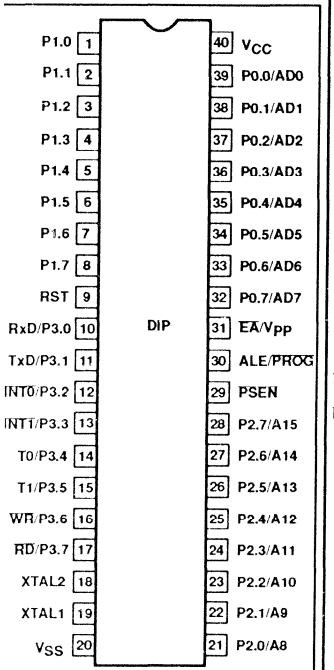
8051-based 8-bit Microcontrollers

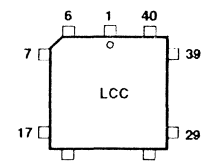
BOOK | IC20 | 1991

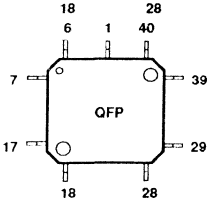
Philips Components

PHILIPS

8031AH/8051AH, 80C31/80C51/87C51, 80/83/87C652*, 80/83/87C654*, 80/83C851

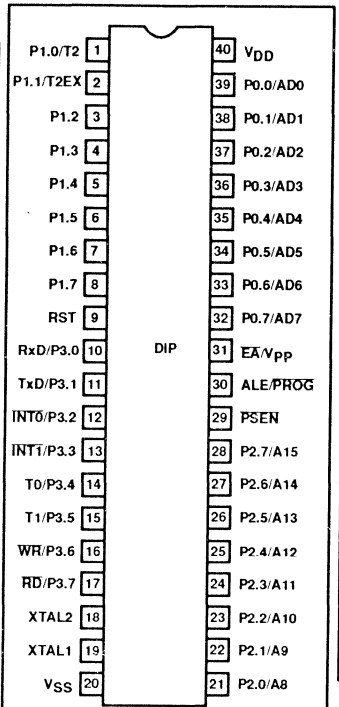


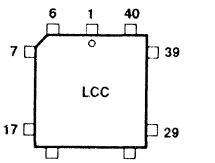


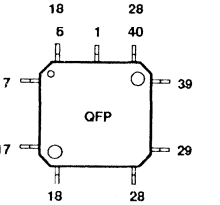


Pin	Function	Pin	Function
1	NC	23	NC/VSS
2	P1.0	24	P2.0/A8
3	P1.1	25	P2.1/A9
4	P1.2	26	P2.2/A10
5	P1.3	27	P2.3/A11
6	P1.4	28	P2.4/A12
7	P1.5	29	P2.5/A13
8	P1.6	30	P2.6/A14
9	P1.7	31	P2.7/A15
10	RST	32	PSEN
11	P3.0/RxD	33	ALE
12	NC	34	NC
13	P3.1/TxD	35	EA
14	P3.2/INT0	36	P0.7/AD7
15	P3.3/INT1	37	P0.6/AD6
16	P3.4/T0	38	P0.5/AD5
17	P3.5/T1	39	P0.4/AD4
18	P3.6/WR	40	P0.3/AD3
19	P3.7/RD	41	P0.2/AD2
20	XTAL2	42	P0.1/AD1
21	XTAL1	43	P0.0/AD0
22	VSS	44	VCC

8032AH/8052AH, 80C32/80C52/87C52, 80/83/87C528*



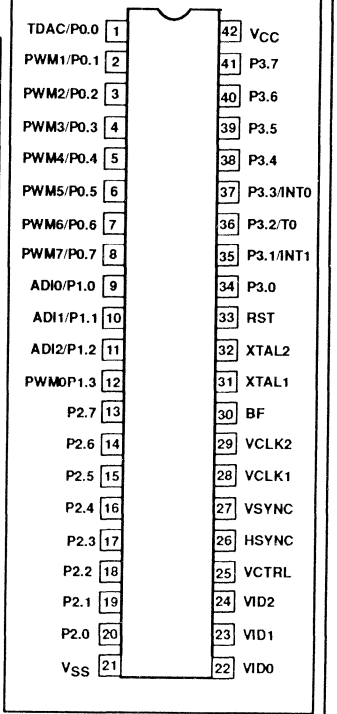




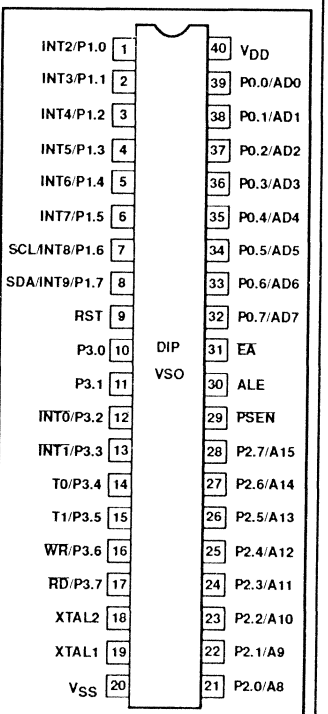
Pin	Function	Pin	Function
1	NC	23	NC/VSS
2	T2/P1.0	24	P2.0/A8
3	T2EX/P1.1	25	P2.1/A9
4	P1.2	26	P2.2/A10
5	P1.3	27	P2.3/A11
6	P1.4	28	P2.4/A12
7	P1.5	29	P2.5/A13
8	P1.6	30	P2.6/A14
9	P1.7	31	P2.7/A15
10	RST	32	PSEN
11	RxD/P3.0	33	ALE/PROG
12	NC	34	NC
13	TxD/P3.1	35	EA/VPP
14	INT0/P3.2	36	P0.7/AD7
15	INT1/P3.3	37	P0.6/AD6
16	T0/P3.4	38	P0.5/AD5
17	T1/P3.5	39	P0.4/AD4
18	WR/P3.6	40	P0.3/AD3
19	RD/P3.7	41	P0.2/AD2
20	XTAL1	42	P0.1/AD1
21	XTAL2	43	P0.0/AD0
22	VSS	44	VCC

P1.6 and P1.7 have the alternate functions SCL and SDA, respectively, on the 80/83/87C652, 80/83/87C654, 80/83/87C528, and 80/83/87C552.

83C053, 87C054

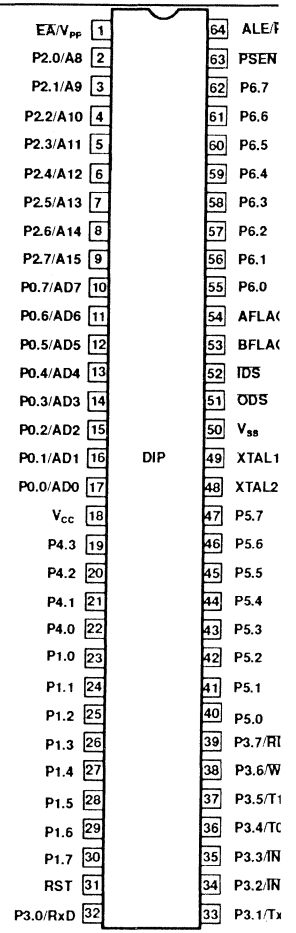


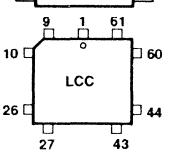
80CL410, 83CL410



See inside of back cover for additional pins.

80/83/87C451





Pin	Function	Pin	Function
1	EA/VPP	35	RST
2	P2.0/A8	36	P3.0/RxD
3	P2.1/A9	37	P3.1/TxD
4	P2.2/A10	38	P3.2/INT0
5	P2.3/A11	39	P3.3/INT1
6	P2.4/A12	40	P3.4/T0
7	P2.5/A13	41	P3.5/T1
8	P2.6/A14	42	P3.6/WR
9	P2.7/A15	43	P3.7/RD
10	P0.7/AD7	44	P5.0
11	P0.6/AD6	45	P5.1
12	P0.5/AD5	46	P5.2
13	P0.4/AD4	47	P5.3
14	P0.3/AD3	48	P5.4
15	P0.2/AD2	49	P5.5
16	P0.1/AD1	50	P5.6
17	P0.0/AD0	51	P5.7
18	VCC	52	XTAL2
19	P4.7	53	XTAL1
20	P4.6	54	VSS
21	P4.5	55	ODS
22	P4.4	56	IDS
23	P4.3	57	BFLAG
24	P4.2	58	AFLAG
25	P4.1	59	P6.0
26	P4.0	60	P6.1
27	P1.0	61	P6.2
28	P1.1	62	P6.3
29	P1.2	63	P6.4
30	P1.3	64	P6.5
31	P1.4	65	P6.6
32	P1.5	66	P6.7
33	P1.6	67	PSEN
34	P1.7	68	ALE/PROG

8051-BASED 8-BIT MICROCONTROLLERS

page

Preface iii

Contents v

Section 1 – 8051 Family 1

Section 2 – Inter-Integrated (I^2C) Circuit Bus 131

Section 3 – 80C51 Family Derivatives 150

Section 4 – Application Notes 463

Section 5 – Development Support Tools 551

Section 6 – Additional Microcontroller Group Data Sheets 577

Section 7 – Package Outlines 656

Preface

Microcontrollers from Philips Components

Philips Components supplies a wide range of microcontrollers based on mainstream architectures, spanning 8-, 16- and 32-bit product lines. By offering a large variety of product derivatives, Philips Components can meet a broad range of specific or unique application requirements. All of our microcontrollers are based on mainstream architectures to allow the user to take advantage of existing software and a vast array of third-party support.

Philips Components 8-bit microcontrollers are based on the popular 80C51 and 80C49 architectures. We offer most of the industry standard products in these architectures as well as a large selection of powerful derivative products. These derivatives offer a wide assortment of features, including: additional memory, A/D, PWM, additional timers, and many more. Many of the derivative microcontrollers have an I^2C serial interface that allows them to be connected easily to over 70 other parts, increasing their capabilities even further. The I^2C serial bus is covered in Section 2 of this book. This data book covers the 80C51 standard products and derivatives that Philips Components manufactures. The 80C49 products are covered in a separate book.

Philips Components 16-bit microcontroller family is based on the powerful 68000 architecture. While these are called 16-bit microcontrollers, the 68000 CPU core architecture is 32-bit. This offers the user a great deal more processing power, when the need arises in a design to move from an 8-bit to a 16-bit microcontroller. Philips Components 16-bit microcontrollers are software compatible with existing 68000 code. As with our popular 8-bit microcontrollers, EPROM and OTP versions of our 16-bit products are available. The 16-bit microcontrollers are also covered in a separate data book.

Philips Components is developing a family of 32-bit microcontrollers based on the SPARC RISC architecture. This family of microcontrollers will offer the ultimate in processing power for those applications that are computation intensive in an embedded control environment.

Philips Components offers uncompromising quality, service, and support with all of its microcontroller products. For a complete family and the best in microcontroller products, look to Philips Components.

Contents

80C51 Microcontroller Family Features Guide vii

8051 Microcontroller Cross-Reference Guide ix

80C51 Microcontroller Development System Support x

CMOS and NMOS 8-Bit Microcontroller Family xi

CMOS 16-Bit Microcontroller Family xvii

8-Bit Microcontroller Demonstration and Evaluation Boards xviii

Section 1 — 80C51 Family 1
- Family Overview 1
- Architecture 5
- Family Instruction Set 8
- Hardware Description 20
- Programmers' Guide and Instruction Set 45
- Instruction Definitions 61
- EPROM Products 100
- 8031AH/8051AH Data Sheet 104
- 80C31/80C51/87C51 Data Sheet 113

Section 2 — Inter-Integrated (I^2C) Circuit Bus 131
- I^2C Bus Specification 131
- I^2C Address Allocation Table 147
- Assigned I^2C Bus Addresses 148
- I^2C Peripheral Selection Guide 149

Section 3 — 80C51 Family Derivatives 150
- 8XC52 Overview 150
- 8032AH/8052AH Data Sheet 160
- 80C32/80C52/87C52 Data Sheet 169
- 8XC053/54 Overview 183
- 83C053, 87C054 Data Sheet 186
- 8XCL410 Overview 201
- 80CL410/83CL410 Data Sheet 208
- 8XC451 Overview 225
- 80C451/83C451/87C451 Data Sheet 228
- 8XC528 Overview 245
- 80C528/83C528/87C528 Data Sheet 250
- 8XC550 Overview 265
- 80C550/83C550/87C550 Data Sheet 270
- 8XC552, 8XC562 Overview 288
- 80C552/83C552/87C552 Data Sheet 345
- 80C562/83C562/87C562 Data Sheet 365
- 8XC652/654 Overview 378
- 80C652/83C652/87C652 Data Sheet 382
- 83C654/87C654 Data Sheet 398
- 8XC751 Overview 414
- 83C751/87C751 Data Sheet 421
- 8XC752 Overview 431
- 83C752/87C752 Data Sheet 436
- 8XC851 Overview 449
- 80C851/83C851 Data Sheet 451

Section 4 — Application Notes 463
AN408 80C451 Operation of Port 6 463
AN417 256k Centronics Printer Buffer Using the 87C451 Microcontroller 474
AN418 Counter/Timer 2 of the 83C552 Microcontroller 488
AN420 Using up to 5 External Interrupts on the 8051 Family Microcontrollers 495
AN422 Using the 8XC751 Microcontroller as an I^2C Bus Master 497
AN423 Software Driven Serial Communication Routines for the 83C751 and 83C752 Microcontrollers 516
AN424 8051 Family Warm Boot Determinations 523
AN425 Interfacing PCD8584 I^2C Bus Controller to 80C51 Family Microcontrollers 525
AN426 Controlling Air Core Meters with the 87C751 and SAA5775 545

Section 5 — Development Support Tools 551
Development Support Tools 551
Nohau EMUL51-PC — PC-Based In-Circuit Emulator 556
MetaLink 561
SDS 8051 Stand-alone Debug Station 563
OM4142 (ASM51, as8051) Cross-Assembler Package 569
OM4144 (plm51, PLMTI51) Compiler Package 571
OM4129 Symbolic Debugging Package XRAY51 574

Section 6 — Additional Microcontroller Group Data Sheets 577
SCN8049 Series Microcontrollers 577
8X305 Microcontroller 592
8X401 Microcontroller 617
82C200 Stand-alone CAN Controller 637

Section 7 — Package Outlines 656
24-Pin Hermetic CERDIP (300 mil wide) 656
24-Pin Plastic Dual In-Line (300 mil wide) 657
28-Pin Ceramic Dual In-Line 658
28-Pin Plastic Dual In-Line (600 mil wide) 658
28-Pin Plastic Leaded Chip Carrier 659
40-Pin Ceramic Dual In-Line 659
40-Pin Plastic Dual In-Line 660
44-Pin Plastic Leaded Chip Carrier 661
44-Pin Square Ceramic Leadless Chip Carrier 661
44-Pin Square Ceramic Leaded Chip Carrier 662
64-Pin Plastic Dual In-Line 663
68-Pin Plastic Leaded Chip Carrier 663
68-Pin Square Ceramic Leadless Chip Carrier 664
84-Pin Square Plastic Leaded Chip Carrier 665
84-Pin Square Ceramic Leadless Chip Carrier 665
40-Lead Mini-Pack; Plastic (VSO40; SOT158A) 666
44-Pin Quad Flat-Pack 667
80-Lead Quad Flat-Pack; Plastic (SOT219) 668

80C51 Microcontroller Family Features Guide

PART NO.			MEMORY		COUNTER/	I/O	SERIAL	EXTERNAL	
ROMless	ROM	EPROM	ROM	RAM	TIMER	PORT	INTERFACES	INTERRUPTS	SPECIAL FEATURES
–	83C751	87C751	2k	64	1 (16-bit)	2–3/8	I^2C (Bit)	2	Low-cost 24-pin skinny-DIP
–	83C752	87C752	2k	64	1 (16-bit)	2–5/8	I^2C (Bit)	2	5-Channel 8-bit A/D, PWM output
8031AH	8051AH	–	4k	128	2	4	UART	2	NMOS
80C31B	80C51B	87C51	4k	128	2	4	UART	2	CMOS
80CL410	83CL410	–	4k	128	2	4	I^2C	10	Low voltage (1.5V–6V), low power
80C451	83C451	87C451	4k	128	2	7	UART	2	Extended I/O, Processor bus interface
80C550	83C550	87C550	4k	128	2 + Watchdog	4	UART	2	8-Channel 8-bit A/D
80C851	83C851	–	4k	128	2	4	UART	2	256 Bytes EEPROM, 8051 pin-for-pin compatible
8032AH	8052AH	–	8k	256	3	4	UART	2	NMOS
80C32	80C52	87C52	8k	256	3	4	UART	2	CMOS
80C552	83C552	87C552	8k	256	3 + Watchdog	6	UART, I^2C	6	8-Channel 10-bit A/D, 2 PWM outputs
80C562	83C562	–	8k	256	3 + Watchdog	6	UART	6	8-Channel 8-bit A/D, 2 PWM outputs
80C652	83C652	87C652	8k	256	2	4	UART, I^2C	2	8051 pin-for-pin compatible with twice the memory
–	83C053	–	8k	192	2 (16-bit)	3-4/8	–	3	On-screen display, 9PWM outputs, 3 software A/D inputs
–	83C054	87C054	16k	192	2 (16-bit)	3-4/8	–	3	On-screen display, 9PWM outputs, 3 software A/D inputs
–	83C654	87C654	16k	256	2	4	UART, I^2C	2	8051 pin-for-pin compatible with four times the ROM and twice the RAM
80C528	83C528	87C528	32k	512	3 + Watchdog	4	UART, I^2C (Bit)	2	The Ultimate Programming Machine

PART NO.	SPEED				TEMPERATURE °C			PACKAGE				
	0.5–12	3.5–12	3.5–16	3.5–20	0 to 70	–40 to +85	–55 to +125	PDIP	CDIP	PLCC	CLCC	QFP
83/87C751	X	X	X		X	X	X (–40 to +125)	24	24	28		
83/87C752	X	X	X		X	X	X (87C751)	28	28	28		
8031/51		X	X (3.5–15MHz)		X	X		40		44		
80C31/51/87C51	X	X	X	X (1.2–30MHz)	X	X	X	40	40	44	44	44
80/83CL410			X (32kHz–16MHz)			X		40				
80/83/87C451	X	X	X		X	X	X	64	64	68	68	
80/83/87C550	X	X	X		X	X		40	40	44	44	
80/83C851		X (1.3–12MHz)			X	X		40		44		44
8032/52		X	X (3.5–15MHz)		X	X		40		44		
80C32/52/87C52	X		X	X	X	X		40	40	44	44	44
80/83/87C552			X (1.2–16MHz)		X	X				68	68	
80/83C562			X (1.2–16MHz)		X		x (–40to+125)			68		
80/83/87C652		X (1.2–12MHz)	X (1.2–16MHz)		X	X	X (–40 to +125)	40	40	44	44	44
83C053		X (6–12MHz)			X			42 SDIP				
83/87C054		X (6–12MHz)			X			42 SDIP				
83/87C654		X (1.2–12MHz)	X (1.2–16MHz)		X	X	X (–40 to +125)	40	40	44	44	44
80/83/87C528	X	X	X		X	X	X	40	40	44	44	44

Note: All combinations of part type, speed, temperature, and package may not be available.

8051 Microcontroller Cross-Reference Guide

	INTEL	AMD	SIEMENS	OKI	MATRA/HARRIS	PHILIPS COMPONENTS/ SIGNETICS
NMOS	8039AL 8049AH 8040AHL 8050AH					MAB8039H/SCN8039H MAB8049H/SCN8049H MAB8040H/SCN8040H MAB8050H/SCN8050H
	8031AH 8051AH 8032AH 8052AH	8031AH 8051AH	SAB 8031A SAB 8051A SAB 8032A SAB 8052A			MAB8031AH/SCN8031H MAB8051AH/SCN8051H MAB8032AH/SCN8032H MAB8052AH/SCN8052H
CMOS	80C31BH 80C31BH-1 80C31BH-2	80C31BH	SAB 80C31	MSM80C31 MSM80C31	80C31 80C31-1 80C31	PCB80C31BH-2/SC80C31BCC PCB80C31BH-3/SC80C31BCG /SC80C31BCB
	80C51BH 80C51BH-1 80C51BH-2	8051BH	SAB 80C51	MSM80C51 MSM80C51	80C51 80C51-1 80C51	PCB80C51BH-2/SC80C51BCC PCB80C51BH-3/SC80C51BCG /SC80C51BCB
	87C51 87C51-1 87C51-2	87C51				SC87C51CC SC87C51CG SC87C51CB
	80C32 80C32-1 80C52 80C52-1	 80C52T2				P80C32EB P80C32GB P80C52EB P80C52GB

NOTES:
1. Siemens 8032A–16 = 16MHz 8032.
2. AMD 80C52T2 = 80C52 without T2.
3. 80XXAHL = 80XX with low power standby pin; H = HMOS.

80C51 Microcontroller Development System Support

COMPANY	ADDRESS	SUPPORT FUNCTIONS
Metalink	325 E. Elliot Road Suite 23 Chandler, AZ 85225 (602) 926-0797	MICROICE Emulator Supports 80C51 Microcontroller Family
Nohau	51 E. Campbell Ave. Campbell, CA 95008 (408) 866-1820	EMUL51-PC Supports 80C51 Microcontroller Family
Ashling Microsystems Limited	Plassey Technological Park Limerick, Ireland (353) 61-334466	Emulators for 80C51 Microcontroller Family
Philips Components	Corporate Centre Building BAE-2 P.O. Box 218 5600 MD Eindhoven The Netherlands 31-40-724223	Stand-Alone Debug Station Emulator for 80C51 Family
Tasking	Tasking Software BV P.O. Box 899 3800 AW Amersfoort The Netherlands 31-33-55-85-84	C Compiler and Cross Assembler for 80C51 Family
BSO/Tasking	128 Technology Center P.O. Box 9164 Waltham, MA 02254–9164 (617) 894-7800	C Compiler and Cross Assembler for 80C51 Family
Franklin Software	888 Saratoga Ave., #2 San Jose, CA 95129 (408) 296-8051	C Compiler for 80C51 Family
Archimedes Software	2159 Union Street San Francisco, CA 94123 (415) 567-4010	C Compiler for 80C51 Family
CEIBO	105 Gleason Rd. Lexington, MA 02173 (617) 863-9927	80C51 Family Demonstration Boards

EPROM PROGRAMMING SUPPORT CONTACTS			
Data I/O Corp. 10525 Williams Road N.E. P.O. Box 97056 Redmond, WA 98073-9746 (206) 867-6899	**Logical Devices, Inc.** 1201 Northwest 65th Place Ft. Lauderdale, FL 33309 (305) 974-0967	**Needhams's Electronics** 4535 Orange Grove Ave. Sacramento, CA 95841 (916) 924-8037	**BP Microsystems** 10681 Haddington, Suite 190 Houston, TX 77043 (800) 225-2102
GTEK, Inc. P.O. Box 2310 Bay St., Lewis, MS 39521-2310 (300) 282-4835	**Logical Systems** P.O. Box 6184 Syracuse, NY 13217-6184 (315) 478-0722	**North Valley Products** P.O. Box 32899 San Jose, CA 95152 (408) 929-5345	**Stag Microsystems** 528-5 Weddell Drive Sunnyvale, CA 94086 (408) 988-1118

NOTE:
For more information on Development Support, see Section 5 of this book.

CMOS and NMOS 8-Bit Microcontroller Family

80C51 Family CMOS

TYPE	ROM/ EPROM	RAM	SPEED (MHz)	PACKAGE	FUNCTIONS	REMARKS	PROBE SDS	METALINK EMULATOR	REMARKS
80C31 80C51 87C51	0 4k ROM 4k EPROM	128 128 128	30 30 16	DIL40, LCC44 QFP44	UART, 2 timers		OM1092 + OM1097	SUI-8051SD	OM1092: Universal probe OM1095: Upgrade unit
80C32 80C52 87C52	0 8k ROM 8k EPROM	256 256 256	20 20 20	DIL40, LCC44 QFP44	UART, 3 timers			SUI-8051SD	
80C451 83C451 87C451	0 4k ROM 4k EPROM	128 128 128	16 16 16	DIP64/LCC68	UART, 2 timers Extended I/O		OM4123	SMI-83C451SD SMI-80C451SD	OM4124: PLCC to DIL OM4125: DIL to PLCC
83C528 87C528	32k ROM 32k EPROM	512 512	16 16	DIL40/LCC44 (QFP44)	UART, 3 timers Watchdog timer Bit I^2C	Samples: Now Production: Q1 '91	OM4111 + OM4110		OM4110: gen. probe OM4111: probe head OM4120-S for max. speed
83C550 87C550	4k ROM 4k EPROM	128 128	16 16	LCC44 DIL44	UART, 2 timers 8 8-bit ADC inputs, watchdog timer	Samples: Now Production: Q1 '91			
80C552 83C552 87C552	0 8k ROM 8k EPROM	256 256 256	16 16 16	LCC68/QFP80	UART, 2 timers Timer with compare and capture, 2 PWM outputs, 8 10-bit ADC inputs, Byte I^2C		OM1092 + OM1095	SMI-80C552SD	OM1092: Universal probe OM1095: Upgrade unit
80C562 83C562	0 8k ROM	256 256	16 16	LCC68/QFP80	UART, 2 timers Timer with compare and capture, 2 PWM outputs, 8 8-bit ADC inputs	Use 87C552 for development	OM1092 + OM1095	SMI-80C552SD	OM1092: Universal probe OM1095: Upgrade unit
80C592 83C592 87C592	0 16k ROM 16k EPROM	512 512 512	16 16 16	LCC68/QFP80	8XC552 + CAN interface No I^2C	Samples: Q4 '90 Production: Q1 '91	OM4110 + OM4112		OM4110: gen. probe OM4112: probe head OM4120S: full speed
80C652 83C652 87C652	0 8k ROM 8k EPROM	256 256 256	12 12 12	DIL40/LCC44 QFP44	UART, 2 timers Byte I^2C	EPROM Production: Q1 '91	OM1092 + OM1096	SMI-80C652SD	

80C51 Family CMOS (Continued)

TYPE	ROM/ EPROM	RAM	SPEED (MHz)	PACKAGE	FUNCTIONS	REMARKS	PROBE SDS	METALINK EMULATOR	REMARKS
83C654 87C654	16k ROM 16k EPROM	256 256	16 16	DIL40/LCC44 QFP44	UART, 2 timers Byte I^2C	EPROM Production: Q1 '91	OM1092 + OM1096		OM1092: Universal probe OM1095: Upgrade unit
83C751 87C751	2k ROM 2k EPROM	64 64	16 16	DIP24 skinny LCC28 DIP24 skinny	1 timer Bit I^2C		OM1094P/WP	SMI-80C751SD	
83C752 83C752	2k ROM 2k EPROM	64 64	16 16	DIP28, LCC28 DIP 28, LCC28	1 timer, PWM output, 5 8-bit ADC inputs, Bit I^2C			SMI-83C752SD	
80C851 83C851 89C851	0 4k ROM 4k EEPROM	128 128 128	12 12 12	DIL40/LCC44 QFP44	UART, 2 timers 256 byte EEPROM	QFP44 In devel.	OM1092		
83C852	6k ROM	256	16		2k byte EEPROM smart card hardware	In devel.	OM4119		
83C053	8k ROM	192	12	DIP42 Shrunk	2 timers, 14-bit PWM, 8-6 bit PWM 128 char. OSD 3 4-bit A/D inp.	Samples: Now Production: Q4 '90			
83C054 87C054	16k ROM 16k EPROM	192 192	12 12	DIP42 Shrunk DIP42 Shrunk	As 8XC053	Samples: Now Production: Q1 '91			

* The following microcontollers have no external memory access: 8XC751, 8XC752, 8XC053, 87C054.

80CLXXX Family CMOS

TYPE	ROM	RAM	SPEED (MHz)	PACKAGE	FUNCTIONS	REMARKS	PROBE SDS	REMARKS
85CL001	0	256	16	PLCC84	UART, 2 timers Byte I^2C	In development		
85CL000	0	256	16		UART, 2 timers Byte I^2C	Piggyback In development		
83CL410	4k	128	16	DIL40 VSO40	2 timers Byte I^2C		OM1079	
83CL710	16k	256	16	DIL40	UART, 2 timers Byte I^2C	In development	OM1079	
80CL51	4k	128	16	DIL40	UART, 2 timers	In development		
83CL780	16k	512	16		256 bytes EEPROM AD converter	In development		

8051 Family NMOS

TYPE	ROM	RAM	SPEED (MHz)	PACKAGE	FUNCTIONS	REMARKS	PROBE SDS	METALINK EMULATOR	REMARKS
8051 8031	4k 0	128 128	15 15	DIL40/PLCC44 DIL40/PLCC44	UART, 2 timers		OM1091P/WP	SUI-8051SD	
8052 8032	8k 0	256 256	15 15	DIL40/PLCC44 DIL40/PLCC44	UART, 3 timers UART, 3 timers			SUI-8051SD	OM1091P/WP: No emulation of T2

8048 Family NMOS

TYPE	ROM	RAM	SPEED (MHz)	PACKAGE
8048 8035	1k 0	64 64	11 11	DIL40/PLCC44 DIL40/PLCC44
8049 8039	2k 0	128 128	11 11	DIL40/PLCC44 DIL40/PLCC44
8050 8040	4k 0	256 256	11 11	DIL40/PLCC44 DIL40/PLCC44

8048 Family CMOS

TYPE	ROM	RAM	SPEED (MHz)	PACKAGE
80C49 80C39	2k 0	128 128	15 15	DIL40/PLCC44 DIL40/PLCC44

8400 Family CMOS

TYPE	ROM	RAM	SPEED (MHz)	PACKAGE	FUNCTIONS	REMARKS	PROBE SDS	REMARKS
84C21 84C41 84C81 84C41C	2k 4k 8k 4k	64 128 256 128	10 10 10 12	DIL28/SO28 DIL28/SO28 DIL28/SO28 DIL28/SO28	20 I/O lines 8-bit timer Byte I^2C		OM1083	
84C12 84C22 84C42 84C12A	1k 2k 4k 1k	64 64 64 64	10 10 10 16	DIL20/SO20 DIL20/SO20 DIL20/SO20 DIL20/SO20	13 I/O lines 8-bit timer	 July '90	OM1083	
84C00B 84C00T	0 0	256 256	10 10	28 pins VSO-56	20 I/O lines 8-bit timer Byte I^2C	Piggyback ROMless	OM1080	
84C121 84C121B	1k 0	64 64	10 10	DIL20/SO20	13 I/O lines 2 8-bit timers 8 bytes EE- PROM	 Piggyback	OM1073	
84C122	1k	32	10	SO20/SO24	Controller for remote control	APP: Sept./ Oct. '90		
84C230	2l	64	10	DIL40/VSO40	12 I/O lines 8-bit timer 16*4 LCD drive		OM1072	
84C430 84C430B	4k 0	128 128	10 10	QFP64	24 I/O lines 8-bit timer Byte I^2C 24*4 LCD drive	 Piggyback for C230 and C430	OM1072	
84C633 84C633B	6k 0	256 256	16 16	VSO56	28 I/O lines 8-bit timer 16-bit up/down counter 16-bit timer with compare and capture 16*4 LCD drive	Julyt '90 Piggyback for C633 Q3 '90	OM1086	Q3 '90
84C440 84C441 84C640 84C641 84C643 84C644 84C840 84C841 84C843 84C844	4k 4k 6k 6k 6k 6k 8k 8k 8k 8k	128 128 128 128 128 128 192 192 192 192	10 10 10 10 10 10 10 10 10 10	DIP42 shrunk	RC: 29 I/O lines LC: 28 I/O lines 8-bit timer 1 14-bit PWM 5 6-bit PWM 3-bit ADC OSD 2L-16	I^2C, RC I^2C, LC I^2C, RC I^2C, LC RC ,Q2 '90 LC ,Q2 '90 I^2C, RC ,Q2 '90 I^2C, LC ,Q2 '90 RC ,Q2 '90 LC ,Q2 '90	OM1074	For emulation of LC versions, use OM1074 + adapter_3
84C646 84C846	6k 8k	192 192	10 10	DIP42 shrunk	tbf	I^2C, RC ,Q4 '90 I^2C, RC ,Q4 '90	tbf	
84C647 84C847	6k 8k	192 192	10 10	DIP42 shrunk	tbf	I^2C, LC ,Q4 '90 I^2C, LC ,Q4 '90	tbf	

8400 Family CMOS (Continued)

TYPE	ROM	RAM	SPEED (MHz)	PACKAGE	FUNCTIONS	REMARKS	PROBE SDS	REMARKS
84C85	8k	256	10	DIL40/VSO40	32 I/O lines 8-bit timer Byte I^2C		CM1070	No trace
84C85B	0	256	10			Piggyback for C85		
84C853	8k	256	16	DIL40/VSO40	33 I/O lines 8-bit timer 16-bit up/down counter 16-bit timer with compare and capture	July '90	OM1081	Q3 '90
84C853B	0	256	16			Piggyback for C853 Q3 '90		
84C270	2k	128	10	DIL40/VSO40	8 I/O lines 16*8 capture keyboard matrix 8-bit timer		OM1077	
84C470	4k	128	10	DIL40/VSO40				
84C270B	0	128	10			Piggyback for C270		
84C470B	0	128	10		470 also handles mech. keys	Piggyback for C470		
84C271	2k	128	10	DIL40	8 I/O lines 16*8 mech. keyboard matrix 8-bit timer		OM1078	

8400 Family NMOS

TYPE	ROM	RAM	SPEED (MHz)	PACKAGE	FUNCTIONS	REMARKS	EMULATOR TOOLS	REMARKS
8411 8421 8441 8461	1k 2k 4k 6k	64 64 128 128	6 6 6 6	DIL28/SO28 DIL28/SO28 DIL28/SO28 DIL28/SO28	20 I/O lines 8-bit timer Byte I^2C		OM1084	
8422 8442	2k 4k	64 128	6 6	DIL20 DIL20	13 I/O lines 8-bit timer Bit I^2C		PM8327/20 + PM8447	On PMDS
8401B 8401WP	0 0	128 128	6 6	28-pin PLCC68		Piggyback for 84X1 Bond out		

3300 Family CMOS

TYPE	ROM	RAM	SPEED (MHz)	PACKAGE	FUNCTIONS	REMARKS	PROBE SDS	REMARKS
3315	1.5k	160	10	DIL28/SO28	20 I/O lines 8-bit timer V_{DD} > 1.8V		OM1083	
3343	3k	224	10	DIL28/SO28	20 I/O lines 8-bit timer V_{DD} > 1.8V Byte I^2C		OM1083	
3344	2k	224	3.58	DIL28/SO28	20 I/O lines 8-bit timer DTMF generator		OM1071	
3346	4k	128	10	DIL28/SO28	20 I/O lines 8-bit timer Byte I^2C 256 bytes EEPROM V_{DD} < 1.8V		OM1076	
3347	1.5k	64	3.58	DIL20/SO20	12 I/O lines 8-bit timer DTMF generator		OM1071	
3348	8k	256	10	DIL28/SO28	20 I/O lines 8-bit timer Byte I^2C V_{DD} < 1.8V		OM1083	
3349	4k	224	3.58	DIL28/SO28	20 I/O lines 8-bit timer DTMF generator		OM1071	
3350	8k	128	3.58	VSO64	20 I/O lines 8-bit timer DTMF generator 256 bytes EEPROM 160 segm. LCD (1:4)	To be developed		
3351	2k	64	3.58	DIL28/SO28	20 I/O lines 8-bit timer DTMF generator 128 bytes EEPROM	In development		
3352	4k				tbd			
3301B						Piggyback for 3315, 3343, 3348		
3344B						Piggyback for 3344, 3347, 3349		
3346B						Piggyback for 3346		

CMOS 16-Bit Microcontroller Family

16-Bit Controllers

TYPE	(EP)ROM	RAM	EEPROM	SPEED (MHz)	FUNCTIONS	REMARKS	DEVELOPMENT TOOLS
68070	–	–	–	17.5	2 DMA channels, MMU, UART, 16-bit timer, I^2C, 68000 bus interface, 16Mb address range		OM4160 Microcore OM4161 (SBE68070) TRACE32-ICE68070 (Lauterbach) OM4222 90C Development system (planned)
93C101					Derivative with low power modes		
93C110 90C110 93C100 90C100 97C100	34k – 34k – 32k (EPROM)	512 512 512 512 512	256 256 – – –	15 15	UART, I^2C, 3 16-bit timers, 80C51 interface, 68000 interface, 40 I/O lines, 2Mb address range		OM4160/3 Microcore 3 OM4201WP (SBE90C110) OM4220 90C Development system TRACE32 – ICE93C110 (Lauterbach)

16-Bit Microcontroller Family[1]

PART NO.	ROM	RAM	EE-PROM	16-BIT I/O PORTS	SERIAL I/O	DMA CHAN-NELS	COUNT-ER/ TIMER	EXTERNAL INTER-RUPTS	SPEED MHz	PACK-AGES	SPECIAL FEATURES
68070	–	–	–	–	UART, I^2C	2	1[2]	6	10, 12, 15, 17.5	PLCC84 QFP120	Memory management unit 68000 bus interface
90C100	–	512	–	2 + 1/2	UART, I^2C	–	1[2]	8	15	PLCC84 QFP80	80C51 bus interface 68000 bus interface
93C100	34k	512	–	2 + 1/2	UART, I^2C	–	1[2]	8	15	PLCC84 QFP80	80C51 bus interface 68000 bus interface
97C100	32k EPROM	512	–	2 + 1/2	UART, I^2C	–	1[2]	8	15	PLCC84 CLCC84 QFP80	80C51 bus interface 68000 bus interface
90C110	–	512	256	2 + 1/2	UART, I^2C	–	1[2]	8	15	PLCC84 QFP80	80C51 bus interface 68000 bus interface
93C110	34k	512	256	2 + 1/2	UART, I^2C	–	1[2]	8	15	PLCC84 QFP80	80C51 bus interface 68000 bus interface

NOTES:
1. 68000 software compatible.
2. 16-bit timer with two match/count/capture registers.

8-Bit Microcontroller Demonstration and Evaluation Boards

I²C demonstration board based on 84CXX derivatives

I²C LCD driver demonstration board with 84CXX microcontroller

I²C bus analyzer (with 84CXX)

I²C demonstration board based on 80C51 derivatives

8051 family demonstration board (P8051DB)

8XC552 evaluation board

CAN controller evaluation board (OM4130)

68070 demonstration and evaluation board (MicroCore I)

90C100 family demonstration and evaluation board (MicroCore III)

Section 1
8051 Family Overview

CONTENTS

Family Overview 1
8051 1
80C51 1
8052 1
80C52 3
83C053 3
83CL410 3
83C451 3
83C528 3
83C550 3
83C552 3
83C562 3
83C652 4
83C654 4
83C751 4
83C752 4
83C851 4

80C51 Architecture 5
Memory Organization 5
Program Memory 5
Data Memory 6

8051 Family Instruction Set 8
Program Status Word 8
Addressing Modes 9
Direct Addressing 9
Indirect Addressing 9
Register Instructions 9
Register-Specific Instructions 9
Immediate Constants 9
Indexed Addressing 9
Arithmetic Instructions 9
Logical Instructions 10
Data Transfers 10
Internal RAM 10
External RAM 11
Lookup Tables 12
Boolean Instructions 12
Relative Offset 13
Jump Instructions 13
CPU Timing 14
Machine Cycles 14
Interrupt Structure 15
Interrupt Enables 15
Interrupt Priorities 19
Simulating a 3rd Priority Level in Software 19

Hardware Description 20
Special Function Registers 21
Accumulator 21
B Register 21
Program Status Word 21
Stack Pointer 21
Data Pointer 21
Ports 0 to 3 21
Serial Data Buffer 21
Timer Registers Basic to 80C51 21
Control Registers for the 80C51 21

Hardware Description (Continued) ....
Port Structures and Operation 21
I/O Configurations 22
Writing to a Port 22
Port Loading and Interfacing 23
Read-Modify-Write Feature 24
Accessing External Memory 24
Timer/Counters 25
Timer 0 and Timer 1 25
Mode 0 26
Mode 1 26
Mode 2 26
Mode 3 26
Standard Serial Interface 26
Multiprocessor Communications 28
Serial Port Control Register 28
Baud Rates 28
Using Timer 1 to Generate Baud Rates 29
More About Mode 0 30
More About Mode 1 30
More About Modes 2 and 3 30
Interrupts 35
Priority Level Structure 35
How Interrupts Are Handled 36
External Interrupts 37
Response Time 37
Single-Step Operation 37
Reset 37
Power-On Reset 38
Power-Saving Modes of Operation 38
CMOS Power Reduction Mode 38
Idle Mode 38
Power-Down Mode 39
On-Chip Oscillators 40
NMOS Versions 40
CMOS Versions 41
Internal Timing 41
80C51 Pin Descriptions 41

Programmers' Guide and Instruction Set 45
Memory Organization 45
Program Memory 45
80C51 Family Instruction Set 57

Instruction Definitions 61

EPROM Products 100
Programming the 87C51 100
Encryption Table 100
Lock Bit 101
Program Verification 101
Signature Bytes 102
EPROM Erasure 102
Programming the 87C751 and 87C752 102
Program Verification 103
87C751 and 87C752 Signature Bytes 103
Programming Features 103
Erasure Characteristics 103
8031AH/8051AH Data Sheet 104
80C31/80C51/87C51 Data Sheet 113

Section 1
8051 Family Overview

8051 FAMILY OVERVIEW

The Philips 8051 family of products is based on the industry standard for 8-bit high performance microcontrollers. The architecture for the family has been optimized for sequential real time control applications. The 8051 family of products are used in a wide range of applications from those that are relatively simple to applications in medical instrumentation and automobile control systems. All of the devices included in the family are available in versions that have either internal ROM, EPROM, or CPU only. With the exception of the 83C751 and 752 (which are limited to on-board memory) all of the devices in the family can address up to 64k bytes of both program and data memory.

The 8051 family of microcontrollers includes the devices listed in Table 1. The basic architecture of these devices is shown in Figure 1.

8051

The 8051 is the original member of the family. Among the features of the 8051 are:

- 8-bit CPU optimized for control applications
- Extensive Boolean processing (single-bit logic) capabilities
- 32 bidirectional and individually addressable I/O lines
- 128 bytes of on-chip data RAM
- Two 16-bit timer/counters
- Full duplex UART
- 5-source interrupt structure with 2 priority levels
- On-chip clock oscillator
- 4k bytes of on-chip program memory
- 64k bytes program memory address space
- 64k bytes data memory address space
- 40-pin DIP and 44-pin PLCC packages

The 8031 is a CPU only version of the 8051 and differs from the 8051 in that it does not have on-chip ROM. The 8031 fetches all instructions from external memory.

80C51

The 80C51 is the CMOS version of the 8051. Functionally it is fully compatible with the 8051, but being CMOS it draws less current than its NMOS counterpart.

The ROMless version of the 80C51 is the 80C31. The EPROM version is the 87C51.

8052

The 8052 is an enhanced version of the 8051. It is fabricated with NMOS technology, and is upwards compatible with the 8051. Its enhancements over the 8051 include:

- 256 bytes of on-chip data RAM
- Three counter/timers
- A 6-source interrupt structure
- 8k bytes of on-chip program memory
- 40-pin DIP and 44-pin PLCC packages

The ROMless version of the 8052 is the 8032.

Table 1. 8051 Family of Microcontrollers

DEVICE NAME	ROMLESS VERSION	EPROM VERSION	ROM BYTES	RAM BYTES	16-BIT TIMERS	CIRCUIT TYPE
8051	8031	–	4k	128	2	NMOS
80C51	80C31	87C51	4k	128	2	HMOS
8052	8032	–	8k	256	3	NMOS
80C52	80C32	87C52	8k	256	3	CMOS
83C053	–	87C054	8k	192	2	CMOS
83CL410	80CL410	–	4k	128	2	CMOS
83C451	80C451	87C451	4k	128	2	CMOS
83C528	80C528	87C528	32k	512	3 + WD	CMOS
83C550	80C550	87C550	4k	128	2 + WD	CMOS
83C552	80C552	87C552	8k	256	3 + WD	CMOS
83C562	80C526	–	8k	256	3 + WD	CMOS
83C652	80C652	87C652	8k	256	2	CMOS
83C654	–	87C654	16k	256	2	CMOS
83C751	–	87C751	2k	64	1	CMOS
83C752	–	87C752	2k	64	1	CMOS
83C851	80C851	–	4k	128	2	CMOS

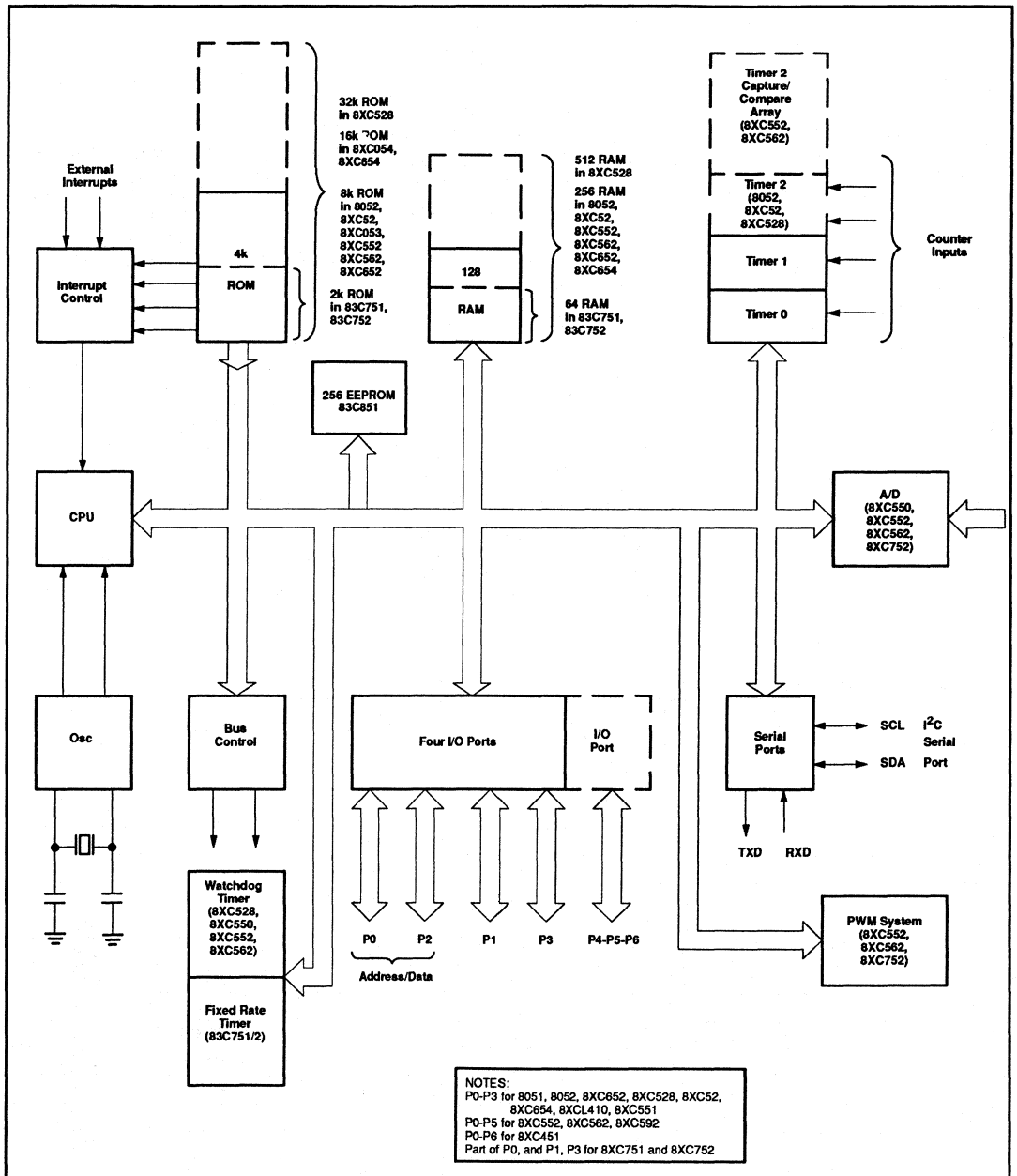


Figure 1. 8051 Family Block Diagram

80C52

The 80C52 is the CMOS version of the 8052. Functionally it is fully compatible with the 8052, but being CMOS it draws less current than its NMOS counterpart.

The ROMless version of the 80C52 is the 80C32. The EPROM version is the 87C52.

83C053

The 83C053 is a derivative of the 80C51 that is intended for use as the central control mechanism in a television. It provides tuner functions and has an on-screen display facility. The main features of the part are:

- 8k ROM (83C053)
- 16k EPROM (87C054)
- 192 bytes RAM
- On-screen display controller
- Three digital video outputs
- Multiplexer/mixer and background intensity controls
- 128 x 10 display RAM
- 60 x 18 x 14 character generator ROM
- Eight 6-bit PWM
- One 14-bit PWM
- Four high current open-drain port outputs
- Twelve high voltage (+12V) open-drain outputs
- Programmable video input and output polarities
- 42-pin shrink DIP

There is no ROMless version of the part. The EPROM version is the 87C054.

83CL410

The 83CL410 is a derivative of the 80C51 that operates at very low supply levels and frequency. At lower supply levels and frequency, the part has dramatically reduced power dissipation. This part is ideally suited for low voltage battery operation. The part has all of the features of an 80C51, except the full-duplex UART. The additional features of the 83CL410 are:

- I^2C serial interface
- Warm start from power-down mode
- 32kHz to 20MHz frequency operation (can be operated to DC with an external oscillator)
- Power supply range: 1.5 to 6V

This part is socket compatible with the 80C51. The ROMless version is the 80CL410. There is no EPROM version.

83C451

The 83C451 is an extended I/O version of the 80C51 with the following features:

- Seven 8-bit quasi-bidirectional I/O ports (PLCC version)
- Six 8-bit and one 4-bit quasi-bidirectional I/O ports (DIP version)
- Mailbox port (port 6) features:
  - Operation as normal quasi-bidirectional I/O port
  - Four handshake control pins
  - Control status register
  - Input and output buffer registers making port 6 suitable for:
    direct MPU interface
    parallel printer interface
- 64-pin DIP and 68-pin PLCC packages

All other aspects of the 83C451 are identical to the 80C51. The ROMless version of the 83C451 is the 80C451, and the 87C451 is the EPROM version.

83C528

This is an extended memory version of the 80C51. It is socket compatible with the 80C51 and has all of the 80C51 features plus:

- 32k bytes of ROM
- 512 bytes of RAM
- A third 16-bit counter/timer (identical to the 80C52 T2)
- A watchdog timer with separate oscillator
- An I^2C serial port
- Warm start from power-down mode

The ROMless version of the 83C528 is the 80C528, and the EPROM version is the 87C528.

83C550

The 83C550 is a 40-pin derivative of the 80C51 that has an 8 channel, 8-bit A/D converter. In addition to having all of the features of an 80C51, the part has:

- 8 channel, 8-bit A/D
- Watchdog timer

Although the 83C550 is available in a 40-pin package, it is not socket compatible with the 80C51, because of its analog features.

The ROMless version of the 83C550 is the 80C550 and the EPROM version is the 87C550.

83C552

The 83C552 is an extended function 80C51 with the following features:

- 8k bytes of ROM
- 256 bytes of RAM
- 10-bit 8 channel A/D
- Counter/timer array with high speed outputs and capture inputs
- Four counter/timers (including a watchdog timer)
- Two PWM outputs
- 8051 full duplex UART
- Six 8-bit I/O ports
- I^2C serial port
- 68-pin PLCC package

The 83C552 is 100% code compatible with the 80C51. The ROMless version of the 83C552 is the 80C552 and the EPROM version is the 87C552.

83C562

The 83C562 is an 80C51 derivative that is identical to the 83C552 except that the I^2C interface has been removed and the A/D converter is 8 bit instead of 10 bit.

The ROMless version of the 83C562 is the 80C562. There is no EPROM version. For development and prototyping, the 87C552 can be used.

83C652

The 83C652 is an 80C51 with the following additions: an 8k ROM, 256 bytes RAM and I²C serial port.

The 83C652 is pin-for-pin compatible with the 80C51 except for minor DC specifications on the I²C serial port pins. The ROMless version of the 83C652 is the 80C652 and the EPROM version is the 87C652.

83C654

This 80C51 derivative is identical to the 83C652 except that it has 16k of program ROM. It is pin-for-pin socket compatible with the 80C51.

There is no ROMless version of this part because it would be identical to an 80C652. The EPROM version is the 87C654.

83C751

The 83C751 is a 24-pin derivative of the 80C51, for applications where small size and cost are of prime consideration. The 83C751 is packaged in a 24-pin "skinny-dip" (.300 wide) and in a 28-pin PLCC package. The following differences exist between the 83C751 and the 80C51. The 83C751 has:

- 2k bytes ROM
- 64 bytes RAM
- I²C serial port (no UART)
- 19 I/O lines
- Single level interrupt structure
- One counter/timer with 16-bit autoload
- No external memory expandability (data memory can be expanded using the I²C serial port and I²C compatible memory devices)

Note that since there is no external expandability, the external memory addressing signals: $\overline{WR}$, $\overline{RD}$, $\overline{PSEN}$, $\overline{EA}$, and ALE are not present. Because of these differences, the instructions LJMP, LCALL, and MOVX execute as NOPs in the 83C751.

For all other instructions the 83C751 is 100% code compatible with the 80C51 and operates at full 80C51 speed. The EPROM version of the 83C751 is the 87C751. There is no ROMless version.

83C752

The 83C752 is a 28-pin derivative of the 80C51 that is intended for use in automotive, electro-mechanical, and consumer applications. The 83C752 contains most of the features of the 80C51 with the following differences:

- 2k bytes ROM
- 64 bytes RAM
- Single level interrupt structure
- One 16-bit counter/timer with 16-bit autoload
- Two 8-bit and one 5-bit bidirectional I/O ports
- I²C serial interface
- One PWM with timer, including overflow interrupt capability
- Five channels of 8-bit A/D
- 28-pin packages, both DIP and PLCC

The 83C752 does not have external memory expandability. The EPROM version of the 83C752 is the 87C752. There is no ROMless version.

83C851

This 80C51 derivative has on-chip EEPROM. It is socket compatible with the 80C51 and has all of the features of the 80C51. In addition to these standard features it has:

- 256 bytes of EEPROM

The ROMless version of the part is the 80C851. There is no EPROM version.

Table 2. 80C51 Derivative Comparisons

DEVICE	A/D	PORTS	PWM	TIMERS	UART
80C51	–	4	–	Two standard	UART
80C52	–	4	–	Two standard, timer 2	UART
83C053	–	$3^1/_2$	9	Two standard	–
83CL410	–	4	–	Two standard	I²C
83C451	–	7	–	Two standard	UART
83C528	–	4	–	Two standard, timer 2, watchdog (4 total)	UART, I²C
83C550	8 channel/8-bit	4	–	Two standard, watchdog	UART
83C552	8 channel/10-bit	6	2	Two standard, timer 2, watchdog (4 total)	UART, I²C
83C562	8 channel/8-bit	6	2	Two standard, timer 2, watchdog (4 total)	UART
83C652	–	4	–	Two standard	UART, I²C
83C654	–	4	–	Two standard	UART, I²C
83C751	–	$2^3/_8$	–	One standard, extended to 16-bit autoload	I²C
83C752	5 channel/8-bit	$2^5/_8$	1	One standard, extended to 16-bit autoload	I²C
83C851	–	4	–	Two standard	UART

80C51 ARCHITECTURE

Memory Organization

All 8051 devices have separate address spaces for program and data memory, as shown in Figures 2 and 3. The logical separation of program and data memory allows the data memory to be accessed by 8-bit addresses, which can be quickly stored and manipulated by an 8-bit CPU. Nevertheless, 16-bit data memory addresses can also be generated through the DPTR register.

Program memory (ROM, EPROM) can only be read, not written to. There can be up to 64k bytes of program memory. In the 80C51, the lowest 4k bytes of program are on-chip. In the ROMless versions, all program memory is external. The read strobe for external program memory is the $\overline{PSEN}$ (program store enable).

Data Memory (RAM) occupies a separate address space from Program Memory. In the 80C51, the lowest 128 bytes of data memory are on-chip. Up to 64k bytes of external RAM can be addressed in the external Data Memory space. In the ROMless version, the lowest 128 bytes are on-chip. The CPU generates read and write signals, $\overline{RD}$ and $\overline{WR}$, as needed during external Data Memory accesses.

External Program Memory and external Data Memory may be combined if desired by applying the RD and PSEN signals to the inputs of an AND gate and using the output of the gate as the read strobe to the external Program/Data memory.

Program Memory

Figure 4 shows a map of the lower part of the Program Memory. After reset, the CPU begins execution from location 0000H. As shown in Figure 4, each interrupt is assigned a fixed location in Program Memory. The interrupt causes the CPU to jump to that location, where it commences execution of the service routine. External Interrupt 0, for example, is assigned to location 0003H. If External Interrupt 0 is going to be used, its service routine must begin at location 0003H. If the interrupt is not going to be used, its service location is available as general purpose Program Memory.

The interrupt service locations are spaced at 8-byte intervals: 0003H for External Interrupt 0, 000BH for Timer 0, 0013H for External Interrupt 1, 001BH for Timer 1, etc. If an interrupt service routine is short enough (as is often the case in control applications), it can reside entirely within that 8-byte interval. Longer service routines can use a jump instruction to skip over subsequent interrupt locations, if other interrupts are in use.

The lowest 4k bytes of Program Memory can either be in the on-chip ROM or in an external ROM. This selection is made by strapping the $\overline{EA}$ (External Access) pin to either V_{CC}, or V_{SS}. In the 80C51, if the $\overline{EA}$ pin is strapped to V_{CC}, then the program fetches to addresses 0000H through 0FFFH are directed to the internal ROM. Program fetches to addresses 1000H through FFFFH are directed to external ROM.

If the $\overline{EA}$ pin is strapped to V_{SS}, then all program fetches are directed to external ROM. The ROMless parts (8031, 80C31, etc.) must have this pin externally strapped to V_{SS} to enable them to execute from external Program Memory.

The read strobe to external ROM, $\overline{PSEN}$, is used for all external program fetches. $\overline{PSEN}$ is not activated for internal program fetches.

The hardware configuration for external program execution is shown in Figure 5. Note that 16 I/O lines (Ports 0 and 2) are dedicated to bus functions during external Program Memory fetches. Port 0 (P0 in Figure 5) serves as a multiplexed address/data bus. It emits the low byte of the Program Counter (PCL) as an address, and then goes into a float state awaiting the arrival of the code byte from the Program Memory. During the time that the low byte of the Program Counter is valid on Port 0, the signal ALE (Address Latch Enable) clocks this byte into an address latch. Meanwhile, Port 2 (P2 in Figure 5) emits the high byte of the Program Counter (PCH). Then $\overline{PSEN}$ strobes the EPROM and the code byte is read into the microcontroller.

Program Memory addresses are always 16 bits wide, even though the actual amount of Program Memory used may be less than 64k bytes. External program execution sacrifices two of the 8-bit ports, P0 and P2, to the function of addressing the Program Memory.

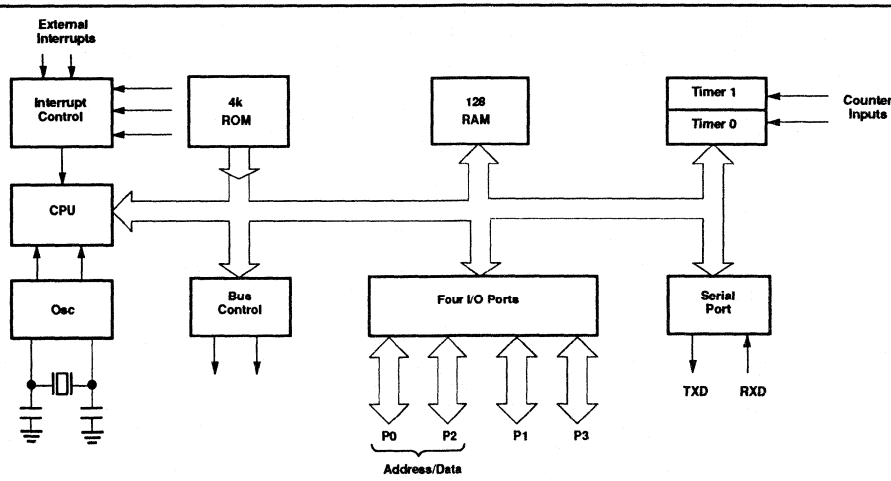


Figure 2. 80C51 Block Diagram

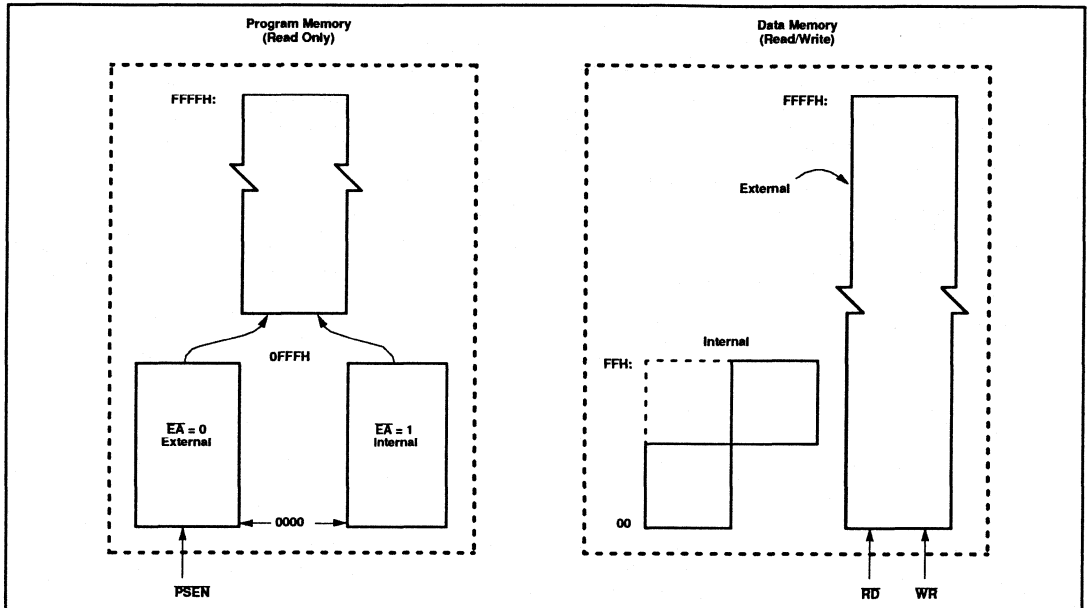


Figure 3. 80C51 Memory Structure

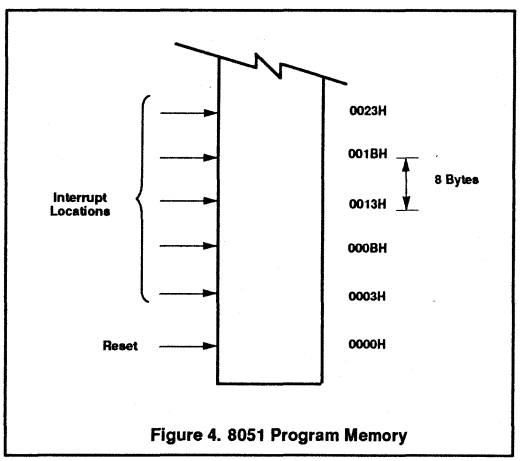


Figure 4. 8051 Program Memory

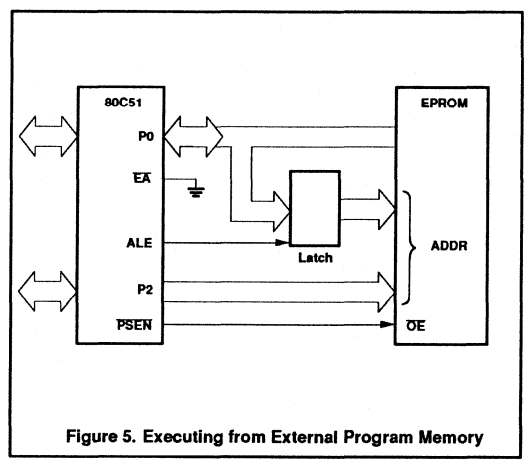


Figure 5. Executing from External Program Memory

DATA MEMORY

The right half of Figure 3 shows the internal and external Data Memory spaces available to the 8051 user. Figure 6 shows a hardware configuration for accessing up to 2k bytes of external RAM. The CPU in this case is executing from internal ROM. Port 0 serves as a multiplexed address/data bus to the RAM, and 3 lines of Port 2 are being used to page the RAM. The CPU generates $\overline{RD}$ and $\overline{WR}$ signals as needed during external RAM accesses. There can be up to 64k bytes of external Data Memory. External Data Memory addresses can be either 1 or 2 bytes wide. One-byte addresses are often used in conjunction with one or more other I/O lines to page the RAM, as shown in Figure 6.

Two-byte addresses can also be used, in which case the high address byte is emitted at Port 2.

Internal Data Memory is mapped in Figure 7. The memory space is shown divided into three blocks, which are generally referred to as the Lower 128, the Upper 128, and SFR space.

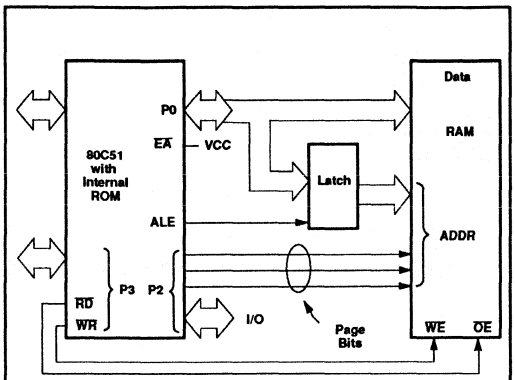


Figure 6. Accessing External Data Memory If the Program Memory Is Internal, the other Bits of P2 Are Available as I/O

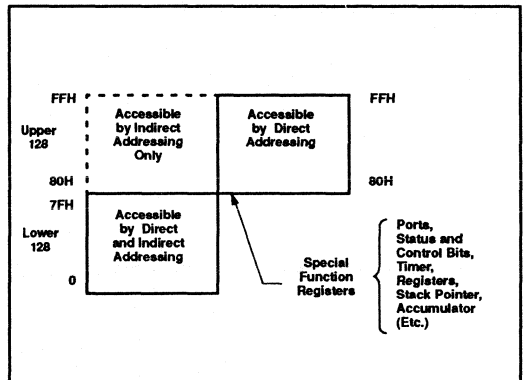


Figure 7. Internal Data Memory

Internal Data Memory addresses are always one byte wide, which implies an address space of only 256 bytes. However, the addressing modes for internal RAM can in fact accommodate 384 bytes, using a simple trick. Direct addresses higher than 7FH access one memory space, and indirect addresses higher than 7FH0 access a different memory space. Thus Figure 7 shows the Upper 128 and SFR space occupying the same block of addresses, 80H through FFH, although they are physically separate entities.

The Lower 128 bytes of RAM are present in all 80C51 devices as mapped in Figure 8.

The lowest 32 bytes are grouped into 4 banks of 8 registers. Program instructions call out these registers as R0 through R7. Two bits in the Program Status Word (PSW) select which register bank is in use. This allows more efficient use of code space, since register instructions are shorter than instructions that use direct addressing.

The next 16 bytes above the register banks form a block of bit-addressable memory space. The 80C51 instruction set includes a wide selection of single-bit instructions, and the 128 bits in this area can be directly addressed by these instructions. The bit addresses in this area are 00H through 7FH.

All of the bytes in the Lower 128 can be accessed by either direct or indirect addressing. The Upper 128 (Figure 9) can only be accessed by indirect addressing.

Figure 10 gives a brief look at the Special Function Register (SFR) space. SFRs include the Port latches, timers, peripheral controls, etc. These registers can only be accessed by direct addressing. Sixteen addresses in SFR space are both byte- and bit-addressable. The bit-addressable SFRs are those whose address ends in 0H or 8H.

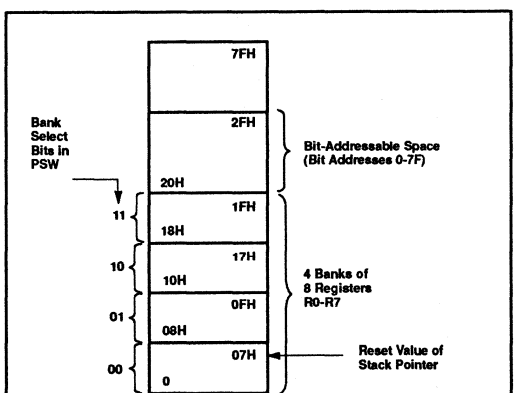


Figure 8. Lower 128 Bytes of Internal RAM

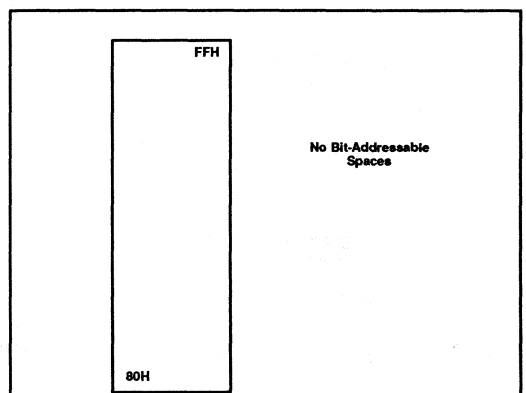


Figure 9. Upper 128 Bytes of Internal RAM

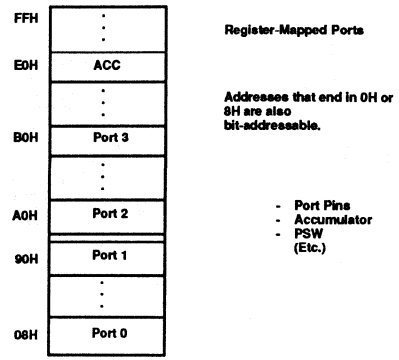


Figure 10. SFR Space

80C51 FAMILY INSTRUCTION SET

The 80C51 instruction set is optimized for 8-bit control applications. It provides a variety of fast addressing modes for accessing the internal RAM to facilitate byte operations on small data structures. The instruction set provides extensive support for one-bit variables as a separate data type, allowing direct bit manipulation in control and logic systems that require Boolean processing.

Program Status Word

The Program Status Word (PSW) contains several status bits that reflect the current state of the CPU. The PSW, shown in Figure 11, resides in the SFR space. It contains the Carry bit, the Auxiliary Carry (for BCD operations), the two register bank select bits, the Overflow flag, a Parity bit, and two user-definable status flags.

The Carry bit, other than serving the function of a Carry bit in arithmetic operations, also serves as the "Accumulator" for a number of Boolean operations.

The bits RS0 and RS1 are used to select one of the four register banks shown in Figure 8. A number of instructions refer to these RAM locations as R0 through R7. The selection of which of the four is being referred to is made on the basis of the RS0 and RS1 at execution time.

The Parity bit reflects the number of 1s in the Accumulator: P = 1 if the Accumulator contains an odd number of 1s, and P = 0 if the Accumulator contains an even number of 1s. Thus the number of 1s in the Accumulator plus P is always even. Two bits in the PSW are uncommitted and may be used as general purpose status flags.

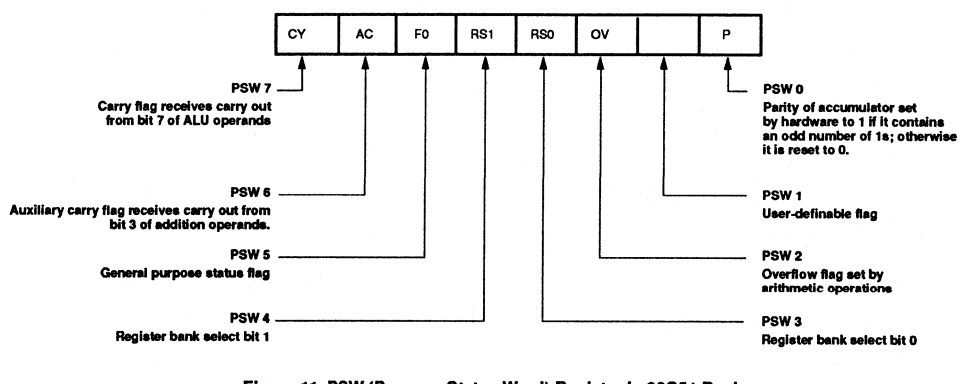


Figure 11. PSW (Program Status Word) Register in 80C51 Devices

Addressing Modes

The addressing modes in the 80C51 instruction set are as follows:

Direct Addressing

In direct addressing the operand is specified by an 8-bit address field in the instruction. Only internal Data RAM and SFRs can be directly addressed.

Indirect Addressing

In indirect addressing the instruction specifies a register which contains the address of the operand. Both internal and external RAM can be indirectly addressed.

The address register for 8-bit addresses can be R0 or R1 of the selected bank, or the Stack Pointer. The address register for 16-bit addresses can only be the 16-bit "data pointer" register, DPTR.

Register Instructions

The register banks, containing registers R0 through R7, can be accessed by certain instructions which carry a 3-bit register specification within the opcode of the instruction. Instructions that access the registers this way are code efficient, since this mode eliminates an address byte. When the instruction is executed, one of the eight registers in the selected bank is accessed. One of four banks is selected at execution time by the two bank select bits in the PSW.

Register-Specific Instructions

Some instructions are specific to a certain register. For example, some instructions always operate on the Accumulator, or Data Pointer, etc., so no address byte is needed to point to it. The opcode itself does that. Instructions that refer to the Accumulator as A assemble as accumulator specific opcodes.

Immediate Constants

The value of a constant can follow the opcode in Program Memory. For example,

```
MOV  A, #100
```

loads the Accumulator with the decimal number 100. The same number could be specified in hex digits as 64H.

Indexed Addressing

Only program Memory can be accessed with indexed addressing, and it can only be read. This addressing mode is intended for reading look-up tables in Program Memory A 16-bit base register (either DPTR or the Program Counter) points to the base of the table, and the Accumulator is set up with the table entry number.

The address of the table entry in Program Memory is formed by adding the Accumulator data to the base pointer.

Another type of indexed addressing is used in the "case jump" instruction. In this case the destination address of a jump instruction is computed as the sum of the base pointer and the Accumulator data.

Arithmetic Instructions

The menu of arithmetic instructions is listed in Table 3. The table indicates the addressing modes that can be used with each instruction to access the <byte> operand. For example, the ADD A,<byte> instruction can be written as:

```
ADD   a, 7FH (direct addressing)
ADD   A, @R0 (indirect addressing)
ADD   a, R7 (register addressing)
ADD   A, #127 (immediate constant)
```

The execution times listed in Table 3 assume a 12MHz clock frequency. All of the arithmetic instructions execute in 1µs except the INC DPTR instruction, which takes 2µs, and the Multiply and Divide instructions, which take 4µs.

Note that any byte in the internal Data Memory space can be incremented without going through the Accumulator.

One of the INC instructions operates on the 16-bit Data Pointer. The Data Pointer is used to generate 16-bit addresses for external memory, so being able to increment it in one 16-bit operation is a useful feature.

The MUL AB instruction multiplies the Accumulator by the data in the B register and puts the 16-bit product into the concatenated B and Accumulator registers.

The DIV AB instruction divides the Accumulator by the data in the B register and leaves the 8-bit quotient in the Accumulator, and the 8-bit remainder in the B register.

Oddly enough, DIV AB finds less use in arithmetic "divide" routines than in radix conversions and programmable shift operations. An example of the use of DIV AB in a radix conversion will be given later. In shift operations, dividing a number by 2n shifts its n bits to the right. Using DIV AB to perform the division completes the shift in 4µs and leaves the B register holding the bits that were shifted out. The DA A instruction is for BCD arithmetic operations. In BCD arithmetic, ADD and ADDC instructions should always be followed by a DA A operation, to ensure that the result is also in BCD. Note that DA A will not convert a binary number to BCD. The DA A operation produces a meaningful result only as the second step in the addition of two BCD bytes.

Table 3. 80C51 Arithmetic Instructions

MNEMONIC	OPERATION	ADDRESSING MODES				EXECUTION
		DIR	IND	REG	IMM	TIME (µs)
ADD A,<byte>	A = A + <byte>	X	X	X	X	1
ADDC A,<byte>	A = A + <byte> + C	X	X	X	X	1
SUBB A,<byte>	A = A – <byte> – C	X	X	X	X	1
INC A	A = A + 1	Accumulator only				1
INC <byte>	<byte> = <byte> + 1	X	X	X		1
INC DPTR	DPTR = DPTR + 1	Data Pointer only				2
DEC A	A = A – 1	Accumulator only				1
DEC <byte>	<byte> = <byte> – 1	X	X	X		1
MUL AB	B:A = B x A	ACC and B only				4
DIV AB	A = Int[A/B] B = Mod[A/B]	ACC and B only				4
DA A	Decimal Adjust	Accumulator only				1

Logical Instructions

Table 4 shows the list of 80C51 logical instructions. The instructions that perform Boolean operations (AND, OR, Exclusive OR, NOT) on bytes perform the operation on a bit-by-bit basis. That is, if the Accumulator contains 00110101B and byte contains 01010011B, then:

```
ANL   A, <byte>
```

will leave the Accumulator holding 00010001B.

The addressing modes that can be used to access the <byte> operand are listed in Table 4.

The ANL A, <byte> instruction may take any of the forms:

```
ANL   A,7FH (direct addressing)
ANL   A,@R1 (indirect addressing)
ANL   A,R6 (register addressing)
ANL   A,#53H (immediate constant)
```

All of the logical instructions that are Accumulator-specific execute in 1µs (using a 12MHz clock). The others take 2µs.

Note that Boolean operations can be performed on any byte in the internal Data Memory space without going through the Accumulator. The XRL <byte>, #data instruction, for example, offers a quick and easy way to invert port bits, as in XRL P1, #OFFH.

If the operation is in response to an interrupt, not using the Accumulator saves the time and effort to push it onto the stack in the service routine.

The Rotate instructions (RL, A, RLC A, etc.) shift the Accumulator 1 bit to the left or right. For a left rotation, the MSB rolls into the LSB position. For a right rotation, the LSB rolls into the MSB position.

The SWAP A instruction interchanges the high and low nibbles within the Accumulator. This is a useful operation in BCD manipulations. For example, if the Accumulator contains a binary number which is known to be less than 100, it can be quickly converted to BCD by the following code:

```
MOVE   B,#10
DIV    AB
SWAP   A
ADD    A,B
```

Dividing the number by 10 leaves the tens digit in the low nibble of the Accumulator, and the ones digit in the B register. The SWAP and ADD instructions move the tens digit to the high nibble of the Accumulator, and the ones digit to the low nibble.

Data Transfers

Internal RAM

Table 5 shows the menu of instructions that are available for moving data around within the internal memory spaces, and the addressing modes that can be used with each one. With a 12MHz clock, all of these instructions execute in either 1 or 2µs.

The MOV <dest>, <src> instruction allows data to be transferred between any two internal RAM or SFR locations without going through the Accumulator. Remember, the Upper 128 bytes of data RAM can be accessed only by indirect addressing, and SFR space only by direct addressing.

Note that in 80C51 devices, the stack resides in on-chip RAM, and grows upwards. The PUSH instruction first increments the Stack Pointer (SP), then copies the byte into the stack. PUSH and POP use only direct addressing to identify the byte being saved or restored, but the stack itself is accessed by indirect addressing using the SP register. This means the stack can go into the Upper 128 bytes of RAM, if they are implemented, but not into SFR space.

The Upper 128 bytes of RAM are not implemented in the 80C51 nor in its ROMless or EPROM counterparts. With these devices, if the SP points to the Upper 128, PUSHed bytes are lost, and POPed bytes are indeterminate.

The Data Transfer instructions include a 16-bit MOV that can be used to initialize the Data Pointer (DPTR) for look-up tables in Program Memory, or for 16-bit external Data Memory accesses.

Table 4. 80C51 Logical Instructions

MNEMONIC	OPERATION	ADDRESSING MODES				EXECUTION
		DIR	IND	REG	IMM	TIME (µs)
ANL A,<byte>	A = A.AND. <byte>	X	X	X	X	1
ANL <byte>,A	<byte> = <byte> .AND.A	X				1
ANL <byte>,#data	<byte> = <byte> .AND.#data	X				2
ORL A,<byte>	A = A.OR.<byte>	X	X	X	X	1
ORL <byte>,A	<byte> = <byte> .OR.A	X				1
ORL <byte>,#data	<byte> = <byte> .OR.#data	X				2
XRL A,<byte>	A = A.XOR. <byte>	X	X	X	X	1
XRL <byte>,A	<byte> = <byte> .XOR.A	X				1
XRL <byte>,#data	<byte> = <byte> .XOR.#data	X				2
CRL A	A = 00H	Accumulator only				1
CPL A	A = .NOT.A	Accumulator only				1
RL A	Rotate ACC Left 1 bit	Accumulator only				1
RLC A	Rotate Left through Carry	Accumulator only				1
RR A	Rotate ACC Right 1 bit	Accumulator only				1
RRC A	Rotate Right through Carry	Accumulator only				1
SWAP A	Swap Nibbles in A	Accumulator only				1

Table 5. Data Transfer Instructions that Access Internal Data Memory Space

MNEMONIC	OPERATION	ADDRESSING MODES				EXECUTION
		DIR	IND	REG	IMM	TIME (µs)
MOV A,<src>	A = <src>	X	X	X	X	1
MOV <dest>,A	<dest> = A	X	X	X		1
MOV <dest>,<src>	<dest> = <src>	X	X	X	X	2
MOV DPTR,#data16	DPTR = 16-bit immediate constant				X	2
PUSH <src>	INC SP:MOV"@SP",<src>	X				2
POP <dest>	MOV <dest>,"@SP":DEC SP	X				2
XCH A,<byte>	ACC and <byte> exchange data	X	X	X		1
XCHD A,@Ri	ACC and @Ri exchange low nibbles		X			1

The XCH A, <byte> instruction causes the Accumulator and addressed byte to exchange data. The XCHD A, @Ri instruction is similar, but only the low nibbles are involved in the exchange.

To see how XCH and XCHD can be used to facilitate data manipulations, consider first the problem of shifting an 8-digit BCD number two digits to the right. Figure 12 shows how this can be done using direct MOVs, and for comparison how it can be done using XCH instructions. To aid in understanding how the code works, the contents of the registers that are holding the BCD number and the content of the Accumulator are shown alongside each instruction to indicate their status after the instruction has been executed.

After the routine has been executed, the Accumulator contains the two digits that were shifted out on the right. Doing the routine with direct MOVs uses 14 code bytes and 9µs of execution time (assuming a 12MHz clock).

The same operation with XCHs uses only 9 bytes and executes almost twice as fast.

To right-shift by an odd number of digits, a one-digit shift must be executed.

Figure 13 shows a sample of code that will right-shift a BCD number one digit, using the XCHD instruction. Again, the contents of the registers holding the number and of the Accumulator are shown alongside each instruction.

First, pointers R1 and R0 are set up to point to the two bytes containing the last four BCD digits. Then a loop is executed which leaves the last byte, location 2EH, holding the last two digits of the shifted number. The pointers are decremented, and the loop is repeated for location 2DH. The CJNE instruction (Compare and Jump if Not Equal) is a loop control that will be described later. The loop executed from LOOP to CJNE for R1 = 2EH, 2DH, 2CH, and 2BH. At that point the digit that was originally shifted out on the right has propagated to location 2AH. Since that location should be left with 0s, the lost digit is moved to the Accumulator.

External RAM

Table 6 shows a list of the Data Transfer instructions that access external Data Memory. Only indirect addressing can be used. The choice is whether to use a one-byte address, @Ri, where Ri can be either R0 or R1 of the selected register bank, or a two-byte address, @DPTR. The disadvantage to using 16-bit addresses if only a few k bytes of external RAM are involved is that 16-bit addresses use all 8 bits of Port 2 as address bus. On the other hand, 8-bit addresses allow one to address a few bytes of RAM, as shown in Figure 6, without having to sacrifice all of Port 2.

All of these instructions execute in 2 µs, with a 12MHz clock.

		2A	2B	2C	2D	2E	ACC
MOV	A,2EH	00	12	34	56	78	78
MOV	2EH,2DH	00	12	34	56	56	78
MOV	2DH,2CH	00	12	34	34	56	78
MOV	2CH,2BH	00	12	12	34	56	78
MOV	2BH,#0	00	00	12	34	56	78

A. Using direct MOVs: 14 bytes, 9 µs

		2A	2B	2C	2D	2E	ACC
CLR	A	00	12	34	56	78	00
XCH	A,2BH	00	00	34	56	78	12
XCH	A,2CH	00	00	12	56	78	34
XCH	A,2DH	00	00	12	34	78	56
XCH	A2EH	00	00	12	34	56	78

B. Using XCHs: 9 bytes, 5 µs

Figure 12. Shifting a BCD Number Two Digits to the Right

			2A	2B	2C	2D	2E	ACC
	MOV	R1,#2EH	00	12	34	56	78	XX
	MOV	R0,#2DH	00	12	34	56	78	XX
	loop for R1 = 2EH:							
LOOP:	MOV	A,@R1	00	12	34	56	78	78
	XCHD	A,@R0	00	12	34	58	78	76
	SWAP	A	00	12	34	58	78	67
	MOV	@R1,A	00	12	34	58	67	67
	DEC	R1	00	12	34	58	67	67
	DEC	R0	00	12	34	58	67	67
	CJNE	R1,#2AH,LOOP						
	loop for R1 = 2DH:		00	12	38	45	67	45
	loop for R1 = 2CH:		00	18	23	45	67	23
	loop for R1 = 2BH:		08	01	23	45	67	01
	CLR	A	08	01	23	45	67	00
	XCH	A,2AH	00	01	23	45	67	08

Figure 13. Shifting a BCD Number One Digit to the Right

Table 6. 80C51 Data Transfer Instructions that Access External Data Memory Space

ADDRESS WIDTH	MNEMONIC	OPERATION	EXECUTION TIME (μs)
8 bits	MOVX A,@Ri	Read external RAM @Ri	2
8 bits	MOVX @Ri,A	Write external RAM @ Ri	2
16 bits	MOVX A,@DPTR	Read external RAM @ DPTR	2
16 bits	MOVX @DPTR,A	Write external RAM @ DPTR	2

Note that in all external Data RAM accesses, the Accumulator is always either the destination or source of the data.

The read and write strobes to external RAM are activated only during the execution of a MOVX instruction. Normally these signals are inactive, and in fact if they're not going to be used at all, their pins are available as extra I/O lines.

Lookup Tables

Table 7 shows the two instructions that are available for reading lookup tables in Program Memory. Since these instructions access only Program Memory, the lookup tables can only be read, not updated.

If the table access is to external Program Memory, then the read strobe is $\overline{PSEN}$.

The mnemonic is MOVC for "move constant." The first MOVC instruction in Table 7 can accommodate a table of up to 256 entries numbered 0 through 255. The number of the desired entry is loaded into the Accumulator, and the Data Pointer is set up to point to the beginning of the table. Then:

```
MOVC    A,@A+DPTR
```

copies the desired table entry into the Accumulator.

The other MOVC instruction works the same way, except the Program Counter (PC) is used as the table base, and the table is accessed through a subroutine. First the number of the desired entry is loaded into the Accumulator, and the subroutine is called:

```
MOV     A,ENTRY NUMBER
CALL    TABLE
```

The subroutine "TABLE" would look like this:

```
TABLE:MOVC      A,@A+PC
      RET
```

The table itself immediately follows the RET (return) instruction in Program Memory. This type of table can have up to 255 entries, numbered 1 through 255. Number 0 cannot be used, because at the time the MOVC instruction is executed, the PC contains the address of the RET instruction. An entry numbered 0 would be the RET opcode itself.

Boolean Instructions

80C51 devices contain a complete Boolean (single-bit) processor. The internal RAM contains 128 addressable bits, and the SFR space can support up to 128 addressable bits as well. All of the port lines are bit-addressable, and each one can be treated as a separate single-bit port. The instructions that access these bits are not just conditional branches, but a complete menu of move, set, clear, complement, OR, and AND instructions. These kinds of bit operations are not easily obtained in other architectures with any amount of byte-oriented software.

The instruction set for the Boolean processor is shown in Table 8. All bit accesses are by direct addressing.

Bit addresses 00H through 7FH are in the Lower 128, and bit addresses 80H through FFH are in SFR space.

Note how easily an internal flag can be moved to a port pin:

```
MOV     C,FLAG
MOV     P1.0,C
```

In this example, FLAG is the name of any addressable bit in the Lower 128 or SFR space. An I/O line (the LSB of Port 1, in this case) is set or cleared depending on whether the flag bit is 1 or 0.

The Carry bit in the PSW is used as the single-bit Accumulator of the Boolean processor. Bit instructions that refer to the Carry bit as C assemble as Carry-specific instructions (CLR C, etc.). The Carry bit also has a direct address, since it resides in the PSW register, which is bit-addressable.

Note that the Boolean instruction set includes ANL and ORL operations, but not the XRL (Exclusive OR) operation. An XRL operation is simple to implement in software. Suppose, for example, it is required to form the Exclusive OR of two bits:

```
C = bit1 .XRL. bit2
```

The software to do that could be as follows:

```
      MOV       C,bit1
      JNB       bit2,OVER
      CPL       C
OVER: (continue)
```

First, bit1 is moved to the Carry. If bit2 = 0, then C now contains the correct result. That is, bit1 .XRL. bit2 = bit1 if bit2 = 0. On the other hand, if bit2 = 1, C now contains the complement of the correct result. It need only be inverted (CPL C) to complete the operation.

This code uses the JNB instruction, one of a series of bit-test instructions which execute a jump if the addressed bit is set (JC, JB, JBC) or if the addressed bit is not set (JNC, JNB). In the above case, bit2 is being tested, and if bit2 = 0, the CPL C instruction is jumped over.

JBC executes the jump if the addressed bit is set, and also clears the bit. Thus a flag can be tested and cleared in one operation. All the PSW bits are directly addressable, so the Parity bit, or the general purpose flags, for example, are also available to the bit-test instructions.

Table 7. 80C51 Lookup Table Read Instructions

MNEMONIC	OPERATION	EXECUTION TIME (μs)
MOVC A,@A+DPTR	Read program memory at (A + DPTR)	2
MOVC A,@A+PC	Read program memory at (A + PC)	2

Table 8. 80C51 Boolean Instructions

MNEMONIC		OPERATION	EXECUTION TIME (μs)
ANL	C,bit	C = C.AND.bit	2
ANL	C,/bit	C = C.AND..NOT.bit	2
ORL	C,bit	C = C.OR.bit	2
ORL	C,/bit	C = C.OR..NOT.bit	2
MOV	C,bit	C = bit	1
MOV	bit,C	bit = C	2
CLR	C	C = 0	1
CLR	bit	bit = 0	1
SETB	C	C = 1	1
SETB	bit	bit = 1	1
CPL	C	C = .NOT.C	1
CPL	bit	bit = .NOT.bit	1
JC	rel	Jump if C = 1	2
JNC	rel	Jump if C = 0	2
JB	bit,rel	Jump if bit = 1	2
JNB	bit,rel	Jump if bit = 0	2
JBC	bit,rel	Jump if bit = 1; CLR bit	2

Relative Offset

The destination address for these jumps is specified to the assembler by a label or by an actual address in Program memory. However, the destination address assembles to a relative offset byte. This is a signed (two's complement) offset byte which is added to the PC in two's complement arithmetic if the jump is executed. The range of the jump is therefore –128 to +127 Program Memory bytes relative to the first byte following the instruction.

Jump Instructions

Table 9 shows the list of unconditional jumps with execution time for a 12MHz clock.

The table lists a single "JMP addr" instruction, but in fact there are three SJMP, LJMP, and AJMP, which differ in the format of the destination address. JMP is a generic mnemonic which can be used if the programmer does not care which way the jump is encoded.

The SJMP instruction encodes the destination address as a relative offset, as described above. The instruction is 2 bytes long, consisting of the opcode and the relative offset byte. The jump distance is limited to a range of –128 to +127 bytes relative to the instruction following the SJMP.

The LJMP instruction encodes the destination address as a 16-bit constant. The instruction is 3 bytes long, consisting of the opcode and two address bytes. The destination address can be anywhere in the 64k Program Memory space.

The AJMP instruction encodes the destination address as an 11-bit constant. The instruction is 2 bytes long, consisting of the opcode, which itself contains 3 of the 11 address bits, followed by another byte containing the low 8 bits of the destination address. When the instruction is executed, these 11 bits are simply substituted for the low 11 bits in the PC. The high 5 bits stay the same. Hence the destination has to be within the same 2k block as the instruction following the AJMP.

In all cases the programmer specifies the destination address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the destination address into the correct format for the given instruction. If the format required by the instruction will not support the distance to the specified destination address, a "Destination out of range" message is written into the List file.

The JMP @A+DPTR instruction supports case jumps. The destination address is computed at execution time as the sum of the 16-bit DPTR register and the Accumulator. Typically, DPTR is set up with the address of a jump table. In a 5-way branch, for example, an integer 0 through 4 is loaded into the Accumulator. The code to be executed might be as follows:

```
MOV   DPTR,#JUMP TABLE
MOV   A,INDEX_NUMBER
RL    A
JMP   @A+DPTR
```

Table 9. Unconditional Jumps in 80C51 Devices

MNEMONIC		OPERATION	EXECUTION TIME (μs)
JMP	addr	Jump to addr	2
JMP	@A+DPTR	Jump to A + DPTR	2
CALL	addr	Call subroutine at addr	2
RET		Return from subroutine	2
RETI		Return from interrupt	2
NOP		No operation	1

The RL A instruction converts the index number (0 through 4) to an even number on the range 0 through 8, because each entry in the jump table is 2 bytes long:

```
JUMP TABLE:
        AJMP    CASE 0
        AJMP    CASE 1
        AJMP    CASE 2
        AJMP    CASE 3
        AJMP    CASE 4
```

Table 9 shows a single "CALL addr" instruction, but there are two of them, LCALL and ACALL, which differ in the format in which the subroutine address is given to the CPU. CALL is a generic mnemonic which can be used if the programmer does not care which way the address is encoded.

The LCALL instruction uses the 16-bit address format, and the subroutine can be anywhere in the 674k Program Memory space. The ACALL instruction uses the 11-bit format, and the subroutine must be in the same 2k block as the instruction following the ACALL.

In any case, the programmer specifies the subroutine address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the address into the correct format for the given instructions.

Subroutines should end with a RET instruction, which returns execution to the instruction following the CALL.

RETI is used to return from an interrupt service routine. The only difference between RET and RETI is that RETI tells the interrupt control system that the interrupt in progress is done. If there is no interrupt in progress at the time RETI is executed, then the RETI is functionally identical to RET.

Table 10 shows the list of conditional jumps available to the 80C51 user. All of these jumps specify the destination address by the relative offset method, and so are limited to a jump distance of –128 to +127 bytes from the instruction following the conditional jump instruction. Important to note, however, the user specifies to the assembler the actual destination address the same way as the other jumps: as a label or a 16-bit constant.

There is no Zero bit in the PSW. The JZ and JNZ instructions test the Accumulator data for that condition.

The DJNZ instruction (Decrement and Jump if Not Zero) is for loop control. To execute a loop N times, load a counter byte with N and terminate the loop with a DJNZ to the beginning of the loop, as shown below for N = 10.

```
        MOV         COUNTER,#10
LOOP:   (begin loop)
          •
          •
          •
        (end loop)
        DJNZ        COUNTER,LOOP
        (continue)
```

The CJNE instruction (Compare and Jump if Not Equal) can also be used for loop control as in Figure 13. Two bytes are specified in the operand field of the instruction. The jump is executed only if the two bytes are not equal. In the example of Figure 13, the two bytes were data in R1 and the constant 2AH. The initial data in R1 was 2EH. Every time the loop was executed, R1 was decremented, and the looping was to continue until the R1 data reached 2AH.

Another application of this instruction is in "greater than, less than" comparisons. The two bytes in the operand field are taken as unsigned integers. If the first is less than the second, then the Carry bit is set (1). If the first is greater than or equal to the second, then the Carry bit is cleared.

CPU Timing

All 80C51 microcontrollers have an on-chip oscillator which can be used if desired as the clock source for the CPU. To use the on-chip oscillator, connect a crystal or ceramic resonator between the XTAL1 and XTAL2 pins of the microcontroller, and capacitors to ground as shown in Figure 14.

Examples of how to drive the clock with an external oscillator are shown in Figure 15. Note that in the NMOS devices (8051, etc.) the signal at the XTAL2 pin actually drives the internal clock generator. In the CMOS devices (80C51, etc.), the signal at the XTAL1 pin drives the internal clock generator. The internal clock generator defines the sequence of states that make up the 80C51 machine cycle.

Machine Cycles

A machine cycle consists of a sequence of 6 states, numbered S1 through S6. Each state time lasts for two oscillator periods. Thus a machine cycle takes 12 oscillator periods or 1μs if the oscillator frequency is 12MHz.

Each state is divided into a Phase 1 half and a Phase 2 half. Figure 16 shows that fetch/execute sequences in states and phases for various kinds of instructions. Normally two program fetches are generated during each machine cycle, even if the instruction being executed doesn't require it. If the instruction being executed doesn't need more code bytes, the CPU simply ignores the extra fetch, and the Program Counter is not incremented.

Execution of a one-cycle instruction (Figure 16A and B) begins during State 1 of the machine cycle, when the opcode is latched into the Instruction Register. A second fetch occurs during S4 of the same machine cycle. Execution is complete at the end of State 6 of this machine cycle.

The MOVX instructions take two machine cycles to execute. No program fetch is generated during the second cycle of a MOVX instruction. This is the only time program fetches are skipped. The fetch/execute sequence for MOVX instructions is shown in Figure 16D.

The fetch/execute sequences are the same whether the Program Memory is internal or external to the chip. Execution times do not depend on whether the Program Memory is internal or external.

Table 10. Conditional Jumps in 80C51 Devices

MNEMONIC	OPERATION	ADDRESSING MODES				EXECUTION
		DIR	IND	REG	IMM	TIME (μs)
JZ rel	Jump if A = 0	Accumulator only				2
JNZ rel	Jump if A ≠ 0	Accumulator only				2
DJNZ <byte>,rel	Decrement and jump if not zero	X		X		2
CJNE A,<byte>,rel	Jump if A ≠ <byte>	X			X	2
CJNE <byte>,#data,rel	Jump if <byte> ≠ #data		X	X		2

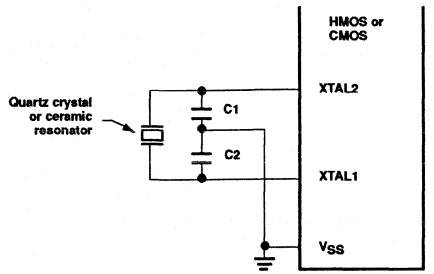


Figure 14. Using the On-Chip Oscillator

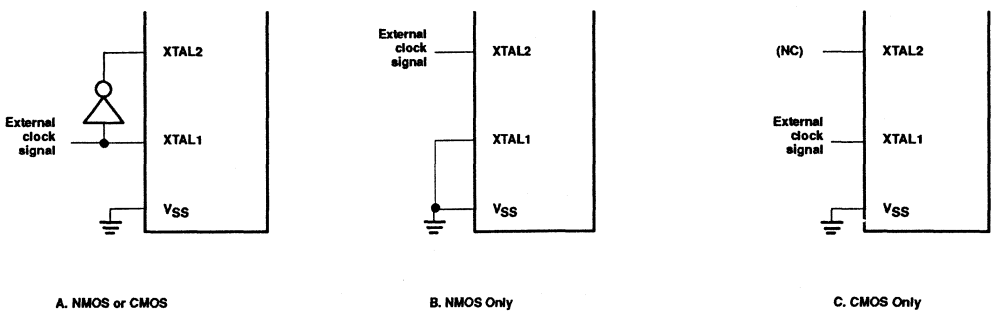


Figure 15. Using an External Clock

Figure 17 shows the signals and timing involved in program fetches when the Program Memory is external. If Program Memory is external, then the Program Memory read strobe $\overline{PSEN}$ is normally activated twice per machine cycle, as shown in Figure 17A. If an access to external Data Memory occurs, as shown in Figure 17B, two $\overline{PSEN}$s are skipped, because the address and data bus are being used for the Data Memory access.

Note that a Data Memory bus cycle takes twice as much time as a Program Memory bus cycle. Figure 17 shows the relative timing of the addresses being emitted at Ports 0 and 2, and of ALE and $\overline{PSEN}$. ALE is used to latch the low address byte from P0 into the address latch.

When the CPU is executing from internal Program Memory, PSEN is not activated, and program addresses are not emitted. However, ALE continues to be activated twice per machine cycle and so it is available as a clock output signal. Note, however, that one ALE is skipped during the execution of the MOVX instruction.

Interrupt Structure

The 80C51 and its ROMless and EPROM versions have 5 interrupt sources: 2 external interrupts, 2 timer interrupts, and the serial port interrupt.

What follows is an overview of the interrupt structure for the device. More detailed information for specific members of the 80C51 derivative family is provided in later chapters of this user's guide.

Interrupt Enables

Each interrupt sources can be individually enabled or disabled by setting or clearing a bit in the SFR named IE (Interrupt Enable). This register also contains a global disable bit, which can be cleared to disable all interrupts at once. Figure 18 shows the IE register.

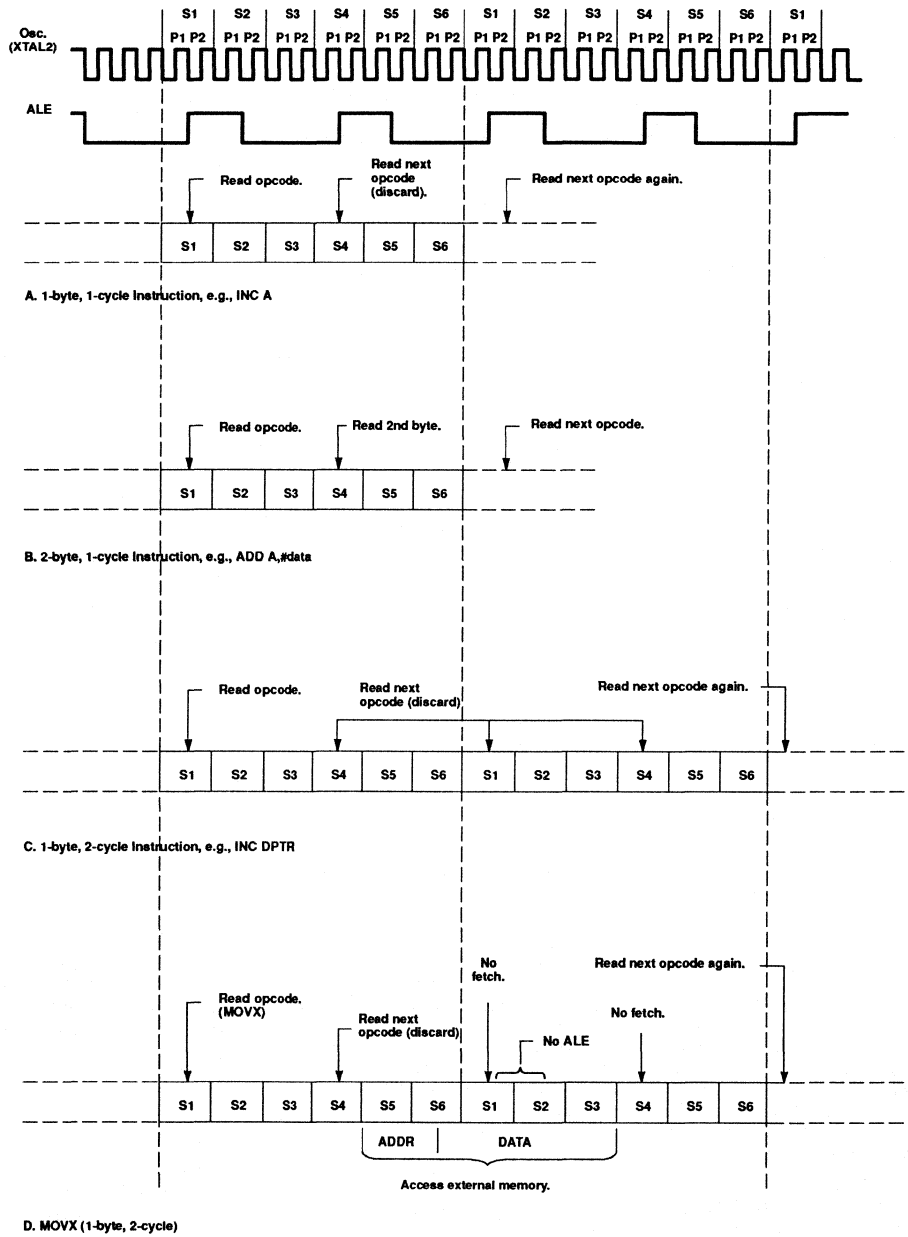
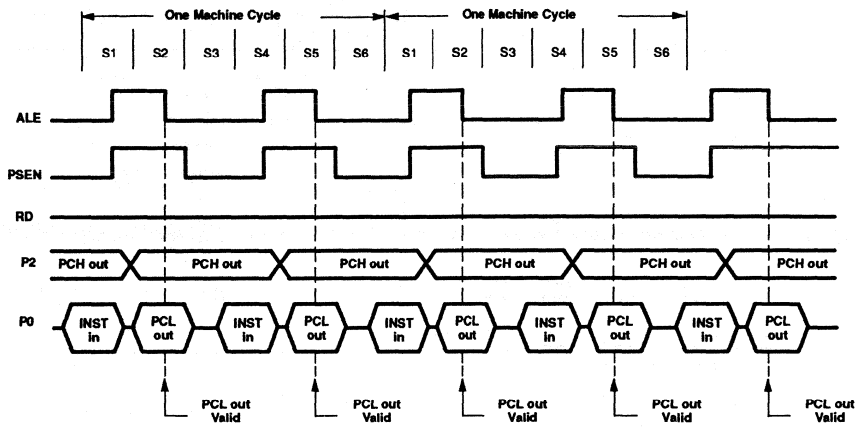
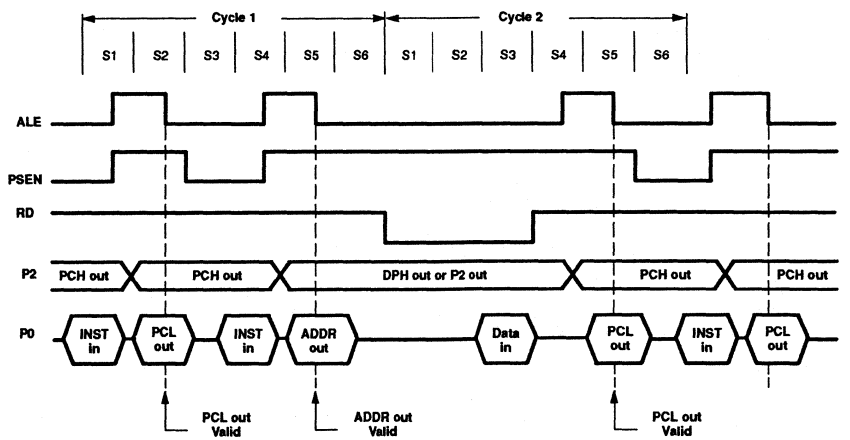


Figure 16. State Sequence in 80C51 Family Devices



A. Without a MOVX



B. With a MOVX

Figure 17. Bus Cycles in 80C51 Family Devices Executing from External Program Memory

(MSB)							(LSB)
EA	X	X	ES	ET1	EX1	ET0	EX0

Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
	IE.6	Reserved.
	IE.5	Reserved.
ES	IE.4	Enables or disables the Serial Port interrupt. If ES = 0, the Serial Port interrupt is disabled.
ET1	IE.3	Enables or disables the Timer 1 Overflow interrupt. If ET1 = 0, the Timer 1 interrupt is disabled.
EX1	IE.2	Enables or disables External Interrupt.1. If EX1 = 0, External Interrupt 1 is disabled.
ET0	IE.1	Enables or disables the Timer 0 Overflow interrupt.. If ET0 = 0, the Timer 0 interrupt is disabled.
EX0	IE.0	Enables or disables External Interrupt. 0. If EX0 = 0, External Interrupt 0 is disabled.

Figure 18. Interrupt Enable (IE) Register

(MSB)							(LSB)
X	X	X	PS	PT1	PX1	PT0	PX0

Symbol	Position	Function
	IP.7	Reserved.
	IP.6	Reserved.
	IP.5	Reserved.
PS	IP.4	Defines the Serial Port interrupt priority level. PS = 1 programs it to the higher priority level.
PT1	IP.3	Defines theTimer 1 interrupt priority level. PT1 = 1 programs it to the higher priority level.
PX1	IP.2	Defines the External Interrupt 1 priority level. PX1 = 1 programs it to the higher priority level.
PT0	IP.1	Enables or disables the Timer 0 Interrupt priority level. PT0 = 1 programs it to the higher priority level.
PX0	IP.0	Defines the External Interrupt 0 priority level. PX0 = 1 programs it to the higher priority level.

Figure 19. Interrupt Priority (IP) Register

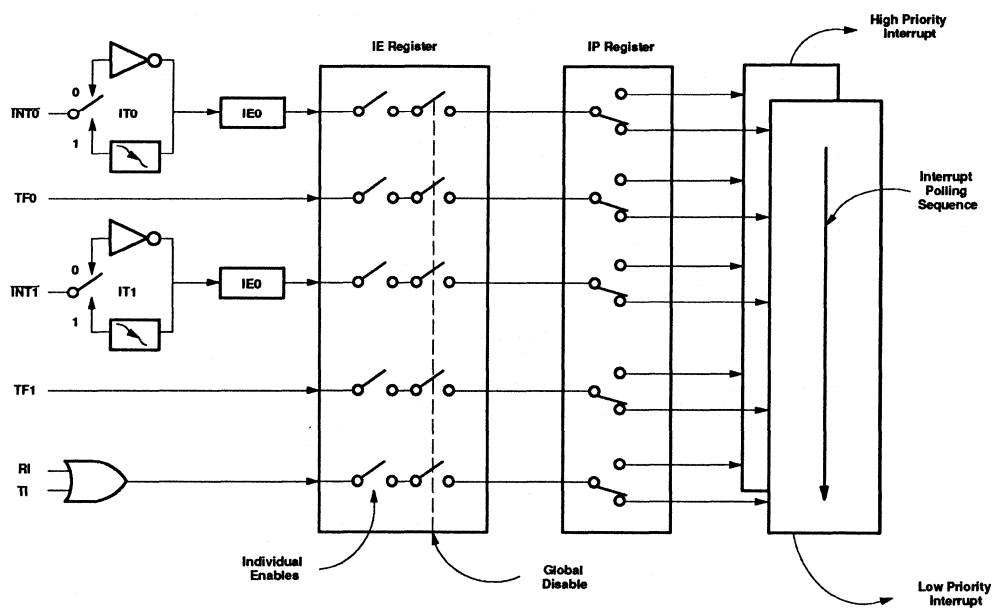


Figure 20. Interrupt Control System

Interrupt Priorities

Each interrupt source can also be individually programmed to one of two priority levels by setting or clearing a bit in the SFR named IP (Interrupt Priority). Figure 19 shows the IP register. A low-priority interrupt can be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

If two interrupt requests of different priority levels are received simultaneously, the request of higher priority is serviced. If interrupt requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence. Figure 20 shows how the IE and IP registers and the polling sequence work to determine which if any interrupt will be serviced.

In operation, all the interrupt flags are latched into the interrupt control system during State 5 of every machine cycle. The samples are polled during the following machine cycle. If the flag for an enabled interrupt is found to be set (1), the interrupt system generates an LCALL to the appropriate location in Program Memory, unless some other condition blocks the interrupt. Several conditions can block an interrupt, among them that an interrupt of equal or higher priority level is already in progress.

The hardware-generated LCALL causes the contents of the Program Counter to be pushed into the stack, and reloads the PC with the beginning address of the service routine. As previously noted (Figure 4), the service routine for each interrupt begins at a fixed location.

Only the Program Counter is automatically pushed onto the stack, not the PSW or any other register. Having only the PC automatically saved allows the programmer to decide how much time should be spent saving other registers. This enhances the interrupt response time, albeit at the expense of increasing the programmer's burden of responsibility. As a result, many interrupt functions that are typical in control applications toggling a port pin for example, or reloading a timer, or unloading a serial buffer can often be completed in less time than it takes other architectures to complete.

Simulating a Third Priority Level in Software

Some applications require more than two priority levels that are provided by on-chip hardware in 80C51 devices. In these cases, relatively simple software can be written to produce the same effect as a third priority level. First, interrupts that are to have higher priority than 1 are assigned to priority 1 in the Interrupt Priority (IP) register. The service routines for priority 1 interrupts that are supposed to be interruptable by priority 2 interrupts are written to include the following code:

```
        PUSH    IE
        MOV     IE,#MASK
        CALL    LABEL
        **************************
        (execute service routine)
        **************************
        POP     IE
        RET
LABEL:  RETI
```

As soon as any priority interrupt is acknowledged, the Interrupt Enable (IE) register is redefined so as to disable all but priority 2 interrupts. Then a CALL to LABEL executes the RETI instruction, which clears the priority 1 interrupt-in-progress flip-flop. At this point any priority 1 interrupt that is enabled can be serviced, but only priority 2 interrupts are enabled.

POPing IE restores the original enable byte. Then a normal RET (rather than another RETI) is used to terminate the service routine. The additional software adds 10µs (at 12MHz) to priority 1 interrupts.

HARDWARE DESCRIPTION

This chapter provides a detailed description of the 80C51 microcontroller (see Figure 21). Included in this description are:

- The port drivers and how they function both as ports and, for Ports 0 and 2, in bus operations
- The Timers/Counters
- The Serial Interface
- The Interrupt System
- Reset
- The Reduced Power Modes in CMOS devices
- The EPROM version of the 80C51

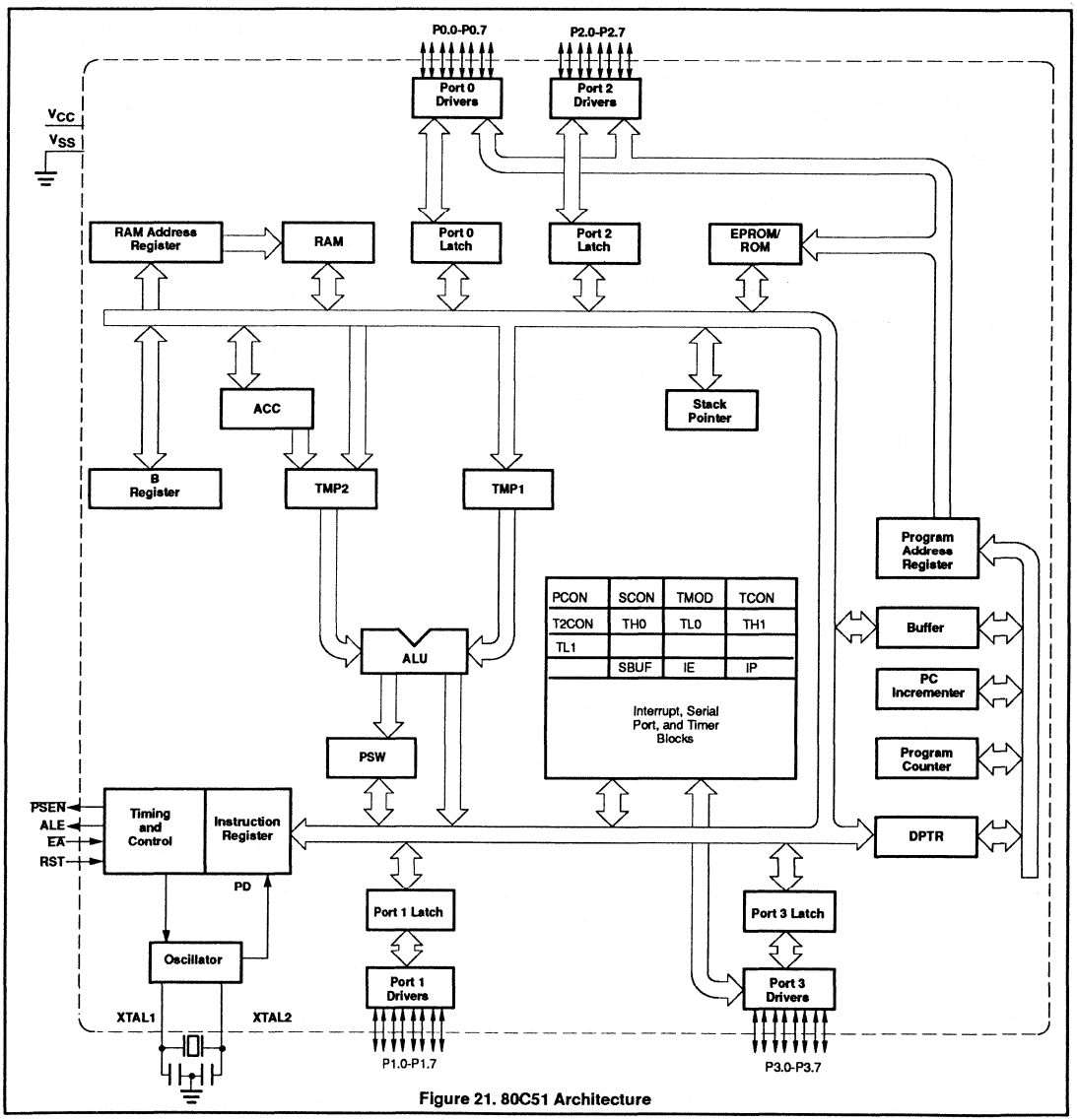


Figure 21. 80C51 Architecture

Special Function Registers

A Map of the on-chip memory area called the Special Function Register (SFR) space is shown in Figure 22.

Note that in the SFRs not all of the addresses are occupied. Unoccupied addresses are not implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have no effect.

User software should not write 1s to these unimplemented locations, since they may be used in other 80C51 Family derivative products to invoke new features. The functions of the SFRs are described in the text that follows.

Accumulator
ACC is the Accumulator register. The mnemonics for Accumulator-Specific instructions, however, refer to the Accumulator simply as A.

B Register
The B register is used during multiply and divide operations. For other instructions it can be treated as another scratch pad register.

Program StatusWord
The PSW register contains program status information as detailed in Figure 23.

Stack Pointer
The Stack Pointer register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on-chip RAM, the Stack Pointer is initialized to 07H after a reset. This causes the stack to begin at locations 08H.

Data Pointer
The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address. It may be manipulated as a 16-bit register or as two independent 8-bit registers.

Ports 0 to 3
P0, P1, P2, and P3 are the SFR latches of Ports 0, 1, 2, and 3, respectively. Writing a one to a bit of a port SFR (P0, P1, P2, or P3) causes the corresponding port output pin to switch high. Writing a zero causes the port output pin to switch low. When used as an input, the external state of a port pin will be held in the port SFR (i.e., if the external state of a pin is low, the corresponding port SFR bit will contain a 0; if it is high, the bit will contain a 1).

Serial Data Buffer
The Serial Buffer is actually two separate registers, a transmit buffer and a receive buffer. When data is moved to SBUF, it goes to the transmit buffer and is held for serial transmission. (Moving a byte to SBUF is what initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

Timer Registers Basic to 80C51
Register pairs (TH0, TL0), and (TH1, TL1) are the 16-bit Counting registers for Timer/Counters 0 and 1, respectively.

Control Register for the 80C51
Special Function Registers IP, IE, TMOD, TCON, SCON, and PCON contain control and status bits for the interrupt system, the Timer/Counters, and the serial port. They are described in later sections.

Port Structures and Operation

All four ports in the 80C51 are bidirectional. Each consists of a latch (Special Function Registers P0 through P3), an output driver, and an input buffer.

The output drivers of Ports 0 and 2, and the input buffers of Port 0, are used in accesses to external memory. In this application, Port 0 outputs the low byte of the external memory address, time-multiplexed with the byte being written or read.

Port 2 outputs the high byte of the external memory address when the address is 16 bits wide. Otherwise, the Port 2 pins continue to emit the P2 SFR content.

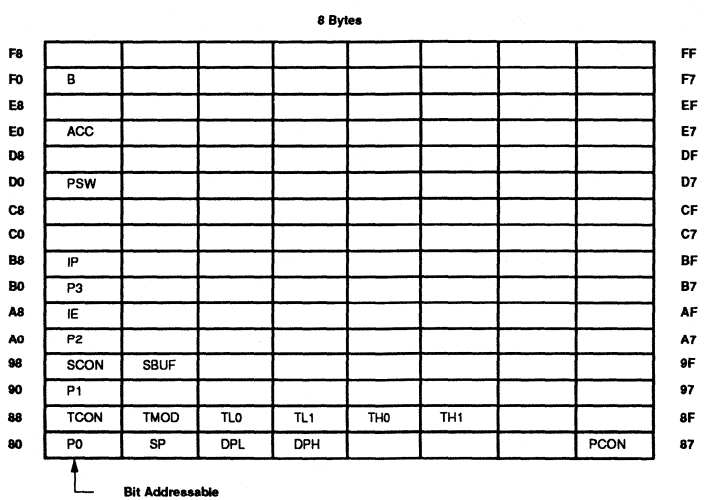
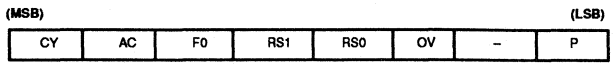


Figure 22. 80C51 SFR Memory Map



Symbol	Position	Name and Significance
CY	PSW.7	Carry flag.
AC	PSW.6	Auxiliary Carry flag. (For BCD operations.)
F0	PSW.5	Flag 0. (Available to the user for general purposes.)
RS1	PSW.4	Register bank select control bits 1 and 0. Set/cleared by software to determine working register bank (see Note).
RS0	PSW.3	

Symbol	Position	Name and Significance
OV	PSW.2	Overflow flag.
–	PSW.1	User-definable flag.
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of "one" bits in the Accumulator, i.e., even parity.

NOTE:
The contents of (RS1, RS0) enable the working register banks as follows:

(0,0)—Bank 0 (00H-07H)
(0,1)—Bank 1 (08H-0FH)
(1,0)—Bank 2 (10H-17H)
(1,1)—Bank 3 (18H-17H)

Figure 23. Program Status Word (PSW) Register

All the Port 3 pins are multifunctional. They are not only port pins, but also serve the functions of various special features as listed below:

Port Pin	Alternate Function
P3.0	RxD (serial input port)
P3.1	TxD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt)
P3.3	$\overline{INT1}$ (external interrupt)
P3.4	T0 (Timer/Counter 0 external input)
P3.5	T1 (Timer/Counter 1 external input)
P3.6	$\overline{WR}$ (external Data Memory write strobe)
P3.7	$\overline{RD}$ (external Data Memory read strobe)

The alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1. Otherwise the port pin remains at 0.

I/O Configurations

Figure 24 shows a functional diagram of a typical bit latch and I/O buffer in each of the four ports. The bit latch (one bit in the port's SFR) is represented as a Type D flip-flop, which will clock in a value from the internal bus in response to a "write to latch" signal from the CPU. The level of the port pin itself is placed on the internal bus in response to a "read pin" signal from the CPU. Some instructions that read a port activate the "read latch" signal, and others activate the "read pin" signal.

As shown in Figure 24, the output drivers of Port 0 and 2 are switchable to an internal ADDR and ADDR/DATA bus by an internal CONTROL signal for use in external memory accesses. During external memory accesses, the P2 SFR remains unchanged, but the P0 SFR gets 1s written to it.

Also shown in Figure 24 is that if a P3 bit latch contains a 1, then the output level is controlled by the signal labeled "alternate output function." The actual P3.X pin level is always available to the pin's alternate input function, if any.

Ports 1, 2, and 3 have internal pullups, and Port 0 has open drain outputs. Each I/O line can be independently used as an input or an output. (Port 0 and 2 may not be used as general purpose I/O when being used as the ADDR/DATA BUS for external memory during normal operation.) To be used as an input, the port bit latch must contain a 1, which turns off the output driver FET. Then, for Ports 1, 2, and 3, the pin is pulled high by a weak internal pullup, and can be pulled low by an external source.

Port 0 differs in that its internal pullups are not active during normal port operation. The pullup FET in the P0 output driver (see Figure 24) is used only when the port is emitting 1s during external memory accesses. Otherwise the pullup FET is off. Consequently P0 lines that are being used as output port lines are open drain. Writing a 1 to the bit latch leaves both output FETs off, so the pin floats. In that condition it can be used as a high-impedance input.

Because Ports 1, 2, and 3 have fixed internal pullups, they are sometimes called "quasi-bidirectional" ports. When configured as inputs they pull high and will source current (I_{IL}, in the data sheets) when externally pulled low. Port 0, on the other hand, is considered "true" bidirectional, because when configured as an input it floats.

All the port latches in the 80C51 have 1s written to them by the reset function. If a 0 is subsequently written to a port latch, it can be reconfigured as an input by writing a 1 to it.

Writing to a Port

In the execution of an instruction that changes the value in a port latch, the new value arrives at the latch during S6P2 of the final cycle of the instruction. However, port latches are in fact sampled by their output buffers only during Phase 1 of an clock period. (During Phase 2 the output buffer holds the value it saw during the previous Phase 1). Consequently, the new value in the port latch won't actually appear at the output pin until the next Phase 1, which will be at S1P1 of the next machine cycle.

If the change requires a 0-to-1 transition in Port 1, 2, or 3, an additional pullup is turned on during S1P1 and S1P2 of the cycle in which the transition occurs. This is done to increase the transition speed. The extra pullup can source about 100 times the current that the normal pullup can. It should be noted that the internal pullups are field-effect transistors, not linear resistors. The pullup arrangements are shown in Figure 25.

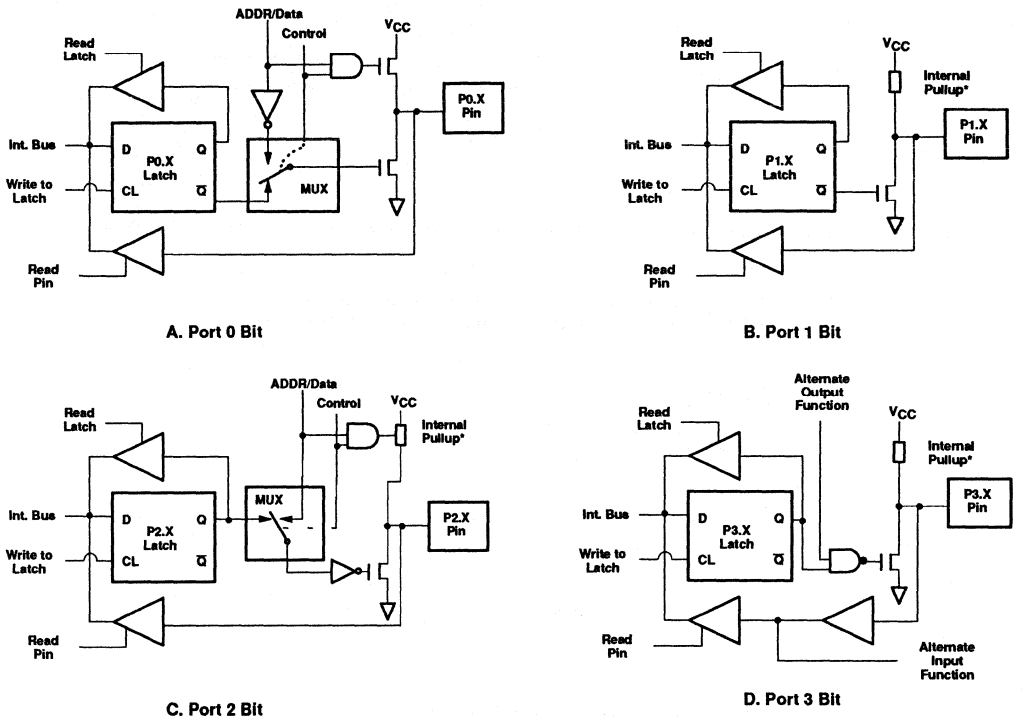


Figure 24. 80C51 Port Bit Latches and I/O Buffers

In the NMOS 8051 part, the fixed part of the pullup is a depletion mode transistor with the gate wired to the source. This transistor will allow the pin to source about 0.25mA when shorted to ground. In parallel with the fixed pullup is an enhancement mode transistor, which is activated during S1 whenever the port bit does a 0-to-1 transition. During this interval, if the port pin is shorted to ground, this extra transistor will allow the pin to source an additional 30mA.

In the CMOS 80C51, the pullup consists of three pFETs. It should be noted that an n-channel FET (nFET) is turned on when a logical 1 is applied to its gate, and is turned off when a logical 0 is applied to its gate. A p-channel FET (pFET) is the opposite: it is on when its gate sees a 0, and off when its gate sees a 1.

pFET1 in Figure 25 is the transistor that is turned on for 2 oscillator periods after a 0-to-1 transition in the port latch. While it's on, it turns on pFET3 (a weak pullup), through the inverter. This inverter and pPET form a latch which holds the 1.

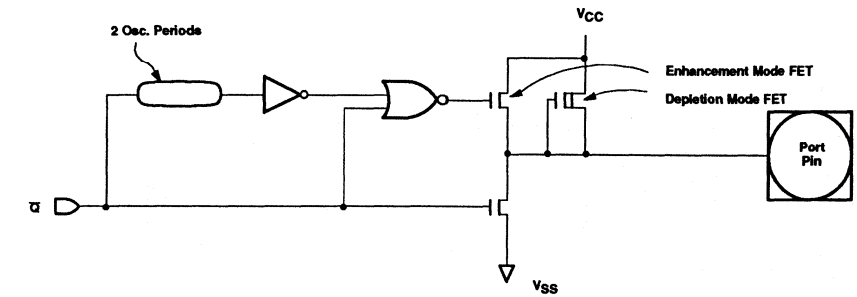
Note that if the pin is emitting a 1, a negative glitch on the pin from some external source can turn off pFET3, causing the pin to go into a float state. pFET2 is a very weak pullup which is on whenever the nFET is off, in traditional CMOS style. It's only about 1/10 the strength of pFET1. Its function is to restore a 1 to the pin in the event the pin had a 1 and lost it to a glitch.

Port Loading and Interfacing

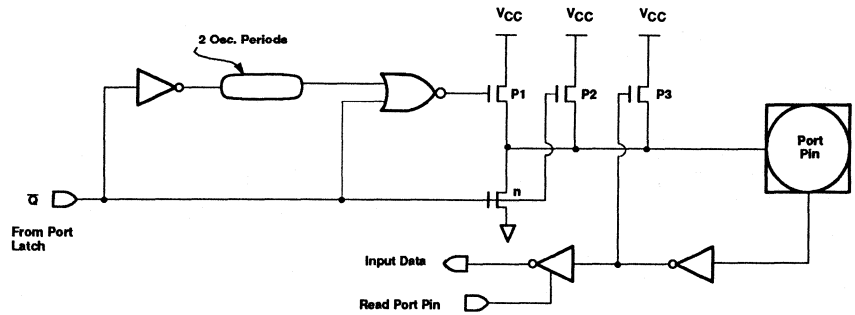
The output buffers of Ports 1, 2, and 3 can each drive 4 LS TTL inputs. These ports on NMOS versions can be driven in a normal manner by an TTL or NMOS circuit. Both NMOS and CMOS pins can be driven by open-collector and open-drain outputs, but note that 0-to-1 transitions will not be fast.

In the NMOS device, if the pin is driven by an open-collector output, a 0-to-1 transition will have to be driven by the relatively weak depletion mode FET in Figure 25A. In the CMOS device, an input 0 turns off pullup pFET3, leaving only the very weak pullup pFET2 to drive the transition.

Port 0 output buffers can each drive 8 LS TTL inputs. They do, however, require external pullups to drive NMOS inputs, except when being used as the ADDRESS/DATA bus for external memory.



A. NMOS Configuration. The enhancement mode transistor is turned on for 2 oscillator periods after Q makes a 1-to-0 transition.



B. CMOS Configuration. pFET1 is turned on for 2 oscillator periods after Q makes a 1-to-0 transition. During this time, pFET1 also turns on pFET3 through the inverter to form a latch which holds the 1. pFET2 is also on.

Figure 25. Ports 1 and 3 NMOS and CMOS Internal Pullup Configurations
Port 2 is similar except that it holds the strong pullup on while emitting 1s that are address bits. (See Accessing External Memory.)

Read-Modify-Write Feature

Some instructions that read a port read the latch and others read the pin. Which ones do which? The instructions that read the latch rather than the pin are the ones that read a value, possibly change it, and then rewrite it to the latch. These are called "read-modify-write" instructions. The instructions listed below are read-modify-write instructions. When the destination operand is a port, or a port bit, these instructions read the latch rather than the pin:

ANL	(logical AND, e.g., ANL P1,A)
ORL	(logical OR, e.g., ORL P2,A)
XRL	(logical EX-OR, e.g., XRL P3,A)
JBC	(jump if bit = 1 and clear bit, e.g., JBC P1.1,LABEL)
CPL	(complement bit, e.g., CPL P3.0)
INC	(increment, e.g., INC P2)
DEC	(decrement, e.g., DEC P2)
DJNZ	(decrement and jump if not zero, e.g., DINZ P3,LABEL)
MOV,PX,Y,C	(move carry bit to bit Y of Port X)
CLR PX.Y	(clear bit Y of Port X)
SET PX Y	(set bit Y of Port X)

It is not obvious that the last three instructions in this list are read-modify-write instructions, but they are. They read the port byte, all 8 bits, modify the addressed bit, then write the new byte back to the latch.

The reason that read-modify-write instructions are directed to the latch rather than the pin is to avoid a possible misinterpretation of the voltage level at the pin. For example, a port bit might be used to drive the base of a transistor. When a 1 is written to the bit, the transistor is turned on. If the CPU then reads the same port bit at the pin rather than the latch, it will read the base voltage of the transistor and interpret it as a 0. Reading the latch rather than the pin will return the correct value of 1.

Accessing External Memory

Accesses to external memory are of two types: accesses to external Program Memory and accesses to external Data Memory. Accesses to external Program Memory use signal $\overline{PSEN}$ (program store enable) as the read strobe. Accesses to external Data Memory use $\overline{RD}$ or $\overline{WR}$ (alternate functions of P3.7 and P3.6) to strobe the memory. Fetches from external Program Memory always use a 16-bit address. Accesses to external Data

Memory can use either a 16-bit address (MOVX @ DPTR) or an 8-bit address (MOVX @Ri).

Whenever a 16-bit address is used, the high byte of the address comes out on Port 2, where it is held for the duration of the read or write cycle. Note that the Port 2 drivers use the strong pullups during the entire time that they are emitting address bits that are 1s. This is during the execution of a MOVX @DPTR instruction. During this time the Port 2 latch (the Special Function Register) does not have to contain 1s, and the contents of the Port 2 SFR are not modified. If the external memory cycle is not immediately followed by another external memory cycle, the undisturbed contents of the Port 2 SFR will reappear in the next cycle.

If an 8-bit address is being used (MOVX @Ri), the contents of the Port 2 SFR remain at the Port 2 pins throughout the external memory cycle. This will facilitate paging.

In any case, the low byte of the address is time-multiplexed with the data byte on Port 0. The ADDR/DATA signals drive both FETs in the Port 0 output buffers. Thus, in this application the Port 0 pins are not open-drain outputs, and do not require external pullups. ALE (Address Latch Enable) should be used to capture the address byte into an external latch. The address byte is valid at the negative transition of ALE. Then, in a write cycle, the data byte to be written appears on Port 0 just before $\overline{WR}$ is activated, and remains there until after $\overline{WR}$ is deactivated. In a read cycle, the incoming byte is accepted at Port 0 just before the read strobe is deactivated.

During any access to external memory, the CPU writes 0FFH to the Port 0 latch (the Special Function Register), thus obliterating whatever information the Port 0 SFR may have been holding.

External Program Memory is accessed under two conditions: Whenever signal $\overline{EA}$ is active; or whenever the program counter (PC) contains a number that is larger than 0FFFH (in the 80C51).

This require that the ROMless versions have $\overline{EA}$ wired low to enable the lower 4k program bytes to be fetched from external memory.

When the CPU is executing out of external Program Memory, all 8 bits of Port 2 are dedicated to an output function and may not be used for general purpose I/O. During external program fetches they output the high byte of the PC. During this time the Port 2 drivers use the strong pullups to emit PC bits that are 1s.

Timer/Counters

The 80C51 has two 16-bit Timer/Counter registers: Timer 0 and Timer 1. Both can be configured to operate either as timers or event counters (see Figure 26).

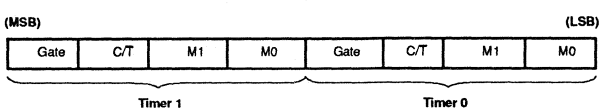
In the "Timer" function, the register is incremented every machine cycle. Thus, one can think of it as counting machine cycles. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

In the "Counter" function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T0 or T1. In this function, the external input is sampled during S5P2 of every machine cycle.

When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since it takes 2 machine cycles (24 oscillator periods) to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full cycle. In addition to the "Timer" or "Counter" selection, Timer 0 and Timer 1 have four operating modes from which to select.

Timer 0 and Timer 1

The "Timer" or "Counter" function is selected by control bits C/T in the Special Function Register TMOD. These two Timer/Counters have four operating modes, which are selected by bit-pairs (M1, M0) in TMOD. Modes 0, 1, and 2 are the same for both Timers/Counters. Mode 3 is different. The four operating modes are described in the following text.



Gate — Gating control when set. Timer/Counter "x" is enabled only while "$\overline{INTx}$" pin is high and "TRx" control pin is set. When cleared Timer "x" is enabled whenever "TRx" control bit is set.

C/T — Timer or Counter Selector cleared for Timer operation (input from internal system clock). Set for Counter operation (input from "Tx" input pin).

M1	M0	Operating
0	0	8048 Timer "TLx" serves as 5-bit prescaler.
0	1	16-bit Timer/Counter "THx" and "TLx" are cascaded; there is no prescaler.
1	0	8-bit auto-reload Timer/Counter "THx" holds a value which is to be reloaded into "TLx" each time it overflows.
1	1	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits.
1	1	(Timer 1) Timer/Counter 1 stopped.

Figure 26. Timer/Counter Mode Control (TMOD) Register

Mode 0
Putting either Timer into Mode 0 makes it look like an 8048 Timer, which is an 8-bit Counter with a divide-by-32 prescaler. Figure 27 shows the Mode 0 operation as it applies to Timer 1.

In this mode, the Timer register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, it sets the Timer interrupt flag TF1. The counted input is enabled to the Timer when TR1 = 1 and either GATE = 0 or $\overline{INT1}$ = 1. (Setting GATE = 1 allows the Timer to be controlled by external input $\overline{INT1}$, to facilitate pulse width measurements). TR1 is a control bit in the Special Function Register TCON (Figure 28). GATE is in TMOD.

The 13-bit register consists of all 8 bits of TH1 and the lower 5 bits of TL1. The upper 3 bits of TL1 are indeterminate and should be ignored. Setting the run flag (TR1) does not clear the registers.

Mode 0 operation is the same for the Timer 0 as for Timer 1. Substitute TR0, TF0, and $\overline{INT0}$ for the corresponding Timer 1 signals in Figure 27. There are two different GATE bits, one for Timer 1 (TMOD.7) and one for Timer 0 (TMOD.3).

Mode 1
Mode 1 is the same as Mode 0, except that the Timer register is being run with all 16 bits.

Mode 2
Mode 2 configures the Timer register as an 8-bit Counter (TL1) with automatic reload, as shown in Figure 29. Overflow from TL1 not only sets TF1, but also reloads TL1 with the contents of TH1, which is preset by software. The reload leaves TH1 unchanged.

Mode 2 operation is the same for Timer/Counter 0.

Mode 3
Timer 1 in Mode 3 simply holds its count. The effect is the same as setting TR1 = 0.

Timer 0 in Mode 3 establishes TL0 and TH0 as two separate counters. The logic for Mode 3 on Timer 0 is shown in Figure 30. TL0 uses the Timer 0 control bits: C/T, GATE, TR0, INT0, and TF0. TH0 is locked into a timer function (counting machine cycles) and takes over the use of TR1 and TF1 from Timer 1. Thus, TH0 now controls the "Timer 1" interrupt.

Mode 3 is provided for applications requiring an extra 8-bit timer on the counter. With Timer 0 in Mode 3, an 80C51 can look like it has three Timer/Counters. When Timer 0 is in Mode 3, Timer 1 can be turned on and off by switching it out of and into its own Mode 3, or can still be used by the serial port as a baud rate generator, or in fact, in any application not requiring an interrupt.

Standard Serial Interface

The serial port is full duplex, meaning it can transmit and receive simultaneously. It is also receive-buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the register. (However, if the first byte still hasn't been read by the time reception of the second byte is complete, one of the bytes will be lost.) The serial port receive and transmit registers are both accessed at Special Function Register SBUF. Writing to SBUF loads the transmit register, and reading SBUF accesses a physically separate receive register.

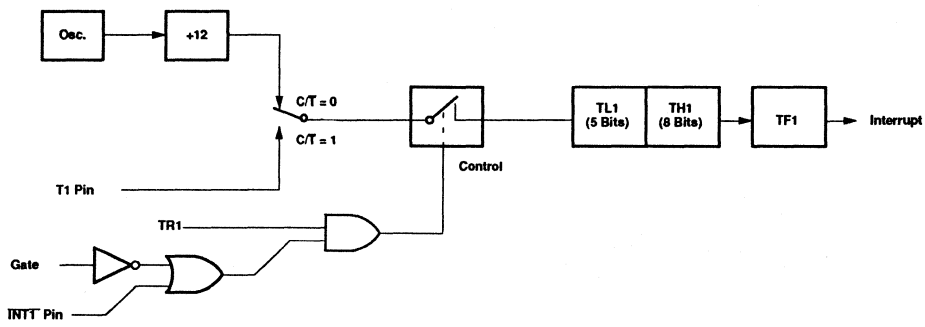


Figure 27. Timer/Counter 1 Mode 0: 13-Bit Counter

(MSB) (LSB)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Symbol	Position	Name and Significance
TF1	TCON.7	Timer 1 overflow flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine.
TR1	TCON.6	Timer 1 Run control bit. Set/cleared by software to turn Timer/Counter on/off.
TF0	TCON.5	Timer 0 overflow flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine.
TR0	TCON.4	Timer 0 Run control bit. Set/cleared by software to turn Timer/Counter on/off.
IE1	TCON.3	Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT1	TCON.2	Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
IE0	TCON.1	Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT0	TCON.0	Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.

Figure 28. Timer/Counter Control (TCON) Register

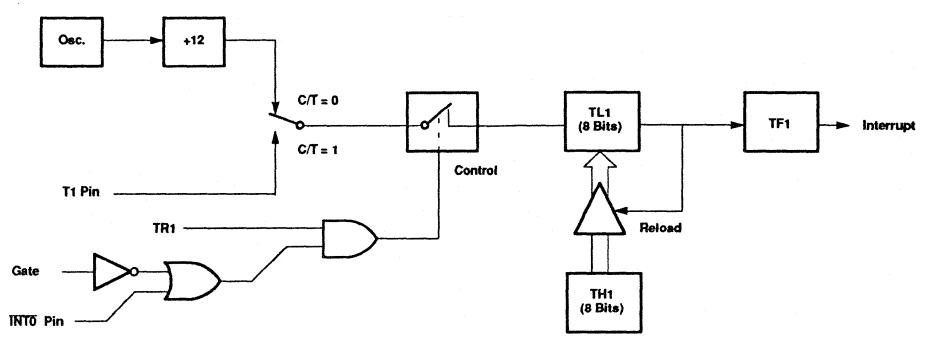


Figure 29. Timer/Counter 1 Mode 2: 8-Bit Auto-Load

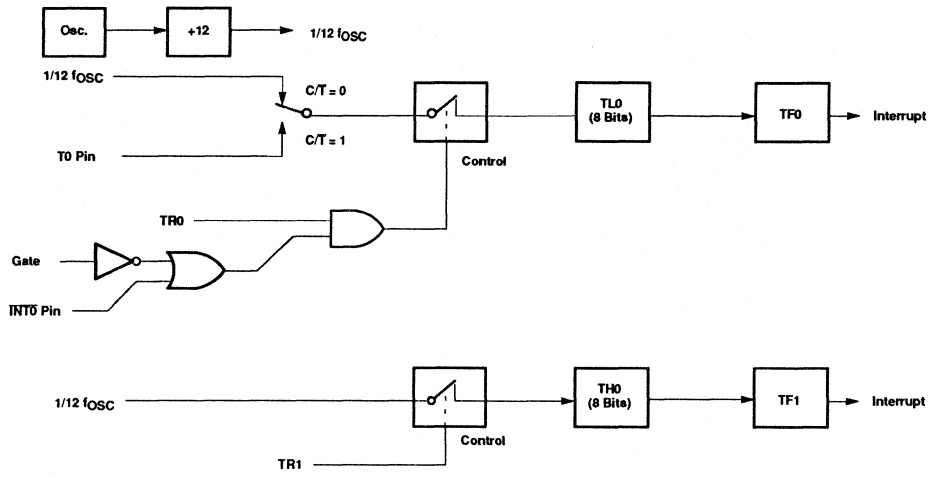


Figure 30. Timer/Counter 0 Mode 3: Two 8-Bit Counters

The serial port can operate in 4 modes:

Mode 0: Serial data enters and exits through RxD. TxD outputs the shift clock. 8 bits are transmitted/received (LSB first). The baud rate is fixed at 1/12 the oscillator frequency.

Mode 1: 10 bits are transmitted (through TxD) or received (through RxD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in Special Function Register SCON. The baud rate is variable.

Mode 2: 11 bits are transmitted (through TxD) or received (through RxD): start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On Transmit, the 9th data bit (TB8 in SCON) can be assigned the value of 0 or 1. Or, for example, the parity bit (P, in the PSW) could be moved into TB8. On receive, the 9th data bit goes into RB8 in Special Function Register SCON, while the stop bit is ignored. The baud rate is programmable to either 1/32 or 1/64 the oscillator frequency.

Mode 3: 11 bits are transmitted (through TxD) or received (through RxD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). In fact, Mode 3 is the same as Mode 2 in all respects except baud rate. The baud rate in Mode 3 is variable.

In all four modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in Mode 0 by the condition RI = 0 and REN = 1. Reception is initiated in the other modes by the incoming start bit if REN = 1.

Multiprocessor Communications

Modes 2 and 3 have a special provision for multiprocessor communications. In these modes, 9 data bits are received. The 9th one goes into RB8. Then comes a stop bit. The port can be programmed such that when the stop bit is received, the serial port interrupt will be activated only if RB8 = 1. This feature is enabled by setting bit SM2 in SCON. A way to use this feature in multiprocessor systems is as follows:

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte which identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM2 = 1, no slave will be interrupted by a data byte. An address byte, however, will interrupt all slaves, so that each slave can examine the received byte and see if it is being addressed. The addressed slave will clear its SM2 bit and prepare to receive the data bytes that will be coming. The slaves that weren't being addressed leave their SM2s set and go on about their business, ignoring the coming data bytes.

SM2 has no effect in Mode 0, and in Mode 1 can be used to check the validity of the stop bit. In a Mode 1 reception, if SM2 = 1, the receive interrupt will not be activated unless a valid stop bit is received.

Serial Port Control Register

The serial port control and status register is the Special Function Register SCON, shown in Figure 31. This register contains not only the mode selection bits, but also the 9th data bit for transmit and receive (TB8 and RB8), and the serial port interrupt bits (TI and RI).

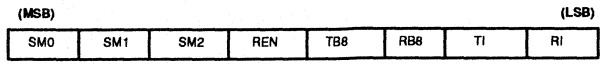
Baud Rates

The baud rate in Mode 0 is fixed: Mode 0 Baud Rate = Oscillator Frequency / 12. The baud rate in Mode 2 depends on the value of bit SMOD in Special Function Register PCON. If SMOD = 0 (which is the value on reset), the baud rate is 1/64 the oscillator frequency. If SMOD = 1, the baud rate is 1/32 the oscillator frequency.

Mode 2 Baud Rate =

$$\frac{2^{SMOD}}{64} \times (\text{Oscillator Frequency})$$

In the 80C51, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate.



Where SM0, SM1 specify the serial port mode, as follows:

SM0	SM1	Mode	Description	Baud Rate
0	0	0	shift register	$f_{OSC}/12$
0	1	1	8-bit UART	variable
1	0	2	9-bit UART	$f_{OSC}/64$ or $f_{OSC}/32$
1	1	3	9-bit UART	variable

- **SM2** enables the multiprocessor communication feature in modes 2 and 3. In mode 2 or 3, if SM2 is set to 1, then RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 = 1, then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0.
- **REN** enables serial reception. Set by software to enable reception. Clear by software to disable reception.
- **TB8** is the 9th data bit that will be transmitted in modes 2 and 3. Set or clear by software as desired.
- **RB8** in modes 2 and 3, is the 9th data bit that was received. In mode 1, if SM2 = 0, RB8 is the stopbit that was received in mode 0, RB8 is not used.
- **TI** is the transmit interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must be cleared by software.
- **RI** is the receive interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM2). Must be cleared by software.

Figure 31. Serial Port Control (SCON) Register

Using Timer 1 to Generate Buad Rates

When Timer 1 is used as the baud rate generator, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate and the value of SMOD as follows:

Mode 1, 3 Baud Rate =

$$\frac{2^{SMOD}}{32} \times (\text{Timer 1 Overflow Rate})$$

The Timer 1 interrupt should be disabled in this application. The Timer itself can be configured for either "timer" or "counter" operation, and in any of its 3 running modes. In the most typical applications, it is configured for "timer" operation, in the auto-reload mode (high nibble of TMOD = 0010B). In that case the baud rate is given by the formula:

Mode 1, 3 Baud Rate =

$$\frac{2^{SMOD}}{32} \times \frac{\text{Oscillator Frequency}}{12 \times [256 - (TH1)]}$$

One can achieve very low baud rates with Timer 1 by leaving the Timer 1 interrupt enabled, and configuring the Timer to run as a 16-bit timer (high nibble of TMOD = 0001B), and using the Timer 1 interrupt to do a 16-bit software reload. Figure 32 lists various commonly used baud rates and how they can be obtained from Timer 1.

Baud Rate	f_{OSC}	SMOD	Timer 1		
			C/T	Mode	Reload Value
Mode 0 Max: 1.67MHz	20MHz	X	X	X	X
Mode 2 Max: 625k	20MHz	1	X	X	X
Mode 1, 3 Max: 104.2k	20MHz	1	0	2	FFH
19.2k	11.059MHz	1	0	2	FDH
9.6k	11.059MHz	0	0	2	FDH
4.8k	11.059MHz	0	0	2	FAH
2.4k	11.059MHz	0	0	2	F4H
1.2k	11.059MHz	0	0	2	E8H
137.5	11.986M	0	0	2	1DH
110	6MHz	0	0	2	72H
110	12MHz	0	0	1	FEEEBH

Figure 32. Timer 1 Generated Commonly Used Baud Rates

More About Mode 0

Serial data enters and exits through RxD. TxD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud rate is fixed a 1/12 the oscillator frequency.

Figure 33 shows a simplified functional diagram of the serial port in Mode 0, and associated timing.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal at S6P2 also loads a 1 into the 9th position of the transmit shift register and tells the TX Control block to commence a transmission. The internal timing is such that one full machine cycle will elapse between "write to SBUF" and activation of SEND.

SEND enables the output of the shift register to the alternate output function line of P3.0 and also enable SHIFT CLOCK to the alternate output function line of P3.1. SHIFT CLOCK is low during S3, S4, and S5 of every machine cycle, and high during S6, S1, and S2. At S6P2 of every machine cycle in which SEND is active, the contents of the transmit shift are shifted to the right one position.

As data bits shift out to the right, zeros come in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position, is just to the left of the MSB, and all positions to the left of that contain zeros. This condition flags the TX Control block to do one last shift and then deactivate SEND and set T1. Both of these actions occur at S1P1 of the 10th machine cycle after "write to SBUF."

Reception is initiated by the condition REN = 1 and R1 = 0. At S6P2 of the next machine cycle, the RX Control unit writes the bits 11111110 to the receive shift register, and in the next clock phase activates RECEIVE.

RECEIVE enable SHIFT CLOCK to the alternate output function line of P3.1. SHIFT CLOCK makes transitions at S3P1 and S6P1 of every machine cycle. At S6P2 of every machine cycle in which RECEIVE is active, the contents of the receive shift register are shifted to the left one position. The value that comes in from the right is the value that was sampled at the P3.0 pin at S5P2 of the same machine cycle.

As data bits come in from the right, 1s shift out to the left. When the 0 that was initially loaded into the rightmost position arrives at the leftmost position in the shift register, it flags the RX Control block to do one last shift and load SBUF. At S1P1 of the 10th machine cycle after the write to SCON that cleared RI, RECEIVE is cleared as RI is set.

More About Mode 1

Ten bits are transmitted (through TxD), or received (through RxD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in SCON. In the 80C51 the baud rate is determined by the Timer 1 overflow rate.

Figure 34 shows a simplified functional diagram of the serial port in Mode 1, and associated timings for transmit receive.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal also loads a 1 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission actually commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the "write to SBUF" signal.)

The transmission begins with activation of SEND which puts the start bit at TxD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TxD. The first shift pulse occurs one bit time after that.

As data bits shift out to the right, zeros are clocked in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position is just to the left of the MSB, and all positions to the left of that contain zeros. This condition flags the TX Control unit to do one last shift and then deactivfate SEND and set TI. This occurs at the 10th divide-by-16 rollover after "write to SBUF."

Reception is initiated by a detected 1-to-0 transition at RxD. For this purpose RxD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written into the input shift register. Resetting the divide-by-16 counter aligns its rollovers with the boundaries of the incoming bit times.

The 16 states of the counter divide each bit time into 16ths. At the 7th, 8th, and 9th counter states of each bit time, the bit detector samples the value of RxD. The value accepted is the value that was seen in at least 2 of the 3 samples. This is done for noise rejection. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. This is to provide rejection of false start bits. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register (which in mode 1 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI. The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated.:

1. R1 = 0, and
2. Either SM2 = 0, or the received stop bit = 1.

If either of these two conditions is not met, the received frame is irretrievably lost. If both conditions are met, the stop bit goes into RB8, the 8 data bits go into SBUF, and RI is activated. At this time, whether the above conditions are met or not, the unit goes back to looking for a 1-to-0 transition in RxD.

More About Modes 2 and 3

Eleven bits are transmitted (through TxD), or received (through RxD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On transmit, the 9th data bit (TB8) can be assigned the value of 0 or 1. On receive, the 9the data bit goes into RB8 in SCON. The baud rate is programmable to either 1/32 or 1/64 the oscillator frequency in Mode 2. Mode 3 may have a variable baud rate generated from Timer 1.

Figures 35 and 36 show a functional diagram of the serial port in Modes 2 and 3. The receive portion is exactly the same as in Mode 1. The transmit portion differs from Mode 1 only in the 9th bit of the transmit shift register.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal also loads TB8 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the "write to SBUF" signal.)

The transmission begins with activation of SEND, which puts the start bit at TxD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TxD. The first shift pulse occurs one bit time after that. The first shift clocks a 1 (the stop bit) into the 9th bit position of the shift register. Thereafter, only zeros are clocked in. Thus, as data bits shift out to the right, zeros are clocked in from the left. When TB8 is at the output position of the shift register, then the stop bit is just to the left of TB8, and all positions to the left of that contain zeros. This condition flags the TX Control unit to do one last shift and then deactivate SEND and set TI. This occurs at the 11th divide-by-16 rollover after "write to SUBF."

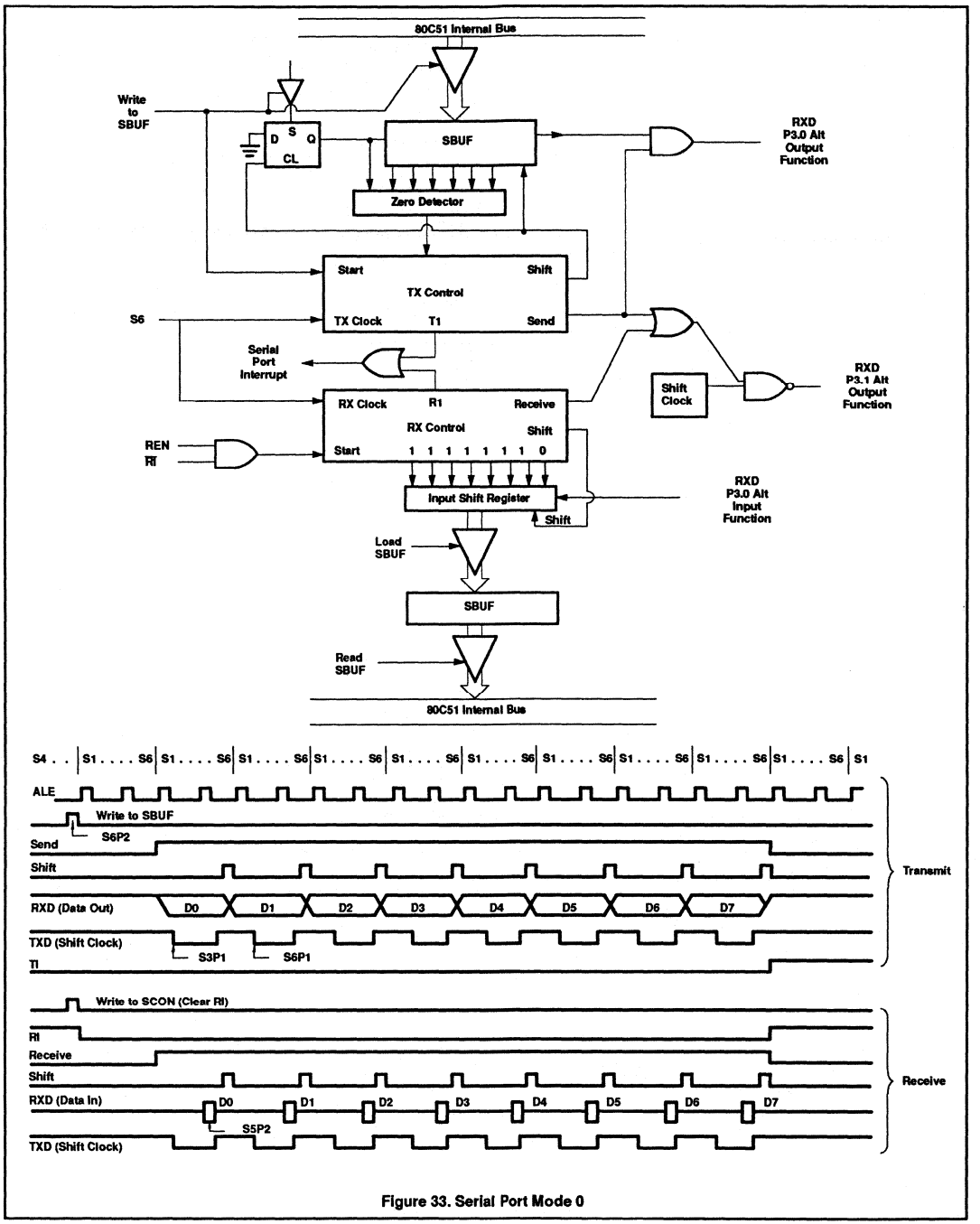


Figure 33. Serial Port Mode 0

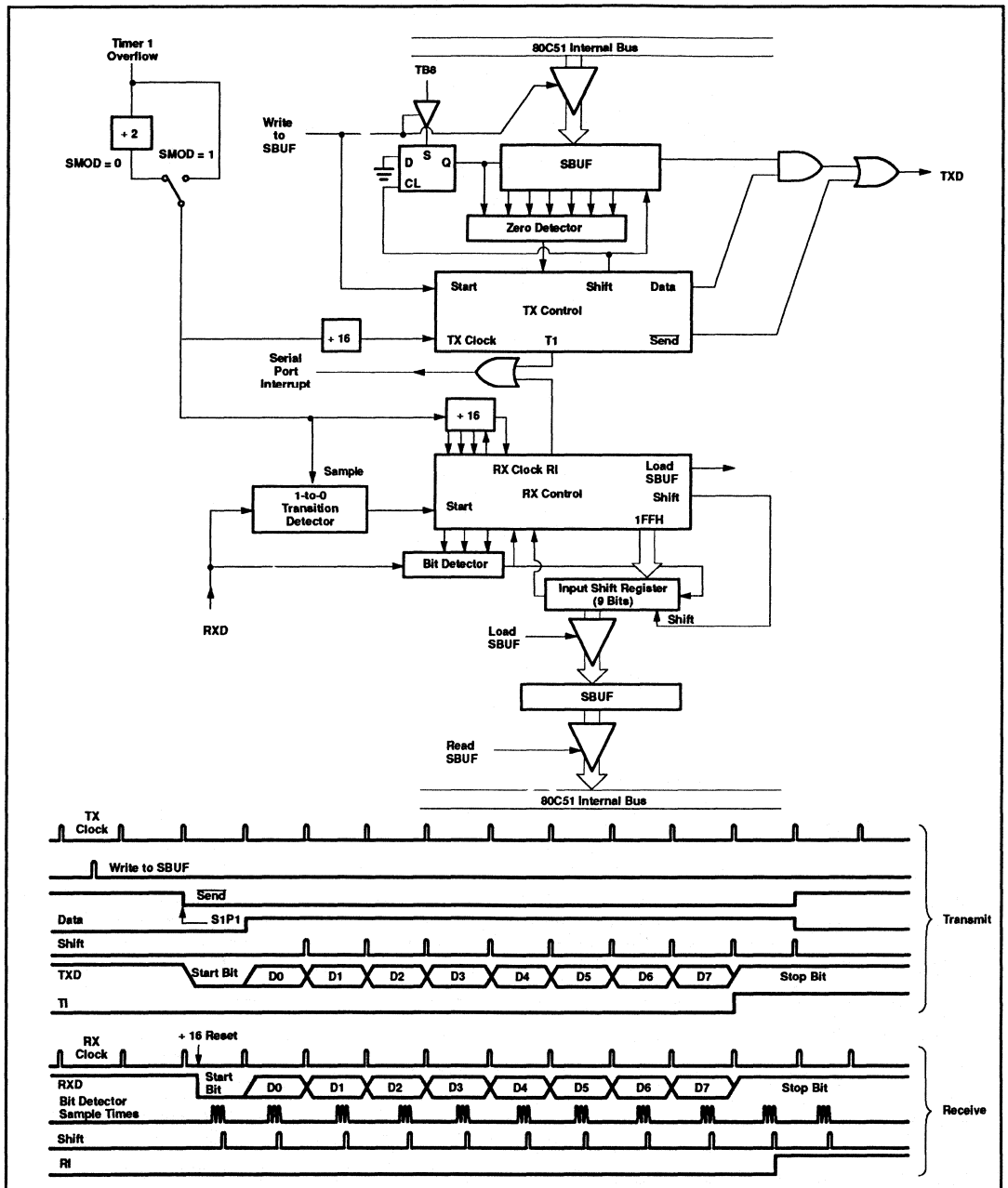


Figure 34. Serial Port Mode 1

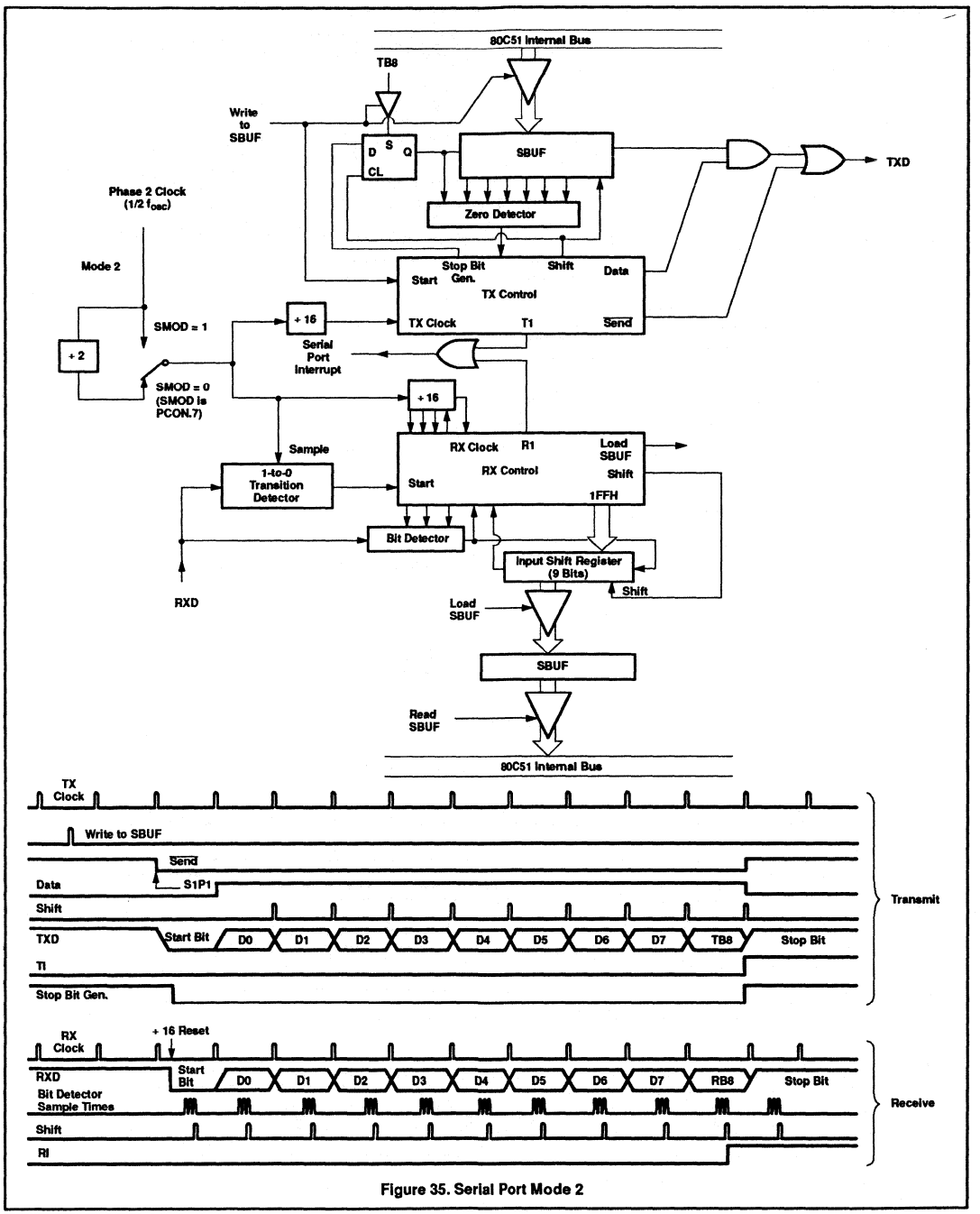


Figure 35. Serial Port Mode 2

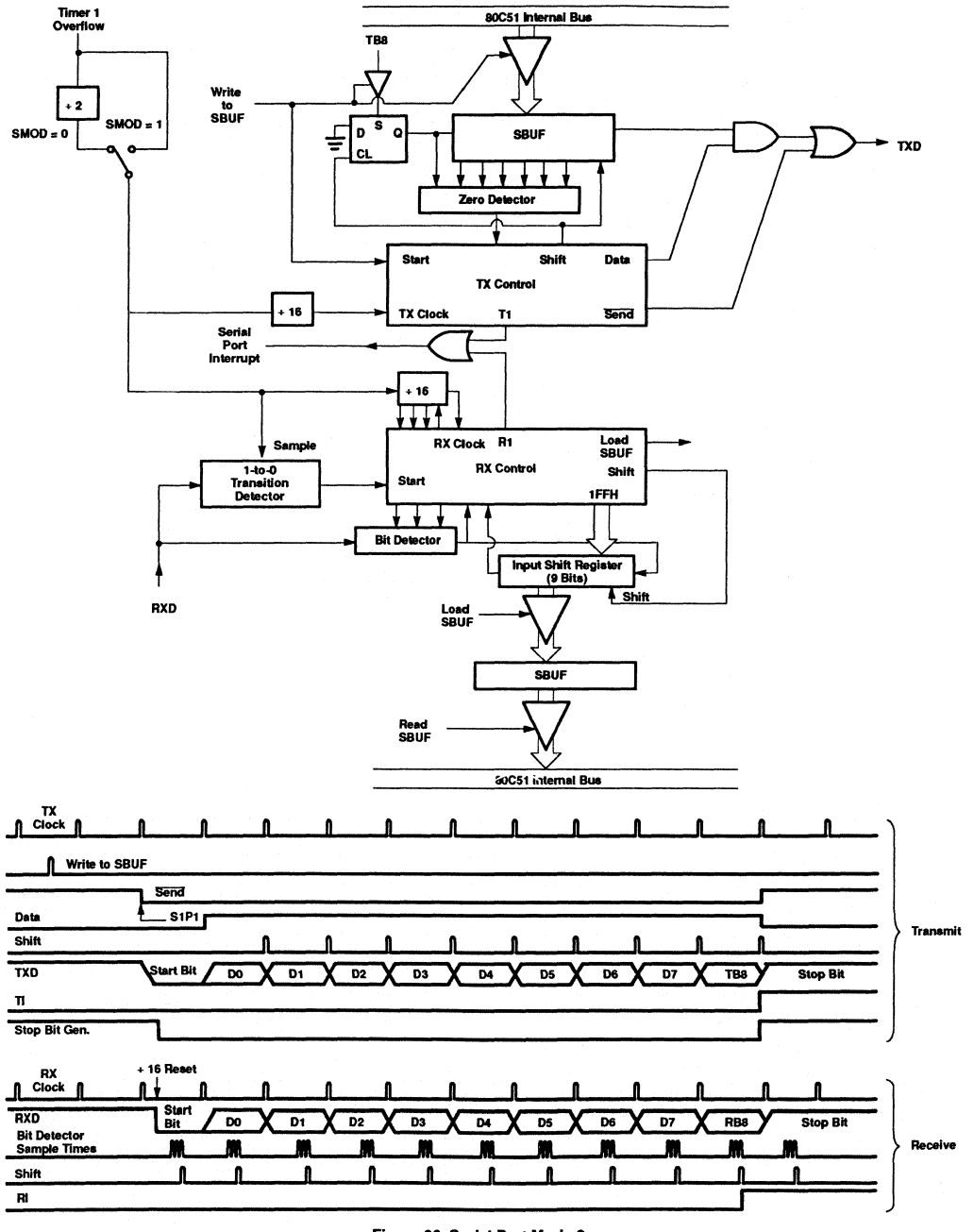


Figure 36. Serial Port Mode 3

Reception is initiated by a detected 1-to-0 transition at RxD. For this purpose RxD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written to the input shift register.

At the 7th, 8th, and 9th counter states of each bit time, the bit detector samples the value of R-D. The value accepted is the value that was seen in at least 2 of the 3 samples. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register (which in Modes 2 and 3 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI.

The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated.

1. RI = 0, and
2. Either SM2 = 0, or the received 9th data bit = 1

If either of these conditions is not met, the received frame is irretrievably lost, and RI is not set. If both conditions are met, the received 9th data bit goes into RB8, and the first 8 data bits go into SBUF. One bit time later, whether the above conditions were met or not, the unit goes back to looking for a 1-to-0 transition at the RxD input.

Interrupts

The 80C51 provides 5 interrupt sources. These are shown in Figure 37. The External Interrupts $\overline{INT0}$ and $\overline{INT1}$ can each be either level-activated or transition-activated, depending on bits IT0 and IT1 in Register TCON. The flags that actually generate these interrupts are bits IE0 and IE1 in TCON. When an external interrupt is generated, the flag that generated it is cleared by the hardware when the service routine is vectored to only if the interrupt was transition-activated. If the interrupt was level-activated, then the external requesting source is what controls the request flag, rather than the on-chip hardware.

The Timer 0 and Timer 1 Interrupts are generated by TF0 and TF1, which are set by a rollover in their respective Timer/Counter registers (except see Timer 0 in Mode 3). When a timer interrupt is generated, the flag that generated it is cleared by the on-chip hardware when the service routine is vectored to.

The Serial Port Interrupt is generated by the logical OR of RI and TI. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine will normally have to determine whether it was RI or TI that generated the interrupt, and the bit will have to be cleared in software.

All of the bits that generate interrupts can be set or cleared by software, with the same result as though it had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be canceled in software.

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE (Figure 38). IE also contains a global disable bit, EA, which disables all interrupts at once.

Priority Level Structure

Each interrupt source can also be individually programmed to one of two priority levels by setting or clearing a bit in Special Function Register IP (Figure 39). A low-priority interrupt can itself be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

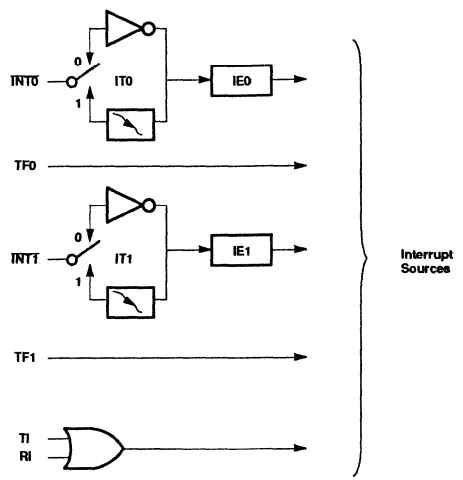
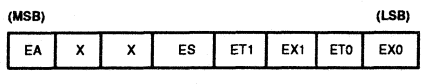
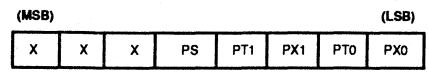


Figure 37. 80C51 Interrupt Sources



Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1 each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
–	IE.6	Reserved.
–	IE.5	Reserved.
ES	IE.4	Enables or disables the Serial Port interrupt. If ES = 0, the Serial Port interrupt is disabled.
ET1	IE.3	Enables or disables the Timer 1 Overflow interrupt. If ET1 = 0, the Timer 1 interrupt is disabled.
EX1	IE.2	Enables or disables External Interrupt 1. If EX1 = 0, External Interrupt 1 is disabled.
ET0	IE.1	Enables or disables the Timer 0 Overflow interrupt. If ET0 = 0, the Timer 0 interrupt is disabled.
EX0	IE.0	Enables or disables External Interrupt 0. If EX0 = 0, External Interrupt 0 is disabled.

Figure 38. Interrupt Enable Register (IE)



Symbol	Position	Function
–	IP.7	Reserved.
–	IP.6	Reserved.
–	IP.5	Reserved.
PS	IP.4	Defines the Serial Port interrupt priority level. PS = 1 programs it to the higher priority level.
PT1	IP.3	Defines theTimer 1 interrupt priority level. PT1 = 1 programs it to the higher priority level.
PX1	IP.2	Defines the External Interrupt 1 priority level. PX1 =1 programs it to the higher priority level.
PT0	IP.1	Enables or disables the Timer 0 Interrupt priority level. PT0 = 1 programs it to the higher priority level.
PX0	IP.0	Defines the External Interrupt 0 priority level. PX0 = 1 programs it to the higher priority level.

Figure 39. Interrupt Priority Register (IP)

If two request of different priority levels are received simultaneously, the request of higher priority level is serviced. If requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence as follows:

Source	Priority Within Level
1. IE0	(highest)
2. TF0	
3. IE1	
4. TF1	
5. RI+TI	(lowest)

Note that the "priority within level" structure is only used to resolve simultaneous requests of the same priority level.

The IP register contains a number of unimplemented bits. IP.7, IP.6, and IP.5 are reserved in the 80C51. User software should not write 1s to these positions, since they may be used in other 8051 Family products.

How Interrupts Are Handled

The interrupt flags are sampled at S5P2 of every machine cycle. The samples are polled during the following machine cycle. If one of the flags was in a set condition at S5P2 of the preceding cycle, the polling cycle will find it and the interrupt system will generate an LCALL to the appropriate service routine, provided this hardware-generated LCALL is not blocked by any of the following conditions:

1. An interrupt of equal or higher priority level is already in progress.
2. The current (polling) cycle is not the final cycle in the execution of the instruction in progress.
3. The instruction in progress is RETI or any write to the IE or IP registers.

Any of these three conditions will block the generation of the LCALL to the interrupt service routine. Condition 2 ensures that the instruction in progress will be completed before vectoring to any service routine. Condition 3 ensures that if the instruction in progress is RETI or any access to IE or IP, then at least one more instruction will be executed before any interrupt is vectored to.

The polling cycle is repeated with each machine cycle, and the values polled are the values that were present at S5P2 of the previous machine cycle. Note that if an interrupt flag is active but not being responded to for one of the above conditions, if the flag is not still active when the blocking condition is removed, the denied interrupt will not be serviced. In other words, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle is new.

The polling cycle/LCALL sequence is illustrated in Figure 40.

Note that if an interrupt of higher priority level goes active prior to S5P2 of the machine cycle labeled C3 in Figure 40, then in accordance with the above rules it will be vectored to during C5 and C6, without any instruction of the lower priority routine having been executed.

Thus the processor acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate servicing routine. In some cases it also clears the flag that generated the interrupt, and in other cases it doesn't. It never clears the Serial Port flag. This has to be done in the user's software. It clears an external interrupt flag (IE0 or IE1) only if it was transition-activated. The hardware-generated LCALL pushes the contents of the Program Counter on to the stack (but it does not save the PSW) and reloads the PC with an address that depends on the source of the interrupt being vectored to, as shown below:

Source	Vector Address
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI+TI	0023H

Execution proceeds from that location until the RETI instruction is encountered. The RETI instruction informs the processor that this interrupt routine is no longer in progress, then pops the top two bytes from the stack and reloads the Program Counter. Execution of the interrupted program continues from where it left off.

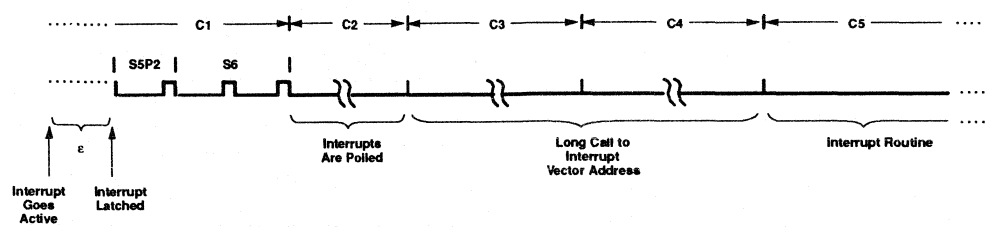


Figure 40. Interrupt Response Timing Diagram

Note that a simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system thinking an interrupt was still in progress, making future interrupts impossible.

External Interrupts

The external sources can be programmed to be level-activated or transition-activated by setting or clearing bit IT1 or IT0 in Register TCON. If ITx = 0, external interrupt x is triggered by a detected low at the INTx pin. If ITx = 1, external interrupt x is edge triggered. In this mode if successive samples of the INTx pin show a high in one cycle and a low in the next cycle, interrupt request flag IEx in TCON is set. Flag bit IEx then requests the interrupt.

Since the external interrupt pins are sampled once each machine cycle, an input high or low should hold for at least 12 oscillator periods to ensure sampling. If the external interrupt is transition-activated, the external source has to hold the request pin high for at least one cycle, and then hold it low for at least one cycle. This is done to ensure that the transition is seen so that interrupt request flag IEx will be set. IEx will be automatically cleared by the CPU when the service routine is called.

If the external interrupt is level-activated, the external source has to hold the request active until the requested interrupt is actually generated. Then it has to deactivate the request before the interrupt service routine is completed, or else another interrupt will be generated.

Response Time

The $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ levels are inverted and latched into IE0 and IE1 at S5P2 of every machine cycle. The values are not actually polled by the circuitry until the next machine cycle. If a request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. The call itself takes two cycles. Thus, a minimum of three complete machine cycles elapse between activation of an external interrupt request and the beginning of execution of the first instruction of the service routine. Figure 40 shows interrupt response timings.

A longer response time would result if the request is blocked by one of the 3 previously listed conditions. If an interrupt of equal or higher priority level is already in progress, the additional wait time obviously depends on the nature of the other interrupt's service routine. If the instruction in progress is not in its final cycle, the additional wait time cannot be more the 3 cycles, since the longest instructions (MUL and DIV) are only 4 cycles long, and if the instruction in progress is RETI or an access to IE or IP, the additional wait time cannot be more than 5 cycles (a maximum of one more cycle to complete the instruction in progress, plus 4 cycles to complete the next instruction if the instruction is MUL or DIV).

Thus, in a single-interrupt system, the response time is always more than 3 cycles and less than 9 cycles.

Single-Step Operation

The 80C51 interrupt structure allows single-step execution with very little software overhead. As previously noted, an interrupt request will not be responded to while an interrupt of equal priority level is still in progress, nor will it be responded to after RETI until at least one other instruction has been executed. Thus, once an interrupt routine has been entered, it cannot be re-entered until at least one instruction of the interrupted program is executed. One way to use this feature for single-step operation is to program one of the external interrupts (e.g., INT0) to be level-activated. The service routine for the interrupt will terminate with the following code:

```
JNB   P3.2,$   ;Wait Till INT0 Goes High
JB    P3.2,$   ;Wait Till INT0 Goes Low
RETI           ;Go Back and Execute One
                Instruction
```

Now if the INT0 pin, which is also the P3.2 pin, is held normally low, the CPU will go right into the External Interrupt 0 routine and stay there until INT0 is pulsed (from low to high to low). Then it will execute RETI, go back to the task program, execute one instruction, and immediately re-enter the External Interrupt 0 routine to await the next pulsing of P3.2. One step of the task program is executed each time P3.2 is pulsed.

Reset

The reset input is the RST pin, which is the input to a Schmitt Trigger. A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. The CPU responds by generating an internal reset, with the timing shown in Figure 41.

The external reset signal is asynchronous to the internal clock. The RST pin is sampled during State 5 Phase 2 of every machine cycle. The port pins will maintain their current activities for 19 oscillator periods after a logic 1 has been sampled at the RST pin; that is, for 19 to 31 oscillator periods after the external reset signal has been applied to the RST pin.

The internal reset algorithm writes 0s to all the SFRs except the port latches, the Stack Pointer, and SBUF. The port latches are initialized to FFH, the Stack Pointer to 07H, and SBUF is indeterminate. Table 11 lists the SFR reset values. The internal RAM is not affected by reset. On power up the RAM content is indeterminate.

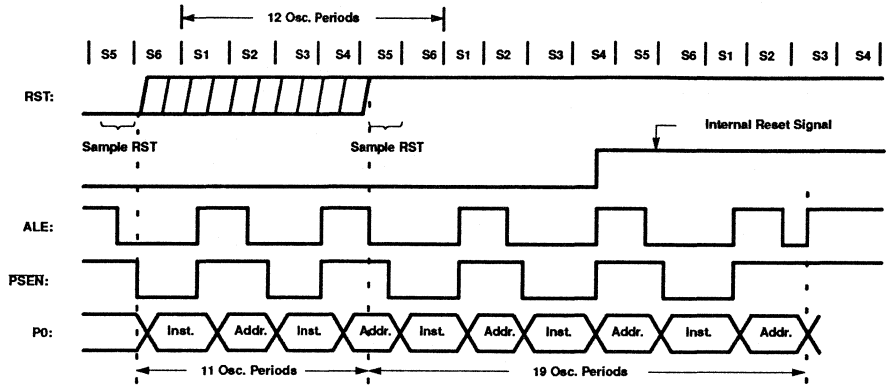


Figure 41. Reset Timing

Table 11. 80C51 SFR Reset Values

REGISTER	RESET VALUE
PC	000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0-P3	FFH
IP	XXX00000B
IE	0XX00000B
TMOD	00H
TCON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
SCON	00H
SBUF	Indeterminate
PCON (HMOS)	0XXXXXXXB
PCON (CHMOS)	0XXX0000B

Power-on Reset

An automatic reset can be obtained when V_{CC} is turned on by connecting the RST pin to V_{CC} through a 10µf capacitor and to V_{SS} through an 8.2k resistor, providing the V_{CC} rise time does not exceed 1 millisecond and the oscillator start-up time does not exceed 10 milliseconds. This power-on reset circuit is shown in Figure 42. The CMOS devices do not require the 8.2k pulldown resistor, although its presence does no harm.

When power is turned on, the circuit holds the RST pin high for an amount of time that depends on the value of the capacitor and the rate at which it charges. To ensure a good reset, the RST pin must be high long enough to allow the oscillator time to start-up (normally a few ms) plus two machine cycles.

Note that the port pins will be in a random state until the oscillator has started and the internal reset algorithm has written 1s to them.

With this circuit, reducing V_{CC} quickly to 0 causes the RST pin voltage to momentarily fall below 0V. However, this voltage is internally limited, and will not harm the device.

Power-Saving Modes of Operation

For applications where power consumption is critical the CMOS version provides power reduced modes of operation as a standard feature. The power down mode in NMOS devices is no longer a standard feature.

CMOS Power Reduction Mode

CMOS versions have two power reducing modes, Idle and Power Down. The input through which backup power is supplied during these operations is V_{CC}. Figure 43 shows the internal circuitry which implements these features. In the Idle modes (IDL = 1), the oscillator continues to run and the Interrupt, Serial Port, and Timer blocks continue to be clocked, but the clock signal is gated off to the CPU. In Power Down (PD = 1), the oscillator is frozen. The Idle and Power Down Modes are activated by setting bits in Special Function Register PCON. The address of this register is 87H. Figure 44 details its contents.

In the NMOS devices the PCON register only contains SMOD. The other four bits are implemented only in the CMOS devices. User software should never write 1s to unimplemented bits, since they may be used in other 80C51 Family products.

Idle Mode

An instruction that sets PCON.0 causes that to be the last instruction executed before going into the Idle mode, the internal clock signal is gated off to the CPU but not to the Interrupt, Timer, and Serial Port functions. The CPU status is preserved in its entirety; the Stack Pointer, Program Counter, Program Status Word, Accumulator, and all other registers maintain their data during Idle. The port pins hold the logical states they had at the time Idle was activated. ALE and $\overline{PSEN}$ hold at logic high levels.

There are two ways to terminate the Idle. Activation of any enabled interrupt will cause PCON.0 to be cleared by hardware, terminating the Idle mode. The interrupt will be serviced, and following RETI, the next instruction to be executed will be the one following the instruction that put the device into Idle.

The flag bits GF0 and GF1 can be used to give an indication if an interrupt occurred during normal operation or during an Idle. For example, an instruction that activates Idle can also set one or both flag bits. When Idle is terminated by an interrupt, the interrupt service routine can examine the flag bits. The other way of terminating the Idle mode is with a hardware reset. Since the clock oscillator is still running, the hardware reset needs to be held active for only two machine cycles (24 oscillator periods) to complete the reset.

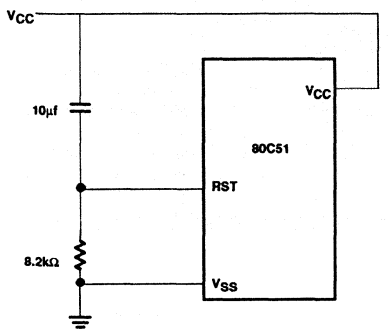


Figure 42. Power On Reset Circuit

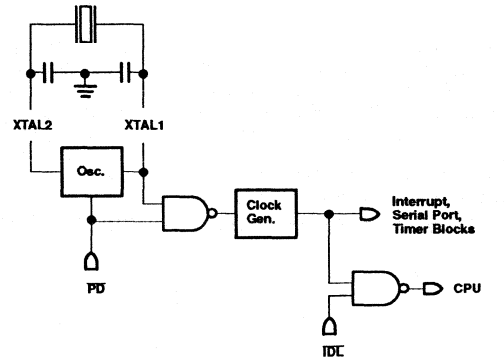


Figure 43. Idle and Power Down Hardware

(MSB)							(LSB)
SMOD	–	–	–	GF1	GF0	PD	IDL

Symbol	Position	Name and Function
SMOD	PCON.7	Double Baud rate bit. When set to a 1 and Timer 1 is used to generate baud rate, and the Serial Port is used in modes 1, 2, or 3.
–	PCON.6	Reserved.
–	PCON.5	Reserved.
–	PCON.4	Reserved.
GF1	PCON.3	General-purpose flag bit.
GF0	PCON.2	General-purpose flag bit.
PD	PCON.1	Power-Down bit. Setting this bit activates power-down operation.
IDL	PCON.0	Idle mode bit. Setting this bit activates idle mode operation.

If 1s are written to PD and IDL at the same time, PD takes precedence. The reset value of PCON is (0XXX0000). In the NMOS devices, the PCON register only contains SMOD. The other four bits are implemented only in the CMOS devices. User software should never write 1s to unimplemented bits, since they may be used in future products.

Figure 44. Power Control (PCON) Register

The signal at the RST pin clears the IDL bit directly and asynchronously. At this time the CPU resumes program execution from where it left off; that is, at the instruction following the one that invoked the Idle Mode. As shown in Figure 41, two or three machine cycles of program execution may take place before the internal reset algorithm takes control. On-chip hardware inhibits access to the internal RAM during this time, but access to the port pins is not inhibited. To eliminate the possibility of unexpected outputs at the port pins, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external Data RAM.

Power-Down Mode

An instruction that sets PCON.1 causes that to be the last instruction executed before going into the Power Down mode. In the Power Down mode, the on-chip oscillator is stopped. With the clock frozen, all functions are stopped, the contents of the on-chip RAM and Special Function Registers are maintained. The port pins output the values held by their respective SFRs. The ALE and $\overline{PSEN}$ output are held low.

The only exit from Power Down is a hardware reset. Reset redefines all the SFRs, but does not change the on-chip RAM.

In the Power Down mode of operation, V_{CC} can be reduced to as low as 2V. Care must be taken, however, to ensure that V_{CC} is not reduced before the Power Down mode is invoked, and that V_{CC} is restored to its normal operating level, before the Power Down mode is terminated. The reset that terminates Power Down also frees the oscillator. The reset should not be activated before V_{CC} is restored to its normal operating level, and must be held active long enough to allow the oscillator to restart and stabilize (normally less than 10ms).

The On-Chip Oscillators

NMOS Version

The on-chip oscillator circuitry for the NMOS-members of the 80C51 family is a single stage linear inverter (Figure 45), intended for use as a crystal-controlled, positive reactance oscillator (Figure 46). In this application the crystal is operated in its fundamental response mode as an inductive reactance in parallel resonance with capacitance external to the crystal.

The crystal specifications and capacitance values (C1 and C2 in Figure 46) are not critical. 30pF can be used in these positions at any frequency with good quality crystals. A ceramic resonator can be used in place of the crystal in cost-sensitive applications. When a ceramic resonator is used, C1 and C2 are normally selected to be of somewhat higher values, typically, 47pF. The manufacturer of the ceramic resonator should be consulted for recommendation on the values of these capacitors.

To drive the NMOS parts with an external clock source, apply the external clock signal to XTAL2, and ground XTAL1, as shown in Figure 47. A pullup resistor may be used (to increase noise margin), but is optional if V_{OH} of the driving gate exceeds the V_{IH} minimum specification of XTAL.

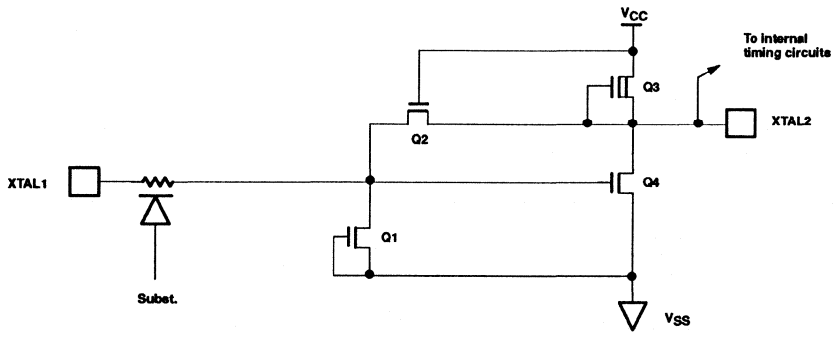


Figure 45. On-Chip Oscillator in the NMOS Version of the 8051 Family

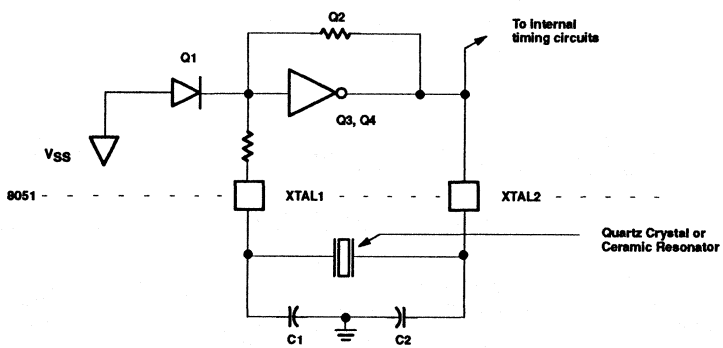


Figure 46. Using the NMOS On-Chip Oscillator

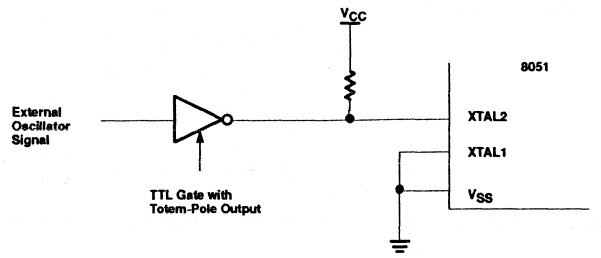


Figure 47. Driving the NMOS 8051 Family Parts with an External Clock

CMOS Versions

The on-chip oscillator circuitry for the 80C51, shown in Figure 48, consists of a single stage linear inverter intended for use as a crystal-controlled, positive reactance oscillator in the same manner as the NMOS parts. However, there are some important differences.

One difference is that the 80C51 is able to turn off its oscillator under software control (by writing a 1 to the PD bit in PCON). Another difference is that, in the 80C51, the internal clocking circuitry is driven by the signal at XTAL1, whereas in the NMOS versions it is by the signal at XTAL2.

The feedback resistor R_f in Figure 48 consists of paralleled n- and p-channel FETs controlled by the PD bit, such that R_f is opened when PD = 1. The diodes D1 and D2, which act as clamps to V_{CC} and V_{SS}, are parasitic to the R_f FETs. The oscillator can be used with the same external components as the NMOS versions, as shown in Figure 49. Typically, C1 = C2 = 30pF when the feedback element is a quartz crystal, and C1 = C2 = 47pF when a ceramic resonator is used.

To drive the CMOS parts with an external clock source, apply the external clock signal to XTAL1, and leave XTAL2 float, as shown in Figure 50.

The reason for this change from the way the NMOS part is driven can be seen by comparing Figures 46 and 48. In the NMOS devices the internal timing circuits are driven by the signal at XTAL2. In the CMOS devices the internal timing circuits are driven by the signal at XTAL1.

Internal Timing

Figures 51 through 54 show when the various strobe and port signals are clocked internally. The figures do not show rise and fall times of the signals, nor do they show propagation delays between the XTAL2 signal and events at other pins.

Rise and fall times are dependent on the external loading that each pin must drive. They are often taken to be something in the neighborhood of 10ns, measured between 0.8V and 2.0V.

Propagation delays are different for different pins. For a given pin they vary with pin loading, temperature, V_{CC}, and manufacturing lot. If the XTAL2 waveform is taken as the timing reference, prop delays may vary up to ±200%.

The AC Timings section of the data sheets do not reference any timing to the XTAL2 waveform. Rather, they relate the critical edges of control and input signals to each other. The timings published in the data sheets include the effects of propagation delays under the specified test conditions.

80C51 Pin Descriptions

ALE/PROG: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. ALE is emitted at a constant rate of 1/6 of the oscillator frequency, for external timing or clocking purposes, even when there are no accesses to external memory. (However, one ALE pulse is skipped during each access to external Data Memory.) This pin is also the program pulse input (PROG) during EPROM programming.

$\overline{PSEN}$: Program Store Enable is the read strobe to external Program Memory. When the device is executing out of external Program Memory, $\overline{PSEN}$ is activated twice each machine cycle (except that two $\overline{PSEN}$ activations are skipped during accesses to external Data Memory). $\overline{PSEN}$ is not activated when the device is executing out of internal Program Memory.

$\overline{EA}$/V_{PP}: When $\overline{EA}$ is held high the CPU executes out of internal Program Memory (unless the Program Counter exceeds 0FFFH in the 80C51). Holding $\overline{EA}$ low forces the CPU to execute out of external memory regardless of the Program Counter value. In the 80C31, $\overline{EA}$ must be externally wired low. In the EPROM devices, this pin also receives the programming supply voltage (V_{PP}) during EPROM programming.

XTAL1: Input to the inverting oscillator amplifier.

XTAL2: Output from the inverting oscillator amplifier.

Port 0: Port 0 is an 8-bit open drain bidirectional port. As an open drain output port, it can sink eight LS TTL loads. Port 0 pins that have 1s written to them float, and in that state will function as high impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external memory. In this application it uses strong internal pullups when emitting 1s. Port 0 emits code bytes during program verification. In this application, external pullups are required.

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, port 1 pins that are externally being pulled low will source current because of the internal pullups.

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. Port 2 emits the high-order address byte during accesses to external memory that use 16-bit addresses. In this application, it uses the strong internal pullups when emitting 1s.

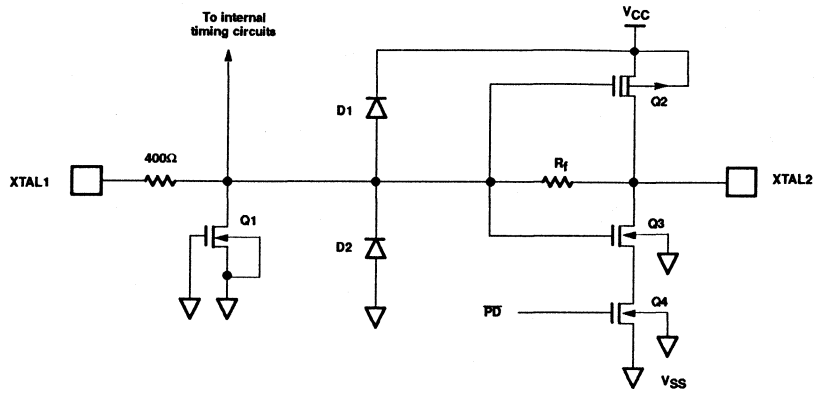


Figure 48. On-Chip Oscillator Circuitry in the CMOS Version of the 80C51 Family

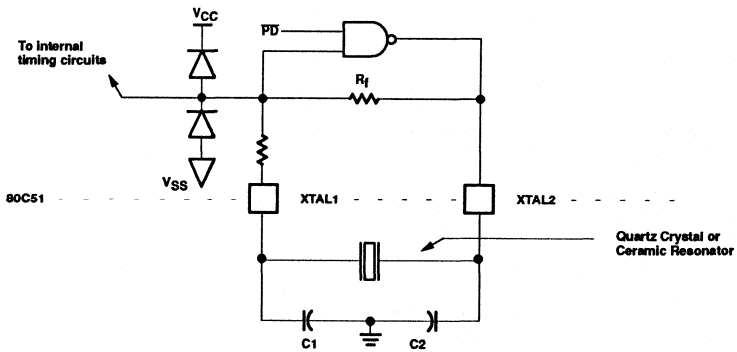


Figure 49. Using the CMOS On-Chip Oscillator

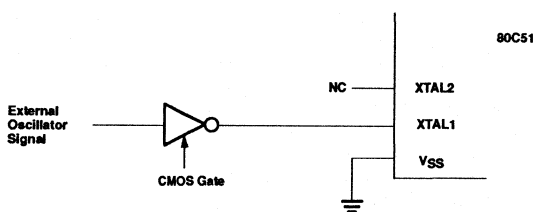


Figure 50. Driving the CMOS Family Parts with an External Clock Source

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. It also serves the functions of various special features of the 80C51 Family as follows:

Port Pin	Alternate Function
P3.0	RxD (serial input port)
P3.1	TxD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

V_{CC}: Supply voltage

V_{SS}: Circuit ground potential

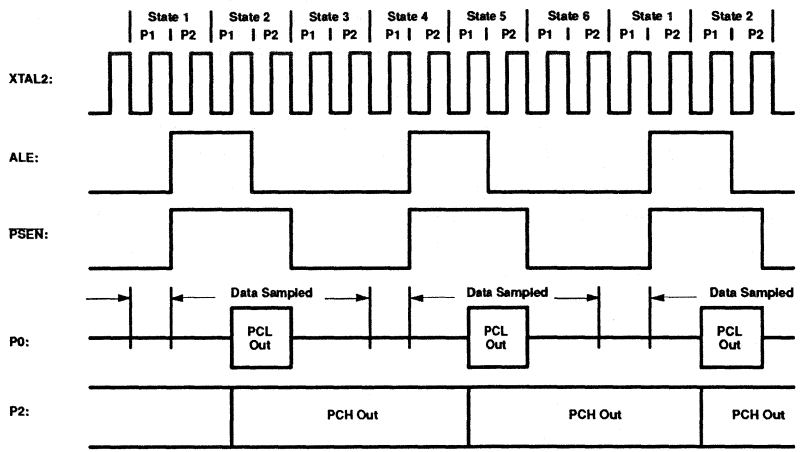


Figure 51. External Program Memory Fetches

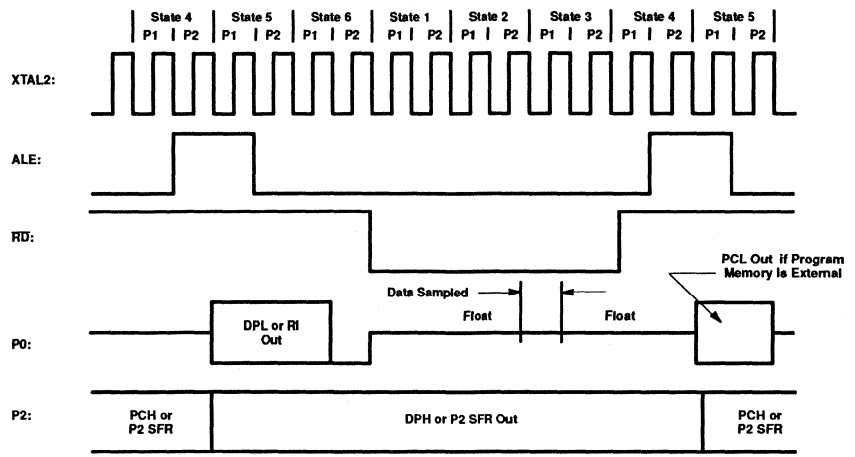


Figure 52. External Data Memory Read Cycle

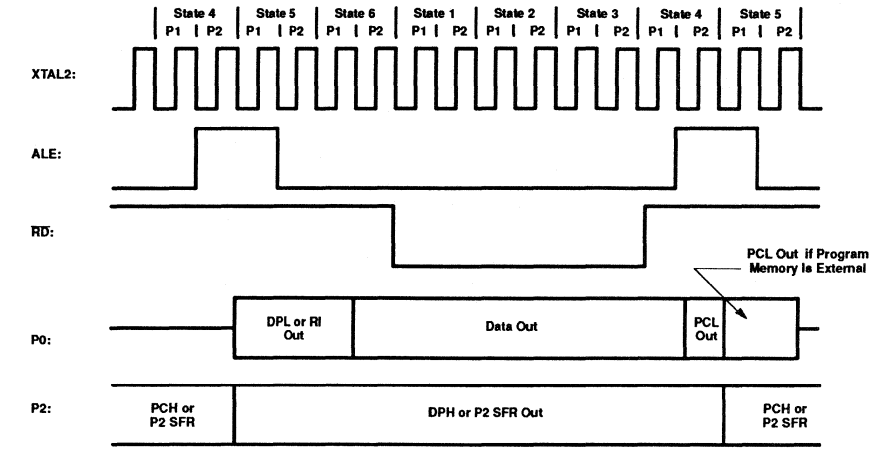


Figure 53. External Data Memory Write Cycle

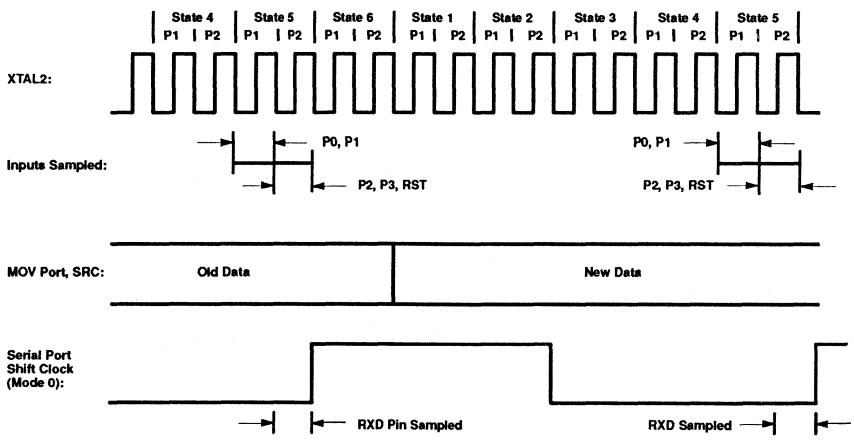


Figure 54. Port Operation

PROGRAMMER'S GUIDE AND INSTRUCTION SET

Memory Organization

Program Memory

The 80C51 has separate address spaces for program and data memory. The Program memory can be up to 64k bytes long. The lower 4k can reside on-chip. Figure 55 shows a map of the 80C51 program memory.

The 80C51 can address up to 64k bytes of data memory to the chip. The MOVX instruction is used to access the external data memory.

The 80C51 has 128 bytes of on-chip RAM, plus a number of Special Function Registers (SFRs). The lower 128 bytes of RAM can be accessed either by direct addressing (MOV data addr) or by indirect addressing (MOV @Ri). Figure 56 shows the Data Memory organization.

Direct and Indirect Address Area

The 128 bytes of RAM which can be accessed by both direct and indirect addressing can be divided into three segments as listed below and shown in Figure 57.

1. Register Banks 0-3: Locations 0 through 1FH (32 bytes). The device after reset defaults to register bank 0. To use the other register banks, the user must select them in software. Each register bank contains eight 1-byte registers 0 through 7. Reset initializes the stack pointer to location 07H, and it is incremented once to start from location 08H, which is the first register (R0) of the second register bank. Thus, in order to use more than one register bank, the SP should be initialized to a different location of the RAM where it is not used for data storage (i.e., the higher part of the RAM).
2. Bit Addressable Area: 16 bytes have been assigned for this segment, 20H-2FH. Each one of the 128 bits of this segment can be directly addressed (0-7FH). The bits can be referred to in two ways, both of which are acceptable by most assemblers. One way is to refer to their address (i.e., 0-7FH). The other way is with reference to bytes 20H to 2FH. Thus, bits 0-7 can also be referred to as bits 20.0-20.7, and bits 8-FH are the same as 21.0-21.7, and so on. Each of the 16 bytes in this segment can also be addressed as a byte.
3. Scratch Pad Area: 30H through 7FH are available to the user as data RAM. However, if the data pointer has been initialized to this area, enough bytes should be left aside to prevent SP data destruction.

Figure 56 shows the different segments of the on-chip RAM.

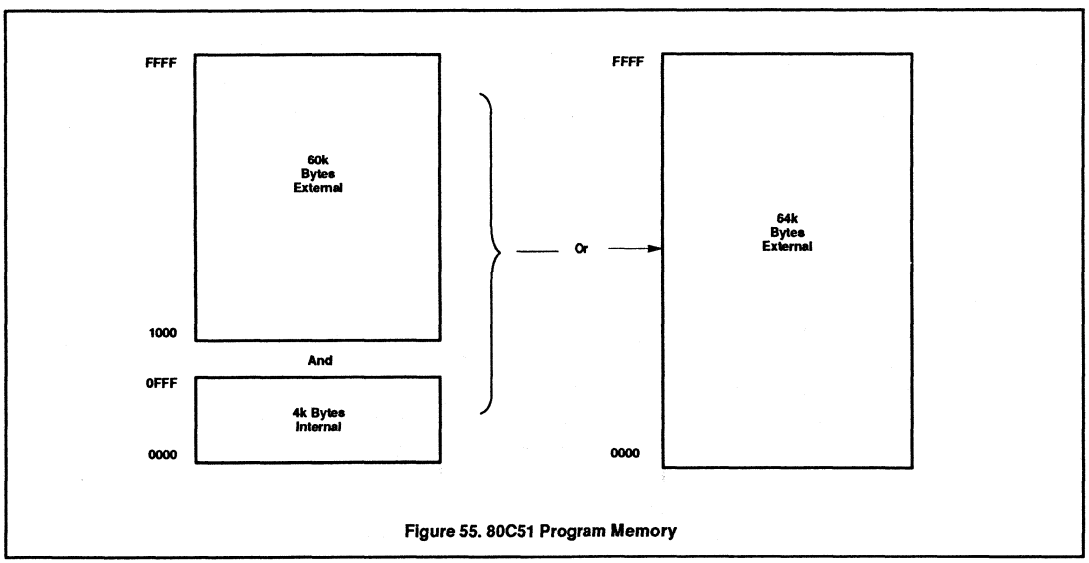


Figure 55. 80C51 Program Memory

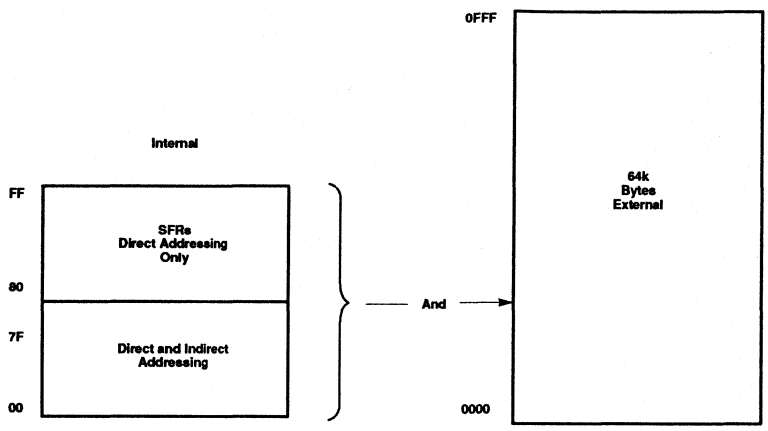


Figure 56. 80C51 Data Memory

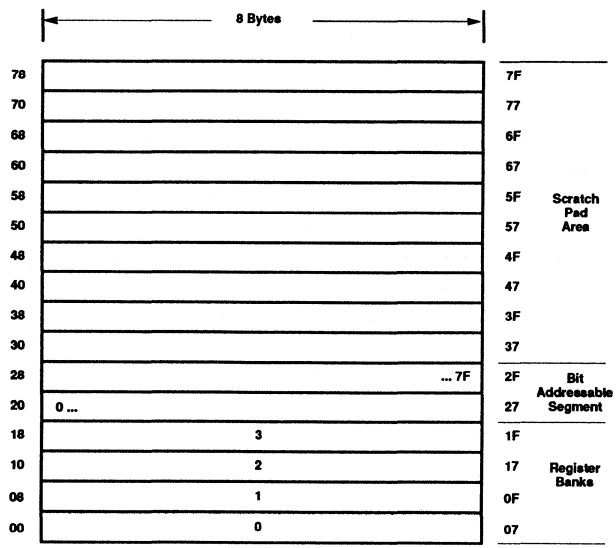


Figure 57. 128 Bytes of RAM Direct and Indirect Addressable

Table 12. 80C51 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR	Data pointer (2 bytes)										
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*	Interrupt enable	A8H	EA	–	–	ES	ET1	EX1	ET0	EX0	0x000000B
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*	Interrupt priority	B8H	–	–	–	PS	PT1	PX1	PT0	PX0	xx000000B
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	FFH
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	–	–	–	–	–	–	T2EX	T2	FFH
			A7	A6	A5	A4	A3	A2	A1	A0	
P2*	Port 2	A0H	A15	A14	A13	A12	A11	A10	A9	A8	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	$\overline{RS}$	$\overline{WR}$	T0	T1	$\overline{INT1}$	$\overline{INT0}$	TxD	Rxd	FFH
PCON[1]	Power control	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxxxxxxB
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
SBUF	Serial data buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
SCON*	Serial controller	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
TH0	Timer high 0	8CH									00H
TH1	Timer high 1	8DH									00H
TL0	Timer low 0	8AH									00H
TL1	Timer low 1	8BH									00H
TMOD	Timer mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H

*Bit addressable

1. Bits GF1, GF0, PD, and IDL of the PCON register are not implemented on the NMOS 8051/8031.

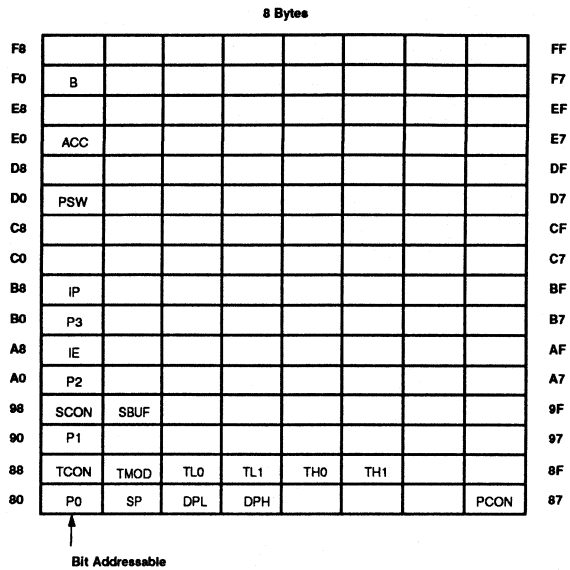


Figure 58. SFR Memory Map

Those SFRs that have their bits assigned for various functions are listed in this section. A brief description of each bit is provided for quick reference. For more detailed information refer to the Architecture Chapter of this book.

PSW: PROGRAM STATUS WORD. BIT ADDRESSABLE.

CY	AC	F0	RS1	RS0	OV	–	P

CY	PSW.7	Carry Flag.
AC	PSW.6	Auxiliary Carry Flag.
F0	PSW.5	Flag 0 available to the user for general purpose.
RS1	PSW.4	Register Bank selector bit 1 (SEE NOTE 1).
RS0	PSW.3	Register Bank selector bit 0 (SEE NOTE 1).
OV	PSW.2	Overflow Flag.
–	PSW.1	Usable as a general purpose flag.
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of '1' bus in the accumulator.

NOTE:
1. The value presented by RS0 and RS1 selects the corresponding register bank.

RS1	RS0	REGISTER BANK	ADDRESS
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

PCON: POWER CONTROL REGISTER. NOT BIT ADDRESSABLE.

SMOD	–	–	–	GF1	GF0	PD	IDL

SMOD	Double baud rate bit. If Timer 1 is used to generate baud rate and SMOD = 1, the baud rate is doubled when the Serial Port is used in modes 1, 2, or 3.
–	Not implemented, reserved for future use.*
–	Not implemented reserved for future use.*
–	Not implemented reserved for future use.*
GF1	General purpose flag bit.
GF0	General purpose flag bit.
PD	Power Down Bit. Setting this bit activates Power Down operation in the 80C51. (Available only in CMOS.)
IDL	Idle mode bit. Setting this bit activates Idle Mode operation in the 80C51. (Available only in CMOS.)

If 1s are written to PD and IDL at the same time, PD takes precedence.

*User software should not write 1s to reserved bits. These bits may be used in future 8051 products to invoke new features.

INTERRUPTS:

To use any of the interrupts in the 80C51 Family, the following three steps must be taken.

1. **Set the EA (enable all) bit in the IE register to 1.**
2. **Set the corresponding individual interrupt enable bit in the IE register to 1.**
3. **Begin the interrupt service routine at the corresponding Vector Address of that interrupt. See Table below.**

INTERRUPT SOURCE	VECTOR ADDRESS
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI & TI	0023H

In addition, for external interrupts, pins INT0 and INT1 (P3.2 and P3.3) must be set to 1, and depending on whether the interrupt is to be level or transition activated, bits IT0 or IT1 in the TCON register may need to be set to 1.

ITx = 0 level activated

ITx = 1 transition activated

IE: INTERRUPT ENABLE REGISTER. BIT ADDRESSABLE.

If the bit is 0, the corresponding interrupt is disabled. If the bit is 1, the corresponding interrupt is enabled.

EA	–	–	ES	ET1	EX1	ET0	EX0

EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	Not implemented, reserved for future use.*
—	IE.5	Not implemented, reserved for future use.*
ES	IE.4	Enable or disable the serial port interrupt.
ET1	IE.3	Enable or disable the Timer 1 overflow interrupt.
EX1	IE.2	Enable or disable External Interrupt 1.
ET0	IE.1	Enable or disable the Timer 0 overflow interrupt.
EX0	IE.0	Enable or disable External Interrupt 0.

*User software should not write 1s to reserved bits. These bits may be used in future 80C51 products to invoke new features.

ASSIGNING HIGHER PRIORITY TO ONE OR MORE INTERRUPTS:

In order to assign higher priority to an interrupt the corresponding bit in the IP register must be set to 1.

Remember that while an interrupt service is in progress, it cannot be interrupted by a lower or same level interrupt.

PRIORITY WITHIN LEVEL:

Priority within level is only to resolve simultaneous requests of the same priority level.

From high to low, interrupt sources are listed below:

IE0
TF0
IE1
TF1
RI or TI

IP: INTERRUPT PRIORITY REGISTER. BIT ADDRESSABLE.

If the bit is 0, the corresponding interrupt has a lower priority and if the bit is 1 the corresponding interrupt has a higher priority.

–	–	–	PS	PT1	PX1	PT0	PX0

–	IP.7	Not implemented, reserved for future use.*
–	IP.6	Not implemented, reserved for future use.*
–	IP.5	Not implemented, reserved for future use.*
PS	IP.4	Defines the Serial Port interrupt priority level.
PT1	IP.3	Defines the Timer 1 interrupt priority level.
PX1	IP.2	Defines External Interrupt 1 priority level.
PT0	IP.1	Defines the Timer 0 interrupt priority level.
PX0	IP.0	Defines the External Interrupt 0 priority level.

*User software should not write 1s to reserved bits. These bits may be used in future 80C51 products to invoke new features.

TCON: TIMER/COUNTER CONTROL REGISTER. BIT ADDRESSABLE.

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1	TCON.7	Timer 1 overflow flag. Set by hardware when the Timer/Counter 1 overflows. Cleared by hardware as processor vectors to the interrupt service routine.
TR1	TCON.6	Timer 1 run control bit. Set/cleared by software to turn Timer/Counter 1 ON/OFF.
TF0	TCON.5	Timer 0 overflow flag. Set by hardware when the Timer/Counter 0 overflows. Cleared by hardware as processor vectors to the service routine.
TR0	TCON.4	Timer 0 run control bit. Set/cleared by software to turn Timer/Counter 0 ON/OFF.
IE1	TCON.3	External Interrupt 1 edge flag. Set by hardware when External Interrupt edge is detected. Cleared by hardware when interrupt is processed.
IT1	TCON.2	Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.
IE0	TCON.1	External Interrupt 0 edge flag. Set by hardware when External Interrupt edge detected. Cleared by hardware when interrupt is processed.
IT0	TCON.0	Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.

TMOD: TIMER/COUNTER MODE CONTROL REGISTER. NOT BIT ADDRESSABLE.

GATE	C/T	M1	M0	GATE	C/T	M1	M0
		Timer 1				Timer 0	

GATE	When TRx (in TCON) is set and GATE = 1, TIMER/COUNTERx will run only while INTx pin is high (hardware control). When GATE = 0, TIMER/COUNTERx will run only while TRx = 1 (software control).
C/T	Timer or Counter selector. Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from Tx input pin).
M1	Mode selector bit. (NOTE 1)
M0	Mode selector bit. (NOTE1)

NOTE 1:

M1	M0		Operating Mode
o	0	0	13-bit Timer (8048 compatible)
o	0	1	16-bit Timer/Counter
1	0	2	8-bit Auto-Reload Timer/Counter
1	1	3	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standart Timer 0 control bits. TH0 is an8-bit Timer and is controlled by Timer 1 control bits.
1	1	3	(Timer 1) Timer/Counter 1 stopped.

TIMER SET-UP

Tables 13 through 16 give some values for TMOD which can be used to set up Timer 0 in different modes.

It is assumed that only one timer is being used at a time. If it is desired to run Timers 0 and 1 simultaneously, in any mode, the value in TMOD for Timer 0 must be ORed with the value shown for Timer 1 (Table 16 and 17).

For example, if it is desired to run Timer 0 in mode 1 GATE (external control), and Timer 1 in mode 2 COUNTER, then the value that must be loaded into TMOD is 69H (09H from Table 14 ORed with 60H from Table 17).

Moreover, it is assumed that the user, at this point, is not ready to turn the timers on and will do that at a different point in the program by setting bit TRx (in TCON) to 1.

TIMER/COUNTER 0

As a Timer:

Table 13

MODE	TIMER 0 FUNCTION	TMOD	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	00H	08H
1	16-bit Timer	01H	09H
2	8-bit Auto-Reload	02H	0AH
3	Two 8-bit Timers	03H	0BH

As a Counter:

Table 14

MODE	COUNTER 0 FUNCTION	TMOD	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	04H	0CH
1	16-bit Timer	05H	0DH
2	8-bit Auto-Reload	06H	0EH
3	One 8-bit Counter	07H	0FH

NOTES:
1. The timer is turned ON/OFF by setting/clearing bit TR0 in the software.
2. The Timer is turned ON/OFF by the 1-to-0 transition on $\overline{INT0}$ (P3.2) when TR0 = 1 (hardware control).

TIMER/COUNTER 1

As a Timer:

Table 15

MODE	TIMER 1 FUNCTION	TMOD	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	00H	80H
1	16-bit Timer	10H	90H
2	8-bit Auto-Reload	20H	A0H
3	Does not run	30H	B0H

As a Counter:

Table 16

MODE	COUNTER 1 FUNCTION	TMOD	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	40H	C0H
1	16-bit Timer	50H	D0H
2	8-bit Auto-Reload	60H	E0H
3	Not available	–	–

NOTES:
1. The timer is turned ON/OFF by setting/clearing bit TR1 in the software.
2. The Timer is turned ON/OFF by the 1-to-0 transition on $\overline{INT1}$ (P3.2) when TR1 = 1 (hardware control).

SCON: SERIAL PORT CONTROL REGISTER. BIT ADDRESSABLE.

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0	SCON.7	Serial Port mode specifier. (NOTE 1)
SM1	SCON.6	Serial Port mode specifier. (NOTE 1)
SM2	SCON.5	Enables the multiprocessor communication feature in modes 2 & 3. In mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0. (See Table 17.)
REN	SCON.4	Set/Cleared by software to Enable/Disable reception.
TB8	SCON.3	The 9th bit that will be transmitted in modes 2 & 3. Set/Cleared by software.
RB8	SCON.2	In modes 2 & 3, is the 9th data bit that was received. In mode 1, if SM2 = 0, RB8 is the stop bit that was received. In mode 0, RB8 is not used.
TI	SCON.1	Transmit interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or at the beginning of the stop bit in the other modes. Must be cleared by software.
RI	SCON.0	Receive interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or halfway through the stop bit time in the other modes (except see SM2). Must be cleared by software.

NOTE 1:

SM0	SM1	Mode	Description	Baud Rate
0	0	0	Shift Register	$F_{OSC.}/12$
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	$F_{OSC.}/64$ or $F_{OSC.}/32$
1	1	3	9-bit UART	Variable

SERIAL PORT SET-UP:

Table 17

MODE	SCON	SM2 VARIATION
0 1 2 3	10H 50H 90H D0H	Single Processor Environment (SM2 = 0)
0 1 2 3	NA 70H B0H F0H	Multiprocessor Environment (SM2 = 1)

GENERATING BAUD RATES

Serial Port in Mode 0:

Mode 0 has a fixed baud rate which is 1/12 of the oscillator frequency. To run the serial port in this mode none of the Timer/Counters need to be set up. Only the SCON register needs to be defined.

$$\text{Baud Rate} = \frac{\text{Osc Freq}}{12}$$

Serial Port in Mode 1:

Mode 1 has a variable baud rate. The baud rate is generated by Timer 1.

USING TIMER/COUNTER 1 TO GENERATE BAUD RATES:

For this purpose, Timer 1 is used in mode 2 (Auto-Reload). Refer to Timer Setup section of this chapter.

$$\text{Baud Rate} = \frac{K \times \text{Osc Freq}}{32 \times 12 \times [256 - (TH1)]}$$

If SMOD = 0, then K = 1.
If SMOD = 1, then K = 2 (SMOD is in the PCON register).

Most of the time the user knows the baud rate and needs to know the reload value for TH1.

$$TH1 = 256 - \frac{K \times \text{Osc Freq}}{384 \times \text{baud rate}}$$

TH1 must be an integer value. Rounding off TH1 to the nearest integer may not produce the desired baud rate. In this case, the user may have to choose another crystal frequency.

Since the PCON register is not bit addressable, one way to set the bit is logical ORing the PCON register (i.e., ORL PCON,#80H). The address of PCON is 87H.

SERIAL PORT IN MODE 2:

The baud rate is fixed in this mode and is 1/32 or 1/64 of the oscillator frequency, depending on the value of the SMOD bit in the PCON register.

In this mode none of the Timers are used and the clock comes from the internal phase 2 clock.

SMOD = 1, Baud Rate = 1/32 Osc Freq.

SMOD = 0, Baude Rate = 1/64 Osc Freq.

To set the SMOD bit: ORL PCON,#80H. The address of PCON is 87H.

SERIAL PORT IN MODE 3:

The baud rate in mode 3 is variable and sets up exactly the same as in mode 1.

80C51 Family Instruction Set

Table 18. 80C51 Instruction Set Summary

Interrupt Response Time: Refer to Hardware Description Chapter.

Instructions that Affect Flag Settings[1]

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	0	X		ANL C,/bit	X		
DIV	0	X		ANL C,bit	X		
DA	X			ORL C,/bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

[1]Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.

Notes on instruction set and addressing modes:

Rn	Register R7-R0 of the currently selected Register Bank.
direct	8-bit internal data location's address. This could be an Internal Data RAM location (0-127) or a SFR [i.e., I/O port, control register, status register, etc. (128-255)].
@Ri	8-bit internal data RAM location (0-255) addressed indirectly through register R1 or R0.
#data	8-bit constant included in the instruction.
#data 16	16-bit constant included in the instruction
addr 16	16-bit destination address. Used by LCALL and LJMP. A branch can be anywhere within the 64k-byte Program Memory address space.
addr 11	11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2k-byte page of program memory as the first byte of the following instruction.
rel	Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is –128 to +127 bytes relative to first byte of the following instruction.
bit	Direct Addressed bit in Internal Data RAM or Special Function Register.

MNEMONIC		DESCRIPTION	BYTE	OSCILLATOR PERIOD
ARITHMETIC OPERATIONS				
ADD	A,Rn	Add register to Accumulator	1	12
ADD	A,direct	Add direct byte to Accumulator	2	12
ADD	A,@Ri	Add indirect RAM to Accumulator	1	12
ADD	A,#data	Add immediate data to Accumulator	2	12
ADDC	A,Rn	Add register to Accumulator with carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with carry	2	12
ADDC	A,@Ri	Add indirect RAM to Accumulator with carry	1	12
ADDC	A,#data	Add immediate data to A_{CC} with carry	2	12
SUBB	A,Rn	Subtract Register from A_{CC} with borrow	1	12
SUBB	A,direct	Subtract direct byte from A_{CC} with borrow	2	12
SUBB	A,@Ri	Subtract indirect RAM from A_{CC} with borrow	1	12
SUBB	A,#data	Subtract immediate data from A_{CC} with borrow	2	12
INC	A	Increment Accumulator	1	12
INC	Rn	Increment register	1	12

MNEMONIC		DESCRIPTION	BYTE	OSCILLATOR PERIOD
ARITHMETIC OPERATIONS (Continued)				
INC	direct	Increment direct byte	2	12
INC	@Ri	Increment indirect RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	Rn	Decrement Register	1	12
DEC	direct	Decrement direct byte	2	12
DEC	@Ri	Decrement indirect RAM	1	12
INC	DPTR	Increment Data Pointer	1	24
MUL	AB	Multiply A and B	1	48
DIV	AB	Divide A by B	1	48
DA	A	Decimal Adjust Accumulator	1	12
LOGICAL OPERATIONS				
ANL	A,Rn	AND Register to Accumulator	1	12
ANL	A,direct	AND direct byte to Accumulator	2	12
ANL	A,@Ri	AND indirect RAM to Accumulator	1	12
ANL	A,#data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
ANL	direct,#data	AND immediate data to direct byte	3	24
ORL	A,Rn	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@Ri	OR indirect RAM to Accumulator	1	12
ORL	A,#data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct,#data	OR immediate data to direct byte	3	24
XRL	A,Rn	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL	A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12
RL	A	Rotate Accumulator left	1	12
RLC	A	Rotate Accumulator left through the carry	1	12
RR	A	Rotate Accumulator right	1	12
RRC	A	Rotate Accumulator right through the carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER				
MOV	A,Rn	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@Ri	Move indirect RAM to Accumulator	1	12

MNEMONIC		DESCRIPTION	BYTE	OSCILLATOR PERIOD
DATA TRANSFER (Continued)				
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	Rn,A	Move Accumulator to register	1	12
MOV	Rn,direct	Move direct byte to register	2	24
MOV	RN,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,Rn	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@Ri	Move indirect RAM to direct byte	2	24
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@Ri,A	Move Accumulator to indirect RAM	1	12
MOV	@Ri,direct	Move direct byte to indirect RAM	2	24
MOV	@Ri,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to A_{CC}	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to A_{CC}	1	24
MOVX	A,@Ri	Move external RAM (8-bit addr) to A_{CC}	1	24
MOVX	A@DPTR	Move external RAM (16-bit addr) to A_{CC}	1	24
MOVX	A,@Ri,A	Move A_{CC} to external RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move A_{CC} to external RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,Rn	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@Ri	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@Ri	Exchange low-order digit indirect RAM with A_{CC}	1	12
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to carry	2	24
ANL	C,/bit	AND complement of direct bit to carry	2	24
ORL	C,bit	OR direct bit to carry	2	24
ORL	C,/bit	OR complement of direct bit to carry	2	24
MOV	C,bit	Move direct bit to carry	2	12
MOV	bit,C	Move carry to direct bit	2	24
JC	rel	Jump if carry is set	2	24
JNC	rel	Jump if carry not set	2	24

MNEMONIC		DESCRIPTION	BYTE	OSCILLATOR PERIOD
BOOLEAN VARIABLE MANIPULATION (Continued)				
JB	rel	Jump if direct bit is set	2	24
JNB	rel	Jump if direct bit is not set	2	24
JBC	bit,rel	Jump if direct bit is set and clear bit	3	24
PROGRAM BRANCHING				
ACALL	addr11	Absolute subroutine call	2	24
LCALL	addr16	Long subroutine call	3	24
RET		Return from subroutine	1	24
RETI		Return from interrupt	1	24
AJMP	addr11	Absolute jump	2	24
LJMP	addr16	Long jump	3	24
SJMP	rel	Short jump (relative addr)	2	24
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is zero	2	24
JNZ	rel	Jump if Accumulator is not zero	2	24
CJNE	A,direct,rel	Compare direct byte to A_{CC} and jump if not equal	3	24
CJNE	A,#data,rel	Compare immediate to A_{CC} and jump if not equal	3	24
CJNE	RN,#data,rel	Compare immediate to register and jump if not equal	3	24
CJNE	@Ri,#data,rel	Compare immediate to indirect and jump if not equal	3	24
DJNZ	Rn,rel	Decrement register and jump if not zero	2	24
DJNZ	direct,rel	Decrement direct byte and jump if not zero	3	24
NOP		No opertion	1	12

INSTRUCTION DEFINITIONS

ACALL addr11

Function: Absolute Call

Description: ACALL unconditionally calls a subroutine located at the indicated address. The instruction increments the PC twice to obtain the address of the following instruction, then pushes the 16-bit result onto the stack (low-order byte first) and increments the Stack Pointer twice. The destination address is obtained by successively concatenating the five high-order bits of the incremented PC, opcode bits 7-5, and the second byte of the instruction. The subroutine called must therefore start within the same 2k block of the program memory as the first byte of the instruction following ACALL. No flags are affected.

Example: Initially SP equals 07H. The label "SUBRTN" is at program memory location 0345 H. After executing the instruction,

ACALL SUBRTN

at location 0123H, SP will contain 09H, internal RAM locations 08H and 09H will contain 25H and 01H, respec tively, and the PC will contain 0345H.

Bytes: 2

Cycles: 2

Encoding:

a10 a9 a8 1	0 0 0 1	a7 a6 a5 a4	a3 a2 a1 a0

Operation: ACALL

(PC) ← (PC) + 2
(SP) ← (SP) + 1
(SP) ← (PC_{7-0})
(SP) ← (SP) + 1
(SP) ← (PC_{15-8})
(PC_{10-0}) ← page address

ADD A,<src-byte>

Function: Add

Description: ADD adds the byte variable indicated to the Accumulator, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B). The instruction,

ADD A,R0

will leave 6DH (01101101B) in the Accumulator with the AC flag cleared and both the Carry flag and OV set to 1.

ADD A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 0 1 0	1 r r r

Operation: ADD

$(A) \leftarrow (A) + (R_n)$

ADD A,direct

Bytes: 2

Cycles: 1

Encoding:

0 0 1 0	0 1 0 1	direct address

Operation: ADD

(A) ← (A) + (direct)

ADD A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 1 0	0 1 1 i

Operation: ADD

$(A) \leftarrow (A) + ((R_i))$

ADD A,#data

Bytes: 2

Cycles: 1

Encoding:

0 0 1 0	0 1 0 0	immediate data

Operation: ADD

(A) ← (A) + #data

ADDC A,<src-byte>

Function: Add with Carry

Description: ADDC simultaneously adds the byte variable indicated, the carry flag and the Accumulator contents, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not out of bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) with the carry flag set. The instruction,

ADDC A,R0

will leave 6EH (01101110B) in the Accumulator with AC cleared and both the Carry flag and OV set to 1.

ADDC A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 0 1 1	1 r r r

Operation: ADDC

(A) ← (A) + (C) + (R_n)

ADDC A,direct

Bytes: 2

Cycles: 1

Encoding:

0 0 1 1	0 1 0 1	direct address

Operation: ADDC

(A) ← (A) + (C) + (direct)

ADDC A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 1 1	0 1 1 i

Operation: ADDC

(A) ← (A) + (C) + ((R_i))

ADDC A,#data

Bytes: 2

Cycles: 1

Encoding:

0 0 1 1	0 1 0 0	immediate data

Operation: ADDC

(A) ← (A) + (C) + #data

AJMP addr11

Function: Absolute Jump

Description: AJMP transfers program execution to the indicated address, which is formed at run-time by concatenating the high-order five bits of the PC (*after* incrementing the PC twice), opcode bits 7-5, and the second byte of the instruction. The destination must therefore be within the same 2k block of program memory as the first byte of the instruction following AJMP.

Example: The label "JMPADR" is at program memory location 0123H. The instruction,

AJMP JMPADR

is at location 0345H and will load the PC with 0123H.

Bytes: 2

Cycles: 2

Encoding:

a10 a9 a8 0	0 0 0 1	a7 a6 a5 a4	a3 a2 a1 a0

Operation: AJMP

(PC) ← (PC) + 2

($PC_{10\text{-}0}$) ← page address

ANL <dest-byte>,<src-byte>

Function: Logical-AND for byte variables

Description: ANL performs the bitwise logical-AND operation between the variables indicated and stores the results in the destination variable. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: If the Accumulator holds 0C3H (11000011B) and register 0 holds 55H (01010101B) then the instruction,

ANL A,R0

will leave 41H (01000001B) in the Accumulator.

When the destination is a directly addressed byte, this instruction will clear combinations of bits in any RAM location or hardware register. The mask byte determining the pattern of bits to be cleared would either be a constant contained in the instruction or a value computed in the Accumulator at run-time. The instruction,

ANL P1,#0111001B

will clear bits 7, 3, and 2 of output port 1.

ANL A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 1 0 1	1 r r r

Operation: ANL

$(A) \leftarrow (A) \wedge (R_n)$

ANL A,direct

Bytes: 2

Cycles: 1

Encoding:

0 1 0 1	0 1 0 1	direct address

Operation: ANL

$(A) \leftarrow (A) \wedge (direct)$

ANL A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0 1 0 1	0 1 1 i

Operation: ANL

$(A) \leftarrow (A) \wedge ((R_i))$

ANL A,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 0 1	0 1 0 0	immediate data

Operation: ANL

$(A) \leftarrow (A) \wedge$ #data

ANL direct,A

Bytes: 2

Cycles: 1

Encoding:

0 1 0 1	0 0 1 0	direct address

Operation: ANL

$(A) \leftarrow (direct) \wedge (A)$

ANL direct,#data

Bytes: 3

Cycles: 2

Encoding:

0 1 0 1	0 0 1 1	direct address	immediate data

Operation: ANL

$(direct) \leftarrow (direct) \wedge$ #data

ANL C,<src-bit>

Function: Logical-AND for bit variables

Description: If the Boolean value of the source bit is a logical 0 then clear the carry flag; otherwise leave the carry flag in its current state. A slash ("/") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, *but the source bit itself is not affected.* No other flags are affected.

Only direct addressing is allowed for the source operand.

Example: Set the carry flag if, and only if, P1.0 = 1, ACC.7 = 1, and OV = 0:

```
MOV   C,P1.0      ;LOAD CARRY WITH INPUT PIN STATE
ANL   C,ACC.7     ;AND CARRY WITH ACCUM. BIT 7
ANL   C,/OV       ;AND WITH INVERSE OF OVERFLOW FLAG
```

ANL C,bit

Bytes: 2

Cycles: 2

Encoding:

1 0 0 0	0 0 1 0	bit address

Operation:

ANL

(C) ← (C) ∧ (bit)

ANL C,/bit

Bytes: 2

Cycles: 2

Encoding:

1 0 1 1	0 0 0 0	bit address

Operation:

ANL

(C) ← (C) ∧ ⌝(bit)

CJNE <dest-byte>,<src-byte>,rel

Function: Compare and Jump if Not Equal

Description: CJNE compares the magnitudes of the first two operands, and branches if their values are not equal. The branch destination is computed by adding the signed relative-displacement in the last instruction byte to the PC, after incrementing the PC to the start of the next instruction. The carry flag is set if the unsigned integer value of <dest-byte> is less than the unsigned integer value of <src-byte>; otherwise, the carry is cleared. Neither operand is affected.

The first two operands allow four addressing mode combinations: the Accumulator may be compared with any directly addressed byte or immediate data, and any indirect RAM location or working register can be compared with an immediate constant.

Example: The Accumulator contains 34H. Register 7 contains 56H. The first instruction in the sequence,

```
         CJNE    R7,#60H,NOT_EQ
;        ...     ....          ;  R7 = 60H.
NOT_EQ   JC      REQ_LOW       ;  IF R7 < 60H.
;        ...     ....          ;  R7 > 60H.
```

sets the carry flag and branches to the instruction at label NOT_EQ. By testing the carry flag, this instruction determines whether R7 is greater or less than 60H.

If the data being presented to Port 1 is also 34H, then the instruction,

```
WAIT:  CJNE  A,P1,WAIT
```

clears the carry flag and continues with the next instruction in sequence, since the Accumulator does equal the data read from P1. (If some other value was being input on P1, the program will loop at this point until the P1 data changes to 34H.)

CJNE A,direct,rel

Bytes: 3

Cycles: 2

Encoding:

1 0 1 1	0 1 0 1	direct address	rel. address

Operation:

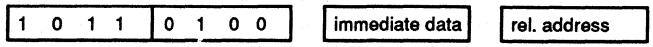
(PC) ← (PC) + 3
IF (A) < > *(direct)*
THEN
    (PC) ← (PC) + *relative offset*
IF (A) < *(direct)*
THEN
    (C) ← 1
ELSE
    (C) ← 0

CJNE A,#data,rel

Bytes: 3

Cycles: 2

Encoding:



Operation:

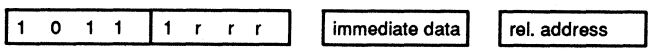
(PC) ← (PC) + 3
IF (A) < > *data*
THEN
(PC) ← (PC) + *relative offset*
IF (A) < *data*
THEN
(C) ← 1
ELSE
(C) ← 0

CJNE Rn,#data,rel

Bytes: 3

Cycles: 2

Encoding:



Operation:

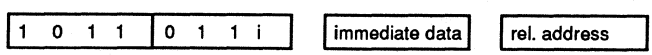
(PC) ← (PC) + 3
IF (R_n) < > *data*
THEN
(PC) ← (PC) + *relative offset*
IF (R_n) < *data*
THEN
(C) ← 1
ELSE
(C) ← 0

CJNE @Ri,#data,rel

Bytes: 3

Cycles: 2

Encoding:



Operation:

(PC) ← (PC) + 3
IF ((R_i))< > *data*
THEN
(PC) ← (PC) + *relative offset*
IF ((R_i)) < *data*
THEN
(C) ← 1
ELSE
(C) ← 0

CLR A

Function: Clear Accumulator

Description: The Accumulator is cleared (all bits reset to zero). No flags are affected.

Example: The Accumulator contains 5CH (01011100B). The instruction,

CLR A

will leave the Accumulator set to 00H (00000000B).

Bytes: 1

Cycles: 1

Encoding:

1 1 1 0	0 1 0 0

Operation:

CLR
(A) ← 0

CLR bit

Function: Clear bit

Description: The indicated bit is cleared (reset to zero). No other flags are affected. CLR can operate on the carry flag or any directly addressable bit.

Example: Port 1 has previously been written with 5DH (01011101B). The instruction,

CLR P1.2

will leave the port set to 59H (01011001B).

CLR C

Bytes: 1

Cycles: 1

Encoding:

1 1 0 0	0 0 1 1

Operation:

CLR
(C) ← 0

CLR bit

Bytes: 2

Cycles: 1

Encoding:

1 1 0 0	0 0 1 0	bit address

Operation:

CLR
(bit) ← 0

CPL A

Function: Complement Accumulator

Description: Each bit of the Accumulator is logically complemented (one's complement). Bits which previously contained a one are changed to a zero and vice-versa. No flags are affected.

Example: The Accumulator contains 5CH (01011100B). The instruction,

CPL A

will leave the Accumulator set to 0A3H (10100011B).

Bytes: 1

Cycles: 1

Encoding:

1 1 1 1	0 1 0 0

Operation: CPL

(A) ← ⌝ (A)

CPL bit

Function: Complement bit

Description: The bit variable specified is complemented. A bit which had been a one is changed to zero and vice-versa. No other flags are affected. CLR can operate on the carry or any directly addressable bit.

Note: When this instruction is used to modify an output pin, the value used as the original data will be read from the output data latch, *not* the input pin.

Example: Port 1 has previously been written with 5DH (01011101B). The instruction sequence,

CPL P1.1

CPL P1.2

will leave the port set to 5BH (01011011B).

CPL C

Bytes: 1

Cycles: 1

Encoding:

1 0 1 1	0 0 1 1

Operation: CPL

(C) ← ⌝ (C)

CPL bit

Bytes: 2

Cycles: 1

Encoding:

1 0 1 1	0 0 1 0	bit address

Operation: CPL

(bit) ← ⌝ (bit)

DA A

Function: Decimal-adjust Accumulator for Addition

Description: DA A adjusts the eight-bit value in the Accumulator resulting from the earlier addition of two variable (each in packed-BCD format), producing two four-bit digits. Any ADD or ADDC instruction may have been used to perform the addition.

If Accumulator bits 3-0 are greater than nine (xxx1010-xxx1111), or if the AC flag is one, six is added to the Accumulator, producing the proper BCD digit in the low-order nibble. This internal addition would set the carry flag if a carry-out of the low-order four-bit field propagated through all high-order bits, but it would not clear the carry flag otherwise.

If the carry flag is now set, or if the four high-order bits now exceed nine (1010xxx-111xxxx), these high-order bits are incremented by six, producing the proper BCD digit in the high-order nibble. Again, this would set the carry flag if there was a carry-out of the high-order bits, but wouldn't clear the carry. The carry flag thus indicates if the sum of the original two BCD variables is greater than 100, allowing multiple precision decimal addition. OV is not affected.

All of this occurs during the one instruction cycle. Essentially, this instruction performs the decimal conversion by adding 00H, 06H, 60H, or 66H to the Accumulator, depending on initial Accumulator and PSW conditions.

Note: DA A *cannot* simply convert a hexadecimal number in the Accumulator to BCD notation, nor does DA A apply to decimal subtraction.

Example: The Accumulator holds the value 56H (01010110B) representing the packed BCD digits of the decimal number 56. Register 3 contains the value 67H (01100111B) representing the packed BCD digits of the decimal number 67. The carry flag is set.. The instruction sequence,

```
ADDC    A,R3
DA      A
```

will first perform a standard two's-complement binary addition, resulting in the value 0BEH (10111110B) in the Accumulator. The carry and auxiliary carry flags will be cleared.

The Decimal Adjust instruction will then alter the Accumulator to the value 24H (00100100B), indicating the packed BCD digits of the decimal number 24, the low-order two digits of the decimal sum of 56, 67, and the carry-in. The carry flag will be set by the Decimal Adjust instruction, indicating that a decimal overflow occurred. The true sum 56, 67, and 1 is 124.

BCD variables can be incremented or decremented by adding 01H or 99H. If the Accumulator initially holds 30H (representing the digits of 30 decimal), the the instruction sequence,

```
ADD     A,#99H
DA      A
```

will leave the carry set and 29H in the Accumulator, since 30 + 99 = 129. The low-order byte of the sum can be interpreted to mean 30 – 1 = 29.

Bytes: 1

Cycles: 1

Encoding:

1	1	0	1	0	1	0	0

Operation:

```
DA
–contents of Accumulator are BCD
IF   [[(A3-0) > 9] v [(AC) = 1]]
     THEN(A3-0) ← (A3-0) + 6
          AND

IF   [[(A7-4) > 9] v [(C) = 1]]
     THEN(A7-4) ← (A7-4) + 6
```

DEC byte

Function: Decrement

Description: The variable indicated is decremented by 1. An original value of 00H will underflow to 0FFH. No flags are affected. Four operand addressing modes are allowed: accumulator, register, direct, or register-indirect.

Note: When this instruction is used to modify an output port, the value used as the original data will be read from the output data latch, *not* the input pin.

Example: Register 0 contains 7FH (01111111B). Internal RAM locations 7EH and 7FH contain 00H and 40H, respectively. The instruction sequence,

```
DEC   @R0
DEC   R0
DEC   @R0
```

will leave register 0 set to 7EH and internal RAM locations 7EH and 7FH set to 0FFH and 3FH.

DEC A

Bytes: 1

Cycles: 1

Encoding:

0 0 0 1	0 1 0 0

Operation: DEC

$(A) \leftarrow (A) - 1$

DEC Rn

Bytes: 1

Cycles: 1

Encoding:

0 0 0 1	1 r r r

Operation: DEC

$(R_n) \leftarrow (R_n) - 1$

DEC direct

Bytes: 2

Cycles: 1

Encoding:

0 0 0 1	0 1 0 1	direct address

Operation: DEC

(direct) ← (direct) – 1

DEC @Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 0 1	0 1 1 i

Operation: DEC

$((R_i)) \leftarrow ((R_i)) - 1$

DIV AB

Function: Divide

Description: DIV AB divides the unsigned eight-bit integer in the Accumulator by the unsigned eight-bit integer in register B. The Accumulator receives the integer part of the quotient; register B receives the integer remainder. The carry and OV flags will be cleared.

Exception: if B had originally contained 00H, the values returned in the Accumulator and B-register will be undefined and the overflow flag will be set. The carry flag is cleared in any case.

Example: The Accumulator contains 251 (0FBH or 11111011B) and B contains 18 (12H or 00010010B). The instruction,

DIV AB

will leave 13 in the Accumulator (0DH or 00001101B) and the value 17 (11H or 00010001B) in B, since 251 = (13 x 18) + 17. Carry and OV will both be cleared.

Bytes: 1

Cycles: 4

Encoding:

1 0 0 0	0 1 0 0

Operation: DIV

$(A)_{15-8} \leftarrow (A)/(B)$

$(B)_{7-0}$

DJNZ <byte>,<rel-addr>

Function: Decrement and Jump if Not Zero

Description: DJNZ decrements the location indicated by 1, and branches to the address indicated by the second operand if the resulting value is not zero. An original value of 00H will underflow to 0FFH. No flags are affected. The branch destination would be computed by adding the signed relative-displacement value in the last instruction byte to the PC, after incrementing the PC to the first byte of the following instruction.

The location decremented may be a register or directly addressed byte.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: Internal RAM locations 40H, 50H, and 60H contain the values 01H, 70H, and 15H, respectively. The instruction sequence,

```
DJNZ    40H,LABEL_1
DJNZ    50H,LABEL_2
DJNZ    60H,LABEL_3
```

will cause a jump to the instruction at LABEL_2 with the values 00h, 6FH, and 15H in the three RAM locations. The first jump was *not* taken because the result was zero.

This instruction provides a simple was of executing a program loop a given number of times, or for adding a moderate time delay (from 2 to 512 machine cycles) with a single instruction. The instruction sequence,

```
        MOV     R2,#8
TOGGLE: CPL     P1.7
        DJNZ    R2,TOGGLE
```

will toggle P1.7 eight times, causing four output pulses to appear at bit 7 of output Port 1. Each pulse will last three machine cycles, two for DJNZ and one to alter the pin.

DJNZ Rn,rel

Bytes: 2

Cycles: 2

Encoding:

1 1 0 1	1 r r r	rel. address

Operation:

DJNZ
(PC) ← (PC) + 2
(R_n) ← (R_n) – 1
IF (R_n) > 0 or (R_n) < 0
THEN
(PC) ← (PC) + rel

DJNZ direct,rel

Bytes: 3

Cycles: 2

Encoding:

1 1 0 1	0 1 0 1	direct data	rel. address

Operation:

DJNZ
(PC) ← (PC) + 2
(direct) ← (direct) – 1
IF (direct) > 0 or (direct) < 0
THEN
(PC) ← (PC) + rel

INC <byte>

Function: Increment

Description: INC increments the indicated variable by 1. An original value of 0FFH will overflow to 00H. No flags are affected. Three addressing modes are allowed: register, direct, or register-indirect.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: Register 0 contains 7EH (01111110B). Internal RAM locations 7EH and 7FH contain 0FFH and 40H, respectively. The instruction sequence,

```
INC   @R0
INC   R0
INC   @R0
```

will leave register 0 set to 7FH and internal RAM locations 7EH and 7FH holding (respectively) 00H and 41H.

INC A

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	0 1 0 0

Operation: INC

$(A) \leftarrow (A) + 1$

INC Rn

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	1 r r r

Operation: INC

$(R_n) \leftarrow (R_n) + 1$

INC direct

Bytes: 2

Cycles: 1

Encoding:

0 0 0 0	0 1 0 1	direct address

Operation: INC

(direct) ← (direct) + 1

INC @Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	0 1 1 i

Operation: INC

$((R_i)) \leftarrow ((R_i)) + 1$

INC DPTR

Function: Increment Data Pointer

Description: Increment the 16-bit data pointer by 1. A 16-bit increment (modulo 2^{16}) is performed; an overflow of the low-order byte of the data pointer (DPL) from 0FFH to 00H will increment the high-order byte (DPH). No flags are affected.

This is the only 16-bit register which can be incremented.

Example: Registers DPH and DPL contain 12H and 0FEH, respectively. The instruction sequence,

```
INC   DPTR
INC   DPTR
INC   DPTR
```

will change DPH and DPL to 13H and 01H.

Bytes: 1

Cycles: 2

Encoding:

1 0 1 0	0 0 1 1

Operation: INC

(DPTR) ← (DPTR) + 1

JB bit,rel

Function: Jump if Bit set

Description: If the indicated bit is a one, jump to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. *The bit tested is not modified.* No flags are affected.

Example: The data present at input port 1 is 11001010B. The Accumulator holds 56 (01010110B). The instruction sequence,

```
JB    P1.2,LABEL1
JB    ACC.2,LABEL2
```

will cause program execution to branch to the instruction at label LABEL2.

Bytes: 3

Cycles: 2

Encoding:

0 0 1 0	0 0 0 0	bit address	rel. address

Operation: JB

(PC) ← (PC) + 3

IF (bit) = 1

THEN

(PC) ← (PC) + rel

JBC bit,rel

Function: Jump if Bit is set and Clear bit

Description: If the indicated bit is a one, branch to the address indicated; otherwise proceed with the next instruction. *The bit will not be cleared if it is already a zero.* The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. No flags are affected.

Note: When this instruction is used to test an output pin, the value used as the original data will read from the output data latch, not the input pin.

Example: The Accumulator holds 56H (01010110B). The instruction sequence,

```
JBC     ACC.3,LABEL1
JBC     ACC.2,LABEL2
```

will cause program execution to continue at the instruction identified by the LABEL2, with the Accumulator modified to 52H (01010010B).

Bytes: 3

Cycles: 2

Encoding:

Operation:

```
JBC
(PC) ← (PC) + 3
IF (bit) = 1
   THEN
(bit) ← 0
(PC) ← (PC) + rel
```

JC rel

Function: Jump if Carry is set

Description: If the carry flag is set, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. No flags are affected.

Example: The carry flag is cleared. The instruction sequence,

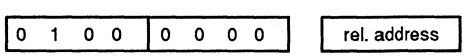
```
JC      LABEL1
CPL     C
JC      LABEL2
```

will set the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:



Operation:

```
JC
(PC) ← (PC) + 2
IF (C) = 1
   THEN
            (PC) ← (PC) + rel
```

JMP @A+DPTR

Function: Jump indirect

Description: Add the eight-bit unsigned contents of the Accumulator with the sixteen-bit data pointer, and load the resulting sum to the program counter. This will be the address for subsequent instruction fetches. Sixteen-bit addition is performed (modulo 2^{16}): a carry-out from the low-order eight bits propagates through the higher-order bits. Neither the Accumulator nor the Data Pointer is altered. No flags are affected.

Example: An even number from 0 to 6 is in the Accumulator. The following sequence of instructions will branch to one of four AJMP instructions in a jump table starting at JMP_TBL:

```
            MOV   DPTR,#JMP_TBL
            JMP   @A+DPTR
JMP_TBL:    AJMP  LABEL0
            AJMP  LABEL1
            AJMP  LABEL2
            AJMP  LABEL3
```

If the Accumulator equals 04H when starting this sequence, execution will jump to label LABEL2. Remember that AJMP is a two-byte instruction, so the jump instructions start at every other address.

Bytes: 1

Cycles: 2

Encoding:

0 1 1 1	0 0 1 1

Operation: JMP

(PC) ← (A) + (DPTR)

JNB bit,rel

Function: Jump if Bit Not set

Description: If the indicated bit is a zero, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. *The bit tested is not modified.* No flags are affected.

Example: The data present at input port 1 is 11001010B. The Accumulator holds 56H (01010110B). The instruction sequence,

```
JNB   P1.3,LABEL1
JNB   ACC.3,LABEL2
```

will cause program execution to continue at the instruction at label LABEL2.

Bytes: 3

Cycles: 2

Encoding:

0 0 1 1	0 0 0 0	bit address	rel. address

Operation: JNB

(PC) ← (PC) + 3

IF (bit) = 0

THEN

(PC) ← (PC) + rel

JNC rel

Function: Jump if Carry Not set

Description: If the carry flag is a zero, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice to point to the next instruction. The carry flag is not modified.

Example: The carry flag is set. The instruction sequence,

```
JNC   LABEL1
CPL   C
JNC   LABEL2
```

will clear the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0 1 0 1	0 0 0 0	rel. address

Operation:

```
JNC
(PC) ← (PC) + 2
IF (C) = 0
   THEN
(PC) ← (PC) + rel
```

JNZ rel

Function: Jump if Accumulator Not Zero

Description: If any bit of the Accumulator is a one, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example: The Accumulator originally holds 00H. The instruction sequence,

```
JNZ   LABEL1
JNC   A
JNZ   LABEL2
```

will set the Accumulator to 01H and continue at label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0 1 1 1	0 0 0 0	rel. address

Operation:

```
JNZ
(PC) ← (PC) + 2
IF A ≠ 0
   THEN (PC) ← (PC) + rel
```

JZ rel

Function: Jump if Accumulator Zero

Description: If all bits of the Accumulator are zero, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example: The Accumulator originally holds 01H. The instruction sequence,

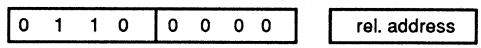
```
JZ      LABEL1
DEC     A
JZ      LABEL2
```

will change the Accumulator to 00H and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:



Operation:

```
JZ
(PC) ← (PC) + 2
IF A = 0
   THEN (PC) ← (PC) + rel
```

LCALL addr16

Function: Long Call

Description: LCALL calls a subroutine located at the indicated address. The instruction adds three to the program counter to generate the address of the next instruction and then pushes the 16-bit result onto the stack (low byte first), incrementing the Stack Pointer by two. The high-order and low-order bytes of the PC are then loaded, respectively, with the second and third bytes of the LCALL instruction. Program execution continues with the instruction at this address. The subroutine may therefore begin anywhere in the full 64k-byte program memory address space. No flags are affected.

Example: Initially the Stack Pointer equals 07H. The label "SUBRTN" is assigned to program memory location 1234H. After executing the instruction,

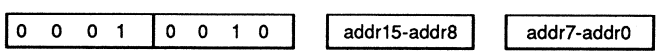
```
LCALL   SUBRTN
```

at location 0123H, the Stack Pointer will contain 09H, internal RAM locations 08H and 09H will contain 26H and 01H, and the PC will contain 1235H.

Bytes: 3

Cycles: 2

Encoding:



Operation:

```
LCALL
(PC) ← (PC) + 3
(SP) ← (SP) + 1
((SP)) ← (PC7-0)
(SP) ← (SP) + 1
((SP)) ← (PC15-8)
(PC) ← addr15-0
```

LJMP addr16

Function: Long Jump

Description: LJMP causes an unconditional branch to the indicated address, by loading the high-order and low-order bytes of the PC (respectively) with the second and third instruction bytes. The destination may therefore be anywhere in the full 64k program memory address space. No flags are affected.

Example: The label "JMPADR" is assigned to the instruction at program memory location 1234H. The instruction,

LJMP JMPADR

at location 0123H will load the program counter with 1234H.

Bytes: 3

Cycles: 2

Encoding:

0 0 0 0	0 0 1 0	addr15-addr8	addr7-addr0

Operation: LJMP

((SP)) ← $addr_{15-8}$

MOV <dest-byte>,<src-byte>

Function: Move byte variable

Description: The byte variable indicated by the second operand is copied into the location specified by the first operand. The source byte is not affected. No other register or flag is affected.

This is by far the most flexible operation. Fifteen combinations of source and destination addressing modes are allowed.

Example: Internal RAM location 30H holds 40H. The value of RAM location 40H is 10H. The data present at input port 1 is 11001010B (0CAH). The instruction sequence,

```
MOV     R0,#30H   ;R0 < = 30H
MOV     A,@R0     ;A < = 40H
MOV     R1,A      ;R1 < = 40H
MOV     R,@R1     ;B < = 10H
MOV     @R1,P1    ;RAM (40H) < = 0CAH
MOV     P2,P1     ;P2 #0CAH
```

leaves the value 30H in register 0, 40H in both the Accumulator and register 1, 10H in register B, and 0CAH (11001010B) both in RAM location 40H and output on port 2.

MOV A,Rn

Bytes: 1

Cycles: 1

Encoding:

1 1 1 0	1 r r r

Operation: MOV

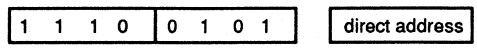
(A) ← (R_n)

***MOV A,direct**

Bytes: 2

Cycles: 1

Encoding:



Operation: MOV

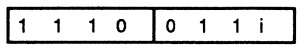
(A) ← (direct)

MOV A,@RI

Bytes: 1

Cycles: 1

Encoding:



Operation: MOV

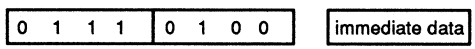
(A) ← (R_i)

MOV A,#data

Bytes: 2

Cycles: 1

Encoding:



Operation: MOV

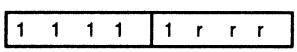
(A) ← #data

MOV Rn,A

Bytes: 1

Cycles: 1

Encoding:



Operation: MOV

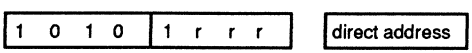
(R_n) ← (A)

MOV Rn,direct

Bytes: 2

Cycles: 2

Encoding:



Operation: MOV

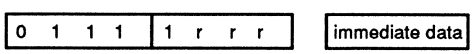
(R_n) ← (direct)

MOV Rn,#data

Bytes: 2

Cycles: 1

Encoding:



Operation: MOV

(R_n) ← #data

*MOV A,ACC is not a valid instruction.

MOV direct,A

Bytes: 2

Cycles: 1

Encoding:

1 1 1 1	0 1 0 1	direct address

Operation: MOV
(direct) ← (A)

MOV direct,Rn

Bytes: 2

Cycles: 2

Encoding:

1 0 0 0	1 r r r	direct address

Operation: MOV
(direct) ← (R_n)

MOV direct,direct

Bytes: 3

Cycles: 2

Encoding:

1 0 0 0	0 1 0 1	dir. addr. (src)	dir. addr. (dest)

Operation: MOV
(direct) ← (direct)

MOV direct,@Ri

Bytes: 2

Cycles: 2

Encoding:

1 0 0 0	0 1 1 i	direct address

Operation: MOV
(direct) ← (R_i)

MOV direct,#data

Bytes: 3

Cycles: 2

Encoding:

0 1 1 1	0 1 0 1	direct address	immediate data

Operation: MOV
(direct) ← #data

MOV @Ri,A

Bytes: 1

Cycles: 1

Encoding:

1 1 1 1	0 1 1 i

Operation: MOV
((R_i)) ← (A)

MOV @Ri,direct

Bytes: 2

Cycles: 2

Encoding:

1 0 1 0	0 1 1 i	direct address

Operation: MOV

$((R_i)) \leftarrow$ (direct)

MOV @Ri,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 1 1	0 1 1 i	immediate data

Operation: MOV

$((R_i)) \leftarrow$ #data

MOV <dest-bit>,<src-bit>

Function: Move bit data

Description: The Boolean variable indicated by the second operand is copied into the location specified by the first operand. One of the operands must be the carry flag; the other may be any directly addressable bit. No other register or flag is affected.

Example: The carry flag is originally set. The data present at input Port 3 is 11000101B. The data previously written to output Port 1 is 35H (00110101B). The instruction sequence,

```
MOV  P1.3,C
MOV  C,P3.3
MOV  P1.2,C
```

will leave the carry cleared and change Port 1 to 39H (00111001B).

MOV C,bit

Bytes: 2

Cycles: 1

Encoding:

1 0 1 0	0 0 1 0	bit address

Operation: MOV

(C) $\leftarrow$ (bit)

MOV bit,C

Bytes: 2

Cycles: 2

Encoding:

1 0 0 1	0 0 1 0	bit address

Operation: MOV

(bit) $\leftarrow$ (C)

MOV DPTR,#data16

Function: Load Data Pointer with a 16-bit constant

Description: The Data Pointer is loaded with the 16-bit constant indicated. The 16-bit constant is loaded into the second and third bytes of the instruction. The second byte (DPH) is the high-order byte, while the third byte (DPL) holds the low-order byte. No flags are affected.

This is the only instruction which moves 16 bits of data at once.

Example: The instruction,

MOV DPTR,#1234H

will load the value 1234H into the Data Pointer: DPH will hold 12H and DPL will hold 34H.

Bytes: 3

Cycles: 2

Encoding:

1 0 0 1	0 0 0 0	immed. data15-8	immed. data7-0

Operation: MOV

(DPTR) ← (#data$_{15-0}$)

DPH □ DPL ← #data$_{15-8}$ □ #data$_{7-0}$

MOVC A,@A+<base-reg>

Function: Move Code byte

Description: The MOVC instructions load the Accumulator with a code byte, or constant from program memory. The address of the byte fetched is the sum of the original unsigned eight-bit Accumulator contents and the contents of a sixteen-bit base register, which may be either the Data Pointer or the PC. In the latter case, the PC is incremented to the address of the following instruction before being added with the Accumulator; otherwise the base register is not altered. Sixteen-bit addition is performed so a carry-out from the low-order eight bits may propagate through higher-order bits. No flags are affected.

Example: A value between 0 and 3 is in the Accumulator. The following instructions will translate the value in the Accumulator to one of four values defined by the DB (define byte) directive:

```
REL_PC:  INC   A
         MOVC  A,@A+PC
         RET
         DB    66H
         DB    77H
         DB    88H
         DB    99H
```

If the subroutine is called with the Accumulator equal to 01H, it will return with 77H in the Accumulator. The INC A before the MOVC instruction is needed to "get around" the RET instruction above the table. If several bytes of code separated the MOVC from the table, the corresponding number would be added to the Accumulator instead.

MOVC A,@A+DPTR

Bytes: 1

Cycles: 2

Encoding:

1 0 0 1	0 0 1 1

Operation: MOVC

(A) ← ((A) + (DPTR))

MOVC A,@A+DPTR

Bytes: 1

Cycles: 2

Encoding:

1 0 0 0	0 0 1 1

Operation: MOVC

(PC) ← (PC) + 1

(A) ← ((A) + (PC))

MOVX <dest-byte>,<src-byte>

Function: Move External

Description: The MOVX instructions transfer data between the Accumulator and a byte of external data memory, hence the "X" appended to MOV. There are two types of instructions, differing in whether they provide an eight-bit or sixteen-bit indirect address to the external data RAM.

In the first type, the contents of R0 or R1 in the current register bank provide an eight-bit address multiplexed with data on P0. Eight bits are sufficient for external I/O expansion decoding or for a relatively small RAM array. For somewhat larger arrays, port pins can be used to output higher-order address bits. These pins would be controlled by an output instruction preceding the MOVX.

In the second type of MOVX instruction, The Data Pointer generates a sixteen-bit address. P2 outputs the high-order eight address bits (the contents of DPH) while P0 multiplexes the low-order eight bits (DPL) with data. The P2 Special Function Register retains its previous contents while the P2 output buffers are emitting the contents of DPH. This form is faster and more efficient when accessing very large data arrays (up to 64k bytes), since no additional instructions are needed to set up the output ports.

It is possible in some situations to mix the two MOVX types. A large RAM array with its high-order address lines driven by P2 can be addressed via the Data Pointer, or with code to output high-order address bits to P2 followed by a MOVX instruction using R0 or R1.

Example: An external 256 byte RAM using multiplexed address/data lines is connected to the 8051 Port 0. Port 3 provides control lines for the external RAM. Ports 1 and 2 are used for normal I/O. Registers 0 and 1 contain 12H and 34H. Location 34H of the external RAM holds the value 56H. The instruction sequence,

```
MOVX    A,@R1
MOVX    @R0,A
```

copies the value 56H into both the Accumulator and external RAM location 12H.

MOVX A,@Ri

Bytes: 1

Cycles: 2

Encoding:

1 1 1 0	0 0 1 i

Operation: MOVX

(A) ← ((R_i))

MOVX A,@DPTR

Bytes: 1

Cycles: 2

Encoding:

1 1 1 0	0 0 0 0

Operation: MOVX

(A) ← ((DPTR))

MOVX @RI,A

Bytes: 1

Cycles: 2

Encoding:

1	1	1	1	0	0	1	i

Operation: MOVX

$((R_i)) \leftarrow (A)$

MOVX @DPTR,A

Bytes: 1

Cycles: 2

Encoding:

1	1	1	1	0	0	0	0

Operation: MOVX

((DPTR)) ← (A)

MUL AB

Function: Multiply

Description: MUL AB multiplies the unsigned eight-bit integers in the Accumulator and register B. The low-order byte of the sixteen-bit product is left in the Accumulator, and the high-order byte in B. If the product is greater than 255 (0FFH) the overflow flag is set; otherwise it is cleared. The carry flag is always cleared.

Example: Originally the Accumulator holds the value 80 (50H). Register B holds the value 160 (0A0H).The instruction,

MUL AB

will give the product 12,800 (3200H), so B is changed to 32H (00110010B) and the Accumulator is cleared. The overflow flag is set, carry is cleared.

Bytes: 1

Cycles: 4

Encoding:

1	0	1	0	0	1	0	0

Operation: MUL

$(A)_{7\text{-}0} \leftarrow (A) \times (B)$

$(B)_{15\text{-}8}$

NOP

Function: No Operation

Description: Execution continues at the following instruction. Other than the PC, no registers or flags are affected.

Example: It is desired to produce a low-going output pulse on bit 7 of Port 2 lasting exactly 5 cycles. A simple SETB/CLR sequence would generate a one-cycle pulse, so four additional cycles must be inserted. This may be done (assuming are enabled) with the instruction sequence,

```
CLR    P2.7
NOP
NOP
NOP
NOP
SETB   P2.7
```

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	0 0 0 0

Operation: NOP

(PC) ← (PC) + 1

ORL <dest-byte>,<src-byte>

Function: Logical-OR for byte variables

Description: ORL performs the bitwise logical-OR operation between the indicated variables, storing the results in the destination byte. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: If the Accumulator holds 0C3H (11000011B) and R0 holds 55H (01010101B) then the instruction,

```
ORL    A,R0
```

will leave the Accumulator holding the value 0D7H (11010111B).

When the destination is a directly addressed byte, the instruction can set combinations of bits in any RAM location or hardware register. The pattern of bits to be set is determined by a mask byte, which may be either a constant data value in the instruction or a variable computed in the Accumulator at run-time. The instruction,

```
ORL    P1,#00110010B
```

will set bits 5, 4, and 1 of output Port 1.

ORL A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 1 0 0	1 r r r

Operation: ORL

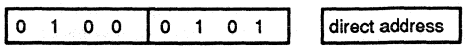
(A) ← (A) ∨ (R_n)

ORL A,direct

Bytes: 2

Cycles: 1

Encoding:



Operation: ORL

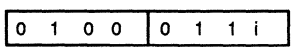
(A) ← (A) ∨ (direct)

ORL A,@Ri

Bytes: 1

Cycles: 1

Encoding:



Operation: ORL

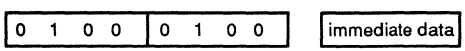
(A) ← (A) ∨ ((Ri))

ORL A,#data

Bytes: 2

Cycles: 1

Encoding:



Operation: ORL

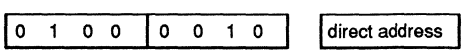
(A) ← (A) ∨ #data

ORL direct,A

Bytes: 2

Cycles: 1

Encoding:



Operation: ORL

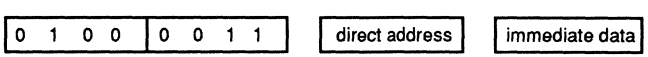
(direct) ← (direct) ∨ (A)

ORL direct,#data

Bytes: 3

Cycles: 2

Encoding:



Operation: ORL

(direct) ← (direct) ∨ #data

ORL C,<src-bit>

Function: Logical-OR for bit variables

Description: Set the carry flag if the Boolean value is a logical 1; leave the carry in its current state otherwise. A slash ("/") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, but the source bit itself is not affected. No other flags are affected.

Example: Set the carry flag if and only if P1.0 = 1, ACC.7 = 1, or OV = 0:

```
ORL   C,P1.0     ;LOAD CARRY WITH INPUT PIN P10
ORL   C,ACC.7    ;OR CARRY WITH THE ACC. BIT 7
ORL   C,/OV      ;OR CARRY WITH THE INVERSE OF OV.
```

ORL C,bit

Bytes: 2

Cycles: 2

Encoding:

0 1 1 1	0 0 1 0	bit address

Operation: ORL

(C) ← (C) ∨ (bit)

ORL C,/bit

Bytes: 2

Cycles: 2

Encoding:

1 0 1 0	0 0 0 0	bit address

Operation: ORL

(C) ← (C) ∨ ($\overline{\text{bit}}$)

POP direct

Function: Pop from stack

Description: The contents of the internal RAM location addressed by the Stack Pointer is read, and the Stack Pointer is decremented by one. The value read is then transferred to the directly addressed byte indicated. No flags are affected.

Example: The Stack Pointer originally contains the value 32H, and internal RAM locations 30H through 32H contain the values 20H, 23H, and 01H, respectively. The instruction sequence,

```
POP   DPH
POP   DPL
```

will leave the Stack Pointer equal to the value 30H and the Data Pointer set to 0123H. At this point the instruction,

```
POP   SP
```

will leave the Stack Pointer set to 20H. Note that in this special case the Stack Pointer was decremented to 2FH before being loaded with the value popped (20H).

Bytes: 2

Cycles: 2

Encoding:

1 1 0 1	0 0 0 0	direct address

Operation:

POP
(direct) ← ((SP))
(SP) ← (SP) – 1

PUSH direct

Function: Push onto stack

Description: The Stack Pointer is incremented by one. The contents of the indicated variable is then copied into the internal RAM location addressed by the Stack Pointer. Otherwise no flags are affected.

Example: On entering an interrupt routine the Stack Pointer contains 09H. The Data Pointer holds the value 0123H. The instruction sequence,

```
PUSH   DPL
PUSH   DPH
```

will leave the Stack Pointer set to 0BH and store 23H and 01H in internal RAM locations 0AH and 0BH, respectively.

Bytes: 2

Cycles: 2

Encoding:

1 1 0 0	0 0 0 0	direct address

Operation:

PUSH
(SP) ← (SP) + 1
((SP)) ← (direct)

RET

Function: Return from subroutine

Description: RET pops the high- and low-order bytes of the PC successively from the stack, decrementing the Stack Pointer by two. Program execution continues at the resulting address, generally the instruction immediately following an ACALL or LCALL. No flags are affected.

Example: The Stack Pointer originally contains the value 0BH. Internal RAM locations 0AH and 0BH contain the values 23H and 01H, respectively. The instruction,

RET

will leave the Stack Pointer equal to the value 09H. Program execution will continue at location 0123H.

Bytes: 1

Cycles: 2

Encoding:

0 0 1 0	0 0 1 0

Operation: RET

$(PC_{15-8}) \leftarrow ((SP))$

$(SP) \leftarrow (SP) - 1$

$(PC_{7-0}) \leftarrow ((SP))$

$(SP) \leftarrow (SP) - 1$

RETI

Function: Return from interrupt

Description: RETI pops the high- and low-order bytes of the PC successively from the stack, and restores the interrupt logic to accept additional interrupts at the same priority level as the one just processed. The Stack Pointer is left decremented by two. No other registers are affected; the PSW is not automatically restored to its pre-interrupt status. Program execution continues at the resulting address, which is generally the instruction immediately after the point at which the interrupt request was detected. If a lower- or same-level interrupt has been pending when the RETI instruction is executed, that one instruction will be executed before the pending interrupt is processed.

Example: The Stack Pointer originally contains the value 0BH. An interrupt was detected during the instruction ending at location 0122H. Internal RAM locations 0AH and 0BH contain the values 23H and 01H, respectively. The instruction,

RETI

will leave the Stack Pointer equal to 09H and return program execution to location 0123H.

Bytes: 1

Cycles: 2

Encoding:

0 0 1 1	0 0 1 0

Operation: RETI

$(PC_{15-8}) \leftarrow ((SP))$

$(SP) \leftarrow (SP) - 1$

$(PC_{7-0}) \leftarrow ((SP))$

$(SP) \leftarrow (SP) - 1$

RL A

Function: Rotate Accumulator Left

Description: The eight bits in the Accumulator are rotated one bit to the left. Bit 7 is rotated into the bit 0 position. No flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B). The instruction,

RL A

leaves the Accumulator holding the value 8BH (10001011B) with the carry unaffected.

Bytes: 1

Cycles: 1

Encoding:

0	0	1	0	0	0	1	1

Operation: RL

$(A_{n+1}) \leftarrow (A_n)$, n = 0 – 6

(A0) ← (A7)

RLC A

Function: Rotate Accumulator Left through the Carry flag

Description: The eight bits in the Accumulator and the carry flag are together rotated one bit to the left. Bit 7 moves into the carry flag; the original state of the carry flag moves into the bit 0 position. No other flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B), and the carry is zero. The instruction,

RLC A

leaves the Accumulator holding the value 8BH (10001010B) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0	0	1	1	0	0	1	1

Operation: RLC

$(A_{n+1}) \leftarrow (A_n)$, n = 0 – 6

(A0) ← (C)

(C) ← (A7)

RR A

Function: Rotate Accumulator Right

Description: The eight bits in the Accumulator are rotated one bit to the right. Bit 0 is rotated into the bit 7 position. No flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B). The instruction,

RR A

leaves the Accumulator holding the value 0E2H (11100010B) with the carry unaffected.

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0	0	0	1	1

Operation: RR

$(A_n) \leftarrow (A_{n+1})$, n = 0 – 6

(A7) ← (A0)

RRC A

Function: Rotate Accumulator Right through the Carry flag

Description: The eight bits in the Accumulator and the carry flag are together rotated one bit to the right. Bit 0 moves into the carry flag; the original state of the carry flag moves into the bit 7 position. No other flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B), and the carry is zero. The instruction,

RRC A

leaves the Accumulator holding the value 62 (01100010B) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0	0	0	1	0	0	1	1

Operation: RRC

$(A_n) \leftarrow (A_{n+1})$, n = 0 – 6

(A7) ← (C)

(C) ← (A0)

SETB <bit>

Function: Set Bit

Description: SETB sets the indicated bit to one. SETB can operate on the carry flag or any directly addressable bit. No other flags are affected.

Example: The carry flag is cleared. Output Port 1 has been written with the value 34H (00110100B). The instructions,

```
SETB  C
SETB  P1.0
```

will leave the carry flag set to 1 and change the data output on Port 1 to 35H (00110101B).

SETB C

Bytes: 1

Cycles: 1

Encoding:

1 1 0 1	0 0 1 1

Operation:
SETB
(C) ← 1

SETB bit

Bytes: 2

Cycles: 1

Encoding:

1 1 0 1	0 0 1 0	bit address

Operation:
SETB
(bit) ← 1

SJMP rel

Function: Short Jump

Description: Program control branches unconditionally to the address indicated. The branch destination is computed by adding the signed displacement in the second instruction byte to the PC, after incrementing the PC twice. Therefore, the range of destinations allowed is from 128 bytes preceding this instruction to 127 bytes following it.

Example: The label "RELADR" is assigned to an instruction at program memory location 0123H. The instruction,

```
SJMP  RELADR
```

will assemble into location 0100H. After the instruction is executed, the PC will contain the value 0123H.

(*Note:* Under the above conditions the instruction following SJMP will be at 102H. Therefore, the displacement byte of the instruction will be the relative offset (0123H-0102H) = 21H. Put another way, an SJMP with a displacement of 0FEH would be a one-instruction infinite loop.)

Bytes: 2

Cycles: 2

Encoding:

1 0 0 0	0 0 0 0	rel. address

Operation:
SJMP
(PC) ← (PC) + 2
(PC) ← (PC) + rel

SUBB A, <src-byte>

Function: Subtract with borrow

Description: SUBB subtracts the indicated variable and the carry flag together from the Accumulator, leaving the result in the Accumulator. SUBB sets the carry (borrow) flag if a borrow is needed for bit 7, and clears C otherwise. (If C was set *before* executing a SUBB instruction, this indicates that a borrow was needed for the previous step in a multiple precision subtraction, so the carry is subtracted from the Accumulator along with the source operand.) AC is set if a borrow is needed for bit 3, and cleared otherwise. OV is set if a borrow is needed into bit 6, but not into bit 7, or into bit 7, but not bit 6.

When subtracting signed integers OV indicates a negative number produced when a negative value is subtracted from a positive value, or a positive result when a positive number is subtracted from a negative number.

The source operand allows four addressing modes: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C9H (11001001B), register 2 holds 54H (01010100B), and the carry flag is set. The instruction,

SUBB A,R2

will leave the value 74H (01110100B) in the Accumulator, with the carry flag and AC cleared but OV set.

Notice that 0C9H minus 54H is 75H The difference between this and the above result is due to the carry (borrow) flag being set before the operation. If the state of the carry is not known before starting a single or multiple-precision subtraction, it should be explicitly cleared by a CLR C instruction

SUBB A,Rn

Bytes: 1

Cycles: 1

Encoding:

1 0 0 1	1 r r r

Operation: SUBB

(A) ← (A) – (C) – (R_n)

SUBB A,direct

Bytes: 2

Cycles: 1

Encoding:

1 0 0 1	0 1 0 1	direct address

Operation: SUBB

(A) ← (A) – (C) – (direct)

SUBB A,@Ri

Bytes: 1

Cycles: 1

Encoding:

1 0 0 1	0 1 1 i

Operation: SUBB

(A) ← (A) – (C) – (R_i)

SUBB A,@Ri

Bytes: 1

Cycles: 1

Encoding:

1 0 0 1	0 1 0 0	immediate data

Operation: SUBB

$(A) \leftarrow (A) - (C) - (R_i)$

SWAP A

Function: Swap nibbles within the Accumulator

Description: SWAP A interchanges the low- and high-order nibbles (four-bit fields) of the Accumulator (bits 3-0 and bits 7-4). The operation can also be thought of as a four-bit rotate instruction. No flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B). The instruction,

SWAP A

leaves the Accumulator holding the value 5CH (01011100B).

Bytes: 1

Cycles: 1

Encoding:

1 1 0 0	0 1 0 0

Operation: SWAP

$(A_{3-0}) \rightleftarrows (A_{7-4})$

XCH A,<byte>

Function: Exchange Accumulator with byte variable

Description: XCH loads the Accumulator with the contents of the indicated variable, at the same time writing the original Accumulator contents to the indicated variable. The source/destination operand can use register, direct, or register-indirect addressing.

Example: R0 contains the address 20H. The Accumulator holds the value 3FH (00111111B). Internal RAM location 20H holds the value 75H (01110101B). The instruction,

XCH A,@R0

will leave the RAM location 20H holding the values 3FH (00111111B) and 75H (01110101B) in the Accumulator.

XCH A,Rn

Bytes: 1

Cycles: 1

Encoding:

1 1 0 0	1 r r r

Operation: XCH

$(A) \rightleftarrows (R_n)$

XCH A,direct

Bytes: 2

Cycles: 1

Encoding:

1 1 0 0	0 1 0 1	direct address

Operation: XCH

$(A) \rightleftarrows$ (direct)

XCH A,@Ri

Bytes: 1

Cycles: 1

Encoding:

1 1 0 0	0 1 1 i

Operation: XCH

(A) $\rightleftarrows$ ((Ri))

XCHD A,@Ri

Function: Exchange Digit

Description: XCHD exchanges the low-order nibble of the Accumulator (bits 3-0), generally representing a hexadecimal or BCD digit, with that of the internal RAM location indirectly addressed by the specified register. The high-order nibbles (bits 7-4) of each register are not affected. No flags are affected.

Example: R0 contains the address 20H. The Accumulator holds the value 36H (00110110B). Internal RAM location 20H holds the value 75H (01110101B). The instruction,

XCHD A,@R0

will leave RAM location 20H holding the value 76H (01110110B) and 35H (00110101B) in the Accumulator.

Bytes: 1

Cycles: 1

Encoding:

1 1 0 1	0 1 1 i

Operation: XCHD

(A_{3-0}) $\rightleftarrows$ ((Ri_{3-0}))

XRL <dest-byte>,<src-byte>

Function: Logical Exclusive-OR for byte variables

Description: XRL performs the bitwise logical Exclusive-OR operation between the indicated variables, storing the results in the destination. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

(*Note:* When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.)

Example: If the Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) then the instruction,

XRL A,R0

will leave the Accumulator holding the value 69H (01101001B).

When the destination is a directly addressed byte, this instruction can complement combinations of bits in any RAM location or hardware register. The pattern of bits to be complemented is then determined by a mask byte, either a constant contained in the instruction or a variable computed in the Accumulator at run-time. The instruction,

XRL P1,#00110001B

will complement bits 5, 4, and 0 of output Port 1.

XRL A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 1 1 0	1 r r r

Operation: XRL

(A) ← (A) ⊻ (R_n)

XRL A,direct

Bytes: 2

Cycles: 1

Encoding:

0 1 1 0	0 1 0 1	direct address

Operation: XRL

(A) ← (A) ⊻ (direct)

XRL A,@RI

Bytes: 1

Cycles: 1

Encoding:

0 1 1 0	0 1 1 i

Operation: XRL

(A) ← (A) ⊻ (R_i)

XRL A,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 1 0	0 1 0 0	immediate data

Operation: XRL

(A) ← (A) ⊻ #data

XRL direct,A

Bytes: 2

Cycles: 1

Encoding:

0 1 1 0	0 0 1 0	direct address

Operation: XRL

(direct) ← (direct) ⊻ (A)

XRL direct,#data

Bytes: 3

Cycles: 2

Encoding:

0 1 1 0	0 0 1 1	direct address	immediate data

Operation: XRL

(direct) ← (direct) ⊻ #data

EPROM PRODUCTS

Most of the 80C51 derivative products offered by Philips are supported with an EPROM version. Currently available EPROM parts are the 87C51, 87C451, 87C552, 87C52, 87C752, and the 87C751. EPROM versions of the 87C550, 87C652, 87C654, and 87C528 are now in development.

All EPROM products are available in both windowed DIP and OTP package configurations. The windowed DIP package allows the EPROM to be erased under a strong UV light source, making program development easier and faster. The OTP (One Time Programmable) version cannot be erased because there is no window through which the die could be exposed to UV light. While the EPROM can only be programmed once in the OTP package, the part costs less than in windowed DIP and therefore offers an advantage for those not desiring to use the masked ROM version of the part.

The EPROM products are fully supported on the industry standard EPROM programmers.

Programming the 87C51

The setup for programming the microcontroller is shown in Figure 59. Note that the part is running with a 4 to 6 MHz oscillator. The clock must be running because the device is executing internal address and program data transfers during the programming.

To program the 87C51, the address of the EPROM location to be programmed is applied to ports 1 and 2 as shown in Figure 59. The code byte to be programmed into this location is applied to port 0. RST, $\overline{PSEN}$, and the pins of ports 2 and 3 specified in Table 20 are held at the "Program Code Data" levels specified in the table. The ALE/$\overline{PROG}$ is then pulsed low 25 times to program the addressed location.

Encryption Table

The encryption table is a feature of the 87C51, and its derivatives, that protects the code from being easily read by anyone other than the programmer. The encryption table is 16 to 64 bytes of code, depending on the microcontroller, that are exclusive NORed with the program code data as it is read out. The first byte is XNORed with the first location read, the second with the second read, etc. through the sixteenth byte read. The seventeenth byte is XNORed with the first byte of the encryption table, the eighteenth with the second, etc. and on in sixteen-byte groups.

After the Encryption table has been programmed the user has to know its contents in order to correctly decode the program code data. The encryption table itself cannot be read out.

The encryption table is programmed in the same manner as the program memory, but using the "Pgm Encryption Table" levels specified in Table 20. After the encryption table is programmed, verification cycles will produce only encrypted information.

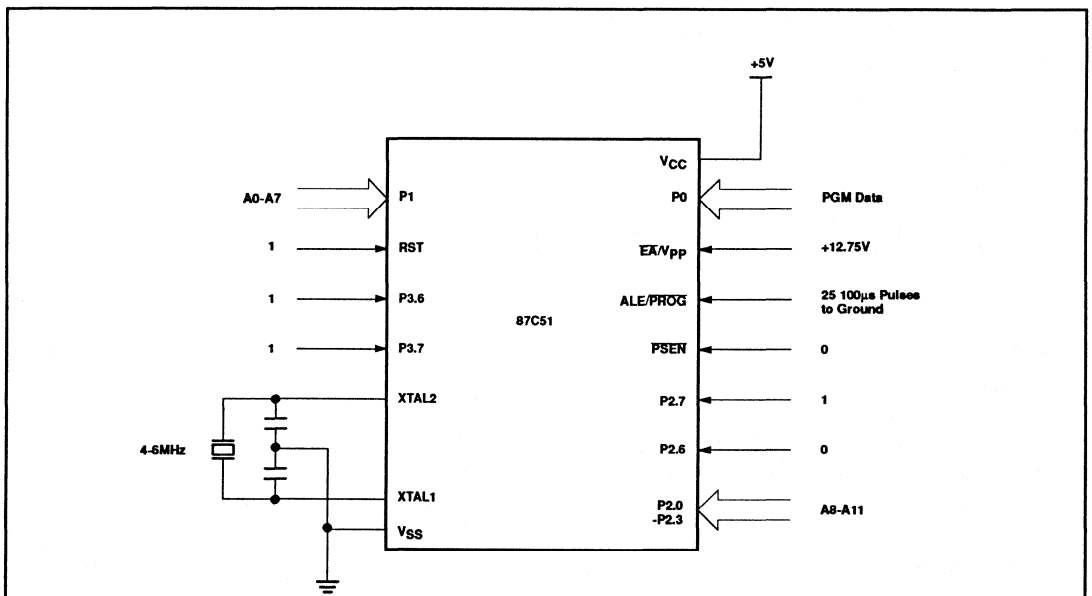


Figure 59. Programming Configuration

Table 20. EPROM Programming Modes

MODE	RST	PSEN	ALE/PROG	EA/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm lock bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm lock bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:
1. "0" = valid low for that pin, "1" = valid high for that pin.
2. V_{PP} = 12.75 ±0.25V.
3. V_{CC} = 5V ±10% during programming and verification.

* ALE/PROG receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100ms (±10µs) and high for a minimum of 10µs.

Lock Bit

There are two lock bits on the 87C51 that, when set, prevent the program data memory from being read out or programmed further. To program the lock bits, repeat the programming sequence using the "Pgm Lock Bit" levels specified in Table 20.

After the first lock bit is programmed, further programming of the code memory or the encryption table is disabled. The other lock bit can of course still be programmed. With only lock bit one programmed, the memory can still be read out for program verification. After the second lock bit is programmed, it is no longer possible to read out (verify) the program memory.

Program Verification

If lock bit 2 has not been programmed the on-chip program memory can be read out for program verification. To verify the contents of the program memory, the address of the location to be read is applied to ports 1 and 2 as shown in Figure 60. The other pins are held at the "Verify Code Data" levels indicated in Table 20. The contents of the addressed location will appear on port 0. For this operation external pull-ups are required on port 0 as shown in Figure 60. Note that if the encryption table has been programmed the data presented at port 0 will be the exclusive NOR of the program byte with a byte from the encryption table.

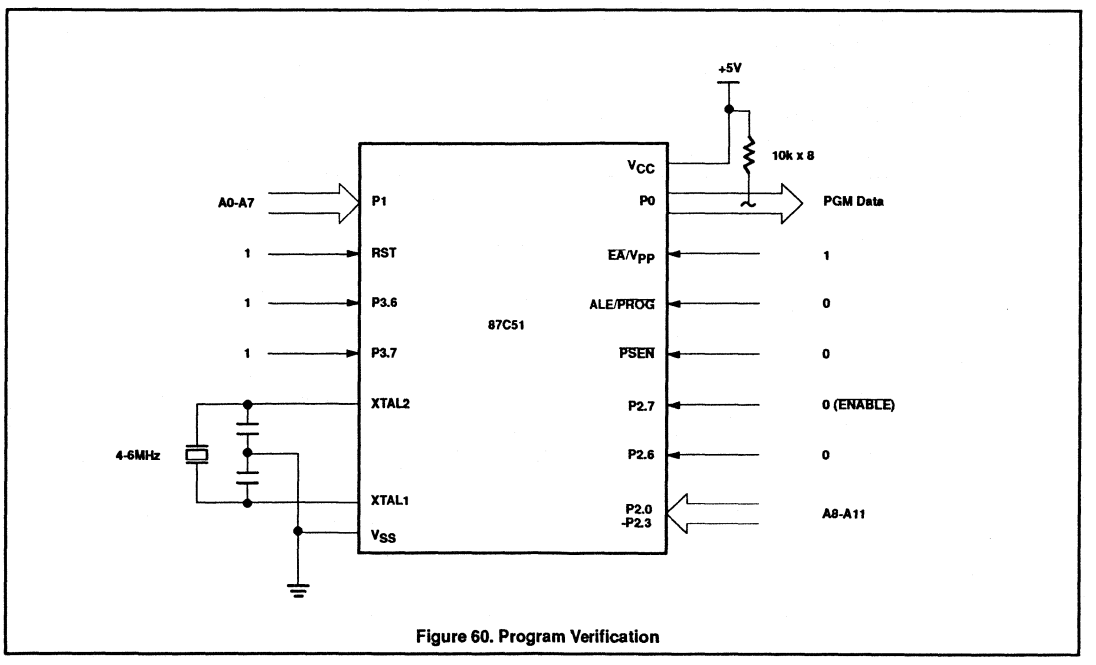


Figure 60. Program Verification

Signature Bytes
The 87C51 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C51 manufactured by Philips.

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates the part is made by Philips

(031H) =	90H 87C451	97H 87C52
	92H 87C51	99H 87C654
	93H 87C652	9AH 80C52
	94H 87C552	9BH 87C528
	96H 87C550	9CH 87C592

EPROM Erasure

Erasure of the EPROM occurs when the chip is exposed to light with wavelengths shorter than 4000 angstroms. Sunlight and fluorescent lighting have wavelengths in this range, so exposure to these light sources over an extended period of time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. It is recommended, for this reason, that an opaque label be placed over the window. If the part is subject to elevated temperatures or an environment where solvents are used, Kapton tape (Fluorglas part number 2345-5 or its equivalent) can be used.

The recommended erasure procedure is to expose the chip to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000µW/cm^2 rating for 20 to 40 minutes, at a distance of 1 inch, is adequate.

Programming the 87C751 and 87C752

The 87C751 and 87C752 are programmed using a Quick-pulse programming algorithm that is similar to that used for the 87C51. It differs from the 87C51 in that a serial data stream is used to place the 87C751 in the programming mode.

Figure 61 shows a block diagram of the programming configuration for the 87C751. Port pin P0.2 is used for the programming voltage supply input (V_{PP} signal). Port pin P0.1 is used for the program (PGM) signal.

Port 3 accepts the address input for the EPROM location to be programmed. Both the high and low components of the eleven-bit address are presented to the part through port 3. Multiplexing of the address components is performed using ASEL (P0.0).

Port 1 is used as a bidirectional data bus during programming and verify operations. During the programming mode, it accepts the byte to be programmed. In the verify mode, it returns the contents of the specified address location.

The X1 pin is the oscillator input and receives the master system clock. This clock should be between 1.2 and 6MHz.

The RESET pin is used to accept the serial data stream that places the 87C751 into various programming modes. This pattern consists of a 10-bit code with the LSB send first. Each bit is synchronized to the clock input X1.

To program the 87C751 the part must be put into the programming mode by presenting the proper serial code (see Table 21) to the RESET pin. To do this RESET should be held high for at least two machine cycles. Port pins P0.1 and P0.2 will be at VOH as a result of this, but they must be driven high prior to sending the serial data stream on the RESET pin. The serial data bits can now be transmitted over the RESET pin placing the 87C751 into one of the programming modes. Following the transmission of the last data bit, the reset pin should be held low.

Next the address information for the location to be programmed is placed on Port 3 and ASEL is used to perform the address multiplexing. ASEL should be driven high and then Port 3 driven with the high-order address bits. ASEL is then driven low, latching the high-order bits internally. Port 3 can now be driven with the low 8 bits of the address, completing the addressing of the location to be programmed.

A high-voltage V_{PP} level is now applied to the V_{PP} input. This sets Port 1 as an input port. The data to be programmed to the EPROM array should be placed on Port 1. A series of 25 programming pulses is now applied to the PGM pin (P0.1) to program the addressed EPROM location.

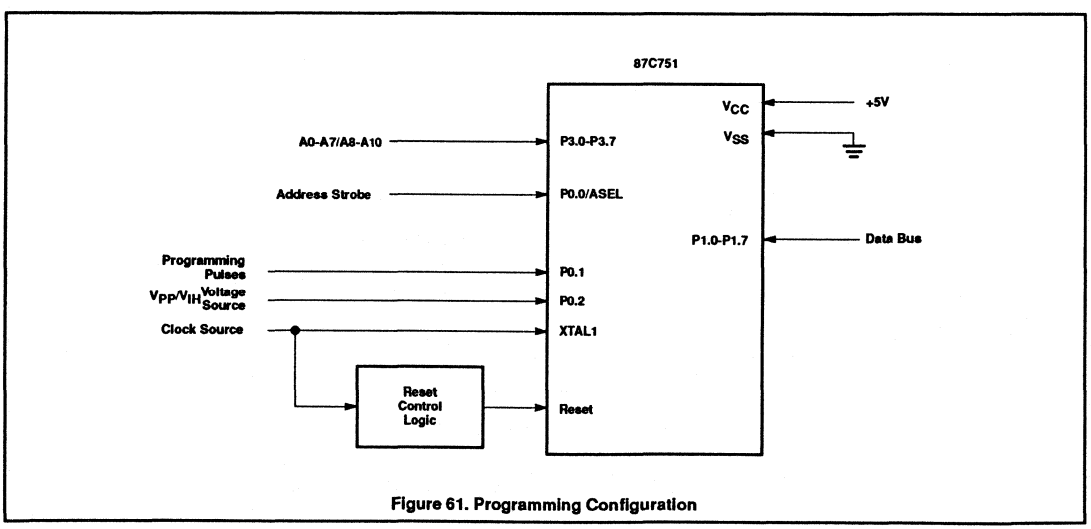


Figure 61. Programming Configuration

Table 21. Serial Codes

OPERATION	SERIAL CODE	P0.1 (PGM/)	P0.2 (V_{PP})
Program user EPROM	296H	–*	V_{PP}
Verify user EPROM	296H	V_{IH}	V_{IH}
Program key EPROM	292H	–*	V_{PP}
Verify key EPROM	292H	V_{IH}	V_{IH}
Program security bit 1	29AH	–*	V_{PP}
Program security bit 2	298H	–*	V_{PP}
Verify security bits	29AH	V_{IH}	V_{IH}
Read signature bytes	19CH	V_{IH}	V_{IH}

NOTE:
*Pulsed from V_{IH} to V_{IL} and returned to V_{IH}.

Program Verification

The EPROM array can be verified by placing the part in the programming mode as described above and forcing the V_{PP} pin to the V_{OH} level. Four machine cycles after addressing a location the contents of the addressed location will appear on Port 1.

87C751 and 87C752 Signature Bytes

The signature bytes for the 87C751 and 87C752 are read differently and are in different locations than those on the 87C51. Due to its reduced pin count, the part has to be put into "Signature Byte Read Mode" by placing a 10-bit serial data stream on the Reset pin. The proper code and the conditions of P0.1 and P0.2, for this mode, are shown in Table 21.

Once the part has been placed into the Signature Byte Read Mode, the signature bytes can be read by the same procedure as a normal verification of locations 01EH and 01FH. The values are:

01EH = 15H indicates the part is made by Philips
01FH = 91H - 87C751
01FH = 95H - 87C752

Programming Features

The 87C751 has all of the special programming features incorporated within its EPROM array that the 87C51 has. It has an encryption key table and two security bits (lock bits). These function exactly as they do in the 87C51. They are programmed or verified by sending the proper code over the RESET pin (see Table 21) and then following the 87C751 programming procedure as described previously.

Erasure Characteristics

The erasure procedure is exactly the same as that described for the 87C51.

Philips Components

Date of Issue	August 26, 1986
Status	Product Specification
Application Specific Product	

8031AH/8051AH

Single-chip 8-bit microcontroller

DESCRIPTION

The Philips 8031AH/8051AH is a high-performance microcontroller fabricated with Philips high-density highly reliable +5V, depletion-load, N-channel, silicon-gate, N500 MOS process technology. It provides the hardware features, architectural enhancements and instructions that are necessary to make it a powerful and cost-effective controller for applications requiring up to 64k bytes of program memory and/or up to 64k bytes of data storage.

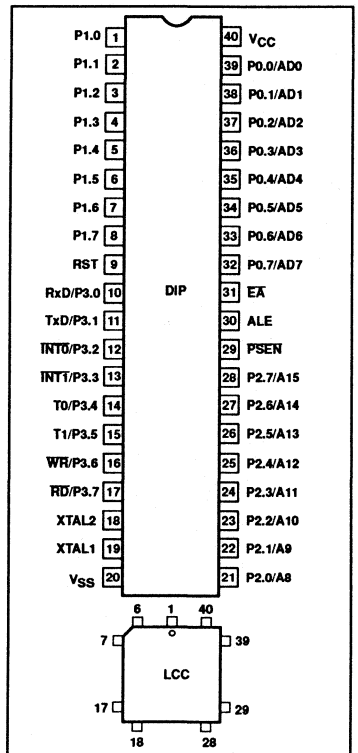
The 8051AH contains a 4k x 8 read-only program memory, a 128 x 8 read-only data memory, 32 I/O lines, two 16-bit counter/timers, a five-source, two-priority level nested interrupt structure, a serial I/O port for either multi-processor communications, I/O expansion or full duplex UART, and on-chip oscillator and clock circuits. The 8031AH is identical, except that it lacks the program memory. For systems that require extra capability, the 8051AH can be expanded using standard TTL compatible memories and byte oriented peripheral controllers.

The 8051AH microcontroller, like its 8048 predecessor, is efficient both as a controller and as an arithmetic processor. It has extensive facilities for binary and BCD arithmetic and excels in bit-handling capabilities. Efficient use of program memory results from an instruction set consisting of 44% one-byte, 41% two-byte, and 15% three-byte instructions. With a 12MHz crystal, 58% of the instructions execute in 1μs, 40% in 2μs and multiply and divide require only 4μs.

FEATURES

- Reduced supply current
- 4k X 8 ROM (8051AH)
- 128 X 8 RAM
- Four 8-bit ports, 32 I/O lines
- Two 16-bit timer/event counters
- High-performance full-duplex serial channel
- External memory expandable to 128k
- Boolean processor
- Industry standard 8051 architecture:
  - Non-paged jumps
  - Direct addressing
  - Four 8-register banks
  - Stack depth up to 128-bytes
  - Multiply, divide, subtract, compare
- Most instructions execute in 1μs
- 4μs multiply and divide

PIN CONFIGURATION

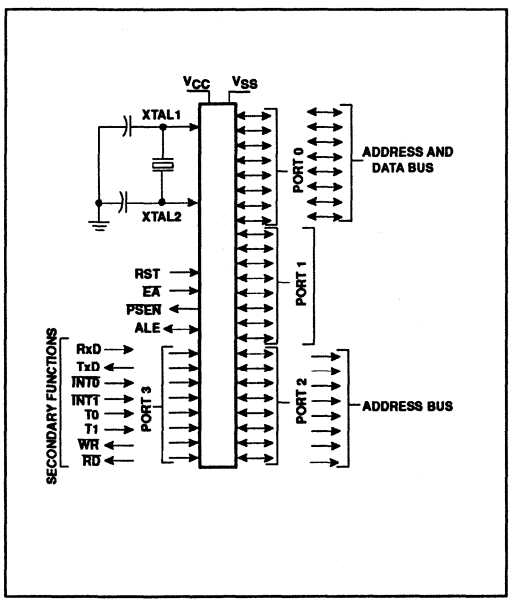


Pin	Function	Pin	Function	Pin	Function
1	NC	15	INT1/P3.3	30	P2.6/A14
2	P1.0	16	T0/P3.4	31	P2.7/A15
3	P1.1	17	T1/P3.5	32	PSEN
4	P1.2	18	WR/P3.6	33	ALE
5	P1.3	19	RD/P3.7	34	NC
6	P1.4	20	XTAL2	35	EA
7	P1.5	21	XTAL1	36	P0.7/AD7
8	P1.6	22	V_{SS}	37	P0.6/AD6
9	P1.7	23	NC	38	P0.5/AD5
10	RST	24	P2.0/A8	39	P0.4/AD4
11	RxD/P3.0	25	P2.1/A9	40	P0.3/AD3
12	NC	26	P2.2/A10	41	P0.2/AD2
13	TxD/P3.1	27	P2.3/A11	42	P0.1/AD1
14	INT0/P3.2	28	P2.4/A12	43	P0.0/AD0
		29	P2.5/A13	44	V_{CC}

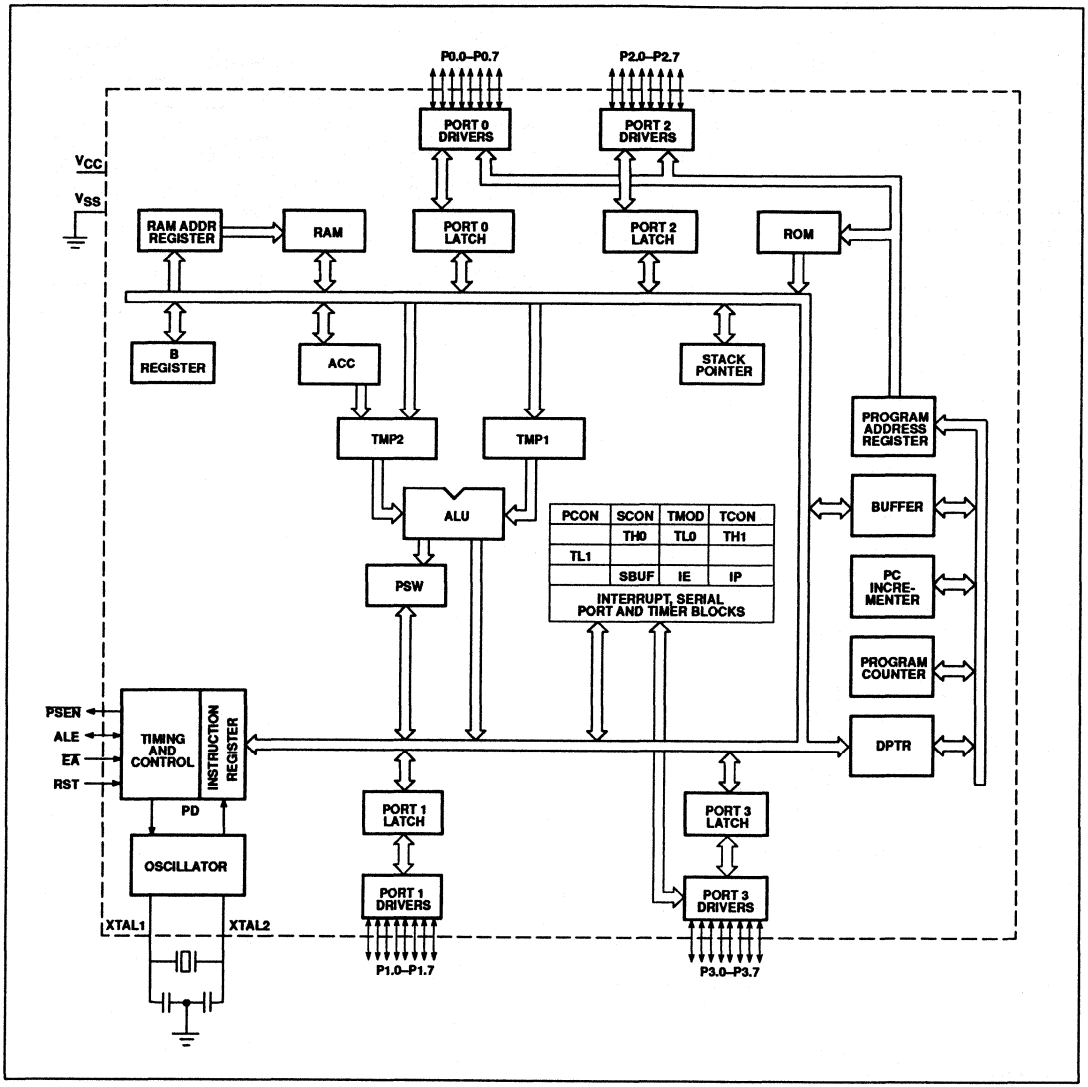
PART NUMBER SELECTION

PHILIPS		PHILIPS COMPONENTS-SIGNETICS		TEMPERATURE °C AND PACKAGE	FREQUENCY MHz
ROMless	ROM	ROMless	ROM		
MAF8031AH-2P	MAF8051AH-2P	SCN8031HACN40	SCN8051HACN40	–40 to +85, plastic DIP	12
MAB8031AH-2P	MAB8051AH-2P	SCN8031HCCN40	SCN8051HCCN40	0 to +70, plastic DIP	12
		SCN8031HCFN40	SCN8051HCFN40	0 to +70, plastic DIP	15
		SCN8031HAFN40	SCN8051HAFN40	–40 to +85, plastic DIP	15
MAB8031AH-2WP	MAB8051AH-2WP	SCN8031HCCA44	SCN8051HCCA44	0 to +70, plastic LCC	12
MAF8031AH-2WP	MAF8051AH-2WP	SCN8031HACA44	SCN8051HACA44	–40 to +85, plastic LCC	12
		SCN8031HCFA44	SCN8051HCFA44	0 to +70, plastic LCC	15
		SCN8031HAFA44	SCN8051HAFA44	–40 TO +85, plastic LCC	15

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	TYPE	NAME AND FUNCTION
V_{SS}	20	22 23	I I	**Ground:** 0V reference. **Ground:** 0V reference. (QFP only)
V_{CC}	40	44	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–46	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s.
P1.0–P1.7	1–8	2–9	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}).
P2.0–P2.7	21–28	24–31	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the 80C51 family, as listed below:
	10	11	I	**RxD (P3.0):** Serial input port
	11	13	O	**TxD (P3.1):** Serial output port
	12	14	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	I	**T0 (P3.4):** Timer 0 external input
	15	17	I	**T1 (P3.5):** Timer 1 external input
	16	18	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE	30	33	I/O	**Address Latch Enable:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory.
$\overline{PSEN}$	29	32	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$	31	35	I	**External Access Enable:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H to 0FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 0FFFH.
XTAL1	19	21	I	**Crystal 1:** Input to the inverting oscillator amplifier.
XTAL2	18	20	O	**Crystal 2:** Output from the inverting oscillator amplifier and input to the internal clock generator circuits.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

DESIGN CONSIDERATIONS

At power-on, the voltage on V_{CC} and RST should come up at the same time for a proper start-up.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Voltage on any other pin to V_{SS}	–0.5 to +7.0	V
Input, output current on any single pin	10	mA
Power dissipation	1.0	W

DC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C, V_{CC} = 5V ±10%, V_{SS} = 0V[4, 5]

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{IL}	Input low voltage		–0.5	0.8	V
V_{IH}	Input high voltage; except XTAL2, RST		2.0	V_{CC}+0.5	V
V_{IH1}	Input high voltage to RST for reset, XTAL2	XTAL1 to V_{SS}	2.5	V_{CC}+0.5	V
V_{OL}	Output low voltage; ports 1, 2, 3[6]	I_{OL} = 1.6mA		0.45	V
V_{OL1}	Output low voltage; port 0, ALE, $\overline{PSEN}$[6]	I_{OL} = 3.2mA		0.45	V
V_{OH}	Output high voltage; ports 1, 2, 3	I_{OH} = –80uA	2.4		V
V_{OH1}	Output high voltage; port 0, ALE, $\overline{PSEN}$[3]	I_{OH} = –400uA	2.4		V
I_{IL}	Logical 0 input current; ports 1, 2, 3	V_{IN} = 0.45V		–500	µA
I_{IH1}	Input high current to RST for reset	V_{IN} < V_{CC} –1.5V		500	µA
I_{LI}	Input leakage current; port 0, $\overline{EA}$	0.45 < V_{IN} < V_{CC}		±10	uA
I_{IL2}	Logical 0 input current for XTAL2	XTAL1 = V_{SS}, V_{IN} = 0.45V		–3.2	mA
I_{CC}	Power supply current	All outputs disconnected and $\overline{EA}$ = V_{CC}		125	mA
C_{IO}	Pin capacitance			10	pF

T_A = –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{IH}	Input high voltage; except XTAL2, RST		2.1	V_{CC}+0.5	V
V_{IH1}	Input high voltage to RST and XTAL2	XTAL1 = V_{SS}	2.6	V_{CC}+0.5	V
I_{IL2}	Logical 0 input current for XTAL2	XTAL1 = V_{SS}, V_{IN} = 0.45V		–4.0	mA
I_{CC}	Power supply current	All outputs disconnected and $\overline{EA}$ = V_{CC}		135	mA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.
5. All voltage measurements are referenced to ground. For testing, all input signals swing between 0.45V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and at output voltages of 0.8V and 2.0V as appropriate.
6. V_{OL} is derated when the device rapidly discharges external capacitance. This AC noise is most pronounced during emission of address data. When using external memory, locate the latch or buffer as close as possible to the device.

Datum	Emitting Ports	Degraded I/O Lines	V_{OL} (Peak Max)
Address	P2, P0	P1, P3	0.8V
Write Data	P0	P1, p3, ALE	0.8V

7. C_L = 100pF for port 0, ALE and $\overline{PSEN}$ outputs: C_L = 80pF for all other ports.

AC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or −40°C to +85°C, V_{CC} = 5V ±20%, V_{SS} = 0V[1, 2]

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK[3]		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$		Oscillator frequency: **Speed Versions** SCN8051/31 C MAB8051/31 −2 MAF8051/31 −2 SCN8051/31 F			 3.5 3.5 3.5 3.5	 12 12 12 15	 MHz MHz MHz MHz
t_{LHLL}	1	ALE pulse width	127		$2t_{CLCL}-40$		ns
t_{AVLL}	1	Address valid to ALE low	43		$t_{CLCL}-40$		ns
t_{LLAX}	1	Address hold after ALE low	48		$t_{CLCL}-35$		ns
t_{LLIV}	1	ALE low to valid instruction in		233		$4t_{CLCL}-100$	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	58		$t_{CLCL}-25$		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	215		$3t_{CLCL}-35$		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		125		$3t_{CLCL}-125$	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		63		$t_{CLCL}-20$	ns
t_{AVIV}	1	Address to valid instruction in		302		$5t_{CLCL}-115$	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		20		20	ns
t_{PXAV}	1	$\overline{PSEN}$ to address valid	75		$t_{CLCL}-8$		ns
Data Memory							
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		252		$5t_{CLCL}-165$	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		97		$2t_{CLCL}-70$	ns
t_{LLDV}	2, 3	ALE low to valid data in		517		$8t_{CLCL}-150$	ns
t_{AVDV}	2, 3	Address to valid data in		585		$9t_{CLCL}-165$	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}-130$		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	23		$t_{CLCL}-60$		ns
t_{QVWH}	2, 3	Data valid to $\overline{WR}$ high	433		$7t_{CLCL}-150$		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	33		$t_{CLCL}-50$		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		20		20	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock							
t_{CHCX}	5	High time	20		20		ns
t_{CLCX}	5	Low time	20		20		ns
t_{CLCH}	5	Rise time		20		20	ns
t_{CHCL}	5	Fall time		20		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

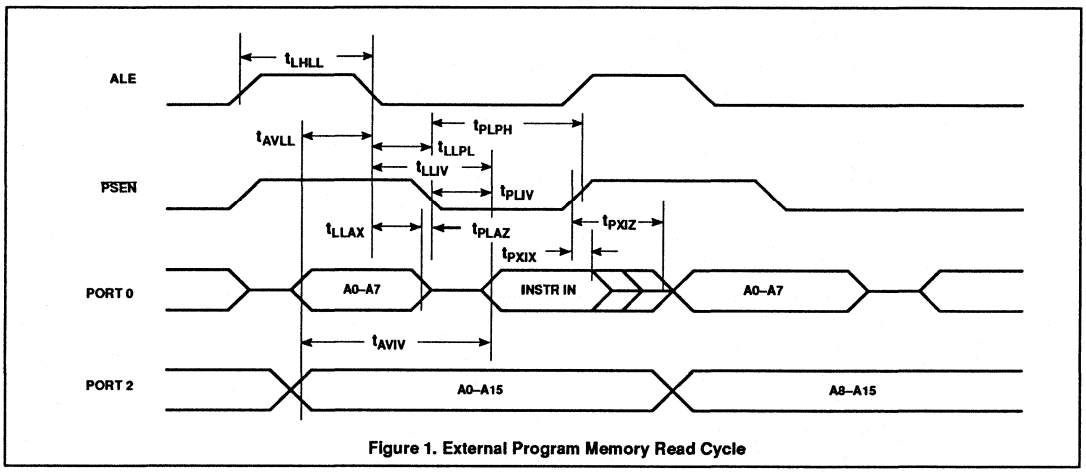


Figure 1. External Program Memory Read Cycle

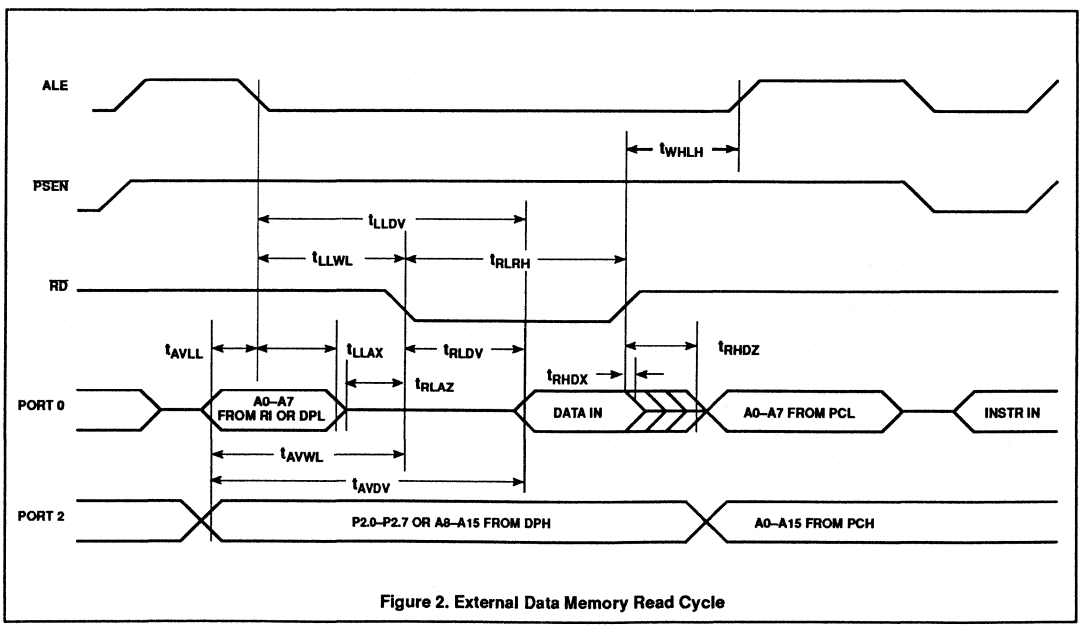


Figure 2. External Data Memory Read Cycle

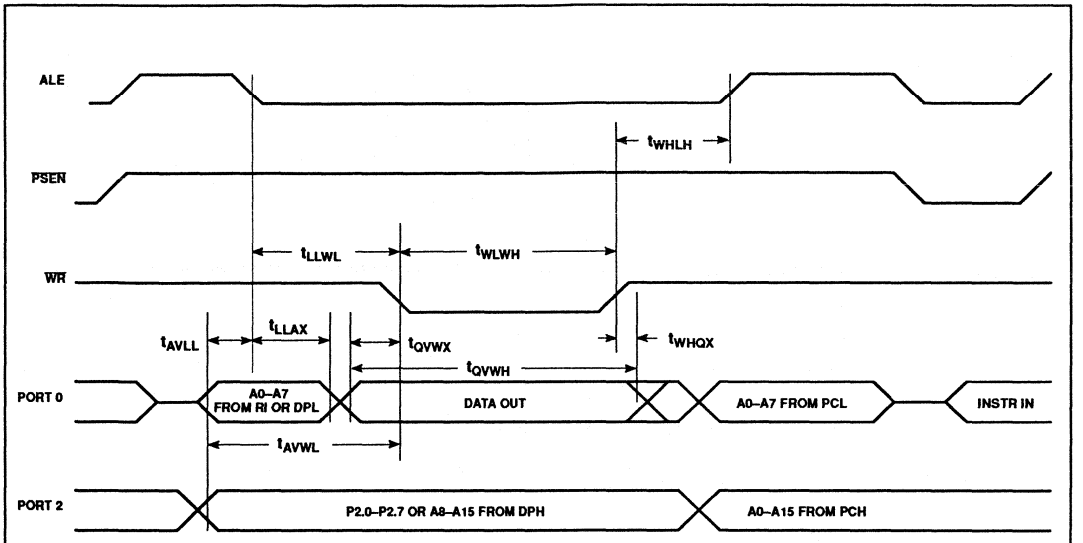


Figure 3. External Data Memory Write Cycle

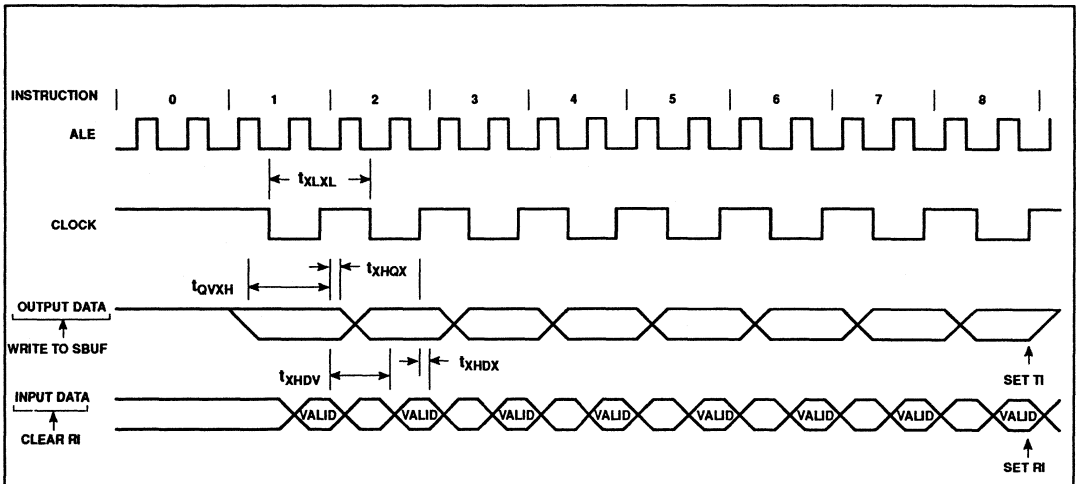


Figure 4. Shift Register Mode Timing

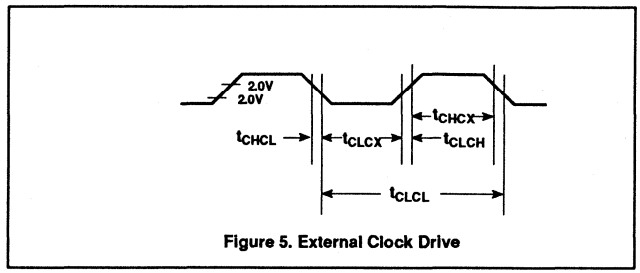
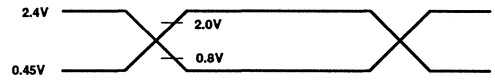
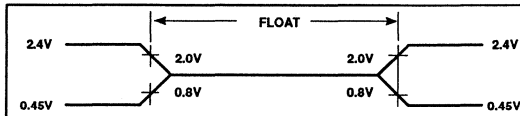


Figure 5. External Clock Drive



NOTE:
AC inputs during testing are driven at 2.4V for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at 2.0V min for a logic '1' and 0.8V for a logic '0'.

Figure 6. AC Testing Input/Output



NOTE:
For timing purposes, the float state is defined as the point at which a P0 pin sinks 2.4mA or sources 400µA at the voltage test levels.

Figure 7. Float Waveform

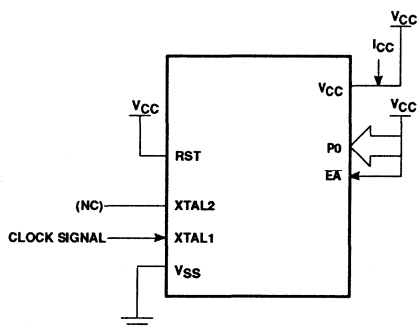


Figure 8. I_{CC} Test Condition, Active Mode
All other pins are disconnected

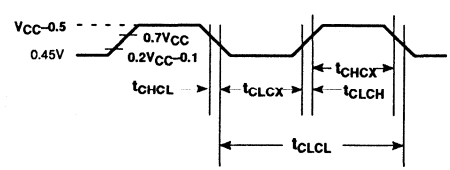


Figure 9. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

Philips Components

Date of Issue	May 23, 1990
Status	Product Specification
Application Specific Product	

80C31/80C51/87C51

CMOS single-chip, 8-bit microcontroller

DESCRIPTION

The Philips 80C31/80C51/87C51 is a high-performance microcontroller fabricated with Philips high-density CMOS technology. The CMOS 8XC51 is functionally compatible with the NMOS 8031/8051 microcontrollers. The Philips CMOS technology combines the high speed and density characteristics of HMOS with the low power attributes of CMOS. Philips epitaxial substrate minimizes latch-up sensitivity.

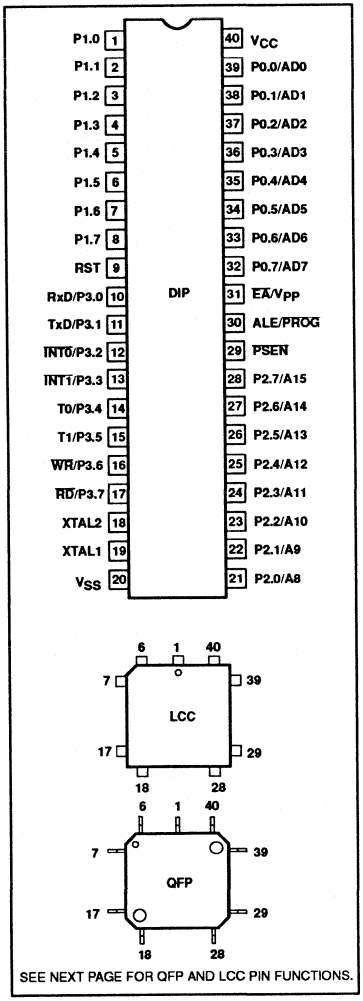
The 8XC51 contains a 4k x 8 ROM (80C51) EPROM (87C51), a 128 x 8 RAM, 32 I/O lines, two 16-bit counter/timers, a five-source, two-priority level nested interrupt structure, a serial I/O port for either multi-processor communications, I/O expansion or full duplex UART, and on-chip oscillator and clock circuits.

In addition, the device has two software selectable modes of power reduction – idle mode and power-down mode. The idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.

FEATURES

- 8031/8051 compatible
  - 4k x 8 ROM (80C51)
  - 4k x 8 EPROM (87C51)
  - ROMless (80C31)
  - 128 x 8 RAM
  - Two 16-bit counter/timers
  - Full duplex serial channel
  - Boolean processor
- Memory addressing capability
  - 64k ROM and 64k RAM
- Power control modes:
  - Idle mode
  - Power-down mode
- CMOS and TTL compatible
- Five speed ranges at V_{CC} = 5V
  - 12MHz
  - 16MHz
  - 20MHz
  - 24MHz
  - 30MHz
- Five package styles
- Extended temperature ranges
- OTP package available

PIN CONFIGURATION

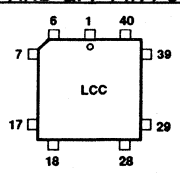


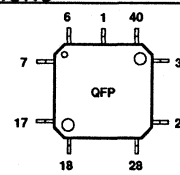
SEE NEXT PAGE FOR QFP AND LCC PIN FUNCTIONS.

PART NUMBER SELECTION

PHILIPS		PHILIPS COMPONENTS-SIGNETICS			TEMPERATURE °C AND PACKAGE	SPEED
ROMless	ROM	ROMless	ROM	EPROM		
				SC87C51CCF40	0 to +70, CDIP	12MHz
				SC87C51CGF40	0 to +70, CDIP	16MHz
				SC87C51CCL44	0 to +70, CLCC	12MHz
				SC87C51CGL44	0 to +70, CLCC	16MHz
PCB80C31BH-2P	PCB80C51BH-2P	SC80C31BCCN40	SC80C51BCCN40	SC87C51CCN40	0 to +70, PDIP	12MHz
PCB80C31BH-3P	PCB80C51BH-3P	SC80C31BCGN40	SC80C51BCGN40	SC87C51CGN40	0 to +70, PDIP	16MHz
PCB80C31BH-2WP	PCB80C51BH-2WP	SC80C31BCCA44	SC80C51BCCA44	SC87C51CCA44	0 to +70, PLCC	12MHz
PCB80C31BH-3WP	PCB80C51BH-3WP	SC80C31BCGA44	SC80C51BCGA44	SC87C51CGA44	0 to +70, PLCC	16MHz
PCF80C31BH-2P	PCF80C51BH-2P	SC80C31BACN40	SC80C51BACN40	SC87C51ACN40	–40 to +85, PDIP	12MHz
PCF80C31BH-3P	PCF80C51BH-3P	SC80C31BAGN40	SC80C51BAGN40	SC87C51AGN40	–40 to +85, PDIP	16MHz
PCB80C31BH-2WP	PCB80C51BH-2WP	SC80C31BACA44	SC80C51BACA44	SC87C51ACA44	–40 to +85, PLCC	12MHz
PCF80C31BH-3WP	PCF80C51BH-3WP	SC80C31BAGA44	SC80C51BAGA44	SC87C51AGA44	–40 to +85, PLCC	16MHz
				SC87C51ACF40	–40 to +85, CDIP	12MHz
				SC87C51ACL44	–40 to +85, CLCC	12MHz
				SC87C51AGL44	–40 to +85, CLCC	16MHz
				SC87C51AGF44	–40 to +85, CDIP	16MHz
PCF80C31BH-2H	PCF80C51BH-2H	SC80C31BACB44	SC80C51BACB44		–40 to +85, PQFP	12MHz
PCF80C31BH-3H	PCF80C51BH-3H	SC80C31BAGB44	SC80C51BAGB44		–40 to +85, PQFP	16MHz
		SC80C31BALB44	SC80C51BALB44		–40 to +85, PQFP	20MHz
PCB80C31BH-2H	PCB80C51BH-2H	SC80C31BCCB44	SC80C51BCCB44		0 to +70, PQFP	12MHz
PCB80C31BH-3H	PCB80C51BH-3H	SC80C31BCGB44	SC80C51BCGB44		0 to +70, PQFP	16MHz
PCA80C31BH-3P	PCA80C51BH-3P				–40 to +125, PDIP	16MHz
PCA80C31BH-3WP	PCA80C51BH-3WP				–40 to +125, PLCC	16MHz
		SC80C31BCLN40	SC80C51BCLN40		0 to +70, PDIP	20MHz
		SC80C31BCLA44	SC80C51BCLA44		0 to +70, PLCC	20MHz
		SC80C31BALN40	SC80C51BALN40		–40 to +85, PDIP	20MHz
		SC80C31BALA44	SC80C51BALA44		–40 to +85, PLCC	20MHz
		SC80C31BCLB44	SC80C51BCLB44		0 to +70, PQFP	20MHz
PCB80C31BH-4P	PCB80C51BH-4P	SC80C31BCPN40	SC80C51BCPN40		0 to +70, PDIP	24MHz
PCB80C31BH-4WP	PCB80C51BH-4WP	SC80C31BCPA44	SC80C51BCPA44		0 to +70, PLCC	24MHz
PCF80C31BH-4P	PCF80C51BH-4P	SC80C31BAPN40	SC80C51BAPN40		–40 to +85, PDIP	24MHz
PCF80C31BH-4WP	PCF80C51BH-4WP	SC80C31BAPA44	SC80C51BAPA44		–40 to +85, PLCC	24MHz
PCB80C31BH-4H	PCB80C51BH-4H	SC80C31BCPB44	SC80C51BCPB44		0 to +70, PQFP	24MHz
PCB80C31BH-5P	PCB80C51BH-5P				0 to +70, PDIP	30MHz
PCB80C31BH-5WP	PCB80C51BH-5WP				0 to +70, PLCC	30MHz
PCB80C31BH-5H	PCB80C51BH-5H				0 to +70, PQFP	30MHz

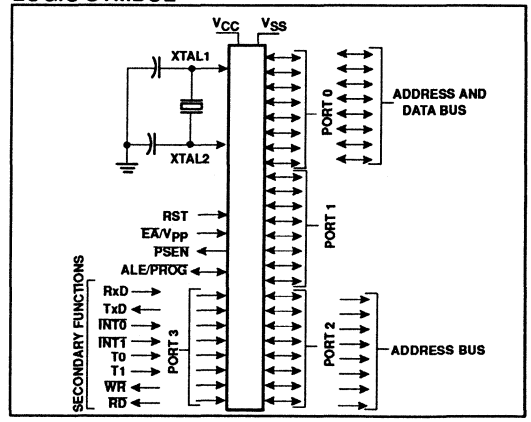
LCC AND QFP PIN FUNCTIONS



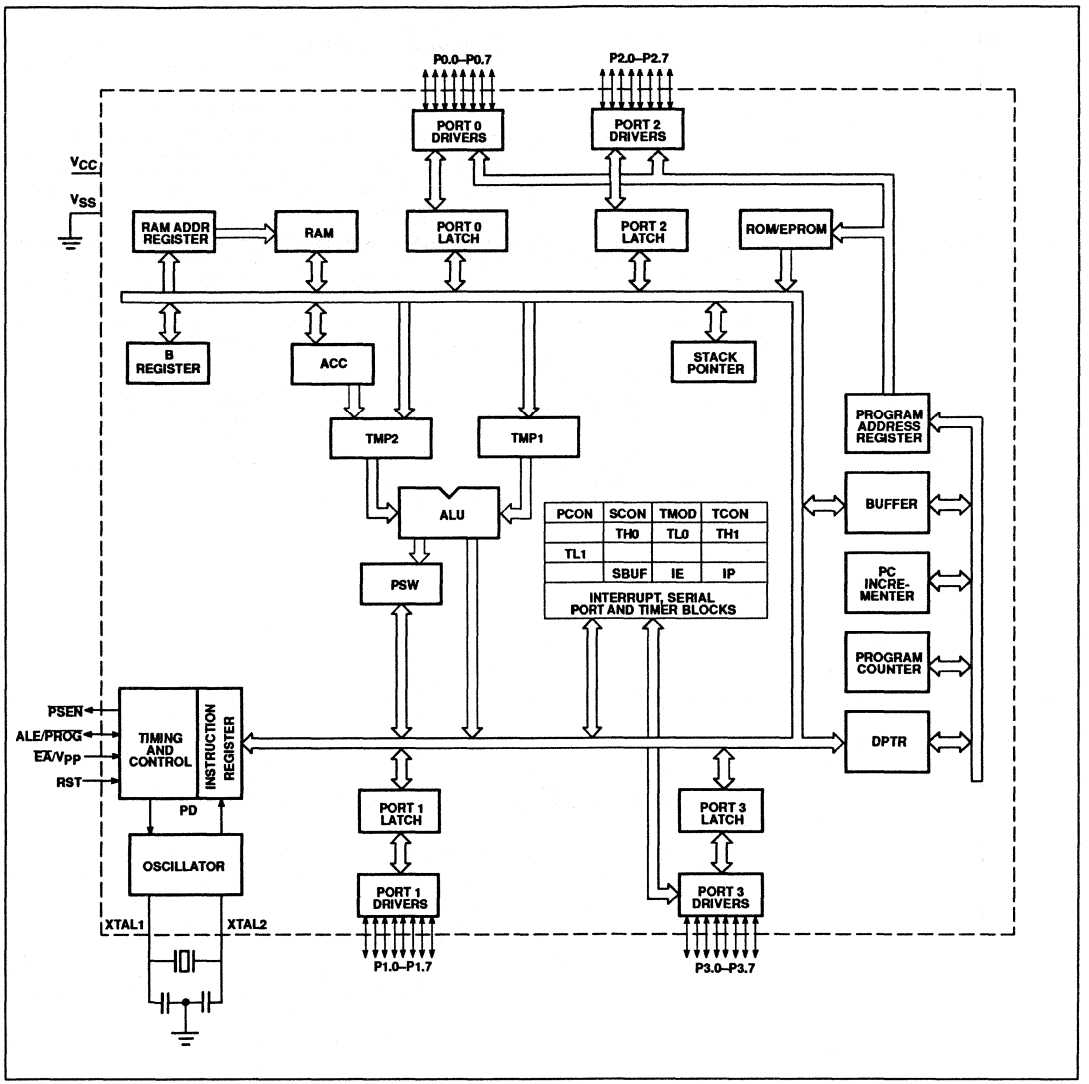


Pin	Function	Pin	Function	Pin	Function
1	NC	16	P3.4/T0	31	P2.7/A15
2	P1.0	17	P3.5/T1	32	PSEN
3	P1.1	18	P3.6/WR	33	ALE
4	P1.2	19	P3.7/RD	34	NC
5	P1.3	20	XTAL2	35	EA
6	P1.4	21	XTAL1	36	P0.7/AD7
7	P1.5	22	V_{SS}	37	P0.6/AD6
8	P1.6	23	V_{SS}	38	P0.5/AD5
9	P1.7	24	P2.0/A8	39	P0.4/AD4
10	RST	25	P2.1/A9	40	P0.3/AD3
11	P3.0/RxD	26	P2.2/A10	41	P0.2/AD2
12	NC	27	P2.3/A11	42	P0.1/AD1
13	P3.1/TxD	28	P2.4/A12	43	P0.0/AD0
14	P3.2/INT0	29	P2.5/A13	44	V_{CC}
15	P3.3/INT1	30	P2.6/A14		

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	TYPE	NAME AND FUNCTION
V_{SS}	20	22	I	**Ground:** 0V reference.
V_{CC}	40	44	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–46	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s. Port 0 also outputs the code bytes during program verification in the 87C51. External pull-ups are required during program verification.
P1.0–P1.7	1–8	2–9	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 1 also receives the low-order address byte during program memory verification.
P2.0–P2.7	21–28	24–31	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the 80C51 family, as listed below:
	10	11	I	**RxD (P3.0):** Serial input port
	11	13	O	**TxD (P3.1):** Serial output port
	12	14	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	I	**T0 (P3.4):** Timer 0 external input
	15	17	I	**T1 (P3.5):** Timer 1 external input
	16	18	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE/$\overline{PROG}$	30	33	I/O	**Address Latch Enable/Program Pulse:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.
$\overline{PSEN}$	29	32	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$/V_{PP}	31	35	I	**External Access Enable/Programming Supply Voltage:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H to 0FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 0FFFH. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
XTAL1	19	21	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	O	**Crystal 2:** Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-up reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-up, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

IDLE MODE

In idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. the control bits for the reduced power modes are in the special function register PCON.

Table 1 shows the state of I/O ports during low current operating modes.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

Electrical Deviations from Commercial Specifications for Extended Temperature Range (87C51)
DC and AC parameters not included here are the same as in the commercial temperature range table.

DC ELECTRICAL CHARACTERISTICS

$T_A = -40^oC$ to $+85^oC$, $V_{CC} = 5V \pm 10\%$, $V_{SS} = 0V$ (Signetics Parts Only)
$T_A = -40^oC$ to $+125^oC$, $V_{CC} = 5V \pm 10\%$, $V_{SS} = 0V$ (Philips Parts Only)

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS		UNIT
			MIN	MAX	
V_{IL}	Input low voltage, except $\overline{EA}$		–0.5	$0.2V_{CC}$–0.15	V
V_{IL}	Input low voltage, except $\overline{EA}$		–0.5	$0.2V_{DD}$–0.25	V
V_{IL1}	Input low voltage to $\overline{EA}$		0	$0.2V_{CC}$–0.35	V
V_{IL1}	Input low voltage to $\overline{EA}$		0	$0.2V_{DD}$–0.45	V
V_{IH}	Input high voltage, except XTAL1, RST		$0.2V_{CC}$+1	V_{CC}+0.5	V
V_{IH1}	Input high voltage to XTAL1, RST		$0.7V_{CC}$+0.1	V_{CC}+0.5	V
I_{IL}	Logical 0 input current, ports 1, 2, 3	$V_{IN} = 0.45V$		–75	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3	$V_{IN} = 2.0V$		–750	µA
I_{CC}	Power supply current: Active mode[1] (Philips) Active mode (Signetics) Idle mode[2] (Philips) Idle mode (Signetics) Power-down mode[3] (Philips and Signetics)	V_{CC} = 4.5–5.5V, Frequency range = 3.5 to 12MHz		 20 35 4.4 6 50	 mA mA mA mA µA

1. The operating supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = V_{DD}.
2. The idle mode supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = Port 0 = V_{DD}; RST = V_{SS}.
3. The power-down current is measured with all output pins disconnected, XTAL2 not connected, $\overline{EA}$ = Port 0 = V_{DD}; RST = V_{SS}.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Operating temperature under bias	0 to +70 or –40 to +85	°C
Storage temperature range	–65 to +150	°C
Voltage on $\overline{EA}/V_{PP}$ pin to V_{SS}	0 to +13.0	V
Voltage on any other pin to V_{SS}	–0.5 to +6.5	V
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.5	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

$T_A = 0^oC$ to $+70^oC$ or -40^oC to $+85^oC$, $V_{CC} = 5V \pm 20\%$, $V_{SS} = 0V$ (80C31/51)
$T_A = 0^oC$ to $+70^oC$ or -40^oC to $+85^oC$, $V_{CC} = 5V \pm 10\%$, $V_{SS} = 0V$ (87C51)
$T_A = -40^oC$ to $+125^oC$, $V_{CC} = 5V \pm 10\%$, $V_{SS} = 0V$ (Philips Only)

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			MIN	TYPICAL[1]	MAX	
V_{IL}	Input low voltage, except $\overline{EA}$[7]		–0.5		$0.2V_{CC}-0.1$	V
V_{IL1}	Input low voltage to $\overline{EA}$[7]		0		$0.2V_{CC}-0.3$	V
V_{IH}	Input high voltage, except XTAL1, RST[7]		$0.2V_{CC}+0.9$		$V_{CC}+0.5$	V
V_{IH1}	Input high voltage, XTAL1, RST[7]		$0.7V_{CC}$		$V_{CC}+0.5$	V
V_{OL}	Output low voltage, ports 1, 2, 3	$I_{OL} = 1.6mA$[2]			0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$	$I_{OL} = 3.2mA$[2]			0.45	V
V_{OH}	Output high voltage, ports 1, 2, 3, ALE, $\overline{PSEN}$[3]	$I_{OH} = -60\mu A$, $I_{OH} = -25\mu A$ $I_{OH} = -10\mu A$	2.4 $0.75V_{CC}$ $0.9V_{CC}$			V V V
V_{OH1}	Output high voltage (port 0 in external bus mode)	$I_{OH} = -800\mu A$, $I_{OH} = -300\mu A$ $I_{OH} = -80\mu A$	2.4 $0.75V_{CC}$ $0.9V_{CC}$			V V V
I_{IL}	Logical 0 input current, ports 1, 2, 3[7]	$V_{IN} = 0.45V$			–50	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3[7]	See note 4			–650	µA
I_{LI}	Input leakage current, port 0	$V_{IN} = V_{IL}$ or V_{IH}			±10	µA
I_{CC}	Power supply current:[7] Active mode @ 12MHz[8] (Philips) Active mode @ 12MHz[5] (Signetics) Idle mode @ 12MHz[9] (Philips) Idle mode @ 12MHz (Signetics) Power-down mode[10] (Philips and Signetics)	See note 6		 11.5 1.3 3	 18 25 4.4 4 50	 mA mA mA mA µA
R_{RST}	Internal reset pull-down resistor		50		300	kohm
C_{IO}	Pin capacitance				10	pF

NOTES:
1. Typical ratings are not guaranteed. The values listed are at room temperature, 5V.
2. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.. I_{OL} can exceed these conditions provided that no single output sinks more than 5mA and no more than two outputs exceed the test conditions.
3. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
4. Pins of ports 1, 2 and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
5. I_{CC}MAX at other frequencies (for Signetics parts) is given by: Active mode: I_{CC}MAX = 0.94 X FREQ + 13.71: Idle mode: I_{CC}MAX = 0.14 X FREQ +2.31,
   where FREQ is the external oscillator frequency in MHz. I_{CC}MAX is given in mA. See Figure 7.
6. See Figures 8 through 11 for I_{CC} test conditions.
7. For Signetics parts when $T_A = -40^oC$ to $+85^oC$ or Philips parts when $T_A = -40^oC$ to $+1255^oC$, see table on previous page.
8. The operating supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = V_{DD}.
9. The idle mode supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = Port 0 = V_{DD}; RST = V_{SS}.
10. The power-down current is measured with all output pins disconnected, XTAL2 not connected, $\overline{EA}$ = Port 0 = V_{DD}; RST = V_{SS}.

AC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±20%, V_{SS} = 0V (80C31/51)[1,2]
T_A = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V (87C51)

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$		Oscillator frequency: **Speed Versions** 8XC31/51 B C G –2 –3			 0.5 3.5 3.5 1.2 1.2	 12 12 16 12 16	 MHz MHz MHz MHz MHz
t_{LHLL}	1	ALE pulse width	127		$2t_{CLCL}$–40		ns
t_{AVLL}	1	Address valid to ALE low	28		t_{CLCL}–55		ns
t_{LLAX}	1	Address hold after ALE low	48		t_{CLCL}–35		ns
t_{LLIV}	1	ALE low to valid instruction in		234		$4t_{CLCL}$–100	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	43		t_{CLCL}–40		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	205		$3t_{CLCL}$–45		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		145		$3t_{CLCL}$–105	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		59		t_{CLCL}–25	ns
t_{AVIV}	1	Address to valid instruction in		312		$5t_{CLCL}$–105	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	400		$6t_{CLCL}$–100		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	400		$6t_{CLCL}$–100		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		252		$5t_{CLCL}$–165	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		97		$2t_{CLCL}$–70	ns
t_{LLDV}	2, 3	ALE low to valid data in		517		$8t_{CLCL}$–150	ns
t_{AVDV}	2, 3	Address to valid data in		585		$9t_{CLCL}$–165	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}$–50	$3t_{CLCL}$+50	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}$–130		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	23		t_{CLCL}–60		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	33		t_{CLCL}–50		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	t_{CLCL}–40	t_{CLCL}+40	ns
External Clock							
t_{CHCX}	4	High time	20		20		ns
t_{CLCX}	4	Low time	20		20		ns
t_{CLCH}	4	Rise time		20		20	ns
t_{CHCL}	4	Fall time		20		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.

AC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or −40°C to +85°C, V_{CC} = 5V ±20%, V_{SS} = 0V (80C31/51)[1,2]

SYMBOL	FIGURE	PARAMETER	VARIABLE CLOCK[3] MIN	VARIABLE CLOCK[3] MAX	UNIT
$1/t_{CLCL}$		Oscillator frequency: **Speed Versions** 80C31/51 L P	 3.5 3.5	 20 24	 MHz MHz
t_{LHLL}	1	ALE pulse width	$2t_{CLCL}-40$		ns
t_{AVLL}	1	Address valid to ALE low	$t_{CLCL}-32$		ns
t_{LLAX}	1	Address hold after ALE low	$t_{CLCL}-20$		ns
t_{LLIV}	1	ALE low to valid instruction in		$4t_{CLCL}-100$	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	$t_{CLCL}-22$		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	$3t_{CLCL}-45$		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		$3t_{CLCL}-105$	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		$t_{CLCL}-25$	ns
t_{AVIV}	1	Address to valid instruction in		$5t_{CLCL}-105$	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		10	ns
Data Memory					
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	$6t_{CLCL}-100$		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	$6t_{CLCL}-100$		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		$5t_{CLCL}-165$	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		$2t_{CLCL}-28$	ns
t_{LLDV}	2, 3	ALE low to valid data in		$8t_{CLCL}-150$	ns
t_{AVDV}	2, 3	Address to valid data in		$9t_{CLCL}-165$	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	$4t_{CLCL}-130$		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	$t_{CLCL}-40$		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	$t_{CLCL}-40$		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		0	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock					
t_{CHCX}	4	High time	20		ns
t_{CLCX}	4	Low time	20		ns
t_{CLCH}	4	Rise time		20	ns
t_{CHCL}	4	Fall time		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. For Signetics speed versions L and P only.

AC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or −40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V (80C31/51)[1,2]

SYMBOL	FIGURE	PARAMETER	VARIABLE CLOCK[3]		UNIT
			MIN	MAX	
$1/t_{CLCL}$		Oscillator frequency: **Speed Versions** 80C31/51 I	1.2	24	MHz
t_{LHLL}	1	ALE pulse width	$2t_{CLCL}-40$		ns
t_{AVLL}	1	Address valid to ALE low	$t_{CLCL}-32$		ns
t_{LLAX}	1	Address hold after ALE low	$t_{CLCL}-20$		ns
t_{LLIV}	1	ALE low to valid instruction in		$4t_{CLCL}-100$	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	$t_{CLCL}-22$		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	$3t_{CLCL}-45$		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		$3t_{CLCL}-80$	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		$t_{CLCL}-25$	ns
t_{AVIV}	1	Address to valid instruction in		$5t_{CLCL}-105$	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		10	ns
Data Memory					
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	$6t_{CLCL}-100$		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	$6t_{CLCL}-100$		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		$5t_{CLCL}-125$	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		$2t_{CLCL}-28$	ns
t_{LLDV}	2, 3	ALE low to valid data in		$8t_{CLCL}-150$	ns
t_{AVDV}	2, 3	Address to valid data in		$9t_{CLCL}-165$	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	$4t_{CLCL}-80$		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	$t_{CLCL}-40$		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	$t_{CLCL}-35$		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		0	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock					
t_{CHCX}	4	High time	17		ns
t_{CLCX}	4	Low time	17		ns
t_{CLCH}	4	Rise time		20	ns
t_{CHCL}	4	Fall time		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. For Philips speed version I only.

AC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C, V_{CC} = 5V ±20%, V_{SS} = 0V (80C31/51)[1, 2, 3]

SYMBOL	FIGURE	PARAMETER	VARIABLE CLOCK[4]		UNIT
			MIN	MAX	
$1/t_{CLCL}$		Oscillator frequency: **Speed Versions** 80C31/51 K	1.2	30	MHz
t_{LHLL}	1	ALE pulse width	$2t_{CLCL}-40$		ns
t_{AVLL}	1	Address valid to ALE low	$t_{CLCL}-25$		ns
t_{LLAX}	1	Address hold after ALE low	$t_{CLCL}-25$		ns
t_{LLIV}	1	ALE low to valid instruction in		$4t_{CLCL}-65$	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	$t_{CLCL}-25$		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	$3t_{CLCL}-45$		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		$3t_{CLCL}-60$	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		$t_{CLCL}-25$	ns
t_{AVIV}	1	Address to valid instruction in		$5t_{CLCL}-80$	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		10	ns
Data Memory					
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	$6t_{CLCL}-100$		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	$6t_{CLCL}-100$		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		$5t_{CLCL}-90$	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		$2t_{CLCL}-28$	ns
t_{LLDV}	2, 3	ALE low to valid data in		$8t_{CLCL}-150$	ns
t_{AVDV}	2, 3	Address to valid data in		$9t_{CLCL}-165$	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	$4t_{CLCL}-75$		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	$t_{CLCL}-30$		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	$t_{CLCL}-25$		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		0	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	$t_{CLCL}-25$	$t_{CLCL}+25$	ns
External Clock					
t_{CHCX}	4	High time	15		ns
t_{CLCX}	4	Low time	15		ns
t_{CLCH}	4	Rise time		20	ns
t_{CHCL}	4	Fall time		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. Interfacing the 80C31/51 to devices with float times up to 30ns is permitted. This limited bus contention will not cause damage to port 0 drivers.
4. For Philips speed version K only.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

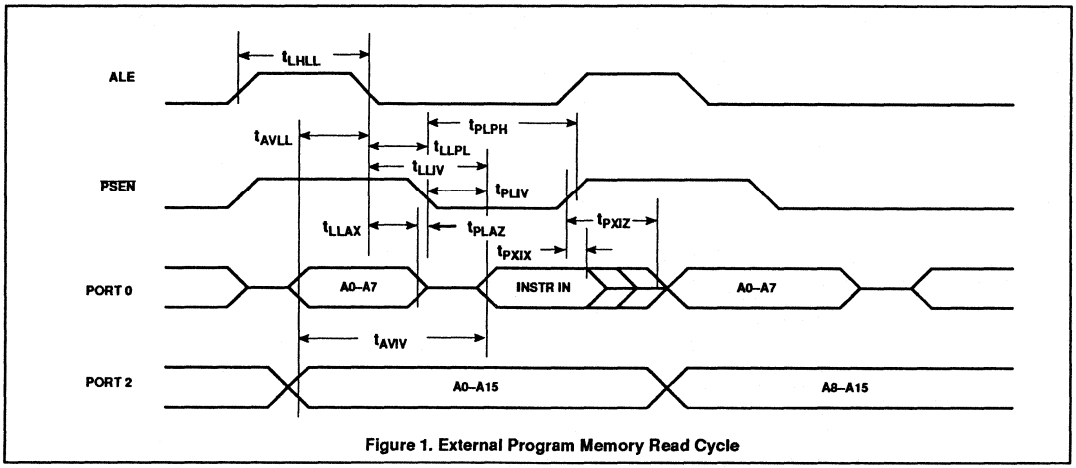


Figure 1. External Program Memory Read Cycle

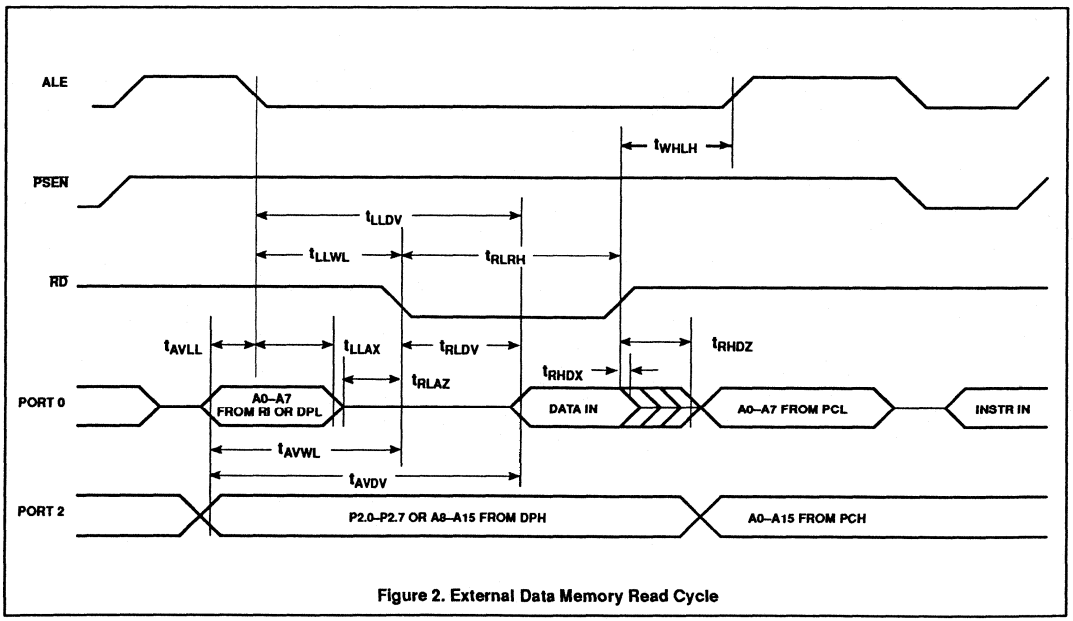


Figure 2. External Data Memory Read Cycle

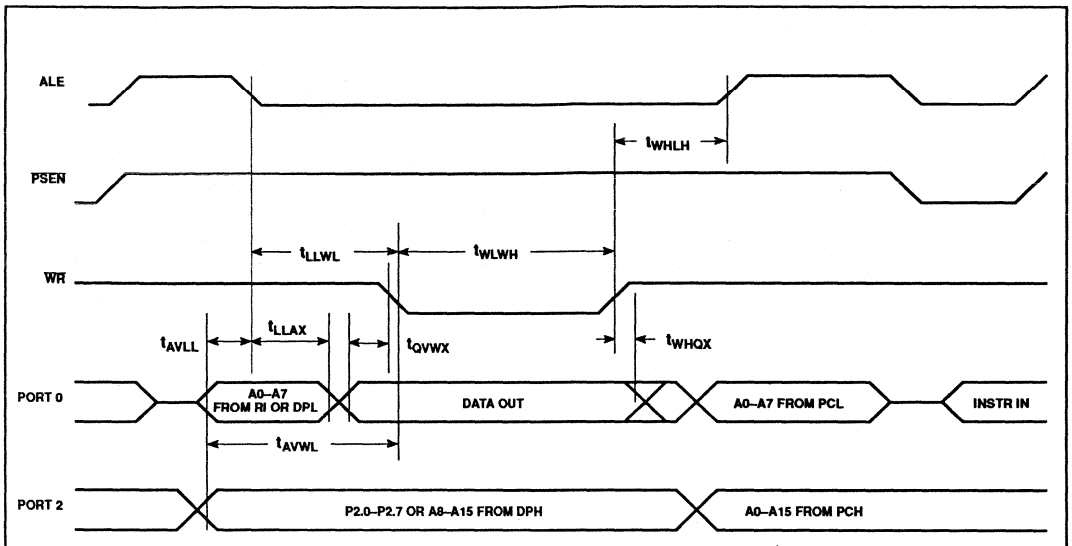


Figure 3. External Data Memory Write Cycle

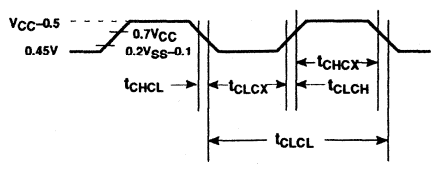
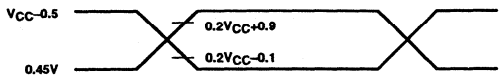
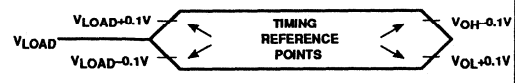


Figure 4. External Clock Drive



NOTE:
AC inputs during testing are driven at V_{CC} –0.5 for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at V_{IH} min for a logic '1' and V_{IL} for a logic '0'.

Figure 5. AC Testing Input/Output



NOTE:
For timing purposes, a port is no longer floating when a 100mV change from load voltage occurs, and begins to float when a 100mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OH}/I_{OL} \geq \pm 20$mA.

Figure 6. Float Waveform

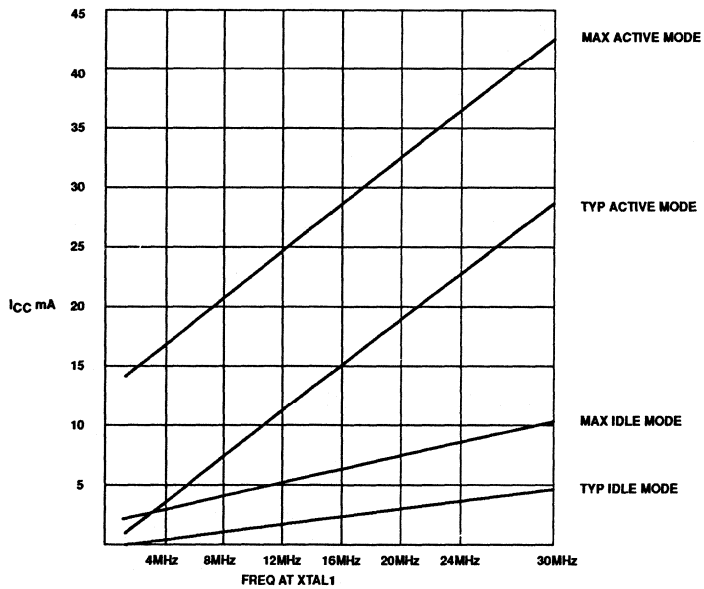


Figure 7. I_{CC} vs. FREQ
Valid only within frequency specifications of the device under test

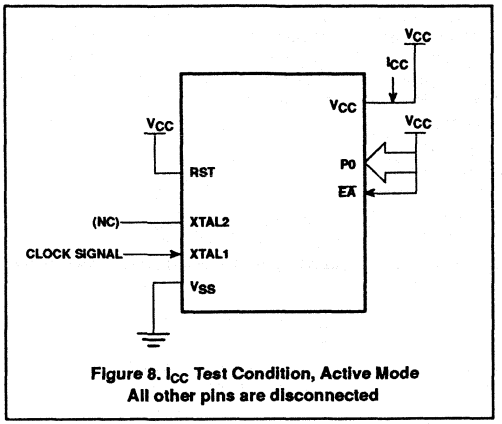


Figure 8. I_{CC} Test Condition, Active Mode
All other pins are disconnected

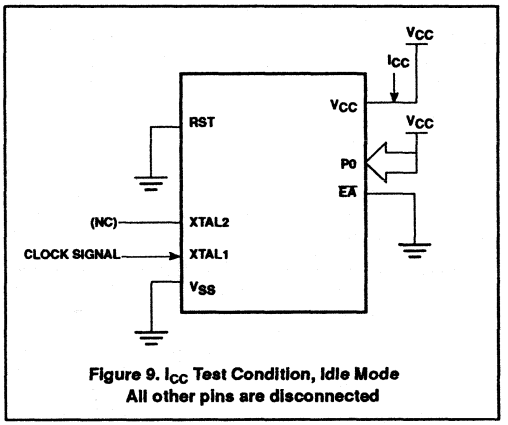


Figure 9. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

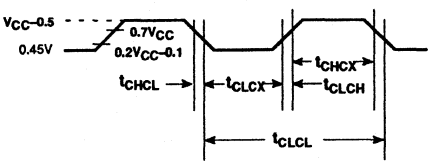


Figure 10. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

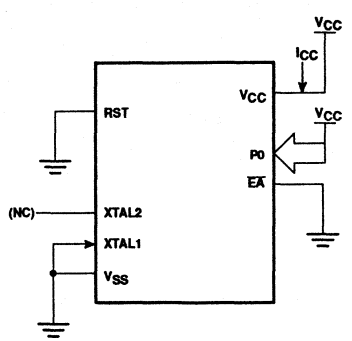


Figure 11. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected. V_{CC} = 2V to 5.5V

EPROM CHARACTERISTICS

The 87C51 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C51 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C51 manufactured by Philips Corporation.

Table 2 shows the logic levels for reading the signature byte, and for programming the program memory, the encryption table, and the lock bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 12 and 13. Figure 14 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 12. Note that the 87C51 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1 and 2, as shown in Figure 12. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$ and pins of ports 2 and 3 specified in Table 2 are held at the 'Program Code Data' levels indicated in Table 2. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 13.

To program the encryption table, repeat the 25 pulse programming sequence for addresses 0 through 1FH, using the 'Pgm Encryption Table' levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the lock bits, repeat the 25 pulse programming sequence using the 'Pgm Lock Bit' levels. After one lock bit is programmed, further programming of the code memory and encryption table is disabled. However, the other lock bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If lock bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be read is applied to ports 1 and 2 as shown in Figure 14. The other pins are held at the 'Verify Code Data' levels indicated in Table 2. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips
(031H) = 92H indicates 87C51

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 2, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345–5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000uW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

Table 2. EPROM Progamming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm lock bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm lock bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:
1. '0' = Valid low for that pin, '1' = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{CC} = 5V±10% during programming and verification.

*ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

™Trademark phrase of Intel Corporation.

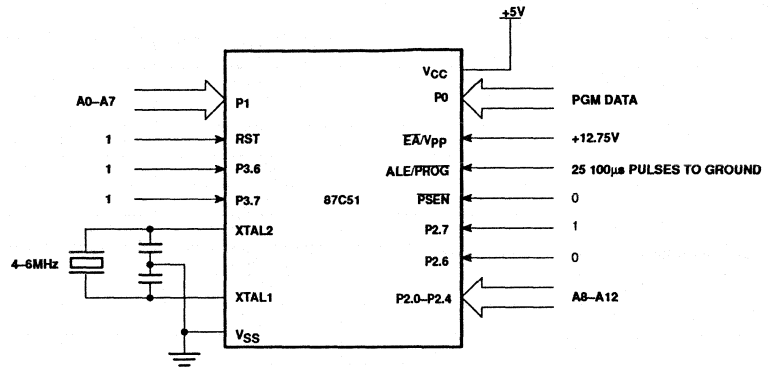


Figure 12. Programming Configuration

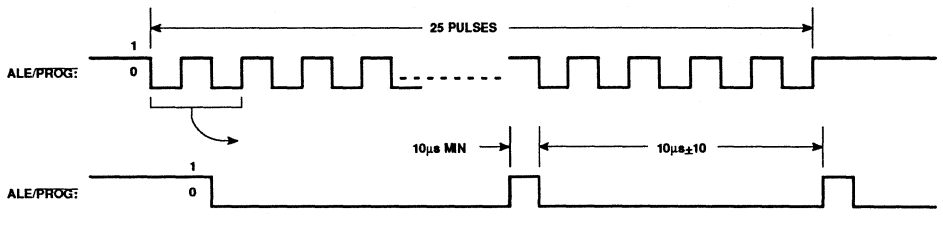


Figure 13. PROG Waveform

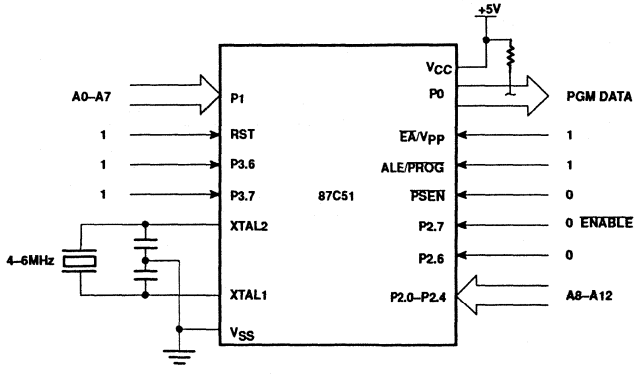


Figure 14. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_A = 21°C to +27°C, V_{CC} = 5V±10%, V_{SS} = 0V (See Figure 15)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		us
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		us
t_{GLGH}	$\overline{PROG}$ width	90	110	us
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	$\overline{ENABLE}$ low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		us

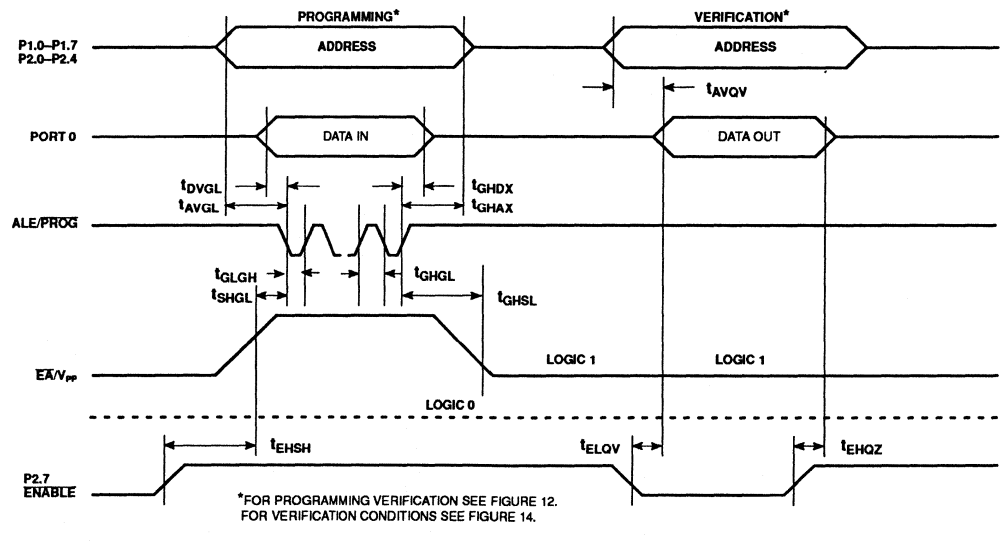


Figure 15. EPROM Programming and Verification

Section 2
I²C Bus Specification

CONTENTS

The I²C Bus Specification 131
I²C Address Allocation Table 147
Assigned I²C Bus Addresses 148
I²C Peripheral Selection Guide 149

Section 2
Inter-Integrated (I^2C) Circuit Bus

INTRODUCTION

For 8-bit applications, such as those requiring single-chip microcomputers, certain design criteria can be established:

- A complete system usually consists of at least one microcomputer and other peripheral devices, such as memories and I/O expanders.
- The cost of connecting the various devices within the system must be kept to a minimum.
- Such a system usually performs a control function and does not require high-speed data transfer.
- Overall efficiency depends on the devices chosen and the interconnecting bus structure.

In order to produce a system to satisfy these criteria, a serial bus structure is needed. Although serial buses don't have the through-put capability of parallel buses, they do require less wiring and fewer connecting pins. However, a bus is not merely an interconnecting wire, it embodies all the formats and procedures for communication within the system.

Devices communicating with each other on a serial bus must have some form of protocol which avoids all possibilities of confusion, data loss, and blockage of information. Fast devices must be able to communicate with slow devices. The system must not be dependent on the devices connected to it; otherwise, modifications or improvements would be impossible. A procedure has also to be resolved to decide which device will be in control of the bus and when. And if different devices with different clock speeds are connected to the bus, the bus clock source must be defined.

All these criteria are involved in the specification of the I^2C bus.

THE I^2C BUS CONCEPT

Any manufacturing process (NMOS, CMOS, I^2L) can be supported by the I^2C bus. Two wires (SDA—serial data, SCL—serial clock) carry information between the devices connected to the bus. Each device is recognized by a unique address—whether it is a microcomputer, LCD driver, memory or keyboard interface—and can operate as either a transmitter or receiver, depending on the function of the device. Obviously an LCD driver is only a receiver, while a memory can both receive and transmit data. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers (see Table 1). A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave.

The I^2C bus is a multimaster bus. This means that more than one device capable of controlling the bus can be connected to it. As masters are usually microcomputers, let's consider the case of a data transfer between two microcomputers connected to the I^2C bus (Figure 1). This highlights the master-slave and receiver-transmitter relationships to be found on the I^2C bus. It should be noted that these relationships are not permanent, but only depend on the direction of data transfer at that time. The transfer of data would follow in this way:

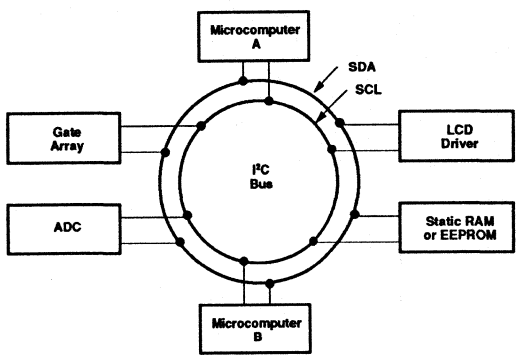


Figure 1. Typical I^2C Bus Configuration

Table 1. Definition of I^2C Bus Terminology

TERM	DESCRIPTION
Transmitter	The device which sends data to the bus
Receiver	The device which receives data from the bus
Master	The device which initiates a transfer, generates clock signals, and terminates a transfer
Slave	The device addressed by a master
Multi-master	More than one master can attempt to control the bus at the same time without corrupting the message
Arbitration	Procedure to ensure that if more than one master simultaneously tries to control the bus, only one is allowed to do so and the message is not corrupted
Synchronization	Procedure to synchronize the clock signals of two or more devices

1. Suppose microcomputer A wants to send information to microcomputer B.
   - –Microcomputer A (master) addresses microcomputer B (slave)
   - –Microcomputer A (master transmitter) sends data to microcomputer B (slave receiver)
   - –Microcomputer A terminates the transfer.
2. If microcomputer A wants to receive information from microcomputer B
   - –Microcomputer A (master) addresses microcomputer B (slave)
   - –Microcomputer A (master receiver) receives data from microcomputer B (slave transmitter)
   - –Microcomputer A terminates the transfer.

Even in this case, the master (microcomputer A) generates the timing and terminates the transfer.

The possibility of more than one microcomputer being connected to the I^2C bus means that more than one master could try to initiate a data transfer at the same time. To avoid the chaos that might ensue from such an event, an arbitration procedure has been developed. This procedure relies on the wired-AND connection of all devices to the I^2C bus.

If two or more masters try to put information on to the bus, the first to produce a one when the other produces a zero will lose the arbitration. The clock signals during arbitration are a synchronized combination of the clocks generated by the masters using the wired-AND connection to the SCL line (for more detailed information concerning arbitration see Arbitration and Clock Generation).

Generation of clock signals on the I^2C bus is always the responsibility of master devices; each master generates its own clock signals when transferring data on the bus. Bus clock signals from a master can only be altered when they are stretched by a slow slave device holding down the clock line or by another master when arbitration takes place.

GENERAL CHARACTERISTICS

Both SDA and SCL are bidirectional lines, connected to a positive supply voltage via a pull-up resistor (see Figure 2). When the bus is free, both lines are High. The output stages of devices connected to the bus must have an open-drain or open-collector in order to perform the wired-AND function. Data on the I^2C bus can be transferred at a rate up to 100kbit/s. The number of devices connected to the bus is solely dependent on the limiting bus capacitance of 400pF.

BIT TRANSFER

Due to the variety of different technology devices (CMOS, NMOS, I^2L) which can be connected to the I^2C bus, the levels of the logical 0 (Low) and 1 (High) are not fixed and depend on the appropriate level of V_{DD} (see Electrical Specifications). One clock pulse is generated for each data bit transferred.

Data Validity

The data on the SDA line must be stable during the High period of the clock. The High or Low state of the data line can only change when the clock signal on the SCL line is Low (Figure 3).

Start and Stop Conditions

Within the procedure of the I^2C bus, unique situations arise which are defined as start and stop conditions (see Figure 4).

A High-to-Low transition of the SDA line while SCL is High is one such unique case. This situation indicates a start condition.

A Low-to-High transition of the SDA line while SCL is High defines a stop condition.

Start and stop conditions are always generated by the master. The bus is considered to be busy after the start condition. The bus is considered to be free again a certain time after the stop condition. This bus free situation will be described later in detail.

Detection of start and stop conditions by devices connected to the bus is easy if they have the necessary interfacing hardware. However, microcomputers with no such interface have to sample the SDA line at least twice per clock period in order to sense the transition.

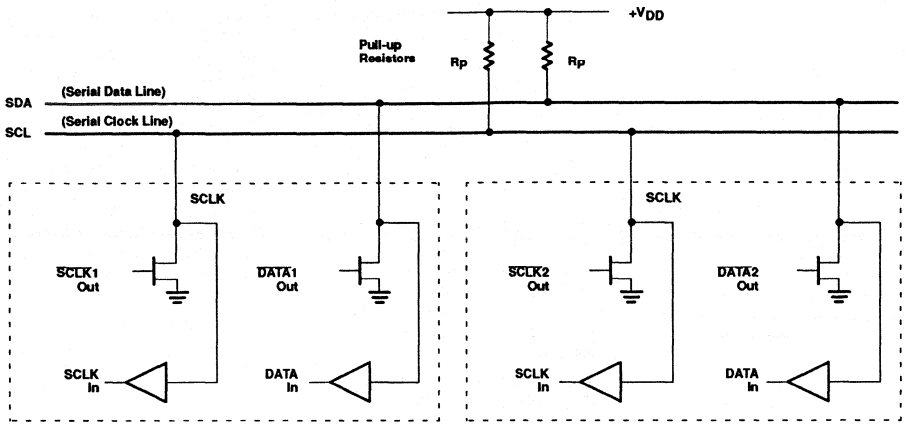


Figure 2. Connection of Devices to the I^2C Bus

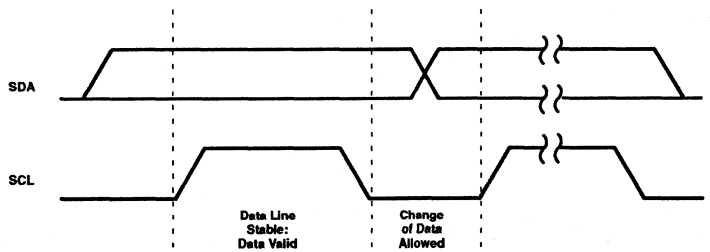


Figure 3. Bit Transfer on the I^2C Bus

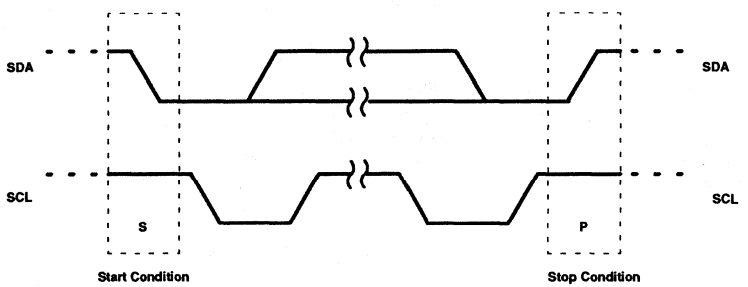


Figure 4. Start and Stop Conditions

TRANSFERRING DATA

Byte Format

Every byte put on the SDA line must be 8 bits long. The number of bytes that can be transmitted per transfer is unrestricted. Each byte must be followed by an acknowledge bit.

Data is transferred with the most significant bit (MSB) first (Figure 5). If a receiving device cannot receive another complete byte of data until it has performed some other function, for example, to service an internal interrupt, it can hold the clock line SCL Low to force the transmitter into a wait state. Data transfer then continues when the receiver is ready for another byte of data and releases the clock line SCL.

In some cases, it is permitted to use a different format from the I^2C bus format, such as CBUS compatible devices. A message which starts with such an address can be terminated by the generation of a stop condition, even during the transmission of a byte. In this case, no acknowledge is generated.

Acknowledge

Data transfer with acknowledge is obligatory. The acknowledge-related clock pulse is generated by the master. The transmitting device releases the SDA line (High) during the acknowledge clock pulse.

The receiving device has to pull down the SDA line during the acknowledge clock pulse so that the SDA line is stable Low during the high period of this clock pulse (Figure 6). Of course, setup and hold times must also be taken into account and these will be described in the Timing section.

Usually, a receiver which has been addressed is obliged to generate an acknowledge after each byte has been received (except when the message starts with a CBUS address).

When a slave receiver does not acknowledge on the slave address, for example, because it is unable to receive while it is performing some real-time function, the data line must be left High by the slave. The master can then generate a STOP condition to abort the transfer.

If a slave receiver does acknowledge the slave address, but some time later in the transfer cannot receive any more data bytes, the master must again abort the transfer. This is indicated by the slave not generating the acknowledge on the first byte following. The slave leaves the data line High and the master generates the STOP condition.

In the case of a master receiver involved in a transfer, it must signal an end of data to the slave transmitter by not generating an acknowledge on the last byte that was clocked out of the slave. The slave transmitter must release the data line to allow the master to generate the STOP condition.

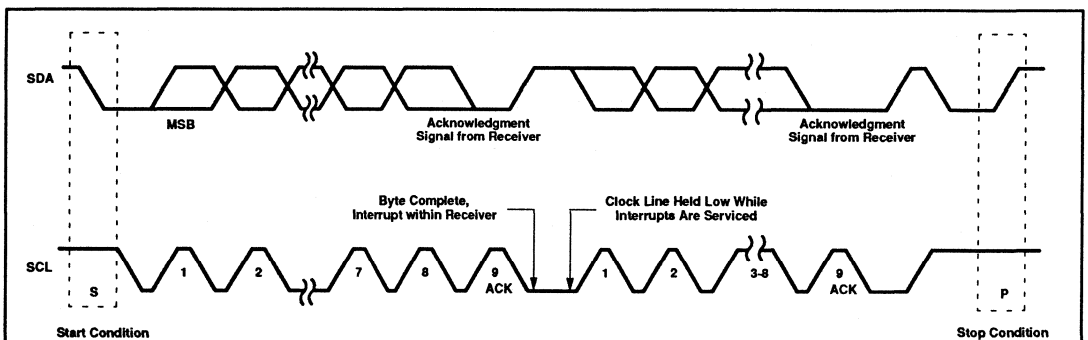


Figure 5. Data Transfer on the I^2C Bus

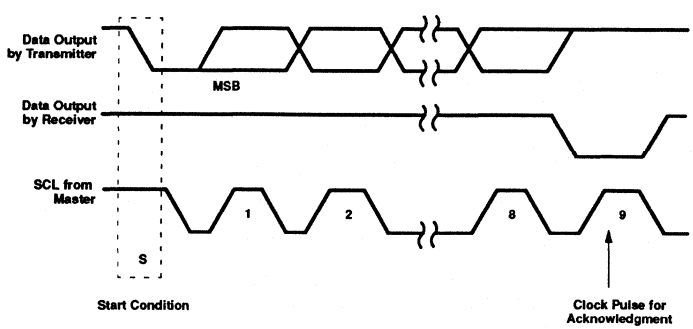


Figure 6. Acknowledge on the I^2C Bus

ARBITRATION AND CLOCK GENERATION

Synchronization

All masters generate their own clock on the SCL line to transfer messages on the I^2C bus. Data is only valid during the clock High period on the SCL line; therefore, a defined clock is needed if the bit-by-bit arbitration procedure is to take place.

Clock synchronization is performed using the wired-AND connection of devices to the SCL LINE. This means that a High-to-Low transition on the SCL line will affect the devices concerned, causing them to start counting off their Low period. Once a device clock has gone Low it will hold the SCL line in that state until the clock High state is reached (Figure 7). However, the Low-to-High change in this device clock may not change the state of the SCL line if another device clock is still within its Low period. Therefore, SCL will be held Low by the device with the longest Low period. Devices with shorter Low periods enter a High wait state during this time.

When all devices concerned have counted off their Low period, the clock line will be released and go High. There will then be no difference between the device clocks and the state of the SCL line and all of them will start counting their High periods. The first device to complete its High period will again pull the SCL line Low.

In this way, a synchronized SCL clock is generated for which the Low period is determined by the device with the longest clock Low period while the High period on SCL is determined by the device with the shortest clock High period.

Arbitration

Arbitration takes place on the SDA line in such a way that the master which transmits a High level, while another master transmits a Low level, will switch off its DATA output stage since the level on the bus does not correspond to its own level.

Arbitration can carry on through many bits. The first stage of arbitration is the comparison of the address bits. If the masters are each trying to address the same device, arbitration continues into a comparison of the data. Because address and data information is used on the I^2C bus for the arbitration, no information is lost during this process.

A master which loses the arbitration can generate clock pulses until the end of the byte in which it loses the arbitration.

If a master does lose arbitration during the addressing stage, it is possible that the winning master is trying to address it. Therefore, the losing master must switch over immediately to its slave receiver mode.

Figure 8 shows the arbitration procedure for two masters. Of course more may be involved, depending on how many masters are connected to the bus. The moment there is a difference between the internal data level of the master generating DATA 1 and the actual level on the SDA line, its data output is switched off, which means that a High output level is then connected to the bus. This will not affect the data transfer initiated by the winning master. As control of the I^2C bus is decided solely on the address and data sent by competing masters, there is no central master, nor any order of priority on the bus.

Use of the Clock Synchronizing Mechanism as a Handshake

In addition to being used during the arbitration procedure, the clock synchronization mechanism can be used to enable receiving devices to cope with fast data transfers, either on a byte or bit level.

On the byte level, a device may be able to receive bytes of data at a fast rate, but needs more time to store a received byte or prepare another byte to be transmitted. Slave devices can then hold the SCL line Low, after reception and acknowledge of a byte, to force the master into a wait state until the slave is ready for the next byte transfer in a type of handshake procedure.

On the bit level, a device such as a microcomputer without a hardware I^2C interface on-chip can slow down the bus clock by extending each clock Low period. In this way, the speed of any master is adapted to the internal operating rate of this device.

Formats

Data transfers follow the format shown in Figure 9. After the start condition, a slave address is sent. This address is 7 bits long; the eighth bit is a data direction bit (R/$\overline{W}$). A zero indicates a transmission (WRITE); a one indicates a request for data (READ). A data transfer is always terminated by a stop condition generated by the master. However, if a master still wishes to communicate on the bus, it can generate another start condition, and address another slave without first generating a stop condition. Various combinations of read/write formats are then possible within such a transfer.

At the moment of the first acknowledge, the master transmitter becomes a master receiver and the slave receiver becomes a slave transmitter. This acknowledge is still generated by the slave.

The stop condition is generated by the master.

During a change of direction within a transfer, the start condition and the slave address are both repeated, but with the R/$\overline{W}$ bit reversed.

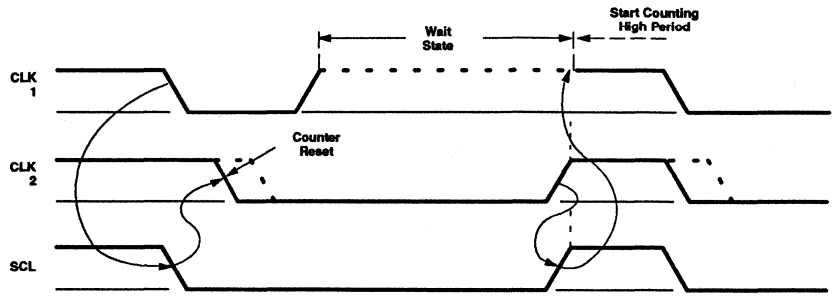


Figure 7. Clock Synchronization During the Arbitration Procedure

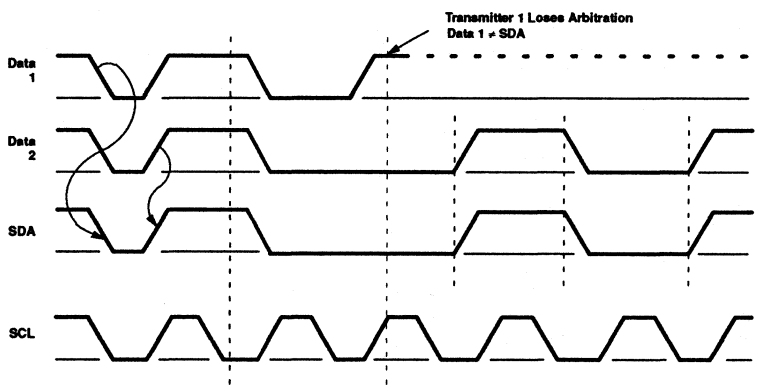


Figure 8. Arbitration Procedure of Two Masters

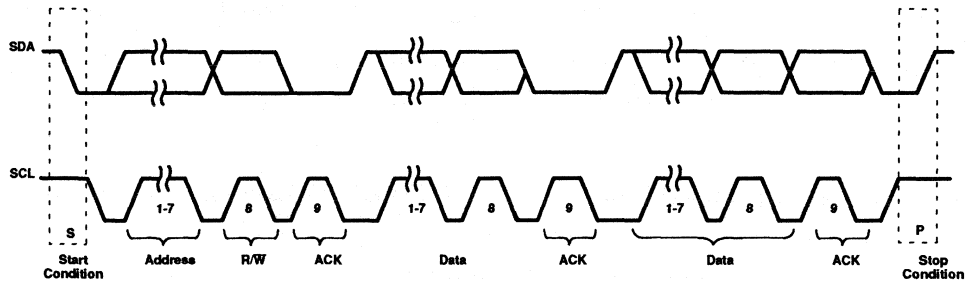
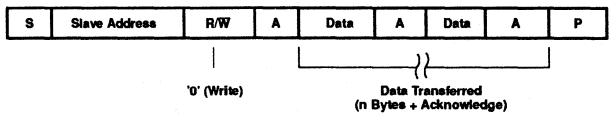


Figure 9. Complete Data Transfer

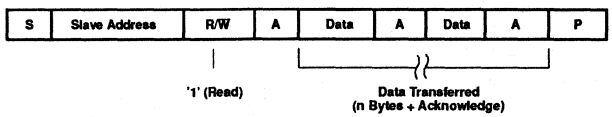
Possible Data Transfer Formats Are:

a) Master transmitter transmits to slave receiver. Direction is not changed.

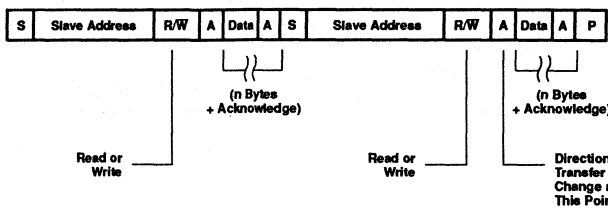
A = Acknowledge
S = Start
P = Stop



b) Master reads slave immediately after first byte.



c) Combined formats.



NOTES:

1. Combined formats can be used, for example, to control a serial memory. During the first data byte, the internal memory location has to be written. After the start condition is repeated, data can then be transferred.
2. All decisions on auto-increment or decrement of previously accessed memory locations, etc., are taken by the designer of the device.
3. Each byte is followed by an acknowledge as indicated by the A blocks in the sequence.
4. I^2C devices have to reset their bus logic on receipt of a start condition so that they all anticipate the sending of a slave address.

ADDRESSING

The first byte after the start condition determines which slave will be selected by the master. Usually, this first byte follows that start procedure. The exception is the general call address which can address all devices. When this address is used, all devices should, in theory, respond with an acknowledge, although devices can be made to ignore this address. The second byte of the general call address then defines the action to be taken.

Definition of Bits in the First Byte

The first seven bits of this byte make up the slave address (Figure 10). The eighth bit (LSB—least significant bit) determines the direction of the message. A zero on the least significant position of the first byte means that the master will write information to a selected slave; a one in this position means that the master will read information from the slave.

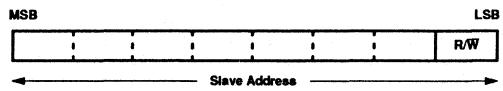


Figure 10. The First Byte After the Start Procedure

When an address is sent, each device in a system compares the first 7 bits after the start condition with its own address. If there is a match, the device will consider itself addressed by the master as a slave receiver or slave transmitter, depending on the R/W bit.

The slave address can be made up of a fixed and a programmable part. Since it is expected that identical ICs will be used more than once in a system, the programmable part of the slave address enables the maximum possible number of such devices to be connected to the I^2C bus. The number of programmable address bits of a device depends on the number of pins available. For example, if a device has 4 fixed and 3 programmable address bits, a total of eight identical devices can be connected to the same bus.

The I^2C bus committee is available to coordinate allocation of I^2C addresses.

The bit combination 1111XXX of the slave address is reserved for future extension purposes.

The address 1111111 is reserved as the extension address. This means that the addressing procedure will be continued in the next byte(s). Devices that do not use the extended addressing do not react at the reception of this byte. The seven other possibilities in group 1111 will also only be used for extension purposes but are not yet allocated.

The combination 0000XXX has been defined as a special group. The following addresses have been allocated:

FIRST BYTE			
SLAVE ADDRESS		R/W	
0000	000	0	General call address
0000	000	1	Start byte
0000	001	X	CBUS address
0000	010	X	Address reserved for different bus format
0000	011	X	
0000	100	X	To be defined
0000	101	X	To be defined
0000	110	X	To be defined
0000	111	X	To be defined

No device is allowed to acknowledge at the reception of the start byte.

The CBUS address has been reserved to enable the intermixing of CBUS and I^2C devices in one system. I^2C bus devices are not allowed to respond at the reception of this address.

The address reserved for a different bus format is included to enable the mixing of I^2C and other protocols. Only I^2C devices that are able to work with such formats and protocols are allowed to respond to this address.

General Call Address

The general call address should be used to address every device connected to the I^2C bus. However, if a device does not need any of the data supplied within the general call structure, it can ignore this address by not acknowledging. If a device does require data from a general call address, it will acknowledge this address and behave as a slave receiver. The second and following bytes will be acknowledged by every slave receiver capable of handling this data. A slave which cannot process one of these bytes must ignore it by not acknowledging.
The meaning of the general call address is always specified in the second byte (Figure 11).

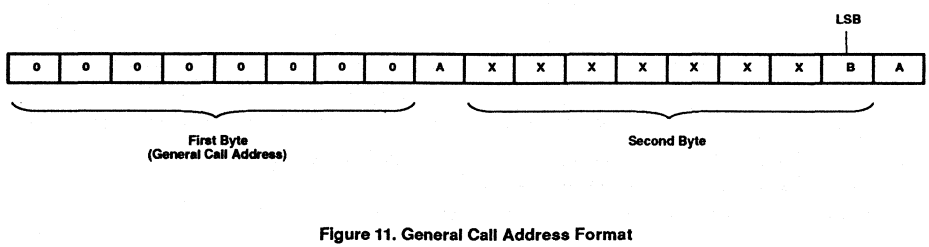


Figure 11. General Call Address Format

There are two cases to consider:
1. When the least significant bit is a zero.
2. When the least significant bit is a one.

When B is a zero, the second byte has the following definition:

00000110 (H'06') Reset and write the programmable part of slave address by software and hardware. On receiving this two-byte sequence, all devices (designed to respond to the general call address) will reset and take in the programmable part of their address.
Precautions must be taken to ensure that a device is not pulling down the SDA or SCL line after applying the supply voltage, since these low levels would block the bus.

00000010 (H'02') Write slave address by software only. All devices which obtain the programmable part of their address by software (and which have been designed to respond to the general call address) will enter a mode in which they can be programmed. The device will not reset.

An example of a data transfer of a programming master is shown in Figure 12 (ABCD represents the fixed part of the address).

00000100 (H'04') Write slave address by hardware only. All devices which define the programmable part of their address by hardware (and which respond to the general call address) will latch this programmable part at the reception of this two-byte sequence. The device will not reset.

00000000 ('H00') This code is not allowed to be used as the second byte.

Sequences of programming procedure are published in the appropriate device data sheets.

The remaining codes have not been fixed and devices must ignore these codes.

When B is a one, the two-byte sequence is a hardware general call. This means that the sequence is transmitted by a hardware master device, such as a keyboard scanner, which cannot be programmed to transmit a desired slave address. Since a hardware master does not know in advance to which device the message must be transferred, it can only generate this hardware general call and its own address, thereby identifying itself to the system (Figure 13).

The seven bits remaining in the second byte contain the device address of the hardware master. This address is recognized by an intelligent device, such as a microcomputer, connected to the bus which will then direct the information coming from the hardware master. If the hardware master can also act as a slave, the slave address is identical to the master address.

In some systems an alternative could be that the hardware master transmitter is brought in the slave receiver mode after the system reset. In this way, a system configuring master can tell the hardware master transmitter (which is now in slave receiver mode) to which address data must be sent (Figure 14). After this programming procedure, the hardware master remains in the master transmitter mode.

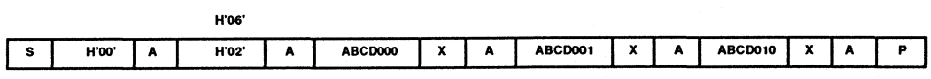


Figure 12. Sequence of a Programming Master

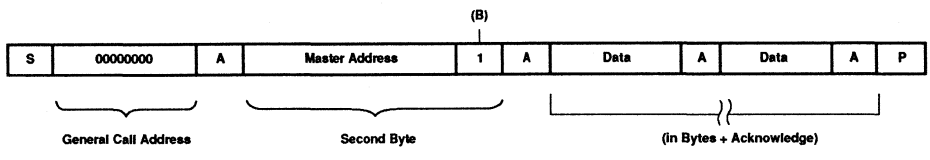


Figure 13. Data Transfer From Hardware Master Transmitter

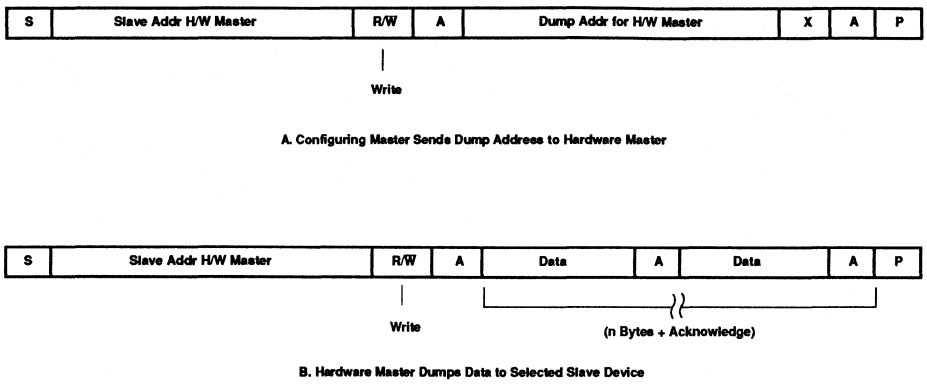


Figure 14. Data Transfer From Hardware Master Transmitter Capable of Dumping Data Directly to Slave Devices

Start Byte

Microcomputers can be connected to the I^2C bus in two ways. If an on-chip hardware I^2C bus interface is present, the microcomputer can be programmed to be interrupted only by requests from the bus. When the device possesses no such interface, it must constantly monitor the bus via software. Obviously, the more times the microcomputer monitors, or polls, the bus, the less time it can spend carrying out its intended function.

Therefore, there is a difference in speed between fast hardware devices and the relatively slow microcomputer which relies on software polling.

In this case, data transfer can be preceded by a start procedure which is much longer than normal (Figure 15). The start procedure consists of:

1. A start condition, (S)
2. A start byte 00000001
3. An acknowledge clock pulse
4. A repeated start condition, (Sr)

After the start condition (S) has been transmitted by a master requiring bus access, the start byte (00000001) is transmitted. Another microcomputer can therefore sample the SDA line on a low sampling rate until one of the seven zeros in the start byte is detected. After detection of the Low level on the SDA line, the microcomputer is then able to switch to a higher sampling rate in order to find the second start condition (Sr), which is then used for synchronization.

A hardware receiver will reset at the reception of the second start condition (Sr) and will therefore ignore the start byte.

After the start byte, an acknowledge-related clock pulse is generated. This is present only to conform with the byte-handling format used on the bus. No device is allowed to acknowledge the start byte.

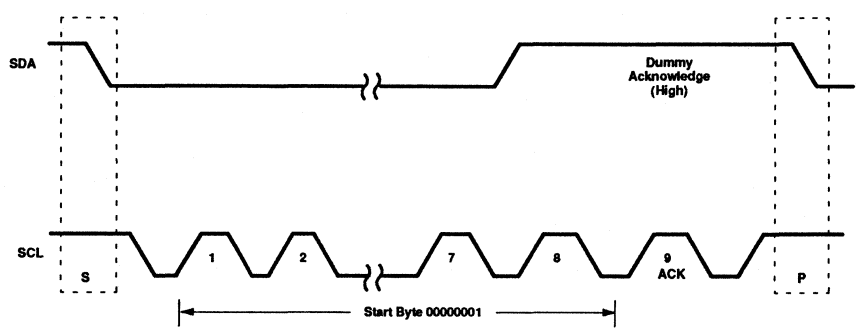


Figure 15. Start Byte Procedure

CBUS Compatibility

Existing CBUS receivers can be connected to the I^2C bus. In this case, a third line, called DLEN, has to be connected and the acknowledge bit omitted. Normally, I^2C transmissions are multiples of 8-bit bytes; however, CBUS devices have different format.

In a mixed bus structure, I^2C devices are not allowed to respond on the CBUS message. For this reason, a special CBUS address (0000001X) has been reserved. No I^2C device will respond to this address. After the transmission of the CBUS address, the DLEN line can be made active and transmission, according to the CBUS format, can be performed (Figure 16).

After the stop condition, all devices are again ready to accept data.

Master transmitters are allowed to generate CBUS formats after having sent the CBUS address. Such a transmission is terminated by a stop condition, recognized by all devices. In the low speed mode, full 8-bit bytes must always be transmitted and the timing of the DLEN signal adapted.

If the CBUS configuration is know and no expansion with CBUS devices is foreseen, the user is allowed to adapt the hold time to the specific requirements of device(s) used.

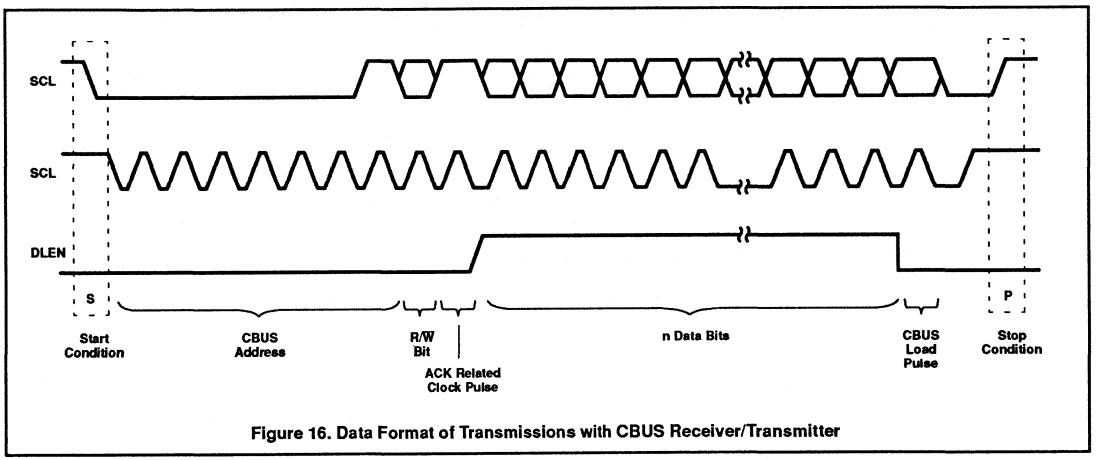


Figure 16. Data Format of Transmissions with CBUS Receiver/Transmitter

ELECTRICAL SPECIFICATIONS OF INPUTS AND OUTPUTS OF I^2C DEVICES

The I^2C bus allows communications between devices made in different technologies which might also use different supply voltages.

For devices with fixed input levels, operating on a supply voltage of +5V ±10%, the following levels have been defined:

V_{ILmax} =1.5V (maximum input Low voltage)
V_{IHmin} =3V (minimum input High voltage)

Devices operating on a fixed supply voltage different from +5 (e.g., I^2L) must also have these input levels of 1.5V and 3V for V_{IL} and V_{IH}, respectively.

For devices operating over a wide range of supply voltages (e.g., CMOS), the following levels have been defined:

V_{ILmax} =0.3V_{DD} (maximum input Low voltage)
V_{IHmin} =0.7V_{DD} (minimum input High voltage)

For both groups of devices, the maximum output Low value has been defined:

V_{OLmax} =0.4V (max. output voltage Low) at 3mA sink current)

The maximum low-level input current at V_{OLmax} of both the SDA pin and the SCL pin of an I^2C device is –10μA, including the leakage current of a possible output stage.

The maximum high-level input current at 0.9V_{DD} of both the SDA pin and SCL pin of an I^2C device is 10μA, including the leakage current of a possible output stage.

The maximum capacitance of both the SDA pin and the SCL pin of an I^2C device is 10pF.

Devices with fixed input levels can each have their own power supply of +5V ±10%. Pull-up resistors can be connected to any supply (see Figure 17).

However, the devices with input levels related to V_{DD} must have one common supply line to which the pull-up resistor is also connected (see Figure 18).

When devices with fixed input levels are mixed with devices with V_{DD}-related levels, the latter devices have to be connected to one common supply line of +5V ±10% along with the pull-up resistors (Figure 19).

Input levels are defined in such a way that:

1. The noise margin on the Low level is 0.1 V_{DD}.
2. The noise margin on the High level is 0.2 V_{DD}.
3. Series resistors (R_S) up to 300Ω can be used for flash-over protection against high voltage spikes on the SDA and SCL line (due to flash-over of a TV picture tube, for example) (Figure 20).

The maximum bus capacitance per wire is 400pF. This includes the capacitance of the wire itself and the capacitance of the pins connected to it.

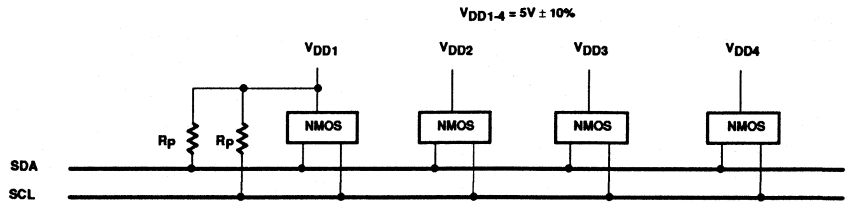


Figure 17. Fixed Input Level Devices Connected to the I^2C Bus

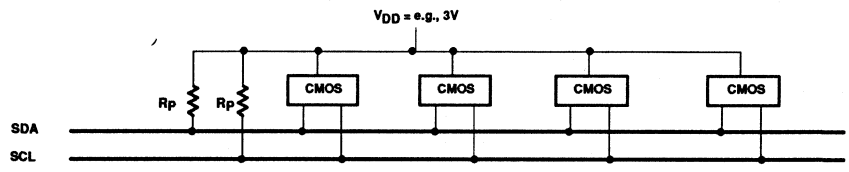


Figure 18. Devices with a Wide Range of Supply Voltages Connected to the I^2C Bus

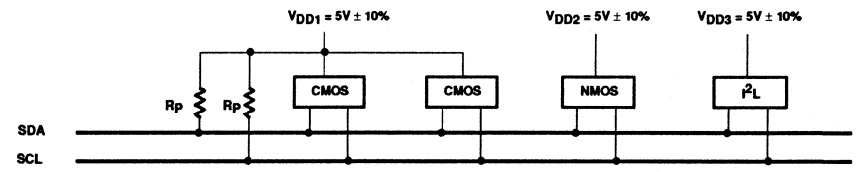


Figure 19. Devices with V_{DD} Related Levels Mixed with Fixed Input Level Devices on the I^2C Bus

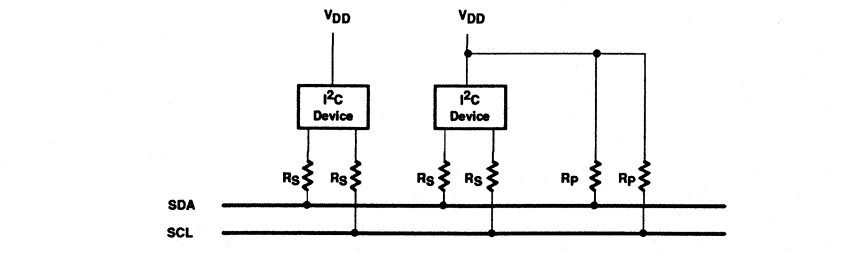


Figure 20. Serial Resistors (R_S) for Protection Against High Voltage

TIMING

The clock on the I^2C bus has a minimum Low period of 4.7µs and a minimum High period of 4µs. Masters in this mode can generate a bus clock with a frequency from 0 to 100kHz.

All devices connected to the bus must be able to follow transfers with frequencies up to 100kHz, either by being able to transmit or receive at that speed or by applying the clock synchronization procedure which will force the master into a wait state and stretch the Low periods. In the latter case the frequency is reduced.

Figure 21 shows the timing requirements in detail. A description of the abbreviations used is shown in Table 2. All timing references are at V_{ILmax} and V_{ILmin}.

LOW-SPEED MODE

As explained previously, there is a difference in speed on the I^2C bus between fast hardware devices and the relatively slow microcomputer which relies on software polling. For this reason a low speed mode is available on the I^2C bus to allow these microcomputers to poll the bus less often.

Start and Stop Conditions

In the low-speed mode, data transfer is preceded by the start procedure.

Data Format and Timing

The bus clock in this mode has a Low period of 130µs +25µs and a High period of 390µs +25µs, resulting in a clock frequency of approximately 2kHz. The duty cycle of the clock has this Low-to-High ratio to allow for more efficient use of microcomputers without an on-chip hardware I^2C bus interface. In this mode also, data transfer with acknowledge is obligatory. The maximum number of bytes transferred is not limited (Figure 22).

In this mode, a transfer cannot be terminated during the transmission of a byte.

The bus is considered busy after the first start condition. It is considered free again one minimum clock Low period, 105µs, after the detection of the stop condition. Figure 23 shows the timing requirements in detail Table 3 explains the abbreviations.

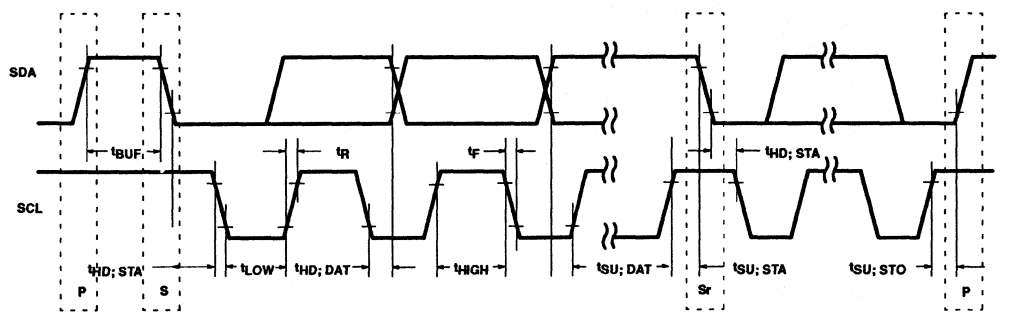


Figure 21. Timing Requirements for the I^2C Bus

Table 2. Timing Requirement for the I^2C Bus

SYMBOL	PARAMETER	LIMITS		UNIT
		MIN	MAX	
f_{SCL}	SCL clock frequency	0	100	kHz
t_{BUF}	Time the bus must be free before a new transmission can start	4.7		µs
$t_{HD;STA}$	Hold time start condition. After this period the first clock pulse is generated	4		µs
t_{LOW}	The Low period of the clock	4.7		µs
t_{HIGH}	The High period of the clock	4		µs
$t_{SU;STA}$	Setup time for start condition (only relevant for a repeated start condition)	4.7		µs
$t_{HD;DAT}$	Hold time DATA for CBUS compatible masters for I^2C devices	 5 0*		 µs µs
$t_{SU;DAT}$	Setup time DATA	250		ns
t_R	Rise time of both SDA and SCL lines		1	µs
t_F	Fall time of both SDA and SCL lines		300	ns
$t_{SU;STO}$	Setup time for stop condition	4.7		µs

NOTES:
All values referenced to V_{IH} and V_{IL} levels.
* Note that a transmitter must internally provide a hold time to bridge the undefined region (300ns max.) of the falling edge of SCL.

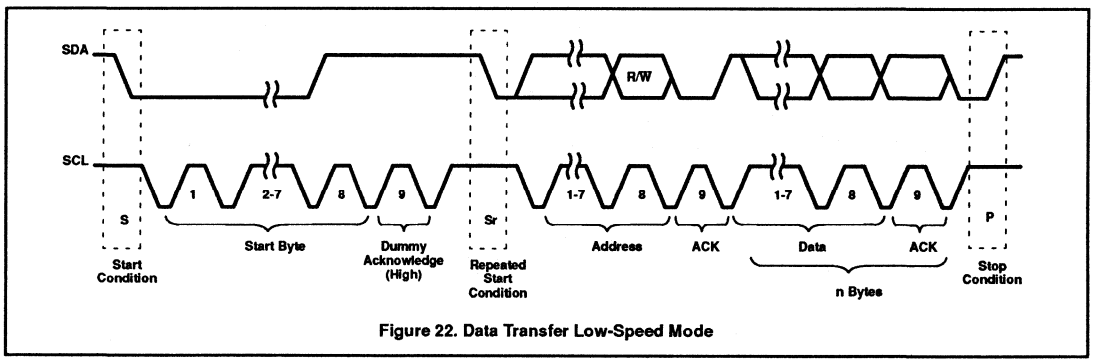


Figure 22. Data Transfer Low-Speed Mode

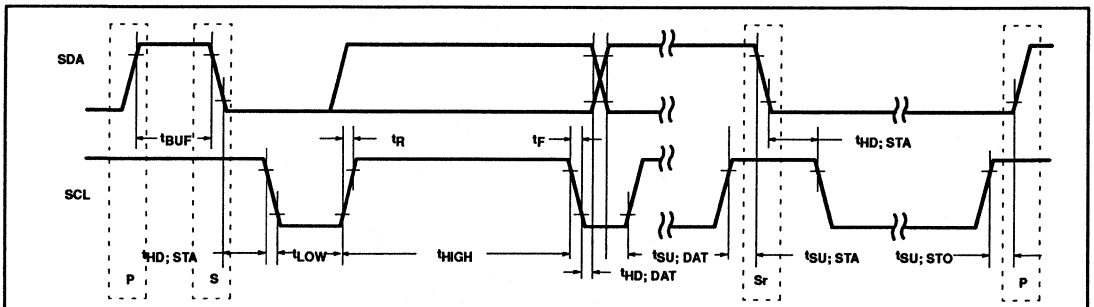


Figure 23. Timing Low-Speed Mode

LOW SPEED MODE

CLOCK	: t_{LOW} = 130μs ± 25μs
DUTY CYCLE	: t_{HIGH} = 390μs ± 25μs
	: 1:3 Low-to-High (Duty cycle of clock generator)
START BYTE	: 0000 0001
MAX. NO. OF BYTES	: UNRESTRICTED
PREMATURE TERMINATION OF TRANSFER	: NOT ALLOWED
ACKNOWLEDGE CLOCK BIT	: ALWAYS PROVIDED
ACKNOWLEDGMENT OF SLAVES	: OBLIGATORY

Table 3. Timing Low-Speed Mode

SYMBOL	PARAMETER	LIMITS		UNIT
		MIN	MAX	
t_{BUF}	Time the bus must be free before a new transmission can start	105		μs
$t_{HD;STA}$	Hold time start condition. After this period the first clock pulse is generated	365		μs
$t_{HD;STA}$	Hold time (repeated start condition only)	210		μs
t_{LOW}	The Low period of the clock	105	155	μs
t_{HIGH}	The High period of the clock	365	415	μs
$t_{SU;STA}$	Setup time for start condition (only relevant for a repeated start condition)	105	155	μs
$t_{HD;DAT}$	Hold time DATA for CBUS compatible masters for I^2C devices	 5 0*		 μs μs
$t_{SU;DAT}$	Setup time DATA	250		ns
t_R	Rise time of both SDA and SCL lines		1	μs
t_F	Fall time of both SDA and SCL lines		300	ns
$t_{SU;STO}$	Setup time for stop condition	105	155	μs

NOTES:
All values referenced to V_{IH} and V_{IL} levels.
* Note that a transmitter must internally provide a hold time to bridge the undefined region (300ns max.) of the falling edge of SCL.

APPENDIX 1

Maximum and minimum values of the pull-up resistors R_P and series resistors R_S (see Figure 20).

In an I^2C bus system these values depend on the following parameters:

- Supply voltage
- Bus capacitance
- Number of devices (input current + leakage current)

1. The supply voltage limits the minimum value of the R_P resistor due to the specified 3mA as minimum sink current of the output stages, at 0.4V as maximum low voltage. In graph 1, V_{DD} against R_{Pmin} is shown.

The desired noise margin of 0.1 V_{DD} for the low level limits the maximum value of R_S.

In Graph 2, R_{Smax} against R_P is shown.

2. The bus capacitance is the total capacitance of wire, connections, and pins. This capacitance limits the maximum value of R_P because of the specified rise time of 1µs.

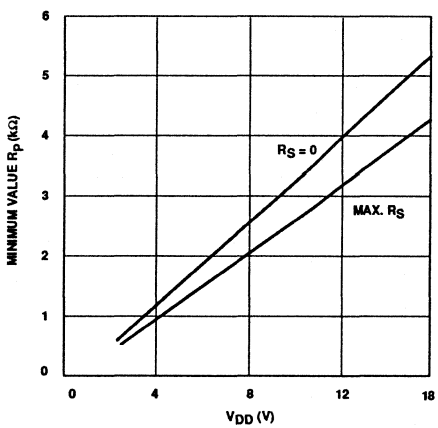
In Graph 3, the bus capacitance-R_{Pmax} relationship is shown.

3. The maximum high-level input current of each input/output connection has a specified value of 10µA max. Due to the desired noise margin of 0.2 V_{DD} for the high level, this input current limits the maximum value of R_P. This limit is dependent on V_{DD}.

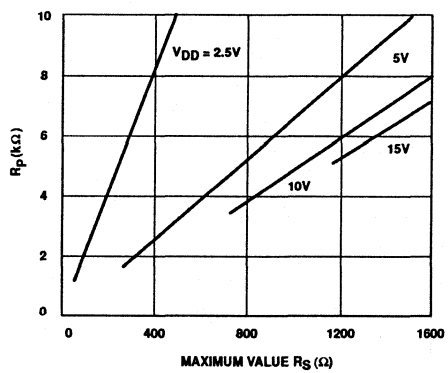
In Graph 4 the total high-level input current-R_{Pmax} relationship is shown.

I^2C LICENSE

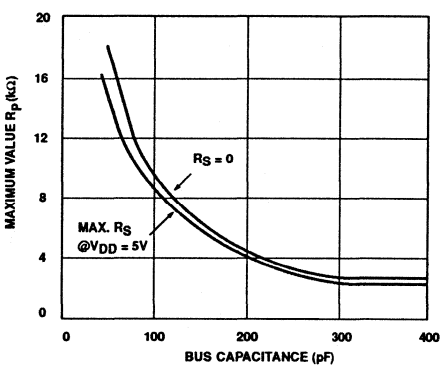
Purchase of Signetics or Philips I^2C components conveys a license under the Philips I^2C patent rights to use these components in an I^2C system, provided that the system conforms to the I^2C standard specification as defined by Philips.



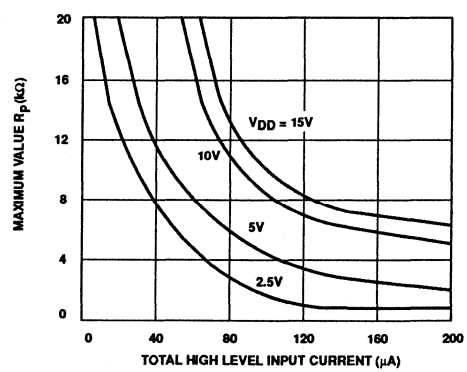
Graph 1



Graph 2



Graph 3



Graph 4

I^2C ADDRESS ALLOCATION TABLE

A6–A3	A2–A0							
	0	1	2	3	4	5	6	7
0	General call address	Reserved—	———	———	———	———	———	⟶
1								
2	PCF8200* SAF1135:—	SAA5243* ——— SAA5245*	——— SAA9068!—	⟶ ⟶	SAA9020*	———	———	⟶
3							SAA1136!	
4	SAA1300!— TDA8444!— PCF8574!—	——— ——— ———	——— ——— ———	⟶ ——— ———	 ——— ——— PCD3311/A!— PCD3312!—	 ——— ——— ⟶ ⟶	 ——— ——— SA3028	 ⟶ ⟶
5								
6								
7	PCF8576!— PCF8574*— SAA1064!—	⟶ ——— ———	PCF8577!	PCF8577A! ——— ⟶	PCF8578* PCF8579*	⟶ ⟶ ———	PCF8566 !— ———	⟶ ⟶
8	TDA8420!— TDA8421!— TEA6300/T!— TEA6310T!—	⟶ ⟶ TDA8425!	TDA8045*		TDA8422! TDA8461!—	SAA9050* ⟶ SAA9051*	SAA9062* SAA9063* SAA9064*	
9	TDA8440!— PCF8591*—	——— ———	——— ———	——— ———	——— ———	——— ———	——— ———	⟶ ⟶
A	PCF8583*— PCF8570*— PCF8571*— PCF8572*— PCF8582A!—	⟶ ——— ——— ——— ———	 ——— ——— ——— ———	 ——— ——— ——— ———	 ——— ——— ——— ———	 ——— ——— ——— ———	 ——— ——— ——— ———	 ⟶ ⟶ ⟶ ⟶
B	PCF8570*—	———	———	———	———	———	———	⟶
C	TSA5510*— SAB3035*— SAB3036*— SAB3037*— TDA8400!—	——— ——— ——— ——— TEA6000*— TEA6100*— ———	——— ——— ——— ——— TSA6057!—	⟶ ⟶ ⟶ ⟶ ⟶ ⟶				
D	TDA8433!— TDA8443A!— TDA8573*—	——— ——— ———	——— ——— ———	——— ——— ⟶	——— ———	——— ———	——— ———	⟶ ⟶
E								
F	Reserved—	———	———	———	———	———	———	⟶

Address Format: | A6 | A5 | A4 | A3 | A2 | A1 | A0 | R/W | Legend: * = R/W, ! = W, : = R

To find a part's address, the most significant 4 bits (A6, A5, A4, and A3) are read from the side of the table. The least significant 3 bits (A2, A1, and A0) are read from the top of the table. For example: SAA1136 is at 36H (0011 110), PCF8577 is at 72H (0111 010). Parts with arrows indicate that a portion of the address is user-configurable. For example: PCF8576 is at address 70H (0111 000) but the last bit (A0) can be set high or low by the user so it can also be at address 71H (0111 001).

ASSIGNED I^2C BUS ADDRESSES

PART NUMBER	FUNCTION	I^2C ADDRESS						
		A6	A5	A4	A3	A2	A1	A0
—	General call address	0	0	0	0	0	0	0
—	Reserved addresses	0	0	0	0	X	X	X
PCF8200	Voice synthesizer (male or female)	0	0	1	0	0	0	0
SAA5243/45	Enhanced teletext circuit	0	0	1	0	0	0	1
SAF1135	Data line decoder	0	0	1	0	0	A	A
SAA9068	(PIPCO) Picture-in-picture controller	0	0	1	0	0	1	A
SAA9020	Field memory controller	0	0	1	0	1	A	A
SAA1136	PCM-Aufio indent-word interface	0	0	1	1	1	1	0
SAA1300	5-bit high current driver	0	1	0	0	0	A	A
TDA8444	Octuple 6-bit DAC	0	1	0	0	A	A	A
PCF8574	I^2C bus to 8-bit bus converter	0	1	0	0	A	A	A
PCD3311/12	Tone generator DTMF/modem/musical	0	1	0	0	1	0	A
SAA3028	Transcoder (RC-5) for IR remote	0	1	0	0	1	1	0
PCF8576	160-segment LCD driver 1:1–1:4 Mux	0	1	1	1	0	0	A
SAA1064	4-digit LED driver	0	1	1	1	0	A	A
PCF8574A	I^2C bus to 8-bit bus converter	0	1	1	1	A	A	A
PCF8577	64-segment LCD driver 1:1–1:2 Mux	0	1	1	1	0	1	0
PCF8577A	64-segment LCD driver 1:1–1:2 Mux	0	1	1	1	0	1	1
PCF8578	Row/column LCD dot-matrix driver	0	1	1	1	1	0	A
PCF8579	Row/column LCD dot-matrix driver	0	1	1	1	1	0	A
PCF8566	96-segment LCD driver 1:1–1:4 Mux	0	1	1	1	1	1	A
TEA6300/6310T	Sound fader control circuit	1	0	0	0	0	0	0
TDA8420/21	Hi-fi stereo audio processor	1	0	0	0	0	0	A
TDA8425	Audio processor w/loudspeaker channel	1	0	0	0	0	0	1
TDA8405	TV and VTR stereo sound processor	1	0	0	0	0	1	0
TDA8442	Interface for color decoders	1	0	0	0	1	0	0
TDA8461	PAL/NTSC color decoder	1	0	0	0	1	0	A
SAA9050/51	Digital multi-standard TV decoder	1	0	0	0	1	0	1
SAA9062/63/64	Digital deflection controller	1	0	0	0	1	1	0
TDA8440	Switch for CTV receivers	1	0	0	1	A	A	A
PCF8591	4-channel, 8-bit A/D plus 8-bit D/A	1	0	0	1	A	A	A
PCF8583	256 x 8 RAM with clock/calendar	1	0	1	0	0	0	A
PCF8570/71	256 x 8, 128 x 8 static RAM	1	0	1	0	A	A	A
PCF8572	128 x 8 EEPROM	1	0	1	0	A	A	A
PCF8582A	256 x 8 EEPROM	1	0	1	0	A	A	A
PCF8570C	256 x 8 static RAM	1	0	1	1	A	A	A
TEA6000/6100	FM/IF and tuning interface	1	1	0	0	0	0	1
TSA5510	PLL frequency synthesizer for TV	1	1	0	0	0	A	A
SAB3035/36/37	(CITAC) CPU interface for tuning and control	1	1	0	0	0	A	A
TDA8400	Computer interface prescaler-synthesizer	1	1	0	0	0	A	A
TSA6057	Radio tuning PLL frequency synthesizer	1	1	0	0	0	1	A
PCF8573	Clock/calendar	1	1	0	1	0	A	A
TDA8443	YUV/RGB interface circuit	1	1	0	1	A	A	A
—	Reserved addresses	1	1	1	1	X	X	X

X = Don't care.
A = Can be connected high or low by the user.

I^2C PERIPHERAL SELECTION GUIDE

General Purpose ICs

LCD Drivers

PCF8566: 96-segment LCD driver 1:1 - 1:4 Mux

PCF8576: 160-segment LCD driver 1:1 - 1:4 Mux

PCF8577A: 64-segment LCD driver 1:1 - 1:2 Mux

PCF8578/79: Row/column LCD dot-matrix driver; 1:8 - 1:32 Mux

I/O Expanders

PCF8574/A: 8-bit remote I/O port (I^2C bus to parallel converter)

PCF8584: 8-bit parallel to I^2C converter

SAA1064: 4-digit LED driver

SAA1300: 5-bit high-current driver

Data Converters

PCF8591: 4-channel, 8-bit Mux ADC + one DAC

TDA8442: Quad 6-bit DAC

TDA8444: Octal 6-bit DAC

Memory

PCF8570/C: 256-byte static RAM

PCF8571: 128-byte static RAM

PCF8581: 128-byte EEPROM*

PCF8582A: 256-byte EEPROM

PCF8583: 256-byte RAM/clock/calendar

Clock/Calendars

PCF8573: Clock/calendar

PCF8583: Clock/calendar/256-byte RAM

68000-Based CMOS µControllers

68070: 68000 CPU/MMU/UART/DMA/timer

80C51-Based CMOS µControllers+

8XCL410: 4k ROM/128 RAM, low power

8XC528: 32k ROM/512 RAM, T2, WD

8XC552: 256-byte RAM/8k ROM/ADC/UART/PWM

8XC652: 256-byte RAM/8k ROM, UART

8XC654: 256-byte RAM/16k ROM, UART

8XC751: 64-byte RAM/2k ROM

8XC752: 64-byte RAM/2k ROM, ADC/PWM

8048 Instruction-Set Based CMOS µControllers

PCF84C00: 256-byte RAM/ bond-out version for prototype development

PCF84C21: 64-byte RAM/2k ROM

PCF84C41: 128-byte RAM/4k ROM

PCF84C81: 256-byte RAM/8k ROM

PCF84C85: 256-byte RAM/8k ROM/ Extended I/O

PCF84C430: 128-byte RAM/4k ROM/ 96-segment LCD driver*

Application-Oriented ICs

Video/Radio/Audio

PCF8200: Voice synthesizer (male/female speech)

SAA1300: Tuner switching circuit

SAA3028: Transcoder (RC-5) for IR remote control

SAA4700: Dataline processor for VPS

SAA5243: Enhanced Computer Controlled Teletext (ECCT) decoder

SAA5245: Enhanced Computer Controlled Teletext (USECCT) decoder

SAA9041: Digital video teletext (DVTB) processor

SAA9050: Digital PAL/NTSC color decoder

SAA9055: Digital SECAM decoder

SAA9068: Picture-in-picture controller

SAB3035/36/37: Digital tuning circuits for computer-controlled TV

SAF1135: Dataline 16 decoder for VCR

TDA8370: Synchronization processor for TV

TDA8405: Stereo/dual sound processor

TDA8420/21: Audio processor with a loudspeaker channel and a headphone channel

TDA8425: Audio processor with a loudspeaker channel only

TDA8440: Video/audio switch

TDA8442: Interface for color decoders

TDA8443/A: YUV/RGB matrix switch

TDA8461: PAL/NTSC color decoder and RGB processor

TEA6000/6100: FM/IF and digital tuning IC for computer-controlled radio

TEA6300: Sound fader control and preamplifier/source selector for car radio

TEA6310T: Sound fader control with tone and volume control for car radio

TSA5510: PLL frequency synthesizer for radio

TSA6057: PLL frequency synthesizer for radio

Telecom

NE5750/51: Audio processor pair*

PCD3311/12: Tone generator (DTMF/modem/musical)

PCD3341: Advanced 10 to 110-number repertory dialer with LCD control

PCD3343: Microntroller with 224-byte RAM/3k ROM

PCD3348: Microntroller with 256-byte RAM/8k ROM

UMA1000T: Data processor for mobile telephones*

UMA1010T: 1GHz frequency synthesizer for mobile telephones*

*Future Device

+Also available with extended temperature ranges.

Section 3
80C51 Family Derivatives

CONTENTS

8XC52 Overview 150
Differences from the 80C51 150
Program Memory 150
Special Function Registers 150
Timer/Counters 150
Serial Port 151
Timer/Counter 2 Set-Up 155
Using Timer/Counter 2 to Generate Baud Rates 155
Interrupts 155
Port Structures 155

8032AH/8052AH Data Sheet 160

80C32/80C52/87C52 Data Sheet 169

8XC053/54 Overview 183
Differences from the 80C51 183
Special Function Registers 183
Memory Organization 183
On-Screen Display Module (OSD) 183
Pulse Width Modulators 183
Software A/D 185
Reduced Power Modes 185
Interrupts 185

83C053, 87C054 Data Sheet 186

8XCL410 Overview 201
Differences from the 80C51 201
Special Function Registers 201
I^2C Serial Interface 201
The Interrupt Structure 204
Oscillator 205
Power-Down Mode 205
Low Power Consumption 205

80CL410/83CL410 Data Sheet 208

8XC451 Overview 225
Differences from the 80C51 225
Special Function Registers 225
I/O Port Structure 225
Processor Bus Interface 226
Standard Quasi-Bidirectional I/O Port 226
Parallel Printer Port 226

80C451/83C451/87C451 Data Sheet 228

8XC528 Overview 245
Differences from the 80C51 245
Data Memory 245
Special Function Registers 246
Watchdog Timer 246
I^2C Interface 248
Interrupts 249
Idle Mode 249
Power-Down Mode 249

80C528/83C528/87C528 Data Sheet 250

8XC550 Overview 265
Differences from the 8C051 265
Special Function Registers 265
I/O Port Structure 265
A/D Converter 265
Sample A/D Routines 267

Watchdog Timer 267
Interrupts 268
Interrupt Control Registers 268
Power-Down and Idle Modes 269

80C550/83C550/87C550 Data Sheet 270

8XC552 Overview 288
83C562 Overview 288
Differences from the 80C51 288
Program Memory 288
Data Memory 288
Special Function Registers 288
Timer T2 289
Timer T3, the Watchdog Timer 295
Serial I/O 296
Reset Circuitry 329
Interrupts 329
I/O Port Structure 333
Port 1 Operation 333
Port 5 Operation 333
Pulse Width Modulated Outputs 333
Analog-to-Digital Converter 333
Power Reduction Modes 339
Memory Organization 340

80C552/83C552/87C552 Data Sheet 345

80C562/83C562/87C562 Data Sheet 365

8XC652/654 Overview 378
Differences from the 80C51 378
Special Function Registers 378
I^2C Serial Communication—SIO1 378
Idle and Power-Down Operation 378
Interrupt System 380

80C652/83C652/87C652 Data Sheet 382

83C654/87C654 Data Sheet 398

8XC751 Overview 414
Differences from the 80C51 414
Instruction Set 414
Memory Organization 414
Special Function Registers 414
Data Pointer (DPTR) 414
I/O Port Latches (P0, P1, P3) 414
I/O Port Structure 416
Timer/Counter 416
I^2C Serial Interface 416
I^2C Interrupts 417
I^2C Register I2CON 417
I^2C Register I2DAT 418
I^2C Register I2CFG 418
I^2C Register I2STA 419
Interrupts 419
Interrupt Enable Register 420

83C751/87C751 Data Sheet 421

8XC752 Overview 431
Differences from the 80C51 431
Instruction Set 431

Memory Organization 431
I/O Ports 431
PWM Outputs 431
A/D Converter 433
Counter/Timer 434
I²C Serial I/O 435
Interrupts 435
Power-Down and Idle Modes 435
Special Function Registers 435
Data Pointer 435

83C752/87C752 Data Sheet 436

8XC851 Overview 449
Differences from the 80C51 449
Special Function Registers 449

80C851/83C851 Data Sheet 451

Section 3
80C51 Family Derivatives

8XC52 OVERVIEW

The 8XC52 is identical to the 8XC51, respectively, except for added features. The 8XC52 has:

- 8k ROM (80C52 only)
- 256 bytes RAM
- Counter/timer 2

As a result, there are some additions to the interrupt structure, I/O pin alternate functions, and baud rate generation for the serial channel.

Since the similarity is so close, only the additional features of the 8XC52 are described. Where necessary, some repetition of 80C51 information will be made for the sake of clarity. The 8XC52 is pin for pin and fully code compatible with the 80C51.

Differences From the 80C51

Program Memory

The data and program memory are organized virtually identically to the 80C51. The 80C52 possesses 8k bytes of on-chip program memory. When $\overline{EA}$ is high, the 80C52 fetches instructions from the internal ROM unless the address exceeds 1FFFh. Locations 2000H to FFFFH are fetched from external program memory. When $\overline{EA}$ is held low all instruction fetches are from external memory. The program memory space is shown in Figure 1.

The data memory organization is identical to the 80C51 except that the 80C52 has an additional 128 bytes of RAM overlapped with the special functio register space. This additional RAM is addressed using indirect addressing only and is available as stack space. The 80C52 data memory space is shown in Figure 2.

Special Function Registers

The special function register space is the same as the 80C51 except that the 80C52 contains the additional special function registers T2CON, RCAP2L, RCAP2H, TL2, and TH2. Since the standard 80C51 on-chip functions are identical in the 80C52, the SFR locations, bit locations, and operation are likewise identical. The only exceptions are in the interrupt mode and interrupt priority SFRs (see Table 1).

Timer/Counters

In addition to timer/counters 0 and 1 of the 80C51, the 80C52 contains timer/counter 2. Like timers 0 and 1, timer 2 can operate as either an event timer or as an event counter. This is selected by bit C/T2 in the special function register T2CON (see Figure 3). It has three operating modes: capture, auto-load, and baud rate generator, which are selected by bits in the T2CON as shown in Table 2.

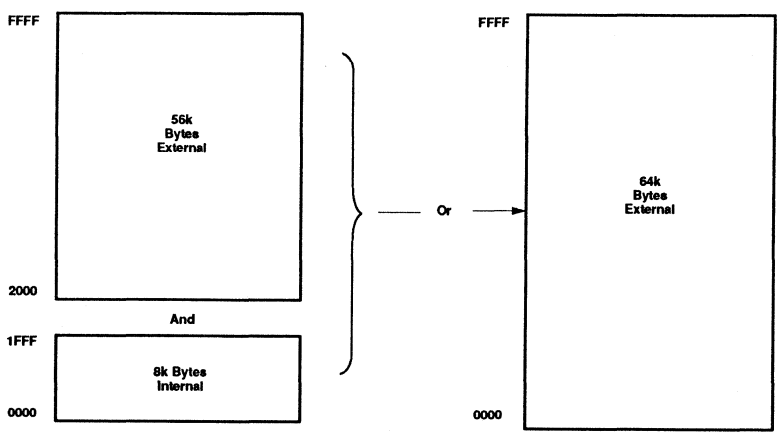


Figure 1. 80C52 Program Memory

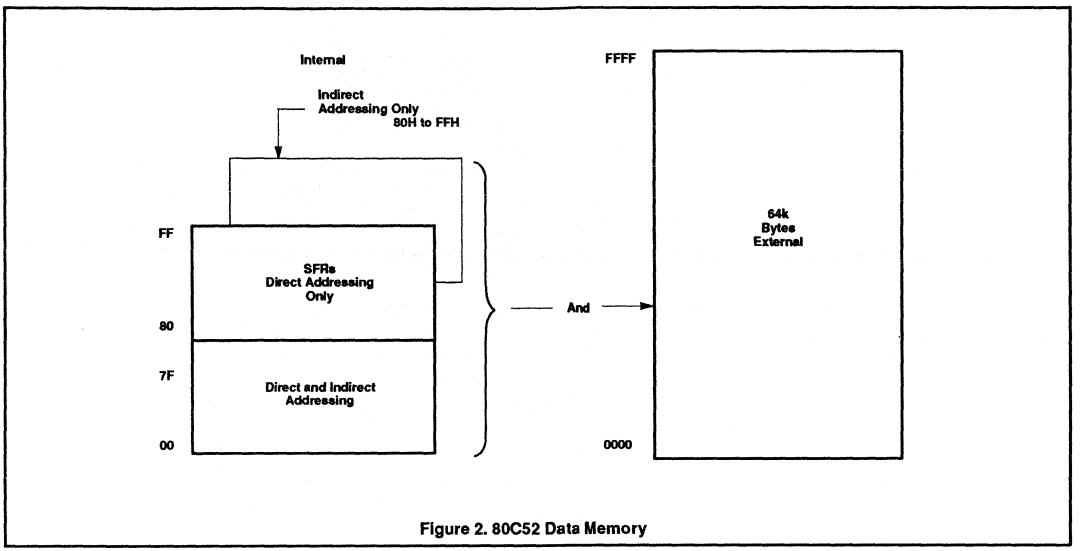


Figure 2. 80C52 Data Memory

In the Capture Mode there are two options which are selected by bit EXEN2 in T2CON. If EXEN2 = 0, then Timer 2 is a 16-bit timer or counter which upon overflowing sets bit TF2, the Timer 2 overflow bit, which can be used to generate an interrupt. If EXEN2 = 1, then Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX causes the current value in the Timer 2 registers, TL2 and TH2, to be captured into registers RCAP2L and RCAP2H, respectively. (RCAP2L and RCAP2H are new special function registers in the 80C52.) In addition, the transition at T2EX causes bit EXF2 in T2CON to be set, and EXF2 like TF2 can generate an interrupt. The Capture Mode is illustrated in Figure 4.

In the auto-reload mode, there are again two options, which are selected by bit EXEN2 in T2CON. If EXEN2 = 0, then when Timer 2 rolls over it not only sets TF2 but also causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2L and RCAP2H, which are preset by software. If EXEN2 = 1, then Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX will also trigger the 16-bit reload and set EXF2. The auto-reload mode is illustrated in Figure 5.

The baud rate generation mode is selected by RCLK = 1 and/or TCLK = 1. It will be described in conjunction with the serial port.

Serial Port

The serial port of the 8XC52 is identical to that of the 80C51 except that counter/timer 2 can be used to generate baud rates.

In the 80C52, Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (see Figure 3). Note that the baud rate for transmit and receive can be simultaneously different. Setting RCLK and/or TCLK puts Timer into its baud rate generator mode, as shown in Figure 6.

The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

Now, the baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate as follows:

$$\text{Modes 1, 3 Baud Rate} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

The timer can be configured for either "timer" or "counter" operation. In the most typical applications, it is configured for "timer" operation (C/T2 = 0). "Timer" operation is a little different for Timer 2 when it's being used as a baud rate generator. Normally, as a timer it would increment every machine cycle (thus at 1/12 the oscillator frequency). As a baud rate generator, however, it increments every state time (thus at 1/2 the oscillator frequency). In that case the baud rate is given by the formula:

$$\text{Modes 1, 3 Baud Rate} = \frac{\text{Oscillator Frequency}}{32\times[65536 - (\text{RCAP2H, RCAP2L})]}$$

where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

(MSB)							(LSB)
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2

Symbol	Position	Name and Significance
TF2	T2CON.7	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK or TCLK = 1.
EXF2	T2CON.6	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software.
RCLK	T2CON.5	Receive clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
TCLK	T2CON.4	Transmit clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EXEN2	T2CON.3	Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	T2CON.2	Start/stop control for Timer 2. A logic 1 starts the timer.
C/T2	T2CON.1	Timer or counter select. (Timer 2) 0 = Internal timer (OSC/12) 1 = External event counter (falling edge triggered).
CP/RL2	T2CON.0	Capture/Reload flag. When set, captures will occur on negative transitions at T2EX if EXEN2 = 1. When cleared, auto-reloads will occur either with Timer 2 overflows or negative transitions at T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

Figure 3. Timer/Counter 2 (T2CON) Control Register

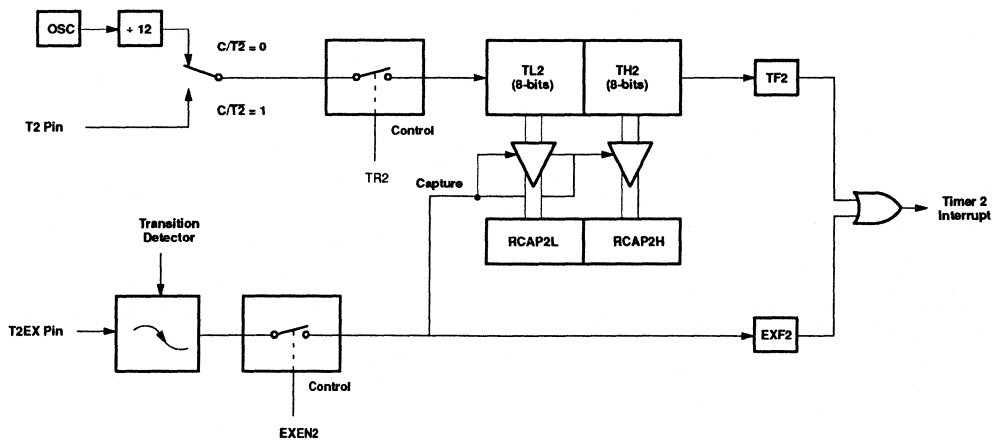


Figure 4. Timer 2 in Capture Mode

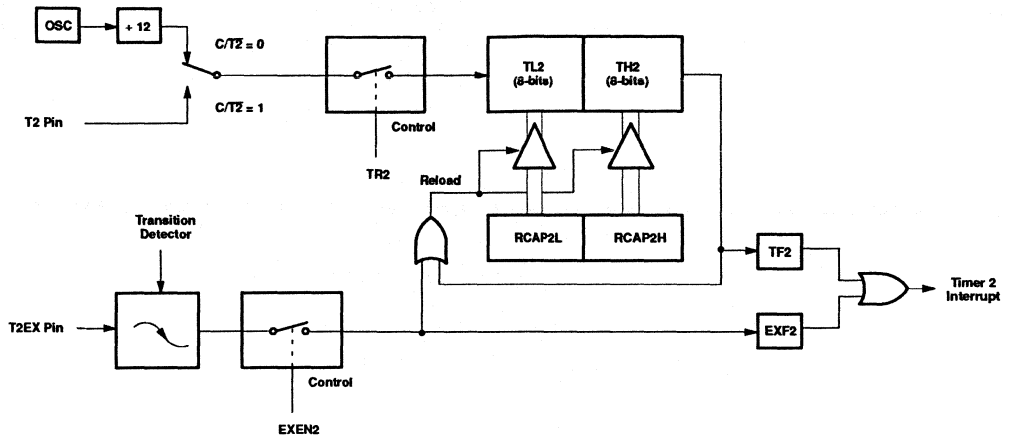


Figure 5. Timer 2 in Auto-Reload Mode

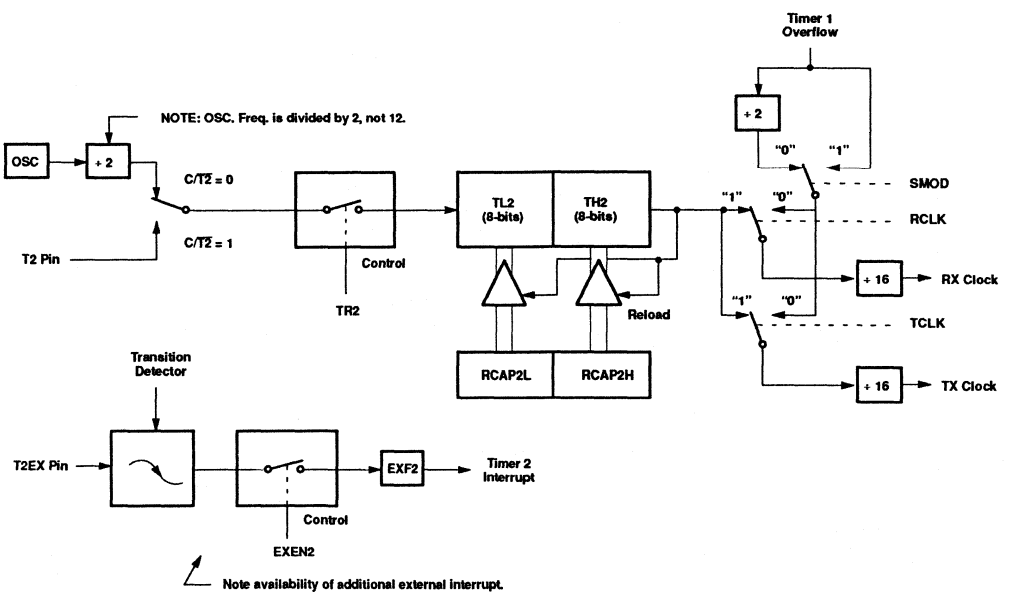


Figure 6. Timer 2 in Baud Rate Generator Mode

Table 1. 8XC52 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR:	Data pointer (2 bytes)										
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*	Interrupt enable	A8H	EA	–	ET2	ES	ET1	EX1	ET0	EX0	0x000000B
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*	Interrupt priority	B8H	–	–	PT2	PS	PT1	PX1	PT0	PX0	xx000000B
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	FFH
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	–	–	–	–	–	–	T2EX	T2	FFH
			A7	A6	A5	A4	A3	A2	A1	A0	
P2*	Port 2	A0H	A15	A14	A13	A12	A11	A10	A9	A8	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	$\overline{RD}$	$\overline{WR}$	T0	T1	$\overline{INT1}$	$\overline{INT0}$	TxD	RxD	FFH
PCON[1]	Power control	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxxxxxxB
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
RCAP2H#	Capture high	CBH									00H
RCAPL#	Capture low	CAH									00H
SBUF	Serial data buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
SCON*	Serial controller	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
			CF	CE	CD	CC	CB	CA	C9	C8	
T2ON*#	Timer 2 control	C8H	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$	00H
TH0	Timer high 0	8CH									00H
TH1	Timer high 1	8DH									00H
TH2#	Timer high 2	CDH									00H
TL0	Timer low 0	8AH									00H
TL1	Timer low 1	8BH									00H
TL2#	Timer low 2	CCH									00H
TMOD	Timer mode	89H	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	00H

* Bit addressable

\# SFRs are modified from or added to the 80C51 SFRs.

[1] Bits GF1, GF0, PD, and IDL of the PCON register are not implemented in the NMOS 8XC52.

Table 2. Timer 2 Operating Modes

RCLK + RCLK	CP/RL2	TR2	MODE
0	0	1	16-bit Auto-reload
0	1	1	16-bit Capture
1	X	1	Baud rate generator
X	X	0	(off)

Timer 2 as a baud rate generator is shown in Figure 6. This figure is valid only if RCLK + TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2, and will not generate an interrupt. Therefore, the Timer 2 interrupt does not have to be disabled when Timer 2 is in the baud rate generator mode. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt, if desired.

It should be noted that when Timer 2 is running (TR2 = 1) in "timer" function in the baud rate generator mode, one should not try to read or write TH2 or TL2. Under these conditions the timer is being incremented every state time, and the results of a read or write may not be accurate. The RCAP registers may be read, but should not be written to, because a write might overlap a reload and cause write and/or reload errors. Turn the timer off (clear TR2) before accessing the Timer 2 or RCAP registers, in this case.

The serial port in Modes 1 and 3 with the timer 2 baud rate interface is shown in Figures 7 and 8.

Timer/Counter 2 Set-up

Except for the baud rate generator mode, the values given for T2CON do not include the setting of the TR2 bit. Therefore, bit TR2 must be set, separately, to turn the timer on. See Table 3 for set-up of timer 2 as a timer. See Table 4 for set-up of timer 2 as a counter.

Using Timer/Counter 2 to Generate Baud Rates

For this purpose, Timer 2 must be used in the baud rate generating mode. If Timer 2 is being clocked through pin T2 (P1.0) the baud rate is:

$$\text{Baud Rate} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

And if it is being clocked internally, the baud rate is:

$$\text{Baud Rate} = \frac{\text{Oscillator Frequency}}{32\text{x}[65536 - (\text{RCAP2H, RCA2PL})]}$$

To obtain the reload value for RCAP2H and RCA02L, the above equation can be rewritten as:

$$\text{RCAP2H, RCAP2L} = 65536 - \frac{\text{Oscillator Frequency}}{32 \times \text{Baud Rate}}$$

Interrupts

The 80C52 has 6 interrupt sources as shown in Figure 9. All except TF2 and EXF2 are identical sources to those in the 80C51.

The Interrupt Enable Register and the Interrupt Priority Register are modified to include the additional 80C52 interrupt sources. The operation of these registers is identical to the 80C51. The registers are detailed in Figures 10, 11, and 12.

In the 80C52, the Timer 2 Interrupt is generated by the logical OR of TF2 and EXF2. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and the bit will have to be cleared in software.

All of the bits that generate interrupts can be set or cleared by software, with the same result as though it has been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be canceled in software.

The interrupt vector addresses and the interrupt priority for requests in the same priority level are given in the following:

Source	Vector Address	Priority Within Level
1. IE0	0003H	(highest)
2. TF0	000BH	
3. IE1	0013H	
4. TF1	001BH	
5. RI + TI	0023H	
6. TF2 + EXF2	002BH	(lowest)

Note that they are identical to those in the 80C51 except for the addition of the Timer 2 (TF1 and EXF2) interrupt at 002BH and at the lowest priority within a level.

Port Structures

The port structures are identical in both parts, except that on the 8XC52, ports P1.0 and P1.1 include the Timer 2 alternate functions as follows:

P1.0 T2 (Timer/counter 2 external input)
P1.1 T2EX (Timer/counter 2 capture/reload trigger)

As with the 80C51, these alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1.

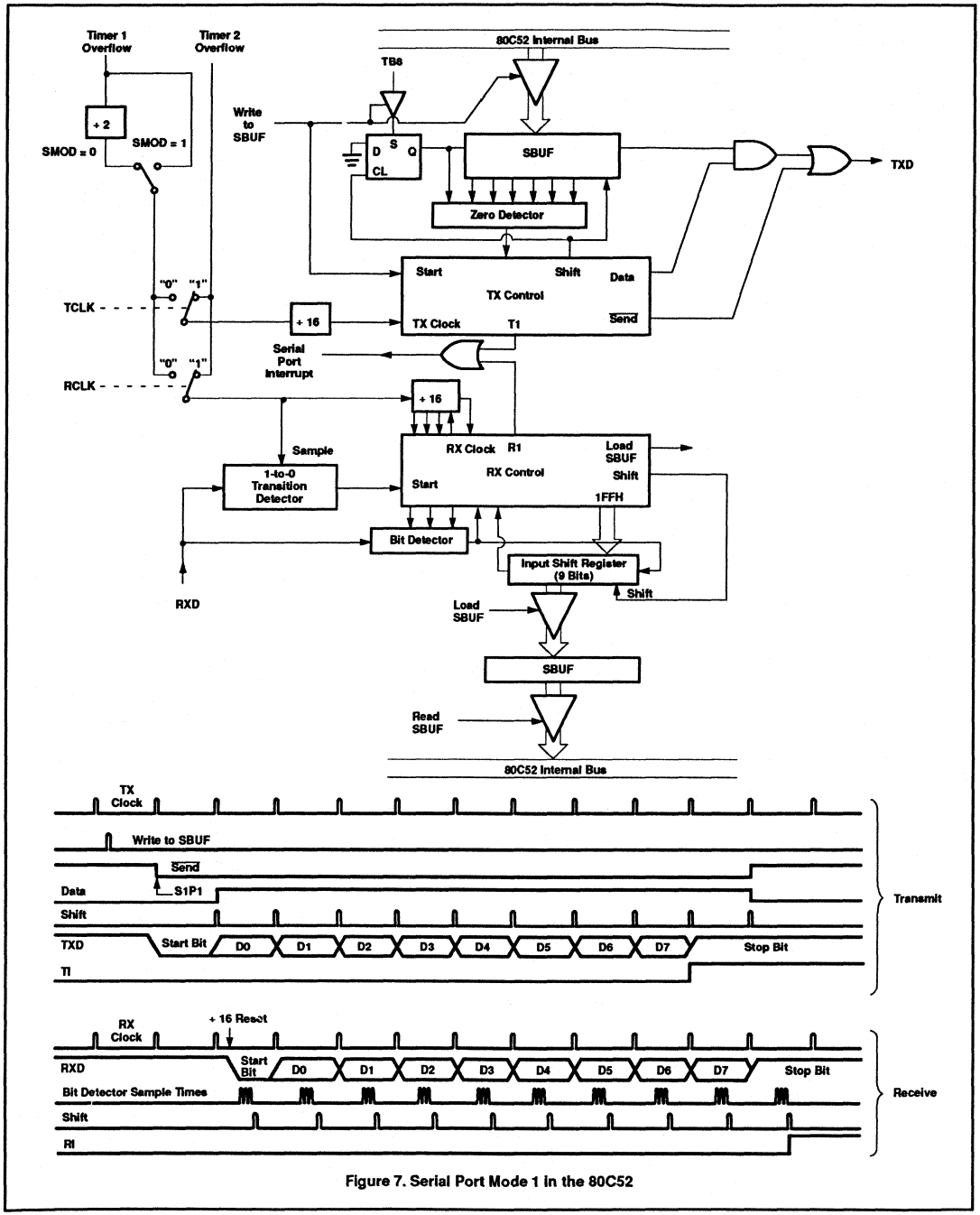


Figure 7. Serial Port Mode 1 in the 80C52

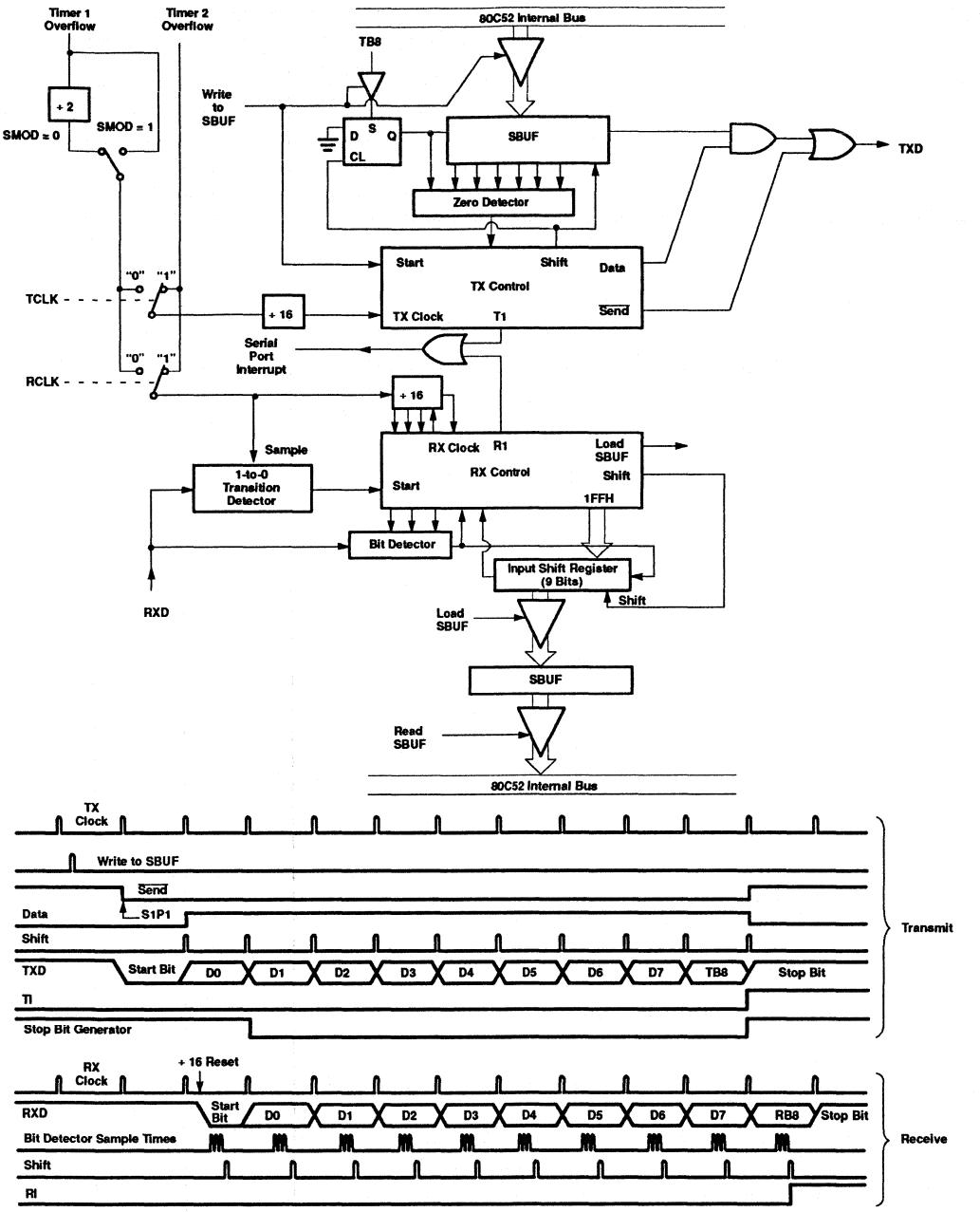


Figure 8. Serial Port Mode 3 in the 80C52

Table 3. Timer 2 as a Timer

MODE	T2CON	
	INTERNAL CONTROL (Note 1)	EXTERNAL CONTROL (Note 2)
16-bit Auto-Reload	00H	08H
16-bit Capture	01H	09H
Baud rate generator receive and transmit same baud rate	34H	36H
Receive only	24H	26H
Transmit only	14H	16H

Table 4. Timer 2 as a Counter

MODE	TMOD	
	INTERNAL CONTROL (Note 1)	EXTERNAL CONTROL (Note 2)
16-bit	02H	0AH
Auto-Reload	03H	0BH

NOTES:
1. Capture/reload occurs only on timer/counter overflow.
2. Capture/reload occurs on timer/counter overflow and a 1-to-0 transition on T2EX (P1.1) pin except when timer 2 is used in the baud rate generator mode.

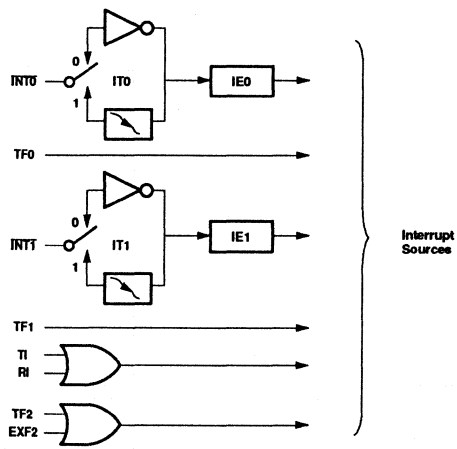


Figure 9. 80C52 Interrupt Sources

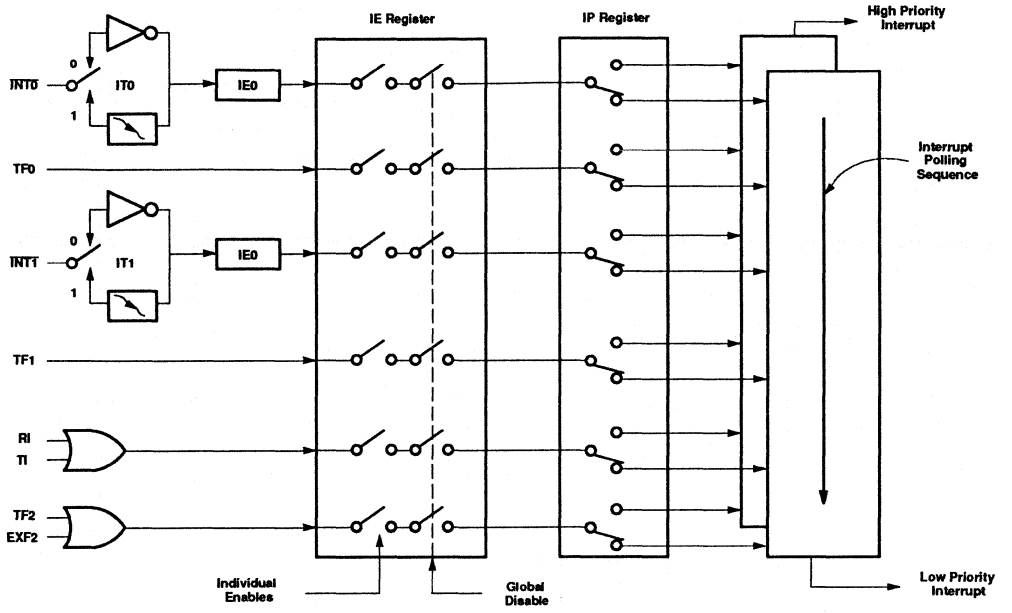


Figure 10. 80C52 Interrupt Control System

(MSB)							(LSB)
EA	X	ET2	ES	ET1	EX1	ET0	EX0

Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
	IE.6	Reserved.
ET2	IE.5	Enables or disables the Timer 2 Overflow or capture interrupt. If ET2 = 0, the Timer 2 interrupt is disabled.
ES	IE.4	Enables or disables the Serial Port interrupt. If ES = 0, the Serial Port interrupt is disabled.
ET1	IE.3	Enables or disables the Timer 1 Overflow interrupt. If ET1 = 0, the Timer 1 interrupt is disabled.
EX1	IE.2	Enables or disables External Interrupt 1. If EX1 = 0, External Interrupt 1 is disabled.
ET0	IE.1	Enables or disables the Timer 0 Overflow interrupt. If ET0 = 0, the Timer 0 interrupt is disabled.
EX0	IE.0	Enables or disables External Interrupt 0. If EX0 = 0, External Interrupt 0 is disabled.

Figure 11. 80C52 Interrupt Enable (IE) Register

(MSB)							(LSB)
X	X	PT2	PS	PT1	PX1	PT0	PX0

Symbol	Position	Function
–	IP.7	Reserved.
–	IP.6	Reserved.
PT2	IP.5	Defines the Timer 2 interrupt priority level. PT2 = 1 programs it to the higher priority level.
PS	IP.4	Defines the Serial Port interrupt priority level. PS = 1 programs it to the higher priority level.
PT1	IP.3	Defines theTimer 1 interrupt priority level. PT1 = 1 programs it to the higher priority level.
PX1	IP.2	Defines the External Interrupt 1 priority level. PX1 = 1 programs it to the higher priority level.
PT0	IP.1	Enables or disables the Timer 0 Interrupt priority level. PT0 = 1 programs it to the higher priority level.
PX0	IP.0	Defines the External Interrupt 0 priority level. PX0 = 1 programs it to the higher priority level.

Figure 12. 80C52 Interrupt Priority (IP) Register

Philips Components

Date of Issue	September 6, 1990
Status	Product Specification
Application Specific Product	

8032AH/8052AH

Single-chip 8-bit microcontroller

DESCRIPTION

The Philips 8032AH/8052AH is a high-performance microcontroller fabricated using the Philips high-density highly reliable +5V, depletion-load, N-channel, silicon-gate, N500 MOS process technology. It provides the hardware features, architectural enhancements and instructions that are necessary to make it a powerful and cost-effective controller for applications requiring up to 64k bytes of program memory and/or up to 64k bytes of data storage.

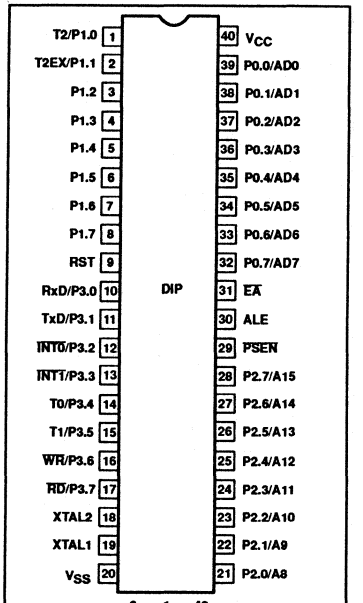
The 8032AH/8052AH contains 256 bytes of read/write data memory, 32 I/O lines configured as four 8-bit ports, three 16-bit counter/timers, a six-source, two-priority-level nested interrupt structure, a programmable serial I/O port and on-chip oscillator and clock circuitry. The 8052AH has all of these features plus 8k bytes of non-volatile read-only program memory. Both microcontrollers have memory expansion capabilities of up to 64k bytes of data storage and 64k bytes of program memory that can be attained with standard TTL compatible memories.

Because of its extensive BCD/binary arithmetic and bit-handling facilities, the 8032AH/8052AH microcontroller is efficient at both computational and control-oriented tasks. Efficient use of program memory is also achieved by using the familiar compact instruction set of the 8031AH/8051AH. Forty-four percent of the instructions are one-byte, 41% two-byte, and 15% three-byte instructions. With a 12MHz crystal, the majority of the instructions execute in just 1.0μs. The longest instructions, multiply and divide, require only 4μs at 12MHz.

FEATURES

- 8032AH – control-oriented CPU with RAM and I/O
- 8052AH – an 8032AH with factory mask-programmable ROM
- 8k X 8 ROM (8052AH only)
- 256 X 8 RAM
- Four 8-bit ports, 32 I/O lines
- Three 16-bit timer/counters
- Programmable full-duplex serial channel
  - –Variable transmit/receive baud rate capability
- Timer 2 capture capability
- External memory
  - –64k ROM and 64k RAM
- Boolean processor
- 128 user bit-addressable locations
- Upward compatible with 8031AH/8051AH

PIN CONFIGURATION

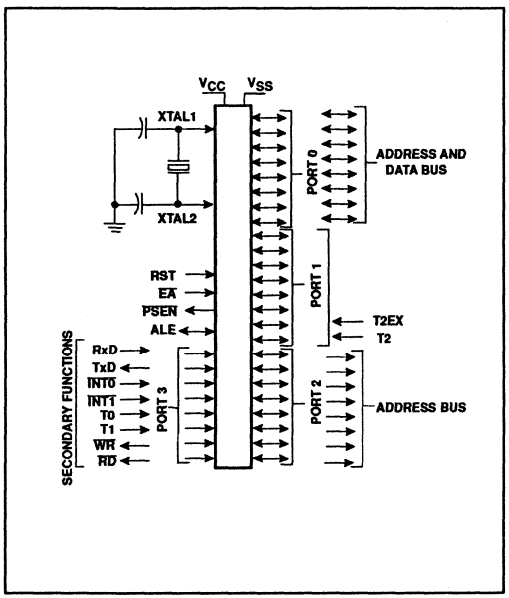


Pin	Function	Pin	Function	Pin	Function
1	NC	15	INT1/P3.3	30	P2.6/A14
2	T2/P1.0	16	T0/P3.4	31	P2.7/A15
3	T2EX/P1.1	17	T1/P3.5	32	PSEN
4	P1.2	18	WR/P3.6	33	ALE
5	P1.3	19	RD/P3.7	34	NC
6	P1.4	20	XTAL2	35	EA
7	P1.5	21	XTAL1	36	P0.7/AD7
8	P1.6	22	V_{SS}	37	P0.6/AD6
9	P1.7	23	NC	38	P0.5/AD5
10	RST	24	P2.0/A8	39	P0.4/AD4
11	RxD/P3.0	25	P2.1/A9	40	P0.3/AD3
12	NC	26	P2.2/A10	41	P0.2/AD2
13	TxD/P3.1	27	P2.3/A11	42	P0.1/AD1
14	INT0/P3.2	28	P2.4/A12	43	P0.0/AD0
		29	P2.5/A13	44	V_{CC}

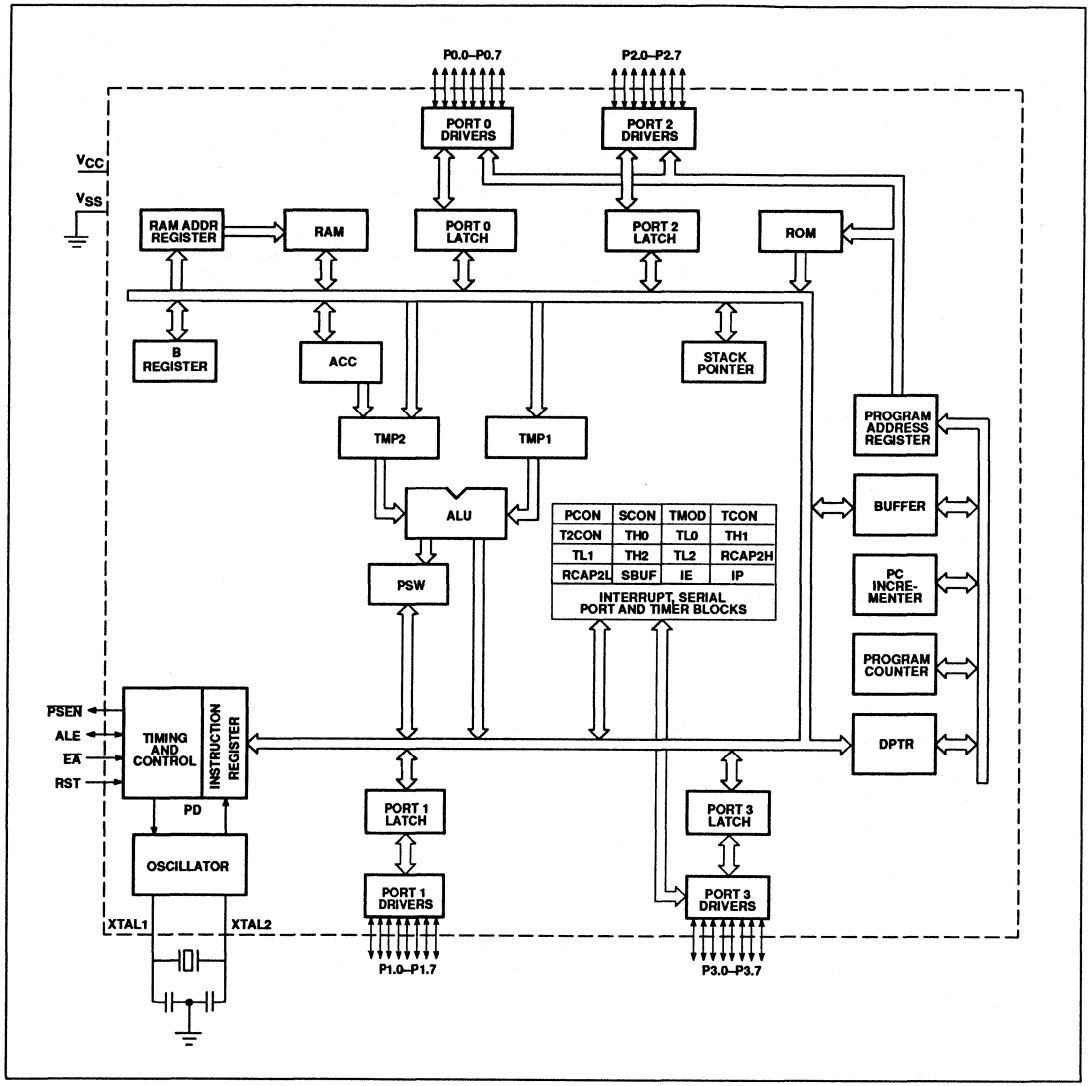
PART NUMBER SELECTION

PHILIPS		PHILIPS COMPONENTS-SIGNETICS		TEMPERATURE °C AND PACKAGE	FREQUENCY MHz
ROMless	ROM	ROMless	ROM		
MAB8032AH-2P	MAB8052AH-2P	SCN8032HCCN40	SCN8052HCCN40	0 to +70, plastic DIP	12
MAB8032AH-2WP	MAB8052AH-2WP	SCN8032HCCA44	SCN8052HCCA44	0 to +70, plastic LCC	12
MAF8032AH-2P	MAF8052AH-2P	SCN8032HACN40	SCN8052HACN40	–40 to +85, plastic DIP	12
MAF8032AH-2WP	MAF8052AH-2WP	SCN8032HACA44	SCN8052HACA44	–40 to +85, plastic LCC	12
		SCN8032HCFN40	SCN8052HCFN40	0 to +70, plastic DIP	15
		SCN8032HCFA44	SCN8052HCFA44	0 to +70, plastic LCC	15
		SCN8032HAFN40	SCN8052HAFN40	–40 to +85, plastic DIP	15
		SCN8032HAFA44	SCN8052HAFA44	–40 TO +85, plastic LCC	15

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	TYPE	NAME AND FUNCTION
V_{SS}	20	22	I	**Ground:** 0V reference.
V_{CC}	40	44	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–46	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s.
P1.0–P1.7	1–8	2–9	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Pins P1.0 and P1.1 also correspond to the special functions T2, timer 2 counter trigger input, and T2EX, external input to timer 2. the output latch on these two special functions must be programmed to one for that function to operate. Port 1 also receives the low-order address byte during program verification.
	1	2	I	**T2 (P1.0):** Timer/counter 2 trigger input.
	2	3	I	**T2EX (P1.1):** Timer/counter 2 external count input.
P2.0–P2.7	21–28	24–31	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 also serves the special features of the SC80C51 family, as listed below:
	10	11	I	**RxD (P3.0):** Serial input port
	11	13	O	**TxD (P3.1):** Serial output port
	12	14	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	I	**T0 (P3.4):** Timer 0 external input
	15	17	I	**T1 (P3.5):** Timer 1 external input
	16	18	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. A small external pull-down resistor (approx 8.2kohm) from RST to V_{SS} permits power-on reset when a capacitor (approx 10uF) is also connected from this pin to V_{CC}.
ALE	30	33	I/O	**Address Latch Enable:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory.
$\overline{PSEN}$	29	32	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$	31	35	I	**External Access Enable:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H and 1FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 1FFFH.
XTAL1	19	21	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	O	**Crystal 2:** Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Voltage on any other pin to V_{SS}	–0.5 to +7.0	V
Input, output current on any single pin	10	mA
Power dissipation	1.5	W

DC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V[4, 5]

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS		UNIT
			MIN	MAX	
V_{IL}	Input low voltage		–0.5	0.8	V
V_{IH}	Input high voltage; except XTAL2, RST		2.0	V_{CC}+0.5	V
V_{IH1}	Input high voltage to RST for reset, XTAL2	XTAL1 to V_{SS}	2.5	V_{CC}+0.5	V
V_{OL}	Output low voltage; ports 1, 2, 3[6]	I_{OL} = 1.6mA		0.45	V
V_{OL1}	Output low voltage; port 0, ALE, $\overline{PSEN}$[6]	I_{OL} = 3.2mA		0.45	V
V_{OH}	Output high voltage; ports 1, 2, 3	I_{OH} = –80uA	2.4		V
V_{OH1}	Output high voltage; port 0 in external bus mode, ALE, $\overline{PSEN}$[3]	I_{OH} = –400uA	2.4		V
I_{IL}	Logical 0 input current; ports 1, 2, 3	V_{IN} = 0.45V		–800	uA
I_{IH1}	Input high current to RST for reset	V_{IN} = V_{CC} –1.5V		500	uA
I_{LI}	Input leakage current; port 0, $\overline{EA}$	0.45 < V_{IN} < V_{CC}		±10	uA
I_{IL2}	Logical 0 input current for XTAL2	XTAL1 = V_{SS}, V_{IN} = 0.45V		–3.2	mA
I_{CC}	Power supply current	All outputs disconnected and $\overline{EA}$ = V_{CC}		175	mA
C_{IO}	Pin capacitance	f_C = 1MHz, T_A = 25°C		10	pF
T_A = –40°C to +85°C – Extended temperature range, V_{CC} = 5V ±10%, V_{SS} = 0V					
V_{IH}	Input high voltage; except XTAL2, RST		2.2	V_{CC}+0.5	V
V_{IH1}	Input high voltage to RST for reset, XTAL2	XTAL1 to V_{SS}	2.7		V
I_{IL2}	Logical 0 input current for XTAL2	XTAL1 = V_{SS}, V_{in} = 0.45V		–3.5	mA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.
5. All voltage measurements are referenced to ground. For testing, all input signals swing between 0.45V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and at output voltages of 0.8V and 2.0V as appropriate.
6. V_{OL} is derated when the device rapidly discharges external capacitance. This AC noise is most pronounced during emission of address data. When using external memory, locate the latch or buffer as close as possible to the device.

Datum	Emitting Ports	Degraded I/O Lines	V_{OL} (Peak Max)
Address	P2, P0	P1, P3	0.8V
Write Data	P0	P1, p3, ALE	0.8V

7. C_L = 100pF for port 0, ALE and $\overline{PSEN}$ outputs: C_L = 80pF for all other ports.

AC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V[1, 2]

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$		Oscillator frequency: **Speed Versions** SCN8052 C 32MAB8052/32 –2 SCN8052 F 32MAF8052/32 –2			 3.5 3.5 3.5 3.5	 12 12 15 12	 MHz MHz MHz MHz
t_{LHLL}	1	ALE pulse width	127		$2t_{CLCL}$–40		ns
t_{AVLL}	1	Address valid to ALE low	43		t_{CLCL}–40		ns
t_{LLAX}	1	Address hold after ALE low	48		t_{CLCL}–35		ns
t_{LLIV}	1	ALE low to valid instruction in		233		$4t_{CLCL}$–100	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	58		t_{CLCL}–25		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	215		$3t_{CLCL}$–35		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		125		$3t_{CLCL}$–125	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		63		t_{CLCL}–20	ns
t_{AVIV}	1	Address to valid instruction in		302		$5t_{CLCL}$–115	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		20		20	ns
t_{PXAV}	1	$\overline{PSEN}$ to address valid	75		t_{CLCL} –8		ns
Data Memory							
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	400		$6t_{CLCL}$–100		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	400		$6t_{CLCL}$–100		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		252		$5t_{CLCL}$–165	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		97		$2t_{CLCL}$–70	ns
t_{LLDV}	2, 3	ALE low to valid data in		517		$8t_{CLCL}$–150	ns
t_{AVDV}	2, 3	Address to valid data in		585		$9t_{CLCL}$–165	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}$–50	$3t_{CLCL}$+50	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}$–130		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	23		t_{CLCL}–60		ns
t_{QVWH}	2, 3	Data valid to $\overline{WR}$ high	433		$7t_{CLCL}$ –150		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	33		t_{CLCL}–50		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		20		20	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	t_{CLCL}–40	t_{CLCL}+40	ns
External Clock							
t_{CHCX}	5	High time	20		20		ns
t_{CLCX}	5	Low time	20		20		ns
t_{CLCH}	5	Rise time		20		20	ns
t_{CHCL}	5	Fall time		20		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

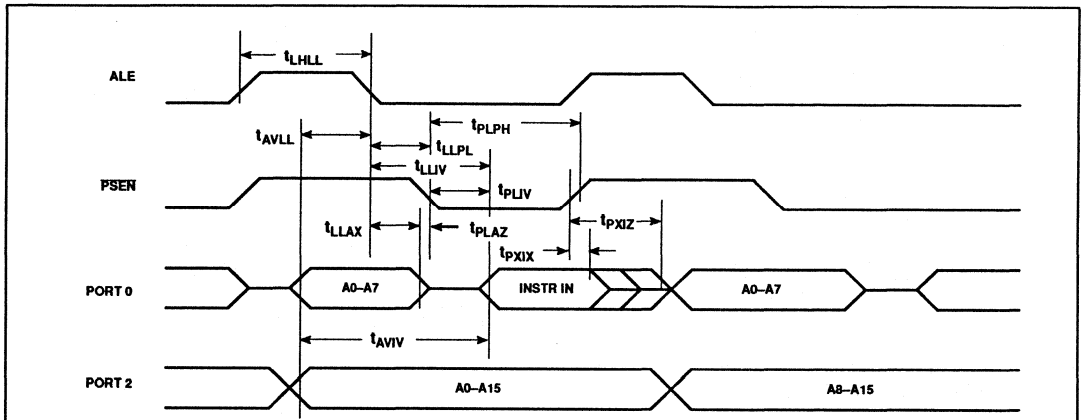


Figure 1. External Program Memory Read Cycle

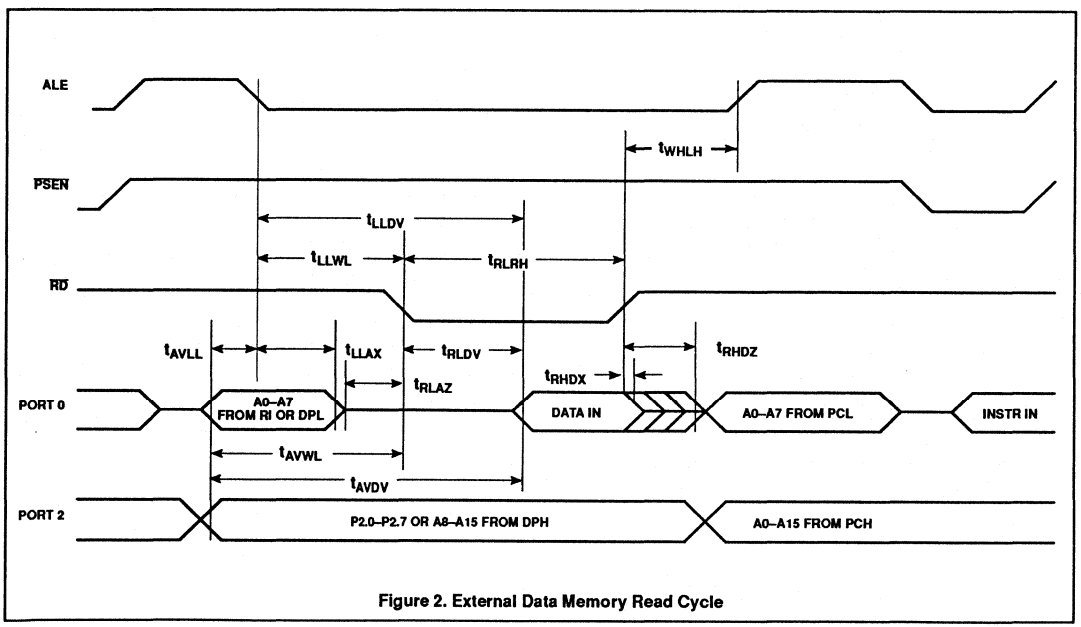


Figure 2. External Data Memory Read Cycle

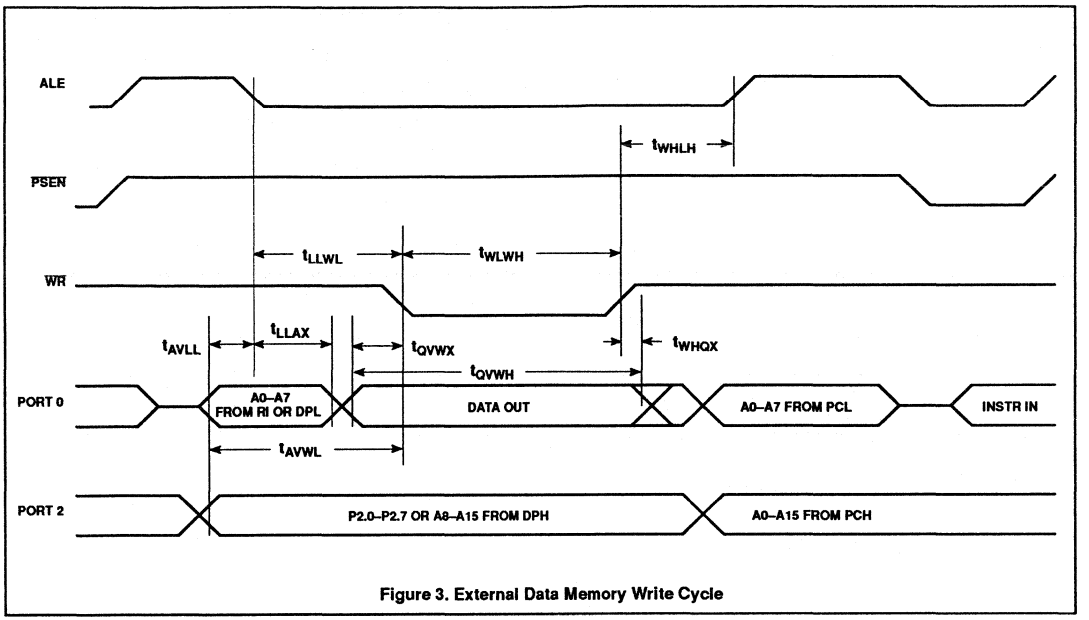


Figure 3. External Data Memory Write Cycle

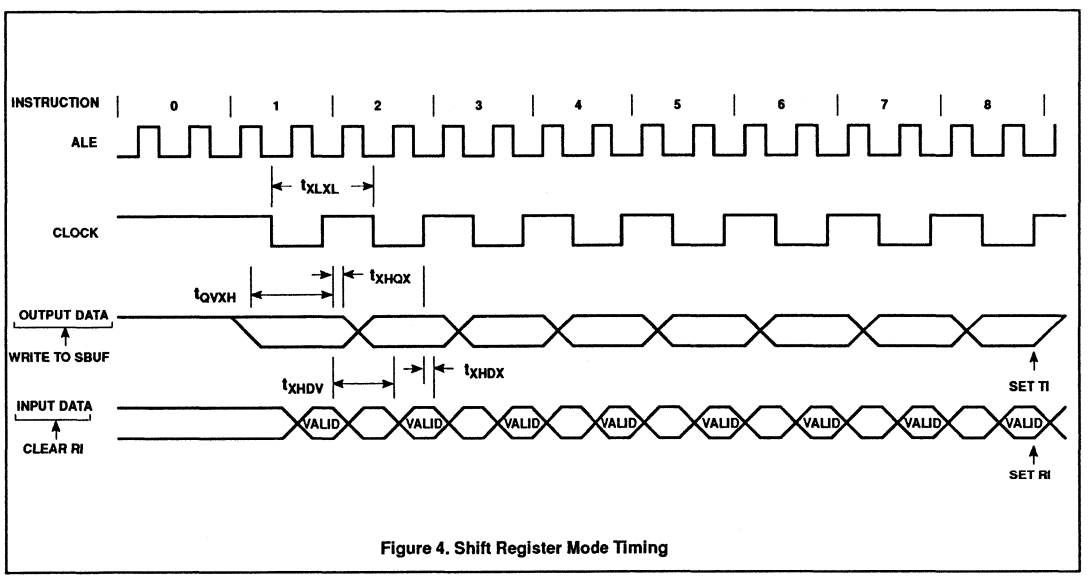


Figure 4. Shift Register Mode Timing

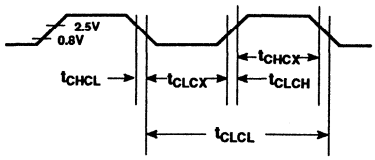
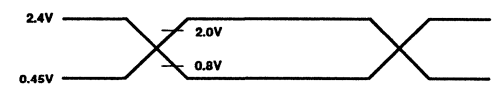
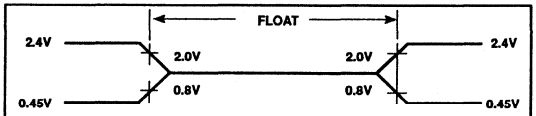


Figure 5. External Clock Drive



NOTE:
AC inputs during testing are driven at 2.4V for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at 2.0V min for a logic '1' and 0.8V for a logic '0'.

Figure 6. AC Testing Input/Output



NOTE:
For timing purposes, the float state is defined as the point at which a P0 pin sinks 2.4mA or sources 400μA at the voltage test levels.

Figure 7. Float Waveform

Philips Components

Document No.	
ECN No.	
Date of Issue	January 1990
Status	Preliminary Specification
Application Specific Product	

80C32/80C52/87C52
CMOS single-chip 8-bit microcontroller

DESCRIPTION

The Philips 80C32/80C52/87C52 is a high-performance microcontroller fabricated with Philips high-density CMOS technology. The CMOS 8XC52 is functionally compatible with the NMOS SCN-8032/8052 microcontrollers. The Philips CMOS technology combines the high speed and density characteristics of HMOS with the low power attributes of CMOS. Philips epitaxial substrate minimizes latch-up sensitivity.

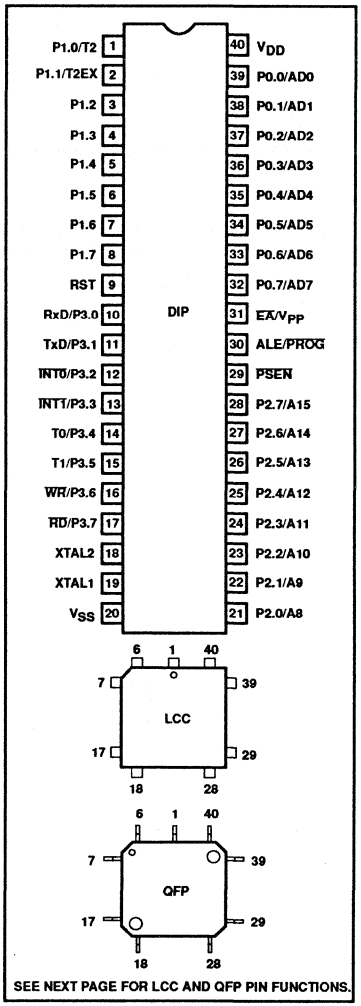
The 8XC52 contains an 8K x 8 ROM (80C52) EPROM (87C52), a 256 x 8 RAM, 32 I/O lines, three 16-bit counter/timers, a six-source, two-priority level nested interrupt structure, a serial I/O port for either multi-processor communications, I/O expansion or full duplex UART, and on-chip oscillator and clock circuits.

In addition, the 8XC52 has two software selectable modes of power reduction – idle mode and power-down mode. The idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.

FEATURES

- 80C51 based architecture
- 8032/8052 compatible
  - 8k x 8 ROM (80C52)
  - 8k x 8 EPROM (87C52)
  - ROMless (80C32)
  - 256 x 8 RAM
  - Three 16-bit counter/timers
  - Full duplex serial channel
  - Boolean processor
- Memory addressing capability
  - 64k ROM and 64k RAM
- Power control modes:
  - Idle mode
  - Power-down mode
- CMOS and TTL compatible
- Two speed ranges:
  - 3.5 to 16MHz
  - 3.5 to 20MHz
- Five package styles
- Extended temperature ranges
- OTP package available

PIN CONFIGURATION

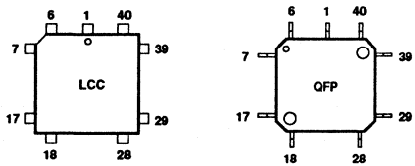


SEE NEXT PAGE FOR LCC AND QFP PIN FUNCTIONS.

PART NUMBER SELECTION

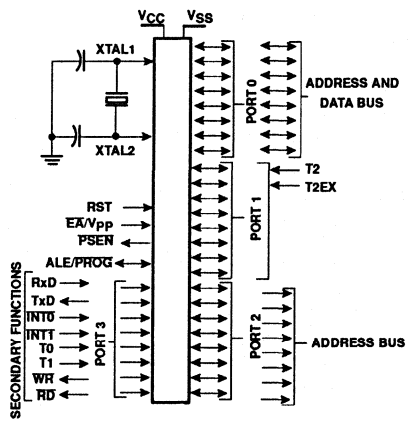
ROMless	ROM	EPROM	TEMPERATURE °C AND PACKAGE	FREQUENCY (MHz)
P80C32EBP	P83C52EBP	P87C52EBP	0 to +70, plastic DIP	16
P80C32EBA	P83C52EBA	P87C52EBA	0 to +70, plastic LCC	16
		P87C52GFF	–40 to +85, ceramic DIP	20
P80C32GFP	P83C52GFP	P87C52GFP	–40 to +85, plastic DIP	20
		P87C52GFL	–40 to +85, ceramic LCC	20
P80C32GFA	P83C52GFA	P87C52GFA	–40 to +85, plastic LCC	20
P80C32GBB	P83C52GBB	P87C52GBB	0 to +70, plastic QFP	20

LCC AND QFP PIN FUNCTIONS

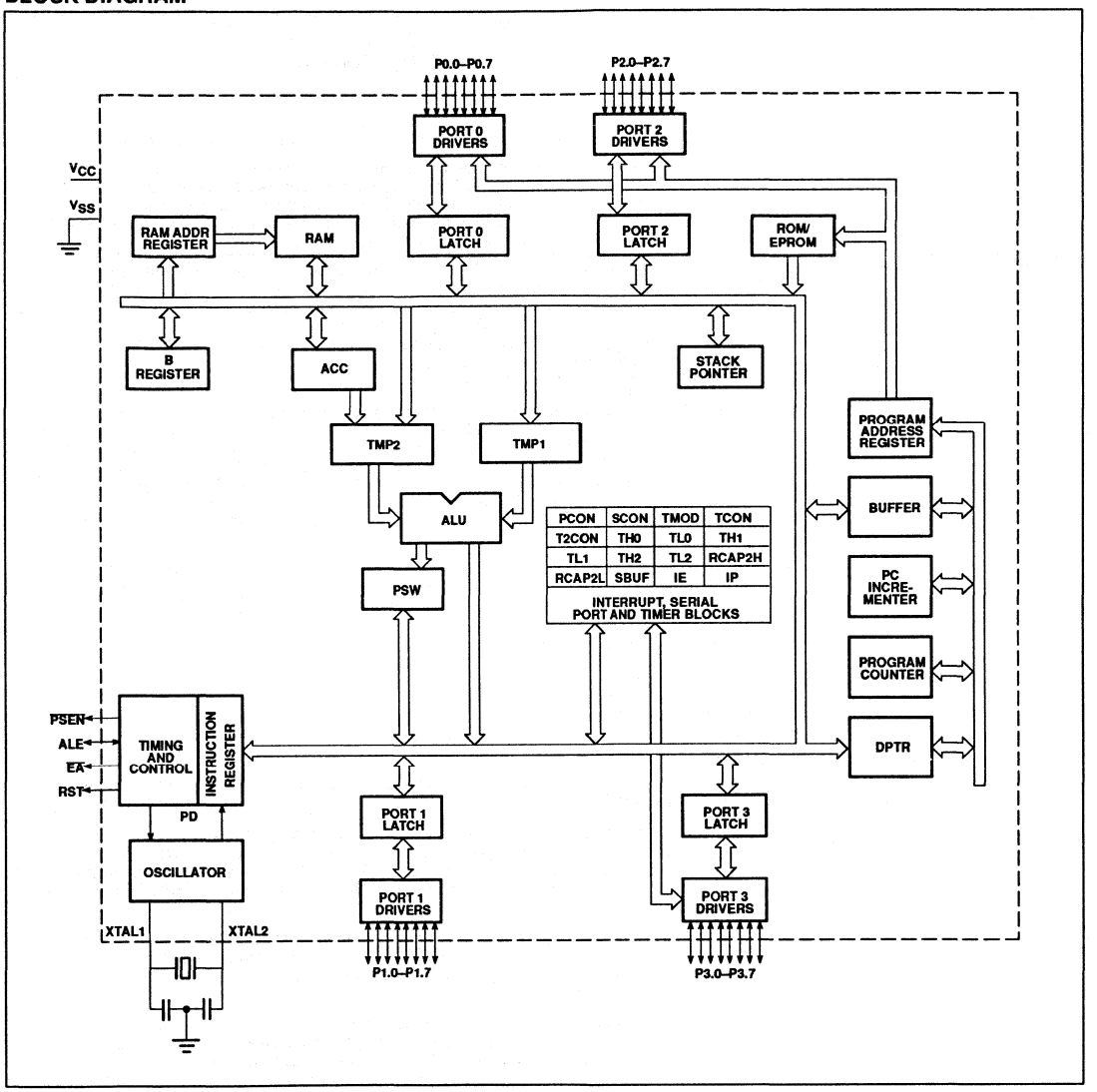


Pin	Function	Pin	Function
1	NC	23	V_{SS}
2	T2/P1.0	24	P2.0/A8
3	T2EX/P1.1	25	P2.1/A9
4	P1.2	26	P2.2/A10
5	P1.3	27	P2.3/A11
6	P1.4	28	P2.4/A12
7	P1.5	29	P2.5/A13
8	P1.6	30	P2.6/A14
9	P1.7	31	P2.7/A15
10	RST	32	$\overline{PSEN}$
11	RxDP3.0	33	ALE/$\overline{PROG}$
12	NC	34	NC
13	TxD/P3.1	35	$\overline{EA}/V_{PP}$
14	$\overline{INT0}$/P3.2	36	P0.7/AD7
15	$\overline{INT1}$/P3.3	37	P0.6/AD6
16	T0/P3.4	38	P0.5/AD5
17	T1/P3.5	39	P0.4/AD4
18	$\overline{WR}$/P3.6	40	P0.3/AD3
19	$\overline{RD}$/P3.7	41	P0.2/AD2
20	XTAL1	42	P0.1/AD1
21	XTAL2	43	P0.0/AD0
22	V_{SS}	44	V_{CC}

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC/QFP	TYPE	NAME AND FUNCTION
V_{SS}	20	22	I	**Ground:** 0V reference.
V_{CC}	40	44	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–46	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s. Port 0 also outputs the code bytes during program verification in the 87C52. External pull-ups are required during program verification.
P1.0–P1.7	1–8	2–9	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Pins P1.0 and P1.1 also. Port 1 also receives the low-order address byte during program memory verification. Port 1 also serves alternate functions for timer 2:
	1	2	I	**T2 (P1.0):** Timer/counter 2 external count input.
	2	3	I	**T2EX (P1.1):** Timer/counter 2 trigger input.
P2.0–P2.7	21–28	24–31	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the 80C51 family, as listed below:
	10	11	I	**RxD (P3.0):** Serial input port
	11	13	O	**TxD (P3.1):** Serial output port
	12	14	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	I	**T0 (P3.4):** Timer 0 external input
	15	17	I	**T1 (P3.5):** Timer 1 external input
	16	18	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE/$\overline{PROG}$	30	33	I/O	**Address Latch Enable/Program Pulse:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.
$\overline{PSEN}$	29	32	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$/V_{PP}	31	35	I	**External Access Enable/Programming Supply Voltage:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H to 1FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 1FFFH. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
XTAL1	19	21	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	O	**Crystal 2:** Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol, page 1.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-up reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-up, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

IDLE MODE

In idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. the control bits for the reduced power modes are in the special function register PCON.

DESIGN CONSIDERATIONS

At power-on, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

When the idle mode is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write when idle is terminated by reset, the instruction following the one that invokes idle should not be one that writes to a port pin or to external memory.

Table 1 shows the state of I/O ports during low current operating modes.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

Electrical Deviations from Commercial Specifications for Extended Temperature Range (87C52)

DC and AC parameters not included here are the same as in the commercial temperature range table.

DC ELECTRICAL CHARACTERISTICS

$T_A = -40°C$ to $+85°C$, $V_{CC} = 5V \pm 10\%$, $V_{SS} = 0V$

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS		UNIT
			MIN	MAX	
V_{IL}	Input low voltage, except $\overline{EA}$		–0.5	$0.2V_{CC}$–0.15	V
V_{IL1}	Input low voltage to $\overline{EA}$		0	$0.2V_{CC}$–0.35	V
V_{IH}	Input high voltage, except XTAL1, RST		$0.2V_{CC}$+1	V_{CC}+0.5	V
V_{IH1}	Input high voltage to XTAL1, RST		$0.7V_{CC}$+0.1	V_{CC}+0.5	V
I_{IL}	Logical 0 input current, ports 1, 2, 3	$V_{IN} = 0.45V$		–75	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3	$V_{IN} = 2.0V$		–750	µA
I_{CC}	Power supply current: Active mode Idle mode Power-down mode	$V_{CC} = 4.5$–$5.5V$, Frequency range = 3.5 to 12MHz		 35 6 50	 mA mA µA

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Operating temperature under bias	0 to +70 or –40 to +85	°C
Storage temperature range	–65 to +150	°C
Voltage on $\overline{EA}/V_{PP}$ pin to V_{SS}	0 to +13.0	V
Voltage on any other pin to V_{SS}	–0.5 to +6.5	V
Input, output current on any two pins	±10	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.5	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V (87C52)
T_A = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±20%, V_{SS} = 0V (80C32/80C52)

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			MIN	TYPICAL[1]	MAX	
V_{IL}	Input low voltage, except $\overline{EA}$[7]		–0.5		$0.2V_{CC}$–0.1	V
V_{IL1}	Input low voltage to $\overline{EA}$[7]		0		$0.2V_{CC}$–0.3	V
V_{IH}	Input high voltage, except XTAL1, RST[7]		$0.2V_{CC}$+0.9		V_{CC}+0.5	V
V_{IH1}	Input high voltage, XTAL1, RST[7]		$0.7V_{CC}$		V_{CC}+0.5	V
V_{OL}	Output low voltage, ports 1, 2, 3	I_{OL} = 1.6mA[2]			0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$	I_{OL} = 3.2mA[2]			0.45	V
V_{OH}	Output high voltage, ports 1, 2, 3, ALE, $\overline{PSEN}$[3]	I_{OH} = –60µA, I_{OH} = –25µA I_{OH} = –10µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$			V V V
V_{OH1}	Output high voltage (port 0 in external bus mode)	I_{OH} = –800µA, I_{OH} = –300µA I_{OH} = –80µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$			V V V
I_{IL}	Logical 0 input current, ports 1, 2, 3[7]	V_{IN} = 0.45V			–50	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3[7]	See note 4			–650	µA
I_{LI}	Input leakage current, port 0	V_{IN} = V_{IL} or V_{IH}			±10	µA
I_{CC}	Power supply current:[7] Active mode @ 12MHz[5] Idle mode @ 12MHz Power-down mode	See note 6		 11.5 1.3 3	 25 4 50	 mA mA µA
R_{RST}	Internal reset pull-down resistor		50		300	kohm
C_{IO}	Pin capacitance				10	pF

NOTES:
1. Typical ratings are not guaranteed. The values listed are at room temperature, 5V.
2. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input. I_{OL} can exceed these conditions provided that no single output sinks more than 5mA and no more than two outputs exceed the test conditions.
3. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
4. Pins of ports 1, 2 and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
5. I_{CC}MAX at other frequencies is given by: Active mode: I_{CC}MAX = 0.94 X FREQ + 13.71: Idle mode: I_{CC}MAX = 0.14 X FREQ +2.31, where FREQ is the external oscillator frequency in MHz. I_{CC}MAX is given in mA. See Figure 8.
6. See Figures 9 through 12 for I_{CC} test conditions.
7. These values apply only to T_A = 0°C to +70°C. For T_A = –40°C to +85°C, see table on previous page.
8. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.

AC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or −40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V (87C52)[1,2]

SYMBOL	FIGURE	PARAMETER	20MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	1	Oscillator frequency: **Speed Versions** 8XC52 E 8XC52 G			 3.5 3.5	 16 20	 MHz MHz
t_{LHLL}	1	ALE pulse width	60		$2t_{CLCL}-40$		ns
t_{AVLL}	1	Address valid to ALE low	18		$t_{CLCL}-32$		ns
t_{LLAX}	1	Address hold after ALE low	30		$t_{CLCL}-20$		ns
t_{LLIV}	1	ALE low to valid instruction in		100		$4t_{CLCL}-100$	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	33		$t_{CLCL}-17$		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	115		$3t_{CLCL}-35$		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		45		$3t_{CLCL}-105$	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		25		$t_{CLCL}-25$	ns
t_{AVIV}	1	Address to valid instruction in		175		$5t_{CLCL}-75$	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	200		$6t_{CLCL}-100$		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	200		$6t_{CLCL}-100$		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		85		$5t_{CLCL}-165$	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		72		$2t_{CLCL}-28$	ns
t_{LLDV}	2, 3	ALE low to valid data in		250		$8t_{CLCL}-150$	ns
t_{AVDV}	2, 3	Address to valid data in		285		$9t_{CLCL}-165$	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	100	200	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	70		$4t_{CLCL}-130$		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	10		$t_{CLCL}-40$		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	10		$t_{CLCL}-40$		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	10	90	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock							
t_{CHCX}	5	High time	20		20		ns
t_{CLCX}	5	Low time	20		20		ns
t_{CLCH}	5	Rise time		20		20	ns
t_{CHCL}	5	Fall time		20		20	ns
Shift Register							
t_{XLXL}	4	Serial port clock cycle time	600		$12t_{CLCL}$		ns
t_{QVXH}	4	Output data setup to clock rising edge	367		$10t_{CLCL}-133$		ns
t_{XHQX}	4	Output data hold after clock rising edge	20		$2t_{CLCL}-80$		ns
t_{XHDX}	4	Input data hold after clock rising edge	0		0		ns
t_{XHDV}	4	Clock rising edge to input data valid		367		$10t_{CLCL}-133$	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

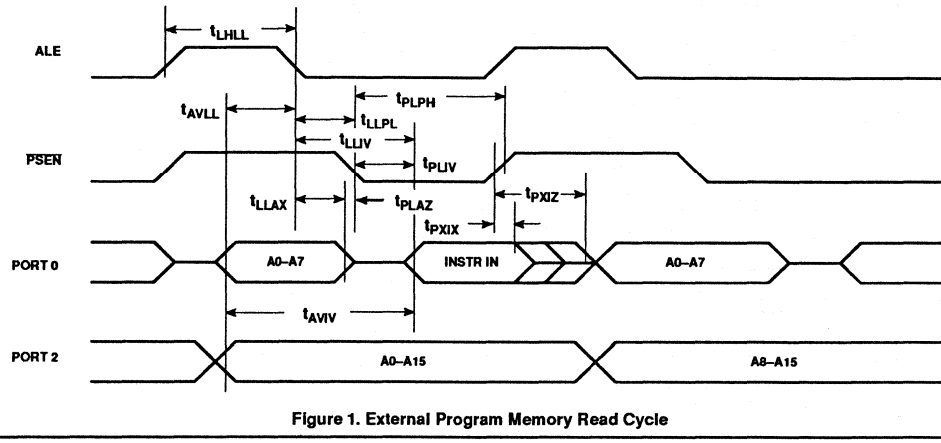


Figure 1. External Program Memory Read Cycle

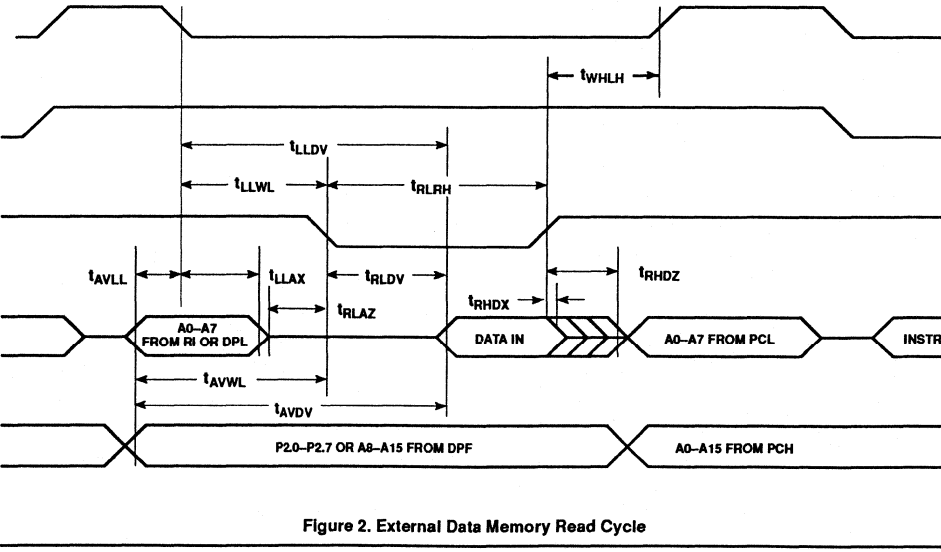


Figure 2. External Data Memory Read Cycle

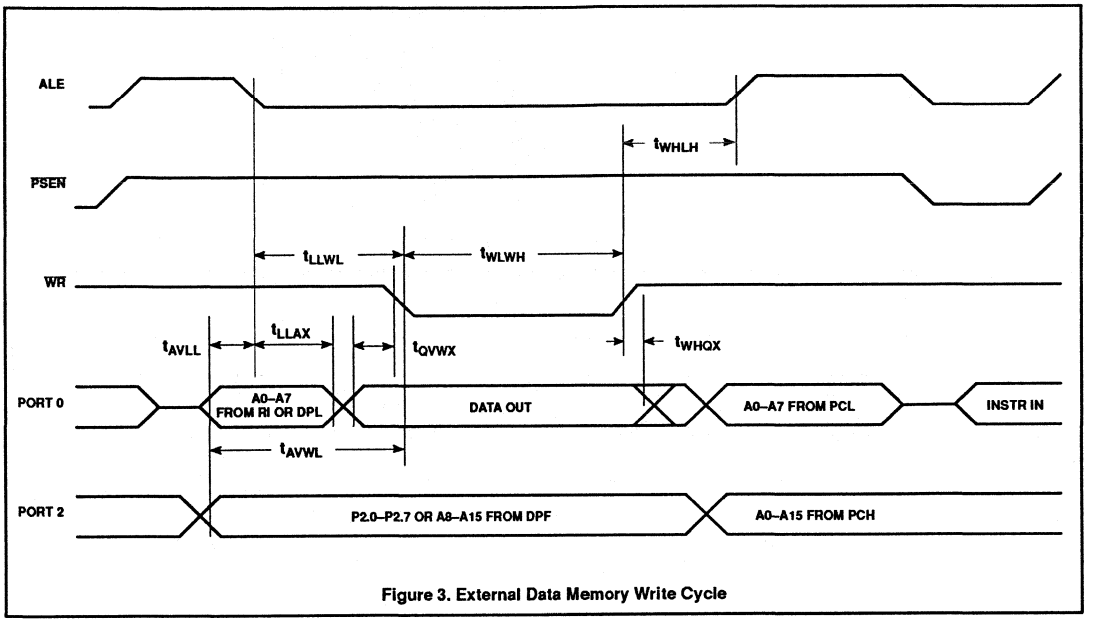


Figure 3. External Data Memory Write Cycle

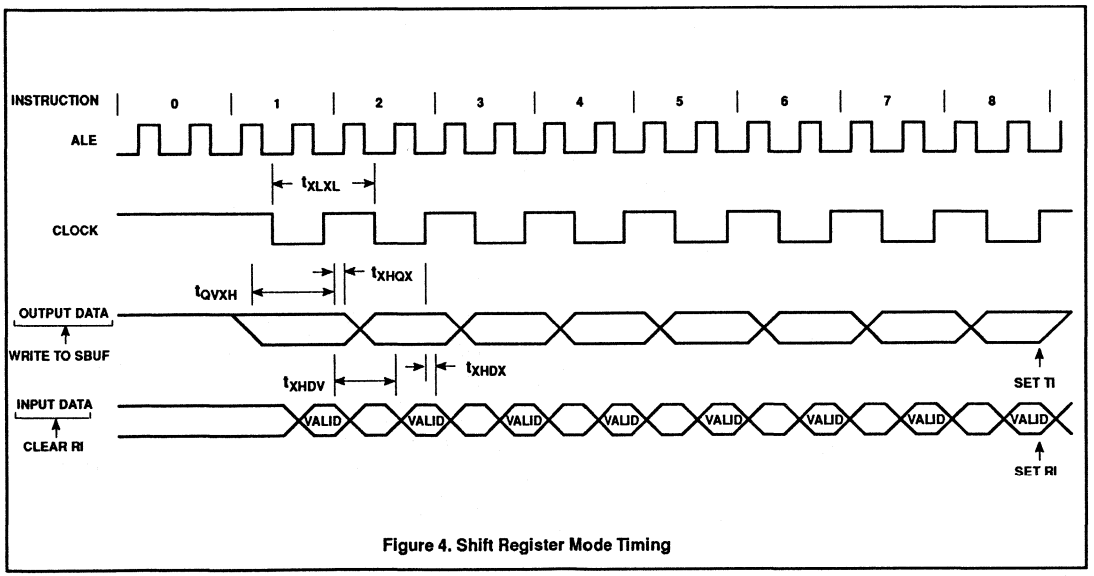


Figure 4. Shift Register Mode Timing

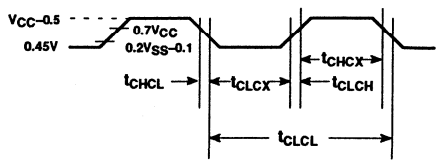
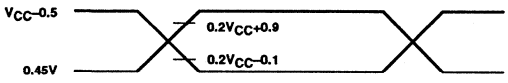
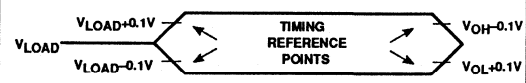


Figure 5. External Clock Drive



NOTE:
AC inputs during testing are driven at V_{CC} –0.5 for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at V_{IH} min for a logic '1' and V_{IL} for a logic '0'.

Figure 6. AC Testing Input/Output



NOTE:
For timing purposes, a port is no longer floating when a 100mV change from load voltage occurs, and begins to float when a 100mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OH}/I_{OL} \geq \pm 20$mA.

Figure 7. Float Waveform

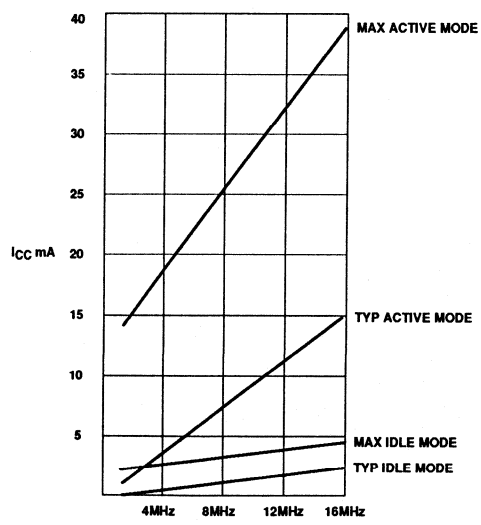


Figure 8. I_{CC} vs. FREQ
Valid only within frequency specifications of the device under test

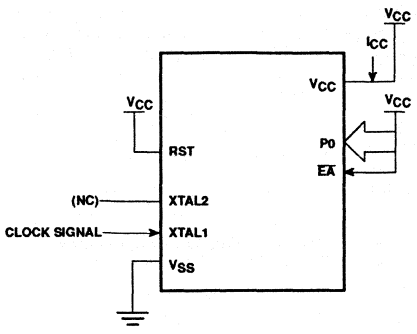


Figure 9. I_{CC} Test Condition, Active Mode
All other pins are disconnected

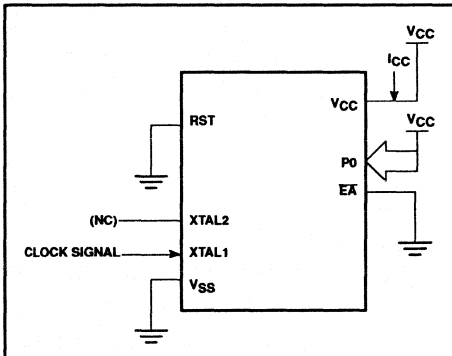


Figure 10. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

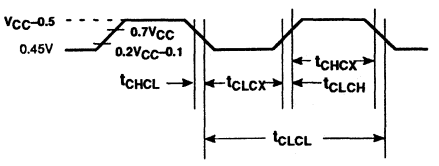


Figure 11. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

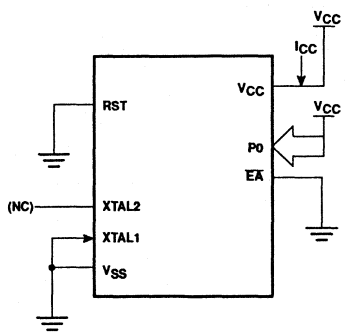


Figure 12. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected. V_{CC} = 2V to 5.5V

EPROM CHARACTERISTICS

The 87C52 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C52 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C52 manufactured by Philips.

Table 2 shows the logic levels for reading the signature byte, and for programming the program memory, the encryption table, and the lock bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 13 and 14. Figure 15 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 13. Note that the 87C52 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1 and 2, as shown in Figure 13. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$ and pins of ports 2 and 3 specified in Table 2 are held at the 'Program Code Data' levels indicated in Table 2. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 14.

To program the encryption table, repeat the 25 pulse programming sequence for addresses 0 through 1FH, using the 'Pgm Encryption Table' levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the lock bits, repeat the 25 pulse programming sequence using the 'Pgm Lock Bit' levels. After one lock bit is programmed, further programming of the code memory and encryption table is disabled. However, the other lock bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If lock bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be read is applied to ports 1 and 2 as shown in Figure 15. The other pins are held at the 'Verify Code Data' levels indicated in Table 2. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips

(031H) = 97H indicates 87C52

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 2, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345–5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-s/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000μW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

Table 2. EPROM Progamming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm lock bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm lock bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:

1. '0' = Valid low for that pin, '1' = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{CC} = 5V±10% during programming and verification.

*ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100μs (±10μs) and high for a minimum of 10μs.

™Trademark phrase of Intel Corporation.

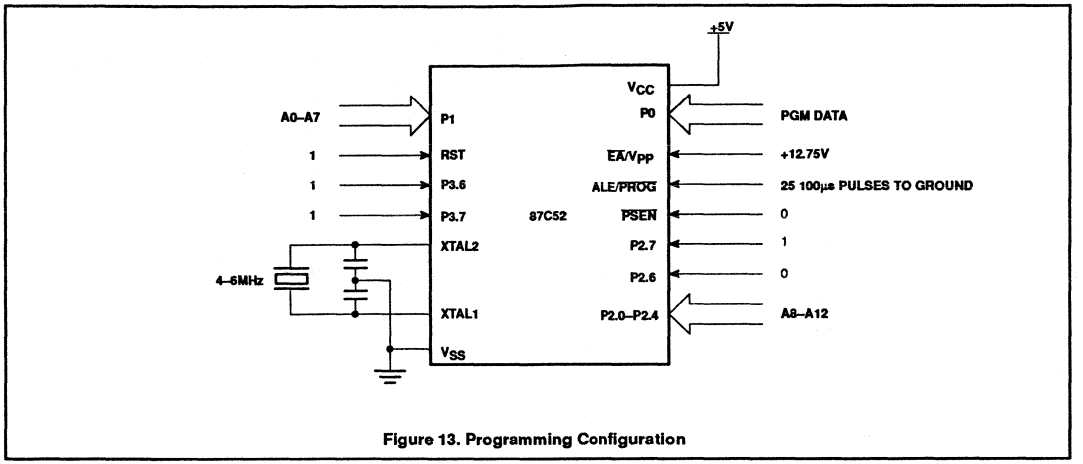


Figure 13. Programming Configuration

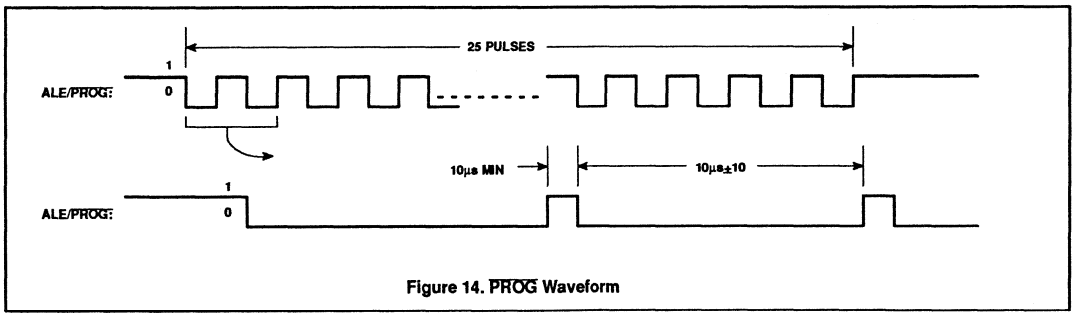


Figure 14. PROG Waveform

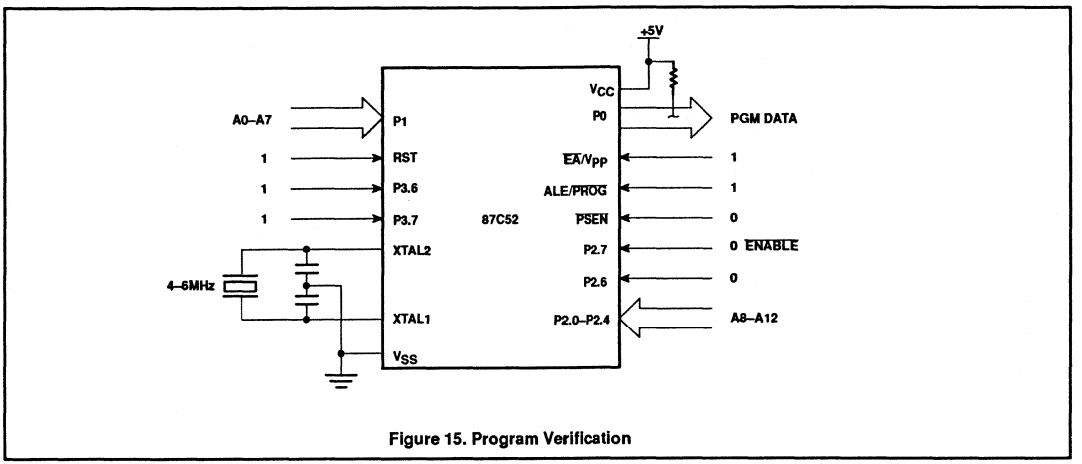


Figure 15. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_A = 21°C to +27°C, V_{CC} = 5V±10%, V_{SS} = 0V (See Figure 16)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		µs
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	$\overline{ENABLE}$ low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		µs

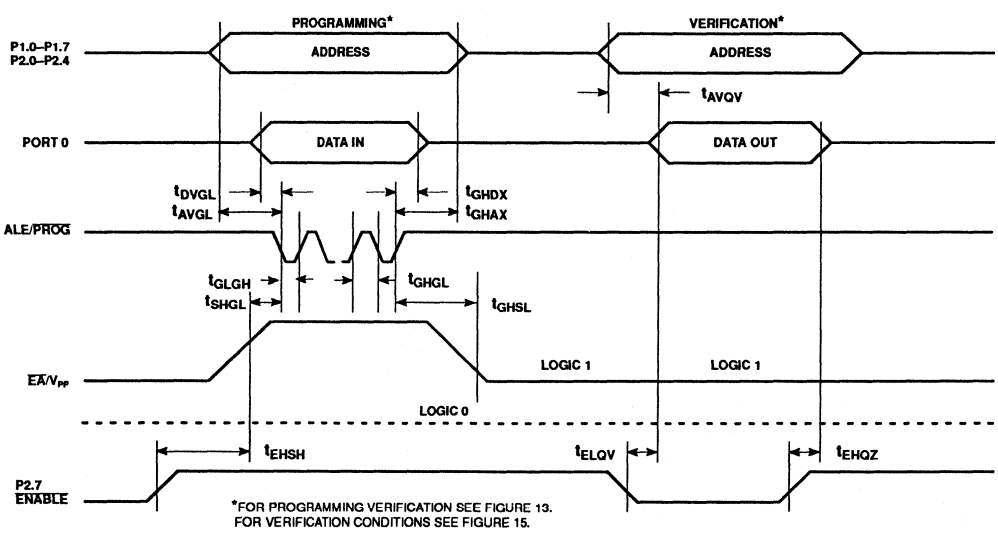


Figure 16. EPROM Programming and Verification

8XC053/54 OVERVIEW

The 8XC053/54 is an 80C51 derivative microcontroller that generates on-screen display for television and has PWM channels to control general TV functions. The applications for this part include television, security systems, medical and industrial imaging and many other applications where it is desirable to superimpose text onto a TV video image.

The part is available with either 8k (83C053) or 16k (83C054) program ROM, or 16k (87C054) of program EPROM. Other than the difference in memory size all of the features of the 83C053 and 83C054 are identical. For development of a system using the 83C053 the EPROM 87C054 can be used, but only the lower 8k of program memory space should be used. The 80C51 external memory interface pins ALE, PSEN, and EA have not been implemented on these parts, so there are no ROMless versions.

For the remainder of this overview the 83C053, 83C054, and 87C054 will be referred to collectively as the 83C053.

The 83C053 offers a number of improvements over the current solutions, which is an on-screen display controller and a separate microcontroller. The 83C053 has a unique carriage return linefeed feature utilizing a new-line character, which eliminates the need to use space characters. This not only reduces memory usage but also adds additional display flexibility. Using the 83C053 a total of 128 characters can be displayed on the screen in any format defined by the user. The vertical and horizontal starting points are programmable over a wide range. The 83C053 is not limited to standard TV synchronization rates (59.94kHz vertical and 15.535kHz horizontal); it can support a wide variety of sync rates for other raster scanned video displays, including monitors that use VGA graphics standards.

To improve text visibility, a number of enhancements have been added to the 83C053. Each character can be displayed in a different foreground color. Background can be switched on or off on a character-by-character basis, and background color may be changed for every new word. Other text enhancements include short row mode, double height characters, double size characters, and reduced character matrix (12 x 18).

The 83C053 also allows programmable shadowing; eight distinctive types of shadowing can be selected for individual characters.

The 83C053 has the following features:

- 80C51 based architecture
- 8k bytes ROM (83C053), 16k bytes ROM (83C054), 16k bytes EPROM (87C054)
- 192 bytes RAM
- On-screen display controller
  - Flexible formatting with OSD new line option
  - Eight text shadowing modes
  - Text color selectable per character
  - Background color selectable per word
  - Background color vs. video selectable per character
- Three digital video outputs
- Multiplexer/mixer and background intensity controls
- 128 x 10 display RAM
- 60 x 18 x 14 character generator ROM
- Eight 6-bit pulse width modulators
- One 14-bit pulse width modulator for high-precision voltage integration
- D/A converter and comparator with a 3-input multiplexer
- Four high-current open drain outputs
- Twelve high voltage (+12V) open drain outputs
- Programmable video input and output polarities

The part is available in a plastic 42-pin shrunk DIP package, or a ceramic 48-pin DIP package.

Differences from the 80C51

Memory Organization

The 83C053 differs from the 80C51 in that it has either 8k or 16k or on-chip program memory, and it is not externally expandable. The size of the on-chip data RAM space also differs in that it is 192 bytes. To support the on-screen display portion of the 83C053, there is a 128 x 10 bit display RAM. The display RAM holds 6 data bits and 4 attribute bits, which together make up its 10-bit words. The display RAM is accessed through three special function registers (OSAD, OSDT, and OSAT). A portion of the display RAM (6 data bits) is used to address the character generator ROM. There are 60 programmable (either mask ROM on the 83C053 or EPROM on the 87C054) 18 x 14 bit characters in the character ROM. Note that there are 60 programmable characters although this is a 64-character memory space because there is a new line and three space characters that are predefined.

Special Function Registers

The 83C053 contains 19 additional special function registers in addition to those found on the 80C51. Six of the additional registers control the on-screen display, ten control the PWMs on the part, one controls the D/A and voltage comparator, and two are implemented to make it possible to test a portion of the part. The six registers that control the on-screen display are: OSAT (on-screen attributes), OSDT (on-screen data), OSAD (on-screen address), OSCON (on-screen display origin). The ten special function register that control the PWMs are: PWM0-7 which control the 6-bit lo-res PWMs, and TDACL and TDACH which are the low and high bytes of the 14-bit hi-res PWM. The D/A is controlled by the SAD register. The remaining two additional special function registers are RAMCHR and RAMATT, which are for test purposes only and will not be discussed.

On-Screen Display Module (OSD)

The basic function of this block is to superimpose text onto a television video image. There are four input pins to the OSD block: two for a video clock, and horizontal and vertical sync signals. The block has four output pins: three color video signals, and a control signal.

As mentioned previously, the OSD function is controlled by six special function registers that provide communication to and from the 80C51 core.

A detailed description of the operation of these registers and the OSD is contained in the data sheet for the 83C053.

Pulse Width Modulators

There are nine pulse width modulators on the 83C053 that can be used to control different functions found in a video environment. The 6-bit PWMs are clocked from the output of a fixed 6-bit counter that is driven at 1/4 of the oscillator frequency. All of the 6-bit PWMs are therefore clocked at the same frequency which is 1/128 of the oscillator frequency. The eight 6-bit PWMs (PWM0-PWM7) each have a special function register that contains an enable bit and the 6-bit value. The 14-bit PWM is clocked at 1/4 the oscillator frequency. The 14-bit PWM has two special function registers (TDACL, TDACH) which contain an enable bit and the 14-bit value. The PWMs are discussed in detail in the data sheet.

Table 5. 83C053 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR	Data pointer (2 bytes)										
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*	Interrupt enable	A8H	EA	–	–	ES	ET1	EX1	ET0	EX0	0x000000B
OSAD	On-screen address	9A									
OSAT	On-screen attributes	98									
OSDT	On-screen data	99									
OSCON	On-screen display control	C0									
OSMOD	On-screen display mode	C1									
OSORG	On-screen display origin	C1									
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	FFH
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	–	–	–	–	–	–	T2EX	T2	FFH
			A7	A6	A5	A4	A3	A2	A1	A0	
P2*	Port 2	A0H	A15	A14	A13	A12	A11	A10	A9	A8	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	$\overline{RS}$	$\overline{WR}$	T0	T1	$\overline{INT1}$	$\overline{INT0}$	TxD	Rxd	FFH
PCON	Power control	87H	SMOD	–	–	–	GF1	GF0	PD		0xxxxxxxB
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
PWM0	Lo-res pulse width modulators	D4									
PWM1	Lo-res pulse width modulators	D5									
PWM2	Lo-res pulse width modulators	D6									
PWM3	Lo-res pulse width modulators	D7									
PWM4	Lo-res pulse width modulators	DC									
PWM5	Lo-res pulse width modulators	DD									
PWM6	Lo-res pulse width modulators	DE									

Table 5. 83C053 Special Function Registers (Continued)

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB LSB								RESET VALUE
PWM7	Lo-res pulse width modulators	DF									
SAD	D/A and voltage comparator	D8									
SP	Stack pointer	81H									07H
TDACH	Hi-res pulse width modulators	D3									
TDACL	Hi-res pulse width modulators	D2									
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
TH0	Timer high 0	8CH									00H
TH1	Timer high 1	8DH									00H
TL0	Timer low 0	8AH									00H
TL1	Timer low 1	8BH									00H
TMOD	Timer mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H

*Bit addressable

Software A/D
This A/D is a voltage comparator that has one side connected to an on-chip 4-bit D/A that is controlled by the four least significant bits of the SAD special function register. A voltage on P1.0, P1.1, or P1.2 is compared with the voltage generated by the internal D/A. The output from the comparator is the most significant bit of the SAD register and can be easily read. The A/D capabilities on the 83C053 are discussed in detail in the data sheet.

Reduced Power Modes
An idle mode has not been implemented on the 83C053. The only reduced power mode is power-down, which is entered by writing the PD bit of the PCON register to a 1 exactly the same as the 80C51. The power-down mode differs from the 80C51 in that the current is only reduced to several mA. This is due to the resistor ladder that is part of the A/D which is connected from V_{CC} to ground on the part. The resistor is always connected, so while the part is power-down and the oscillator is stopped, there is still considerable current drawn through the resistor. Note that during power-down mode the voltage across the 83C053 can be reduced, which will reduce the current drawn by the part.

Interrupts
Since the 83C053 does not have a UART, the corresponding interrupt as found on the 80C51 has been removed. In its place has been added an interrupt that occurs at the leading edge of the vsync pulse. The vector addresses for the interrupts are given in the data sheet. The IP register as found on the 80C51 is not implemented on the 83C053.

Philips Components

Document No.	
ECN No.	
Date of Issue	May 1990
Status	Preliminary Specification
Application Specific Product	

83C053, 87C054

Microcontroller for Television and Video (MTV)

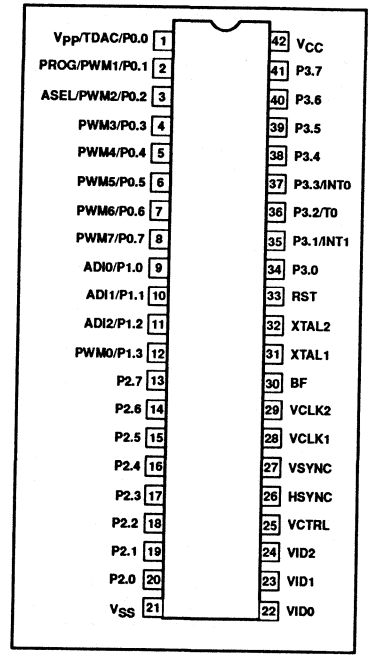
DESCRIPTION

The Microcontroller for Television and Video (MTV) applications is a derivative of Philips' industry-standard 80C51 microcontroller that is intended for use as the central control mechanism in a television receiver or tuner. Providing tuner functions and an On Screen Display facility, it represents a next-generation replacement for the currently available parts.

FEATURES

- 8192x8 masked ROM (83C) or 16384x8 OTPROM (87C)
- 192x8 RAM
- On Screen Display (OSD) Controller
- 3 digital video outputs
- Multiplexer/mixer and background intensity controls
- Flexible formatting with OSD New Line Option
- 128x10 display RAM
- 60x18x14 character generator ROM
- 8 text shadowing modes
- Text color selectable per character
- Background color selectable per word
- Background color vs. video selectable per character
- Eight 6-bit pulse width modulators for analog voltage integration
- One 14-bit PWM for high-precision voltage integration
- D/A converter and comparator with 3-input multiplexer
- 9 dedicated I/Os plus 28 port bits
- 15 port bits have alternate uses
- 4 high-current open-drain port outputs
- 12 high-voltage (+12V) open drain outputs
- Programmable video input and output polarities
- 80C51 instruction set
- No external memory capability
- 42 pin shrunk DIP (.07 inch center pins)
- High speed CMOS technology
- 5V ± 10% operation

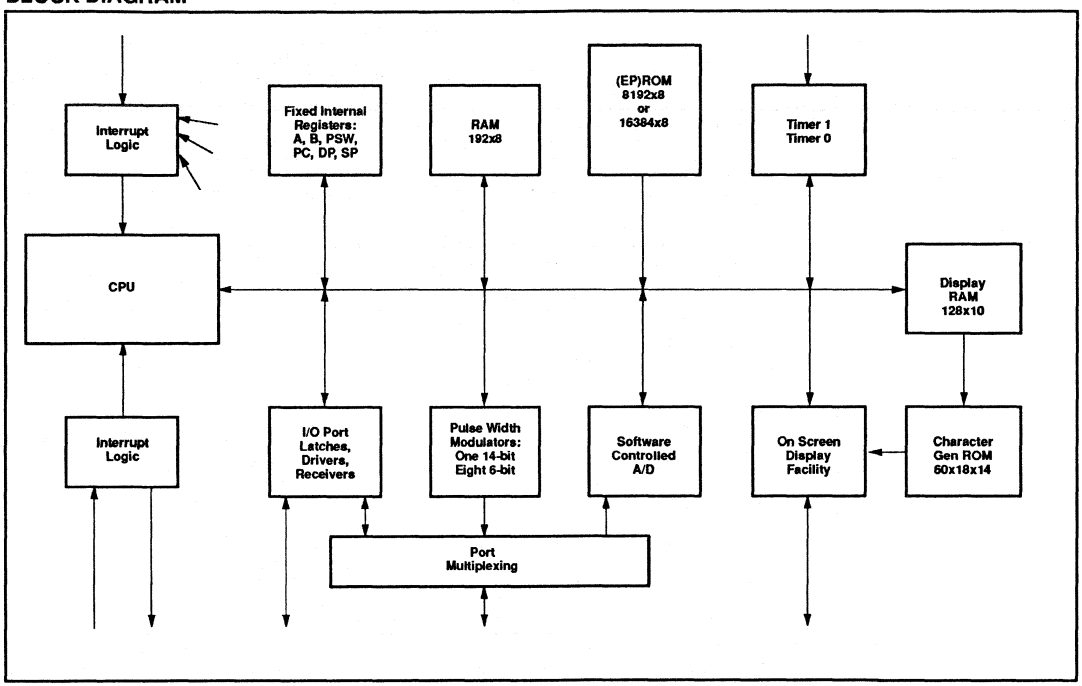
PIN CONFIGURATION



PART NUMBER SELECTION

ROM	EPROM	TEMPERATURE AND PACKAGE	FREQUENCY
P83C053BBP	P87C054BBP	0 to +70°C, plastic DIP	3.5 to 12MHz
	P87C054BBF	0 to +70°C, ceramic DIP	3.5 to 12MHz

BLOCK DIAGRAM



PIN DESCRIPTIONS

MNEMONIC	PIN NO. DIP	TYPE	NAME AND FUNCTION
VCLK1	28	I	**Video Clock 1:** Input for the horizontal timing reference for the On Screen Display facility. VCLK1 and VCLK2 are intended to be used with an external LC circuit to provide an on-chip oscillator. The period of the video clock is determined such that the width of a pixel in the On Screen Display is equal to the inter-line separation of the raster.
VCLK2	29	O	**Video Clock 2:** Output from the on-chip video oscillator.
HSYNC	26	I	**Horizontal Sync:** A dedicated input for a TTL-level version of the horizontal sync pulse. The polarity of this pulse is programmable; its trailing edge is used by the On Screen Display facility as the reference for horizontal positioning.
VSYNC	27	I	**Vertical Sync:** A dedicated input for a TTL-level version of the vertical sync pulse. The polarity of this pulse is programmable, and either edge can serve as the reference for vertical timing.
VID2:0	22–24	O	**Digital Video bus:** Three totem pole outputs comprising digital RGB (or other color encoding) from the On Screen Display facility. The polarity of these outputs is controlled by a programmable register bit.
VCTRL	25	O	**Video Control:** A totem-pole output indicating whether the On Screen Display facility is currently presenting active video on the VID2:0 outputs. This signal should be used to control an external multiplexer (mixer) between normal video and the video derived from VID2:0. The polarity of this outputs is controlled by a programmable register bit.
BF	30	O	**Background/Foreground:** A totem-pole output which, when VCTRL is active, indicates whether the current video data represents a foreground (low) or background (high) dot in a character. This signal can be used to reduce the intensity of the background color and thus emphasize the text. If a 40-pin version of this part is ever produced, BF will not be pinned out.
P0.0-P0.7	1–8	I/O	**Port 0:** An 8-bit open-drain bidirectional port. Port 0 pins that have ones written to them float, and in that state can be used as high-impedance inputs. The port 0 pins can also serve as outputs from the high-precision Pulse Width Modulator (TDAC) and seven of the eight lower-precision Pulse Width Modulator functions. For each PWM block, a register bit controls whether the corresponding pin is controlled by the block or by port 0; port 0 controls the pin immediately after a Reset. Regardless of how each pin is controlled, it can be externally pulled up as high as +12V±5%, and the state of the pin can be read from the Port 0 register by the program.
	1	I	**V_{PP} (P0.0)** – This pin receives the 12V programming supply voltage during EPROM programming.
	2	I	**PROG (P0.1)** – This pin receives the programming pulses during EPROM programming.
	3	I	**ASEL (P0.2)** – Input which indicates which bits of the EPROM address are applied to port 2.
	1	O	**TDAC (P0.0)** – This is the output for the 14-bit high-precision PWM.
	2–8	O	**PWM1–7 (P0.1–P0.7)** – Outputs for the 6-bit PWMs 1 through 7.
P1.0-P1.3	9–12	I/O	**Port 1:** A 4-bit open-drain bidirection port. Port 1 pins that have ones written to them float, and in that state can be used as high-impedance inputs. P1.3 can also serve as the eighth lower-precision Pulse Width Modulator output (PWM0), and can be externally pulled up as high as +12V±5%. P1.2:0 have optional alternate use as ADI2:0, inputs to the Software A/D conversion faciiity. If a 40-pin version of this part is ever produced, P1.3/PWM0 will not be pinned out. Any of the port 1 pins are driven low if the corresponding port register bit is written as 0, or, for P1.3 only, if the TDAC module presents a 0. The state of the pin can always be read from the port register by the program.
	9–11	I	**ADI0–2 (P1.0–P1.2)** – Inputs for the software A/D facility.
	12	O	**PWM0 (P1.3)** – Output for the PWM0 6-bit PWM.
P2.0-P2.7	20–13	I/O	**Port 2:** An 8-bit open-drain bidirectional port. Port 2 pins that have ones written to them float, and in that state can be used as high-impedance inputs. P2.3:0 have high current capability (10 mA at 0.5V) for LEDs. Any of the port 2 pins are driven low if the port register bit is written as 0. The state of the pin can always be read from the port register by the program.

PIN DESCRIPTIONS (Continued)

MNEMONIC	PIN NO. DIP	TYPE	NAME AND FUNCTION
P3.0-P3.7	34–42	I/O	**Port 3:** An 8-bit open-drain bidirectional port. Port 3 pins that have ones written to them float, and in that state can be used as high-impedance inputs. P3.0, P3.4, and P3.7 can be externally pulled up as high as +12V±5%, while P3.5 and P3.6 have 10mA drive capability. Some of the port 3 pins can also serve alternate functions, as follows:
	35	I	**INT1 (P3.1)** – External Interrupt 1.
	36	I	**T0 (P3.2)** – Timer 0 external input.
	37	I	**INT0 (P3.3)** – External Interrupt 0.
RST	33	I	**Reset:** If this pin is high for two machine cycles (24 oscillator periods) while the oscillator is running, the MTV is reset. Also, this pin is used as a serial input to enter a test or EPROM programming mode, as on the 87C751.
XTAL1	31	I	**Crystal 1:** Input to the inverting oscillator amplifier and clock generator circuit that provides the timing reference for all MTV logic other than the OSD facility. XTAL1 and XTAL2 can be used with a quartz crystal or ceramic resonator to provide an on-chip oscillator. Alternatively, XTAL1 can be conncected to an external clock, and XTAL2 left unconnected.
XTAL2	32	O	**Crystal 2:** Output from the inverting oscillator amplifier.
V_{CC}		I	**Power Supply:** This is the power supply for normal and power-down modes.
V_{SS}		I	**Ground:** 0V reference.

COMPARISON TO THE 80C51

The elements of the MTV are shown in the Block Diagram. The features of the MTV are identical to those of the 80C51, except as noted herein.

Pinout and Testing

Since neither data nor program memory is externally expandable on the MTV, the 80C51 pins ALE, $\overline{EA}$, and $\overline{PSEN}$ are not implemented on the MTV.

I/O Ports

On both the 80C51 and the MTV, port 0 is open-drain, but on the 80C51 it can be used for external memory expansion while on the MTV its alternate use is for Pulse Width Modulated outputs.

On the 80C51, port 1 is 8 bits, is mostly unallocated (general purpose), and is quasi-bidirectional (that is, having a weak pullup transistor that can be overdriven). On the MTV it is a 4-bit open-drain port, and includes alternate uses for analog inputs and a PWM output.

On the 80C51, port 2 is quasi-bidirectional and can be used for external memory expansion; on the MTV, port 2 is open-drain and unallocated.

On the 80C51, port 3 is quasi-bidirectional and all eight bits have alternate uses. On the MTV, three port 3 bits have some of the same alternate uses as on the 80C51 but not necessarily on the same pins, while five pins are open-drain and unallocated.

Idle Mode

The idle mode is not implemented on the MTV.

Power-Down Mode

The power-down mode of the 80C51 is retained without change. The PCON register has the following format::

PCON

7	6	5	4	3	2	1	0
–	–	–	–	GF1	GF0	PD	–

Interrupts

The interrupt facilities of the MTV differ from those of the 80C51 as follows:

1. Since there is not a serial port, there are no interrupts nor control bits relating to this interrupt. The interrupts and their vector addresses are as follows:

Event	Program Memory Address
Reset	000
External INT0	003
Timer 0	00B
External INT1	013
Timer 1	01B
VSync Start	023

2. The VSYNC input used by the On Screen Display facility can generate an interrupt. The active polarity of the pulse is programmable, as described in a later section. The interrupt occurs at the leading edge of the pulse.

3. External Interrupt 1 is modified so that an interrupt is generated when the input switches in either direction (on the 8051, there is a programmable choice between interrupt on a negative edge or a low level on INT1). This facility allows for software pulse-width measurement handling of a remote control.

4. The IP register is not used, and the IE register is similar to that on the 80C51:

IE

7	6	5	4	3	2	1	0
EA	–	–	EVS	ET1	EX1	ET0	EX0

Six-Bit PWM DACs

The structure of these modules is shown in Figure 2. First, the basic MCU clock is divided by 4 to get a waveform that clocks a 6-bit counter which is common to all the PWMs, including the 14-bit one. This divided clock is hereafter called the PWM counter clock.

Each PWM block has a special function register PWMn arranged as follows:

PWM0-PWM7

7	6	5	4	3	2	1	0
PWE	–	PV5	PV4	PV3	PV2	PV1	PV0

If the PWE bit for a particular PWM block is 1, the block is active and controls its assigned port pin; if PWE is 0 the corresponding port pin is controlled by the port. The "value" field (PV5 ... PV0) of each PWM register is compared to (the LS 6 bits of) the common counter. When the value matches, the output FF is cleared, so that the output pin is driven low. When the value rolls over to zero, the output FF is set, so that the output pin is released. Thus the output waveform has a fixed period of 64 PWM counter clocks; its duty cycle is determined by PWMn.5:0.

Three of the nine total PWMs operate as described above; for three others, both the rising and falling edges of the output are delayed by one PWM clock; for the remaining three, both edges are delayed by two PWM clocks. This feature reduces the radio-frequency emission that would otherwise occur when the counter rolled over to zero and all nine open-drain outputs were released.

14-Bit PWM DAC (TDAC)

This feature was partially described in the preceding section. As shown in Figure 3, the 6-bit counter used for the lower precision PWMs is in fact the least significant part of a 14-bit counter used for this facility. The nature of the counter is such that it can achieve a stable output value through its MSB, and the value can propagate through logic like that shown in Figure 3, and the logic output can be stable within one period of the PWM counter clock (e.g., 250 ns) if edge-triggered logic is used to capture the logic output, or within one phase of the PWM counter clock (e.g., 125 ns) if a phase of the PWM counter clock is used to capture the logic output. For cost and die-size reasons, it is preferable that the TDAC counter be a ripple counter.

This feature is controlled by two special function registers:

TDACL

7	6	5	4	3	2	1	0
TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0

TDACH

7	6	5	4	3	2	1	0
TDE	–	TD13	TD12	TD11	TD10	TD9	TD8

When software wishes to change the 14-bit value (TD0 – TD13), it should first write TDACL and then write TDACH. Alternatively, if the required precision of the duty cycle is satisfied by 6 bits or less, software can simply write TDACH. Note from Figure 3 that this block includes an "extra" 14-bit latch between TDACL/H and the comparator and other logic. The programmed value is clocked into the operative latch when the 7 low-order bits of the counter roll over to zero, provided that the software is not in the midst of loading a new 14-bit value (that is, it is not between writing TDACL and writing TDACH).

In a similar fashion to the lower-precision PWMs, this facility has an output FF that is set when the lower 7 bits of the counter overflow/wrap. The *more* significant 7 bits of the operative latch's programmed value are compared for equality against the *less* significant 7 bits of the counter, and the output FF is cleared when they match. Thus this output has a fixed period of 128 PWM counter clocks, and the duty cycle is determined by the programmed value.

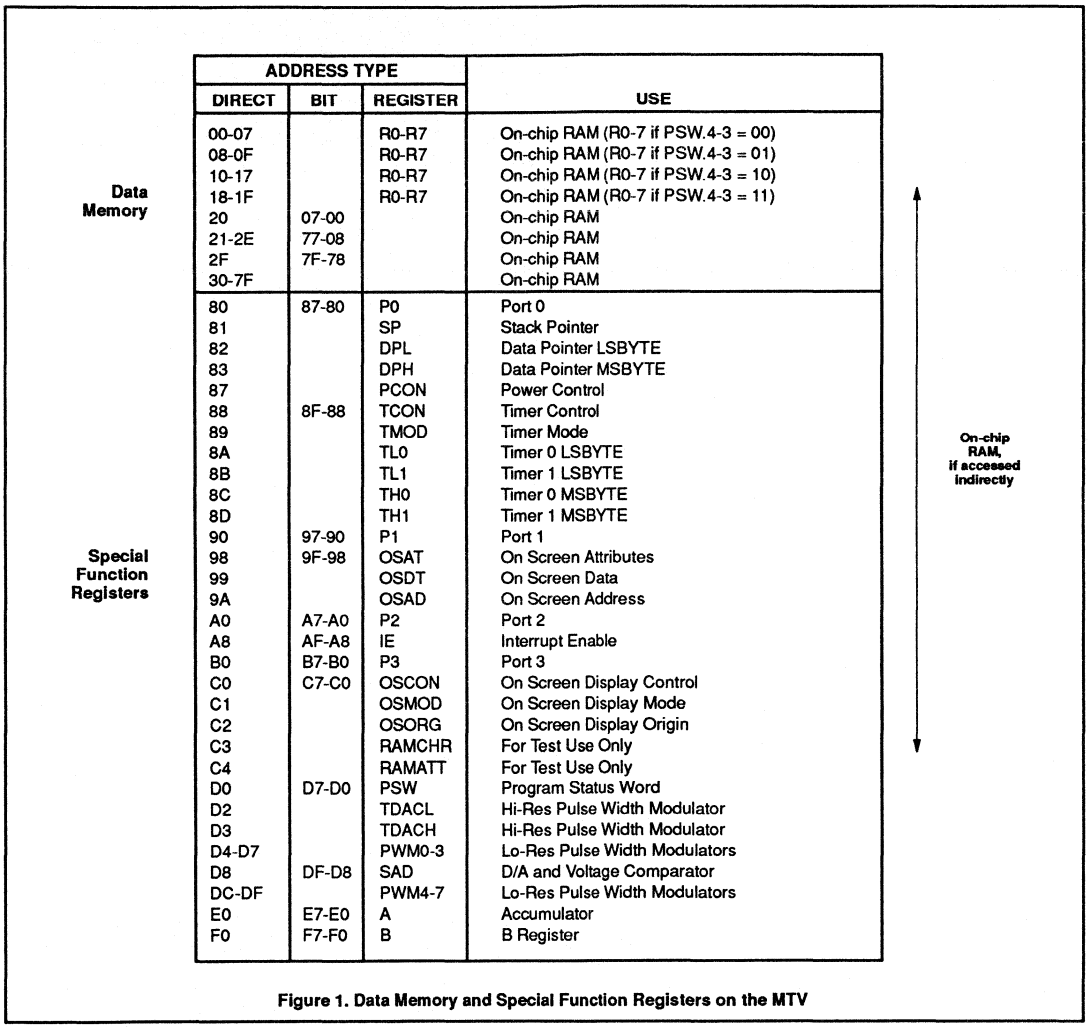
For the higher-precision aspect of this feature, the 7 more-significant bits of the counter are used in a logic block with the 7 less-significant bits of the programmed value. The 7th LSB (binary value 64) of the programmed value is ANDed with the 7th MSB (128) of the counter, the 6th LSB of the value is ANDed with the counter's 6th and 7th MSBs being 10, and so on through the LSB of the programmed value being ANDed with the counter's 7MSBs being 100000. Then these 7 ANDed terms are ORed. If the result is true/1 at the time the 7 LSBs of the counter match the MSBs of the programmed value, the output is forced high for 1 (additional) PWM counter clock.

The result is that, if the value-64 bit of the 14-bit value is programmed to 1, every other cycle of 128 PWM counter clocks has its duty cycle stretched by one counter clock; if the value-32 bit is programmed to 1, every 4th cycle is stretched, and so on through, if the value-1 bit is programmed to 1, one cycle out of each 128 is stretched.

Assuming the external integrator can handle all this, the net effect is a PWM DAC that has the period of a 7-bit design (which makes the integrator easier and more feasible to design) with the accuracy of a 14-bit one. There is some question whether all of the least significant bits can be effectively integrated, or whether they simply act as a source of ripple in the integrated voltage. An obvious prerequisite for such precision is that the load on the voltage must be very light, like a single op amp or comparator.

The TDAC feature differs from the corresponding features of predecessor parts in several ways:

1. The 14-bit value is functionally composed of major and minor portions of 7 bits each.
2. The 14-bit value is programmed as a contiguous multi-register value that can be manipulated straight-forwardly via arithmetic instructions.
3. As discussed for the 6-bit DACs, both of the preceding parts had a feature whereby the PWM output could be inverted, redundantly with complementing the 14-bit value. This feature has been eliminated.



	ADDRESS TYPE			USE
	DIRECT	BIT	REGISTER	
Data Memory	00-07		R0-R7	On-chip RAM (R0-7 if PSW.4-3 = 00)
	08-0F		R0-R7	On-chip RAM (R0-7 if PSW.4-3 = 01)
	10-17		R0-R7	On-chip RAM (R0-7 if PSW.4-3 = 10)
	18-1F		R0-R7	On-chip RAM (R0-7 if PSW.4-3 = 11)
	20	07-00		On-chip RAM
	21-2E	77-08		On-chip RAM
	2F	7F-78		On-chip RAM
	30-7F			On-chip RAM
Special Function Registers	80	87-80	P0	Port 0
	81		SP	Stack Pointer
	82		DPL	Data Pointer LSBYTE
	83		DPH	Data Pointer MSBYTE
	87		PCON	Power Control
	88	8F-88	TCON	Timer Control
	89		TMOD	Timer Mode
	8A		TL0	Timer 0 LSBYTE
	8B		TL1	Timer 1 LSBYTE
	8C		TH0	Timer 0 MSBYTE
	8D		TH1	Timer 1 MSBYTE
	90	97-90	P1	Port 1
	98	9F-98	OSAT	On Screen Attributes
	99		OSDT	On Screen Data
	9A		OSAD	On Screen Address
	A0	A7-A0	P2	Port 2
	A8	AF-A8	IE	Interrupt Enable
	B0	B7-B0	P3	Port 3
	C0	C7-C0	OSCON	On Screen Display Control
	C1		OSMOD	On Screen Display Mode
	C2		OSORG	On Screen Display Origin
	C3		RAMCHR	For Test Use Only
	C4		RAMATT	For Test Use Only
	D0	D7-D0	PSW	Program Status Word
	D2		TDACL	Hi-Res Pulse Width Modulator
	D3		TDACH	Hi-Res Pulse Width Modulator
	D4-D7		PWM0-3	Lo-Res Pulse Width Modulators
	D8	DF-D8	SAD	D/A and Voltage Comparator
	DC-DF		PWM4-7	Lo-Res Pulse Width Modulators
	E0	E7-E0	A	Accumulator
	F0	F7-F0	B	B Register

On-chip RAM, if accessed indirectly

Figure 1. Data Memory and Special Function Registers on the MTV

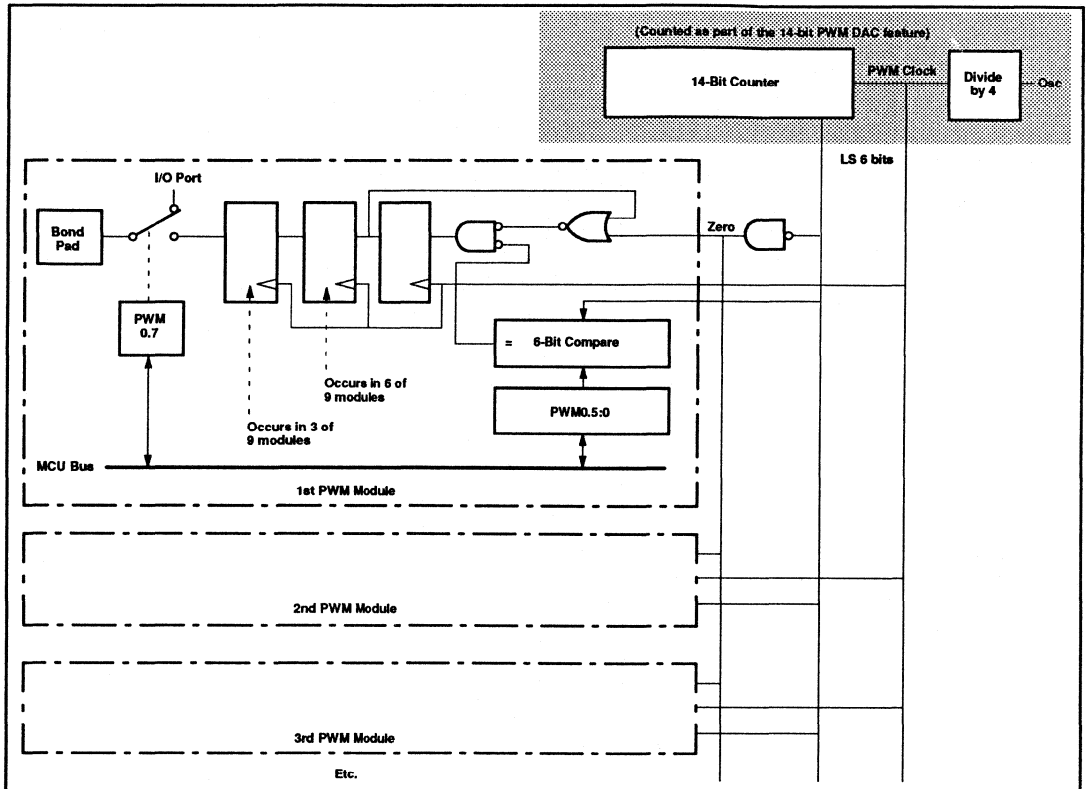


Figure 2. 6-Bit PWM Logic

Software A/D Facility

This facility is shown in Figure 4. It represents an alternate use whereby any of the P1.0 through P1.2 pins can be selected as one input of a linear voltage comparator. The block includes one special function register:

SAD

7	6	5	4	3	2	1	0
VHi	CH1	CH0	St	SAD3	SAD2	SAD1	SAD0

As shown in Figure 4 the other input of the comparator is connected to a 4-bit D/A that is controlled by the 4 LSBs of the SAD register, producing a reference voltage nominally 0.15625V to 4.84375V by steps of 0.3125V. The output of the comparator (high/low) can be read by the program as the MSB of the register, which is bit addressable.

The St bit should be written as 1 in order to initiate a voltage comparison. After writing St=1, the program should include intervening instructions totalling at least six machine cycles (72 CLK periods or 6 microseconds at 12MHz), before the instruction that accesses and tests VHi.

The chan field controls which pin, if any, is connected to this facility:

CH1	CH0	pin
0	0	none
0	1	P1.0
1	0	P1.1
1	1	P1.2

Port 1 has open-drain drivers which will not materially affect an analog voltage as long as any and all pins used for software A/D measurement have corresponding ones in the port register.

On Screen Display (OSD) Module

This block is the largest of the additions that are specific to this product. Its basic function is to superimpose text on the television video image, to indicate various parameters and settings of the receiver or tuner. External circuitry handles the mixing (multiplexing) of the text and the TV video.

The overall OSD block has four input pins: two for a video clock, plus the horizontal and vertical sync signals. The video clock pins are used to connect an LC circuit to an on-chip video oscillator that is independent of the normal MCU clock. The L and C values are chosen so that a video pulse, of a duration equal to the VCLK period, will produce a more-or-less square dot on the screen, that is, a dot having a width approximately equal to the vertical distance between consecutive scan lines.

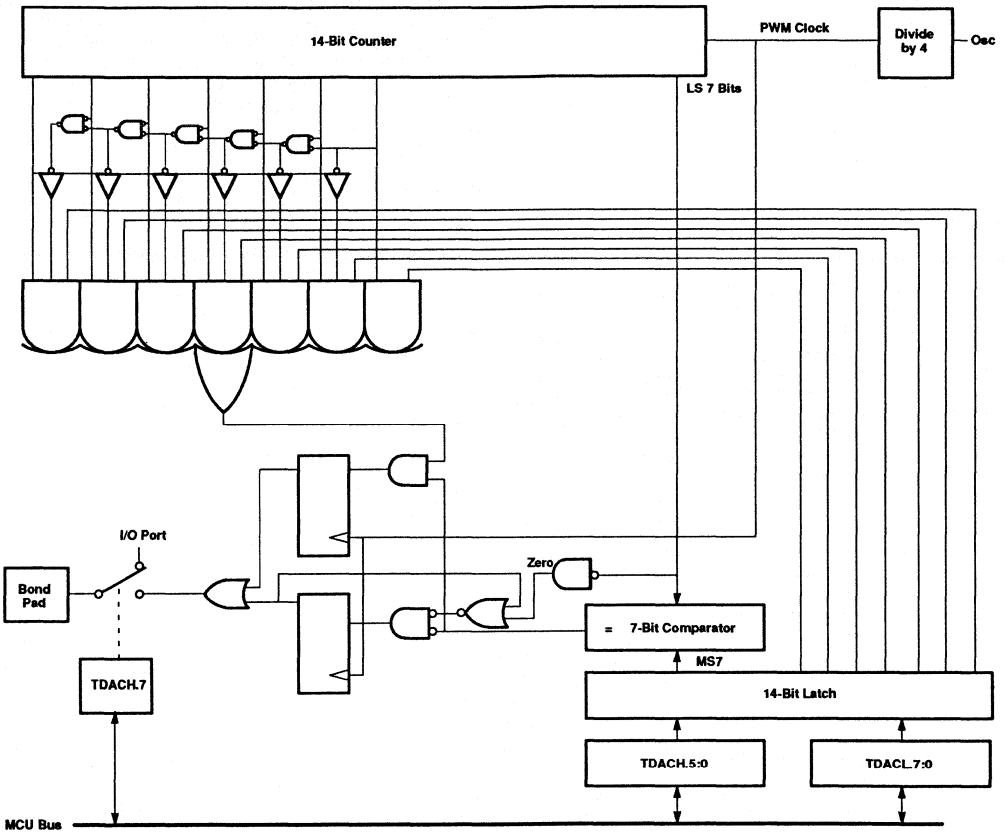


Figure 3. 14-Bit PWM (TDAC) Logic

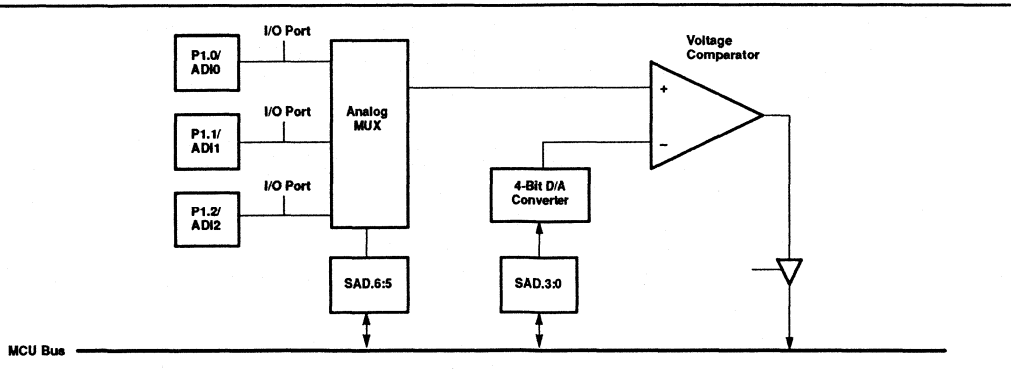


Figure 4. Software A/D Facility

The video oscillator is stopped (with VCLK2 low) while horizontal sync is asserted, and is released to operate at the trailing edge of horizontal sync. This technique helps provide uniform horizontal positioning of characters/dots from one scan line to the next.

The block has four outputs, three color video signals, and a control signal. Since this block is the major feature of the part, its main inputs and outputs are dedicated pins, without alternate port bits.

Display RAM

The OSD of the MTV differs from that in preceding devices in one major way: It does not fix the number and size of displayed rows of text. Several predecessor parts allowed two displayed rows of 16 characters each. The MTV simply has 128 locations of display RAM, each of which can contain a displayed character or a New Line character that indicates the end of a row. A variant of the New Line character is used to indicate the end of displayed data.

The three major elements of the OSD facility are shown in the Block Diagram. Each display RAM location includes 6 data bits and 4 attribute bits. The 6 data bits from display RAM, along with a line-within-row count, act as addresses into the character generator ROM, which contains 60 displayable bit maps (64 minus one for each of New Line and three Space characters). Each bit map includes 18 scan lines by 14 dots. The character generator ROM is maskable or programmable along with the program ROM to allow for various character sets and languages.

The programming interface to display RAM is provided by three special function registers:

OSAD

7	6	5	4	3	2	1	0
–	OSAD6	OSAD5	OSAD4	OSAD3	OSAD2	OSAD1	OSAD0

OSDT

7	6	5	4	3	2	1	0
–	–	OSDT5	OSDT4	OSDT3	OSDT2	OSDT1	OSDT0

OSAT (with OSDT = New Line)

7	6	5	4	3	2	1	0
–	–	–	E	–	SR	D	Sh

OSAT (with OSDT = BSpace or SplitBSpace)

7	6	5	4	3	2	1	0
–	–	–	B	–	BC2	BC1	BC0

OSAT (with OSDT = any other)

7	6	5	4	3	2	1	0
–	–	–	B	–	FC2	FC1	FC0

OSAD ("On Screen ADress") contains the address at which data will next be written into display RAM, while the ten active bits in OSDT ("On Screen DaTa") plus OSAT ("On Screen ATtributes") correspond exactly to the 10 bits in each display RAM location. FColor indicates the color of foreground (1) pixels in the ROM bit map for this character, while B indicates whether background (0) pixels should show the current background color (B=1) or television video (B=0). Thus, for the 1 bits in a character's bit map, the VID2:0 pins are driven with (FColor) and VCTRL is driven active, while for 0 bits VID2:0 are driven with the background color (except for shadow bits) and VCTRL is driven with the B bit.

Writing OSAT simply latches the attribute bits into a register, while writing OSDT causes the data bus information, plus the contents of the OSAT register, to be written into display RAM. Thus, for a given display RAM location, OSAT should be written before OSDT. If successive characters are to be written into display RAM with the same attributes, OSAT need not be rewritten for each character, only prior to writing OSDT for the first character with those particular attributes.

In reality, there is a potential conflict between the timing of a write to OSDT and an access to display RAM by the OSD logic for data display. This is resolved by the use of a true dual-ported RAM for display memory.

OSAD is automatically incremented by one after each time OSDT and display RAM are written. Except in special test modes that are beyond the scope of this spec release, display RAM cannot be read by the MCU program.

The OSAT attribute bits associated with the BSpace (data=111110), SplitBSpace (111111), and New Line (111101) characters are interpreted differently from those that accompany other data characters. With BSpace and SplitBSpace, B is interpreted as described above, but the 3 color bits specify the Background color (BColor) for subsequent characters. For BSpace, a change in B and BColor becomes effective at the left edge of the character's bit map. For SplitBSpace, a change in B and BColor occurs halfway through the character horizontally. The normal Space character (111100) has no effect on the BColor value.

BColor values 000 and 111 minimize the occurrence of transient states among the VID2:0 outputs.

The background color defined by the most recently encountered BSpace or SplitBSpace character is maintained on the VID2:0 pins except at the following times:

1. During the active time of HSYNC,
2. During the active time of VSYNC,
3. During those pixels of an active character that correspond to a 1 in the character's bit map,
4. During a "shadow" bit.

The BColor value is not cleared between vertical scans, so that if a single background color is all that is needed in an application, it can be set via a single BSpace character during program initialization, and never changed thereafter. In order for such a BSpace to actually affect the MTV's internal BColor register the Mode field of the OSMOD register must be set to 01 (or higher) so that the OSD hardware is operating.

With a New Line character, if the E bit is 1, no further rows are displayed on the screen. If E is 0 and D is 1, all of the characters in the following row are displayed with Double height and width. If E is 0 and Sh is 1, all of the characters in the following row are displayed with shadowing, as described in a later section. If E is 0 and SR is 1, the next row is a "short row": It is only 4 or 8 scan lines high rather than 18 or 36. Short rows can be used for underlined text.

The latches in which the E, D, Sh, and SR bits are captured are cleared to zero at the start of each vertical scan. This means that if the first text line on the screen is a short row, or if it contains either double size or shadowing, the text must be preceded by a New Line character. Like all such characters, this initial New Line advances the vertical screen position; the VStart value (see below) should take this fact into account.

Other OSD Registers

A number of changes in the OSD architecture have reduced the number of other special function registers involved in the feature, below the number needed with predecessor devices:

1. The elimination of certain options that are present in the 47C634 and 84C640, such as 4, 6, or 8X character sizes and alternate use of two of the video outputs.
2. The moving of certain other options from central registers to display RAM, such as foreground color codes and background selection.

OSCON

7	6	5	4	3	2	1	0
IV	Pv	Lv	Ph	Pc	Po	DH	BFe

The IV bit is the interrupt flag for the OSD feature. It is set by the leading edge of the VSYNC pulse, and is cleared by the hardware when the VSYNC interrupt routine is vectored to. It can also be set or cleared by software writing a 1 or 0 to this bit.

NOTE

It is theoretically possible that a VSYNC interrupt could be missed, or an extra one generated, if OSCON is read, then modified internally (e.g., in Ac), and the result written back to OSCON. However, none of the other bits in OSCON are reasonable candidates for dynamic change. Special provisions are included in the MTV logic so that IV will not be changed by a single "read-modify-write" instruction such as SETB or CLR, unless the instruction specifically changes IV.

A 0 (1) in Pv designates that the VSYNC input is high-active (low-active). One effect of this bit is that the VID2:0 and VCTRL outputs are blocked (held at black/inactive) during the active time of VSYNC. The IV bit is set on the leading edge of the VSYNC pulse; thus Pv controls whether the OSD interrupt occurs in response to a high-to-low or low-to-high transition on VSYNC.

A 0 (1) in Lv designates that the leading edge (active level) of VSYNC, as defined by Pv, clears the state counter that is used to determine the vertical start of on-screen data. In effect, Lv=0(1) says that the leading (trailing) edge of VSYNC is the time reference for the video field.

A 0 (1) in Ph designates that the HSYNC input is high-active (low-active).

A 0 (1() in Pc designates that a high (low) on the VCTRL output means "show the color on VID2:0".

A 0 (1) in Po designates that a 0 (1) internal to the MTV corresponds to a low on one of the VID2:0 pins. This control bit is needed only because the Shadowing feature needs to generate black pixels without reference to a register value: Internally, the 3-bit code 000 always designates black.

If DH is 1, character sizes are doubled vertically but not horizontally. This feature allows the MTV to be used in "improved definition" systems that are not interlaced. The vertical doubling imposed by DH does not affect the VStart logic described below: It operates in HSync units regardless of DH or D.

If BFe is 1, the BF output tracks whether each bit in displayed characters is a foreground bit (low) or a background bit (high). If BFe is 0, the BF pin remains high.

OSORG

7	6	5	4	3	2	1	0
HS4	HS3	HS2	HS1	HS0	VS2	VS1	VS0

The HStart field (HS4 – HS0) defines the left end (start) of all of the on-screen character rows, as a multiple of four VCLKs. Active display begins 4(HStart+1) VCLKs plus one single-sized character width after the trailing edge of HSYNC. Counting variations in Wc, there may be 16 to 142 VCLKs from the end of HSYNC to the start of the first character of each row.

The VStart field (VS2 – VS0) defines the top (start) of the first on-screen character row, as a multiple of four HSYNC pulses. Active display begins 4(VStart+1) HSYNCs after the field's time reference point, a range of 4 to 32. Subsequent character rows occur directly below the first, such that the last scan line of one row is directly followed by the first scan line of the next row. Successive New Line characters (with or without the Short Row designation) can be used to vertically separate text rows on the screen.

Neither the HStart nor VStart parameter is affected by the D line attribute that is used to display double-sized characters.

OSMOD

7	6	5	4	3	2	1	0
Wc	–	Mode1	Mode0	–	SHM2	SHM1	SHM0

If the mode bits (Mode 1, Mode 0) are 00, the OSD feature is disabled. The VCLK oscillator is disabled, VID2:0 are set to black, and VCTRL is held inactive.This is the mode to which the MTV OSD logic is reset. A direct transition from this mode to active display (1x) would result in undefined operation and visual effects for the duration of the current video field (until the next VSYNC).

If the mode is 01, the VCLK oscillator is enabled and the OSD logic operates normally internally, but VID2:0 are set to black and VCTRL is held inactive. The OSD feature can be toggled between this state and 1x as desired to achieve real-time special effects such as "vertical wiping."

Mode 10 represents normal OSD operation. Active characters can be shown against TV video (for characters with B=0) or (for characters with B=1) against a background of the color defined as an attribute of BSpace and SplitBSpace characters.

In mode 11, characters can be displayed but all of the receiver's normal video is inhibited by holding VCTRL asserted throughout the active portion of each scan line. Since VID2:0 are driven with the current background color during this time, except during the foreground portion of displayed characters, this produces text against a solid background. This mode is useful for extensive displays that require user concentration.

If Wc is 1, then each displayed character is horizontally terminated after 12 bits have been output, as opposed to after 14 bits if Wc is 0. This allows text to be "packed" more tightly so that more characters can be displayed per line. In effect, the 2 bits out of the display ROM, which would otherwise be the rightmost 2 of the 14, are ignored when Wc is 1. Clearly, if this feature is to be used, it must be accounted for in the design of the bit maps in the display ROM.

The 3-bit ShMode field (SHM2 – SHM0) determines how characters are shadowed in rows for which the SH row attribute is 1. As shown in Figure 5, the values 000-110 indicate an apparent light source position ranging from the lower left clockwise to the lower right, while the value 111 indicates full-surround shadowing.

DC ELECTRICAL CHARACTERISTICS

V_{CC} = 5V ±10%, T_A = 0°C to +70°C

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT	NOTES
V_{IL}	Input low voltage		–0.5	$0.2V_{CC}$–0.1	V	
V_{IH1}	Input high voltage (P1.2:0, P2.7:0, P3.6:5, P3.3:1, VSYNC, HSYNC)		$0.2V_{CC}$+0.9	V_{CC}+0.5	V	
V_{IH2}	Input high voltage (port 0, P1.3, P3.7, P3.4, P3.0)	IIH = 2mA	$0.2V_{CC}$+0.9	12.6	V	
V_{IH} – V_{CC}	Input high voltage (port 0, P1.3, P3.7, P3.4, P3.0) with respect to V_{CC}			8	V	1
V_{IH}	Input high voltage (XTAL1, VCLK1, RST)		$0.7V_{CC}$	V_{CC}+0.5	V	
V_{OL1}	Output low voltage (P2.3:0, P3.6:5)	I_{OL} = 10mA		0.5	V	
V_{OL2}	Output low voltage (TDAC, PWM0:7)	I_{OL} = 700µA		0.5	V	2
V_{OL3}	Output low voltage (all other outputs)	I_{OL} = 1.6mA		0.45	V	
V_{OH}	Output high voltage (port 1, VID2:0, VCTRL, BF)	I_{OH} = –60µA	2.4		V	
R_{RST}	Reset pulldown resistor		50	300	kΩ	
C_{IO}	Pin capacitance	Test freq = 1MHz, TA = 25°C		10	pF	
I_{PD}	Power-down current	V_{CC} = 2 to 6V		5	mA	
I_{CC}	Normal mode supply current	V_{CC} = 5.5V		30	mA	3

NOTES:
1. This maximum applies at all times, including during power switching, and must be accounted for in power supply design. During a power-on process, the +12 volt source used for external pullup resistors should not precede the V_{CC} of the MTV up their respective voltage ramps by more than this margin, nor, during a power-down process, should V_{CC} precede +12V down their respective voltage ramps by more than this margin.
2. The specified current rating applies when any of these pins is used as a Pulse Width modulated output. For use as a port output, the rating is as given subsequently.
3. I_{CC} measured with OSD block initialized.

AC ELECTRICAL CHARACTERISTICS

V_{CC} = 5V ±10%, T_A = 0°C to +70°C

SYMBOL	PARAMETER	TENTATIVE LIMITS MIN	TENTATIVE LIMITS MAX	UNIT	NOTES
$1/t_{CLCL}$	XTAL Frequency	6	12	MHz	4
t_{CHCX}	XTAL1 Clock high time	20		ns	5
t_{CLCX}	XTAL1 Clock low time	20		ns	5
t_{CLCH}	XTAL1 Clock rise time		20	ns	5
t_{CLCL}	XTAL1 Clock fall time		20	ns	5
$1/t_{VCLCL}$	VCLK Frequency	5	8	MHz	
$\lvert t_{VCOH}–t_{VCOL} \rvert$	Rise vs. fall time skew on any one of VID2:0, VCTRL, BF		40	ns	6
$\lvert t_{VCOH1}–t_{VCOH2} \rvert$	Rise time skew between any two of VID2:0, VCTRL, BF		10	ns	6
$\lvert t_{VCOL1}–t_{VCOL2} \rvert$	Fall time skew between any two of VID2:0, VCTRL, BF		10	ns	6

NOTES:
4. The MTV is tested at its maximum XTAL frequency, but not at any other (lower) rate.
5. These parameters apply only when an external clock signal is used.
6. These parameters assume equal loading at C_L = 100pF, for all the referenced outputs.

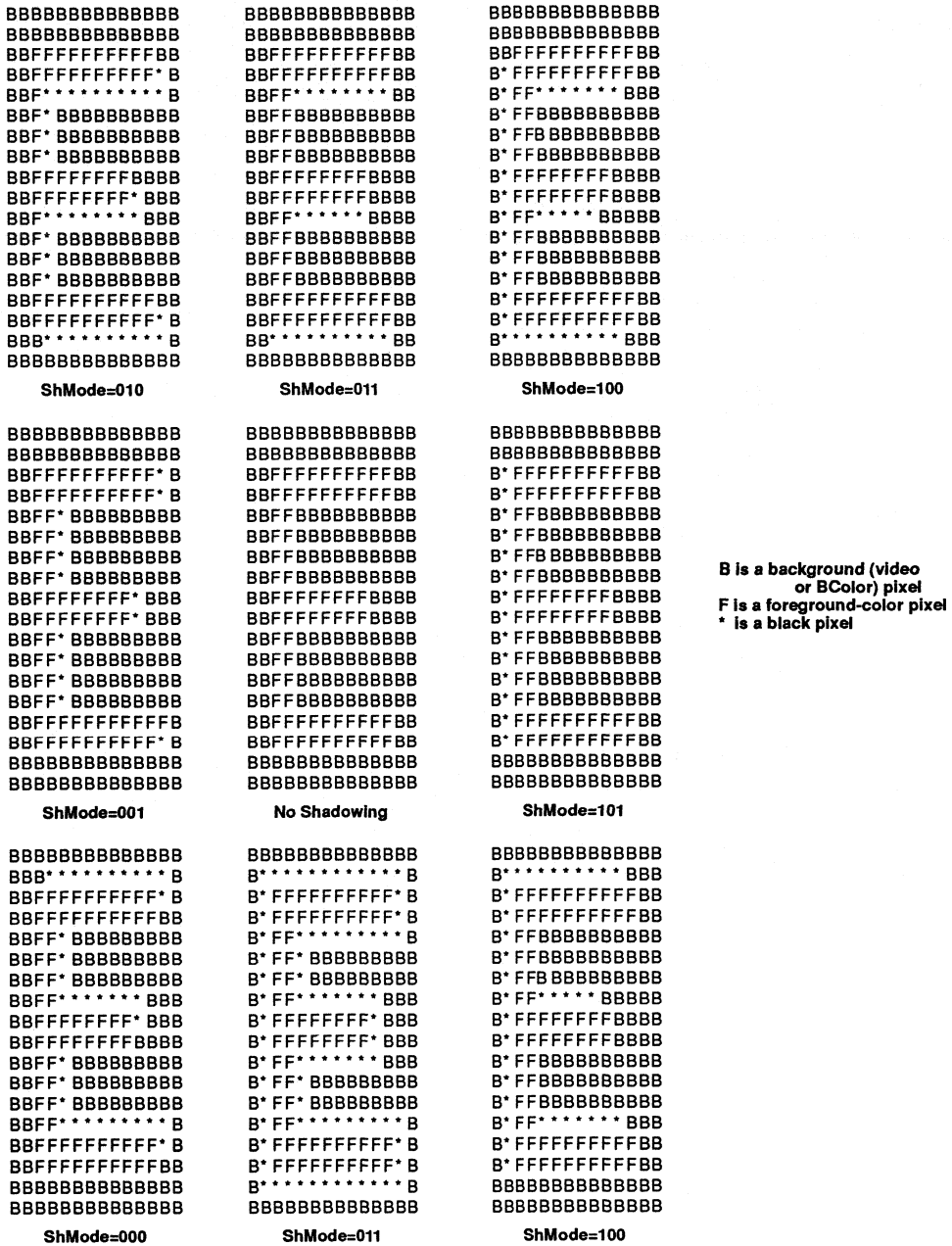


Figure 5. Effect of Shadowing on the Letter "E"

PROGRAMMING CONSIDERATIONS

EPROM Characteristics

The 87C054 is programmed by using a modified Quick-Pulse Programming algorithm similar to that used for devices such as the 87C51. It differs from these devices in that a serial data stream is used to place the 87C751 in the programming mode.

Figure 6 shows a block diagram of the programming configuration for the 87C054. Port pin P0.2 is used as the programming voltage supply input (V_{PP} signal). Port pin P0.1 is used as the program (PGM/) signal. This pin is used for the 25 programming pulses.

Port 3 is used as the address input for the byte to be programmed and accepts both the high and low components of the eleven bit address. Multiplexing of these address components is performed using the ASEL input. The user should drive the ASEL input high and then drive port 3 with the high order bits of the address. ASEL should remain high for at least 13 clock cycles. ASEL may then be driven low which latches the high order bits of the address internally. the high address should remain on port 3 for at least two clock cycles after ASEL is driven low. Port 3 may then be driven with the low byte of the address. The low address will be internally stable 13 clock cycles later. The address will remain stable provided that the low byte placed on port 3 is held stable and ASEL is kept low. **Note:** ASEL needs to be pulsed high only to change the high byte of the address.

Port 1 is used as a bidirectional data bus during programming and verify operations. During programming mode, it accepts the byte to be programmed. During verify mode, it provides the contents of the EPROM location specified by the address which has been supplied to Port 3.

The XTAL1 pin is the oscillator input and receives the master system clock. This clock should be between 1.2 and 6MHz.

The RESET pin is used to accept the serial data stream that places the 87C054 into various programming modes. This pattern consists of a 10-bit code with the LSB sent first. Each bit is synchronized to the clock input, X1.

Programming Operation

Figures 7 and 8 show the timing diagrams for the program/verify cycle. RESET should initially be held high for at least two machine cycles. P0.1 (PGM/) and P0.2 (V_{PP}) will be at V_{OH} as a result of the RESET operation. At this point, these pins function as normal quasi-bidirectional I/O ports and the programming equipment may pull these lines low. However, prior to sending the 10-bit code on the RESET pin, the programming equipment should drive these pins high (V_{IH}). The RESET pin may now be used as the serial data input for the data stream which places the 87C054 in the programming mode. Data bits are sampled during the clock high time and thus should only change during the time that the clock is low. Following transmission of the last data bit, the RESET pin should be held low.

Next the address information for the location to be programmed is placed on port 3 and ASEL is used to perform the address multiplexing, as previously described. At this time, port 1 functions as an output.

A high voltage V_{PP} level is then applied to the V_{PP} input (P0.2). (This sets Port 1 as an input port). The data to be programmed into the EPROM array is then placed on Port 1. This is followed by a series of programming pulses applied to the PGM/ pin (P0.1). These pulses are created by driving P0.1 low and then high. This pulse is repeated until a total of 25 programming pulses have occurred. At the conclusion of the last pulse, the PGM/ signal should remain high.

The V_{PP} signal may now be driven to the V_{OH} level, placing the 87C054 in the verify mode. (Port 1 is now used as an output port). After four machine cycles (48 clock periods), the contents of the addressed location in the EPROM array will appear on Port 1.

The next programming cycle may now be initiated by placing the address information at the inputs of the multiplexed buffers, driving the V_{PP} pin to the V_{PP} voltage level, providing the byte to be programmed to Port1 and issuing the 26 programming pulses on the PGM/ pin, bringing V_{PP} back down to the V_C level and verifying the byte.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Flourless part number 2345–5 or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-s/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000µW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

Table 1. Implementing Program/Verify Modes

OPERATION	SERIAL CODE	P0.1 (PGM/)	P0.0 (V_{PP})
Program user EPROM	286H	–*	V_{PP}
Verify user EPROM	286H	V_{IH}	V_{IH}

NOTE:
*Pulsed from V_{IH} to V_{IL} and returned to V_{IH}.

EPROM PROGRAMMING AND VERIFICATION

T_A = 21°C to +27°C, V_{CC} = 5V ±10%, V_{SS} = 0V

SYMBOL	PARAMETER	MIN	MAX	UNIT
$1/t_{CLCL}$	Oscillator/clock frequency	1.2	6	MHz
t_{AVGL}*	Address setup to P0.1 (PROG–) low	10µs + 24t_{CLCL}		
t_{GHAX}	Address hold after P0.1 (PROG–) high	48t_{CLCL}		
t_{DVGL}	Data setup to P0.1 (PROG–) low	38t_{CLCL}		
t_{DVGL}	Data setup to P0.1 (PROG–) low	38t_{CLCL}		
t_{GHDX}	Data hold after P0.1 (PROG–) high	36t_{CLCL}		
t_{SHGL}	V_{PP} setup to P0.1 (PROG–) low	10		µs
t_{GHSL}	V_{PP} hold after P0.1 (PROG–)	10		µs
t_{GLGH}	P0.1 (PROG–) width	90	110	µs
t_{AVQV}**	V_{PP} low (V_{CC}) to data valid		48t_{CLCL}	
t_{GHGL}	P0.1 (PROG–) high to P0.1 (PROG–) low	10		µs
t_{SYNL}	P0.0 (sync pulse) low	4t_{CLCL}		
t_{SYNH}	P0.0 (sync pulse) high	8t_{CLCL}		
t_{MASEL}	ASEL high time	13t_{CLCL}		
t_{MAHLD}	Address hold time	2t_{CLCL}		
t_{HASET}	Address setup to ASEL	13t_{CLCL}		
t_{ADSTA}	Low address to address stable	13t_{CLCL}		

NOTES:
*Address should be valid at least 24t_{CLCL} before the rising edge of P0.2 (V_{PP}).
**For a pure verify mode, i.e., no program mode in between, t_{AVQV} is 14t_{CLCL} maximum.

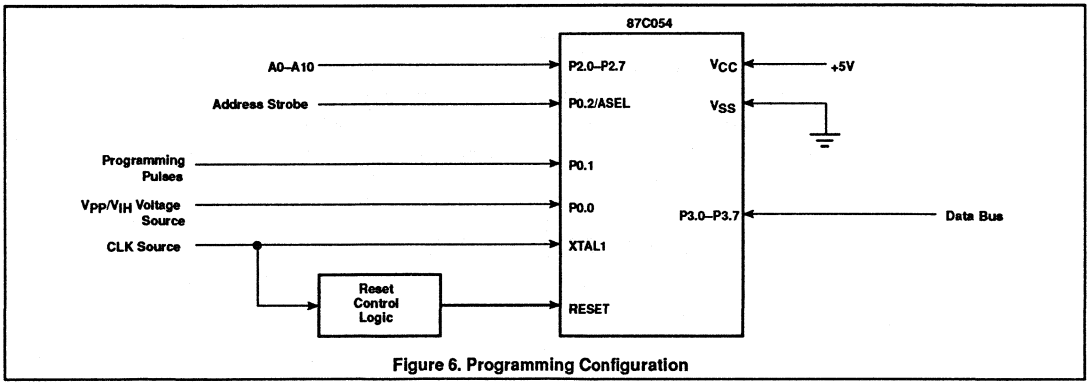


Figure 6. Programming Configuration

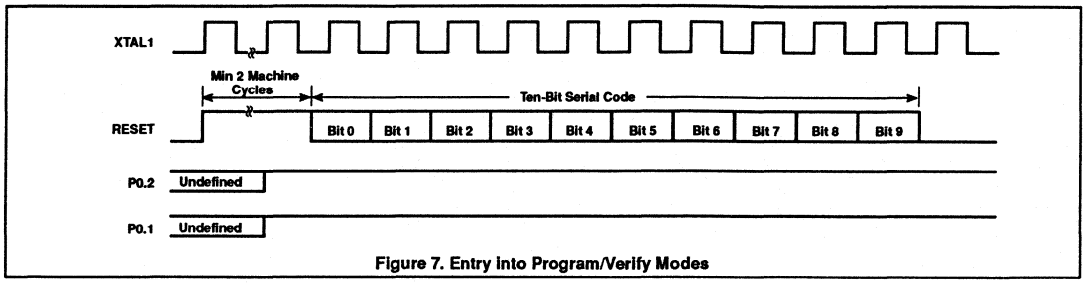


Figure 7. Entry into Program/Verify Modes

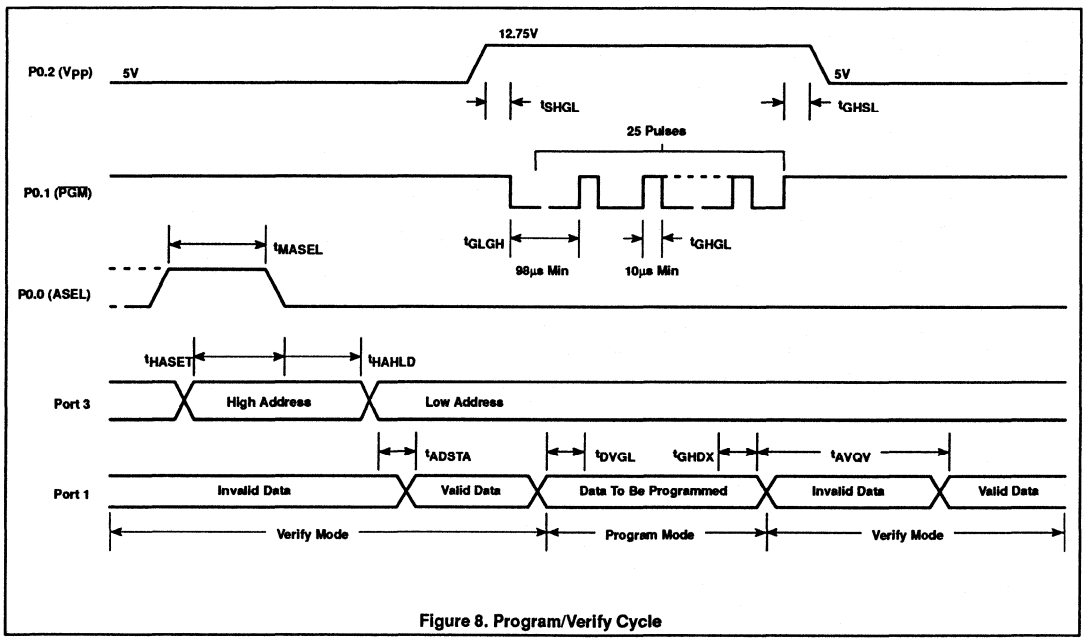


Figure 8. Program/Verify Cycle

8XCL410 OVERVIEW

The 80CL410 (ROMless version) and 83CL410 (ROM version) are referred to collectively in this overview as the 8XCL410. The 8XCL410 is socket compatible with the 80C51 and has many of the same features, but at a much lower power and clock frequency.

The 8XCL410 is the first member of a family of low-power 80C51 derivative parts. The features of the 8XCL410 include:

- 4k x 8 ROM
- 128 x 8 RAM
- Two 16-bit timer/counters
- A two level nested priority interrupt structure
- An I^2C serial interface
- Thirteen interrupt sources (with eight additional external interrupts)
- Idle and power-down modes
- Interrupt or reset return from power-down
- Six oscillator configurations
  - Quartz crystal
  - Ceramic resonator
  - RC
  - LC
  - External
- Input supply range from 1.5V to 6V
- Operating frequency from DC to 20MHz

The 8XCL410 can be operated from a single battery supply which can vary between 1.5V and 6V, and for an AC supply the part requires only a simple voltage regulator. In some cases the part can be operated from an unstabilized supply, eliminating altogether the need for a regulator.

The power consumption of the part is much lower than that of a standard 80C51. Operating at 3.5MHz from a 3V supply, the 8XCL410 typically draws less than 1mA of current. The power consumption of the part is directly related to the supply voltage and clock frequency. As the supply voltage or clock frequency are increased, the power consumption also increases. At 12MHz and a supply voltage of 5V, the 8XCL410 will draw about 10mA of current, which is slightly less than the current that an 80C51 would require.

The advantage that the 8XCL410 has over the 80C51 is in its ability to operate at very low frequencies and supply voltages. This makes the part an ideal choice for applications where power is supplied from batteries, or where low supply voltages or clock frequency are necessary. The 8XCL410 features a fully static design. Using the on-chip oscillator, the clock frequency is limited to a minimum of 32kHz, but using an external oscillator the part can be operated down to DC. This means that the clock can be turned off, and when it is started again the microcontroller will continue with the action that it was performing when the clock was stopped. This is something that is impossible with dynamic devices because their internal nodes must be constantly refreshed. The static design of the 8XCL410 offers the user the ultimate in power-down modes, because the part can be stopped until it is needed and then started from where it was at when it was stopped, with no loss of internal states or data. The power consumption of the 8XCL410 when the clock is stopped is less than 1μA.

Differences from the 80C51

Special Function Registers

The 8XCL410 contains most of the special function registers found in the 80C51 as well as eight additional SFRs that have been added to handle the I^2C serial interface and eight additional external interrupts. The standard UART found on the 80C51 has been replaced with an I^2C serial interface, so the SFRs SCON and SBUF have been removed. Four SFRs have been added to handle the I^2C interface; they are: S1CON, S1DAT, S1STA, and S1ADR.

The interrupt structure on the 8XCL410 has been upgraded to include eight additional external interrupts. The IE and IP registers on the 80C51 have had their names changed on the 8XCL410 to IEN0 and IP0. In addition, two SFRs have been added to handle the additional external interrupts. The registers are IEN1 and IP1.

Two more SFRs are added to allow the user to set the polarity of the additional external interrupts and to hold the interrupt request flags for those added interrupts. The SFRs are the interrupt polarity register (IX1) and the interrupt request flag register (IRQ1).

Table 6 shows the special function registers, their locations, and their reset values for the 8XCL410.

I^2C Serial Interface

The serial port supports the two-wire I^2C bus. The I^2C bus consists of a data line (SDA) and a clock line (SCL). These lines are multiplexed functions of I/O port pins P1.7 and P1.6, respectively. The main features of the bus are:

- Bidirectional data transfer between masters and slaves
- True multimaster bus
- Arbitration between simultaneously transmitting masters without loss or corruption of the serial data on the bus
- Synchronized clock allows devices with different bit rates to communicate
- The serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer.

The CPU of the 8XCL410 interfaces to the I^2C logic via four special function registers. The registers are S1CON (I^2C control register), S1DAT (data register), S1STA (status register), and S1ADR (slave address register).

A detailed discussion of the I^2C bus is given in section 2 of this users' guide. The I^2C interface used on the 8XCL410 is functionally identical to the one on the 8XC552. A detailed discussion of this I^2C hardware is given in the 8XC552 section, so the discussion here will be limited to a review of each of the four I^2C special function registers. A block diagram of the I^2C serial interface is shown in Figure 13.

The functions of the I^2C interface are controlled by the S1CON register.

Table 6. 8XCL410 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR:	Data pointer (2 bytes):										
DPH	High byte	83H									00H
DPL	Low byte	82H									00H
			BF	BE	BD	BC	BB	BA	B9	B8	
IP0*#	Interrupt priority 0	B8H	–	–	PS1	–	PT1	PX1	PT0	PX0	xx0x0000B
			FF	FE	FD	FC	FB	FA	F9	F8	
IP1*#	Interrupt priority 1	F8H	PX9	PX8	PX7	PX6	PX5	PX4	PX3	PX2	00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IEN0*#	Interrupt enable 0	A8H	EA	–	ES1	–	ET1	EX1	ET0	EX0	0x0x0000B
			EF	EE	ED	EC	EB	EA	E9	E8	00H
IEN1*#	Interrupt enable 1	E8H	EX9	EX8	EX7	EX6	EX5	EX4	EX3	EX2	00H
			C7	C6	C5	C4	C3	C2	C1	C0	00H
IRQ1*#	Interrupt request flag	C0H	IQ9	IQ8	IQ7	IQ6	IQ5	IQ4	IQ3	IQ2	00H
IX1#	Interrupt polarity	E9H									00H
P0*	Port 0	80H	87	86	85	84	83	82	81	80	FFH
P1*	Port 1	90H	97	96	95	94	93	92	91	90	FFH
P2*	Port 2	A0H	A7	A6	A5	A4	A3	A2	A1	A0	FFH
P3*	Port 3	B0H	B7	B6	B5	B4	B3	B2	B1	B0	FFH
PCON	Power control	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxx0000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
S1ADR#	Slave address	DBH									00H
			DF	DE	DD	DC	DB	DA	D9	D8	
S1ON*#	Serial control	D8H	–	ENS1	STA	STO	SI	AA	CR1	CR0	x0000000B
S1DAT#	Serial data	DAH									00H
S1STA#	Serial status	D9H									11111000B
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer/counter control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
TMOD	Timer/counter mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H
TH0	Timer 0 high byte	8CH									00H
TH1	Timer 1 high byte	8DH									00H
TL0	Timer 0 low byte	8AH									00H
TL1	Timer 1 low byte	8BH									00H

*SFRs are bit addressable.
#SFRs are modified from or added to the 80C51 SFRs.

S1CON (D8H)

7	6	5	4	3	2	1	0
–	ENSI	STA	STO	SI	AA	CR1	CR0

CR0 and CR1—Clock Rate bits. These bits determine the serial clock (SCL) frequency when the 8XCL410 is in the master mode. The various SCL serial clock rates that can be achieved are shown in Table 7.

The SCL rate generated when both CR0 and CR1 are low is usually used when the I²C interface on other parts are software driven and slow. The maximum SCL rate is 100kHz and can be derived from either a 12MHz or 6MHz oscillator. A variable bit rate can be used if timer 1 is not required for another purpose while the 8XCL410 is in the master mode.

The CR0 and CR1 bits have no effect when the part is in the slave mode, because SCL is generated only by the bus master. In the slave mode, the part will automatically synchronize with the I²C bus clock frequency.

AA—Assert Acknowledge. When the AA bit is set (1), an acknowledge will be returned when the 8XCL410 recognizes its own slave address, recognizes the general call address if S1ADR.0 is set (1), or receives a data byte while it is either the bus master or the selected slave. If AA is not set (0), then no acknowledge will be returned for any condition. The I²C bus hardware is not disabled, but the part will not respond to its own slave address or the general call address (even is S1ADR is set), nor will it acknowledge received bytes.

SI—Serial Interrupt flag SI is set by hardware when one of 25 of the 26 possible I²C hardware states on the 8XCL410 is entered. The only state that does not cause SI to be set is F8H, which indicates that no relevant state information is available. An interrupt will only be requested while SI is set (1) and the serial interrupt is enabled in the IEN0 (interrupt enable) SFR. When SI is set, the low period of the I²C clock (SCL) is stretched (that is, held low) until SI is cleared. SI can only be cleared by software.

STO—STOp flag. When the 8XCL410 is in the master mode and STO is set (1), a stop condition will be forced on the I²C bus by the part. When the stop is detected on the bus by the 8XCL410's hardware, it will clear the STO flag.

STA—STArt flag. When the 8XCL410 is set to enter the master mode and STA is set (1), the part will check the status of the I²C bus and generate a START condition if the bus is free. If the bus is not free, the 8XCL410 will wait for a STOP condition on the I²C bus and then after a half period delay (of SCL) it will generate a START.

If both STA and STO are set, and the part is in the master mode, a STOP will be forced on the bus, and then following that with the appropriate delays, a START will be forced.

ENS1—Enable I²C Serial Port. When ESN1 is low, the part will not respond to its address or any activity on the I²C bus. The SDA and SCL outputs are in a high impedance state. P1.6 and P1.7 can be used as open drain port pins. When ENS1 is set (1), the I²C serial port is enabled. Port latches P1.6 and P1.7 must be set (1).

ENS1 should not be used to temporarily release the bus, because the I²C bus status S1STA is cleared and the part's bus status lost when ESN1 is reset (0). To temporarily idle the bus, the AA flag should be used.

The data to be transmitted to or received from the I²C bus is written into or read from the S1DAT register.

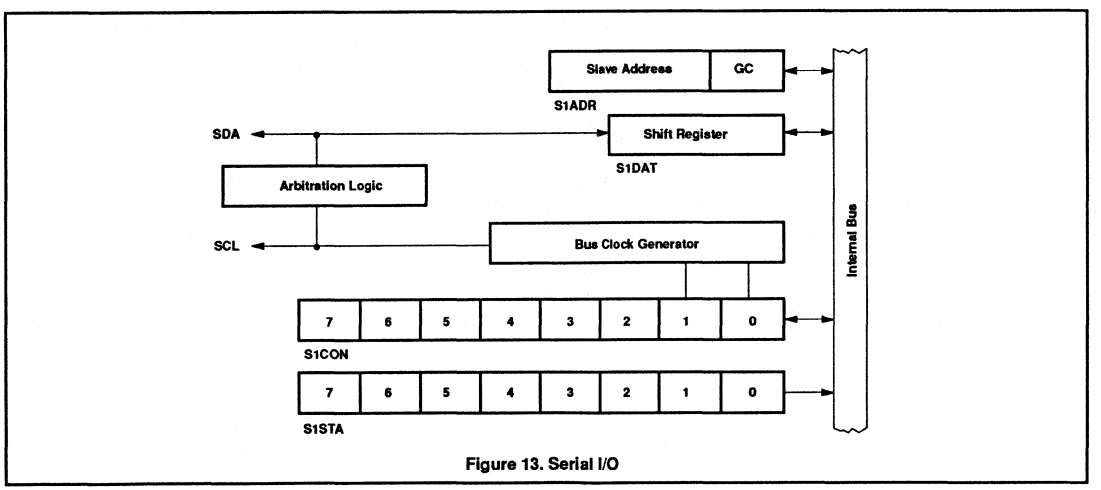


Figure 13. Serial I/O

Table 7. SCL Frequency (kHz) at f_{OSC}

CR1	CR0	6MHz	12MHz	16MHz	20MHz	f_{OSC} DIVIDED BY
0	0	6.25	12.5	17	21	960
0	1	50	100	NA	NA	120
1	0	100	NA	NA	NA	60
1	1	0.25 < 31.25	0.5 < 62.5	0.65 < 83.3	0.8 < 104	96 x (256 – Timer 1 reload range 0–254 in mode 1)

S1DAT (DAH)

7	6	5	4	3	2	1	0
SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0

SD7-SD0—Serial Data bits. A byte to be transmitted is written into this register, and a byte received is read from this register. A 1 in the register corresponds to a high level on the bus, and a 0 corresponds to a low level. Data shifts into or out of the register from left to right.

The status of the bus can be determined at any time by reading the S1STA register. This is a read only register in which the three least significant bits are always zero (0).

S1STA (D9H)

7	6	5	4	3	2	1	0
SC4	SC3	SC2	SC1	SC0	0	0	0

SC4-SC0—Status Code bits. These bits hold a status code that indicates the current status of the bus. The contents of these bits can be used to vector to a service routine, which optimizes the response time of the software and consequently that of the I^2C bus.

The following is a list of the status codes for each S1STA value.

Master Transmitter Mode

08H – A START condition has been transmitted.
10H – A repeated START condition has been transmitted.
18H – Slave address and write bit transmitted, acknowledge received.
20H – Slave address and write bit transmitted, acknowledge not received.
28H – Data transmitted, acknowledge received.
30H – Data transmitted, acknowledge not received.
38H – Arbitration lost while transmitting slave address, R/W bit, or data.

Master Receiver Mode

38H – Arbitration lost while returning acknowledge.
40H – Slave address and read bit transmitted, acknowledge returned.
48H – Slave address and read bit transmitted, acknowledge not returned.
50H – Data received, acknowledge returned.
58H – Data received, acknowledge not returned.

Slave Receiver Mode

60H – Own slave address and write bit received, acknowledge returned.
68H – Arbitration lost. Own slave address and write bit received, acknowledge returned.
70H – General call received, acknowledge returned.
78H – Arbitration lost. General call received.
80H – Received own slave address and data byte, acknowledge returned.
88H – Received own slave address and data byte, acknowledge not returned.
90H – Received general call and data byte, acknowledge returned.
98H – Received general call and data byte, acknowledge not returned.
A0H – Stop or repeated start received while still addressed as slave transmitter or receiver.

Slave Transmitter Mode

A8H – Own slave address and read bit received, acknowledge returned.
B0H – Arbitration lost. Own slave address and read bit received, acknowledge returned.
B8H – Data byte transmitted, acknowledge received.
C0H – Data byte transmitted, acknowledge not received.
C8H – Last data byte transmitted, acknowledge received.

All Modes

00H – Bus error due to an erroneous start or stop condition

The slave address that the part is to respond to is put into the S1ADR special function register.

S1ADR (DBH)

7	6	5	4	3	2	1	0
SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0

SA7-SA1—Slave Address bits. The 7-bit slave addresses that the part is to respond to is loaded into these seven bits.

SA0—This bit can be set so that the part will respond to a general call address on the I^2C bus. Clearing (0) this bit will prevent the part from responding to a general call address.

The Interrupt Structure

External events and the real-time-driven on-chip peripherals require service by the CPU asynchronous to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution, a multiple-source, two-priority level, nested interrupt system is provided. The 8XCL410 acknowledges interrupt requests from 13 sources as follows:

Priority	Source	Vector Address	Function
Highest	INT0	0003H	External interrupt 0
	S1	002BH	I2C serial interrupt
	INT5	0053H	External interrupt 5
	T0	000BH	Timer 0 interrupt
	INT6	005BH	External interrupt 6
	INT1	0013H	External interrupt 1
	INT2	003BH	External interrupt 2
	INT7	0063H	External interrupt 7
	T1	001BH	Timer 1 interrupt
	INT3	0043H	External interrupt 3
	INT8	006BH	External interrupt 8
	INT4	004BH	External interrupt 4
Lowest	INT9	0073H	External interrupt 9

There are six special function registers associated with the interrupt portion of the 8XCL410. They are IE0, IP0, IE1, IP1, IX1, and IRQ1. Following is a detailed description of each.

IE0—Interrupt Enable register zero. This register has the same function as the IE register found on the 80C51. The only difference between the two is that the bit that controls the serial interface interrupt has been moved from bit 4 to bit 5. This has been done because the 8XCL410 has an I^2C serial interface and bit 5 is used on other parts for the I^2C interrupt enable bit.

IE0 (A8H)

7	6	5	4	3	2	1	0
EA	–	ES1	–	ET1	EX1	ET0	EX0

EA – General Enable/Disable control
0 = All interrupts disabled.
1 = Interrupts can be individually enabled or disabled.
ES1 – I^2C interrupt enable. (1 = enabled, 0 = disabled)
ET1 – Timer 1 interrupt enable. (1 = enabled, 0 = disabled)

EX1 – External interrupt 1 enable. (1 = enabled, 0 = disabled)
ET0 – Timer 0 interrupt enable. (1 = enabled, 0 = disabled)
EX0 – External interrupt 0 enable. (1 = enabled, 0 = disabled)

IP0—Interrupt Priority register zero. This register has the same function as the IP register on the 80C51. The only difference is that the serial interface priority is set on bit 5.

IP0 (B8H)

7	6	5	4	3	2	1	0
–	–	PS1	–	PT1	–	PT0	PX0

PS1 – I^2C interrupt priority. (1 = high, 0 = low)
PT1 – Timer 1 interrupt priority. (1 = high, 0 = low)
PX1 – External interrupt 1 priority. (1 = high, 0 = low)
PT0 – Timer 0 interrupt priority. (1 = high, 0 = low)
PX0 – External interrupt 0 priority. (1 = high, 0 = low)

IE1—Interrupt Enable register one. This register contains the interrupt enables for the eight external interrupts that have been added to the 8XCL410. Clearing (0) the $\overline{EA}$ bit in the IE0 register disables all of the interrupts in this register as well as those in the IE0 register.

IE1 (E8H)

7	6	5	4	3	2	1	0
EX9	EX8	EX7	EX6	EX5	EX4	EX3	EX2

EX9-EX2—External interrupt enables for external interrupts 2 – 9. (0 = disabled, 1 = enabled)

IP1—Interrupt priority register one. This register allows the priority level of each of the additional external interrupt enables to be set.

IP1 (D8H)

7	6	5	4	3	2	1	0
PX9	PX8	PX7	PX6	PX5	PX4	PX3	PX2

PX9-PX2—External interrupt priority for external interrupts 2 – 9. (0 = low, 1 = high)

IX1—Interrupt Polarity register. This register allows the programmer to determine the polarity of external interrupts 2 – 9 that will be sensed for the interrupt. If the bit for an interrupt is set to 1, then the interrupt will be triggered by a high input on that external interrupt. If the bit is cleared to 0, then the interrupt will be triggered by a low on that external interrupt.

IX1 (E9H)

7	6	5	4	3	2	1	0
IL9	IL8	IL7	IL6	IL5	IL4	IL3	IL2

IL9-IL2—External Interrupt polarity bits corresponding to external interrupts 9 – 2. (1 = high trigger, 0 = low trigger)

IRQ1—Interrupt Request flag register. This register contains flags that are set when one of the external interrupts 2 – 9 are requested. The flags will only be set if the corresponding interrupt is enabled in the IE1 register. The flags must be cleared by software.

IRQ1 (C0H)

7	6	5	4	3	2	1	0
IQ9	IQ8	IQ7	IQ6	IQ5	IQ4	IQ3	IQ2

IQ9-IQ2—External interrupt request flags for external interrupts 9 – 2.

Oscillator

The on-chip oscillator circuitry of the 8XCL410 is a single stage inverting amplifier biased by an internal feedback resistor. The oscillator circuitry has been significantly enhanced over the circuitry on the 80C51 and for operation with a standard quartz crystal, no external components are needed. The oscillator circuitry on the 8XCL410 is capable of being driven from a number of sources, including: quartz crystal, ceramic resonator, RC networks, LC networks, and external sources. Figure 14 shows the various oscillator configurations that can be used on the part.

It is necessary when ordering the 8XCL410 to specify an oscillator option. This is necessary because of the wide range of potential external components that can be used with the oscillator. It is not possible to optimize one internal configuration of the operation of the oscillator for use with all of the external components. An option must therefore be specified that will allow the oscillator to be optimized in the specific target application that it is being purchased for. The available options are:

32kHz – When very low-speed operation is desired (<0.5MHz).
Osc.2 – For operation between 0.5MHz and 4MHz
Osc.3 – For operation between 4MHz and 10MHz
Osc.4 – For operation above 10MHz.
R-C – For operation with an RC network.

The 8XCL410 microcontroller features a fully static design. Using the on-chip oscillator, the clock frequency is limited to a minimum of 32kHz, but using external clock circuitry, this family of parts can be operated down to DC. This means that the clock can be turned off completely, and when it is started again, the microcontroller will continue with the action that it was performing when the clock was stopped. Notice that it will continue with the action that it was performing without changing anything inside of the part. It is just as if the clock was never stopped. This is something that is impossible with dynamic devices, because their internal nodes must be constantly refreshed. Besides the obvious benefit of being able to stop the clock, there is another benefit that the power consumption of the part can be reduced to very low levels by starting and stopping the clock and only using the part when it is needed. Starting and stopping the clock is only possible with an external clock source and is not possible when a quartz crystal, ceramic resonator, LC, or RC is used.

Power-Down Mode

In addition to being able to reduce the power consumption by stopping the clock, there are both idle and power-down modes available. These operate exactly the same as the idle and power-down modes on the 80C51. There is only one difference and that is that it is possible to terminate a power-down condition with either a reset or an external interrupt.

To be able to wake up the part from the power-down state with an external interrupt, both the PD and IDL bits of the PCON register must be set when entering the power-down mode. If only the PD bit is set, the power-down mode will only be terminated by a hardware reset. With both bits set, an interrupt on any of the additional external interrupts, INT2-INT9, will cause the part to wake up. To ensure that the oscillator is stable before the controller restarts, the internal clock will remain inactive for 1536 oscillator periods after the interrupt is detected. After this, the PD flag will be reset, and the part will be in the idle mode, and the interrupt will be handled in the normal way. Figure 15 shows the different oscillator delays associated with the two methods of waking the part up from the power-down mode.

Low Power Consumption

The 8XCL410 is targeted toward low power applications in industrial control, portable instrumentation, intelligent computer peripherals, portable consumer products, and smart cards. Working from a single supply which can vary between 1.5V and 6V, the 8XCL410 requires only a simple voltage regulator. In many cases it can be operated from an unstabilized supply, eliminating altogether the need

for a regulator. A typical 8XCL410 device draws 1mA from a 3V supply when running at a 3.5MHz clock frequency. Current consumption in the idle and power-down modes is reduced further to less than 0.5mA and 1μA, respectively.

The 8XCL410 can be operated at clock frequencies up to 20MHz. At these frequencies and a 5V supply voltage, the part will draw current similar to that of a standard 80C51. For the 8XCL410, the reduction in power is a function of both the clock frequency and the supply voltage. Power dissipation is reduced by lowering the clock frequency and/or reducing the supply voltage. A standard Philips 80C51 will operate down to a clock frequency of 0.5MHz and a supply voltage of 4V. The advantage of the low-power 8XCL410 is that it can be run at frequencies as low as 32kHz with the internal oscillator (DC when an external oscillator circuit is used), and that the voltage supply levels can be reduced down to 1.5V. To obtain maximum power reduction, the part can be operated with both reduced clock frequency and reduced voltage supply.

The low voltage operation of the 8XCL410 is due in part to the Philips SACMOS process. This is a self aligned contact CMOS process in which the isolation regions between the contacts and the edge of the isolation have been eliminated. This significantly reduces the size of the die, which in turn reduces the parasitic capacitances and drain resistance. This means that for a given clock speed, parts fabricated in the SACMOS process will require less power, and this is most apparent at low frequencies and voltage supply levels.

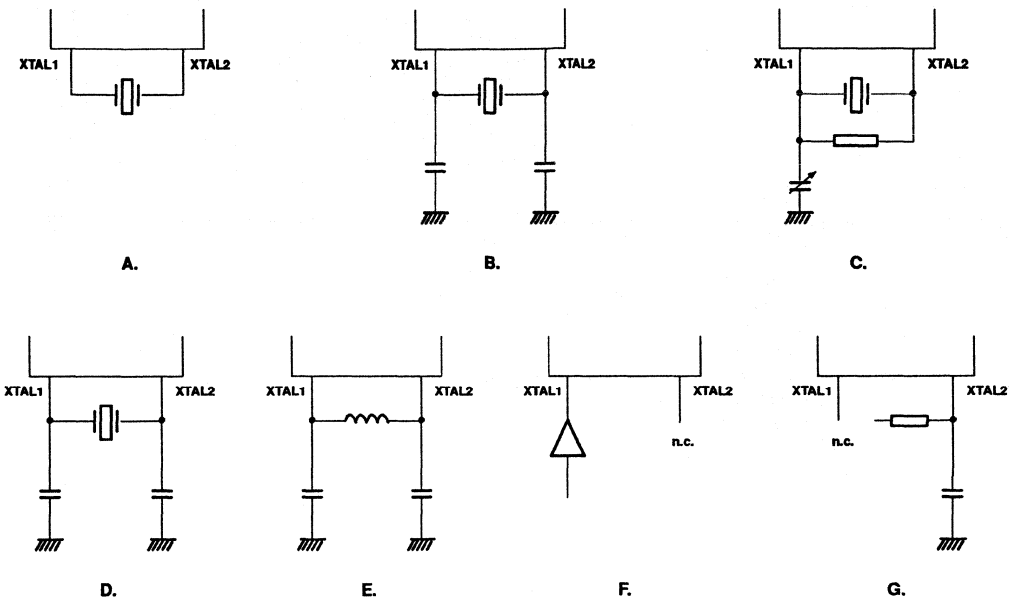


Figure 14. Alternative Oscillator Configurations

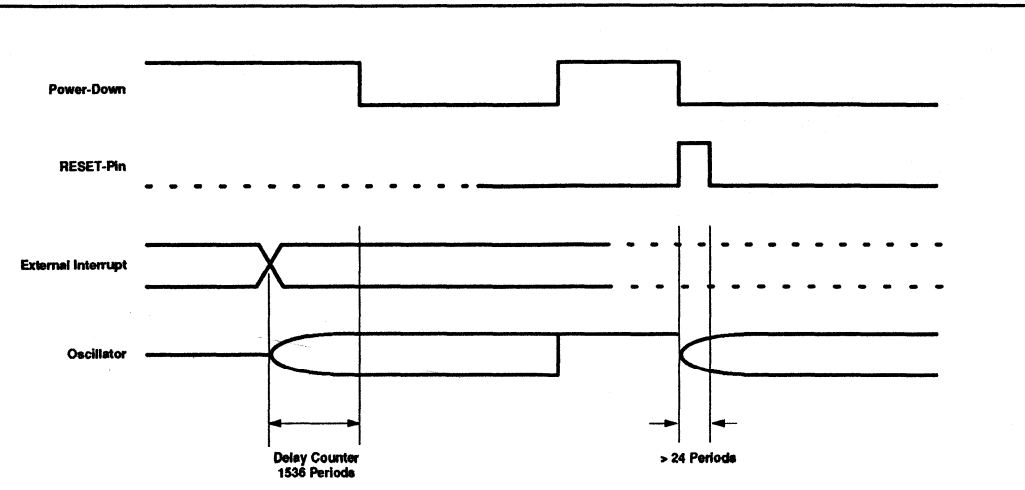


Figure 15. Wake-up Operation

Philips Components

Date of Issue	September 6, 1990
Status	Product Specification
Application Specific Products	

80CL410/83CL410

Low voltage/low power single-chip 8-bit microcontroller

DESCRIPTION

The 80CL410/83CL410 (hereafter generically referred to as 8XCL410) is manufactured in an advanced CMOS process that allows the part to operate at supply voltages down to 1.5V and oscillator frequencies down to DC. The 8XCL410 has the same instruction set as the 80C51.

The 8XCL410 features a 4k byte ROM (83CL410), 128 bytes RAM (both ROM and RAM are externally expandable to 64k bytes), four 8-bit ports, two 16-bit timer/counters, an I^2C serial interface, a thirteen source, two priority level nested interrupt structure, and on-chip oscillator circuitry suitable for quartz crystal, ceramic resonator, RC, or LC.

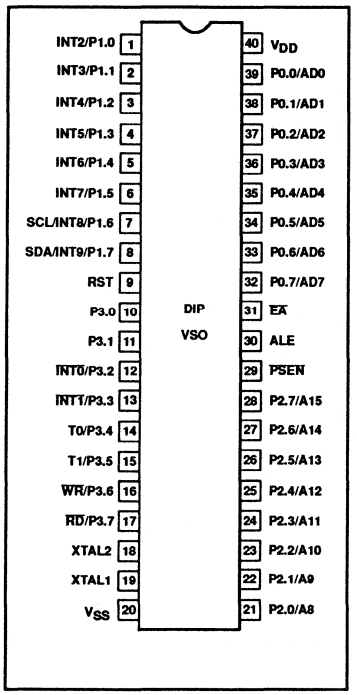
The 8XCL410 has two reduced power modes that are the same as those on the standard 80C51. The special reduced power feature of this part is that it can be stopped and then restarted. Running from an external clock source, the clock can be stopped and after a period of time restarted. The 8XCL410 will resume operation from where it was when the code stopped with no loss of internal state, RAM contents, or Special Function Register contents. If the internal oscillator is used the part cannot be stopped and started, but the power-down mode, which can be terminated via an interrupt, can be used to achieve similar power savings and then restart without loss of on-chip RAM and Special Function Register values.

For emulation purposes, the P85CL000 (Piggyback version) with 256 bytes of RAM is recommended.

FEATURES

- Supply voltage from 1.5 to 5.5V
- Operating frequency from 32kHz to 20MHz
- 80C51 based architecture
  - 4k x 8 ROM (64k external)
  - 128 x 8 RAM (64k external)
  - Four 8-bit I/O ports
  - Two 16-bit timer/counters
  - A thirteen source, two level, nested priority interrupt structure
  - 10 external interrupts
- Fully static 80C51 CPU
- I^2C Serial Interface
- Two power control modes
  - Idle mode
  - Power-down mode – can be terminated by reset or external interrupt
- Wake-up via external interrupts at port 1
- On-chip oscillator (quartz crystal, ceramic resonator, RC, LC)
- Very low power consumption
- Operating temperature range: –40 to +85°C

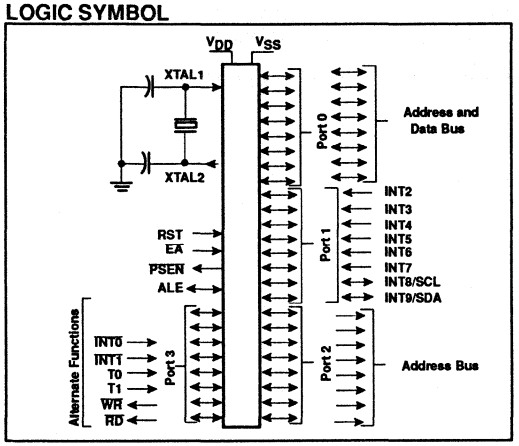
PIN CONFIGURATION



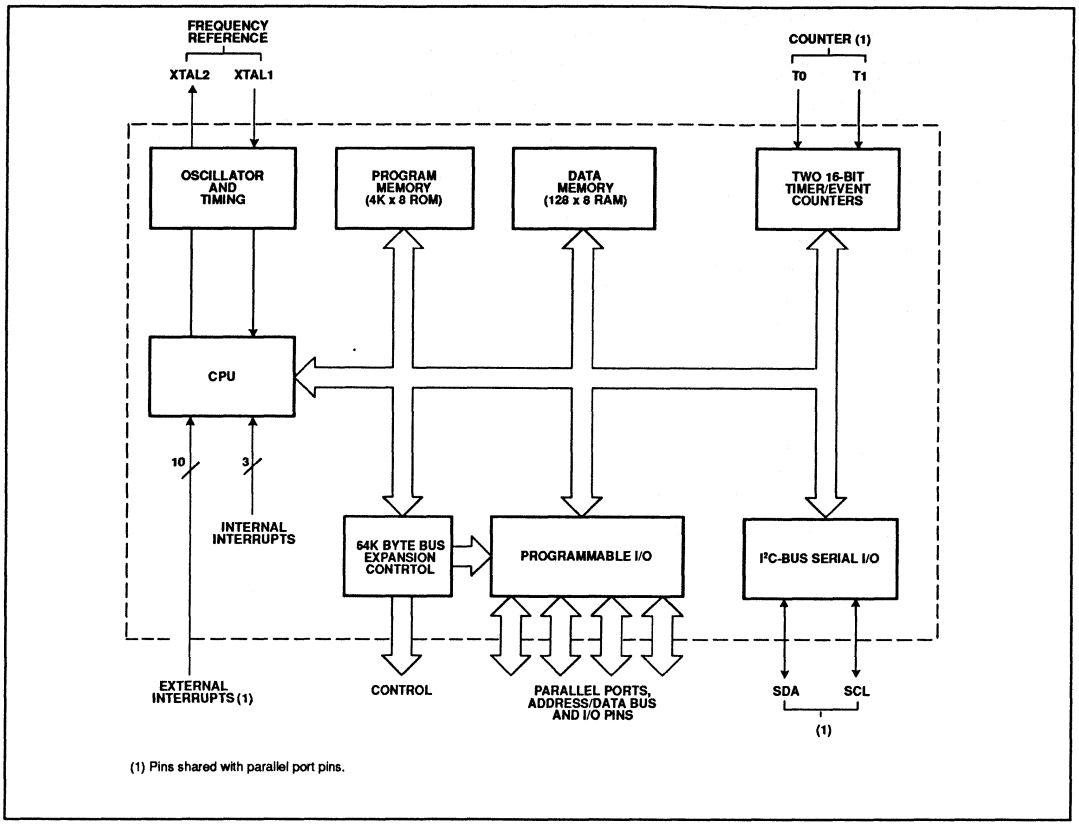
ORDERING CODE

T_A = –40 to +85°C; Frequency 32kHz to 20MHz	
ROM	PACKAGE
P83CL410HFP	40-pin plastic DIP
P83CL410HFT	40-pin plastic VSO

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
V_{SS}	20	I	**Ground:** 0V reference.
V_{DD}	40	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s.
P1.0–P1.7	1–8	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Additional functions include:
	7	I/O	**SCL (P1.6):** I^2C serial bus clock.
	8	I/O	**SDA (P1.7):** I^2C serial bus data.
	1–8	I	**INT2–INT9 (P1.0–P1.7):** Additional external interrupts.
P2.0–P2.7	21–28	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the 80C51 family, as listed below:
	12	I	**$\overline{INT0}$ (P3.2):** External interrupt 0
	13	I	**$\overline{INT1}$ (P3.3):** External interrupt 1
	14	I	**T0 (P3.4):** Timer 0 external input
	15	I	**T1 (P3.5):** Timer 1 external input
	16	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{DD}.
ALE	30	I/O	**Address Latch Enable:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory.
$\overline{PSEN}$	29	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$	31	I	**External Access Enable:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H to 1FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 0FFFH.
XTAL1	19	I	**Crystal 1:** Input to the inverting oscillator amplifier and input for an external clock source.
XTAL2	18	O	**Crystal 2:** Output from the inverting oscillator amplifier.

PORT OPTIONS

The pins of port 1 (not P1.6/SCL or P1.7/SDA), port 2, and port 3 may be individually configured with one of the following port options (see Figure 1):

Option 1: **Standard Port**—quasi-bidirectional I/O with pull-up. The strong booster pull-up p1 is turned on for two oscillator periods after a 0-to-1 transition in the port latch. See Figure 1(a).

Option 2: **Open Drain**—quasi-bidirectional I/O with n-channel open drain output. Use as an output requires the connection of an external pull-up resistor. See Figure 1(c).

Option 3: **Push-Pull**—output with drive capability in both polarities. Under this option, pins can only be used as outputs.

The definition of port options for port 0 is slightly different.

Two cases have to be examined. First, accesses to external memory (EA = 0 or access above the built-in memory boundary), and second, I/O accesses.

External Memory Accesses

Option 1: True 0 and 1 are written as address to the external memory (strong pull-up is used).

Option 2: An external pull-up resistor is needed for external accesses.

Option 3: True 0 and 1 are written as address to the external memory (strong pull-up is used).

I/O Accesses

Option 1: When writing a 1 to the port latch, the strong pull-up p1 will be on for two oscillator periods. No weak pull-up exists. Without an external pull-up, this option can be used as a high-impedance input. See Figure 1(b).

Option 2: Open drain—quasi-bidirectional I/O with n-channel open drain output. Use as an output requires the connection of an external pull-up resistor. See Figure 1(c).

Option 3: Push-Pull—output with drive capability in both polarities. Under this option, pins can only be used as outputs.

Individual mask selection of the post-reset state is available on any of the above pins. Make your selection by appending "S" or "R" to option 1, 2, or 3 above (e.g., 1S for a standard I/O to be set after RESET or 2R for an open-drain I/O to be reset after RESET.

Option S: **Set**—after reset, this pin will be initialized High..

Option R: **Reset**—after reset, this pin will be initialized Low.

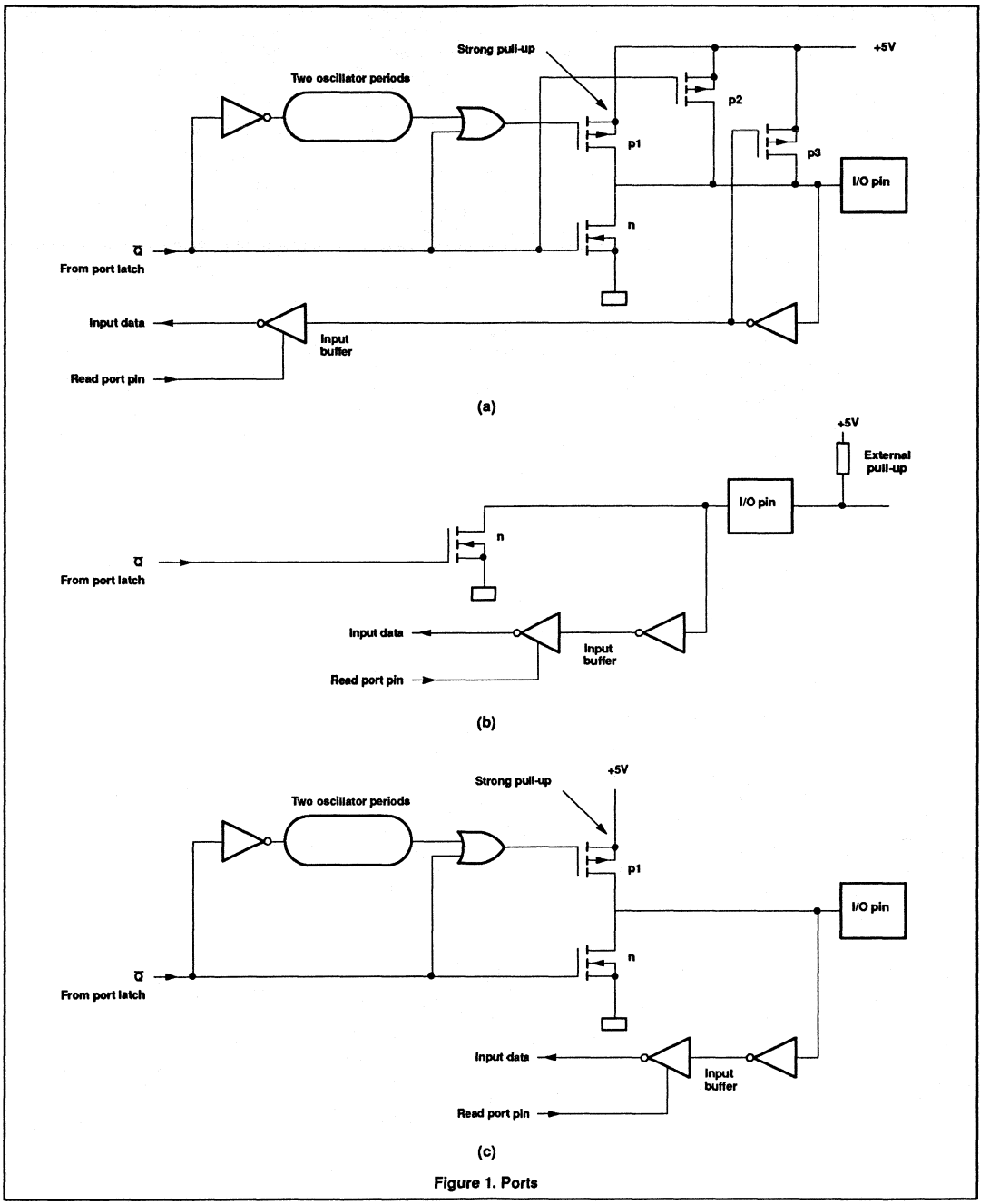


Figure 1. Ports

POWER-DOWN MODE

The instruction setting PCON.1 is the last executed prior to going into the power-down mode. In power-down mode, the oscillator is stopped. The contents of the the on-chip RAM and SFRs are preserved. The port pins output the values held by their respective SFRs. ALE and $\overline{PSEN}$ are held low. Power-down operates in wake-up mode and reset mode.

Wake-Up Mode

Setting both PD and IDL flags in the PCON register forces the controller into the power-down mode. Setting both flags enable the controller to be woken-up from the power-down mode with either the external interrupts INT2–INT9, or a reset operation.

An external interrupt INT2–INT9 at port 1 releases both the oscillator and the delay counter. To ensure that the oscillator is stable before the controller restarts, the internal clock will remain inactive for 1536 oscillator periods after the interrupt is detected. After this, the PD flag will be reset, the controller is now in the Idle mode and the interrupt will be handled in the normal way.

Reset Mode

Setting only the PD bit in the PCON register again forces the controller into the power-down mode, but in this case it can only be restored to normal operation with a direct reset operation.

IDLE MODE

The instruction that sets PCON.0 is the last instruction executed before going into idle mode. In idle mode, the internal clock is stopped for the CPU, but not for the interrupt, timer, and serial port functions. The CPU status is preserved along with the stack pointer, program counter, program status word and accumulator. The RAM and all other registers maintain their data during idle mode. The port pins retain the logical states they held at idle mode activation. ALE and $\overline{PSEN}$ hold at the logic high level.

There are two methods used to terminate the idle mode. Activation of any interrupt will cause PCON to be cleared by hardware; terminating idle mode. The interrupt is serviced, and following the instruction RETI, the next instruction to be executed will be the one following the instruction that put the device in the the idle mode.

Flag bits GF0 and GF1 can be used to determine whether the interrupt was received during normal execution or idle mode. For example, the instruction that writes to PCON.0 can also set or clear one or both flag bits. When idle mode is terminated by an interrupt, the service routine can examine the status of the flag bits.

The second method of terminating the idle mode is with an external hardware reset. Since the oscillator is still running, the hardware reset is required to be active for only two machine cycles to complete the reset operation. Reset redefines all SFRs, but does not affect the on-chip RAM.

In the power-down mode, V_{DD} may be reduced to minimize power consumption. However, the supply voltage must not be reduced until the power-down mode is active, and must be restored before the hardware reset is applied and frees the oscillator. Reset must be held active until the oscillator has restarted and stabilized.

The status of the external pins during idle and power-down mode is shown in Table 1. If the power-down mode is activated while accessing external memory, port data held in the special function register P2 is restored to port 2. If the data is a logic 1, the port pin is held high during the power-down mode.

INTERRUPT SYSTEM

External events and the real-time-driven on-chip peripherals require service by the CPU asynchronous to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution, a multiple-source, two-priority level, nested interrupt system is provided. The 8XCL410 acknowledges interrupt requests from thirteen sources, as follows:

- INT0 and INT1
- Timer 0 and timer 1
- I^2C-bus serial I/O interrupt
- INT2 to INT9 (port 1)

Each interrupt vectors to a separate location in program memory for its service routine. Each source can be individually enabled or disabled by corresponding bits in the internal enable registers (IEN0, IEN1) The priority level is selected via the interrupt priority register (IP0, IP1). All enabled sources can be globally disabled or enabled.

External Interrupts INT2–INT9

Port 1 lines serve an alternative purpose as eight additional interrupts INT2–INT9. When enabled, each of these lines can "wake-up" the device from power-down mode. Using the IX1 register, each pin may be initialized to either active high or low. IRQ1 is the interrupt request flag register. Each flag, if the interrupt is enabled, will be set on an interrupt request but it must be cleared by software.

IEN1 (E8H)
Interrupt enable register

7	6	5	4	3	2	1	0
EX9	EX8	EX7	EX6	EX5	EX4	EX3	EX2

Bit	Symbol	Function
IEN1.7	EX9	Enable external interrupt 9
IEN1.6	EX8	Enable external interrupt 8
IEN1.5	EX7	Enable external interrupt 7
IEN1.4	EX6	Enable external interrupt 6
IEN1.3	EX5	Enable external interrupt 5
IEN1.2	EX4	Enable external interrupt 4
IEN1.1	EX3	Enable external interrupt 3
IEN1.0	EX2	Enable external interrupt 2

where 0 = interrupt disabled
1 = interrupt enabled

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Floating	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Floating	Data	Data	Data

IP1 (F8H)
Interrupt priority register

7	6	5	4	3	2	1	0
PX9	PX8	PX7	PX6	PX5	PX4	PX3	PX2

Bit	Symbol	Function
IP1.7	PX9	External interrupt 9 priority level
IP1.6	PX8	External interrupt 8 priority level
IP1.5	PX7	External interrupt 7 priority level
IP1.4	PX6	External interrupt 6 priority level
IP1.3	PX5	External interrupt 5 priority level
IP1.2	PX4	External interrupt 4 priority level
IP1.1	PX3	External interrupt 3 priority level
IP1.0	PX2	External interrupt 2 priority level

Interrupt priority is as follows:
0 – low priority
1 – high priority

IX1 (E9H)
Interrupt polarity register

7	6	5	4	3	2	1	0
IL9	IL8	IL7	IL6	IL5	IL4	IL3	IL2

Bit	Symbol	Function
IX1.7	IL9	External interrupt 9 polarity level
IX1.6	IL8	External interrupt 8 polarity level
IX1.5	IL7	External interrupt 7 polarity level
IX1.4	IL6	External interrupt 6 polarity level
IX1.3	IL5	External interrupt 5 polarity level
IX1.2	IL4	External interrupt 4 polarity level
IX1.1	IL3	External interrupt 3 polarity level
IX1.0	IL2	External interrupt 2 polarity level

Writing either a "1" or "0" to an IX1 register bit sets the priority level of the corresponding external interrupt to active High or Low, respectively.

IRQ1 (C0H)
Interrupt request flag register

7	6	5	4	3	2	1	0
IQ9	IQ8	IQ7	IQ6	IQ5	IQ4	IQ3	IQ2

Bit	Symbol	Function
IRQ1.7	IQ9	External interrupt 9 request flag
IRQ1.6	IQ8	External interrupt 8 request flag
IRQ1.5	IQ7	External interrupt 7 request flag
IRQ1.4	IQ6	External interrupt 6 request flag
IRQ1.3	IQ5	External interrupt 5 request flag
IRQ1.2	IQ4	External interrupt 4 request flag
IRQ1.1	IQ3	External interrupt 3 request flag
IRQ1.0	IQ2	External interrupt 2 request flag

Priority	Vector	Source
X0 (highest)	0003H	External 0
S1	002BH	I^2C port
X5	0053H	External 5
T0	000BH	Timer 0
X6	005BH	External 6
X1	0013H	External 1
X2	003BH	External 2
X7	0063H	External 7
T1	001BH	Timer 1
X3	0043H	External 3
X8	006BH	External 8
X4	004BH	External 4
X9 (lowest)	0073H	External 9

Register	Function	SFR Address
IX1	Interrupt polarity register	E9H
IRQ1	Interrupt request flag register	C0H
IEN0	Interrupt enable register	A8H
IEN1	Interrupt enable register (INT2–INT9)	E8H
IP0	Interrupt priority register	B8H
IP1	Interrupt priority register (INT2–INT9)	F8H

OSCILLATOR CIRCUITRY

The on-chip oscillator circuitry of the 8XCL410 is a single stage inverting amplifier biased by an internal feedback resistor. (See Figure 2.) The oscillator can be operated with a quartz crystal, ceramic resonator, LC network or RC network. See Figure 3 for different configurations. When ordering parts, it is necessary to specify an oscillator option. The options are: RC when an RC network will be used, OSC 2 for oscillator operation below 4MHz, OSC 3 for oscillator operation from 4MHz to 10MHz, OSC 4 for oscillator operation above 10MHz, and 32kHz if 32kHz operation is desired.

For operation as a standard quartz oscillator, no external components are needed. When using external capacitors, ceramic resonators, coils, and RC networks to drive the oscillator, five different configurations are supported (see Figure 3 and Table 2).

In the power-down mode the oscillator is stopped and both XTAL1 and XTAL2 are pulled high. The feedback resistor is switched off to ensure no current will flow regardless of the voltage at XTAL1. To drive the device with an external clock source, apply the external clock signal to XTAL1, and leave XTAL2 to float, as shown in Figure 3(f). There are no requirements on the duty cycle of the external clock, since the input to the internal clocking circuitry is split using a flip-flop.

The following options are provided for optimum on-chip oscillator performance. Please state option when ordering:

32kHz: Figure 3(c). An option for 32kHz clock applications with external trimmer for frequency adjustment.

A 4.7MΩ bias resistor is recommended for use in parallel with the crystal.

Osc.2: Figure 3(e). An option for low-power, low-frequency operations using LC components.

Osc.3: An option for medium frequency range applications.

Osc.4: An option for high frequency range applications.

RC: Figure 3(g). An option for an RC oscillator.

The equivalent circuit data of the internal oscillator compares with that of matched crystals.

The externally adjustable RC oscillator has a frequency range from 100kHz to 500kHz. (See Figure 5.)

Power-on Reset

The 8XCL410 contains on-chip circuitry which switch the port pins to the customer-defined logic level as soon as V_{DD} exceeds 1.2V. (See Figures 6 and 7.) As soon as the minimum supply voltage is reached, the oscillator will start up. However, to ensure that the oscillator is stable before the controller starts, the clock signals are gated away from the CPU for a further 1536 oscillator periods.

An hysteresis of approximately 100mV at a typical power-on switching level of 1.3V will ensure correct operation.

An automatic reset can be obtained at power-on by connecting the RST pin to V_{DD} via a 10µF capacitor. At power-on, the voltage on the RST pin is equal to V_{DD} minus the capacitor voltage, and decreases from V_{DD} as the capacitor discharges through the internal resistor R_{RST} to ground. The larger the capacitor, the more slowly V_{RST} decreases. V_{RST} must remain above the lower threshold of the Schmitt trigger long enough to effect a complete reset. The time required is the oscillator start-up time, plus 2 machine cycles.

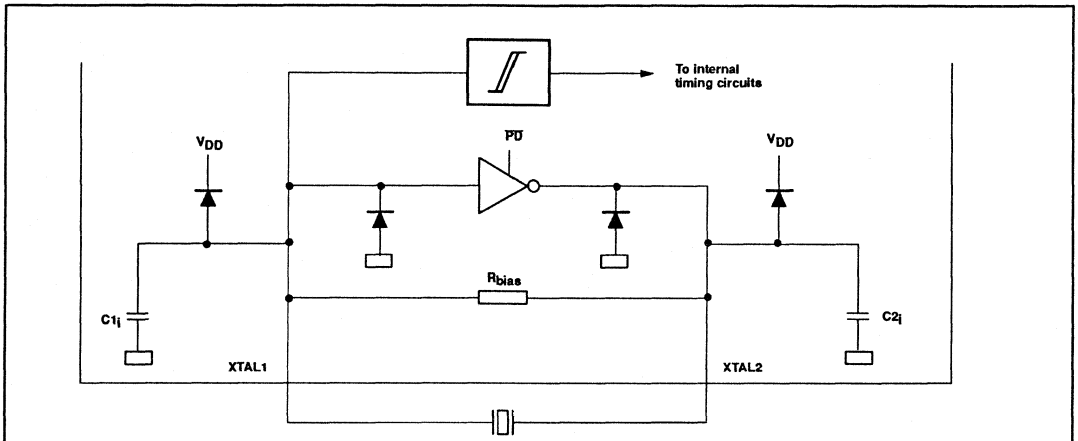


Figure 2. Oscillator

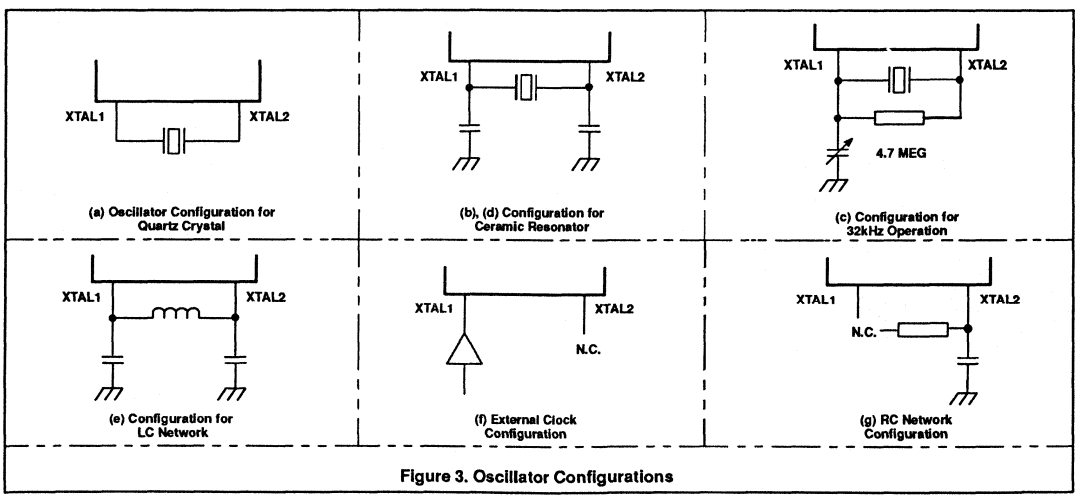


Figure 3. Oscillator Configurations

Table 2. Oscillator Type Selection Guide

RESONATOR	f (MHZ)	OPTION	C1 EXT. MIN	C1 EXT. MAX	C2 EXT. MIN	C2 EXT. MAX	MAXIMUM RESONATOR SERIES RESISTANCE
Quartz	0.032	32kHz	5	15	0	0	15kΩ[1]
Quartz	1.0	Osc.2	0	30	0	30	600Ω
Quartz	3.58	Osc.2	0	15	0	15	100Ω
Quartz	4.0	Osc.2	0	20	0	20	75Ω
Quartz	6.0	Osc.3	0	10	0	10	60Ω
Quartz	10.0	Osc.4	0	15	0	15	60Ω
Quartz	12.0	Osc.4	0	10	0	10	40Ω
Quartz	16.0	Osc.4	0	15	0	15	20Ω
PXE	0.455	Osc.2	40	50	40	50	10Ω
PXE	1.0	Osc.2	15	50	15	50	100Ω
PXE	3.58	Osc.2	0	40	0	40	10Ω
PXE	4.0	Osc.2	0	40	0	40	10Ω
PXE	6.0	Osc.2	0	20	0	20	5Ω
PXE	10.0	Osc.3	0	15	0	15	6Ω
PXE	12.0	Osc.4	10	40	10	40	6Ω
LC		Osc.2	20	90	20	90	10μH = 1Ω 100μH = 5Ω 1mH = 75Ω

NOTE:

1. 32kHz quartz crystals with a series resistance higher than 15kΩ will reduce the guaranteed supply voltage range to 2.5 to 3.5V.

Table 3. Oscillator Equivalent Circuit Parameters (see Figure 4)

PARAMETER	OPTION	SYMBOL	CONDITION	MIN	TYP	MAX	UNIT
Transconductance	All	g_m	T = +85°C	–	–	–	S
	32kHz	g_m	T = +25°C; V_{DD} = 4.5V	–	15	–	S
	Osc.2	g_m	T = +25°C; V_{DD} = 4.5V	–	600	–	S
	Osc.3	g_m	T = +25°C; V_{DD} = 4.5V	–	1500	–	S
	Osc.4	g_m	T = +25°C; V_{DD} = 4.5V	–	4000	–	S
Input capacitance	32kHz	$c1_i$		–	3.0	–	pF
	Osc.2	$c1_i$		–	8.0	–	pF
	Osc.3	$c1_i$		–	8.0	–	pF
	Osc.4	$c1_i$		–	8.0	–	pF
Output capacitance	32kHz	$c2_i$		–	23.0	–	pF
	Osc.2	$c2_i$		–	8.0	–	pF
	Osc.3	$c2_i$		–	8.0	–	pF
	Osc.4	$c2_i$		–	8.0	–	pF
Output resistance	32kHz	R2		–	3800	–	kΩ
	Osc.2	R2		–	65	–	kΩ
	Osc.3	R2		–	18	–	kΩ
	Osc.4	R2		–	5.0	–	kΩ

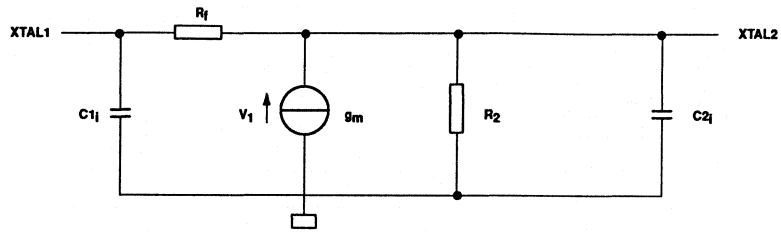


Figure 4. Equivalent Circuit Diagram

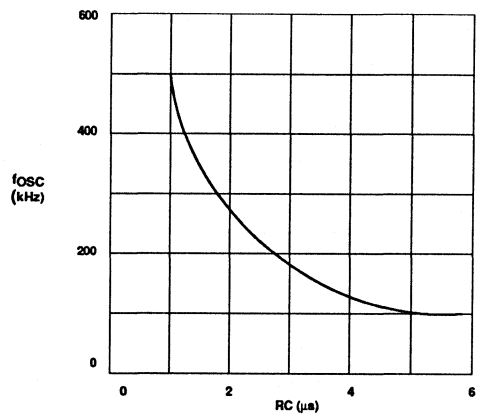


Figure 5. Frequency as a Function of RC

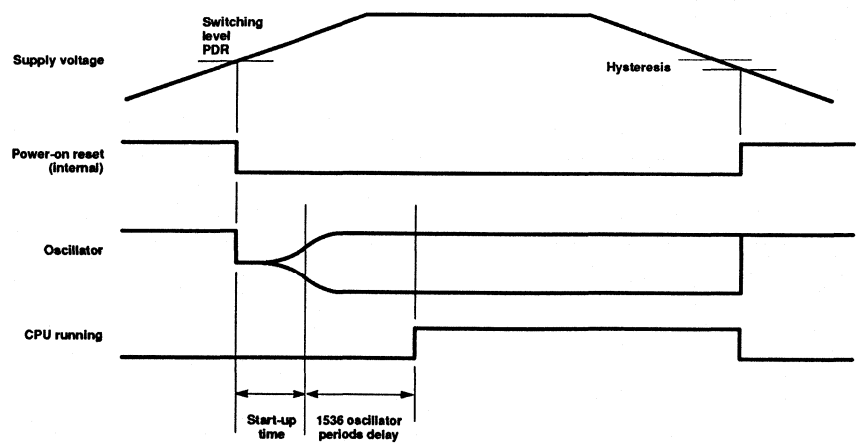


Figure 6. Power-on Reset Switching Level

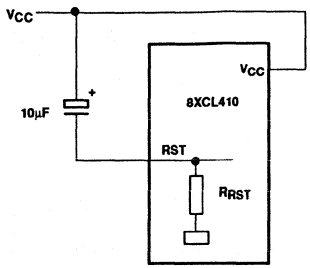


Figure 7. Recommended Power-on Reset Circuitry

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Supply voltage	–0.5 to +6.5	V
All input voltages	–0.5 to V_{DD} +0.5	V
DC current into any input or output	5	mA
Total power dissipation	300	mW
Storage temperature range	–65 to +150	°C
Operating ambient temperature range	–40 to +85	°C
Operating junction temperature	125	°C

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS*

T_A = –40°C to +85°C, V_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			MIN	TYPICAL[1]	MAX	
V_{DD}	Supply voltage RAM retention voltage in power-dov.n mode		2.5 1.0		5.5 5.5	V V
I_{DD}	Power supply current: Operating[2,3] Idle mode[3,4] Power-down mode[5]	V_{DD} = 5V, f_{CLK} = 16MHz V_{DD} = 3V, f_{CLK} = 3.58MHz V_{DD} = 5V, f_{CLK} = 16MHz V_{DD} = 3V, f_{CLK} = 3.58MHz V_{DD} = 2V, T = 25°C		 0.5 	 22 2.5 9 10	 mA mA mA mA µA
V_{IL}	Input low voltage		V_{SS}		0.3V_{DD}	V
V_{IH}	Input high voltage		0.7V_{DD}		V_{DD}	V
I_{OL}	Output sink current, except SDA, SCL[6,7]	V_{DD} = 5V, V_{OL} = 0.4V V_{DD} = 2.5V, V_{OL} = 0.4V	1.6 0.7			mA
I_{OL1}	Output sink current, SDA, SCL	V_{DD} = 5V, V_{OL} = 0.4V	3.0			mA
I_{OH}	Output source current[8] (push-pull options only)	V_{DD} = 5V, V_{OH} = V_{DD} – 0.4V V_{DD} = 2.5V, V_{OH} = V_{DD} – 0.4V	1.6 0.7			mA
I_{IL}	Logical 0 input current	V_{DD} = 5V, V_{IN} = 0.4V V_{DD} = 2.5V, V_{IN} = 0.4V			–100 –50	µA µA
I_{TL}	Logical 1-to-0 transition current,[9] ports 1, 2, 3	V_{DD} = 5V, V_{IN} = V_{DD}/2 V_{DD} = 2.5V, V_{IN} = V_{DD}/2			–1.0 –500	mA µA
I_{LI}	Input leakage current, port 0	V_{SS} < V_I < V_{DD}			±10	µA
R_{RST}	Internal reset pull-down resistor		50		200	kohm

NOTES:

* Contact local sales office regarding availability of 1.5V version.

1. Typical ratings are based on a limited number of samples taken from early manufacturing lots and are not guaranteed. The values listed are at room temperature, 5V.
2. The operating supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f$ = 10ns; $V_{IL} = V_{SS}$, $V_{IH} = V_{DD}$; XTAL2 not connected; EA = RST = Port 0 = V_{DD}; all open drain outputs connected to V_{SS}.
3. These values were based on measurements made with the high-frequency oscillator option (worst case). As this option would not typically be used for a 3.58MHz application, a typical value is quoted for this low frequency.
4. The idle supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f$ = 10ns; $V_{IL} = V_{SS}$, $V_{IH} = V_{DD}$; XTAL2 not connected; EA = Port 0 = V_{DD}; RST = V_{SS}; all open drain outputs connected to V_{SS}.
5. The power-down current is measured with all output pins disconnected; XTAL1 not connected; $\overline{EA}$ = port 0 = V_{DD}; RST = V_{SS}.
6. PSEN and ALE are inputs during Reset, internally pulled high. To start proper operation, the maximum sink current is 20µA at 5V, respectively, 5µA at 2.5V.
7. For port pin 2.7, only the open drain option is allowed. This is a temporary limitation and will be removed with the next version of the device.
8. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the 0.9V_{DD} specification when the address bits are stabilizing.
9. Pins of ports 1, 2 and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V. (See Figure 12.)

AC ELECTRICAL CHARACTERISTICS

T_A = –40°C to +85°C, V_{DD} = 5V ±10%, V_{SS} = 0V[1, 2]

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
Program Memory							
$1/t_{CLCL}$		Oscillator frequency			0	20	MHz
t_{LL}	8	ALE pulse width	127		$2t_{CLCL}$–40		ns
t_{AL}	8	Address valid to ALE low	43		t_{CLCL}–40		ns
t_{LA}	8	Address hold after ALE low	48		t_{CLCL}–35		ns
t_{LIV}	8	ALE low to valid instruction in		233		$4t_{CLCL}$–100	ns
t_{LC}	8	ALE low to $\overline{PSEN}$ low	58		t_{CLCL}–25		ns
t_{CC}	8	$\overline{PSEN}$ pulse width	215		$3t_{CLCL}$–35		ns
t_{CIV}	8	$\overline{PSEN}$ low to valid instruction in		125		$3t_{CLCL}$–125	ns
t_{CI}	8	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{CIF}	8	Input instruction float after $\overline{PSEN}$		63		t_{CLCL}–20	ns
t_{AVI}	8	Address to valid instruction in		302		$5t_{CLCL}$–115	ns
t_{AFC}	8	$\overline{PSEN}$ low to address float	0		0		ns
Data Memory							
t_{RR}	9	$\overline{RD}$ pulse width	400		$6t_{CLCL}$–100		ns
t_{WW}	10	$\overline{WR}$ pulse width	400		$6t_{CLCL}$–100		ns
t_{LA}	9, 10	Address hold time after ALE	48	–	t_{CLCL}–35	–	ns
t_{RD}	9	$\overline{RD}$ low to valid data in		250		$5t_{CLCL}$–165	ns
t_{DFR}	9	Data float after $\overline{RD}$		97		$2t_{CLCL}$–70	ns
t_{LD}	9	ALE low to valid data in		517		$8t_{CLCL}$–150	ns
t_{AD}	9	Address to valid data in		585		$9t_{CLCL}$–165	ns
t_{LW}	9, 10	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}$–50	$3t_{CLCL}$+50	ns
t_{AW}	9, 10	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}$–130		ns
t_{DWX}	10	Data valid to $\overline{WR}$ transition	23		t_{CLCL}–60		ns
t_{DW}	9	Data valid to $\overline{WR}$	433	–	$7t_{CLCL}$–150	–	ns
t_{WD}	10	Data hold after $\overline{WR}$	33		t_{CLCL}–50		ns
t_{AFR}	9	$\overline{RD}$ low to address float[3]		12		12	ns
t_{WHLH}	9, 10	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	t_{CLCL}–40	t_{CLCL}+40	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 50pF, load capacitance for all other outputs = 40pF.
3. Interfacing the 8XCL410 to devices with float time up to 75ns is permitted. This limited bus connection will not cause damage to port 0 drivers.

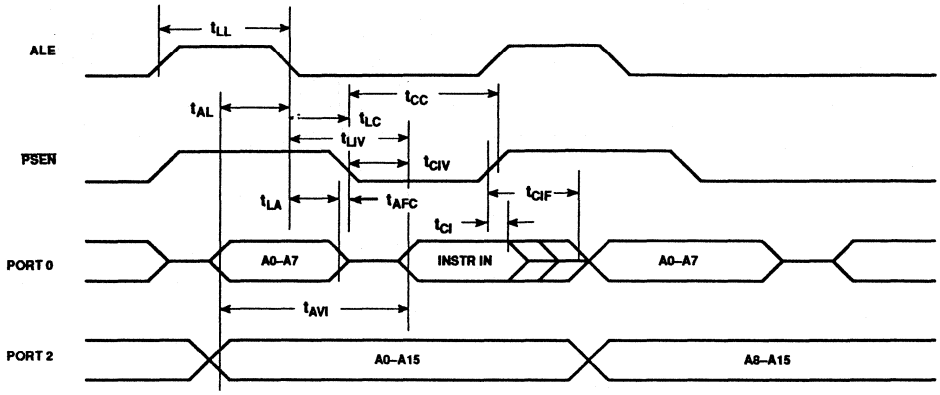


Figure 8. External Program Memory Read Cycle

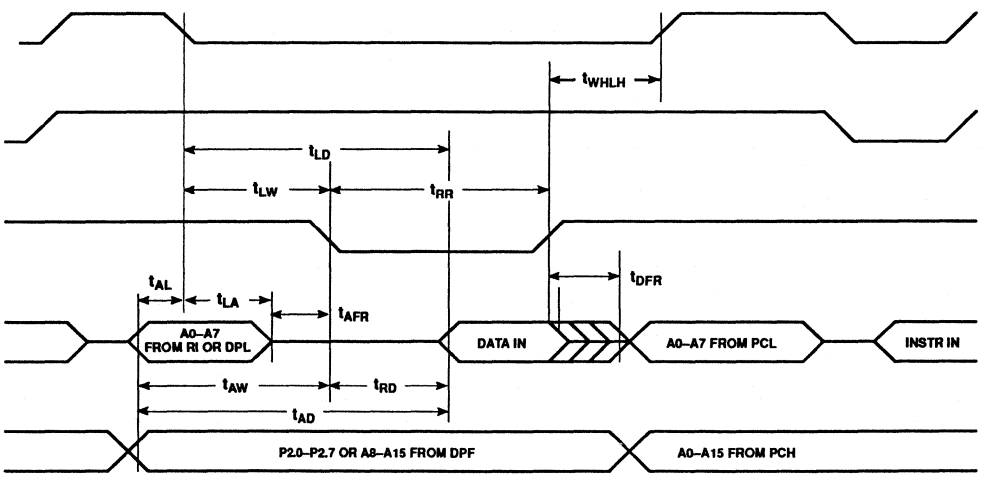


Figure 9. External Data Memory Read Cycle

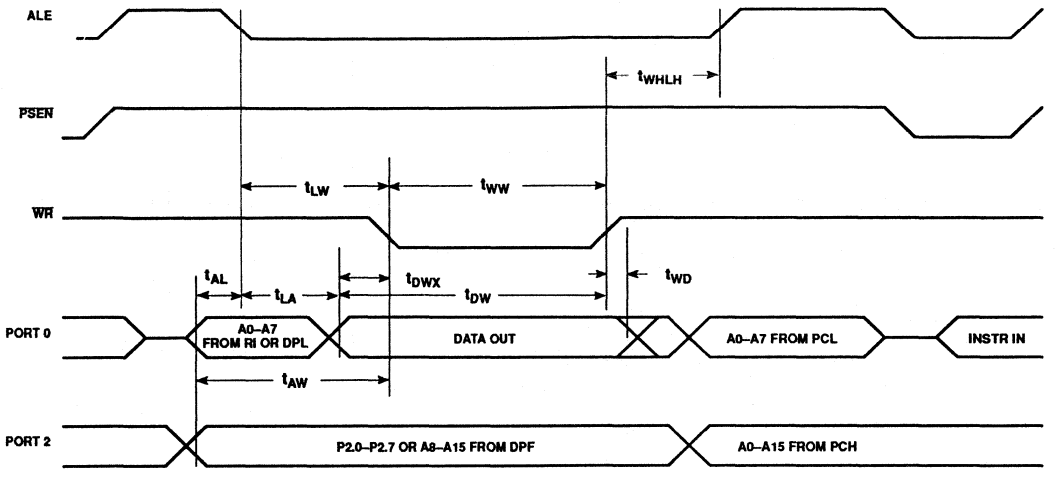


Figure 10. External Data Memory Write Cycle

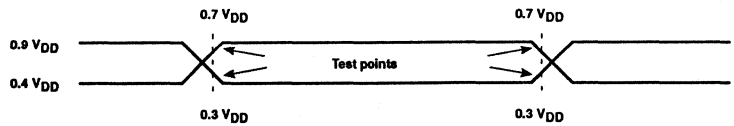


Figure 11. AC Testing Input Waveform

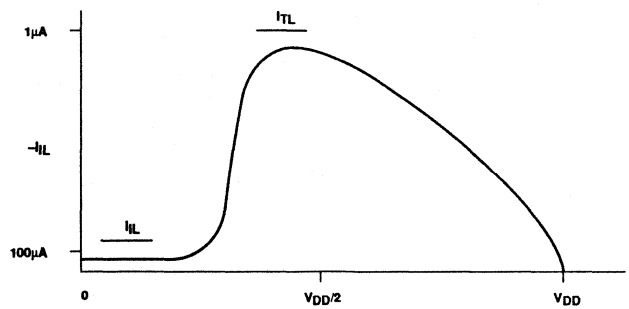


Figure 12. Input Current

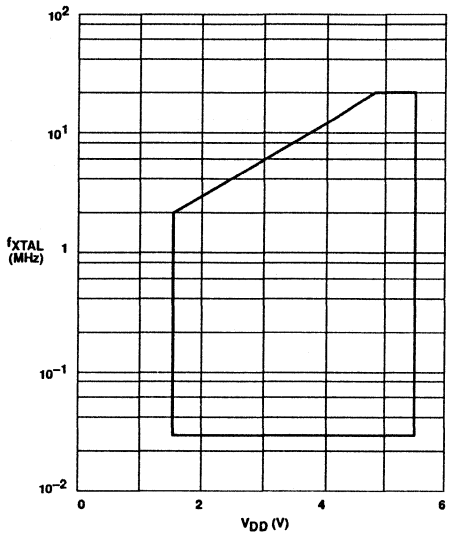


Figure 13. Frequency Operating Range

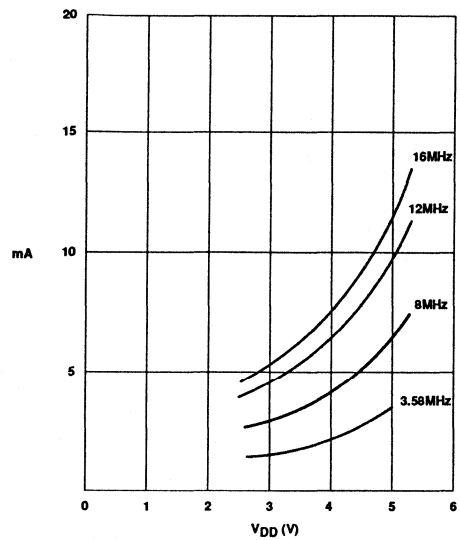


Figure 14. Typical Operating Current Vs. Frequency and V_{DD}, T_{amb} = 25°C

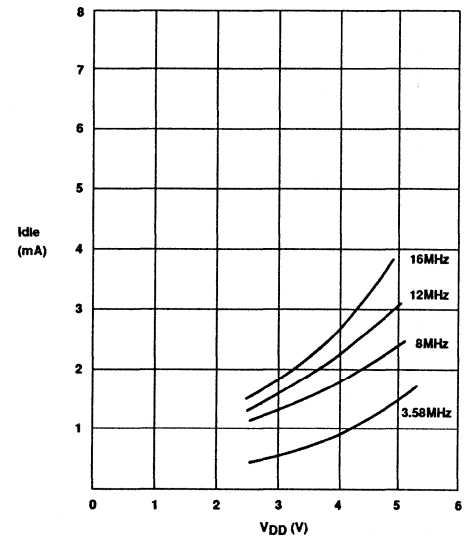


Figure 15. Typical Idle Current Vs. Frequency and V_{DD}, T_{amb} = 25°C

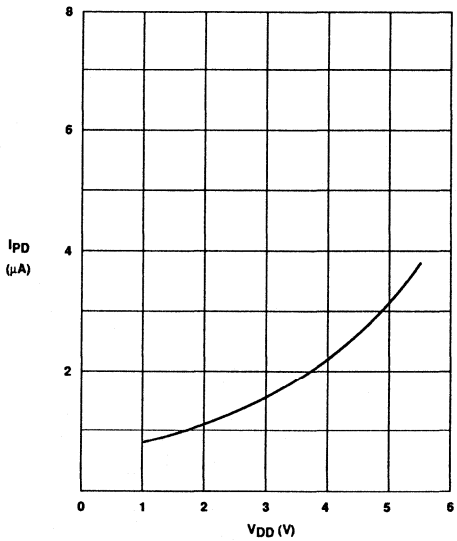


Figure 16. Typical Power-Down Current Vs. Frequency and V_{DD}, T_{amb} = 25°C

8XC451 OVERVIEW

The 80C451, the 83C451, and the 87C451 (hereafter referred to collectively as the 8XC451) are I/O expanded versions of the 80C51. Three I/O ports have been added to the basic 80C51 architecture for a total of 7 on-chip I/O ports. The LCC version has a total of 68 pins. The DIP version has 64 pins. Port 6 has 4 control lines to facilitate high-speed asynchronous I/O functions.

The 83C451/87C451 includes a 4k x 8 ROM/EPROM, a 128 x 8 RAM, 56 (LCC) or 52 (DIP) I/O lines, two 16-bit timer/counters, a five source, two priority, level nested interrupt structure, a serial I/O port for either full duplex UART, I/O expansion, or multiprocessor communications, and an on-chip oscillator and clock circuits. The 80C451 includes all of the 83C451 features except the on-board 4k x 8 ROM.

The 8XC451 has two software selectable modes of reduced activity for further power reduction: idle mode and power-down mode. Idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. Power-down mode freezes the oscillator, causing all other chip functions to be inoperative while maintaining the RAM contents.

The 8XC451 features include:

- 80C51 based architecture
- 68-pin LCC and 64-pin DIP packages
- Seven 8-bit I/O ports (LCC version)
- Six 8-bit ports and one 4-bit port (DIP version)
- 4k x 8 ROM or EPROM
- 128 x 8 RAM
- Two 16-bit counter/timers
- Two external interrupts
- External memory addressing capability
  - 64k ROM and 64k RAM
- Low power consumption
  - Idle mode
  - Power-down mode

Differences From the 80C51

Special Function Registers

The SFRs are identical to those of the standard 80C51 with the exception of four registers that have been added to allow control of the three additional I/O ports P4, P5, and P6. The additional registers are P4, P5, P6, and CSR. Registers P4, P5, and P6 function as port latches for ports 4, 5, and 6, respectively. These registers operate identically to those for ports 0 through 3 of the 80C51.

The Control Status Register (CSR) is used to control the mode of operation of port 6 and indicates the current status of port 6. All control status register bits can be read and written by the CPU except bits 0 and 1, which are read only. A Reset writes ones to bits 2-7 and zeros to bits 0 and 1. See Table 8 for the specific function of each bit in the Control Status register.

I/O Port Structure

The 8XC451 has a total of seven parallel I/O ports. The first four ports, P0 through P3, are identical in function to those present on the 80C51 family. The added ports 4 and 5 are identical in function to port 1; that is, they are standard quasi-bidirectional ports with no alternate functions and the standard output drive characteristics. Note that on the 68-pin LCC packages, port 4 is an 8-bit port, while on the 64-pin DIP packages, only the lower four bits of port 4 are available. Port 6 is a specialized 8-bit bidirectional I/O port with internal pullups. This special port can sink/source three LS TTL inputs and drive CMOS inputs without external pullups. The flexibility of this port facilitates high-speed parallel data communications. Port 6 operating modes are controlled by the port 6 Control Status Register (CSR). Port 6 and the CSR are addressed at the Special Function Addresses shown in Table 9. Port 6 can be used as a standard I/O port, or in strobed modes of operation in conjunction with the four port 6 control lines listed below:

Table 8. Control Status Register (CSR)

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
BFLAG Mode Select		AFLAG Mode Select		Output Buffer Flag Clear Mode	Input Data Strobe Mode	Output Buffer Full Flag	Input Buffer Full Flag
0/0 = Logic 0 output* 0/1 = Logic 1 output* 1/0 = IBF output 1/1 = PE input (0 = Select) (1 = Disable I/O)		0/0 = Logic 0 output 0/1 = Logic 1 output 1/0 = OBF output* 1/1 = SEL input (0 = Data) (1 = Control/status)		0 = Negative edge of $\overline{ODS}$ 1 = Positive edge of $\overline{ODS}$	0 = Positive edge of $\overline{IDS}$ 1 = Low level of $\overline{IDS}$	0 = Output data buffer empty 1 = Output data buffer full	0 = Input data buffer empty 1 = Input data buffer full

NOTE:
*Output-always mode: MB1 = 0, MA1 = 1, AND MA0 = 0. In this mode, port 6 is always enabled for output. $\overline{ODS}$ only clears the OBF flag.

Table 9 Special Function Register Addresses

REGISTER ADDRESS			BIT ADDRESS							
NAME	SYMBOL	ADDRESS	MSB							LSB
Port 4	P4	C0	C7	C6	C5	C4	C3	C2	C1	C0
Port 5	P5	C8	CF	CE	CD	CC	CB	CA	C9	C8
Port 6 data	P6	D8	DF	DE	DD	DC	DB	DA	D9	D8
Port 6 control status	CSR	E8	EF	EE	ED	EC	EB	EA	E9	E8

ODS	Output data strobe (active low)
IDS	Input data strobe (active low)
BFLAG	Bidirectional I/O pin. Can be programmed to output the Input Buffer Full flag (IBF), input an active low Port Enable (PE) signal, or output a high or low logic level.
AFLAG	Bidirectional I/O pin. Can be programmed to output the Output Buffer Full (OBF) flag, input a register select signal (SEL), or output a high or low logic level.

Port 6 can be used in a number of different ways to facilitate data communication. It can be used as a processor bus interface, as a standard quasi-bidirectional I/O port, or as a parallel printer port (either polled or interrupt driven).

Processor Bus Interface

Port 6 allows the use of an 8XC451 as an element on a microprocessor type bus. The host processor could be a general purpose MPU or the data bus of a microcontroller like the 8XC451 itself. Setting up the 8XC451 as a processor bus interface allows single or multiple microcontrollers to be used on a bus as flexible peripheral processing elements. Applications can include: keyboard scanners, serial I/O controllers, servo controllers, etc.

On reset, port 6 is programmed correctly (that is, Special Function registers CSR and P6) for use as a bus interface. This prevents the interface from disrupting data on the bus of a host processor during power-up.

Standard Quasi-bidirectional I/O Port

To use port 6 as a common I/O port, all of the control pins should be tied to ground. On hardware reset, bits 2-7 of the CSR are set to one. With the control pins grounded, the port's operation and electrical characteristics will be identical to port 1 on the 80C51. No further software initialization is required.

Parallel Printer Port

The 8XC451 has the capacity to permit all of the intelligent features of a common printer to be handled by a single chip. The features of port 6 allow a parallel port to be designed with only line driving and receiving chips required as additional hardware. The onboard UART allows RS232 interfacing with only level shifting chips added. The 8-bit parallel ports 0 to 6 are ample to drive onboard control functions, even when ports are used for external memory access, interrupts, and other functions. The RAM addressing ability of ports 0 to 2 can be used to address up to 64k bytes of a hardware buffer/spooler.

In addition, either end of a parallel interface can be implemented using port 6, and the interfaces can be interrupt driven or polled in either case. For more detailed information on port 6 usage, refer to the application notes contained in Section 4, entitled "80C451 Operation of Port 6" and "256k Centronics Printer Buffer Using the SC87C451 Microcontroller."

Table 10. 8X451 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT NAMES AND ADDRESSES MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
			EF	EE	ED	EC	EB	EA	E9	E8	
CSR*#	Port 6 command/status	E8H	MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF	FCH
DPTR	Data pointer (2 bytes)										
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*	Interrupt priority	B8H	–	–	–	PS	PT1	PX1	PT0	PX0	xxx00000B
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*	Interrupt enable	A8H	EA	–	–	ES	ET1	EX1	ET0	EX0	0xx00000B
P0*	Port 0	80H	87	B6	85	84	83	82	81	80	FFH
P1*	Port 1	90H	97	96	95	94	93	92	91	90	FFH
P2*	Port 2	A0H	A7	A6	A5	A4	A3	A2	A1	A0	FFH
P3*	Port 3	B0H	B7	B6	B5	B4	B3	B2	B1	B0	FFH
P4*#	Port 4	C0H	C7	C6	C5	C4	C3	C2	C1	C0	FFH
P5*#	Port 5	C8H	CF	CE	CD	CC	CB	CA	C9	C8	FFH
P6*#	Port 6	D8H	DF	DE	DD	DC	DB	DA	D9	D8	FFH
PCON	Power control	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxx0000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
SBUF	Serial data buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
SCON*	Serial port control	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer/counter control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
TMOD	Timer/counter mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H
TH0	Timer 0 high byte	8CH									00H
TH1	Timer 1 high byte	8DH									00H
TL0	Timer 0 low byte	8AH									00H
TL1	Timer 1 low byte	8BH									00H

*SFRs are bit addressable.
#SFRs are modified from or added to the 80C51 SFRs.

Philips Components

Date of Issue	February 1, 1990
Status	Product Specification
Application Specific Product	

80C451/83C451/87C451

CMOS single-chip 8-bit microcontroller

DESCRIPTION

The Philips 8XC451 is an I/O expanded single-chip microcontroller fabricated with Philips high-density CMOS technology. Philips epitaxial substrate minimizes latch-up sensitivity.

The 8XC451 is a functional extension of the 87C51 microcontroller with three additional I/O ports and four I/O control lines. The LCC version has a total of 68 pins. Four control lines associated with port 6 facilitate high-speed asynchronous I/O functions.

The 8XC451 includes a 4k X 8 ROM (83C451) EPROM (87C451), a 128 X 8 RAM, 56 (LCC) or 52 (DIP) I/O lines, two 16-bit timer/counters, a five source, two priority level, nested interrupt structure, a serial I/O port for either a full duplex UART, I/O expansion, or muti-processor communications, and on-chip oscillator and clock circuits. The 80C451 includes all of the 83C451 features except the on-board 4k X 8 ROM.

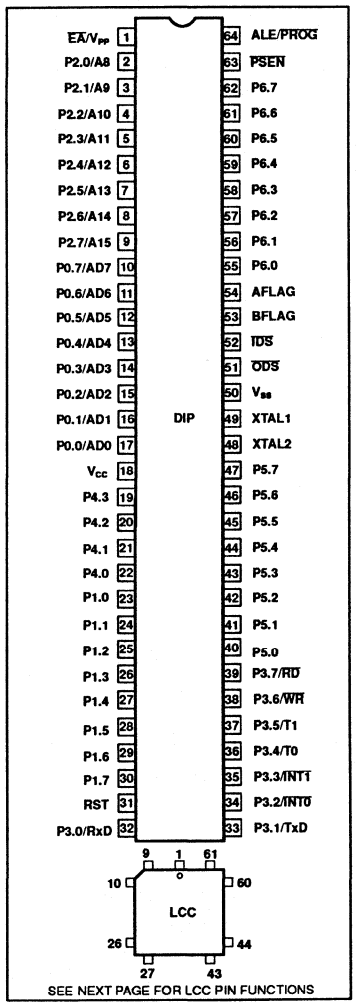
The 87C451 has 4k of EPROM on-chip as program memory and is otherwise identical to the 83C451.

The 8XC451 has two software selectable modes of reduced activity for further power reduction; idle mode and power-down mode. Idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. Power-down mode freezes the oscillator, causing all other chip functions to be inoperative while maintaining the RAM contents.

FEATURES

- 80C51 based architecture
- 68-pin LCC and 64-pin DIP packages:
  - Seven 8-bit I/O ports (LCC version)
  - Six 8-bit ports and one 4-bit port (DIP version)
- Port 6 features:
  - 8 data pins
  - 4 control pins
  - Direct MPU bus interface
  - Parallel printer interface
- On the microcontroller:
  - 4k X 8 ROM (83C451) 4k X 8 EPROM (87C451) ROMless version (80C451)
  - 128 X 8 RAM
  - Two 16-bit counter/timers
  - Two external interrupts
- External memory addressing capability
  - 64k ROM and 64k RAM
- Low power consumption:
  - Normal operation: less than 24mA at 5V, 12MHz
  - Idle mode
  - Power-down mode

PIN CONFIGURATION

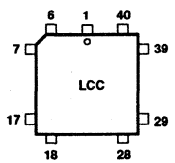


SEE NEXT PAGE FOR LCC PIN FUNCTIONS

PART NUMBER SELECTION

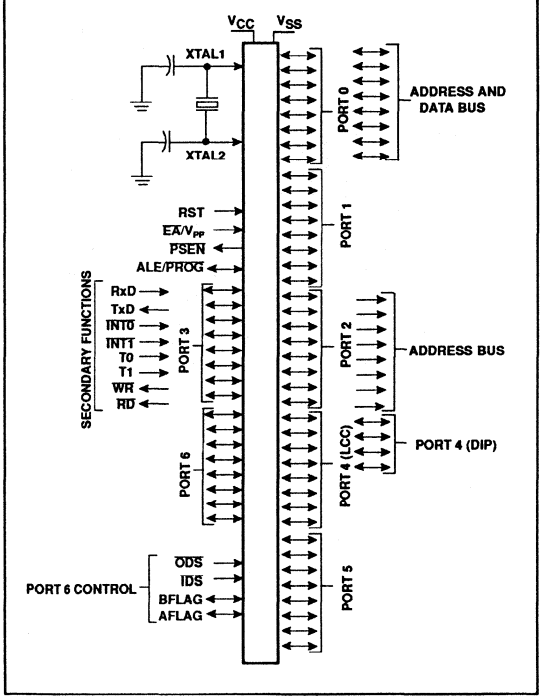
ROMless	ROM	EPROM	TEMPERATURE °C AND PACKAGE	FREQUENCY
SC80C451CCN64	SC83C451CCN64	SC87C451CCN64	0 to +70, plastic DIP	3.5 to 12MHz
SC80C451CGN64	SC83C451CGN64	SC87C451CGN64	0 to +70, plastic DIP	3.5 to 16MHz
SC80C451CCA68	SC83C451CCA68	SC87C451CCA68	0 to +70, plastic LCC	3.5 to 12MHz
SC80C451CGA68	SC83C451CGA68	SC87C451CGA68	0 to +70, plastic LCC	3.5 to 16MHz
SC80C451ACN64	SC83C451ACN64	SC87C451ACN64	–40 to +85, plastic DIP	3.5 to 12MHz
SC80C451AGN64	SC83C451AGN64	SC87C451AGN64	–40 to +85, plastic DIP	3.5 to 16MHz
SC80C451ACA68	SC83C451ACA68	SC87C451ACA68	–40 to +85, plastic LCC	3.5 to 12MHz
SC80C451AGA68	SC83C451AGA68	SC87C451AGA68	–40 to +85, plastic LCC	3.5 to 16MHz
		SC87C451CCIA	0 to +70, ceramic DIP	3.5 to 12MHz
		SC87C451CGIA	0 to +70, ceramic DIP	3.5 to 16MHz
		SC87C451CCL68	0 to +70, ceramic LCC	3.5 to 12MHz
		SC87C451CGL68	0 to +70, ceramic LCC	3.5 to 16MHz
		SC87C451ACIA	–40 to +85, ceramic DIP	3.5 to 12MHz
		SC87C451ACL68	–40 to +85, ceramic LCC	3.5 to 12MHz
		SC87C451AGIA	–40 to +85, ceramic DIP	3.5 to 16MHz
		SC87C451AGL68	–40 to +85, ceramic LCC	3.5 to 16MHz

LCC PIN FUNCTIONS

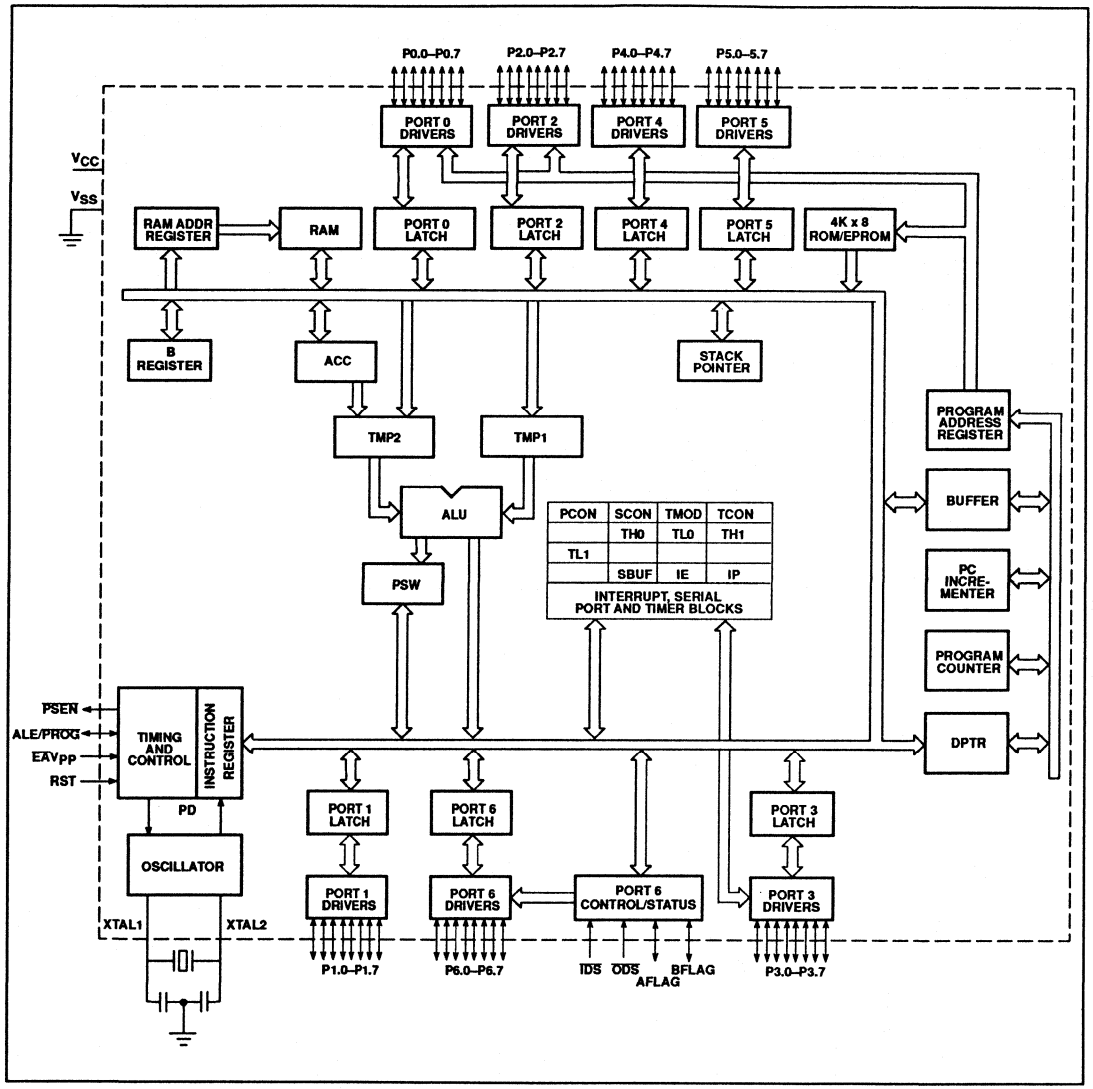


Pin	Function	Pin	Function	Pin	Function
1	$\overline{EA}/V_{PP}$	24	P4.2	47	P5.3
2	P2.0/A8	25	P4.1	48	P5.4
3	P2.1/A9	26	P4.0	49	P5.5
4	P2.2/A10	27	P1.0	50	P5.6
5	P2.3/A11	28	P1.1	51	P5.7
6	P2.4/A12	29	P1.2	52	XTAL2
7	P2.5/A13	30	P1.3	53	XTAL1
8	P2.6/A14	31	P1.4	54	V_{SS}
9	P2.7/A15	32	P1.5	55	$\overline{ODS}$
10	P0.7/AD7	33	P1.6	56	$\overline{IDS}$
11	P0.6/AD6	34	P1.7	57	BFLAG
12	P0.5/AD5	35	RST	58	AFLAG
13	P0.4/AD4	36	P3.0/RxD	59	P6.0
14	P0.3/AD3	37	P3.1/TxD	60	P6.1
15	P0.2/AD2	38	P3.2/$\overline{INT0}$	61	P6.2
16	P0.1/AD1	39	P3.3/$\overline{INT1}$	62	P6.3
17	P0.0/AD0	40	P3.4/T0	63	P6.4
18	V_{CC}	41	P3.5/T1	64	P6.5
19	P4.7	42	P3.6/$\overline{WR}$	65	P6.6
20	P4.6	43	P3.7/$\overline{RD}$	66	P6.7
21	P4.5	44	P5.0	67	$\overline{PAEN}$
22	P4.4	45	P5.1	68	ALE/$\overline{PROG}$
23	P4.3	46	P5.2		

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	TYPE	NAME AND FUNCTION
V_{SS}	50	54	I	**Ground:** 0V reference.
V_{CC}	18	18	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	17–10	17–10	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 is also the multiplexed data and low-order address bus during accesses to external memory. External pull-ups are required during program verification. Port 0 can sink/source eight LS TTL inputs.
P1.0–P1.7	23–30	27–34	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 receives the low-order address bytes during program memory verification. Port 1 can sink/source three LS TTL inputs, and drive CMOS inputs without external pull-ups.
P2.0–P2.7	2–9	2–9	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 emits the high-order address bytes during access to external memory and receives the high-order address bits and control signals during program verification. Port 2 can sink/source three LS TTL inputs, and drive CMOS inputs without external pull-ups.
P3.0–P3.7	32–39	36–43	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 can sink/source three LS TTL inputs, and drive CMOS inputs without external pull-ups. Port 3 also serves the special functions listed below:
	32	36	I	**RxD (P3.0):** Serial input port
	33	37	O	**TxD (P3.1):** Serial output port
	34	38	I	**$\overline{INT0}$ (P3.2):** External interrupt
	35	39	I	**$\overline{INT1}$ (P3.3):** External interrupt
	36	40	I	**T0 (P3.4):** Timer 0 external input
	37	41	I	**T1 (P3.5):** Timer 1 external input
	38	42	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	39	43	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
P4.0–P4.3	22–19		I/O	**Port 4:** Port 4 is a 4/8-bit (DIP/LCC) bidirectional I/O port with internal pull-ups. Port 4 can sink/source three LS TTL inputs and drive CMOS inputs without external pull-ups.
P4.0–P4.7		26–19	I/O	
P5.0–P5.7	40–47	44–51	I/O	**Port 5:** Port 5 is a 4/8-bit (DIP/LCC) bidirectional I/O port with internal pull-ups. Port 5 can sink/source three LS TTL inputs and drive CMOS inputs without external pull-ups.
P6.0–P6.7	55–62	59–66	I/O	**Port 6:** Port 6 is a specialized 8-bit bidirectional I/O port with internal pull-ups. This special port can sink/source three LS TTL inputs and drive CMOS inputs without external pull-ups. Port 6 can be used in a strobed or non-strobed mode of operation. Port 6 works in conjunction with four control pins that serve the functions listed below:
	51	55	I	**$\overline{ODS}$:** Output data strobe
	52	56	I	**$\overline{IDS}$:** Input data strobe
	53	57	I/O	**BFLAG:** Bidirectional I/O pin with internal pull-ups
	54	58	I/O	**AFLAG:** Bidirectional I/O pin with internal pull-ups
RST	31	35	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal pull-down resistor permits a power-on reset using only an external capacitor connected to V_{CC}.
ALE/$\overline{PROG}$	64	68	I/O	**Address Latch Enable/Program Pulse:** Output pulse for latching the low byte of the address during an access to external memory. ALE is activated at a constant rate of 1/6 the oscillator frequency except during an external data memory access, at which time one ALE is skipped. ALE can sink/source three LS TTL inputs and drive CMOS inputs without external pull-ups. This pin is also the program pulse during EPROM programming.
$\overline{PSEN}$	63	67	O	**Program Store Enable:** The read strobe to external program memory. $\overline{PSEN}$ is activated twice each machine cycle during fetches from external program memory. However, when executing out of external program memory, two activations of $\overline{PSEN}$ are skipped during each access to external program memory. $\overline{PSEN}$ is not activated during fetches from internal program memory. $\overline{PSEN}$ can sink/source eight LS TTL inputs and drive CMOS inputs without an external pull-up. This pin should be tied low during programming.
$\overline{EA}$/V_{PP}	1	1	I	**Instruction Execution Control/Programming Supply Voltage:** When $\overline{EA}$ is held high, the CPU executes out of internal program memory, unless the program counter exceeds 0FFFH. When $\overline{EA}$ is held low, the CPU executes out of external program memory. $\overline{EA}$ must never be allowed to float. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
XTAL1	49	53	I	**Crystal 1:** Input to the inverting oscillator amplifier that forms the oscillator. This input receives the external oscillator when an external oscillator is used.
XTAL2	48	52	O	**Crystal 2:** An output of the inverting amplifier that forms the oscillator. This pin should be floated when an external oscillator is used.

PORTS 4 AND 5

Ports 4 and 5 are bidirectional I/O ports with internal pull-ups. Port 4 is an 8-bit port (LCC version) or a 4-bit port (DIP version). Port 4 and port 5 pins with ones written to them, are pulled high by the internal pull-ups, and in that state can be used as inputs. Port 4 and 5 are addressed at the special function register addresses shown in Table 1.

PORT 6

Port 6 is a special 8-bit bidirectional I/O port with internal pull-ups (see Figure 1). This port can be used as a standard I/O port, or in strobed modes of operation in conjunction with four special control lines: $\overline{ODS}$, $\overline{IDS}$, AFLAG, and BFLAG. Port 6 operating modes are controlled by the port 6 control status register (CSR). Port 6 and the CSR are addressed at the special function register addresses shown in Table 1. The following four control pins are used in conjunction with port 6:

$\overline{ODS}$ – Output data strobe for port 6. $\overline{ODS}$ can be programmed to control the port 6 output drivers and the output buffer full flag (OBF), or to clear only the OBF flag bit in the CSR (output-always mode). $\overline{ODS}$ is active low for output driver control. the OBF flag can be programmed to be cleared on the negative or positive edge of $\overline{ODS}$.

$\overline{IDS}$ – Input data strobe for port 6. $\overline{IDS}$ is used to control the port 6 input latch and input buffer full flag (IBF) bit in the CSR. The input data latch can be programmed to be transparent when $\overline{IDS}$ is low and latched on the positive transition of $\overline{IDS}$, or to latch only on the positive transition of $\overline{IDS}$. Correspondingly, the IBF flag is set on the negative or positive transition of $\overline{IDS}$.

AFLAG – AFLAG is a bidirectional I/O pin which can be programmed to be an output set high or low under program control, or to output the state of the output buffer full flag. AFLAG can also be programmed to be an input which selects whether the contents of the output buffer, or the contents of the port 6 control status register will output on port 6. This feature grants complete port 6 status to external devices.

BFLAG – BFLAG is a bidirectional I/O pin which can be programmed to be an output, set high or low under program control, or to output the state of the input buffer full flag. BFLAG can also be programmed to input an enable signal for port 6. When BFLAG is used as an enable input, port 6 output drivers are in the high-impedance state, and the input latch does not respond to the $\overline{IDS}$ strobe when BFLAG is high. Both features are enabled when BFLAG is low. This feature facilitates the use of the SC8XC451 in bused multiprocessor systems.

CONTROL STATUS REGISTER

The control status register (CSR) establishes the mode of operation for port 6 and indicates the current status of port 6 I/O registers. All control status register bits can be read and written by the CPU, except bits 0 and 1, which are read only. Reset writes ones to bits 2 through 7, and writes zeros to bits 0 and 1 (see Table 2).

CSR.0 Input Buffer Full Flag (IBF) (Read Only) – The IBF bit is set to a logic 1 when port 6 data is loaded into the input buffer under control of $\overline{IDS}$. This can occur on the negative or positive edge of $\overline{IDS}$, as determined by CSR.2 IBF is cleared when the CPU reads the input buffer register.

CSR.1 Output Buffer Full Flag (OBF) (Read Only) – The OBF flag is set to a logic 1 when the CPU writes to the port 6 output data buffer. OBF is cleared by the positive or negative edge of $\overline{ODS}$, as determined by CSR.3.

Table 1. Special Function Register Addresses

REGISTER ADDRESS			BIT ADDRESS							
Name	Symbol	Address	MSB							LSB
Port 4	P4	C0	C7	C6	C5	C4	C3	C2	C1	C0
Port 5	P5	C8	CF	CE	CD	CC	CB	CA	C9	C8
Port 6 data	P6	D8	DF	DE	DD	DC	DB	DA	D9	D8
Port 6 control status	CSR	E8	EF	EE	ED	EC	EB	EA	E9	E8

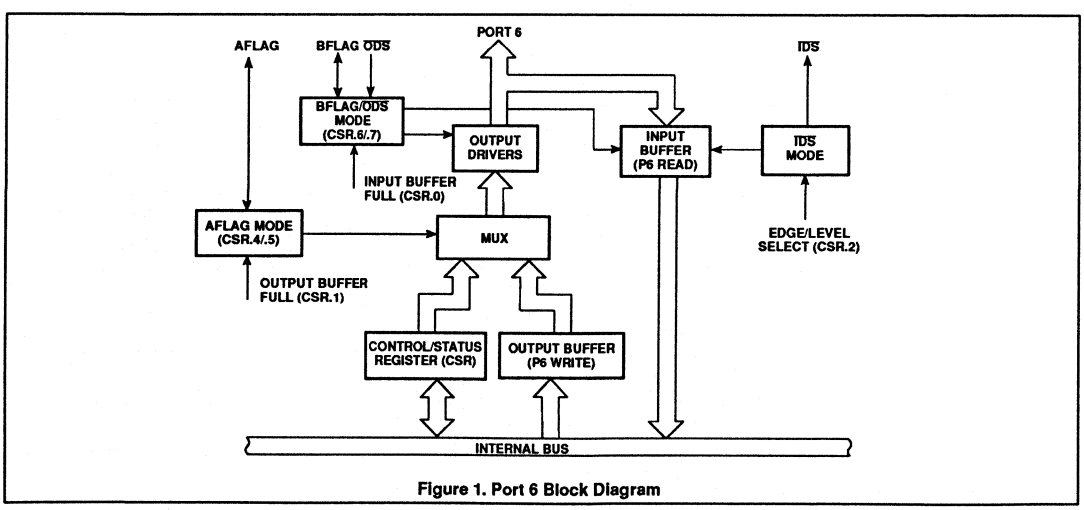


Figure 1. Port 6 Block Diagram

CSR.2 IDS Mode Select (IDSM) – When CSR.2 = 0, a low-to-high transition on the $\overline{IDS}$ pin sets the IBF flag. The Port 6 input buffer is loaded on the $\overline{IDS}$ positive edge. When CSR.2 = 1, a high-to-low transition on the $\overline{IDS}$ pin sets the IBF flag. Port 6 input buffer is transparent when $\overline{IDS}$ is low, and latched when $\overline{IDS}$ is high.

CSR.3 Output Buffer Full Flag Clear Mode (OBFC) – When CSR.3 = 1, the positive edge of the $\overline{ODS}$ input clears the OBF flag. When CSR.3 = 0, the negative edge of the $\overline{ODS}$ input clears the OBF flag.

CSR.4, CSR.5 AFLAG Mode Select (MA0, MA1) – Bits 4 and 5 select the mode of operation for the AFLAG pin as follows:

MA1	MA0	AFLAG Function
0	0	Logic 0 output
0	1	Logic 1 output
1	0	OBF flag output (CSR.1)
1	1	Select (SEL) input mode

The select (SEL) input mode is used to determine whether the port 6 data register or the control status register is output on port 6. When the select feature is enabled, the AFLAG input controls the source of port 6 output data. A logic 0 on AFLAG input selects the port 6 data register, and a logic 1 on AFLAG input selects the control status register.

CSR.6, CSR.7 BFLAG Mode Select (MB0, MB1) – Bits 6 and 7 select the mode operation as follows:

MB1	MB0	BFLAG Function
0	0	Logic 0 output
0	1	Logic 1 output
1	0	IBF flag output (CSR.0)
1	1	Port enable ($\overline{PE}$)

In the port enable mode, $\overline{IDS}$ and $\overline{ODS}$ inputs are disabled when BFLAG input is high. When the BFLAG input is low, the port is enabled for I/O.

SPECIAL FUNCTION REGISTER ADDRESSES

Special function register addresses for the device are identical to those of the 80C51, except for the additional registers listed in Table 1.

Table 2. Control Status Register (CSR)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
BFLAG Mode Select		AFLAG Mode Select		Output Buffer Flag Clear Mode	Input Data Strobe Mode	Output Buffer Flag Full	Input Buffer Flag Full
0/0 = Logic 0 output* 0/1 = Logic 1 output* 1/0 = IBF output 1/1 = $\overline{PE}$ input (0 = Select) (1 = Disable I/O)		0/0 = Logic 0 output* 0/1 = Logic 1 output* 1/0 = OBF output 1/1 = SEL input (0 = Select) (1 = Control/status)		0 = Negative edge of $\overline{ODS}$ 1 = Positive edge o $\overline{ODS}$	0 = Negative edge of $\overline{IDS}$ 1 = Positive edge o $\overline{IDS}$	0 = Output data buffer empty 1 = Output data buffer full	0 = Input data buffer empty 1 = Input data buffer full

NOTE:
*Output-always mode: MB1 = 0, MA1 = 1, and MA0 = 0. In this mode, port 6 is always enabled for output. $\overline{ODS}$ only clears the OBF flag.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Operating temperature under bias	0 to +70 –40 to +85	°C
Storage temperature range	–65 to +150	°C
Voltage on any other pin to V_{SS}	–0.5 to +6.5	V
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.5	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±20%, V_{SS} = 0V (80C451, 83C451)

T_A = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V (87C451)

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			MIN	TYPICAL[1]	MAX	
V_{IL}	Input low voltage; except $\overline{EA}$		–0.5		$0.2V_{CC}$–0.1	V
V_{IL1}	Input low voltage to $\overline{EA}$		0		$0.2V_{CC}$–0.3	V
V_{IH}	Input high voltage; except XTAL1, RST		$0.2V_{CC}$+0.9		V_{CC}+0.5	V
V_{IH1}	Input high voltage; XTAL1, RST		$0.7V_{CC}$		V_{CC}+0.5	V
V_{OL}	Output low voltage; ports 1, 2, 3	I_{OL} = 1.6mA[2]			0.45	V
V_{OL1}	Output low voltage; port 0, ALE, $\overline{PSEN}$	I_{OL} = 3.2mA[2]			0.45	V
V_{OH}	Output high voltage; ports 1, 2, 3, 4, 5, 6	I_{OH} = –60µA, I_{OH} = –25µA I_{OH} = –10µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$			V V V
V_{OH1}	Output high voltage (port 0 in external bus mode, ALE, $\overline{PSEN}$)[3]	I_{OH} = –800µA, I_{OH} = –300µA I_{OH} = –80µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$			V V V
I_{IL}	Logical 0 input current,; ports 1, 2, 3, 4, 5, 6	V_{IN} = 0.45V			–50	µA
I_{TL}	Logical 1-to-0 transition current; ports 1, 2, 3	See note 4			–650	µA
I_{LI}	Input leakage current; port 0	V_{IN} = V_{IL} or V_{IH}			±10	µA
I_{CC}	Power supply current: Active mode @ 12MHz[5] Idle mode @ 12MHz[5] Power down mode	See note 6		 11.5 1.3 3	 25 4 50	 mA mA µA
R_{RST}	Internal reset pull-down resistor		50		300	kohm
C_{IO}	Pin capacitance[7] – DIP package – PLCC package				15 10	pF pF

NOTES:
1. Typical ratings are based on a limited number of samples taken from early manufacturing lots and are not guaranteed. The values listed are at room temperature, 5V.
2. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input..
3. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
4. Pins of ports 1, 2 and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
5. I_{CC}MAX at other frequencies is given by:
   Active mode: I_{CC}MAX = 0.94 X FREQ + 13.71
   Idle mode: I_{CC}MAX = 0.14 X FREQ +2.31
   where FREQ is the external oscillator frequency in MHz. I_{CC}MAX is given in mA. See Figure 13.
6. See Figures 14 through 17 for I_{CC} test conditions.
7. C_{IO} applies to ports 1 through 6, AFLAG, BFLAG, XTAL1, XTAL2.

AC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±20%, V_{SS} = 0V (80C451, 83C451)[1,2]

T_A = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V (87C451)

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$		Oscillator frequency: **Speed Versions** SC8XC451 B SC8XC451 C SC8XC451 G			 0.5 3.5 3.5	 12 12 16	 MHz MHz MHz
t_{LHLL}	2	ALE pulse width	127		$2t_{CLCL}$–40		ns
t_{AVLL}	2	Address valid to ALE low	28		t_{CLCL}–55		ns
t_{LLAX}	2	Address hold after ALE low	48		t_{CLCL}–35		ns
t_{LLIV}	2	ALE low to valid instruction in		234		$4t_{CLCL}$–100	ns
t_{LLPL}	2	ALE low to $\overline{PSEN}$ low	43		t_{CLCL}–40		ns
t_{PLPH}	2	$\overline{PSEN}$ pulse width	205		$3t_{CLCL}$–45		ns
t_{PLIV}	2	$\overline{PSEN}$ low to valid instruction in		145		$3t_{CLCL}$–105	ns
t_{PXIX}	2	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	2	Input instruction float after $\overline{PSEN}$		59		t_{CLCL}–25	ns
t_{AVIV}	2	Address to valid instruction in		312		$5t_{CLCL}$–105	ns
t_{PLAZ}	2	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{RLRH}	3, 4	$\overline{RD}$ pulse width	400		$6t_{CLCL}$–100		ns
t_{WLWH}	3, 4	$\overline{WR}$ pulse width	400		$6t_{CLCL}$–100		ns
t_{RLDV}	3, 4	$\overline{RD}$ low to valid data in		252		$5t_{CLCL}$–165	ns
t_{RHDX}	3, 4	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	3, 4	Data float after $\overline{RD}$		97		$2t_{CLCL}$–70	ns
t_{LLDV}	3, 4	ALE low to valid data in		517		$8t_{CLCL}$–150	ns
t_{AVDV}	3, 4	Address to valid data in		585		$9t_{CLCL}$–165	ns
t_{LLWL}	3, 4	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}$–50	$3t_{CLCL}$+50	ns
t_{AVWL}	3, 4	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}$–130		ns
t_{QVWX}	3, 4	Data valid to $\overline{WR}$ transition	23		t_{CLCL}–60		ns
t_{WHQX}	3, 4	Data hold after $\overline{WR}$	33		t_{CLCL}–50		ns
t_{RLAZ}	3, 4	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	3, 4	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	t_{CLCL}–40	t_{CLCL}+40	ns
Shift Register							
t_{XLXL}	5	Serial port clock cycle time	1.0		$12t_{CLCL}$		µs
t_{QVXH}	5	Output data setup to clock rising edge	700		$10t_{CLCL}$–133		ns
t_{XHQX}	5	Output data hold after clock rising edge	50		$2t_{CLCL}$–117		ns
t_{XHDX}	5	Input data hold after clock rising edge	0		0		ns
t_{XHDV}	5	Clock rising edge to input data valid		700		$10t_{CLCL}$–133	ns

NOTES:

1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.

AC ELECTRICAL CHARACTERISTICS (Continued)

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
Port 6 input (input rise and fall times = 5ns)							
t_{FLFH}	8	$\overline{PE}$ width	270		$3t_{CLCL}+20$		ns
t_{ILIH}	8	$\overline{IDS}$ width	270		$3t_{CLCL}+20$		ns
t_{DVIH}	8	Data setup to $\overline{IDS}$ high or $\overline{PE}$ high	0		0		ns
t_{IHDX}	8	Data hold after $\overline{IDS}$ high or $\overline{PE}$ high	30		30		ns
t_{IVFV}	9	$\overline{IDS}$ to BFLAG (IBF) delay		130		130	ns
Port 6 output							
t_{OLOH}	6	$\overline{ODS}$ width	270		$3t_{CLCL}+20$		ns
t_{FVDV}	7	SEL to data out delay		85		85	ns
t_{OLDV}	6	$\overline{ODS}$ to data out delay		80		80	ns
t_{OHDZ}	6	$\overline{ODS}$ to data float delay		35		35	ns
t_{OVFV}	6	$\overline{ODS}$ to AFLAG (OBF) delay		100		100	ns
t_{FLDV}	6	$\overline{PE}$ to data out delay		120		120	ns
t_{OHFH}	7	$\overline{ODS}$ to AFLAG (SEL) delay	100		100		ns
External Clock							
t_{CHCX}	10	High time	20		20		ns
t_{CLCX}	10	Low time	20		20		ns
t_{CLCH}	10	Rise time		20		20	ns
t_{CHCL}	10	Fall time		20		20	ns

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

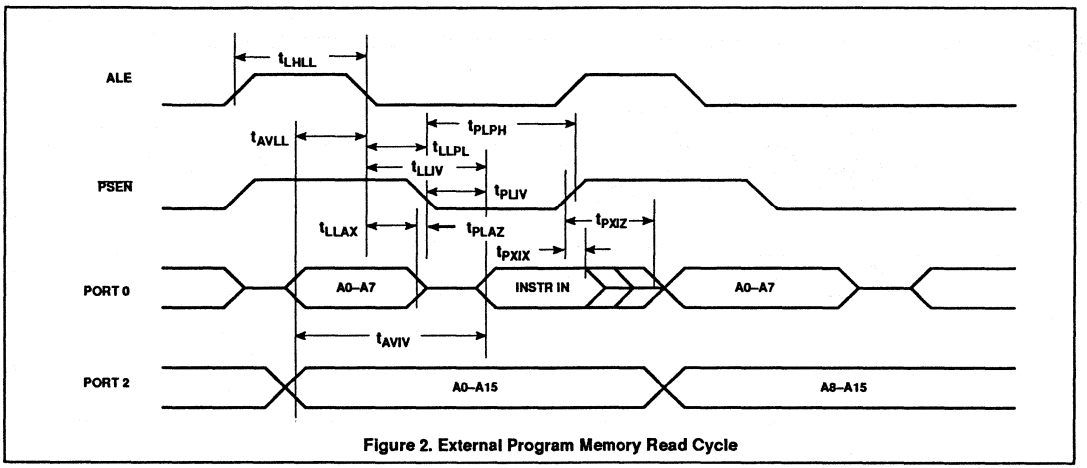


Figure 2. External Program Memory Read Cycle

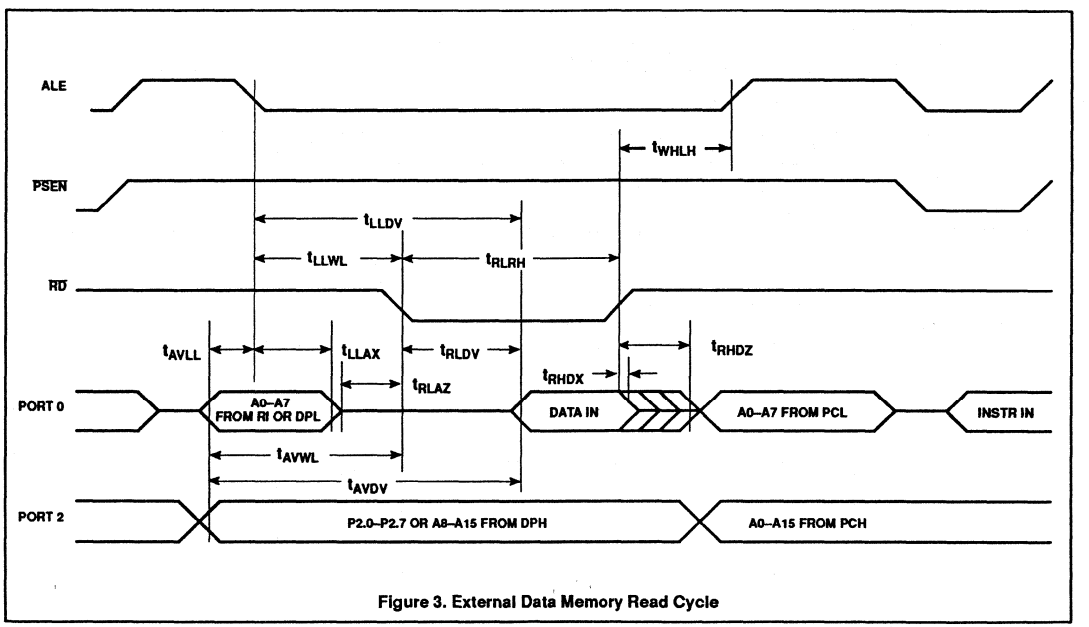


Figure 3. External Data Memory Read Cycle

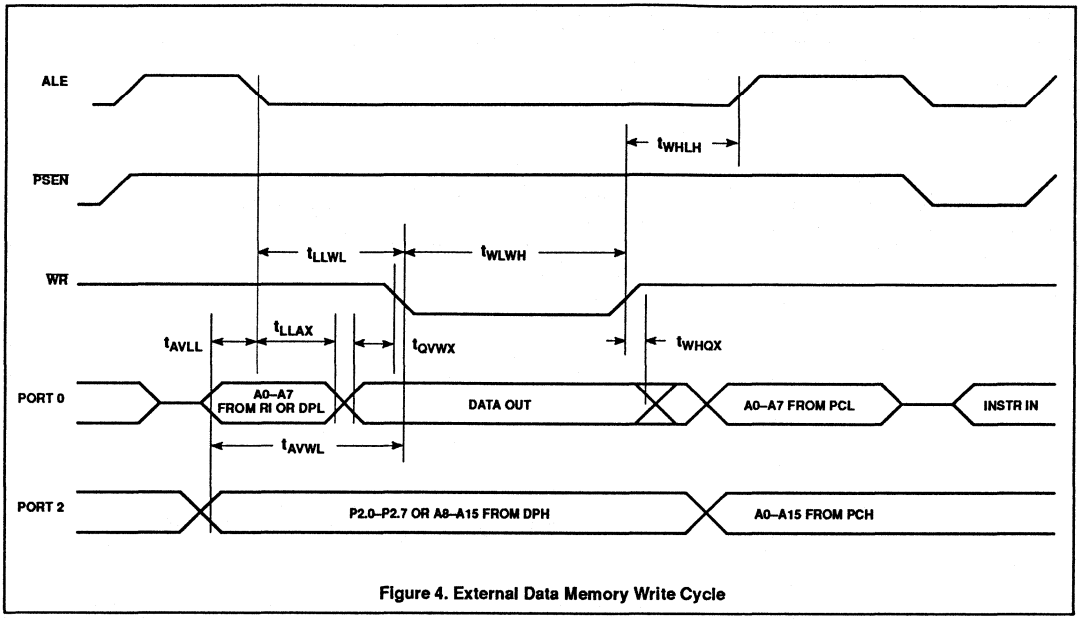


Figure 4. External Data Memory Write Cycle

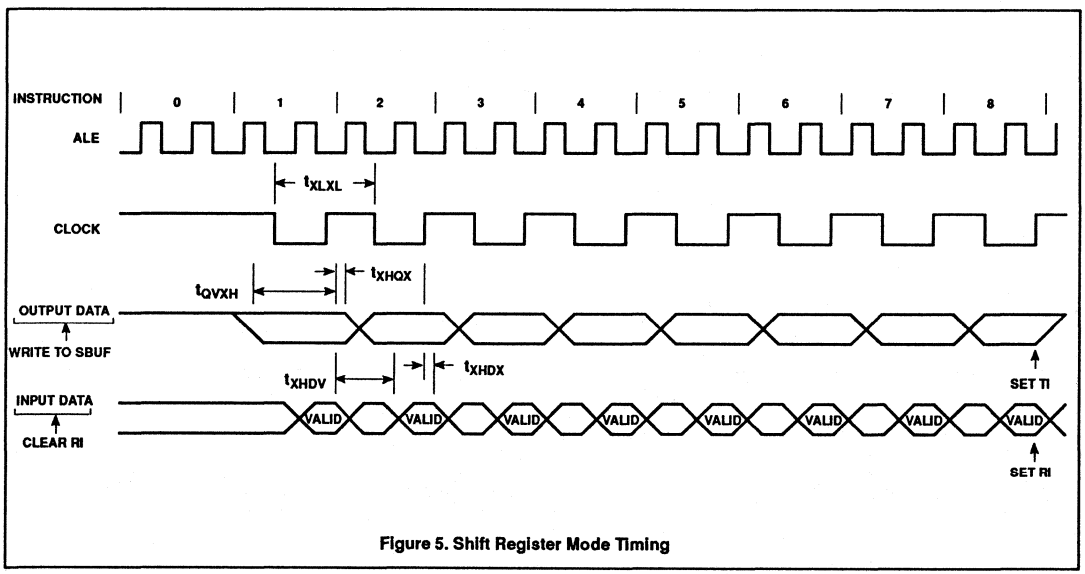


Figure 5. Shift Register Mode Timing

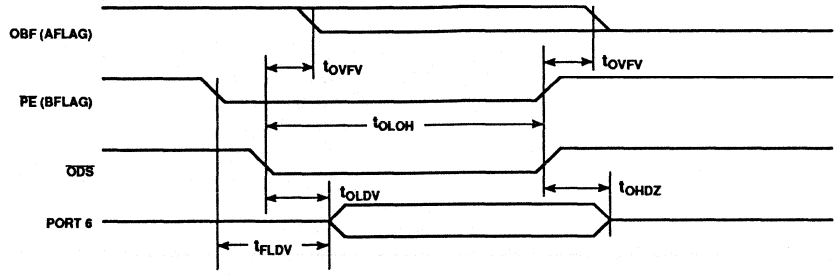


Figure 6. Port 6 Output

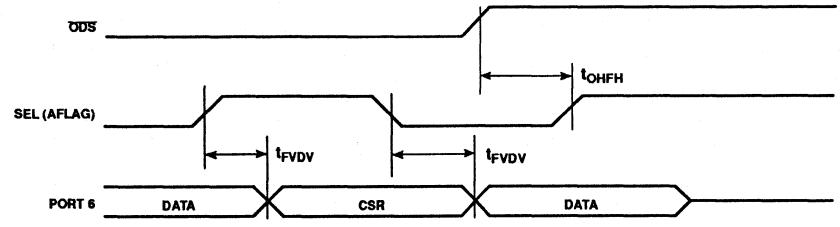


Figure 7. Port 6 Select Mode

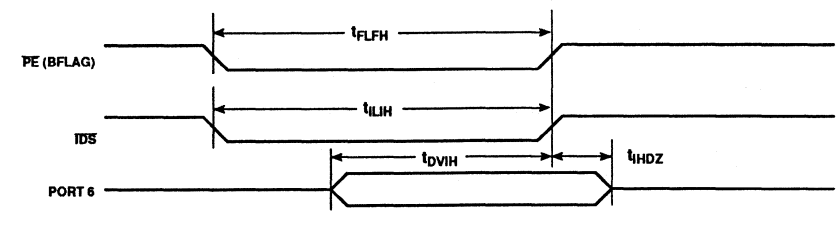


Figure 8. Port 6 Input

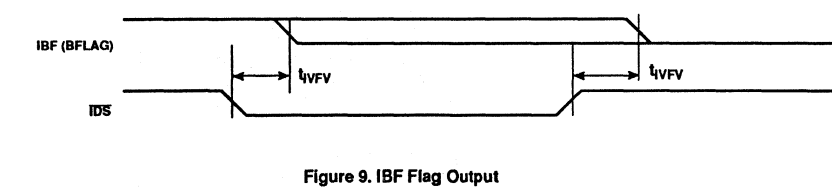


Figure 9. IBF Flag Output

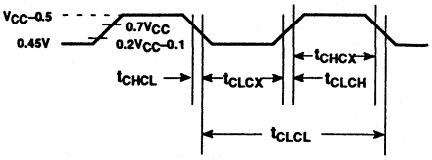
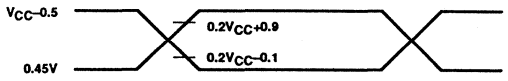
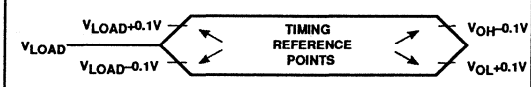


Figure 10. External Clock Drive



NOTE:
AC inputs during testing are driven at V_{CC} –0.5 for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at V_{IH} min for a logic '1' and V_{IL} for a logic '0'.

Figure 11. AC Testing Input/Output



NOTE:
For timing purposes, a port is no longer floating when a 100mV change from load voltage occurs, and begins to float when a 100mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OH}/I_{OL} \geq \pm 20$mA.

Figure 12. Float Waveform

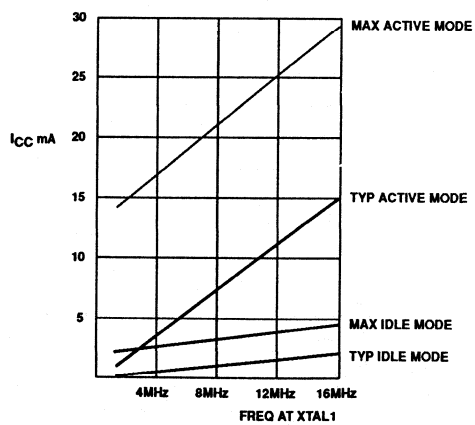


Figure 13. I_{CC} vs. FREQ
Valid only within frequency specifications of the device under test

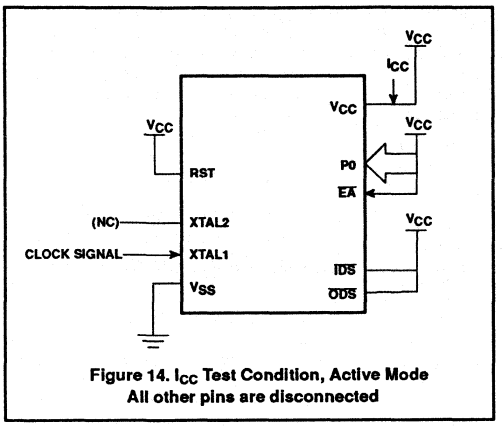


Figure 14. I_{CC} Test Condition, Active Mode
All other pins are disconnected

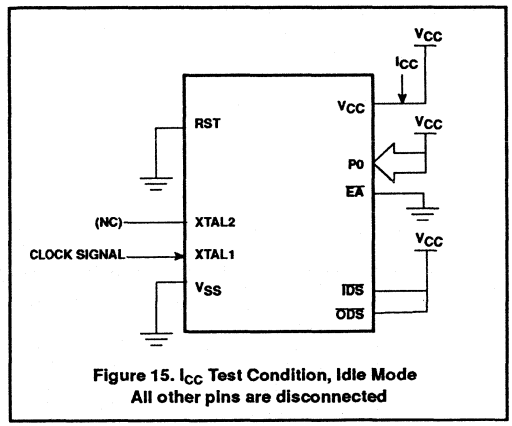


Figure 15. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

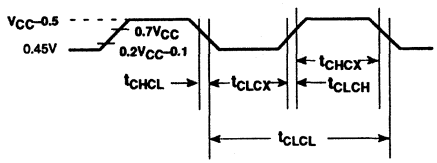


Figure 16. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

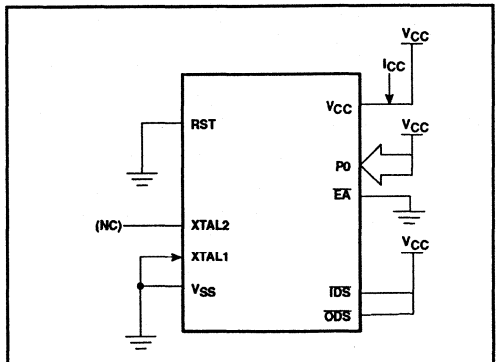


Figure 17. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected. V_{CC} = 2V to 5.5V

EPROM CHARACTERISTICS

The 87C451 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C451 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C451 manufactured by Philips Corporation.

Table 3 shows the logic levels for reading the signature byte, and for programming the program memory, the encryption table, and the lock bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 18 and 19. Figure 20 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 18. Note that the 87C451 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1 and 2, as shown in Figure 18. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$ and pins of ports 2 and 3 specified in Table 3 are held at the 'Program Code Data' levels indicated in Table 3. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 19.

To program the encryption table, repeat the 25 pulse programming sequence for addresses 0 through 1FH, using the "Pgm Encryption Table' levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the lock bits, repeat the 25 pulse programming sequence using the 'Pgm Lock Bit' levels. After one lock bit is programmed, further programming of the code memory and encryption table is disabled. However, the other lock bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If lock bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be read is applied to ports 1 and 2 as shown in Figure 20. The other pins are held at the 'Verify Code Data' levels indicated in Table 3. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips

(031H) = 90H indicates 87C451

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 3, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345–5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000uW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

Table 3. EPROM Progamming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm lock bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm lock bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:

1. '0' = Valid low for that pin, '1' = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{CC} = 5V±10% during programming and verification.

*ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

™Trademark phrase of Intel Corporation.

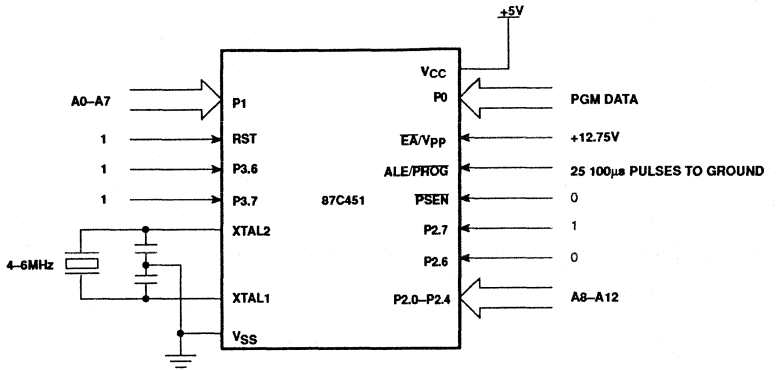


Figure 18. Programming Configuration

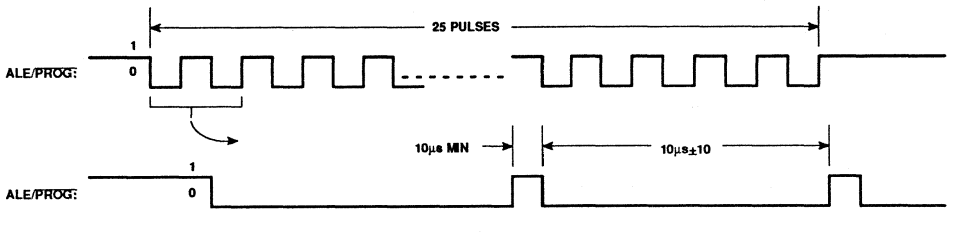


Figure 19. PROG Waveform

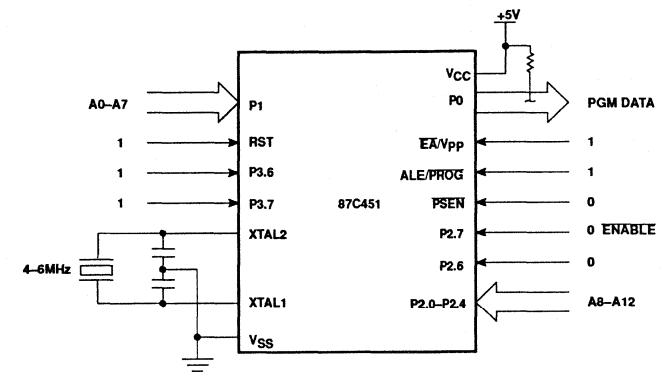


Figure 20. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_A = 21°C to +27°C, V_{CC} = 5V±10%, V_{SS} = 0V (See Figure 21)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		µs
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	$\overline{ENABLE}$ low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		µs

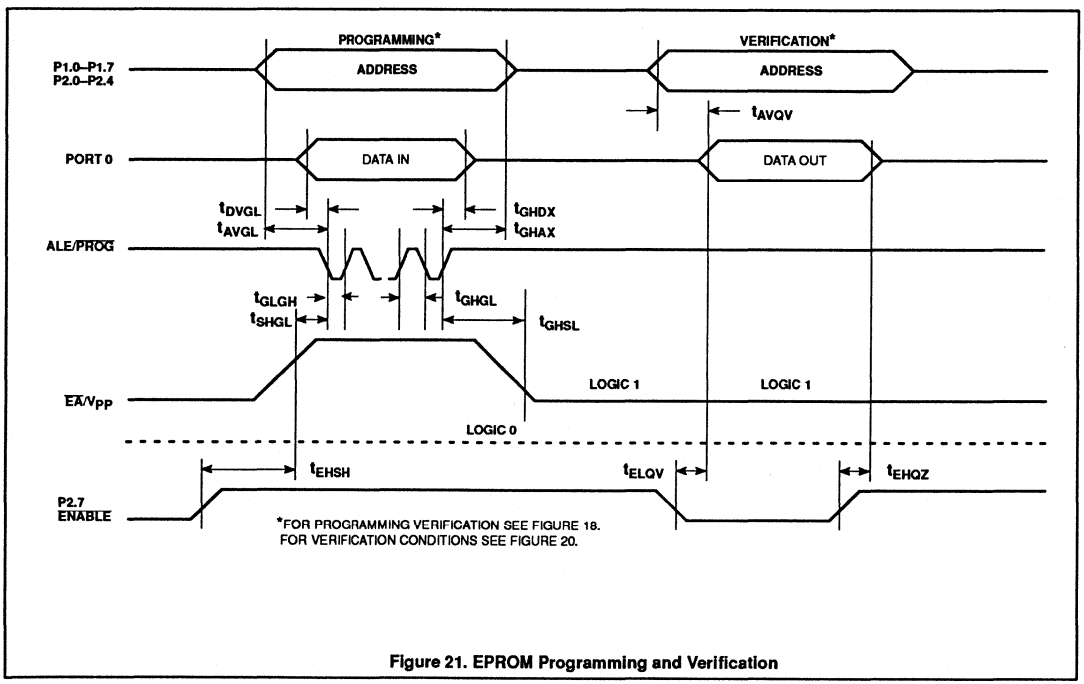


Figure 21. EPROM Programming and Verification

8XC528 OVERVIEW

The 8XC528 is a high-performance single-chip microcontroller manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. With the exception of open-drain outputs on two port pins, it is fully compatible with the industry standard 80C51 and includes a number of additional enhancements. It is well suited for applications requiring large amounts of on-chip ROM and RAM. Additional special function registers are incorporated to control the on-chip peripherals.

The 8XC528 contains a non-volatile 32k x 8 read-only program memory, a volatile 512 x 8 read/write data memory, four 8-bit I/O ports, two 16-bit timer/counters (identical to those in the 80C51), a 16-bit timer coupled to capture and compare registers (identical to T2 of the 80C52), a multisource two-priority level nested interrupt structure, two serial interfaces (a standard UART and I^2C bus), a watchdog timer with separate oscillator, and a master oscillator.The 8XC528 also includes ROM code protection and enhanced recovery from power-down mode.

Differences from the 80C51

The 8XC528 contains 32 kbytes of on-chip program memory which can be extended to 64 kbytes with external memories (see Figure 16).

When the $\overline{EA}$ pin is held high, the 8XC528 fetches instructions from internal ROM unless the address exceeds 7FFFH. Locations 8000H to FFFFH are fetched from external program memory. When the $\overline{EA}$ pin is held low, all instruction fetches are from external memory.

By setting a mask programmable security bit, the internal ROM contents are protected and cannot be read with the help of test modes or by execution of MOVC instructions from external program memory. The operation of the MOVC instructions in internal and external program memory with the security bit is as follows:

FUNCTION	ACCESS TO INTERNAL PROGRAM MEMORY	ACCESS TO EXTERNAL PROGRAM MEMORY
MOVC in internal program memory	Yes	Yes
MOVC in external program memory	No	Yes

There are no restrictions on the operation of MOVC instructions if the security bit is cleared (logic zero). The state of the EA pin is latched during reset, and its status following reset is ignored. This prevents reading from internal program memory by switching from external to internal mode during the execution of a MOVC instruction.

Data Memory

The internal data memory is divided into four physically separate sections: the lower 128 bytes of RAM, a second 128 bytes of RAM, a 256-byte auxiliary RAM (AUX-RAM), and the 128-byte special function register (SFR) space.

The lower 128 bytes of RAM (addresses 0 to 7FH) are directly and indirectly addressable and correspond to the 128 bytes of RAM in the 80C51. The second 128 bytes of RAM and the special function registers share the same address space but are accessed through different addressing modes.

RAM locations 80H to FFH are only indirectly addressable while the special function registers are only directly addressable. This is the same addressing method used in the 80C52.

The 256-byte AUX-RAM, while physically located on-chip, logically occupies the first 256 bytes of external data memory. As such, it is indirectly addressed in the same way as external data memory using the MOVX instructions in combination with any of the registers R0, R1, or DPTR. Accesses to AUX-RAM locations (0 to FFH) will not affect ports P0, P2, or pins P3.6 or P3.7.

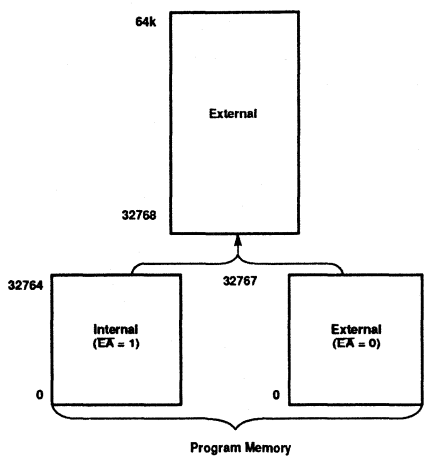


Figure 16. Program Memory

Access to external data memory locations 100H to FFFFH will perform normally. The stack may be located anywhere in the internal data memory by loading the 8-bit stack pointer and has a maximum depth of 256 bytes. The stack may not be located in AUX-RAM. Figure 17 shows the data memory map of the 8XC528 along with a summary of accessing modes.

Special Function Registers

The special function registers contain all of the 8XC528 registers except the program counter and the four register banks. Most of the 31 special function registers are used to control on-chip peripheral hardware. Other registers include arithmetic registers (ACC, B, PSW), stack pointer (SP), and data pointer registers (DPH, DPL). Thirteen of the SFRs are bit addressable. Table 11 lists the 8XC528 special function registers.

The standard 80C52 SFRs are present and function identically in the 8XC528. SFRs SCON and SBUF of the 80C51 have been renamed S0CON and S0BUF, respectively.

Watchdog Timer

In addition to timers T0, T1, and T2, the 8XC528 also includes a watchdog timer, T3. The purpose of a watchdog timer is to reset the microcontroller within a reasonable time should it enter an erroneous processor state (possibly caused by electrical noise or RFI). When enabled, the watchdog circuitry will generate a system reset if the user program fails to reload the watchdog timer prior to the timer overflowing.

The watchdog timer consists of an 11-bit prescaler and an 8-bit timer, T3, shown in Figure 18. The prescaler is incremented by a dedicated on-chip oscillator with a fixed frequency of 1MHz with a tolerance of +100% and –50%. The 8-bit timer, T3, increments every 2048 cycles of this dedicated oscillator.

When a timer overflow occurs, the 8XC528 is reset, and a reset pulse of 16 x 2048 cycles of the dedicated oscillator is generated at the reset pin.

The internal reset signal is not inhibited when the external reset pin is held low by an external circuit. The watchdog timer is controlled by the special function register WDCON. After a reset signal, WDCON will contain the value A5H, which halts the dedicated oscillator and clears both the prescaler and timer T3. Any value stored in WDCON other than A5H will enable the watchdog timer.

Timer T3 can be read on the fly. Timer T3 can only be written if WDCON contains the value 5AH. A successful write operation to T3 will clear the prescaler and WDCON, leaving the watchdog enabled and preventing inadvertent changes of T3.

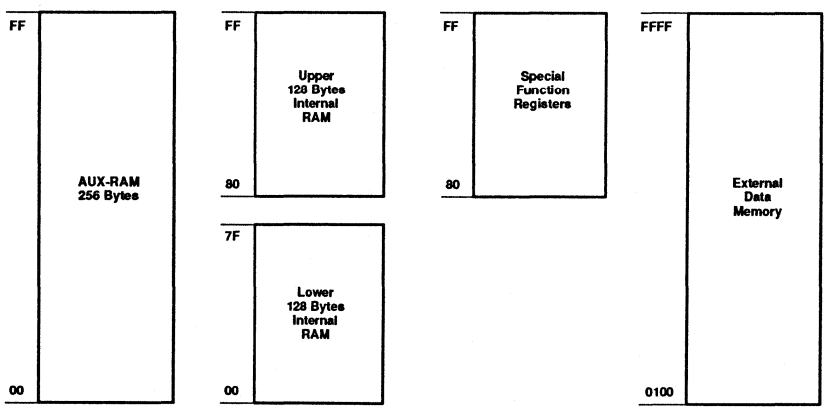


Figure 17. Internal Data Memory Address Space

Table 11. 8XC528 Special Function Register

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR:	Data pointer (2 bytes):										
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*	Interrupt enable	A8H	EA	ES1	ET2	ES0	ET1	EX1	ET0	EX0	00H
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*	Interrupt priority	B8H	–	PS1	PT2	PS0	PT1	PX1	PT0	PX0	x0000000B
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	FFH
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	–	–	–	–	–	–	T2EX	T2	FFH
			A7	A6	A5	A4	A3	A2	A1	A0	
P2*	Port 2	A0H	A15	A14	A13	A12	A11	A10	A9	A8	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	RD	WR	T0	T1	INT1	INT0	TxD	RxD	FFH
PCON	Power control	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxx0000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
RCAP2H#	Capture high	CBH									00H
RCAP2L#	Capture low	CAH									00H
S0BUF	Serial data buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
S0CON*	Serial controller	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
S1BIT	Serial I^2C data	D9H/RD	SDI	0	0	0	0	0	0	0	80H
		WR	SD0	X	X	X	X	X	X	X	
S1INT	Serial I^2C interrupt	DAH	INT	X	X	X	X	X	X	X	00H
			DF	DE	DD	DC	DB	DA	D9	D8	
S1SCS	Serial I^2C control	D8H/RD	SDI	SCI	CLH	BB	RBF	WBF	STR	ENS	00H
		WR	SD0	SC0	CLH	X	X	X	STR	ENS	
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
			CF	CE	CD	CC	CB	CA	C9	C8	
T2CON*#	Timer 2 control	C8H	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	00H
TH0	Timer high 0	8CH									00H
TH1	Timer high 1	8DH									00H
TH2#	Timer high 2	CDH									00H
TL0	Timer low 0	8AH									00H
TL1	Timer low 1	8BH									00H
TL2#	Timer low 2	CCH									00H
T3	Watchdog timer	FFH									00H
TMOD	Timer mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H
WDCON	Watchdog control	A5H									A5H

*SFRs are bit addressable.
#SFRs are modified from or added to the 80C51 SFRs.

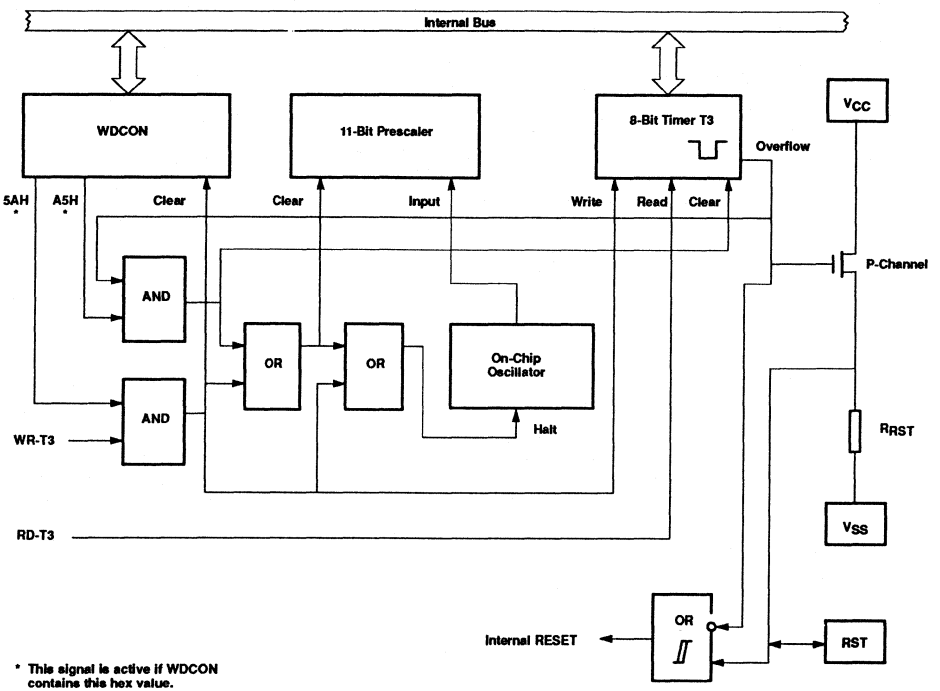


Figure 18. Watchdog Timer T3

During a read or write operation to T3, the output of the dedicated oscillator is inhibited to prevent timing problems due to asynchronous increments of T3. To prevent an overflow of the watchdog timer, the user program has to reload the watchdog timer within periods that are shorter than the programmed watchdog interval. This time interval is determined by the 8-bit value loaded into T3.

$$\text{Watchdog interval} = \frac{[256 - (T3)] \times 2048}{\text{dedicated osc freq}}$$

I²C Interface

Support of the I²C bus on the 8XC528 is provided by a bit-level serial interface. This interface is supported by registers S1INT, S1BIT, and S1SCS in conjunction with pins P1.6/SCL and P1.7/SDA. These latter two pins meet the I²C bus specifications for input and output drive levels and consequently have open drain outputs. All four modes of the I²C bus are supported: master transmitter, master receiver, slave transmitter, and slave receiver. A 100kbps data rate can be achieved with a master oscillator frequency of 12MHz.

The I²C interface on the 8XC528 performs the following functions:

- Generates an interrupt on reception of a START condition
- Recognizes a STOP condition and indicates busy or free status of bus
- Latches a received serial bit
- Generates a single serial clock pulse on SCL pin
- Performs serial clock synchronization
- Detects bit-level arbitration loss

The three SFRs used for I²C are:

S1INT

7	6	5	4	3	2	1	0
INT	X	X	X	X	X	X	X

S1BIT (READ))

7	6	5	4	3	2	1	0
SDI	0	0	0	0	0	0	0

(WRITE)

7	6	5	4	3	2	1	0
SDO	X	X	X	X	X	X	X

S1SCS (READ)

7	6	5	4	3	2	1	0
SDI	SCI	CLH	BB	RBF	WBF	STR	ENS

(WRITE)

7	6	5	4	3	2	1	0
SDO	SCO	CLH	X	X	X	STR	ENS

Interrupts

The interrupt structure of the 8XC528 is the same as that used in the 80C51 but includes two additional interrupt sources: one for the third timer/counter, T2, and one for the I²C interface. The interrupt enable and interrupt priority registers are IE and IP.

IE (A8H)

7	6	5	4	3	2	1	0
EA	ES1	ET2	ES0	ET1	EX1	ET0	EX0

Symbol	Position	Function
EA	IE.7	General enable/disable control 0 – No interrupt is enabled 1 – Any individually enabled interrupt will be accepted
ES1	IE.6	Enable bit-level I²C I/O interrupt
ET2	IE.5	Enable timer 2 interrupt
ES0	IE.4	Enable serial port interrupt
ET1	IE.3	Enable timer 1 interrupt
EX1	IE.2	Enable external 1 interrupt
ET0	IE.1	Enable timer 0 interrupt
EX0	IE.0	Enable external 0 interrupt

IP (B8H)

7	6	5	4	3	2	1	0
–	PS1	PT2	PS0	PT1	PX1	PT0	PX0

Symbol	Position	Function
–	IP.7	Reserved
PS1	IP.6	Bit-level I²C interrupt priority level
PT2	IP.5	Timer 2 interrupt priority level
PS0	IP.4	Serial port interrupt priority level
PT1	IP.3	Timer 1 interrupt priority level
PX1	IP.2	External interrupt 1 priority level
PT0	IP.1	Enable timer 0 interrupt
PX0	IP.0	External interrupt 0 priority level

The interrupt vector locations and the interrupt priorities are:

Source	Priority within Level
Vector Address	Highest
0003H TF2+EXF2	
002BH SI (I2C)	
0053H TF0	
000BH IE1	
0013H TF1	
001BH RI+TI	
0023H IE0	Lowest

Idle Mode

Idle and power-down operation is similar to that used in the 80C51. Idle mode permits the interrupts, serial ports, and timers to function while the CPU operation is halted. During idle mode, the following functions remain active and may generate an interrupt or reset ending the idle mode:

- Timer 0, 1, 2, or 3 (watchdog)
- Standard async UART
- I²C interface
- External interrupts

Idle mode is entered by setting bit PCON.0. Once in the idle mode the CPU status is preserved, and all registers and RAM maintain their data. The status of device pins during idle mode is shown in Table 12. Idle mode is terminated by the activation of any enabled interrupt or by the occurrence of a reset signal (including the watchdog timer overflow).

Power-Down Mode

During power-down mode, the master oscillator is stopped, CPU status is preserved, and all registers and RAM retain their data. The status of device pins during power down is the same as with the idle mode and is shown in Table 12. The power-down mode is terminated by a reset signal (including a watchdog timer overflow), or by the occurrence of either of the two external interrupts.

To terminate power-down mode with the external interrupts, the given interrupt must be programmed to be level-sensitive and must be enabled. The interrupt pin must be held low until the master oscillator has restarted and is stabilized.

Table 12. Status of the External Pins During Idle and Power-Down Modes

MODE	MEMORY	ALE	PSEN	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Port data	Port data	Port data	Port data
Idle	External	1	1	Floating	Port data	Address	Port data
Power-down	Internal	0	0	Port data	Port data	Port data	Port data
Power-down	External	0	0	Floating	Port data	Port data	Port data

Philips Components

Document No.	
ECN No.	
Date of Issue	February 1990
Status	Preliminary Specification
Application Specific Product	

80C528/83C528/87C528

CMOS single-chip 8-bit microcontroller

DESCRIPTION

The 8XC528 single-chip 8-bit microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroler family. The 8XC528 has the same instruction set as the 80C51.

This device provides architectural enhancements that make it applicable in a variety of applications in general control systems, especially in those systems which need large ROM and RAM capacity on-chip.

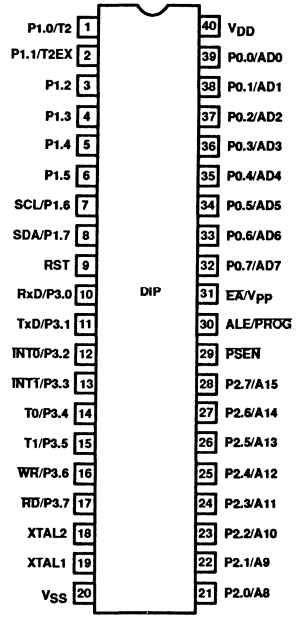
The 8XC528 contains a 32k x 8 ROM (83C528)/EPROM (87C528), a 512 x 8 RAM, four 8-bit I/O ports, two 16-bit timer/event counters (identical to the timers of the 80C51), a 16-bit timer (identical to the timer 2 of the 80C52), a watchdog timer with a separate oscillator, a multi-source, two-priority-level, nested interrupt structure, two serial interfaces (UART and I^2C-bus), and on-chip oscillator and timing circuits.

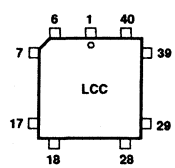
In addition, the 8XC528 has two software selectable modes of power reduction – idle mode and power-down mode. The idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.

FEATURES

- 80C51 instruction set
  - 32k x 8 ROM (83C528)
  - 32k x 8 EPROM (87C528)
  - ROMless (80C528)
  - 512 x 8 RAM
  - Memory addressing capability – 64k ROM and 64k RAM
  - Three 16-bit counter/timers
  - On-chip watchdog timer
  - Full duplex UART
  - I^2C serial interface
- Power control modes:
  - Idle mode
  - Power-down mode
  - Warm start from power-down
- CMOS and TTL compatible
- Three speed ranges at V_{DD} = 5V ±10%
  - 3.5 to 12MHz
  - 3.5 to 16MHz
  - 0.5 to 12MHz
- Extended temperature ranges
- OTP package available
- ROM/EPROM code protection

PIN CONFIGURATION



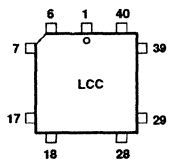


SEE NEXT PAGE FOR LCC PIN FUNCTIONS.

PART NUMBER SELECTION

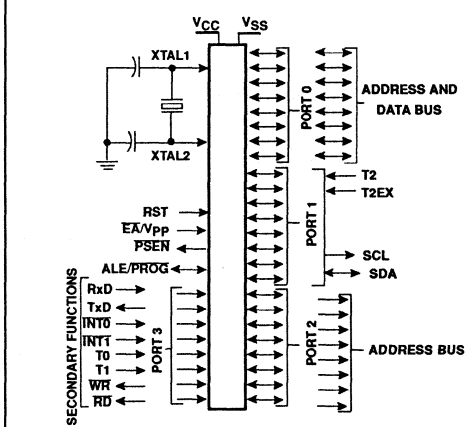
ROMless	ROM	EPROM	TEMPERATURE °C AND PACKAGE	FREQUENCY
		P87C528BBF	0 to +70, ceramic DIP	3.5 to 12MHz
		P87C528EBF	0 to +70, ceramic DIP	3.5 to 16MHz
		P87C528BHF	–40 to +125, ceramic DIP	3.5 to 12MHz
		P87C528EHF	–40 to +125, ceramic DIP	3.5 to 16MHz
P80C528BBP	P83C528BBP	P87C528BBP	0 to +70, plastic DIP	3.5 to 12MHz
P80C528EBP	P83C528EBP	P87C528EBP	0 to +70, plastic DIP	3.5 to 16MHz
P80C528BHP	P83C528BHP	P87C528BHP	–40 to +125, plastic DIP	3.5 to 12MHz
P80C528EHP	P83C528EHP	P87C528EHP	–40 to +125, plastic DIP	3.5 to 16MHz
		P87C528BBK	0 to +70, ceramic LCC	3.5 to 12MHz
		P87C528EBK	0 to +70, ceramic LCC	3.5 to 16MHz
		P87C528BHK	–40 to +125, ceramic LCC	3.5 to 12MHz
		P87C528EHK	–40 to +125, ceramic LCC	3.5 to 16MHz
P80C528BBA	P83C528BBA	P87C528BBA	0 to +70, plastic LCC	3.5 to 12MHz
P80C528EBA	P83C528EBA	P87C528EBA	0 to +70, plastic LCC	3.5 to 16MHz
P80C528BHA	P83C528BHA	P87C528BHA	–40 to +125, plastic LCC	3.5 to 12MHz
P80C528EHA	P83C528EHA	P87C528EHA	–40 to +125, plastic LCC	3.5 to 16MHz

LCC PIN FUNCTIONS

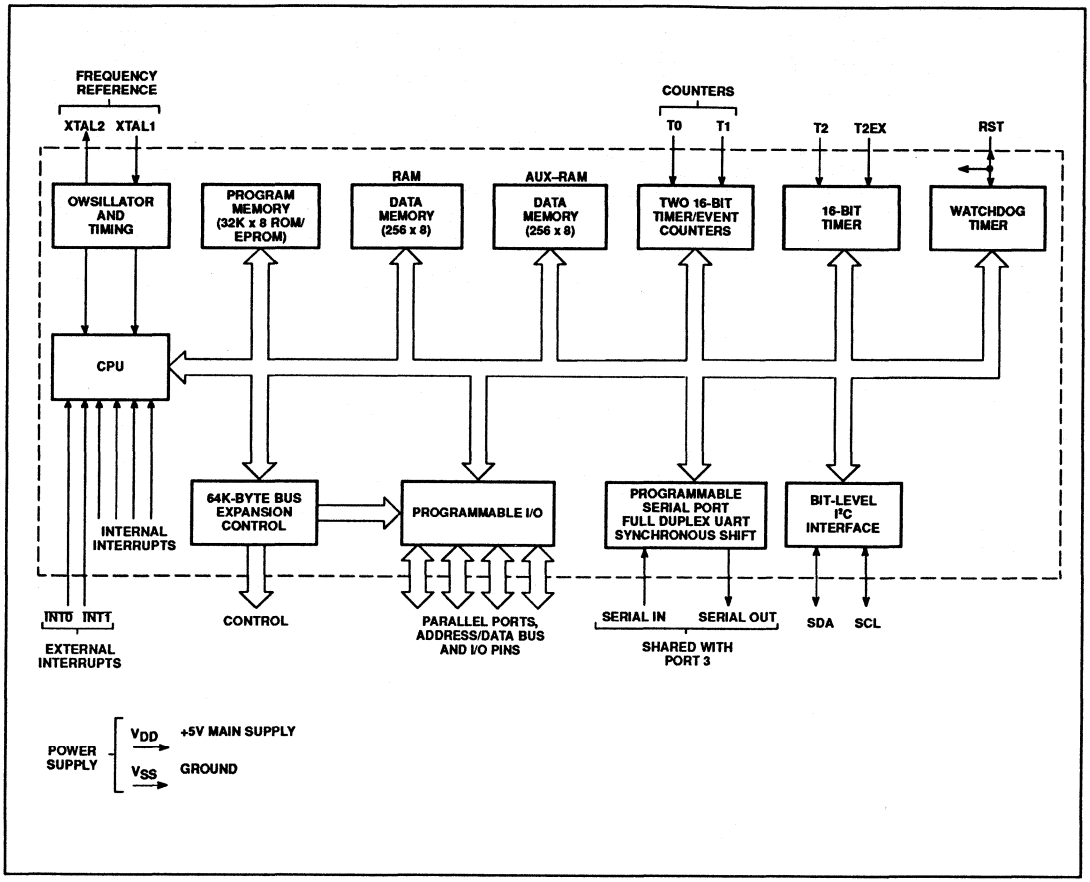


Pin	Function	Pin	Function
1	NC	23	NC
2	P1.0/T2	24	P2.0/A8
3	P1.1/T2EX	25	P2.1/A9
4	P1.2	26	P2.2/A10
5	P1.3	27	P2.3/A11
6	P1.4	28	P2.4/A12
7	P1.5	29	P2.5/A13
8	P1.6/SCL	30	P2.6/A14
9	P1.7/SDA	31	P2.7/A15
10	RST	32	$\overline{PSEN}$
11	P3.0/RxD	33	ALE/$\overline{PROG}$
12	NC	34	NC
13	P3.1/TxD	35	$\overline{EA}/V_{PP}$
14	P3.2/$\overline{INT}$	36	P0.7/AD7
15	P3.3/$\overline{INT}$	37	P0.6/AD6
16	P3.4/T0	38	P0.5/AD5
17	P3.5/T1	39	P0.4/AD4
18	P3.6/$\overline{WR}$	40	P0.3/AD3
19	P3.7/$\overline{RD}$	41	P0.2/AD2
20	XTAL2	42	P0.1/AD1
21	XTAL1	43	P0.0/AD0
22	V_{SS}	44	V_{CC}

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	TYPE	NAME AND FUNCTION
V_{SS}	20	22	I	**Ground:** 0V reference.
V_{CC}	40	44	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–46	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s. Port 0 also outputs the code bytes during program verification in the P87C528. External pull-ups are required during program verification.
P1.0–P1.7	1–8	2–9	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Pins P1.0 and P1.1 also. Port 1 also receives the low-order address byte during program memory verification. Port 1 also serves alternate functions for timer 2:
	1	2	I	**T2 (P1.0):** Timer/counter 2 external count input.
	2	3	I	**T2EX (P1.1):** Timer/counter 2 trigger input.
	7	8	O	**SCL (P1.6):** I^2C serial clock output.
	8	9	I/O	**SDA (P1.7):** I^2C serial data port.
P2.0–P2.7	21–28	24–31	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the SC80C51 family, as listed below:
	10	11	I	**RxD (P3.0):** Serial input port
	11	13	O	**TxD (P3.1):** Serial output port
	12	14	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	I	**T0 (P3.4):** Timer 0 external input
	15	17	I	**T1 (P3.5):** Timer 1 external input
	16	18	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE/$\overline{PROG}$	30	33	I/O	**Address Latch Enable/Program Pulse:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.
$\overline{PSEN}$	29	32	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$/V_{PP}	31	35	I	**External Access Enable/Programming Supply Voltage:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H to 7FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 7FFFH. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
XTAL1	19	21	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	O	**Crystal 2:** Output from the inverting oscillator amplifier.

ROM CODE PROTECTION

By setting a mask programmable security bit, the ROM content in the 83C528 is protected, i.e., it cannot be read out with the help of a test mode or by instructions in the external program memory space. When the security bit is set, a MOVE instruction in external memory will be able to access code in the external memory, but it will not have access to code in the external memory. A MOVE in the external program memory will have access to code in both internal and external memory.

If the security bit has been programmed as a zero, then there are no restrictions for the MOVC instructions. The EA input is latched during RESET and is don't care after RESET. This status prevents the reading of internal program code by switching from external program memory to internal program memory during a MOVC instruction or an instruction that handles immediate data.

INTERNAL DATA MEMORY

The internal data memory is divided into three physically separated segments: 256 bytes of RAM, 256 bytes of AUX-RAM, and a 128 bytes special function area. These can be addressed each in a different way.

- RAM 0 to 127 can be addressed directly and indirectly as in the 80C51. Address pointers are R0 and R1 of the selected register bank.
- RAM 128 to 255 can only be addressed indirectly as in the 80C51. Address pointers are R0 and R1 of the selected register bank.
- AUX-RAM 0 to 255 is indirectly addressed in the same way as external data memory with the MOVX instructions. Address pointers are R0, R1 of the selected register bank and DPTR. An access to AUX-RAM 0 to 255 will not affect ports P0, P2, P3.6 and P3.7.

An access to external data memory locations higher than 255 will be performed with the MOVX DPTR instructions in the same way as in the 8052 structure, so with P0 and P2 as data/address bus and P3.6 and P3.7 as write and read timing signals. Note that these external data memory cannot be accessed with R0 and R1 as address pointer.

TIMER 2

Timer 2 is functionally equal to the Timer 2 of the 8052AH. Timer 2 is a 16-bit timer/counter. These 16 bits are formed by two special function registers TL2 and TH2. Another pair of special function register RCAP2L and RCAP2H form a 16-bit capture register or a 16-bit reload register. Like Timer 0 and 1, it can operate either as a timer or as an event counter. This is selected by bit C/T2N in the special function register T2CON. It has three operating modes: capture, autoload, and baud rate generator mode which are selected by bits in T2CON.

WATCHDOG TIMER T3

The watchdog timer consists of an 11-bit and an 8-bit timer formed by special function register T3. The prescaler is incremented by an on-chip oscillator with a fixed frequency of 1MHz. The maximum tolerance on this frequency is –50% and +100%. The 8-bit timer increments every 2048 cycles of the on-chip oscillator. When a timer overflow occurs, the microcontroller is reset and a reset output pulse of 16 x 2048 cycles of the on-chip oscillator is generated at pin RST. The internal RESET signal is not inhibited when the external RST pin is kept low by, for example, an external reset circuit. The RESET signal drives all port outputs into the high state, independent if the XTAL-clock is running or not.

The watchdog timer is controlled by one special function register WDCON with the direct address location A5H. WDCON can be read and written by software. A value of A5H in WDCON halts the on-chip oscillator and clears both the prescaler and timer T3. After the RESET signal, WDCON contains A5H. Every value other than A5H in WDCON enables the watchdog timer. When the watchdog timer is enabled, it runs independently of the XTAL-clock.

To prevent an overflow of the watchdog timer, the user program has to reload the watchdog timer within periods that are shorter than the programmed watchdog timer internal. This time interval is determined by an 8-bit value that has to be loaded in register T3 while at the same time the prescaler is cleared by hardware.

Watchdog timer interval =

$$\frac{[256 - (T3)] \times 2048}{\text{on-chip oscillator frequency}}$$

BIT-LEVEL I²C INTERFACE

This bit-level serial I/O interface supports the I²C-bus. P1.6/SCL and P1.7/SDA are the serial I/O pins, shared with P1.6 and P1.7.

These two pins met the I²C specification concerning the input levels and output drive capability. Consequently these outputs have an open drain output configuration. All the four modes of the I²C-bus are supported:

- master/transmitter
- master/receiver
- slave/transmitter
- slave/receiver

In all modes, a bit rate of 100Kbit/sec can be achieved @ f_{osc} > 12MHz.

The hardware performs the following functions:

- Generates an interrupt on the reception of a START condition.
- Recognizes a STOP condition and indicates bus-busy or bus-free status.
- Latches a received serial bit.
- Generates a single serial clock pulse to the SCL line.
- Serial clock synchronization (low level stretching of SCL).
- Arbitration loss detection on bit-level.

Three special function registers control the bit-level I²C interface, S1SINT, S1BIT and S1SCS.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol, page 1.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-up reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-up, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

IDLE MODE

In idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. The power-down mode can be terminated by a RESET in the same way as in the 80C51 or in addition by one of two external interrupts, INT0 or INT1. A termination with an external interrupt does not affect the internal data memory and does not affect the special function registers. This makes it possible to exit power-down without changing the port output levels. To terminate the power-down mode with an external interrupt $\overline{INT0}$ or $\overline{INT1}$ must be switched to level-sensitive and must be enabled. The external interrupt input signal $\overline{INT0}$ and $\overline{INT1}$ must be kept low until the oscillator has restarted and stabilized. An instruction following the instruction that puts the device in the power-down mode will be executed. A reset generated by the watchdog timer terminates the power-down mode in the same way as an external RESET, and only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. the control bits for the reduced power modes are in the special function register PCON.

DESIGN CONSIDERATIONS

At power-on, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

When the idle mode is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write when idle is terminated by reset, the instruction following the one that invokes idle should not be one that writes to a port pin or to external memory.

Table 1 shows the state of I/O ports during low current operating modes.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

Electrical Deviations from Commercial Specifications for Extended Temperature Range (87C528)
DC and AC parameters not included here are the same as in the commercial temperature range table.

DC ELECTRICAL CHARACTERISTICS

$T_A = -40°C$ to $+85°C$, $V_{CC} = 5V \pm 10\%$, $V_{SS} = 0V$

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
V_{IL}	Input low voltage, except $\overline{EA}$		–0.5	$0.2V_{CC}$–0.15	V
V_{IL1}	Input low voltage to $\overline{EA}$		0	$0.2V_{CC}$–0.35	V
V_{IH}	Input high voltage, except XTAL1, RST		$0.2V_{CC}$+1	V_{CC}+0.5	V
V_{IH1}	Input high voltage to XTAL1, RST		$0.7V_{CC}$+0.1	V_{CC}+0.5	V
I_{IL}	Logical 0 input current, ports 1, 2, 3	$V_{IN} = 0.45V$		–75	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3	$V_{IN} = 2.0V$		–750	µA
I_{CC}	Power supply current: Active mode Idle mode Power down mode	$V_{CC} = 4.5$–$5.5V$, Frequency range = 3.5 to 12MHz		 35 6 50	 mA mA µA

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Operating temperature under bias	0 to +70 or –40 to +125	°C
Storage temperature range	–65 to +150	°C
Voltage on $\overline{EA}/V_{PP}$ pin to V_{SS}	0 to +13.0	V
Voltage on any other pin to V_{SS}	–0.5 to V_{DD} +6.5	V
Input, output current on any two pins	±10	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.0	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or –40°C to +125°C, V_{CC} = 5V ±10%, V_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			Min	Typical[1]	Max	
V_{IL}	Input low voltage, except $\overline{EA}$[7]		–0.5		0.2V_{CC}–0.1	V
V_{IL1}	Input low voltage to $\overline{EA}$[7]		0		0.2V_{CC}–0.3	V
V_{IH}	Input high voltage, except XTAL1, RST[7]		0.2V_{CC}+0.9		V_{CC}+0.5	V
V_{IH1}	Input high voltage, XTAL1, RST[7]		0.7V_{CC}		V_{CC}+0.5	V
V_{OL}	Output low voltage, ports 1, 2, 3	I_{OL} = 1.6mA[2]			0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$	I_{OL} = 3.2mA[2]			0.45	V
V_{OH}	Output high voltage, ports 1, 2, 3, ALE, $\overline{PSEN}$[3]	I_{OH} = –60µA, I_{OH} = –25µA I_{OH} = –10µA	2.4 0.75V_{CC} 0.9V_{CC}			V V V
V_{OH1}	Output high voltage (port 0 in external bus mode)	I_{OH} = –800µA, I_{OH} = –300µA I_{OH} = –80µA	2.4 0.75V_{CC} 0.9V_{CC}			V V V
I_{IL}	Logical 0 input current, ports 1, 2, 3[7]	V_{IN} = 0.45V			–50	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3[7]	See note 4			–650	µA
I_{LI}	Input leakage current, port 0	V_{IN} = V_{IL} or V_{IH}			±10	µA
I_{CC}	Power supply current:[7] Active mode @ 12MHz[5] Idle mode @ 12MHz Power down mode	See note 6		 11.5 1.3 3	 25 4 50	 mA mA µA
R_{RST}	Internal reset pull-down resistor		50		300	kohm
C_{IO}	Pin capacitance				10	pF

NOTES:
1. Typical ratings are not guaranteed. The values listed are at room temperature, 5V.
2. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows: 10mA per port pin, port 0 total (all bits) 26mA, ports 1, 2, and total each (all bits) 15mA.
3. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the 0.9V_{CC} specification when the address bits are stabilizing.
4. Pins of ports 1, 2 and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
5. I_{CC}MAX at other frequencies is given by: Active mode: I_{CC}MAX = 0.94 X FREQ + 13.71: Idle mode: I_{CC}MAX = 0.14 X FREQ +2.31, where FREQ is the external oscillator frequency in MHz. I_{CC}MAX is given in mA. See Figure 8.
6. See Figures 9 through 12 for I_{CC} test conditions.
7. These values apply only to T_A = 0°C to +70°C. For T_A = –40°C to +85°C, see table on previous page.

AC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or −40°C to +125°C, V_{CC} = 5V ±10%, V_{SS} = 0V[1,2]

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			Min	Max	Min	Max	
$1/t_{CLCL}$	1	Oscillator frequency: **Speed Versions** 8XC528 –1, 2 8XC528 –4, 5			 3.5 3.5	 12 16	 MHz MHz
t_{LHLL}	1	ALE pulse width	127		$2t_{CLCL}-40$		ns
t_{AVLL}	1	Address valid to ALE low	28		$t_{CLCL}-55$		ns
t_{LLAX}	1	Address hold after ALE low	48		$t_{CLCL}-35$		ns
t_{LLIV}	1	ALE low to valid instruction in		234		$4t_{CLCL}-100$	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	43		$t_{CLCL}-40$		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	205		$3t_{CLCL}-45$		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		145		$3t_{CLCL}-105$	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		59		$t_{CLCL}-25$	ns
t_{AVIV}	1	Address to valid instruction in		312		$5t_{CLCL}-105$	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		252		$5t_{CLCL}-165$	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		97		$2t_{CLCL}-70$	ns
t_{LLDV}	2, 3	ALE low to valid data in		517		$8t_{CLCL}-150$	ns
t_{AVDV}	2, 3	Address to valid data in		585		$9t_{CLCL}-165$	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}-130$		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	23		$t_{CLCL}-60$		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	33		$t_{CLCL}-50$		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock							
t_{CHCX}	5	High time	20		20		ns
t_{CLCX}	5	Low time	20		20		ns
t_{CLCH}	5	Rise time		20		20	ns
t_{CHCL}	5	Fall time		20		20	ns
Shift Register							
t_{XLXL}	4	Serial port clock cycle time	1.0		$12t_{CLCL}$		µs
t_{QVXH}	4	Output data setup to clock rising edge	700		$10t_{CLCL}-133$		ns
t_{XHQX}	4	Output data hold after clock rising edge	50		$2t_{CLCL}-117$		ns
t_{XHDX}	4	Input data hold after clock rising edge	0		0		ns
t_{XHDV}	4	Clock rising edge to input data valid		700		$10t_{CLCL}-133$	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

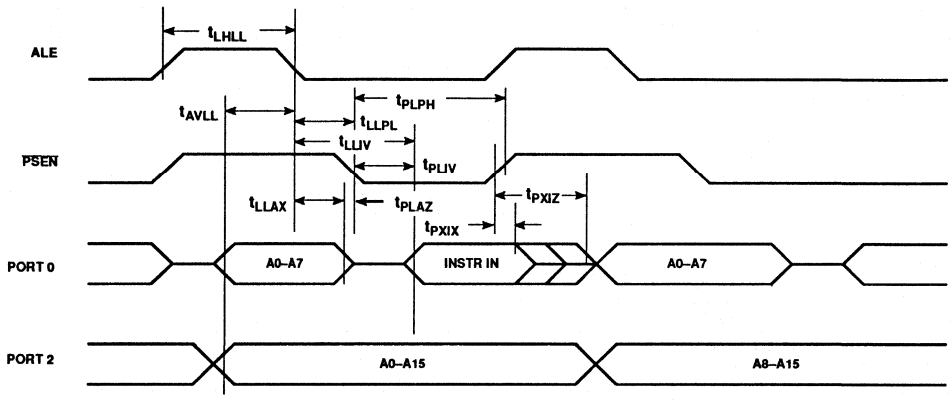


Figure 1. External Program Memory Read Cycle

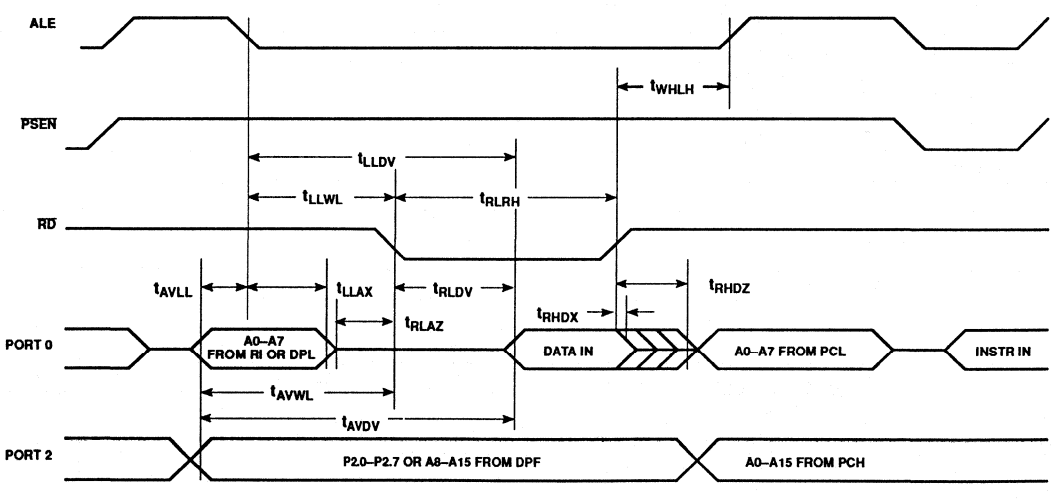


Figure 2. External Data Memory Read Cycle

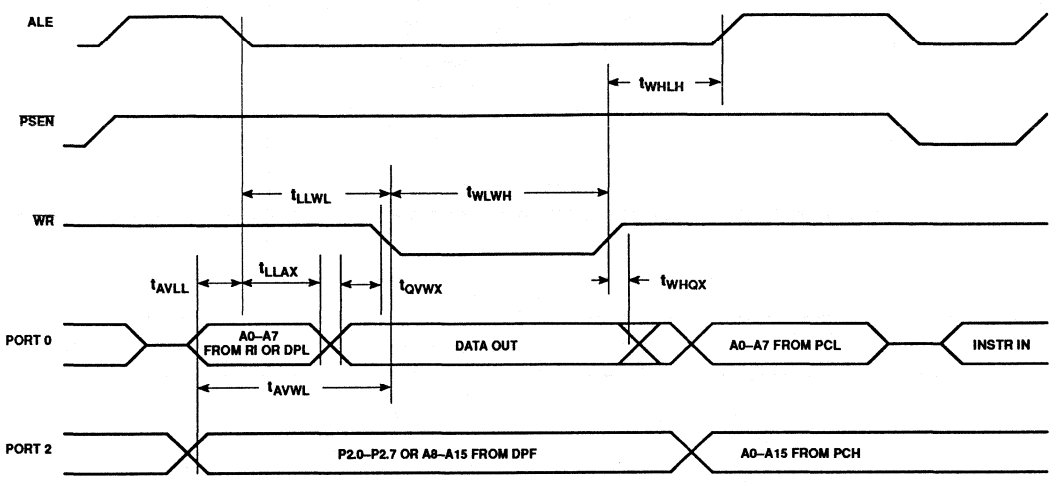


Figure 3. External Data Memory Write Cycle

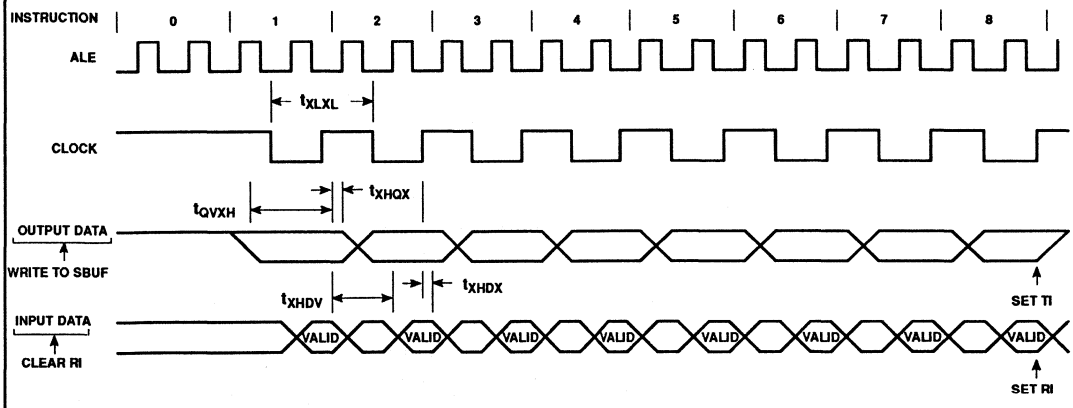


Figure 4. Shift Register Mode Timing

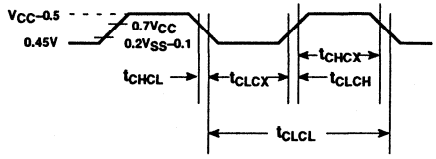
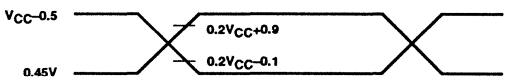
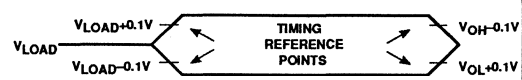


Figure 5. External Clock Drive



NOTE:
AC inputs during testing are driven at V_{CC} –0.5 for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at V_{IH} min for a logic '1' and V_{IL} for a logic '0'.

Figure 6. AC Testing Input/Output



NOTE:
For timing purposes, a port is no longer floating when a 100mV change from load voltage occurs, and begins to float when a 100mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OH}/I_{OL} \geq \pm 20$mA.

Figure 7. Float Waveform

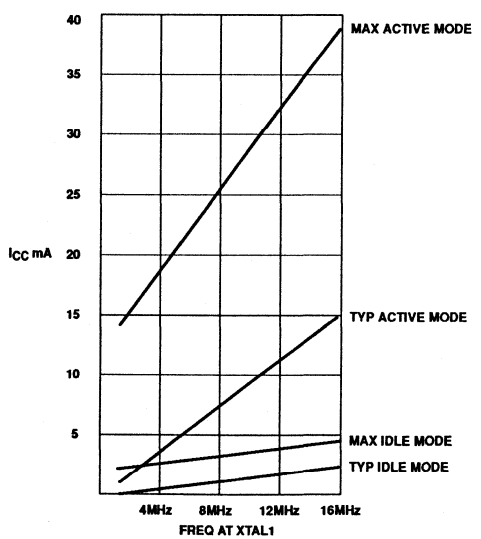


Figure 8. I_{CC} vs. FREQ
Valid only within frequency specifications of the device under test

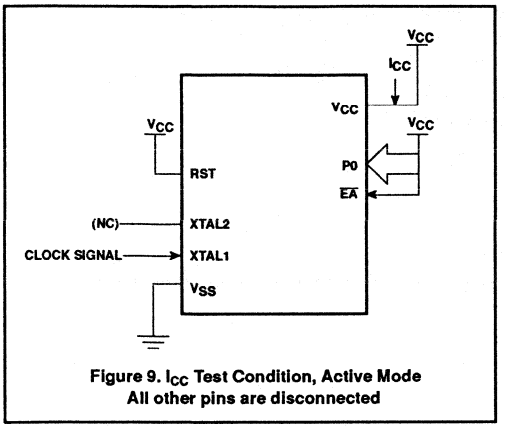


Figure 9. I_{CC} Test Condition, Active Mode
All other pins are disconnected

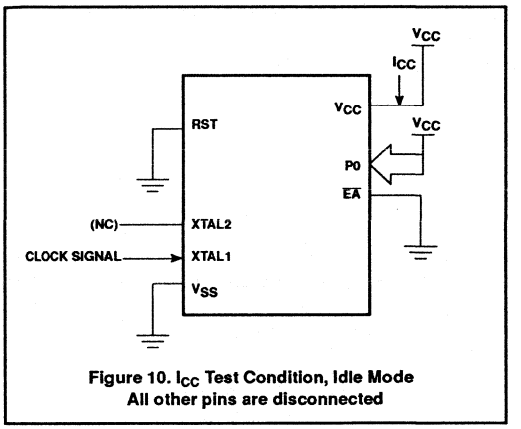


Figure 10. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

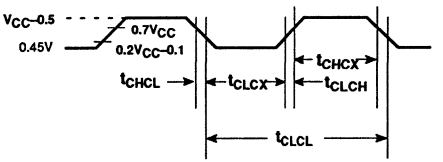


Figure 11. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

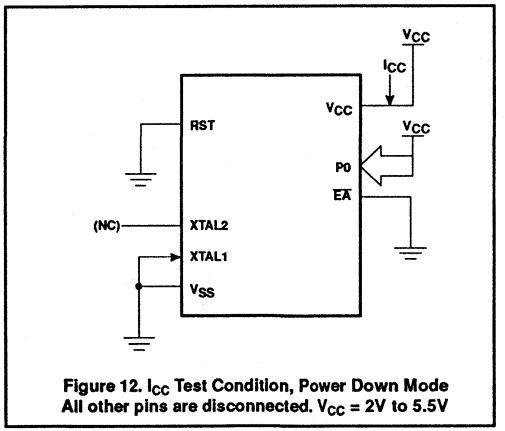


Figure 12. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected. V_{CC} = 2V to 5.5V

EPROM CHARACTERISTICS

The 87C528 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C528 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C528 manufactured by Philips.

Table 2 shows the logic levels for reading the signature byte, and for programming the program memory, the encryption table, and the lock bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 13 and 14. Figure 15 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 13. Note that the 87C528 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1 and 2, as shown in Figure 13. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$ and pins of ports 2 and 3 specified in Table 2 are held at the 'Program Code Data' levels indicated in Table 2. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 14.

To program the encryption table, repeat the 25 pulse programming sequence for addresses 0 through 1FH, using the "Pgm Encryption Table' levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the lock bits, repeat the 25 pulse programming sequence using the 'Pgm Lock Bit' levels. After one lock bit is programmed, further programming of the code memory and encryption table is disabled. However, the other lock bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If lock bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be read is applied to ports 1 and 2 as shown in Figure 15. The other pins are held at the 'Verify Code Data' levels indicated in Table 2. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips

(031H) = 97H indicates 87C528

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 2, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345–5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000uW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

Table 2. EPROM Progamming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm lock bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm lock bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:

1. '0' = Valid low for that pin, '1' = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{CC} = 5V±10% during programming and verification.

*ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

™Trademark phrase of Intel Corporation.

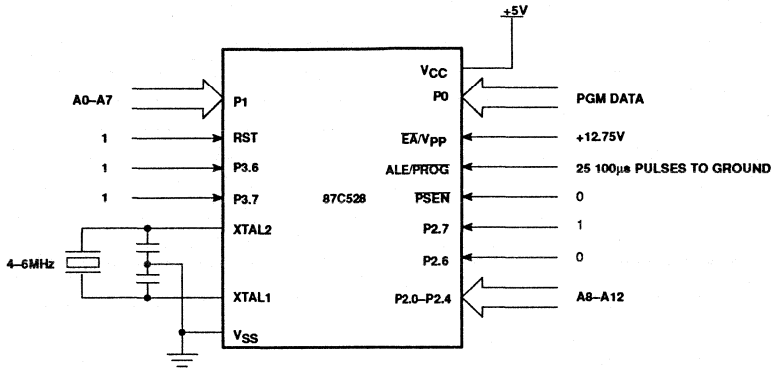


Figure 13. Programming Configuration

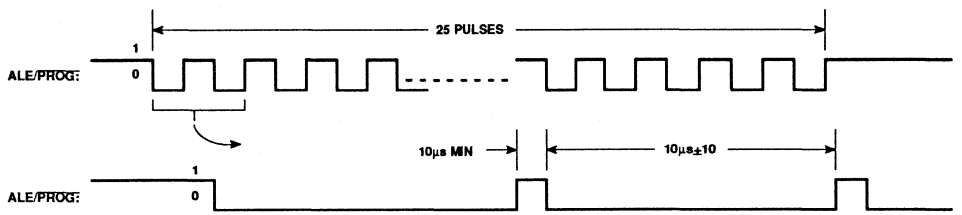


Figure 14. PROG Waveform

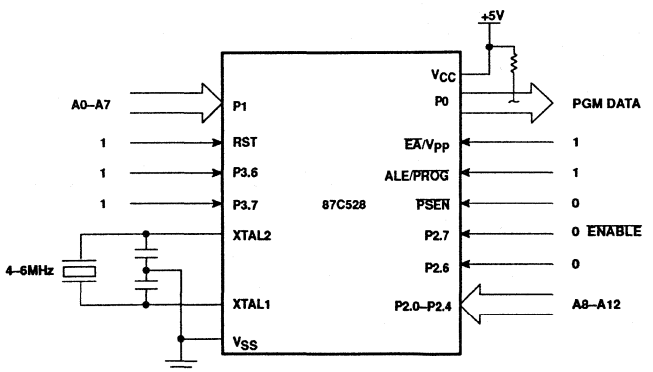


Figure 15. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_A = 21°C to +27°C, V_{CC} = 5V±10%, V_{SS} = 0V (See Figure 16)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		µs
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	$\overline{ENABLE}$ low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		µs

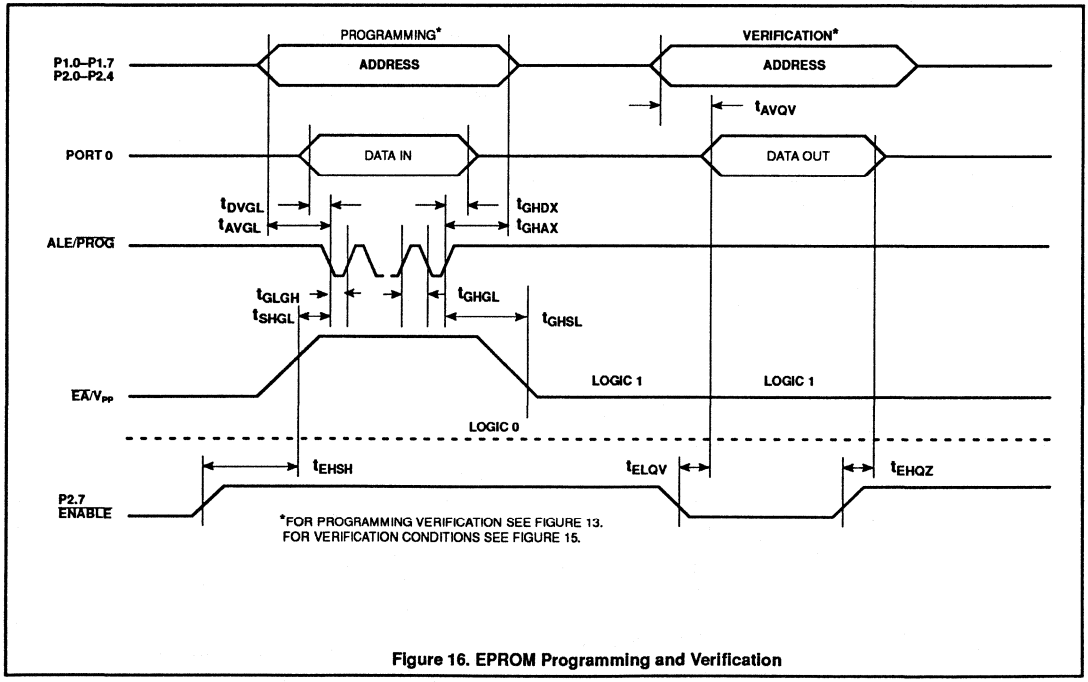


Figure 16. EPROM Programming and Verification

8XC550 OVERVIEW

The Signetics 83C550 is a single-chip control oriented microcontroller in the 80C51 family. The 83C550 has the same basic architecture and instruction set as the industry standard 80C51, with the addition of an eight-channel multiplexed 8-bit A/D converter and a versatile watchdog timer.

The part is available in three versions, collectively referred to as the 83C550: the 83C550 with 4k bytes of masked ROM program memory; the 87C550 with 4k bytes of EPROM program memory; and the 80C550 with no on-chip program memory. The EPROM version of this device is available in both quartz-lid erasable and one-time programmable (OTP) packages. Once the EPROM array of the 87C550 is programmed, the part will generally be functionally equivalent to the masked ROM 83C550. The watchdog timer setup, however, is accomplished in a different manner on the ROM part than it is on the EPROM and ROMless parts. Refer to the description of the watchdog timer for details.

The 83C550 supports two power reduction modes of operation referred to as the idle mode and the power-down mode. The 83C550 features include:

- 80C51 based architecture
- Eight-channel multiplexed 8-bit A/D converter in the LCC package (six channels on the DIP package)
- 40-microsecond A/D conversion time (when used with a 12MHz oscillator)
- Separate A/D power supply and references (LCC part only)
- Watchdog timer
- 4k byte ROM or EPROM program memory
- 128-byte RAM
- 40-pin DIP and 44-pin LCC packages
- 32 I/O lines on the LCC package (30 on the DIP package), port 1 is input only
- Two 16-bit counter/timers
- Two external interrupts
- External memory addressing capability of 64k program memory and 64k data memory
- Full duplex UART
- Low power consumption
  - Idle mode
  - Power-down mode

Differences From the 80C51

Special Function Registers

The 83C550 contains all of the special function registers (SFRs) that are present on the standard 80C51. There are several SFRs that have been added as well as several others that have been modified somewhat in order to accommodate the additional functions of this chip.

Table 13 contains a summary of all of the SFRs and their addresses.

The A/D control register (ADCON) and the A/D data register (ADAT) have been added to allow access to the on-chip A/D converter. ADCON contains all of the control and status bits required to operate the A/D converter.

Four other registers have been added to allow control of the watchdog timer function. The WDCON register controls watchdog timer operation, and WDL holds the watchdog timer reload value. The WFEED1 and WFEED2 registers, when properly manipulated, cause the watchdog timer to be reloaded from WDL. The current watchdog counter value may also be read back from the WDL register address.

I/O Port Structure

The 83C550 has the same I/O ports as the standard 80C51 with the following exceptions: the port 1 pins are shared with the A/D converter inputs and are input only pins even when the A/D is not being used; port 1 has only 6 pins on the DIP version of the part.

A/D Converter

The LCC packaged versions of the 83C550 contain an eight-channel multiplexed 8-bit A/D converter, while the DIP versions implement only six channels. The conversion requires 40 machine cycles (40µs at 12MHz oscillator frequency).

The A/D converter is controlled by the A/D control register, ADCON. Input channels are selected by the analog multiplexer by bits ADCON.0 through ADCON.2. The ADCON register is not bit addressable.

The completion of the 8-bit ADC conversion is flagged by ADCI in the ADCON register, and the result is stored in the special function register ADAT.

An ADC conversion in progress is unaffected by an ADC start. The result of a completed conversion remains unaffected provided ADCI remains at a logic 1. While ADCS is a logic 1 or ADCI is a logic 1, a new ADC START will be blocked and consequently lost. An ADC conversion in progress is aborted when the idle or power-down mode is entered. The result of a completed conversion (ADCI = logic 1) remains unaffected when entering the idle mode. See Figure 70 (8XC752 section 3) for an A/D input equivalent circuit.

The analog input pins ADC0-ADC7 may still be used as digital inputs. The analog input channel that is selected by the ADDR2-ADDR0 bits in ADCON cannot be used as a digital input. Reading the selected A/D channel as a digital input will always return a 1. The unselected A/D inputs may always be used as digital inputs.

Table 13. 83C550 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
ADAT#	A/D result	C6H									xxH
ADCON#	A/D control	C5H	–	–	–	ADCI	ADCS	AADR2	AADR1	AADR0	xxx00000B
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR:	Data pointer (2 bytes):										
DPH	High byte	83H									00H
DPL	Low byte	82H									00H
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*#	Interrupt priority	B8H	–	PWD	PAD	PS	PT1	PX1	PT0	PX0	x0000000B
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*#	Interrupt enable	A8H	EA	EWD	EAD	ES	ET1	EX1	ET0	EX0	00H
P0*	Port 0	80H	87	86	85	84	83	82	81	80	FFH
P1*	Port 1	90H	97	96	95	94	93	92	91	90	FFH
P2*	Port 2	A0H	A7	A6	A5	A4	A3	A2	A1	A0	FFH
P3*	Port 3	B0H	B7	B6	B5	B4	B3	B2	B1	B 0	FFH
PCON#	Power control	87H	SMOD	SIDL	–	–	GF1	GF0	PD	IDL	00xx0000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
SBUF	Serial data buffer	99H									xxH
			9F	9E	9D	9C	9B	9A	99	98	
SCON*	Serial port control	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	00H
TCON*	Timer counter/control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
TMOD	Timer/counter mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H
TH0	Timer 0 high byte	8CH									00H
TH1	Timer 1 high byte	8DH									00H
TL0	Timer 0 low byte	8AH									00H
TL1	Timer 1 low byte	8BH									00H
			C7	C6	C5	C4	C3	C2	C1	C0	
WDCON*#	Watchdog timer control	C0H	PRE2	PRE1	PRE0	–	–	WDRUN	WDTOF	WDMOD	000xx000B**
WDL#	Watchdog timer reload	C1H									FFH**
WFEED1#	Watchdog timer feed 1	C2H									xxH
WFEED2#	Watchdog timer feed 2	C3H									xxH

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.
**This value is not valid for a masked ROM part (83C550) when running from internal memory ($\overline{EA}$ = 1). See data sheet for details.

ADCON Register

MSB							LSB
X	X	X	ADCI	ADCS	AADR2	AADR1	AADR0

ADCI	ADCS	Operation
0	0	ADC not busy, a conversion can be started.
0	1	ADC busy, start of a new conversion is blocked.
1	0	Conversion completed, start of a new is blocked.
1	1	Not possible.

INPUT CHANNEL SELECTION			
ADDR2	ADDR1	ADDR0	INPUT PIN
0	0	0	P1.0
0	0	1	P1.1
0	1	0	P1.2
0	1	1	P1.3
1	0	0	P1.4
1	0	1	P1.5
1	1	0	P1.6*
1	1	1	P1.7*

*Not present on 40-pin DIP versions.

Symbol	Position	Function
ADCI	ADCON.4	ADC interrupt flag. This flag is set when an ADC conversion is complete. If IE.5 = 1, an interrupt is requested when ADCI = 1. The ADCI flag must be cleared by software after A/D data is read, before the next conversion can begin.
ADCS	ADCON.3	ADC start and status. Setting this bit starts an A/D conversion. Once set, ADCS remains high throughout the conversion cycle. On completion of the conversion, it is reset at the same time the ADCI interrupt flag is set. ADCS cannot be reset by software.
AADR2	ADCON.2	Analog input selects.
AADR1	ADCON.1	Binary coded address
AADR0	ADCON.0	selects one of the five analog input port pins of P1 to be input to the converter. It can only be changed when ADCI and ADCS are both low. AADR2 is the most significant bit.

Sample A/D Routines

The following routines demonstrate two methods of operating the A/D converter. The first method uses polling to determine when the A/D conversion is complete. The second method uses the A/D interrupt to flag the end of conversion.

The routine ReadAD will start a read of the A/D channel identified by R7, and wait for the conversion to complete, polling the A/D interrupt flag. The result is returned in the accumulator.

```
ReadAD: MOV  A,#08h        ;Basic A/D
                           ;start
                           ;command.
        ORL  A,R7          ;Add channel
                           ;# to be read.
        MOV  ADCON,A;      ;Start A/D.
ADLoop: MOV  A,ADCON       ;Get A/D
                           ;status.
        JNB  ACC.4,ADLoop;Wait for
                           ;ADCI
                           ;(A/D
                           ;finished).
        MOV  A,ADAT        ;Get
                           ;conversion
                           ;result
        MOV  ADCON,#0      ;Clear ADCI.
        RET
```

The routine StartAD will start a read of the A/D channel identified by R7 and exit back to the calling program. When the conversion is complete, the A/D interrupt occurs, calling the A/D interrupt service routine. The result of the conversion is returned in register R6.

```
StartAD: MOV  A,#08h       ;Basic A/D
                           ;start
                           ;command.
         ORL  A,R7         ;Add channel
                           ;# to be read.
         MOV  ADCON,A      ;Start A/D.
         RET
         .
         .
         .
         ORG  2Bh          ;A/D interrupt
                           ;address.
ADInt:   MOV  R6,ADAT      ;Get
                           ;conversion
                           ;result.
         MOV  ADCON,#0     ;Clear ADCI.
         RETI
```

Watchdog Timer

The purpose of the watchdog timer is to reset the microcontroller within a reasonable amount of time if it enters an erroneous state, possibly due to a programming error, electrical noise, or RFI. When enabled, the watchdog circuit will generate a system reset if the user program fails to "feed" (or reload) the watchdog within a predetermined amount of time.

The watchdog timer implemented on the 83C550 has a programmable interval and can thus be fine tuned to a particular application. If the watchdog function is not used, the timer may still be used as a versatile general purpose timer.

The watchdog function consists of a programmable 13-bit prescaler, and an 8-bit main timer. The main timer is clocked by a tap taken from one of the top 8 bits of the prescaler. The prescaler is incremented once every machine cycle, or 1/12 of the oscillator frequency. Thus, the main counter can be clocked as often as once every 64 machine cycles or as seldom as once every 8192 machine cycles.

When clocked, the main counter decrements. If the main watchdog counter reaches zero, a system reset will occur. To prevent the watchdog timer from under-flowing, the watchdog must be fed before it counts down to zero. When the watchdog is fed, the contents of the WDL register are loaded into the main watchdog counter and the prescaler is cleared.

WDCON Register

MSB							LSB
PRE2	PRE1	PRE0	X	X	WDRUN	WDTOF	WDMOD

Symbol	Position	Function
WDCON.7	PRE2	Prescaler select (read/write).
WDCON.6	PRE1	These bits select the prescaler divide ratio according to the following table:
WDCON.5	PRE0	

PRE2	PRE1	PRE0	DIVISOR (FROM f_{OSC})
0	0	0	12 X 64
0	0	1	12 X 64 X 2
0	1	0	12 X 64 X 4
0	1	1	12 X 64 X 8
1	0	0	12 X 64 X 16
1	0	1	12 X 64 X 32
1	1	0	12 X 64 X 64
1	1	1	12 X 64 X 128

WDCON.4	–	Not used
WDCON.3	–	Not used
WDCON.2	WDRUN	Run control (read/write). This bit turns the timer on (WDRUN = 1) or off (WDRUN = 0) if the timer mode has been selected.
WDCON.1	WDTOF	Timeout flag (read /write). This bit is set when the watchdog timer underflows. It is cleared by an external reset and can be cleared by software.
WDCON.0	WDMOD	Mode selection (read/write). When WDMOD = 1, the watchdog is selected; when WDMOD = 0, the timer is selected. Selecting the watchdog mode automatically disables power-down mode. WDMOD is cleared by external reset. Once the watchdog mode is selected, this bit can only be cleared by writing a 0 to this bit and then performing a feed operation.

A very specific sequence of events must take place to feed the watchdog timer; it cannot be fed accidentally by a runaway program. The following routines demonstrate setting up and feeding the watchdog timer. These routines apply to all versions of the 83C550 except the ROM part when running from internal program memory.

This routine sets up and starts the watchdog timer. This is not necessary for internal ROM operation, because setup of the watchdog timer on masked ROM parts is accomplished directly via ROM mask options.

```
SetWD:  MOV   WDL,#0FFh     ;Set watch-
                            ;dog
                            ;reload value.
        MOV   WDCON,#0E5    ;Set up timer
                            ;prescaler,
                            ;mode, and
                            ;run bits.
        ACALL FeedWD        ;Start watch-
                            ;dog with a
                            ;feed
                            ;operation.
        RET
```

This routine executes a watchdog timer feed operation, causing the timer to reload from WDL. Interrupts must be disabled during this operation due to the fact that the two feed registers must be loaded on consecutive instruction cycles, or a system reset will occur immediately.

```
FeedWD:CLR  EA              ;This
                            ;sequence
                            ;must not be
                            ;interrupted.
       MOV  WFEED1,#0A5h    ;First
                            ;instruction
                            ;of feed
                            ;sequence.
       MOV  WFEED2,#05Ah    ;Second
                            ;instruction
                            ;of feed
                            ;sequence.
       SETB EA              ;Turn
                            ;interrupts
                            ;back on.
       RET
```

An interrupt is available to allow the watchdog timer to be used as a general purpose timer in applications where the watchdog function is not needed. The timer operates in the same manner when used as a general purpose timer except that the timer interrupt is generated on timer underflow instead of a chip reset. Refer to the 87C550 data sheet for additional information on watchdog timer operation.

Interrupts

The 83C550 interrupt structure is a seven-source, two-priority level interrupt system similar to that of the standard 80C51 microcontroller. The interrupt sources are listed below in the order of their internal polling sequence. This is the order in which simultaneous interrupts of the same priority level would be serviced.

Interrupt Priorities

PRIORITY	SOURCE	VECTOR ADDRESS	FUNCTION
Highest	INT0	0003H	External interrupt 0
	TF0	000BH	Counter/timer 0 overflow
	INT1	0013H	External interrupt 1
	TF1	001BH	Counter/timer 1 overflow
	TI & RI	0023H	Serial port transmit/receive
	ADCI	002BH	A/D converter conversion complete
Lowest	WDTOF	0033H	Watchdog timer overflow (only when not in watchdog mode)

Interrupt Control Registers

The standard 80C51 interrupt enable and priority registers have been modified slightly to take into account the additional interrupt sources of the 83C550.

Interrupt Enable Register

MSB							LSB
EA	EWD	EAD	ES	ET1	EX1	ET0	EX0

Symbol	Position	Function
EA	IE.7	Global interrupt enable
EWD	IE.6	Watchdog timer overflow
EAD	IE.5	A/D conversion complete
ES	IE.4	Serial port transmit or receive
ET1	IE.3	Timer 1 overflow
EX1	IE.2	External interrupt 1
ET0	IE.1	Timer 0 overflow
EX0	IE.0	External interrupt 0

Interrupt Priority Register

MSB							LSB
–	PWD	PAD	PS	PT1	PT1	PX0	PX0

Symbol	Position	Function
PWD	IP.6	Watchdog timer
PAD	IP.5	A/D conversion
PS	IP.4	Serial port interrupt
PT1	IP.3	Timer 1 interrupt
PX1	IP.2	External interrupt 1
PT0	IP.1	Timer 0 interrupt
PX0	IP.0	External interrupt 0

Power-Down and Idle Modes

The 83C550 includes the standard 80C51 power-down and idle modes of reduced power consumption. In addition, the 83C550 includes an option to separately turn off the serial port for extra power savings when it is not needed. Also, the individual functional blocks such as the counter/timers are automatically disabled when they are not running. This actually turns off the clocks to the block in question, resulting in additional power savings. Note that when the watchdog timer is operating, the processor is inhibited from entering the power-down mode. This is due to the fact that the oscillator is stopped in the power-down mode, which would effectively turn off the watchdog timer. In keeping with the purpose of the watchdog timer, the processor is prevented from accidentally entering power-down due to some erroneous operation.

Power Control Register

MSB							LSB
SMOD	SIDL	–	–	GF1	GF0	PD	IDL

Symbol	Position	Function
SMOD	PCON.7	Double baud rate bit. When set to a 1 and Timer 1 is used to generate baud rate, and the serial port is used in modes 1, 2, or 3.
SIDL	PCON.6	Separately idles the serial port for additional power savings.
–	PCON.5	Reserved
–	PCON.4	Reserved
GF1	PCON.3	General-purpose flag bit.
GF0	PCON.2	General-purpose flag bit.
PD	PCON.1	Power-down bit. Starting this bit activates power-down operation.
IDL	PCON.0	Idle mode bit. Setting this bit activates idle mode operation.

If 1s are written to PD and IDL at the same time, PD takes precedence.

Document No.	
ECN No.	
Date of Issue	August 1990
Status	Preliminary Specification
Application Specific Product	

80C550/83C550/87C550

CMOS single-chip 8-bit microcontroller with A/D and watchdog timer

DESCRIPTION

The Philips 8XC550 is a high-performance microcontroller fabricated with Philips high-density CMOS technology. This Philips CMOS technology combines the high speed and density characteristics of HMOS with the low power attributes of CMOS. Philips epitaxial substrate minimizes latch-up sensitivity. The CMOS 8XC550 has the same instruction set as the 80C51.

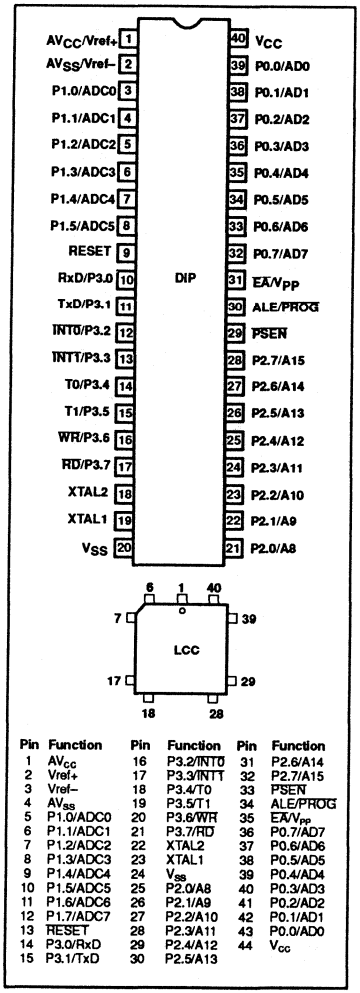
The 8XC550 contains a 4k x 8 EPROM (87C550)/ROM (83C550)/ROMless (80C550 has no program memory on-chip), a 128 x 8 RAM, 8 channels of 8-bit A/D, four 8-bit ports (port 1 is input only), a watchdog timer, two 16-bit counter/timers, a seven-source, two-priority level nested interrupt structure, a serial I/O port for either multi-processor communications, I/O expansion or full duplex UART, and an on-chip oscillator and clock circuits.

In addition, the 8XC550 has two software selectable modes of power reduction – idle mode and power-down mode. The idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.

FEATURES

- 80C51 based architecture
  - 4k x 8 EPROM (87C550)/ ROM (83C550)
  - 128 x 8 RAM
  - 8 channels of 8-bit A/D
  - Two 16-bit counter/timers
  - Watchdog timer
  - Full duplex serial channel
  - Boolean processor
- Memory addressing capability
  - 64k ROM and 64k RAM
- Power control modes:
  - Idle mode
  - Power-down mode
- CMOS and TTL compatible
- Three speed ranges at V_{CC} = 5V ±10%
  - 3.5 to 12MHz
  - 3.5 to 16MHz
- Four package styles
- Extended temperature ranges
- OTP package available

PIN CONFIGURATION



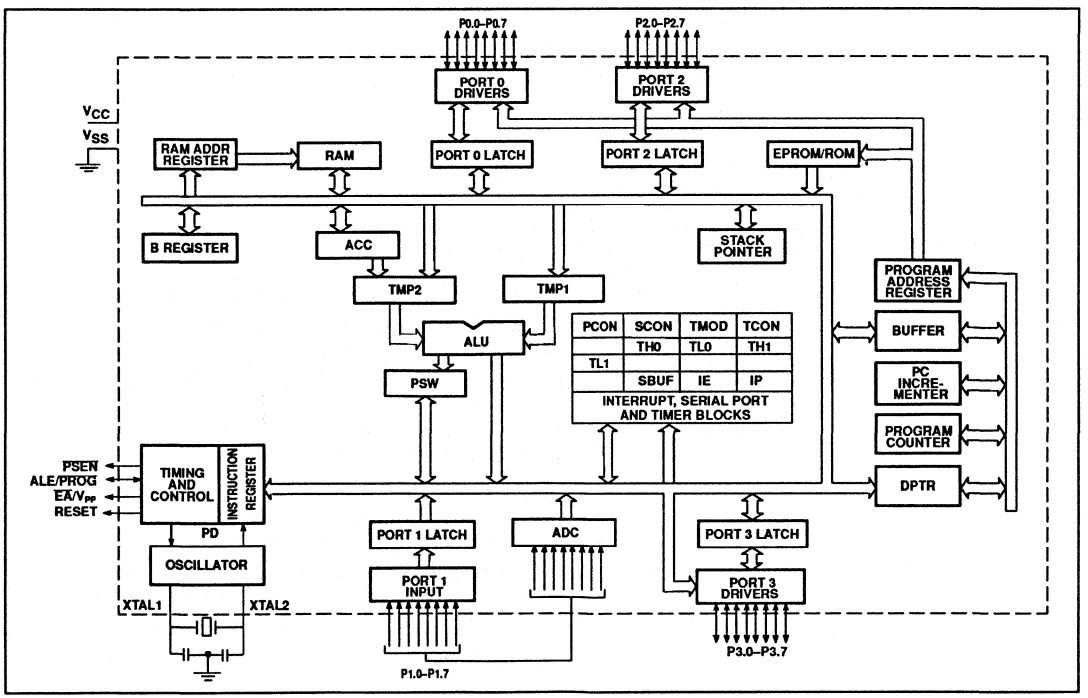
Pin	Function	Pin	Function	Pin	Function
1	AV_{CC}	16	P3.2/INT0	31	P2.6/A14
2	Vref+	17	P3.3/INT1	32	P2.7/A15
3	Vref–	18	P3.4/T0	33	PSEN
4	AV_{SS}	19	P3.5/T1	34	ALE/PROG
5	P1.0/ADC0	20	P3.6/WR	35	EA/V_{PP}
6	P1.1/ADC1	21	P3.7/RD	36	P0.7/AD7
7	P1.2/ADC2	22	XTAL2	37	P0.6/AD6
8	P1.3/ADC3	23	XTAL1	38	P0.5/AD5
9	P1.4/ADC4	24	V_{SS}	39	P0.4/AD4
10	P1.5/ADC5	25	P2.0/A8	40	P0.3/AD3
11	P1.6/ADC6	26	P2.1/A9	41	P0.2/AD2
12	P1.7/ADC7	27	P2.2/A10	42	P0.1/AD1
13	RESET	28	P2.3/A11	43	P0.0/AD0
14	P3.0/RxD	29	P2.4/A12	44	V_{CC}
15	P3.1/TxD	30	P2.5/A13		

CMOS single-chip 8-bit microcontroller 80C550/83C550/87C550

PART NUMBER SELECTION

ROMless	ROM	EPROM	TEMPERATURE AND PACKAGE	FREQUENCY
P80C550BBF	P83C550BBF	P87C550BBF	0 to +70°C, ceramic DIP	3.5 to 12MHz
P80C550EBF	P83C550EBF	P87C550EBF	0 to +70°C, ceramic DIP	3.5 to 16MHz
P80C550BBK	P83C550BBK	P87C550BBK	0 to +70°C, ceramic LCC	3.5 to 12MHz
P80C550EBK	P83C550EBK	P87C550EBK	0 to +70°C, ceramic LCC	0.5 to 16MHz
P80C550BBP	P83C550BBP	P87C550BBP	0 to +70°C, plastic DIP	3.5 to 12MHz
P80C550EBP	P83C550EBP	P87C550EBP	0 to +70°C, plastic DIP	3.5 to 16MHz
P80C550BBA	P83C550BBA	P87C550BBA	0 to +70°C, plastic LCC	3.5 to 12MHz
P80C550EBA	P83C550EBA	P87C550EBA	0 to +70°C, plastic LCC	3.5 to 16MHz
P80C550BFP	P83C550BFP	P87C550BFP	–40 to +85°C, plastic DIP	3.5 to 12MHz
P80C550EFP	P83C550EFP	P87C550EFP	–40 to +85°C, plastic DIP	3.5 to 16MHz
P80C550BFA	P83C550BFA	P87C550BFA	–40 to +85°C, plastic LCC	3.5 to 12MHz
P80C550EFA	P83C550EFA	P87C550EFA	–40 to +85°C, plastic LCC	3.5 to 16MHz
P80C550BFF	P83C550BFF	P87C550BFF	–40 to +85°C, ceramic DIP	3.5 to 12MHz
P80C550EFF	P83C550EFF	P87C550EFF	–40 to +85°C, ceramic LCC	3.5 to 12MHz
P80C550BFK	P83C550BFK	P87C550BFK	–40 to +85°C, ceramic LCC	3.5 to 16MHz
P80C550EFK	P83C550EFK	P87C550EFK	–40 to +85°C, ceramic DIP	3.5 to 16MHz

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	TYPE	NAME AND FUNCTION
V_{SS}	20	24	I	**Ground:** 0V reference.
V_{CC}	40	44	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
AV_{CC}	1	1	I	**Analog Power Supply:** Analog supply voltage.
AV_{SS}	2	4	I	**Analog Ground:** Analog 0V reference.
Vref+		2	I	**Vref:** A/D converter reference level inputs. Note that these references are combined with AV_{CC} and AV_{SS} in the 40-pin DIP package.
Vref–		3	I	
P0.0–0.7	39–32	43–36	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s. Port 0 also outputs the code bytes during program verification in the S87C550. External pull-ups are required during program verification.
P1.0–P1.7	3–8	5–12	I	**Port 1:** Port 1 is an 8-bit input only port (6-bit in the DIP package; bits P1.6 and P1.7 are not implemented). Port 1 digital input can be read out any time.
ADC0–ADC7	3–8	5–12		**ADCx:** Inputs to the analog multiplexer input of the 8-bit A/D. There are only six A/D inputs in the DIP package.
P2.0–P2.7	21–28	25–32	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	14–21	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the SC80C51 family, as listed below:
	10	14	I	**RxD (P3.0):** Serial input port
	11	15	O	**TxD (P3.1):** Serial output port
	12	16	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	17	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	18	I	**T0 (P3.4):** Timer 0 external input
	15	19	I	**T1 (P3.5):** Timer 1 external input
	16	20	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	21	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	13	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE/$\overline{PROG}$	30	34	I/O	**Address Latch Enable/Program Pulse:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.
$\overline{PSEN}$	29	33	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$/V_{PP}	31	35	I	**External Access Enable/Programming Supply Voltage:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H to 1FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 1FFFH. For the 80C550 ROMless part, $\overline{EA}$ must be held low for the part to operate properly. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
XTAL1	19	23	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	22	O	**Crystal 2:** Output from the inverting oscillator amplifier.

Table 1. Interrupt Priorities

Priority	Source	Vector Address	Function
Highest	$\overline{INT0}$	0003H	External interrupt 0
	T0	000BH	Timer flag 0
	$\overline{INT1}$	0013H	External interrupt 1
	T1	001BH	Timer flag 1
	SIO	0023H	Serial port interrupt
	ADC	002BH	A/D conversion complete
Lowest	WD	0033H	Watchdog timer

INTERRUPTS

The interrupt structure is a seven source, two level interrupt system similar to that of the 80C51. The interrupt sources are listed in Table 1 in order of polling sequence priority (highest to lowest). Note that the watchdog timer function is available only if the watchdog timer function is disabled and the watchdog timer is used as a general purpose timer.

Interrupt Control Registers

The interrupt enable and the interrupt priority registers have been modified to take into account the different interrupt sources of the 8XC550. In all other respects, their operation is identical to that of the 80C51. Setting a bit in the IE register enables the interrupt; clearing the bit disables the interrupt. All bits are cleared by reset. See Figure 1 for interrupt register formats.

SERIAL COMMUNICATIONS

The serial port operation is identical to that of the 80C51. In order to conserve power, another bit (SIDL) has been added to the PCON register that idles the serial port when it is not being used. This bit is cleared by reset. See Figure 2.

A/D CONVERTER

The analog input circuitry consists of an 8-input analog multiplexer and an analog-to-digital converter with 8-bit resolution. In the LCC package, the analog reference voltage and analog power supplies are connected via separate input pins; in the DIP package, Vref+ is combined with AV_{CC} and Vref– is combined with AV_{SS}. The analog inputs are alternate functions to port 1, which is an input only port. Digital input to port 1 can be read any time during an A/D conversion. Care should be exercised in mixing analog and digital signals on port 1, because cross talk from the digital input signals can degrade the A/D conversion accuracy of the analog input. An A/D conversion requires 40 machine cycles.

The A/D converter is controlled by the ADCON special function register. The input channel to be converted is selected by the analog multiplexer by setting ADCON register bits, ADDR2–ADDR0 (see Figure 3). These bits can only be changed when ADCI and ADCS are both low.

The completion of the 8-bit ADC conversion is flagged by ADCI in the ADCON register and the result is stored in the special function register ADAT. The functions of the ADCI and ADCS are:

ADCI	ADCS	Operation
0	0	ADC not busy. A conversion can be started.
0	1	ADC busy. Start of a new conversion is blocked.
1	0	Conversion completed. start of new conversion is blocked.
1	1	Not possible.

An ADC conversion in progress is unaffected by a software ADC start. The result of a completed conversion remains unaffected provided ADCI remains at a logic 1. While ADCS is a logic 1 or ADCI is a logic 1, a new ADC START will be blocked and consequently lost. An A/D conversion in progress will be aborted when the idle or power-down mode is entered. The result of a completed conversion (ADCI = logic 1) remains unaffected when entering the idle mode, but will be lost if power-down mode is entered.

MSB INTERRUPT ENABLE REGISTER (IE) LSB

EA	EWD	EAD	ES	ET1	EX1	ET0	EX0

BIT	SYMBOL	FUNCTION
IE.7	EA	GLOBAL INTERRUPT DISABLE WHEN EA = 0
IE.6	EWD	WATCHDOG TIMER OVERFLOW
IE.5	EAD	A/D CONVERSION COMPLETE
IE.4	ES	SIO INTERRUPT
IE.3	ET1	TIER 1 OVERFLOW
IE.2	EX1	EXTERNAL INTERRUPT 1
IE.1	ET0	TIMER 0 OVERFLOW
IE.0	EX0	EXTERNAL INTERRUPT 0

MSB INTERRUPT PRIORITY REGISTER (IP) LSB

X	PWD	PAD	PS	PT1	PX1	PT0	PX0

BIT	SYMBOL	FUNCTION
IP.6	PWD	WATCHDOG TIMER
IP.5	PAD	A/D CONVERTER
IP.4	PS	SIO INTERRUPT
IP.3	PT1	T1 INTERRUPT
IP.2	PX1	$\overline{INT1}$ INTERRUPT
IP.1	PT0	T0 INTERRUPT
IP.0	PX0	$\overline{INT0}$ INTERRUPT

NOTE:
The interrupt priority register function is identical to that of the 80C51. A '1' sets that interrupt to the high priority level and a '0' sets the interrupt to the low priority level.

Figure 1. Interrupt Registers

MSB							LSB
SMOD	SIDL	X	X	GF1	GF0	PD	IDL

BIT	SYMBOL	FUNCTION
PCON.7	SMOD	DOUBLE BAUD RATE
PCON.7	SIDL	SERIAL PORT IDLE
PCON.5	X	RESERVED FOR FUTURE USE
PCON.4	X	RESERVED FOR FUTURE USE
PCON.3	GF1	GENERAL PURPOSE FLAG BIT
PCON.2	GF0	GENERAL PURPOSE FLAG BIT
PCON.1	PD	POWER DOWN BIT
PCON.0	IDL	IDLE MODE BIT

NOTE:
The PCON register is at SFR byte address 87H. Its contents following a reset are 00XX0000.

Figure 2. Power Control Register (PCON)

MSB							LSB
X	X	X	ADCI	ADCS	AADR2	AADR1	AADR0

INPUT CHANNEL SELECTION

ADDR2	ADDR1	ADDR0	INPUT PIN
0	0	0	ADC0
0	0	1	ADC1
0	1	0	ADC2
0	1	1	ADC3
1	0	0	ADC4
1	0	1	ADC5
1	1	0	ADC6
1	1	1	ADC7

BIT	SYMBOL	FUNCTION
ADCON.7	–	NOT USED
ADCON.6	–	NOT USED
ADCON.5	–	NOT USED
ADCON.4	ADCI	ADC INTERRUPT FLAG. This flag is set when an ADC conversion result is ready to be read. An interrupt is invoked if the A/D interrupt is enabled. The flag must be cleared by software. It cannot be set by software.
ADCON.3	ADCS	ADC START AND STATUS. Setting this flag starts an A/D conversion. The ADC logic insures that this signal is high while the ADC is busy. On completion of the conversion, ADCS is reset at the same time the interrupt flag ADCI is set. ADCS cannot be reset by software.
ADCON.2	ADDR2	ANALOG INPUT SELECT 2
ADCON.1	ADDR1	ANALOG INPUT SELECT 1
ADCON.0	ADDR0	ANALOG INPUT SELECT 0

Figure 3. A/D Control Register (ADCON)

WATCHDOG TIMER

The watchdog timer is not directly loadable by the user. Instead, the value to be loaded into the main timer is held in an autoload register or is part of the mask ROM programming. In order to cause the main timer to be loaded with the appropriate value, a special sequence of software action must take place. This operation is referred to as feeding the watchdog timer.

To feed the watchdog, two instructions must be sequentially executed successfully. No intervening instruction fetches are allowed, so interrupts should be disabled before feeding the watchdog. The instructions should move A5H to the WFEED1 register and then 5AH to the WFEED2 register. If WFEED1 is correctly loaded and WFEED2 is not correctly loaded, then an immediate underflow will occur.

The watchdog timer subsystem has two modes of operation. Its principal function is a watchdog timer. In this mode it protects the system from incorrect code execution by causing a system reset when the watchdog timer underflows as a result of a failure of software to feed the timer prior to the timer reaching its terminal count. If the user does not employ the watchdog function, the watchdog subsystem can be used as a timer. In this mode, reaching the terminal count sets a flag which can be used to generate an interrupt. In most other respects, the timer mode possesses the characteristics of the watchdog mode. This is done to protect the integrity of the watchdog function.

The watchdog timer subsystem consists of a prescaler and a main counter. The prescaler has 8 selectable taps off the final stages and the output of a selected tap provides the clock to the main counter. The main counter is the section that is loaded as a result of the software feeding the watchdog and it is the section that causes the system reset (watchdog mode) or time-out flag to be set (timer mode) if allowed to reach its terminal count.

Programming the Watchdog Timer

Both the EPROM and ROM devices have a set of SFRs for holding the watchdog autoload values and the control bits. The watchdog time-out flag is present in the watchdog control register and operates the same in all versions. In the EPROM device, the watchdog parameters (autoload value and control) are always taken from the SFRs. In the ROM device, the watchdog parameters can be mask programmed or taken from the SFRs. The selection to take the watchdog parameters from the SFRs or from the mask programmed values is controlled by EA (external access). When EA is high (internal ROM access), the watchdog parameters are taken from the mask programmed values. If the watchdog is masked programmed to the timer mode, then the autoload values and the pre-scaler taps are taken from the SFRs. When EA is low (external access), the watchdog parameters are taken from the SFRs. The user should be able to leave code in his program which initializes the watchdog SFRs even though he has migrated to the mask ROM part. This allows no code changes from EPROM prototyping to ROM coded production parts.

Watchdog Detailed Operation

EPROM Device (and ROMless Operation: EA = 0)

In the ROMless operation (ROM part, EA = 0) and in the EPROM device, the watchdog operates in the following manner.

Whether the watchdog is in the watchdog or timer mode, when external RESET is applied, the following takes place:

- Watchdog mode bit set to timer mode.
- Watchdog run control bit set to OFF.
- Autoload register set to FF (max count).
- Watchdog time-out flag cleared.
- Prescaler is cleared.
- Prescaler tap set to the highest divide.
- Autoload takes place.

The watchdog can be fed even though it is in the timer mode.

Note that the operational concept is for the watchdog mode of operation, when coming out of a hardware reset, the software should load the autoload registers, set the mode to watchdog, and then feed the watchdog (cause an autoload). The watchdog will now be starting at a known point.

If the watchdog is in the watchdog mode and running and happens to underflow at the time the external RESET is applied, the watchdog time-out flag will be cleared.

When the watchdog is in the watchdog mode and the watchdog underflows, the following action takes place:

- Autoload takes place.
- Watchdog time-out flag is set
- Timer mode interrupt flag unchanged.
- Mode bit unchanged.
- Watchdog run bit unchanged.
- Autoload register unchanged.
- Prescaler tap unchanged.
- All other device action same as external reset.

Note that if the watchdog underflows, the program counter will start from 00H as in the case of an external reset. The watchdog time-out flag can be examined to determine if the watchdog has caused the reset condition. The watchdog time-out flag bit can be cleared by software.

When the watchdog is in the timer mode and the timer software underflows, the following action takes place:

- Autoload takes place.
- Watchdog time-out flag is set
- Mode bit unchanged.
- Watchdog run bit unchanged.
- Autoload register unchanged.
- Prescaler tap unchanged.

The timer mode interrupt flag is cleared when the interrupt routine is invoked. This bit can also be cleared directly by software without a software feed operation.

Mask ROM Device (EA = 1)
In the mask ROM device, the watchdog mode bit (WDMOD) is mask programmed and the bit in the watchdog command register is read only and reflects the mask programmed selection. If the mask programmed mode bit selects the timer mode, then the watchdog run bit (WDRUN) operates as described under EPROM Device. If the mask programmed bit selects the watchdog mode, then the watchdog run bit has no effect on the timer operation.

Watchdog Function
The watchdog consists of a programmable prescaler and the main timer. The prescaler derives its clock from the on-chip oscillator.

The prescaler consists of a divide by 12 followed by a 13 stage counter with taps from stage 6 through stage 13. The tap selection is programmable. The watchdog main counter is a down counter clocked (decremented) each time the programmable prescaler underflows. The watchdog generates an underflow signal (and is autoloaded) when the watchdog is at count 0 and the clock to decrement the watchdog occurs. The watchdog is 8 bits long and the autoload value can range from 0 to FFH. (The autoload value of 0 is permissible since the prescaler is cleared upon autoload).

This leads to the following user design equations. Definitions :t_{OSC} is the oscillator period, N is the selected prescaler tap value, W is the main counter autoload value, t_{MIN} is the minimum watchdog time-out value (when the autoload value is 0), t_{MAX} is the maximum time-out value (when the autoload value is FFH), t_D is the design time-out value.

$t_{MIN} = t_{OSC} \times 12 \times 64$

$t_{MAX} = t_{MIN} \times 128 \times 256$

$t_D = t_{MIN} \times 2^{PRESCALER} \times W$
(where prescaler = 0, 1, 2, 3, 4, 5, 6, or 7)

Note that the design procedure is anticipated to be as follows. A t_{MAX} will be chosen either from equipment or operation considerations and will most likely be the next convenient value higher than t_D. (If the watchdog were inadvertently to start from FFH, an overflow would be guaranteed, barring other anomalies, to occur within t_{MAX}). Then the value for the prescaler would be chosen from:

$prescaler = \log2 (t_{MAX} / (t_{OSC} \times 12 \times 256)) - 6$

This then also fixes t_{MIN}. An autoload value would then be chosen from:

$W = t_D / t_{MIN} - 1$

The software must be written so that a feed operation takes place every t_D seconds from the last feed operation. Some tradeoffs may need to be made. It is not advisable to include feed operations in minor loops or in subroutines unless the feed operation is a specific subroutine.

Watchdog Control Register (WDCON) (Bit Addressable) Address C0
The following bits of this register are read only in the ROM part when EA is high: WDMOD, PRE0, PRE1, and PRE2. That is, the register will reflect the mask programmed values. In the ROM part with EA high, these bits are taken from mask coded bits and are not readable by the program. WDRUN is read only in the ROM part when EA is high and MDMOD is in the watchdog mode. When WDMOD is in the timer mode, WDRUN functions normally (see Figure 4).

The parameters written into WDMOD, PRE0, PRE1, and PRE2 by the program are not applied directly to the watchdog timer subsystem. The watchdog timer subsystem is directly controlled by a second register which stores these bits. The transfer of these bits from the user register (WDMOD) to the second control register takes place when the watchdog is fed. This prevents random code execution from directly foiling the watchdog function. This does not affect the operation where these bits are taken from mask coded values.

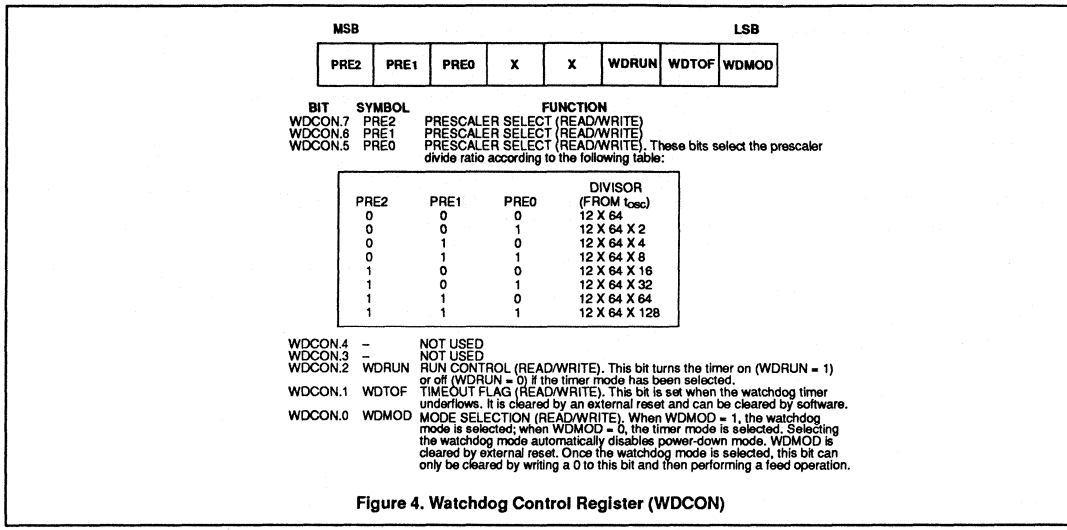
OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol, page 1.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

IDLE MODE

In idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. the instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. An A/D conversion in progress will be aborted when idle mode is entered. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.



MSB							LSB
PRE2	PRE1	PRE0	X	X	WDRUN	WDTOF	WDMOD

BIT	SYMBOL	FUNCTION
WDCON.7	PRE2	PRESCALER SELECT (READ/WRITE)
WDCON.6	PRE1	PRESCALER SELECT (READ/WRITE)
WDCON.5	PRE0	PRESCALER SELECT (READ/WRITE). These bits select the prescaler divide ratio according to the following table:

PRE2	PRE1	PRE0	DIVISOR (FROM t_{osc})
0	0	0	12 X 64
0	0	1	12 X 64 X 2
0	1	0	12 X 64 X 4
0	1	1	12 X 64 X 8
1	0	0	12 X 64 X 16
1	0	1	12 X 64 X 32
1	1	0	12 X 64 X 64
1	1	1	12 X 64 X 128

WDCON.4	–	NOT USED
WDCON.3	–	NOT USED
WDCON.2	WDRUN	RUN CONTROL (READ/WRITE). This bit turns the timer on (WDRUN = 1) or off (WDRUN = 0) if the timer mode has been selected.
WDCON.1	WDTOF	TIMEOUT FLAG (READ/WRITE). This bit is set when the watchdog timer underflows. It is cleared by an external reset and can be cleared by software.
WDCON.0	WDMOD	MODE SELECTION (READ/WRITE). When WDMOD = 1, the watchdog mode is selected; when WDMOD = 0, the timer mode is selected. Selecting the watchdog mode automatically disables power-down mode. WDMOD is cleared by external reset. Once the watchdog mode is selected, this bit can only be cleared by writing a 0 to this bit and then performing a feed operation.

Figure 4. Watchdog Control Register (WDCON)

Programmable Idle Modes

The programmable idle modes have been dispersed throughout the functional blocks. Each block has its own ability to be disabled. For example, if timer 0 is not commanded to be running (TR = 0), then the clock to the timer is disabled resulting in an idle mode power saving. An additional idle control bit has been added to the serial communications port.

A/D Operation in Idle Mode

When in the idle mode, the A/D converter will be disabled. However, the current through the Vref pins will be present and will not be reduced internally in either the idle or the power-down modes. It is the responsibility of the user to disconnect Vref to reduce power supply current.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. The power-down mode can be terminated by a Reset in the same way as in the 80C51 or in addition by one of two external interrupts, INT0 or INT1. A termination with an external interrupt does not affect the internal data memory and does not affect the special function registers. This makes it possible to exit power-down without changing the port output levels. To terminate the power-down mode with an external interrupt, INT0 or INT1 must be switched to level-sensitive and must be enabled. The external interrupt input signal INT0 and INT1 must be kept low until the oscillator has restarted and stabilized. An instruction following the instruction that puts the device in the power-down mode will be executed. A reset generated by the watchdog timer terminates the power-down mode in the same way as an external Reset, and only the contents of the on-chip RAM are preserved.

The control bits for the reduced power modes are in the special function register PCON.

DESIGN CONSIDERATIONS

At power-on, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

When the idle mode is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write when idle is terminated by reset, the instruction following the one that invokes idle should not be one that writes to a port pin or to external memory. Table 2 shows the state of I/O ports during low current operating modes.

Table 2. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

Encryption Table

The encryption table is a feature of the 83C550 and 87C550 that protects the code from being easily read by anyone other than the programmer. The encryption table is 32 bytes of code that are exclusive NORed with the program code data as it is read out. The first byte is XNORed with the first location read, the second with the second read, etc.

After the encryption table has been programmed, the user has to know its contents in order to correctly decode the program code data. The encryption table itself cannot be read out.

For the EPROM (87C550) part, the encryption table is programmed in the same manner as the program memory, but using the "Pgm Encryption Table" levels specified in Table 4. After the encryption table is programmed, verification cycles will produce only encrypted information.

For the ROM part (83C550) the encryption table information is submitted with the ROM code as shown in Table 3.

Security Bits

There are two security bits on the 83C550 and 87C550 that, when set, prevent the program data memory from being read out or programmed further.

After the first security bit is programmer, the external MOVC instruction is disabled, and for the 87C550, further programming of the code memory or the encryption table is disabled. The other security bit can of course still be programmed. With only security bit one programmed, the memory can still be read out for program verification. After the second security bit is programmed, it is no longer possible to read out (verify) the program memory.

To program the security bits for the 87C550, repeat the programming sequence using the "Pgm Lock Bit" levels specified in Table 4. For the masked ROM 83C550 the security bit information is submitted with the ROM code as shown in Table 3.

ROM Code Submission

When submitting a ROM code for the 83C550, the following must be specified:

1. The 4k byte user ROM program.
2. The 32 byte ROM encryption key.
3. The ROM security bits.
4. The watchdog timer parameters.

This information can be submitted in an EPROM (2764) or hex file with the format specified in Table 3.

Table 3. ROM Code Submittal Requirements

ADDRESS	CONTENT	BIT(s)	COMMENT
0000H to 0FFFH	Data	7:0	User ROM data
1000H to 101FH	Key	7:0	ROM encryption key; FFH = no encryption
1020H	Security	0	ROM security bit 1
1020H	Security	1	ROM security bit 2 0 = enable security feature 1 = disable security feature
1020H	Security	2	ROM security bit 3
1030H	WMOD	0	Watchdog mode bit; 00H = timer mode 01H = watchdog mode
1031H	PRE2:0	2:0	Watchdog prescaler selection; 00H = divide by 12 x 64 07H = divide by 12 x 64 x 128 (see specification)
1032H	WD	7:0	Watchdog autoload value (see specification)

Electrical Deviations from Commercial Specifications for Extended Temperature Range
DC and AC parameters not included here are the same as in the commercial temperature range table.

DC ELECTRICAL CHARACTERISTICS

T_A = –40°C to +85°C, V_{CC} = 5V ±10% (87C550), V_{CC} = 5V ±20% (80/83C550), V_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
V_{IL}	Input low voltage, except $\overline{EA}$		–0.5	$0.2V_{CC}$–0.15	V
V_{IL1}	Input low voltage to $\overline{EA}$		0	$0.2V_{CC}$–0.35	V
V_{IH}	Input high voltage, except XTAL1, RST		$0.2V_{CC}$+1	V_{CC}+0.5	V
V_{IH1}	Input high voltage to XTAL1, RST		$0.7V_{CC}$+0.1	V_{CC}+0.5	V
I_{IL}	Logical 0 input current, ports 1, 2, 3	V_{IN} = 0.45V		–75	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3	V_{IN} = 2.0V		–750	µA
I_{CC}	Power supply current: Active mode Idle mode Power down mode	V_{CC} = 4.5–5.5V, Frequency range = 3.5 to 12MHz		 35 6 50	 mA mA µA

ADC DC ELECTRCIAL CHARACTERISTICS

AV_{CC} = 5V ±10%, AV_{SS} = 0V, t_{AMB} = –40°C to 85°C, unless otherwise specified

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS		UNIT
			MIN	MAX	
AV_{CC}	Analog supply	AV_{CC} = V_{CC} ±0.2	4.5	5.5	V
Vref+	Analog reference			AV_{cc} + 0.2	V
AI_{CC}	Analog operating supply current	See note 1		6.0	mA
AV_{IN}	Analog input voltage		AV_{SS} – 0.2	AV_{CC} + 0.2	V
A_{IC}	Analog input capacitance			15	pF
t_{ADS}	Sampling time			$8t_{CY}$	
t_{ADC}	Conversion time			$40t_{CY}$	
	Differential nonlinearity	See note 1		±1	LSB
	Integral nonlinearity	See note 1		±1	LSB
OSe	Offset error	See note 1		±10	mV
Ge	Gain error	See note 1		0.4	%
M_{CTC}	Channel-to-channel matching			±1	LSB
Ct	Crosstalk	0 – 100kHz		–60	dB

NOTES:
1. Conditions: Vref+ = 4.99712V, Vref– = 0V.
2. The resistor ladder network is not disconnected in the power-down or idle modes. Thus to conserve power, the user must remove AV_{CC} and AVref.
3. If the A/D function is not required, or if the A/D function is only needed periodically, AV_{CC} can be removed without affecting the operation of the digital circuitry. Contents of ADCON and ADAT are not guaranteed to be valid. Digital inputs function normally. No digital outputs are present.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Operating temperature under bias	–40 to +85	°C
Storage temperature range	–65 to +150	°C
Voltage on $\overline{EA}/V_{PP}$ pin to V_{SS} (87C550 only)	0 to +13.0	V
Voltage on any other pin to V_{SS}	–0.5 to +6.5	V
Input, output current on any two I/O pins	±10	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.5	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10% (87C550), V_{CC} = 5V ±20% (80/83C550), V_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			MIN	TYPICAL[1]	MAX	
V_{IL}	Input low voltage, except $\overline{EA}$[7]		–0.5		$0.2V_{CC}$–0.1	V
V_{IL1}	Input low voltage to $\overline{EA}$[7]		0		$0.2V_{CC}$–0.3	V
V_{IH}	Input high voltage, except XTAL1, RST[7]		$0.2V_{CC}$+0.9		V_{CC}+0.5	V
V_{IH1}	Input high voltage, XTAL1, RST[7]		$0.7V_{CC}$		V_{CC}+0.5	V
V_{OL}	Output low voltage, ports 2, 3	I_{OL} = 1.6mA[2]			0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$	I_{OL} = 3.2mA[2]			0.45	V
V_{OH}	Output high voltage, ports 2, 3, ALE, $\overline{PSEN}$[3]	I_{OH} = –60µA, I_{OH} = –25µA I_{OH} = –10µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$			V V V
V_{OH1}	Output high voltage (port 0 in external bus mode)	I_{OH} = –800µA, I_{OH} = –300µA I_{OH} = –80µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$			V V V
I_{IL}	Logical 0 input current, ports 1, 2, 3[7]	V_{IN} = 0.45V			–50	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3[7]	See note 4			–650	µA
I_{LI}	Input leakage current, port 0	$V_{IN} = V_{IL}$ or V_{IH}			±10	µA
I_{CC}	Power supply current:[7] Active mode @ 12MHz[5] Idle mode @ 12MHz Power down mode	See note 6		 11.5 1.3 3	 25 4 50	 mA mA µA
R_{RST}	Internal reset pull-down resistor		50		300	kohm
C_{IO}	Pin capacitance				10	pF

NOTES:
1. Typical ratings are not guaranteed. The values listed are at room temperature, 5V.
2. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.
3. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
4. Pins of ports 2 and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
5. I_{CC}MAX at other frequencies is given by: Active mode; I_{CC}MAX = 0.94 X FREQ + 13.71: Idle mode; I_{CC}MAX = 0.14 X FREQ +2.31, where FREQ is the external oscillator frequency in MHz. I_{CC}MAX is given in mA. See Figure 12.
6. See Figures 13 through 16 for I_{CC} test conditions.
7. These values apply only to T_A = 0°C to +70°C. For T_A = –40°C to +85°C. See table on previous page.

AC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or −40°C to +85°C, V_{CC} = 5V ±10% (87C550), V_{CC} = 5V ±20% (80/83C550), V_{SS} = 0V[1, 2]

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	5	Oscillator frequency: **Speed Versions** S8XC550 −1, 2 S8XC550 −4, 5			 3.5 3.5	 12 16	 MHz MHz
t_{LHLL}	5	ALE pulse width	127		$2t_{CLCL}-40$		ns
t_{AVLL}	5	Address valid to ALE low	28		$t_{CLCL}-55$		ns
t_{LLAX}	5	Address hold after ALE low	48		$t_{CLCL}-35$		ns
t_{LLIV}	5	ALE low to valid instruction in		234		$4t_{CLCL}-100$	ns
t_{LLPL}	5	ALE low to $\overline{PSEN}$ low	43		$t_{CLCL}-40$		ns
t_{PLPH}	5	$\overline{PSEN}$ pulse width	205		$3t_{CLCL}-45$		ns
t_{PLIV}	5	$\overline{PSEN}$ low to valid instruction in		145		$3t_{CLCL}-105$	ns
t_{PXIX}	5	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	5	Input instruction float after $\overline{PSEN}$		59		$t_{CLCL}-25$	ns
t_{AVIV}	5	Address to valid instruction in		312		$5t_{CLCL}-105$	ns
t_{PLAZ}	5	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{RLRH}	6, 7	$\overline{RD}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{WLWH}	6, 7	$\overline{WR}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{RLDV}	6, 7	$\overline{RD}$ low to valid data in		252		$5t_{CLCL}-165$	ns
t_{RHDX}	6, 7	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	6, 7	Data float after $\overline{RD}$		97		$2t_{CLCL}-70$	ns
t_{LLDV}	6, 7	ALE low to valid data in		517		$8t_{CLCL}-150$	ns
t_{AVDV}	6, 7	Address to valid data in		585		$9t_{CLCL}-165$	ns
t_{LLWL}	6, 7	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	6, 7	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}-130$		ns
t_{QVWX}	6, 7	Data valid to $\overline{WR}$ transition	23		$t_{CLCL}-60$		ns
t_{WHQX}	6, 7	Data hold after $\overline{WR}$	33		$t_{CLCL}-50$		ns
t_{RLAZ}	6, 7	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	6, 7	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock							
t_{CHCX}	9	High time	20		20		ns
t_{CLCX}	9	Low time	20		20		ns
t_{CLCH}	9	Rise time		20		20	ns
t_{CHCL}	9	Fall time		20		20	ns
Shift Register							
t_{XLXL}	8	Serial port clock cycle time	1.0		$12t_{CLCL}$		µs
t_{QVXH}	8	Output data setup to clock rising edge	700		$10t_{CLCL}-133$		ns
t_{XHQX}	8	Output data hold after clock rising edge	50		$2t_{CLCL}-117$		ns
t_{XHDX}	8	Input data hold after clock rising edge	0		0		ns
t_{XHDV}	8	Clock rising edge to input data valid		700		$10t_{CLCL}-133$	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

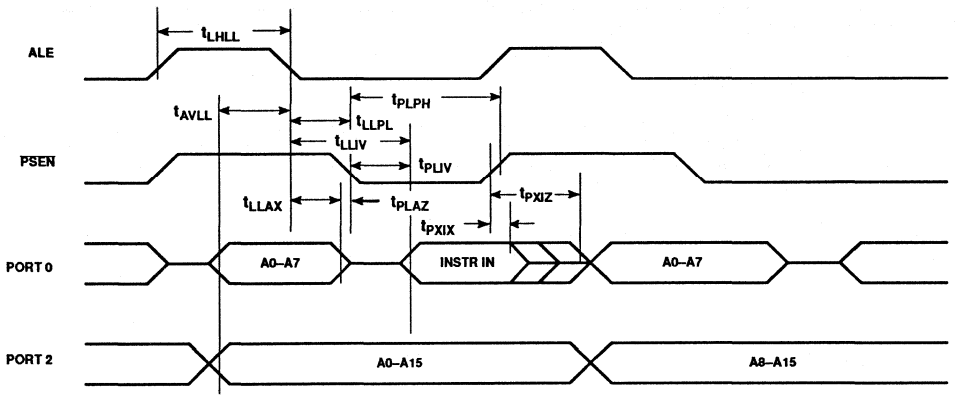


Figure 5. External Program Memory Read Cycle

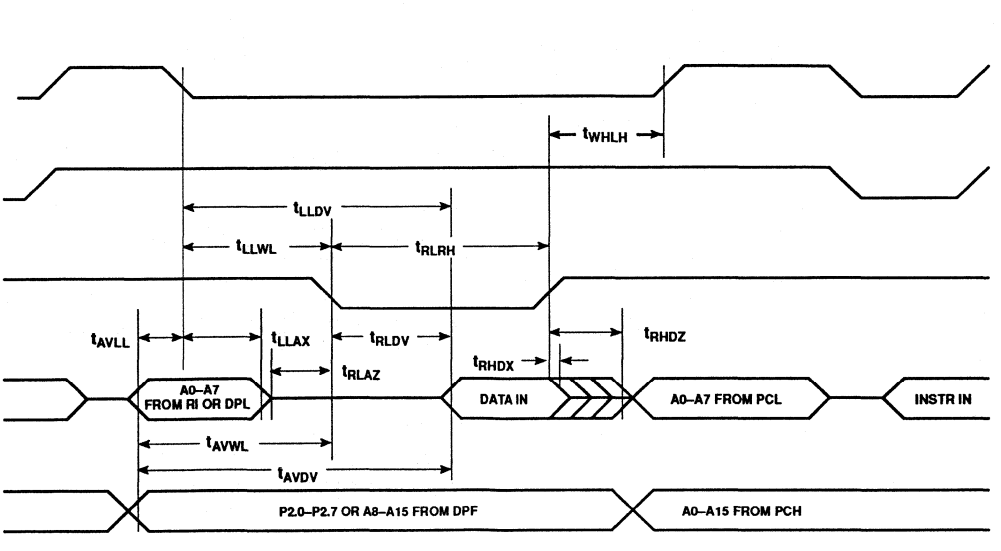


Figure 6. External Data Memory Read Cycle

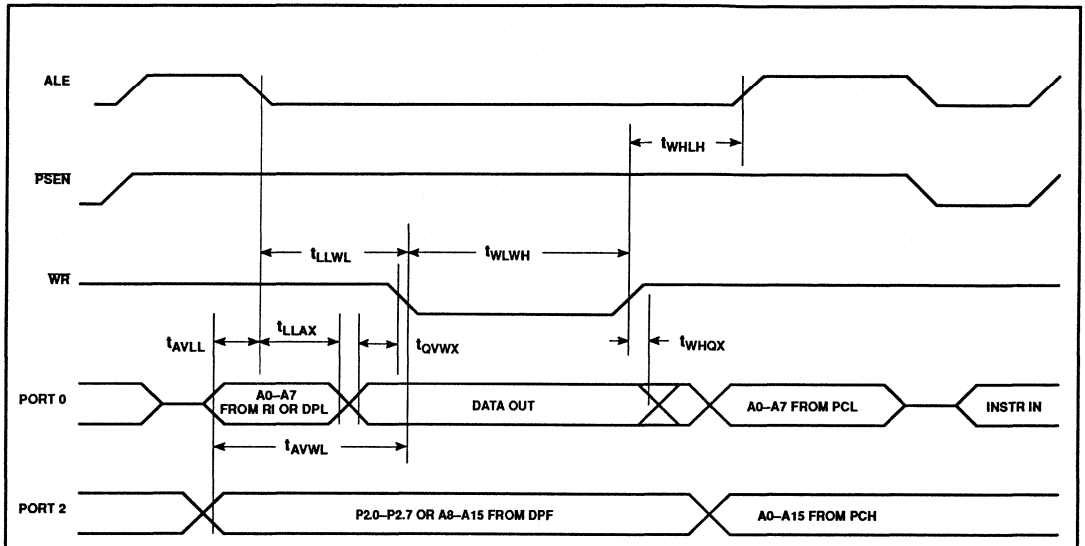


Figure 7. External Data Memory Write Cycle

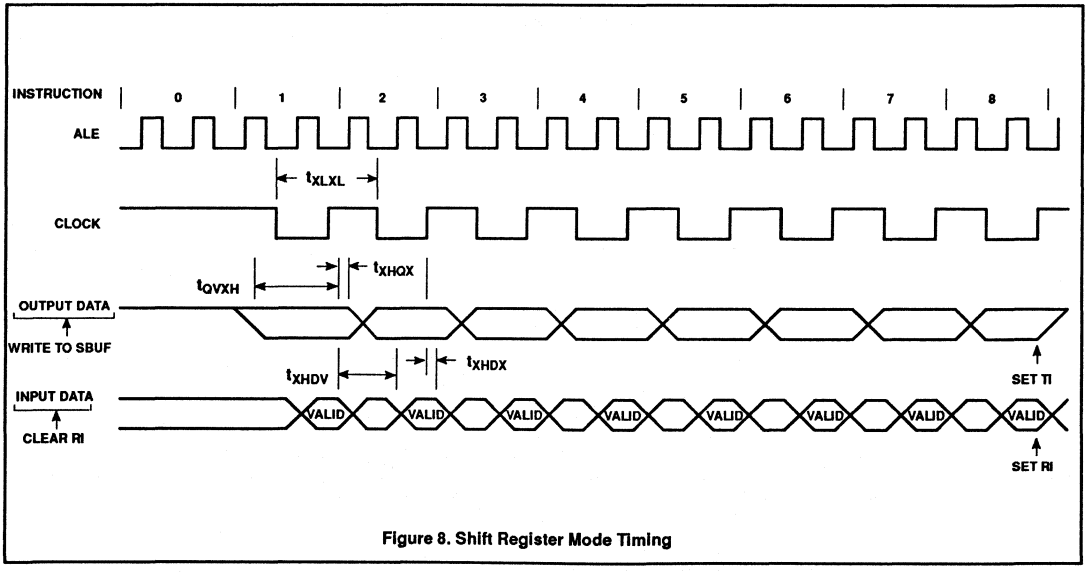


Figure 8. Shift Register Mode Timing

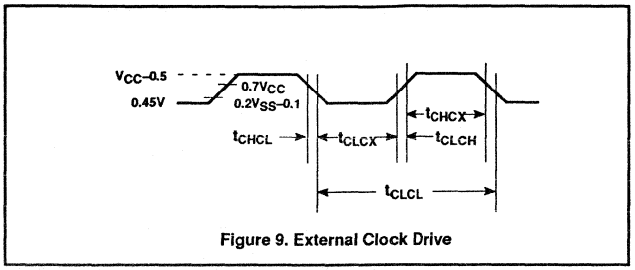
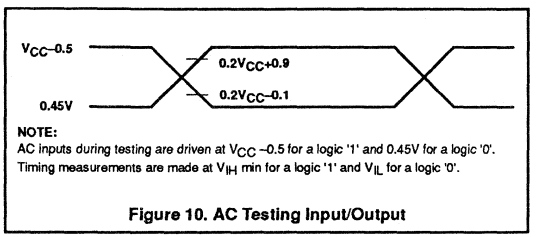
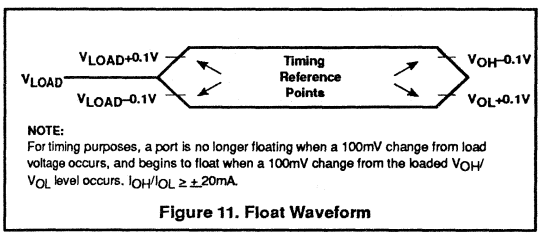


Figure 9. External Clock Drive



NOTE:
AC inputs during testing are driven at V_{CC} –0.5 for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at V_{IH} min for a logic '1' and V_{IL} for a logic '0'.

Figure 10. AC Testing Input/Output



NOTE:
For timing purposes, a port is no longer floating when a 100mV change from load voltage occurs, and begins to float when a 100mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OH}/I_{OL} \geq \pm 20$mA.

Figure 11. Float Waveform

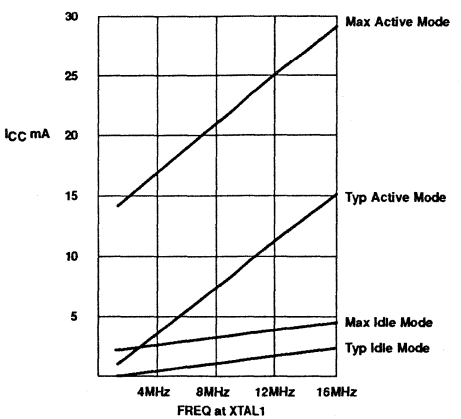


Figure 12. I_{CC} vs. FREQ
Valid only within frequency specifications of the device under test

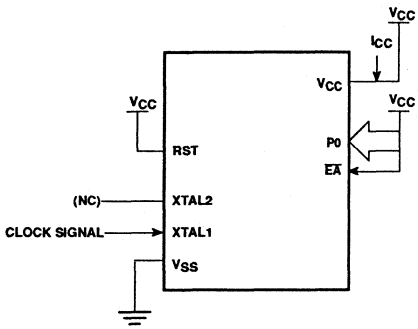


Figure 13. I_{CC} Test Condition, Active Mode
All other pins are disconnected

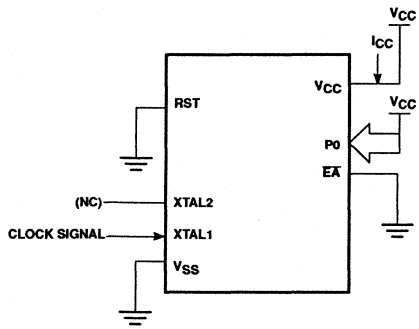


Figure 14. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

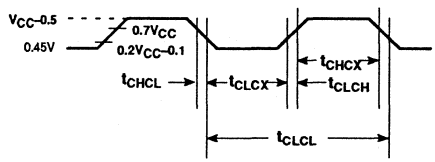


Figure 15. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

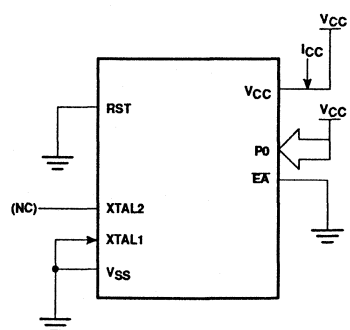


Figure 16. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected. V_{CC} = 2V to 5.5V

EPROM CHARACTERISTICS

The 87C550 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C550 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an S87C550 manufactured by Philips.

Table 4 shows the logic levels for reading the signature byte, and for programming the program memory, the encryption table, and the lock bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 17 and 18. Figure 19 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 17. Note that the 87C550 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 2 and 3, as shown in Figure 17. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$ and pins of ports 1 and 2 specified in Table 4 are held at the 'Program Code Data' levels indicated in Table 4. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 18.

To program the encryption table, repeat the 25 pulse programming sequence for addresses 0 through 1FH, using the "Pgm Encryption Table' levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the lock bits, repeat the 25 pulse programming sequence using the 'Pgm Lock Bit' levels. After one lock bit is programmed, further programming of the code memory and encryption table is disabled. However, the other lock bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If security bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be read is applied to ports 2 and 3 as shown in Figure 19. The other pins are held at the 'Verify Code Data' levels indicated in Table 4. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P1.0 and P1.1 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips

(031H) = 96H indicates S87C550

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 4, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345–5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000uW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

Table 4. EPROM Programming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P1.0	P1.1
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm security bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm security bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:

1. '0' = Valid low for that pin, '1' = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{CC} = 5V±10% during programming and verification.

*ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

™Trademark phrase of Intel Corporation.

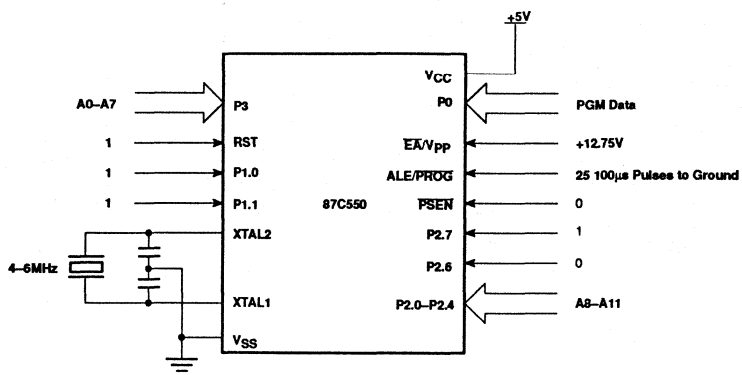


Figure 17. Programming Configuration

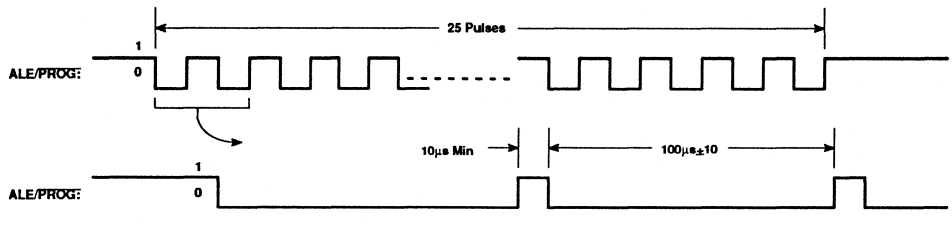


Figure 18. $\overline{\text{PROG}}$ Waveform

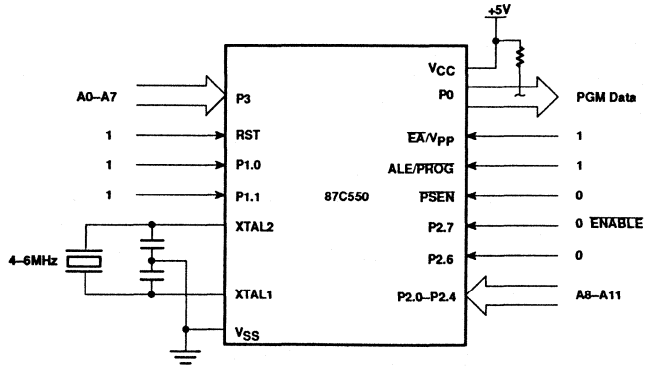


Figure 19. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_A = 21°C to +27°C, V_{CC} = 5V±10%, V_{SS} = 0V (See Figure 20)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		µs
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	$\overline{ENABLE}$ low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		µs

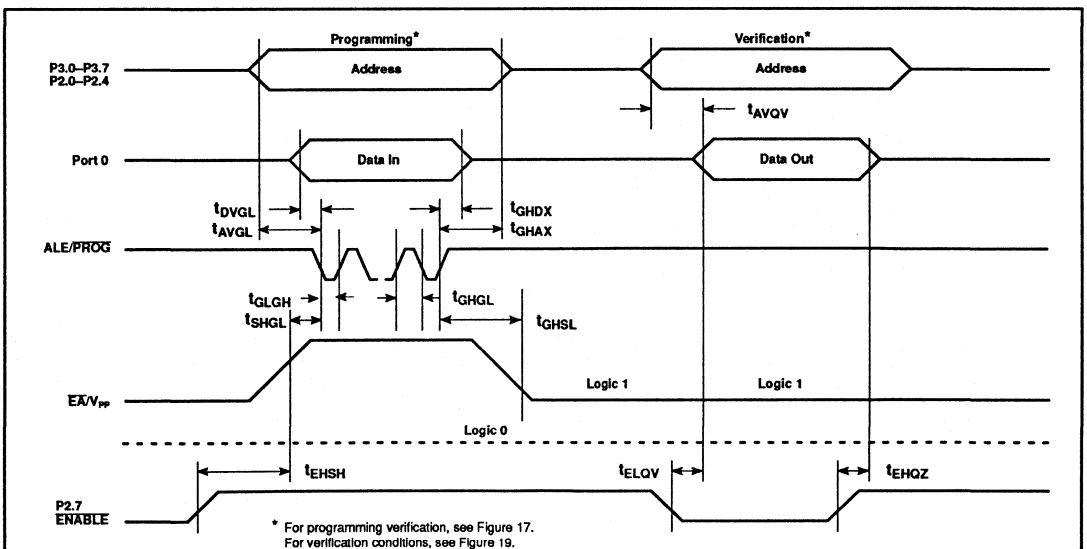


Figure 20. EPROM Programming and Verification

8XC552 OVERVIEW

The 8XC552 is a stand-alone high-performance microcontroller designed for use in real-time applications such as instrumentation, industrial control, and automotive control applications such as engine management and transmission control. The device provides, in addition to the 80C51 standard functions, a number of dedicated hardware functions for these applications.

The 8XC552 single-chip 8-bit microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. The 8XC552 uses the powerful instruction set of the 80C51. Additional special function registers are incorporated to control the on-chip peripherals. Three versions of the derivative exist although the generic term "8XC552" is used to refer to family members:

83C552: 8k bytes mask-programmable ROM, 256 bytes RAM

87C552: 8k bytes EPROM, 256 bytes RAM

80C552: ROMless version of the 83C552

The 8XC552 contains a nonvolatile 8k x 8 read-only program memory, a volatile 256 x 8 read/write data memory, six 8-bit I/O ports, two 16-bit timer/event counters (identical to the timers of the 80C51), an additional 16-bit timer coupled to capture and compare latches, a fifteen-source, two-priority-level, nested interrupt structure, an 8-input ADC, a dual DAC pulse width modulated interface, two serial interfaces (UART and I^2C bus), a "watchdog" timer, and on-chip oscillator and timing circuits. For systems that require extra capability, the 8XC552 can be expanded using standard TTL compatible memories and logic

The 8XC552 has two software selectable modes of reduced activity for further power reduction—Idle and Power-down. The idle mode freezes the CPU and resets Timer T2 and the ADC and PWM circuitry but allows the other timers, RAM, serial ports, and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to become inoperative.

83C562 OVERVIEW

The 83C562 has been derived from the 8XC552 with the following changes:

- The SIO1 (I^2C) interface has been omitted.
- The output of port lines P1.6 and P1.7 have a standard configuration instead of open drain.
- The resolution of the A/D converter is decreased from 10 bits to 8 bits.
- The speed of an A/D conversion has decreased from 50 machine cycles to 24 machine cycles.

All other functions, pinning and packaging are unchanged.

This chapter of the users' guide can be used for the 83C562 by omitting or changing the following:

- Disregard the description of SIO1 (I^2C).
- The SFRs for the interface: S1ADR, S1DAT, S1STA, and S1CON are not implemented. The two SIO1 related flags ES1 in SFR IEN0 and PS1 in SFR IP0 are also not implemented. These two flag locations are undefined after RESET. The interrupt vector for SIO1 is not used.
- Port lines P1.6 and P1.7 are not open drain but have the same standard configuration and electrical characteristics as P1.0-P1.5. Port lines P1.6 and P1.7 have alternative functions.
- The A/D converter has a resolution of 8 bits instead of 10 bits and consequently the two high-order bits 6 and 7 of SFR ADCON are not implemented. These two locations are undefined after RESET. The 8-bit result of an A/D conversion is present in SFR ADCH. The result can always be calculated from the formula:

$$256 \times \frac{V_{IN} - V_{ref-}}{AV_{ref+} - AV_{ref-}}$$

  The A/D conversion time is 24 machine cycles instead of 50 machine cycles, and the sampling time is 6 machine cycles instead of 8 machine cycles. The conversion time takes 3 machine cycles per bit.
- The serial I/O function SIO0 and its SFRs S0BUF and S0CON are renamed to SIO, SBUF, and SCON. The interrupt related flags ES0 and PS0 are renamed ES and PS. Interrupt source S0 is renamed S. The serial I/O function remains the same.

Differences From the 80C51

Program Memory

The 8XC552 contains 8k bytes of on-chip program memory which can be extended to 64k bytes with external memories (see Figure 19). When the $\overline{EA}$ pin is held high, the 8XC552 fetches instructions from internal ROM unless the address exceeds 1FFFH. Locations 2000H to FFFFH are fetched from external program memory. When the $\overline{EA}$ pin is held low, all instruction fetches are from external memory. ROM locations 0003H to 0073H are used by interrupt service routines.

Data Memory

The internal data memory is divided into 3 sections: the lower 128 bytes of RAM, the upper 128 bytes of RAM, and the 128-byte special function register areas. The lower 128 bytes of RAM are directly and indirectly addressable. While RAM locations 128 to 255 and the special function register area share the same address space, they are accessed through different addressing modes. RAM locations 128 to 255 are only indirectly addressable, and the special function registers are only directly addressable. All other aspects of the internal RAM are identical to the 8051.

The stack may be located anywhere in the internal RAM by loading the 8-bit stack pointer. Stack depth is 256 bytes maximum.

Special Function Registers

The special function registers (directly addressable only) contain all of the 8XC552 registers except the program counter and the four register banks. Most of the 56 special function registers are used to control the on-chip peripheral hardware. Other registers include arithmetic registers (ACC, B, PSW), stack pointer (SP), and data pointer registers (DHP, DPL). Sixteen of the SFRs contain 128 directly addressable bit locations. Table 14 lists the 8XC552's special function registers.

The standard 80C51 SFRs are present and function identically in the 8XC552 except where noted in the following sections.

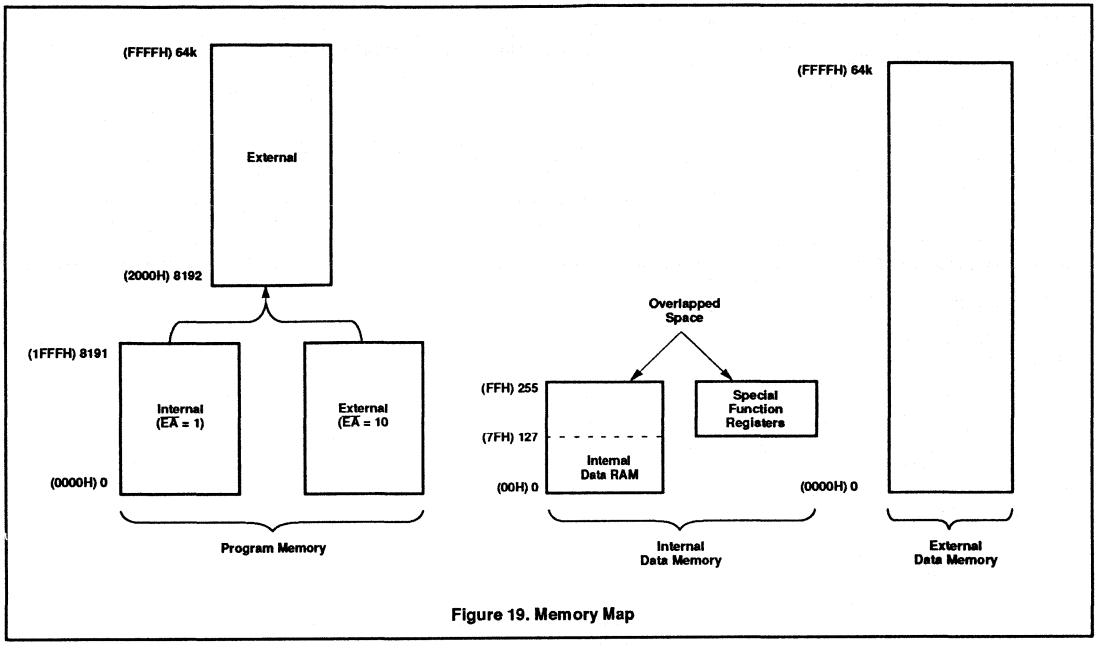


Figure 19. Memory Map

Timer T2

Timer T2 is a 16-bit timer consisting of two registers TMH2 (HIGH byte) and TML2 (LOW byte). The 16-bit timer/counter can be switched off or clocked via a prescaler from one of two sources: fosc/12 or an external signal. When Timer T2 is configured as a counter, the prescaler is clocked by an external signal on T2 (P1.4). A rising edge on T2 increments the prescaler, and the maximum repetition rate is one count per machine cycle (1MHz with a 12MHz oscillator).

The maximum repetition rate for Timer T2 is twice the maximum repetition rate for Timer 0 and Timer 1. T2 (P1.4) is sampled at S2P1 and again at S5P1 (i.e., twice per machine cycle). A rising edge is detected when T2 is LOW during one sample and HIGH during the next sample. To ensure that a rising edge is detected, the input signal must be LOW for at least 1/2 cycle and then HIGH for at least 1/2 cycle. If a rising edge is detected before the end of S2P1, the timer will be incremented during the following cycle; otherwise it will be incremented one cycle later. The prescaler has a programmable division factor of 1, 2, 4, or 8 and is cleared if its division factor or input source is changed, or if the timer/counter is reset.

Timer T2 may be read "on the fly" but possesses no extra read latches, and software precautions may have to be taken to avoid misinterpretation in the event of an overflow from least to most significant byte while Timer T2 is being read. Timer T2 is not loadable and is reset by the RST signal or by a rising edge on the input signal RT2, if enabled. RT2 is enabled by setting bit T2ER (TM2CON.5).

When the least significant byte of the timer overflows or when a 16-bit overflow occurs, an interrupt request may be generated. Either or both of these overflows can be programmed to request an interrupt. In both cases, the interrupt vector will be the same. When the lower byte (TML2) overflows, flag T2B0 (TM2CON) is set and flag T20V (TM2IR) is set when TMH2 overflows. These flags are set one cycle after an overflow occurs. Note that when T20V is set, T2B0 will also be set. To enable the byte overflow interrupt, bits ET2 (IEN1.7, enable overflow interrupt, see Figure 20) and T2IS0 (TM2CON.6, byte overflow interrupt select) must be set. Bit TWB0 (TM2CON.4) is the Timer T2 byte overflow flag.

To enable the 16-bit overflow interrupt, bits ET2 (IE1.7, enable overflow interrupt) and T2IS1 (TM2CON.7, 16-bit overflow interrupt select) must be set. Bit T2OV (TM2IR.7) is the Timer T2 16-bit overflow flag. All interrupt flags must be reset by software. To enable both byte and 16-bit overflow, T2IS0 and T2IS1 must be set and two interrupt service routines are required. A test on the overflow flags indicates which routine must be executed. For each routine, only the corresponding overflow flag must be cleared.

Timer T2 may be reset by a rising edge on RT2 (P1.5) if the Timer T2 external reset enable bit (T2ER) in T2CON is set. This reset also clears the prescaler. In the idle mode, the timer/counter and prescaler are reset and halted. Timer T2 is controlled by the TM2CON special function register (see Figure 21).

Table 14. 8XC552 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
ADCH#	A/D converter high	C6H									xxxxxxxxB
ADCON#	Adc control	C5H	ADC.1	ADC.0	ADEX	ADCI	ADCS	AADR2	AADR1	AADR0	xx000000B
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
CTCON#	Capture control	EBH	CTN3	CTP3	CTN2	CTP2	CTN1	CTP	CTN0	CTP0	00H
CTH3#	Capture high 3	CFH									xxxxxxxxB
CTH2#	Capture high 2	CEH									xxxxxxxxB
CTH1#	Capture high 1	CDH									xxxxxxxxB
CTH0#	Capture high 0	CCH									xxxxxxxxB
CMH2#	Compare high 2	CBH									00H
CMH1#	Compare high 1	CAH									00H
CMH0#	Compare high 0	C9H									00H
CTL3#	Capture low 3	AFH									xxxxxxxxB
CTL2#	Capture low 2	AEH									xxxxxxxxB
CTL1#	Capture low 1	ADH									xxxxxxxxB
CTL0#	Capture low 0	ACH									xxxxxxxxB
CML2#	Compare low 2	ABH									00H
CML1#	Compare low 1	AAH									00H
CML0#	Compare low 0	A9H									00H
DPTR:	Data pointer (2 bytes)										
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IEN0*	Interrupt enable 0	A8H	EA	EAD	ES1	ES0	ET1	EX1	ET0	EX0	00H
			EF	EE	ED	EC	EB	EA	E9	E8	
IEN1*#	Interrupt enable 1	E8H	ET2	ECM2	ECM1	ECM0	ECT3	ECT2	ECT1	ECT0	00H
			BF	BE	BD	BC	BB	BA	B9	B8	
IP0*	Interrupt priority 0	B8H	–	PAD	PS1	PS0	PT1	PX1	PT0	PX0	x0000000B
			FF	FE	FD	FC	FB	FA	F9	F8	
IP1*#	Interrupt priority 1	F8H	PT2	PCM2	PCM1	PCM0	PCT3	PCT2	PCT1	PCT0	00H
P5#	Port 5	C4H	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	xxxxxxxxB
			C7	C6	C5	C4	C3	C2	C1	C0	
P4#	Port 4	C0H	CMT1	CMT2	CMSR5	CMSR4	CMSR3	CMSR2	CMSR1	CMSR0	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	$\overline{RD}$	$\overline{WR}$	T1	T0	$\overline{INT1}$	$\overline{INT0}$	TXD	RXD	FFH
			A7	A6	A5	A4	A3	A2	A1	A0	
P2*	Port 2	A0H	A15	A14	A13	A12	A11	A10	A9	A8	FFH
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	SDA	SCL	RT2	T2	CT3I	CT2I	CT1I	CT0I	FFH
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	FFH
PCON	Power control	87H	SMOD	–	–	WLE	GF1	GF0	PD	IDL	00xx0000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	F1	P	00H

*SFRs are bit addressable.
#SFRs are modified from or added to the 80C51 SFRs.

Table 14. 8XC552 Special Function Registers (Continued)

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
PWMP#	PWM prescaler	FEH									00H
PWM1#	PWM register 1	FDH									00H
PWM0#	PWM register 0	FCH									00H
RTE#	Reset/toggle enable	EFH	TP47	TP46	RP45	RP44	RP43	RP42	RP41	RP40	00H
SP	Stack pointer	81H									07H
S0BUF	Serial 0 data buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
S0CON*	Serial 0 control	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
S1ADR#	Serial 1 address	DBH	SLAVE ADDRESS							GC	00H
S1DAT#	Serial 1 data	DAH									00H
S1STA#	Serial 1 status	D9H	SC4	SC3	SC2	SC1	SC0	0	0	0	F8H
			DF	DE	DD	DC	DB	DA	D9	D8	
S1CON#*	Serial 1 control	D8H	CR2	ENS1	STA	STO	SI	AA	CR1	CR0	00H
STE#	Set enable	EEH	TG47	TG46	SP45	SP44	SP43	SP42	SP41	SP40	C0H
TH1	Timer high 1	8DH									00H
TH0	Timer high 0	8CH									00H
TL1	Timer low 1	8BH									00H
TL0	Timer low 0	8AH									00H
TMH2#	Timer high 2	EDH									00H
TML2#	Timer low 2	ECH									00H
TMOD	Timer mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
TM2CON#	Timer 2 control	EAH	T2IS1	TSIS0	T2ER	T2B0	T2P1	T2P0	T2MS1	T2MS0	00H
			CF	CE	CD	CC	CB	CA	C9	C8	
TM2IR#	Timer 2 int flag reg	C8H	T20V	CMI2	CMI1	CMI0	CTI3	CTI2	CTI1	CTI0	00H
T3#	Timer 3	FFH									00H

*SFRs are bit addressable.
#SFRs are modified from or added to the 80C51 SFRs.

	7	6	5	4	3	2	1	0
IEN1 (E8H	ET2	ECM2	ECM1	ECM0	ECT3	ECT2	ECT1	ECT0
	(MSB)							(LSB)

Bit	Symbol	Function
IEN1.7	ET2	Enable Timer T2 overflow interrupt(s)
IEN1.6	ECM2	Enable T2 Comparator 2 interrupt
IEN1.5	ECM1	Enable T2 Comparator 1 interrupt
IEN1.4	ECM0	Enable T2 Comparator 0 interrupt
IEN1.3	ECT3	Enable T2 Capture register 3 interrupt
IEN1.2	ECT2	Enable T2 Capture register 2 interrupt
IEN1.1	ECT1	Enable T2 Capture register 1 interrupt
IEN1.0	ECT0	Enable T2 Capture register 0 interrupt

Figure 20. Timer T2 Interrupt Enable Register (IEN1)

TM2CON (EAH)

7	6	5	4	3	2	1	0
T2IS1	T2IS0	T2ER	T2BO	T2P1	T2P0	T2MS1	T2MS0

(MSB) (LSB)

Bit	Symbol	Function
TM2CON.7	T2IS1	Timer T2 16-bit overflow interrupt select
TM2CON.6	T2IS0	Timer T2 byte overflow interrupt select
TM2CON.5	T2ER	Timer T2 external reset enable. When this bit is set, Timer T2 may be reset by a rising edge on RT2 (P1.5).
TM2CON.4	T2BO	Timer T2 byte overflow interrupt flag
TM2CON.3	T2P1	Timer T2 prescaler select
TM2CON.2	T2P0	

T2P1	T2P0	Timer T2 Clock
0	0	Clock source
0	1	Clock source/2
1	0	Clock source/4
1	1	Clock source/8

TM2CON.1	T2MS1	Timer T2 mode select
TM2CON.2	T2MS0	

T2MS1	T2MS0	Mode Selected
0	0	Timer T2 halted (off)
0	1	T2 clock source = $f_{osc}/12$
1	0	Test mode; do not use
1	1	T2 clock source = pin T2

Figure 21. T2 Control Register (TM2CON)

Timer T2 Extension: When a 12MHz oscillator is used, a 16-bit overflow on Timer T2 occurs every 65.5, 131, 262, or 524 ms, depending on the prescaler division ratio; i.e., the maximum cycle time is approximately 0.5 seconds. In applications where cycle times are greater than 0.5 seconds, it is necessary to extend Timer T2. This is achieved by selecting fosc/12 as the clock source (set T2MS0, reset T2MS1), setting the prescaler division ration to 1/8 (set T2P0, set T2P1), disabling the byte overflow interrupt (reset T2IS0) and enabling the 16-bit overflow interrupt (set T2IS1). The following software routine is written for a three-byte extension which gives a maximum cycle time of approximately 2400 hours.

```
OVINT: PUSH  ACC      ;save accumulator
       PUSH  PSW      ;save status
       INC   TIMEX1   ;increment first
                      ;byte (low order)
                      ;of extended timer
       MOV   A,TIMEX1
       JNZ   INTEX    ;jump to INTEX if
                      ;there is no
                      ;overflow
       INC   TIMEX2   ;increment second
                      ;byte
       MOV   A,TIMEX2
       JNZ   INTEX    ;jump to INTEX if
                      ;there is no
                      ;overflow
       INC   TIMEX3   ;increment third
                      ;byte (high order)
INTEX: CLR   T2OV     ;reset interrupt
                      ;flag
       POP   PSW      ;restore status
       POP   ACC      ;restore accumula-
                      tor
       RETI           ;return from
                      ;interrupt
```

Timer T2, Capture and Compare Logic: Timer T2 is connected to four 16-bit capture registers and three 16-bit compare registers. A capture register may be used to capture the contents of Timer T2 when a transition occurs on its corresponding input pin. A compare register may be used to set, reset, or toggle port 4 output pins at certain pre-programmable time intervals.

The combination of Timer T2 and the capture and compare logic is very powerful in applications involving rotating machinery, automotive injection systems, etc. Timer T2 and the capture and compare logic are shown in Figure 22.

Capture Logic: The four 16-bit capture registers that Timer T2 is connected to are: CT0, CT1, CT2, and CT3. These registers are loaded with the contents of Timer T2, and an interrupt is requested upon receipt of the input signals CT0I, CT1I, CT2I, or CT3I. These input signals are shared with port 1. The four interrupt flags are in the Timer T2 interrupt register (TM2IR special function register). If the capture facility is not required, these inputs can be regarded as additional external interrupt inputs.

Using the capture control register CTCON (see Figure 23), these inputs may capture on a rising edge, a falling edge, or on either a rising or falling edge. The inputs are sampled during S1P1 of each cycle. When a selected edge is detected, the contents of Timer T2 are captured at the end of the cycle.

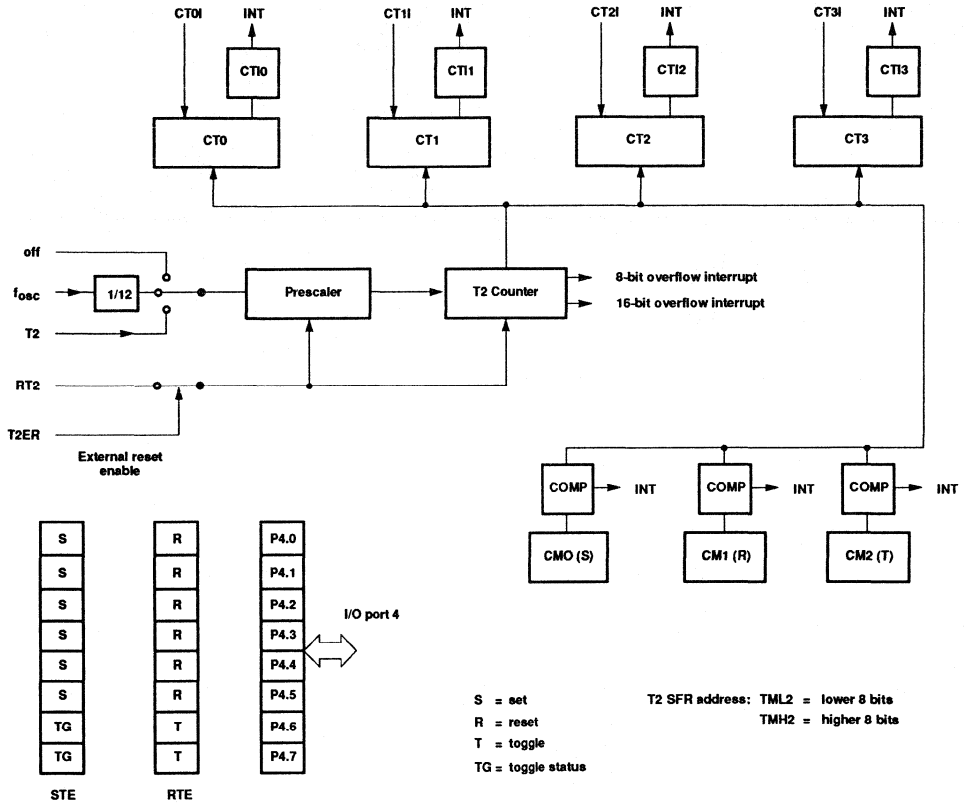


Figure 22. Block Diagram of Timer 2

Measuring Time Intervals Using Capture Registers: When a recurring external event is represented in the form of rising or falling edges on one of the four capture pins, the time between two events can be measured using Timer T2 and a capture register. When an event occurs, the contents of Timer T2 are copied into the relevant capture register and an interrupt request is generated. The interrupt service routine may then compute the interval time if it knows the previous contents of Timer T2 when the last event occurred. With a 12MHz oscillator, Timer T2 can be programmed to overflow every 524ms. When event interval times are shorter than this, computing the interval time is simple, and the interrupt service routine is short. For longer interval times, the Timer T2 extension routine may be used.

Compare Logic: Each time Timer T2 is incremented, the contents of the three 16-bit compare registers CM0, CM1, and CM2 are compared with the new counter value of Timer T2. When a match is found, the corresponding interrupt flag in TM2IR is set at the end of the following cycle. When a match with CM0 occurs, the controller sets bits 0-5 of port 4 if the corresponding bits of the set enable register STE are at logic 1.

When a match with CM1 occurs, the controller resets bits 0-5 of port 4 if the corresponding bits of the reset/toggle enable register RTE are at logic 1 (see Figure 24 for RTE register function). If RTE is "0", then P4.n is not affected by a match between CM1 or CM2 and Timer 2. When a match with CM2 occurs, the controller "toggles" bits 6 and 7 of port 4 if the corresponding bits of the RTE are at logic 1. The port latches of bits 6 and 7 are not toggled. Two additional flip-flops store the last operation, and it is these flip-flops that are toggled.

Thus, if the current operation is "set," the next operation will be "reset" even if the port latch is reset by software before the "reset" operation occurs. The first "toggle" after a chip RESET will set the port latch. The contents of these two flip-flops can be read at STE.6 and STE.7 (corresponding to P4.6 and P4.7, respectively). Bits STE.6 and STE.7 are read only (see Figure 25 for STE register function). A logic 1 indicates that the next toggle will set the port latch; a logic 0 indicates that the next toggle will reset the port latch. CM0, CM1, and CM2 are reset by the RST signal.

The modified port latch information appears at the port pin during S5P1 of the cycle following the cycle in which a match occurred. If the port is modified by software, the outputs change during S1P1 of the following cycle. Each port 4 bit can be set or reset by software at any time. A hardware modification resulting from a comparator match takes precedence over a software modification in the same cycle. When the comparator results require a "set" and a "reset" at the same time, the port latch will be reset.

Timer T2 Interrupt Flag Register TM2IR: Eight of the nine Timer T2 interrupt flags are located in special function register TM2IR (see Figure 26). The ninth flag is TM2CON.4.

The CT0I and CT1I flags are set during S4 of the cycle in which the contents of Timer T2 are captured. CT0I is scanned by the interrupt logic during S2, and CT1I is scanned during S3. CT2I and CT3I are set during S6 and are scanned during S4 and S5. The associated interrupt requests are recognized during the following cycle. If these flags are polled, a transition at CT0I or CT1I will be recognized one cycle before a transition on CT2I or CT3I since registers are read during S5. The CMI0, CMI1, and CMI2 flags are set during S6 of the cycle following a match. CMI0 is scanned by the interrupt logic during S2; CMI1 and CMI2 are scanned during S3 and S4. A match will be recognized by the interrupt logic (or by polling the flags) two cycles after the match takes place.

The 16-bit overflow flag (T2OV) and the byte overflow flag (T2BO) are set during S6 of the cycle in which the overflow occurs. These flags are recognized by the interrupt logic during the next cycle.

Special function register IP1 (Figure 26) is used to determine the Timer T2 interrupt priority. Setting a bit high gives that function a high priority, and setting a bit low gives the function a low priority. The functions controlled by the various bits of the IP1 register are shown in Figure 26.

	7	6	5	4	3	2	1	0
CTCON (E8H)	CTN3	CTP3	CTN2	CTP2	CTN1	CTP1	CTN0	CTP0
	(MSB)							(LSB)

Bit	Symbol	Capture/Interrupt on:
CTCON.7	CTN3	Capture Register 3 triggered by a falling edge on CT3I
CTCON.6	CTP3	Capture Register 3 triggered by a rising edge on CT3I
CTCON.5	CTN2	Capture Register 2 triggered by a falling edge on CT2I
CTCON.4	CTP2	Capture Register 2 triggered by a rising edge on CT2I
CTCON.3	CTN1	Capture Register 1 triggered by a falling edge on CT1I
CTCON.2	CTP1	Capture Register 1 triggered by a rising edge on CT1I
CTCON.1	CTN0	Capture Register 0 triggered by a falling edge on CT0I
CTCON.0	CTP0	Capture Register 0 triggered by a rising edge on CT0I

Figure 23. Capture Control Register (CTCON)

	7	6	5	4	3	2	1	0
RTE (EFH)	TP47	TP46	RP45	RP44	RP43	RP42	RO41	RP40
	(MSB)							(LSB)

Bit	Symbol	Function
RTE.7	TP47	If "1" then P4.7 toggles on a match between CM2 and Timer T2
RTE.6	TP46	If "1" then P4.6 toggles on a match between CM2 and Timer T2
RTE.5	RP45	If "1" then P4.5 is reset on a match between CM1 and Timer T2
RTE.4	RP44	If "1" then P4.4 is reset on a match between CM1 and Timer T2
RTE.3	RP43	If "1" then P4.3 is reset on a match between CM1 and Timer T2
RTE.2	RP42	If "1" then P4.2 is reset on a match between CM1 and Timer T2
RTE.1	RP41	If "1" then P4.1 is reset on a match between CM1 and Timer T2
RTE.0	RP40	If "1" then P4.0 is reset on a match between CM1 and Timer T2

Figure 24. Reset/Toggle Enable Register (RTE)

	7	6	5	4	3	2	1	0
STE (EEH)	TG47	TG46	SP45	SP44	SP43	SP42	SP41	SP40
	(MSB)							(LSB)

Bit	Symbol	Function
STE.7	TG47	Toggle flip-flops
STE.6	TG46	Toggle flip-flops
STE.5	SP45	If "1" then P4.5 is set on a match between CM0 and Timer T2
STE.4	SP44	If "1" then P4.4 is set on a match between CM0 and Timer T2
STE.3	SP43	If "1" then P4.3 is set on a match between CM0 and Timer T2
STE.2	SP42	If "1" then P4.2 is set on a match between CM0 and Timer T2
STE.1	SP41	If "1" then P4.1 is set on a match between CM0 and Timer T2
STE.0	SP40	If "1" then P4.0 is set on a match between CM0 and Timer T2

Figure 25. Set Enable Register (STE)

	7	6	5	4	3	2	1	0
TM2IR (C8H)	T2OV	CMI2	CMI1	CMI0	CTI3	CTI2	CTI1	CTI0
	(MSB)							(LSB)

Bit	Symbol	Function
TM2IR.7	T2OV	Timer T2 16-bit overflow interrupt flag
TM2IR.6	CMI2	CM2 interrupt flag
TM2IR.5	CMI1	CM1 interrupt flag
TM2IR.4	CMI0	CM0 interrupt flag
TM2IR.3	CTI3	CT3 interrupt flag
TM2IR.2	CTI2	CT2 interrupt flag
TM2IR.1	CTI1	CT1 interrupt flag
TM2IR.0	CTI0	CT0 interrupt flag

Interrupt Flag Register (TM2IR)

	7	6	5	4	3	2	1	0
IP1 (F8H)	PT2	PCM2	PCM1	PCM0	PCT3	PCT2	PCT1	PCT0
	(MSB)							(LSB)

Bit	Symbol	Function
IP1.7	PT2	Timer T2 overflow interrupt(s) priority level
IP1.6	PCM2	Timer T2 comparator 2 interrupt priority level
IP1.5	PCM1	Timer T2 comparator 1 interrupt priority level
IP1.4	PCM0	Timer T2 comparator 0 interrupt priority level
IP1.3	PCT3	Timer T2 capture register 3 interrupt priority level
IP1.2	PCT2	Timer T2 capture register 2 interrupt priority level
IP1.1	PCT1	Timer T2 capture register 1 interrupt priority level
IP1.0	PCT0	Timer T2 capture register 0 interrupt priority level

Timer 2 Interrupt Priority Register (IP1)

Figure 26. Interrupt Flag Register (TM2IR) and Timer T2 Interrupt Priority Register (IP1)

Timer T3, The Watchdog Timer

In addition to Timer T2 and the standard timers, a watchdog timer is also incorporated on the 8XC552. The purpose of a watchdog timer is to reset the microcontroller if it enters erroneous processor states (possibly caused by electrical noise or RFI) within a reasonable period of time. An analogy is the "dead man's handle" in railway locomotives. When enabled, the watchdog circuitry will generate a system reset if the user program fails to reload the watchdog timer within a specified length of time known as the "watchdog interval."

Watchdog Circuit Description: The watchdog timer (Timer T3) consists of an 8-bit timer with an 11-bit prescaler as shown in Figure 27. The prescaler is fed with a signal whose frequency is 1/12 the oscillator frequency (1MHz with a 12MHz oscillator). The 8-bit timer is incremented every "t" seconds, where:

t = 12 x 2048 x 1/fosc (=2ms at fosc = 12 MHz)

If the 8-bit timer overflows, a short internal reset pulse is generated which will reset the 8XC552. A short output reset pulse is also generated at the RST pin. This short output pulse (3 machine cycles) may be destroyed if the RST pin is connected to a capacitor. This would not, however, affect the internal reset operation.

Watchdog operation is activated when external pin $\overline{EW}$ is tied low. When $\overline{EW}$ is tied low, it is impossible to disable the watchdog operation by software.

How to Operate the Watchdog Timer: The watchdog timer has to be reloaded within periods that are shorter than the programmed watchdog interval; otherwise the watchdog timer will overflow and a system reset will be generated. The user program must therefore continually execute sections of code which reload the watchdog timer. The period of time elapsed between execution of these sections of code must never exceed the watchdog interval. When using a 12MHz oscillator, the watchdog interval is programmable between 2ms and 510ms.

In order to prepare software for watchdog operation, a programmer should first determine how long his system can sustain an erroneous processor state. The result will be the maximum watchdog interval. As the maximum watchdog interval becomes shorter, it becomes more difficult for the programmer to ensure that the user program always reloads the watchdog timer within the watchdog interval, and thus it becomes more difficult to implement watchdog operation.

The programmer must now partition the software in such a way that reloading of the watchdog is carried out in accordance with the above requirements. The programmer must determine the execution times of all software modules. The effect of possible conditional branches, subroutines, external and internal interrupts must all be taken into account. Since it may be very difficult to evaluate the execution times of some sections of code, the programmer should use worst case estimations. In any event, the programmer must make sure that the watchdog is not activated during normal operation.

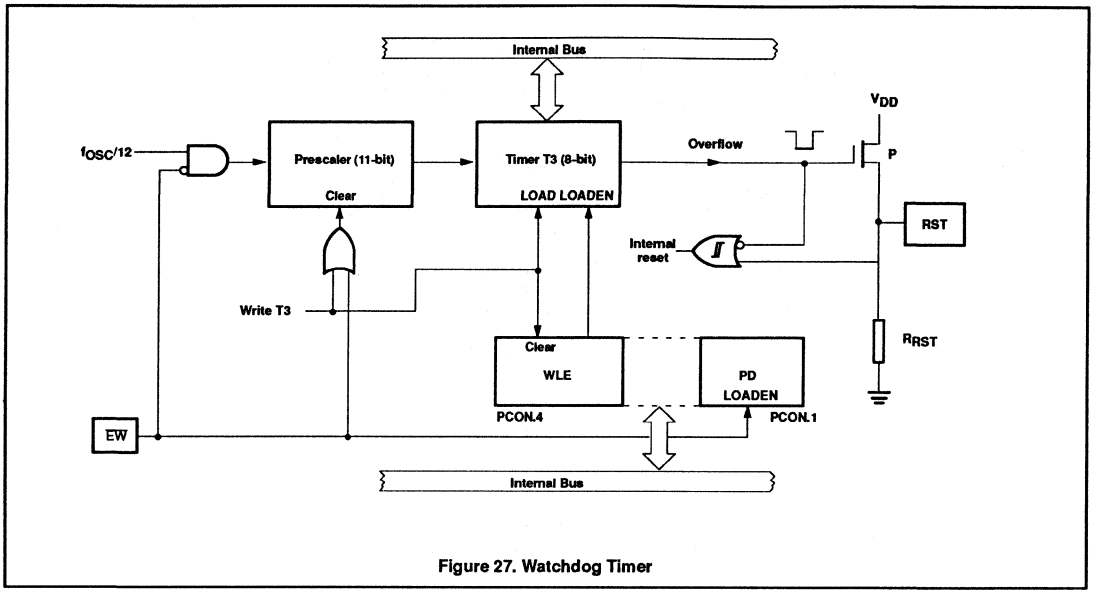


Figure 27. Watchdog Timer

The watchdog timer is reloaded in two stages in order to prevent erroneous software from reloading the watchdog. First PCON.4 (WLE) must be set. The T3 may be loaded. When T3 is loaded, PCON.4 (WLE) is automatically reset. T3 cannot be loaded if PCON.4 (WLE) is reset. Reload code may be put in a subroutine as it is called frequently. Since Timer T3 is an up-counter, a reload value of 00H gives the maximum watchdog interval (510ms with a 12MHz oscillator), and a reload value of 0FFH gives the minimum watchdog interval (2ms with a 12MHz oscillator).

In the idle mode, the watchdog circuitry remains active. When watchdog operation is implemented, the power-down mode cannot be used since both states are contradictory. Thus, when watchdog operation is enabled by tying external pin $\overline{EW}$ low, it is impossible to enter the power-down mode, and an attempt to set the power-down bit (PCON.1) will have no effect. PCON.1 will remain at logic 0.

During the early stages of software development/debugging, the watchdog may be disabled by tying the $\overline{EW}$ pin high. At a later stage, $\overline{EW}$ may be tied low to complete the debugging process.

Watchdog Software Example: The following example shows how watchdog operation might be handled in a user program.

```
;at the program start
        :
T3          EQU  0FFH ;address of
                      ;watchdog
                      ;timer T3
PCON        EQU  087H ;address of
                      ;PCON SFR
WATCH-INTV  EQU  156  ;watchdog
                      ;interval
                      ;(e.g., 2x100ms)
;to be inserted at each watchdog reload
;location within the user program:
LCALL WATCHDOG
;watchdog service routine:
WATCHDOG: ORL  PCON,#10H;set
                        ;condition
                        ;flag
                        (PCON.4)
          MOV T3,WATCH-INV ;load T3
                           ;with
                           ;watchdog
                           ;interval
          RET
```

If it is possible for this subroutine to be called in an erroneous state, then the condition flag WLE should be set at different parts of the main program.

Serial I/O

The 8XC552 is equipped with two independent serial ports: SIO0 and SIO1. SIO0 is a full duplex UART port and is identical to the 80C51 serial port. SIO1 accommodates the I^2C bus.

SIO0: SIO0 is a full duplex serial I/O port identical to that on the 80C51. Its operation is the same, including the use of timer 1 as a baud rate generator.

SIO1, I2C Serial I/O: The I^2C bus uses two wires (SDA and SCL) to transfer information between devices connected to the bus. The main features of the bus are:

- Bidirectional data transfer between masters and slaves
- Multimaster bus (no central master)
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus
- Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer
- The I^2C bus may be used for test and diagnostic purposes

The output latches of P1.6 and P1.7 must be set to logic 1 in order to enable SIO1.

The 8XC552 on-chip I^2C logic provides a serial interface that meets the I^2C bus specification and supports all transfer modes (other than the low-speed mode) from and to the I^2C

bus. The SIO1 logic handles bytes transfer autonomously. It also keeps track of serial transfers, and a status register (S1STA) reflects the status of SIO1 and the I^2C bus.

The CPU interfaces to the I^2C logic via the following four special function registers: S1CON (SIO1 control register), S1STA (SIO1 status register), S1DAT (SIO1 data register), and S1ADR (SIO1 slave address register). The SIO1 logic interfaces to the external I^2C bus via two port 1 pins: P1.6/SCL (serial clock line) and P1.7/SDA (serial data line).

A typical I^2C bus configuration is shown in Figure 28, and Figure 29 shows how a data transfer is accomplished on the bus. Depending on the state of the direction bit (R/W), two types of data transfers are possible on the I^2C bus:

1. Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte.
2. Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. Next follows the data bytes transmitted by the slave to the master. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a "not acknowledge" is returned.

The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the I^2C bus will not be released.

Modes of Operation: The on-chip SIO1 logic may operate in the following four modes:

1. Master Transmitter Mode:

Serial data output through P1.7/SDA while P1.6/SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the data direction bit. In this case the data direction bit ($R/\overline{W}$) will be logic 0, and we say that a "W" is transmitted. Thus the first byte transmitted is SLA+W. Serial data is transmitted 8 bits at a time. After each byte is transmitted, an acknowledge bit is received. START and STOP conditions are output to indicate the beginning and the end of a serial transfer.

2. Master Receiver Mode:

The first byte transmitted contains the slave address of the transmitting device (7 bits) and the data direction bit. In this case the data direction bit ($R/\overline{W}$) will be logic 1, and we say that an "R" is transmitted. Thus the first byte transmitted is SLA+R. Serial data is received via P1.7/SDA while P1.6/SCL outputs the serial clock. Serial data is received 8 bits at a time. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are output to indicate the beginning and end of a serial transfer.

3. Slave Receiver Mode:

Serial data and the serial clock are received through P1.7/SDA and P1.6/SCL. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and direction bit.

4. Slave Transmitter Mode:

The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will indicate that the transfer direction is reversed. Serial data is transmitted via P1.7/SDA while the serial clock is input through P1.6/SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer.

In a given application, SIO1 may operate as a master and as a slave. In the slave mode, the SIO1 hardware looks for its own slave address and the general call address. If one of these addresses is detected, an interrupt is requested. When the microcontroller wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave action is not interrupted. If bus arbitration is lost in the master mode, SIO1 switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

SIO1 Implementation and Operation: Figure 30 shows how the on-chip I^2C bus interface is implemented, and the following text describes the individual blocks.

Input Filters and Output Stages

The input filters have I^2C compatible input levels. If the input voltage is less than 1.5V, the input logic level is interpreted as 0; if the input voltage is greater than 3.0V, the input logic level is interpreted as 1. Input signals are synchronized with the internal clock (fosc/4), and spikes shorter than three oscillator periods are filtered out.

The output stages consist of open drain transistors that can sink 3mA at $V_{OUT} < 0.4V$. These open drain outputs do not have clamping diodes to V_{DD}. Thus, if the device is connected to the I^2C bus and V_{DD} is switched off, the I^2C bus is not affected.

Address Register, S1ADR

This 8-bit special function register may be loaded with the 7-bit slave address (7 most significant bits) to which SIO1 will respond when programmed as a slave transmitter or receiver. The LSB (GC) is used to enable general call address (00H) recognition.

Comparator

The comparator compares the received 7-bit slave address with its own slave address (7 most significant bits in S1ADR). It also compares the first received 8-bit byte with the general call address (00H). If an equality is found, the appropriate status bits are set and an interrupt is requested.

Shift Register, S1DAT

This 8-bit special function register contains a byte of serial data to be transmitted or a byte which has just been received. Data in S1DAT is always shifted from right to left; the first bit to be transmitted is the MSB (bit 7) and, after a byte has been received, the first bit of received data is located at the MSB of S1DAT. While data is being shifted out, data on the bus is simultaneously being shifted in; S1DAT always contains the last byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in S1DAT.

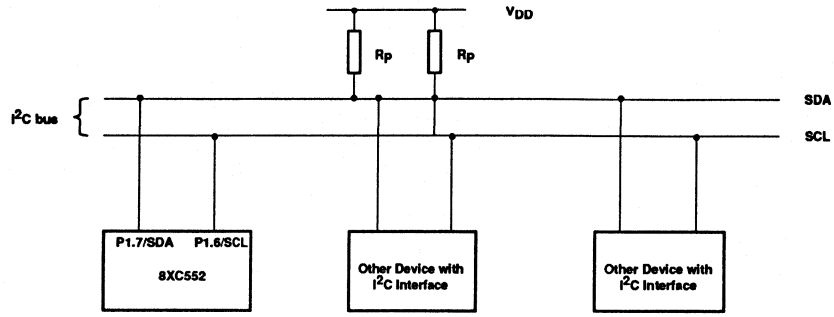


Figure 28. Typical I²C Bus Configuration

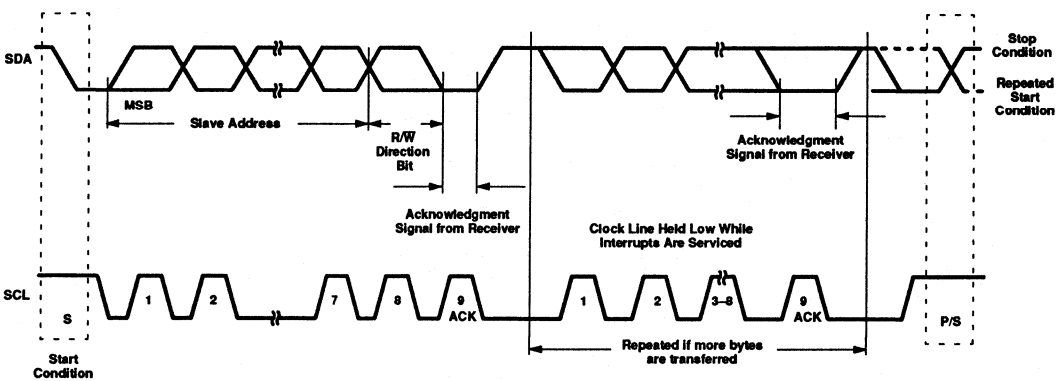


Figure 29. Data Transfer on the I²C Bus

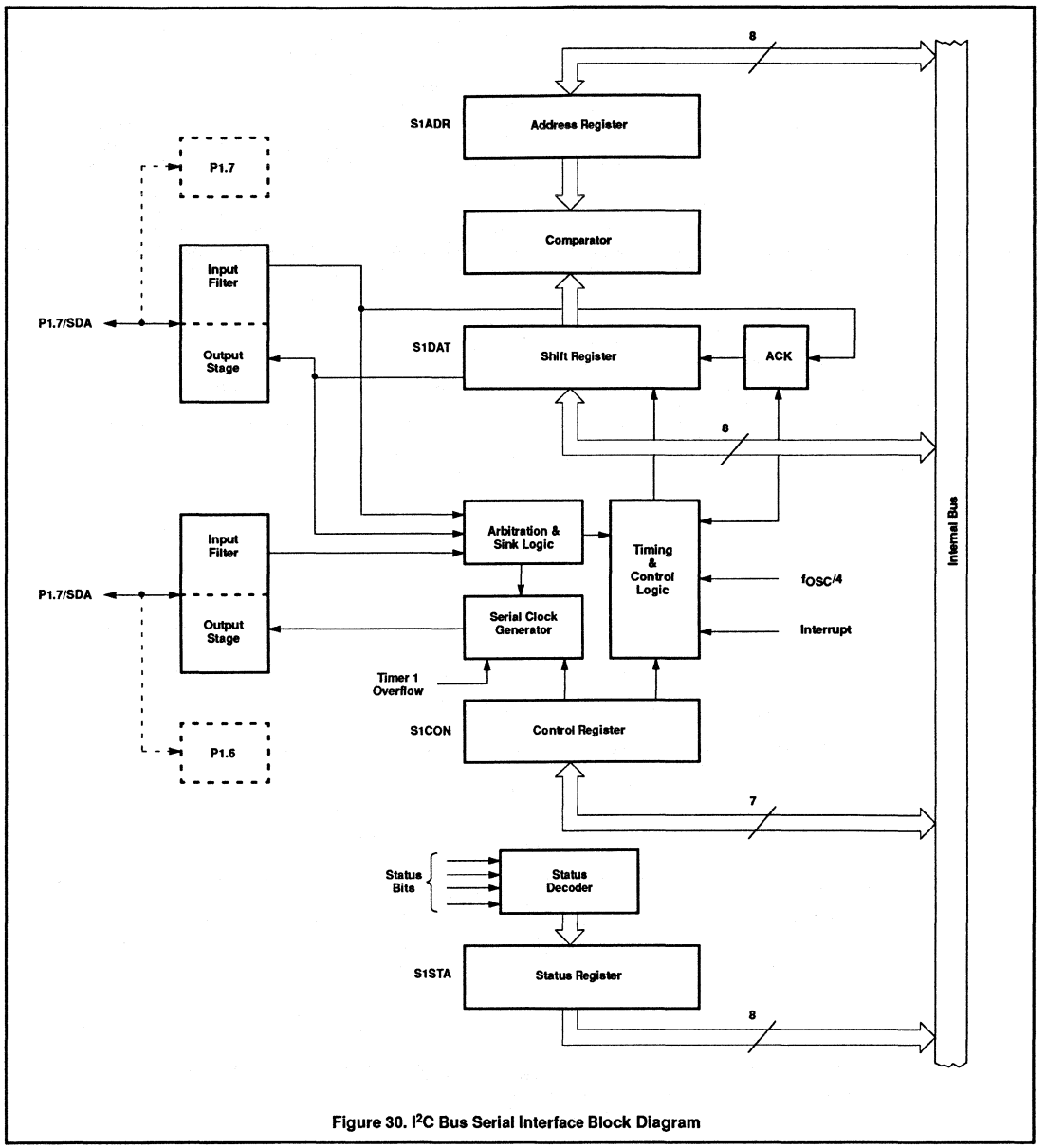
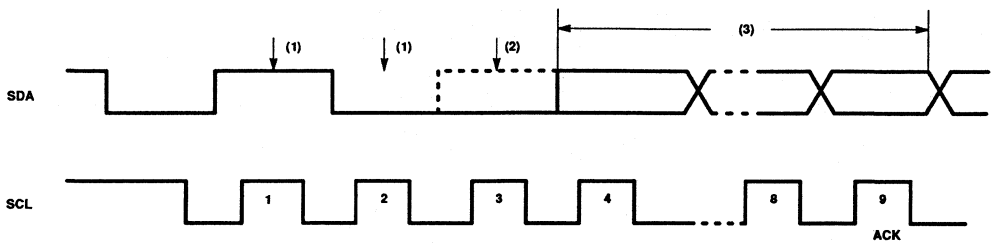
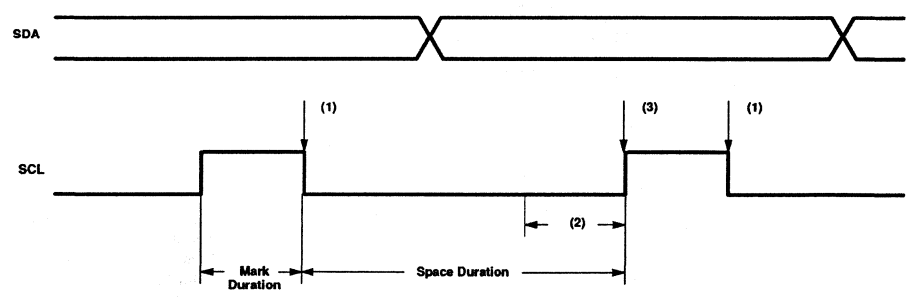


Figure 30. I²C Bus Serial Interface Block Diagram



1. Another device transmits identical serial data.
2. Another device overrules a logic 1 (dotted line) transmitted by SIO1 (master) by pulling the SDA line low. Arbitration is lost, and SIO1 enters the slave receiver mode.
3. SIO1 is in the slave receiver mode but still generates clock pulses until the current byte has been transmitted. SIO1 will not generate clock pulses for the next byte. Data on SDA originates from the new master once it has won arbitration.

Figure 31. Arbitration Procedure



1. Another service pulls the SCL line low before the SIO1 "mark" duration is complete. The serial clock generator is immediately reset and commences with the "space" duration by pulling SCL low.
2. Another device still pulls the SCL line low after SIO1 releases SCL. The serial clock generator is forced into the wait state until the SCL line is released.
3. The SCL line is released, and the serial clock generator commences with the mark duration.

Figure 32. Serial Clock Synchronization

Arbitration and Synchronization Logic

In the master transmitter mode, the arbitration logic checks that every transmitted logic 1 actually appears as a logic 1 on the I^2C bus. If another device on the bus overrules a logic 1 and pulls the SDA line low, arbitration is lost, and SIO1 immediately changes from master transmitter to slave receiver. SIO1 will continue to output clock pulses (on SCL) until transmission of the current serial byte is complete.

Arbitration may also be lost in the master receiver mode. Loss of arbitration in this mode can only occur while SIO1 is returning a "not acknowledge: (logic 1) to the bus. Arbitration is lost when another device on the bus pulls this signal LOW. Since this can occur only at the end of a serial byte, SIO1 generates no further clock pulses. Figure 31 shows the arbitration procedure.

The synchronization logic will synchronize the serial clock generator with the clock pulses on the SCL line from another device. If two or more master devices generate clock pulses, the "mark" duration is determined by the device that generates the shortest "marks," and the "space" duration is determined by the device that generates the longest "spaces." Figure 32 shows the synchronization procedure.

A slave may stretch the space duration to slow down the bus master. The space duration may also be stretched for handshaking purposes. This can be done after each bit or after a complete byte transfer. SIO1 will stretch the SCL space duration after a byte has been transmitted or received and the acknowledge bit has been transferred. The serial interrupt flag (SI) is set, and the stretching continues until the serial interrupt flag is cleared.

Serial Clock Generator

This programmable clock pulse generator provides the SCL clock pulses when SIO1 is in the master transmitter or master receiver mode. It is switched off when SIO1 is in a slave mode. The programmable output clock frequencies are: fosc/120, fosc/9600, and the Timer 1 overflow rate divided by eight. The output clock pulses have a 50% duty cycle unless the clock generator is synchronized with other SCL clock sources as described above.

Timing and Control

The timing and control logic generates the timing and control signals for serial byte handling. This logic block provides the shift pulses for S1DAT, enables the comparator, generates and detects start and stop conditions, receives and transmits acknowledge bits, controls the master and slave modes, contains interrupt request logic, and monitors the I^2C bus status.

Control Register, S1CON

This 7-bit special function register is used by the microcontroller to control the following SIO1 functions: start and restart of a serial transfer, termination of a serial transfer, bit rate, address recognition, and acknowledgment.

Status Decoder and Status Register

The status decoder takes all of the internal status bits and compresses them into a 5-bit code. This code is unique for each I^2C bus status. The 5-bit code may be used to generate vector addresses for fast processing of the various service routines. Each service routine processes a particular bus status. There are 26 possible bus states if all four modes of SIO1 are used. The 5-bit status code is latched into the five most significant bits of the status register when the serial interrupt flag is set (by hardware) and remains stable until the interrupt flag is cleared by software. The three least significant bits of the status register are always zero. If the status code is used as a vector to service routines, then the routines are displaced by eight address locations. Eight bytes of code is sufficient for most of the service routines (see the software example in this section).

More Information on SIO1 Operating Modes: The four operating modes are:

- Master Transmitter
- Master Receiver
- Slave Receiver
- Slave Transmitter

Data transfers in each mode of operation are shown in Figures 33-37. These figures contain the following abbreviations:

Abbreviation	Explanation
S	Start condition
SLA	7-bit slave address
R	Read bit (high level at SDA)
W	Write bit (low level at SDA)
A	Acknowledge bit (low level at SDA)
$\overline{A}$	Not acknowledge bit (high level at SDA)
Data	8-bit data byte
P	Stop condition

In Figures 33-37, circles are used to indicate when the serial interrupt flag is set. The numbers in the circles show the status code held in the S1STA register. At these points, a service routine must be executed to continue or complete the serial transfer. These service routines are not critical since the serial transfer is suspended until the serial interrupt flag is cleared by software.

When a serial interrupt routine is entered, the status code in S1STA is used to branch to the appropriate service routine. For each status code, the required software action and details of the following serial transfer are given in Tables 15-18.

Master Transmitter Mode: In the master transmitter mode, a number of data bytes are transmitted to a slave receiver (see Figure 33). Before the master transmitter mode can be entered, S1CON must be initialized as follows:

	7	6	5	4	3	2	1	0
S1CON (D8H)	CR2	ENS1	STA	STO	SI	AA	CR1	CR0
	X	1	0	0	0	X	bit rate	

CR0, CR1, and CR2 define the serial bit rate. ENS1 must be set to logic 1 to enable SIO1. If the AA bit is reset, SIO1 will not acknowledge its own slave address or the general call address in the event of another device becoming master of the bus. In other words, if AA is reset, SIO0 cannot enter a slave mode. STA, STO, and SI must be reset.

The master transmitter mode may now be entered by setting the STA bit using the SETB instruction. The SIO1 logic will now test the I^2C bus and generate a start condition as soon as the bus becomes free. When a START condition is transmitted, the serial interrupt flag (SI) is set, and the status code in the status register (S1STA) will be 08H. This status code must be used to vector to an interrupt service routine that loads S1DAT with the slave address and the data direction bit (SLA+W). The SI bit in S1CON must then be reset before the serial transfer can continue.

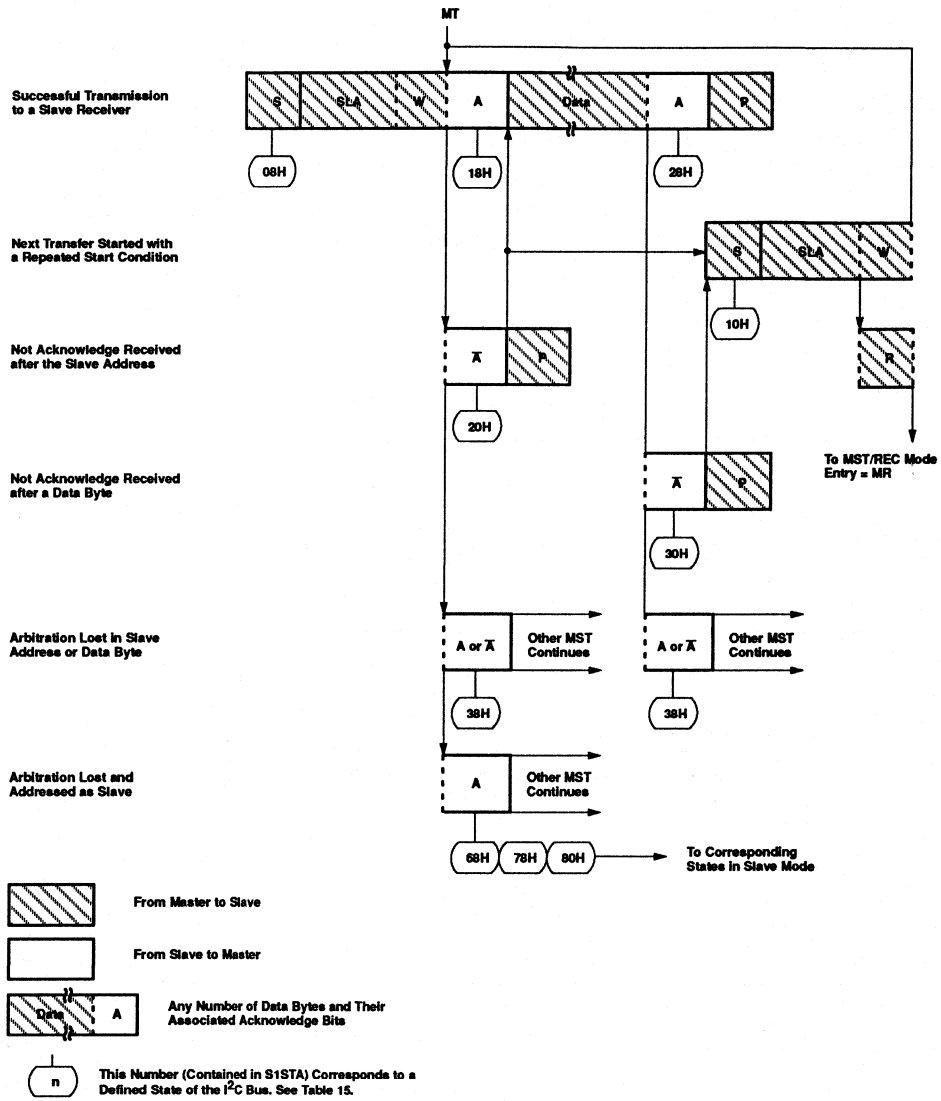


Figure 33. Format and States in the Master Transmitter Mode

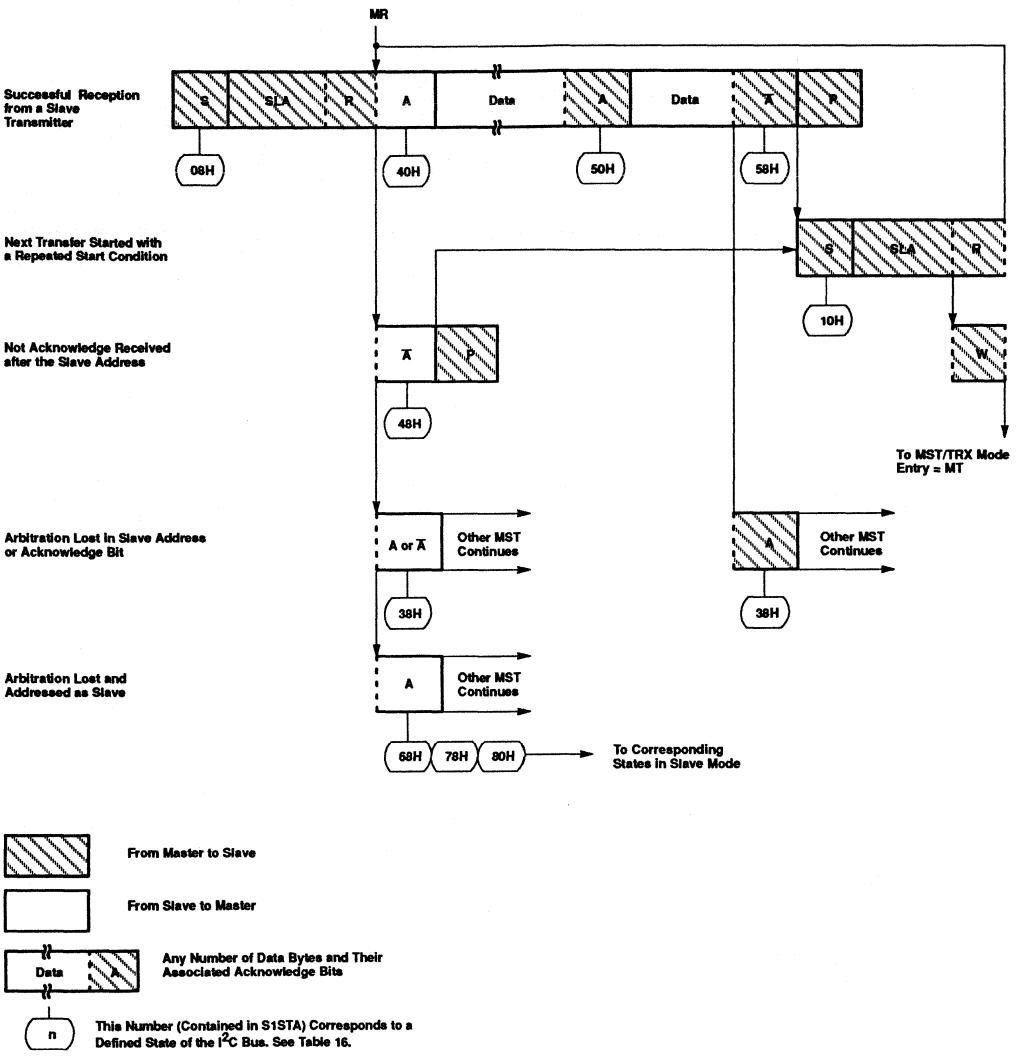


Figure 34. Format and States in the Master Receiver Mode

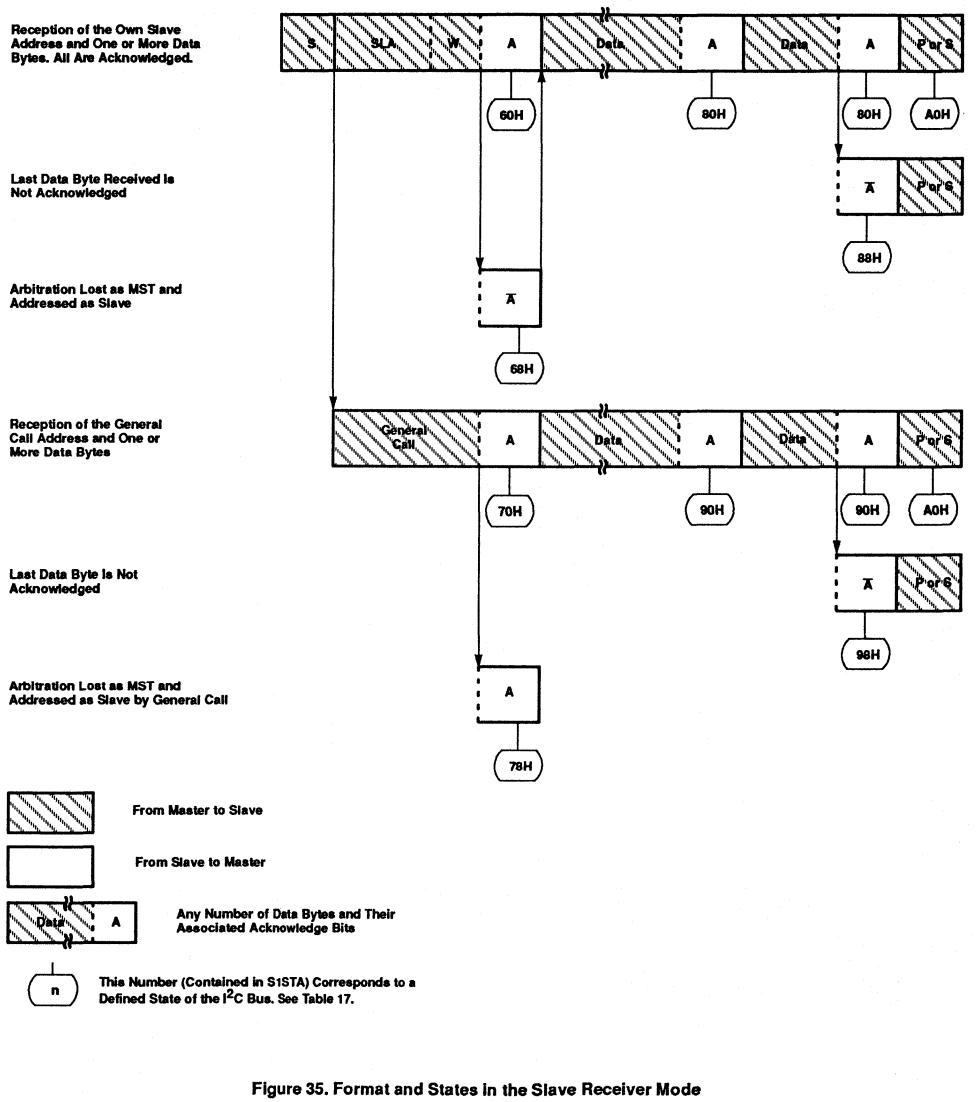


Figure 35. Format and States in the Slave Receiver Mode

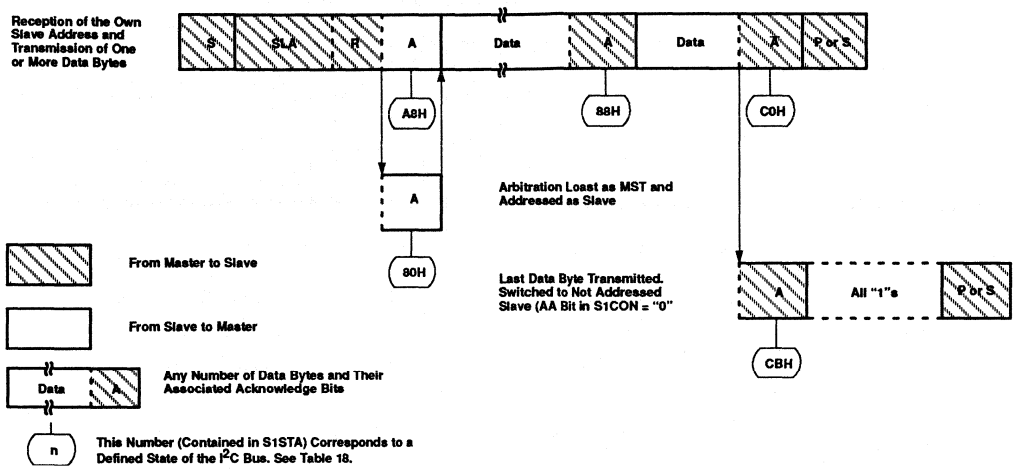


Figure 36. Format and States of the Slave Transmitter Mode

When the slave address and the direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in S1STA are possible. There are 18H, 20H, or 38H for the master mode and also 68H, 78H, or B0H if the slave mode was enabled (AA = logic 1). The appropriate action to be taken for each of these status codes is detailed in Table 15. After a repeated start condition (state 10H). SIO1 may switch to the master receiver mode by loading S1DAT with SLA+R).

Master Receiver Mode: In the master receiver mode, a number of data bytes are received from a slave transmitter (see Figure 34). The transfer is initialized as in the master transmitter mode. When the start condition has been transmitted, the interrupt service routine must load S1DAT with the 7-bit slave address and the data direction bit (SLA+R). The SI bit in S1CON must then be cleared before the serial transfer can continue.

When the slave address and the data direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in S1STA are possible. These are 40H, 48H, or 38H for the master mode and also 68H, 78H, or B0H if the slave mode was enabled (AA = logic 1). The appropriate action to be taken for each of these status codes is detailed in Table 16. ENS1, CR1, and CR0 are not affected by the serial transfer and are not referred to in Table 16. After a repeated start condition (state 10H), SIO1 may switch to the master transmitter mode by loading S1DAT with SLA+W.

Slave Receiver Mode: In the slave receiver mode, a number of data bytes are received from a master transmitter (see Figure 35). To initiate the slave receiver mode, S1ADR and S1CON must be loaded as follows:

	7	6	5	4	3	2	1	0
S1ADR (DBH)	X	X	X	X	X	X	X	GC
	own slave address							

The upper 7 bits are the address to which SIO1 will respond when addressed by a master. If the LSB (GC) is set, SIO1 will respond to the general call address (00H); otherwise it ignores the general call address.

	7	6	5	4	3	2	1	0
S1CON (D8H)	CR2	ENS1	STA	STO	SI	AA	CR1	CR0
	X	1	0	0	0	1	X	X

CR0, CR1, and CR12 do not affect SIO1 in the slave mode. ENS1 must be set to logic 1 to enable SIO1. The AA bit must be set to enable SIO1 to acknowledge its own slave address or the general call address. STA, STO, and SI must be reset.

When S1ADR and S1CON have been initialized, SIO1 waits until it is addressed by its own slave address followed by the data direction bit which must be "0" (W) for SIO1 to operate in the slave receiver mode. After its own slave address and the W bit have been received, the serial interrupt flag (I) is set and a valid status code can be read from S1STA. This status code is used to vector to an interrupt service routine, and the appropriate action to be taken for each of these status codes is detailed in Table 17. The slave receiver mode may also be entered if arbitration is lost while SIO1 is in the master mode (see status 68H and 78H).

If the AA bit is reset during a transfer, SIO1 will return a not acknowledge (logic 1) to SDA after the next received data byte. While AA is reset, SIO1 does not respond to its own slave address or a general call address. However, the I²C bus is still monitored and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate SIO1 from the I²C bus.

Table 15. Master Transmitter Mode

STATUS CODE (S1STA)	STATUS OF THE I²C BUS AND SIO1 HARDWARE	APPLICATION SOFTWARE RESPONSE					NEXT ACTION TAKEN BY SIO1 HARDWARE
		TO/FROM S1DAT	TO S1CON				
			STA	STO	SI	AA	
08H	A START condition has been transmitted	Load SLA+W	X	0	0	X	SLA+W will be transmitted; ACK bit will be received
10H	A repeated START condition has been transmitted	Load SLA+W or Load SLA+R	X X	0 0	0 0	X X	As above SLA+W will be transmitted; SIO1 will be switched to MST/REC mode
18H	SLA+W has been transmitted; ACK has been received	Load data byte or no S1DAT action or no S1DAT action or no S1DAT action	0 1 0 1	0 0 1 1	0 0 0 0	X X X X	Data byte will be transmitted; ACK bit will be received Repeated START will be transmitted; STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset
20H	SLA+W has been transmitted; NOT ACK has been received	Load data byte or no S1DAT action or no S1DAT action or no S1DAT action	0 1 0 1	0 0 1 1	0 0 0 0	X X X X	Data byte will be transmitted; ACK bit will be received Repeated START will be transmitted; STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset
28H	Data byte in S1DAT has been transmitted; ACK has been received	Load data byte or no S1DAT action or no S1DAT action or no S1DAT action	0 1 0 1	0 0 1 1	0 0 0 0	X X X X	Data byte will be transmitted; ACK bit will be received Repeated START will be transmitted; STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset
30H	Data byte in S1DAT has been transmitted; NOT ACK has been received	Load data byte or no S1DAT action or no S1DAT action or no S1DAT action	0 1 0 1	0 0 1 1	0 0 0 0	X X X X	Data byte will be transmitted; ACK bit will be received Repeated START will be transmitted; STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset
38H	Arbitration lost in SLA+R/W or Data bytes	No S1DAT action or No S1DAT action	0 1	0 0	0 0	X X	I²C bus will be released; not addressed slave will be entered A START condition will be transmitted when the bus becomes free

Table 16. Master Receiver Mode

STATUS CODE (S1STA)	STATUS OF THE I²C BUS AND SIO1 HARDWARE	APPLICATION SOFTWARE RESPONSE					NEXT ACTION TAKEN BY SIO1 HARDWARE
		TO/FROM S1DAT	TO S1CON				
			STA	STO	SI	AA	
08H	A START condition has been transmitted	Load SLA+R	X	0	0	X	SLA+R will be transmitted; ACK bit will be received
10H	A repeated START condition has been transmitted	Load SLA+R or Load SLA+W	X X	0 0	0 0	X X	As above SLA+W will be transmitted; SIO1 will be switched to MST/TRX mode
38H	Arbitration lost in NOT ACK bit	No S1DAT action or No S1DAT action	0 1	0 0	0 0	X X	I²C bus will be released; SIO1 will enter a slave mode A START condition will be transmitted when the bus becomes free
40H	SLA+R has been transmitted; ACK has been received	No S1DAT action or no S1DAT action	0 0	0 0	0 0	0 1	Data byte will be received; NOT ACK bit will be returned Data byte will be received; ACK bit will be returned
48H	SLA+R has been transmitted; NOT ACK has been received	No S1DAT action or no S1DAT action or no S1DAT action	1 0 1	0 1 1	0 0 0	X X X	Repeated START condition will be transmitted STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset
50H	Data byte has been transmitted; ACK has been returned	Read data byte or read data byte	0 0	0 0	0 0	0 1	Data byte will be received; NOT ACK bit will be returned Data byte will be received; ACK bit will be returned
58H	Data byte has been received; NOT ACK has been returned	Read data byte or read data byte or read data byte	1 0 1	0 1 1	0 0 0	X X X	Repeated START condition will be transmitted STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset

Table 17. Slave Receiver Mode

STATUS CODE (S1STA)	STATUS OF THE I²C BUS AND SIO1 HARDWARE	APPLICATION SOFTWARE RESPONSE					NEXT ACTION TAKEN BY SIO1 HARDWARE
		TO/FROM S1DAT	TO S1CON				
			STA	STO	SI	AA	
60H	Own SLA+W has been received; ACK has been returned	No S1DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned
		no S1DAT action	X	0	0	1	Data byte will be received and ACK will be returned
68H	Arbitration lost in SLA+R/W as master; Own SLA+W has been received, ACK returned	No S1DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned
		no S1DAT action	X	0	0	1	Data byte will be received and ACK will be returned
70H	General call address (00H) has been received; ACK has been returned	No S1DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned
		no S1DAT action	X	0	0	1	Data byte will be received and ACK will be returned
78H	Arbitration lost in SLA+R/W as master; General call address has been received, ACK has been returned	No S1DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned
		no S1DAT action	X	0	0	1	Data byte will be received and ACK will be returned
80H	Previously addressed with own SLV address; DATA has been received; ACK has been returned	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned
		read data byte	X	0	0	1	Data byte will be received and ACK will be returned
88H	Previously addressed with own SLA; DATA byte has been received; NOT ACK has been returned	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address
		read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1
		read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free
		read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.
90H	Previously addressed with General Call; DATA byte has been received; ACK has been returned	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned
		read data byte	X	0	0	1	Data byte will be received and ACK will be returned
98H	Previously addressed with General Call; DATA byte has been received; NOT ACK has been returned	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address
		read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1
		read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free
		read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.

Table 17. Slave Receiver Mode (Continued)

STATUS CODE (S1STA)	STATUS OF THE I²C BUS AND SIO1 HARDWARE	APPLICATION SOFTWARE RESPONSE					NEXT ACTION TAKEN BY SIO1 HARDWARE
		TO/FROM S1DAT	TO S1CON				
			STA	STO	SI	AA	
A0H	A STOP condition or repeated START condition has been received while still addressed as SLV/REC or SLV/TRX	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address
		read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1
		read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free
		read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.

Table 18. Slave Transmitter Mode

STATUS CODE (S1STA)	STATUS OF THE I²C BUS AND SIO1 HARDWARE	APPLICATION SOFTWARE RESPONSE					NEXT ACTION TAKEN BY SIO1 HARDWARE
		TO/FROM S1DAT	TO S1CON				
			STA	STO	SI	AA	
A8H	Own SLA+R has been received; ACK has been returned	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received
		load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received
B0H	Arbitration lost in SLA+R/$\overline{W}$ as master; Own SLA+R has been received, ACK has been returned	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received
		load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received
B8H	Data byte in S1DAT has been transmitted; ACK has been received	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received
		load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received
C0H	Data byte in S1DAT has been transmitted; NOT ACK has been received	No S1DAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address
		no S1DAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1
		no S1DAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free
		no S1DAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.
C8H	Last data byte in S1DAT has been transmitted (AA = 0); ACK has been received	No S1DAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address
		no S1DAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1
		no S1DAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free
		no S1DAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.

Slave Transmitter Mode: In the slave transmitter mode, a number of data bytes are transmitted to a master receiver (see Figure 36). Data transfer is initialized as in the slave receiver mode. When S1ADR and S1CON have been initialized, SIO1 waits until it is addressed by its own slave address followed by the data direction bit which must be "1" (R) for SIO1 to operate in the slave transmitter mode. After its own slave address and the R bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from S1STA. This status code is used to vector to an interrupt service routine, and the appropriate action to be taken for each of these status codes is detailed in Table 18. The slave transmitter mode may also be entered if arbitration is lost while SIO1 is in the master mode (see state B0H).

If the AA bit is reset during a transfer, SIO1 will transmit the last byte of the transfer and enter state C0H or C8H. SIO1 is switched to the not addressed slave mode and will ignore the master receiver if it continues the transfer. Thus the master receiver receives all 1s as serial data. While AA is reset, SIO1 does not respond to its own slave address or a general call address. However, the I^2C bus is still monitored, and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate SIO1 from the I^2C bus.

Miscellaneous States: There are two S1STA codes that do not correspond to a defined SIO1 hardware state (see Table 19). These are discussed below.

S1STA = F8H:

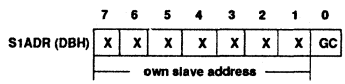
This status code indicates that no relevant information is available because the serial interrupt flag, SI, is not yet set. This occurs between other states and when SIO1 is not involved in a serial transfer.

S1STA = 00H:

This status code indicates that a bus error has occurred during an SIO1 serial transfer. A bus error is caused when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. A bus error may also be caused when external interference disturbs the internal SIO1 signals. When a bus error occurs, SI is set. To recover from a bus error, the STO flag must be set and SI must be cleared. This causes SIO1 to enter the "not addressed" slave mode (a defined state) and to clear the STO flag (no other bits in S1CON are affected). The SDA and SCL lines are released (a STOP condition is not transmitted).

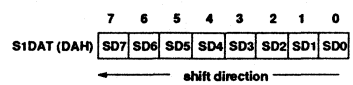
The Four SIO1 Special Function Registers: The microcontroller interfaces to SIO1 via four special function registers. These four SFRs (S1ADR, S1DAT, S1CON, and S1STA) are described individually in the following sections.

The Address Register, S1ADR: The CPU can read from and write to this 8-bit, directly addressable SFR. S1ADR is not affected by the SIO1 hardware. The contents of this register are irrelevant when SIO1 is in a master mode. In the slave modes, the seven most significant bits must be loaded with the microcontroller's own slave address, and, if the least significant bit is set, the general call address (00H) is recognized; otherwise it is ignored.



The most significant bit corresponds to the first bit received from the I^2C bus after a start condition. A logic 1 in S1ADR corresponds to a high level on the I^2C bus, and a logic 0 corresponds to a low level on the bus.

The Data Register, S1DAT: S1DAT contains a byte of serial data to be transmitted or a byte which has just been received. The CPU can read from and write to this 8-bit, directly addressable SFR while it is not in the process of shifting a byte. This occurs when SIO1 is in a defined state and the serial interrupt flag is set. Data in S1DAT remains stable as long as SI is set. Data in S1DAT is always shifted from right to left: the first bit to be transmitted is the MSB (bit 7), and, after a byte has been received, the first bit of received data is located at the MSB of S1DAT. While data is being shifted out, data on the bus is simultaneously being shifted in; S1DAT always contains the last data byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in S1DAT.



SD7 - SD0:

Eight bits to be transmitted or just received. A logic 1 in S1DAT corresponds to a high level on the I^2C bus, and a logic 0 corresponds to a low level on the bus. Serial data shifts through S1DAT from right to left. Figure 37 shows how data in S1DAT is serially transferred to and from the SDA line.

Table 19. Miscellaneous States

STATUS CODE (S1STA)	STATUS OF THE I^2C BUS AND SIO1 HARDWARE	APPLICATION SOFTWARE RESPONSE					NEXT ACTION TAKEN BY SIO1 HARDWARE
		TO/FROM S1DAT	TO S1CON				
			STA	STO	SI	AA	
F8H	No relevant state information available; S1 = 0	No S1DAT action	No S1CON action				Wait or proceed current transfer
00H	Bus error during MST or selected slave modes, due to an illegal START or STOP condition. State 00H can also occur when interference causes SIO1 to enter an undefined state.	No S1DAT action	0	1	0	X	Only the internal hardware is affected in the MST or addressed SLV modes. In all cases, the bus is released and SIO1 is switched to the not addressed SLV mode. STO is reset.

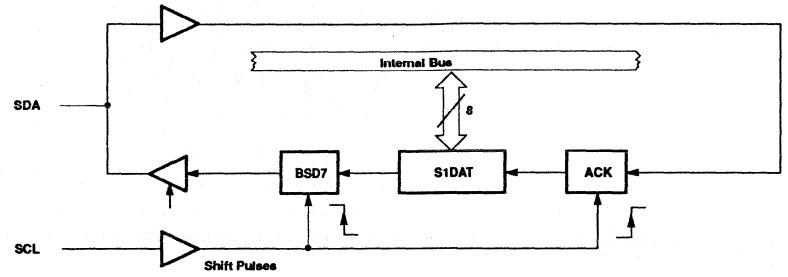


Figure 37. Serial Input/Output Configuration

S1DAT and the ACK flag form a 9-bit shift register which shifts in or shifts out an 8-bit byte, followed by an acknowledge bit. The ACK flag is controlled by the SIO1 hardware and cannot be accessed by the CPU. Serial data is shifted through the ACK flag into S1DAT on the rising edges of serial clock pulses on the SCL line. When a byte has been shifted into S1DAT, the serial data is available in S1DAT, and the acknowledge bit is returned by the control logic during the ninth clock pulse. Serial data is shifted out from S1DAT via a buffer (BSD7) on the falling edges of clock pulses on the SCL line.

When the CPU writes to S1DAT, BSD7 is loaded with the content of S1DAT.7, which is the first bit to be transmitted to the SDA line (see Figure 38). After nine serial clock pulses, the eight bits in S1DAT will have been transmitted to the SDA line, and the acknowledge bit will be present in ACK. Note that the eight transmitted bits are shifted back into S1DAT.

The Control Register, S1CON: The CPU can read from and write to this 8-bit, directly addressable SFR. The most significant bit is not used, and a logic 1 will be returned if S1CON.7 is read. Two bits are affected by the SIO1 hardware: the SI bit is set when a serial interrupt is requested, and the STO bit is cleared when a STOP condition is present on the I²C bus. The STO bit is also cleared when ENS1 = "0".

	7	6	5	4	3	2	1	0
S1CON (D8H)	CR2	ENS1	STA	STO	SI	AA	CR1	CR0

ENS1, the SIO1 Enable Bit

ENS1 = "0": When ENS1 is "0", the SDA and SCL outputs are in a high impedance state. SDA and SCL input signals are ignored, SIO1 is in the "not addressed" slave state, and the STO bit in S1CON is forced to "0". No other bits are affected. P1.6 and P1.7 may be used as open drain I/O ports.

ENS1 = "1": When ENS1 is "1", SIO1 is enabled. The P1.6 and P1.7 port latches must be set to logic 1.

ENS1 should not be used to temporarily release SIO1 from the I2C bus since, when ENS1 is reset, the I2C bus status is lost. The AA flag should be used instead (see description of the AA flag in the following text).

In the following text, it is assumed that ENS1 = "1".

STA, the START Flag

STA = "1": When the STA bit is set to enter a master mode, the SIO1 hardware checks the status of the I2C bus and generates a START condition if the bus is free. If the bus is not free, then SIO1 waits for a STOP condition (which will free the bus) and generates a START condition after a delay of a half clock period of the internal serial clock generator.

If STA is set while SIO1 is already in a master mode and one or more bytes are transmitted or received, SIO1 transmits a repeated START condition. STA may be set at any time. STA may also be set when SIO1 is an addressed slave.

STA = "0": When the STA bit is reset, no START condition or repeated START condition will be generated.

STO, the STOP Flag

STO = "1": When the STO bit is set while SIO1 is in a master mode, a STOP condition is transmitted to the I²C bus. When the STOP condition is detected on the bus, the SIO1 hardware clears the STO flag. In a slave mode, the STO flag may be set to recover from an error condition. In this case, no STOP condition is transmitted to the I²C bus. However, the SIO1 hardware behaves as if a STOP condition has been received and switches to the defined "not addressed" slave receiver mode. The STO flag is automatically cleared by hardware.

If the STA and STO bits are both set, the a STOP condition is transmitted to the I²C bus if SIO1 is in a master mode (in a slave mode, SIO1 generates an internal STOP condition which is not transmitted). SIO1 then transmits a START condition.

STO = "0": When the STO bit is reset, no STOP condition will be generated.

SI, the Serial Interrupt Flag

SI = "1": When the SI flag is set, then, if the EA and ES1 (interrupt enable register) bits are also set, a serial interrupt is requested. SI is set by hardware when one of 25 of the 26 possible SIO1 states is entered. The only state that does not cause SI to be set is state F8H, which indicates that no relevant state information is available.

While SI is set, the low period of the serial clock on the SCL line is stretched, and the serial transfer is suspended. A high level on the SCL line is unaffected by the serial interrupt flag. SI must be reset by software.

SI = "0": When the SI flag is reset, no serial interrupt is requested, and there is no stretching of the serial clock on the SCL line.

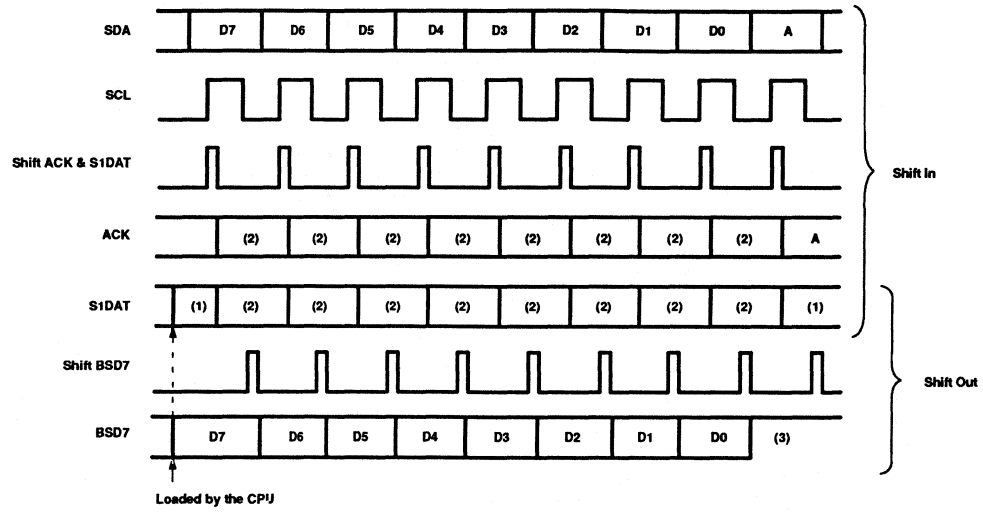


Figure 38. Shift-In and Shift-out Timing

AA, the Assert Acknowledge Flag

AA = "1": If the AA flag is set, an acknowledge (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line when:

- The "own slave address" has been received
- The general call address has been received while the general call bit (GC) in S1ADR is set
- A data byte has been received while SIO1 is in the master receiver mode
- A data byte has been received while SIO1 is in the addressed slave receiver mode

AA = "0": if the AA flag is reset, a not acknowledge (high level to SDA) will be returned during the acknowledge clock pulse on SCL when:

- A data has been received while SIO1 is in the master receiver mode
- A data byte has been received while SIO1 is in the addressed slave receiver mode

When WIO1 is in the addressed slave transmitter mode, state C8H will be entered after the last serial is transmitted (see Figure 36). When SI is cleared, SIO1 leaves state C8H, enters the not addressed slave receiver mode, and the SDA line remains at a high level. In state C8H, the AA flag can be set again for future address recognition.

When SIO1 is in the not addressed slave mode, its own slave address and the general call address are ignored. Consequently, no acknowledge is returned, and a serial interrupt is not requested. Thus, SIO1 can be temporarily released from the I^2C bus while the bus status is monitored. While SIO1 is released from the bus, START and STOP conditions are detected, and serial data is shifted in. Address recognition can be resumed at any time by setting the AA flag. If the AA flag is set when the part's own slave address or the general call address has been partly received, the address will be recognized at the end of the byte transmission.

CR0, CR1, and CR2, the Clock Rate Bits

These three bits determine the serial clock frequency when SIO1 is in a master mode. The various serial rates are shown in Table 20.

Table 20. Serial Clock Rates

CR2	CR1	CR0	BIT FREQUENCY (kHz) AT f_{OSC}			f_{OSC} DIVIDED BY
			6MHz	12MHz	16MHz	
0	0	0	23	47	63	256
0	0	1	27	54	71	224
0	1	0	31	63	83	192
0	1	1	37	75	100	160
1	0	0	6.25	12.5	17	960
1	0	1	50	100	–	120
1	1	0	100	–	–	60
1	1	1	0.25 < 62.5	0.5 < 62.5	0.67 < 56	96 x (256 – reload value Timer 1) (Reload value range: 0 – 254 in mode 2)

A 12.5kHz bit rate may be used by devices that interface to the I²C bus via standard I/O port lines which are software driven and slow. 100kHz is usually the maximum bit rate and can be derived from a 16MHz, 12MHz, or a 6MHz oscillator. A variable bit rate (0.5kHz to 62.5kHz) may also be used if Timer 1 is not required for any other purpose while SIO1 is in a master mode.

The frequencies shown in Table 20 are unimportant when SIO1 is in a slave mode. In the slave modes, SIO1 will automatically synchronize with any clock frequency up to 100kHz.

The Status Register, S1STA: S1STA is an 8-bit read-only special function register. The three least significant bits are always zero. The five most significant bits contain the status code. There are 26 possible status codes. When S1STA contains F8H, no relevant state information is available and no serial interrupt is requested. All other S1STA values correspond to defined SIO1 states. When each of these states is entered, a serial interrupt is requested (SI = "1"). A valid status code is present in S1STA one machine cycle after SI is set by hardware and is still present one machine cycle after SI has been reset by software.

Some Special Cases: The SIO1 hardware has facilities to handle the following special cases that may occur during a serial transfer:

Simultaneous Repeated START Conditions from Two Masters

A repeated START condition may be generated in the master transmitter or master receiver modes. A special case occurs if another master simultaneously generates a repeated START condition (see Figure 39). Until this occurs, arbitration is not lost by either master since they were both transmitting the same data.

If the SIO1 hardware detects a repeated START condition on the I²C bus before generating a repeated START condition itself, it will release the bus, and no interrupt request is generated. If another master frees the bus by generating a STOP condition, SIO1 will transmit a normal START condition (state 08H), and a retry of the total serial data transfer can commence.

Data Transfer After Loss of Arbitration

Arbitration may be lost in the master transmitter and master receiver modes (see Figure 31). Loss of arbitration is indicated by the following states in S1STA; 38H, 68H, 78H, and B0H (see Figures 33 and 34).

If the STA flag in S1CON is set by the routines which service these states, then, if the bus is free again, a START condition (state 08H) is transmitted without intervention by the CPU, and a retry of the total serial transfer can commence.

Forced Access to the I²C Bus

In some applications, it may be possible for an uncontrolled source to cause a bus hang-up. In such situations, the problem may be caused by interference, temporary interruption of the bus or a temporary short-circuit between SDA and SCL.

If an uncontrolled source generates a superfluous START or masks a STOP condition, then the I²C bus stays busy indefinitely. If the STA flag is set and bus access is not obtained within a reasonable amount of time, then a forced access to the I²C bus is possible. This is achieved by setting the STO flag while the STA flag is still set. No STOP condition is transmitted. The SIO1 hardware behaves as if a STOP condition was received and is able to transmit a START condition. The STO flag is cleared by hardware (see Figure 40).

I²C Bus Obstructed by a Low Level on SCL or SDA

An I²C bus hang-up occurs if SDA or SCL is pulled LOW by an uncontrolled source. If the SCL line is obstructed (pulled LOW) by a device on the bus, no further serial transfer is possible, and the SIO1 hardware cannot resolve this type of problem. When this occurs, the problem must be resolved by the device that is pulling the SCL bus line LOW.

If the SDA line is obstructed by another device on the bus (e.g., a slave device out of bit synchronization), the problem can be solved by transmitting additional clock pulses on the SCL line (see Figure 41). The SIO1 hardware transmits additional clock pulses when the STA flag is set, but no START condition can be generated because the SDA line is pulled LOW while the I²C bus is considered free. The SIO1 hardware attempts to generate a START condition after every two additional clock pulses on the SCL line. When the SDA line is eventually released, a normal START condition is transmitted, state 08H is entered, and the serial transfer continues.

If a forced bus access occurs or a repeated START condition is transmitted while SDA is obstructed (pulled LOW), the SIO1 hardware performs the same action as described above. In each case, state 08H is entered after a successful START condition is transmitted and normal serial transfer continues. Note that the CPU is not involved in solving these bus hang-up problems.

Bus Error

A bus error occurs when a START or STOP condition is present at an illegal position in the format frame. Examples of illegal positions are during the serial transfer of an address byte, a data or an acknowledge bit.

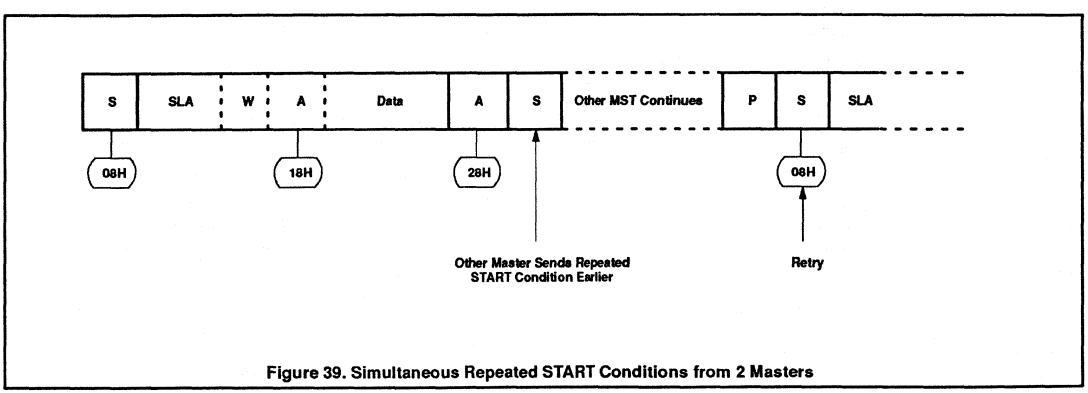


Figure 39. Simultaneous Repeated START Conditions from 2 Masters

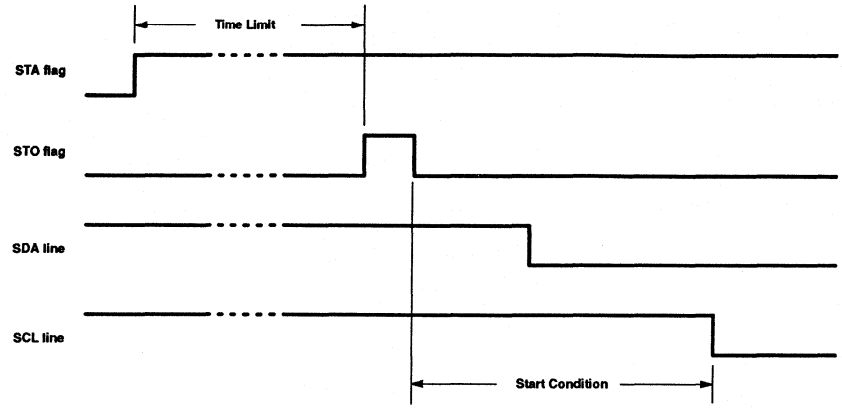
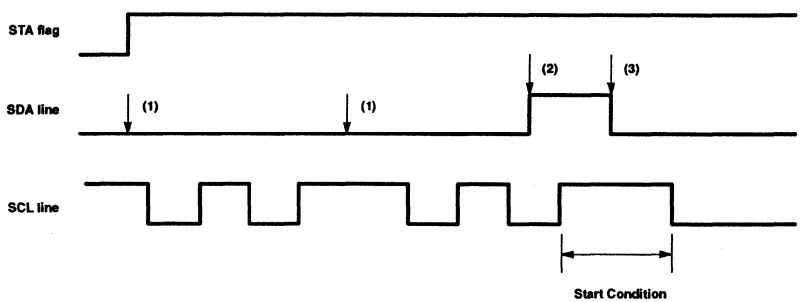


Figure 40. Forced Access to a Busy I²C Bus



(1) Unsuccessful attempt to send a Start condition
(2) SDA line released
(3) Succesful attempt to send a Start condition; state 08H is entered

Figure 41. Recovering from a Bus Obstruction Caused by a Low Level on SDA

The SIO1 hardware only reacts to a bus error when it is involved in a serial transfer either as a master or an addressed slave. When a bus error is detected, SIO1 immediately switches to the not addressed slave mode, releases the SDA and SCL lines, sets the interrupt flag, and loads the status register with 00H. This status code may be used to vector to a service routine which either attempts the aborted serial transfer again or simply recovers from the error condition as shown in Table 19.

Software Examples of SIO1 Service Routines: This section consists of a software example for:

- Initialization of SIO1 after a RESET
- Entering the SIO1 interrupt routine
- The 26 state service routines for the
  - Master transmitter mode
  - Master receiver mode
  - Slave receiver mode
  - Slave transmitter mode

Initialization

In the initialization routine, SIO1 is enabled for both master and slave modes. For each mode, a number of bytes of internal data RAM are allocated to the SIO to act as either

a transmission or reception buffer. In this example, 8 bytes of internal data RAM are reserved for different purposes. The data memory map is shown in Figure 42. The initialization routine performs the following functions:

- S1ADR is loaded with the part's own slave address and the general call bit (GC)
- P1.6 and P1.7 bit latches are loaded with logic 1s
- RAM location HADD is loaded with the high-order address byte of the service routines
- The SIO1 interrupt enable and interrupt priority bits are set
- The slave mode is enabled by simultaneously setting the ENS1 and AA bits in S1CON and the serial clock frequency (for master modes) is defined by loading CR0 and CR1 in S1CON. The master routines must be started in the main program.

The SIO1 hardware now begins checking the I²C bus for its own slave address and general call. If the general call or the own slave address is detected, an interrupt is requested and S1STA is loaded with the appropriate state information. The following text describes a fast method of branching to the appropriate service routine.

SIO1 Interrupt Routine

When the SIO1 interrupt is entered, the PSW is first pushed on the stack. Then S1STA and HADD (loaded with the high-order address byte of the 26 service routines by the initialization routine) are pushed on to the stack. S1STA contains a status code which is the lower byte of one of the 26 service routines. The next instruction is RET, which is the return from subroutine instruction. When this instruction is executed, the high and low order address bytes are popped from stack and loaded into the program counter.

The next instruction to be executed is the first instruction of the state service routine. Seven bytes of program code (which execute in eight machine cycles) are required to branch to one of the 26 state service routines.

```
SI      PUSH   PSW     Save PSW
        PUSH   S1STA   Push status code
                       (low order address
                       byte)
        PUSH   HADD    Push high order
                       address byte
        RET            Jump to state
                       service routine
```

The state service routines are located in a 256-byte page of program memory. The location of this page is defined in the initialization routine. The page can be located anywhere in program memory by loading data RAM register HADD with the page number. Page 01 is chosen in this example, and the service routines are located between addresses 0100H and 01FFH.

The State Service Routines

The state service routines are located 8 bytes from each other. Eight bytes of code are sufficient for most of the service routines. A few of the routines require more than 8 bytes and have to jump to other locations to obtain more bytes of code. Each state routine is part of the SIO1 interrupt routine and handles one of the 26 states. It ends with a RETI instruction which causes a return to the main program.

Master Transmitter and Master Receiver Modes

The master mode is entered in the main program. To enter the master transmitter mode, the main program must first load the internal data RAM with the slave address, data bytes, and the number of data bytes to be transmitted. To enter the master receiver mode, the main program must first load the internal data RAM with the slave address and the number of data bytes to be received. The R/$\overline{W}$ bit determines whether SIO1 operates in the master transmitter or master receiver mode.

Master mode operation commences when the STA bit in S1CION is set by the SETB instruction and data transfer is controlled by the master state service routines in accordance with Table 15, Table 16, Figure 33, and Figure 34. In the example below, 4 bytes are transferred. There is no repeated START condition. In the event of lost arbitration, the transfer is restarted when the bus becomes free. If a bus error occurs, the I²C bus is released and SIO1 enters the not selected slave receiver mode. If a slave device returns a not acknowledge, a STOP condition is generated.

A repeated START condition can be included in the serial transfer if the STA flag is set instead of the STO flag in the state service routines vectored to by status codes 28H and 58H. Additional software must be written to determine which data is transferred after a repeated START condition.

Slave Transmitter and Slave Receiver Modes

After initialization, SIO1 continually tests the I²C bus and branches to one of the slave state service routines if it detects its own slave address or the general call address (see Table 17, Table 18, Figure 35, and Figure 36). If arbitration was lost while in the master mode, the master mode is restarted after the current transfer. If a bus error occurs, the I²C bus is released and SIO1 enters the not selected slave receiver mode.

In the slave receiver mode, a maximum of 8 received data bytes can be stored in the internal data RAM. A maximum of 8 bytes ensures that other RAM locations are not overwritten if a master sends more bytes. If more than 8 bytes are transmitted, a not acknowledge is returned, and SIO1 enters the not addressed slave receiver mode. A maximum of one received data byte can be stored in the internal data RAM after a general call address is detected. If more than one byte is transmitted, a not acknowledge is returned and SIO1 enters the not addressed slave receiver mode.

In the slave transmitter mode, data to be transmitted is obtained from the same locations in the internal data RAM that were previously loaded by the main program. After a not acknowledge has been returned by a master receiver device, SIO1 enters the not addressed slave mode.

Adapting the Software for Different Applications

The following software example shows the typical structure of the interrupt routine including the 26 state service routines and may be used as a base for user applications. If one or more of the four modes are not used, the associated state service routines may be removed but, care should be taken that a deleted routine can never be invoked.

This example does not include any time-out routines. In the slave modes, time-out routines are not very useful since, in these modes, SIO1 behaves essentially as a passive device. In the master modes, an internal timer may be used to cause a time-out if a serial transfer is not complete after a defined period of time. This time period is defined by the system connected to the I²C bus.

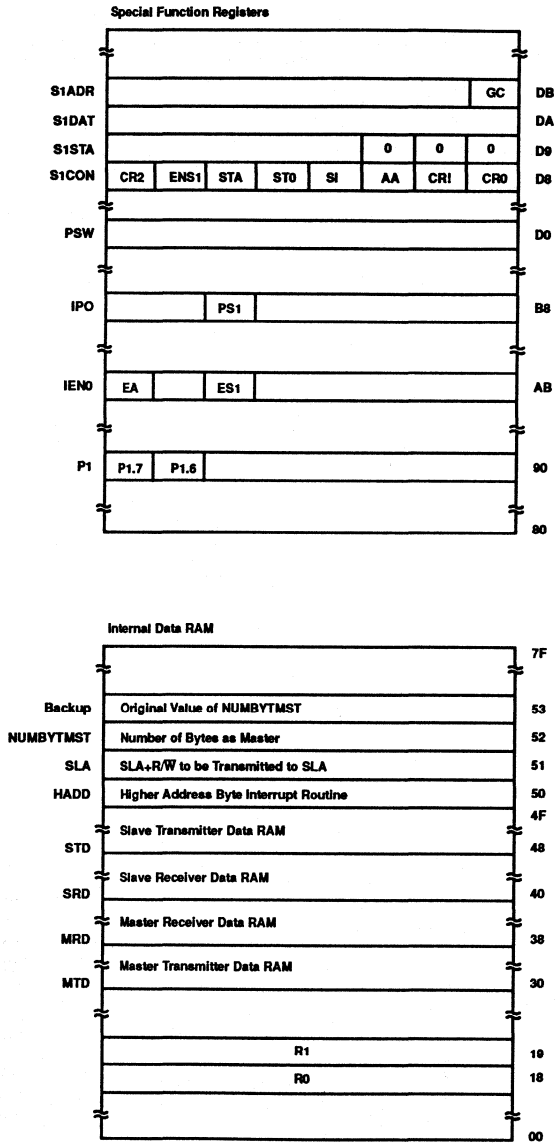


Figure 42. SIO1 Data Memory Map

```
                    !*****************************************************************************
                    ! SI01 EQUATE LIST
                    !*****************************************************************************
                    !*****************************************************************************
                    ! LOCATIONS OF THE SI01 SPECIAL FUNCTION REGISTERS
                    !*****************************************************************************
00D8                S1CON     -0xd8
00D9                S1STA     -0xd9
00DA                S1DAT     -0xda
00DB                S1ADR     -0xdb

00A8                IEN0      -0xa8
00B8                IP0       -02b8

                    !*****************************************************************************
                    ! BIT LOCATIONS
                    !*****************************************************************************

00DD                STA       -0xdd          ! STA bit in S1CON
00BD                SI01HP    -0xbd          ! IP0, SI01 Priority bit

                    !*****************************************************************************
                    ! IMMEDIATE DATA TO WRITE INTO REGISTER S1CON
                    !*****************************************************************************
00D5                ENS1_NOTSTA_STO_NOTSI_AA_CR0          -0xd5   ! Generates STOP
                                                                  ! (CR0 = 100kHz)
00C5                ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0       -0xc5   ! Releases BUS and
                                                                  ! ACK
00C1                ENS1_NOTSTA_NOTSTO_NOTSI_NOTAA_CR0    -0xc1   ! Releases BUS and
                                                                  ! NOT ACK
00E5                ENS1_STA_NOTSTO_NOTSI_AA_CR0          -0xe5   ! Releases BUS and
                                                                  ! set STA

                    !*****************************************************************************
                    ! GENERAL IMMEDIATE DATA
                    !*****************************************************************************
0031                OWNSLA    -0x31          ! Own SLA+General Call
                                             ! must be written into S1ADR
00A0                ENSI01    -0xa0          ! EA+ES1, enable SIO1 interrupt
                                             ! must be written into IEN0
0001                PAG1      -0x01          ! select PAG1 as HADD
00C0                SLAW      -0xc0          ! SLA+W to be transmitted
00C1                SLAR      -0xc1          ! SLA+R to be transmitted
0018                SELRB3    -0x18          ! Select Register Bank 3

                    !*****************************************************************************
                    ! LOCATIONS IN DATA RAM
                    !*****************************************************************************
0030                MTD       -0x30          ! MST/TRX/DATA base address
0038                MRD       -0x38          ! MST/REC/DATA base address
0040                SRD       -0x40          ! SLV/REC/DATA base address
0048                STD       -0x48          ! SLV/TRX/DATA base address

0053                BACKUP    -0x53          ! Backup from NUMBYTMST
                                             ! To restore NUMBYTMST in case
                                             ! of an Arbitration Loss.
0052                NUMBYTMST -0x52          ! Number of bytes to transmit
                                             ! or receive as MST.
0051                SLA       -0x51          ! Contains SLA+R/W to be
                                             ! transmitted.
0050                HADD      -0x50          ! High Address byte for STATE 0
                                             ! till STATE 25.
```

```
                                        |*****************************************************************************
                                        ! INITIALIZATION ROUTINE
                                        ! Example to initialize IIC Interface as slave receiver or
                                        ! slave transmitter and start a MASTER TRANSMIT or a MASTER
                                        ! RECEIVE function. 4 bytes will be transmitted or received.
                                        |*****************************************************************************
                                        .sect  strt
                                        .base  0x00
0000   4100                                     ajmp   INIT                  ! RESET

                                        .sect  initial
                                        .base  0x200
0200   75DB31                           INIT:  mov    S1ADR,#OWNSLA         ! Load own SLA + enable
                                                                             ! general call recognition
0203   D296                                     setb   pl(6)                 ! P1.6 High level.
0205   D297                                     setb   pl(7)                 ! P1.7 High level.
0207   755001                                   mov    HADD,#PAG1
020A   43A8A0                                   orl    IEN0,#ENSI01          ! Enable SI01 interrupt
020D   C2BD                                     clr    SI01HP                ! SI01 interrupt low
                                                                             ! priority
020F   75D8C5                                   mov    S1CON, #ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                             ! Initialize SLV funct.
                                        |*****************************************************************************

                                        |-----------------------------------------------
                                        ! START MASTER TRANSMIT FUNCTION
                                        |-----------------------------------------------

0212   755204                                   mov    NUMBYTMST,#0x4        ! Transmit 4 bytes.
0215   7551C0                                   mov    SLA,#SLAW             ! SLA+W, Transmit funct.
0218   D2DD                                     setb   STA                   ! set STA in S1CON

                                        |-----------------------------------------------
                                        ! START MASTER RECEIVE FUNCTION
                                        |-----------------------------------------------

021A   755204                                   mov    NUMBYTMST,#0x4        ! Receive 4 bytes.
021D   7551C1                                   mov    SLA,#SLAR             ! SLA+R, Receive funct.
0220   D2DD                                     setb   STA                   ! set STA in S1CON
```

```
                        !*****************************************************************************
                        ! SI01 INTERRUPT ROUTINE
                        !*****************************************************************************
                        .sect  intvec                                   ! SI01 interrupt vector
                        .base  0x00

                        ! S1STA and HADD are pushed onto the stack. They serve as
                        ! return address for the RET instruction.
                        ! The RET instruction sets the Program Counter to address
                        ! HADD, S1STA and jumps to the right subroutine.

002B    C0D0                            push  psw                       ! save psw
002D    C0D9                            push  S1STA
002F    C050                            push  HADD
0031    22                              ret                             ! JMP to address
                                                                        ! HADD,S1STA.

                        !-------------------------------------------------------------
                        ! STATE  :  00, Bus error.
                        ! ACTION:  Enter not addressed SLV mode and release bus.
                                   STO reset.
                        !-------------------------------------------------------------
                        .sect  st0
                        .base  0x100

0100    75D8D5                          mov   S1CON,#ENS1_NOTSTA_STO_NOTSI_AA_CR0  ! clr SI
                                                                                   ! set STO,AA
0103    D0D0                            pop   psw
0105    32                              reti
```

```
                        !*****************************************************************************
                        !*****************************************************************************
                        ! MASTER STATE SERVICE ROUTINES
                        !*****************************************************************************
                        ! State 08 and State 10 are both for MST/TRX and MST/REC.
                        ! The R/W bit decides whether the next state is within
                        ! MST/TRX mode or within MST/REC mode.
                        !*****************************************************************************

                        !-----------------------------------------------------------
                        ! STATE  :  08, A, START condition has been transmitted.
                        ! ACTION :  SLA+R/W are transmitted, ACK bit is received.
                        !-----------------------------------------------------------
                        .sect  mts8
                        .base  0x108

0108  8551DA                          mov   S1DAT,SLA            ! Load SLA+R/W
010B  75D8C5                          mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                 ! clr SI
010E  01A0                            ajmp  INITBASE1

                        !-----------------------------------------------------------
                        ! STATE  :  10, A repeated START condition has been
                        !           transmitted.
                        ! ACTION :  SLA+R/W are transmitted, ACK bit is received.
                        !-----------------------------------------------------------
                        .sect  mts10
                        .base  0x110

0110  8551DA                          mov   S1DAT,SLA            ! Load SLA+R/W
0113  75D8C5                          mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                 ! clr SI
010E  01A0                            ajmp  INITBASE1

                        .sect  ibase1
                        .base  0xa0
00A0  75D018  INITBASE1:              mov   psw,#SELRB3
00A3  7930                            mov   rl,#MTD
00A5  7838                            mov   r0,#MRD
00A7  855253                          mov   BACKUP,NUMBYTMST           ! Save initial value
00AA  D0D0                            pop   psw
00AC  32                              reti
```

```
                        |**********************************************************************
                        |**********************************************************************
                        ! MASTER TRANSMITTER STATE SERVICE ROUTINES
                        |**********************************************************************
                        |**********************************************************************

                        |----------------------------------------------------
                        ! STATE  :  18, Previous state was STATE 8 or STATE 10,
                        !           SLA+W have been transmitted, ACK has
                        !           been received.
                        ! ACTION :  First DATA is transmitted, ACK bit
                        !           is received.
                        |----------------------------------------------------
                        .sect  mts18
                        .base  0x118

0118  75D018                        mov   psw,#SELRB3
011B  87DA                          mov   S1DAT,@r1
011D  01B5                          ajmp  CON

                        |----------------------------------------------------
                        ! STATE  :  20, SLA+W have been transmitted, NOT ACK
                        !           has been received
                        ! ACTION :  Transmit STOP condition.
                        |----------------------------------------------------
                        .sect  mts20
                        .base  0x110

0120  75D8D5                        mov   S1CON,#ENS1_NOTSTA_STO_NOTSI_AA_CR0
                                                          ! set STO, clr SI
0123  D0D0                          pop   psw
0125  32                            reti

                        |----------------------------------------------------
                        ! STATE  :  28, DATA of S1DAT have been transmitted,
                        !           ACK received.
                        ! ACTION :  If Transmitted DATA is last DATA then
                        !           transmit a STOP condition, else transmit
                        !           next DATA.
                        |----------------------------------------------------
                        .sect  mts28
                        .base  0x128

0128  D55285                        djnz  NUMBYTMST,NOTLDAT1      ! JMP if NOT last DATA
012B  75D8D5                        mov   S1CON,#ENS1_NOTSTA_STO_NOTSI_AA_CR0
                                                          ! clr SI, set AA
012E  01B9                          ajmp  RETmt

                        .sect  mts28sb
                        .base  0x0b0
00B0  75D018            NOTLDAT1:   mov   psw,#SELRB3
00B3  87DA                          mov   S1DAT,@r1
00B5  75D8C5            CON:        mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                          ! clr SI, set AA
00B8  09                            inc   r1
00B9  D0D0              RETmt    :  pop   psw
00BB  32                            reti
```

```
                        |-------------------------------------------------
                        ! STATE  :  30, DATA of S1DAT have been transmitted,
                        !           NOT ACK received.
                        ! ACTION :  Transmit a STOP condition.
                        |-------------------------------------------------
                        .sect  mts30
                        .base  0x130

0130  75D8D5                          mov   S1CON,#ENS1_NOTSTA_STO_NOTSI_AA_CR0
                                                          ! set STO, clr SI
0133  D0D0                            pop   psw
0135  32                              reti

                        |-------------------------------------------------
                        ! STATE  :  38, Arbitration lost in SLA+W or DATA.
                        ! ACTION :  Bus is released, not addressed SLV mode is
                        !           entered. A new START condition is transmitted
                        !           when the IIC bus is free again.
                        |-------------------------------------------------
                        .sect  mts38
                        .base  0x138

0138  75D8E5                          mov   S1CON,#ENS1_STA_NOTSTO_NOTSI_AA_CR0
013B  855352                          mov   NUMBYTMST,BACKUP
013E  01B9                            ajmp  RETmt


                        !*************************************************************************
                        !*************************************************************************
                        ! MASTER RECEIVER STATE SERVICE ROUTINES
                        !*************************************************************************
                        !*************************************************************************

                        |-------------------------------------------------
                        ! STATE  :  40, Previous state was STATE 08 or STATE 10,
                        !           SLA+R have been transmitted, ACK received.
                        ! ACTION :  DATA will be received, ACK returned.
                        |-------------------------------------------------
                        .sect  mts40
                        .base  0x140

0140  75D8C5                          mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                          ! clr STA, STO, SI set AA
0143  01CB                            ajmp  RETmr

                        |-------------------------------------------------
                        ! STATE  :  48, SLA+R have been transmitted, NOT ACK
                        !           received.
                        ! ACTION :  STOP condition will be generated.
                        |-------------------------------------------------
                        .sect  mts48
                        .base  0x148

0148  75D8D5                          mov   S1CON,#ENS1_NOTSTA_STO_NOTSI_AA_CR0
                                                          ! set STO, clr SI
014B  D0D0                            pop   psw
014D  32                              reti
```

```
                        !-------------------------------------------------
                        ! STATE  :  50, DATA have been received, ACK returned.
                        ! ACTION:  Read DATA of S1DAT.
                        !          DATA will be received, if it is last
                        !          DATA then NOT ACK will be returned else ACK
                        !          will be returned.
                        !-------------------------------------------------
                        .sect  mrs50
                        .base  0x150

0150  75D018                        mov   psw,#SELRB3
0153  A6DA                          mov   @r0,S1DAT          ! Read received DATA
0155  01C0                          ajmp  REC1

                        .sect  mrs50s
                        .base  0xc0

00C0  D55205            REC1:       djnz  NUMBYTMST,NOTLDAT2
00C3  75D8C1                        mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_NOTAA_CR0
                                                             ! clr SI,AA
00C6  8003                          sjmp  RETmr
00C8  75D8C5            NOTLDAT2:   mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                             ! clr SI, set AA
00CB  08                RETmr:      inc   r0
00CC  D0D0                          pop   psw
00CE  32                            reti

                        !-------------------------------------------------
                        ! STATE  :  58, DATA have been received, NOT ACK
                        !          returned.
                        ! ACTION:  Read DATA of S1DAT and generate a STOP
                        !          condition.
                        !-------------------------------------------------
                        .sect  mrs58
                        .base  0x158

0158  75D018                        mov   psw,#SELRB3
015B  A6DA                          mov   @0,S1DAT
015D  80E9                          sjmp  STOP
```

```
                        !*****************************************************************************
                        !*****************************************************************************
                        ! SLAVE RECEIVER STATE SERVICE ROUTINES
                        !*****************************************************************************
                        !*****************************************************************************

                        !-----------------------------------------------------------
                        ! STATE  :  60, Own SLA+W have been received, ACK
                        !           returned.
                        ! ACTION :  DATA will be received and ACK returned.
                        !-----------------------------------------------------------
                        .sect  srs60
                        .base  0x160
0160  75D8C5                        mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                          ! clr SI, set AA
0163  75D018                        mov   psw,#SELRB3
0166  01D0                          ajmp  INITSRD
                        .sect  insrd
                        .base  0xd0

00D0  7840              INITSRD:    mov   r0,#SRD
00D2  7908                          mov   rl,#8
00D4  D0D0                          pop   psw
00D6  32                            reti

                        !-----------------------------------------------------------
                        ! STATE  :  68, Arbitration lost in SLA and R/W as MST
                        !           Own SLA+W have been received, ACK returned
                        ! ACTION :  DATA will be received and ACK returned.
                        !           STA is set to restart MST mode after the
                        !           bus is free again.
                        !-----------------------------------------------------------
                        .sect  srs68
                        .base  0x168
0168  75D8E5                        mov   S1CON,#ENS1_STA_NOTSTO_NOTSI_AA_CR0
016B  75D018                        mov   psw,#SELRB3
016E  01D0                          ajmp  INITSRD

                        !-----------------------------------------------------------
                        ! STATE  :  70, General call has been received, ACK
                        !           returned.
                        ! ACTION :  DATA will be received and ACK returned.
                        !-----------------------------------------------------------
                        .sect  srs70
                        .base  0x170
0170  75D8C5                        mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                          ! clr SI, set AA
0173  75D018                        mov   psw,#SELRB3     ! Initialize SRD counter

                        !-----------------------------------------------------------
                        ! STATE  :  78, Arbitration lost in SLA+R/W as MST.
                        !           General call has been received, ACK returned.
                        ! ACTION :  DATA will be received and ACK returned.
                        !           STA is set to restart MST mode after the
                        !           bus is free again.
                        !-----------------------------------------------------------
                        .sect  srs78
                        .base  0x178
0178  75D8E5                        mov   S1CON,#ENS1_STA_NOTSTO_NOTSI_AA_CR0
017B  75D018                        mov   psw,#SELRB3     ! Initialize SRD counter
017E  01D0                          ajmp  INITSRD
```

```
                        !-----------------------------------------------
                        ! STATE  :  80, Previously addressed with own SLA.
                        !          DATA received, ACK returned.
                        ! ACTION :  Read DATA.
                        !          IF received DATA was the last THEN superfluous
                        !          DATA will be received and NOT ACK returned ELSE
                        !          next DATA will be received and ACK returned.
                        !-----------------------------------------------
                        .sect  srs80
                        .base  0x180

0180  75D018                        mov   psw,#SELRB3
0183  A6DA                          mov   @r0,S1DAT          ! Read received DATA
0185  01D8                          ajmp  REC2

                        .sect  srs80s
                        .base  0xd8

00D8  D906              REC2:       djnz  r1,NOTLDAT3
00DA  75D8C1                        mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_NOTAA_CR0
                                                          ! clr SI,AA
00DD  D0D0                          pop   psw
00DF  32                            reti
00E0  75D8C5            NOTLDAT2:   mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                          ! clr SI, set AA
00E3  08                            inc   r0
00E4  D0D0              RETsr:      pop   psw
00E6  32                            reti

                        !-----------------------------------------------
                        ! STATE  :  88, Previously addressed with own SLA.
                        !          DATA received NOT ACK returned.
                        ! ACTION :  No save of DATA, Enter NOT addressed SLV
                        !          mode. Recognition of own SLA. General call
                        !          recognized, if S1ADR. 0–1.
                        !-----------------------------------------------
                        .sect  srs88
                        .base  0x188

0188  75D8C5                        mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                          ! clr SI, set AA
018B  01E4                          ajmp  RETsr

                        !-----------------------------------------------
                        ! STATE  :  90, Previously addressed with general call.
                        !          DATA has been received, ACK has been returned.
                        ! ACTION :  Read DATA.
                        !          After General call only one byte will be
                        !          received with ACK the second DATA will be
                        !          received with NOT ACK.
                        !          DATA will be received and NOT ACK
                        !          returned.
                        !-----------------------------------------------
                        .sect  srs90
                        .base  0x190

0190  75D018                        mov   psw,#SELRB3
0193  A6DA                          mov   @r0,S1DAT          ! Read received DATA
0195  01DA                          ajmp  LDAT
```

```
                        |-------------------------------------------------
                        ! STATE  :  98, Previously addressed with general call.
                        !           DATA has been received, NOT ACK has been
                        !           returned.
                        ! ACTION :  No save of DATA, Enter NOT addressed SLV
                        !           mode. Recognition of own SLA. General call
                        !           recognized, if S1ADR. 0–1.
                        |-------------------------------------------------
                        .sect  srs98
                        .base  0x198

0198  75D8C5                        mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                          ! clr SI, set AA
019B  D0D0                          pop   psw
019D  32                            reti


                        |-------------------------------------------------
                        ! STATE  :  A0, A STOP condition or repeated START has
                        !           been received, while still addressed as
                        !           SLV/REC or SLV/TRX.
                        ! ACTION :  No save of DATA, Enter NOT addressed SLV
                        !           mode. Recognition of own SLA. General call
                        !           recognized, if S1ADR. 0–1.
                        |-------------------------------------------------
                        .sect  srsA0
                        .base  0x1a0

01A0  75D8C5                        mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                          ! clr SI, set AA
01A3  D0D0                          pop   psw
01A5  32                            reti
```

```
                        !*****************************************************************************
                        !*****************************************************************************
                        ! SLAVE TRANSMITTER STATE SERVICE ROUTINES
                        !*****************************************************************************
                        !*****************************************************************************

                        !-----------------------------------------------------------
                        ! STATE  :  A8, Own SLA+R received, ACK returned.
                        ! ACTION :  DATA will be transmitted, A bit received.
                        !-----------------------------------------------------------
                        .sect  stsa8
                        .base  0x1a8

01A8  8548DA                            mov   S1DAT,STD          ! load DATA in S1DAT
01AB  75D8C5                            mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                 ! clr SI, set AA
01AE  01E8                              ajmp  INITBASE2

                        .sect  ibase2
                        .base  0xe8
00E8  75D018            INITBASE2:      mov   psw,#SELRB3
00EB  7948                              mov   r1, #STD
00ED  09                                inc   r1
00EE  D0D0                              pop   psw
00F0  32                                reti

                        !-----------------------------------------------------------
                        ! STATE  :  B0, Arbitration lost in SLA and R/W as MST.
                        !           Own SLA+R received, ACK returned.
                        ! ACTION :  DATA will be transmitted, A bit received.
                        !           STA is set to restart MST mode after the
                        !           bus is free again.
                        !-----------------------------------------------------------
                        .sect  stsb0
                        .base  0x1b0

01B0  8548DA                            mov   S1DAT,STD          ! load DATA in S1DAT
01B3  75D8E5                            mov   S1CON,#ENS1_STA_NOTSTO_NOTSI_AA_CR0
01B6  01E8                              ajmp  INITBASE2

                        !-----------------------------------------------------------
                        ! STATE  :  B8, DATA has been transmitted, ACK received.
                        ! ACTION :  DATA will be transmitted, ACK bit is received.
                        !-----------------------------------------------------------
                        .sect  stsb8
                        .base  0x1b8
01B8  75D018                            mov   psw,#SELRB3
01BB  87DA                              mov   S1DAT,@r1
01BD  01F8                              ajmp  SCON

                        .sect  scn
                        .base  0xf8

00F8  75D8C5            SCON:           mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                 ! clr SI, set AA

00FB  09                                inc   r1
00FC  D0D0                              pop   psw
00FE  32                                reti
```

```
                        !-------------------------------------------------
                        ! STATE  :  C0, DATA has been transmitted, NOT ACK
                        !          received.
                        ! ACTION :  Enter not addressed SLV mode.
                        !-------------------------------------------------
                        .sect  stsc0
                        .base  0x1c0
01C0  75D8C5                       mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                          ! clr SI, set AA
01C3  D0D0                         pop   psw
01C5  32                           reti

                        !-------------------------------------------------
                        ! STATE  :  C8, Last DATA has been transmitted (AA=0),
                        !          ACK received.
                        ! ACTION :  Enter not addressed SLV mode.
                        !-------------------------------------------------
                        .sect  stsc8
                        .base  0x1c8
01C8  75D8C5                       mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                          ! clr SI, set AA
01CB  D0D0                         pop   psw
01CD  32                           reti

                        !*************************************************************************************
                        !*************************************************************************************
                        ! END OF SI01 INTERRUPT ROUTINE
                        !*************************************************************************************
                        !*************************************************************************************
```

Reset Circuitry

The reset circuitry for the 8XC552 is connected to the reset pin RST. A Schmitt trigger is used at the input for noise rejection (see Figure 43). The output of the Schmitt trigger is sampled by the reset circuitry every machine cycle.

A reset is accomplished by holding the RST pin HIGH for at least two machine cycles (24 oscillator periods) while the oscillator is running. The CPU responds by executing an internal reset. During reset, ALE and PSEN output a HIGH level. In order to perform a correct reset, this level must not be affected by external elements. The RST line can also be pulled HIGH internally by a pull-up transistor activated by the watchdog timer T3. The length of the output pulse from T3 is 3 machine cycles. A pulse of such short duration is necessary in order to recover from a processor or system fault as fast as possible.

Note that the short reset pulse from Timer T3 cannot discharge the power-on reset capacitor (see Figure 44). Consequently, when the watchdog timer is also used to set external devices, this capacitor arrangement should not be connected to the RST pin, and a different circuit should be used to perform the power-on reset operation. A timer T3 overflow, if enabled, will force a reset condition to the 8XC552 by an internal connection, whether the output RTS is tied LOW or not.

The internal reset is executed during the second cycle in which RST is HIGH and is repeated every cycle until RST goes low. It leaves the internal registers as follows:

REGISTER	CONTENT	
ACC	0000	0000
ADCON	xx00	0000
ADCH	xxxx	xxxx
B	0000	0000
CML0-CML2	0000	0000
CMH0-CMH2	0000	0000
CTCON	0000	0000
CTL0-CTL3	xxxx	xxxx
CTH0-CTH3	xxxx	xxxx
DPL	0000	0000
DPH	0000	0000
IEN0	0000	0000
IEN1	0000	0000
IP0	x000	0000
IP1	0000	0000
PCH	0000	0000
PCL	0000	0000
PCON	0xx0	0000
PSW	0000	0000
PWM0	0000	0000
PWM1	0000	0000
PWMP	0000	0000
P0-P4	1111	1111
P5	xxxx	xxxx
RTE	0000	0000
S0BUF	xxxx	xxxx
S0CON	0000	0000
S1ADR	0000	0000
S1CON	0000	0000
S1DAT	0000	0000
S1STA	1111	1000
SP	0000	0111
STE	1100	0000
TCON	0000	0000
TH0, TH1	0000	0000
TMH2	0000	0000
TL0, TL1	0000	0000
TML2	0000	0000
TMOD	0000	0000
TM2CON	0000	0000
TM2IR	0000	0000
T3	0000	0000

The internal RAM is not affected by reset. At power-on, the RAM content is indeterminate.

Interrupts

The 8XC552 has fifteen interrupt sources, each of which can be assigned one of two priority levels, as shown in Figure 45. The five interrupt sources common to the 80C51 are the external interrupts ($\overline{INT0}$ and $\overline{INT1}$), the timer 0 and timer 1 interrupts (IT0 and IT1), and the serial I/O interrupt (RI or TI). In the 8XC552, the standard serial interrupt is called SIO0. Since the subsystems which create these interrupts are identical on both parts, their functionality is likewise identical. The only differences are the locations of the enable and priority register configurations and the priority structure. This is detailed below along with the specifics of the interrupts unique to the 8XC552.

The eight Timer T2 interrupts are generated by flags CTI0-CT13, CMI0-CMI2, and by the logical OR of flags T2OV and T2BO. Flags CTI0 to CT13 are set by input signals CT0I to CT3i. Flags CMI0 to CMI2 are set when a match occurs between Timer T2 and the compare registers CM0, CM1, and CM2. When an 8-bit or 16-bit overflow occurs, flags T2BO and T2OV are set, respectively. These nine flags are not cleared by hardware and must be reset by software to avoid recurring interrupts.

The ADC interrupt is generated by the ADCI flag in the ADC control register (ADCON). This flag is set when an ADC conversion result is ready to be read. ADCI is not cleared by hardware and must be reset by software to avoid recurring interrupts.

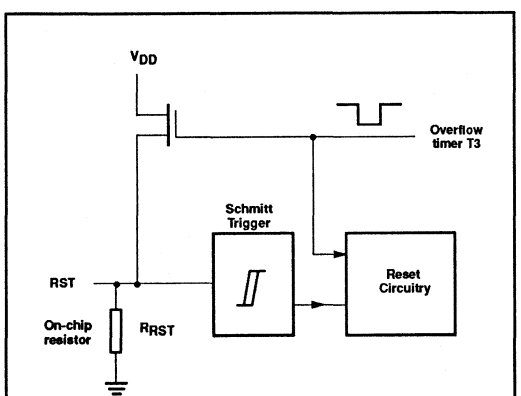


Figure 43. On-Chip Reset Configuration

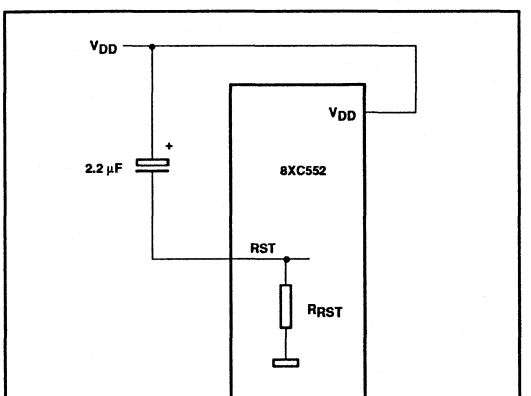


Figure 44. Power-On Reset

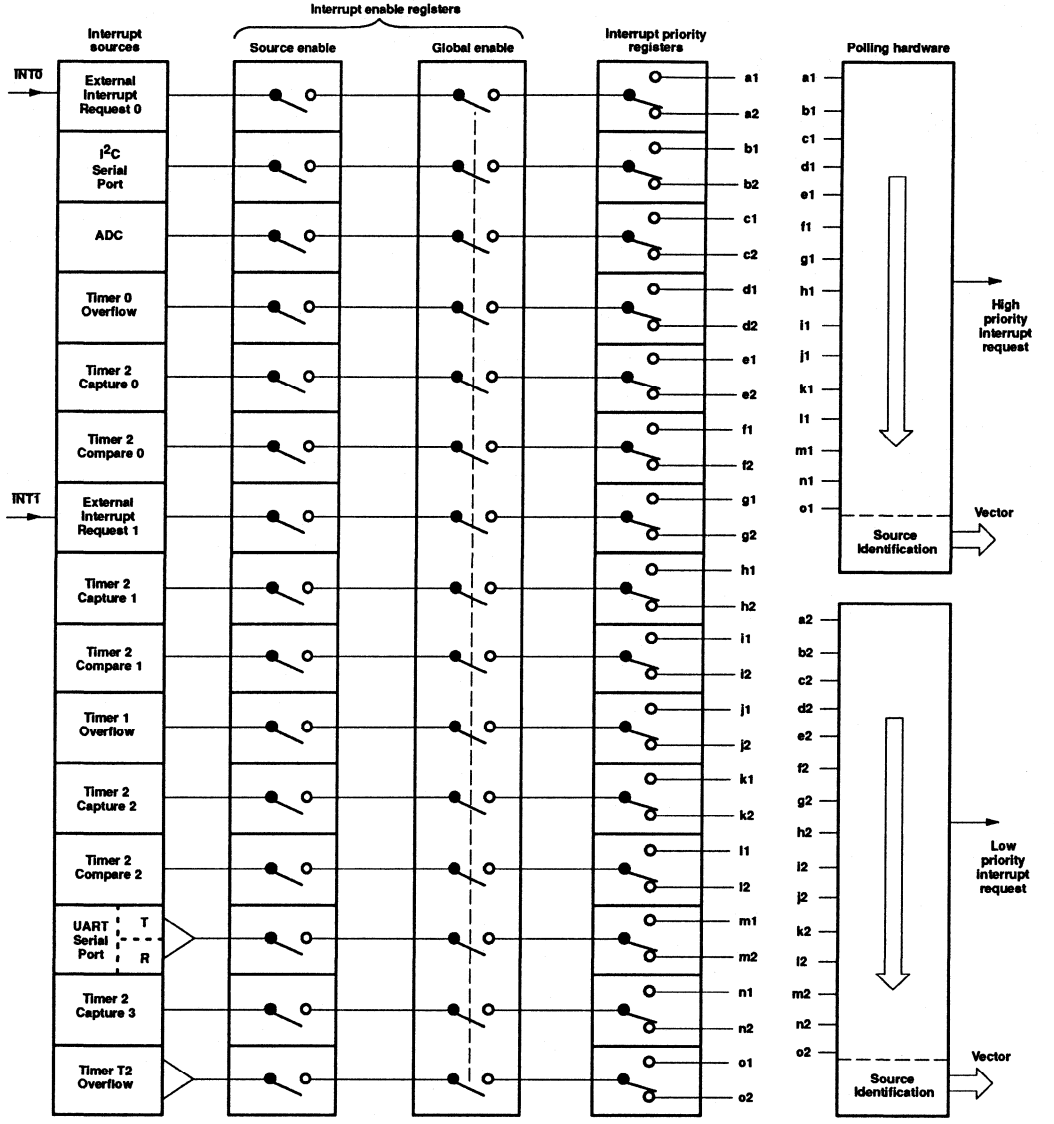


Figure 45. The Interrupt System

The SIO1 (I^2C) interrupt is generated by the SI flag in the SIO1 control register (S1CON). This flag is set when S1STA is loaded with a valid status code.

The ADCI flag may be reset by software. It cannot be set by software. All other flags that generate interrupts may be set or cleared by software, and the effect is the same as setting or resetting the flags by hardware. Thus, interrupts may be generated by software and pending interrupts can be canceled by software.

Interrupt Enable Registers: Each interrupt source can be individually enabled or disabled by setting or clearing a bit in the interrupt enable special function registers IEN0 and IEN1. All interrupt sources can also be globally enabled or disabled by setting or clearing bit EA in IEN0. The interrupt enable registers are described in Figures 46 and 47.

Interrupt Priority Structure: Each interrupt source can be assigned one of two priority levels. Interrupt priority levels are defined by the interrupt priority special function registers IP0 and IP1. IP0 and IP1 are described in Figures 48 and 49.

Interrupt priority levels are as follows:
"0"—low priority
"1"—high priority

A low priority interrupt may be interrupted by a high priority interrupt. A high priority interrupt cannot be interrupted by any other interrupt source. If two requests of different priority occur simultaneously, the high priority level request is serviced. If requests of the same priority are received simultaneously, an internal polling sequence determines which request is serviced. Thus, within each priority level, there is a second priority structure determined by the polling sequence. This second priority structure is shown in Table 21.

The above Priority Within Level structure is only used when there are simultaneous requests of the same priority level.

Interrupt Handling: The interrupt sources are sampled at S5P2 of every machine cycle. The samples are polled during the following machine cycle. If one of the flags was in a set condition at S5P2 of the previous machine cycle, the polling cycle will find it and the interrupt system will generate an LCALL to the appropriate service routine, provided this hardware-generated LCALL is not blocked by any of the following conditions:

1. An interrupt of higher or equal priority level is already in progress.
2. The current machine cycle is not the final cycle in the execution of the instruction in progress. (No interrupt request will be serviced until the instruction in progress is completed.)
3. The instruction in progress is RETI or any access to the interrupt priority or interrupt enable registers. (No interrupt will be serviced after RETI or after a read or write to IP0, IP1, IE0, or IE1 until at least one other instruction has been subsequently executed.)

The polling cycle is repeated with every machine cycle, and the values polled are the values present at S5P2 of the previous machine cycle. Note that if an interrupt flag is active but is not being responded to because of one of the above conditions, and if the flag is inactive when the blocking condition is removed, then the blocked interrupt will not be serviced. Thus, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle is new.

	7	6	5	4	3	2	1	0
IEN0 (A8H)	EA	EAD	ES1	ES0	ET1	EX1	ET0	EX0
	(MSB)							(LSB)

Bit	Symbol	Function
IEN0.7	EA	Global enable/disable control 0 = No interrupt is enabled 1 = Any individually enabled interrupt will be accepted
IEN0.6	EAD	Enable ADC interrupt
IEN0.5	ES1	Enable SIO1 (I^2C) interrupt
IEN0.4	ES0	Enable SIO0 (UART) interrupt
IEN0.3	ET1	Enable Timer 1 interrupt
IEN0.2	EX1	Enable External interrupt 1
IEN0.1	ET0	Enable Timer 0 interrupt
IEN0.0	EX0	Enable External interrupt 0

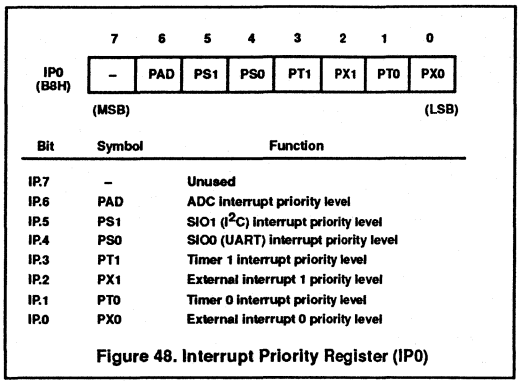
Figure 46. Interrupt Enable Register (IEN0)

	7	6	5	4	3	2	1	0
IEN1 (E8H)	ET2	ECM2	ECM1	ECM0	ECT3	ECT2	ECT1	ECT0
	(MSB)							(LSB)

Bit	Symbol	Function
IEN1.7	ET2	Enable T2 overflow interrupt(s)
IEN1.6	ECM2	Enable T2 comparator 2 interrupt
IEN1.5	ECM1	Enable T2 comparator 1 interrupt
IEN1.4	ECM0	Enable T2 comparator 0 interrupt
IEN1.3	ECT3	Enable T2 capture register 3 interrupt
IEN1.2	ECT2	Enable T2 capture register 2 interrupt
IEN1.1	ECT1	Enable T2 capture register 1 interrupt
IEN1.0	ECT0	Enable T2 capture register 0 interrupt

In all cases, if the enable bit is 0, then the interrupt is disabled, and if the enable bit is 1, then the interrupt is enabled.

Figure 47. Interrupt Enable Register (IEN1)

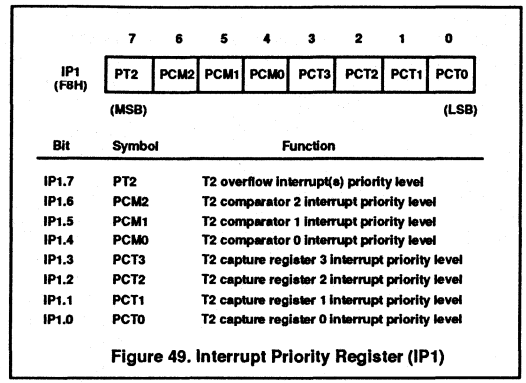


	7	6	5	4	3	2	1	0
IP0 (B8H)	–	PAD	PS1	PS0	PT1	PX1	PT0	PX0

(MSB) (LSB)

Bit	Symbol	Function
IP.7	–	Unused
IP.6	PAD	ADC interrupt priority level
IP.5	PS1	SIO1 (I^2C) interrupt priority level
IP.4	PS0	SIO0 (UART) interrupt priority level
IP.3	PT1	Timer 1 interrupt priority level
IP.2	PX1	External interrupt 1 priority level
IP.1	PT0	Timer 0 interrupt priority level
IP.0	PX0	External interrupt 0 priority level

Figure 48. Interrupt Priority Register (IP0)



	7	6	5	4	3	2	1	0
IP1 (F8H)	PT2	PCM2	PCM1	PCM0	PCT3	PCT2	PCT1	PCT0

(MSB) (LSB)

Bit	Symbol	Function
IP1.7	PT2	T2 overflow interrupt(s) priority level
IP1.6	PCM2	T2 comparator 2 interrupt priority level
IP1.5	PCM1	T2 comparator 1 interrupt priority level
IP1.4	PCM0	T2 comparator 0 interrupt priority level
IP1.3	PCT3	T2 capture register 3 interrupt priority level
IP1.2	PCT2	T2 capture register 2 interrupt priority level
IP1.1	PCT1	T2 capture register 1 interrupt priority level
IP1.0	PCT0	T2 capture register 0 interrupt priority level

Figure 49. Interrupt Priority Register (IP1)

Table 21. Interrupt Priority Structure

SOURCE	NAME	PRIORITY WITHIN LEVEL
External interrupt 0	X0	(highest)
SIO1 (I^2C)	S1	↑
ADC completion	ADC	
Timer 0 overflow	T0	
T2 capture 0	CT0	
T2 compare 0	CM0	
External interrupt 1	X1	
T2 capture 1	CT1	
T2 compare 1	CM1	
Timer 1 overflow	T1	
T2 capture 2	CT2	
T2 compare 2	CM2	
SIO0 (UART)	S0	
T2 capture 3	CT3	
Timer T2 overflow	T2	↓ (lowest)

The processor acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate service routine. In some cases it also clears the flag which generated the interrupt, and in others it does not. It clears the Timer 0, Timer 1, and external interrupt flags. An external interrupt flag (IE0 or IE1) is cleared only if it was transition-activated. All other interrupt flags are not cleared by hardware and must be cleared by the software. The LCALL pushes the contents of the program counter on to the stack (but it does not save the PSW) and reloads the PC with an address that depends on the source of the interrupt being vectored to as shown in Table 22.

Execution proceeds from the vector address until the RETI instruction is encountered. The RETI instruction clears the "priority level active" flip-flop that was set when this interrupt was acknowledged. It then pops the top two bytes from the stack and reloads the program counter. Execution of the interrupted program continues from where it was interrupted.

Table 22. Interrupt Vector Addresses

SOURCE	NAME	VECTOR ADDRESS
External interrupt 0	X0	0003H
Timer 0 overflow	T0	000BH
External interrupt 1	X1	0013H
Timer 1 overflow	T1	001BH
SIO0 (UART)	S0	0023H
SIO1 (I^2C)	S1	002BH
T2 capture 0	CT0	0033H
T2 capture 1	CT1	003BH
T2 capture 2	CT2	0043H
T2 capture 3	CT3	004BH
ADC completion	ADC	0053H
T2 compare 0	CM0	005BH
T2 compare 1	CM1	0063H
T2 compare 2	CM2	006BH
T2 overflow	T2	0073H

I/O Port Structure

The 8XC552 has six 8-bit ports. Each port consists of a latch (special function registers P0 to P5), an input buffer, and an output driver (port 0 to 4 only). Ports 0-3 are the same as in the 80C51, with the exception of the additional functions of port 1. The parallel I/O function of port 4 is equal to that of ports 1, 2, and 3. Port 5 may be used as an input port only.

Figure 50 shows the bit latch and I/O buffer functional diagrams of the unique 8XC552 ports. A bit latch corresponds to one bit in a port's SFR and is represented as a D type flip-flop. A "write to latch" signal from the CPU latches a bit from the internal bus and a "read latch" signal from the CPU places the Q output of the flip-flop on the internal bus. A "read pin" signal from the CPU places the actual port pin level on the internal bus. Some instructions that read a port read the actual port pin levels, and other instructions read the latch (SFR) contents.

Port 1 Operation

Port 1 operates the same as it does in the 8051 with the exception of port lines P1.6 and P1.7, which may be selected as the SCL and SDA lines of serial port SIO1 (I^2C). Because the I^2C bus may be active while the device is disconnected from V_{DD}, these pins are provided with open drain drivers. Therefore pins P1.6 and P1.7 do not have internal pull-ups.

Port 5 Operation

Port 5 may be used to input up to 8 analog signals to the ADC. Unused ADC inputs may be used to input digital inputs. These inputs have an inherent hysteresis to prevent the input logic from drawing excessive current from the power lines when driven by analog signals. Channel to channel crosstalk (Ct) should be taken into consideration when both analog and digital signals are simultaneously input to Port 5 (see, D.C. characteristics in data sheet).

Port 5 is not bidirectional and may not be configured as an output port. All six ports are multifunctional, and their alternate functions are listed in Table 23. A more detailed description of these features can be found in the relevant parts of this section.

Pulse Width Modulated Outputs

The 8XC552 contains two pulse width modulated output channels (see Figure 51). These channels generate pulses of programmable length and interval. The repetition frequency is defined by an 8-bit prescaler PWMP, which supplies the clock for the counter. The prescaler and counter are common to both PWM channels. The 8-bit counter counts modulo 255, i.e., from 0 to 254 inclusive. The value of the 8-bit counter is compared to the contents of two registers: PWM0 and PWM1. Provided the contents of either of these registers is greater than the counter value, the corresponding $\overline{PWM0}$ or $\overline{PWM1}$ output is set LOW. If the contents of these registers are equal to, or less than the counter value, the output will be HIGH. The pulse-width-ratio is therefore defined by the contents of the registers PWM0 and PWM1. The pulse-width-ratio is in the range of 0 to 1 and may be programmed in increments of 1/255.

Buffered PWM outputs may be used to drive DC motors. The rotation speed of the motor would be proportional to the contents of PWMn. The PWM outputs may also be configured as a dual DAC. In this application, the PWM outputs must be integrated using conventional operational amplifier circuitry. If the resulting output voltages have to be accurate, external buffers with their own analog supply should be used to buffer the PWM outputs before they are integrated. The repetition frequency fpwm, at the PWMn outputs is give by:

$$fpwm = \frac{fosc}{2 \times (1 + PWMP) \times 255}$$

This gives a repetition frequency range of 92Hz to 23.5kHz (fosc = 12MHz). By loading the PWM registers with either 00H or FFH, the PWM channels will output a constant HIGH or LOW level, respectively. Since the 8-bit counter counts modulo 255, it can never actually reach the value of the PWM registers when they are loaded with FFH.

When a compare register (PWM0 or PWM1) is loaded with a new value, the associated output is updated immediately. It does not have to wait until the end of the current counter period. Both $\overline{PWMn}$ output pins are driven by push-pull drivers. These pins are not used for any other purpose.

Prescaler frequency control register PWMP

PWMP (FEH)	7	6	5	4	3	2	1	0
	MSB							LSB

PWMP.0-7 Prescaler division factor = PWMP + 1.

Reading PWMP gives the current reload value. The actual count of the prescaler cannot be read.

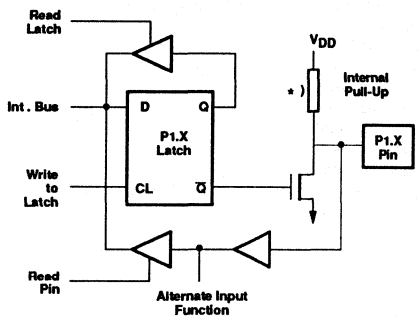
PWM0 (FCH) PWM1 (FDH)	7	6	5	4	3	2	1	0
	MSB							LSB

PWM0/1.0-7} Low/high ratio of $\overline{PWMn}$ =

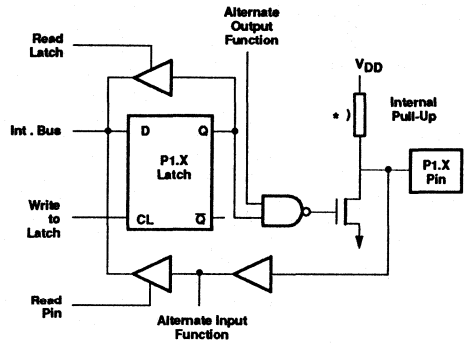
$$\frac{(PWMn)}{255 - (PWMn)}$$

Analog-to-Digital Converter

The analog input circuitry consists of an 8-input analog multiplexer and a 10-bit, straight binary, successive approximation ADC. The analog reference voltage and analog power supplies are connected via separate input pins. The conversion takes 50 machine cycles, i.e., 50µs at an oscillator frequency of 12MHz. Input voltage swing is from 0V to +5V. Because the internal DAC employs a ratiometric potentiometer, there are no discontinuities in the converter characteristic. Figure 52 shows a functional diagram of the analog input circuitry.

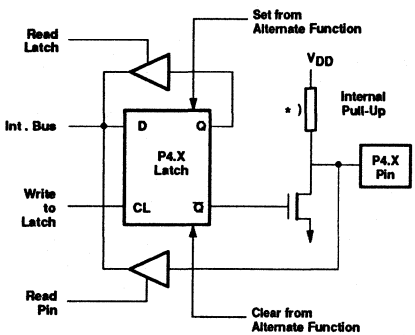


A.

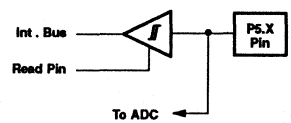


B.

NOTE:
Pull-up not present on P1.6 and P1.7.
*Two period active pull-up as in the 80C51.



C.



D.

Figure 50. Port Bit Latches and I/O Buffers

Table 23. Input/Output Ports

PORT PIN	ALTERNATE FUNCTION	
P0.0	AD0	Multiplexed lower order address/data bus used during external memory accesses
P0.1	AD1	
P0.2	AD2	
P0.3	AD3	
P0.4	AD4	
P0.5	AD5	
P0.6	AD6	
P0.7	AD7	
P1.0	CT0I	Capture timer input signals for timer T2
P1.1	CT1I	
P1.2	CT2I	
P1.3	CT3I	
P1.4	T2	T2 event input
P1.5	RT2	T2 timer reset signal. Rising edge triggered
P1.6	SCL	Serial port clock line I^2C bus
P1.7	SDA	Serial port data line I^2C bus
P2.0	A8	High order address byte used during external memory accesses
P2.1	A9	
P2.2	A10	
P2.3	A11	
P2.4	A12	
P2.5	A13	
P2.6	A14	
P2.7	A15	
P3.0	RXD	Serial input port (UART)
P3.1	TXD	Serial output port (UART)
P3.2	$\overline{INT0}$	External interrupt 0
P3.3	$\overline{INT1}$	External interrupt 1
P3.4	T0	Timer 0 external input
P3.5	T1	Timer 1 external input
P3.6	$\overline{WR}$	External data memory write strobe
P3.7	$\overline{RD}$	External data memory read strobe
P4.0	CMSR0	Timer T2: compare and set/reset outputs on a match with timer T2
P4.1	CMSR1	
P4.2	CMSR2	
P4.3	CMSR3	
P4.4	CMSR4	
P4.5	CMSR5	
P4.6	CMT0	Timer T2: compare and toggle outputs on a match with timer T2
P4.7	CMT1	
P5.0	ADC0	Eight analogue ADC inputs
P5.1	ADC1	
P5.2	ADC2	
P5.3	ADC3	
P5.4	ADC4	
P5.5	ADC5	
P5.6	ADC6	
P5.7	ADC7	

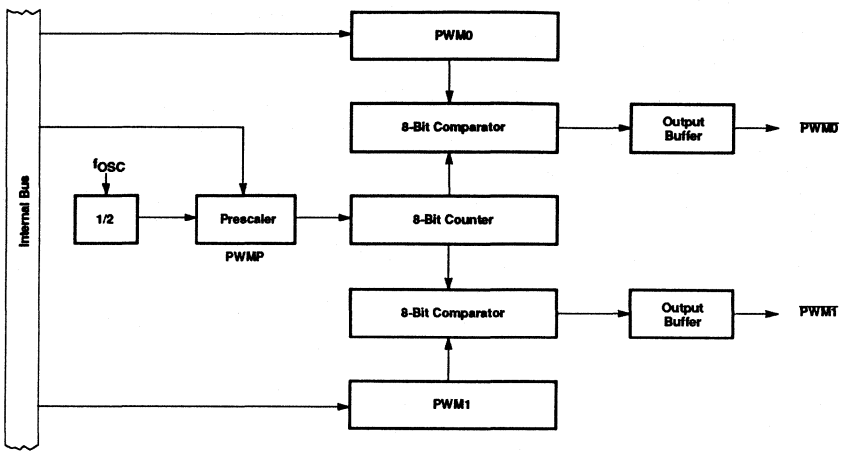


Figure 51. Functional Diagram of Pulse Width Modulated Outputs

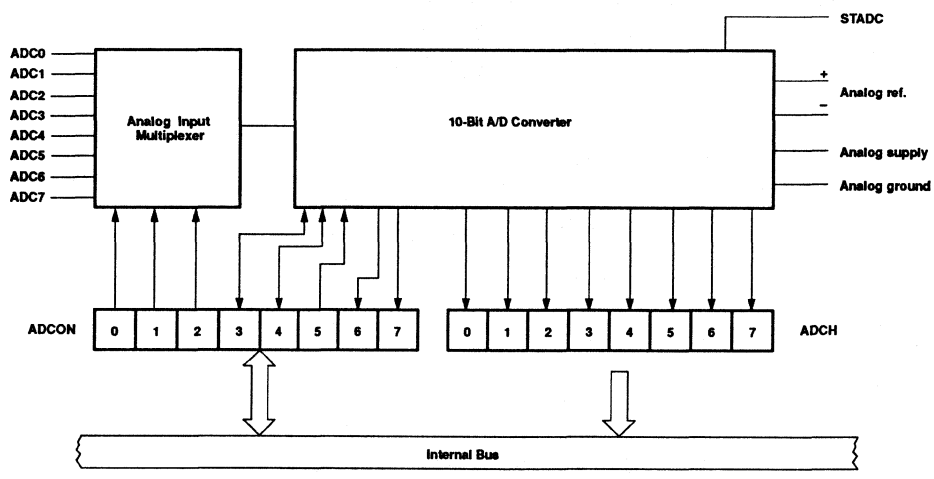


Figure 52. Functional Diagram of Analog Input Circuitry

Analog-to-Digital Conversion: Figure 53 shows the elements of a successive approximation (SA) ADC. The ADC contains a DAC which converts the contents of a successive approximation register to a voltage (VDAC) which is compared to the analog input voltage (Vin). The output of the comparator is fed to the successive approximation control logic which controls the successive approximation register. A conversion is initiated by setting ADCS in the ADCON register. ADCS can be set by software only or by either hardware or software.

The software only start mode is selected when control bit ADCON.5 (ADEX) = 0. A conversion is then started by setting control bit ADCON.3 (ADCS). The hardware or software start mode is selected when ADCON.5 = 1, and a conversion may be started by setting ADCON.3 as above or by applying a rising edge to external pin STADC. When a conversion is started by applying a rising edge, a low level must be applied to STADC for at least one machine cycle followed by a high level for at least one machine cycle.

The low-to-high transition of STADC is recognized at the end of a machine cycle, and the conversion commences at the beginning of the next cycle. When a conversion is initiated by software, the conversion starts at the beginning of the machine cycle which follows the instruction that sets ADCS. ADCS is actually implemented with two flip-flops: a command flip-flop which is affected by set operations, and a status flag which is accessed during read operations.

The next two machine cycles are used to initiate the converter. At the end of the first cycle, the ADCS status flag is set and a value of "1" will be returned if the ADCS flag is read while the conversion is in progress. Sampling of the analog input commences at the end of the second cycle.

During the next eight machine cycles, the voltage at the previously selected pin of port 5 is sampled, and this input voltage should be stable in order to obtain a useful sample.vent, the input voltage slew rate must be less than 10V/ms in order to prevent an undefined result.

The successive approximation control logic first sets the most significant bit and clears all other bits in the successive approximation register (10 0000 0000B). The output of the DAC (50% full scale) is compared to the input voltage Vin. If the input voltage is greater than VDAC, then the bit remains set; otherwise it is cleared.

The successive approximation control logic now sets the next most significant bit (11 0000 0000B or 01 0000 0000B, depending on the previous result), and VDAC is compared to Vin again. If the input voltage is greater than VDAC, then the bit being tested remains set; otherwise the bit being tested is cleared.

This process is repeated until all ten bits have been tested, at which stage the result of the conversion is held in the successive approximation register. Figure 54 shows a conversion flow chart. The bit pointer identifies the bit under test. The conversion takes four machine cycles per bit.

The end of the 10-bit conversion is flagged by control bit ADCON.4 (ADCI). The upper 8 bits of the result are held in special function register ADCH, and the two remaining bits are held in ADCON.7 (ADC.1) and ADCON.6 (ADC.0). The user may ignore the two least significant bits in ADCON and use the ADC as an 8-bit converter (8 upper bits in ADCH). In any event, the total actual conversion time is 50 machine cycles. ADCI will be set and the ADCS status flag will be reset 50 cycles after the command flip-flop (ADCS) is set.

Control bits ADCON.0, ADCON.1, and ADCON.2 are used to control an analog multiplexer which selects one of eight analog channels (see Figure 55). An ADC conversion in progress is unaffected by an external or software ADC start. The result of a completed conversion remains unaffected provided ADCI = logic 1; a new ADC conversion already in progress is aborted when the idle or power-down mode is entered. The result of a completed conversion (ADCI = logic 1) remains unaffected when entering the idle mode.

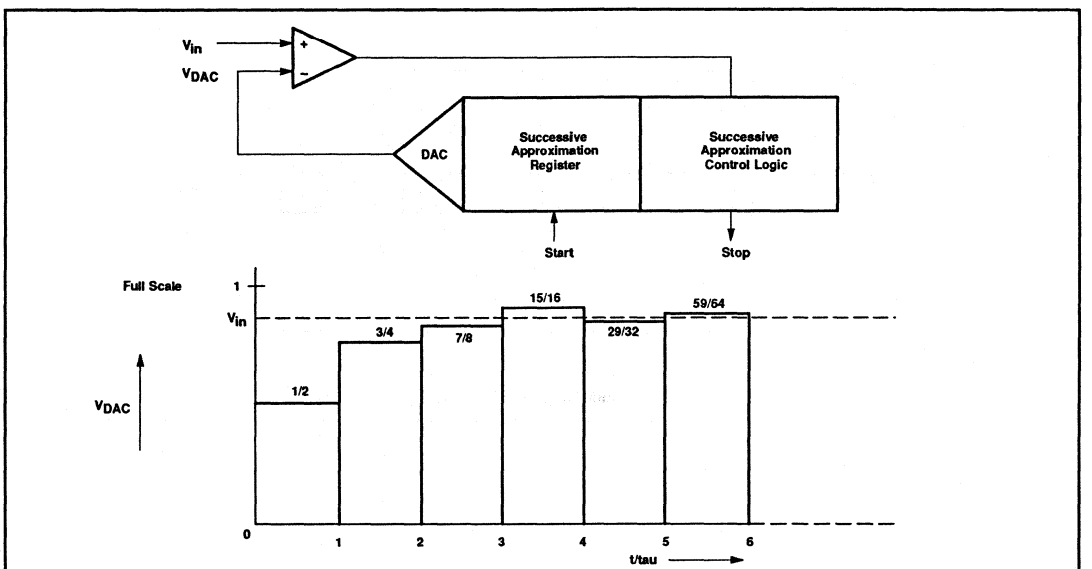


Figure 53. Successive Approximation ADC

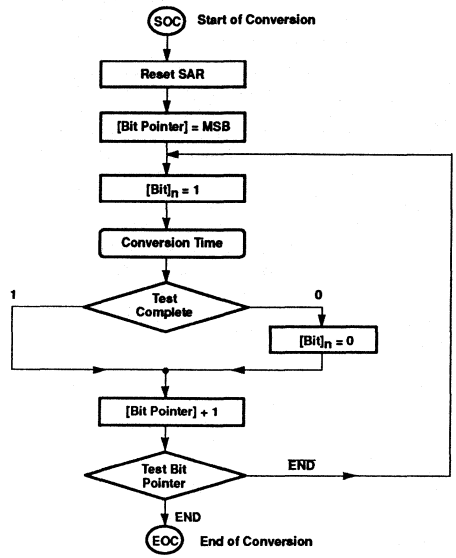


Figure 54. A/D Conversion Flowchart

ADCON (C5H)

7	6	5	4	3	2	1	0
ADC.1	ADC.0	ADEX	ADCI	ADCS	AADR2	AADR1	AADR0
(MSB)							(LSB)

Bit	Symbol	Function
ADCON.7	ADC.1	Bit 1 of ADC result
ADCON.6	ADC.0	Bit 0 of ADC result
ADCON.5	ADEX	Enable external start of conversion by STADC 0 = Conversion can be started by software only (by setting ADCS) 1 = Conversion can be started by software or externally (by a rising edge on STADC)
ADCON.4	ADCI	ADC interrupt flag: this flag is set when an A/D conversion result is ready to be read. An interrupt is invoked if it is enabled. The flag may be cleared by the interrupt service routine. While this flag is set, the ADC cannot start a new conversion. ADCI cannot be set by software.
ADCON.3	ADCS	ADC start and status: setting this bit starts an A/D conversion. It may be set by software or by the external signal STADC. The ADC logic ensures that this signal is HIGH while the ADC is busy. On completion of the conversion, ADCS is reset at the same time the interrupt flag is set. ADCS cannot be reset by software. A new conversion may not be started while either ADCS or ADCI is high.

ADCI	ADCS	ADC Status
0	0	ADC not busy; a conversion can be started
0	1	ADC busy; start of a new conversion is blocked
1	0	Conversion completed; start of a new conversion is blocked
1	1	Not possible

ADCON.2 ADCON.1 ADCON.0	AADR2 AADR1 AADR0	Analogue input select: this binary coded address selects one of the eight analogue port bits of P5 to be input to the converter. It can only be changed when ADCI and ADCS are both LOW.

AADR2	AADR1	AADR0	Selected Analog Channel
0	0	0	ADC0 (P5.0)
0	0	1	ADC1 (P5.1)
0	1	0	ADC2 (P5.2)
0	1	1	ADC3 (P5.3)
1	0	0	ADC4 (P5.4)
0	1	0	ADC5 (P5.5)
1	1	0	ADC6 (P5.6)
1	1	1	ADC7 (P5.7)

Figure 55. ADC Control Register (ADCON)

ADC Resolution and Analog Supply: Figure 56 shows how the ADC is realized. The ADC has its own supply pins (AV_{DD} and AV_{SS}) and two pins (Vref+ and Vref–) connected to each end of the DAC's resistance-ladder. The ladder has 1023 equally spaced taps, separated by a resistance of "R". The first tap is located 0.5 x R above Vref–, and the last tap is located 1.5 x R below Vref+. This gives a total ladder resistance of 1024 x R. This structure ensures that the DAC is monotonic and results in a symmetrical quantization error as shown in Figure 57.

For input voltages between Vref– and (Vref–) + 1/2 LSB, the 10-bit result of an A/D conversion will be 00 0000 0000B = 000H. For input voltages between (Vref+) – 3/2 LSB and Vref+, the result of a conversion will be 11 1111 1111B = 3FFH. AVref+ and AVref– may be between AV_{DD} + 0.2V and AV_{SS} – 0.2V. AVref+ should be positive with respect to AVref–, and the input voltage (Vin) should be between AVref+ and AVref–. If the analog input voltage range is from 2V to 4V, then 10-bit resolution can be obtained over this range if AVref+ = 4V and AVref– = 2V.

The result can always be calculated from the following formula:

$$\text{Result} = 1024 \times \frac{\text{Vin} - \text{AVref–}}{\text{AVref+} - \text{AVref–}}$$

Power Reduction Modes

The 8XC552 has two reduced power modes of operation: the idle mode and the power-down mode. These modes are entered by setting bits in the PCON special function register. When the 8XC552 enters the idle mode, the following functions are disabled:

CPU	(halted)
Timer T2	(halted and reset)
PWM0, PWM1	(reset; outputs are high)
ADC	(conversion aborted if in progress).

In idle mode, the following functions remain active:

Timer 0
Timer 1
Timer T3
SIO0 SIO1
External interrupts

When the 8XC552 enters the power-down mode, the oscillator is stopped. The power-down mode is entered by setting the PD bit in the PCON register. The PD bit can only be set if the $\overline{EW}$ input is tied HIGH.

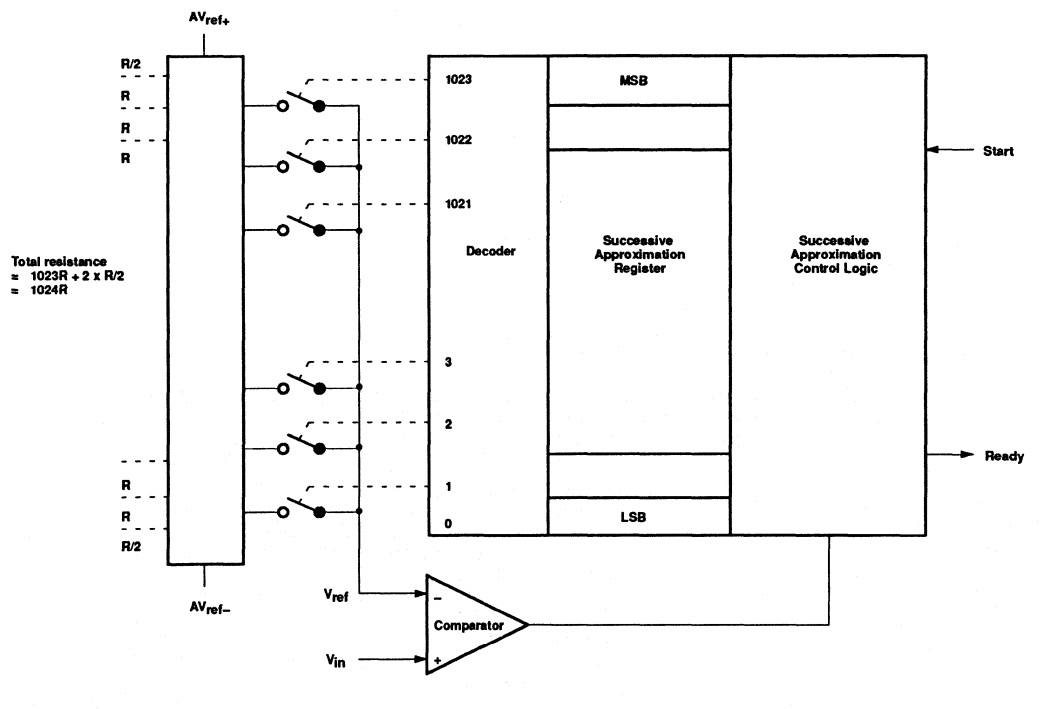


Figure 56. ADC Realization

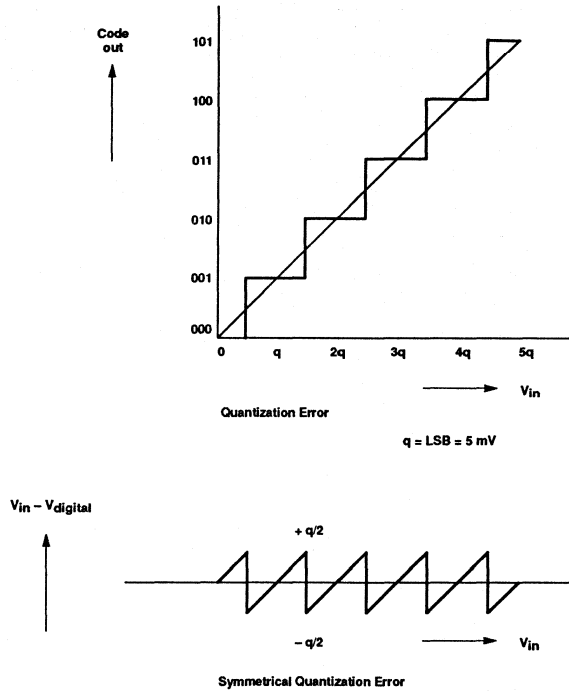


Figure 57. Effective Conversion Characteristic

Power-Down Mode: The instruction that sets PCON.1 will be the last instruction executed in the normal operating mode before the power-down mode is entered. In the power-down mode, the on-chip oscillator is stopped. This freezes all functions; only the on-chip RAM and special function registers are held. The port pins output the contents of their respective special function registers. A hardware reset is the only way to terminate the power-down mode. Reset re-defines all the special function registers, but does not change the on-chip RAM.

In the power-down mode, V_{DD} and AV_{DD} can be reduced to minimize power consumption. V_{DD} and AV_{DD} must not be reduced before the power-down mode is entered and must be restored to the normal operating voltage before the power-down mode is terminated. The reset that terminates the power-down mode also freezes the oscillator. The reset should not be activated before V_{DD} and AV_{DD} are restored to their normal operating level, and must be held active long enough to allow the oscillator to restart and stabilize (normally less than 10ms).

The status of the external pins during power-down is shown in Table 24. If the power-down mode is entered while the 8XC552 is executing out of external program memory, the port data that is held in the P2 special function register is restored to port 2. If a port latch contains a "1", the port pin is held HIGH during the power-down mode by the strong pull-up transistor.

Power Control Register PCON: The idle and power-down modes are entered by writing to bits in PCON. PCON is not bit addressable. See Figure 58.

Memory Organization

The memory organization of the 8XC552 is the same as in the 80C51, with the exception that the 8XC552 has 8k ROM, 256 bytes RAM, and additional SFRs. Addressing modes are the same in the 8XC552 and the 80C51. Details of the differences are given in the following paragraphs.

In the 8XC552, the lower 8k of the 64k program memory address space is filled by internal ROM. By tying the $\overline{EA}$ pin high, the processor fetches instructions from internal program ROM. Bus expansion for accessing program memory from 8k upwards is automatic since external instruction fetches occur automatically when the program counter exceeds 8191. If the $\overline{EA}$ pin is tied low, all program memory fetches are from external memory. The execution speed of the 8XC552 is the same regardless of whether fetches are from external or internal program memory. If all storage is on-chip, then byte location 8191 should be left vacant to prevent an undesired pre-fetch from external program memory address 8192.

Table 24. External Pin Status During Idle and Power-Down Modes

MODE	MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3	PORT 4	$\overline{PWM0}$/$\overline{PWM1}$
Idle (1)	Internal	1	1	Port data	Port data	Port data	Port data	Port data	HIGH
Idle (1)	External	1	1	Floating	Port data	Address	Port data	Port data	HIGH
Power-down	Internal	0	0	Port data	Port data	Port data	Port data	Port data	HIGH
Power-down	External	0	0	Floating	Port data	Port data	Port data	Port data	HIGH

Certain locations in program memory are reserved for specific programs. Locations 0000H to 0002H are reserved for the initialization program. Following reset, the CPU always begins execution at locations 0000H. Locations 0003H to 0075H are reserved for the fifteen interrupt request service routines.

Functionally, the internal data memory is the most flexible of the address spaces. The internal data memory space is subdivided into a 256-byte internal data RAM address space and a 128-byte special function register (SFR) address space, as shown in Figure 59.

The internal data RAM address space is 0 to 255. Four 8-bit register banks occupy locations 0 to 31. 128 bit locations of the internal data RAM are accessible through direct addressing. These bits reside in 16 bytes of internal data RAM at locations 20H to 2FH. The stack can be located anywhere in the internal data RAM address space by loading the 8-bit stack pointer. The stack depth may be 256 bytes maximum.

The SFR address space is 128 to 255. All registers except the program counter and the four 8-bit register banks reside in this address space. Memory mapping the SFRs allows them to be accessed as easily as internal RAM, and as such, they can be operated on by most instructions. The 56 SFRs are listed in Figure 60, and their mapping in the SFR address space is shown in Figures 61 and 62. RAM bit addresses are the same as in the 80C51 and are summarized in Figure 63. The special function bit addresses are summarized in Figure 64.

	7	6	5	4	3	2	1	0
PCON (87H)	SMOD	–	–	WLE	GF1	GF0	PD	IDL

Bit	Symbol	Function
PCON.7	SMOD	Double Baud rate bit. When set to logic 1 the baud rate is doubled when the serial port SIO0 is being used in modes 1, 2, or 3.
PCON.6	–	(Reserved)
PCON.5	–	(Reserved)
PCON.4	WLE	Watchdog load enable. This flag must be set by software prior to loading timer T3 (watchdog timer). It is cleared when timer T3 is loaded.
PCON.3	GF1	General-purpose flag bit
PCON.2	GF0	General-purpose flag bit
PCON.1	PD	Power-down bit. Setting this bit activates the power-down mode. It can only be set if input EW is high.
PCON.0	IDL	Idle mode bit. Setting this bit activates the idle mode.

If logic 1s are written to PD and IDL at the same time, PD takes precedence. The reset value of PCON is (0XX00000).

Figure 58. Power Control Register (PCON)

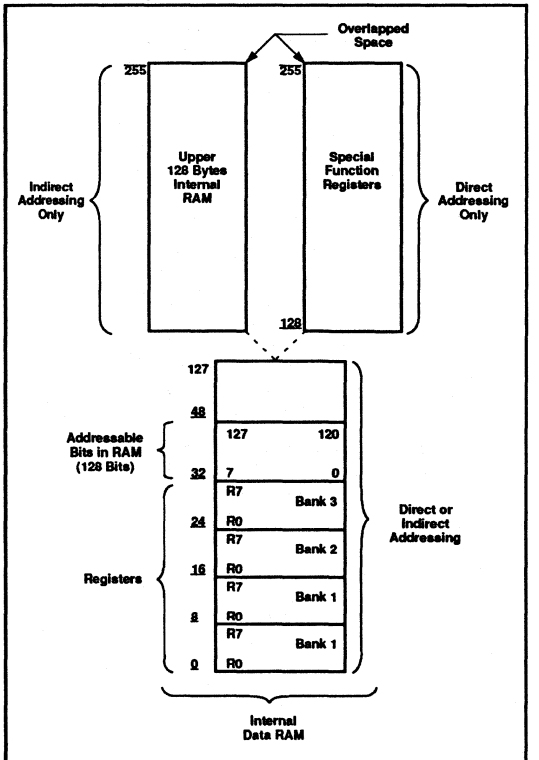


Figure 59. Internal Data Memory Address Space

ARITHMETIC REGISTERS:
ACCumulator,* B register,*
Program Status Word*

POINTERS:
Stack Pointer,
Data Pointer (High and Low)

PARALLEL I/O PORTS:
Port 5,* Port 4,*Port 3,*
Port 2,* Port 1,* Port 0*

INTERRUPT SYSTEM:
Interrupt Priority 0,*
Interrupt Priority 1,*
Interrupt Enable 0,*
Interrupt Enable 1*

PULSE WIDTH MODULATED O/Ps:
Pulse Width Modulation Prescaler
Pulse Width Modulation Register 0,
Pulse Width Modulation Register 1

SERIAL I/O PORTS:
Serial 0 CONtrol,* Serial 0 data BUFfer,
Serial 1 CONtrol,* Serial 1 DATa,
Serial 1 STAtus, Serial 1 ADDress, PCON

TIMERS:
Timer MODe, Timer CONtrol,*
Timer Low 0, Timer High 0,
Timer Low 1, Timer High 1,
TiMer T2 CONtrol, TiMer Low 2,
Timer High 2, Timer T3

CAPTURE AND COMPARE LOGIC:
CapTure CONtrol,
TiMer T2 Interrupt flag Register,
CapTure Low 0, CapTure High 0,
CapTure Low 1, CapTure High 1,
CapTure Low 2, CapTure High 2,
CapTure Low 3, CapTure High 3,
CoMpare Low 0, CoMpare High 0,
CoMpare Low 1, CoMpare High 1,
CoMpare Low 2, CoMpare High 2
SeT Enable, ReseT Enable

ADC
ADC cONtrol, ADC High byte

*NOTE: Bit and byte addressable

Figure 60. Special Function Registers

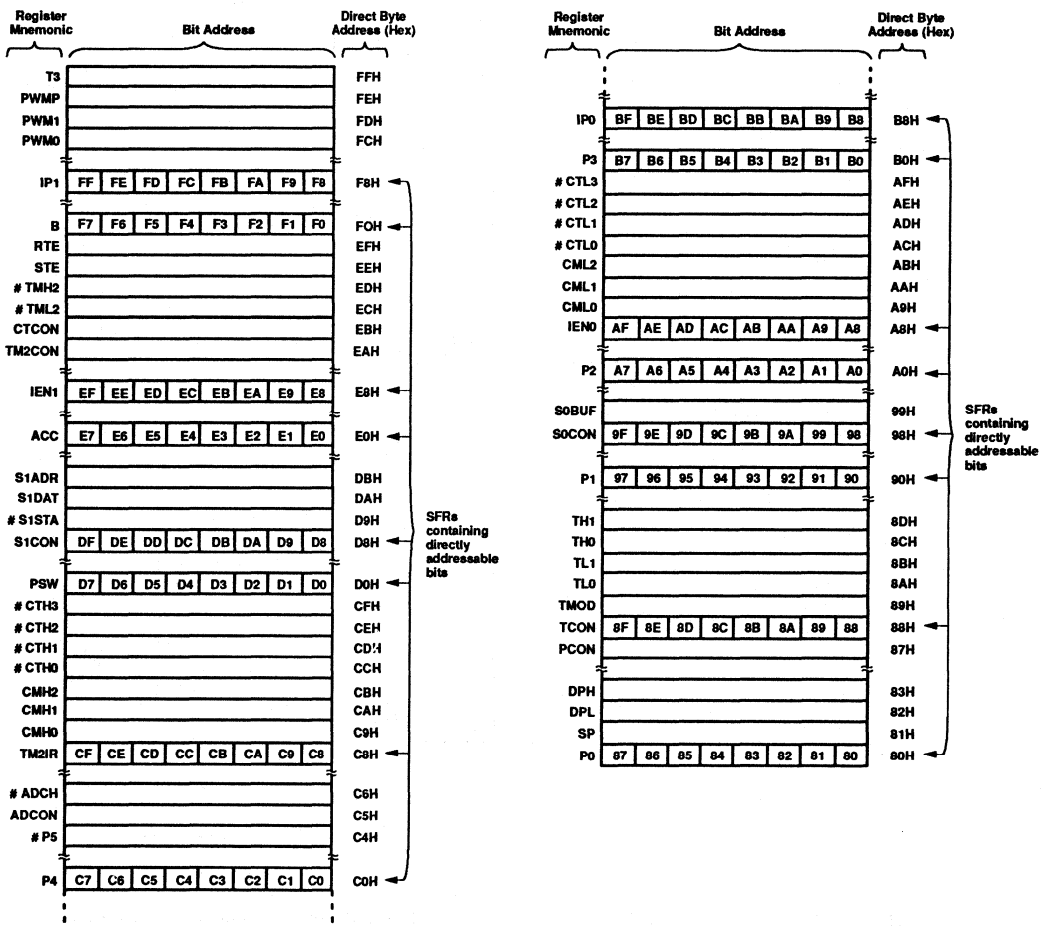


Figure 61. Mapping of Special Function Registers

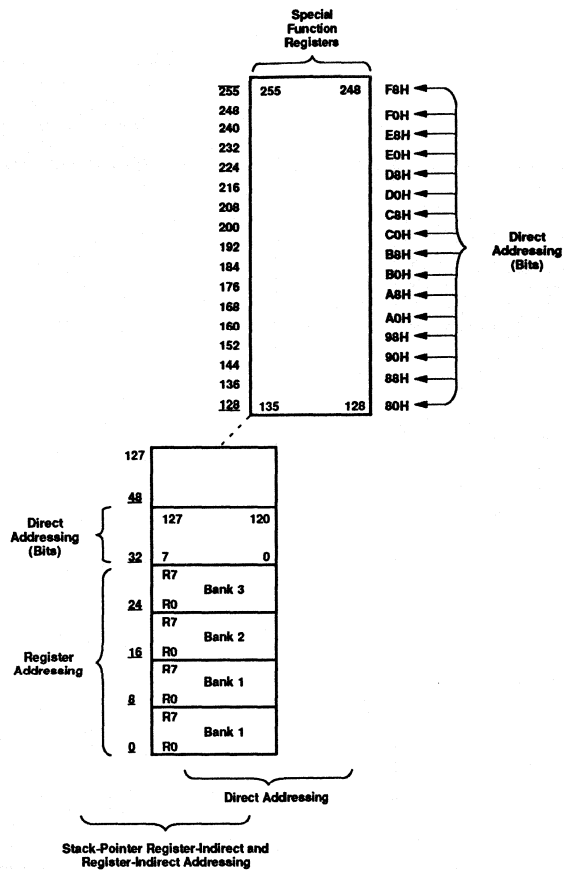


Figure 62. Special Function Register Bit Address

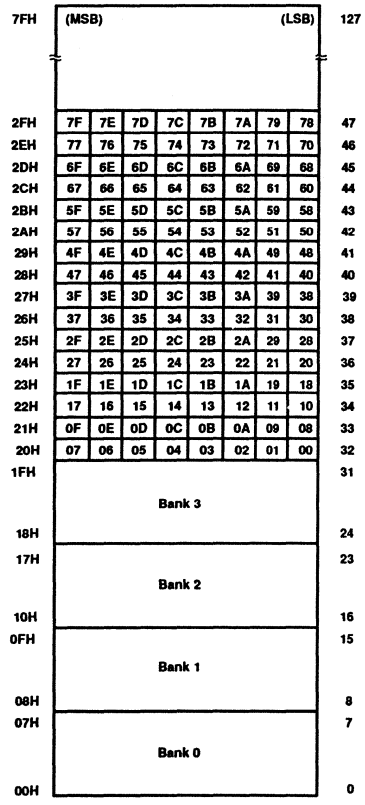


Figure 63. RAM Bit Addresses

Direct Byte Address (Hex)	Bit Addresses								Register Mnemonic
	PT2	PCM2	PCM1	PCM0	PCT3	PCT2	PCT1	PCT0	
F8H	FF	FE	FD	FC	FB	FA	F9	F8	IP1
F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
	ET2	ECM2	ECM1	ECM0	ECT3	ECT2	ECT1	ECT0	
E8H	EF	EE	ED	EC	EB	EA	E9	E8	IEN1
E0H	E7	E6	E5	E4	E3	E2	E1	E0	ACC
	CR2	ENS1	STA	STO	SI	AA	CR1	CR0	
D8H	DF	DE	DD	DC	DB	DA	D9	D8	S1CON
	CY	AC	F0	RS1	RS0	OV	F1	P	
D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
	T2OV	CMI2	CMI1	CMI0	CTI3	CTI2	CTI1	CTI0	
C8H	CF	CE	CD	CC	CB	CA	C9	C8	TM2IR
C0H	C7	C6	C5	C4	C3	C2	C1	C0	P4
	–	PAD	PS1	PS0	PT1	PX1	PT0	PX0	
B8H	BF	BE	BD	BC	BB	BA	B9	B8	IP0
B0H	B7	B6	B5	B4	B3	B2	B1	B0	P3
	EA	EAD	ES1	ES0	ET1	EX1	ET0	EX0	
A8H	AF	AE	AD	AC	AB	AA	A9	A8	IEN0
A0H	A7	A6	A5	A4	A3	A2	A1	A0	P2
	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	
98H	9F	9E	9D	9C	9B	9A	99	98	SOCON
90H	97	96	95	94	93	92	91	90	P1
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
80H	87	86	85	84	83	82	81	80	P0

Figure 64. Special Function Register Bit Address

Date of Issue	September 6, 1990
Status	Product Specification
Application Specific Product	

80C552/83C552/87C552

Single-chip 8-bit microcontroller with 10-bit A/D, capture/compare timer, high-speed outputs, PWM

DESCRIPTION

The 80C552/83C552/87C552 (hereafter generically referred to as 8XC552) Single-Chip 8-Bit Microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. The 8XC552 has the same instruction set as the 80C51. Three versions of the derivative exist:

- 83C552 — 8k bytes mask programmable ROM
- 80C552 — ROMless version of the 83C552
- 87C552 — 8k bytes EPROM

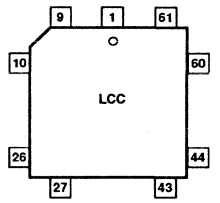
The 8XC552 contains a non-volatile 8k x 8 read-only program memory (83C552) EPROM (87C552), a volatile 256 x 8 read/write data memory, five 8-bit I/O ports, one 8-bit input port, two 16-bit timer/event counters (identical to the timers of the 80C51), an additional 16-bit timer coupled to capture and compare latches, a 15-source, two-priority-level, nested interrupt structure, an 8-input ADC, a dual DAC pulse width modulated interface, two serial interfaces (UART and I^2C-bus), a "watchdog" timer and on-chip oscillator and timing circuits. For systems that require extra capability, the 8XC552 can be expanded using standard TTL compatible memories and logic.

In addition, the 8XC552 has two software selectable modes of power reduction—idle mode and power-down mode. The idle mode freezes the CPU while allowing the RAM, timers, serial ports, and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.

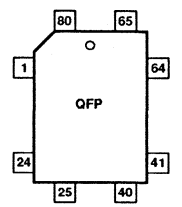
FEATURES

- 80C51 central processing unit
- 8k x 8 ROM expandable externally to 64k bytes
- An additional 16-bit timer/counter coupled to four capture registers and three compare registers
- Two standard 16-bit timer/counters
- 256 x 8 RAM, expandable externally to 64k bytes
- Capable of producing 8 synchronized, timed outputs
- A 10-bit ADC with 8 multiplexed analog inputs
- Two 8-bit resolution, pulse width modulation outputs
- Five 8-bit I/O ports plus one 8-bit input port shared with analog inputs
- I^2C-bus serial I/O port with byte oriented master and slave functions
- Full-duplex UART compatible with the standard 80C51
- On-chip watchdog timer
- Two speed ranges:
  - 12MHz
  - 16MHz
- Extended temperature ranges
- OTP package available

PIN CONFIGURATION



Pin	Function	Pin	Function
1	P5.0/ADC0	35	XTAL1
2	V_{DD}	36	V_{SS}
3	STADC	37	V_{SS}
4	$\overline{PWM0}$	38	NC
5	$\overline{PWM1}$	39	P2.0/A08
6	$\overline{EW}$	40	P2.1/A09
7	P4.0/CMSR0	41	P2.2/A10
8	P4.1/CMRS1	42	P2.3/A11
9	P4.2CMSR2	43	P2.4/A12
10	P4.3/CMSR3	44	P2.5/A13
11	P4.4/CMSR4	45	P2.6/A14
12	P4.5/CMSR5	45	P2.7/A15
13	P4.6/CMT0	47	$\overline{PSEN}$
14	P4.7/CMT1	48	ALE/$\overline{PROG}$
15	RST	49	$\overline{EA}/V_{PP}$
16	P1.0/CT0I	50	P0.7/AD7
17	P1.1/CT1I	51	P0.6/AD6
18	P1.2/CT2I	52	P0.5/AD5
19	P1.3/CT3I	53	P0.4/AD4
20	P1.4/T2	54	P0.3/AD3
21	P1.5/RT2	55	P0.2/AD2
22	P1.6/SCL	56	P0.1/AD1
23	P1.7/SDA	57	P0.0/AD0
24	P3.0/RxD	58	AVref−
25	P3.1TxD	59	AVref+
26	P3.2/$\overline{INT0}$	60	AV_{SS}
27	P3.3/$\overline{INT1}$	61	AV_{DD}
28	P3.4/T0	62	P5.7/ADC7
29	P3.5/T1	63	P5.6/ADC6
30	P3.6/$\overline{WR}$	64	P5.5/ADC5
31	P3.7/$\overline{RD}$	65	P5.4/ADC4
32	NC	66	P5.3/ADC3
33	NC	67	P5.2/ADC2
34	XTAL2	68	P5.1/ADC1

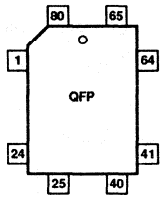


SEE PAGE 347 FOR QFP PIN FUNCTIONS.

PART NUMBER SELECTION

PHILIPS		PHILIPS COMPONENTS-SIGNETICS			TEMPERATURE °C	FREQ.
ROMless	ROM	ROMless	ROM	EPROM	AND PACKAGE	MHz
				S87C552-1A68	0 to +70, PLCC 68	12
				S87C552-1K68	0 to +70, CLCC 68	12
				S87C552-1B80	0 to +70, PQFP 80	12
				S87C552-2A68	–40 to +85, PLCC 68	12
				S87C552-2K68	–40 to +85, CLCC 68	12
				S87C552-2B80	–40 to +85, PQFP 80	12
PCB80C552-4WP	PCB83C552-4WP	S80C552-1A68	S83C552-1A68	S87C552-4A68	0 to +70, PLCC 68	16
				S87C552-4K68	0 to +70, CLCC 68	16
PCB80C552-4H	PCB83C552-4H	S80C552-1B80	S83C552-1B80	S87C552-4B80	0 to +70, PQFP 80	16
PCF80C552-4WP	PCF83C552-4WP	S80C552-2A68	S83C552-2A68	S87C552-5A68	–40 to +85, PLCC 68	16
				S87C552-5K68	–40 to +85, CLCC 68	16
PCF80C552-4H	PCF83C552-4H	S80C552-2B80	S83C552-2B80	S87C552-5B80	–40 to +85, PQFP 80	16
PCA80C552-4WP	PCA83C552-4WP	S80C552-6A68	S83C552-6A68		–40 to +125, PLCC 68	16
PCA80C552-4H	PCA83C552-4H	S80C552-6B80	S83C552-6B80		–40 to +125, PQFP 80	16

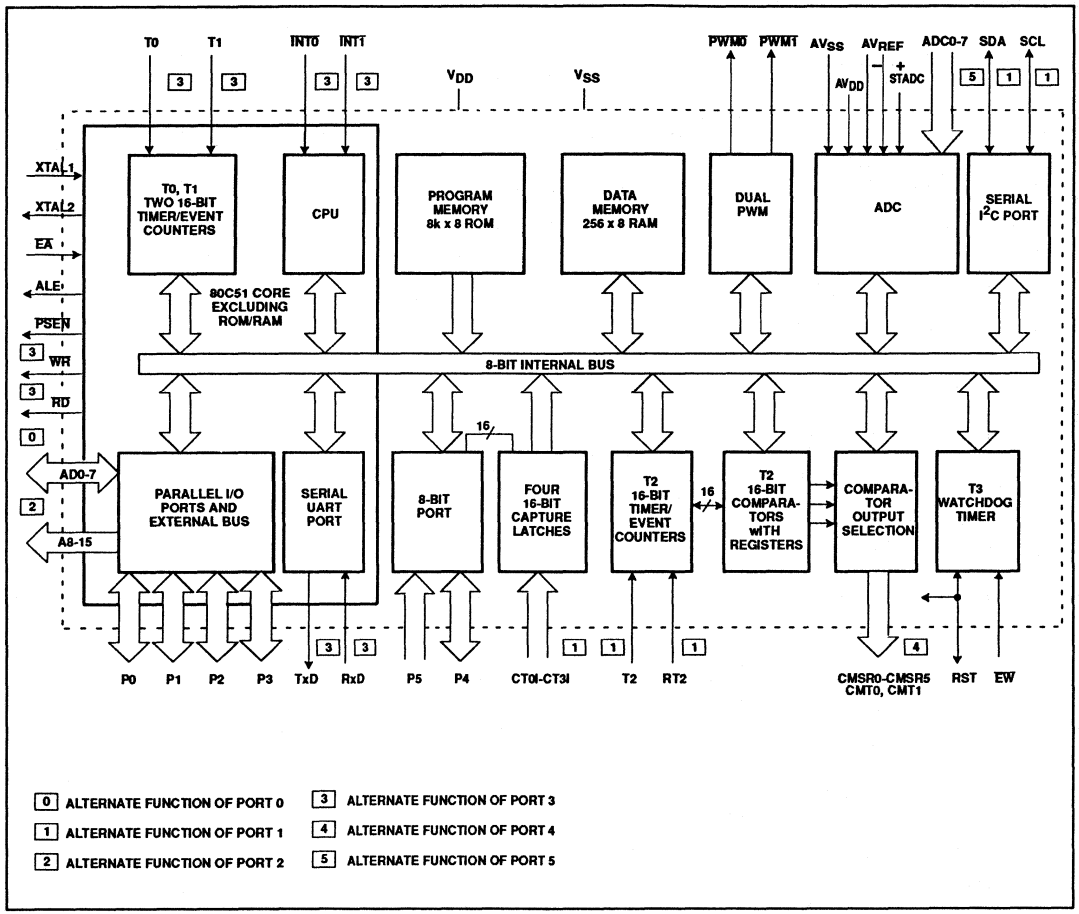
QFP PIN FUNCTIONS



Pin	Function	Pin	Function
1	P4.1/CMSR1	41	P2.3/A11
2	P4.2/CMSR2	42	P2.4/A12
3	NC	43	NC
4	P4.3/CMSR3	44	NC
5	P4.4/CMSR4	45	P2.5/A13
6	P4.5/CMSR5	46	P2.6/A14
7	P4.6/CMT0	47	P2.7/A15
8	P4.7/CMT1	48	$\overline{PSEN}$
9	RST	49	ALE
10	P1.0/CT0I	50	$\overline{EA}$
11	P1.1/CT1I	51	P0.7/AD7
12	P1.2/CT2I	52	P0.6/AD6
13	P1.3/CT3I	53	P0.5/AD5
14	P1.4/T2	54	P0.4/AD4
15	P1.5/RT2	55	P0.3/AD3
16	P1.6/SCL	56	P0.2/AD2
17	P1.7/SDA	57	P0.1/AD1
18	P3.0/RXD	58	P0.0/AD0
19	P3.1/TXD	59	AVref–
20	P3.2/$\overline{INT0}$	60	AVref+
21	NC	61	AV_{ss}
22	NC	62	NC
23	P3.3/$\overline{INT1}$	63	AV_{DD}
24	P3.4/T0	64	P5.7/ADC7
25	P3.5/T1	65	P5.6/ADC6
26	P3.6/$\overline{WR}$	66	P5.5/ADC5
27	P3.7/$\overline{RD}$	67	P5.4/ADC4
28	NC	68	P5.3/ADC3
29	NC	69	P5.2/ADC2
30	NC	70	P5.1/ADC1
31	XTAL2	71	P5.0/ADC0
32	XTAL1	72	V_{DD}
33	IC	73	IC
34	V_{SS}	74	STADC
35	V_{SS}	75	$\overline{PWM0}$
36	V_{SS}	76	$\overline{PWM1}$
37	NC	77	$\overline{EW}$
38	P2.0/A08	78	NC
39	P2.1/A09	79	NC
40	P2.2/A10	80	P4.0/CMSR0

NC = not connected
IC = internally connected (do not use)

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. PLCC	PIN NO. QFP	TYPE	NAME AND FUNCTION
V_{DD}	2	72	I	**Digital Power Supply:** +5V power supply pin during normal operation, idle and power-down mode.
STADC	3	74	I	**Start ADC Operation:** Input starting analog to digital conversion (ADC operation can also be started by software).
$\overline{PWM0}$	4	75	O	**Pulse Width Modulation:** Output 0.
$\overline{PWM1}$	5	76	O	**Pulse Width Modulation:** Output 1.
$\overline{EW}$	6	77	I	**Enable Watchdog Timer:** Enable for T3 watchdog timer and disable power-down mode.
P0.0-P0.7	57-50	58-51	I/O	**Port 0:** Port 0 is an 8-bit open-drain bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application it uses strong internal pull-ups when emitting 1s. Port 0 is also used to input the code byte during programming and to output the code byte during verification.
P1.0-P1.7	16-23	10-17	I/O	**Port 1:** 8-bit I/O port. Alternate functions include:
	16-21	10-15	I/O	**(P1.0-P1.5):** Quasi-bidirectional port pins.
	22-23	16-17	I/O	**(P1.6, P1.7):** Open drain port pins.
	16-19	10-13	I	**CT0I-CT3I (P1.0-P1.3):** Capture timer input signals for timer T2.
	20	14	I	**T2 (P1.4):** T2 event input
	21	15	I	**RT2 (P1.5):** T2 timer reset signal. Rising edge triggered.
	22	16	I/O	**SCL (P1.6):** Serial port clock line I^2C-bus.
	23	17	I/O	**SDA (P1.7):** Serial port data line I^2C-bus.
				Port 1 is also used to input the lower order address byte during EPROM programming and verification. A0 is on P1.0, etc.
P2.0-P2.7	39-46	38-42, 45-47	I/O	**Port 2:** 8-bit quasi-bidirectional I/O port. Alternate function: High-order address byte for external memory (A08-A15). Port 2 is also used to input the upper order address during EPROM programming and verification. A8 is on P2.0, A9 on P2.1, through A12 on P2.4.
P3.0-P3.7	24-31	18-20, 23-27	I/O	**Port 3:** 8-bit quasi-bidirectional I/O port. Alternate functions include:
	24	18		**RxD(P3.0):** Serial input port.
	25	19		**TxD (P3.1):** Serial output port.
	26	20		**$\overline{INT0}$ (P3.2):** External interrupt.
	27	23		**$\overline{INT1}$ (P3.3):** External interrupt.
	28	24		**T0 (P3.4):** Timer 0 external input.
	29	25		**T1 (P3.5):** Timer 1 external input.
	30	26		**$\overline{WR}$ (P3.6):** External data memory write strobe.
	31	27		**$\overline{RD}$ (P3.7):** External data memory read strobe.
P4.0-P4.7	7-14	80, 1-2 4-8	I/O	**Port 4:** 8-bit quasi-bidirectional I/O port. Alternate functions include:
	7-12	80, 1-2 4-6	O	**CMSR0-CMSR5 (P4.0-P4.5):** Timer T2 compare and set/reset outputs on a match with timer T2.
	13, 14	7, 8	O	**CMT0, CMT1 (P4.6, P4.7):** Timer T2 compare and toggle outputs on a match with timer T2.
P5.0-P5.7	68-62, 1	71-64,	I	**Port 5:** 8-bit input port. **ADC0-ADC7 (P5.0-P5.7):** Alternate function: Eight input channels to ADC.
RST	15	9	I/O	**Reset:** Input to reset the 87C552. It also provides a reset pulse as output when timer T3 overflows.
XTAL1	35	32	I	**Crystal Input 1:** Input to the inverting amplifier that forms the oscillator, and input to the internal clock generator. Receives the external clock signal when an external oscillator is used.
XTAL2	34	31	O	**Crystal Input 2:** Output of the inverting amplifier that forms the oscillator. Left open-circuit when an external clock is used.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO. PLCC	PIN NO. QFP	TYPE	NAME AND FUNCTION
V_{SS}	36, 37	34-36	I	**Digital ground.**
$\overline{PSEN}$	47	48	O	**Program Store Enable:** Active-low read strobe to external program memory.
ALE/$\overline{PROG}$	48	49	O	**Address Latch Enable:** Latches the low byte of the address during accesses to external memory. It is activated every six oscillator periods. During an external data memory access, one ALE pulse is skipped. ALE can drive up to eight LS TTL inputs and handles CMOS inputs without an external pull-up. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.
$\overline{EA}$/V_{PP}	49	50	I	**External Access:** When $\overline{EA}$ is held at TTL level high, the CPU executes out of the internal program ROM provided the program counter is less than 8192. When $\overline{EA}$ is held at TTL low level, the CPU executes out of external program memory. $\overline{EA}$ is not allowed to float. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
AV_{REF-}	58	59	I	**Analog to Digital Conversion Reference Resistor:** Low-end.
AV_{REF+}	59	60	I	**Analog to Digital Conversion Reference Resistor:** High-end.
AV_{SS}	60	61	I	**Analog Ground**
AV_{DD}	61	63	I	**Analog Power Supply**

NOTE:
1. To avoid "latch-up" effect at power-on, the voltage on any pin at any time must not be higher or lower than V_{DD} + 0.5V or V_{SS} – 0.5V, respectively.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol, page 1.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-on reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-on, the voltage on V_{DD} and RST must come up at the same time for a proper start-up.

IDLE MODE

In the idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. The control bits for the reduced power modes are in the special function register PCON. Table 1 shows the state of the I/O ports during low current operating modes.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3	PORT 4	PWM0/ PWM1
Idle	Internal	1	1	Data	Data	Data	Data	Data	High
Idle	External	1	1	Float	Data	Address	Data	Data	High
Power-down	Internal	0	0	Data	Data	Data	Data	Data	High
Power-down	External	0	0	Float	Data	Data	Data	Data	High

Serial Control Register (S1CON) – See Table 2

S1CON (D8H)	CR2	ENS1	STA	STO	SI	AA	CR1	CR0

Bits CR0, CR1 and CR2 determine the serial clock frequency that is generated in the master mode of operation.

Table 2. Serial Clock Rates

CR2	CR1	CR0	BIT FREQUENCY (kHz) AT f_{OSC}			f_{OSC} DIVIDED BY
			6MHz	12MHz	16MHz	
0	0	0	23	47	62.5	256
0	0	1	27	54	71	224
0	1	0	31.25	62.5	83.3	192
0	1	1	37	75	100	160
1	0	0	6.25	12.5	17	960
1	0	1	50	100	133[1]	120
1	1	0	100	200[1]	267[1]	60
1	1	1	0.25 < 31.25	0.5 < 62.5	0.67 < 83.3	96 x (256 – (reload value Timer 1)) (Reload value range: 0 – 254 in mode 2)

NOTE:
1. These frequencies exceed the upper limit of 100kHz of the I²C-bus specification and cannot be used in an I²C-bus application.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Voltage on $\overline{EA}/V_{PP}$ to V_{SS} (87C552 only)	–0.5 to +13	V
Voltage on any other pin to V_{SS}	–0.5 to +6.5	V
Input, output DC current on any single I/O pin	5.0	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.0	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

V_{SS}, AV_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{DD}	Supply voltage				
	PCB8XC552		4.0	6.0	V
	PCF8XC552		4.0	6.0	V
	PCA8XC552		4.5	5.5	V
	87C552		4.5	5.5	V
I_{DD}	Supply current operating:	See notes 1 and 2			
	PCB8XC552	f_{OSC} = 16MHz		45	mA
	PCF8XC552	f_{OSC} = 16MHz		45	mA
	PCA8XC552	f_{OSC} = 16MHz		40	mA
	87C552	f_{OSC} = 12MHz		30	mA
I_{ID}	Idle mode:	See notes 1 and 3			
	PCB8XC552	f_{OSC} = 16MHz		10	mA
	PCF8XC552	f_{OSC} = 16MHz		10	mA
	PCA8XC552	f_{OSC} = 16MHz		9	mA
	87C552	f_{OSC} = 12MHz		7	mA
I_{PD}	Power-down current:	See notes 1 and 4; 2V < VPD < VDD max			
	PCB8XC552			50	µA
	PCF8XC552			50	µA
	PCA8XC552			150	µA
	87C552			50	µA
Inputs					
V_{IL}	Input low voltage, except $\overline{EA}$, P1.6, P1.7		–0.5	$0.2V_{DD}$–0.1	V
V_{IL1}	Input low voltage to $\overline{EA}$		–0.5	$0.2V_{DD}$–0.3	V
V_{IL2}	Input low voltage to P1.6/SCL, P1.7/SDA[5]		–0.5	$0.3V_{DD}$	V
V_{IH}	Input high voltage, except XTAL1, RST		$0.2V_{DD}$+0.9	V_{DD}+0.5	V
V_{IH1}	Input high voltage, XTAL1, RST		$0.7V_{DD}$	V_{DD}+0.5	V
V_{IH2}	Input high voltage, P1.6/SCL, P1.7/SDA[5]		$0.7V_{DD}$	6.0	V
$-I_{IL}$	Logical 0 input current, ports 1, 2, 3, 4, except P1.6, P1.7	V_{IN} = 0.45V		–50	µA
$-I_{TL}$	Logical 1-to-0 transition current, ports 1, 2, 3, 4, except P1.6, P1.7	See note 6		–650	µA
$\pm I_{IL1}$	Input leakage current, port 0, $\overline{EA}$, STADC, $\overline{EW}$	0.45V < V_I < V_{DD}		10	µA
$\pm I_{IL2}$	Input leakage current, P1.6/SCL, P1.7/SDA	0V < V_I < 6V 0V < V_{DD} < 5.5V		10	µA
Outputs					
V_{OL}	Output low voltage, ports 1, 2, 3, 4, except P1.6, P1.7	I_{OL} = 1.6mA[7]		0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$, $\overline{PWM0}$, $\overline{PWM1}$	I_{OL} = 3.2mA[7]		0.45	V
V_{OL2}	Output low voltage, P1.6/SCL, P1.7/SDA	I_{OL} = 3.0mA[7]		0.4	V
V_{OH}	Output high voltage, ports 1, 2, 3, 4, except P1.6/SCL, P1.7/SDA				V
		$-I_{OH}$ = 60µA	2.4		V
		$-I_{OH}$ = 25µA	$0.75V_{DD}$		V
		$-I_{OH}$ = 10µA	$0.9V_{DD}$		V
V_{OH1}	Output high voltage (port 0 in external bus mode, ALE, $\overline{PSEN}$, $\overline{PWM0}$, $\overline{PWM1}$)[8]				V
		$-I_{OH}$ = 400µA	2.4		V
		$-I_{OH}$ = 150µA	$0.75V_{DD}$		V
		$-I_{OH}$ = 40µA	$0.9V_{DD}$		V
V_{OH2}	High level output voltage (RST)				V
		$-I_{OH}$ = 400µA	2.4		V
		$-I_{OH}$ = 120µA	$0.8V_{DD}$		V

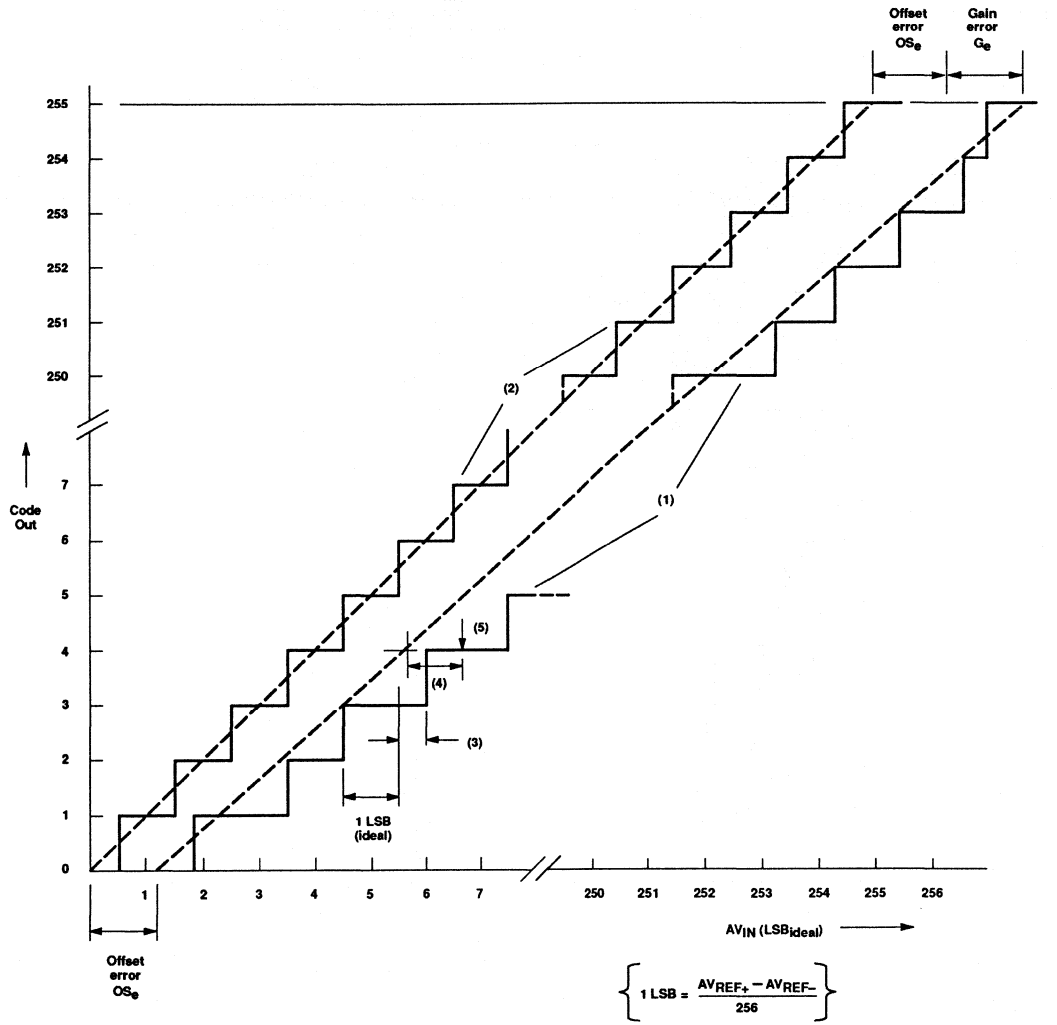
DC ELECTRICAL CHARACTERISTICS (Continued)

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
Outputs (Continued)					
R_{RST}	Internal reset pull-down resistor		50	150	kohm
C_{IO}	Pin capacitance	Test freq = 1MHz, T_A = 25°C		10	pF
Analog Inputs					
AV_{DD}	Analog supply voltage:				
	PCB8XC552	$AV_{DD} = V_{DD}\pm0.2V$	4.0	6.0	V
	PCF8XC552	$AV_{DD} = V_{DD}\pm0.2V$	4.0	6.0	V
	PCA8XC552	$AV_{DD} = V_{DD}\pm0.2V$	4.5	5.5	V
	87C552[9]	$AV_{DD} = V_{DD}\pm0.2V$	4.5	5.5	V
AI_{DD}	Analog supply current: operating:	Port 5 = 0 to AV_{DD}		1.2	mA
AI_{ID}	Idle mode:				
	PCB8XC552			50	µA
	PCF8XC552			50	µA
	PCA8XC552			100	µA
	87C552			50	µA
AI_{PD}	Power-down mode:	$2V < AV_{PD} < AV_{DD}$ max			
	PCB8XC552			50	µA
	PCF8XC552			50	µA
	PCA8XC552			100	µA
	87C552			50	µA
AV_{IN}	Analog input voltage		AV_{SS}–0.2	AV_{DD}+0.2	V
AV_{REF}	Reference voltage:				
	AV_{REF-}		AV_{SS}–0.2		V
	AV_{REF+}			AV_{DD}+0.2	V
R_{REF}	Resistance between AV_{REF+} and AV_{REF-}		10	50	kohms
C_{IA}	Analog input capacitance			15	pF
t_{ADS}	Sampling time			$8t_{CY}$	µs
t_{ADC}	Conversion time (including sampling time)			$50t_{CY}$	µs
DL_e	Differential non-linearity[10, 11, 12]			±1	LSB
IL_e	Integral non-linearity[10, 13]			±2	LSB
OS_e	Offset error[10, 14]			±2	LSB
G_e	Gain error[10, 15]			±0.4	%
A_e	Absolute voltage error[10, 16]			±3	LSB
M_{CTC}	Channel to channel matching			±1	LSB
C_t	Crosstalk between inputs of port 5[17]	0–100kHz		–60	dB

NOTES: See Next Page.

NOTES FOR DC ELECTRICAL CHARACTERISTICS:

1. See Figures 8 through 12 for I_{DD} test conditions.
2. The operating supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = $\overline{EW} = V_{DD}$; STADC = V_{SS}.
3. The idle mode supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = Port 0 = $\overline{EW} = V_{DD}$; RST = STADC = V_{SS}.
4. The power-down current is measured with all output pins disconnected; XTAL2 not connected; $\overline{EA}$ = Port 0 = $\overline{EW} = V_{DD}$; RST = STADC = XTAL1 = V_{SS}.
5. The input threshhold voltage of P1.6 and P1.7 (SIO1) meets the I2C specification, so an input voltage below 1.5V will be recognized as a logic 0 while an input voltage above 3.0V will be recognized as a logic 1.
6. Pins of ports 1 (except P1.6, P1.7), 2, 3, and 4 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
7. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input. I_{OL} can exceed these conditions provided that no single output sinks more than 5mA and no more than two outputs exceed the test conditions.
8. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{DD}$ specification when the address bits are stabilizing.
9. The following condition must not be exceeded: $V_{DD} - 0.2V < AV_{DD} < V_{DD} = 0.2V$.
10. Conditions: $AV_{REF-} = 0V$; $AV_{DD} = 5.0V$, $AV_{REF+} = 4.977V$. ADC is monotonic with no missing codes.
11. The differential non-linearity (DL_e) is the difference between the actual step width and the ideal step width. (See Figure 1.)
12. The ADC is monotonic; there are no missing codes.
13. The integral non-linearity (IL_e) is the peak difference between the center of the steps of the actual and the ideal transfer curve after appropriate adjustment of gain and offset error. (See Figure 1.)
14. The offset error (OS_e) is the absolute difference between the straight line which fits the actual transfer curve (after removing gain error), and a straight line which fits the ideal transfer curve. (See Figure 1.)
15. The gain error (G_e) is the relative difference in percent between the straight line fitting the actual transfer curve (after removing offset error), and the straight line which fits the ideal transfer curve. Gain error is constant at every point on the transfer curve. (See Figure 1.)
16. The absolute voltage error (A_e) is the maximum difference between the center of the steps of the actual transfer curve of the non-calibrated ADC and the ideal transfer curve.
17. This should be considered when both analog and digital signals are simultaneously input to port 5.



(1) Example of an actual transfer curve.
(2) The ideal transfer curve.
(3) Differential non-linearity (DL_e).
(4) Integral non-linearity (IL_e).
(5) Center of a step of the actual transfer curve.

Figure 1. ADC Conversion Characteristic

AC ELECTRICAL CHARACTERISTICS[1, 2]

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	2	Oscillator frequency[3]			1.2	16	MHz
t_{LHLL}	2	ALE pulse width	127		$2t_{CLCL}-40$		ns
t_{AVLL}	2	Address valid to ALE low	28		$t_{CLCL}-55$		ns
t_{LLAX}	2	Address hold after ALE low	48		$t_{CLCL}-35$		ns
t_{LLIV}	2	ALE low to valid instruction in		234		$4t_{CLCL}-100$	ns
t_{LLPL}	2	ALE low to $\overline{PSEN}$ low	43		$t_{CLCL}-40$		ns
t_{PLPH}	2	$\overline{PSEN}$ pulse width	205		$3t_{CLCL}-45$		ns
t_{PLIV}	2	$\overline{PSEN}$ low to valid instruction in		145		$3t_{CLCL}-105$	ns
t_{PXIX}	2	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	2	Input instruction float after $\overline{PSEN}$		59		$t_{CLCL}-25$	ns
t_{AVIV}	2	Address to valid instruction in		312		$5t_{CLCL}-105$	ns
t_{PLAZ}	2	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{AVLL}	3, 4	Address valid to ALE low	43		$t_{CLCL}-40$		ns
t_{RLRH}	3	$\overline{RD}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{WLWH}	4	$\overline{WR}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{RLDV}	3	$\overline{RD}$ low to valid data in		252		$5t_{CLCL}-165$	ns
t_{RHDX}	3	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	3	Data float after $\overline{RD}$		97		$2t_{CLCL}-70$	ns
t_{LLDV}	3	ALE low to valid data in		517		$8t_{CLCL}-150$	ns
t_{AVDV}	3	Address to valid data in		585		$9t_{CLCL}-165$	ns
t_{LLWL}	3, 4	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	3, 4	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}-130$		ns
t_{QVWX}	4	Data valid to $\overline{WR}$ transition	23		$t_{CLCL}-60$		ns
t_{DW}	4	Data before $\overline{WR}$	433		$7t_{CLCL}-150$		ns
t_{WHQX}	4	Data hold after $\overline{WR}$	33		$t_{CLCL}-50$		ns
t_{RLAZ}	3	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	3, 4	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock							
t_{CHCX}	5	High time[4]	20		20		ns
t_{CLCX}	5	Low time[4]	20		20		ns
t_{CLCH}	5	Rise time[4]		20		20	ns
t_{CHCL}	5	Fall time[4]		20		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. 87C552: $1/t_{CLCL}$ = 3.5 to 16 MHz.
4. These values are characterized but not 100% production tested.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

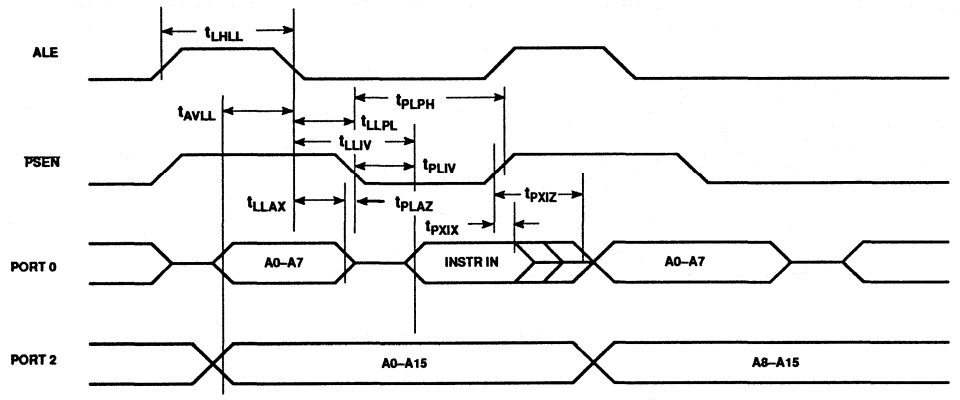


Figure 2. External Program Memory Read Cycle

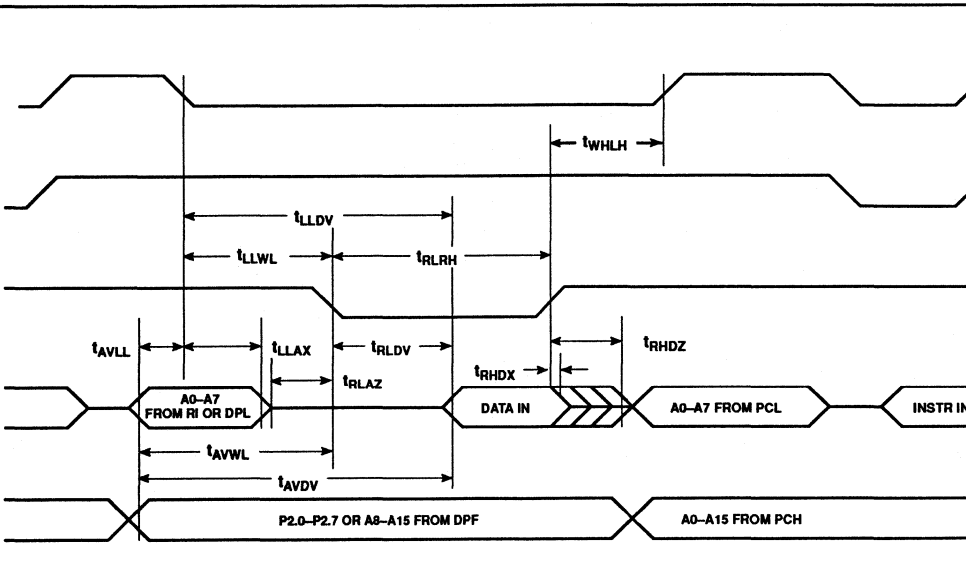


Figure 3. External Data Memory Read Cycle

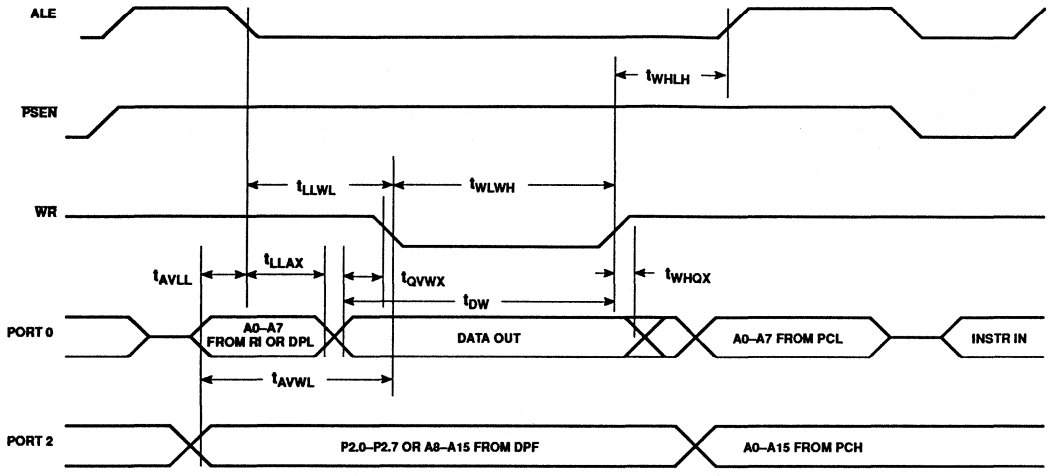


Figure 4. External Data Memory Write Cycle

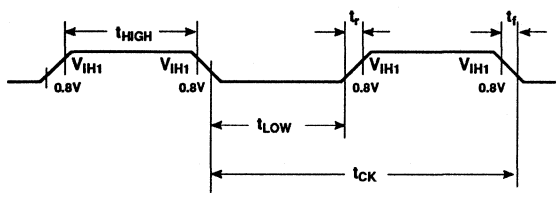
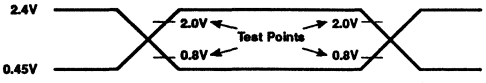
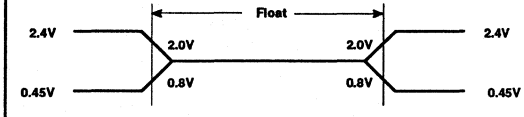


Figure 5. External Clock Drive XTAL1



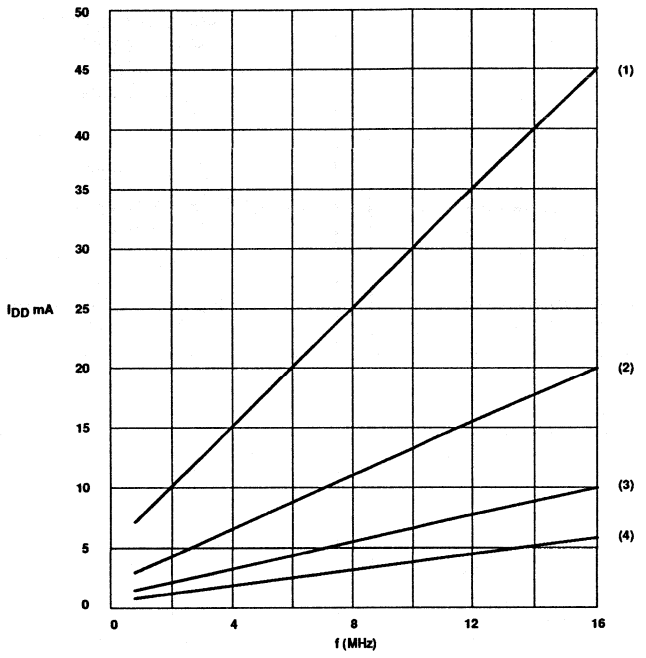
NOTE:
AC inputs during testing are driven at 2.4V for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at 2.0V for a logic '1' and 0.8V for a logic '0'.

Figure 6. AC Testing Input/Output



NOTE:
The float state is defined as the point at which a port 0 pin sinks 3.2mA or sources 400μA at the voltage test levels.

Figure 7. AC Testing Input, Float Waveform



(1) Maximum operating mode; V_{DD} = 6V.
(2) Maximum operating mode; V_{DD} = 4V.
(3) Maximum idle mode; V_{DD} = 6V.
(4) Maximum idle mode; V_{DD} = 4V.

NOTE:
These values are valid only within the frequency specifications of the device under test.

Figure 8. Supply Current (I_{DD}) as a Function of Frequency at XTAL1 (f_{OSC})

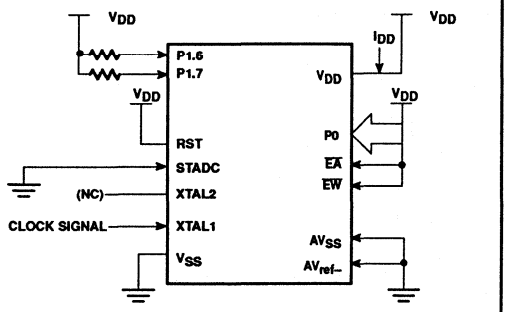


Figure 9. I_{DD} Test Condition, Active Mode
All other pins are disconnected[1]

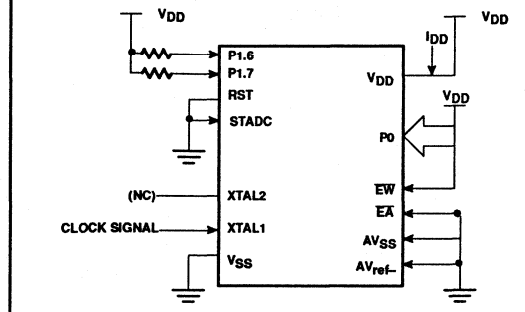


Figure 10. I_{DD} Test Condition, Idle Mode
All other pins are disconnected[2]

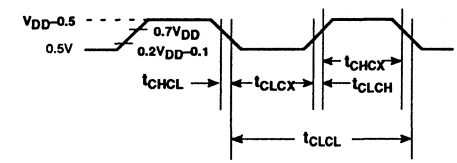


Figure 11. Clock Signal Waveform for I_{DD} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

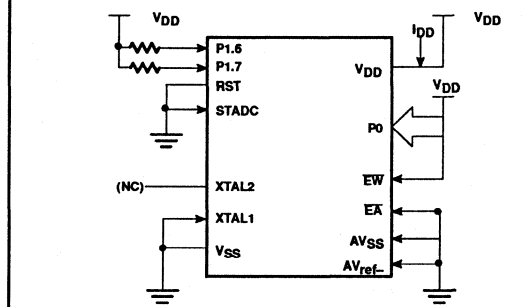
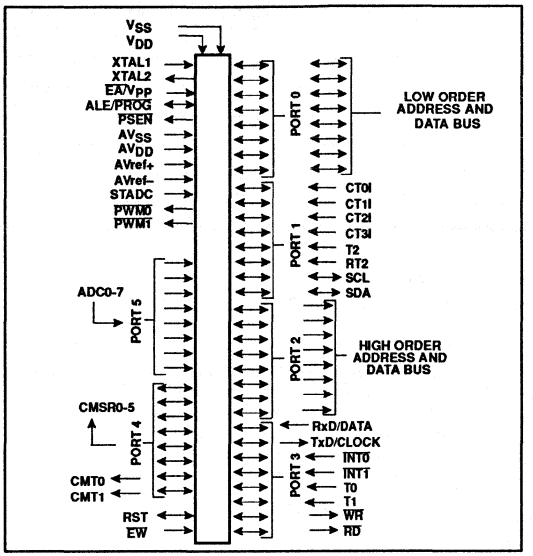


Figure 12. I_{DD} Test Condition, Power Down Mode
All other pins are disconnected. V_{DD} = 2V to 5.5V[3]

NOTES:

1. Active Mode:
   a. The following pins must be forced to V_{DD}: $\overline{EA}$, RST, Port 0, and $\overline{EW}$.
   b. The following pins must be forced to V_{SS}: STADC, AV_{ss}, and AV_{ref-}.
   c. Ports 1.6 and 1.7 should be connected to V_{DD} through resistors of sufficiently high value such that the sink current into these pins cannot exceed the I_{OL1} spec of these pins.
   d. The following pins must be disconnected: XTAL2 and all pins not specified above.
2. Idle Mode:
   a. The following pins must be forced to V_{DD}: Port 0 and $\overline{EW}$.
   b. The following pins must be forced to V_{SS}: RST, STADC, AV_{ss}, AV_{ref-}, and $\overline{EA}$.
   c. Ports 1.6 and 1.7 should be connected to V_{DD} through resistors of sufficiently high value such that the sink current into these pins cannot exceed the I_{OL1} spec of these pins. These pins must not have logic 0 written to them prior to this measurement.
   d. The following pins must be disconnected: XTAL2 and all pins not specified above.
3. Power Down Mode:
   a. The following pins must be forced to V_{DD}: Port 0 and $\overline{EW}$.
   b. The following pins must be forced to V_{SS}: RST, STADC, XTAL1, AV_{ss}, AV_{ref-}, and $\overline{EA}$.
   c. Ports 1.6 and 1.7 should be connected to V_{DD} through resistors of sufficiently high value such that the sink current into these pins cannot exceed the I_{OL1} spec of these pins. These pins must not have logic 0 written to them prior to this measurement.
   d. The following pins must be disconnected: XTAL2 and all pins not specified above.

LOGIC SYMBOL



EPROM CHARACTERISTICS

The 87C552 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C552 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C552 manufactured by Signetics.

Table 3 shows the logic levels for reading the signature byte, and for programming the program memory, theencryption table, and the lock bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 13 and 14. Figure 15 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 13. Note that the 87C552 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1 and 2, as shown in Figure 13. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$, and pins of ports 2 and 3 specified in Table 3 are held at the "Program Code Data" levels indicated in Table 3. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 14.

To program the encryption table, repeat the 25-pulse programming sequence for addresses 0 through 1FH, using the "Pgm Encryption Table" levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the lock bits, repeat the 25-pulse programming sequence using the "Pgm Lock Bit" levels. After one lock bit is programmed, further programming of the code memory and encryption table is disabled. However, the other lock bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If lock bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be red is applied to ports 1 and 2 as shown in Figure 15. The other pins are held at the "Verify Code Data" levels indicated in Table 3. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips Components

(031H) = 94H indicates 87C552

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 3, and which satisfies the timing specifications, is suitable.

™Trademark phrase of Intel Corporation.

Table 3. EPROM Programming Modes

MODE	RST	PSEN	ALE/PROG	EA/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signagure	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm lock bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm lock bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:
1. 0 = Valid low for that pin; 1 = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{DD} = 5V ±10% during programming and verification.

*ALE/PROG receives 25 programming pulses whill V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to the light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345-5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000µW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient. Erasure leaves the array in an all 1s state.

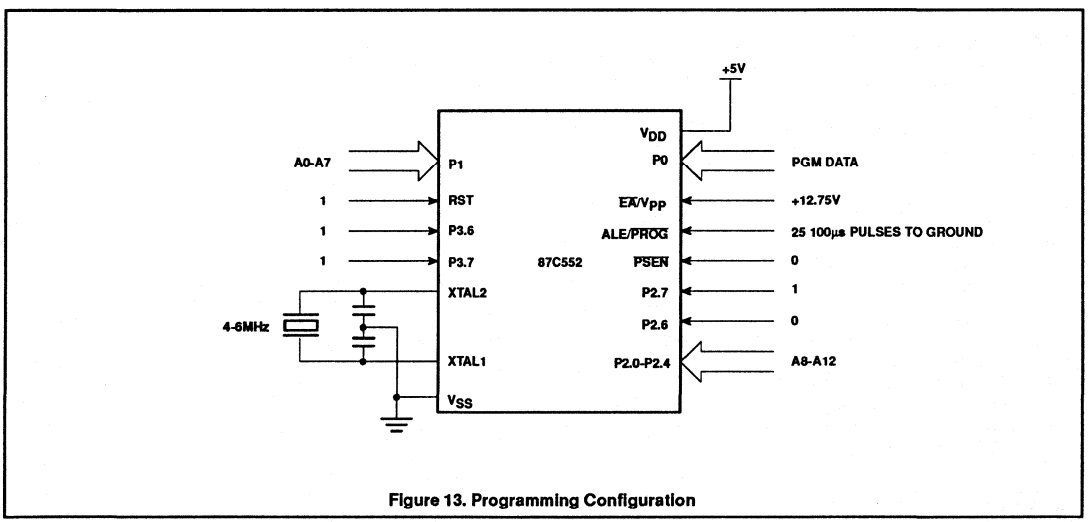


Figure 13. Programming Configuration

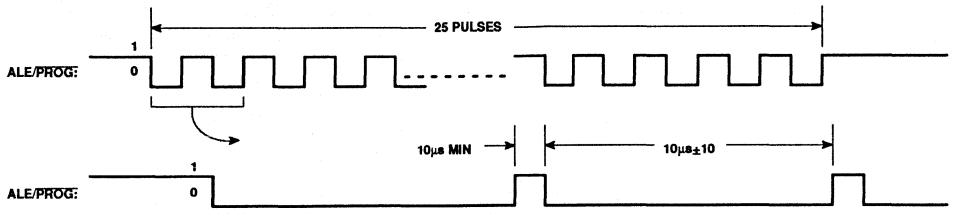


Figure 14. PROG Waveform

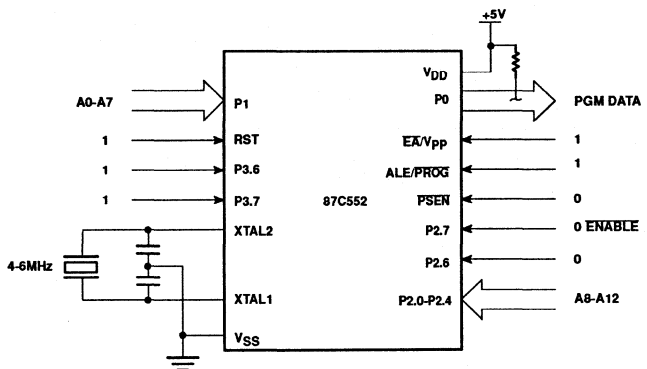


Figure 15. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_A = 21°C to +27°C, V_{DD} = 5V±10%, V_{SS} = 0V (See Figure 16)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to PROG low	$48t_{CLCL}$		
t_{GHAX}	Address hold after PROG	$48t_{CLCL}$		
t_{DVGL}	Data setup to PROG low	$48t_{CLCL}$		
t_{GHDX}	Data hold after PROG	$48t_{CLCL}$		
t_{EHSH}	P2.7 (ENABLE) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to PROG low	10		µs
t_{GHSL}	V_{PP} hold after PROG	10		µs
t_{GLGH}	PROG width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	ENABLE low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after ENABLE	0	$48t_{CLCL}$	
t_{GHGL}	PROG high to PROG low	10		µs

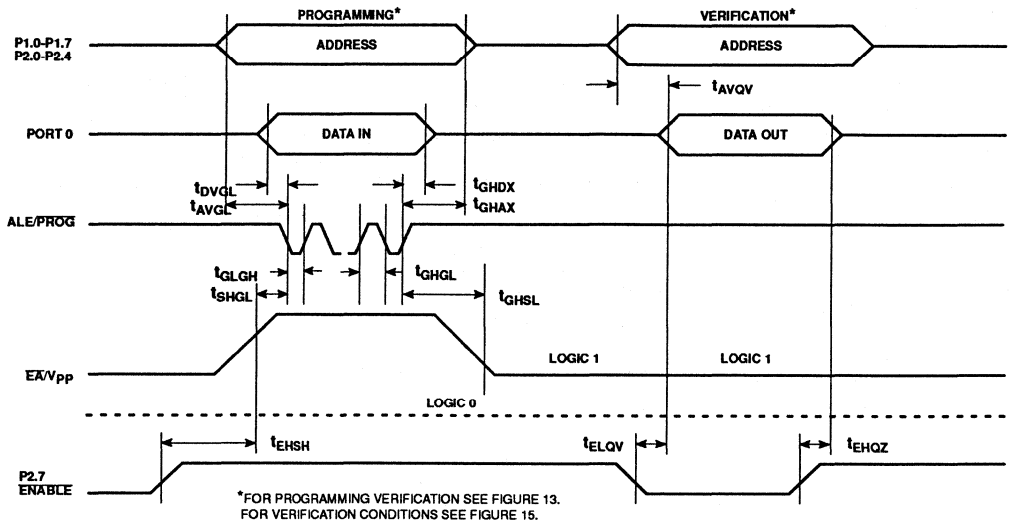


Figure 16. EPROM Programming and Verification

Philips Components

Date of Issue	September 6, 1990
Status	Product Specification
Application Specific Product	

80C562/83C562

Single-chip 8-bit microcontroller with 8-bit A/D, capture/compare timer, high-speed outputs, PWM

DESCRIPTION

The 80C562/83C562 (hereafter generically referred to as 8XC562) Single-Chip 8-Bit Microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. The 83C562/83C562 has the same instruction set as the 80C51.

The 8XC562 contains a non-volatile 256 x 8 read-only program memory, a volatile 256 X 8 read/write data memory (83C562) (the 80C562 is ROMless), a volatile 256 x 8 read/write data memory, six 8-bit I/O ports, two 16-bit timer/event counters (identical to the timers of the 80C51), an additional 16-bit timer coupled to capture and compare latches, a 15-source, two-priority-level, nested interrupt structure, an 8-input ADC, two pulse width modulated outputs, standard 80C51 UART, a "watchdog" timer and on-chip oscillator and timing circuits. For systems that require extra capability, the 83C562 can be expanded using standard TTL compatible memories and logic.

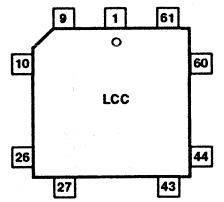
The device also functions as an arithmetic processor having facilities for both binary and BCD arithmetic plus bit-handling capabilities. The instruction set consists of over 100 instructions: 49 one-byte, 45 two-byte and 17 three-byte. With a 12MHz crystal, 58% of the instructions are executed in 1μs and 40% in 2μs. Multiply and divide instructions require 4μs.

For emulation purposes, the 87C552 is recommended.

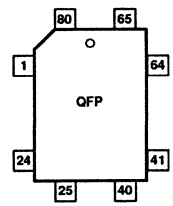
FEATURES

- 80C51 instruction set
- 8k x 8 ROM expandable externally to 64k bytes
- 256 x 8 RAM, expandable externally to 64k bytes
- Two standard 16-bit timer/counters
- An additional 16-bit timer/counter coupled to four capture registers and three compare registers
- Capable of producing 8 synchronized, timed outputs
- A 8-bit ADC with 8 multiplexed analog inputs
- Two 8-bit resolution, pulse width modulated outputs
- Five 8-bit I/O ports plus one 8-bit input port shared with analog inputs
- Full-duplex UART compatible with the standard 80C51
- On-chip watchdog timer
- Three temperature ranges
  - 0 to +70°C
  - –40 to +85°C
  - –40 to +125°C

PIN CONFIGURATION



Pin	Function	Pin	Function
1	P5.0/ADC0	35	XTAL1
2	V_{DD}	36	V_{SS}
3	STADC	37	V_{SS}
4	$\overline{PWM0}$	38	NC
5	$\overline{PWM1}$	39	P2.0/A08
6	$\overline{EW}$	40	P2.1/A09
7	P4.0/CMSR0	41	P2.2/A10
8	P4.1/CMRS1	42	P2.3/A11
9	P4.2CMSR2	43	P2.4/A12
10	P4.3/CMSR3	44	P2.5/A13
11	P4.4/CMSR4	45	P2.6/A14
12	P4.5/CMSR5	46	P2.7/A15
13	P4.6/CMT0	47	$\overline{PSEN}$
14	P4.7/CMT1	48	ALE
15	RST	49	$\overline{EA}$
16	P1.0/CT01	50	P0.7/AD7
17	P1.1/CT1I	51	P0.6/AD6
18	P1.2/CT2I	52	P0.5/AD5
19	P1.3/CT3I	53	P0.4/AD4
20	P1.4/T2	54	P0.3/AD3
21	P1.5/RT2	55	P0.2/AD2
22	P1.6/SCL	56	P0.1/AD1
23	P1.7/SDA	57	P0.0/AD0
24	P3.0/RxD	58	AVref–
25	P3.1TxD	59	AVref+
26	P3.2/$\overline{INT0}$	60	AV_{SS}
27	P3.3/$\overline{INT1}$	61	AV_{DD}
28	P3.4/T0	62	P5.7/ADC7
29	P3.5/T1	63	P5.6/ADC6
30	P3.6/$\overline{WR}$	64	P5.5/ADC5
31	P3.7/$\overline{RD}$	65	P5.4/ADC4
32	NC	66	P5.3/ADC3
33	NC	67	P5.2/ADC2
34	XTAL2	68	P5.1/ADC1



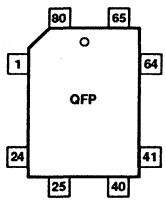
SEE NEXT PAGE FOR QFP PIN FUNCTIONS.

Single-chip 8-bit microcontroller 80C562/83C562

PART NUMBER SELECTION

ROMless	ROM	TEMPERATURE °C AND PACKAGE	FREQUENCY
PCB80C562WP	PCB83C562WP	0 to +70, PLCC	1.2 to 16MHz
PCB80C562H	PCB83C562H	0 to +70, PQFP	1.2 to 16MHz
PCF80C562WP	PCF83C562WP	–40 to +85, PLCC	1.2 to 12MHz
PCF80C562H	PCF83C562H	–40 to +85, PQFP	1.2 to 12MHz
PCA80C562WP	PCA83C562WP	–40 to +125, PLCC	1.2 to 12MHz
PCA80C562H	PCA83C562H	–40 to +125, PQFP	1.2 to 12MHz

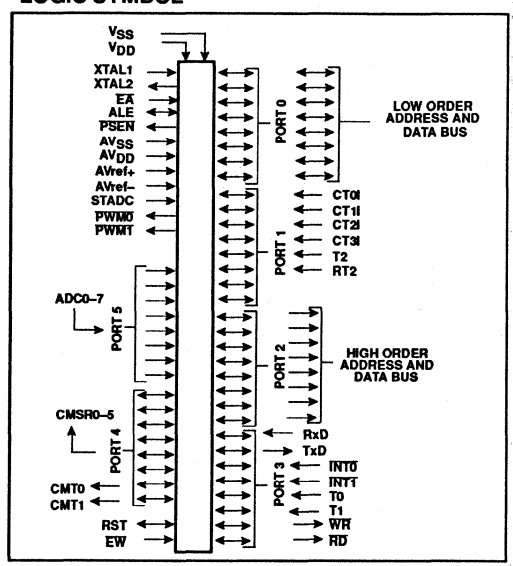
QFP PIN FUNCTIONS



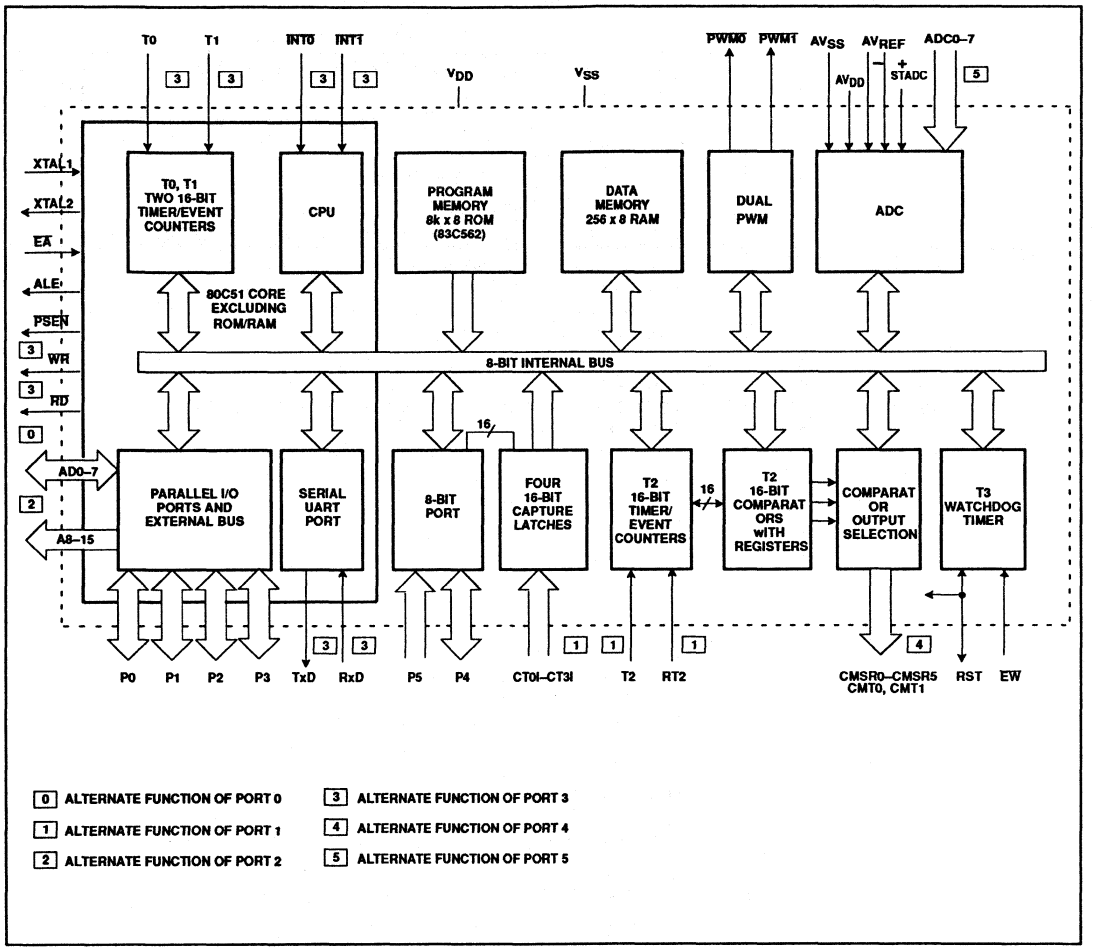
Pin	Function	Pin	Function
1	P4.1/CMSR1	41	P2.3/A11
2	P4.2/CMSR2	42	P2.4/A12
3	NC	43	NC
4	P4.3/CMSR3	44	NC
5	P4.4/CMSR4	45	P2.5/A13
6	P4.5/CMSR5	46	P2.6/A14
7	P4.6/CMT0	47	P2.7/A15
8	P4.7/CMT1	48	$\overline{PSEN}$
9	RST	49	ALE
10	P1.0/CT0I	50	$\overline{EA}$
11	P1.1/CT1I	51	P0.7/AD7
12	P1.2/CT2I	52	P0.6/AD6
13	P1.3/CT3I	53	P0.5/AD5
14	P1.4/T2	54	P0.4/AD4
15	P1.5/RT2	55	P0.3/AD3
16	P1.6	56	P0.2/AD2
17	P1.7	57	P0.1/AD1
18	P3.0/RXD	58	P0.0/AD0
19	P3.1/TXD	59	AVref–
20	P3.2/$\overline{INT0}$	60	AVref+
21	NC	61	AV_{SS}
22	NC	62	NC
23	P3.3/$\overline{INT1}$	63	AV_{DD}
24	P3.4/T0	64	P5.7/ADC7
25	P3.5/T1	65	P5.6/ADC6
26	P3.6/$\overline{WR}$	66	P5.5/ADC5
27	P3.7/$\overline{RD}$	67	P5.4/ADC4
28	NC	68	P5.3/ADC3
29	NC	69	P5.2/ADC2
30	NC	70	P5.1/ADC1
31	XTAL2	71	P5.0/ADC0
32	XTAL1	72	V_{DD}
33	IC	73	IC
34	V_{SS}	74	STADC
35	V_{SS}	75	$\overline{PWM0}$
36	V_{SS}	76	$\overline{PWMT}$
37	NC	77	$\overline{EW}$
38	P2.0/A08	78	NC
39	P2.1/A09	79	NC
40	P2.2/A10	80	P4.0/CMSR0

NC = not connected
IC = internally connected (do not use)

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. PLCC	PIN NO. QFP	TYPE	NAME AND FUNCTION
V_{DD}	2	72	I	**Digital Power Supply:** +5V power supply pin during normal operation, idle and power-down mode.
STADC	3	74	I	**Start ADC Operation:** Input starting analog to digital conversion (ADC operation can also be started by software).
$\overline{PWM0}$	4	75	O	**Pulse Width Modulation:** Output 0.
$\overline{PWM1}$	5	76	O	**Pulse Width Modulation:** Output 1.
$\overline{EW}$	6	77	I	**Enable Watchdog Timer:** Enable for T3 watchdog timer and disable power-down mode.
P0.0–P0.7	57–50	58–51	I/O	**Port 0:** Port 0 is an 8-bit open-drain bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application it uses strong internal pull-ups when emitting 1s.
P1.0–P1.7	16–23	10–17	I/O	**Port 1:** 8-bit I/O port. Alternate functions include:
	16–23	10–17	I/O	**(P1.0–P1.7):** Quasi-bidirectional port pins.
	16–19	10–13	I/O	**CT0I–CT3I (P1.0–P1.3):** Capture timer input signals for timer T2.
	20	14	I	**T2 (P1.4):** T2 event input
	21	15	I	**RT2 (P1.5):** T2 timer reset signal. Rising edge triggered.
P2.0–P2.7	39–46	38–47, 38–42, 45–47	I/O	**Port 2:** 8-bit quasi-bidirectional I/O port. Alternate function: High-order address byte for external memory (A08–A15).
P3.0–P3.7	24–31	18–20, 23–27	I/O	**Port 3:** 8-bit quasi-bidirectional I/O port. Alternate functions include:
	24	18		**RxD(P3.0):** Serial input port.
	25	19		**TxD (P3.1):** Serial output port.
	26	20		**$\overline{INT0}$ (P3.2):** External interrupt.
	27	23		**$\overline{INT1}$ (P3.3):** External interrupt.
	28	24		**T0 (P3.4):** Timer 0 external input.
	29	25		**T1 (P3.5):** Timer 1 external input.
	30	26		**$\overline{WR}$ (P3.6):** External data memory write strobe.
	31	27		**$\overline{RD}$ (P3.7):** External data memory read strobe.
P4.0–P4.7	7–14	80, 1–2, 4–8	I/O	**Port 4:** 8-bit quasi-bidirectional I/O port. Alternate functions include:
	7–12	80, 1–2 4–6	O	**CMSR0–CMSR5 (P4.0–P4.5):** Timer T2 compare and set/reset outputs on a match with timer T2.
	13, 14	7, 8	O	**CMT0, CMT1 (P4.6, P4.7):** Timer T2 compare and toggle outputs on a match with timer T2.
P5.0–P5.7	68–62, 1	71–64	I	**Port 5:** 8-bit input port. **ADC0–ADC7 (P5.0–P5.7):** Alternate function: Eight input channels to ADC.
RST	15	9	I/O	**Reset:** Input to reset the 87C552. It also provides a reset pulse as output when timer T3 overflows.
XTAL1	35	32	I	**Crystal Input 1:** Input to the inverting amplifier that forms the oscillator, and input to the internal clock generator. Receives the external clock signal when an external oscillator is used.
XTAL2	34	31	O	**Crystal Input 2:** Output of the inverting amplifier that forms the oscillator. Left open–circuit when an external clock is used.
V_{SS}	36, 37	34–36	I	**Digital ground.**
$\overline{PSEN}$	47	48	O	**Program Store Enable:** Active-low read strobe to external program memory.
ALE	48	49	O	**Address Latch Enable:** Latches the low byte of the address during accesses to external memory. It is activated every six oscillator periods. During an external data memory access, one ALE pulse is skipped. ALE can drive up to eight LS TTL inputs and handles CMOS inputs without an external pull-up.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO. PLCC	PIN NO. QFP	TYPE	NAME AND FUNCTION
$\overline{EA}$	49	50	I	**External Access:** When $\overline{EA}$ is held at TTL level high, the CPU executes out of the internal program ROM provided the program counter is less than 8192. When $\overline{EA}$ is held at TTL low level, the CPU executes out of external program memory. $\overline{EA}$ is not allowed to float.
AV_{REF-}	58	59	I	**Analog to Digital Conversion Reference Resistor:** Low-end.
AV_{REF+}	59	60	I	**Analog to Digital Conversion Reference Resistor:** High-end.
AV_{SS}	60	61	I	**Analog Ground**
AV_{DD}	61	63	I	**Analog Power Supply**

NOTE:
1. To avoid 'latch-up' effect at power-on, the voltage on any pin at any time must not be higher or lower than V_{DD} +0.5V or V_{SS} – 0.5V, respectively.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol, page 2.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To ensure a good power-on reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-on, the voltage on V_{DD} and RST must come up at the same time for a proper start-up.

IDLE MODE

In the idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. the control bits for the reduced power modes are in the special function register PCON. Table 1 shows the state of the I/O ports during low current operating modes.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3	PORT 4	PWM0/ PWM1
Idle	Internal	1	1	Data	Data	Data	Data	Data	High
Idle	External	1	1	Float	Data	Address	Data	Data	High
Power-down	Internal	0	0	Data	Data	Data	Data	Data	High
Power-down	External	0	0	Float	Data	Data	Data	Data	High

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Voltage on any other pin to V_{SS}	–0.5 to +6.5	V
Input, output DC current on any single I/O pin	5.0	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.0	W
Storage temperature range	–65 to +150	°C

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

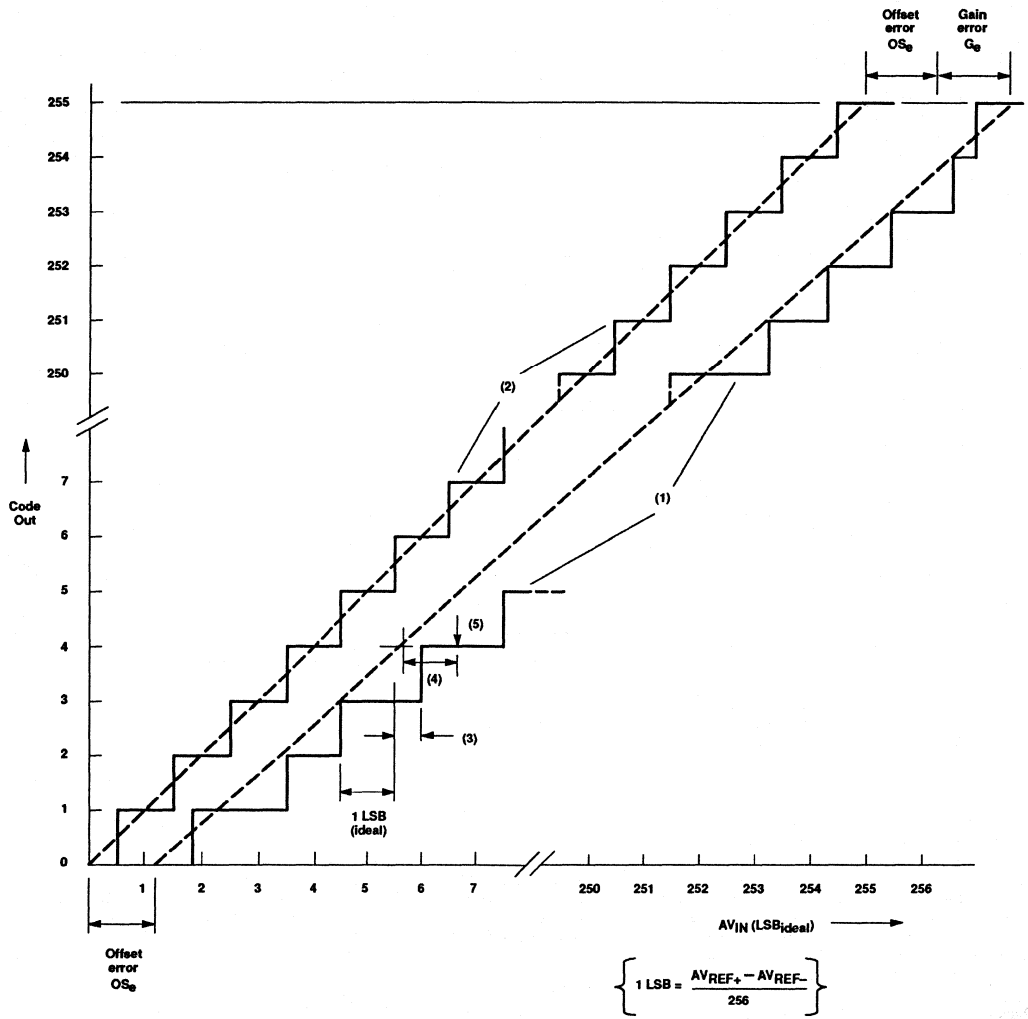
V_{SS}, AV_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{DD}	Supply voltage PCB8XC562 PCF8XC562 PCA8XC562		 4.0 4.0 4.5	 6.0 6.0 5.5	 V V V
I_{DD}	Supply current operating: PCB8XC562 PCF8XC562 PCA8XC562	See notes 1 and 2 f_{OSC} = 16MHz f_{OSC} = 12MHz f_{OSC} = 12MHz		 45 34 30	 mA mA mA
I_{ID}	Idle mode: PCB8XC562 PCF8XC562 PCA8XC562	See notes 1 and 3 f_{OSC} = 16MHz f_{OSC} = 12MHz f_{OSC} = 12MHz		 10 8 7	 mA mA mA
I_{PD}	Power-down current: PCB8XC562 PCF8XC562 PCA8XC562	See notes 1 and 4; 2V < VPD < VDD max		 50 50 100	 µA µA µA
Inputs					
V_{IL}	Input low voltage, except $\overline{EA}$		–0.5	$0.2V_{DD}$–0.1	V
V_{IL1}	Input low voltage to $\overline{EA}$		–0.5	$0.2V_{DD}$–0.3	V
V_{IH}	Input high voltage, except XTAL1, RST		$0.2V_{DD}$+0.9	V_{DD}+0.5	V
V_{IH1}	Input high voltage, XTAL1, RST		$0.7V_{DD}$	V_{DD}+0.5	V
$-I_{IL}$	Logical 0 input current, ports 1, 2, 3, 4	V_{IN} = 0.45V		–50	µA
$-I_{TL}$	Logical 1-to-0 transition current, ports 1, 2, 3, 4	See note 5		–650	µA
$\pm I_{IL1}$	Input leakage current, port 0, $\overline{EA}$, STADC, $\overline{EW}$	0.45V < V_I < V_{DD}		10	µA
Outputs					
V_{OL}	Output low voltage, ports 1, 2, 3, 4	I_{OL} = 1.6mA[6]		0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$, $\overline{PWM0}$, $\overline{PWM1}$	I_{OL} = 3.2mA[6]		0.45	V
V_{OH}	Output high voltage, ports 1, 2, 3, 4	V_{DD} + 5V±10% $-I_{OH}$ = 60µA $-I_{OH}$ = 25µA $-I_{OH}$ = 10µA	 2.4 $0.75V_{DD}$ $0.9V_{DD}$		V V V V
V_{OH1}	Output high voltage (port 0 in external bus mode, ALE, $\overline{PSEN}$, $\overline{PWM0}$, $\overline{PWM1}$)[7]	V_{DD} + 5V±10% $-I_{OH}$ = 400µA $-I_{OH}$ = 150µA $-I_{OH}$ = 40µA	 2.4 $0.75V_{DD}$ $0.9V_{DD}$		V V V V
V_{OH2}	High level output voltage (RST)	$-I_{OH}$ = 400µA $-I_{OH}$ = 120µA	2.4 $0.8V_{DD}$		V V
R_{RST}	Internal reset pull-down resistor		50	150	kohm
C_{IO}	Pin capacitance	Test freq = 1MHz, T_A = 25°C		10	pF
Analog Inputs					
AV_{DD}	Analog supply voltage: PCB8XC562 PCF8XC562 PCA8XC562	 AV_{DD} = V_{DD}±0.2V AV_{DD} = V_{DD}±0.2V AV_{DD} = V_{DD}±0.2V	 4.0 4.0 4.5	 6.0 6.0 5.5	 V V V
AI_{DD}	Analog supply current: operating:	Port 5 = 0 to AV_{DD}		1.2	mA

DC ELECTRICAL CHARACTERISTICS (Continued)

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
Analog Inputs (Continued)					
AI_{ID}	Idle mode: PCB8XC562 PCF8XC562 PCA8XC562			 50 50 100	 µA µA µA
AI_{PD}	Power-down mode: PCB8XC562 PCF8XC562 PCA8XC562	2V < AV_{PD} < AV_{DD} max		 50 50 100	 µA µA µA
AV_{IN}	Analog input voltage		AV_{SS}–0.2	AV_{DD}+0.2	V
AV_{REF}	Reference voltage: AV_{REF-} AV_{REF+}		AV_{SS}–0.2	 AV_{DD}+0.2	 V V
R_{REF}	Resistance between AV_{REF+} and AV_{REF-}		5	25	kohms
C_{IA}	Analog input capacitance			15	pF
t_{ADS}	Sampling time			$6t_{CY}$	µs
t_{ADC}	Conversion time (including sampling time)			$24t_{CY}$	µs
DL_e	Differential non-linearity[8, 9, 10]			±1	LSB
IL_e	Integral non-linearity[8, 11]			±1	LSB
OS_e	Offset error[8, 12]			±1	LSB
G_e	Gain error[8, 13]			0.4	%
M_{CTC}	Channel to channel matching			±1	LSB
C_t	Crosstalk between inputs of port 5[14]	0–100kHz		–60	dB

NOTES:
1. See Figures 8 through 12 for I_{DD} test conditions.
2. The operating supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = $\overline{EW}$ = V_{DD}; STADC = V_{SS}.
3. The idle mode supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = Port 0 = $\overline{EW}$ = V_{DD}; RST = STADC = V_{SS}.
4. The power-down current is measured with all output pins disconnected; XTAL2 not connected; $\overline{EA}$ = Port 0 = $\overline{EW}$ = V_{DD}; RST = STADC = XTAL1 = V_{SS}.
5. Pins of ports 1, 2, 3, and 4 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
6. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input. I_{OL} can exceed these conditions provided that no single output sinks more than 5mA and no more than two outputs exceed the test conditions.
7. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{DD}$ specification when the address bits are stabilizing.
8. Conditions: AV_{REF-} = 0V; AV_{DD} = 5.0V, AV_{REF+} = 5.12V. ADC is monotonic with no missing codes.
9. The differential non-linearity (DL_e) is the difference between the actual step width and the ideal step width. (See Figure 1.)
10. The ADC is monotonic; there are no missing codes.
11. The integral non-linearity (IL_e) is the peak difference between the center of the steps of the actual and the ideal transfer curve after appropriate adjustment of gain and offset error. (See Figure 1.)
12. The offset error (OS_e) is the absolute difference between the straight line which fits the actual transfer curve (after removing gain error), and a straight line which fits the ideal transfer curve. (See Figure 1.)
13. The gain error (G_e) is the relative difference in percent between the straight line fitting the actual transfer curve (after removing offset error), and the straight line which fits the ideal transfer curve. Gain error is constant at every point on the transfer curve. (See Figure 1.)
14. This should be considered when both analog and digital signals are simultaneously input to port 5.



(1) Example of an actual transfer curve.
(2) The ideal transfer curve.
(3) Differential non-linearity (DL_e).
(4) Integral non-linearity (IL_e).
(5) Center of a step of the actual transfer curve.

Figure 1. ADC Conversion Characteristic

AC ELECTRICAL CHARACTERISTICS[1, 2]

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	2	Oscillator frequency			1.2	16	MHz
t_{LHLL}	2	ALE pulse width	127		$2t_{CLCL}-40$		ns
t_{AVLL}	2	Address valid to ALE low	28		$t_{CLCL}-55$		ns
t_{LLAX}	2	Address hold after ALE low	48		$t_{CLCL}-35$		ns
t_{LLIV}	2	ALE low to valid instruction in		234		$4t_{CLCL}-100$	ns
t_{LLPL}	2	ALE low to $\overline{PSEN}$ low	43		$t_{CLCL}-40$		ns
t_{PLPH}	2	$\overline{PSEN}$ pulse width	205		$3t_{CLCL}-45$		ns
t_{PLIV}	2	$\overline{PSEN}$ low to valid instruction in		145		$3t_{CLCL}-105$	ns
t_{PXIX}	2	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	2	Input instruction float after $\overline{PSEN}$		59		$t_{CLCL}-25$	ns
t_{AVIV}	2	Address to valid instruction in		312		$5t_{CLCL}-105$	ns
t_{PLAZ}	2	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{AVLL}	3, 4	Address valid to ALE low	43		$t_{CLCL}-35$		ns
t_{RLRH}	3	$\overline{RD}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{WLWH}	4	$\overline{WR}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{RLDV}	3	$\overline{RD}$ low to valid data in		252		$5t_{CLCL}-165$	ns
t_{RHDX}	3	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	3	Data float after $\overline{RD}$		97		$2t_{CLCL}-70$	ns
t_{LLDV}	3	ALE low to valid data in		517		$8t_{CLCL}-150$	ns
t_{AVDV}	3	Address to valid data in		585		$9t_{CLCL}-165$	ns
t_{LLWL}	3, 4	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	3, 4	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}-130$		ns
t_{QVWX}	4	Data valid to $\overline{WR}$ transition	23		$t_{CLCL}-60$		ns
t_{DW}	4	Data before $\overline{WR}$	433		$7t_{CLCL}-150$		ns
t_{WHQX}	4	Data hold after $\overline{WR}$	33		$t_{CLCL}-50$		ns
t_{RLAZ}	3	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	3, 4	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock							
t_{CHCX}	5	High time[3]	20		20		ns
t_{CLCX}	5	Low time[3]	20		20		ns
t_{CLCH}	5	Rise time[3]		20		20	ns
t_{CHCL}	5	Fall time[3]		20		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. These values are characterized but not 100% production tested.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

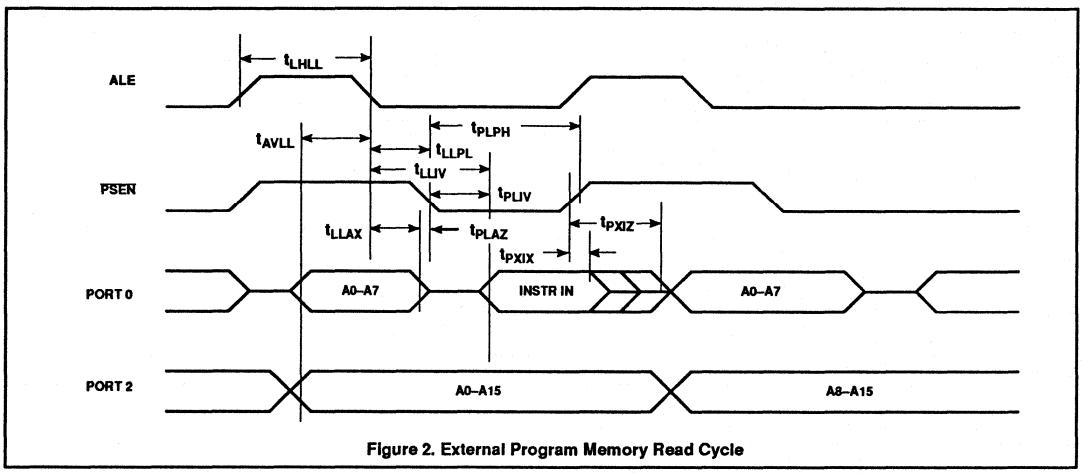


Figure 2. External Program Memory Read Cycle

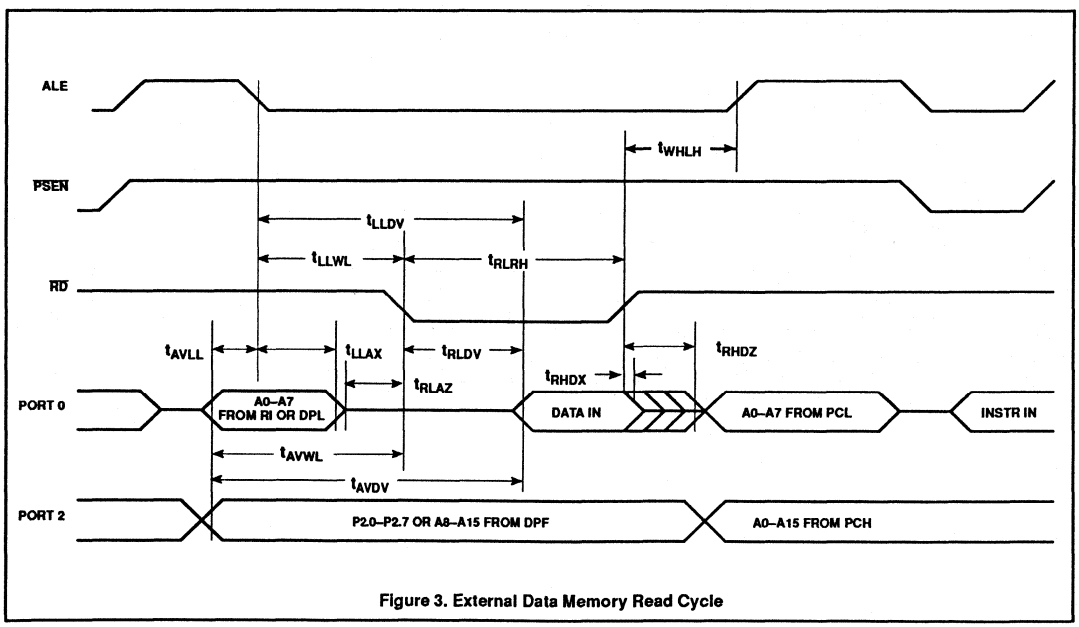


Figure 3. External Data Memory Read Cycle

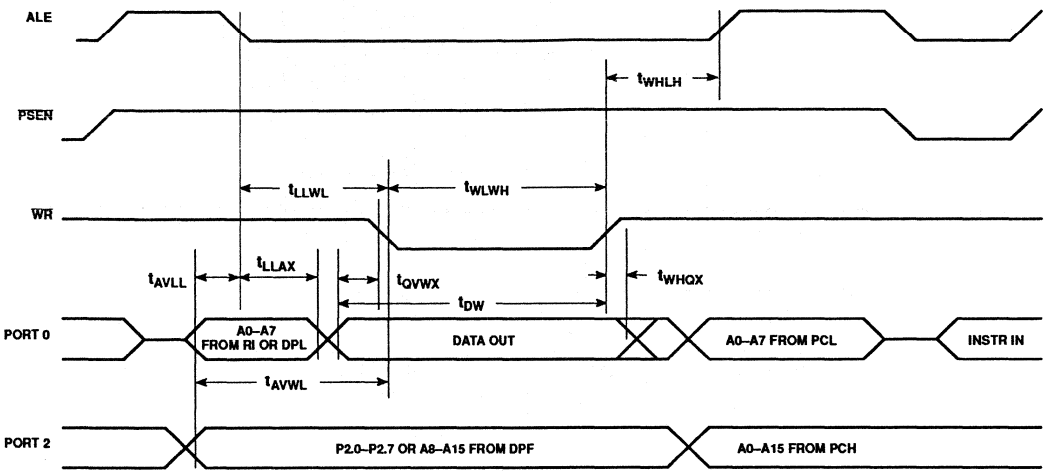


Figure 4. External Data Memory Write Cycle

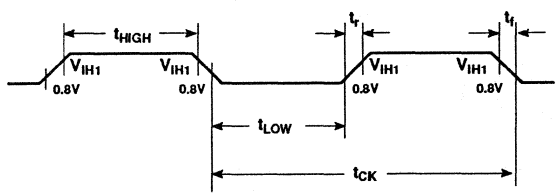
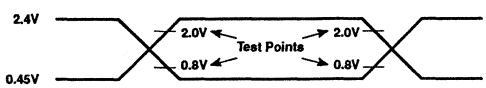
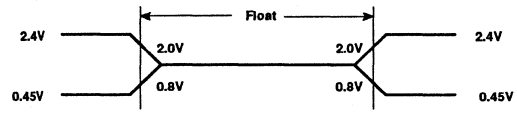


Figure 5. External Clock Drive XTAL1



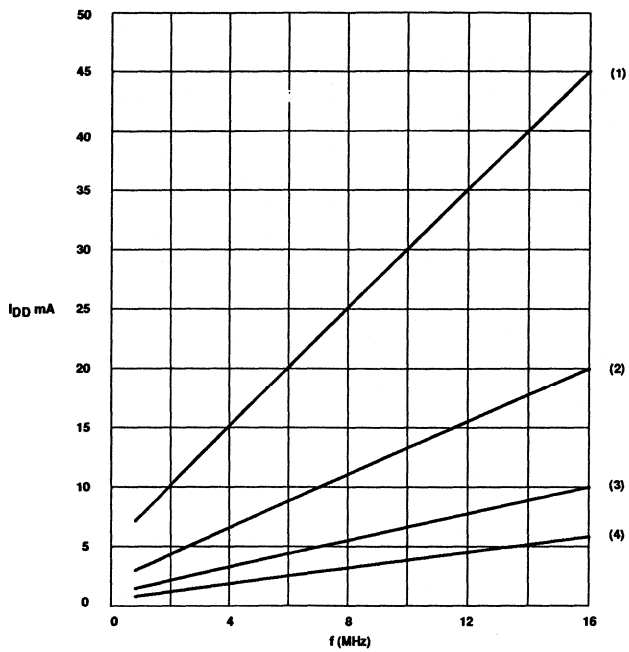
NOTE:
AC inputs during testing are driven at 2.4V for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at 2.0V for a logic '1' and 0.8V for a logic '0'.

Figure 6. AC Testing Input/Output



NOTE:
The float state is defined as the point at which a port 0 pin sinks 3.2mA or sources 400μA at the voltage test levels.

Figure 7. AC Testing Input, Float Waveform



(1) Maximum operating mode; V_{DD} = 6V.
(2) Maximum operating mode; V_{DD} = 4V.
(3) Maximum idle mode; V_{DD} = 6V.
(4) Maximum idle mode; V_{DD} = 4V.

NOTE:
These values are valid only within the frequency specifications of the device under test.

Figure 8. Supply Current (I_{DD}) as a Function of Frequency at XTAL1 (f_{OSC})

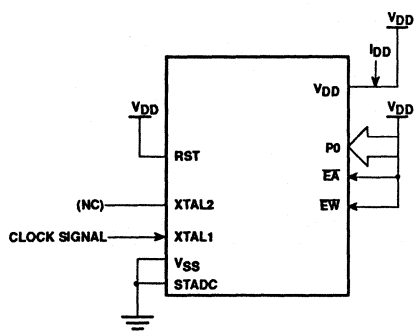


Figure 9. I_{DD} Test Condition, Active Mode
All other pins are disconnected

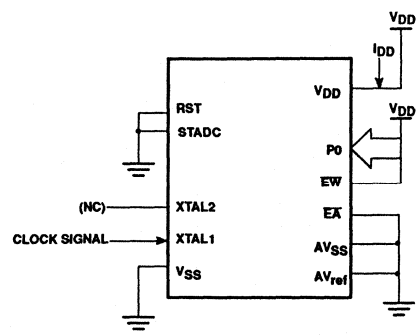


Figure 10. I_{DD} Test Condition, Idle Mode
All other pins are disconnected

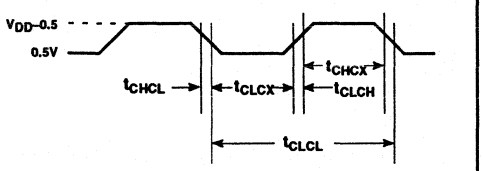


Figure 11. Clock Signal Waveform for I_{DD} Tests in Active and Idle Modes $t_{CLCH} = t_{CHCL} = 10ns$

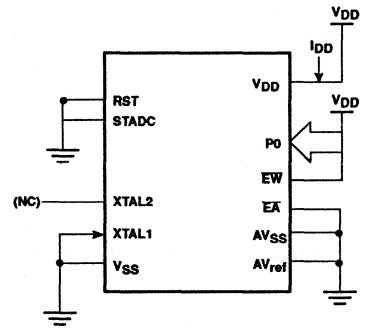


Figure 12. I_{DD} Test Condition, Power Down Mode All other pins are disconnected. V_{DD} = 2V to 5.5V

8XC652/654 OVERVIEW

The 8XC652 and 8XC654 (hereafter referred to collectively as 8XC652/4) are derivatives of the 80C51 8-bit CMOS microcontroller. The 8XC652/4 contains all of the features of the 80C51 (that is, the standard counter/timers T0 and T1, the standard serial I/O (UART), and four 8-bit I/O ports). In addition, the 8XC652/4 has the following:

- 8k bytes of ROM (8XC652)
- 16k bytes of ROM (8XC654)
- 256 bytes of RAM
- I^2C bus serial I/O

The only difference between the 8XC652 and the 8XC654 is that the 8XC654 has 16k bytes of program memory while the 8XC652 has 8k bytes. All other features of these parts are identical.

The 8XC652/4 is pin-for-pin compatible and fully code compatible with the 80C51. There are some differences in the P1.6 and P1.7 pin functions that are described in detail later in this section. All of the 80C51 functions are present, including the external 64k program and data memory expansion, Boolean processing, and two reduced power modes.

Differences from the 80C51

The data and program memory are organized similar to the 80C51. The 8XC652/4 program memory differs in that it has 8k/16k bytes of on-chip ROM. When $\overline{EA}$ is high the 8XC652/4 fetches instructions from the internal ROM unless the address exceeds 1FFFH/2FFFH. Locations 2000H/3000H to FFFFH are fetched from external program memory. When $\overline{EA}$ is held low, all instruction fetches are from external memory.

The organization of the data memory is similar to the 80C51 except that the 8XC652/4 has an additio: al 128 bytes of RAM overlapped with the special function register space. This additional RAM is addressed using indirect addressing only and is available as stack space. (This memory addition is the same as in the 80C52 and 83C552. See Figure 65 for a memory map.)

Special Function Registers

The 8XC652/4 special function register space is the same as that on the 80C51 except that it contains four additional SFRs. The added registers are: S1CON, S1STA, S1DAT,, and S1ADR. In addition to these, the standard UART special function registers SCON and SBUF have been renamed S0CON and S0BUF for clarity.

Since the standard 80C51 on-chip functions are the same on the 8XC652/4, the SFR locations, bit locations, and operation are unchanged. The only exception is in the interrupt enable and interrupt priority SFRs. These have been changed to include the interrupt from the I^2C serial port. Table 25 shows the special function registers, their direct address, the bit addresses, and the value in the register after a reset.

I^2C Serial Communication—SIO1

The I^2C serial port is identical to the I^2C serial port on the 8XC552. The operation of this subsystem is described in detail in the 8XC552 section of this manual.

Note that in both the 8XC652/4 and the 8XC552 the I^2C pins are alternate functions to port pins P1.6 and P1.7. Because of this, P1.6 and P1.7 on these parts do not have a pull-up structure as found on the 80C51. Therefore P1.6 and P1.7 have open drain outputs on the 8XC652/4.

Idle and Power-Down Operation

Idle mode operation permits the interrupt, serial ports, and timer blocks to continue to function while the CPU is halted. The following functions remain active during idle mode. These functions may generate an interrupt or reset and thus end the idle mode:

- Timer 1 overflow
- I^2C serial I/O interrupt
- UART serial I/O interrupt
- Timer 0, Timer 1
- SIO0, SIO1
- External interrupt

In idle mode, port pins P1.6 and P1.7 function as SCL and SDA, respectively, if the I^2C serial port is enabled. The power-down operation freezes the oscillator. The power-down mode can only be activated by setting the PD bit in the PCON register. The power-down mode in the 8XC652/4 operates exactly the same as in the 80C51.

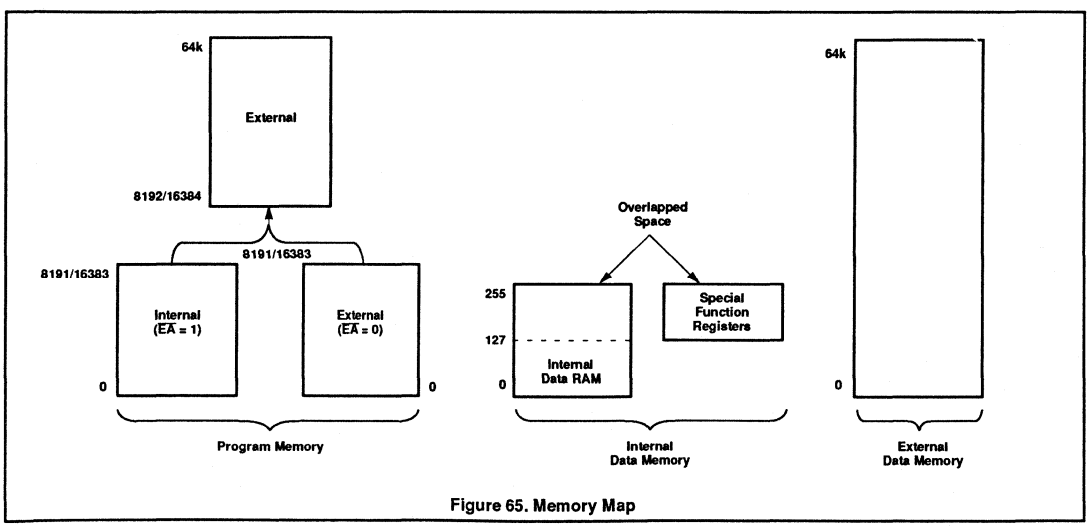


Figure 65. Memory Map

Table 25. 8XC652/654 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR:	Data pointer (2 bytes)										
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*	Interrupt enable	A8H	EA		ES1	ES0	ET1	EX1	ET0	ET1	0x000000B
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*	Interrupt priority	B8H	–		PS1	PS0	PT1	PX1	PT0	PX0	xx000000B
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	FFH
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	SDA	CSCL							FFH
			A7	A6	A5	A4	A3	A2	A1	A0	
P2*	Port 2	A0H	A15	A14	A13	A12	A11	A10	A9	A8	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	RD	WR	T1	T0	INT1	INT0	TXD	RXD	FFH
PCON	Power control	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxx0000B
			9F	9E	9D	9C	9B	9A	99	98	
S0CON*#	Serial 0 port control	98H	SM0	SM1	SM2	REN	TBB	TRB	TI	RI	00H
S0BUF#	Serial 0 data buffer	99H									xxxxxxxxB
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	F1	P	00H
S1DAT#	Serial 1 data	DAH									00H
SP	Stack pointer	81H									07H
S1ADR#	Serial 1 address	DBH	SLAVE ADDRESS							GC	00H
S1STA#	Serial 1 status	D9H	SC4	SC3	SC2	SC1	SC0	0	0	0	F8H
			DF	DE	DD	DC	DB	DA	D9	D8	
S1CON*#	Serial 1 control	D8H	–	ENS1	STA	STO	SI	AA	CR1	CR0	x0000000B
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
TH1	Timer high 1	8DH									00H
TH0	Timer high 0	8CH									00H
TL1	Timer low 1	8BH									00H
TH0	Timer low 0	8AH									00H
TMOD	Timer mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H

*SFRs are bit addressable.
#SFRs are modified from or added to the 80C51 SFRs.

Interrupt System

The interrupt system is the same as in the 80C51 except that the 8XC652/4 acknowledges interrupt requests from six sources as follows:

- $\overline{INT0}$ external interrupt 0
- $\overline{INT1}$ external interrupt 1
- Timer 0 overflow
- Timer 1 overflow
- I^2C serial I/O interrupt
- UART serial interrupt

See Figure 66 for a function diagram of the 8XC652/4 interrupt structure. Each interrupt vectors to a separate location in program memory for its service program. Each source can be individually enabled or disabled by a corresponding bit in the IE register; moreover, each interrupt may be programmed to a high or low priority level using a corresponding bit in the IP register. Also, all enabled sources can be globally disabled or enabled.

Both external interrupts can be programmed to be level-activated or transition-activated; an active LOW level allows "Wire-ORing" or several input sources to the input pin.

Each interrupt source can be set for either high priority or low priority. If two separate interrupts are requested simultaneously, the processor will branch to the vector associated with the interrupt that has the higher priority. If there are simultaneous requests from sources that have the same priority, then the interrupts will be serviced in the following order:

1. $\overline{INT0}$ external interrupt 0
2. I^2C serial I/O interrupt
3. Timer 0 overflow
4. $\overline{INT1}$ external interrupt 1
5. Timer 1 overflow
6. UART serial I/O interrupt

A low priority interrupt routine can be interrupted by an interrupt having a higher priority. A high priority interrupt cannot be interrupted. All of the features of the 8XC652/4 that have not been discussed in this section are the same as those on the 80C51.

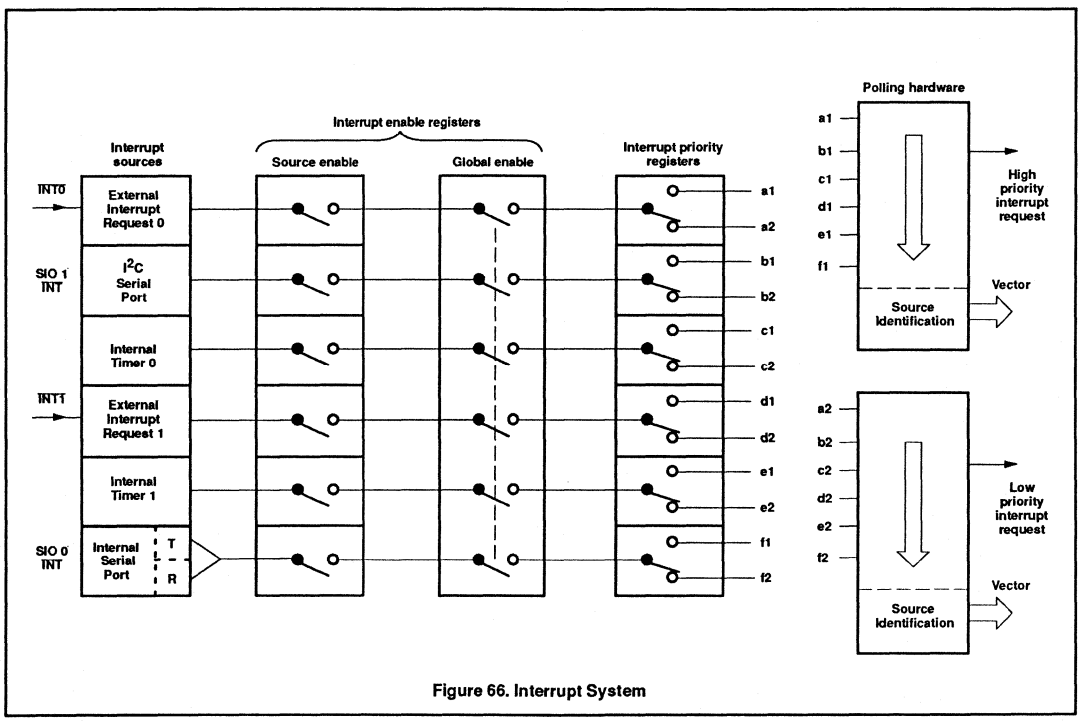
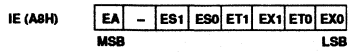


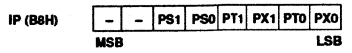
Figure 66. Interrupt System

Interrupt Enable Register



Bit	Symbol	Function
IE.7	EA	General enable/disable control 0 = No interrupt enabled 1 = Any individually enabled interrupt will be accepted
IE.6	–	Unused
IE.5	ES1	Enable SIO1 (I^2C) interrupt
IE.4	ES0	Enable SIO0 (UART) interrupt
IE.3	ET1	Enable timer 1 interrupt
IE.2	EX1	Enable external 1 interrupt
IE.1	ET0	Enable timer 0 interrupt
IE.0	EX0	Enable external 0 interrupt 0 = interrupt disabled 1 = interrupt enabled

Interrupt Priority Register



Bit	Symbol	Function
IP.7	–	Unused
IP.6	–	Unused
IP.5	PS1	SIO1 (I^2C) interrupt priority level
IP.4	PS0	SIO0 (UART) interrupt priority level
IP.3	PT1	Timer 1 interrupt priority level
IP.2	PX1	Enable interrupt 1 priority level
IP.1	PT0	Timer 0 interrupt priority level
IP.0	PX0	External interrupt 0 priority level 0 = Low priority 1 = High priority

The following vectors indicate the ROM location where the appropriate interrupt service routine starts.

Source	Vector
External 0 (X0)	0003H
Timer 0 overflow (T0)	000BH
External 1 (X1)	0013H
Timer 1 overflow (T1)	001BH
Serial I/O 0 (UART) (S0)	0023H
Serial I/O 1 (I^2C) (S1)	002BH

Philips Components

Date of Issue	September 6, 1990
Status	Product Specification
Application Specific Products	

80C652/83C652/87C652

CMOS single-chip 8-bit microcontroller

DESCRIPTION

The 80C652/83C652/87C652 Single-Chip 8-Bit Microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. The 80C652/83C652/87C652 has the same instruction set as the 80C51. Three versions of the derivative exist:

83C652 – 8k bytes mask programmable ROM

80C652 – ROMless version

87C652 – EPROM version

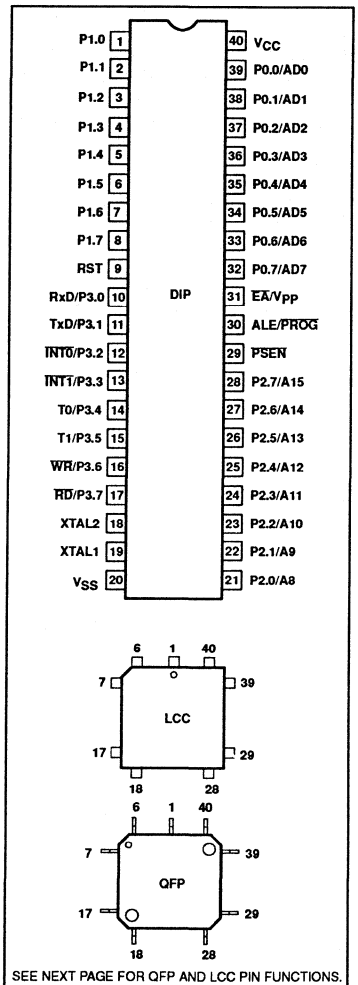
This device provides architectural enhancements that make it applicable in a variety of applications for general control systems. The 8XC652 contains a non-volatile 8k X 8 read-only program memory (83C652) EPROM (87C652), a volatile 256 X 8 read/write data memory, four 8-bit I/O ports, two 16-bit timer/event counters (identical to the timers of the 80C51), a multi-source, two-priority-level, nested interrupt structure, an I^2C interface, UART and on-chip oscillator and timing circuits. For systems that require extra capability, the 8XC652 can be expanded using standard TTL compatible memories and logic.

The device also functions as an arithmetic processor having facilities for both binary and BCD arithmetic plus bit-handling capabilities. The instruction set consists of over 100 instructions: 49 one-byte, 45 two-byte and 17 three-byte. With a 12MHz crystal, 58% of the instructions are executed in 1μs and 40% in 2μs. Multiply and divide instructions require 4μs.

FEATURES

- 80C51 central processing unit
- 8k x 8 ROM expandable externally to 64k bytes (87C652 EPROM is not expandable)
- 256 x 8 RAM, expandable externally to 64k bytes
- Two standard 16-bit timer/counters
- Four 8-bit I/O ports
- I^2C-bus serial I/O port with byte oriented master and slave functions
- Full-duplex UART facilities
- Power control modes
  - Idle mode
  - Power-down mode
- Five package styles
- Extended temperature ranges
- OTP package available

PIN CONFIGURATION

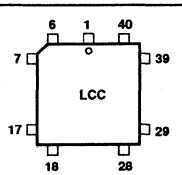


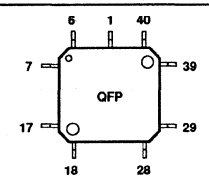
SEE NEXT PAGE FOR QFP AND LCC PIN FUNCTIONS.

PART NUMBER SELECTION

PHILIPS		PHILIPS COMPONENTS-SIGNETICS			TEMPERATURE °C AND PACKAGE	FREQ. MHz
ROMless	ROM	ROMless	ROM	EPROM		
PCB80C652P	PCB83C652P	S80C652-1N40	S83C652-1N40	S87C652-1N40	0 to +70, PDIP	12
				S87C652-1F40	0 to +70, CDIP	12
PCB80C652WP	PCB83C652WP	S80C652-1A44	S83C652-1A44	S87C652-1A44	0 to +70, PLCC	12
				S87C652-1K44	0 to +70, CLCC	12
PCB80C652H	PCB83C652H	S80C652-1B44	S83C652-1B44		0 to +70, PQFP	12
PCF80C652P	PCF83C652P	S80C652-2N40	S83C652-2N40	S87C652-2N40	–40 to +85, PDIP	12
				S87C652-2F40	–40 to +85, CDIP	12
PCF80C652WP	PCF83C652WP	S80C652-2A44	S83C652-2A44	S87C652-2A44	–40 to +85, PLCC	12
				S87C652-2K44	–40 to +85, CLCC	12
PCF80C652H	PCF83C652H	S80C652-2B44	S83C652-2B44		–40 to +85, PQFP	12
				S87C652-4N40	0 to +70, PDIP	16
				S87C652-4F40	0 to +70, CDIP	16
				S87C652-4A44	0 to +70, PLCC	16
				S87C652-4K44	0 to +70, CLCC	16
				S87C652-5N40	–40 to +85, PDIP	16
				S87C652-5F40	–40 to +85, CDIP	16
				S87C652-5A44	–40 to +85, PLCC	16
				S87C652-5K44	–40 to +85, CLCC	16
PCA80C652P	PCA83C652P	S80C652-6N40	S83C652-6N40		–40 to +125, PDIP	12
PCA80C652WP	PCA83C652WP	S80C652-6A44	S83C652-6A44		–40 to +125, PLCC	12
PCA80C652H	PCA83C652H	S80C652-6B44	S83C652-6B44		–40 to +125, PQFP	12

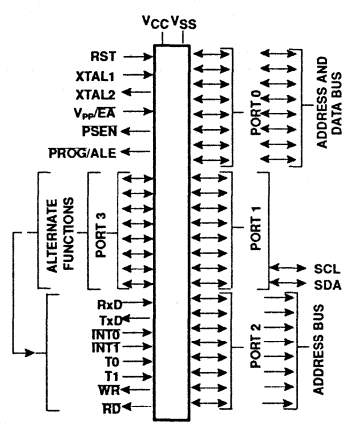
LCC AND QFP PIN FUNCTIONS



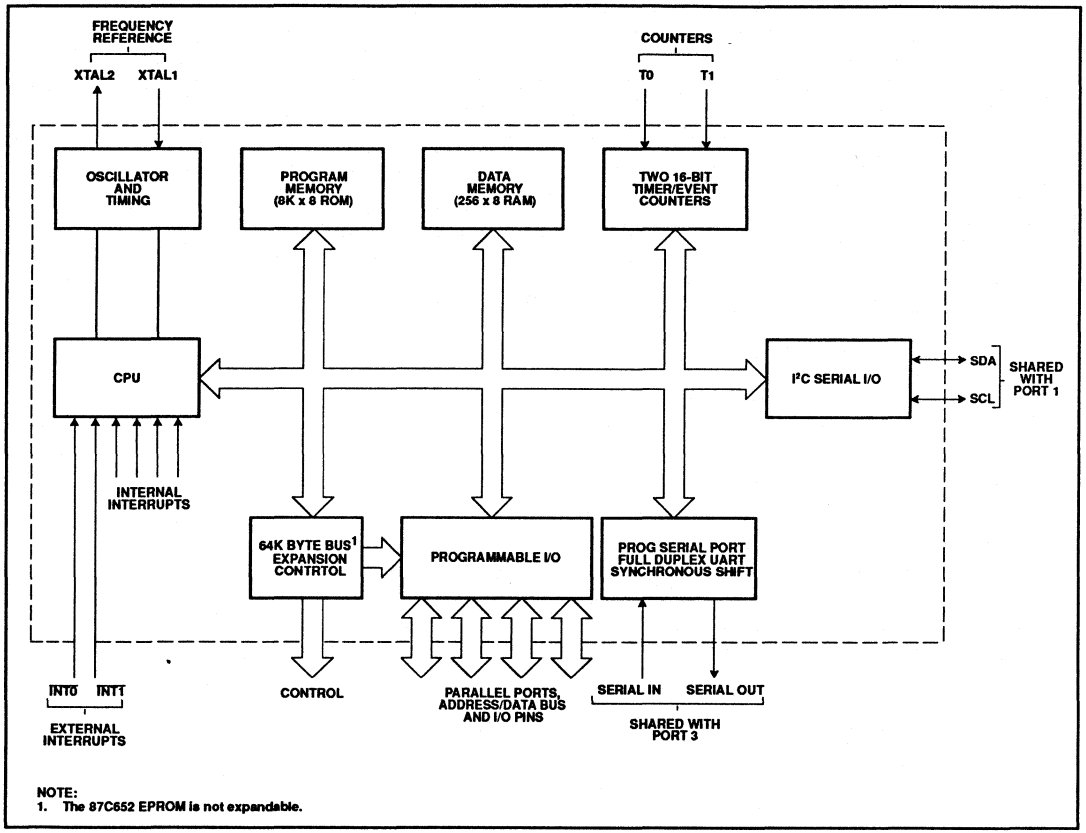


Pin	Function	Pin	Function
1	NC	23	V_{SS}
2	P1.0	24	P2.0/A8
3	P1.1	25	P2.1/A9
4	P1.2	26	P2.2/A10
5	P1.3	27	P2.3/A11
6	P1.4	28	P2.4/A12
7	P1.5	29	P2.5/A13
8	P1.6/SCL	30	P2.6/A14
9	P1.7/SDA	31	P2.7/A15
10	RST	32	$\overline{PSEN}$
11	P3.0/RxD	33	ALE/$\overline{PROG}$
12	NC	34	NC
13	P3.1/TxD	35	$\overline{EA}/V_{PP}$
14	P3.2/$\overline{INT0}$	36	P0.7/AD7
15	P3.3/$\overline{INT1}$	37	P0.6/AD6
16	P3.4/T0	38	P0.5/AD5
17	P3.5/T1	39	P0.4/AD4
18	P3.6/$\overline{WR}$	40	P0.3/AD3
19	P3.7/$\overline{RD}$	41	P0.2/AD2
20	XTAL2	42	P0.1/AD1
21	XTAL1	43	P0.0/AD0
22	V_{SS}	44	V_{CC}

LOGIC SYMBOL



BLOCK DIAGRAM



NOTE:
1. The 87C652 EPROM is not expandable.

PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC/QFP	TYPE	NAME AND FUNCTION
V_{SS}	20	22	I	**Ground:** 0V reference.
V_{CC}	40	44	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–46	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s. Port 0 also outputs the code bytes during program verification in the 87C652. External pull-ups are required during program verification.
P1.0–P1.7	1–8	2–9	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups, except P1.6 and P1.7 which are open drain. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 1 also receives the low-order address byte during program memory verification. Alternate functions include:
P1.6	7	8	I/O	**SCL:** I^2C-bus serial port clock line.
P1.7	8	9	I/O	**SDA:** I^2C-bus serial port data line.
P2.0–P2.7	21–28	24–31	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the 80C51 family, as listed below:
	10	11	I	**RxD (P3.0):** Serial input port
	11	13	O	**TxD (P3.1):** Serial output port
	12	14	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	I	**T0 (P3.4):** Timer 0 external input
	15	17	I	**T1 (P3.5):** Timer 1 external input
	16	18	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE/$\overline{PROG}$	30	33	I/O	**Address Latch Enable/Program Pulse:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.
$\overline{PSEN}$	29	32	O	**Program Store Enable:** The read strobe to external program memory. When the 87C652 is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$/V_{PP}	31	35	I	**External Access Enable/Programming Supply Voltage:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H and 1FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 1FFFH. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
XTAL1	19	21	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	O	**Crystal 2:** Output from the inverting oscillator amplifier.

NOTE:
To avoid "latch-up" effect at power-on, the voltage on any pin at any time must not be higher than V_{CC} + 0.5V or V_{SS} – 0.5V, respectively.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol, page 2.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-on reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-on, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

IDLE MODE

In the idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. the control bits for the reduced power modes are in the special function register PCON. Table 1 shows the state of the I/O ports during low current operating modes.

Table 1. External Pin Status During Idle and Power-Down Mode

MODE	PROGRAM MEMORY	ALE	PSEN	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

Serial Control Register (S1CON) – See Table 2

S1CON (D8H)	CR2	ENS1	STA	STO	SI	AA	CR1	CR0

Bits CR0, CR1 and CR2 determine the serial clock frequency that is generated in the master mode of operation.

Table 2. Serial Clock Rates

CR2[1]	CR1	CR0	BIT FREQUENCY (kHz) AT f_{osc} 6MHz	12MHz	16MHz[1]	f_{osc} DIVIDED BY
0	0	0	23	47	62.5	256[1]
0	0	1	27	54	71	224[1]
0	1	0	31.25	62.5	83.3	192[1]
0	1	1	37	75	100	160[1]
1	0	0	6.25	12.5	17	960
1	0	1	50	100	133[2]	120
1	1	0	100	200[2]	267[2]	60
1	1	1	0.25 < 31.25	0.5 < 62.5	0.67 < 83.3	96 x (256 – (reload value Timer 1)) (Reload value range: 0 – 254 in mode 2)

NOTES:
1. The CR2 control bit is only implemented on the 16MHz version.
2. These frequencies exceed the upper limit of 100kHz of the I²C-bus specification and cannot be used in an I²C-bus application.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Voltage on $\overline{EA}/V_{PP}$ to V_{SS} (87C652 only)	–0.5 to + 13	V
Voltage on any other pin to V_{SS}	–0.5 to + 6.5	V
Input, output current on any single pin	±5	mA
Input, output current on any two pins	±10	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C, or T_A = –40°C to +85°C/+125°C, V_{CC} = 5V ±10%, V_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{CC}	Supply voltage		4.5	5.5	V
I_{CC}	Supply current: Active mode Idle mode Power-down mode	See note 1 See notes 2 and 3 See notes 2 and 4 See notes 3 and 5 See notes 4 and 5 See notes 3, 6, and 7 See notes 4, 6, and 7		 24 27 5.0 6.0 50 120	 mA mA mA mA µA µA
Inputs					
V_{IL}	Input low voltage, except $\overline{EA}$, P1.6/SCL, P1.7/SDA		–0.5	$0.2V_{CC}$–0.1	V
V_{IL1}	Input low voltage to $\overline{EA}$		–0.5	$0.2V_{CC}$–0.3	V
V_{IL2}	Input low voltage to P1.6/SCL, P1.7/SDA[8]		–0.5	1.5	V
V_{IH}	Input high voltage, except XTAL1, RST, P1.6/SCL, P1.7/SDA		$0.2V_{CC}$+0.9	V_{CC}+0.5	V
V_{IH1}	Input high voltage, XTAL1, RST		$0.7V_{CC}$	V_{CC}+0.5	V
V_{IH2}	Input high voltage, P1.6/SCL, P1.7/SDA[8]		3.0	6.0	V
$-I_{IL}$	Logical 0 input current, ports 1, 2, 3, 4, except P1.6/SCL, P1.7/SDA	V_{IN} = 0.45V		50	µA
$-I_{TL}$	Logical 1-to-0 transition current, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA	See note 9		–650	µA
$\pm I_{IL1}$	Input leakage current, port 0, $\overline{EA}$	0.45V < V_I < V_{CC}		10	µA
$\pm I_{IL2}$	Input leakage current, P1.6/SCL, P1.7/SDA	0V < V_I < 6V 0V < V_{CC} < 5.5V		10	µA
Outputs					
V_{OL}	Output low voltage, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA	I_{OL} = 1.6mA[10]		0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$	I_{OL} = 3.2mA[10]		0.45	V
V_{OL2}	Output low voltage, P1.6/SCL, P1.7/SDA	I_{OL} = 3.0mA		0.4	V
V_{OH}	Output high voltage, ports 1, 2, 3	$-I_{OH}$ = 60µA $-I_{OH}$ = 25µA $-I_{OH}$ = 10µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$		V V V
V_{OH1}	Output high voltage (port 0 in external bus mode, ALE, $\overline{PSEN}$)[11]	$-I_{OH}$ = 400µA $-I_{OH}$ = 150µA $-I_{OH}$ = 40µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$		V V V
R_{RST}	Internal reset pull-down resistor		50	150	kohm
C_{IO}	Pin capacitance	Test freq = 1MHz, T_A = 25°C		10	pF

NOTES: See next page.

NOTES FOR DC ELECTRICAL CHARACTERISTICS:

1. See Figures 8 through 11 for I_{CC} test conditions.
2. The operating supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{CC} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = P1.6 = P1.7 = V_{CC}; f_{CLK} = 12MHz. See Figure 8.
3. This applies to 0 to 70°C and –40 to +85°C temperature range devices.
4. This applies to the –40 to +125°C temperature range device.
5. The idle mode supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{CC} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = Port 0 = P1.6 = P1.7 = V_{CC}; f_{CLK} = 12MHz. See Figure 9.
6. The power-down current is measured with all output pins disconnected; XTAL2 not connected; $\overline{EA}$ = Port 0 = P1.6 = P1.7 = V_{CC}; RST = V_{SS}. See Figure 11.
7. $2V \leq V_{PD} \leq V_{CC}max$.
8. The input threshold voltage of P1.6 and P1.7 (SIO1) meets the I^2C specification, so an input voltage below 1.5V will be recognized as a logic 0 while an input voltage above 3.0V will be recognized as a logic 1.
9. Pins of ports 1 , 2, and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
10. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input. I_{OL} can exceed these conditions provided that no single output sinks more than 5mA and no more than two outputs exceed the test conditions.
11. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.

AC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C, or T_A = −40°C to +85°C/+125°C, V_{CC} = 5V ±10%, V_{SS} = 0V[1, 2]

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	1	Oscillator frequency			1.2	16	MHz
t_{LHLL}	1	ALE pulse width	127		$2t_{CLCL}-40$		ns
t_{AVLL}	1	Address valid to ALE low	28		$t_{CLCL}-55$		ns
t_{LLAX}	1	Address hold after ALE low	48		$t_{CLCL}-35$		ns
t_{LLIV}	1	ALE low to valid instruction in		233		$4t_{CLCL}-100$	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	43		$t_{CLCL}-40$		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	205		$3t_{CLCL}-45$		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		145		$3t_{CLCL}-105$	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		59		$t_{CLCL}-25$	ns
t_{AVIV}	1	Address to valid instruction in		312		$5t_{CLCL}-105$	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{AVLL}	2, 3	Address valid to ALE low	48		$t_{CLCL}-35$		ns
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		252		$5t_{CLCL}-165$	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		97		$2t_{CLCL}-70$	ns
t_{LLDV}	2, 3	ALE low to valid data in		517		$8t_{CLCL}-150$	ns
t_{AVDV}	2, 3	Address to valid data in		585		$9t_{CLCL}-165$	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}-130$		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	23		$t_{CLCL}-60$		ns
t_{DW}	2, 3	Data setup time before $\overline{WR}$	433		$7t_{CLCL}-150$		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	33		$t_{CLCL}-50$		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
Shift Register[3]							
t_{XLXL}	4	Serial port clock cycle time[4]	1.0		$12t_{CLCL}$		µs
t_{QVXH}	4	Output data setup to clock rising edge[4]	700		$10t_{CLCL}-133$		ns
t_{XHQX}	4	Output data hold after clock rising edge[4]	50		$2t_{CLCL}-117$		ns
t_{XHDX}	4	Input data hold after clock rising edge[4]	0		0		ns
t_{XHDV}	4	Clock rising edge to input data valid[4]		700		$10t_{CLCL}-133$	ns
External Clock							
t_{CHCX}	5	High time[4]	20		20	$t_{CLCL}-t_{LOW}$	ns
t_{CLCX}	5	Low time[4]	20		20	$t_{CLCL}-t_{HIGH}$	ns
t_{CLCH}	5	Rise time[4]		20		20	ns
t_{CHCL}	5	Fall time[4]		20		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. The shift register has been characterized for the 87C652 only.
4. These values are characterized but not 100% production tested.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

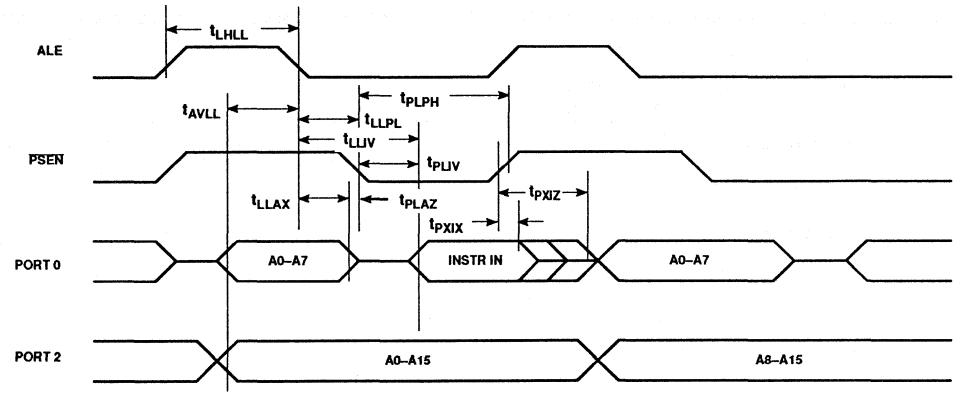


Figure 1. External Program Memory Read Cycle

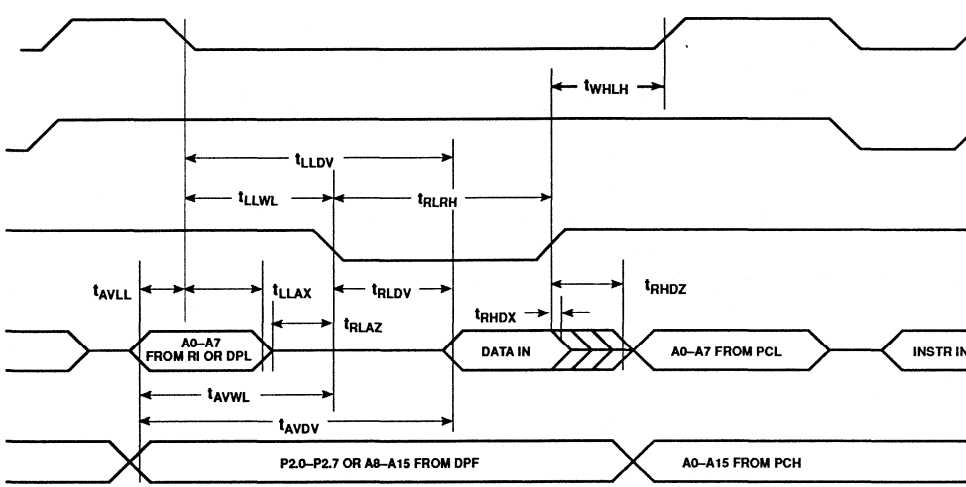


Figure 2. External Data Memory Read Cycle

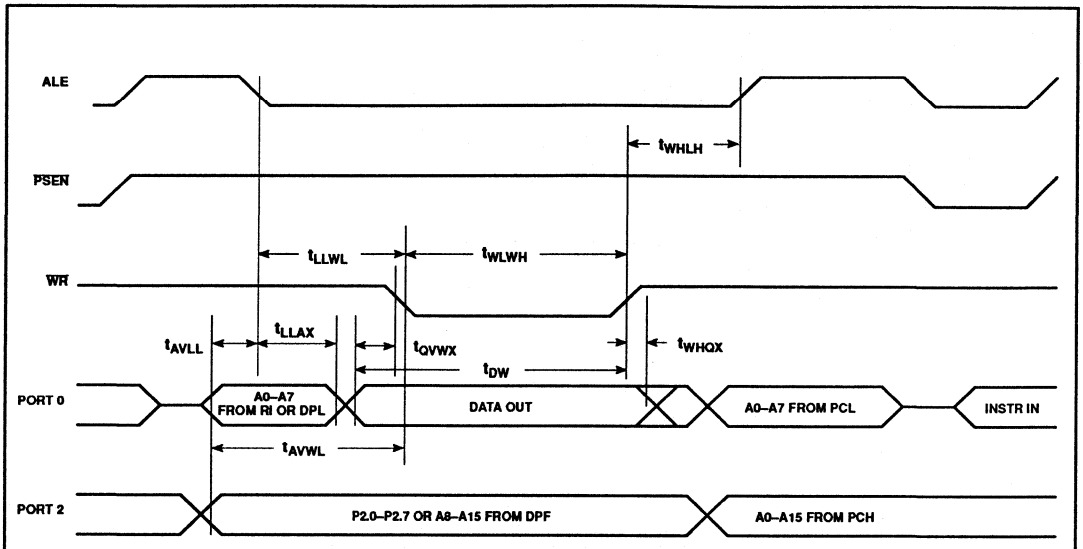


Figure 3. External Data Memory Write Cycle

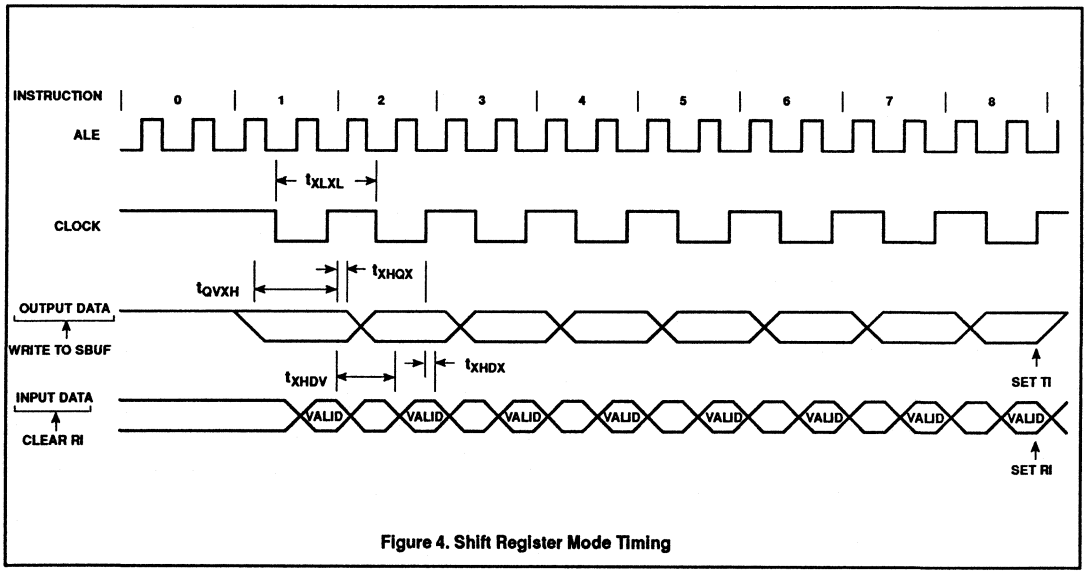


Figure 4. Shift Register Mode Timing

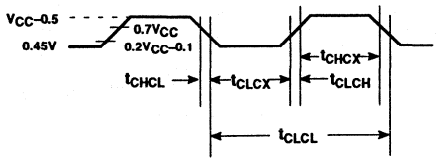
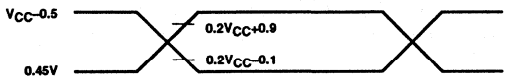
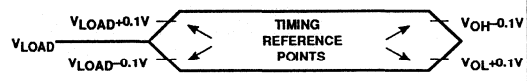


Figure 5. External Clock Drive



NOTE:
AC INPUTS DURING TESTING ARE DRIVEN AT V_{CC}–0.5 FOR A LOGIC '1' AND 0.45V FOR A LOGIC '0'. TIMING MEASUREMENTS ARE MADE AT V_{IH} MIN FOR A LOGIC '1' AND V_{IL} MAX FOR A LOGIC '0'.

Figure 6. AC Testing Input/Output



NOTE:
FOR TIMING PURPOSES, A PORT IS NO LONGER FLOATING WHEN A 100MV CHANGE FROM LOAD VOLTAGE OCCURS, AND BEGINS TO FLOAT WHEN A 100mV CHANGE FROM THE LOADED V_{OH}/V_{OL} LEVEL OCCURS. $I_{OH}/I_{OL} \geq \pm$ 20mA.

Figure 7. Float Waveform

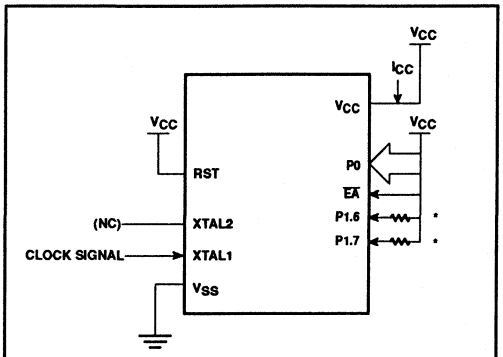


Figure 8. I_{CC} Test Condition, Active Mode
All other pins are disconnected

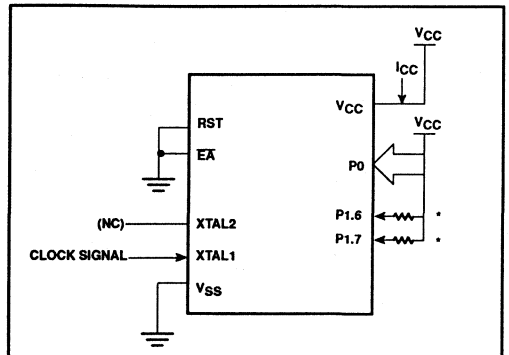


Figure 9. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

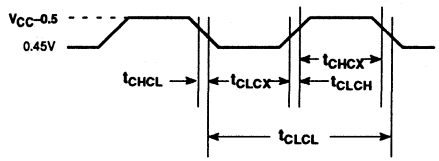


Figure 10. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 10ns$

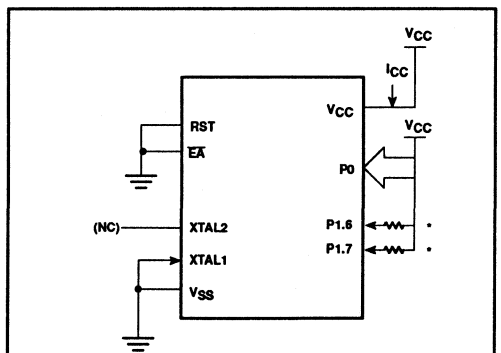


Figure 11. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected. V_{CC} = 2V to 5.5V

* NOTE:
Ports 1.6 and 1.7 should be connected to V_{CC} through resistors of sufficiently high value such that the sink current into these pins does not exceed the I_{OL1} specification.

EPROM CHARACTERISTICS

The 87C652 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C652 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C652 manufactured by Philips Components.

Table 3 shows the logic levels for reading the signature byte, and for programming the program memory, the encryption table, and the lock bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 12 and 13. Figure 14 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 12. Note that the 87C652 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1 and 2, as shown in Figure 12. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$ and pins of ports 2 and 3 specified in Table 3 are held at the 'Program Code Data' levels indicated in Table 3. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 13.

To program the encryption table, repeat the 25 pulse programming sequence for addresses 0 through 1FH, using the "Pgm Encryption Table' levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the lock bits, repeat the 25 pulse programming sequence using the 'Pgm Lock Bit' levels. After one lock bit is programmed, further programming of the code memory and encryption table is disabled. However, the other lock bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If lock bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be read is applied to ports 1 and 2 as shown in Figure 14. The other pins are held at the 'Verify Code Data' levels indicated in Table 3. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 5H indicates manufactured by Philips

(031H) = 99H indicates 87C652

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 3, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345–5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000uW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient. Erasure leaves the array in an all 1s state.

Table 3. EPROM Programming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm lock bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm lock bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:

1. '0' = Valid low for that pin, '1' = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{CC} = 5V±10% during programming and verification.

*ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

™Trademark phrase of Intel Corporation.

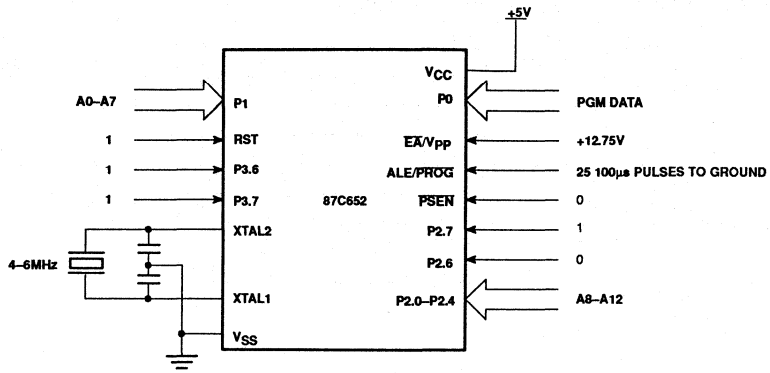


Figure 12. Programming Configuration

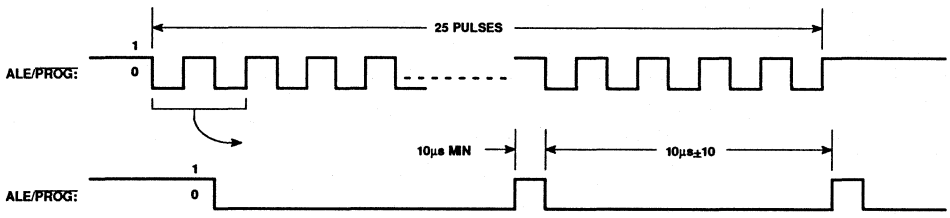


Figure 13. PROG Waveform

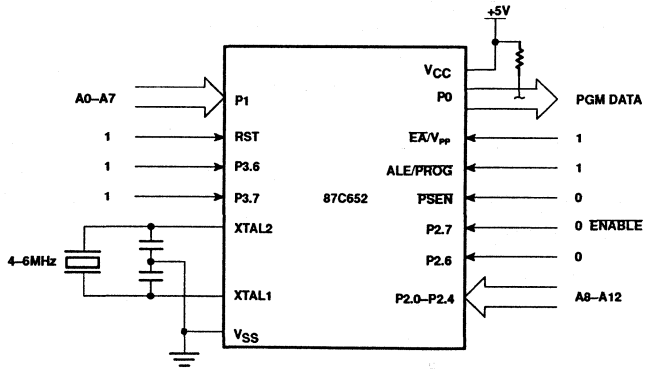


Figure 14. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_A = 21°C to +27°C, V_{CC} = 5V±10%, V_{SS} = 0V (See Figure 15)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 (ENABLE) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		µs
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	$\overline{ENABLE}$ low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		µs

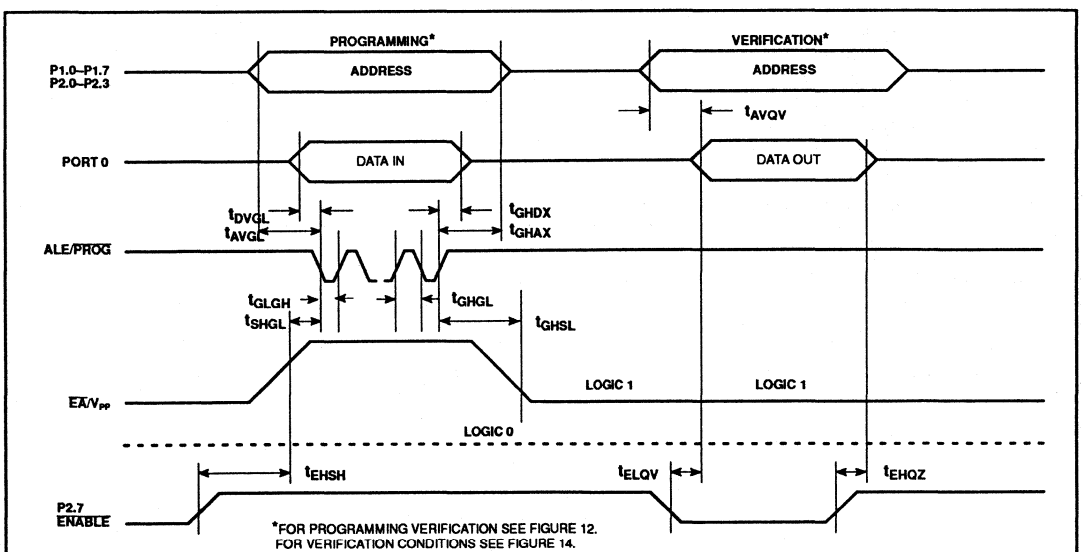


Figure 15. EPROM Programming and Verification

Philips Components

Date of Issue	September 6, 1990
Status	Product Specification
Application Specific Products	

83C654/87C654

CMOS single-chip 8-bit microcontroller

DESCRIPTION

The 83C654/87C654 Single-Chip 8-Bit Microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. The 83C654/87C654 has the same instruction set as the 80C51. Two versions of the derivative exist:

83C654 – 16k bytes mask programmable ROM

87C654 – EPROM version

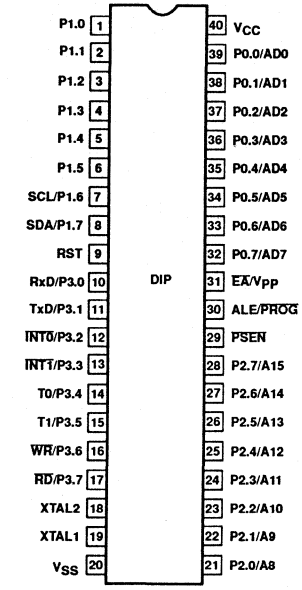
This device provides architectural enhancements that make it applicable in a variety of applications for general control systems. The 8XC654 contains a non-volatile 16k X 8 read-only program memory (83C654) EPROM (87C654), a volatile 256 X 8 read/write data memory, four 8-bit I/O ports, two 16-bit timer/event counters (identical to the timers of the 80C51), a multi-source, two-priority-level, nested interrupt structure, an I^2C interface, UART and on-chip oscillator and timing circuits. For systems that require extra capability, the 8XC654 can be expanded using standard TTL compatible memories and logic.

The device also functions as an arithmetic processor having facilities for both binary and BCD arithmetic plus bit-handling capabilities. The instruction set consists of over 100 instructions: 49 one-byte, 45 two-byte and 17 three-byte. With a 12MHz crystal, 58% of the instructions are executed in 1μs and 40% in 2μs. Multiply and divide instructions require 4μs.

FEATURES

- 80C51 central processing unit
- 16k X 8 ROM expandable externally to 64k bytes
- 256 X 8 RAM, expandable externally to 64k bytes
- Two standard 16-bit timer/counters
- Four 8-bit I/O ports
- I^2C-bus serial I/O port with byte oriented master and slave functions
- Full-duplex UART facilities
- Power control modes
  - Idle mode
  - Power-down mode
- Five package styles
- Extended temperature ranges
- OTP package available

PIN CONFIGURATION



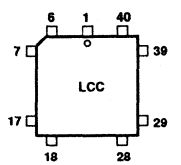
SEE NEXT PAGE FOR QFP AND LCC PIN FUNCTIONS.

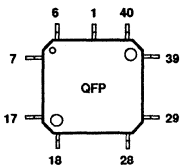
CMOS single-chip 8-bit microcontroller 83C654/87C654

PART NUMBER SELECTION

PHILIPS		PHILIPS COMPONENTS-SIGNETICS			TEMPERATURE °C AND PACKAGE	FREQ. MHz
ROMless	ROM	ROMless	ROM	EPROM		
PCB80C652P	PCB83C654P	S80C652-1N40	S83C654-1N40	S87C654-1N40	0 to +70, PDIP	12
				S87C654-1F40	0 to +70, CDIP	12
PCB80C652WP	PCB83C654WP	S80C652-1A44	S83C654-1A44	S87C654-1A44	0 to +70, PLCC	12
				S87C654-1K44	0 to +70, CLCC	12
PCB80C652H	PCB83C654H	S80C652-1B44	S83C654-1B44		0 to +70, PQFP	12
PCF80C652P	PCF83C654P	S80C652-2N40	S83C654-2N40	S87C654-2N40	–40 to +85, PDIP	12
				S87C654-2F40	–40 to +85, CDIP	12
PCF80C652WP	PCF83C654WP	S80C652-2A44	S83C654-2A44	S87C654-2A44	–40 to +85, PLCC	12
				S87C654-2K44	–40 to +85, CLCC	12
PCF80C652H	PCF83C654H	S80C652-2B44	S83C654-2B44		–40 to +85, PQFP	12
				S87C654-4N40	0 to +70, PDIP	16
				S87C654-4F40	0 to +70, CDIP	16
				S87C654-4A44	0 to +70, PLCC	16
				S87C654-4K44	0 to +70, CLCC	16
				S87C654-5N40	–40 to +85, PDIP	16
				S87C654-5F40	–40 to +85, CDIP	16
				S87C654-5A44	–40 to +85, PLCC	16
				S87C654-5K44	–40 to +85, CLCC	16
PCA80C652P	PCA83C654P	S80C652-6N40	S83C654-6N40		–40 to +125, PDIP	12
PCA80C652WP	PCA83C654WP	S80C652-6A44	S83C654-6A44		–40 to +125, PLCC	12
PCA80C652H	PCA83C654H	S80C652-6B44	S83C654-6B44		–40 to +125, PQFP	12

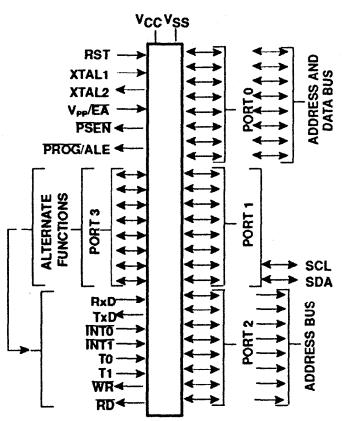
LCC AND QFP PIN FUNCTIONS



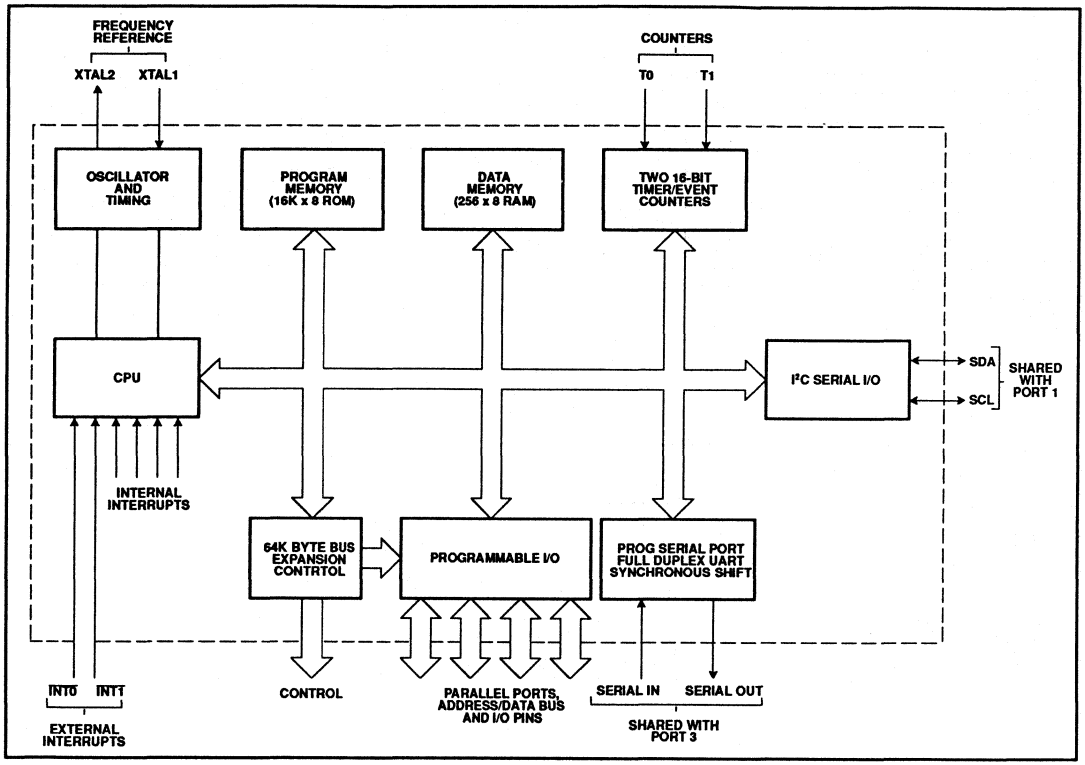


Pin	Function	Pin	Function
1	NC	23	V_{SS}
2	P1.0	24	P2.0/A8
3	P1.1	25	P2.1/A9
4	P1.2	26	P2.2/A10
5	P1.3	27	P2.3/A11
6	P1.4	28	P2.4/A12
7	P1.5	29	P2.5/A13
8	P1.6/SCL	30	P2.6/A14
9	P1.7/SDA	31	P2.7/A15
10	RST	32	PSEN
11	P3.0/RxD	33	ALE/PROG
12	NC	34	NC
13	P3.1/TxD	35	$\overline{EA}/V_{PP}$
14	P3.2/$\overline{INT0}$	36	P0.7/AD7
15	P3.3/$\overline{INT1}$	37	P0.6/AD6
16	P3.4/T0	38	P0.5/AD5
17	P3.5/T1	39	P0.4/AD4
18	P3.6/$\overline{WR}$	40	P0.3/AD3
19	P3.7/$\overline{RD}$	41	P0.2/AD2
20	XTAL2	42	P0.1/AD1
21	XTAL1	43	P0.0/AD0
22	V_{SS}	44	V_{CC}

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC/QFP	TYPE	NAME AND FUNCTION
V_{SS}	20	22	I	**Ground:** 0V reference.
V_{CC}	40	44	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–46	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s. Port 0 also outputs the code bytes during program verification in the 87C654. External pull-ups are required during program verification.
P1.0–P1.7	1–8	2–9	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups, except P1.6 and P1.7 which are open drain. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 1 also receives the low-order address byte during program memory verification. Alternate functions include:
P1.6	7	8	I/O	**SCL:** I^2C-bus serial port clock line.
P1.7	8	9	I/O	**SDA:** I^2C-bus serial port data line.
P2.0–P2.7	21–28	24–31	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the 80C51 family, as listed below:
	10	11	I	**RxD (P3.0):** Serial input port
	11	13	O	**TxD (P3.1):** Serial output port
	12	14	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	I	**T0 (P3.4):** Timer 0 external input
	15	17	I	**T1 (P3.5):** Timer 1 external input
	16	18	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE/$\overline{PROG}$	30	33	I/O	**Address Latch Enable/Program Pulse:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.
$\overline{PSEN}$	29	32	O	**Program Store Enable:** The read strobe to external program memory. When the 87C654 is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$/V_{PP}	31	35	I	**External Access Enable/Programming Supply Voltage:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H and 1FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 1FFFH. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
XTAL1	19	21	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	O	**Crystal 2:** Output from the inverting oscillator amplifier.

NOTE:
To avoid "latch-up" effect at power-on, the voltage on any pin at any time must not be higher than V_{CC} + 0.5V or V_{SS} – 0.5V, respectively.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol, page 2.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

Reset

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-on reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-on, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

Idle Mode

In the idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

Power-Down Mode

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. the control bits for the reduced power modes are in the special function register PCON. Table 1 shows the state of the I/O ports during low current operating modes.

Table 1. External Pin Status During Idle and Power-Down Mode

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

Serial Control Register (S1CON) – See Table 2

S1CON (D8H)	CR2	ENS1	STA	STO	SI	AA	CR1	CR0

Bits CR0, CR1 and CR2 determine the serial clock frequency that is generated in the master mode of operation.

Table 2. Serial Clock Rates

CR2[1]	CR1	CR0	BIT FREQUENCY (kHz) AT f_{osc} 6MHz	12MHz	16MHz[1]	f_{osc} DIVIDED BY
0	0	0	23	47	62.5	256[1]
0	0	1	27	54	71	224[1]
0	1	0	31.25	62.5	83.3	192[1]
0	1	1	37	75	100	160[1]
1	0	0	6.25	12.5	17	960
1	0	1	50	100	133[2]	120
1	1	0	100	200[2]	267[2]	60
1	1	1	0.25 < 31.25	0.5 < 62.5	0.67 < 83.3	96 x (256 – (reload value Timer 1)) (Reload value range: 0 – 254 in mode 2)

NOTES:
1. The CR2 control bit is only implemented on the 16MHz version.
2. These frequencies exceed the upper limit of 100kHz of the I²C-bus specification and cannot be used in an I²C-bus application.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Voltage on $\overline{EA}/V_{PP}$ to V_{SS} (87C654 only)	–0.5 to + 13	V
Voltage on any other pin to V_{SS}	–0.5 to + 6.5	V
Input, output current on any single pin	±5	mA
Input, output current on any two pins	±10	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C, or T_A = –40°C to +85°C/+125°C, V_{CC} = 5V ±10%, V_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{CC}	Supply voltage		4.5	5.5	V
I_{CC}	Supply current:	See note 1			
	Active mode	See notes 2 and 3		38	mA
		See notes 2 and 4		38	mA
	Idle mode	See notes 3 and 5		8	mA
		See notes 4 and 5		8	mA
	Power-down mode	See notes 3, 6, and 7		50	µA
		See notes 4, 6, and 7		135	µA
Inputs					
V_{IL}	Input low voltage, except $\overline{EA}$,P1.6/SCL, P1.7/SDA	T_A = 0°C to +70°C	–0.5	$0.2V_{CC}$–0.1	V
		T_A = –40°C to +85°C	–0.5	$0.2V_{CC}$–0.15	
		T_A = –40°C to +125°C	–0.5	$0.2V_{CC}$–0.25	
V_{IL1}	Input low voltage to $\overline{EA}$	T_A = 0°C to +70°C	–0.5	$0.2V_{CC}$–0.3	V
		T_A = –40°C to +85°C	–0.5	$0.2V_{CC}$–0.35	
		T_A = –40°C to +125°C	–0.5	$0.2V_{CC}$–0.45	
V_{IL2}	Input low voltage to P1.6/SCL, P1.7/SDA[8]		–0.5	$0.3V_{CC}$	V
V_{IH}	Input high voltage, except XTAL1, RST, P1.6/SCL, P1.7/SDA	T_A = 0°C to +70°C	$0.2V_{CC}$+0.9	V_{CC}+0.5	V
		T_A = –40°C to +85°C	$0.2V_{CC}$+1.0	V_{CC}+0.5	
		T_A = –40°C to +125°C	$0.2V_{CC}$+1.0	V_{CC}+0.5	
V_{IH1}	Input high voltage, XTAL1, RST	T_A = 0°C to +70°C	$0.7V_{CC}$	V_{CC}+0.5	V
		T_A = –40°C to +85°C	$0.7V_{CC}$ + 0.1	V_{CC}+0.5	
		T_A = –40°C to +125°C	$0.7V_{CC}$ + 0.1	V_{CC}+0.5	
V_{IH2}	Input high voltage, P1.6/SCL, P1.7/SDA[8]		$0.7V_{CC}$	6.0	V
$-I_{IL}$	Logical 0 input current, ports 1, 2, 3, 4, except P1.6/SCL, P1.7/SDA	V_{IN} = 0.45V			µA
		T_A = 0°C to +70°C		50	
		T_A = –40°C to +85°C		75	
		T_A = –40°C to +125°C		75	
$-I_{TL}$	Logical 1-to-0 transition current, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA	See note 9			µA
		T_A = 0°C to +70°C		650	
		T_A = –40°C to +85°C		750	
		T_A = –40°C to +125°C		750	
$\pm I_{IL1}$	Input leakage current, port 0, $\overline{EA}$	0.45V < V_I < V_{CC}		10	µA
$\pm I_{IL2}$	Input leakage current, P1.6/SCL, P1.7/SDA	0V < V_I < 5.5V 0V < V_{CC} < 5.5V		10	µA
Outputs					
V_{OL}	Output low voltage, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA	I_{OL} = 1.6mA[10, 11]		0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$	I_{OL} = 3.2mA[10, 11]		0.45	V
V_{OL2}	Output low voltage, P1.6/SCL, P1.7/SDA	I_{OL} = 3.0mA[10, 11]		0.4	V
V_{OH}	Output high voltage, ports 1, 2, 3	$-I_{OH}$ = 60µA	2.4		V
		$-I_{OH}$ = 25µA	$0.75V_{CC}$		V
		$-I_{OH}$ = 10µA	$0.9V_{CC}$		V
V_{OH1}	Output high voltage (port 0 in external bus mode, ALE, $\overline{PSEN}$)[12]	$-I_{OH}$ = 800µA	2.4		V
		$-I_{OH}$ = 300µA	$0.75V_{CC}$		V
		$-I_{OH}$ = 80µA	$0.9V_{CC}$		V
R_{RST}	Internal reset pull-down resistor		50	150	kohm
C_{IO}	Pin capacitance	Test freq = 1MHz, T_A = 25°C		10	pF

NOTES: See Next Page.

NOTES FOR DC ELECTRICAL CHARACTERISTICS

1. See Figures 8 through 11 for I_{CC} test conditions.
2. The operating supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{CC} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = P1.6 = P1.7 = V_{CC}; f_{CLK} = 16MHz. See Figure 8.
3. This applies to 0 to 70°C and –40 to +85°C temperature range devices.
4. This applies to the –40 to +125°C temperature range device.
5. The idle mode supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{CC} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = Port 0 = P1.6 = P1.7 = V_{CC}; RST = V_{SS}; f_{CLK} = 16MHz. See Figure 9.
6. The power-down current is measured with all output pins disconnected; XTAL2 not connected; $\overline{EA}$ = Port 0 = P1.6 = P1.7 = V_{CC}; RST = V_{SS}. See Figure 11.
7. $2V \leq V_{PD} \leq V_{CC}max$.
8. The input threshold voltage of P1.6 and P1.7 (SIO1) meets the I^2C specification, so an input voltage below $0.3V_{CC}$ will be recognized as a logic 0 while an input voltage above $0.7V_{CC}$ will be recognized as a logic 1.
9. Pins of ports 1 , 2, and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
10. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input. I_{OL} can exceed these conditions provided that no single output sinks more than 5mA and no more than two outputs exceed the test conditions.
11. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows: Maximum I_{OL} = 10mA per port pin; Maximum I_{OL} = 26mA total for Port 0; Maximum I_{OL} = 15mA total for Ports 1, 2, and 3; Maximum I_{OL} = 71mA total for all output pins. If I_{OL} exceeds the test conditions, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
12. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.

AC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C, or T_A = −40°C to +85°C/+125°C, V_{CC} = 5V ±10%, V_{SS} = 0V[1, 2]

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	1	Oscillator frequency			1.2	16	MHz
t_{LHLL}	1	ALE pulse width	127		$2t_{CLCL}-40$		ns
t_{AVLL}	1	Address valid to ALE low	28		$t_{CLCL}-55$		ns
t_{LLAX}	1	Address hold after ALE low	48		$t_{CLCL}-35$		ns
t_{LLIV}	1	ALE low to valid instruction in		233		$4t_{CLCL}-100$	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	43		$t_{CLCL}-40$		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	205		$3t_{CLCL}-45$		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		145		$3t_{CLCL}-105$	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		59		$t_{CLCL}-25$	ns
t_{AVIV}	1	Address to valid instruction in		312		$5t_{CLCL}-105$	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{AVLL}	2, 3	Address valid to ALE low	48		$t_{CLCL}-35$		ns
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		252		$5t_{CLCL}-165$	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		97		$2t_{CLCL}-70$	ns
t_{LLDV}	2, 3	ALE low to valid data in		517		$8t_{CLCL}-150$	ns
t_{AVDV}	2, 3	Address to valid data in		585		$9t_{CLCL}-165$	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}-130$		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	23		$t_{CLCL}-60$		ns
t_{DW}	2, 3	Data setup time before $\overline{WR}$	433		$7t_{CLCL}-150$		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	33		$t_{CLCL}-50$		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
Shift Register[3]							
t_{XLXL}	4	Serial port clock cycle time[4]	1.0		$12t_{CLCL}$		µs
t_{QVXH}	4	Output data setup to clock rising edge[4]	700		$10t_{CLCL}-133$		ns
t_{XHQX}	4	Output data hold after clock rising edge[4]	50		$2t_{CLCL}-117$		ns
t_{XHDX}	4	Input data hold after clock rising edge[4]	0		0		ns
t_{XHDV}	4	Clock rising edge to input data valid[4]		700		$10t_{CLCL}-133$	ns
External Clock							
t_{CHCX}	5	High time[4]	20		20	$t_{CLCL}-t_{LOW}$	ns
t_{CLCX}	5	Low time[4]	20		20	$t_{CLCL}-t_{HIGH}$	ns
t_{CLCH}	5	Rise time[4]		20		20	ns
t_{CHCL}	5	Fall time[4]		20		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. The shift register has been characterized for the 87C654 only.
4. These values are characterized but not 100% production tested.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

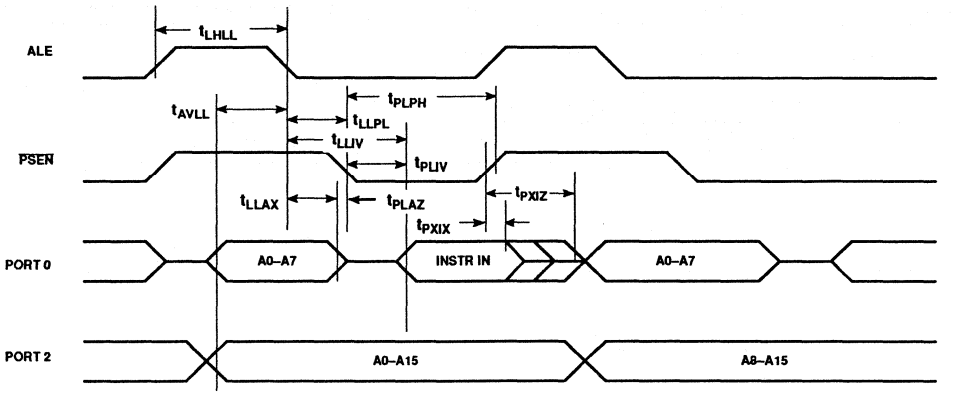


Figure 1. External Program Memory Read Cycle

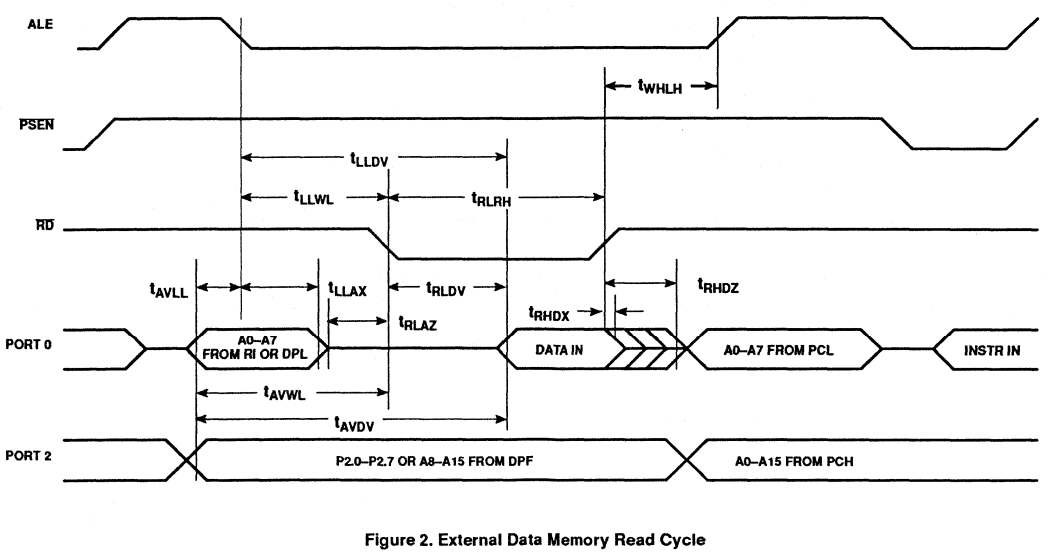


Figure 2. External Data Memory Read Cycle

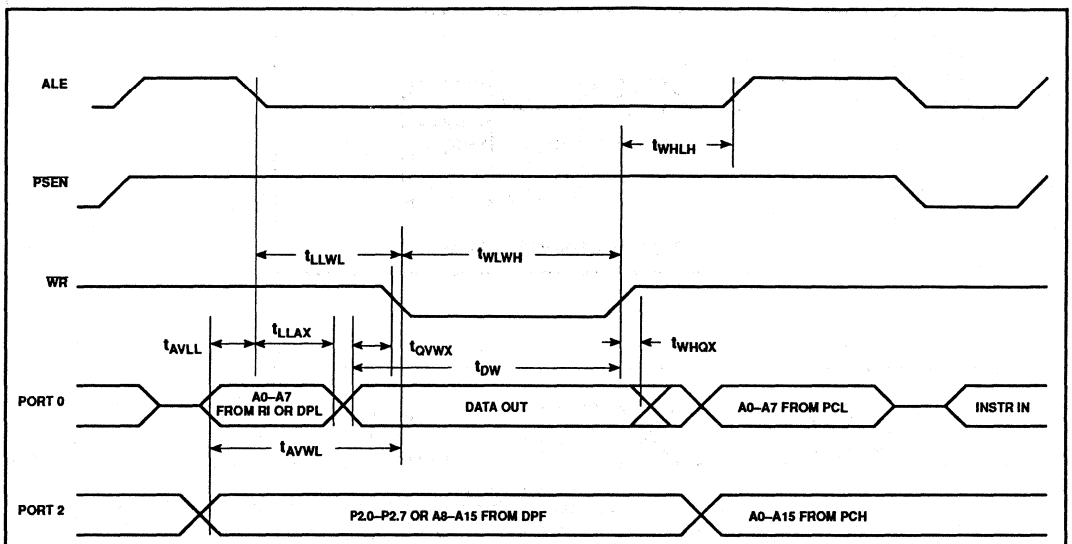


Figure 3. External Data Memory Write Cycle

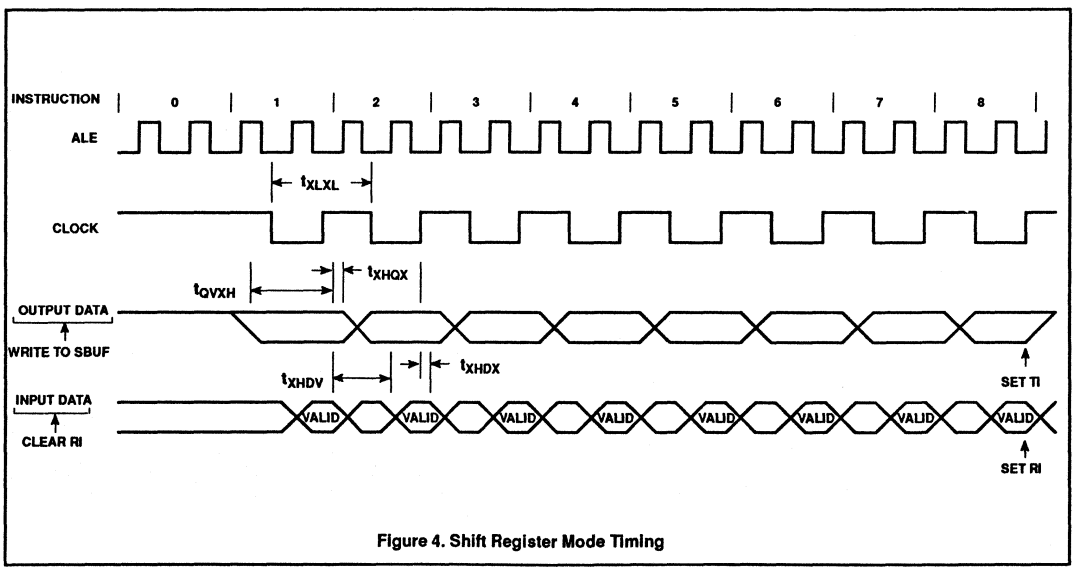


Figure 4. Shift Register Mode Timing

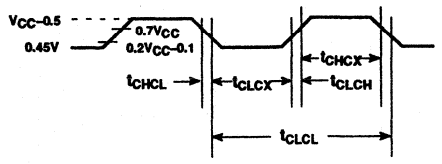
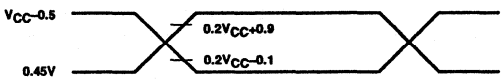
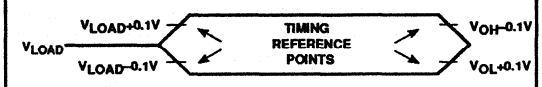


Figure 5. External Clock Drive



NOTE:
AC INPUTS DURING TESTING ARE DRIVEN AT V_{CC}–0.5 FOR A LOGIC '1' AND 0.45V FOR A LOGIC '0'. TIMING MEASUREMENTS ARE MADE AT V_{IH} MIN FOR A LOGIC '1' AND V_{IL} MAX FOR A LOGIC '0'.

Figure 6. AC Testing Input/Output



NOTE:
FOR TIMING PURPOSES, A PORT IS NO LONGER FLOATING WHEN A 100MV CHANGE FROM LOAD VOLTAGE OCCURS, AND BEGINS TO FLOAT WHEN A 100mV CHANGE FROM THE LOADED V_{OH}/V_{OL} LEVEL OCCURS. $I_{OH}/I_{OL} \geq \pm$ 20mA.

Figure 7. Float Waveform

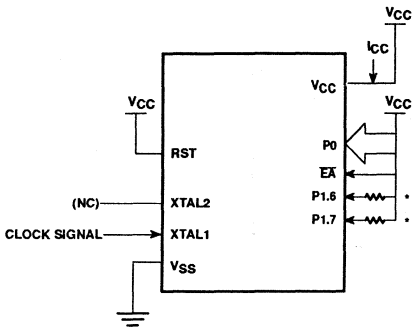


Figure 8. I_{CC} Test Condition, Active Mode
All other pins are disconnected

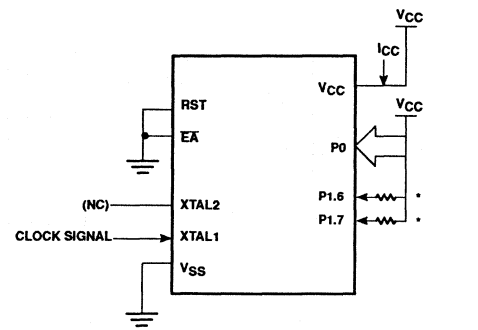


Figure 9. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

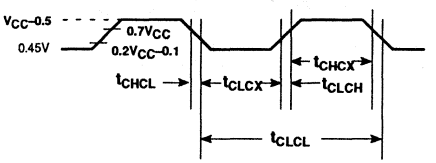


Figure 10. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 10ns$

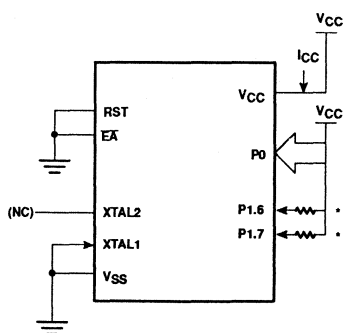


Figure 11. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected. V_{CC} = 2V to 5.5V

* NOTE:
Ports 1.6 and 1.7 should be connected to V_{CC} through resistors of sufficiently high value such that the sink current into these pins does not exceed the I_{OL1} specification.

EPROM CHARACTERISTICS

The 87C654 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C654 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C654 manufactured by Philips Components.

Table 3 shows the logic levels for reading the signature byte, and for programming the program memory, the encryption table, and the lock bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 12 and 13. Figure 14 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 12. Note that the 87C654 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1 and 2, as shown in Figure 12. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$ and pins of ports 2 and 3 specified in Table 3 are held at the 'Program Code Data' levels indicated in Table 3. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 13.

To program the encryption table, repeat the 25 pulse programming sequence for addresses 0 through 1FH, using the "Pgm Encryption Table' levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the lock bits, repeat the 25 pulse programming sequence using the 'Pgm Lock Bit' levels. After one lock bit is programmed, further programming of the code memory and encryption table is disabled. However, the other lock bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If lock bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be read is applied to ports 1 and 2 as shown in Figure 14. The other pins are held at the 'Verify Code Data' levels indicated in Table 3. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips

(031H) = 99H indicates 87C654

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 3, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345–5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000uW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient. Erasure leaves the array in an all 1s state.

Table 3. EPROM Programming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm lock bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm lock bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:

1. '0' = Valid low for that pin, '1' = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{CC} = 5V±10% during programming and verification.

*ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

™Trademark phrase of Intel Corporation.

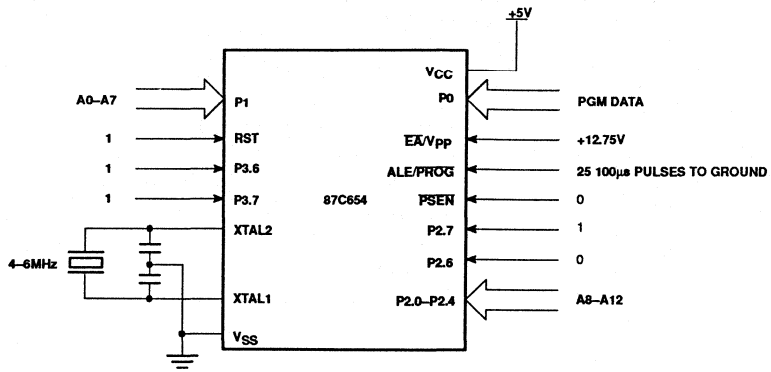


Figure 12. Programming Configuration

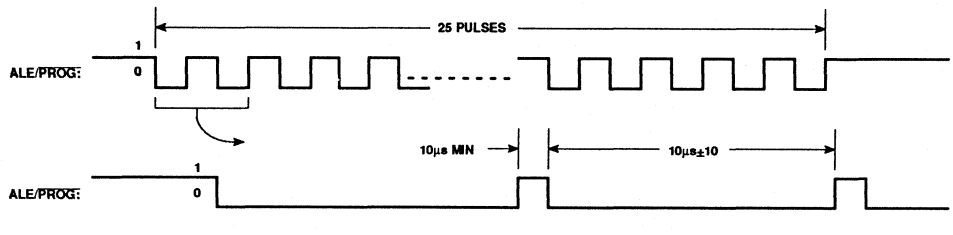


Figure 13. PROG Waveform

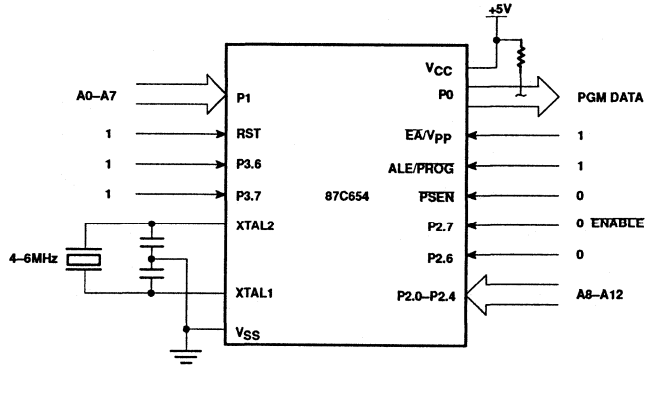


Figure 14. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_A = 21°C to +27°C, V_{CC} = 5V±10%, V_{SS} = 0V (See Figure 15)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 (ENABLE) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		µs
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	$\overline{ENABLE}$ low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		µs

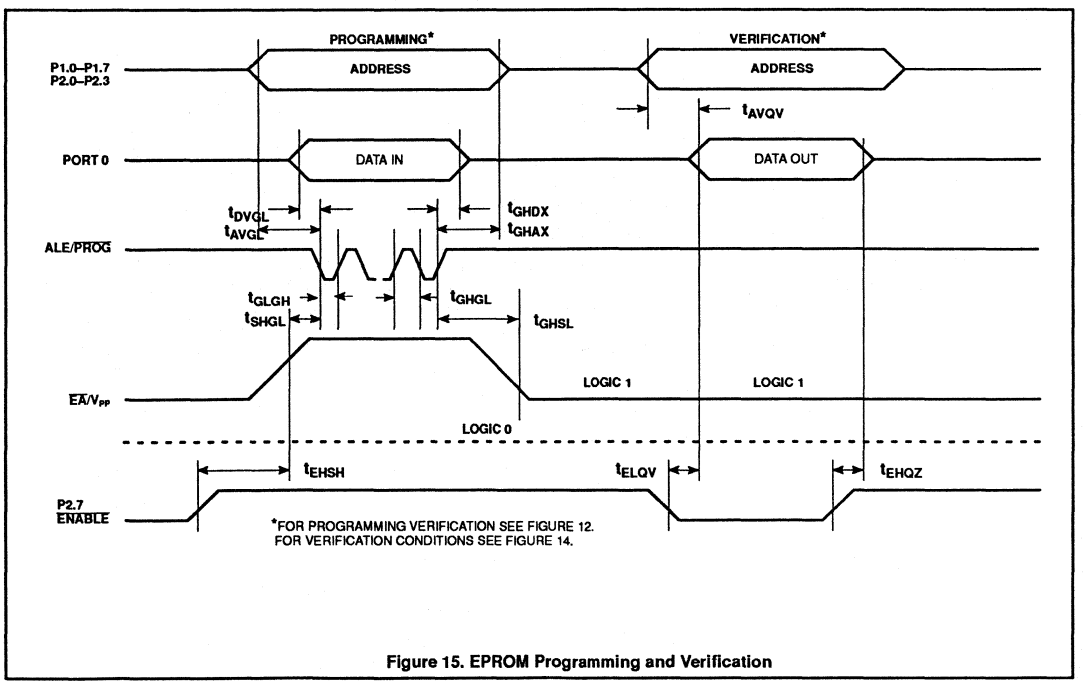


Figure 15. EPROM Programming and Verification

8XC751 OVERVIEW

The Signetics 83C751/87C751 offers the advantages of the SC80C51 architecture in a small package and at a low cost. This microcontroller is fabricated with Signetics high-density CMOS technology. Signetics epitaxial substrate minimizes CMOS latch-up sensitivity. The 83C751/87C751 (hereafter referred to collectively as the 83C751) contains a 2k x 8 ROM/EPROM, a 64 x 8 RAM, 19 I/O lines, a 16-bit auto-reload counter/timer, a fixed rate timer, a five source fixed priority interrupt structure, a bidirectional Inter-Integrated Circuit (I^2C) serial bus interface, and an on-chip oscillator. The onboard inter-integrated circuit (I^2C) bus interface allows the 83C751 to operate as a master or slave device on the I^2C small area network. This capability facilitates I/O and RAM expansion, access to EPROM, processor to processor communication, and efficient interface to a wide variety of dedicated I^2C peripherals. The 83C751 has the following features:

- SC80C51 based architecture
- Boolean processor
- Inter-integrated Circuit (I^2C) serial bus interface
- Fixed-rate timer
- 16-bit auto reloadable counter/timer
- Small package sizes
  - 24-pin DIP (300 mil "skinny DIP")
  - 28-pin PLCC
- 2k x 8 ROM/EPROM
- Available in erasable quartz lid (87C751), one-time programmable (87C751), or mask programmable versions (83C751)
- Wide oscillator frequency range
- Low power consumption:
  - Normal operation: less than 11mA @ 5V, 12MHz
- Idle mode
- Power-down mode
- CMOS and TTL compatible

This part is well suited for logic replacement in consumer and industrial applications.

Differences from the 80C51

Instruction Set

PLEASE NOTE: The instruction set of the 83C751 is identical to the 80C51 except for the instructions: MOVX, LCALL, and LJUMP, which are not implemented. Care must be taken not to use any of these instructions in a user program, especially when using a high level language such as C.

Memory Organization

The central processing unit (CPU) manipulates operands in two address spaces as shown in Figure 67. The part's internal memory space consists of 2k bytes of program memory, and 64 bytes of data RAM overlapped with the 128-byte special function register area. The differences from the 80C51 are in RAM size (64 bytes vs. 128 bytes), in external RAM access (not available on the 83C751), in internal ROM size (2k bytes vs. 4k bytes), and in external program memory expansion (not available on the 83C751). The 128-byte special function register (SFR) space is accessed as on the 80C51 with some of the registers having been changed to reflect changes in the 83C751 peripheral functions. The stack may be located anywhere in internal RAM by loading the 8-bit stack pointer (SP). It should be noted that stack depth is limited to 64 bytes, the amount of available RAM. A reset loads the stack pointer with 07 (which is pre-incremented on a PUSH instruction).

Special Function Registers

The 83C751 contains many of the special function registers (SFR) that are found on the 80C51. Due to the different peripheral features on the 83C751, there are several additional SFRs and several that have been changed. There is no port 2 on the 83C751 so the P2 SFR isn't used. The standard UART found on the 80C51 has been replaced by the I^2C serial interface, so the UART SFRs, SCON, and SBUF have been replaced by I2CON and i2DAT, and two additional I^2C registers have been added (I2STA and I2CFG).

Because the interrupt structure is single level on the 83C751, there is no need for the IP SFR, so it is not used. The counter/timer has only one mode of operation, so the TMOD SFR is not used. There is also only one counter/timer, so there is no need for the TL1 and TH1 SFRs found on the 80C51. These have been replaced on the 83C751 by RTL and RTH, the counter/timer reload registers. Table 26 shows the special function registers, their locations, and reset values.

Data Pointer (DPTR)

The data pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). In the 80C51 this register allows the access of external data memory using the MOVX instruction. Since the 83C751 does not support MOVX or external memory accesses, this register is generally used as a 16-bit offset pointer of the accumulator in a MOVC instruction. DPTR may also be manipulated as two independent 8-bit registers.

I/O Port Latches (P0, P1, P3)

The port latches function the same as those on the 80C51. Since there is no port 2 on the 83C751, the P2 latch is not used. Port 0 on the 83C751 has only 3 bits, so only 3 bits of the P0 SFR have a useful function.

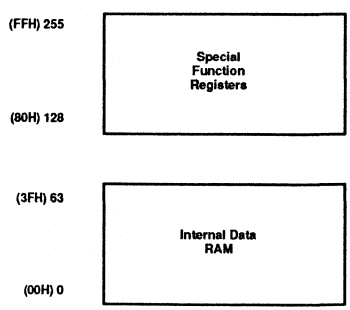


Figure 67. Memory Map

Table 26. 8XC751 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR:	Data pointer (2 bytes)										
DPH	High byte	83H									00H
DPL	Low byte	82H									00H
			DF	DE	DD	DC	DB	DA	D9	D8	
I²CFG*#	I²C configuration	D8H/RD	SLAVEN	MASTRQ	0	TIRUN	–	–	CT1	CT0	0000xx00B
		WR	SLAVEN	MASTRQ	CLRTI	TIRUN	–	–	CT1	CT0	
			9F	9E	9D	9C	9B	9A	99	98	
I²CON*#	I²C control	98H/RD	RDAT	ATN	DRDY	ARL	STR	STP	MASTER	–	81H
		WR	CXA	IDLE	CDR	CARL	CSTR	CSTP	XSTR	XSTP	
I²DAT*#	I²C data	99H/RD	RDAT	0	0	0	0	0	0	0	80H
		WR	XDAT	X	X	X	X	X	X	X	
			FF	FE	FD	FC	FB	FA	F9	F8	
I²STA*#	I²C control	F8H	–	IDLE	XDATA	XACTV	MAKSTR	MAKSTP	XSTR	XSTP	x0100000B
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*#	Interrupt enable	A8H	EA	–	–	EI2	ETI	EX1	ET0	EX0	00H
P0*#	Port 0	80H	–	–	–	–	–	82	81	80	xxxxx111B
P1*	Port 1	90H	97	96	95	94	93	92	91	90	FFH
P3*	Port 3	B0H	B7	B6	B5	B4	B3	B2	B1	B0	FFH
PCON	Power control	87H	–	–	–	–	–	–	PD	IDL	xxxxxx00B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*#	Timer/counter control	88H	GATE	C/T	TF	TR	IE0	IT0	IE1	IT1	00H
TI#	Timer low byte	8AH									00H
TH#	Timer high byte	8CH									00H
RTL#	Timer low reload	8BH									00H
RTH#	Timer high reload	8DH									00H

*SFRs are bit addressable.
#SFRs are modified from or added to the 80C51 SFRs.

I/O Port Structure

The 8XC751 has two 8-bit ports (ports 1 and 3) and one 3-bit port (port 0). All three ports on the 8XC751 are bidirectional. Each consists of a latch (special function register P0, P1, P3), an output driver, and an input buffer. Three port 1 pins and two port 0 pins are multifunctional. In addition to being port pins, these pins serve the function of special features as follows:

Port Pin	Alternate Function
P0.0	I²C clock (SCL)
P0.1	I²C data (SDA)
P1.5	INT0 (external interrupt 0 input)
P1.6	INT1 (external interrupt 1 input)
P1.7	T0 (timer 0 external input)

Ports 1 and 3 are identical in structure to the same ports on the 80C51. The structure of port 0 on the 8XC751 is similar to that of the 80C51 but does not include address/data input and output circuitry. As on the 80C51, ports 1 and 3 are quasi-bidirectional while port 0 is bidirectional with no internal pullups.

Timer/Counter

The 8XC751 has two timers: a 16-bit timer/counter and a 10-bit fixed-rate timer. The 16-bit timer/counter's operation is similar to mode 2 operation on the 80C51, but is extended to 16 bits. The timer/counter is clocked by either 1/12 the oscillator frequency or by transitions on the T0 pin. The C/T pin in special function register TCON selects between these two modes. When the TCON TR bit is set, the timer/counter is enabled. Register pair TH and TL are incremented by the clock source. When the register pair overflows, the register pair is reloaded with the values in registers RTH and RTL. The value in the reload registers is left unchanged. See the 83C751 counter/timer block diagram in Figure 68. The TF bit in special function register TCON is set on counter overflow and, if the interrupt is enabled, will generate an interrupt.

TCON Register

GATE	C/T	TF	TR	IE0	IT0	IE1	IT1
MSB							LSB

GATE 1 – Timer/counter is enabled only when INT0 pin is high, and TR is 1.
0 – Timer/counter is enabled when TR is 1.

C/T 1 – Counter/timer operation from T0 pin.
0 – Timer operation from internal clock.

TF 1 – Set on overflow of TH.
0 – Cleared when processor. vectors to interrupt routine and by reset.

TR 1 – Timer/counter enabled.
0 – Timer/counter disabled.

IE0 1 – Edge detected in $\overline{INT0}$.

IT0 1 – $\overline{INT0}$ is edge triggered.
0 – $\overline{INT0}$ is level sensitive.

IE1 1 – Edge detected on $\overline{INT1}$.

IT1 1 – $\overline{INT1}$ is edge triggered.
0 – $\overline{INT1}$ is level sensitive.

These flags are functionally identical to the corresponding 80C51 flags, except that there is only one timer on the 83C751 and the flags are therefore combined into one register.

Note that the positions of the IE0/IT0 and IE1/IT1 bits are transposed from the positions used in the standard 80C51 TCON register.

Timer I is used to control the timing of the I²C bus and also to detect a "bus locked" condition, by causing an interrupt when nothing happens on the I²C bus for an inordinately long period of time while a transmission is in progress. If the interrupt does not occur, the program can attempt to correct the fault and allow the last I²C transmission to be repeated.

The I²C watchdog timer, timer I, is also available as a general-purpose fixed-rate timer when the I²C interface is not being used. A clock rate of 1/12 the oscillator frequency forms the input to the timer. Timer I has a timeout interval of 1024 machine cycles when used as a fixed-rate timer.

I²C Serial Interface

The I²C bus uses two wires (SDA and SCL) to transfer information between devices connected to the bus. The main features of the bus are:

- Bidirectional data transfer between masters and slaves

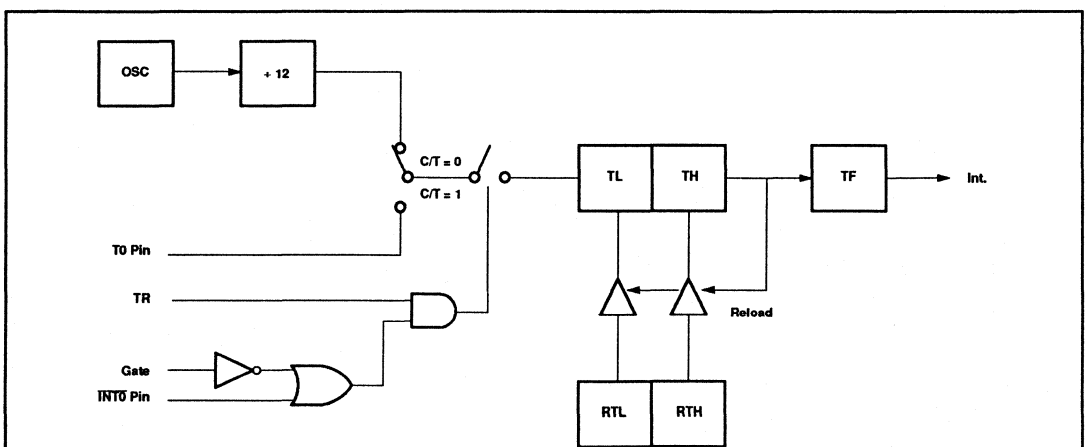


Figure 68. 83C751 Counter/Timer Block Diagram

- Serial addressing of slaves (no added wiring)
- Acknowledgment after each transferred byte
- Multimaster bus
- Arbitration between simultaneously transmitting masters without corruption of serial data on bus

A large family of I²C compatible ICs is available. See the I²C section of this manual for more details on the bus and available ICs.

The 83C751 I²C subsystem includes hardware to simplify the software required to drive the I²C bus. The hardware is a single bit interface which in addition to including the necessary arbitration and framing error checks, includes clock stretching and a bus timeout timer. The interface is synchronized to software either through polled loops or interrupts. Refer to the application note AN422, in Section 4, entitled "Using the 8XC751 Microcontroller as an I²C Bus Master" for additional discussion of the 83C751 I²C interface and sample driver routines.

Six time spans are important in I²C operation and are insured by timer I:

- The MINIMUM HIGH time for SCL when this device is the master.
- The MINIMUM LOW time for SCL when this device is a master. This is not very important for a single-bit hardware interface like this one, because the SCL low time is stretched until the software responds to the I²C flags. The software response time normally meets or exceeds the MIN LO time. In cases where the software responds within MIN HI + MIN LO) time, timer I will ensure that the minimum time is met.
- The MINIMUM SCL HIGH TO SDA HIGH time in a stop condition.
- The MINIMUM SDA HIGH TO SDA LOW time between I²C stop and start conditions (4.7µs, see spec.).
- The MINIMUM SDA LOW TO SCL LOW time in a start condition.
- The MAXIMUM SCL CHANGE time while an I²C frame is in progress. A frame is in progress between a start condition and the following stop condition. This time span serves to detect a lack of software response on this 8XC751 as well as external I²C problems. SCL "stuck low" indicates a faulty master or slave. SCL "stuck high" may mean a faulty device, or that noise induced onto the I²C bus caused all masters to withdraw from I²C arbitration.

The first five of these times are 4.7µs (see I²C specification) and are covered by the low order three bits of timer I. Timer I is clocked by the 8XC751 oscillator, which can vary in frequency from 0.5 to 16MHz. Timer I can be preloaded with one of four values to optimize timing for different oscillator frequencies. At lower frequencies, software response time is increased and will degrade maximum performance of the I²C bus. See special function register I2CFG description for prescale values (CT0, CT1).

The MAXIMUM SCL CHANGE time is important, but its exact span is not critical. The complete 10 bits of timer I are used to count out the maximum time. When I²C operation is enabled, this counter is cleared by transitions on the SCL pin. The timer does not run between I²C frames (i.e., whenever reset or stop occurred more recently than the last start). When this counter is running, it will carry out after 1020 to 1023 machine cycles have elapsed since a change on SCL. A carry out causes a hardware reset of the 83C751 I²C interface and generates an interrupt if the timer I interrupt is enabled. In cases where the bus hangup is due to a lack of software response by this 83C751, the reset releases SCL and allows I²C operation among other devices to continue.

I²C Interrupts

If I²C interrupts are enabled (EA and EI2 are both set to 1), an I²C interrupt will occur whenever the ATN flag is set by a start, stop, arbitration loss, or data ready condition (refer to the description of ATN following). In practice, it is not efficient to operate the I²C interface in this fashion because the I²C interrupt service routine would somehow have to distinguish between hundreds of possible conditions. Also, since I²C can operate at a fairly high rate, the software may execute faster if the code simply waits for the I²C interface.

Typically, the I²C interrupt should only be used to indicate a start condition at an idle slave device, or a stop condition at an idle master device (if it is waiting to use the I²C bus). This is accomplished by enabling the I²C interrupt only during the aforementioned conditions.

I²C Register I2CON

	7	6	5	4	3	2	1	0
Read	RDAT	ATN	DRDY	ARL	STR	STP	MASTER	–
Write	CXA	IDLE	CDR	CARL	CSTR	CSTP	XSTR	XSTP

Reading I2CON

RDAT The data from SDA is captured into "Receive DATa" whenever a rising edge occurs on SCL. RDAT is also available (with seven low-order zeros) in the I2DAT register. The difference between reading it here and there is that reading I2DAT clears DRDY, allowing the I²C to proceed on to another bit. Typically, the first seven bits of a received byte are read from I2DAT, while the 8th is read here. Then I2DAT can be written to send the Ack bit and clear DRDY.

ATN "ATteNtion" is 1 when one or more of DRDY, ARL, STR, or STP is 1. Thus, ATN comprises a single bit that can be tested to release the I²C service routine from a "wait loop."

DRDY "Data ReaDY" (and thus ATN) is set when a rising edge occurs on SCL, except at idle slave. DRDY is cleared by writing CDR = 1, or by writing or reading the I2DAT register. The following low period on SCL is stretched until the program responds by clearing DRDY.

Checking ATN and DRDY

When a program detects ATN = 1, it should next check DRDY. If DRDY = 1, then if it receives the last bit, it should capture the data from RDAT (in I2DAT or I2CON). Next, if the next bit is to be sent, it should be written to I2DAT. One way or another, it should clear DRDY and then return to monitoring ATN. Note that if any of ARL, STR, or STP is set, clearing DRDY will not release SCL to high, so that the I²C will not go on to the next bit. If a program detects ATN = 1, and DRDY = 0, it should go on to examine ARL, STR, and STP.

ARL "Arbitration Loss" is 1 when transmit Active was set, but this 83C751 lost arbitration to another transmitter. Transmit Active is cleared when ARL is 1. There are four separate cases in which ARL is set..

1. If the program sent a 1 or repeated start, but another device sent a 0, or a stop, so that SDA is 0 at the rising edge of SCL. (If the other device sent a stop, the setting of ARL will be followed shortly by STP being set.)
2. If the program sent a 1, but another device sent a repeated start, and it drove SDA low before the 83C751 could drive SCL low. (This type of ARL is always accompanied by STR = 1.)
3. In master mode, if the program sent a repeated start, but another device sent a 1, and it drove SCL low before this 83C751 could drive SDA low.
4. In master mode, if the program sent stop, but it could not be sent because another device sent a 0.

STR "STaRt" is set to a 1 when an I²C start condition is detected at a non-idle slave or at a master. (STR is not set when an idle slave becomes active due to a start bit; the slave has nothing useful to do until the rising edge of SCL sets DRDY.)

STP "SToP" is set to 1 when an I²C stop condition is detected at a non-idle slave or at a master. (STP is not set for a stop condition at an idle slave.)

MASTER "MASTER" is 1 if this 83C751 is currently a master on the I²C. MASTER is set when MASTRQ is 1 and the bus is not busy (i.e., if a start bit hasn't been received since reset or a "Timer I" time-out, or if a stop has been received since the last start). MASTER is cleared when ARL is set, or after the software writes MASTRQ = 0 and then XSTP = 1.

Writing I2CON

Typically, for each bit in an I²C message, a service routine waits for ATN = 1. Based on DRDY, ARL, STR, and STP, and on the current bit position in the message, it may then write I2CON with one or more of the following bits, or it may read or write the I2DAT register.

CXA Writing a 1 to "Clear Xmit Active" clears the Transmit Active state. (Reading the I2DAT register also does this.)

Regarding Transmit Active

Transmit Active is set by writing the I2DAT register, or by writing I2CON with XSTR = 1 or XSTP = 1. The I²C interface will only drive the SDA line low when Transmit Active is set, and the ARL bit will only be set to 1 when Transmit Active is set. Transmit Active is cleared by reading the I2DAT register, or by writing I2CON with CXA = 1. Transmit Active is automatically cleared when ARL is 1.

IDLE Writing 1 to "IDLE" causes a slave's I²C hardware to ignore the I²C until the next start condition (but if MASTRQ is 1, then a stop condition will make the 83C751 into a master).

CDR Writing a 1 to "Clear Data Ready" clears DRDY. (Reading or writing the I2DAT register also does this.)

CARL Writing a 1 to "Clear Arbitration Loss" clears the ARL bit.

CSTR Writing a 1 to "Clear STaRt" clears the STR bit.

CSTP Writing a 1 to "Clear SToP" clears the STP bit. Note that if one or more of DRDY, ARL, STR, or STP is 1, the low time of SCL is stretched until the service routine responds by clearing them.

XSTR Writing 1s to "Xmit repeated STaRt" and CDR tells the I²C hardware to send a repeated start condition. This should only be at a master. Note that XSTR need not and should not be used to send an "initial" (nonrepeated) start; it is sent automatically by the I²C hardware. Writing XSTR = 1 includes the effect of writing I2DAT with XDAT = 1; it sets Transmit Active and releases SDA to high during the SCL low time. After SCL goes high, the I²C hardware waits for the suitable minimum time and then drives SDA low to make the start condition.

XSTP Writing 1s to "Xmit SToP" and CDR tells the I²C hardware to send a stop condition. This should only be done at a master. If there are no more messages to initiate, the service routine should clear the MASTRQ bit in I2CFG to 0 before writing XSTP with 1. Writing XSTP = 1 includes the effect of writing I2DAT with XDAT = 0; it sets Transmit Active and drives SDA low during the SCL low time. After SCL goes high, the I²C hardware waits for the suitable minimum time and then releases SDA to high to make the stop condition.

I²C Register I2DAT

	7	6	5	4	3	2	1	0
Read	RDAT	0	0	0	0	0	0	0
Write	XDAT	X	X	X	X	X	X	X

RDAT "Receive DATa" is captured from SDA every rising edge of SCL. Reading I2DAT also clears DRDY and the Transmit Active state.

XDAT "Xmit Data" sets the data for the next bit. Writing I2DAT also clears DRDY and sets the Transmit Active state.

Regarding Software Response Time

Because the 83C751 can run at 16MHz, and because the I²C interface is optimized for high-speed operation, it is quite likely that an I²C service routine will sometimes respond to DRDY (which is set at a rising edge of SCL) and write I2DAT before SCL has gone low again. If XDAT were applied directly to SDA, this situation would produce an I²C protocol violation. The programmer need not worry about this possibility because XDAT is applied to SDA only when SCL is low.

Conversely, a program that includes an I²C service routine may take a long time to respond to DRDY. Typically, an I²C routine operates on a flag-polling basis during a message, with interrupts from other peripheral functions enabled. If an interrupt occurs, it will delay the response of the I²C service routine. The programmer need not worry about this very much either, because the I²C hardware stretches the SCL low time until the service routine responds. The only constraint on the response is that it must not exceed the Timer I time-out, which is at least 765 microseconds.

I²C Register I2CFG

	7	6	5	4	3	2	1	0
Read	SLAVEN	MASTRQ	0	TIRUN	–	–	CT1	CT0
Write	SLAVEN	MASTRQ	CLRTI	TIRUN	–	–	CT1	CT0

SLAVEN Writing a 1 to "SLAVe ENable" enables the slave functions of the I²C subsystem. If SLAVEN and MASTRQ are 0, the I²C hardware is disabled. This bit is cleared to 0 by reset and by an I²C time-out.

MASTRQ Writing a 1 to "MASTRQ" requests mastership of the I²C. If a frame from another master is in progress when this bit is changed from 0 to 1, action is delayed until a stop condition is detected. Then, or immediately if a frame is not in progress, a start condition is sent and DRDY is set (thus making ATN 1 and generating an I²C interrupt). When a master wishes to release mastership status of the I²C, it writes a 1 to XSTP in I2CON. MASTRQ is cleared by reset and by an I²C time-out.

CLRTI Writing a 1 to this bit clears the Timer I interrupt flag. This bit position always reads as a 0.

TIRUN Writing a 1 to this bit lets Timer I run; a zero stops and clears it. Together with SLAVEN, MASTRQ, and MASTER, this bit determines operational modes as shown in Table 27.

CT1,0 These two bits are programmed as a function of the OSC rate, to optimize the MIN HI and LO time of SCL when this 83C751 is a master on the I²C. The time value determined by these bits controls both of these parameters, and also the timing for stop and start conditions. These bits are cleared to 00 by reset.

Table 27. Interaction of TIRUN with SLAVEN, MASTRQ, and MASTER

SLAVEN, MASTRQ, MASTER	TIRUN	OPERATING MODE
All 0	0	The I²C interface is disabled. Timer I is cleared and does not run. This is the state assumed after a reset. If an I²C application wants to ignore the I²C at certain times, it should write SLAVEN, MASTRQ, and TIRUN all to zero.
All 0	1	The I²C interface is disabled. Timer I operates as a free-running time base. Use this mode only in non-I²C applications.
Any or all 1	0	The I²C interface is enabled. The 3 low-order bits of Timer I run for min-time generation, but the hi-order bits do not, so that there is no checking for I²C being "hung." This configuration can be used for very slow I²C operation.
Any or all 1	1	The I²C interface is enabled. Timer I runs during frames on the I²C, and is cleared by transitions on SCL, and by Start and Stop conditions. This is the normal state for I²C operation.

Values to be used in the CT1 and CT0 bits are shown in Table 28. To allow the I²C bus to run at the maximum rate for a particular oscillator frequency, compare the actual oscillator rate to the f_{OSC} max column in the table. The value for CT1 and CT0 is found in the first line of the table where f_{OSC} max is greater than or equal to the actual frequency.

The table also shows the osc/12 count for various settings of CT1/CT0. This allows calculation of the actual minimum high and low times for SCL as follows:

$$\text{SCL min high/low time (in microseconds)} = \frac{12 * \text{count}}{\text{osc (in MHz)}}$$

For instance, at a 16MHz frequency, with CT1/CT0 set to 10, the minimum SCL high and low times will be 5.25µs

The table also shows the Timer I timeout period (given in machine cycles) for each CT1/CT0 combination. The timeout period varies because of the way in which minimum SCL high and low times are measured. When the I²C interface is operating, Timer I is preloaded at every SCL transition with a value dependent upon CT1/CT0. The preload value is chosen such that a minimum SCL high or low time has elapsed when Timer I reaches a count of 008 (the actual value preloaded into Timer I is 8 minus the osc/12 count).

I²C Register I2STA

Read only

7	6	5	4	3	2	1	0
–	IDLE	XDATA	XACTV	MAKSTR	MAKSTP	XSTR	XSTP
MSB							LSB

This register is read only and reflects the internal status of the I²C hardware. IDLE, XSTR, and XSTP reflect the status of the like named bits in the I2CON register.

XDATA The content of the transmitter buffer.

XACTV Transmitter active.

MAKSTR This bit is high while the hardware is effecting a start condition.

MAKSTP This bit is high while the hardware is effecting a stop condition.

XSTR This bit is active while the hardware is effecting a repeated start condition.

XSTP This bit is active while the hardware is effecting a repeated stop condition.

Interrupts

The interrupt structure is a five-source, one-level interrupt system. Interrupt sources common to the 80C51 are the external interrupts ($\overline{INT0}$, $\overline{INT1}$) and the timer/counter interrupt (ET0). The I²C interrupt (EI2) and Timer I interrupt (ETI) are the other two interrupt sources. The interrupt sources are listed below in their order of polling sequence priority.

Upon interrupt or reset the program counter is loaded with specific values for the appropriate interrupt service routine in program memory. These values are:

Event	Program Memory Address	Priority
Reset	000	Highest
INT0	003	
Counter/Timer 0	00B	
INT1	013	
Timer I	01B	
I²C	023	Lowest

The interrupt enable register (IE) is used to individually enable or disable the five sources. Bit $\overline{EA}$ in the interrupt enable register can be used to globally enable or disable all interrupt sources. The interrupt enable register is described below. All other interrupt details are based on the 80C51 interrupt architecture.

Table 28. CT1, CT0 Values

CT1, CT0	OSC/12 COUNT	f_{OSC} MAX	TIMEOUT PERIOD
10	7	16.8MHz	1023 cycles
01	6	14.25MHz	1022 cycles
00	5	11.7MHz	1021 cycles
11	4	9.14MHz	1020 cycles

Interrupt Enable Register

EA	X	X	EI2	ETI	EX1	ET0	EX0

Symbol	Position	Function
$\overline{EA}$	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each inter rupt source is individu ally enabled or dis abled by setting or clearing its enable bit
–	IE.6	Reserved
–	IE.5	Reserved
EI2	IE.4	Enables or disables the I^2C interrupt. If EI2 = 0, the I^2C interrupt is disabled
ETI	IE.3	Enables or disables the Timer I overflow interrupt. If ETI = 0, the Timer I interrupt is disabled.
EX1	IE.2	Enables or disables external interrupt 1. If EX1 = 0, external interrupt 1 is disabled.
ET0	IE.1	Enables or disables the Timer 0 overflow interrupt. If ET0 = 0, theTimer 0 interrupt is disabled.
EX0	IE.0	Enables or disables external interrupt 0. If EX0 = 0, external interrupt 0 is disabled.

Philips Components

Date of Issue	June 15, 1990
Status	Product Specification
Application Specific Product	

83C751/87C751

CMOS single-chip 8-bit microcontroller

DESCRIPTION

The Philips 83C751/87C751 offers the advantages of the 80C51 architecture in a small package and at low cost.

The 8XC751 Microcontroller is fabricated with Philips high-density CMOS technology. Philips epitaxial substrate minimizes CMOS latch-up sensitivity.

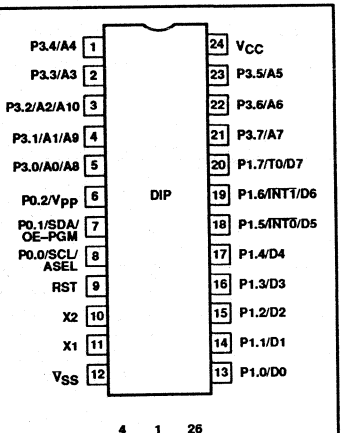
The 8XC751 contains a 2k x 8 ROM (83C751) EPROM (87C751), a 64 x 8 RAM, 19 I/O lines, a 16-bit auto-reload counter/timer, a five-source, fixed-priority level interrupt structure, a bidirectional inter-integrated circuit (I^2C) serial bus interface, and an on-chip oscillator.

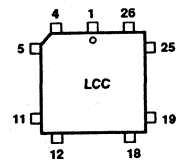
The on-board inter-integrated circuit (I^2C) bus interface allows the 8XC751 to operate as a master or slave device on the I^2C small area network. This capability facilitates I/O and RAM expansion, access to EEPROM, processor-to-processor communication, and efficient interface to a wide variety of dedicated I^2C peripherals.

FEATURES

- 80C51 based architecture
- Inter-Integrated Circuit (I^2C) serial bus interface
- Small package sizes
  - 24-pin DIP (300 mil "skinny DIP")
  - 28-pin PLCC
- 87C751 available in erasable quartz lid or one-time programmable plastic packages
- Wide oscillator frequency range
- Low power consumption:
  - Normal operation: less than 11mA @ 5V, 12MHz
  - Idle mode
  - Power-down mode
- 2k x 8 ROM (83C751)
  2k x 8 EPROM (87C751)
- 64 x 8 RAM
- 16-bit auto reloadable counter/timer
- Fixed-rate timer
- Boolean processor
- CMOS and TTL compatible
- Well suited for logic replacement, consumer and industrial applications

PIN CONFIGURATION



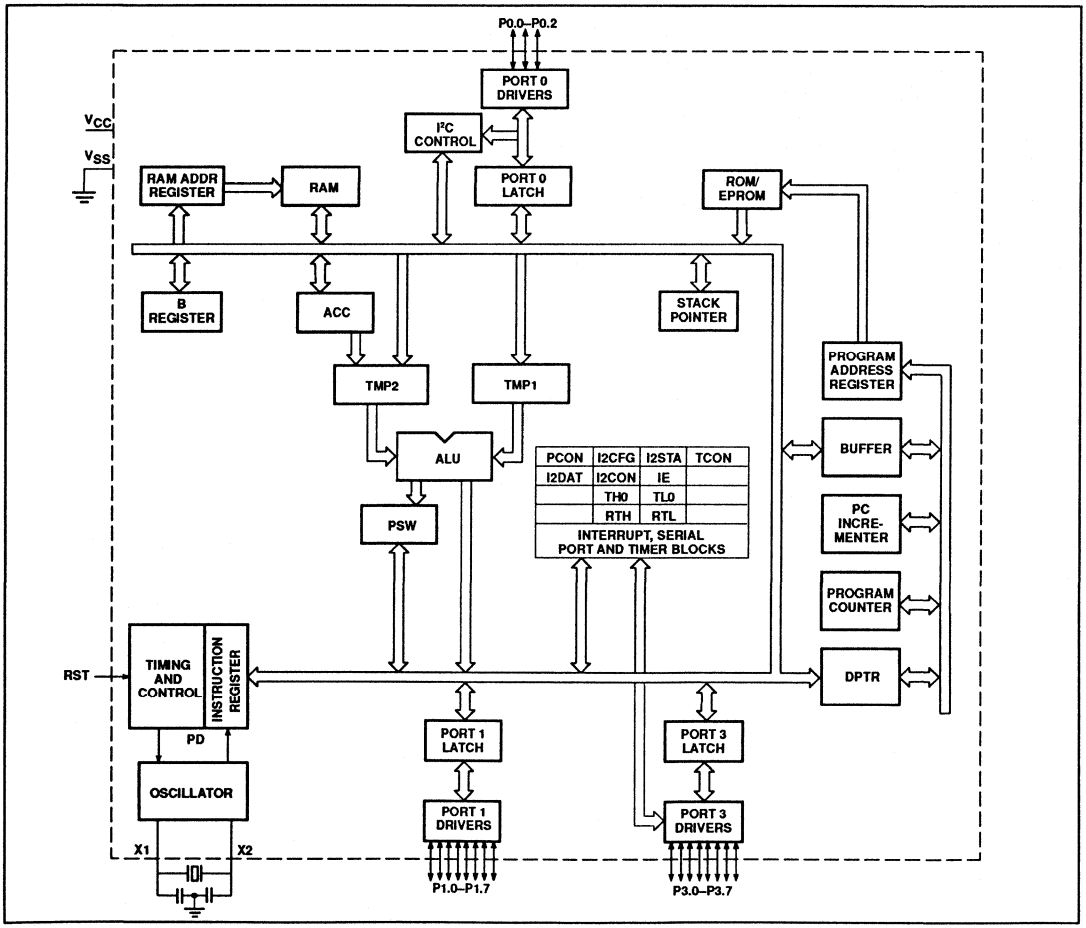


Pin	Function	Pin	Function
1	P3.4/A4	15	P1.0/D0
2	P3.3/A3	16	P1.1/D1
3	P3.2/A2/A10	17	P1.2/D2
4	P3.1/A1/A9	18	P1.3/D3
5	N.C.	19	P1.4/D4
6	P3.0/A0/A8	20	P1.5/INT0/D5
7	P0.2/V_{PP}	21	N.C.
8	P0.1/SDA/OE–PGM	22	N.C.
9	P0.0/SCL/ASEL	23	P1.6/INT1/D6
10	N.C.	24	P1.7/T0/D7
11	RST	25	P3.7/A7
12	X2	26	P3.6/A6
13	X1	27	P3.5/A5
14	V_{SS}	28	V_{CC}

PART NUMBER SELECTION

ROM	EPROM	SPEED	TEMPERATURE AND PACKAGE
	S87C751–1F24	3.5 to 12MHz	0 to +70°C, ceramic DIP
	S87C751–2F24	3.5 to 12MHz	–40 to +85°C, ceramic DIP
	S87C751–4F24	3.5 to 16MHz	0 to +70°C, ceramic DIP
	S87C751–5F24	3.5 to 16MHz	–40 to +85°C, ceramic DIP
S83C751–1N24	S87C751–1N24	3.5 to 12MHz	0 to +70°C, plastic DIP
S83C751–2N24	S87C751–2N24	3.5 to 12MHz	–40 to +85°C, plastic DIP
S83C751–4N24	S87C751–4N24	3.5 to 16MHz	0 to +70°C, plastic DIP
S83C751–5N24	S87C751–5N24	3.5 to 16MHz	–40 to +85°C, plastic DIP
S83C751–1A28	S87C751–1A28	3.5 to 12MHz	0 to +70°C, plastic LCC
S83C751–2A28	S87C751–2A28	3.5 to 12MHz	–40 to +85°C, plastic LCC
S83C751–4A28	S87C751–4A28	3.5 to 16MHz	0 to +70°C, plastic LCC
S83C751–5A28	S87C751–5A28	3.5 to 16MHz	–40 to +85°C, plastic LCC

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	TYPE	NAME AND FUNCTION
V_{SS}	12	14	I	**Circuit Ground Potential**
V_{CC}	24	28	I	**Supply voltage during normal, idle, and power-down operation.**
P0.0–P0.2	8–6	9–7	I/O	**Port 0:** Port 0 is a 3-bit open-drain, bidirectional port. Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs. Port 0 also serves as the serial I^2C interface. When this feature is activated by software, SCL and SDA are driven low in accordance with the I^2C protocol. These pins are driven low if the port register bit is written with a 0 or if the I^2C subsystem presents a 0. The state of the pin can always be read from the port register by the program. To comply with the I^2C specification, P0.0 and P0.1 are open drain bidirectional I/O pins with the electrical characteristics listed in the tables that follow. While these differ from "standard TTL" characteristics, they are close enough for the pins to still be used as general-purpose I/O in non-I^2C applications. Port 0 also provides alternate functions for programming the EPROM memory as follows:
	6	7	N/A	**V_{PP} (P0.2)** – Programming voltage input.
	7	8	I	**OE/PGM (P0.1)** – Input which specifies verify mode (output enable) or the program mode. OE/PGM = 1 output enabled (verify mode). OE/PGM = 0 program mode.
	8	9	I	**ASEL (P0.0)** – Input which indicates which bits of the EPROM address are applied to port 3. ASEL = 0 low address byte available on port 3. ASEL = 1 high address byte available on port 3 (only the three least significant bits are used).
	7	8	I/O	**SDA (P0.1)** – I^2C data.
	8	9	I/O	**SCL (P0.0)** – I^2C clock.
P1.0–P1.7	13–20	15–20, 23, 24	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 1 serves to output the addressed EPROM contents in the verify mode and accepts as inputs the value to program into the selected address during the program mode. Port 1 also serves the special function features of the 80C51 family as listed below:
	18	20	I	**$\overline{INT0}$ (P1.5):** External interrupt.
	19	23	I	**$\overline{INT1}$ (P1.6):** External interrupt.
	20	24	I	**T0 (P1.7):** Timer 0 external input.
P3.0–P3.7	5–1, 23–21	4–1, 6, 27–25	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also functions as the address input for the EPROM memory location to be programmed (or verified). The 11-bit address is multiplexed into this port as specified by P0.0/ASEL.
RST	9	11	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on RESET using only an external capacitor to V_{CC}. After the device is reset, a 10-bit serial sequence, sent LSB first, applied to RESET, places the device in the programming state allowing programming address, data and V_{PP} to be applied for programming or verification purposes. The RESET serial sequence must be synchronized with the X1 input.
X1	11	13	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits. X1 also serves as the clock to strobe in a serial bit stream into RESET to place the device in the programming state.
X2	10	12	O	**Crystal 2:** Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

X1 and X2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator.

To drive the device from an external clock source, X1 should be driven while X2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-up reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-up, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

IDLE MODE

In idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. the control bits for the reduced power modes are in the special function register PCON.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	Port 0	Port 1	Port 2
Idle	Data	Data	Data
Power-down	Data	Data	Data

DIFFERENCES BETWEEN THE 8XC751 AND THE 80C51

Program Memory

On the 8XC751, program memory is 2048 bytes long and is not externally expandable, so the 80C51 instructions MOVX, LJMP, and LCALL are not implemented. The only fixed locations in program memory are the addresses at which execution is taken up in response to reset and interrupts, which are as follows:

Event	Program Memory Address
Reset	000
External $\overline{INT0}$	003
Counter/timer 0	00B
External $\overline{INT1}$	013
Timer I	01B
I^2C serial	023

Counter/Timer Subsystem

The 8XC751 has one counter/timer called timer/counter 0. Its operation is similar to mode 2 operation on the 80C51, but is extended to 16 bits with 16 bits of autoload. The controls for this counter are centralized in a single register called TCON.

A watchdog timer, called Timer I, is for use with the I^2C subsystem. In I^2C applications, this timer is dedicated to time-generation and bus monitoring of the I^2C. In non-I^2C applications, it is available for use as a fixed time-base.

Interrupt Subsystem – Fixed Priority

The IP register and the 2-level interrupt system of the 80C51 are eliminated. Simultaneous interrupt conditions are resolved by a single-level, fixed priority as follows:

Highest priority:	Pin $\overline{INT0}$
	Counter/timer flag 0
	Pin $\overline{INT1}$
	Timer I
Lowest priority:	Serial I^2C

Serial Communications

The 8XC751 contains an I^2C serial communications port instead of the 80C51 UART. The I^2C serial port is a single bit hardware interface with all of the hardware necessary to support multimaster and slave operations. Also included are receiver digital filters and timer (timer I) for communication watch-dog purposes. The I^2C serial port is controlled through four special function registers; I^2C control, I^2C data, I^2C status, and I^2C configuration.

Special Function Register Addresses

Special function registers for the 8XC751 are identical to those of the 80C51, except for the changes listed below:

80C51 special function registers not present in the 8XC751 are TMOD (89), P2 (A0) and IP (B8). The 80C51 registers TH1, TL1, SCON, and SBUF are replaced with the 8XC751 registers RTH, RTL, I2CON, and I2DAT, respectively. Additional special function registers are I2CFG (D8) and I2STA (F8). See Table 2.

Table 2. I^2C Special Function Register Addresses

REGISTER ADDRESS			BIT ADDRESS							
NAME	SYMBOL	ADDRESS	MSB							LSB
I^2C control	I2CON	98	9F	9E	9D	9C	9B	9A	99	98
I^2C data	I2DAT	99	–	–	–	–	–	–	–	–
I^2C configuration	I2CFG	D8	DF	DE	DD	DC	DB	DA	D9	D8
I^2C status	I2STA	F8	FF	FE	FD	FC	FB	FA	F9	F8

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Voltage from V_{CC} to V_{SS}	–0.5 to +6.5	V
Voltage from any pin to V_{SS} (except V_{PP})	–0.5 to V_{CC} + 0.5	V
Power dissipation	1.0	W
Voltage on V_{PP} pin to V_{SS}	0 to +13.0	V

DC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10% for 87C751, V_{CC} = 5V ±20% for 83C751, V_{SS} = 0V[3]

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{IL} V_{IH} V_{IH1}	Input low voltage, except SDA, SCL Input high voltage, except X1, RST Input high voltage, X1, RST		–0.5 0.2V_{CC}+0.9 0.7V_{CC}	0.2V_{DD}–0.1 V_{CC}+0.5 V_{CC}+0.5	V V V
 V_{IL1} V_{IH2}	SDA, SCL: Input low voltage Input high voltage		 –0.5 0.7V_{CC}	 0.3V_{CC} V_{CC}+0.5	 V V
V_{OL} V_{OL1}	Output low voltage, ports 1 and 3 Output low voltage, port 0.2	I_{OL} = 1.6mA I_{OL} = 3.2mA		0.45 0.45	V V
V_{OH}	Output high voltage, ports 1 and 3	I_{OH} = –60µA, I_{OH} = –25µA I_{OH} = –10µA	2.4 0.75V_{CC} 0.9V_{CC}		V V V
V_{OL2} C	Port 0.0 and 0.1 (I^2C) – Drivers Output low voltage Driver, receiver combined: Capacitance	I_{OL} = 3mA (over V_{CC} range)		0.4 10	V pF
I_{IL} I_{TL} I_{LI}	Logical 0 input current, ports 1 and 3 Logical 1 to 0 transition current, ports 1 and 3[8] Input leakage current, port 0	V_{IN} = 0.45V V_{IN} = 2V 0.45 < V_{IN} < V_{CC}		–50 –650 ±10	µA µA µA
R_{RST}	Internal pull-down resistor		25	175	kohm
C_{IO}	Pin capacitance	Test freq = 1MHz, T_A = 25°C		10	pF
I_{PD}	Power-down current[4]	V_{CC} = 2 to V_{CC} max		50	µA
V_{PP}	V_{PP} program voltage (for 87C751 only)	V_{SS} = 0V V_{CC} = 5V±10% T_A = 21°C to 27°C	12.5	13.0	V
I_{PP}	Program current (for 87C751 only)	V_{PP} = 13.0V		10	mA
I_{CC}	Supply current (see Figure 3)				

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.
4. Power-down I_{CC} is measured with all output pins disconnected; port 0 = V_{CC}; X2, X1 n.c.; RST = V_{SS}.
5. Active I_{CC} is measured with all output pins disconnected; X1 driven with t_{CLCH}, t_{CHCL} = 5ns, V_{IL} = V_{SS} + 0.5V, V_{IH} = V_{CC} – 0.5V; X2 n.c.; RST = port 0 = V_{CC}. I_{CC} will be slightly higher if a crystal oscillator is used.
6. Idle I_{CC} is measured with all output pins disconnected; X1 driven with t_{CLCH}, t_{CHCL} = 5ns, V_{IL} = V_{SS} + 0.5V, V_{IH} = V_{CC} – 0.5V; X2 n.c.; port 0 = V_{CC}; RST = V_{SS}.
7. Load capacitance for ports = 80pF.
8. Pins of ports 1 and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.

AC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or −40°C to +85°C, V_{CC} = 5V ±10% FOR 87C751, V_{CC} = 5V ±20% FOR 83C751, V_{SS} = 0V[3, 7]

SYMBOL	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
		MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	Oscillator frequency:			3.5 3.5 0.5	12 16 12	MHz MHz MHz
External Clock (Figure 1)						
t_{CHCX}	High time	20		20		ns
t_{CLCX}	Low time	20		20		ns
t_{CLCH}	Rise time		20		20	ns
t_{CHCL}	Fall time		20		20	ns

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

C – Clock
D – Input data
H – Logic level high
L – Logic level low
Q – Output data
T – Time
V – Valid
X – No longer a valid logic level
Z – Float

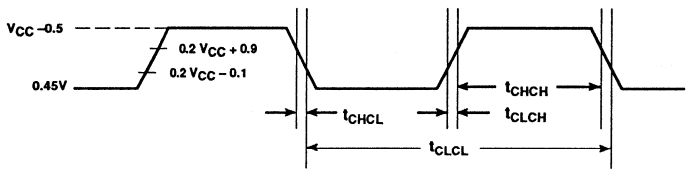


Figure 1. External Clock Drive

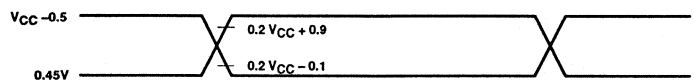


Figure 2. AC Testing Input/Output

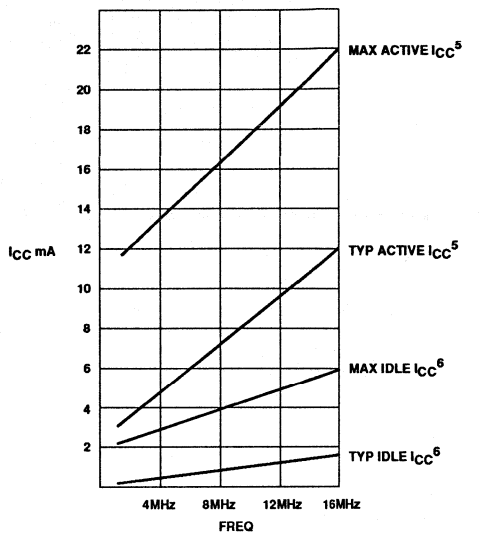


Figure 3. I_{CC} vs. FREQ
Maximum I_{CC} values taken at V_{CC} max and worst case temperature.
Typical I_{CC} values taken at V_{CC} = 5.0V and 25°C.
Notes 5 and 6 refer to AC Electrcial Characteristics.

PROGRAMMING CONSIDERATIONS

EPROM Characteristics

The 87C751 is programmed by using a modified Quick-Pulse Programming algorithm similar to that used for devices such as the 87C451 and 87C51. It differs from these devices in that a serial data stream is used to place the 87C751 in the programming mode.

Figure 4 shows a block diagram of the programming configuration for the 87C751. Port pin P0.2 is used as the programming voltage supply input (V_{PP} signal). Port pin P0.1 is used as the program (PGM/) signal. This pin is used for the 25 programming pulses.

Port 3 is used as the address input for the byte to be programmed and accepts both the high and low components of the eleven bit address. Multiplexing of these address components is performed using the ASEL input. The user should drive the ASEL input high and then drive port 3 with the high order bits of the address. ASEL should remain high for at least 13 clock cycles. ASEL may then be driven low which latches the high order bits of the address internally. the high address should remain on port 3 for at least two clock cycles after ASEL is driven low. Port 3 may then be driven with the low byte of the address. The low address will be internally stable 13 clock cycles later. The address will remain stable provided that the low byte placed on port 3 is held stable and ASEL is kept low. **Note:** ASEL needs to be pulsed high only to change the high byte of the address.

Port 1 is used as a bidirectional data bus during programming and verify operations. During programming mode, it accepts the byte to be programmed. During verify mode, it provides the contents of the EPROM location specified by the address which has been supplied to Port 3.

The XTAL1 pin is the oscillator input and receives the master system clock. This clock should be between 1.2 and 6MHz.

The RESET pin is used to accept the serial data stream that places the 87C751 into various programming modes. This pattern consists of a 10-bit code with the LSB sent first. Each bit is synchronized to the clock input, X1.

Programming Operation

Figures 5 and 6 show the timing diagrams for the program/verify cycle. RESET should initially be held high for at least two machine cycles. P0.1 (PGM/) and P0.2 (V_{PP}) will be at V_{OH} as a result of the RESET operation. At this point, these pins function as normal quasi-bidirectional I/O ports and the programming equipment may pull these lines low. However, prior to sending the 10-bit code on the RESET pin, the programming equipment should drive these pins high (V_{IH}). The RESET pin may now be used as the serial data input for the data stream which places the 87C751 in the programming mode. Data bits are sampled during the clock high time and thus should only change during the time that the clock is low. Following transmission of the last data bit, the RESET pin should be held low.

Next the address information for the location to be programmed is placed on port 3 and ASEL is used to perform the address multiplexing, as previously described. At this time, port 1 functions as an output.

A high voltage V_{PP} level is then applied to the V_{PP} input (P0.2). (This sets Port 1 as an input port). The data to be programmed into the EPROM array is then placed on Port 1. This is followed by a series of programming pulses applied to the PGM/ pin (P0.1). These pulses are created by driving P0.1 low and then high. This pulse is repeated until a total of 25 programming pulses have occurred. At the conclusion of the last pulse, the PGM/ signal should remain high.

The V_{PP} signal may now be driven to the V_{OH} level, placing the 87C751 in the verify mode. (Port 1 is now used as an output port). After four machine cycles (48 clock periods), the contents of the addressed location in the EPROM array will appear on Port 1.

The next programming cycle may now be initiated by placing the address information at the inputs of the multiplexed buffers, driving the V_{PP} pin to the V_{PP} voltage level, providing the byte to be programmed to Port1 and issuing the 26 programming pulses on the PGM/ pin, bringing V_{PP} back down to the V_C level and verifying the byte.

Programming Modes

The 87C751 has four programming features incorporated within its EPROM array. These include the USER EPROM for storage of the application's code, a 16-byte encryption key array and two security bits. Programming and verification of these four elements are selected by a combination of the serial data stream applied to the RESET pin and the voltage levels applied to port pins P0.1 and P0.2. The various combinations are shown in Table 3.

Encryption Key Table

The 87C751 includes a 16-byte EPROM array that is programmable by the end user. The contents of this array can then be used to encrypt the program memory contents during a program memory verify operation. When a program memory verify operation is performed, the contents of the program memory location is XNOR'ed with one of the bytes in the 16-byte encryption table. The resulting data pattern is then provided to port 1 as the verify data. The encryption mechanism can be disable, in essence, by leaving the bytes in the encryption table in their erased state (FFH) since the XNOR product of a bit with a logical one will result in the original bit. The encryption bytes are mapped with the code memory in 16-byte groups. the first byte in code memory will be encrypted with the first byte in the encryption table; the second byte in code memory will be encrypted with the second byte in the encryption table and so forth up to and including the 16the byte. The encryption repeats in 16-byte groups; the 17th byte in the code memory will be encrypted with the first byte in the encryption table, and so forth.

Security Bits

Two security bits, security bit 1 and security bit 2, are provided to limit access to the USER EPROM and encryption key arrays. Security bit 1 is the program inhibit bit, and once programmed performs the following functions:

1. Additional programming of the USER EPROM is inhibited.
2. Additional programming of the encryption key is inhibited.
3. Verification of the encryption key is inhibited.
4. Verification of the USER EPROM and the security bit levels may still be performed.

(If the encryption key array is being used, this security bit should be programmed by the user to prevent unauthorized parties from reprogramming the encryption key to all logical zero bits. Such programming would provide data during a verify cycle that is the logical complement of the USER EPROM contents).

Security bit 2, the verify inhibit bit, prevents verification of both the USER EPROM array and the encryption key arrays. The security bit levels may still be verified.

Programming and Verifying Security Bits

Security bits are programmed employing the same techniques used to program the USER EPROM and KEY arrays using serial data streams and logic levels on port pins indicated in Table 3. When programming either security bit, it is not necessary to provide address or data information to the 87C751 on ports 1 and 3.

Verification occurs in a similar manner using the RESET serial stream shown in Table 3.

Port 3 is not required to be driven and the results of the verify operation will appear on ports 1.6 and 1.7.

Ports 1.7 contains the security bit 1 data and is a logical one if programmed and a logical zero if erased. Likewise, P1.6 contains the security bit 2 data and is a logical one if programmed and a logical zero if erased.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Flourless part number 2345–5 or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-s/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000µW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

Table 3. Implementing Program/Verify Modes

OPERATION	SERIAL CODE	P0.1 (PGM/)	P0.2 (V_{PP})
Program user EPROM	296H	–*	V_{PP}
Verify user EPROM	296H	V_{IH}	V_{IH}
Program key EPROM	292H	–*	V_{PP}
Verify key EPROM	292H	V_{IH}	V_{IH}
Program security bit 1	29AH	–*	V_{PP}
Program security bit 2	298H	–*	V_{PP}
Verify security bits	29AH	V_{IH}	V_{IH}

NOTE:
*Pulsed from V_{IH} to V_{IL} and returned to V_{IH}.

EPROM PROGRAMMING AND VERIFICATION

T_A = 21°C to +27°C, V_{CC} = 5V ±10%, V_{SS} = 0V

SYMBOL	PARAMETER	MIN	MAX	UNIT
$1/t_{CLCL}$	Oscillator/clock frequency	1.2	6	MHz
t_{AVGL}*	Address setup to P0.1 (PROG–) low	10µs + 24t_{CLCL}		
t_{GHAX}	Address hold after P0.1 (PROG–) high	48t_{CLCL}		
t_{DVGL}	Data setup to P0.1 (PROG–) low	38t_{CLCL}		
t_{DVGL}	Data setup to P0.1 (PROG–) low	38t_{CLCL}		
t_{GHDX}	Data hold after P0.1 (PROG–) high	36t_{CLCL}		
t_{SHGL}	V_{PP} setup to P0.1 (PROG–) low	10		µs
t_{GHSL}	V_{PP} hold after P0.1 (PROG–)	10		µs
t_{GLGH}	P0.1 (PROG–) width	90	110	µs
t_{AVQV}**	V_{PP} low (V_{CC}) to data valid		48t_{CLCL}	
t_{GHGL}	P0.1 (PROG–) high to P0.1 (PROG–) low	10		µs
t_{SYNL}	P0.0 (sync pulse) low	4t_{CLCL}		
t_{SYNH}	P0.0 (sync pulse) high	8t_{CLCL}		
t_{MASEL}	ASEL high time	13t_{CLCL}		
t_{MAHLD}	Address hold time	2t_{CLCL}		
t_{HASET}	Address setup to ASEL	13t_{CLCL}		
t_{ADSTA}	Low address to address stable	13t_{CLCL}		

NOTES:
*Address should be valid at least 24t_{CLCL} before the rising edge of P0.2 (V_{PP}).
**For a pure verify mode, i.e., no program mode in between, t_{AVQV} is 14t_{CLCL} maximum.

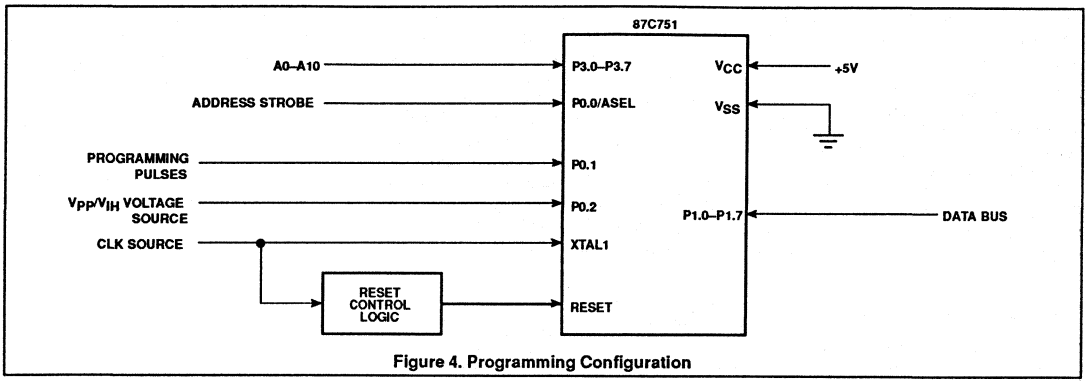


Figure 4. Programming Configuration

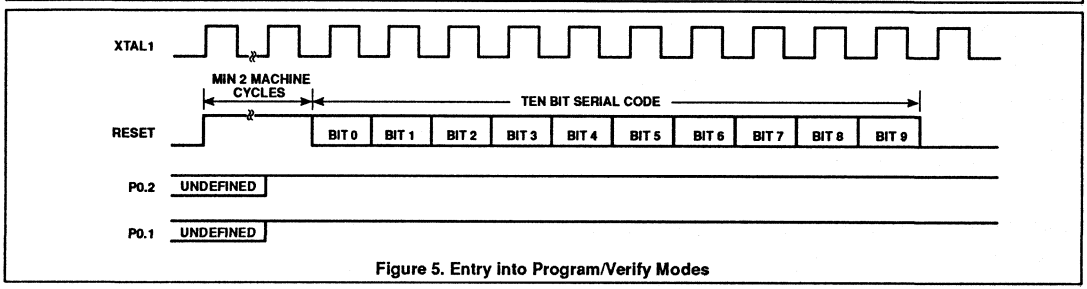


Figure 5. Entry into Program/Verify Modes

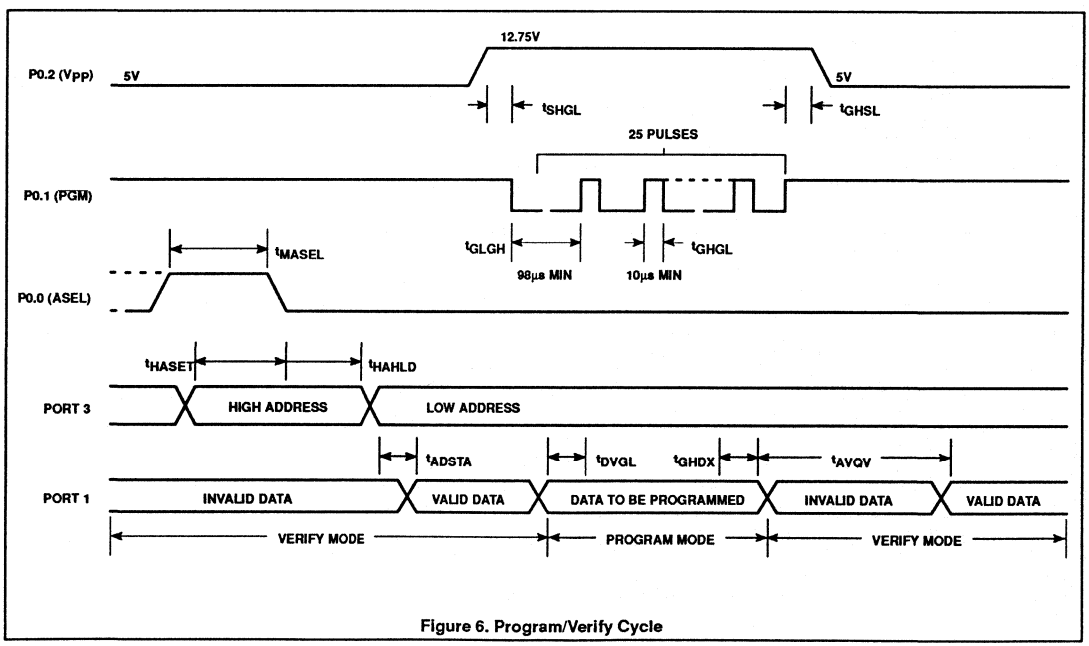


Figure 6. Program/Verify Cycle

8XC752 OVERVIEW

The Signetics 83C752/87C752 is a single-chip control oriented microcontroller fabricated with Signetics high-density CMOS technology minimizing CMOS latch-up sensitivity. Being a member of the 80C51 family, the 83C752 has a powerful instruction set, and has the same basic architecture as the 80C51. The 83C752 is essentially the popular industry-standard 83C751 with the inclusion of a five-channel multiplexed 8-bit ADC and a PWM output.

The 83C752 contains a 2k x 8 masked ROM, 64 bytes of RAM, 21 I/O lines, a 16-bit auto-reload timer/counter, a fixed-rate timer, a seven-source fixed-priority interrupt structure, a bidirectional Inter-Integrated Circuit (I^2C) serial bus interface, and an on-chip oscillator. This device also includes a five-channel multiplexed 8-bit ADC and a PWM output.

The on-board I^2C bus interface allows the 83C752 to operate as a master or slave device on the I^2C small area network. This capability facilitates I/O and RAM expansion, access to EEPROM, processor-to-processor communications, and efficient interface to a wide variety of dedicated I^2C peripherals.

The EPROM version of this device, the 87C752, is also available in both quartz-lid erasable and plastic one-time programmable (OTP) packages. Once the array has been programmed, it is functionally equivalent to the masked ROM 83C752. Thus, unless explicitly stated otherwise, all references made to the 83C752 apply equally to the 87C752.

The 83C752 supports two power reduction modes of operation referred to as the idle mode and the power-down mode.

Differences from the 80C51

Instruction Set

The instruction set of the 83C752 is identical to the 80C51 except that:

MOVX, LCALL, and LJMP are not implemented.

If these instructions are executed, the appropriate number of instruction cycles will take place along with external fetches; however, no operation will take place. The LJMP may not respond to all program address bits.

Memory Organization

The 83C752 manipulates operands in three memory address spaces. The first is the program memory space which contains program instructions as well as constants such as look-up tables. The program memory space contains 2k bytes in the 83C752.

The second memory space is the data memory array which has a logical address space of 128 bytes. However, only the first 64 (0 to 3FH) are implemented in the 83C752.

The third memory space is the special function register array having a 128-byte address space (80H to FFH). Only selected locations in this memory space are used (see Table 29). Note that the architecture of these memory spaces (internal program memory, internal data memory, and special function registers) is identical to the 80C51, and the 83C752 varies only in the amount of memory physically implemented.

The 83C752 does not directly address any external data or program memory spaces. For this reason, the MOVX instructions in the 80C51 instruction set are not implemented in the 83C752, nor are the alternate I/O pin functions $\overline{RD}$ and $\overline{WR}$.

I/O Ports

The I/O pins provided by the 83C752 consist of port 0, port 1, and port 3.

Port 0

Port 0 is a 5-bit bidirectional I/O port and includes alternate functions on some pins of this port. Pins P0.3 and P0.4 are provided with internal pullups while the remaining pins (P0.0, P0.1, and P0.2) have open drain output structures. The alternate functions for port 0 are:

P0.0 SCL – the I^2C bus clock
P0.1 SDA – the I^2C bus data
P0.4 PWM – the PWM output

If the alternate functions, I^2C and PWM, are not being used, then these pins may be used as I/O ports.

Port 1

Port 1 is an 8-bit bidirectional I/O port whose structure is identical to the 80C51, but also includes alternate input functions on all pins. The alternate pin functions for port 1 are:

P1.0-P1.4 - ADC0-ADC4 - A/D converter analog inputs
P1.5 $\overline{INT0}$ - external interrupt 0 input
P1.6 $\overline{INT1}$ - external interrupt 1 input
P1.7 - T0 - timer 0 external input

If the alternate functions $\overline{INT0}$, $\overline{INT1}$, or T0 are not being used, these pins may be used as standard I/O ports, provided that the A/D is disabled. It is necessary to connect AV_{CC} and AV_{SS} to V_{CC} and V_{SS}, respectively, in order to use these pins as standard I/O pins. When the A/D converter is enabled, the analog channel connected to the A/D may not be used as a digital input; however, the remaining analog inputs may be used as digital inputs. They may not be used as digital outputs. While the A/D is enabled, the analog inputs are floating.

Port 3

Port 3 is an 8-bit bidirectional I/O port whose structure is identical to the 80C51. Note that the alternate functions associated with port 3 of the 80C51 have been moved to port 1 of the 83C752 (as applicable). See Figure 69 for port bit configurations.

PWM Outputs

The single PWM output is an alternate function assigned to P0.4 and can be used to output pulses of programmable length and interval. The repetition frequency is defined by an 8-bit prescaler which generates the clock for the counter. This prescaler is contained in the PWMP register.

The 8-bit counter counts from 0 to 254 inclusive. The value of the 8-bit counter is compared to the contents of the compare register, PWM. When the content's value matches that of the PWCM register, the PWCM output is set high. When the counter reaches zero, the PWM output is set low. The pulse width ratio (duty cycle) is defined by the contents of the compare register and is in the range of 0 to 1, programmed in increments of 1/255.

Table 29. 8XC752 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
ADAT#	A/D result	84H									00H
ADCON#	A/D control	A0H	–	–	ENADC	ADCI	ADCS	AADR2	AADR1	AADR0	C0H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR:	Data pointer (2 bytes)										
DPL	Data pointer low	82H									00H
DPH	Data pointer high	83H									00H
			DF	DE	DD	DC	DB	DA	D9	D8	
I²CFG*#	I²C configuration	D8H/RD	SLAVEN	MASTRQ	0	TIRUN	–	–	CT1	CT0	0000xx00B
		WR	SLAVEN	MASTRQ	CLRTI	TIRUN	–	–	CT1	CT0	
			9F	9E	9D	9C	9B	9A	99	98	
I²CON*#	I²C control	98H/RD	RDAT	ATN	DRDY	ARL	STR	STP	MASTER	–	81H
		WR	CXA	IDLE	CDR	CARL	CSTR	CSTP	XSTR	XSTP	
I²DAT*#	I²C data	99H/RD	RDAT	0	0	0	0	0	0	0	80H
		WR	XDAT	X	X	X	X	X	X	X	
			FF	FE	FD	FC	FB	FA	F9	F8	
I²STA*#	I²C status	F8H	–	IDLE	XDATA	XACTV	MAKSTR	MAKSTP	XSTR	XSTP	x0100000B
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*#	Interrupt enable	A8H	EA	EAD	ETI	ES	EPWM	EX1	ET0	EX0	00H
P0*#	Port 0	80H	–	–	–	84	83	82	81	80	xxx11111B
P1*#	Port 1	90H	97	96	95	94	93	92	91	90	FFH
P3*	Port 3	B0H	B7	B6	B5	B4	B3	B2	B1	B0	FFH
PCON#	Power control	87H	–	–	–	–	–	–	PD	IDL	xxxx0000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
PWCM#	PWM compare	8EH									xxxxxxxxB
PWENA#	PWM enable	FEH	–	–	–	–	–	–	–	PWE	FEH
PWMP#	PWM prescaler	8FH									00H
RTL#	Timer low reload	8BH									00H
RTH#	Timer high reload	8DH									00H
SP	Stack pointer	81H									07H
TL#	Timer low	8AH									00H
			8F	8E	8D	8C	8B	8A	89	88	00H
TCON*#	Timer control	88H	GATE	C/T	TF	TR	IE0	IT0	IE1	IT1	00H

*SFRs are bit addressable.
#SFRs are modified from or added to the 80C51 SFRs.

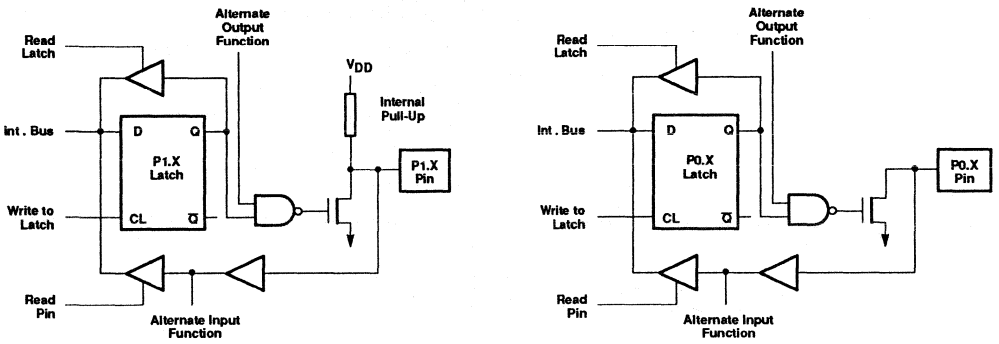


Figure 69. Port Bit Latches and I/O Buffers

The PWM output can be set continuously high by loading the compare register with 00H and continuously low by loading the compare register with FFH. The PWM output is enabled by setting the PWE bit in the PWM enable register, PWENA. When enabled, the output is driven with a fully active strong pullup. When disabled, the pin behaves as a normal bidirectional I/O pin. When disabled, the counter remains active. The PWM function is disabled by a reset condition. The PWM output is high during power-down and idle modes and the counter is disabled.

The repetition frequency is given by:

$$f_{PWM} = \frac{f_{osc}}{510 \times (1 + PWMP)}$$

An oscillator frequency of 12MHz results in a repetition range of 92Hz to 23.5kHz.

The low/high ratio of the PWM output is PWM/(255 – PWM) for PWM values except 255. A PWM value of 255 results in a low PWM output.

If enabled, a PWM interrupt will occur when the PWM counter overflows.

In order for the PWM output to be used as a standard I/O pin, the PWM function needs to be disabled. The PWM counter can still be used as an internal timer by enabling the PWM interrupt.

A/D Converter

The 83C752 contains a five-channel multiplexed 8-bit A/D converter. The conversion requires 40 machine cycles (40µs at 12MHz oscillator frequency).

The A/D converter is controlled by the A/D control register, ADCON. Input channels are selected by the analog multiplexer by bits ADCON.0 through ADCON.2. The ADCON register is not bit addressable.

ADCON Register

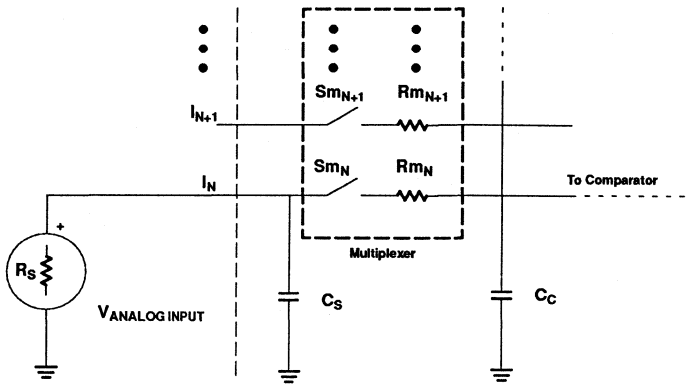
MSB							LSB
X	X	ENADC	ADCI	ADCS	AADR2	AADR1	AADR0

ADCI	ADCS	Operation
0	0	ADC not busy, a conversion can be started.
0	1	ADC busy, start of a new conversion is blocked.
1	0	Conversion completed, start of a new conversion is blocked.
1	1	Not possible.

INPUT CHANNEL SELECTION			
ADDR2	ADDR1	ADDR0	INPUT PIN
0	0	0	P1.0
0	0	1	P1.1
0	1	0	P1.2
0	1	1	P1.3
1	0	0	P1.4

Position	Symbol	Function
ADCON.5	ENADC	Enable A/D function when ENADC = 1. Reset forces ENADC = 0.
ADCON.4	ADCI	ADC interrupt flag. This flag is set when an ADC conversion is complete. If IE.6 = 1, an interrupt is requested when ADCI = 1. The ADCI flag is cleared when conversion data is read. This flag is read only.
ADCON.3	ADCS	ADC start and status. Setting this bit starts an A/D conversion. Once set, ADCS remains high throughout the conversion cycle. On completion of the conversion, it is reset at the same time the ADCI interrupt flag is set. ADCS cannot be reset by software.
ADCON.2	AADR2	Analog input select.
ADCON.1	AADR1	Analog input select.
ADCON.0	AADR0	Analog input select. This binary coded address selects one of the five analog input port pins of P1 to be input to the converter. It can only be changed when ADCI and ADCS are both low. AADR2 is the most significant bit.

The completion of the 8-bit ADC conversion is flagged by ADCI in the ADCON register, and the result is stored in the special function register ADAT.



Rm = 0.5 - 3 kohms
CS + CC = 15pF maximum
RS = Recommended < 9.6 kohms for 1 LSB @ 12MHz

NOTE:
Because the analog to digital converter has a sampled-data comparator, the input looks capacitive to a source. When a conversion is initiated, switch Sm closes for 8tcy (8µs @ 12MHz crystal frequency) during which time capacitance Cs + Cc is charged. It should be noted that the sampling causes the analog input to present a varying load to an analog source.

Figure 70. A/D Input: Equivalent Circuit

An ADC conversion in progress is unaffected by an ADC start. The result of a completed conversion remains unaffected provided ADCI remains at a logic 1. While ADCS is a logic 1 or ADCI is a logic 1, a new ADC START will be blocked and consequently lost. An ADC conversion in progress is aborted when the idle or power-down mode is entered. The result of a completed conversion (ADCI = logic 1) remains unaffected when entering the idle mode. See Figure 70 for an A/D input equivalent circuit.

The analog input pins ADC0-ADC4 may be used as digital inputs and outputs when the A/D converter is disabled by a 0 in the EN-ADC bit in ADCON. When the A/D is enabled, the analog input channel that is selected by the ADDR2-ADDR0 bits in ADCON cannot be used as a digital input. Reading the selected A/D channel as a digital input will always return a 1. The unselected A/D inputs may always be used as digital inputs. Unselected analog inputs will be floating and may not be used as digital outputs.

Counter/Timer

The 8XC752 counter/timer is designated Timer 0 and is separate from Timer I of the I²C serial port and from the PWM. Its operation is similar to mode 2 of the 80C51 counter/timer, extended to 16 bits. When Timer 0 is used in the external counter mode, the T0 input (P1.7) is sampled every S4P1. The counter/timer function is controlled using the timer control register (TCON).

TCON Register

MSB							LSB
GATE	C/T	TF	TR	IE0	IT0	IE1	IT1

Position	Symbol	Function
TCON.7	GATE	1 – Timer 0 is enabled only when INT0 pin is high and TR is 1. 0 – Timer 0 is enabled only when TR is 1.
TCON.6	C/T	1 – Counter operation from T0 pin 0 – Timer operation from internal clock
TCON.5	TF	1 – Set on overflow of T0. 0 – Cleared when processor vectors to interrupt routine and by reset.
TCON.4	TR	1 – Enable timer 0 0 – Disable timer 0
TCON.3	IE	0 – Edge detected on INT0
TCON.2	IT0	1 – INT0 is edge triggered. 0 – INT0 is level sensitive.
TCON.1	IE1	1 – Edge detected on INT1
TCON.0	IT1	1 – INT1 is edge triggered. 0 – INT1 is level sensitive.

These flags are functionally identical to the corresponding 80C51 flags except that there is only one of the 80C51 style timers, and the flags are combined into one register.

Note that the positions of the IE0/IT0 and IE1/IT1 bits are transposed from the positions used in the standard 80C51 TCON register.

A communications watchdog timer, Timer I, is described in the I²C section. In I²C applications, this timer is dedicated to time generation and bus monitoring for the I²C. In non-I²C applications, it is available for use as a fixed time base.

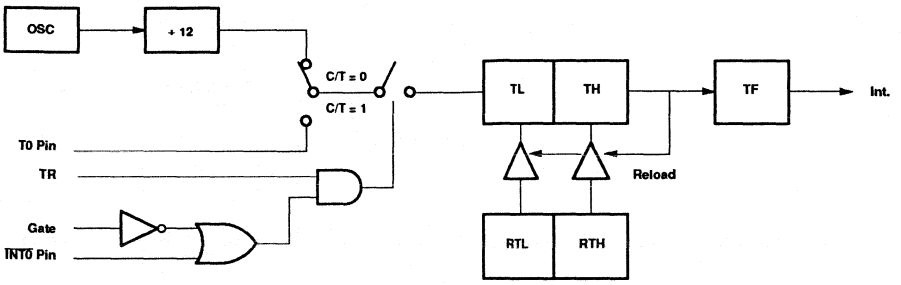


Figure 71. 83C752 Counter/Timer Block Diagram

The 16-bit timer/counter's operation is similar to mode 2 operation on the 80C51, but is extended to 16 bits. The timer/counter is clocked by either 1/12 the oscillator frequency or by transitions on the T0 pin. The C/T pin in special function register TCON selects between these two modes. When the TCON TR bit is set, the timer/counter is enabled. Register pair TH and TL are incremented by the clock source. When the register pair overflows, the register pair is reloaded with the values in registers RTH and RTL. The value in the reload registers is left unchanged. The TF bit in special function register TCON is set on counter overflow and, if the interrupt is enabled, will generate an interrupt (see Figure 71).

I²C Serial I/O

The I²C bus uses two wires (SDA and SCL) to transfer information between devices connected to the bus. The main technical features of the bus are:

- Bidirectional data transfer between masters and slaves
- Serial addressing of slaves
- Acknowledgment after each transferred byte
- Multimaster bus
- Arbitration between simultaneously transmitting master without corruption of serial data on bus

A large family of I²C compatible ICs is available. See the I²C section for more details on the bus and available ICs.

The 83C752 I²C subsystem includes hardware to simplify the software required to drive the I²C bus. This circuitry is the same as that on the 83C751. (See the 83C751 section for a detailed discussion of this subsystem).

Interrupts

The interrupt structure is a seven-source, one-level interrupt system similar to the 8XC751. The interrupt sources are listed below in their order of polling sequence priority (highest to lowest):

Priority	Source	Function
Highest	INT0	External interrupt 0
	TF0	Timer flag 0
	INT1	External interrupt 1
	PWM	PWM counter overflow
	TI	I²C timer overflow
	SIO	Serial port interrupt
Lowest	ADC	A/D conversion complete

The vector addresses are as follows:

Source	Vector Address
INT0	0003H
TF0	000BH
INT1	0013H
TIMER I	001BH
SIO	0023H
ADC	002BH
PWM	0033H

Interrupt Control Registers

The 80C51 interrupt enable register is modified to take into account the different interrupt sources of the 8XC752.

Interrupt Enable Register

MSB							LSB
EA	EAD	ETI	ES	EPWM	EX1	ET0	EX0

Position	Symbol	Function
IE.7	EA	Global interrupt disable when EA = 0
IE.6	EAD	A/D conversion complete
IE.5	ETI	Timer I
IE.4	ES	I²C serial port
IE.3	EPWM	PWM counter overflow
IE.2	EX1	External interrupt 1
IE.1	ET0	Timer 0 overflow
IE.0	EX0	External interrupt 0

Power-Down and Idle Modes

The 8XC752 includes the 80C51 power-down and idle mode features. The functions that continue to run while in the idle mode are Timer 0, the I²C interface including Timer I, and the interrupts. Upon powering-up the circuit, or exiting from idle mode, sufficient time must be allowed for stabilization of the internal analog reference voltages before an A/D conversion is started.

Special Function Registers

The special function registers (directly addressable only) contain all of the 8XC751 registers except the program counter and the four register banks. Most of the 21 special function registers are used to control the on-chip peripheral hardware. Other registers include arithmetic registers (ACC, B, PSW), stack pointer (SP) and data pointer registers (DPH, DPL). Nine of the SFRs are bit addressable.

Data Pointer

The data pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). In the 80C51 this register allows the access of external data memory using the MOVX instruction. Since the 83C752 does not support MOVX or external memory accesses, this register is generally used as a 16-bit offset pointer of the accumulator in a MOVC instruction. DPTR may also be manipulated as two independent 8-bit registers.

Date of Issue	June 15, 1990
Status	Product Specification
Application Specific Product	

83C752/87C752

CMOS single-chip 8-bit microcontroller with A/D, PWM

DESCRIPTION

The Philips 83C752/87C752 offers many of the advantages of the 80C51 architecture in a small package and at low cost.

The 8XC752 Microcontroller is fabricated with Philips high-density CMOS technology. Philips epitaxial substrate minimizes CMOS latch-up sensitivity.

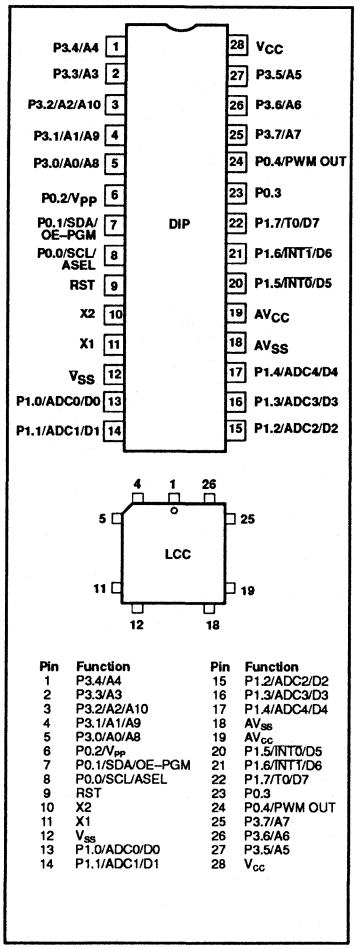
The 8XC752 contains a 2k x 8 ROM (83C752) EPROM (87C752), a 64 x 8 RAM, 21 I/O lines, a 16-bit auto-reload counter/timer, a fixed-priority level interrupt structure, a bidirectional inter-integrated circuit (I^2C) serial bus interface, an on-chip oscillator, a five channel multiplexed 8-bit A/D converter, and an 8-bit PWM output.

The onboard inter-integrated circuit (I^2C) bus interface allows the 8XC752 to operate as a master or slave device on the I^2C small area network. This capability facilitates I/O and RAM expansion, access to EEPROM, processor-to-processor communication, and efficient interface to a wide variety of dedicated I^2C peripherals.

FEATURES

- Available in erasable quartz lid or One-Time Programmable plastic packages
- 80C51 based architecture
- Inter-integrated Circuit (I^2C) serial bus interface
- Small package sizes
  - 28-pin DIP
  - 28-pin PLCC
- Wide oscillator frequency range
- Low power consumption:
  - Normal operation: less than 11mA @ 5V, 12MHz
  - Idle mode
  - Power-down mode
- 2k x 8 ROM (83C752) EPROM (87C752)
- 64 x 8 RAM
- 16-bit auto reloadable counter/timer
- 5-channel 8-bit A/D converter
- 8-bit PWM output/timer
- Fixed-rate timer
- Boolean processor
- CMOS and TTL compatible
- Well suited for logic replacement, consumer and industrial applications

PIN CONFIGURATION

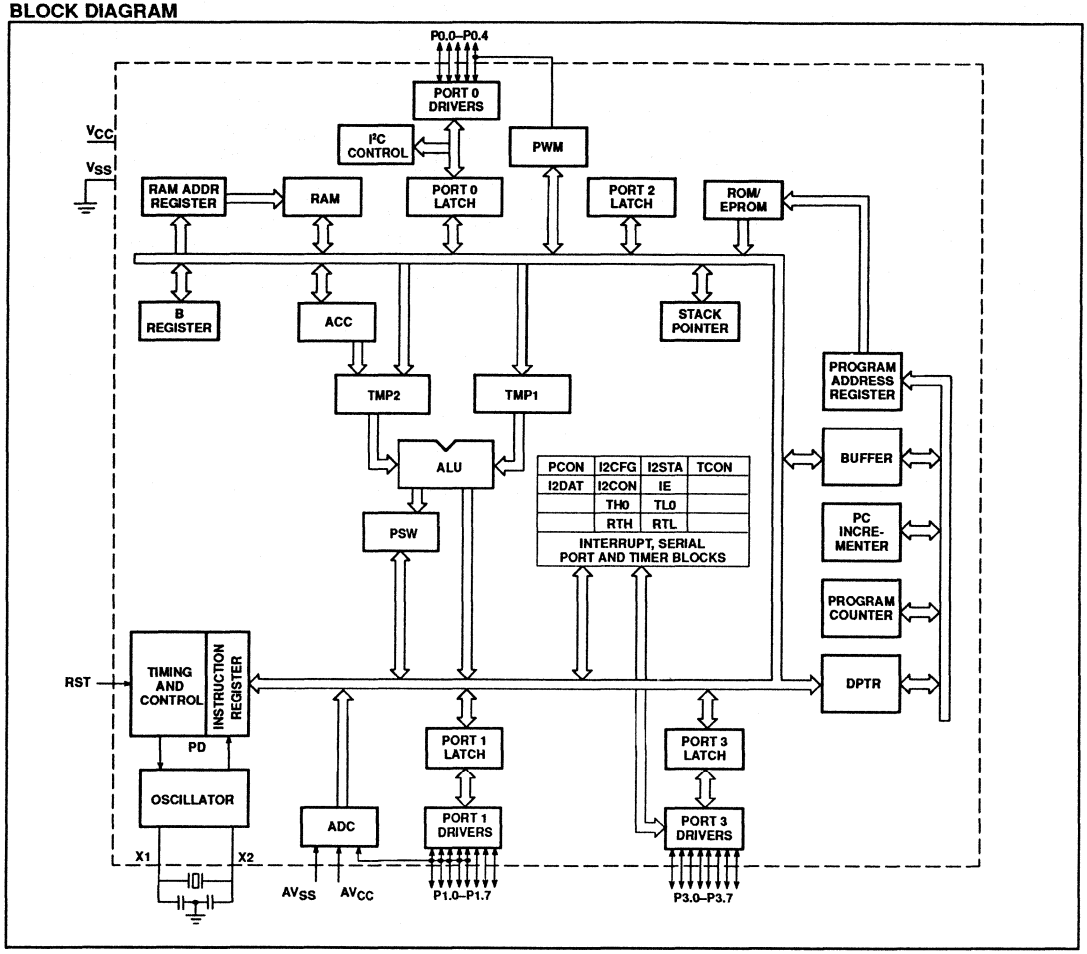


Pin	Function	Pin	Function
1	P3.4/A4	15	P1.2/ADC2/D2
2	P3.3/A3	16	P1.3/ADC3/D3
3	P3.2/A2/A10	17	P1.4/ADC4/D4
4	P3.1/A1/A9	18	AV_{SS}
5	P3.0/A0/A8	19	AV_{CC}
6	P0.2/V_{PP}	20	P1.5/INT0/D5
7	P0.1/SDA/OE–PGM	21	P1.6/INT1/D6
8	P0.0/SCL/ASEL	22	P1.7/T0/D7
9	RST	23	P0.3
10	X2	24	P0.4/PWM OUT
11	X1	25	P3.7/A7
12	V_{SS}	26	P3.6/A6
13	P1.0/ADC0/D0	27	P3.5/A5
14	P1.1/ADC1/D1	28	V_{CC}

PART NUMBER SELECTION

ROM	EPROM	SPEED	TEMPERATURE AND PACKAGE
	87C752-1F28	3.5 to 12MHz	0 to +70°C, ceramic DIP
	87C752-2F28	3.5 to 12MHz	–40 to +85°C, ceramic DIP
	87C752-4F28	3.5 to 16MHz	0 to +70°C, ceramic DIP
	87C752-5F28	3.5 to 16MHz	–40 to +85°C, ceramic DIP
83C752-1N28	87C752-1N28	3.5 to 12MHz	0 to +70°C, plastic DIP
83C752-2N28	87C752-2N28	3.5 to 12MHz	–40 to +85°C, plastic DIP
83C752-4N28	87C752-4N28	3.5 to 16MHz	0 to +70°C, plastic DIP
83C752-5N28	87C752-5N28	3.5 to 16MHz	–40 to +85°C, plastic DIP
83C752-1A28	87C752-1A28	3.5 to 12MHz	0 to +70°C, plastic LCC
83C752-2A28	87C752-2A28	3.5 to 12MHz	–40 to +85°C, plastic LCC
83C752-4A28	87C752-4A28	3.5 to 16MHz	0 to +70°C, plastic LCC
83C752-5A28	87C752-5A28	3.5 to 16MHz	–40 to +85°C, plastic LCC
83C752-6A28	87C752-6A28	3.5 to 12MHz	–55 to +125°C, plastic LCC
83C752-6F28	87C752-6F28	3.5 to 12MHz	–55 to +125°C, ceramic DIP
83C752-6N28	87C752-6N28	3.5 to 12MHz	–55 to +125°C, plastic DIP

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
V_{SS}	12	I	**Circuit Ground Potential.**
V_{CC}	28	I	**Supply voltage during normal, idle, and power-down operation.**
P0.0–P0.4	8–6 23, 24	I/O	**Port 0:** Port 0 is a 5-bit bidirectional port. Port 0.0–P0.2 are open drain. Port 0.0–P0.2 pins that have 1s written to them float, and in that state can be used as high-impedance inputs. P0.3–P0.4 are bidirectional I/O port pins with internal pull-ups. Port 0 also serves as the serial I^2C interface. When this feature is activated by software, SCL and SDA are driven low in accordance with the I^2C protocol. These pins are driven low if the port register bit is written with a 0 or if the I^2C subsystem presents a 0. The state of the pin can always be read from the port register by the program. Port 0.3 and 0.4 have internal pull-ups that function identically to port 3. Pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. To comply with the I^2C specification, P0.0 and P0.1 are open drain bidirectional I/O pins with the electrical characteristics listed in the tables that follow. While these differ from "standard TTL" characteristics, they are close enough for the pins to still be used as general-purpose I/O in non-I^2C applications.
	6	I	**V_{PP} (P0.2)** – Programming voltage input.
	7	I	**OE/PGM (P0.1)** – Input which specifies verify mode (output enable) or the program mode. OE/PGM = 1 output enabled (verify mode). OE/PGM = 0 program mode.
	8	I	**ASEL (P0.0)** – Input which indicates which bits of the EPROM address are applied to port 3. ASEL = 0 low address byte available on port 3. ASEL = 1 high address byte available on port 3 (only the three least significant bits are used).
P1.0–P1.7	13–17, 20–22	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. P0.3–P0.4 pins are bidirectional I/O port pins with internal pull-ups. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 1 also serves the special function features of the SC80C51 family as listed below:
	20	I	**$\overline{INT0}$ (P1.5):** External interrupt.
	21	I	**$\overline{INT1}$ (P1.6):** External interrupt.
	22	I	**T0 (P1.7):** Timer 0 external input.
	13–17	I	**ADC0 (P1.0)–ADC4 (P1.4):** Port 1 also functions as the inputs to the five channel multiplexed A/D converter. These pins can be used as outputs only if the A/D function has been disabled. These pins can be used as inputs while the A/D converter is enabled. Port 1 serves to output the addressed EPROM contents in the verify mode and accepts as inputs the value to program into the selected address during the program mode.
P3.0–P3.7	5–1, 27–25	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also functions as the address input for the EPROM memory location to be programmed (or verified). The 11-bit address is multiplexed into this port as specified by P0.0/ASEL.
RST	9	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running resets the device. An internal diffused resistor to V_{SS} permits a power-on RESET using only an external capacitor to V_{CC}. After the device is reset, a 10-bit serial sequence, sent LSB first, applied to RESET, places the device in the programming state allowing programming address, data and V_{PP} to be applied for programming or verification purposes. The RESET serial sequence must be synchronized with the X1 input.
X1	11	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits. X1 also serves as the clock to strobe in a serial bit stream into RESET to place the device in the programming state.
X2	10	O	**Crystal 2:** Output from the inverting oscillator amplifier.
AV_{CC}	19	I	**Analog supply voltage and reference input.**
AV_{CC}	18	I	**Analog supply and reference ground.**

OSCILLATOR CHARACTERISTICS

X1 and X2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator.

To drive the device from an external clock source, X1 should be driven while X2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

IDLE MODE

In idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. the control bits for the reduced power modes are in the special function register PCON.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	Port 0*	Port 1	Port 2
Idle	Data	Data	Data
Power-down	Data	Data	Data

*Except for PWM output (P0.4).

DIFFERENCES BETWEEN THE 8XC752 AND THE 80C51

Program Memory

On the 8XC752, program memory is 2048 bytes long and is not externally expandable, so the 80C51 instructions MOVX, LJMP, and LCALL are not implemented. The only fixed locations in program memory are the addresses at which execution is taken up in response to reset and interrupts, which are as follows:

Event	Program Memory Address
Reset	000
External $\overline{INT0}$	003
Counter/timer 0	00B
External $\overline{INT1}$	013
Timer I	01B
I^2C serial	023
ADC	02B
PWM	033

Counter/Timer Subsystem

The 8XC752 has one counter/timer called timer/counter 0. Its operation is similar to mode 2 operation on the 80C51, but is extended to 16 bits with 16 bits of autoload. The controls for this counter are centralized in a single register called TCON.

A watchdog timer, called Timer I, is for use with the I^2C subsystem. In I^2C applications, this timer is dedicated to time-generation and bus monitoring of the I^2C. In non-I^2C applications, it is available for use as a fixed time-base.

Interrupt Subsystem – Fixed Priority

The IP register and the 2-level interrupt system of the SC80C51 are eliminated. Simultaneous interrupt conditions are resolved by a single-level, fixed priority as follows:

Highest priority:	Pin $\overline{INT0}$
	Counter/timer flag 0
	Pin $\overline{INT1}$
	PWM
	Timer I
	Serial I^2C
Lowest priority:	ADC

Serial Communications

The 8XC752 contains an I^2C serial communications port instead of the 80C51 UART. The I^2C serial port is a single bit hardware interface with all of the hardware necessary to support multimaster and slave operations. Also included are receiver digital filters and timer (timer I) for communication watch-dog purposes. The I^2C serial port is controlled through four special function registers; I^2C control, I^2C data, I^2C status, and I^2C configuration.

Pulse Width Modulation Output (P0.4)

The PWM outputs pulses of programmable length and interval. The repetition frequency is defined by an 8-bit prescaler which generates the clock for the counter. The prescaler register is PWMP. The prescaler and counter are not associated with any other timer. The 8-bit counter counts modulo 255, that is from 0 to 254 inclusive. The value of the 8-bit counter is compared to the contents of a compare register, PWM. When the counter value matches the contents of this register, the output of the PWM is set high. When the counter reaches zero, the output of the PWM is set low. The pulse width ratio (duty cycle) is defined by the contents of the compare register and is in the range of 0 to 1 programmed in increments of 1/255. The PWM output can be set to be continuously high by loading the compare register with 0 and the output can be set to be continuously low by loading the compare register with 255. The PWM output is enabled by a bit in a special function register, PWENA. When enabled, the pin output is driven with a fully active pull-up. That is, when the output is high, a strong pull-up is continuously applied. when disabled, the pin functions as a normal bidirectional I/O pin, however, the counter remains active.

The PWM function is disabled during RESET and remains disabled after reset is removed until re-enabled by software. The PWM output is high during power down and idle. The counter is disabled during idle. The repetition frequency of the PWM is given by:

$$f_{PWM} = f_{OSC} / 2\ (1 + PWMP)\ 255$$

The low/high ratio of the PWM signal is PWM / (255 – PWM) for PWM not equal to 255. For PWM = 255, the output is always low.

the repetition frequency range is 92Hz to 23.5kHz for an oscillator frequency of 12MHz.

An interrupt will be asserted upon PWM counter overflow if the interrupt is not masked off.

The PWM output is an alternative function of P0.4. In order to use this port as a bidirectional I/O port, the PWM output must be disabled by clearing the enable/disable bit in PWENA. In this case, the PWM subsystem can be used as an interval timer by enabling the PWM interrupt.

A/D Converter

The analog input circuitry consists of a 5-input analog multiplexer and an A to D converter with 8-bit resolution. The conversion takes 40 machine cycles, i.e., 40µs at 12MHz oscillator frequency. The A/D converter is controlled using the ADCON control register. Input channels are selected by the analog multiplexer through ADCON register bits 0–2.

Special Function Register Addresses

Special function registers for the 8XC752 are identical to those of the SC80C51, except for the changes listed below:

SC80C51 special function registers not present in the 8XC752 are TMOD (89), P2 (A0) and IP (B8). The SC80C51 registers TH1, TL1, SCON, and SBUF are replaced with the 8XC752 registers RTH, RTL, I2CON, and I2DAT, respectively. Additional special function registers are I2CFG (D8) and I2STA (FB), ADCON (A0), ADAT (84), PWM (8E), PWMP (8F), and PWENA (FE). See Table 2.

Table 2. I²C Special Function Register Addresses

REGISTER ADDRESS			BIT ADDRESS							
NAME	SYMBOL	ADDRESS	MSB							LSB
I²C control	I2CON	98	9F	9E	9D	9C	9B	9A	99	98
I²C data	I2DAT	99	–	–	–	–	–	–	–	–
I²C configuration	I2CFG	D8	DF	DE	DD	DC	DB	DA	D9	D8
I²C status	I2STA	F8	FF	FE	FD	FC	FB	FA	F9	F8

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Voltage from V_{CC} to V_{SS}	–0.5 to +6.5	V
Voltage from any pin to V_{SS} (except V_{PP})	–0.5 to V_{CC} + 0.5	V
Power dissipation	1.0	W
Voltage from V_{PP} pin to V_{SS}	–0.5 to + 13.0	V

DC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or –40°C to +85°C, AV_{CC} = 5V ±5, AV_{SS} = 0V[3]

83C752: V_{CC} = 5V ± 20%, 87C752: V_{CC} = 5V ± 10%, VSS = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS[3]			UNIT
			MIN	TYPICAL[1]	MAX	
I_{CC}	Supply current (see Figure 3)					
Inputs						
V_{IL}	Input low voltage, except SDA, SCL		–0.5		$0.2V_{CC}$–0.	V
V_{IH}	Input high voltage, except X1, RST		$0.2V_{CC}$+0.		1	V
V_{IH1}	Input high voltage, X1, RST		9 $0.7V_{CC}$		V_{CC}+0.5 V_{CC}+0.5	V
	SDA, SCL:					
V_{IL1}	Input low voltage		–0.5		$0.3V_{CC}$	V
V_{IH2}	Input high voltage		$0.7V_{CC}$		V_{CC}+0.5	V
Outputs						
V_{OL}	Output low voltage, ports 1, 3, 0.3, and 0.4 (PWM disabled)	I_{OL} = 1.6mA			0.45	V
V_{OL1}	Output low voltage, port 0.2	I_{OL} = 3.2mA			0.45	V
V_{OH}	Output high voltage, ports 1, 3, 0.3, and 0.4 (PWM disabled)	I_{OH} = –60µA,	2.4			V
		I_{OH} = –25µA	$0.75V_{CC}$			V
		I_{OH} = –10µA	$0.9V_{CC}$			V
V_{OH2}		I_{OH} = –400µA	2.4			V
	Output high voltage, 0.4 (PWM enabled)	I_{OH} = –40µA	$0.9V_{CC}$			V
	Port 0.0 and 0.1 (I²C) – Drivers					
V_{OL2}	Output low voltage	I_{OL} = 3mA (over V_{CC} range)			0.4	V
	Driver, receiver combined:					
C	Capacitance				10	pF
I_{IL}	Logical 0 input current, ports 1, 3, 0.3, and 0.4 (PWM disabled)	V_{IN} = 0.45V			–50	µA
I_{TL}	Logical 1 to 0 transition current, ports 1, 3, 0.3 and 0.4	V_{IN} = 2V			–650	µA
I_{LI}	Input leakage current, port 0.0, 0.1 and 0.2	0.45 < V_{IN} < V_{CC}			±10	µA
R_{RST}	Reset pull-down resistor		25		175	kohm

DC ELECTRICAL CHARACTERISTICS (continued)

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS[3] MIN	TYPICAL[1]	MAX	UNIT
Outputs (Continued)						
C_{IO}	Pin capacitance	Test freq = 1MHz, T_A = 25°C			10	pF
I_{PD}	Power-down current[6]	V_{CC} = 2 to 5.5V V_{CC} = 2 to 6.0V (83C752)			50	µA
V_{PP}	V_{PP} program voltage (87C752 only)	V_{SS} = 0V V_{CC} = 5V±10% T_A = 21°C to 27°C	12.5		13.0	V
I_{PP}	Program current (87C752 only)	V_{PP} = 13.0V			10	mA
Analog Inputs (A/D guaranteed only with quartz window covered).						
AV_{CC}	Analog supply voltage[9]	AV_{CC} = V_{CC}±0.2V	4.5		5.5	V
AI_{CC}	Analog operating supply current	AV_{CC} = 5.12V			3[8]	mA
AV_{IN}	Analog input voltage		AV_{SS}–0.2		AV_{CC}+0.2	V
C_{IA}	Analog input capacitance				15	pF
t_{ADS}	Sampling time				$8t_{CY}$	s
t_{ADC}	Conversion time				$40t_{CY}$	s
R	Resolution				8	bits
E_{RA}	Relative accuracy				±1	LSB
OS_e	Zero scale offset				±1	LSB
G_e	Full scale gain error				0.4	%
M_{CTC}	Channel to channel matching				±1	LSB
C_t	Crosstalk	0–100kHz			–60	dB

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.
4. Power-down I_{CC} is measured with all output pins disconnected; port 0 = V_{CC}; X2, X1 n.c.; RST = V_{SS}.
5. I_{CC} is measured with all output pins disconnected; X1 driven with t_{CLCH}, t_{CHCL} = 5ns, V_{IL} = V_{SS} + 0.5V, V_{IH} = V_{CC} – 0.5V; X2 n.c.; RST = port 0 = V_{CC}. I_{CC} will be slightly higher if a crystal oscillator is used.
6. Idle I_{CC} is measured with all output pins disconnected; X1 driven with t_{CLCH}, t_{CHCL} = 5ns, V_{IL} = V_{SS} + 0.5V, V_{IH} = V_{CC} – 0.5V; X2 n.c.; port 0 = V_{CC}; RST = V_{SS}.
7. Load capacitance for ports = 80pF.
8. The resistor ladder network is not disconnected in the power down or idle modes. Thus, to conserve power, the user may remove AV_{CC}.
9. If the A/D function is not required, or if the A/D function is only needed periodically, AV_{CC} may be removed without affecting the operation of the digital circuitry. Contents of ADCON and ADAT are not guaranteed to be valid. Digital inputs on P1.0–P1.4 will not function normally.

A/D CONVERTER PARAMETER DEFINITIONS

The following definitions are included to clarify some specifications given and do not represent a complete set of A/D parameter definitions.

Absolute Accuracy Error

Absolute accuracy error of a given output is the difference between the theoretical analog input voltage to produce a given output and the actual analog input voltage required to produce the same code. Since the same output code is produced by a band of input voltages, the "required input voltage" is defined as the midpoint of the band of input voltage that will produce that code. Absolute accuracy error not specified with a code is the maximum over all codes.

Nonlinearity

If a straight line is drawn between the end points of the actual converter characteristics such that zero offset and full scale errors are removed, then non-linearity is the maximum deviation of the code transitions of the actual characteristics from that of the straight line so constructed. This is also referred to as relative accuracy and also integral non-linearity.

Differential Non-Linearity

Differential non-linearity is the maximum difference between the actual and ideal code widths fo the converter. The code widths are the differences expressed in LSB between the code transition points, as the input voltage is varied through the range for the complete set of codes.

Gain Error
Gain error is the deviation between the ideal and actual analog input voltage required to cause the final code transition to a full-scale output code after the offset error has been removed. This may sometimes be referred to as full scale error.

Offset Error
Offset error is the difference between the actual input voltage that causes the first code transition and the ideal value to cause the first code transition. This ideal value is 1/2 LSB above V_{ref-}.

Channel to Channel Matching
Channel to channel matching is the maximum difference between the corresponding code transitions of the actual characteristics taken from different channels under the same temperature, voltage and frequency conditions.

Crosstalk
Crosstalk is the measured level of a signal at the output of the converter resulting from a signal applied to one deselected channel.

Total Error
Maximum deviation of any step point from a line connecting the ideal first transition point to the ideal last transition point.

Relative Accuracy
Relative accuracy error is the deviation of the ADC's actual code transition points from the ideal code transition points on a straight line which connects the ideal first code transition point and the final code transition point, after nulling offset error and gain error. It is generally expressed in LSBs or in percent of FSR.

AC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or −40°C to +85°C, V_{CC} = 5V ±10% (87C752), V_{CC} = 5V ±20% (83C752), V_{SS} = 0V[3, 7]

SYMBOL	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
		MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	Oscillator frequency:			3.5 3.5 0.5	12 16 12	MHz MHz MHz
External Clock (Figure 1)						
t_{CHCX}	High time	20		20		ns
t_{CLCX}	Low time	20		20		ns
t_{CLCH}	Rise time		20		20	ns
t_{CHCL}	Fall time		20		20	ns

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

C – Clock
D – Input data
H – Logic level high
L – Logic level low
Q – Output data
T – Time
V – Valid
X – No longer a valid l;ogic level
Z – Float

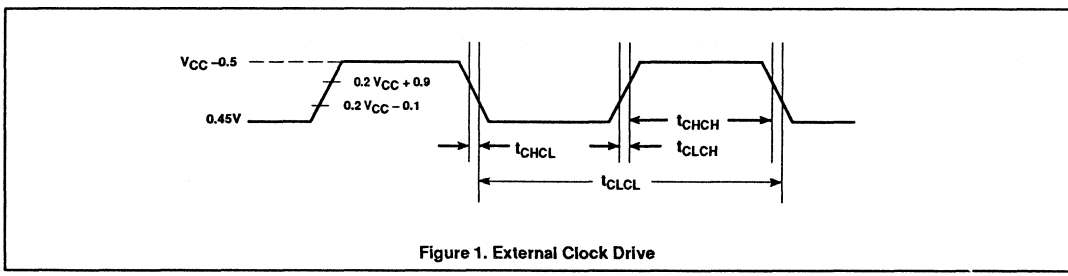


Figure 1. External Clock Drive

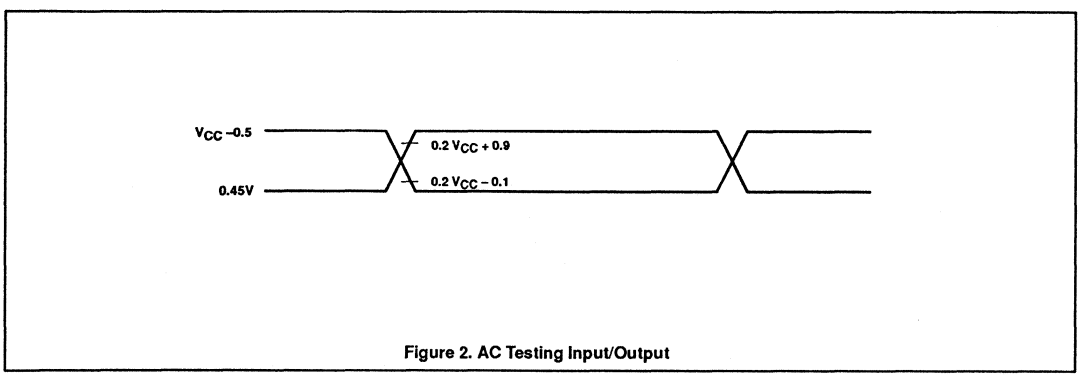


Figure 2. AC Testing Input/Output

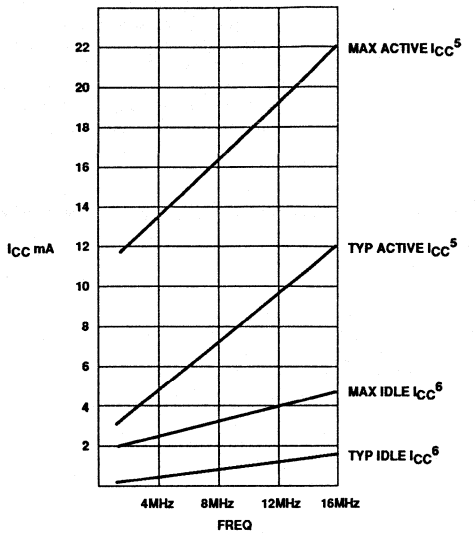


Figure 3. I_{CC} vs. FREQ
Maximum I_{CC} values taken at V_{CC} = 5.5V and worst case temperature.
Typical I_{CC} values taken at V_{CC} = 5.0V and 25°C.
Notes 5 and 6 refer to AC Electrcial Characteristics.

PROGRAMMING CONSIDERATIONS

EPROM Characteristics

The 87C752 is programmed by using a modified Quick-Pulse Programming algorithm similar to that used for devices such as the 87C451 and 87C51. It differs from these devices in that a serial data stream is used to place the 87C752 in the programming mode.

Figure 4 shows a block diagram of the programming configuration for the 87C752. Port pin P0.2 is used as the programming voltage supply input (V_{PP} signal). Port pin P0.1 is used as the program (PGM/) signal. This pin is used for the 25 programming pulses.

Port 3 is used as the address input for the byte to be programmed and accepts both the high and low components of the eleven bit address. Multiplexing of these address components is performed using the ASEL input. The user should drive the ASEL input high and then drive port 3 with the high order bits of the address. ASEL should remain high for at least 13 clock cycles. ASEL may then be driven low which latches the high order bits of the address internally. the high address should remain on port 3 for at least two clock cycles after ASEL is driven low. Port 3 may then be driven with the low byte of the address. The low address will be internally stable 13 clock cycles later. The address will remain stable provided that the low byte placed on port 3 is held stable and ASEL is kept low. **Note:** ASEL needs to be pulsed high only to change the high byte of the address.

Port 1 is used as a bidirectional data bus during programming and verify operations. During programming mode, it accepts the byte to be programmed. During verify mode, it provides the contents of the EPROM location specified by the address which has been supplied to Port 3.

The XTAL1 pin is the oscillator input and receives the master system clock. This clock should be between 1.2 and 6MHz.

The RESET pin is used to accept the serial data stream that places the 87C752 into various programming modes. This pattern consists of a 10-bit code with the LSB sent first. Each bit is synchronized to the clock input, X1.

Programming Operation

Figures 5 and 6 show the timing diagrams for the program/verify cycle. RESET should initially be held high for at least two machine cycles. P0.1 (PGM/) and P0.2 (V_{PP}) will be at V_{OH} as a result of the RESET operation. At this point, these pins function as normal quasi-bidirectional I/O ports and the programming equipment may pull these lines low. However, prior to sending the 10-bit code on the RESET pin, the programming equipment should drive these pins high (V_{IH}). The RESET pin may now be used as the serial data input for the data stream which places the 87C752 in the programming mode. Data bits are sampled during the clock high time and thus should only change during the time that the clock is low. Following transmission of the last data bit, the RESET pin should be held low.

Next the address information for the location to be programmed is placed on port 3 and ASEL is used to perform the address multiplexing, as previously described. At this time, port 1 functions as an output.

A high voltage V_{PP} level is then applied to the V_{PP} input (P0.2). (This sets Port 1 as an input port). The data to be programmed into the EPROM array is then placed on Port 1. This is followed by a series of programming pulses applied to the PGM/ pin (P0.1). These pulses are created by driving P0.1 low and then high. This pulse is repeated until a total of 25 programming pulses have occurred. At the conclusion of the last pulse, the PGM/ signal should remain high.

The V_{PP} signal may now be driven to the V_{OH} level, placing the 87C752 in the verify mode. (Port 1 is now used as an output port). After four machine cycles (48 clock periods), the contents of the addressed location in the EPROM array will appear on Port 1.

The next programming cycle may now be initiated by placing the address information at the inputs of the multiplexed buffers, driving the V_{PP} pin to the V_{PP} voltage level, providing the byte to be programmed to Port1 and issuing the 26 programming pulses on the PGM/ pin, bringing V_{PP} back down to the V_C level and verifying the byte.

Programming Modes

The 87C752 has four programming features incorporated within its EPROM array. These include the USER EPROM for storage of the application's code, a 16-byte encryption key array and two security bits. Programming and verification of these four elements are selected by a combination of the serial data stream applied to the RESET pin and the voltage levels applied to port pins P0.1 and P0.2. The various combinations are shown in Table 3.

Encryption Key Table

The 87C752 includes a 16-byte EPROM array that is programmable by the end user. The contents of this array can then be used to encrypt the program memory contents during a program memory verify operation. When a program memory verify operation is performed, the contents of the program memory location is XNOR'ed with one of the bytes in the 16-byte encryption table. The resulting data pattern is then provided to port 1 as the verify data. The encryption mechanism can be disable, in essence, by leaving the bytes in the encryption table in their erased state (FFH) since the XNOR product of a bit with a logical one will result in the original bit. The encryption bytes are mapped with the code memory in 16-byte groups. the first byte in code memory will be encrypted with the first byte in the encryption table; the second byte in code memory will be encrypted with the second byte in the encryption table and so forth up to and including the 16the byte. The encryption repeats in 16-byte groups; the 17th byte in the code memory will be encrypted with the first byte in the encryption table, and so forth.

Security Bits

Two security bits, security bit 1 and security bit 2, are provided to limit access to the USER EPROM and encryption key arrays. Security bit 1 is the program inhibit bit, and once programmed performs the following functions:

1. Additional programming of the USER EPROM is inhibited.
2. Additional programming of the encryption key is inhibited.
3. Verification of the encryption key is inhibited.
4. Verification of the USER EPROM and the security bit levels may still be performed.

(If the encryption key array is being used, this security bit should be programmed by the user to prevent unauthorized parties from reprogramming the encryption key to all logical zero bits. Such programming would provide data during a verify cycle that is the logical complement of the USER EPROM contents).

Security bit 2, the verify inhibit bit, prevents verification of both the USER EPROM array and the encryption key arrays. The security bit levels may still be verified.

Programming and Verifying Security Bits

Security bits are programmed employing the same techniques used to program the USER EPROM and KEY arrays using serial data streams and logic levels on port pins indicated in Table 3. When programming either security bit, it is not necessary to provide address or data information to the 87C752 on ports 1 and 3.

Verification occurs in a similar manner using the RESET serial stream shown in Table 3.

Port 3 is not required to be driven and the results of the verify operation will appear on ports 1.6 and 1.7.

Ports 1.7 contains the security bit 1 data and is a logical one if programmed and a logical zero if erased. Likewise, P1.6 contains the security bit 2 data and is a logical one if programmed and a logical zero if erased.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Flourless part number 2345–5 or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000uW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

Table 3. Implementing Program/Verify Modes

OPERATION	SERIAL CODE	P0.1 (PGM/)	P0.2 (V_{PP})
Program user EPROM	296H	–*	V_{PP}
Verify user EPROM	296H	V_{IH}	V_{IH}
Program key EPROM	292H	–*	V_{PP}
Verify key EPROM	292H	V_{IH}	V_{IH}
Program security bit 1	29AH	–*	V_{PP}
Program security bit 2	298H	–*	V_{PP}
Verify security bits	29AH	V_{IH}	V_{IH}

NOTE:
*Pulsed from V_{IH} to V_{IL} and returned to V_{IH}.

EPROM PROGRAMMING AND VERIFICATION

T_A = 21°C to +27°C, V_{CC} = 5V ±10%, V_{SS} = 0V

SYMBOL	PARAMETER	MIN	MAX	UNIT
$1/t_{CLCL}$	Oscillator/clock frequency	1.2	6	MHz
t_{AVGL}*	Address setup to P0.1 (PROG–) low	10µs + 24t_{CLCL}		
t_{GHAX}	Address hold after P0.1 (PROG–) high	48t_{CLCL}		
t_{DVGL}	Data setup to P0.1 (PROG–) low	38t_{CLCL}		
t_{DVGL}	Data setup to P0.1 (PROG–) low	38t_{CLCL}		
t_{GHDX}	Data hold after P0.1 (PROG–) high	36t_{CLCL}		
t_{SHGL}	V_{PP} setup to P0.1 (PROG–) low	10		µs
t_{GHSL}	V_{PP} hold after P0.1 (PROG–)	10		µs
t_{GLGH}	P0.1 (PROG–) width	90	110	µs
t_{AVQV}**	V_{PP} low (V_{CC}) to data valid		48t_{CLCL}	
t_{GHGL}	P0.1 (PROG–) high to P0.1 (PROG–) low	10		µs
t_{SYNL}	P0.0 (sync pulse) low	4t_{CLCL}		
t_{SYNH}	P0.0 (sync pulse) high	8t_{CLCL}		
t_{MASEL}	ASEL high time	13t_{CLCL}		
t_{MAHLD}	Address hold time	2t_{CLCL}		
t_{HASET}	Address setup to ASEL	13t_{CLCL}		
t_{ADSTA}	Low address to address stable	13t_{CLCL}		

NOTES:
*Address should be valid at least 24t_{CLCL} before the rising edge of P0.2 (V_{PP}).
**For a pure verify mode, i.e., no program mode in between, t_{AVQV} is 14t_{CLCL} maximum.

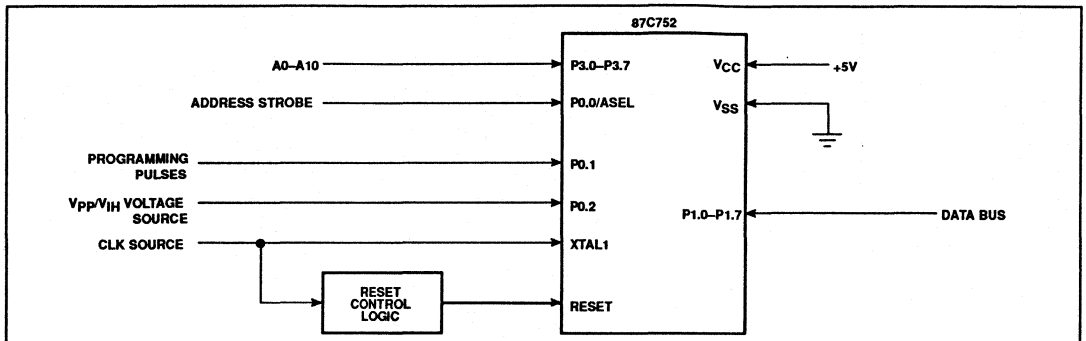


Figure 4. Programming Configuration

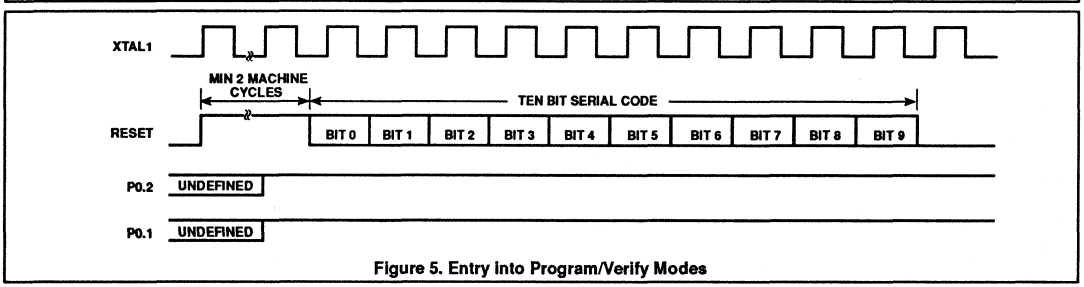


Figure 5. Entry into Program/Verify Modes

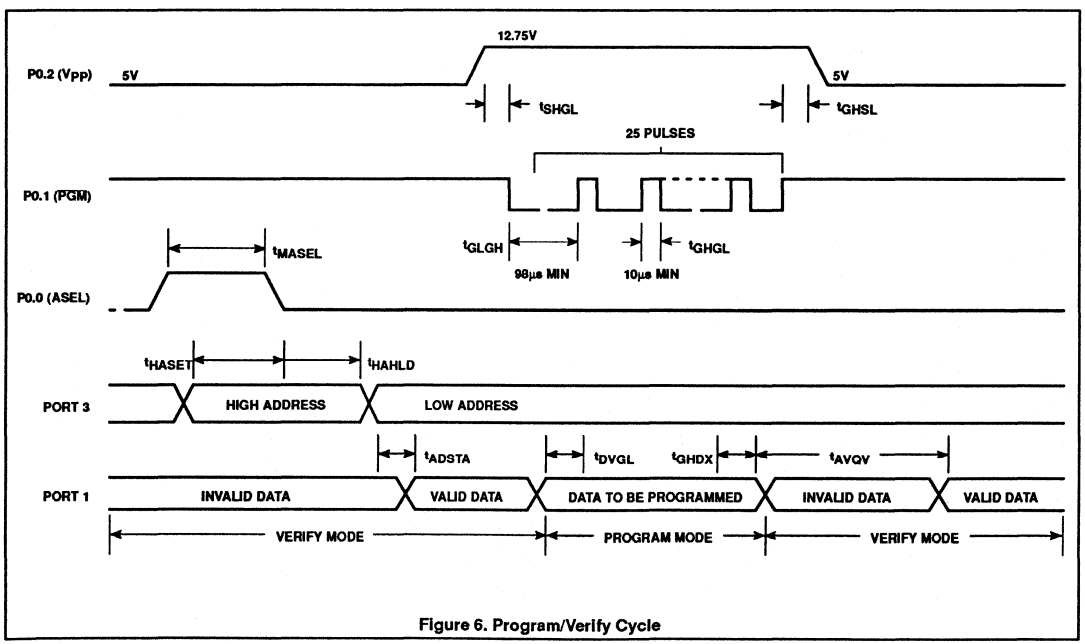


Figure 6. Program/Verify Cycle

8XC851 OVERVIEW

The 80C851 and the 83C851 (hereafter referred to collectively as the 8XC851) are EEPROM expanded versions of the 80C51. These devices are pin-for-pin compatible with the 80C51 with the addition of 256 bytes of EEPROM. The addition of the EEPROM makes these devices suitable for a variety of applications, specifically control and security systems.

The 8XC851 includes a 4k x 8 ROM, a 128 x 8 RAM, a 256 x 8 electrically erasable programmable read-only memory (EEPROM), 32 I/O lines, two 16-bit timer/counters, a seven-source, two priority level nested interrupt structure, a serial I/O port for either full duplex UART or I/O expansion, and an on-chip oscillator and clock circuit. The 80C851 includes all of the 83C851 features except the on-board 4k x 8 ROM.

The 8XC851 has two software selectable modes of reduced activity for further power reduction: idle mode and power-down mode. Idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. Power-down mode freezes the oscillator, causing all other chip functions to be inoperative while maintaining the RAM contents.

In addition, some special security features are implemented:

- ROM code protection: mask-programmable. When implemented, access to the internal ROM is possible only when executing internal program memory; it is not possible to access the internal ROM when executing external program memory.
- In the security mode (enabled when the security bit is set), the contents of the EEPROM are protected and, when executing external program memory, no read or write operation to the EEPROM is possible except "Blockerase" ("Blockerase" clears all bytes, including the byte containing the security bits).

The 8XC851 features include:

- 80C51 pin-for-pin compatibility
- 44-pin PLCC and 40-pin DIP packages
- 4k x 8 ROM
- 128 x 8 RAM
- 256 x 8 EEPROM
  - On-chip voltage multiplier for erase/write
  - 10,000 erase/write cycles per byte
  - 10 years non-volatile data retention
  - Infinite number of read cycles
- Two 16-bit counter/timers
- Two external interrupts
- External memory addressing capability
  - 64k ROM and 64k RAM
- Low power consumption
  - Idle mode
  - –Power-down mode
- ROM code protection
- EEPROM security mode

Differences from the 80C51

Special Function Registers

The SFRs are identical to those of the standard 80C51 with the exception of five registers (EADRL, EADRH, EDAT, ECNTRL, and ETIM) that have been added to allow control of the 256 bytes of EEPROM. Table 30 is a detailed expansion of the special function registers.

Refer to the 80C851 data sheet for additional information.

Table 30. 8XC851 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
			EF	EE	ED	EC	EB	EA	E9	E8	
DPTR:	Data pointer (2 bytes):										
DPH	High byte	83H									00H
DPL	Low byte	82H									00H
EADRH#	EEPROM addr reg-high	F3H									80H
EADRL#	EEPROM addr reg-low	F2H									00H
ECNTRL#	EEPROM control reg	F6H	IFE	EEINT	EWP	–	ECNTR L3	ECNTR L2	ECNTR L1	ECNTR L0	00H
EDAT#	EEPROM data register	F4H									xxH
ETIM#	EEPROM timer register	F5H									08H
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*	Interrupt priority	B8H	–	–	–	PS	PT1	PX1	PT0	PX0	xxx00000B
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*	Interrupt enable	A8H	EA	–	–	ES	ET1	EX1	ET0	EX0	0xx00000B
P0*	Port 0	80H	87	86	85	84	83	82	81	80	FFH
P1*	Port 1	90H	97	96	95	94	93	92	91	90	FFH
P2*	Port 2	A0H	A7	A6	A5	A4	A3	A2	A1	A0	FFH
P3*	Port 3	B0H	B7	B6	B5	B4	B3	B2	B1	B0	FFH
PCON	Power control	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxx0000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
SBUF	Serial data buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
SCON*	Serial port control	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	00H
TCON*	Timer/counter control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
TMOD	Timer/counter mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H
TH0	Timer 0 high byte	8CH									00H
TH1	Timer 1 high byte	8DH									00H
TL0	Timer 0 low byte	8AH									00H
TL1	Timer 1 low byte	8BH									00H

*SFRs are bit addressable.
#SFRs are modified from or added to the 80C51 SFRs.

Date of Issue	September 6, 1990
Status	Product Specification
Application Specific Product	

80C851/83C851

CMOS single-chip 8-bit microcontroller with on-chip EEPROM

DESCRIPTION

The Philips 80C851/83C851 is a high-performance microcontroller fabricated with Philips high-density CMOS technology. The 80C851/83C851 has the same instruction set as the 80C51. The Philips CMOS technology combines the high speed and density characteristics of HMOS with the low power attributes of CMOS. The Philips epitaxial substrate minimizes latch-up sensitivity.

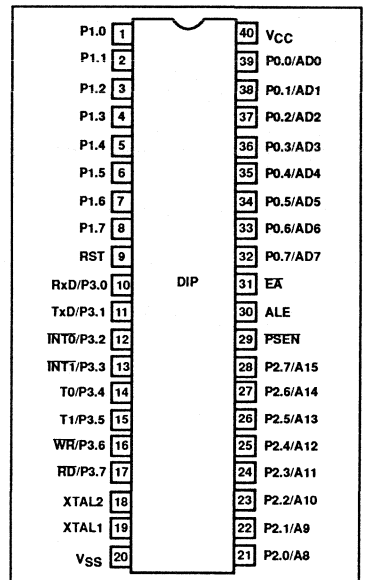
The 80C851/83C851 contains a 4k x 8 ROM with mask-programmable ROM code protection, a 128 x 8 RAM, 256 x 8 EEPROM, 32 I/O lines, two 16-bit counter/timers, a seven-source, five vector, two-priority level nested interrupt structure, a serial I/O port for either multi-processor communications, I/O expansion or full duplex UART, and on-chip oscillator and clock circuits.

In addition, the 80C851/83C851 has two software selectable modes of power reduction – idle mode and power-down mode. The idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. The power-down mode saves the RAM and EEPROM contents but freezes the oscillator, causing all other chip functions to be inoperative.

FEATURES

- 80C51 based architecture
  - 4k x 8 ROM
  - 128 x 8 RAM
  - Two 16-bit counter/timers
  - Full duplex serial channel
  - Boolean processor
- Non-volatile 256 x 8-bit EEPROM (electrically erasable programmable read only memory)
  - On-chip voltage multiplier for erase/write
  - 10,000 erase/write cycles per byte
  - 10 years non-volatile data retention
  - Infinite number of read cycles
  - User selectable security mode
  - Block erase capability
- Mask-programmable ROM code protection
- Memory addressing capability
  - 64k ROM and 64k RAM
- Power control modes:
  - Idle mode
  - Power-down mode
- CMOS and TTL compatible
- 1.2 to 12MHz
- Two temperature ranges
- Three package styles

PIN CONFIGURATION



Pin	Function	Pin	Function	Pin	Function
1	NC	15	INT1/P3.3	30	P2.6/A14
2	P1.0	16	T0/P3.4	31	P2.7/A15
3	P1.1	17	T1/P3.5	32	PSEN
4	P1.2	18	WR/P3.6	33	ALE
5	P1.3	19	RD/P3.7	34	NC
6	P1.4	20	XTAL2	35	EA
7	P1.5	21	XTAL1	36	P0.7/AD7
8	P1.6	22	V_{SS}	37	P0.6/AD6
9	P1.7	23	NC	38	P0.5/AD5
10	RST	24	P2.0/A8	39	P0.4/AD4
11	RxD/P3.0	25	P2.1/A9	40	P0.3/AD3
12	NC	26	P2.2/A10	41	P0.2/AD2
13	TxD/P3.1	27	P2.3/A11	42	P0.1/AD1
14	INT0/P3.2	28	P2.4/A12	43	P0.0/AD0
		29	P2.5/A13	44	V_{CC}

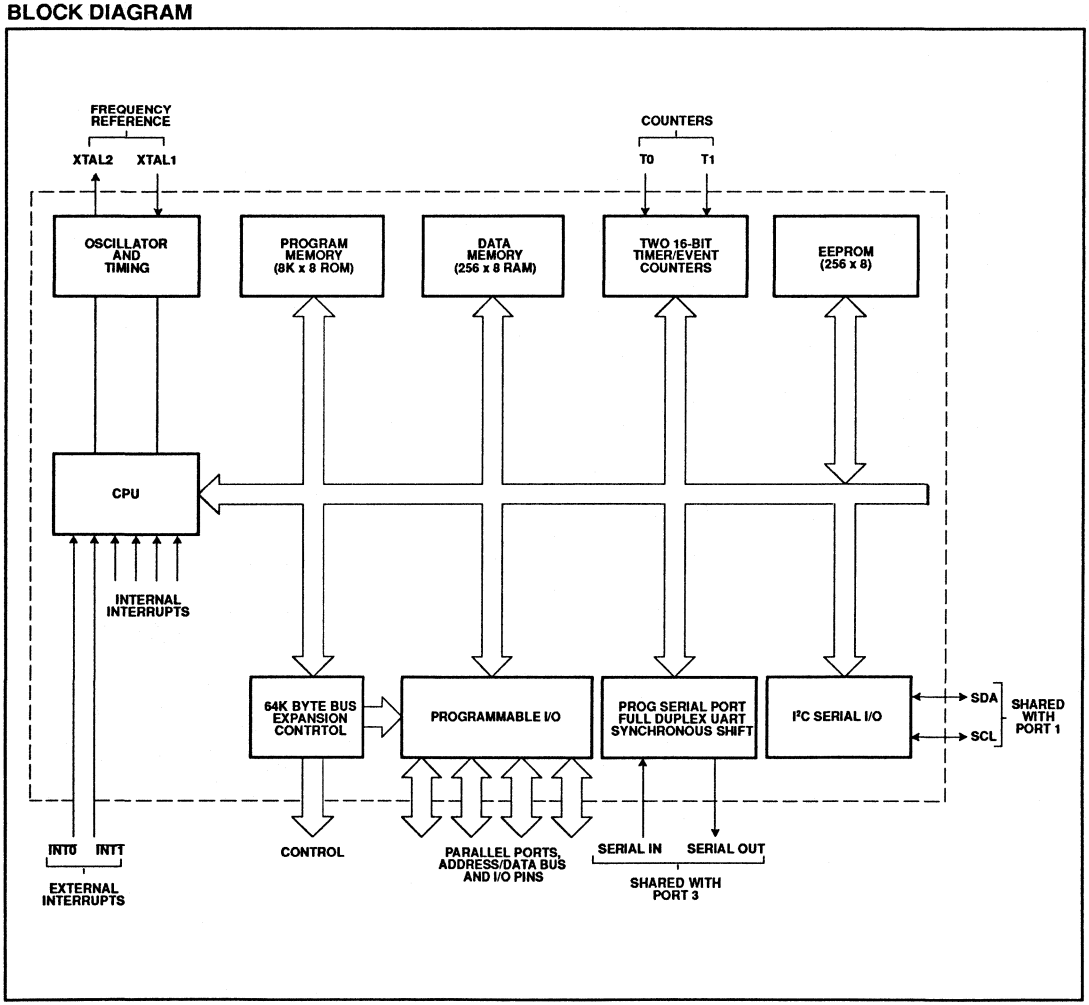
CMOS single-chip EEPROM 8-bit microcontroller

80C851/83C851

PART NUMBER SELECTION

PHILIPS		PHILIPS COMPONENTS-SIGNETICS		TEMPERATURE AND PACKAGE	FREQUENCY (MHz)
ROMless Version	ROM Version	ROMless Version	ROM Version		
PCB80C851P	PCB83C851P	S80C851-1N40	S83C851-1N40	0 to +70°C plastic DIP	1.2 to 12
PCB80C851WP	PCB83C851WP	S80C851-1A44	S83C851-1A44	0 to +70°C plastic LCC	1.2 to 12
PCB80C851H	PCB83C851H	S80C851-1B44	S83C851-1B44	0 to +70°C plastic QFP	1.2 to 12
PCF80C851P	PCF83C851P	S80C851-2N40	S83C851-2N40	–40 to +85°C plastic DIP	1.2 to 12
PCF80C851WP	PCF83C851WP	S80C851-2A44	S83C851-2A44	–40 to +85°C plastic LCC	1.2 to 12
PCF80C851H	PCF83C851H	S80C851-2B44	S83C851-2B44	–40 to +85°C plastic QFP	1.2 to 12

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	PIN NO. QFP	TYPE	NAME AND FUNCTION
V_{SS}	20	22	22	I	**Ground:** 0V reference.
V_{CC}	40	44	44	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	32–39	36–43	36–43	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s.
P1.0–P1.7	1–8	2–9	2–9	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 1 also receives the low-order address byte during program memory verification.
P2.0–P2.7	21–28	24–31	24–31	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	11, 13–19	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the SC80C51 family, as listed below:
	10	11	11	I	**RxD (P3.0):** Serial input port
	11	13	13	O	**TxD (P3.1):** Serial output port
	12	14	14	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	15	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	16	I	**T0 (P3.4):** Timer 0 external input
	15	17	17	I	**T1 (P3.5):** Timer 1 external input
	16	18	18	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	19	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	10	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE	30	33	33	I/O	**Address Latch Enable:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory.
$\overline{PSEN}$	29	32	32	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$	31	35	35	I	**External Access Enable:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H and 0FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 0FFFH.
XTAL1	19	21	21	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	20	O	**Crystal 2:** Output from the inverting oscillator amplifier.

EEPROM

Communications between the CPU and the EEPROM is accomplished via 5 special function registers; 2 address registers (high and low byte), 1 data register for read and write operations, 1 control register, and 1 timer register to adapt the erase/write time to the clock frequency. All registers can be read and written. Figure 1 shows a block diagram of the CPU, the EEPROM and the interface.

Register and Functinal Description

Address Register (EADRH, EADRL)
The lower byte contains the address of one of the 256 bytes. The higher byte (EADRH) is for future extensions and for addressing the security bits (see Security Facilities). The The EADRH register address is F3H. the EADRL register address is F2H.

Data Register (EDAT)
This register is required for read and write operations and also for row/block erase. In write mode, its contents are written to the addressed byte (for "row erase" and "block erase" the contents are don't care). the write pulse starts all operations, except read. In read mode, EDAT contains the data of the addressed byte. The EDAT register address is F4H.

Timer Register (ETIM)
The timer register is required to adapt the eraswe/write time to the oscillator frequency. The user has to ensure that the erase or write (program) time is neither too short or too long.

The ETIM register address is F5H.Table 1 contains the values which must be written to the ETIM register by software for various oscillator frequencies (the default value is 08H after RESET).

The general formula is:

Value (decimal) = f_{XTAL1} (kHz) / 96 – 2

The adjustment range is 0 to 255 corresponding to frequencies of 192kHz to 24.7MHz.

Control Register (ECNTRL)
See Figure 2 for a description of this register. The ECNTRL register address is F6H.

Table 1. Values for the Timer Register (ETIM)

f_{XTAL1} (MHz)	VALUES FOR ETIM		
	BINARY	HEXADECIMAL	DECIMAL
1.2	00001011	0B	11
2.0	00010011	13	19
3.0	00011101	1D	29
4.0	00101000	28	40
5.0	00110010	32	50
6.0	00111100	3C	60
7.0	01000111	47	71
8.0	01010001	51	81
9.0	01011100	5C	92
10.0	01100110	66	102
11.0	01110001	71	113
12.0	01111011	7B	123

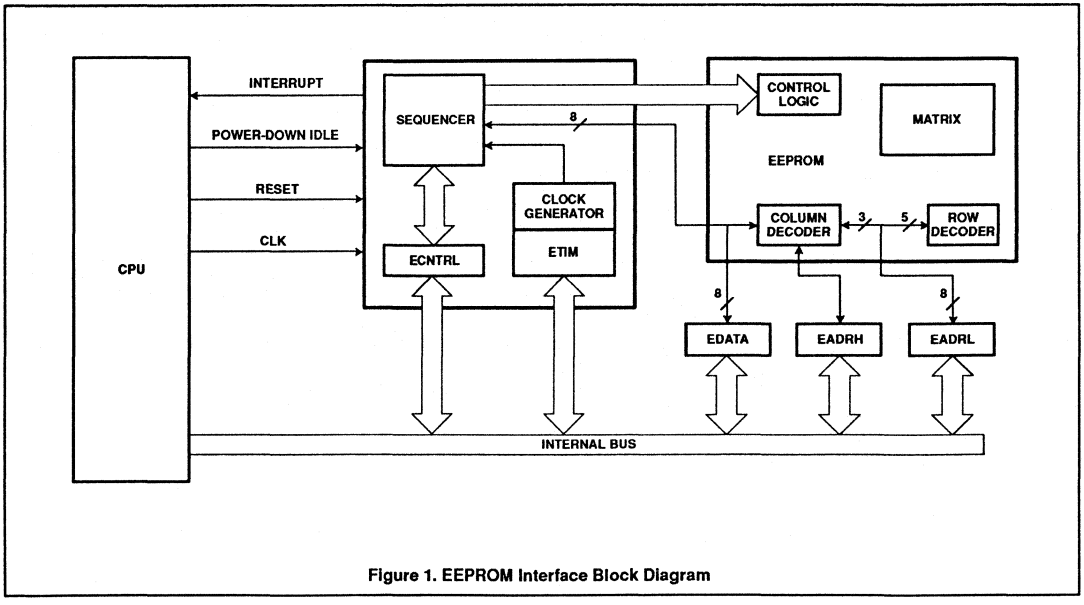


Figure 1. EEPROM Interface Block Diagram

7	6	5	4	3	2	1	0
IFE	EEINT	EWP	--	ECNTRL3	ECNTRL2	ECNTRL1	ECNTRL0

Bit	Symbol	Function
ECNTRL.7	IFE	Active high EEPROM interrupt flag set by the sequencer or by software; reset by software. When set and enabled, this flag forces an interrupt to the same vector as the serial port interrupt (see Interrupt section).
ECNTRL.6	EEINT	EEPROM interrupt enable set and reset by software (active high).
ECNTRL.5	EWP	Erase/write in progress flag set and reset by the sequencer (active high). When EWP is set, access to the EEPROM is not possible. EWP cannot be set or reset by software.
ECNTRL.4		Reserved.
ECCTRL.3–ECNTRL.0		See table below.

Operation	ECNTRL.3	ECNTRL.2	ECNTRL.1	ECNTRL.0
Byte mode	0	0	0	0
Row erase	1	1	0	0
Page write*	–	–	–	–
Page erase/write*	–	–	–	–
block erase	1	0	1	0

*Future products.

Byte mode: Normal EEPROM mode, default mode after reset. In this mode, data can be read and written to one byte at a time.

Read mode: This is the default mode when byte mode is selected. This means that the contents of the addressed byte are available in the data register.

Write mode: This mode is activated by writing to the data register. The address register must be loaded first. Since the old contents are read first (by default), this allows the sequencer to decide whether an erase/write or write cycle only (data = 00H) is required.

Row erase: In this mode, the addressed row is cleared. The three LSBs of EADRL are not significant, i.e. the 8 bytes addressed by EADRL are cleared in the same time normally needed to clear one byte ($t_{ROWERASE} = t_E = t_W$). For the following write modes, only the write and not the erase/write cycle is required. For example, using the row erase mode, programming 8 bytes takes $t_{TOTAL} = t_E + 8 \times t_W$ compared to $t_{TOTAL} = 8 \times t_E + 8 \times t_W$ ($t_E = t_{ERASE}$. $t_W = t_{WRITE}$).

Page write: For future products.

Page erase/write: For future products

Block erase: In this mode all 256 bytes are cleared. The byte containing the security bits is also cleared. $t_{BLOCKERASE} = t_E$. The contents of EADRH, EADRL and EDAT are insignificant.

Program Sequences and Register Contents after Reset

The contents of the EEPROM registers after a Reset are the default values:

EADRH	= 1xxxxxxB	(security bit address)
EADRL	= 00H	(security bit address)
ETIM	= 08H	(minimum erase time with the lowest permissible oscillator frequency)
ECNTRL	= 00H	(Byte mode, read)
EDAT	= xxH	(security bit)

```
Initialize:   MOV ETIM, ..
              MOV EADRH, ..

Read:         MOV EADRL, ..
              MOV ..., EDAT

Write:        MOV EADRL, ..
              MOV EDAT, ..

Erase row:    MOV EADRL, ..          Row address. 3LSBs don't care
              MOV ECNTRL, 0CH        Erase row mode
              MOV EDAT, ..  (EDAT) don't care

Erase block:  MOV ECNTRL, 0AH        Erase block mode
              MOVE EDAT, .. (EDAT) don't care
```

If the security bit is to be altered, the program generally starts as follows:

```
MOV EADRH, #80H
MOV EADRL, #00H
```

Figure 2. Control Register (ECNTRL)

EEPROM Protection
The EEPROM is protected using four security bits which are contained in an extra EEPROM byte at address 8000H (EADRH/EADRL). They can be set or cleared by software. To activate the EEPROM protection, the program sequence in byte mode must be as follows:

```
MOV  EADRH, #80H
MOV  EADRL, #00H
MOV  EDAT, #FFH
```

If two or more of these bit are reset, SB = 0, the security mode is disabled and the EEPROM is not protected. If three or four bits are set, SB = 1 and the $\overline{EA}$ mode differs from the internal access mode.

In this case, access to the EEPROM is only possible in one mode regardless of how the external access mode is reached (by pulling the $\overline{EA}$ pin low or by passing the 4K boundary). For SB = 1 and "external access" only, the "block erase" mode is enabled. The program sequence has to be as follows:

```
MOV  EADRH, #80H (security byte address)
MOV  EADRL, #00H (security byte address)
MOV  ECNTRL, 0AH (block erase mode)
MOV  EDAT, #xxH     (start block erase)
```

All 256 data bytes, the security bits, and SB will be cleared after completing this mode (EWP = 0). SB will also be affected in byte mode when writing to the security byte (not for SB = 1 and "external access"). Figure 3 illustrates the access to SB.

Security Facilities

ROM Code Protection
Since the external access mode can only be selected by pulling the $\overline{EA}$ pin low during reset, it is not possible to read the internal program memory using the MOVC instruction while executing external program memory. Furthermore, it is not possible to change this mode to internal access within the MOVC cycle.

Additionally, a mask-programmable ROM code protection facility is available. When the program memory passes the 4K boundary using both the internal and external ROMs, it is not possible to access the internal ROM from the external program memory if the mask-programmable ROM security bit is set. An access to the lower 4K bytes of program memory using the MOVC instruction is only possible while executing internal program memory.

Also the verification mode (test-mode which writes the ROM contents to a port for comparison with a reference code) is not implemented for security reasons. A different test-mode is implemented for test purpose. This mode allows every bit to be tested. However, the internal code cannot be accessed via a port.

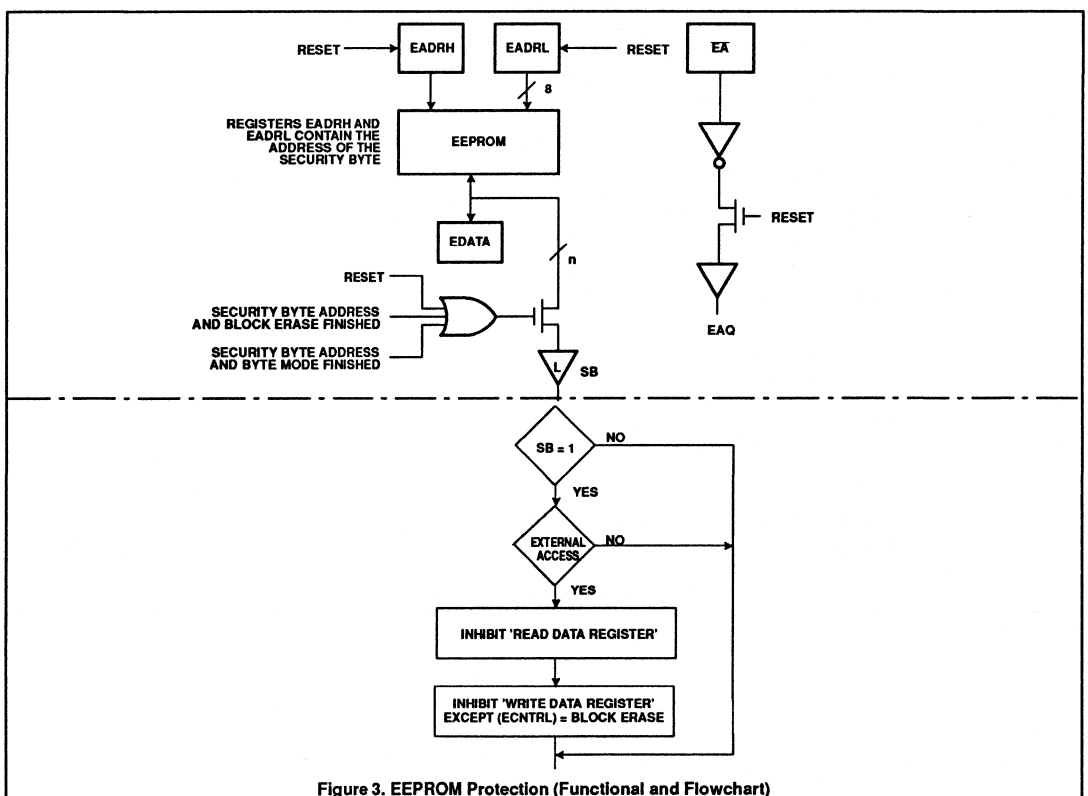


Figure 3. EEPROM Protection (Functional and Flowchart)

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol, page 1.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-up reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-up, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

Note: Before entering the idle or power-down modes, the user has to ensure that there is no EEPROM erase/write cycle in progress (i.e. the EWP bit has to be reset before activating the idle or power-down modes; otherwise EEPROM accesses will be aborted).

IDLE MODE

In idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM and EEPROM are preserved. A hardware reset is the only way to terminate the power-down mode. the control bits for the reduced power modes are in the special function register PCON. Table 2 shows the state of the I/O ports during low current operating modes.

INTERRUPT SYSTEM

External events and the real-time-driven on-chip peripherals require service by the CPU asynchronous to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution, a multiple-source, two-priority-level, nested interrupt system is provided.

Interrupt response latency is from 3µs to 7µs when using a 12MHz crystal. The S83C851 acknowledges interrupt requests from 7 sources as follows:

- $\overline{INT0}$ and $\overline{INT1}$: externally via pins 12 and 13, respectively
- Timer 0 and timer 1: from the two internal counters
- Serial port: from the internal serial I/O port or EEPROM (1 vector)

Each interrupt vectors to a separate location in program memory for its service program. Each source can be individually enabled (the EEPROM interrupt can only be enabled when the serial port interrupt is enabled) or disabled and can be programmed to a high or low priority level. All enabled sources can also be globally disabled or enabled. Both external interrupts can be programmed to be level-actiavted and are active low to allow "wire-ORing" of several interrupt sources to one input pin.

Note: The serial port and EEPROM interrupt flags must be cleared by software; all other flags are cleared by hardware.

Table 2. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Voltage on any other pin to V_{SS}	–0.5 to +6.5	V
Input or output DC current on any single I/O pin	±5	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.0	W

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or −40°C to +85°C, V_{SS} = 0V, V_{CC} = 5V ±10%

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{IL}	Input low voltage, except $\overline{EA}$		−0.5	$0.2V_{CC}$−0.1	V
V_{IL1}	Input low voltage to $\overline{EA}$		0	$0.2V_{CC}$−0.3	V
V_{IH}	Input high voltage, except XTAL1, RST		$0.2V_{CC}$+0.9	V_{CC}+0.5	V
V_{IH1}	Input high voltage, XTAL1, RST		$0.7V_{CC}$	V_{CC}+0.5	V
V_{OL}	Output low voltage, ports 1, 2, 3	I_{OL} = 1.6mA[1]		0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$	I_{OL} = 3.2mA[1]		0.45	V
V_{OH}	Output high voltage, ports 1, 2, 3	V_{CC} = 5V±10%, I_{OH} = −60µA, I_{OH} = −25µA I_{OH} = −10µA	 2.4 $0.75V_{CC}$ $0.9V_{CC}$		 V V V
V_{OH1}	Output high voltage (port 0 in external bus mode, ALE, $\overline{PSEN}$)[2]	V_{CC} = 5V±10%, I_{OH} = −400µA, I_{OH} = −150µA I_{OH} = −40µA	 2.4 $0.75V_{CC}$ $0.9V_{CC}$		 V V V
I_{IL}	Logical 0 input current, ports 1, 2, 3	V_{IN} = 0.45V		−50	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3	See note 3		−650	µA
I_{LI}	Input leakage current, port 0	$0.45V<V_{IN}<V_{CC}$		±10	µA
I_{CC}	Power supply current: Active mode @ 12MHz Idle mode @ 12MHz Power down mode	 See note 4 See note 5 See note 6 See note 7		 24 5 100	 mA mA µA
R_{RST}	Internal reset pull-down resistor		50	150	kohm
C_{IO}	Pin capacitance	f = 1MHz, t_A = +25°C		10	pF

NOTES:

1. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input..
2. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
3. Pins of ports 1, 2 and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
4. See Figures 11 through 14 for I_{CC} test conditions.
5. The operating supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f$ = 10ns; $V_{IL} = V_{SS}$ + 0.5V; $V_{IH} = V_{CC}$ − 0.5V; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = V_{CC}.
6. The idle mode supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f$ = 10ns; $V_{IL} = V_{SS}$ + 0.5V; $V_{IH} = V_{CC}$ − 0.5V; XTAL2 not connected; $\overline{EA}$ = Port 0 = V_{CC}; RST = V_{SS}.
7. The power-down current is measured with all output pins disconnected; XTAL2 not connected; $\overline{EA}$ = Port 0 = V_{CC}; RST = XTAL1 = V_{SS}.

I_{CC} (MAX) AS A FUNCTION OF f_{OSC} AND V_{CC}

f_{OSC} (MHz)	OPERATIONAL MODE (V_{CC})			IDLE MODE (V_{CC})		
	4.5	5.0	5.5	4.5	5.0	5.5
1.2	9.7	10.0	10.3	2.7	2.8	2.9
3.5	11.5	12.5	13.5	3.1	3.2	3.4
8.0	16.0	17.5	19.0	3.8	4.0	4.2
12.0	20.2	22.2	24.0	4.3	4.6	5.0

AC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C or −40°C to +85°C, V_{SS} = 0V[1, 2], V_{CC} = 5V ±10%

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	4	Oscillator frequency			1.2	12	MHz
t_{LHLL}	4	ALE pulse width	127		$2t_{CLCL}-40$		ns
t_{AVLL}	4	Address valid to ALE low	28		$t_{CLCL}-55$		ns
t_{LLAX}	4	Address hold after ALE low	48		$t_{CLCL}-35$		ns
t_{LLIV}	4	ALE low to valid instruction in		233		$4t_{CLCL}-100$	ns
t_{LLPL}	4	ALE low to $\overline{PSEN}$ low	43		$t_{CLCL}-40$		ns
t_{PLPH}	4	$\overline{PSEN}$ pulse width	205		$3t_{CLCL}-45$		ns
t_{PLIV}	4	$\overline{PSEN}$ low to valid instruction in		145		$3t_{CLCL}-105$	ns
t_{PXIX}	4	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	4	Input instruction float after $\overline{PSEN}$		59		$t_{CLCL}-25$	ns
t_{AVIV}	4	Address to valid instruction in		312		$5t_{CLCL}-105$	ns
t_{PLAZ}	4	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{RLRH}	5, 6	$\overline{RD}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{WLWH}	5, 6	$\overline{WR}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{RLDV}	5, 6	$\overline{RD}$ low to valid data in		252		$5t_{CLCL}-165$	ns
t_{RHDX}	5, 6	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	5, 6	Data float after $\overline{RD}$		97		$2t_{CLCL}-70$	ns
t_{LLDV}	5, 6	ALE low to valid data in		517		$8t_{CLCL}-150$	ns
t_{AVDV}	5, 6	Address to valid data in		585		$9t_{CLCL}-165$	ns
t_{LLWL}	5, 6	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AW}	5, 6	Address to $\overline{RD}$ or $\overline{WR}$	203		$4t_{CLCL}-130$		ns
t_{QW}	5, 6	Data setup time before $\overline{WR}$	433		$7t_{CLCL}-150$		ns
t_{QVWX}	5, 6	Data valid to $\overline{WR}$ transition	23		$t_{CLCL}-60$		ns
t_{WHQX}	5, 6	Data hold after $\overline{WR}$	33		$t_{CLCL}-50$		ns
t_{RLAZ}	5, 6	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	5, 6	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock							
t_{CHCX}	8	High time	20		20		ns
t_{CLCX}	8	Low time	20		20		ns
t_{CLCH}	8	Rise time		20		20	ns
t_{CHCL}	8	Fall time		20		20	ns
Erase/write timer constant[3]							
$t_{E/W}$		Erase/write cycle time	20	100	20	100	ms
t_E		Erase time	10	100	10	100	ms
t_W		Write time	10	100	10	100	ms
t_S		Data retention time[4]	10		10		years
NE/W		Erase/write cycles[5]	10,000		10,000		cycles

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. The power-off fall-time of V_{CC} must be less than 1ms to prevent an overwrite pulse from being generated in the EEPROM which can cause spurious parasitic writing to EEPROM cells. If the V_{CC} power-off full-time is greater than 1ms, a power-off reset signal should be generated to prevent this condition from occurring.
4. Test condition: t_A = +55°C
5. Number of erase/write cycles for each EEPROM byte.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

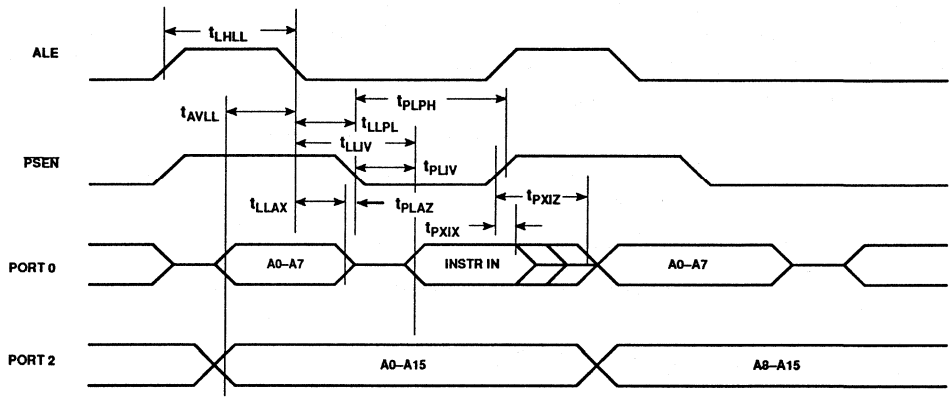


Figure 4. External Program Memory Read Cycle

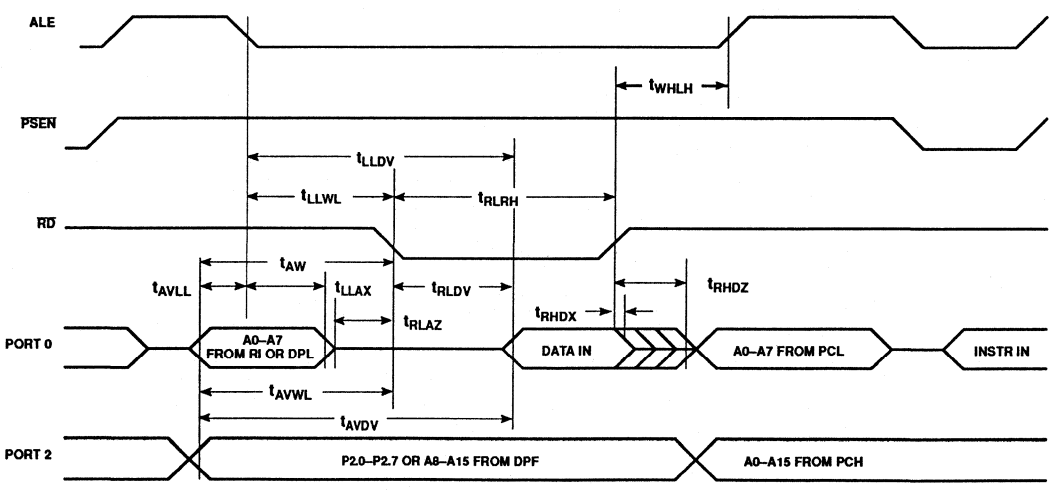


Figure 5. External Data Memory Read Cycle

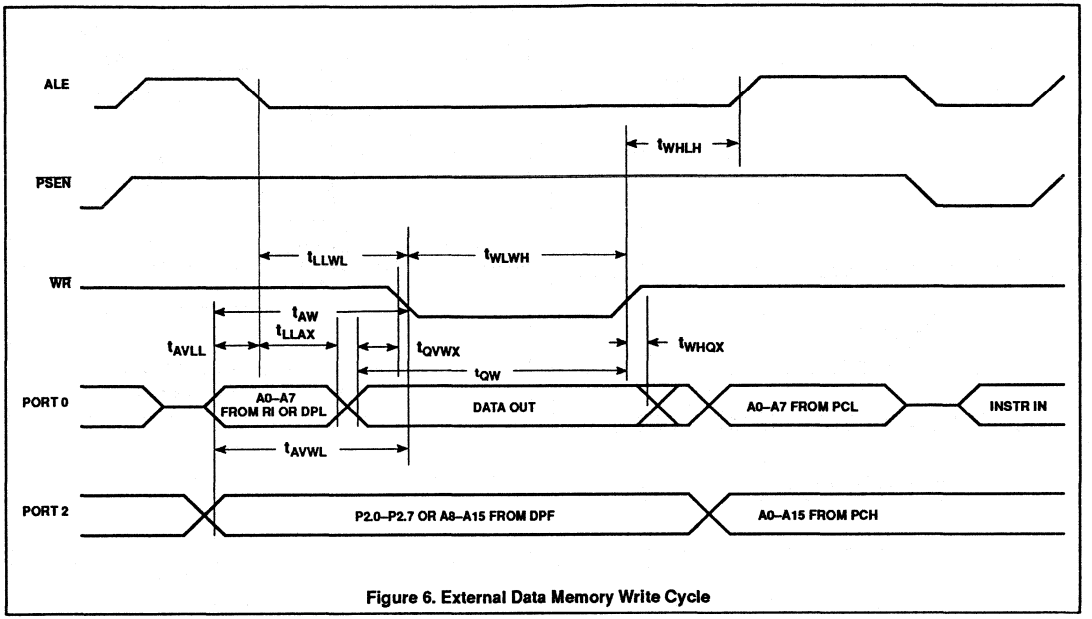


Figure 6. External Data Memory Write Cycle

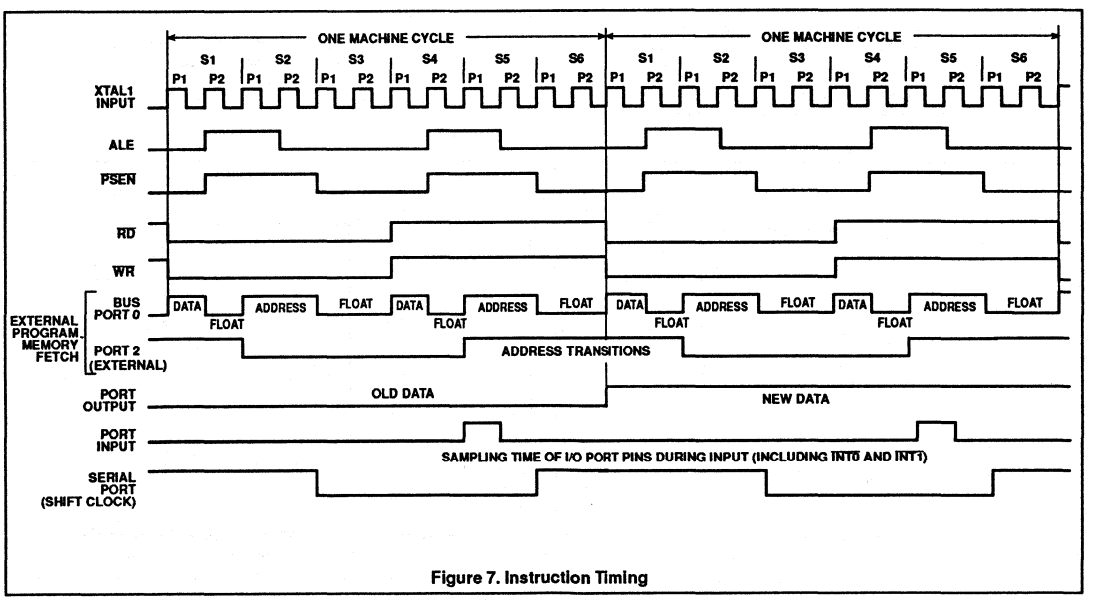


Figure 7. Instruction Timing

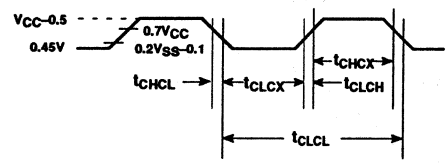
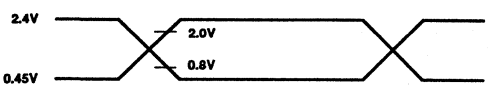
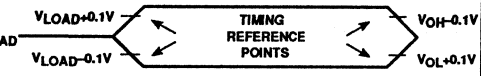


Figure 8. External Clock Drive



NOTE:
AC inputs during testing are driven at 2.4V for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at 2.0V min for a logic '1' and 0.8V for a logic '0'.

Figure 9. AC Testing Input/Output



NOTE:
For timing purposes, a port is no longer floating when a 100mV change from load voltage occurs, and begins to float when a 100mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OH}/I_{OL} \geq \pm 20$mA.

Figure 10. Float Waveform

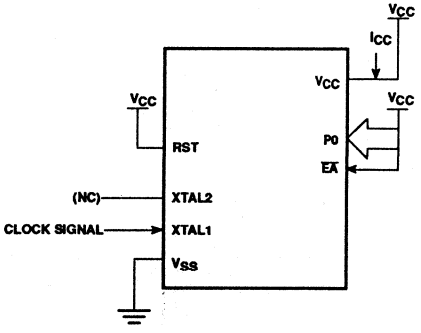


Figure 11. I_{CC} Test Condition, Active Mode
All other pins are disconnected

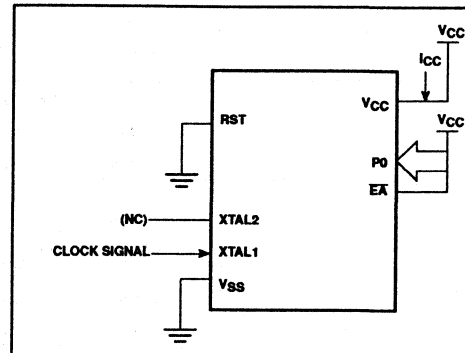


Figure 12. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

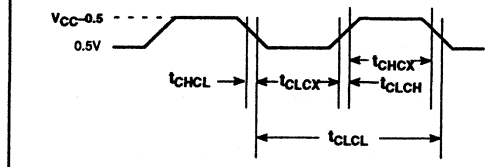


Figure 13. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 10$ns

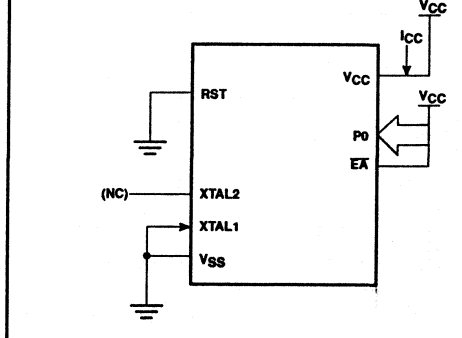


Figure 14. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected. V_{CC} = 2V to 5.5V

Section 4
Application Notes

CONTENTS

AN408	SC80C451 Operation of Port 6	463
AN417	256k Centronics Printer Buffer Using the SC87C451 Microcontroller	474
AN418	Counter/Timer 2 of the S83C552 Microcontroller	488
AN420	Using up to 5 External Interrupts on the 80C51 Family Microcontrollers	495
AN422	Using the 8XC751 Microcontroller as an I^2C Bus Master	497
AN423	Software Driven Serial Communication Routines for the 83C751 and 83C752 Microcontrollers	516
AN424	8051 Family Warm Boot Determinations	523
AN425	Interfacing PCD8584 I^2C Bus Controller to 80C51 Family Microcontrollers	525
AN426	Controlling Air Core Meters with the 87C751 and SAA5775	545

AN408
80C451 operation of port 6

INTRODUCTION

The features of the 80C451 are shared with the 80C51 or are conventional except for the operation of port 6. The flexibility of this port facilitates high–speed parallel data communications. This application note discusses the use of port 6 and is divided into the following sections:

1. Port 6 as a processor bus interface.
2. Using port 6 as a standard pseudo bidirectional I/O port.
3. Implementation of parallel printer ports.

This information applies to all versions of the part: 80C451, 83C451, and the 87C451.

PORT 6 AS A PROCESSOR BUS INTERFACE

Port 6 allows use of the 80C451 as an element on a microprocessor type bus. The host processor could be a general purpose MPU or the data bus of a microcontroller like the 80C451 itself. This feature allows single or multiple 80C451 controllers to be used on a bus as flexible peripheral processing elements. Applications could include keyboard scanners, serial I/O controllers, servo controllers, etc.

OPERATION

On reset, port 6 is programmed correctly for use as a bus interface (see Figure 2). This prevents the interface from disrupting data on the bus of the host processor during power–up. Software initialization of the CSR (Control Status Register) is not required. A dummy read of port 6 may be required to clear the IBF (Input Buffer Full) flag since it could be set by turn on transients on the bus of the host processor. On reset, the CSR of the 83C451 is programmed to allow the following:

1. AFLAG is an input controlling the port select function. If AFLAG is high, the contents of the CSR is output on port 6 when the port is read by the host. If AFLAG is low, then the contents of the output latch is output when port 6 is read by the host.
2. BFLAG is an input controlling the port enable function. In this mode when BFLAG is high, the input latch and the output drivers are disabled and the flags are not affected by the IDS (Input Data Strobe) or ODS (Output Data Strobe) signals. When BFLAG is low, the port is enabled for reading and writing under the control of IDS and ODS pins.

Figure 1 shows one possible example of an 80C451 on a memory bus. This arrangement allows the main processor to query port 6 for flag status without interrupting the 80C451. If the address decoder, shown in Figure 1, enables port 6 on the 80C451 when the address is 8000H or 8001H, and the address line A0 controls the port select feature, then the host processor can read and write to port 6 using address 8000H. Since the port select function is being controlled by the address line A0, the CSR contents can be read by the host processor at address 8001H.

By testing the CSR contents in this way, the host processor can tell if new data has been written to the port 6 output latch since it last read the port or if the 80C451 has read the last byte that the host wrote to the port. Conversely, the 80C451 can poll the flags in its CSR to see if the host processor has written to or read from port 6 since the last time it serviced the port.

If desired, an interrupt source for the 80C451 can be derived easily from the port enable source as shown by the dashed line in Figure 1.

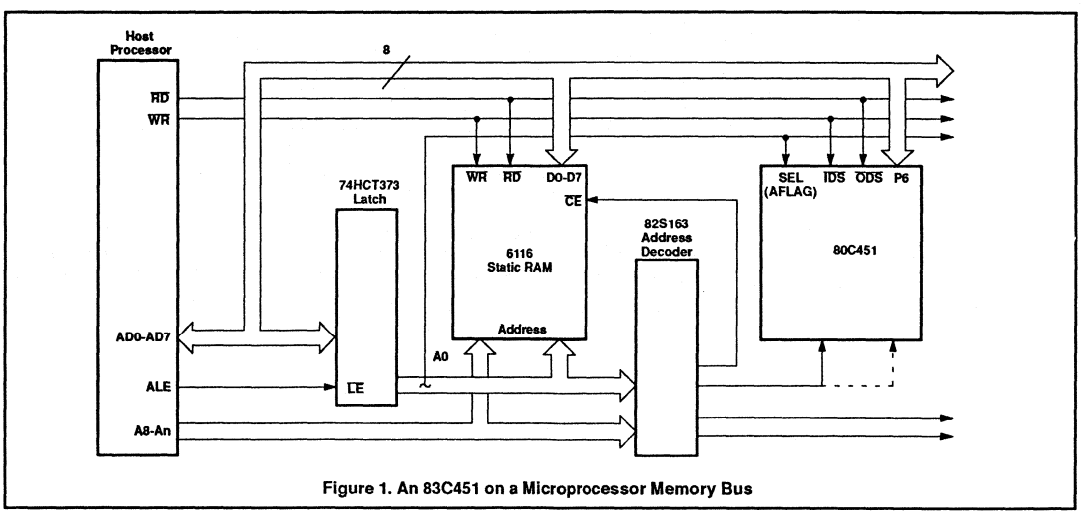


Figure 1. An 83C451 on a Microprocessor Memory Bus

80C451 operation of port 6 AN408

SOFTWARE EXAMPLES

To write to port 6 on the bus shown in Figure 1, the host processor first reads the CSR contents at address 8001H, and tests the input buffer full flag (CSR bit 0). If the flag is clear, the host writes a byte to address 8000H. This loads the input buffer latch of port 6 and sets the input buffer full flag.

Conversely, the 80C451 polls the IBF flag and reads a byte from port 6 when it finds the flag set. The flag is automatically reset when this internal read occurs.

80C451 ROUTINE TO READ ONE BYTE FROM HOST VIA PORT 6

```
RCVR:   JNB CSR.0,RCVR   ;TEST IBF FLAG
        MOV A,P6         ;WHEN FLAG IS SET READ BYTE
        RET
```

80C451 ROUTINE TO WRITE ONE BYTE TO THE 83C451 PORT 6

If the host processor is an 80C51, the following routine will write a byte of data to the 80C451. The data involved is passed to the routine through register 1.

```
XMIT:   MOV DPTR,8001H
TEST:   MOVX A,@DPTR     ;READ THE CSR
        JB ACC.0,TEST    ;TEST IBF FLAG
        MOV DPTR,8000H
        MOV A,R1
        MOVX @DPTR,A     ;WRITE DATA TO THE 451
        RET
```

80C451 ROUTINE TO WRITE ONE BYTE TO HOST VIA PORT 6

Routines for data transfer in the opposite direction are similar to the above two. The 80C451 version is given below.

```
XMIT:   JB CSR.1,XMIT    ;TEST OBF FLAG
        MOV P6,A         ;WRITE DATA
        RET
```

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
1	1	1	1	1	1		

Figure 2. CSR Programmed to Allow Port 6 as a Bus Interface

USING PORT 6 AS A STANDARD QUASI-BIDIRECTIONAL I/O PORT

To use port 6 as a common I/O port, all of the control pins are tied to ground (see Figure 3). On hardware reset, bits 2 - 7 in the CSR are set to one. Port operation and electrical characteristics become identical to port 1 on the 80C51 and the 80C451 ports 1, 4, and 5. No software initialization is required.

If desired, AFLAG and BFLAG can be used as outputs while port 6 is operating as a standard quasi-bidirectional I/O port (see Figure 4). In this case, only $\overline{IDS}$ and $\overline{ODS}$ are tied to ground and the CSR is initialized to allow operation of AFLAG and BFLAG as simple outputs (see Figure 5).

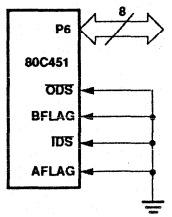


Figure 3. Standard I/O Port on Reset

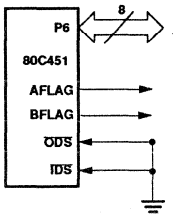


Figure 4. Standard I/O Port on Reset with AFLAG and BFLAG as Outputs

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
1	X	0	X	X	1		

Figure 5. CSR Programmed to Allow AFLAG and BFLAG to Operate as Outputs and Port 6 as a Standard I/O Port

DATA TRANSFER SIGNAL PINS		
Pin No.	Groung Return Pin No.	Signal
1	19	STROBE
2	20	DATA 1
3	21	DATA 2
4	22	DATA 3
5	23	DATA 4
6	24	DATA 5
7	25	DATA 6
8	26	DATA 7
9	27	DATA 8
10	28	ACKNLG
11	29	BUSY

TYPICAL AUXILIARY PIN FUNCTIONS	
Pin No.	Signal
12	PAPER OUT
14	AUTO LINE FEED
16	LOGIC GROUND
17	CHASSIS GND
30	GROUND RETURN
31	RESET PRINTER
32	ERROR
33	GROUND RETURN
36	SLCTIN

Figure 6. Parallel Printer Interface Pin Functions

IMPLEMENTATION OF PARALLEL PRINTER PORTS USING PORT 6

The 80C451 is an excellent choice for a printer controller. The 80C451 has the facilities to permit all of the intelligent features of a common printer to be handled by a single chip:

1. The features of port 6 allow a parallel printer port to be designed with only line driving and receiving chips required as additional hardware.
2. The onboard UART allows RS232 interfacing with only level shifting chips added.
3. The 8-bit parallel ports 0 to 6 are ample to drive onboard control functions, even when ports are used for external memory access, interrupts, and other functions.
4 The RAM addressing ability of ports 0 and 2 can be used to address up to 64k bytes of a hardware buffer/spooler. AFLAG and BFLAG as simple outputs (see Figure 5).
5. The 64k byte ROM addressing capability allows space for the most sophisticated software.

In addition, either end of a parallel interface can be implemented using port 6, and the interfaces can be interrupt driven or polled in either case.

THE INTERFACE

Data transfer on a parallel printer interface occurs across eleven signal lines. The other conductors on the standard plug are used as ground returns or for auxiliary functions (see Figure 6). Only the data transfer signals will be considered.

The Data Transfer Format

The parallel printer interfaces are far more standardized in features than their serial counterpart. However, at least three significant variations exist in handshake style in printers using generic parallel interfaces. This fact influences the design of both port hardware and software. A good transmitter should be able to drive devices with all three styles of handshakes, and a good receiver should generate the handshake most likely compatible with any transmitter.

The Variations

Type 1—Figure 7 shows a common style of handshake and is the style that will be implemented in the receiver examples. A busy signal and an acknowledge strobe pulse are generated for every byte received.

Type 2—Another style of handshake generates a busy signal only when the printer will not be able to accept more data for a relatively long time. Acknowledge pulses are created after every byte received. When the busy signal is generated after a byte is received, the associated acknowledge pulse does not occur until *after* the busy signal returns to logic zero (see Figure 7).

Type 3—A third handshake style does not generate acknowledge pulses, but a busy signal is produced after every byte is received.

PARALLEL PRINTER INTERFACES USING POLLING

Transmitter Operation

This application illustrates the flexibility of the port 6 logic in solving an applications problem. We need to be able to handle all types of acknowledge signals that might be received by the transmitter. We will use the $\overline{ODS}$ pin and output buffer full flag logic to record the receipt of the acknowledge pulse (see Figure 8), but not all parallel receivers generate acknowledge pulses. We could poll the busy signal line, but not all receivers generate busy signals for each byte received; so lack of a busy signal does not imply that we can send another byte. We can, however, expect an acknowledge pulse very shortly after the end of a busy signal if one is going to arrive at all. So we can send a new data byte after having received either a positive transition on the acknowledge line, or shortly after receiving a negative edge on the busy line.

The CSR is programmed to the output only mode. In this mode, the $\overline{ODS}$ pin does not control the output drivers but only the output buffer full flag. The flag serves to record the positive transition of the acknowledge signal. The input latch is not used, but the $\overline{IDS}$ pin is used to set the input buffer full flag. This is used to record the negative transition at the end of the busy signal. Dummy reads by the 80C451 of port 6 will be used to clear the flag. In this example, the AFLAG mode is set only to place the port in the output only mode. The AFLAG pin is not actually used (see Figure 10).

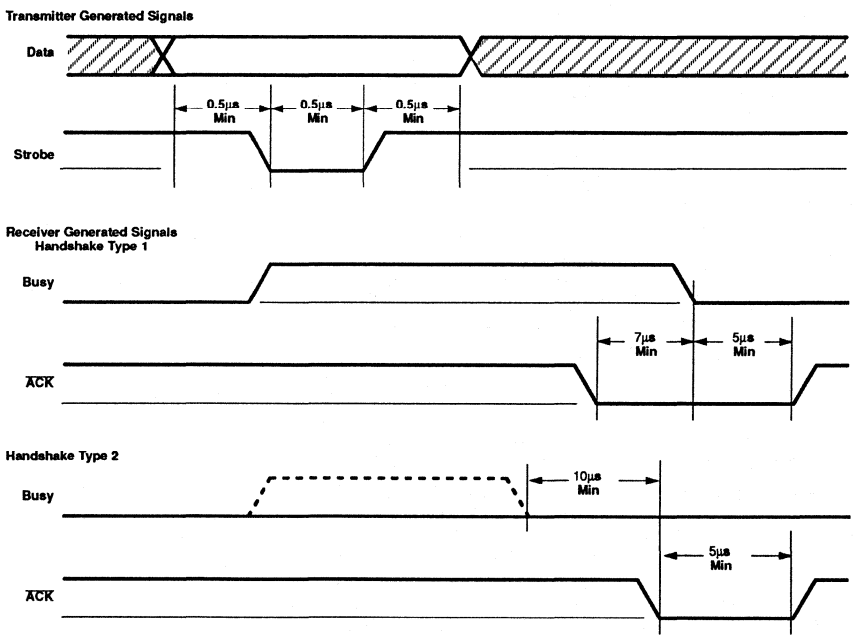


Figure 7. Parallel Printer Interface Signals

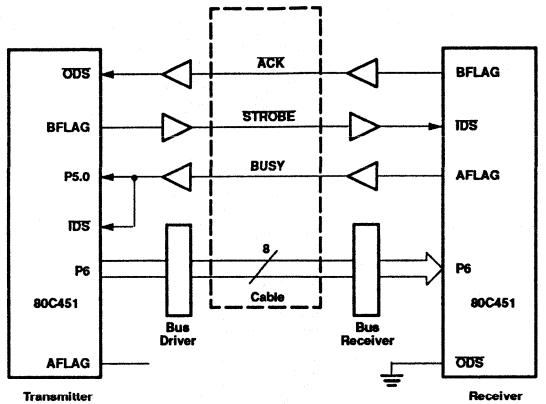


Figure 8. Interconnection for a Parallel Interface Using Polling

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
0	1	1	0	0	1		

Figure 9. CSR Programmed for Polled Transmitter Operation

The transmitter's CSR (control status register) is programmed to the following mode (see Figure 9):

1. CSR bit 6 controls the BFLAG output and therefore the strobe line.
2. The OBF (output buffer full) flag controls the AFLAG output.
3. The OBF is cleared on the positive edge of the ODS input.
4. The IBF flag is cleared on the negative edge of the IDS strobe.

NOTE:
With this combination of modes set, port 6 is in the output only mode.

Receiver Operation

In receiver operation, the IDS input is used to latch in the data transmitted on receipt of the strobe pulse. The receiver's CSR is programmed to allow the following (see Figure 11):

1. The input buffer full flag is output through the BFLAG pin and is used as the busy signal to the transmitter.
2. The IBF flag is set and data is latched on the positive edge of IDS.
3. Writing to the CSR bit 4 controls the AFLAG output and therefore the acknowledge line.

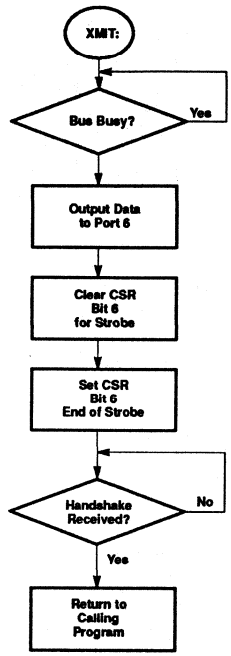


Figure 10. Flow Chart of Polled Parallel Transmitter Operation

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
1	0	0	1	1	0		

Figure 11. CSR Programmed for Polled Parallel Receiver Operation

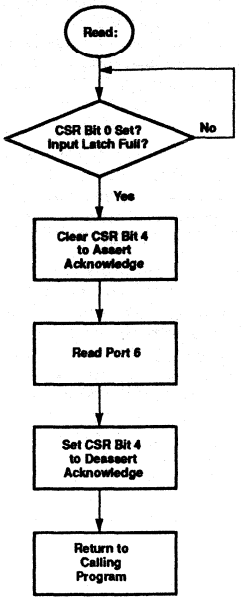


Figure 12. Flow Chart of Polled Parallel Receiver Operation

SOFTWARE EXAMPLES

This polled parallel transmit routine outputs one byte passed to it in the accumulator.

```
P_INIT:   MOV CSR,#064H     ;INITIALIZE PORT 6 OPERATING MODE
P_OUT:    JB P5.0           ;WAIT IF BUSY SIGNAL IS HIGH
          MOV P6,ACC        ;OUTPUT DATA
          MOV R1,P6         ;DUMMY READ TO CLEAR IBF FLAG
          MOV R1,#02H       ;INITIALIZE DELAY COUNTER
          CLEAR CSR.6       ;START STROBE PULSE
          DJNZ R1,$         ;TIME 6 MICROSECOND STROBE PULSE
          SETB CSR.6        ;END STROBE PULSE
WAIT:     JNB CSR.1,OUT     ;EXIT IF ACKNOWLEDGE RCV'D
          JNB CSR.0,WAIT    ;EXIT IF NEGATIVE BUSY EDGE RCV'D
```

This polled parallel receive routine places one byte in the accumulator each time it is called.

```
P_INIT:   MOV CSR,#09CH     ;INITIALIZE PORT 6 OPERATING MODE
          MOV R7,P6         ;DUMMY READ TO CLEAR IBF FLAG
P_IN      JNB CSR.0         ;INPUT BUFFER LATCH FULL?
          CLR CSR.4         ;BEGIN ACKNOWLEDGE PULSE
          MOV R7,#02H       ;INITIALIZE DELAY COUNTER
          DJNZ R7,$         ;TIME ACKNOWLEDGE PULSE
          MOV A,P6          ;READ BYTE – CLEAR BUSY SIGNAL
          MOV R7,#02H       ;INITIALIZE DELAY COUNTER
          DJNZ R7,$         ;TIME ACKNOWLEDGE PULSE
          SETB CSR.4        ;END ACKNOWLEDGE PULSE
          RET
```

INTERRUPT DRIVEN PARALLEL PRINTER INTERFACE (See Figure 13)

Transmitter Operation

The transmitter's CSR (control status register) is programmed to the following mode (see Figure 14):

1. CSR bit 6 controls the BFLAG output and therefore the strobe line.
2. The OBF (output buffer full) flag controls the AFLAG output.
3. The OBF is cleared on the positive edge of the $\overline{ODS}$ (output data strobe) input.
4. The IBF flag is set on the negative edge of the $\overline{IDS}$ (input data strobe) pin.

NOTE:
With this combination of AFLAG and BFLAG modes set, port 6 is in the output only mode. The output drivers are always enabled and the $\overline{ODS}$ input is only used to clear the OBF flag.

INTO is programmed to be negative edge sensitive and is connected to the OBF flag through the AFLAG pin. The OBF is cleared on the positive edge of $\overline{ODS}$. The net result is that INTO is triggered on the end of the ACK pulse (a positive edge). This signals the transmitter that another byte may be transmitted. The transmitting 83C451 is free to do other tasks prior to this interrupt.

In this routine, Figure 15, the main program establishes a buffer in data memory ended by an ASCII end of text character. To begin outputting the buffer, the routine PSEND is called. The rest of the buffer is emptied by the interrupt vectors to PSEND1.

For printers which generate acknowledge pulses, output rates of 25k transfers per second are achieved. Timer generated interrupts are used to periodically return program execution to the routine to service non-acknowledging printers and to provide a timeout feature. Non-acknowledging printers are serviced at a rate of about 2.5k transfers per second. This maximum rate may be varied by adjusting the timer reload value. As written, the time out procedure attempts to retransmit a byte when the printer has not acknowledged for an excessively long time.

Receiver Operation

In receiver operation, the IDS input is used to latch in the data transmitted on receipt of the strobe pulse. The receiver's CSR is programmed to allow the following (see Figure 16):

1. The input buffer full flag is output through the BFLAG pin and is used as the busy signal to the transmitter. The IBF flag is set and data is latched on the positive edge of $\overline{IDS}$.
2. Writing to the CSR bit 4 controls the AFLAG output and therefore the acknowledge line.

The receiver is interrupted on the negative edge of the data strobe. Data is latched in on the positive edge of the strobe pulse (see Figure 17). Since the strobe pulse is normally very short, there is little time lost between receiving the interrupt and having valid data in the input latch. The receiver is free to do other tasks prior to receiving the $\overline{INTO}$ interrupt.

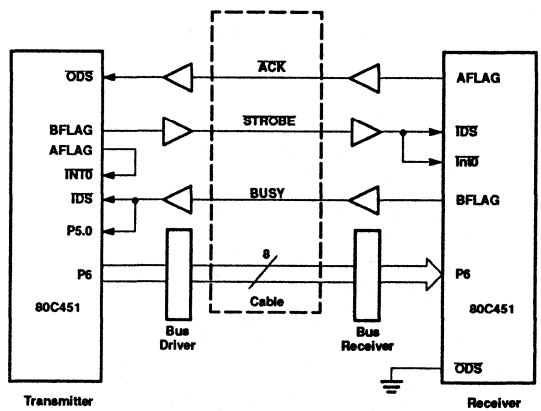


Figure 13. Interrupt Driven Parallel Interfaces Using 80C451 Controllers

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
0	1	1	0	1	1		

Figure 14. CSR Programmed for Use as an Interrupt Driven Parallel Transmitter

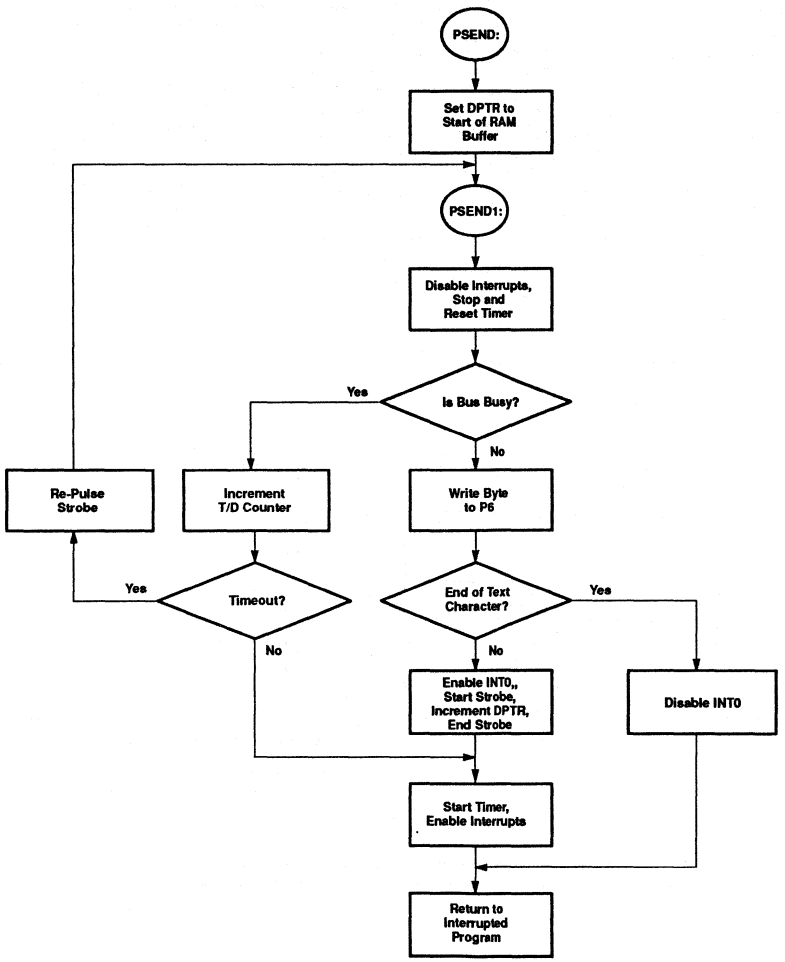


Figure 15. Flow Chart for an Interrupt Driven Parallel Transmitter

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
1	0	0	1	1	0		

Figure 16. CSR Programmed for Use as an Interrupt Driven Parallel Receiver

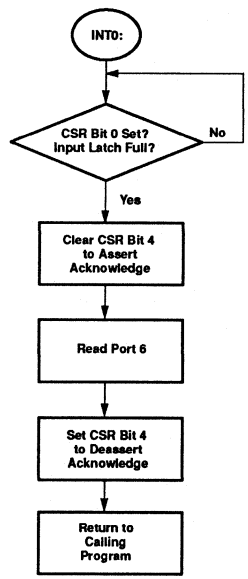


Figure 17. Flow Chart of Interrupt Driven Parallel Receiver Operation

SOFTWARE EXAMPLES

The software for the interrupt driven parallel receiver is similar to the polled receiver example. However, after an interrupt is received, this routine checks to confirm that data has been latched by the positive edge of the strobe pulse before proceeding with the routine.

```
INIT:       MOV CSR,#090H     ;INITIALIZE CSR
            SETB EX0          ;ENABLE INTERRUPT 0
            SETB IT0          ;SET NEG EDGE TRIGGERED INTERRUPTS
            SETB EA           ;ENABLE ALL INTERRUPTS
ORG EXTI0                     ;INTERRUPT 0 VECTOR
            JMP RCVR

            RCVR:
RCVR:       JNB CSR.0,#       ;CONFIRM DATA LATCHED
            CLR CSR.4         ;START ACKNOWLEDGE PULSE
            MOV R7,#02H       ;INITIALIZE THE DELAY COUNTER
            DJNZ R7,#         ;TIME ACK PULSE
            MOV A,P6          ;READ BYTE – RESET BUSY LINE
            MOV R7,#02H       ;INITIALIZE THE DELAY COUNTER
            DJNZ R7,$         ;TIME ACK PULSE
            SETB CSR.4        ;END ACK PULSE
            RET1
```

This is the software for the interrupt driven parallel transmitter example.

```
; XMIT ROUTINE DRIVEN BY ACK PULSE GENERATED INTERRUPTS, OR TIME GENERATED INTERRUPTS
; FOR NON ACKNOWLEDGING PRINTERS. READS DATA BUFFER IN EXTERNAL RAM STARTING AT 100H
; AND READING UNTIL 04H IS FOUND.

        ORG  RESET
        JMP  26H
        ORG  TIMER0
        JMP  PSEND1
        ORG  EXTI0
        JMP  PSEND1
        ORG  26H
             MOV CSR,#064H              ;PORT 6 MODE
             MOV TMOD,#002H             ;CONFIGURE TIMER 0 TO 16 BITS
             SETB TO0                   ;INT0 IS EDGE TRIGGERED
             SETB EA                    ;ENABLE INTERRUPTS
PSEND:       MOV DPTR,#0100H            ;SET DPTR TO START OF TEXT
                                        ;BUFFER
PSEND1:      CLR EA                     ;DISABLE INTERRUPTS AND STOP
                                        ;TIMER
             CLR TR0                    ;IF ENABLED
             CLR ET0
             MOV R7,00H                 ;CLEAR TIMEOUT COUNTER
             MOV R6,00H
             MOV TH0,#–4                ;SET TIMER INTERRUPT PERIOD
             MOV TL0,#00H
             JB 0C8H,BB                 ;BUS BUSY
             MOV ACC,#00H               ;CLEAR ACCUMULATOR
             MOVX 1,@DPTR               ;RETRIEVE FIRST BYTE
             MOV 06,ACC                 ;OUTPUT FIRST BYTE
             CJNE A,#004H, CONT1        ;LOOK FOR END OF TEXT
             JMP EOTB
CONT1:       SETB ERX0                  ;ENABLE INTO
             CKR 0EEH                   ;START STROBE PULSE
             INC DPTR
             MOV ACC,DPH                ;LOOK FOR PHYSICAL END OF
             JB ACC.2,EOTB              ;TEXT BUFFER
             SETB 0EEH
             JMP CONT
EOTB:        CLR EX0                    ;END OF TEXT FOUND, DISABLE
                                        ;INT0
             SETB 0EEH
             SETB EA
             RETI
BB:          INC R7                     ;COUNT TIMER TIMEOUTS ON
                                        ;BUS BUSY
             CJNE R7,#00H, CONT         ;LOOK FOR OVERFLOW
             INC R6                     ;COUNT OVERFLOWS
             CJNE R6,#10H, CONT         ;TIMEOUT APPROX 5 SEC
             JMP TO
CONT:        SETB TR0                   ;ENABLE TIMER INTERRUPT
             SETB ET0                   ;START TIMER
             SETB EA
             RETI
TO:          CLR 0C9H                   ;SEND NEW STROBE PULSE IN
                                        ;RESPONSE TO TIMEOUT
             NOP
             NOP
             MOV R6,#00H                ;RESET TO COUNTER
             MOV R7,#00H
             SETB 0C9H                  ;END OF STROBE PULSE
             JMP PSEND1
```

AN417

256k centronics printer buffer using the 87C451 microcontroller

DESCRIPTION

This application note describes a stand alone Centronics type parallel printer buffer using the 87C451 expanded I/O microcontroller. This type of unit would typically be placed between a personal computer and its printer. It captures the data to be printed at high speed, freeing the personal computer to go to other tasks, and sends data to the printer as required. As described here, 256k dynamic RAMs are used, providing over one quarter million characters of storage. If desired the design is easily modified to work with 1 megabit DRAMs. Although written with the 87C451 in mind, this design is applicable to the 80C451 and 83C451.

Design Objectives

The objectives kept in mind during the design of this device were: provide a substantial size of buffer, keep the parts count and the power consumption to a minimum, and use readily available components.

A buffer size of 256k bytes was chosen because, although a 64k byte buffer is very easily implemented using the 8051 family's 64k external data storage capabilities, it is a little too small for today's printing applications that print a page of text in graphics mode, using up twenty times as many bytes as standard printing mode. Presenting a method for controlling 256k DRAMs shows off the I/O capabilities of the 87C451, and it is very easy to add the extra address line for one megabit devices if a larger buffer is needed.

The 8XC451 Microcontroller

The 8XC451 is an 8-bit microcontroller based on the familiar 8051 family of devices. In fact, it is an 80C51 with three added ports: P4, P5, and P6. Ports 4 and 5 give 12 (16 in PLCC) additional quasi-bidirectional I/O lines. Port 6 provides another 8 bits of I/O, plus 4 handshake lines that can be programmed to operate in several useful modes for interfacing. The *XC451 comes in three versions: ROMless 80C451, 83C451 with 4k x 8 ROM, and 87C451 with 4k x 8 EPROM.

In this note, port 6 is used in the I/O mode as a Centronics compatible printer output port. Additionally, the /IDS and BFLAG pins normally associated with port 6 are used as part of the input port logic. For a complete discussion of port 6 operating modes and programming, see the application note AN408 titled "83C451 Microcontroller Operation of Port 6."

Circuit Description

Figure 1 is a schematic diagram of the printer buffer circuit. Other than the 87C451 (U1), and the eight 256k DRAMs (U5-U12), only two 74LS244 buffers (U2, U3) and a 76HCT374 (U4) octal flip-flop are needed. The U2 and U3 buffers are included to provide full drive capability for the output port and some of the handshake signals on the input port, as the output buffers on the 87C451 can only drive 3 LSTTL loads. U4 has 8-bit data strobed into it by the /STB pulse of the input port.

As the code size for this application is quite small (less than 1k bytes), the on-chip instruction memory is quite sufficient for program storage. For a production version, the 87C451 could be replaced with the 83C451 with a 4k x 8 masked ROM on chip. Note that port 0 and port 1 are not used in the present design; thus the 80C451 may be used in this application with the addition of an external address latch and EPROM.

The /RAS, /CAS, and /WR signals for the DRAM array are provided by port 3 bits /WR, /RD, and T1. Note that as in the 80C51, all port 3 signals are multifunctional. That is, each can be treated as a regular quasi-bidirectional port bit, or as having the special function indicated by its name. This feature is an advantage when using /WR and /RD as /RAS and /CAS control signals for a DRAM array. Treated as a normal port bit, the /WR pin is cleared and set by individual CLR and SETB instructions for a normal length RAM read or write cycle. However, when performing a refresh cycle, /RAS (port 3/WR) can be pulsed low using a dummy MOVX @R0,A (move to external data memory) instruction. This allows DRAM refresh to be done much more quickly than would otherwise be possible.

Port 1 and one bit from port 4 form the 9-bit address required when addressing the DRAM array. The data inputs to the array come from the parallel input data lines which are latched by U4. The RAM data outputs are fed to port 5. By making the data outputs available to the processor, it is possible to add some additional features to the firmware, such as control codes for printing multiple copies of a document, data compression, data conversion, etc. which are not implemented in this design.

Port 6 Operation

The /IDS (input data strobe) and BFLAG pins are normally used in conjunction with the port 6 bidirectional mode. In this mode, the /IDS pin is used to strobe data into the port 6 input latches, and BFLAG is used as flag output. In this application, however, these two bits are used to good effect as part of the (separate) input port logic. When a byte of data is strobed into U4 by the printer port of the host computer, the /STB signal connected to /IDS sets the input buffer full flag (IBF). BFLAG is programmed to mirror the contents of IBF, and therefore becomes asserted. This makes it ideal to be used as the BUSY output for the input port. After the input port data has been read and stored in the RAM buffer, BFLAG is de-asserted by performing a dummy read of port 6, which clears IBF. To complete the input port logic, one of the port 3 pins, P3.4, is used as the acknowledge signal, and is asserted/de-asserted by software. The /ODS pin is tied to ground to permanently

enable the port 6 output drivers. This does not cause difficulty as no data is being input into the port.

Note that programming port 6 to operate in the bidirectional mode as described above means the loss of /ODS as an acknowledge input. The acknowledge input is normally used to clear the OBF (output buffer full) flag, indicating that the printer is ready for another character. On the other hand, operating port 6 in the "output only" mode causes the loss of BFLAG as BUSY output. Because the input port requires an instant BUSY indication while the output port only needs to remember the occurrence of an acknowledge pulse, it makes sense to program port 6 to operate in the bidirectional mode, with /ODS grounded to enable the output drivers. The /INT1 pin can be used instead of /ODS to record the occurrence of an acknowledge pulse with the interrupt system.

Priority and Execution of Tasks

There are three tasks that must be performed in this system: Receive—servicing the input port and storing the input character; Transmit—sending stored characters to the output port as required; and Refresh—performing DRAM refresh. The timers and interrupt system are used to manage the execution and priority of these tasks. Figures 2 and 3 illustrate the flow charts of these tasks. Firmware, broken into sections, performing these three frunctions as well as an initialization routine is provided.

The 51C256 DRAMs require a 256 row refresh every 4 milliseconds. Rather than do an entire refresh cycle every 4 milliseconds, it is done as 64 rows every millisecond. This leaves time for other tasks to get service "slices" more frequently. As DRAM refresh is obviously the highest priority, timer 0 is used as the refresh interval timer, and is programmed to the 16-bit mode, and set to the higher priority level in the interrupt priority (IP) register. The refresh code is written in-line rather than in a loop to maximize speed.

An interesting point to note is that when there are no characters stored, the DRAM does not need to be refreshed. If power consumption is of concern, the 87C451 could be programmed to go into idle mode whenever the buffer were empty. A character strobed into the input port would cause an interrupt, restarting the 87C451; DRAM refresh would be maintained until the buffer was once again empty.

The next highest priority should be input port service, as the reason for having a printer buffer is to get the data out of the computer as quickly as possible. Therefore, the input port /STB signal is connected to the /INT0 pin (as well as U4's clock pin and /IDS). Interrupt 0 is programmed in the interrupt priority register to be at the lower interrupt level so it cannot prevent refresh service. The interrupt 0 service routine stores the input character at the next location in the DRAM array, using the technique of a circular FIFO buffer. The routine also sends back an acknowledge pulse by clearing and setting the P3.4 pin, and then clears the BUSY (BFLAG) pin by performing a dummy read of port 6 (unless this character caused the buffer to be completely full).

During periods of access to the DRAM array by the input and output routines, the global interrupt enable bit (EA) is cleared so that the refresh interrupt does not disturb the contents of ports 1 and 4, or the /RAS, /CAS, and /WR signals.

The printer (output port) service routine runs all the time, except when the CPU is called to service the other conditions, therefore having the lowest priority. If there are characters in the buffer, polling is used to check for output port BUSY status. If the printer is not busy, then the character is sent, and the output port /STB pin (P4.3) is cleared and set. The output port /ACK line is connected to the /INT1 pin, so that the negative going edge of the /ACK signal is recorded as an interrupt pending. A very short INT1 service routine sets a software flag to indicate that the printer acknowledge the last character.

Possible Enhancements

There are a number of features that could be added to this design. As mentioned previously, the microcontroller could be put into the idle mode when the buffer is empty, conserving power.

The software could be enhanced to provide features such as multiple copies of a document, data compression, data conversion, automatic printer setup, etc. The PC operating system could be suitably modified to send a header for each file to be printed, containing these parameters. There is plenty of room for operating firmware expansion, and plenty of horsepower left in the 87C451 to handle these features.

The two serial port pins RxD and TxD were deliberately left unused so that input and/or output ports are easily implemented for serial interfaces or printers using the built-in UART. The pins used for parallel port handshaking could then be used as serial handshaking lines, providing the standard "modem" signals.

Combining the above two features, this circuit could act as a "splitter." By connecting a daisy-wheel printer to the serial port, a dot-matrix printer to the parallel port, and sending an "address" flag in the file header, simultaneous letter-quality and draft printing could be done.

The size of the DRAM array is easily expanded to one megabyte or large devices by connecting the additional address pins to port 4 bits 1 and 2. Only slight modifications to the operating firmware would be required.

Conclusion

The SC8XC451 microcontrollers provide plenty of I/O pins that previously had to be implemented by clumsy I/O expansion methods. The flexibility of port 6 means that this device can be used in a wide variety of applications requiring special port functions, while still using the industry standard 8051 instruction set.

The Application Note, describing a typical parallel printer buffer, makes full use of the 8XC451 features, yet allows room for enhancement and expansion.

256k centronics printer buffer AN417

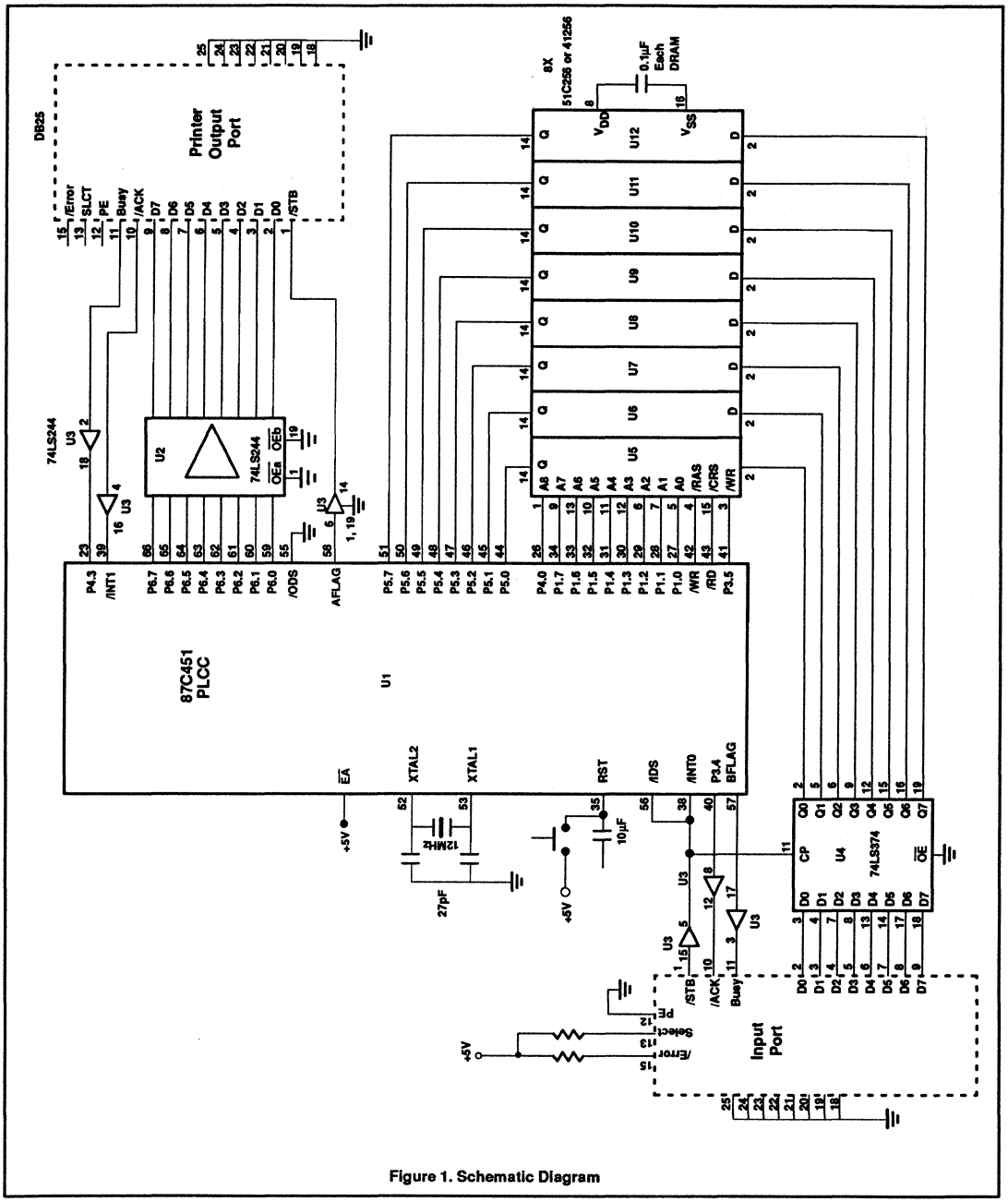


Figure 1. Schematic Diagram

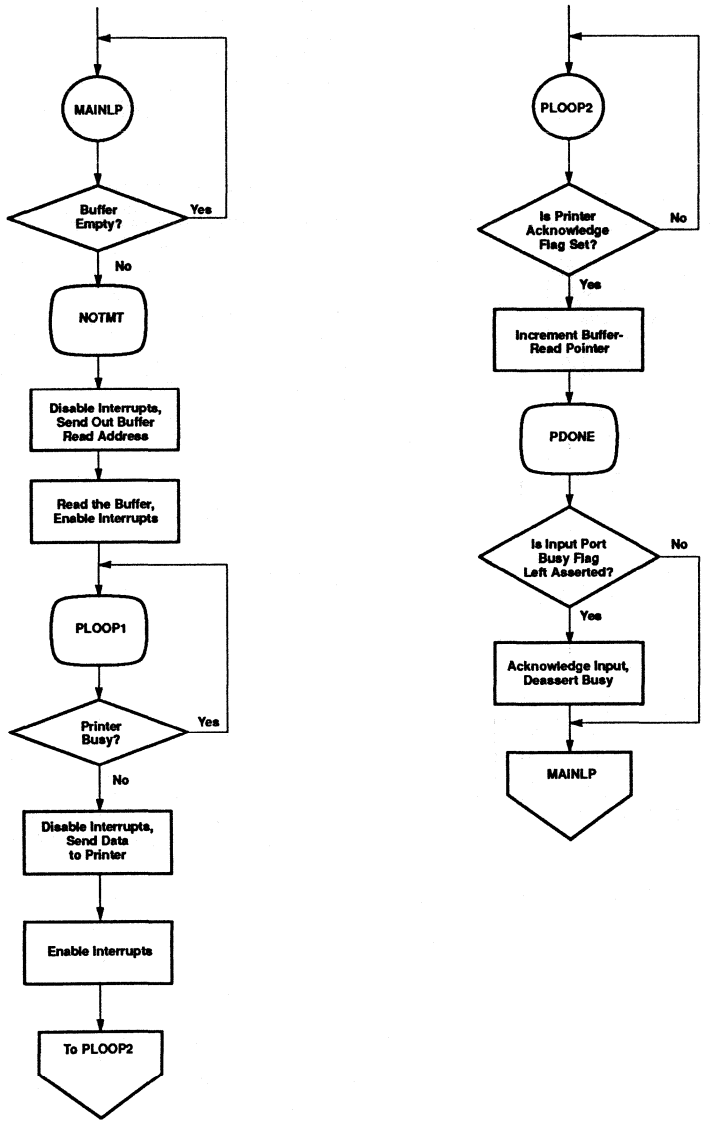


Figure 2. Flowchart of Transmit Operation

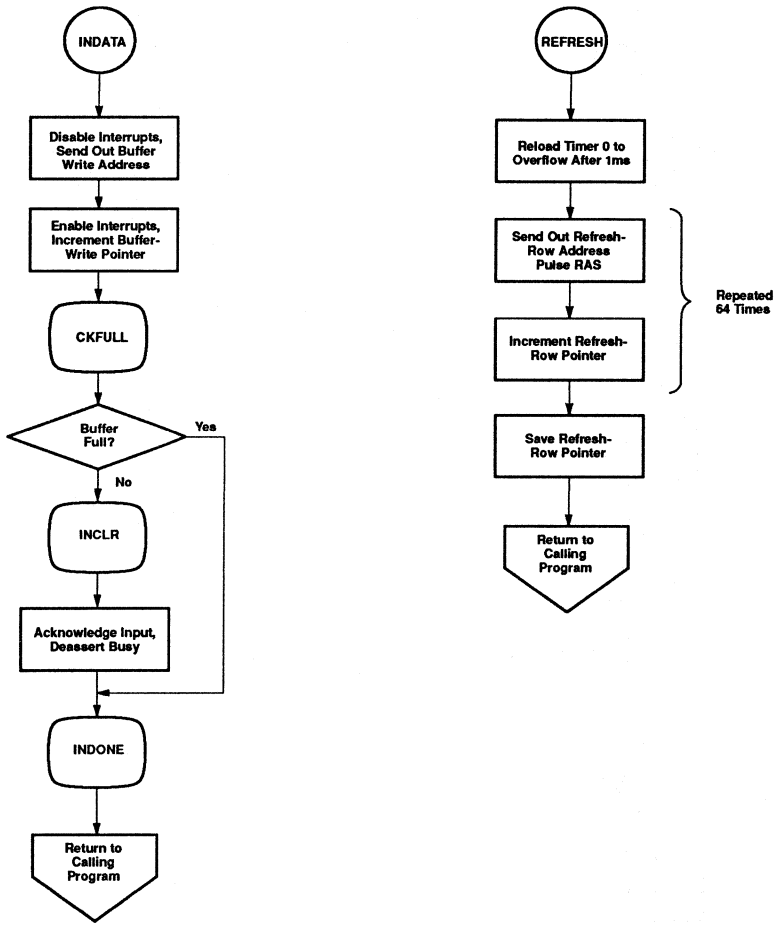


Figure 3. Flowchart of Receive and Refresh Operation

```
XMIT:       MOV DPTR,8001H
TEST:       MOVX A,@DPTR        ;READ THE CSR
            JB ACC.0,TEST       ;TEST IBF FLAG

;*****************************************************************************;
;                                                                             ;
;  256K PRINTER BUFFER PROGRAM USING THE 8xC451                               ;
;    FOR CENTRONICS PARALLEL PRINTER PORTS                                    ;
;                                                                             ;
;                    SIGNETICS CORPORATION                                    ;
;                       OCTOBER, 1988                                         ;
;                                                                             ;
;*****************************************************************************;

$Mod451
$Title(*XC451 Printer Buffer)
$Date(10/28/88)

;  PORT USAGE:
;
;  P0                   Not used (reserved for data/address bus when external
;                         program memory is used).
;  P1                   Lower 8 bits of DRAM address (A0 – A7).
;  P2                   Not used (reserved for high-order address bus when external
;                         program memory is used).
;
;  P3.0                 (Reserved for serial port.)
;  P3.1                 (Reserved for serial port.)
;  P3.2 (/INT0)         Input port strobe input (interrupt).
;  P3.3 (/INT1)         Output port acknowledge input (interrupt).
;  P3.4                 Input port acknowledge output.
;  P3.5                 DRAM write enable output.
;  P3.6 (/WR)           DRAM row address select output.
;  P3.7 (/RD)           DRAM column address select output.
;
;  P4.0                 Upper bit of DRAM address (A8).
;  P4.1                 Reserved as an extra address line for 1 megabit DRAMS.
;  P4.2                 Not used.
;  P4.3                 Output port busy input (OBUSY).
;  P4.4–P4.7            Unused (not available on 64-pin DIP package).
;
;  P5                   DRAM output data.
;
;  P6                   Parallel output port.
;  /IDS                 Input port strobe input (ISTB).
;  BFLAG                Input port busy output (IBUSY).
;
;  AFLAG                Output port strobe output (OSTB).
;  /ODS                 Port 6 output enable, tied low.
;
;
;  Internal Register/RAM Usage:
;
REFCNT      EQU     020h            ; Low order refresh byte.;
;
;  The following refer to the circular FIFO buffer
;  implemented in the DRAM array.
;
INLOW       EQU     22h             ; Incoming address low byte.
INMID       EQU     23h             ; Incoming address mid byte.
INHI        EQU     24h             ; Incoming address high byte.
OUTLOW      EQU     25h             ; Outgoing address low byte.
OUTMID      EQU     26h             ; Outgoing address mid byte.
OUTHI       EQU     27h             ; Outgoing address high byte.
```

```
;
OACK    EQU   28h          ; Holds flag for output port acknowledge.
FOACK   BIT   OACK.0       ; Bit-address of output port acknowledge flag.
;
;  Miscellaneous Equates:
;
TIME    EQU   -1000        ; Value for 1000 timer clocks = 1 millisecond.
TIMEHI  EQU   HIGH TIME    ; High byte of timer value.
TIMELO  EQU   LOW TIME     ;Low byte of timer value.
RAS     BIT   P3.6         ; DRAM column address select.
CAS     BIT   P3.7         ; DRAM row address select.
DRAMWR  BIT   P3.5         ; DRAM write control line.
IACK    BIT   P3.4         ; Input port ACK output.
ISTB    BIT   P3.2         ; Input port strobe line (INT0).
OBUSY   BIT   P4.3         ; Output port BUSY input.
OSTB    BIT   MA0          ; Output port strobe (MA0 bit in port 6 CSR).
;
;*****************************************************************************************
;
;  Reset and Interrupt Jump Table
;
        ORG   00h          ; Power-on reset.
        AJMP  START
;
        ORG   03h          ; INT 0.
        AJMP  INDATA       ; Data at input port.
;
        ORG   0Bh          ; Timer 0.
        AJMP  REFRESH      ; Refresh DRAM array.
;
        ORG   13h          ; INT 1.
        AJMP  OPACK        ; Output port acknowledge.
;*****************************************************************************************
;
;  Power up reset routine:
;  Set up refresh timer, enable timer interrupt and
;  external interrupt, initialize circular buffer pointers.
;
        ORG   18h
START:  MOV   SP,#40h      ; Initialize stack pointer.
        MOV   A,#00
;       MOV   REFCNT,A     ; Initialize refresh counter.
        MOV   INLOW,A      ; Initialize FIFO pointers.
        MOV   INMID,A
        MOV   INHI,A
        MOV   OUTLOW,A
        MOV   OUTMID,A
        MOV   OUTHI,A
;
```

```
;Initialize interrupt priority register so that DRAM refresh
;  (TF0) gets high priority, input port service (IE0) and output
;  port acknowledge service get lower priority. All other
;  interrupts set to lower priority level.
;
            MOV     IE,#00000111b       ; Timer0, INT0, and INT1 enabled.
            MOV     IP,#00000010b       ; Timer0 high priority.
            MOV     TL0,#TIMELO
            MOV     TH0,#TIMEHI
            MOV     TMOD,#00000001b     ; Operate Timer0 in mode 1.
            MOV     TCON,#00010101b     ; Timer0 run, I0 and I1 = edge.
;
;
;  Initialize Port 6 Control and Status Register.
;  -  'BFLAG' mode set to output value of IBF
;  -  (input port BUSY signal : IBUSY)
;  -  'AFLAG' set as logic 1 output
;     (output port strobe signal : OSTB)
;  -  'IDS' active on negative level
;     (input port strobe signal : ISTB)
;
            MOV     CSR,#10011100b
            MOV     A,P6                ; Dummy read of P6 to clear IBF (IBUSY).
            SETB    EA                  ; Enable interrupts.
;
;*****************************************************************************************************
;
;  Main Routine:
;  Executes while not performing DRAM refresh or servicing
;  input port interrupt.
;
;  Check if buffer is not empty by comparing input and output
;  pointers. If not empty, go to NOTMT to output a byte.
;
MAINLP:     MOV     A,INLOW         ; Compare pointers.
            CJNE    A,OUTLOW,NOTMT
            MOV     A,INMID
            CJNE    A,OUTMID,NOTMT
            MOV     A,INHI
            CJNE    A,OUTHI,NOTMT
            SJMP    MAINLP
;
;  Buffer is not empty: compute row & column addresses for
;  a read cycle from DRAM.
;
NOTMT:      MOV     R4,OUTLOW   ; Save low byte of row.
            MOV     R5,OUTMID   ; Save upper bit of row.
            MOV     A,OUTHI     ; Shift to align correctly.
            RRC     A
            MOV     R7,A        ; Save upper column bit.
            MOV     A,OUTMID    ; Get low byte of column.
            RRC     A           ; Shift in bit from OUTHI.
            MOV     R6,A        ; Save.
;
;  Now do actual DRAM access to get the data byte at computed
;  address. Disable interrupts so we don't lose what we put
;  out on the ports.
;
```

```
            CLR     EA              ; Disable interrupts.
            MOV     P1,R4           ; Low byte row address.
            MOV     A,R5            ; Get high byte row address.
            ORL     A,#0FEh         ; Make sure OBUSY stays high.
            MOV     P4,A
            CLR     RAS             ; /RAS low.
            MOV     P1,R6           ; Low byte column address.
            MOV     A,R7            ; High byte column address.
            ORL     A,#0FEh         ; Make sure OBUSY stays high.
            MOV     P4,A
            CLR     CAS             ; /CAS low.
            MOV     R4,P5           ; Get the data byte
;
            SETB    CAS             ; /CAS high.
            SETB    RAS             ; /RAS high.
            CLR     FOACK           ; Clear acknowledge flag.
            SETB    EA              ; Re-enable interrupts.
;
PLOOP1:     JB      OBUSY,PLOOP1    ; Loop if printer busy.
;
            CLR     EA              ; Disable interrupts.
            MOV     P6,R4           ; Move byte to output port.
            CLR     MA0             ; Assert output port strobe.
            NOP                     ; Kill some time.
            NOP
            NOP
            NOP
            SETB    MA0             ; De-assert output port strobe.
            SETB    EA              ; Re-enable interrupts.
;
;   Following waits for /ACk to occur on output port. Loops on
;   acknowledge flag which is set by INT1 service routine when
;   /ACK occurs.
;
PLOOP2:     JNB     FOACK,PLOOP2    ; Wait till /ACK occurs.
;
            INC     OUTLOW          ; Increment output buffer pointer.
            MOV     A,OUTLOW
            CJNE    A,#00,PDONE
            INC     OUTMID
            MOV     A,OUTMID
            CJNE    A,#00,PDONE
            MOV     A,OUTHI
            INC     A
            ANL     A,#03h          ; Eliminate unused address bits
            MOV     OUTHI,A         ; and save.
;
;   Check if input port busy flag was left asserted, indicating that
;   the buffer was full after last input. If so, acknowledge input
;   port and de-assert input busy signal.
;
PDONE:      JNB     IBF,MAINLP      ; Not busy, return to main loop.
            CLR     EA              ; Disable interrupts.
            CLR     IACK            ; Assert /IACK.
            NOP                     ; Wait 7 microseconds.
            NOP
            NOP
            NOP
            NOP
```

```
            NOP
            NOP
            MOV     A,P6            ; Dummy read of P6 clears IBF (IBUSY).
            NOP                     ; Wait 5 microseconds.
            NOP
            NOP
            NOP
            NOP
            SETB    IACK            ; De-assert /IACK.
            SETB    EA              ; Re-enable interrupts.
            AJMP    MAINLP          ; Return to main loop.
;
;*****************************************************************************
;
;  Interrupt 1 Service Routine:
;
;  - Called when output port asserts /ACK.
;  - Sets FOACK flag and returns.
;
OPACK:      SETB    FOACK
            RETI
;
;*****************************************************************************
;
;  DRAM Refresh (Timer0) Interrupt Service:
;
;  - Called once every millisecond by timer interrupt.
;  - Refreshes 64 rows and then returns.
;  - Therefore refreshes all rows every 4 milliseconds.
;    (Note that 41256/51C256 DRAM only requires a 256 row refresh.)
;
REFRESH:    PUSH    PSW
            MOV     TH0,#TIMEHI     ; Reload timer registers.
            MOV     TL0,#TIMELO

            MOV     P1,REFCNT       ; Get next row to refresh.
            MOVX    @R0,A           ; Pulse /RAS (/WR).
            INC     P1
            MOVX    @R0,A           ; 1
            INC     P1
            MOVX    @R0,A           ; 2
            INC     P1
            MOVX    @R0,A           ; 3
            INC     P1
            MOVX    @R0,A           ; 4
            INC     P1
            MOVX    @R0,A           ; 5
            INC     P1
            MOVX    @R0,A           ; 6
            INC     P1
            MOVX    @R0,A           ; 7
            INC     P1
            MOVX    @R0,A           ; 8
            INC     P1
            MOVX    @R0,A           ; 9
            INC     P1
            MOVX    @R0,A           ; 10
            INC     P1
            MOVX    @R0,A           ; 11
            INC     P1
```

```
MOVX    @R0,A           ; 12
INC     P1
MOVX    @R0,A           ; 13
INC     P1
MOVX    @R0,A           ; 14
INC     P1
MOVX    @R0,A           ; 15
INC     P1
MOVX    @R0,A           ; 16
INC     P1
MOVX    @R0,A           ; 17
INC     P1
MOVX    @R0,A           ; 18
INC     P1
MOVX    @R0,A           ; 19
INC     P1
MOVX    @R0,A           ; 20
INC     P1
MOVX    @R0,A           ; 21
INC     P1
MOVX    @R0,A           ; 22
INC     P1
MOVX    @R0,A           ; 23
INC     P1
MOVX    @R0,A           ; 24
INC     P1
MOVX    @R0,A           ; 25
INC     P1
MOVX    @R0,A           ; 26
INC     P1
MOVX    @R0,A           ; 27
INC     P1
MOVX    @R0,A           ; 28
INC     P1
MOVX    @R0,A           ; 29
INC     P1
MOVX    @R0,A           ; 30
INC     P1
MOVX    @R0,A           ; 31
INC     P1
MOVX    @R0,A           ; 32
INC     P1
MOVX    @R0,A           ; 33
INC     P1
MOVX    @R0,A           ; 34
INC     P1
MOVX    @R0,A           ; 35
INC     P1
MOVX    @R0,A           ; 36
INC     P1
MOVX    @R0,A           ; 37
INC     P1
MOVX    @R0,A           ; 38
INC     P1
MOVX    @R0,A           ; 39
INC     P1
MOVX    @R0,A           ; 40
INC     P1
MOVX    @R0,A           ; 41
INC     P1
```

```
        MOVX    @R0,A           ; 42
        INC     P1
        MOVX    @R0,A           ; 43
        INC     P1
        MOVX    @R0,A           ; 44
        INC     P1
        MOVX    @R0,A           ; 45
        INC     P1
        MOVX    @R0,A           ; 46
        INC     P1
        MOVX    @R0,A           ; 47
        INC     P1
        MOVX    @R0,A           ; 48
        INC     P1
        MOVX    @R0,A           ; 49
        INC     P1
        MOVX    @R0,A           ; 50
        INC     P1
        MOVX    @R0,A           ; 51
        INC     P1
        MOVX    @R0,A           ; 52
        INC     P1
        MOVX    @R0,A           ; 53
        INC     P1
        MOVX    @R0,A           ; 54
        INC     P1
        MOVX    @R0,A           ; 55
        INC     P1
        MOVX    @R0,A           ; 56
        INC     P1
        MOVX    @R0,A           ; 57
        INC     P1
        MOVX    @R0,A           ; 58
        INC     P1
        MOVX    @R0,A           ; 59
        INC     P1
        MOVX    @R0,A           ; 60
        INC     P1
        MOVX    @R0,A           ; 61
        INC     P1
        MOVX    @R0,A           ; 62
        INC     P1
        MOVX    @R0,A           ; 63
        INC     P1

        INC     P1              ; Adjust for next time
        MOV     REFCNT,P1       ; and save.
        POP     PSW
        RETI
```

****

Data at Input Port:

This routine is called via interrupt INT0 whenever data is strobed into the input port. It saves the data into the DRAM array and increments the input pointer. If the output pointer is now equal to the input pointer, then the buffer is full, and we leave the busy flag set so that no more data can be input until some is output and the buffer is no longer full.

```
;
;
INDATA:  PUSH    PSW
         PUSH    ACC
         MOV     R1,INLOW        ; Lower 8 bits of row to R1.
         MOV     R2,INMID        ; Upper bit of row to R2.
         MOV     A,INHI          ; Get upper 2 bits.
         RRC     A               ; LSB to carry.
         MOV     R0,A
         MOV     A,INMID
         RRC     A               ; Shift bit into MSB.
         MOV     R3,A            ; Save.
;
         CLR     EA              ; Disable interrupts.
         MOV     P1,R1           ; LSB Row address.
         MOV     A,R2            ; MSB row address.
         ORL     A,#0FEh         ; Make sure OBUSY stays high.
         MOV     P4,A            ; MSB row address.
STBLP:   JNB     ISTB,STBLP      ; Check for end of strobe before DRAM write.
         CLR     RAS             ; /RAS low.
         CLR     DRAMWR          ; /WR low.
         MOV     P1,R3           ; LSB column address.
         MOV     1,R0            ; MSB column address.
         ORL     A,#0FEh         ; Make sure OBUSY stays high.
         MOV     P4,A            ; MSB column address.
         MOVX    A,@R0           ; Pulse /CAS low.
         SETB    RAS             ; /RAS high.
         SETB    DRAMWR          ; /WR high.
         SETB    EA              ; Re-enable interrupts.
;
         INC     INLOW           ; Increment input buffer pointer.
         MOV     A,INLOW
         CJNE    A,#00,CKFULL
         INC     INMID
         MOV     A,INMID
         CJNE    A,#00,CKFULL
         MOV     A,INHI
         INC     A
         ANL     A,#03h          ; Eliminate unused address bits.
         MOV     INHI,A
;
;  Compare input pointer to output pointer to see if the buffer is full.
;
CKFULL:  MOV     A,INLOW
         CJNE    A,OUTLOW,INCLR
         MOV     A,INMID
         CJNE    A,OUTMID,INCLR
         MOV     A,INHI
         CJNE    A,OUTHI,INCLR
;
;  If we get here, the buffer is full, so skip the acknowledge pulse.
;
         SJMP    INDONE
;
;
;  Send acknowledge pulse on /IACK line for 7 microseconds,
;  de-assert input BUSY signal halfway through.
;
INCLR:   CLR     EA              ; Disable interrupts.
         CLR     IACK            ; Assert /IACK.
```

```
        NOP             ; Wait 7 microseconds.
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        MOV     A,P6    ; Dummy read of P6 clears IBF (IBUSY).
        NOP             ; Wait 5 microseconds before clearing /IACK.
INDONE: POP     ACC
        POP     PSW
        SETB    IACK    ; De-assert /IACK.
        SETB    EA      ; Re-enable interrupts.
        RETI
;
        END
```

AN418
Counter/timer 2 of the 83C552 microcontroller

INTRODUCTION TO THE 83C552

The 83C552 is an 80C51 derivative with several extended features: 8k ROM, 256 bytes RAM, 10-bit A/D converter, two PWM channels, two serial I/O channels, six 8-bit I/O ports, and four counter/timers. The architecture of the 83C552 is identical to that of the 80C51, making the two devices fully code compatible. The additional peripheral functions are added to the 80C51 Special Function Register space, and the interrupt structure is modified accordingly. This information is detailed in other references on the 83C552. The focus of this application note is on one of the timers of the 83C552, Counter/Timer 2.

This counter/timer includes capture, compare, and high-speed output capabilities which facilitate many control oriented tasks. The objective of this note is to make users of the 83C552 aware of this counter/timer subsystem and assist the use of this subsystem by a detailed explanation of its operation supported by actual application examples.

TIMER 2 OF THE 83C552

Timer 2 of the 83C552 is in fact a timing controller and has an associated programmable array. The Timer 2 subsystem consists of three parts:

1. The time base consists of a 16-bit timer with a 3-bit prescalar. The master clock for the subsystem can be derived from the on-chip oscillator (fosc) or an external input, T2. It has an external reset, RT2, by which a signal applied to this input can reset the timer if the external reset is enabled.
2. A capture system consisting of four capture registers and four capture inputs which can be used for a wide variety of time measurements on external signals.
3. A compare system consisting of three compare registers and eight associated high-speed outputs which can be activated upon a match between the 16-bit timer and one of the compare registers.

For reference a complete block diagram of the 83C552 Counter/Timer 2 subsystem is shown in Figure 1.

16-BIT COUNTER/TIMER

The description of Counter/Timer 2 in the following paragraphs is intended to be a general overview. Details on architecture, address locations, interrupt structure, and timer operation are given in the 83C552 Users Manual. This users manual may be useful to complement the material presented in this application note. References to registers, bits, I/O ports, and on-chip hardware will relate directly to 83C552 Users Manual nomenclature. This application note will focus on the use of Counter/Timer 2 as a powerful input capture and high-speed output facilitator through some specific examples and not on the detailed coding.

The counter/timer consists of a 16-bit counter which is readable by software through special function registers TM2L and TM2H. The timer itself has two overflow flags, one after the entire 16-bit counter and one attached to the eighth stage. This latter flag reflects an overflow from the first byte of the counter. These two flags are present in register TM2IR and are labeled T2BO for the overflow from the first byte and T2OV for the overflow from the entire 16 bits. These flags may be used to generate an interrupt.

The counter timer is controlled directly through the special function register TM2CON, the timer 2 control register. This register also contains certain status flags.

The prescaler divides the input clock by a programmable ratio. The prescaler divide value is programmable to divide by 1, 2, 4, or 8 as controlled by T2PO and T2P1 in TM2CON.

The input clock to the prescaler is either fosc/12 or the external input, T2. The clock input to the prescaler may also be shut off. This clock input selection is controlled by bits T2MS0 and T2MS1 in TM2CON.

If T2 is used as the input clock to the timer 2 subsystem, the hardware logic samples this input and looks for a low-to-high transition. If the logic detects a logic 0 at the T2 input in state S2P1 of the microcontroller and a logic 1 in state S5P1, then this is recognized as a low-to-high transition, and the prescaler is incremented. The prescaler is incremented in the second cycle after the cycle in which the transition was detected. If the transition is detected before S2P1 is finished, the prescaler is incremented in the next cycle. This timing is shown in Figure 2. Note that this sampling rate is twice that of the normal 80C51 timers, T0 and T1; therefore T2 has twice the maximum external counting rate as compared to the standard timers.

Any programming of the clock source or the prescaler divide ratio results in a reset of the prescaler. This allows the state of the timer subsystem to be in a known state upon programming. The main 16-bit timer cannot be reset by software but it is reset by activating the reset pin or using the external reset, RT2. The external reset, RT2, can be enabled or disabled by bit T2ER in TM2CON. These resets reset the prescaler as well as the 16-bit counter.

Only one interrupt is available from the 16-bit counter timer. Two bits in TM2CON control whether TM2L, TM2H, or both flags will be used to generate the interrupt. A selection for no interrupt is also possible.

Capture System

The capture system is a powerful tool to measure the width of pulses or repetition rates. There are four independent inputs for the signals to be analyzed, CTI0 through CTI3. These inputs are alternate functions to port 1. Each input is connected to a dedicated capture register. A transition at any of these inputs will cause the content of the 16-bit counter/timer to be loaded into the respective capture register. The capture can occur upon various conditions of the input signal as specified by certain bits in the capture control register, CTCON. Each input can be set to cause a capture on a low-to-high transition, a high-to-low transition, or on both transitions. Upon a capture taking place, each input causes an interrupt flag to be set in the Timer 2 Interrupt Flag Register, TM2IR. If enabled, an interrupt will be generated.

One of the capture inputs is shown in more detail in Figure 3. All of the other capture inputs are similar to this one. The capture input is gated with the capture enable bits CTN0 and CTP0, which are located in CTCON. According to the status of these bits, the desired edges are selected to generate the capture enable pulse. The input pulse transient detection is at the input of the enable pulse generator. The input signal is sampled at S1P1 of the machine cycle. If a logic 1 is detected when a logic 0 was detected at the same time in the previous cycle, then the event is taken as a transition. An enable pulse is sent to the capture register, and the contents of timer 2 is copied into the capture register at the end of this machine cycle. The interrupt flag CTI0 is also set.

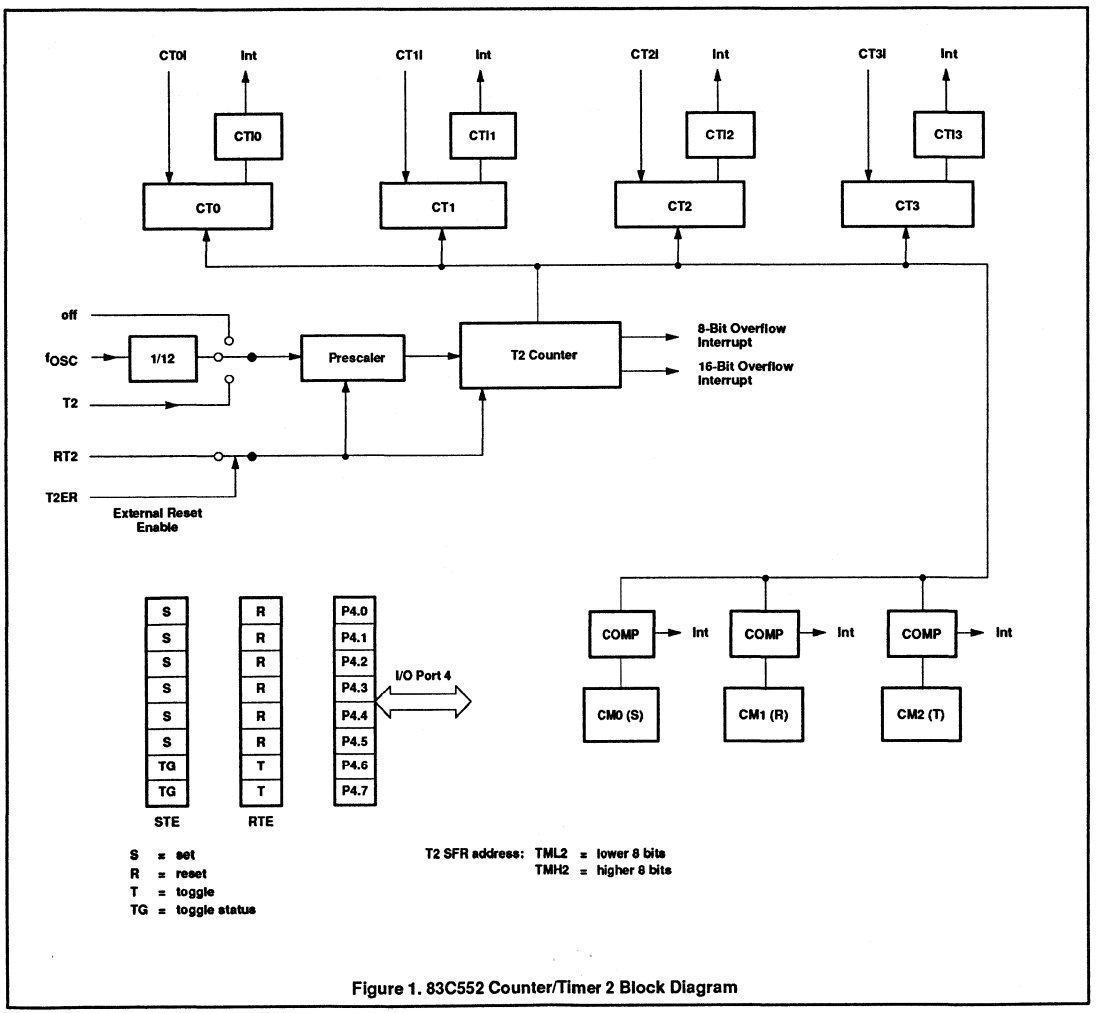


Figure 1. 83C552 Counter/Timer 2 Block Diagram

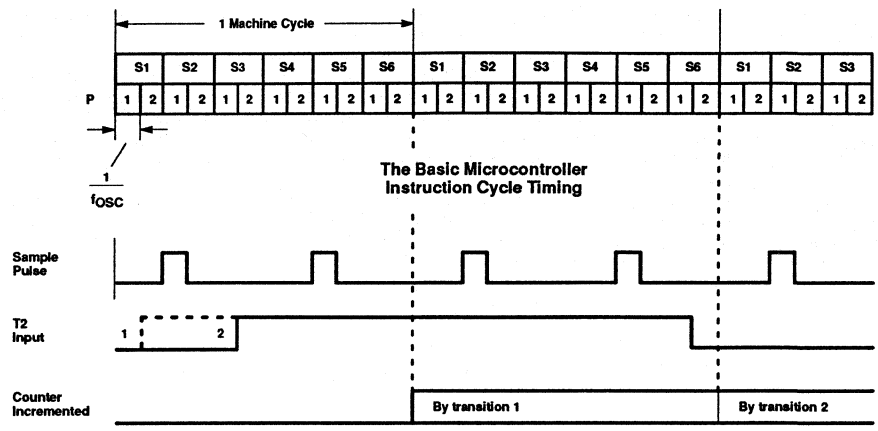


Figure 2. The Sampling of the External Clock Signal

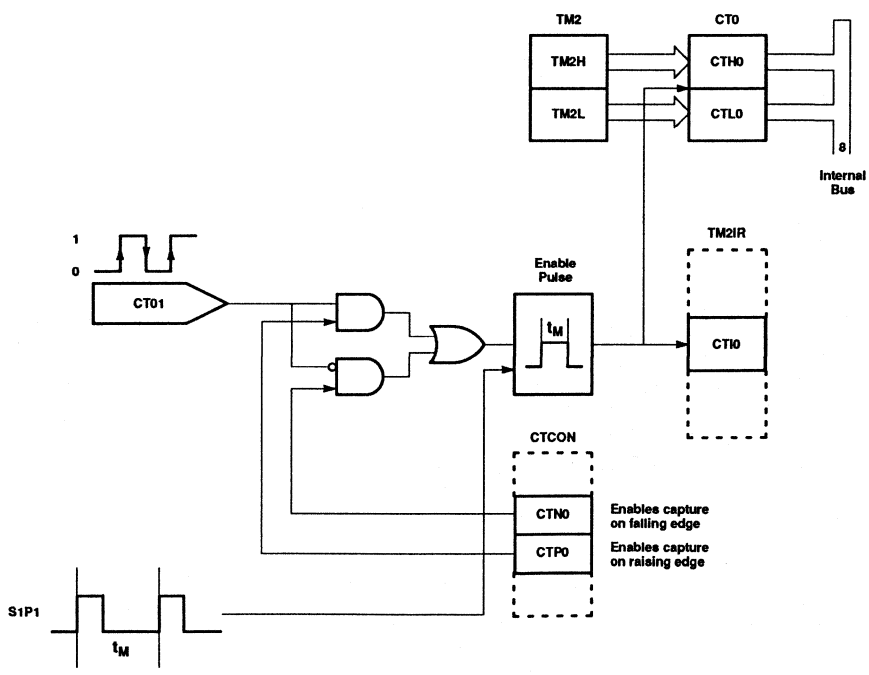


Figure 3. Capture Subsystem for CTO1

Compare System

The compare system of Timer 2 can be used to generate a set of outputs whose transitions are controlled directly by the time defined by the 16-bit counter/timer. There are eight of these high-speed outputs which are directly controlled from Counter/Timer 2. These outputs are alternate functions to port 4. Six of these outputs are set-reset controlled (CMSR0 through CMSR5), and two are toggle controlled (CMT0 and CMT1). To clarify the operation, these two types will be discussed separately. In the following discussions, refer to Figure 4, which shows the compare system for P4.5 (a set-reset high-speed output) and P4.6 (a toggle high-speed output).

There are two compare registers associated with the set-reset outputs. These registers are CM0 and CM1. In addition, there are two enable registers: one to enable setting of an output and the other to enable resetting of an output. These registers are STE and RTE, respectively. The contents of CM0 and CM1 are continuously compared to the contents of the 16-bit counter. Whenever there is a match between the 16-bit counter and the contents of CM0, a SET pulse is generated. Similarly, whenever there is a match between the timer and the contents of CM1, a RESET pulse is generated. The set pulse is applied to the set-reset outputs, CMSR0 through CMSR5, through gates controlled by bits in STE. Bits 0 through 5 in STE control the application of the SET pulse to one of the high-speed outputs. For example, STE.0 controls the gating of the SET pulse to CMSR0, STE.1 controls the gating of the SET pulse to CMSR1, and so forth. Thus, if the corresponding SET enable bit in STE is a 1 and a compare occurs with CM0, then that high-speed output will become set. Similarly, the reset pulse from CM1 is applied to the high-speed outputs CMSR0 through CMSR5 through gates controlled by bits in RTE. As with STE, bits 0 through 5 in RTE control the application of the reset pulse to one of the high-speed outputs. If a compare occurs between the timer and CM1, a high-speed output will be reset if its corresponding enable bit in RTE is a 1. Compares with CM0 and CM1 set interrupt flags which, if enabled, can be used to generate an interrupt.

The two toggle-controlled outputs are CMT0 and CMT1, and these are associated with compare register CM2. These outputs are also alternate functions on port 4. Upon a compare between the counter and the contents of CM2, CM2 generates a toggle pulse which is applied to the high-speed outputs CMT0 and CMT1 through a set of gates. The gates control the application of the toggle pulse to the toggle outputs as specified by the high-order bits of register RTE. RET.6 controls CMT0, and RET.7 controls CMT1. Should the corresponding bit of RTE be set, then the toggle pulse is enabled to the associated high-speed output and that output will toggle upon generation of the toggle pulse from CM2. The structure of these toggled outputs is different from the other high-speed outputs in that the toggling is actually accomplished in a separate toggled flip-flop and not directly in the port latch. This toggle flip-flop cannot be controlled directly by software and powers up in an indeterminate state. The state of the toggle flip-flops is readable in STE bits 6 and 7.

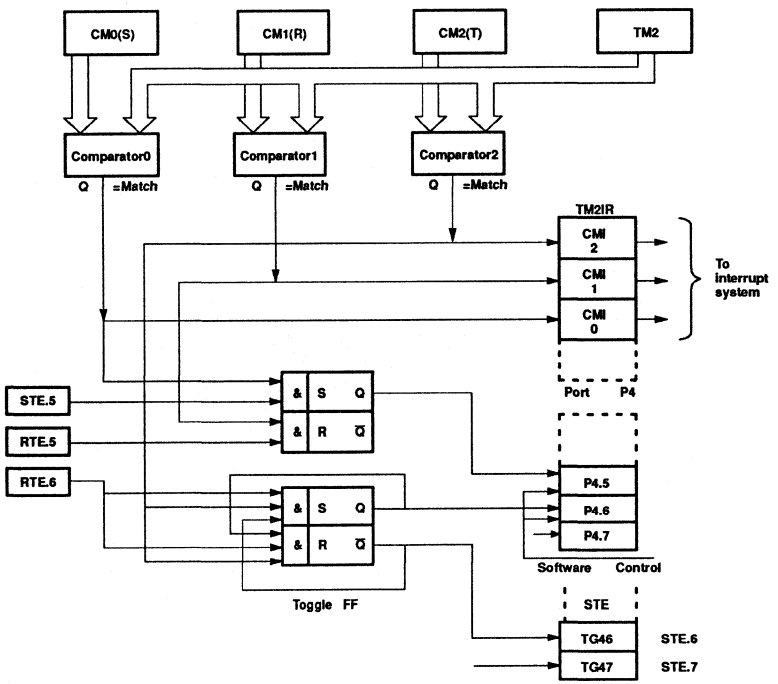


Figure 4. Compare System for P4.5 and P4.6

APPLICATION EXAMPLE—TIMED FUEL INJECTION

In modern automobiles, optimal combustion is necessary to meet emission standards and improve fuel consumption. Optimal combustion depends on several factors and is enhanced by proper fuel injection based upon these factors which vary according to engine speed and other factors. Thus the task is to control the opening and closing of the engine fuel injectors of each cylinder relative to the crankshaft reference point.

For the application example here, we will not consider the factors which determine the timing relationships. These are assumed to be given quantities. The example here will focus upon the implementation of the injector timing control signals and how they are generated using the Counter/Timer 2 system. The illustration considers a four cylinder engine. While this is an automotive application which serves to clearly illustrate Counter/Timer 2 Subsystem operation, it is clear that many systems share similar timing requirements, and the techniques employed here are applicable to a wide class of timing tasks. The 83C552 will also support six cylinder engine control.

Figure 5 shows the injection timing required for two consecutive revolutions of the engine crankshaft.. Start and stop of the injection are given relative to a reference point on the crankshaft. The cylinders are numbered in the order of the injection sequence (not with reference to their physical location). Start of the injection is usually given in angular measure with respect to top dead center, and the injection duration is assumed to be a time value calculated from engine environmental factors and operating parameters. The angle for the start of the injection must be converted into time with respect to the reference point.

The injector drivers are assumed to be connected to the port 4 high-speed outputs CMSR0 through CMSR3. To obtain the top dead center reference point, the signal from the appropriate sensor is connected to the capture input CTOI. The interrupt for this capture input is enabled so that software can synchronize its operation to this time reference and make use of the top dead center time in the injector timing calculations. The software synchronization takes two forms. First, the captured time is an absolute reference for all real-time output operations. This time is available in capture register CT0. Note that at 12 MHz operation, Timer 2 can have a resolution as fine as 1 microsecond with a total time before overflow of over 65 milliseconds, and these times are adjustable by increasing the prescaler divide ratio. A proper selection can make the timing calculations relatively simple. Second, at the time the input is captured, flags which keep track of the phases of the crankshaft cycle are reset when cylinder 1 is at top dead center. These flags are used in the interrupt service routines to tell which action is required for that phase of the crankshaft.

Consider now the sequence of events in one rotation of the engine crankshaft and refer to Figure 5 during the discussion. Assume that the engine is running, that all relevant parameters are available, and that it has been determined that the processor is responding to the interrupt associated with the top dead center capture, CTOI. Interrupts for CTOI, CM0 (compare register 0), and CM1 (compare register 1) are enabled. Upon entering the interrupt service routine for CTOI, the previous value of the captured top dead center time is subtracted from the present value, and the crankshaft rotation time is determined. This is used to compute the time to open the first injector from the required angle at which the injector is to open. This time is made available for the interrupt service routine which responds to a compare from CM0. The interrupt service routine is exited.

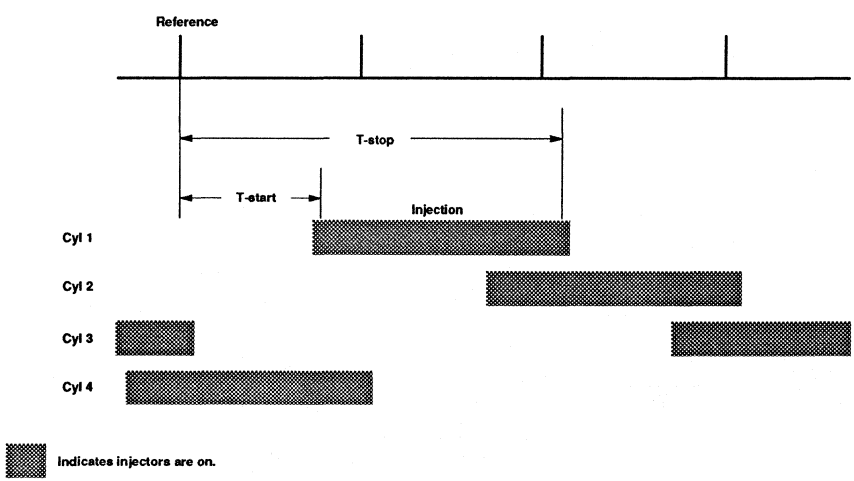


Figure 5. Four Cylinder Injection Timing

The next interrupt to occur per the figure for this example is a result of a compare with CM1 which will be a result of the injector stop time for cylinder 4 having been reached. The flags in an internal status register are employed to keep track of the cylinder number that is presently active for both injection stop and injection start times. After identifying this interrupt from the flags, the processor uses the injector start time for cylinder 1 (previously loaded into CM0) and the predetermined duration to calculate the injector stop time for cylinder 1. This value is loaded into compare register CM1 and the reset enable bit for high-speed output CMSR0 is programmed to a 1. This is bit RTE.0 The reset enable bit for the cylinder 4 injector is set to 0 (bit RET.3). The interrupt routine is now exited.

The next interrupt to occur will be for the start time for the injector for cylinder 2. This and all subsequent cases follow the same sequence of events as for the cylinder 1 CM0 interrupt described above. In this case, calculations are made for cylinder 2 and loaded into CM0,STE.1 is programmed to a 1, and STE.0 is programmed to a 0. Similarly, the next interrupt for CM1 is treated in the same way, and the sequence of events rotates around through all cylinders in turn. The flag bits associated with this operation keep track of the injector sequencing.

While this example shows the injection stop time of one cylinder overlapping into the injector on time of the subsequent cylinder, close examination of the operations described above reveal that the start and stop events are independent and can overlap or not as required. In this way all injectors may be driven independently and have overlapping on times.

Given that this is an example applicable to general usage, it is possible that interrupt service routine could be relatively long as it would be in an actual injector application. Since the service routine has other interrupts disabled, the length may cause real-time conflicts. To eliminate this potential problem, the interrupt service routines are divided into two parts. In the first part, all other interrupts are disabled, and the essential register loading is done to prepare for the next interrupt. After this is completed, all interrupts are enabled and the ancillary service routine functions are performed prior to a return to the main routine.

As an example, consider the interrupt service routine for CM0. Upon entering the routine, all interrupts are disabled. Then the following actions are performed:

- Set bit in STE to start next injector
- Clear bit in STE for injector just started
- Load CM0 with start time for next injector
- Clear CMIO interrupt flag in TM2IR

Now that the essential set-up is made for the next interrupt, all interrupts are now enabled. However, the return to the main program is not invoked until the following ancillary processing is completed:

- Calculate the next absolute start time for the next injector (the next load value for CM0)
- Increment the flag so that the next entry to this interrupt service routine will be able to identify the next injector to start.

The process performing these calculations can be interrupted to service real-time functions.

APPLICATION EXAMPLE—TIMED IGNITION

In electronic ignition systems, multiple ignition coils may be used and each coil is fired by electronic means rather than with the old style mechanical breakers. In a four cylinder engine, there may be two ignition coils, one coil providing spark for a pair of cylinders. Both plugs fire at the same time. For one cylinder, the spark occurs at the appropriate time while for the other cylinder, the spark occurs at the end of the exhaust stroke and has no effect. With timing references to crankshaft top dead center provided by an external sensor, the ignition timing for the engine may be generated in the 83C552 and applied to the electronic drivers for the ignition coils.

To illustrate the toggle high-speed outputs of the 83C552 Counter/Timer 2 subsystem, the following example will discuss the ignition timing in a four cylinder engine employing the two coil approach with one coil for a pair of cylinders. The coil timing is illustrated in Figure 6. A reference time is used which is a given interval prior to top dead center so that the times used in the illustration can be always after the reference. There are two times of interest for each coil: the load time and the ignition point.

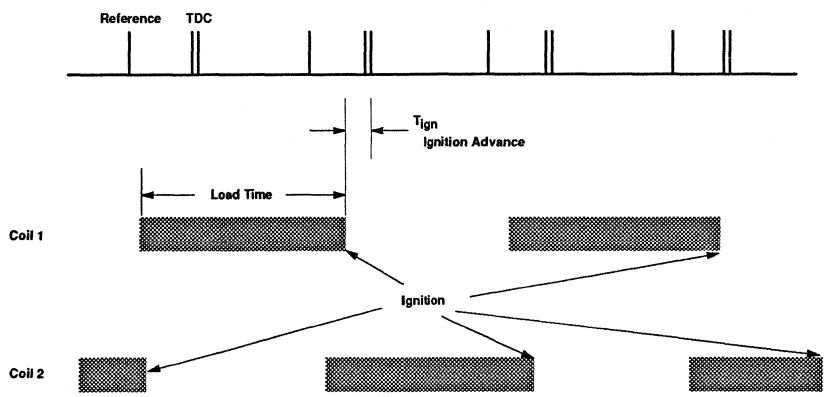


Figure 6. Four Cylinder, Two Coil Ignition Timing

Counter/timer 2 of the 83C552 microcontroller AN418

Ignition advance is usually given in degrees crankshaft angle prior to top dead center. As with injection, this angle is assumed to be derived from other calculations and is a given value for this illustration. This angle must be converted in to a time with respect to the reference point. The load time (the time at which the coil has to be switched on to reach the current that will give sufficient energy for an adequate spark) must be subtracted from the desired ignition point. At the ignition time, the coil will be switched off and the spark will be generated.

The coil driver electronics are connected to port bits P4.6 and P4.7. Ignition coil 1 is connected to P4.6, and ignition coil 2 is connected to P4.7. These outputs are the toggle high-speed outputs controlled by the 16-bit compare register, CM2. The program simply needs to set up the compare and control registers to turn the coils on and off at the appropriate times. It is assumed in this example that the ignition and load times are given quantities and have been determined previously.

Consider now the sequence of events in two rotations of the engine crankshaft and refer to Figure 6. Assume that the engine is running, that all relevant parameters are available, and that it has been determined that the processor is responding to the interrupt associated with a compare to CM2. The top dead center time and crankshaft rotation speed have been already determined through the top dead center capture, CTOI. This is the same as in the injector example. The interrupt for CTOI is enabled. From the top dead center time, the times to turn on and turn off the coil drivers are computed and made available in data storage locations in the microcontroller. It is also convenient to have flags to identify the step in the complete ignition cycle. The flags are cleared in the interrupt service routine for top dead center of cylinder 1.

Upon entering the interrupt service routine, other interrupts are disabled. Examination of the flags reveals that the state of the ignition sequence is that coil 1 has been turned on to begin the current build up (load time). The next event will therefore be turning off coil 2 to cause ignition. The interrupt service routine then performs the following actions: The time to turn off coil 2 is moved into compare register 2, CM2. Bit 6 of RTE is cleared; this disconnects the output of CM2 from the toggle flip-flop of P4.6 (coil 1). Bit 7 of RTE is set; this connects the output of CM2 to the toggle flip-flop of P4.7 (coil 2). The flags are incremented to indicate that the next interrupt will be a result of coil 2 turning off and causing ignition. The other interrupts can be enabled and a return to the main program can be executed. After the other interrupts are enabled and before a return is made to the main program, it may be convenient to do any necessary calculations to determine the time value to be loaded into CM2 in the next CM2 interrupt.

Since the flip-flops are toggled, it is likely that upon power up of the microcontroller, the toggle flip-flops will not be in the desired state. To get the toggle flip-flops in the correct state in the ignition cycle, the flip-flops must be toggled if they are in the wrong state. To determine if this is necessary, the state of the toggle flip-flops can be read from the STE register. The state of the P4.6 flip-flop is present in STE bit 6 and the state of the P4.7 flip-flop is present in STE bit 7. Comparing the actual state to the required state determines which if any or both of the flip-flops must be toggled. If a toggle is necessary to put one or both of the flip-flops in the correct state, the corresponding bits in RTE would be set for those flip-flops requiring the toggle, and CM2 would be loaded with a value that is slightly larger than the present contents of timer 2. If desired for reliability purposes, the state of the flip-flops could be checked periodically against the ignition cycle flags to determine if a correction is necessary.

CONCLUSION

This application note has examined one aspect of the 83C552 CMOS 80C51 derivative microcontroller. The Counter/Timer 2 Subsystem has been applied to a complex timing task of gasoline engine injector valve and ignition coil timing control. While this is a specific application to the automotive interests, the results are applicable to a wide variety of time measurement and control applications. The 83C552 would be ideal for many electromechanical systems such as copy machines, fax machines, industrial process control equipment, automatic transmission control, and anti-skid and anti-lock braking control.

These application areas are those which can successfully employ the 83C552 Counter/Timer 2; however, the other features should not be overlooked. When combined with the 10-bit A to D Converter, the Pulse Width Modulator, the I^2C serial bus, and peripheral device family, the 83C552 provides minimum component count solutions for cellular radio systems, professional audio systems, and medical instrumentation products such as bedside patient monitors and analyzers for home care and sports use.

AN420

Using up to 5 external interrupts on 80C51 family microcontrollers

80C51 family microcontrollers are equipped with up to two inputs which may be used as general-purpose interrupts. A typical device provides a total of 5 interrupt sources. Timer 0 and Timer 1 generate vectored interrupts, as does the Serial Port. Applications that require more than two externally signaled vectored interrupts, and do not use one or more of the counters or the serial port, can be configured to use these facilities for additional external interrupt inputs.

This note describes a method to configure the timer/counters and the serial port for use as interrupt inputs (see Figure 1). Minimum response time is a goal for this configuration.

Another popular method to implement extra interrupt inputs is to poll under software control a port pin configured as an input. This method is necessary when the on-chip peripherals are in use. Applications where this approach is recommended are ones in which the processor spends more than half of the time executing a "wait loop," or a short code sequence which jumps or branches back on itself without performing any functions. In this case, the instructions that will check the state of input used as an interrupt source are inserted into this sequence. Consequently, this input is ignored when other routines are being executed. This input may have to be latched externally, or the processor may miss the signal while executing other routines.

Dedicated interrupt inputs that vector the processor to individual service routines (as the two general-purpose interrupt inputs work) do not have the drawbacks of the method described above.

COUNTER/TIMER CONFIGURATION

Timers 0 and 1 are placed in mode 2, which configures the timer/register as an 8-bit counter with automatic reload. The counter and reload register are loaded with FF hexadecimal which is stored in TH1 and TL1 or TH0 and TL0.

To prepare one of the timers for this kind of operation, a number of control bits have to be set up. The following is a list of these bits and their values:

In TMOD:	In TCON:	In IE:
GATE = 0	TRi = 1	ETi = 1
C/T = 1		EA = 1
M1 = 1		
M0 = 0		

Where "i" is the timer number being used as the external interrupt. The TMOD value would be 66 hexadecimal if both timers are being used as external interrupt sources, x6 hex for timer 0, and 6x hex for timer 1. The interrupt priority may also be set in the IP register.

A falling edge on the corresponding Timer 0 or Timer 1 input (T0 or T1) will cause the counter to overflow and generate a timer interrupt. The counter will be automatically loaded with another FF from the reload register, so the interrupt can occur again as soon as the interrupt service routine completes. Counter/Timer operation is described in detail elsewhere in this manual.

SERIAL PORT CONFIGURATION

The serial port can be placed in mode 2, which is a 9-bit UART with the baud rate derived from the oscillator. The external interrupt is signaled through this port on the RxD receive data pin. Reception is initiated by a detected 1-to-0 transition at RxD. The signal must stay at 0 for at least five-eighths of a bit period for this level to be recognized. Refer to the description of baud rates to determine the length of a bit period at the oscillator frequency selected for the application. The input signal should remain low for at least one bit period and for not more than 9 bit periods.
To prepare the serial port for use as an external interrupt, the following bits must be set up:

In SCON:

SM0 = 1
SM1 = 0
SM2 = 0
REN = 1

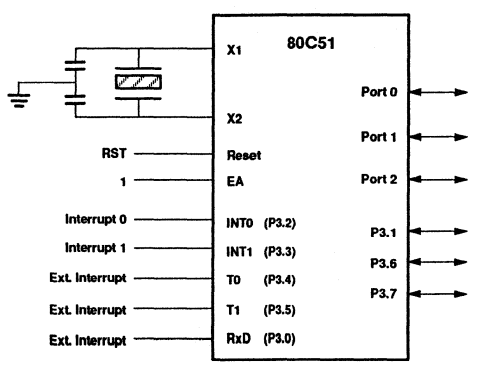


Figure 1. 80C51 Five Interrupt Configuration

Using up to 5 external interrupts on 80C51 family microcontrollers

AN420

The Serial Port Interrupt is then used as a general-purpose interrupt. The contents of receive buffer should be ignored, and will subsequently be overwritten during the next interrupt.

Note that the response time for this input will be slower than for the Counter/Timer inputs. This is due to the fact that the RI is generated after the eighth serial data bit time after the falling edge on RxD.

```
; Demonstration program for five external interrupts.

$MOD51
$TITLE (Five Vectored External Interrupts)

; Interrupt Jump Table
            ORG     0H              ;Reset
            AJMP    Setup

            ORG     3H              ;External interrupt 0.
            RETI                    ;(not implemented in this demo)

            ORG     0BH             ;Timer 0 interrupt.
            AJMP    Tim0

            ORG     13H             ;External interrupt 1.
            RETI                    ;(not implemented in this demo)

            ORG     1BH             ;Timer 1 interrupt.
            AJMP    Tim1

            ORG     23H             ;Serial port interrupt.
            AJMP    Serial

; Begin setup code
Setup       MOV     SP,#7FH         ;Initialize the stack pointer.

; Configure both timers
            MOV     TMOD,#66H       ;Put both counters into mode 2.
            MOV     A,#0FFH
            MOV     TL0,A           ;Load FF hex into both counters
            MOV     TH0,A
            MOV     TL1,A
            MOV     TH1,A
            SETB    ET0             ;Enable Timer 0 interrupt.
            SETB    ET1             ;Enable Timer 1 interrupt.
            SETB    TR0             ;Enable Timer 0 to run.
            SETB    TR1             ;Enable Timer 1 to run.

; Configure the serial port
            SETB    ES              ;Enable serial port interrupt.
            MOV     SCON,#90H       ;Put the serial port in mode 2.
            SETB    EA              ;Enable interrupt system.

Wait:       NOP                     ;Wait for an interrupt.
            JMP     Wait

Serial:     NOP                     ;Serial interrupt service routine.
            CLR     RI              ;Clear receiver interrupt flag.
            RETI

Tim0:       NOP                     ;Timer 0 interrupt service routine.
            RETI

Tim1:       NOP                     ;Timer 0 interrupt service routine.
            RETI

            END
```

AN422

Using the 8XC751 microcontroller as an I²C bus master

DESCRIPTION

The 83C751/87C751 Microcontroller offers the advantages of the 80C51 architecture in a small package and at a low cost. It combines the benefits of a high-performance microcontroller with on-board hardware supporting the Inter-Integrated Circuit (I^2C) bus interface.

The I^2C bus, developed and patented by Philips, allows integrated circuits to communicate directly with each other via a simple bidirectional 2-wire bus. The comprehensive family of CMOS and bipolar ICs incorporating the on-chip I^2C interface offers many advantages to designers of digital control for industrial, consumer and telecommunications equipment. A typical system configuration is shown in Figure 1.

Interfacing the devices in an I^2C based system is very simple because they connect directly to the two bus lines: a serial data line (SDA) and a serial clock line (SCL). System design can rapidly progress from block diagram to final schematic, as there is no need to design bus interfaces, and functional blocks on a block diagram correspond to actual ICs. A prototype system or a final product version can easily be modified or upgraded by 'clipping' or 'unclipping' ICs to or from the bus. The simplicity of designing with the I^2C bus does not reduce its effectiveness; it is a reliable, multi-master bus with integrated addressing and data-transfer protocols (see Figure 2). In addition, the I^2C-bus compatible ICs provide cost reduction benefits to equipment manufacturers, some of which are smaller IC packages and a minimization of PCB traces and glue logic.

The availability of microcontrollers like the 83C751, with on-board I^2C interface, is a very powerful tool for system designers. The integrated protocols allow systems to be completely software defined. Software development time of different products can be reduced by assembling a library of reusable software modules. In addition, the multi-master capability allows rapid testing and alignment of end-products via external connections to an assembly-line computer.

The mask programmable 83C751 and its EPROM version, the 87C751, can operate as a master or a slave device on the I^2C small area network. In addition to the efficient interface to the dedicated function ICs in the I^2C family, the on-board interface facilities I/O and RAM expansion, access to EEPROM and processor-to- processor communications.

The multi-master capability of the I^2C is very important but many designs do not require it. For many systems, it is sufficient that all communications between devices are initiated by a single, master processor. In this application note, use of the 8XC751 as an I^2C bus master is described. Some of the technical features of the bus and the 83C751's special hardware associated with the I^2C are discussed. Also included is a software example demonstrating I^2C single master communications. Note that the sample routines are quite general, and therefore may be transferred easily to many applications

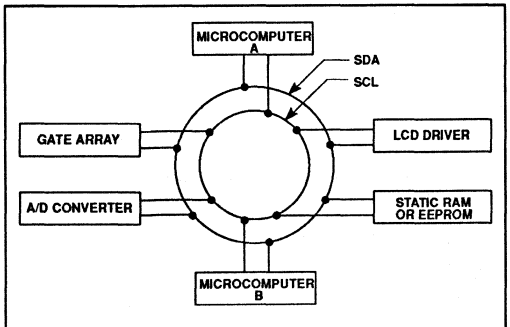


Figure 1. Typical I^2C Bus Configuration

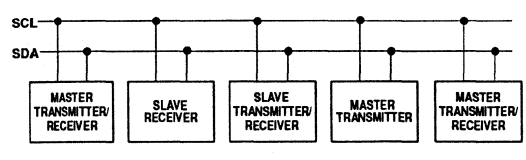


Figure 2. I^2C Bus Connection

Using the 8XC751 as an I²C bus master

AN422

The discussion of the I²C bus characteristics in this application note is by no means complete. Additional information for the I²C bus and the S83C751 Microcontroller can be found in the Microcontroller Users' Guide.

THE I²C BUS

The two lines of the I2C-bus are a serial data line (SDA) and a serial clock line (SCL). Both lines are connected to a positive supply via a pull-up resistor, and remain HIGH when the bus is not busy. Each device is recognized by a unique address – whether it is a microcomputer, LCD driver, memory or keyboard interface – and can operate as either a transmitter or receiver, depending on the function of the device. A device generating a message or data is a transmitter, and a device receiving the message or data is a receiver. Obviously, a passive function like an LCD driver could only be a receiver, while a microcontroller or a memory can both transmit and receive data.

Masters and Slaves

When a data transfer takes place on the bus, a device can either be a master or a slave. The device which initiates the transfer, and generates the clock signals for this transfer, is the master. At that time any device addressed is considered a slave. It is important to note that a master could either be a transmitter or a receiver; a master microcontroller may send data to a RAM acting as a transmitter, and then interrogate the RAM for its contents acting as a receiver – in both cases performing as the master initiating the transfer. In the same manner, a slave could be both a receiver and a transmitter.

The I²C is a multi-master bus. It is possible to have, in one system, more than one device capable of initiating transfers and controlling the bus (Figure 2). A microcontroller may act as a master for one transfer, and then be the slave for another transfer, initiated by another processor on the network. The master/slave relationships on the bus are not permanent, and may change on each transfer.

As more than one master may be connected to the bus, it is possible that two devices will try to initiate a transfer at the same time. Obviously, in order to eliminate bus collisions and communications chaos, an arbitration procedure is necessary. The I²C design has an inherent arbitration and clock synchronization procedure relying on the wired-AND connection of the devices on the bus. In a typical multi-master system, a microcontroller program should allow it to gracefully switch between master and slave modes and preserve data integrity upon loss of arbitration. In this note, a simple case is presented describing the S83C751 operating as a single master on the bus.

Data Transfers

One data bit is transferred during each clock pulse (see Figure 3). The data on the SDA line must remain stable during the HIGH period of the clock pulse in order to be valid. Changes in the data line at this time will be interpreted as control signals. A HIGH-to-LOW transition of the data line (SDA) while the clock signal (SCL) is HIGH indicates a Start condition, and a LOW-to-HIGH transition of the SDA while SCL is HIGH defines a Stop condition (see Figure 4). The bus is considered to be busy after the Start condition and free again at a certain time interval after the Stop condition. The Start and Stop conditions are always generated by the master.

The number of data bytes transferred between the Start and Stop condition from transmitter to receiver is not limited. Each byte, which must be eight bits long, is transferred serially with the most significant bit first, and is followed by an acknowledge bit. (see Figure 5). The clock pulse related to the acknowledge bit is generated by the master. The device that acknowledges has to pull down the SDA line during the acknowledge clock pulse, while the transmitting device releases the SDA line (HIGH) during this pulse (see Figure 6).

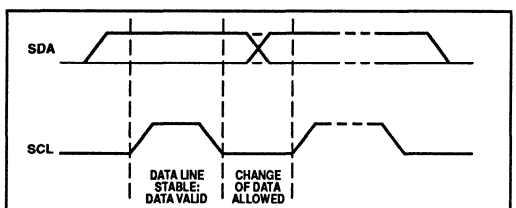


Figure 3. Bit Transfer on the I²C Bus

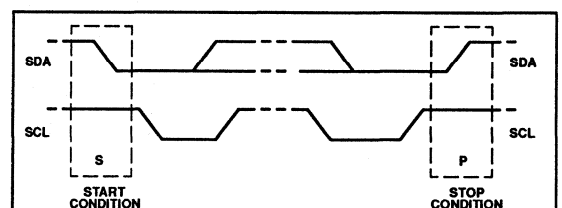


Figure 4. Start and Stop Conditions

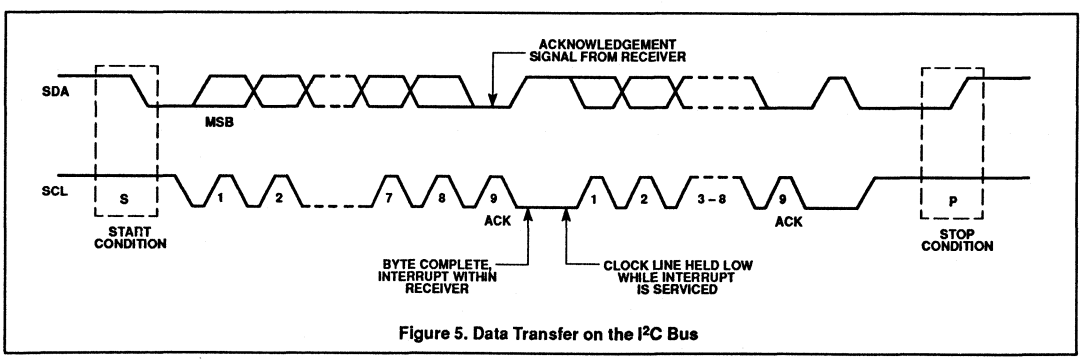


Figure 5. Data Transfer on the I²C Bus

Using the 8XC751 as an I²C bus master AN422

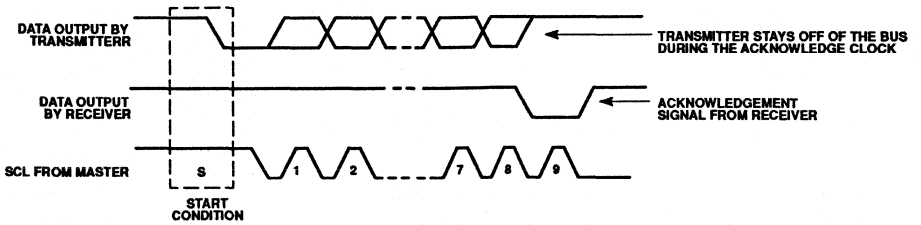


Figure 6. Acknowledge on the I²C Bus

A slave receiver must generate an acknowledge after the reception of each byte, and a master must generate one after the reception of each byte clocked out of the slave transmitter. If a receiving device cannot receive the data byte immediately, it can force the transmitter into a wait state by holding the clock line (SCL) LOW. When designing a system, it is necessary to take into account cases when acknowledge is not received. This happens, for example, when the addressed device is busy in a real time operation. In such a case the master, after an appropriate "time-out", should abort the transfer by generating a Stop condition, allowing other transfers to take place. These "other transfers" could be initiated by other masters in a multi-master system, or by this same master.

There are two exceptions to the "acknowledge after every byte" rule. The first occurs when a master is a receiver: it must signal an end of data to the transmitter by NOT signalling an acknowledge on the last byte that has been clocked out of the slave. The acknowledge related clock, generated by the master should still take place, but the SDA line will not be pulled down. In order to indicate that this is an active and intentional lack of acknowledgement, we shall term this special condition as a "negative acknowledge".

The second exception is that a slave will send a negative acknowledge when it can no longer accept additional data bytes. This occurs after an attempted transfer that cannot be accepted.

The bus design includes special provisions for interfacing to microprocessors which implement all of the I²C communications in software only – it is called "Slow Mode". When all of the devices on the network have built-in I²C hardware support, the Slow Mode is irrelevant.

Addressing and Transfer Formats

Each device on the bus has its own unique address. Before any data is transmitted on the bus, the master transmits on the bus the address of the slave to be accessed for this transaction. A well-behaved slave with a matching address, if it exists on the network, should of course acknowledge the master's addressing. The addressing is done by the first byte transmitted by the master after the Start condition.

An address on the network is seven bits long, appearing as the most significant bits of the address byte. The last bit is a direction (R/W) bit. A zero indicates that the master is transmitting (WRITE) and a one indicates that the master requests data (READ). A complete data transfer, comprised of an address byte indicating a WRITE and two data bytes is shown in Figure 7.

When an address is sent, each device in the system compares the first seven bits after the Start with its own address. If there is a match, the device will consider itself addressed by the master, and will send an acknowledge. The device could also determine if in this transaction it is assigned the role of a slave receiver or slave transmitter, depending on the R/W bit.

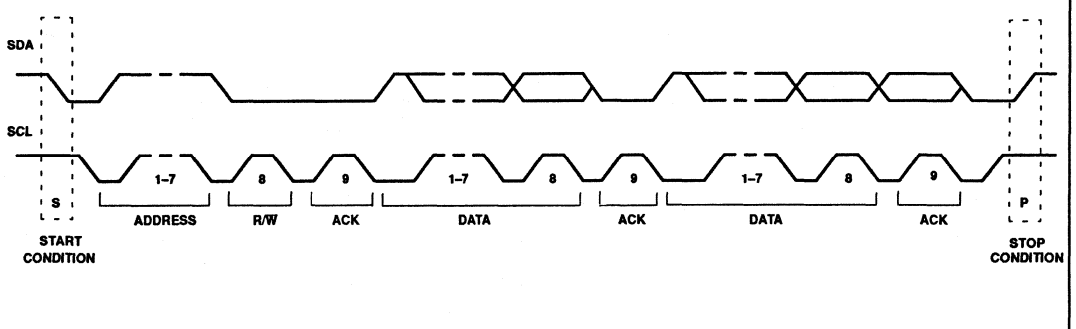


Figure 7. A Complete Data Transfer on the I²C-Bus

Using the 8XC751 as an I²C bus master AN422

Each node of the I²C network has a unique seven bit address. The address of a microcontroller is of course fully programmable, while peripheral devices usually have fixed and programmable address portions. In addition to the "standard" addressing discussed here, the I²C bus protocol allows for "general call" addressing and interfacing to CBUS devices.

When the master is communicating with one device only, data transfers follow the format of Figure 7, where the R/W bit could indicate either direction. After completing the transfer and issuing a Stop condition, if a master would like to address some other device on the network, it could of course start another transaction, issuing a new Start.

Another way for a master to communicate with several different devices would be by using a "repeated start". After the last byte of the transaction was transferred, including its acknowledge (or negative acknowledge), the master issues another Start, followed by address byte and data – without effecting a Stop. The master may communicate with a number of different devices, combining READS and WRITES. After the last transfer takes place, the master issues a Stop and releases the bus. Possible data formats are demonstrated in Figure 8. Note that the repeated start allows for both change of a slave and a change of direction, without releasing the bus. We shall see later on that the change of direction feature can come in handy even when dealing with a single device.

In a single master system, the repeated start mechanism may be more efficient than terminating each transfer with a Stop and starting again. In a multi-master environment, the determination of which format is more efficient could be more complicated, as when a master is using repeated starts it occupies the bus for a long time and thus preventing other devices from initiating transfers.

Use of Sub-Addresses

For some ICs on the I²C bus, the device address alone is not sufficient for effective communications, and a mechanism for addressing the internals of the device is necessary. A typical example when we want to access a specific word inside the device is addressing memories, or a sequence of memory locations starting at a specific internal address.

A typical I²C memory device like the PCF8570 RAM contains a built-in word address register that is incremented automatically after each data byte which is a read or written data byte. When a master communicates with the PCF- 8570 it must send a sub-address in the byte following the slave address byte. This sub-address is the internal address of the word the master wants to access for a single byte transfer, or the beginning of a sequence of locations for a multi-byte transfer. A sub–address is an 8-bit byte, unlike the device address, it does not contain a direction (R/W) bit, and like any byte transferred on the bus it must be followed by an acknowledge.

A memory write cycle is shown in Figure 9(a). The Start is followed by a slave byte with the direction bit set to WRITE, a sub-address byte, a number of data bytes and a Stop signal. The sub-address is loaded into the word address memory, and the data bytes which follow will be written one after the other starting with the sub-address location, as the register is incremented automatically.

The memory read cycle (see Figure 9(b)) commences in a similar manner, with the master sending a slave address with the direction bit set to WRITE with a following sub-address. Then, in order to reverse the direction of the transfer, the master issues a repeated Start followed again by the memory device address, but this time with the direction bit set to READ. The data bytes starting at the internal sub-address will be clocked out of the device, each followed by a master-generated acknowledge. The last byte of the read cycle will be followed by a negative acknowledge, signalling the end of transfer. The cycle is terminated by a Stop signal.

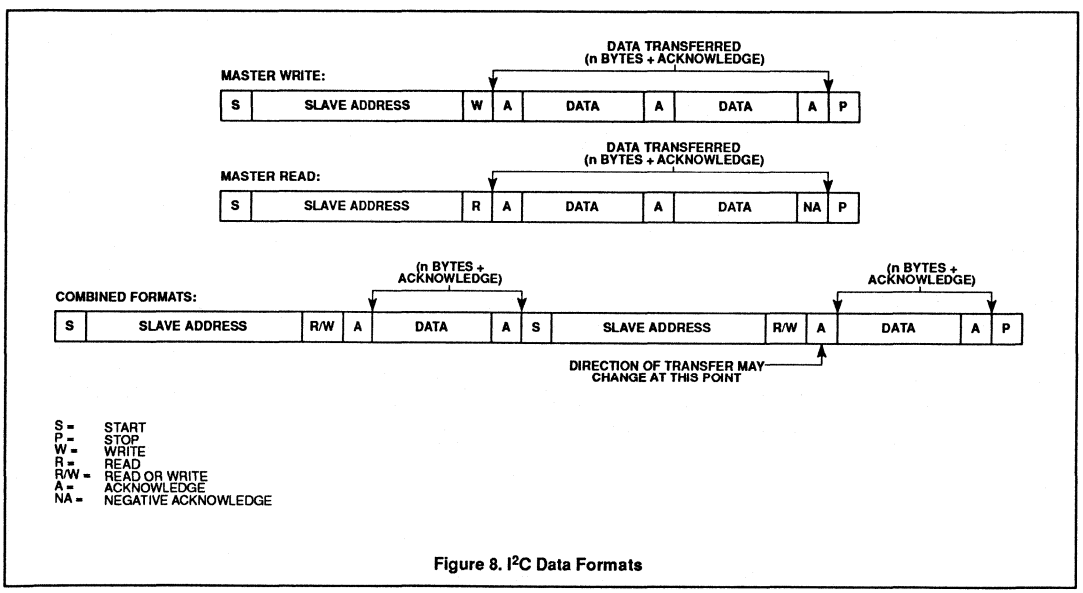


Figure 8. I²C Data Formats

Using the 8XC751 as an I²C bus master AN422

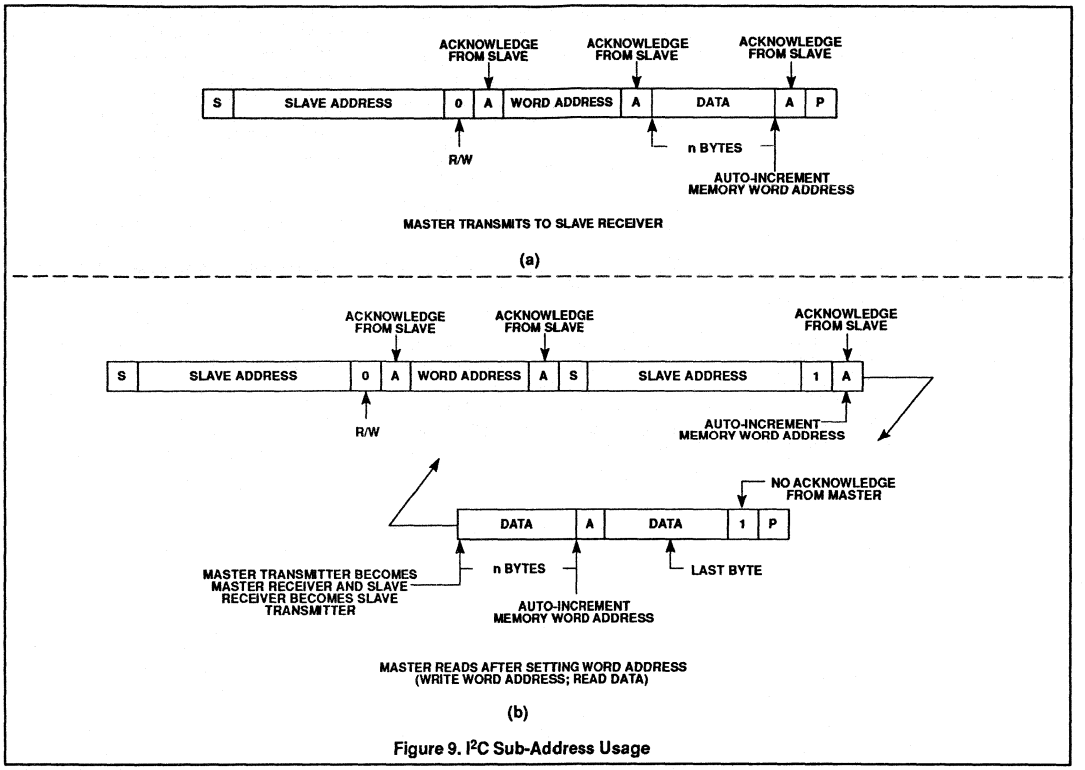


Figure 9. I²C Sub-Address Usage

8XC751 I²C HARDWARE

The on-chip I²C bus hardware support of the 8XC751 allows operation on the bus at full speed, and simplifies the software needed for effective communications on the network. The hardware activates and monitors the SDA and SCL lines, performs the necessary arbitration and framing errors checks, and takes care of clock stretching and synchronization. The hardware support includes a bus time-out timer, called Timer I. The hardware is synchronized to the software either through polled loops or interrupts.

Two of the port 0 pins are multi-functional. When the I²C is active, the pin associated with P0.0 functions as SCL, and the pin associated with P0.1 functions as SDA. These pins have an open drain output.

Two of the five 8XC751 interrupt sources may be used for I²C support. The I²C interrupt is enabled by the EI2 flag of the interrupt enable register, and its service routine should start at address 023h. An I²C interrupt is usually requested (if enabled) when a rising edge of SCL indicates a new data bit on the bus, or a special condition occurs: Start, Stop or arbitration loss. The interrupt is induced by the ATN flag – see below for the conditions for setting this flag. The Timer I overflow interrupt is enabled by the ETI flag, and the service routine starts at 01Bh.

The I²C port is controlled through three special function registers: I²C Control (I2CON), I²C Configuration (I2CFG), and I²C Data (I2DAT). The register addresses are shown in Table 1.

Although the following discussion of the hardware and register details is not complete, it should give a better understanding of the programming examples.

Timer I

In I²C applications, Timer I is dedicated to the port timing generation and bus monitoring. In non-I²C applications, it is available for use as a fixed time base.

Table 1. I²C Special Function Register Addresses

REGISTER			BIT ADDRESS							
Name	Symbol	Address	MSB							LSB
I²C Control	I2CON	98	9F	9E	9D	9C	9B	9A	99	98
I²C Data	I2DAT	99	–	–	–	–	–	–	–	–
I²C Configuration	I2CFG	D8	DF	DE	DD	DC	DB	DA	D9	D8

Using the 8XC751 as an I²C bus master

AN422

In its port timing generation function, Timer I is used to generate SCL, the I²C clock. Timer I is clocked once per machine cycle (osc/12), so that the toggle rate of SCL will be some multiple of that rate. Because the 83C751 can be run over a wide range of oscillator frequencies, it is necessary to adjust SCL for the part's oscillator frequency. This allows the I²C bus to be used at its highest transfer rates independent of the oscillator frequency. SCL is adjusted by writing to two bits (CT0 and CT1) in the I2CFG special function register (see Table 2). The inverse of the values in CT0 and CT1 are loaded into the least significant two bit locations of Timer I every time the fourth bit of the timer is toggled. (A value is actually loaded into the least significant three bits, the third bit being 0 unless both CT0 and CT1 are programmed high and in that case the third bit is 1). SCL is then toggled every time the fourth bit of Timer I is toggled. For example: if CT1 = 0 and CT0 = 1 then the least significant three bits of Timer I would be preloaded with 2 (010 binary). Timer I would then count 3, 4, 5, 6, 7, 8 (6 counts or machine cycles). On 8, the fourth bit of Timer I will toggle, SCL will toggle and the 3 least significant bits will again be preloaded with the value 2 (010).

Table 2. CT0, CT1 Timer I Settings

CT1, CT0 Values	Timer I Counts	Oscillator Freq (MHz)
1 0	7	16
0 1	6	15, 14, 13
0 0	5	12, 11
1 1	4	10 or less

Timer I counts = f_{OSC} (MHz) x 0.39 (rounded up to next integer).

For the bus monitoring function, Timer I is used as a "watchdog timer" for bus hang-ups. It creates an interrupt when the SCL line stays in one state for an extended period of time while the bus is active (between a Start condition and a following Stop condition). SCL "stuck low" indicates a faulty master or slave. SCL "stuck high" may mean a faulty device, or that noise induced unto the I²C caused all masters to withdraw from the I²C arbitration.

The time-out interval of Timer I is fixed (cannot be set): it carries out and interrupts (if enabled) when about 1024 machine cycles have elapsed since a change on SCL within a frame. In other words, whenever I²C is active and Timer I is enabled, the falling edge of SCL will reset Timer I. If SCL is not toggled low for 1024 machine cycles, Timer I will overflow and cause an interrupt. (Note: we wrote "about 1024 machine cycles" although for the sake of accuracy – this number is affected by the setting of the CT0 and CT1 bits mentioned above and may vary by up to three machine cycles) The exact number of cycles for a time-out is not critical; what is important is that it indicates SCL is stuck.

In addition to the interrupt, upon Timer I overflow the I²C port hardware is reset. This is useful for multiple master systems in situations where a bus fault might cause the bus to hang-up due to a lack of software response. When this happens, SCL will be released, and I²C operation between other devices can continue.

I2CON Register

The I²C control register (I2CON) can be written to (see Figure 10). When writing to the I2CON register, one should use bit masks as demonstrated in the example program. Trying to clear or set the bits in the register using the bit addressing capabilities of the 8XC751 may lead to undesirable results. The reason is that a command like CLR reads the register, sets the bit and writes it back, and the write-back may affect other bits.

I2CFG Register

The configuration register (I2CFG) is a read/write register (see Figure 11).

I2DAT Register

The I²C data register (I2DAT) is a read/write register, where the MSB represents the data received or data to be sent. The other seven bits are read as 0 (see Figure 12).

Transmit Active State

The transmit active state – Xmit Active – is an internal state in the I²C interface that is affected by the I²C registers as explained above. The I²C interface will only drive the SDA line low when Xmit Active is set. Xmit Active is set by writing the I2DAT register, or by writing I2CON with XSTR = 1 or XSTP = 1. The ARL bit will be set to 1 only when Xmit Active is set – in such a case Xmit Active will be automatically reset upon arbitration loss. Xmit Active is cleared by writing 1 to CXA at I2CON register or by reading the I2DAT register.

PROGRAMMING EXAMPLE

The listing demonstrates communications routines for the 8XC751 as an I²C bus master in a single-master system.

The single-master system is less complicated than a multi-master environment. The programmer does not have to worry about switching between master and slave roles, or the con- sequences of an arbitration loss.

The I²C interrupt is not used, and therefore disabled. There is no need for frame Start interrupts, as this processor is the only bus master and all data transfers are initiated by it when the appropriate routines are called by the application. No one else generates frame Starts which could be an interrupt source in a multi-master system. Within the frames we monitor bus activity with a wait-loop which polls the ATN flag. As we expect the bus to operate in its full-speed mode, we can assume that only a small amount of time will be wasted in those loops, and the use of interrupts would be less efficient.

The 8XC751 has single-bit I²C hardware interface, where the registers may directly affect the levels on the bus and the software interacting with the register takes part in the protocol implementation. The hardware and the low-level routines dealing with the registers are tightly coupled. Therefore, one should take extra care if trying to modify these lower level routines.

The beginning of the program, at address 0, contains the reset vector, where the microcontroller begins executing code after a hardware reset. In this case, the code simply jumps to the main part of the program, which begins at the label 'Reset' near the end of the listing.

The main program is a simple demonstration of the I²C routines which comprise the balance of the listing. It first enables the Timer I interrupt, and sets up some sample data to be transmitted. Beginning at the label Main-Loop, the program then proceeds to transmit one byte of data to a slave device at address 48 hexadecimal, using the routine titled 'SendData'. In our demonstration hardware, this address corresponds to an 8–bit I/O port that drives eight monitor LEDs. The program then reads back one byte of data from the same port using the routine 'RcvData'. The SendData and RcvData routines can send or receive multiple bytes of data, the number of which is determined by the variable 'ByteCnt'.

Upon return from both SendData and RcvData, the program checks the system flag named 'Retry' to see if the transfer was completed correctly. If not, it loops back and attempts the same transfer again.

Next, the program sends four bytes of data to a 256-byte EEPROM device, an 8-pin part called the PCF8582. The routine 'SendSub' is used for this purpose. The EEPROM was located at address A0 hexadecimal on our board. This device uses the sub-addressing feature to select a starting location to address in the EEPROM array. When data is written to the EEPROM, the address is automatically incre- mented so that the data bytes are stored in consecutive locations.

Finally the program reads back four bytes of data from the EEPROM using the routine 'RcvSub'. Calls to SendSub and RcvSub should also be followed by a test of the Retry flag to insure that all went according to plan.

I2CON READ	RDAT	ATN	DRDY	ARL	STR	STP	MASTER	—

RDAT — Received DATa bit. The value of SDA latched by the rising edge of SCL. Its contents is identical to RDAT in the I2DAT register. Reading the received data here allows doing so without clearing DRDY and releasing SCL.

ATN — An "ATteNtion" flag, set when any one of DRDY, ARL, STR or STP is set. This flag allows a single bit testing for terminating "wait loops", indicating a meaningful event on the bus. This flag also activates the I²C interrupt request.

DRDY — Data ReaDY flag. Set by a rising edge of SCL when I²C is active, except at an idle slave. This flag is cleared by reading or writing the I2DAT register, or by writing a 1 to CDR (at the same address, when I2CON is written).

ARL — ARbitration Loss flag. Indicates that this device lost arbitration while trying to take control of the bus.

STR — STaRt flag. Set when a Start condition is detected, except at an idle slave.

STP — STop flag. Set when a Stop condition is detected, except at an idle slave.

MASTER — This flag is set when the controller is a bus master (or a potential master, prior to arbitration).

I2CON WRITE	CXA	IDLE	CDR	CARL	CSTR	CSTP	XSTR	XSTP

CXA — "Clear Xmit Active". Writing a 1 to CXA clears the internal transmit-active state.

IDLE — Setting this bit will cause a slave to enter idle mode and ignore the I²C bus until the next Start is detected. If the software sets the MASTRQ flag, the device may stop idling by turning into a master.

CDR — Clear Data Ready. Clears the DRDY flag.

CARL — Clear Arbitration Lost. Clears the ARL flag.

CSTR — Clear STaRt. Clears the STR flag.

CSTP — Clear STop. Clears the STP flag.

XSTR — "Xmit repeated STaRt". Writing a 1 to this bit causes the hardware to issue a Repeated Start signal. A side effect will be setting the internal Xmit Active state. This should be used only when the device is a master.

XSTP — "Xmit STop". Issues a Stop condition. The Xmit active state is set.

Figure 10. I2CON Register

	SLAVEN	MASTRQ	CLRTI	TIRUN	—	—	CT1	CT0

SLAVEN — Writing a 1 to this flag enables the slave functions of the I²C interface.

MASTRQ — Request control of the bus as a master.

CLRTI — Clear the Timer I interrupt flag. This bit is always read as 0.

TIRUN — Writing a 1 will let Timer I run. When I²C is active, it will run only inside frames, and will be cleared by SCL transitions, Start and Stop. Writing a 0 will stop and clear the timer.

CT1, CT0 — These bits should be programmed according to the frequency of the crystal oscillator used in the hardware. They determine the minimum high and low times for SCL, and are used to optimize performance at different oscillator speeds.

Figure 11. I2CFG Register

I2DAT READ	RDAT	—	—	—	—	—	—	—

RDAT — Received DATa bit, captured from SDA every rising edge of SCL. Reading I2CAT clears DRDY and the Xmit Active state. If it is necessary to read the data without affecting the flags, it can be read out of RDAT in the I2CON register.

I2DAT WRITE	XDAT	—	—	—	—	—	—	—

XDAT — Xmit DATa bit. Writing XDAT determines the data for the next bit to be transmitted on the I²C bus. Writing I2DAT also clears DRDY and sets the Xmit Active state.

Figure 12. I2DAT Register

Using the 8XC751 as an I²C bus master AN422

This entire process is repeated indefinitely by jumping back to MainLoop.

Back at the beginning of the program, the next location after the reset vector is the Timer I interrupt service routine. The microcontroller will go to address 1B hexadecimal if Timer I overflows. This routine stops the timer, clears the timer interrupt, clears the pending interrupt so that other interrupts will be enabled, restores the stack pointer, and jumps to the 'Recover' routine to try to correct whatever stopped the I²C bus and allowed Timer I to overflow.

Next in the listing come the main I²C service routines. These are the routines SendData, RcvData, SendSub, and RcvSub that were called from the main program. Both of the send routines use the data area labeled 'XmtDat' as the transmit data buffer. In this sample program, four bytes were reserved for this area, but it could be larger or smaller depending on the application. The two receive routines use another four byte buffer labeled 'RcvDat' to store received data. All of these routines use the variables 'SlvAdr' and 'ByteCnt' to determine the slave address and the number of bytes to be sent or received, respectively. The SendSub and RcvSub routines use the variable 'SubAdr' as the sub-address to send to the slave device.

Following the main I²C service routines in the listing are the subroutines that are called by the main routines to deal intimately with the I²C hardware.

The 'SendAddr' subroutine requests mastership of the I²C bus and calls the routine 'XmitAddr' to complete sending the slave address. The bulk of the XmitAddr routine is shared with the 'XmitByte' subroutine which sends data bytes on the I²C bus. XmitByte is also used to send I²C sub-addresses. Both subroutines check for an acknowledge from the slave device after every byte is sent on the I²C bus.

The next subroutine 'RDAck' calls the 'RcvByte' routine to read in a byte of data. It then sends an acknowledge to the slave device. RDAck is used to receive all data except for the last byte of a receive data frame, where the acknowledge is omitted by the bus master. The RcvByte subroutine is called directly for the last byte of a frame.

The 'SendStop' subroutine causes a stop condition on the I²C, thus ending a frame. The 'RepStart' subroutine sends a repeated start condition on the I²C bus, to allow the master to start a new frame without first having to send a intervening stop.

The lower level subroutines deal directly with the hardware. The tight coupling between hardware and software is best demonstrated by the following explanations, relating to two cases in which the code is not self evident.

Sending the Address

When sending the address byte in the Send Addr subroutine, the first bit is written to I2DAT prior to the loop where the other seven bits are sent (SendAd2). The reason is that we need to clear the Start condition in order to release the SCL line, and this is done explicitly by the subsequent command. When SCL is released, the correct bit (MSB of address) must already be in I2DAT.

Capturing the Received Data

Typically, a program receiving data waits in a loop for ATN, and when detected, checks DRDY. If DRDY = 1 then there was a rising SCL, and the new data can be read from RDAT in I2CON or I2DAT. Reading or writing I2DAT clears DRDY, thus releasing SCL.

When reading the last bit in a byte, it should be read from I2CON, and not I2DAT (see the end of the RcvByte routine). This way the Data Ready (DRDY) flag is not cleared, and the low period on SCL is stretched. The reason for doing so is that upon reception of the last bit of a received byte the master must react with an acknowledge. In order to ensure that we "wait" with the acknowledge clock (release of SCL) until the acknowledge level is issued on SDA, the last bit is read out of I2CON and not I2DAT. SCL is stretched low until the acknowledge level is written into I2DAT by the software.

Bus Faults and Other Exceptions

Bus exceptions are detected either by Timer I time-out, or "illegal" logic states tested for and detected by the software. Upon Timer I time-out, a bus recovery is attempted by the Recover routine. The final section of the listing is this 'Recover' routine. Its job is to try to restore control of the I²C bus to the main program. First, the subroutine 'FixBus' is called. It checks to see if only the SDA line is 'stuck', and if so, tries to correct it by sending some extra clocks on the SCL line, and forcing a stop condition on the bus. If this does not work, another subroutine 'BusReset' is called. This generally happens when a severe bus error occurs, such as a shorted clock line. The philosophy used in this code is that the only chance of recovering from a severe error is to cause a reset of the I¹C hardware by deliberately forcing Timer I to time out. This method allows recovery from a temporary short or other serious condition on the I²C bus.

```
I2CAPP          83C751 Single Master I2C Routines                09/07/89


                 1
                 2    ;*******************************************************
                 3    ;
                 4    ;   Sample I2C Single Master Routines for the 83C751
                 5
                 6    ;*******************************************************
                 7
                 8    $TITLE(83C751 Single Master I2C Routines)
                 9    $DATE(09/07/89)
                10    $MOD751
                11    $DEBUG
                12
                13
                14    ; Value definitions.
                15
0002            16    CTVAL       EQU     02h       ;CT1, CT0 bit values for I2C.
                17
                18
                19    ; Masks for I2CFG bits.
                20
0010            21    BTIR        EQU     10h       ;Mask for TIRUN bit.
0040            22    BMRQ        EQU     40h       ;Mask for MASTRQ bit.
                23
                24
                25    ; Masks for I2CON bits.
                26
0080            27    BCXA        EQU     80h       ;Mask for CXA bit.
0040            28    BIDLE       EQU     40h       ;Mask for IDLE bit.
0020            29    BCDR        EQU     20h       ;Mask for CDR bit.
0010            30    BCARL       EQU     10h       ;Mask for CARL bit.
0008            31    BCSTR       EQU     08h       ;Mask for CSTR bit.
0004            32    BCSTP       EQU     04h       ;Mask for CSTP bit.
0002            33    BXSTR       EQU     02h       ;Mask for XSTR bit.
0001            34    BXSTP       EQU     01h       ;Mask for XSTP bit.
                35
                36
                37    ; RAM locations used by I2C routines.
                38
0021            39    BitCnt      DATA    21h      ;I2C bit counter.
0022            40    ByteCnt     DATA    22h
0023            41    SlvAdr      DATA    23h      ;Address of active slave.
0024            42    SubAdr      DATA    24h
                43
0025            44    RcvDat      DATA    25h      ;I2C receive data buffer (4 bytes).
                45                                   ; addresses 25h through 28h.
                46
0029            47    XmtDat      DATA    29h      ;I2C transmit data buffer (4 bytes).
                48                                   ;  addresses 29h through 2Ch.
                49
002D            50    StackSave   DATA    2Dh      ;Saves stack addr for bus recovery.
                51
0020            52    Flags       DATA    20h      ;I2C software status flags.
0000            53    NoAck       BIT     Flags.0  ;Indicates missing acknowledge.
0001            54    Fault       BIT     Flags.1  ;Indicates a bus fault of some kind.
0002            55    Retry       BIT     Flags.2  ;Indicates that last I2C transmission
                56                                   ; failed and should be repeated.
                57
0080            58    SCL         BIT     P0.0     ;Port bit for I2C serial clock line.
```

```
0081          59   SDA        BIT     P0.1      ;Port bit for I2C serial data line.
              60
              61   ;*************************************************************
              62   ;                                  Begin Code
              63   ;*************************************************************
              64
              65   ; Reset and interrupt vectors.
              66
0000 21E1     67               AJMP    Reset             ;Reset vector at address 0.
              68
              69
              70   ; A timer I timeout usually indicates a 'hung' bus.
              71
001B          72               ORG     1Bh               ;Timer I (I2C timeout)
                                                         ; interrupt.
001B D2DD     73   TimerI:     SETB    CLRTI             ;Clear timer I interrupt.
001D C2DC     74               CLR     TIRUN
001F 1126     75               ACALL   ClrInt            ;Clear interrupt pending.
0021 852D81   76               MOV     SP,StackSave      ;Restore stack for return
                                                         ; to main.
0024 218A     77               AJMP    Recover           ;Attempt bus recovery.
0026 32       78   ClrInt:     RETI
              79
              80
              81   ;*************************************************************
              82   ;               Main Transmit and Receive Routines
              83   ;*************************************************************
              84
              85   ; Send data byte(s) to slave.
              86   ;  Enter with slave address in SlvAdr, data in XmtDat buffer,
              87   ;   # of data bytes to send in ByteCnt.
              88
0027 C200     89   SendData:  CLR     NoAck             ;Clear error flags.
0029 C201     90               CLR     Fault
002B C202     91               CLR     Retry
002D 85812D   92               MOV     Stack             ;Save,SP  Save stack address
                                                           for bus fault.
0030 E523     93               MOV     A,SlvAdr          ;Get slave address.
0032 310C     94               ACALL   SendAddr          ;Get bus and send slave addr.
0034 200012   95               JB      NoAck,SDEX        ;Check for missing
                                                         ; acknowledge.
0037 200112   96               JB      Fault,SDatErr     ;Check for bus fault.
003A 7829     97               MOV     R0,#XmtDat        ;Set start of transmit
                                                         ; buffer.
              98
003C E6       99   SDLoop:     MOV     A,@R0             ;Get data for slave.
003D 08       100              INC     R0
003E 3125     101              ACALL   XmitByte          ;Send data to slave.
0040 200006   102              JB      NoAck,SDEX        ;Check for missing
                                                         ; acknowledge.
0043 200106   103              JB      Fault,SDatErr     ;Check for bus fault.
0046 D522F3   104              DJNZ    ByteCnt,SDLoop
              105
0049 3166     106    SDEX:     ACALL   SendStop          ;Send an I2C stop.
004B 22       107              RET
              108
              109
              110    ; Handle a transmit bus fault.
              111
004C 218A     112    SDatErr:   AJMP    Recover         ;Attempt bus recovery.
              113
              114
```

```
                    115    ; Receive data byte(s) from slave.
                    116    ; Enter with slave address in SlvAdr,
                           ;  # of data bytes requested in ByteCnt.
                    117    ;  Data returned in RcvDat buffer.
                    118
004E C200           119    RcvData:   CLR     NoAck            ;Clear error flags.
0050 C201           120               CLR     Fault
0052 C202           121               CLR     Retry
0054 85812D         122               MOV     StackSave,SP     ;Save stack address
                                                               ; for bus fault.
0057 E523           123               MOV     A,SlvAdr         ;Get slave address.
0059 D2E0           124               SETB    ACC.0            ;Aet bus read bit.
005B 310C           125               ACALL   SendAddr         ;Send slave address.
005D 200023         126               JB      NoAck,RDEX       ;Check for missing
                                                               ; acknowledge.
0060 200123         127               JB      Fault,RDatErr    ;Check for bus fault.
                    128
0063 7825           129               MOV     R0,#RcvDat       ;Set start of receive
                                                               ; buffer.
0065 D52202         130               DJNZ    ByteCnt,RDLoop   ;Check for count = 1
                                                               ; byte only.
0068 800A           131               SJMP    RDLast
                    132
006A 3143           133    RDLoop:    ACALL   RDAck            ;Get data and send
                                                               ; an acknowledge.
006C 200117         134               JB      Fault,RDatErr    ;Check for bus fault.
006F F6             135               MOV     @R0,A            ;Save data.
0070 08             136               INC     R0
0071 D522F6         137               DJNZ    ByteCnt,RDLoop   ;Repeat until last
                                                               ; byte.
                    138
0074 314F           139    RDLast:    ACALL   RcvByte          ;Get last data byte
                                                               ; from slave.
0076 20010D         140               JB      Fault,RDatErr    ;Check for bus
                                                               ; fault.
0079 F6             141               MOV     @R0,A            ;Save data.
                    142
007A 759980         143               MOV     I2DAT,#80h       ;Send negative
                                                               ; acknowledge.
007D 309EFD         144               JNB     ATN,$            ;Wait for NAK sent.
0080 309D03         145               JNB     DRDY,RDatErr     ;Check for bus
                                                               ; fault.
                    146
0083 3166           147    RDEX:      ACALL   SendStop         ;Send an I2C bus
                                                               ; stop.
0085 22             148               RET
                    149
                    150
                    151    ; Handle a receive bus fault.
                    152
0086 218A           153    RDatErr:   AJMP    Recover          ;Attempt bus recovery.
                    154
                    155
                    156    ; Send data byte(s) to slave with subaddress.
                    157    ; Enter with slave address in ACC, subaddress in
                           ; SubAdr, # of bytes to send in ByteCnt,
                    158    ; data in XmtDat buffer.
                    159
0088 C200           160    SendSub:   CLR     NoAck            ;Clear error flags.
008A C201           161               CLR     Fault
```

```
008C C202          162                CLR     Retry
008E 85812D        163                MOV     StackSave,SP    ;Save stack address
                                                              ; for bus fault.
0091 E523          164                MOV     A,SlvAdr        ;Get slave address.
0093 310C          165                ACALL   SendAddr        ;Get bus and send
                                                              ; slave address.
0095 20001C        166                JB      NoAck,SSEX      ;Check for missing
                                                              ; acknowledge.
0098 20011C        167                JB      Fault,SSubErr   ; Check for bus
                                                              ; fault.
                   168
009B E524          169                MOV     A,SubAdr        ;Get slave subaddress.
009D 3125          170                ACALL   XmitByte        ;Send subaddress.
009F 200012        171                JB      NoAck,SSEX      ;Check for missing
                                                              ; acknowledge.
00A2 200112        172                JB      Fault,SSubErr   ;Check for bus fault.
00A5 7829          173                MOV     R0,#XmtDat      ;Set start of
                                                              ; transmit buffer.
                   174
00A7 E6            175    SSLoop:     MOV     A,@R0           ;Get data for slave.
00A8 08            176                INC     R0
00A9 3125          177                ACALL   XmitByte        ;Send data to slave.
00AB 200006        178                JB      NoAck,SSEX      ;Check for missing
                                                              ; acknowledge.
00AE 200106        179                JB      Fault,SSubErr   ;Check for bus fault.
00B1 D522F3        180                DJNZ    ByteCnt,SSLoop
                   181
00B4 3166          182    SSEX:       ACALL   SendStop        ;Send an I2C stop.
00B6 22            183                RET
                   184
                   185
                   186    ; Handle a transmit bus fault.
                   187
00B7 218A          188    SSubErr:    AJMP    Recover         ;Attempt bus recovery.
                   189
                   190
                   191    ; Receive data byte(s) from slave with subaddress.
                   192    ; Enter with slave address in SlvAdr, subaddress in SubAdr,
                          ; # of data bytes  requested in ByteCnt.
                   193    ; Data returned in RcvDat buffer.
                   194
00B9 C200          195    RcvSub:     CLR     NoAck           ;Clear error flags.
00BB C201          196                CLR     Fault
00BD C202          197                CLR     Retry
00BF 85812D        198                MOV     StackSave,SP    ;Save stack address
                                                              ; for bus fault.
00C2 E523          199                MOV     A,SlvAdr        ;Get slave address.
00C4 310C          200                ACALL   SendAddr        ;Send slave address.
00C6 20003E        201                JB      NoAck,RSEX      ;Check for missing
                                                              ; acknowledge.
00C9 20013E        202                JB      Fault,RSubErr   ;Check for bus fault.
                   203
00CC E524          204                MOV     A,SubAdr        ;Get slave subaddress.
00CE 3125          205                ACALL   XmitByte        ;Send subaddress.
00D0 200034        206                JB      NoAck,RSEX      ;Check for missing
                                                              ; acknowledge.
00D3 200134        207                JB      Fault,RSubErr   ;Check for bus fault.

                   208
00D6 317A          209                ACALL   RepStart        ;Send repeated start.
00D8 20012F        210                JB      Fault,RSubErr   ;Check for bus fault.
00DB E523          211                MOV     A,SlvAdr        ;Get slave address.
```

```
00DD D2E0       212             SETB    ACC.0           ;Set bus read bit.
00DF 3115       213             ACALL   SendAd2         ;Send slave address.
00E1 200023     214             JB      NoAck,RSEX      ;Check for missing
                                                        ; acknowledge.
00E4 200123     215             JB      Fault,RSubErr   ;Check for bus fault.
                216
00E7 7825       217             MOV     R0,#RcvDat      ;Set start of
                                                        ; receive buffer.
00E9 D52202     218             DJNZ    ByteCnt,RSLoop  ;Check for count = 1
                                                        ; byte only.
00EC 800A       219             SJMP    RSLast
                220
00EE 3143       221 RSLoop:     ACALL   RDAck           ;Get data and send
                                                        ; an acknowledge.
00F0 200117     222             JB      Fault,RSubErr   ;Check for bus fault.
00F3 F6         223             MOV     @R0,A           ;Save data.
00F4 08         224             INC     R0
00F5 D522F6     225             DJNZ    ByteCnt,RSLoop  ;Repeat until last byte.
                226
00F8 314F       227 RSLast:     ACALL   RcvByte         ;Get last data byte
                                                        ; from slave.
00FA 20010D     228             JB      Fault,RSubErr   ;Check for bus fault.
00FD F6         229             MOV     @R0,A           ;Save data.
                230
00FE 759980     231             MOV     I2DAT,#80h      ;Send negative
                                                        ; acknowledge.
0101 309EFD     232             JNB     ATN,$           ;Wait for NAK sent.
0104 309D03     233             JNB     DRDY,RSubErr    ;Check for bus fault.
                234
0107 3166       235 RSEX:       ACALL   SendStop        ;Send an I2C bus stop.
0109 22         236             RET
                237
                238
                239 ; Handle a receive bus fault.
                240
010A 218A       241 RSubErr:    AJMP    Recover         ;Attempt bus recovery.
                242
                243
                244 ;*******************************************************
                245 ;                       Subroutines
                246 ;*******************************************************
                247
                248 ; Send address byte.
                249 ;  Enter with address in ACC.
                250
010C 75D852     251 SendAddr:   MOV     I2CFG,#BMRQ+BTIR+CTVAL  ;Request I2C bus.
010F 309EFD     252             JNB     ATN,$                   ;Wait for bus
                                                                ; granted.
0112 309908     253             JNB     Master,SAErr            ;Should have
                                                                ; become the bus
                                                                ; master.
0115 F599       254 SendAd2:    MOV     I2DAT,A                 ;Send first bit,
                                                                ; clears DRDY.
0117 75981C     255             MOV     I2CON,#BCARL+BCSTR+BCSTP ;Clear start,
                                                                ; releases SCL.
011A 3120       256             ACALL   XmitAddr                ;Finish sending
                                                                ; address.

011C 22         257             RET
                258
011D D201       259 SAErr:      SETB    Fault                   ;Return bus fault
                                                                ; status.
```

```
011F 22          260              RET
                 261
                 262
                 263    ; Byte transmit routine.
                 264    ;   Enter with data in ACC.
                 265    ;   XmitByte : transmits 8 bits.
                 266    ;   XmitAddr : transmits 7 bits (for address only).
                 267
0120 752108      268    XmitAddr: MOV     BitCnt,#8          ;Set 7 bits of
                                                             ; address count.
0123 8005        269              SJMP    XmBit2
                 270
0125 752108      271    XmitByte: MOV     BitCnt,#8          ;Set 8 bits of data
                                                             ; count.
0128 F599        272    XmBit:    MOV     I2DAT,A            ;Send this bit.
012A 23          273    XmBit2:   RL      A                  ;Get next bit.
012B 309EFD      274              JNB     ATN,$              ;Wait for bit sent.
012E 309D0F      275              JNB     DRDY,XMErr         ;Should be data ready.
0131 D521F4      276              DJNZ    BitCnt,XmBit       ;Repeat until all bits sent.
0134 7598A0      277              MOV     I2CON,#BCDR+BCXA   ;Switch to
                                                             ; receive mode.
0137 309EFD      278              JNB     ATN,$              ;Wait for acknowledge
                                                             ; bit.
013A 309F02      279              JNB     RDAT,XMBX          ;Was there an ack?
013D D200        280              SETB    NoAck              ;Return no acknowledge
                                                             ; status.
013F 22          281    XMBX:     RET
                 282
0140 D201        283    XMErr:    SETB    Fault              ;Return bus fault
                                                             ; status.
0142 22          284              RET
                 285
                 286
                 287    ; Byte receive routines.
                 288    ;   RDAck    : receives a byte of data, then sends
                        ;   an acknowledge.
                 289    ;   RcvByte : receives a byte of data.
                 290    ;   Data returned in ACC.
                 291
0143 314F        292    RDAck:    ACALL   RcvByte            ;Receive a data byte.
0145 759900      293              MOV     I2DAT,#0           ;Send receive
                                                             ; acknowledge.
0148 309EFD      294              JNB     ATN,$              ;Wait for acknowledge
                                                             ; sent.
014B 309D15      295              JNB     DRDY,RdErr         ;Check for bus fault.
014E 22          296              RET
                 297
014F 752108      298    RcvByte:  MOV     BitCnt,#8          ;Set bit count.
0152 E4          299              CLR     A                  ;Init received byte
                                                             ; to 0.
0153 4599        300    RBit:     ORL     A,I2DAT            ;Get bit, clear ATN.
0155 23          301              RL      A                  ;Shift data.
0156 309EFD      302              JNB     ATN,$              ;Wait for next bit.
0159 309D07      303              JNB     DRDY,RdErr         ;Should be data ready.

015C D521F4      304              DJNZ    BitCnt,RBit        ;Repeat until 7 bits
                                                             ; are in.
015F A29F        305              MOV     C,RDAT             ;Get last bit, don't
                                                             ; clear ATN.
0161 33          306              RLC     A                  ;Form full data byte.
0162 22          307              RET
                 308
```

```
0163 D201          309   RdErr:     SETB    Fault              ;Return bus fault status.
0165 22            310              RET
                   311
                   312
                   313   ; I2C stop routine.
                   314
0166 C2DE          315   SendStop:  CLR     MASTRQ             ;Release bus
                                                                ; mastership.
0168 759821        316              MOV     I2CON,#BCDR+BXSTP  ;Generate a bus stop.
016B 309EFD        317              JNB     ATN,$              ;Wait for atn.
016E 759820        318              MOV     I2CON,#BCDR        ;Clear data ready.
0171 309EFD        319              JNB     ATN,$              ;Wait for stop sent.
0174 759894        320              MOV     I2CON,#BCARL+BCSTP+BCXA ;Clear I2C bus.
0177 C2DC          321              CLR     TIRUN              ;Stop timer I.
0179 22            322              RET
                   323
                   324
                   325   ; I2C repeated start routine.
                   326   ;   Enter with address in ACC.
                   327
017A 759822        328   RepStart:  MOV     I2CON,#BCDR+BXSTR  ;Send repeated start.
017D 309EFD        329              JNB     ATN,$              ;Wait for ATN.
0180 759820        330              MOV     I2CON,#BCDR        ;Clear data ready.
0183 309EFD        331              JNB     ATN,$              ;Wait for repeated
                                                                ; start sent.
0186 759818        332              MOV     I2CON,#BCARL+BCSTR ;Clear start.
0189 22            333              RET
                   334
                   335
                   336   ; Bus fault recovery routine.
                   337
018A 31A4          338   Recover:   ACALL   FixBus             ;See if bus is dead or
                                                                ; can be 'fixed'.
018C 400D          339              JC      BusReset           ;If not 'fixed', try
                                                                ; extreme measures.
018E D202          340              SETB    Retry              ;If bus OK, return to
                                                                ; main routine.
0190 C201          341              CLR     Fault
0192 C200          342              CLR     NoAck
0194 D2DD          343              SETB    CLRTI
0196 D2DC          344              SETB    TIRUN              ;Enable timer I.
0198 D2AB          345              SETB    ETI                ;Turn on timer I
                                                                ; interrupts.
019A 22            346              RET
                   347
                   348   ;This routine tries a more extreme method of bus recovery.
                   349   ; This is used if SCL or SDA are stuck and cannot
                         ; otherwise be freed.
                   350   ; (will return to the Recover routine when Timer I times out)
                   351
019B C2DE          352   BusReset:  CLR     MASTRQ             ;Release bus.
019D 7598BC        353              MOV     I2CON,#0BCh        ;Clear all I2C flags.
01A0 D2DC          354              SETB    TIRUN
01A2 80FE          355              SJMP    $                  ;Wait for timer I
                                                                ; timeout (this will re-
                   356                                          ; set the I2C hardware).
                   357
                   358
                   359   ; This routine attempts to regain control of the I2C
                         ;                bus after a bus fault.
```

```
                    360   ;  Returns carry clear if successful, carry set if failed.
                    361
01A4 C2DE           362   FixBus:   CLR     MastRQ            ;Turn off I2C functions.
01A6 D3             363             SETB    C
01A7 D280           364             SETB    SCL               ;Insure I/O port is not
                                                              ; locking I2C.
01A9 D281           365             SETB    SDA
01AB 308029         366             JNB     SCL,FixBusEx      ;If SCL is low, bus
                                                              ; cannot be 'fixed'.
01AE 208113         367             JB      SDA,RStop         ;If SCL & SDA are high,
                                                              ; force a stop.
01B1 752109         368             MOV     BitCnt,#9         ;Set max # of tries to
                                                              ; clear bus.
01B4 C280           369   ChekLoop: CLR     SCL               ;Force an I2C clock.
01B6 31D8           370             ACALL   SDelay
01B8 208109         371             JB      SDA,RStop         ;Did it work?
01BB D280           372             SETB    SCL
01BD 31D8           373             ACALL   SDelay
01BF D521F2         374             DJNZ    BitCnt,ChekLoop   ;Repeat clocks until
                                                              ; either SDA clears or
                    375                                       ; we run out of tries.
01C2 8013           376             SJMP    FixBusEx          ;Failed to fix bus by
                                                              ; this method.
                    377
01C4 C281           378   RStop:    CLR     SDA               ;Try forcing a stop
                                                              ; since SCL & SDA
01C6 31D8           379             ACALL   SDelay            ; are both high.
01C8 D280           380             SETB    SCL
01CA 31D8           381             ACALL   SDelay
01CC D281           382             SETB    SDA
01CE 31D8           383             ACALL   SDelay
01D0 308004         384             JNB     SCL,FixBusEx      ;Are SCL & SDA still
                                                              ; high? If so, assume bus
01D3 308101         385             JNB     SDA,FixBusEx      ; is now OK, and return
01D6 C3             386             CLR     C                 ; with carry cleared.
01D7 22             387   FixBusEx: RET
                    388
                    389
                    390   ; Short delay routine (10 machine cycles).
                    391
01D8 00             392   SDelay:   NOP
01D9 00             393             NOP
01DA 00             394             NOP
01DB 00             395             NOP
01DC 00             396             NOP
01DD 00             397             NOP
01DE 00             398             NOP
01DF 00             399             NOP
01E0 22             400             RET
                    401
                    402
                    402   ;*************************************************************
                    404   ;                       Main Program
                    405   ;*************************************************************
                    406
01E1 758107         407   Reset:    MOV     SP,#07h         ;Set stack location.
01E4 D2AB           408             SETB    ETI             ;Enable timer I interrupts.
01E6 D2AF           409             SETB    EA              ;Enable global interrupts.
01E8 75290B         410             MOV     XmtDat,#11      ;Set up transmit data.
01EB 752A16         411             MOV     XmtDat+1,#22    ;Set up transmit data.
01EE 752B2C         412             MOV     XmtDat+2,#44    ;Set up transmit data.
```

```
01F1 752C58          413              MOV     XmtDat+3,#88    ;Set up transmit data.
01F4 752500          414              MOV     RcvDat,#0       ;Clear receive data.
01F7 752600          415              MOV     RcvDat+1,#0     ;Clear receive data.
01FA 752700          416              MOV     RcvDat+2,#0     ;Clear receive data.
01FD 752800          417              MOV     RcvDat+3,#0     ;Clear receive data.
                     418
0200 752348          419  MainLoop:   MOV     SlvAdr,#48h     ;Set slave address
                                                               ; (8-bit I/O port).
0203 752201          420              MOV     ByteCnt,#1      ;Set up byte count.
0206 1127            421              ACALL   SendData        ;Send data to slave.
0208 2002F5          422              JB      Retry,MainLoop
                     423
020B 752201          424  ML2:        MOV     ByteCnt,#1      ;Set up byte count.
020E 114E            425              ACALL   RcvData         ;Read data from slave.
0210 2002F8          426              JB      Retry,ML2
                     427
0213 7523A0          428  SL1:        MOV     SlvAdr,#0A0h    ;Set slave address
                                                               ; (RAM chip).
0216 752400          429              MOV     SubAdr,#0h      ;Set slave subaddress.
0219 752204          430              MOV     ByteCnt,#4      ;Set up byte count.
021C 1188            431              ACALL   SendSub
021E 2002F2          432              JB      Retry,SL1
                     433
0221 752204          434  SL2:        MOV     ByteCnt,#4      ;Set up byte count.
0224 11B9            435              ACALL   RcvSub
0226 2002F8          436              JB      Retry,SL2
                     437
0229 0529            438              INC     XmtDat
022B 052A            439              INC     XmtDat+1
022D 052B            440              INC     XmtDat+2
022F 052C            441              INC     XmtDat+3
0231 80CD            442              SJMP    MainLoop        ;Do it all again.
                     443
                     444              ENDASSEMBLY COMPLETE, 0 ERRORS FOUND


I2CAPP          83C751 Single Master I2C Routines

ACC. . . . . . . . . . . . . . . D ADDR  00E0H  PREDEFINED
ATN. . . . . . . . . . . . . . . B ADDR  009EH  PREDEFINED
BCARL. . . . . . . . . . . . . .   NUMB  0010H
BCDR . . . . . . . . . . . . . .   NUMB  0020H
BCSTP. . . . . . . . . . . . . .   NUMB  0004H
BCSTR. . . . . . . . . . . . . .   NUMB  0008H
BCXA . . . . . . . . . . . . . .   NUMB  0080H
BIDLE. . . . . . . . . . . . . .   NUMB  0040H  NOT USED
BITCNT . . . . . . . . . . . . . D ADDR  0021H
BMRQ . . . . . . . . . . . . . .   NUMB  0040H
BTIR . . . . . . . . . . . . . .   NUMB  0010H
BUSRESET . . . . . . . . . . . . C ADDR  019BH
BXSTP. . . . . . . . . . . . . .   NUMB  0001H
BXSTR. . . . . . . . . . . . . .   NUMB  0002H
BYTECNT. . . . . . . . . . . . . D ADDR  0022H
CHEKLOOP . . . . . . . . . . . . C ADDR  01B4H
CLRINT . . . . . . . . . . . . . C ADDR  0026H
CLRTI. . . . . . . . . . . . . . B ADDR  00DDH  PREDEFINED
CTVAL. . . . . . . . . . . . . .   NUMB  0002H
```

```
DRDY . . . . . B ADDR  009DH  PREDEFINED
EA . . . . . B ADDR  00AFH  PREDEFINED
ETI. . . . . B ADDR  00ABH  PREDEFINED
FAULT. . . . . B ADDR  0001H
FIXBUS . . . . . C ADDR  01A4H
FIXBUSEX . . . . . C ADDR  01D7H
FLAGS. . . . . D ADDR  0020H
I2CFG. . . . . D ADDR  00D8H  PREDEFINED
I2CON. . . . . D ADDR  0098H  PREDEFINED
I2DAT. . . . . D ADDR  0099H  PREDEFINED
MAINLOOP . . . . . C ADDR  0200H
MASTER . . . . . B ADDR  0099H  PREDEFINED
MASTRQ . . . . . B ADDR  00DEH  PREDEFINED
ML2. . . . . C ADDR  020BH
NOACK. . . . . B ADDR  0000H
P0 . . . . . D ADDR  0080H  PREDEFINED
RBIT . . . . . C ADDR  0153H
RCVBYTE. . . . . C ADDR  014FH
RCVDAT . . . . . D ADDR  0025H
RCVDATA. . . . . C ADDR  004EH
RCVSUB . . . . . C ADDR  00B9H
RDACK. . . . . C ADDR  0143H
RDAT . . . . . B ADDR  009FH  PREDEFINED
RDATERR. . . . . C ADDR  0086H
RDERR. . . . . C ADDR  0163H
RDEX . . . . . C ADDR  0083H
RDLAST . . . . . C ADDR  0074H
RDLOOP . . . . . C ADDR  006AH
RECOVER. . . . . C ADDR  018AH
REPSTART . . . . . C ADDR  017AH
RESET. . . . . C ADDR  01E1H
RETRY. . . . . B ADDR  0002H
RSEX . . . . . C ADDR  0107H
RSLAST . . . . . C ADDR  00F8H
RSLOOP . . . . . C ADDR  00EEH
RSTOP. . . . . C ADDR  01C4H
RSUBERR. . . . . C ADDR  010AH
SAERR. . . . . C ADDR  011DH
SCL. . . . . B ADDR  0080H
SDA. . . . . B ADDR  0081H
SDATERR. . . . . C ADDR  004CH
SDELAY . . . . . C ADDR  01D8H
SDEX . . . . . C ADDR  0049H
SDLOOP . . . . . C ADDR  003CH
SENDAD2. . . . . C ADDR  0115H
SENDADDR . . . . . C ADDR  010CH
SENDDATA . . . . . C ADDR  0027H
SENDSTOP . . . . . C ADDR  0166H
SENDSUB. . . . . C ADDR  0088H
```

```
SL1. . . . . C ADDR  0213H
SL2. . . . . C ADDR  0221H
SLVADR . . . . . D ADDR  0023H
SP . . . . . D ADDR  0081H  PREDEFINED
SSEX . . . . . C ADDR  00B4H
SSLOOP . . . . . C ADDR  00A7H
SSUBERR. . . . . C ADDR  00B7H
STACKSAVE. . . . . D ADDR  002DH
SUBADR . . . . . D ADDR  0024H
TIMERI . . . . . C ADDR  001BH  NOT USED
TIRUN. . . . . B ADDR  00DCH  PREDEFINED
XMBIT. . . . . C ADDR  0128H
XMBIT2 . . . . . C ADDR  012AH
XMBX . . . . . C ADDR  013FH
XMERR. . . . . C ADDR  0140H
XMITADDR . . . . . C ADDR  0120H
XMITBYTE . . . . . C ADDR  0125H
XMTDAT . . . . . D ADDR  0029H
```

AN423

Software driven serial communication routines for the 83C751 and 83C752 mirocontrollers

DESCRIPTION

The need often arises to make use of a serial port in connection with a microcontroller that does not have a hardware UART on-chip. Aside from the obvious cases where the microcontroller application intrinsically requires RS-232 communications to achieve its purpose, a serial output may often be a simple and convenient method of providing detailed diagnostic information to the outside world while using only a single I/O port pin. In many cases, the solution may be to implement the UART function in software. The routines included here demonstrate a method to add such a function to a microcontroller without the benefit of a hardware UART.

Examples of microcontrollers that do not have on-chip UARTs are the 83C751 and 83C752. While it is possible to connect an external UART chip to these microcontrollers, it tends to use up many I/O port pins and begins to become less economical than simply using a standard 80C51. The are several factors to be considered in deciding if the software UART method will be usable in a particular application. The first is whether the serial communication channel is to be simplex (transmit only or receive only), half-duplex (transmit and receive, but not simultaneously), or full-duplex (simultaneous transmit and receive). Both simplex and half-duplex operation are fairly easy to implement in software on an 80C51-type microcontroller, and will be covered by this application note. Full-duplex operation is more difficult to implement in software and can use up a large portion of the microcontroller's time and resources.

A second consideration to be taken into account is the amount of system resources that will be "'used up" by the serial communication software. First of all, such software routines will almost always require the use of at least one counter/timer to generate the time slices for the serial bit cells. Next, the physical connection to the outside world will require one I/O port pin each for the serial input and the serial output. Moreover, the port pin used for serial input should be an external interrupt input pin. This allows the software to be interrupted automatically at the beginning of an incoming start bit and synchronizes the timer accurately to the serial data stream. Additional port pins may be used to implement signals such as Request to Send (RTS), Clear to Send (CTS), etc.

Finally, serial communication software will take up a certain amount of CPU time, more than would be required to operate a hardware UART. The overhead of software implemented serial communication may or may not be an issue, depending on the application, the throughput of the serial channel(s), the baud rate, other tasks the CPU is handling and how time-critical they are, etc.

The program listing that is included here is a demonstration of half-duplex serial routines on the 83C751 or 83C752 microcontrollers. The operation of the software would be the same on other 80C51 derivatives, except that the counter/timer operation is slightly different. The program, as listed, will send a canned message to the serial output (port pin P1.0 in this case), then wait for data on the serial input (port pin P1.5/INT0). When a character has been received on the serial input, it will be echoed through the serial output. Since the software is inherently half-duplex, the rate at which characters are received must be less than half the rate that would be possible on a full-duplex channel. This example has been set up to receive and transmit at 9600 baud when run with a 16 MHz crystal.

The operation of the routines is fairly straightforward. Beginning with a start bit occurring on the serial input line, an interrupt (external interrupt 0) will occur. At the interrupt service routine Int0, the counter/timer is loaded with a value that will result in a time delay that is approximately equivalent to half a bit cell time for the baud rate being used, less some constant to account for the elapsed time between a timer interrupt and the point where the serial input is actually sampled. The timer reload register is loaded with a value that will result in a time delay that is as close as can be calculated to one full bit cell time. The program then starts the timer and simply returns to the main program, waiting for the timer to time out, generating another interrupt.

At that point, the serial start bit should be about halfway through its nominal duration. When the first timer interrupt occurs, the timer interrupt routine Timr0 calls the receive bit routine RxBit which checks to make sure that the start bit is still valid and flags an error if it is not. The RxBit routine will then return control to the main program routine, waiting for the next timer interrupt.

Software driven serial communication routines AN423

On the second timer interrupt, the RxBit routine reads the serial input line and shifts the value into the serial holding register RxDat. This process is repeated until 8 bits have been read in on consecutive timer interrupts. Finally, on the tenth timer interrupt, the receive routine looks for a valid stop bit and flags an error if one is not detected. At this point, the RcvRdy flag is set to inform the main program that a character is waiting in the holding register.

The transmit routine works in a somewhat similar fashion, beginning with a call to the byte transmit routine XmtByte, which first checks to make sure that a byte receive operation is not already in progress. The RSXmt routine will then set up the timer and timer reload registers to correspond to one bit cell time, start the timer, and assert a start bit.

At each subsequent timer interrupt, the routine TxBit shifts out the next bit from the transmit holding register XmtDat, until all 8 bits have been transmitted. Once all of the data has been sent, the stop bit is asserted on the next timer interrupt. A final timer interrupt is required to insure that the stop bit lasts at least one full bit cell time. At this point, transmit flag TxFlag is cleared in order to inform the main program that the transmission is completed.

A few other useful routines are embedded in the sample program: PrByte, which converts a byte of data to hexadecimal form and transmits it; HexAsc, which converts one nibble of raw data to hexadecimal form; and Mess, which transmits an absolute string of data (usually a text message) which is terminated by a 0 byte.

This demonstration of software driven serial port routines uses 5 bytes of microcontroller RAM, two port bits (including one external interrupt input), one counter/timer, and about 256 bytes of code space, excluding the message string at the end of the listing.

```
RS751          Half-Duplex Serial Communication Routines                    11/14/89


                    1
                    2   ;***************************************************************************
                    3
                    4   ;       Software Driven Half-Duplex Serial Communication Routines
                    5   ;             for 83C751 and 83C752 series Microcontrollers
                    6
                    7   ;
                    8
                    9   ;***************************************************************************
                   10
                   11   $Title(Half-Duplex Serial Communication Routines)
                   12   $Date(11/14/89)
                   13   $MOD751
                   14
                   15   ;***************************************************************************
                   16
FF75               17   BaudVal  EQU     -139            ;Timer value for 9600 baud @ 16 MHz.
                   18                                    ;(one bit cell time)
FFD9               19   StrtVal  EQU     -39             ;Timer value to start receive.
                   20                                    ;(half of one bit cell time, minus the
                   21                                    ;time it takes the code to sample RxD)
                   22
0010               23   XmtDat   DATA    10h             ;Data for RS-232 transmit routine.
0011               24   RcvDat   DATA    11h             ;Data from RS-232 receive routine.
0012               25   BitCnt   DATA    12h             ;RS-232 transmit & receive bit count.
0013               26   LoopCnt  DATA    13h             ;Loop counter for test routine.
                   27
0020               28   Flags    DATA    20h
0000               29   TxFlag   BIT     Flags.0         ;Receive-in-progress flag.
0001               30   RxFlag   BIT     Flags.1         ;Transmit-in-progress flag.
0002               31   RxErr    BIT     Flags.2         ;Receiver framing error.
0003               32   RcvRdy   BIT     Flags.3         ;Receiver ready flag.
                   33
0090               34   TxD      BIT     P1.0            ;Port bit for RS-232 transmit.
0095               35   RxD      BIT     P1.5            ;Port bit for RS-232 receive (INT0).
                   36
                   37   ;***************************************************************************
                   38
                   39   ; Interrupt Vectors
                   40
0000               41            ORG     0               ;Reset vector.
0000 0124          42            AJMP    Reset
                   43
0003               44            ORG     03H             ;External interrupt 0.
0003 019F          45            AJMP    ExInt0          ;Indicates RS-232 start bit received.
                   46
000B               47            ORG     0BH             ;Timer 0 interrupt.
000B 0175          48            AJMP    Timr0           ;Baud rate generator.
                   49
0013               50            ORG     13H             ;External interrupt 1 (not used).
0013 32            51            RETI
                   52
001B               53            ORG     1BH             ;Timer I interrupt (not used).
001B 32            54            RETI
                   55
0023               56            ORG     23H             ;I2C interrupt (not used).
0023 32            57            RETI
                   58
```

```
                        59  ;***************************************************************************
                        60
                        61  ;Simple test of RS-232 transmit and receive.
                        62
0024 758130             63  Reset:  MOV     SP,#30h
0027 752000             64          MOV     Flags,#0            ;Clear RS-232 flags.
002A C201               65          CLR     RxFlag
002C 758800             66          MOV     TCON,#00h           ;Set up timer controls.
002F 75A882             67          MOV     IE,#82h             ;Enable timer 0 interrupts.
                        68
0032 751310             69          MOV     LoopCnt,#16         ;Test transmit first.
0035 7900               70          MOV     R1,#0               ;Zero line count.
0037 90010C             71          MOV     DPTR,#Msg1          ;Point to message string.
003A 11FB               72  Loop1:  ACALL   Mess                ;Send an RS-232 message repeatedly.
003C 743A               73          MOV     A,#':'
003E 1154               74          ACALL   XmtByte
0040 E9                 75          MOV     A,R1
0041 11DD               76          ACALL   PrByte              ;Print R1 contents.
0043 09                 77          INC     R1                  ;Advance R1 value.
0044 D513F3             78          DJNZ    LoopCnt,Loop1
                        79
0047 D2A8               80  Loop2:  SETB    EX0                 ;Enable interrupt 0 (RS-232 receive).
0049 3003FD             81          JNB     RcvRdy,$            ;Wait for data available.
004C C203               82          CLR     RcvRdy
004E E511               83          MOV     A,RcvDat            ;Echo same byte.
0050 1154               84          ACALL   XmtByte
0052 80F3               85          SJMP    Loop2
                        86
                        87
                        88  ; Send a byte out RS-232 and wait for completion before returning.
                        89  ; (use if there is nothing else to do while RS-232 is busy)
                        90
0054 2001FD             91  XmtByte: JB      RxFlag,$            ;Wait for receive complete.
0057 115D               92          ACALL   RSXmt               ;Send ACC to RS-232 output.
0059 2000FD             93          JB      TxFlag,$            ;Wait for transmit complete.
005C 22                 94          RET
                        95
                        96
                        97  ; Begin RS-232 transmit.
                        98
005D F510               99  RSXmt:  MOV     XmtDat,A            ;Save data to be transmitted.
005F 75120A            100          MOV     BitCnt,#10          ;Set bit count.
0062 758CFF            101          MOV     TH,#High BaudVal    ;Set timer for baud rate.
0065 758A75            102          MOV     TL,#Low BaudVal
0068 758DFF            103          MOV     RTH,#High BaudVal   ;Also set timer reload value.
006B 758B75            104          MOV     RTL,#Low BaudVal
006E D28C              105          SETB    TR                  ;Start timer.
0070 C290              106          CLR     TxD                 ;Begin start bit.
0072 D200              107          SETB    TxFlag              ;Set transmit-in-progress flag.
0074 22                108          RET
                       109
                       110
                       111  ; Timer 0 timeout: RS-232 receive bit or transmit bit.
                       112
0075 C0E0              113  Timr0:  PUSH    ACC
0077 C0D0              114          PUSH    PSW
0079 20013E            115          JB      RxFlag,RxBit        ;Is this a receive timer interrupt?
007C 200007            116          JB      TxFlag,TxBit        ;Is this a transmit timer interrupt?
```

```
007F C28C          117   T0Ex1:  CLR     TR                ;Stop timer.
0081 D0D0          118   T0Ex2:  POP     PSW
0083 D0E0          119           POP     ACC
0085 32            120           RETI
                   121
                   122
                   123   ; RS-232 transmit bit routine.
                   124
0086 D51204        125   TxBit:  DJNZ    BitCnt,TxBusy     ;Decrement bit count, test for done.
0089 C200          126           CLR     TxFlag            ;End of stop bit, release timer.
008B 80F2          127           SJMP    T0Ex1             ;Stop timer and exit.
                   128
008D E512          129   TxBusy: MOV     A,BitCnt          ;Get bit count.
008F B40104        130           CJNE    A,#1,TxNext       ;Is this a stop bit?
0092 D290          131           SETB    TxD               ;Set stop bit.
0094 80EB          132           SJMP    T0Ex2             ;Exit.
                   133
0096 E510          134   TxNext: MOV     A,XmtDat          ;Get data.
0098 13            135           RRC     A                 ;Advance to next bit.
0099 F510          136           MOV     XmtDat,A
009B 9290          137           MOV     TxD,C             ;Send data bit.
009D 80E2          138           SJMP    T0Ex2             ;Exit.
                   139
                   140
                   141   ;Begin RS-232 receive (after external interrupt 0).
                   142
009F 75120A        143   ExInt0: MOV     BitCnt,#10        ;Set receive bit count.
00A2 758CFF        144           MOV     TH,#High StrtVal  ;First timeout in HALF a bit time.
00A5 758AD9        145           MOV     TL,#Low StrtVal
00A8 758DFF        146           MOV     RTH,#High BaudVal ;Set timer reload for baud rate.
00AB 758B75        147           MOV     RTL,#Low BaudVal
00AE 751100        148           MOV     RcvDat,#0         ;Initialize received data to 0.
00B1 C2A8          149           CLR     EX0               ;Disable external interrupt 0.
00B3 C202          150           CLR     RxErr             ;Clear error flag.
00B5 D28C          151           SETB    TR                ;Start timer.
00B7 D201          152           SETB    RxFlag            ;Set receive-in-progress flag.
00B9 32            153           RETI
                   154
                   155
                   156   ; RS-232 receive bit routine.
                   157
00BA D5120D        158   RxBit:  DJNZ    BitCnt,RxBusy     ;Decrement bit count, test for stop.
00BD 209502        159           JB      RxD,RxBitEx       ;Valid stop bit?
00C0 D202          160   RxBtErr: SETB   RxErr             ;Bad stop bit, tell mainline.
00C2 C201          161   RxBitEx: CLR    RxFlag            ;Release timer for other purposes.
00C4 D2A8          162           SETB    EX0               ;Re-enable external interrupt 0.
00C6 D203          163           SETB    RcvRdy            ;Tell mainline that a byte is ready.
00C8 80B5          164           SJMP    T0Ex1             ;Stop timer and exit.
                   165
00CA E512          166   RxBusy: MOV     A,BitCnt          ;Get bit count.
00CC B40905        167           CJNE    A,#9,RxNext       ;Is this a start bit?
00CF 2095EE        168           JB      RxD,RxBtErr       ;Valid start bit?
00D2 80AD          169           SJMP    T0Ex2             ;Exit.
                   170
00D4 E511          171   RxNext: MOV     A,RcvDat          ;Get partial receive byte.
00D6 A295          172           MOV     C,RxD             ;Get receive pin value.
00D8 13            173           RRC     A                 ;Shift in new bit.
00D9 F511          174           MOV     RcvDat,A          ;Save updated receive byte.
```

```
00DB 80A4          175             SJMP      T0Ex2               ;Exit.
                   176
                   177
                   178     ; Print byte routine: print ACC contents as ASCII hexadecimal.
                   179
00DD C0E0          180     PrByte:  PUSH   ACC
00DF C4            181              SWAP   A
00E0 11EB          182              ACALL  HexAsc
00E2 1154          183              ACALL  XmtByte
00E4 D0E0          184              POP    ACC
00E6 11EB          185              ACALL  HexAsc            ;Print nibble in ACC as ASCII hex.
00E8 1154          186              ACALL  XmtByte
00EA 22            187              RET
                   188
                   189
                   190     ; Hexadecimal to ASCII conversion routine.
                   191
00EB 540F          192     HexAsc:  ANL    A,#0FH             ;Convert a nibble to ASCII hex.
00ED 30E308        193              JNB    ACC.3,NoAdj
00F0 20E203        194              JB     ACC.2,Adj
00F3 30E102        195              JNB    ACC.1,NoAdj
00F6 2407          196     Adj:     ADD    A,#07H
00F8 2430          197     NoAdj:   ADD    A,#30H
00FA 22            198              RET
                   199
                   200
                   201     ; Message string transmit routine.
                   202
00FB C0E0          203     Mess:    PUSH       ACC
00FD 7800          204              MOV        R0,#0               ;R0 is character pointer (string
00FF E8            205     Mes1:    MOV        A,R0                ; length is limited to 256 bytes).
0100 93            206              MOVC       A,@A+DPTR           ;Get byte to send.
0101 B40003        207              CJNE       A,#0,Send           ;End of string is indicated by a 0.
0104 D0E0          208              POP        ACC
0106 22            209              RET
                   210
0107 1154          211     Send:    ACALL      XmtByte             ;Send a character.
0109 08            212              INC        R0                  ;Next character.
010A 80F3          213              SJMP       Mes1
                   214
010C 0D0A          215     Msg1:    DB         0Dh, 0Ah
010E 54686973      216              DB         'This is a test of the software serial routines.', 0
0112 20697320
0116 61207465
011A 7374206F
011E 66207468
0122 6520736F
0126 66747761
012A 72652073
012E 65726961
0132 6C20726F
0136 7574696E
013A 65732E00
                   217
                   218              END

ASSEMBLY COMPLETE, 0 ERRORS FOUND
```

```
ACC. . . . . D ADDR  00E0H  PREDEFINED
ADJ. . . . . C ADDR  00F6H
BAUDVAL. . . . . NUMB  FF75H
BITCNT . . . . . D ADDR  0012H
EX0. . . . . B ADDR  00A8H  PREDEFINED
EXINT0 . . . . . C ADDR  009FH
FLAGS. . . . . D ADDR  0020H
HEXASC . . . . . C ADDR  00EBH
IE . . . . . D ADDR  00A8H  PREDEFINED
LOOP1. . . . . C ADDR  003AH
LOOP2. . . . . C ADDR  0047H
LOOPCNT. . . . . D ADDR  0013H
MESL . . . . . C ADDR  00FFH
MESS . . . . . C ADDR  00FBH
MSG1 . . . . . C ADDR  010CH
NOADJ. . . . . C ADDR  00F8H
P1 . . . . . D ADDR  0090H  PREDEFINED
PRBYTE . . . . . C ADDR  00DDH
PSW. . . . . D ADDR  00D0H  PREDEFINED
RCVDAT . . . . . D ADDR  0011H
RCVRDY . . . . . B ADDR  0003H
RESET. . . . . C ADDR  0024H
RSXMT. . . . . C ADDR  005DH
RTH. . . . . D ADDR  008DH  PREDEFINED
RTL. . . . . D ADDR  008BH  PREDEFINED
RXBIT. . . . . C ADDR  00BAH
RXBITEX. . . . . C ADDR  00C2H
RXBTERR. . . . . C ADDR  00C0H
RXBUSY . . . . . C ADDR  00CAH
RXD. . . . . B ADDR  0095H
RXERR. . . . . B ADDR  0002H
RXFLAG . . . . . B ADDR  0001H
RXNEXT . . . . . C ADDR  00D4H
SEND . . . . . C ADDR  0107H
SP . . . . . D ADDR  0081H  PREDEFINED
STRTVAL. . . . . NUMB  FFD9H
T0EX1. . . . . C ADDR  007FH
T0EX2. . . . . C ADDR  0081H
TCON . . . . . D ADDR  0088H  PREDEFINED
TH . . . . . D ADDR  008CH  PREDEFINED
TIMR0. . . . . C ADDR  0075H
TL . . . . . D ADDR  008AH  PREDEFINED
TR . . . . . B ADDR  008CH  PREDEFINED
TXBIT. . . . . C ADDR  0086H
TXBUSY . . . . . C ADDR  008DH
TXD. . . . . B ADDR  0090H
TXFLAG . . . . . B ADDR  0000H
TXNEXT . . . . . C ADDR  0096H
XMTBYTE. . . . . C ADDR  0054H
XMTDAT . . . . . D ADDR  0010H
```

AN424
8051 family warm boot determinations

DESCRIPTION

For some classes of applications, it may be desirable to know if the application of the reset signal to a microcontroller is due to an initial power-on sequence, or is the result of an external signal such as an operator pressing a reset pushbutton, or the result of a watchdog timer or similar event.

While there are perhaps numerous hardware solutions that can be employed, a simple software solution can offer a high degree of confidence in making this determination. The task is to determine the differences in state of resources internal to the microcontroller that would occur as a result of these two types of reset conditions. With respect to the 80C51 family of microcontrollers, on-chip resources consist of the special function registers (SFRs) and the internal data memory (RAM). Most of the SFR locations are initialized as a result of a reset condition and thus cannot be used for this determination. The data memory contents are unaffected by reset. Thus, valid data loaded into the RAM of the 80C51 while executing a program would not be affected by the application of an external reset signal provided the power source for the microcontroler has not been removed (as is the case for a "warm boot").

The contents of data memory as a result of an initial application of power, however, is indeterminate. While this effect has not been extensively characterized, empirical observation suggests that it is highly random in nature. If it is assumed, for the moment, that the behavior of a given byte of data memory is such that it will power-up with a value that is totally random, then there is a one in eight chance that it will power-up with a predetermined value. If the assumption is extended to two bytes, a 16-bit number, then there is one in 2^{16} chance that both bytes will power-up with predetermined values. Extending this to four bytes results in a one in 2^{32} chance; a very small probability. This is the basis for the software determination of a warm or cold boot condition.

The technique consists of evaluating the contents of four consecutive bytes of data memory following a reset condition to determine whether these bytes had been previously loaded with known data values. If the contents of all four bytes match predetermined values, this is interpreted to be a warm boot condition. If there is no match, it is then interpreted to be a cold boot condition. At this point, it is necessary to load these four bytes with predetermined data to prepare for the possibility of a subsequent warm boot condition.

The software example included in this application brief can be used to perform this warm or cold boot determination.

The symbols WARM1 thorough WARM4 represent the predetermined values. The symbol WARM is the address of the first of the four consecutive bytes in data memory. It is set to 30H to avoid conflict with the four register banks, the stack, and the bit-addressable locations in data memory. The symbol WARMBT is a bit-addressable location used as a status bit. It is set as the result of a warm boot and cleared as a result of a cold boot.

The label START is the location of the first instruction to be executed following a reset (address = 0000H). An instruction is located here to jump into the main body of the program to bypass the interrupt vector locations.

The main program body begins by loading register R0 with the address of the first byte in data memory to be evaluated. The contents of this first byte is compared with the first predetermined value. If there is no match, the conclusion is that it is a cold boot. However, if a match is found, this does not imply that it is a warm boot since all four bytes must match, and therefore the remaining three bytes must also be evaluated. Register R0 is incremented to point to the second byte and then compared to the second predetermined value. Comparison of the bytes proceeds until either a no match condition is found or until all four bytes have been evaluated successfully. If all four bytes compared favorable, then a status bit (WARMBT) is set to indicate a warm boot and the remainder of the application program is completed.

An unsuccessful comparison results in branching to the label COLD. This section of code clears the status bit (WARMBT) to indicate a cold boot, and loads the four bytes of data memory with the predetermined values preparing the system for a subsequent possible warm boot. Program flow then continues with the remainder of the application program.

```
                        1
                        2    ;warm boot application example
                        3
  0030                  4    WARM    EQU     30H             ;first location of the four bytes in RAM
  0055                  5    WARM1   EQU     55H             ;first predetermined value
  00AA                  6    WARM2   EQU     0AAH            ;second predetermined value
  0033                  7    WARM3   EQU     33H             ;third predetermined value
  00CC                  8    WARM4   EQU     0CCH            ;fourth predetermined value
  0000                  9    WARMBT  EQU     0               ;warm boot status bit
                       10
0000                   11         ORG     0
                       12
0000 020026            13    START: JMP     MAIN            ;bypass interrupt vectors
                       14
0026                   15         ORG     26H
                       16
0026 7830              17    MAIN:  MOV     R0,#WARM        ;pointer for first byte
0028 B65511            18           CJNE    @R0,#WARM1,COLD ;test first byte
002B 08                19           INC     R0              ;pointer for second byte
002C B6AA0D            20           CJNE    @R0,#WARM2,COLD ;test second byte
002F 08                21           INC     R0              ;pointer for third byte
0030 B63309            22           CJNE    @R0,#WARM3,COLD ;test third byte
0033 08                23           INC     R0              ;pointer for fourth byte
0034 B6CC05            24           CJNE    @R0,#WARM4,COLD ;test fourth byte
0037 D200              25           SETB    WARMBT          ;this is a warm start
0039 02004B            26           JMP     INIT            ;continue with rest of application
003C C200              27    COLD:  CLR     WARMBT          ;this is a cold boot
003E 7830              28           MOV     R0,#WARM        ;pointer for first byte
0040 7655              29           MOV     @R0,#WARM1      ;load the four bytes for future test
0042 08                30           INC     R0              ;
0043 76AA              31           MOV     @R0,#WARM2      ;
0045 08                32           INC     R0              ;
0046 7633              33           MOV     @R0,#WARM3      ;
0048 08                34           INC     R0              ;
0049 76CC              35           MOV     @R0,#WARM4      ;
004B                   36    INIT:                          ;continue with the application
                       37
                       38    END

ASSEMBLY COMPLETE, 0 ERRORS FOUND

COLD . . . . . C ADDR  003CH
INIT . . . . . C ADDR  004BH
MAIN . . . . . C ADDR  0026H
START. . . . . C ADDR  0000H  NOT USED
WARM . . . . . NUMB  0030H
WARM1. . . . . NUMB  0055H
WARM2. . . . . NUMB  00AAH
WARM3. . . . . NUMB  0033H
WARM4. . . . . NUMB  00CCH
WARMBT . . . . . NUMB  0000H
```

AN425

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

DESCRIPTION

This application note shows how to use the PCD8584 I²C-bus controller with 80C51 family microcontrollers. One typical way of connecting the PCD8584 to an 80C31 is shown. Some basic software routines are described showing how to transmit and receive bytes in a single master system. An example is given of how to use these routines in an application that makes use of the I²C circuits on an I²C demonstration board.

The PCD8584 is used to interface between parallel microprocessor or microcontroller buses and the serial I²C bus. For a description of the I²C bus protocol refer to the I²C bus specification which is printed in the microcontroller user guide.

The PCD8584 controls the transmission and reception of data on the I²C bus, arbitration, clock speeds and transmission and reception of data on the parallel bus. The parallel bus is compatible with 80C51, 68000, 8085 and Z80 buses. Communication with the I²C-bus can be done on an interrupt or polled basis. This application note focuses on interfacing with 8051 microcontrollers in single master systems.

PCD8584

In Figure 1, a block diagram is shown of the PCD8584. Basically it consists of an I²C-interface similar to the one used in 84Cxx family microcontrollers, and a control block for interfacing to the microcontroller.

The control block can automatically determine whether the control signals are from 80xx or 68xxx type of microcontrollers.

This is determined after the first write action from the microcontroller to the PCD-8584. The control block also contains a programmable divider which allows the selection of different PCD8584 and I²C clocks.

The I²C interface contains several registers which can be written and read by the microcontroller.

S1 is the control/status register. This register is accessed while the A0 input is 1. The meaning of the bits depends on whether the register is written to or read from. When used as a single master system the following bits are important:

PIN: Interrupt bit. This bit is made active when a byte is sent/received to/from the I²C-bus. When ENI is made active, PIN also controls the external INT line to interrupt the microcontroller.

ES0–ES2: These bits are used as pointer for addressing S0, S0', S2 and S3. Setting ES0 also enables the Serial I/O.

ENI: Enable Interrupt bit. Setting this bit enables the generation of interrupts on the INT line.

STA, STO: These bits allow the generation of START or STOP conditions.

ACK: With this bit set and the PCD8584 is in master/receiver mode, no acknowledge is generated by the PCD8584. The slave/transmitter now knows that no more data must be sent to the I²C-bus.

BER: This bit may be read to check if bus errors have occurred.
BB: This bit may be read to check whether the bus is free for I²C-bus transmission.

S2 is the clock register. It is addressed when A0 = 0 and ES0–ES2 = 010 in the previous write cycle to S1. With the bits S24–S20 it is possible to select 5 input clock frequencies and 4 I²C clock frequencies.

S3 is the interrupt vector register. It is addressed when A0 = 0 and ES0–ES2 = 001 in the previous write cycle to S1. This register is not used when an 80C51 family microcontroller is used. An 80C51 micro- controller has fixed interrupt vector addresses.

S0' is the own address register. It is addressed when A0 = 0 and ES0–ES2 = 000. This register contains the slave address of the PCD8584. In the single master system described here, this register has no functional use. However, by writing a value to S0', the PCD8584 determines whether an 80Cxx or 68xxx type microcontroller is the controlling microcontroller by looking at the $\overline{CS}$ and $\overline{WR}$ lines. So independent of whether the PCD8584 is used as master or slave, the microcontroller should always first write a value to S0' after reset.

S0 is the I²C data register. It is addressed when A0 = 0 and ES0–ES2 = 1x0. Transmission of a byte on the I²C bus is done by writing this byte to S0. When the transmission is finished, the PIN bit in S1 is reset and if ENI is set, an interrupt will be generated. Reception of a byte is signaled by resetting PIN and by generating an interrupt if ENI is set. The received byte can be read from S0.

Interfacing PCD8584 I²C-bus controller AN425

The SDA and SCL lines have no protection diodes to V_{DD}. This is important for multi-master systems. A system with a PCD8584 can now be switched off without causing the I²C-bus to hang-up. Other masters still can use the bus.

For more information of the PCD8584 refer to the data sheet.

PCD8584/8031 Hardware Interface

Figure 2 shows a minimum system with an 8051 family controller and a PCD8584. In this example, an 80C31 is used. However any 80C51 family controller with external addressing capability can be used.

The software resides in EPROM U3. For addressing this device, latch U2 is necessary to demultiplex the lower address bits from the data bits. The PCD8584 is mapped in the external data memory area. It is selected when A1 = 0. Because in this example no external RAM or other mapped peripherals are used, no extra address decoding components are necessary. A0 is used by the PCD8584 for proper register selection in the PCD8584.

U5A is an inverter with Schmitt trigger input and is used to buffer the oscillator signal of the microcontroller. Without buffering, the rise and fall time specifications of the CLK signal are not met. It is also important that the CLK signal has a duty cycle of 50%. If this is not possible with certain resonators or microcontrollers, then an extra flip-flop may me necessary to obtain the correct duty cycle.

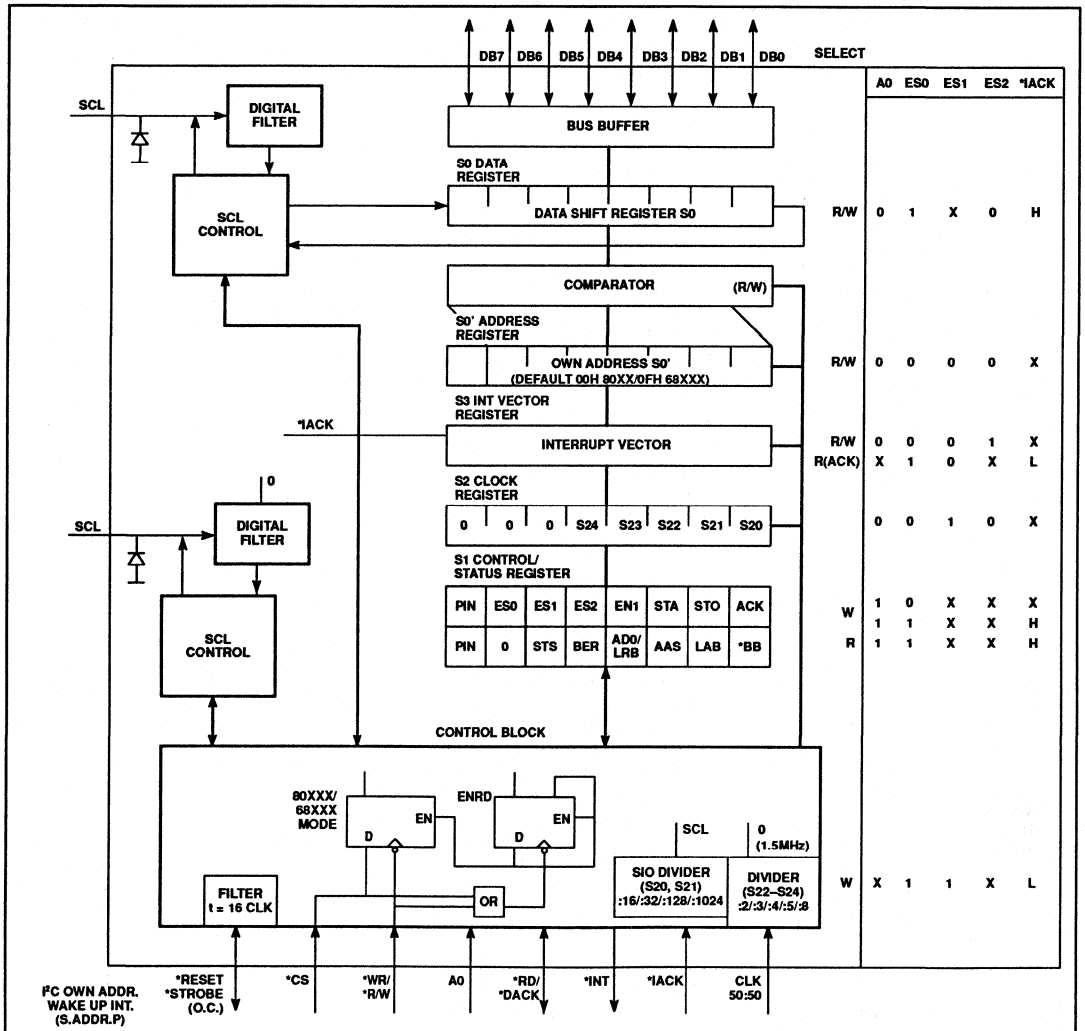


Figure 1. PCD8584 Block Diagram

Interfacing PCD8584 I²C-bus controller AN425

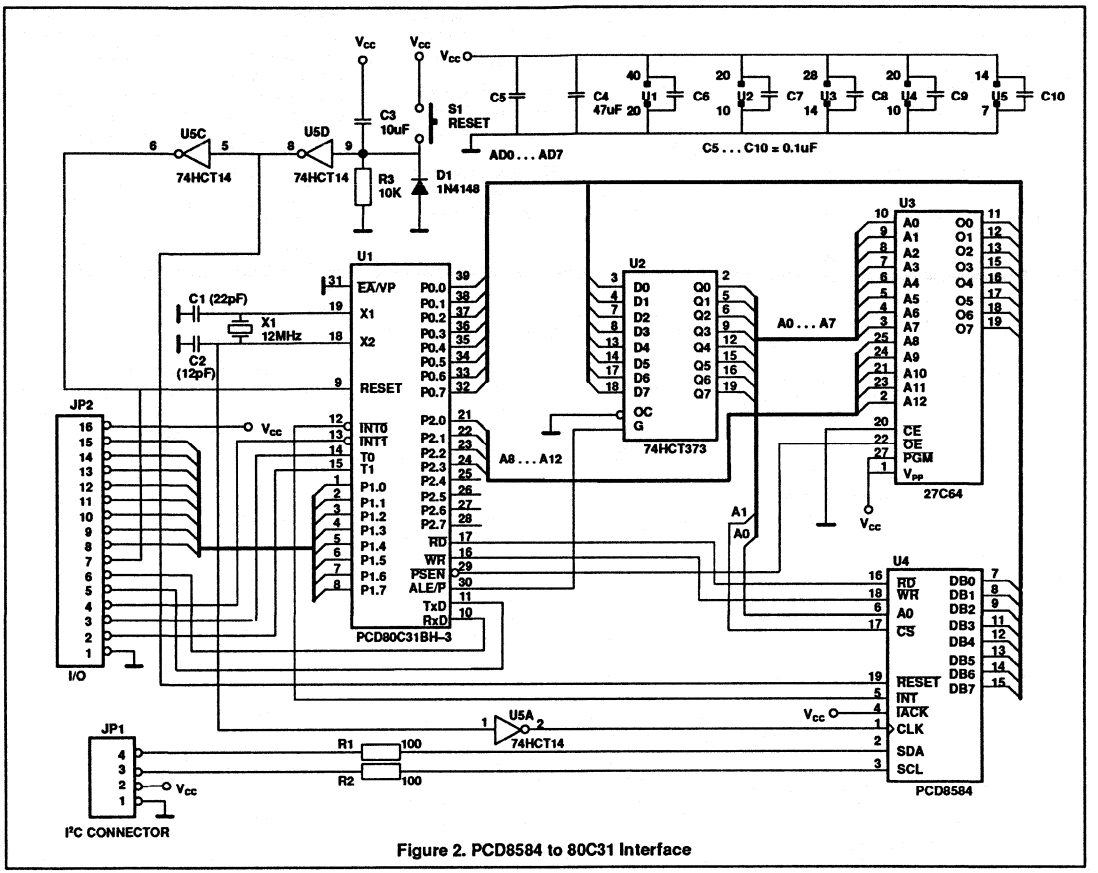


Figure 2. PCD8584 to 80C31 Interface

U5C and U5D are used to generate the proper reset signals for the microcontroller and the PCD8584.

Basic PCD8584/8031 Driver Routines

In the listing section (page 4), some basic routines are shown. The routines are divided in two modules. The module ROUTINE contains the driver routines and initialization of the PCD- 8584. The module INTERR contains the interrupt handler. These modules may be linked to a module with the user program that uses the routines in INTERR and ROUTINE. In this application note, this module will be called USER. A description of ROUTINE and INTERR follows.

Module ROUTINE

Routine Sendbyte (Lines 17–20)
This routine sends the contents of the accumulator to the PCD8584. The address is such that A0 = 0. Which register is accessed depends on the contents of ES0–ES2 of the control register. The address of the PCD8584 is in variable 'PCD8584'. This must have been previously defined in the user program. The DPTR is used as a pointer for addressing the peripheral. If the address is less than 255, then R0 or R1 may be used as the address pointer.

Routine Sendcontr (Lines 25, 26)
This routine is similar to Sendbyte, except that now A0 = 1. This means that the contents of the accumulator are sent to the control register S1 in the PCD8584.

Routine Readbyte (Lines 30–33)
This routine reads a register in the PCD8584 with A0 = 0. Which register depends on ES0 –ES2 of the control register. The result of the read operation is returned in the accumulator.

Routine Readcontr (Lines 37–39)
This routine is similar to Readbyte, except that now A0 = 1. This means that the accumulator will contain the value of status register S1 of the PCD8584.

Routine Start Lines (44–56)
This routine generates a START–condition and the slave address with a R/W bit. In line 44, the variable IIC_CNT is reset. This variable is used as a byte counter to keep track of the number of bytes that are received or transmitted. IIC_CNT is defined in module INTERR.

Lines 45–46 increment the variable NR_BYTES if the PCD8584 must receive data. NR_BYTES is a variable that indicates how many bytes have to be received or transmitted. It must be given the correct value in the USER module. Receiving or transmitting is distinguished by the value of the DIR bit.

This must also be given the correct value in the USER module.

Then the status register of PCD8584 must be read to check if the I²C bus is free. First the status register must be addressed by giving ESO– ES2 of the control register the correct value (lines 47–48). Then the Bus Busy bit is tested until the bus is free (lines 49–50). If this is the case, the slave address is sent to data register S0 and the I2C_END bit is cleared (lines 51–53). The slave address is set by the user program in variable USER. The LSB of the slave address is the R/W bit. I2C_END can be tested by the user program whether an I2C reception/transmission is in progress or not.

Next the START condition will be generated and interrupt generation enabled by setting the appropriate bits in control register S1. (lines 54–55).

Now the routine will return back to the user program and other tasks may be performed. When the START condition, slave address and R/W bit are sent, and the ACK is received, the PCD8584 will generate an interrupt. The interrupt routine will determine if more bytes have to be received or transmitted.

Routine Stop (Lines 59–62)
Calling this routine, a STOP condition will be sent to the I²C bus. This is done by sending the correct value to control register S1 (lines 59– 61). After this the I2C_END bit is set, to indicate to the user program that a complete I²C sequence has been received or transmitted.

Routine I2C_Init (Lines 65–76)
This routine initializes the PCD8584. This must be done directly after reset. Lines 67–70 write data to 'own address' register S0'. First the correct address of S0' is set in control register S1 (lines 67–68), then the correct value is written to it (lines 69–70). The value for S0' is in variable SLAVE_ADR and set by the user program. As noted previously, register S0' must always be the first register to be accessed after reset, because the PCD8584 now determines whether an 80Cxxx or 68xxx microcontroller is connected. Lines 72–76 set the clock register S2. The variable I2C_CLOCK is also set by the user program.

Module INTERR
This module contains the I²C interrupt routine. This routine is called every time a byte is received or transmitted on the I²C bus. In lines 12–15 RAM space for variables is reserved.

BASE is the start address in the internal 80C51 RAM where the data is stored that is received, or where the data is stored that has to be transmitted.

NR_BYTES, IIC_CNT and SLAVE were explained earlier. I2C_END and DIR are flags that are used in the program. I2C_END indicates whether an I²C transmission or reception is in progress. DIR indicates whether the PCD8584 has to receive or transmit bytes. The interrupt routine makes use of register bank 1.

The transmission part of the routine starts at line 42. In lines 42–43, a check is made whether IIC_CNT = NR_BYTES. If true, all bytes are sent and a STOP condition may be generated (lines 44–45).

Next the pointer for the internal RAM is restored (line 46) and the byte to be transmitted is fetched from the internal RAM (line 47). Then this byte is sent to the PCD8584 and the variables are updated (lines 47–49). The interrupt routine is left and the user program may proceed. The receive part starts from line 55. First a check is made if the next byte to be received is the last byte (lines 56–59). If true the ACK must be disabled when the last byte is received. This is accomplished by resetting the ACK bit in the control register S1 (lines 60–61).

Next the received byte may be read (line 62) from data register S0. The byte will be temporary stored in R4 (line 63). Then a check is made if this interrupt was the first after a START condition. If so, the byte read has no meaning and the interrupt routine will be left (lines 68–70). However by reading the data register S0 the next read cycle is started.

If valid data is received, it will be stored in the internal RAM addressed by the value of BASE (lines 71–73). Finally a check is made if all bytes are received. If true, a STOP condition will be sent (lines 75–78).

EXAMPLES
In the listing section (starting on page 8), some examples are shown that make use of the routines described before. The examples are transmission of a sequence, reception of I²C data and an example that combines both.

The first example sends bytes to the PCD8577 LCD driver on the OM1016 demonstration board. Lines 7 to 10 define the interface with the other modules and should be included in every user program. Lines 14 to 16 define the segments in the user module. It is completely up to the user how to organize this.

Lines 24 and 28 are the reset and interrupt vectors. The actual user program starts at line 33. Here three variables are defined that are used in the I²C driver routines. Note that PCD8584 must be an even address, otherwise the wrong internal registers will be accessed! Lines 37–42 initialize the interrupt logic of the microcontroller. Next the PCD8584 will be initialized (line 45).

The PCD8584 is now ready to transmit data. A table is made in the routine at line 61. For the PCD8577, the data is a control byte and the segment data. Note that the table does not contain the slave address of the LCD driver. In lines 51–54, variables are made ready to start the transmission. This consists of defining the direction of the transmission (DIR), the address where the data table starts (BASE), the number of bytes to transmit (NR_BYTES, without slave address!) and the slave address (SLAVE) of the I²C peripheral that has to be accessed.

In line 55 the transmission is started. Once the I²C transmission is started, the user program can do other tasks because the transmission works on interrupts. In this example a loop is performed (line 58). The user can check the end of the transmission during the other tasks, by testing the I2C_END bit regularly.

The second example program receives 2 bytes from the PCF8574P I/O expander on the OM1016 demonstration board. Until line 45 the program is identical to the transmit routine because it consists of initialization and variable definition. From line 48, the variables are set for I²C reception. The received bytes are stored in RAM area from label TABLE. During reception, the user program can do other tasks. By testing the I2C_END bit the user can determine when to start processing the data in the TABLE.

The third example program displays time from the PCF8583P clock/calendar/RAM on the LCD display driven by the PCF8577. The LED display (driven by SAA1064) shows the value of the analog inputs of the A/D converter PCF8591. The four analog inputs are scanned consecutively.

In this example, both transmit and receive sequences are implemented as shown in the previous examples. The main clock part is from lines 62–128. This contains the calls to the I²C routines. From lines 135–160, routines are shown that prepare the data to be transmitted. Lines 171 to 232 are the main program for the AD converter and LED display. Lines 239 to 340 contain routines used by the main program. This demo program can also be used with the I²C peripherals on the OM1016 demonstration board.

```
ASM51  TSW  ASSEMBLER   Routines for PCD8584

LOC   OBJ          LINE  SOURCE

                      1  $TITLE (Routines for PCD8584)
                      2  $PAGELENGTH(40)
                      3  ;Program written for PCD8584 as master
                      4  ;
                      5          PUBLIC READBYTE,READCONTR,SENDBYTE
                                 PUBLIC SENDCONTR,START,STOP
                      6          PUBLIC I2C_INIT
                      7          EXTRN  BIT(I2C_END,DIR)
                      8          EXTRN  DATA(SLAVE,IIC_CNT,NR_BYTES)
                      9          EXTRN  NUMBER(SLAVE_ADR,I2C_CLOCK,PCD8584)
                     10  ;
                     11  ;Define code segment
                     12  ROUTINE    SEGMENT CODE
----                 13             RSEG    ROUTINE
                     14  ;
                     15  ;SENDBYTE sends a byte to PCD8584 with A0=0
                     16  ;Byte to be send must be in accu
0000:            R   17  SENDBYTE:
0000: 900000     R   18          MOV DPTR,#PCD8584 ;Register address
0003: F0             19  SEND:    MOVX @DPTR,A      ;Send byte
0004: 22             20          RET
                     21  ;
                     22  ;SENDCONTR sends a byte to PCD8584 with A0=1
                     23  ;Byte to be send must be in accu
0005:                24  SENDCONTR:
0005: 900001     R   25          MOV DPTR,#PCD8584+01H ;Register address
0008: 80F9           26          JMP SEND
                     27  ;
                     28  ;READBYTE reads a byte from PCD8584 with A0=0
                     29  ;Received byte is stored in accu
000A:                30  READBYTE:
000A: 900000     R   31          MOV DPTR,#PCD8584 ;Register address
000D: E0             32  REC:     MOVX A,@DPTR      ;Receive byte
000E: 22             33          RET
                     34  ;
                     35  ;READCONTR reads a byte from PCD8584 with A0=1
                     36  ;Received byte is stored in accu
000F:                37  READCONTR:
000F: 900001     R   38          MOV DPTR,#PCD8584+01H ;Register address
0012: 80F9           39          JMP REC
                     40  ;
                     41  ;START tests if the I2C bus is ready. If ready a
                     42  ;START-condition will be sent, interrupt generation
                     43  ;and acknowledge will be enabled.
0014: 750000     R   44  START:   MOV IIC_CNT,#00 ;Clear  I2C byte counter
0017: 200002     R   45          JB DIR,PROCEED  ;If DIR is 'receive' then
001A: 0500       R   46          INC NR_BYTES    ;increment NR_BYTES
001C: 7440           47  PROCEED:MOV A,#40H       ; Read STATUS register of
                                                  ; 8584
001E: 120005     R   48          CALL SENDCONTR
0021: 12000F     R   49  TESTBB: CALL READCONTR
0024: 30E0FA         50          JNB ACC.0,TESTBB; Test BB/ bit
0027: E500       R   51          MOV A,SLAVE
0029: C200       R   52          CLR I2C_END     ;Reset I2C ready bit
002B: 120000     R   53          CALL SENDBYTE   ;Send slave address
002E: 744D           54          MOV A,#01001101B;Generate START, set ENI,
                                                  ;set ACK
```

```
0030: 120005   R     55          CALL SENDCONTR
0033: 22             56          RET
                     57  ;
                     58  ;STOP will generate a STOP condition and set the
                         ;I2C_END bit
0034: 74C3           59  STOP:   MOV A,#11000011B
0036: 120005   R     60          CALL SENDCONTR  ;Send STOP condition
0039: D200     R     61          SETB I2C_END    ;Set I2C_END bit
003B: 22             62          RET
                     63  ;
                     64  ;I2C_init does the initialisation of the PCD8584
003C:                65  I2C_INIT:
                     66  ;Write own slave address
003C: E4             67          CLR A
003D: 120005   R     68          CALL SENDCONTR  ;Write to control register
0040: 7400     R     69          MOV A,#SLAVE_ADR
0042: 120000   R     70          CALL SENDBYTE    ;Write to own slave
                                                  ;register
                     71  ;Write clock register
0045: 7420           72          MOV A,#20H
0047: 120005   R     73          CALL SENDCONTR  ;Write to control register
004A: 7400     R     74          MOV A,#I2C_CLOCK
004C: 120000   R     75          CALL SENDBYTE   ;Write to clock register
004F: 22             76          RET
                     77  ;
0050:                78          END
```

```
ASM51  TSW  ASSEMBLER   I2C INTERRUPT ROUTINE

LOC   OBJ          LINE  SOURCE

                      1  $TITLE (I2C INTERRUPT ROUTINE)
                      2  $PAGELENGTH(40)
                      3  ;
                      4          PUBLIC INT0_SRV
                      5          PUBLIC DIR,I2C_END
                      6          PUBLIC BASE,NR_BYTES,IIC_CNT,SLAVE
                      7          EXTRN  CODE(SENDBYTE,SENDCONTR,STOP)
                                 EXTRN  CODE(READBYTE,READCONTR)
                      8  ;
                      9  ;Define variables in RAM
                     10  IIC_VAR SEGMENT DATA
----                 11          RSEG IIC_VAR
0000:           R    12  BASE:   DS 1            ;Pointer to I2C table (till
                                                 ;256)
0001:                13  NR_BYTES: DS 1          ;Number of bytes to rcv/trm
0002:                14  IIC_CNT:DS 1            ;I2C byte counter
0003:                15  SLAVE:  DS 1            ;Slave address after START
                     16  ;
                     17  ;Define variable segment
                     18  BIT_VAR SEGMENT DATA BITADDRESSABLE
----                 19          RSEG BIT_VAR
0000:           R    20  STATUS: DS 1            ;Byte with flags
0000            R    21  I2C_END BIT STATUS.0    ;Defines if a I2C
                                                 ;transmission is finished
                     22                          ;'1' is finished
                     23                          ;'0' is not ready
0000            R    24  DIR     BIT STATUS.3    ;Defines direction of I2C
                                                 ;transmission
                     25                          ;'1':Transmit    '0':Receive
                     26  ;
                     27  ;Define code segment for routine
                     28  IIC_INT SEGMENT CODE PAGE
----                 29          RSEG IIC_INT
                     30  ;
                     31  ;Program uses registers in RB1
                     32          USING 1
                     33  ;
0000:           R    34  INT0_SRV:
0000: C0E0           35          PUSH ACC        ;Save acc. en psw on stack
0002: C0D0           36          PUSH PSW
0004: 75D008         37          MOV PSW,#08H    ;Select register bank 1
0007: 300016    R    38          JNB DIR,RECEIVE ;Test direction bit
                     39                          ;8584 is MST/TRM
                     40
                     41  ;Program part to transmit bytes to IIC bus
000A: E502      R    42          MOV A,IIC_CNT   ;Compare IIC_CNT and
                                                 ;NR_BYTES
000C: B50105    R    43          CJNE A,NR_BYTES,PROCEED
000F: 120000    R    44          CALL STOP       ;All bytes transmitted
0012: 8032           45          JMP EXIT
0014: A800      R    46  PROCEED:MOV R0,BASE     ;RAM pointer
0016: E6             47          MOV A,@R0       ;Source is internal RAM
0017: 0500      R    48          INC BASE        ;Update pointer of table
0019: 120000    R    49          CALL SENDBYTE   ;Send byte to IIC bus
001C: 0502      R    50          INC IIC_CNT     ;Update byte counter
001E: 8026           51          JMP EXIT
```

```
                        52  ;
                        53  ;
                        54  ;Program to receive byte from IIC bus
0020:                   55  RECEIVE:
0020: E502      R       56          MOV A,IIC_CNT   ;Test if last byte is to be
                                                    ;received
0022: 04                57          INC A
0023: 04                58          INC A
0024: B50105    R       59          CJNE A,NR_BYTES,PROC_RD
0027: 7448              60          MOV A,#01001000B;Last byte to be received.
                                                    ;Disable ACK
0029: 120000    R       61          CALL SENDCONTR  ;Write control word to
                                                    ;PCD8584
002C: 120000    R       62  PROC_RD:CALL READBYTE    ;Read I2C byte
002F: FC                63          MOV R4,A        ;Save accu
                        64  ;If RECEIVE is entered after the transmission of
                        65  ;START+address then the result of READBYTE is not
                        66  ;relevant. READBYTE is used to start the generation
                            ;of the clock pulses for the next byte to read.
                        67  ;This situation occurs when IIC_CNT is 0
0030: E4                68          CLR A           ;Test IIC_CNT
0031: B50202    R       69          CJNE A,IIC_CNT,SAVE
0034: 8006              70          JMP END_TEST    ;START is send. No relevant
                                                    ;data in data reg. of 8584
0036: A800      R       71  SAVE:   MOV R0,BASE
0038: EC                72          MOV A,R4        ;Destination is internal RAM
0039: F6                73          MOV @R0,A
003A: 0500      R       74          INC BASE
003C: 0502      R       75  END_TEST:INC IIC_CNT    ;Test if all bytes are
                                                    ;received
003E: E501      R       76          MOV A,NR_BYTES
0040: B50203    R       77          CJNE A,IIC_CNT,EXIT
0043: 120000    R       78          CALL STOP       ;All bytes received
                        79  ;
0046: D0D0              80  EXIT:   POP PSW         ;Restore PSW and accu
0048: D0E0              81          POP ACC
004A: 32                82          RETI
                        83  ;
004B:                   84          END
```

```
ASM51  TSW  ASSEMBLER   Send a string of bytes to the PCF8577 on OM1016

LOC   OBJ          LINE  SOURCE

                      1  $TITLE (Send a string of bytes to the PCF8577 on
                             OM1016)
                      2  $PAGELENGTH(40)
                      3  ;
                      4  ;This program is an example to transmit bytes via
                         ;PCD8584
                      5  ;to the I2C-bus
                      6  ;
                      7          PUBLIC  SLAVE_ADR,I2C_CLOCK,PCD8584
                      8          EXTRN   CODE(I2C_INIT,INT0_SRV,START)
                      9          EXTRN   BIT(I2C_END,DIR)
                     10          EXTRN   DATA(BASE,NR_BYTES,IIC_CNT,SLAVE)
                     11  ;
                     12  ;
                     13  ;Define used segments
                     14  USER    SEGMENT CODE    ;Segment for user program
                     15  RAMTAB  SEGMENT DATA    ;Segment for table in
                                                 ;internal RAM
                     16  RAMVAR  SEGMENT DATA    ;Segment for RAM variables
                                                 ;in RAM
                     17  ;
                     18  ;
----                 19          RSEG    RAMVAR
0000:          R     20  STACK:  DS 20           ;Reserve stack area
                     21  ;
                     22  ;
----                 23          CSEG AT 00H
0000: 020000   R     24          JMP MAIN        ;Reset vector
                     25  ;
                     26  ;
----                 27          CSEG AT 03H
0003: 020000   R     28          JMP INT0_SRV    ;I2C interrupt vector
                                                 ;(INT0/)
                     29  ;
                     30  ;
----                 31          RSEG USER
                     32  ;Define I2C clock, own slave address and PCD8584
                         ;hardware address
0055                 33  SLAVE_ADR EQU 55H       ;Own slave address is 55H
001C                 34  I2C_CLOCK EQU 00011100B ;12.00MHz/90kHz
0000                 35  PCD8584   EQU 0000H     ;PCD8584 address with A0=0
                     36  ;0000: 7581FF   R    37  MAIN:   MOV SP,#STACK-1 ;Initialise stack pointer
                     38  ;Initialise 8031 interrupt registers for I2C
                         ;interrupt
0003: D2A8           39          SETB EX0        ;Enable interrupt INT0/
0005: D2AF           40          SETB EA         ;Set global enable
0007: D2B8           41          SETB PX0        ;Priority level '1'
0009: D288           42          SETB IT0        ;INT0/ on falling edge
                     43  ;
                     44  ;Initialise PCD8584
000B: 120000   R     45          CALL I2C_INIT
                     46  ;
                     47  ;Make a table in RAM with data to be transmitted.
000E: 120021   R     48          CALL MAKE_TAB
                     49  ;
                     50  ;Set variables to control PCD8584
```

```
0011: D200     R    51          SETB DIR          ;DIR='transmission'
0013: 750000   R    52          MOV BASE,#TABLE ;Start address of I2C-data
0016: 750005   R    53          MOV NR_BYTES,#05H ;5 bytes must be
                                                  ;transferred
0019: 750074   R    54          MOV SLAVE,#01110100B ;Slave address PCF8577
                                                  ; + WR/
001C: 120000   R    55          CALL START        ;Start I2C transmission
                    56  ;
                    57  ;
001F: 80FE          58  LOOP:   JMP LOOP          ;Endless loop when program
                                                  ;is finished
                    59  ;
                    60  ;
0021:               61  MAKE_TAB:
0021: 7800     R    62          MOV R0,#TABLE     ;Make data ready for I2C
                                                  ;transmission
0023: 7600          63          MOV @R0,#00       ;Controlword PCF8577
0025: 08            64          INC R0
0026: 76FC          65          MOV @R0,#0FCH     ;'0'
0028: 08            66          INC R0
0029: 7660          67          MOV @R0,#60H      ;'1'
002B: 08            68          INC R0
002C: 76DA          69          MOV @R0,#0DAH     ;'2'
002E: 08            70          INC R0
002F: 76F2          71          MOV @R0,#0F2H     ;'3'
0031: 22            72          RET
                    73  ;
                    74  ;
----                75          RSEG RAMTAB
0000:          R    76  TABLE:  DS 10             ;Reserve space in internal
                                                  ;data RAM
                    77                            ;for I2C data to transmit
                    78  ;
                    79  ;
000A:               80          END
```

```
ASM51  TSW  ASSEMBLER   Receive 2 bytes from the PCF8574P on OM1016

LOC   OBJ          LINE  SOURCE

                      1  $TITLE (Receive 2 bytes from the PCF8574P on OM1016)
                      2  $PAGELENGTH(40)
                      3  ;
                      4  ;This program is an example to receive bytes via
                         ;PCD8584
                      5  ;from the I2C-bus
                      6  ;
                      7          PUBLIC  SLAVE_ADR,I2C_CLOCK,PCD8584
                      8          EXTRN   CODE(I2C_INIT,INT0_SRV,START)
                      9          EXTRN   BIT(I2C_END,DIR)
                     10          EXTRN   DATA(BASE,NR_BYTES,IIC_CNT,SLAVE)
                     11  ;
                     12  ;
                     13  ;Define used segments
                     14  USER    SEGMENT CODE    ;Segment for user program
                     15  RAMTAB  SEGMENT DATA    ;Segment for table in
                                                 ;internal RAM
                     16  RAMVAR  SEGMENT DATA    ;Segment for RAM variables
                                                 ;in RAM
                     17  ;
                     18  ;
----                 19          RSEG    RAMVAR
0000:          R     20  STACK:  DS 20           ;Reserve stack area
                     21  ;
                     22  ;
----                 23          CSEG AT 00H
0000: 020000   R     24          JMP MAIN        ;Reset vector
                     25  ;
                     26  ;
----                 27          CSEG AT 03H
0003: 020000   R     28          JMP INT0_SRV    ;I2C interrupt vector
                                                 ;(INT0/)
                     29  ;
                     30  ;
----                 31          RSEG USER
                     32  ;Define I2C clock, own slave address and PCD8584
                         ;hardware address
0055                 33  SLAVE_ADR EQU 55H       ;Own slave address is 55H
001C                 34  I2C_CLOCK EQU 00011100B ;12.00MHz/90kHz
0000                 35  PCD8584   EQU 0000H     ;PCD8584 address with A0=0
                     36  ;0000: 7581FF   R    37  MAIN:   MOV SP,#STACK-1 ;Initialise stack pointer
                     38  ;Initialise 8031 interrupt registers for I2C
                         ;interrupt
0003: D2A8           39          SETB EX0        ;Enable interrupt INT0/
0005: D2AF           40          SETB EA         ;Set global enable
0007: D2B8           41          SETB PX0        ;Priority level '1'
0009: D288           42          SETB IT0        ;INT0/ on falling edge
                     43  ;
                     44  ;Initialise PCD8584
000B: 120000   R     45          CALL I2C_INIT
                     46  ;
                     47  ;Set variables to control PCD8584
000E: C200     R     48          CLR  DIR        ;DIR='receive'
0010: 750000   R     49          MOV BASE,#TABLE ;Start address of I2C-data
0013: 750002   R     50          MOV NR_BYTES,#02H ;2 bytes must be received
0016: 75004F   R     51          MOV SLAVE,#01001111B ;Slave address PCF8574
                                                 ; + RD
```

```
0019: 120000    R     52            CALL START      ;Start I2C transmission
                      53  ;
                      54  ;
001C: 80FE            55  LOOP:     JMP LOOP        ;Endless loop when program
                                                    ;is finished
                      56  ;
                      57  ;
----                  58            RSEG RAMTAB
0000:           R     59  TABLE:    DS 10           ;Reserve space in internal
                                                    ;data RAM
                      60                            ;for received I2C data
                      61  ;
                      62  ;
000A:                 63            END
```

```
ASM51  TSW  ASSEMBLER   Demo program for PCD8584 I2C-routines

LOC   OBJ          LINE  SOURCE

                      1  $TITLE (Demo program for PCD8584 I2C-routines)
                      2  $PAGELENGTH(40)
                      3  ;Program displays on the LCD display  the time (with
                         ;PCF8583). Dots on LCD display blink every second.
                      5  ;On the LED display the values of the successive
                         ;analog input channels are shown.
                      7  ;Program reads analog channels of PCF8591P.
                      8  ;Channel number and channel value are displayed
                         ;successively.
                      9  ;Values are displayed on LCD and LED display on I2C
                         ;demo board.
                     10  ;
                     11          PUBLIC   SLAVE_ADR,I2C_CLOCK,PCD8584
                     12          EXTRN    CODE(I2C_INIT,INT0_SRV,START)
                     13          EXTRN    BIT(I2C_END,DIR)
                     14          EXTRN    DATA(BASE,NR_BYTES,IIC_CNT,SLAVE)
                     15  ;
                     16  ;
                     17  ;Define used segments
                     18  USER    SEGMENT  CODE   ;Segment for user program
                     19  RAMTAB  SEGMENT  DATA   ;Segment for table in
                                                 ;internal RAM
                     20  RAMVAR  SEGMENT  DATA   ;Segment for variables
                     21  ;
----                 22          RSEG RAMVAR
0000:          R     23  STACK:  DS 20           ;Stack area (20 bytes)
0014:                24  PREVIOUS: DS 1          ;Store for previous seconds
0015:                25  CHANNEL:DS 1            ;Channel number to be
                                                 ;sampled
0016:                26  AN_VAL: DS 1            ;Analog value sampled
                                                 ;channel
0017:                27  CONVAL: DS 3            ;Converted BCD value sampled
                                                 ;channel
                     28  ;
----                 29          CSEG AT 00H
0000: 020000   R     30          LJMP MAIN       ;Reset vector
                     31  ;
----                 32          CSEG AT 03H     ;INT0/
0003: 020000   R     33          LJMP INT0_SRV   ;Vector I2C-interrupt
                     34  ;
                     35  ;
----                 36          RSEG USER37  ;Define I2C clock, own slave address and address for
                         ;main processor
0055                 38  SLAVE_ADR EQU 55H       ;Own slaveaddress is 55h
001C                 39  I2C_CLOCK EQU 00011100B ;12.00MHz/90kHz
0000                 40  PCD8584   EQU 0000H     ;Address of PCD8584. This
                                                 ;must be an EVEN number!!
                     41  ;Define addresses of I2C peripherals
00A3                 42  PCF8583R  EQU 10100011B ;Address PCF8583 with Read
                                                 ;active
00A2                 43  PCF8583W  EQU 10100010B ;Address PCF8583 with Write
                                                 ;active
009F                 44  PCF8591R  EQU 10011111B ;Address PCF8591 with Read
                                                 ;active
009E                 45  PCF8591W  EQU 10011110B ;Address PCF8591 with Write
                                                 ;active
```

```
ASM51  TSW  ASSEMBLER   Demo program for PCD8584 I2C-routines

LOC   OBJ          LINE  SOURCE

0074                 46  PCF8577W  EQU 01110100B ;Address PCF8577 with Write
                                                 ;active
0076                 47  SAA1064W  EQU 01110110B ;Address SAA1064 with Write
                                                 ;active
                     48  ;
0000: 7581FF   R     49  MAIN:   MOV SP,#STACK-1 ;Define stack pointer
                     50  ;Initialise 80C31 interruptregisters for I2C
                         ;interrupt (INT0/)
0003: D2A8           51          SETB EX0       ;Enable interrupt INT0/
0005: D2AF           52          SETB EA        ;Set global enable
0007: D2B8           53          SETB PX0       ;Priority level is '1'
0009: D288           54          SETB IT0       ;INT0/ on falling edge
                     55  ;Initialise PCD8584
000B: 120000   R     56          CALL I2C_INIT
                     57  ;
000E: 751500   R     58          MOV CHANNEL,#00 ;Set AD-channel
                     59  ;
                     60  ;Time must be read from PCD8583.
                     61  ;First write word address and control register of
                         ;PCD8583.
0011: D200     R     62          SETB DIR       ;DIR='transmission'
0013: 750000   R     63          MOV BASE,#TABLE ;Start address I2C data
0016: 750002   R     64          MOV NR_BYTES,#02H ;Send 2 bytes
0019: 7500A2   R     65          MOV SLAVE,#PCF8583W
001C: E4             66          CLR A
001D: F500     R     67          MOV TABLE,A    ;Data to be sent (word
                                                 ;address).
001F: F501     R     68          MOV TABLE+1,A  ;      "         (control
                                                 ;byte)
0021: 120000   R     69          CALL START     ;Start transmission.
0024: 3000FD   R     70  FIN_1:  JNB I2C_END,FIN_1 ;Wait till transmission
                                                 ;finished
                     71  ;Send word address before reading time
0027: D200     R     72  REPEAT: SETB DIR       ;'transmission
0029: 750000   R     73          MOV BASE,#TABLE ;I2C data
002C: 7500A2   R     74          MOV SLAVE,#PCF8583W
002F: 7401           75          MOV A,#01
0031: F500     R     76          MOV NR_BYTES,A  ;Send 1 byte
0033: F500     R     77          MOV TABLE,A    ;Data to be sent is '1'
0035: 120000   R     78          CALL START     ;Start I2C transmission
0038: 3000FD   R     79  FIN_2:  JNB I2C_END,FIN_2 ;Wait till transmission
                                                 ;finished
                     80  ;
                     81  ;Time can now be read from PCD8583. Data read is
                     82  ;hundredths of sec's, sec's, min's and hr's
003B: C200     R     83          CLR DIR        ;DIR='receive'
003D: 750000   R     84          MOV BASE,#TABLE ;I2C table
0040: 750004   R     85          MOV NR_BYTES,#04; 4 bytes to receive
0043: 7500A3   R     86          MOV SLAVE,#PCF8583R
0046: 120000   R     87          CALL START     ;Start I2C reception
0049: 3000FD   R     88  FIN_3:  JNB I2C_END,FIN_3 ;Wait till finished
                     89  ;
                     90  ;Transfer data to R2...R5
004C: 7800     R     91          MOV R0,#TABLE  ;Set pointers
004E: 7902           92          MOV R1,#02H    ;Pointer R2
0050: E6             93  TRANSFER:MOV A,@R0
```

```
ASM51  TSW  ASSEMBLER   Demo program for PCD8584 I2C-routines

LOC   OBJ          LINE  SOURCE

0051: F7            94          MOV @R1,A
0052: 08            95          INC R0
0053: 09            96          INC R1
0054: D500F9   R    97          DJNZ NR_BYTES,TRANSFER
0057: ED            98          MOV A,R5        ;Mask of hour counter
0058: 543F          99          ANL A,#3FH
005A: FD           100          MOV R5,A
                   101  ;
                   102  ;Data must now be displayed on LCD display.
                   103  ;First minutes and hours (in R4 and R5) must be
                   104  ;converted from BCD to LCD segment data.The segment
                        ;data
                   105  ;will be transferred to TABLE. R0 is pointer to
                        ;table
005B: 7800     R   106          MOV R0,#TABLE
005D: 7600         107          MOV @R0,#00H    ;Control word for PCF8577
005F: 08           108          INC R0 0060: 120080   R   109          CALL CONV
                   110  ;
                   111  ;Switch on dp between hours and minutes
0063: 430301   R   112          ORL TABLE+3,#01H
                   113  ;If lsb of seconds is '0' then switch on dp.
0066: EB           114          MOV A,R3        ;Get seconds
0067: 13           115          RRC A           ;lsb in carry
0068: 4003         116          JC PROCEED
006A: 430101   R   117          ORL TABLE+1,#01H;switch on dp
                   118  ;
                   119  ;Now the time (hours,minutes) can be displayed on
                        ;the LCD
006D:              120  PROCEED:
006D: D200     R   121          SETB DIR        ;Direction 'transmit'
006F: 750000   R   122          MOV BASE,#TABLE
0072: 750005   R   123          MOV NR_BYTES,#05H
0075: 750074   R   124          MOV SLAVE,#PCF8577W
0078: 120000   R   125          CALL START      ;Start transmission
                   126  ;
007B: 3000FD   R   127  FIN_4:  JNB I2C_END,FIN_4
007E: 8026         128          JMP ADCON       ;Proceed with AD-conversion
                                                ;part
                   129  ;
                   130 ;*************************************************************
                   131  ;Routines used by clock part of demo
                   132  ;
                   133  ;CONV converts hour and minute data to LCD data and
                        ;stores
                   134  ;it in TABLE.
0080: 90009C   R   135  CONV:   MOV DPTR,#LCD_TAB ;Base for LCD segment
                                                ;table
0083: ED           136          MOV A,R5        ;Hours to accu
0084: C4           137          SWAP A          ;Swap nibbles
0085: 120096   R   138          CALL LCD_DATA   ;Convert 10's hours to LCD
                                                ;data in table
0088: ED           139          MOV A,R5        ;Get hours
0089: 120096   R   140          CALL LCD_DATA
008C: EC           141          MOV A,R4        ;Get minutes
008D: C4           142          SWAP A
008E: 120096   R   143          CALL LCD_DATA   ;Convert 10's minutes
```

```
ASM51  TSW  ASSEMBLER   Demo program for PCD8584 I2C-routines

LOC   OBJ          LINE  SOURCE

0091: EC           144          MOV A,R4
0092: 120096   R   145          CALL LCD_DATA   ;Convert minutes
0095: 22           146          RET
                   147  ;
                   148  ;LCD_DATA gets data from segment table and stores it
                        ;in TABLE
0096: 540F         149  LCD_DATA:ANL A,#0FH     ;Mask off LS-nibble
0098: 93           150          MOVC A,@A+DPTR  ;Get LCD segment data
0099: F6           151          MOV @R0,A       ;Save data in table
009A: 08           152          INC R0
009B: 22           153          RET
                   154  ;
                   155  ;LCD_TAB is conversion table for LCD
009C:              156  LCD_TAB:
009C: FC60DA       157          DB 0FCH,60H,0DAH; '0','1','2'
009F: F266B6       158          DB 0F2H,66H,0B6H; '3','4','5'
00A2: 3EE0FE       159          DB 3EH,0E0H,0FEH; '6','7','8'
00A5: E6           160          DB 0E6H         ; '9'
                   161  ;
                   162 ;*************************************************************************
                   163  ;
                   164  ;
                   165  ;These part of the program reads an analog
                        ;input-channel.
                   166  ;Displaying is done on the LED-display
                   167  ;On odd-seconds the channel number will be
                        ;displayed.
                   168  ;On even-seconds the analog value of this channel is
                        ;displayed
                   169  ;Then the next channel is displayed.
                   170  ;
00A6: EB           171  ADCON:  MOV A,R3         ;Get seconds
00A7: 13           172          RRC A           ;lsb to carry
00A8: 503C         173          JNC NEW_MEAS    ;Even seconds; do a
                                                ;measurement on the current
                                                ;channel
                   174  ;
                   175  ;Display and/or update channel
00AA: 33           176          RLC A           ;Restore accu
00AB: B51402   R   177          CJNE A,PREVIOUS,NEW_CH ;If new seconds,
                                                ;update channel number
00AE: 800A         178          JMP DISP_CH
00B0: 0515     R   179  NEW_CH: INC CHANNEL
00B2: E515     R   180          MOV A,CHANNEL   ;If channel=4 then
                                                ;channel:=0
00B4: B40403       181          CJNE A,#04,DISP_CH
00B7: 751500   R   182          MOV CHANNEL,#00
00BA: 8B14     R   183  DISP_CH:MOV PREVIOUS,R3 ;Update previous seconds
00BC: E515     R   184          MOV A,CHANNEL   ;Get segment value of
                                                ;channel
00BE: 900193   R   185          MOV DPTR,#LED_TAB
00C1: 93           186          MOVC A,@A+DPTR
                   187  ;
00C2: 7800     R   188          MOV R0,#TABLE   ;Fill table with I2C data
```

```
ASM51  TSW  ASSEMBLER   Demo program for PCD8584 I2C-routines

LOC   OBJ          LINE  SOURCE

00C4: 7600         189          MOV @R0,#00     ;SAA1064 instruction byte
00C6: 08           190          INC R0
00C7: 7677         191          MOV @R0,#77H    ;SAA1064 control byte
00C9: 08           192          INC R0
00CA: F6           193          MOV @R0,A       ;Channel number
00CB: E4           194          CLR A
00CC: 08           195          INC R0
00CD: F6           196          MOV @R0,A       ;Second digit
00CE: 08           197          INC R0
00CF: F6           198          MOV @R0,A       ;Third digit
00D0: 08           199          INC R0
00D1: F6           200          MOV @R0,A       ;Fourth byte
                   201  ;
00D2: D200     R   202          SETB DIR        ;I2C transmission of channel
                                                ;number
00D4: 750000   R   203          MOV BASE,#TABLE
00D7: 750006   R   204          MOV NR_BYTES,#06H
00DA: 750076   R   205          MOV SLAVE,#SAA1064W
00DD: 120000   R   206          CALL START
                   207  ;
00E0: 3000FD   R   208  FIN_5:  JNB I2C_END,FIN_5
00E3: 020027   R   209          JMP REPEAT      ; Repeat clock and AD cycle
                                                ; again
                   210  ;
                   211  ;
                   212  ;Measure and display the value of an AD-channel
00E6: 120108   R   213  NEW_MEAS: CALL AD_VAL   ;Do measurement
                   214  ;Wait till values are available
00E9: 3000FD   R   215  FIN_6:  JNB I2C_END,FIN_6
                   216  ;Relevant byte in TABLE+1. Transfer to AN_VAL
00EC: 7801     R   217          MOV R0,#TABLE+1
00EE: 8616     R   218          MOV AN_VAL,@R0
00F0: E516     R   219          MOV A,AN_VAL    ;Channel value in accu for
                                                ;conversion
                   220  ;AN_VAL is converted to BCD value of the measured
                        ;voltage.
                   221  ;Input value for CONVERT in accu
                   222  ;Address for MSByte in R1
00F2: 7917     R   223          MOV R1,#CONVAL
00F4: 120154   R   224          CALL CONVERT
                   225  ;Convert 3 bytes of CONVAL to LED-segments
00F7: 900193   R   226          MOV DPTR,#LED_TAB ;Base of segment table
00FA: 7817     R   227          MOV R0,#CONVAL
00FC: 12018A   R   228          CALL SEG_LOOP
                   229  ;Display value of channel to LED display
00FF: 12012C   R   230          CALL LED_DISP
0102: 3000FD   R   231  FIN_8:  JNB I2C_END,FIN_8 ;Wait till I2C
                                                ;transmission is ended
0105: 020027   R   232          JMP REPEAT      ;Repeat clock and AD cycle
                   233  ;
                   234 ;*************************************************************
                   235  ;Routines used for AD converter.
                   236  ;
                   237  ;AIN reads an analog values from channel denoted by
                        ;CHANNEL.
```

```
ASM51  TSW  ASSEMBLER   Demo program for PCD8584 I2C-routines

LOC   OBJ          LINE  SOURCE

                    238  ;Send controlbyte:
0108: D200     R    239  AD_VAL: SETB DIR         ;I2C transmission
010A: 7800     R    240          MOV R0,#TABLE   ;Define control word
010C: A615     R    241          MOV @R0,CHANNEL
010E: 750000   R    242          MOV BASE,#TABLE ;Set base at table
0111: 750001   R    243          MOV NR_BYTES,#01H ;Number of bytes to be
                                                   ;send
0114: 75009E   R    244          MOV SLAVE,#PCF8591W ;Slave address PCF8591
0117: 120000   R    245          CALL START      ;Start transmission of
                                                   ;controlword
011A: 3000FD   R    246  FIN_7:  JNB I2C_END,FIN_7 ;Wait until tranmission is
                                                   ;finished
                    247  ;Read 2 data bytes from AD-converter
                    248  ;First data byte is from previous conversion and not
                    249  ;relevant
011D: C200     R    250          CLR DIR         ;I2C reception
011F: 750000   R    251          MOV BASE,#TABLE ;Bytes must be stored in
                                                   ;TABLE
0122: 750002   R    252          MOV NR_BYTES,#02H; Receive 3 bytes
0125: 75009F   R    253          MOV SLAVE,#PCF8591R ;Slave address PCF8591
0128: 120000   R    254          CALL START
012B: 22            255          RET
                    256  ;
                    257  ;LED_DISP displays the data of 3 bytes from address
                         ;CONVAL
012C:               258  LED_DISP:
012C: 431780   R    259          ORL CONVAL,#80H ;Set decimal point
012F: 7800     R    260          MOV R0,#TABLE
0131: 7917     R    261          MOV R1,#CONVAL
0133: 7600          262          MOV @R0,#00     ;SAA1064 instruction byte
0135: 08            263          INC R0
0136: 7677          264          MOV @R0,#01110111B ;SAA1064 control byte
0138: 08            265          INC R0
0139: 7600          266          MOV @R0,#00     ;First LED digit
013B: 08            267          INC R0
013C: 120185   R    268          CALL GETBY      ;Second digit
013F: 120185   R    269          CALL GETBY      ;Third digit
0142: 120185   R    270          CALL GETBY      ;Fourth digit
0145: D200     R    271          SETB DIR        ;I2C transmission
0147: 750000   R    272          MOV BASE,#TABLE
014A: 750006   R    273          MOV NR_BYTES,#06
014D: 750076   R    274          MOV SLAVE,#01110110B
0150: 120000   R    275          CALL START      ;Start I2C transmission
0153: 22            276          RET
                    277  ;
                    278  ;CONVERT calculates the voltage of the analog value.
                    279  ;Analog value must be in accu
                    280  ;BCD result (3 bytes) is stored from address stored
                         ;in R1
                    281  ;Calculation: AN_VAL*(5/256)
0154: 75F005        282  CONVERT:MOV B,#05
0157: A4            283          MUL AB
                    284  ;b2..b0 of reg. B : 2E+2..2E0
                    285  ;b7..b0 of accu   : 2E-1..2E-8
0158: A7F0          286          MOV @R1,B       ;Store MSB (10E0-units)
015A: 09            287          INC R1
```

```
ASM51  TSW  ASSEMBLER   Demo program for PCD8584 I2C-routines

LOC   OBJ          LINE  SOURCE

015B: 7700         288          MOV @R1,#00      ;Calculate 10E-1 unit
                                                 ;(10E-1 is 19h)
015D: B41C02       289  TEN_CH: CJNE A,#19H+03H,V1  ;Check if accu <= 0.11
0160: 8002         290          JMP TENS         ;accu=0.11; update tens
0162: 4006         291  V1:     JC NX_CON        ;accu<0.11; update hundreds
0164: C3           292  TENS:   CLR C           ;Calculate new value
0165: 9419         293          SUBB A,#19H
0167: 07           294          INC @R1          ;Update BCD byte
0168: 80F3         295          JMP TEN_CH
                   296  ;Correction may be neccessary. With 8 bits '0.1' is
                        ;in fact 0.0976.
                   297  ;A digit of '0A' may appear. Correct this by
                        ;decrementing the digit.
                   298  ;The intermediate result result must be corrected
                        ;with 10*(0.1-0.0976)
                   299  ;This is 06H
016A: B70A03       300  NX_CON: CJNE @R1,#0AH,PROC_CON ; If digit is '0A'
                                                 ;then correct
016D: 17           301          DEC @R1
016E: 2419         302          ADD A,#19H
0170: 09           303  PROC_CON:INC R1
0171: 7700         304          MOV @R1,#00      ;Calculate 10E-2 units
0173: B40302       305  HUND:   CJNE A,#03H,V2  ;Check if accu <= 10E-2
0176: 8002         306          JMP HUNS         ;accu=10E-2; update hundreds
0178: 4006         307  V2:     JC FINISH        ;accu<10E-2; conversion
                                                 ;finished
017A: C3           308  HUNS:   CLR C           ;Calculate new value
017B: 9403         309          SUBB A,#03H
017D: 07           310          INC @R1          ;Update BCD byte
017E: 80F3         311          JMP HUND
0180: B70A01       312  FINISH: CJNE @R1,#0AH,FIN ;Check if result is '0A'.
                                                 ;Then correct.
0183: 17           313          DEC @R1
0184: 22           314  FIN:    RET
                   315  ;
                   316  ;CALLBY tranfers byte from @R1 to @R0
0185: E7           317  GETBY:  MOV A,@R1
0186: F6           318          MOV @R0,A
0187: 08           319          INC R0
0188: 09           320          INC R1
0189: 22           321          RET
                   322  ;
                   323  ;SEG_LOOP converts 3 values to segment values.
                   324  ;R0 contains address of source and destination
                   325  ;DPTR contains base of table
018A: 7903         326  SEG_LOOP: MOV R1,#03     ;Loop counter
018C: E6           327  INLOOP: MOV A,@R0      ;Get value to be displayed
018D: 93           328          MOVC A,@A+DPTR  ;Get segment value from
                                                 ;table
018E: F6           329          MOV @R0,A       ;Store segment data
018F: 08           330          INC R0
0190: D9FA         331          DJNZ R1,INLOOP
0192: 22           332          RET
                   333  ;
                   334  ;
```

```
ASM51  TSW  ASSEMBLER   Demo program for PCD8584 I2C-routines

LOC   OBJ          LINE  SOURCE

                    335  ;LED_TAB is conversion table for BCD to LED segments
0193:               336  LED_TAB:
0193: 7D483E        337          DB 7DH,48H,3EH  ; '0','1','2'
0196: 6E4B67        338          DB 6EH,4BH,67H  ; '3','4','5'
0199: 734C7F        339          DB 73H,4CH,7FH  ; '6','7','8'
019C: 4F            340          DB 4FH          ; '9'
                    341  ;
                    342 ;*************************************************************
                    343  ;
----                344          RSEG RAMTAB
0000:          R    345  TABLE:  DS 10
                    346  ;
000A:               347          END
```

AN426

Controlling air core meters with the 87C751 and SAA5775

INTRODUCTION

Often, certain classes of microcontroller applications surface where large amounts of on-chip resources such as a large program memory space and numerous I/O pins are not required. These applications are typicially cost sensitive and desirable attributes of the MCU include low cost and modest on-chip resources such as program and data memory, I/O, and timer-counters. Substaintial benefits of reduced design cycle time can be realized by using an industry-standard architecture having software compatibility with existing popular microcontrollers.

THE 87C751

The Philips 87C751 is one such microcontroller that easily meets these requirements. This device, shown in Figure 1, has a 2k x 8 program memory, 64 bytes of RAM, 19 parallel I/O lines, and a 16-bit autoreload timer-counter. It also includes an I^2C serial interface and a fixed rate timer. The 87C751 is based on the 80C51 core and thus uses an industry-standard architecture and instruction set. The device is available in both ROM (83C751) and EPROM (87C751) versions. The EPROM version is available in both UV erasable and OTP packages. References to the 87C751 in this document also apply to the 83C751, unless explicitly stated.

TYPICIAL APPLICATION

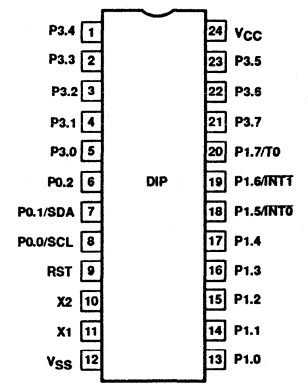
A typicial example of such an application is the interface between the 87C751 and the Philips SA 5775 Air Core Meter Driver shown in Figure 2. This circuit includes the 87C751 microcontroller, the SA 5775 air core meter driver, an NE555 timer, and discrete support components.

An air core meter differs from a conventional (d'Arsonval) meter movement in that it has no spring to return the needle to a predetermined position, no zeroing adjustment, and no permanent magnet in the classical sense. Instead, it consists of two coils of wire wound in quadrature with each other around a central core in which there is a disc magnetized along its diameter. A shaft is placed through the center of this disc so that the shaft rotates with the disc. An indicating needle attached to this shaft will rotate with it.

An air core meter differs from a conventional (d'Arsonval) meter movement in that it has no spring to return the needle to a predetermined position, no zeroing adjustment, and no permanent magnet in the classical sense. Instead, it consists of two coils of wire wound in quadrature with each other around a central core in which there is a disc magnetized along its diameter. A shaft is placed through the center of this disc so that the shaft rotates with the disc. An indicating needle attached to this shaft will rotate with it.

SA 5775 Air Core Meter Driver

The SA 5775 is a monolithic driver for controlling air core meters typically used in automotive instrument clusters and is shown in Figure 3. The SA 5775 receives a 10-bit serial word and converts that word to four voltage outputs that appear at the SINE+, SINE–, COSINE+, and COSINE– outputs. The differential voltage at the SINE outputs are applied to one coil of the meter and the COSINE outputs are applied to the other coil of the meter.



Pin	Function	Pin	Function
1	P3.4	15	P1.0
2	P3.3	16	P1.1
3	P3.2	17	P1.2
4	P3.1	18	P1.3
5	N.C.	19	P1.4
6	P3.0	20	P1.5/INT0
7	P0.2	21	N.C.
8	P0.1/SDA	22	N.C.
9	P0.0/SCL	23	P1.6/INT1
10	N.C.	24	P1.7/T0
11	RST	25	P3.7
12	X2	26	P3.6
13	X1	27	P3.5
14	VSS	28	VCC

Figure 1. Pin Configuration

Controlling air core meters with the 87C751 and SAA5775 AN426

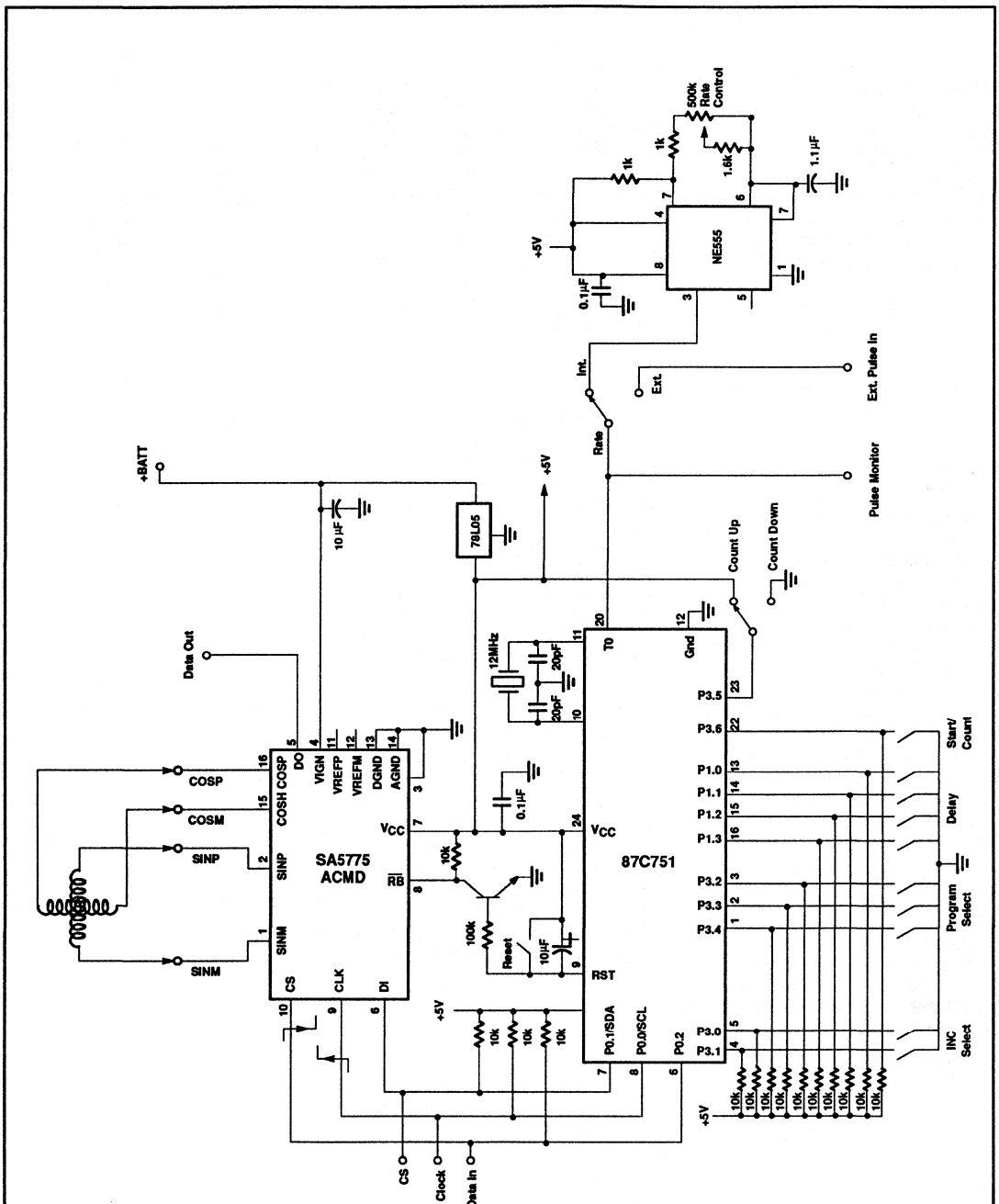


Figure 2. Interface Between the 87C751 and the Philips SA 5775

Controlling air core meters with the 87C751 and SAA5775 AN426

The currents through these coils produce a resultant magnetic force which is the vector sum of the magnetic forces produced by each of the two coils. Since the currents through the coils are bidirectional this magnetic vector can rotate through a full 360 degrees. The magnetized disc within the air core meter will follow the rotating vector and the needle will indicate the vector's current position. Since 10 bits are used, there are 1024 discrete words available resulting in an angular displacement of 0.3516 degrees per bit. This is small enough to provide an apparently smooth movement of the needle. The smoothness of the motion will depend greatly on the damping factor of the meter movement.

A simplified block diagram of the SA5775 is shown in Figure 4. This device consists of a serial-in/parallel-out shift register, a data latch, a D/A converter, buffers, and an internal voltage reference.

A logic high must be present on the chip select (CS) input to clock in the data. Data appearing on the data input (DI) pin is clocked into the shift register on the rising edge of the clock (CLK) input. The data output (DO) pin is the overflow from the shift register, allowing the user to daisy chain multiple SA5775 devices. Note that data is clocked out of this pin on the falling edge of the clock. The CS pin is also used to latch the parallel outputs of the shift register into the data latch. The outputs of the data latch feed the inputs to the D/A converter. The D/A converter outputs are buffered to form the drive signals for the meter coils.

A voltage reference for the D/A converter is provided internally. It is possible to externally force different values for these voltages and pins are provided for this purpose. However, this is not generally recommended as this could lead to increased power dissipation.

The D/A converter circuits and its associated output buffers are purposely designed such that the span of these circuits does not include the power supply rails. This is to avoid inaccuracies that would otherwise occur if the output were to become very close to either supply rail. With a supply voltage of 14 volts (VIGN), the positive reference (VREF+) is approximately 8 volts and the negative reference (VREF–) is approximately 1 volt. The outputs will then span a range from 1 volt to approximately 11 volts. The maximum output is [(VREF+) + 0.41((VREF+)–(VREF–))]. The SA5775 is designed to drive air core meters having a minimum winding impedance of 200 ohms.

The clock high and low time requirements are each 200 ns minimum, implying a maximum data rate approaching 2.5 megabits per second. At this rate it would require approximately 4 ms to ramp from zero to full scale if all binary codes were loaded into the SA5775. However, the air core meter cannot respond to such data rates.

87C751 Microcontroller

The 87C751 microcontroller provides all of the intelligence in this application. It samples various input ports to determine which demonstration programs to run, the incremental step sizes for angular displacement of the meter core, and the time delay between increments. In one of the demonstration modes, it also samples a variable frequency input and positions the meter core in response to the frequency of that input. The 87C751 also transmits the 10-bit serial data to the SA5775. Data input (DI), Clock (CLK), and Chip Select (CS) lines are driven from the 87C751.

Port 0 of the 87C751 is a 3-bit wide port and is used for communicating data to the ACMD. Data is transmitted, MSB first, in a serial stream clocked into the DI of the SA5775 on the rising edge of the clock. In order to clock in data, the CS pin of the SA5775 must be high. The data in the input register is shifted into a latch that drives the DAC on the high to low transition of the CS line. As data is shifted into the ACMD, it overflows through the Data Out (DO) pin on the falling edge of the clock. With this facility, multiple ACMDs can be daisy-drained with DO of one ACMD being connected to DI of the next one, and common clock and chip select lines may be used. This simplifies the interfacing to multiple meter drivers.

The 78L05 regulator (Q2) provides 5 Volt power for the board so that single supply of +14 volts can be applied to the board.

Three rotary switches are used on this board. The PROGRAM SELECT switch (S3) is used to select the program routine that is executed, the INC SELECT (S2) switch selects the incremental step sizes of the two of the routines, and the DELAY switch (S4) is used to set the delay between successive word transmissions in one of the routines.

The START/COUNT button (S5) is used to begin execution of a routine, and to cause the next incremental step in Routine #1.

The COUNT UP/DOWN switch (S6) is used in Routine #1 to determine whether the count is increased or decreased with transmission of successive words.

NE555 Timer

The NE555 timer shown in this application example is used as a free running square-wave generator used to simulate sensor inputs such as those which might be found in an automobile, etc. The NE555 timer (U4) operates in the astable mode to produce an output frequency that can be varied from about 1Hz to about 200 Hz. Three of the program routines measure the input period and produce an output code that is proportional to the frequency present at pin 20 (TO) of the microcontroller. A RATE switch (S7) is used to select between the on board oscillator or an external source.

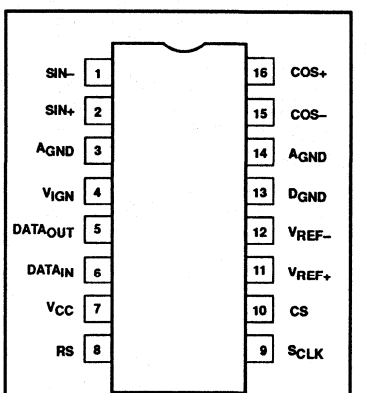
The program listing is included at the end of this application note.

Program Entry

The program starts at address 030(hex) on line 21 of the program listing. The first task is to write 1's to all pins of each port.

Lines 25 and 26 clear registers 6 and 7. These registers are used in this program only to hold the data that is sent out to the ACMD. The registers are cleared to be sure that the starting value is zero.

At line 27 the program waits until the START/COUNT button (S5) is depressed before continuing. Lines 28 and 29 set the timer to overflow after 10ms. This is done by setting the timer registers for a count of 10,000 microseconds less than full scale. When the timer counter overflows the timer flag is set, and the timer is reloaded with the value in the timer register. By examining the timer flag we know when 10ms has expired.



NOTE: D package available in large SO (SOL) package only.

Figure 3. D and N Packages

Controlling air core meters with the 87C751 and SAA5775 AN426

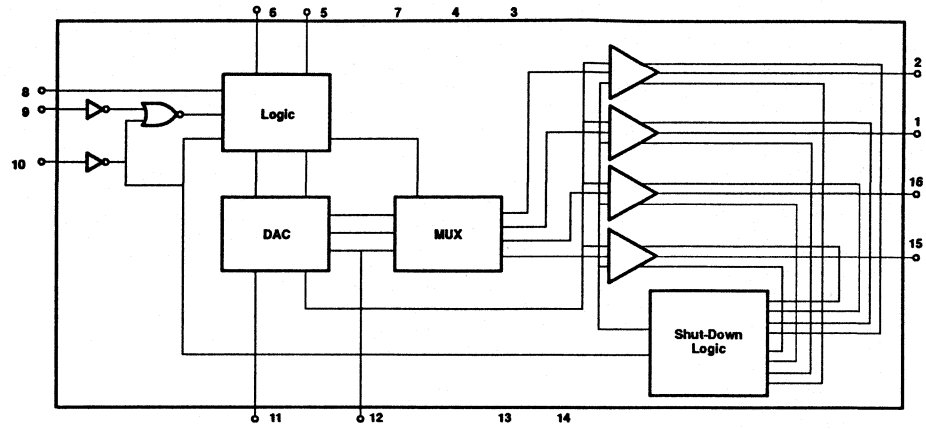


Figure 4. Block Diagram of the SA5775

Line 30 calls subroutine RPS (Read Port Selected), which reads Port 3 to determine which routine has been selected. Since the PROGRAM SELECT switch (S3) is connected to port pins P3.2 through P3.4, subroutine RPS (lines 507 through 511 at the end of the program) first reads Port 3 into the accumulator, then complements it because the switches used are complementary binary. The reading is then rotated right once and the upper nibble and the LSB (least significant bit) are masked off, leaving twice the value of the port selected in the accumulator. Twice the read value is needed for the next few main program lines that determine which routine to execute.

Line 31 moves the address of label JMPTBL (Jump Table) to the 16-bit Data Pointer (DPTR) register. Line 32 causes a program jump to the address that is the sum of the value in the accumulator (two times the routine number selected) plus the DPTR register. Since each of the commands on lines 33 through 40 are two byte commands, these addresses are all separated by two bytes; hence, the need for the accumulator to contain a number that is twice the number of the selected routine.

Routine 0

This routine begins on line 41 by incrementing the 10-bit word in registers 7 and 6 by the amount indicated by the setting of the INCREMENT SELECT switch, then sending that word to the SA5775. When a full scale overflow is detected, a full scale code (3FF hex) is sent out, followed by a delay of 500 ms, then successive output codes are sent out, decremented by an amount indicated by the INCREMENT SELECT switch. When an underflow is detected a code of zero scale is sent and the routine returns to the beginning of the program. This routine is implemented with a series of subroutine calls.

The SO subroutine begins on line 356 and starts by sending out whatever ten bits that in the two LSBs of register 7 (R7) plus the 8 bits of R6 by calling the SENDIT subroutine. Then it calls the UP subroutine, which increases the word value to be sent out. The program then jumps to the beginning of this subroutine, repeating the process of sending out a word and incrementing to the next word until an overflow from the tenth bit (bit 2 of R7) is detected at line 362.

The SENDIT subroutine (beginning on line 476) brings the CS line high, sets a bit counter (R1) to 2 (to send out two bits of R7), brings the value of R7 to the accumulator, rotates the accumulator to the right three times through the carry bit to bring the two LSBs to the position of the two MSBs, calls the SEND1 routine, which sends the number of bits in the accumulator, starting with the MSB, indicated by R1. Counter R1 is then set to 8 to send out all 8 bit of R6 and the accumulator is loaded with the contents of R6. The SEND1 routine is again called to send out the final 8 bits, and, on line 491, the CS line is brought low, loading the SA5775 internal parallel latch with the contents of the input shift register.

The SEND1 routine rotates the accumulator left through the carry bit, moves the value of the carry bit to port pin P0.1 (SDA – Serial Data pin), waits to provide a setup time, brings the clock low, waits, brings the clock high, waits, then decrements bit counter sends the next bit if the counter is not zero. A return is executed when the counter becomes zero.

The UP subroutine, beginning at line 364, reads the delay selected by switch S4 at port pin P1, complements it (again, because the rotary switches are complementary binary), masks off the upper four bits (because the delay switch has just four positions and is connected to the lower four bits of the port), multiplies it by 4 (rotates left twice), then moves the result to R1. If R1 is not zero, the program jumps around line 376 and calls a 10ms delay (subroutine DLY10MS) the number of times entered into R1.

The 10ms delay subroutine (starting at line 436) sets the timer for 10ms, waits at line 446 for the timer flag to be set, clears the timer flag, stops the timer, and returns, in this case, to line 379, where the program decrements R1 and repeats the 10ms delay until R1 is zero.

If the selected delay was zero, the program jumps from line 376 to line 380 and reads port 3 to determine the amount the sent out word is to change from the value previously sent out. The accumulator is complemented and the upper 6 bits masked off to recover only the two bits of the selected increment amount. Since increments of 1, 2, 3, or 4 LSBs are hardly noticeable, the program then

multiplies the result by 8 (rotate left three times). To insure a minimum change amount, the accumulator is increment by one at line 386. This all means that the increment amounts that can be selected are 1, 9, 17, or 25 LSBs. This amount is added, in lines 387 through 391, to the word previously send out and we return from this subroutine.

After calling the S0 subroutine, PR0GO call the FULLSC (full scale) subroutine, which sends out the full scale code of 3E8(hex). Although a 10-bit full scale code would be 3FF(hex), going only to 3E8 allows an easy distinction between zero scale and full scale when looking at the display. The FULLSC subroutine is found at line 352.

After advancing to full scale, there is a 500ms delay, found at line 464 and called from line 48, then 49 calls the S0D subroutine to send out decreasing word values.

The S0D subroutine begins at line 393 and begins by sending out the current word in R7 and R6 from line 398, then calling subroutine DOWN, which calculates the next (decreasing) word to send out. DOWN begins at line 402. It essentially does the same thing as the UP subroutine, but subtracts the INCREMENT SELECT value from the previously sent word rather than adding to it.

At line 50 subroutine ZEROSC is called to send a zero scale code to the SA5775, then the program branches back to the beginning.

Routine 1

This routine is selected with the PROGRAM SELECT switch is in position 1 or position 9. Routine 1 (PROG1) increments or decrements the word send out, depending upon the setting of the COUNT UP/COUNT DOWN switch, S6. The amount of change is determined by the setting of the INC SELECT switch, S2.

At line 63, the program examines S6 at port pin P3.6 and jumps to the decrement portion of the routine if the pin is low. If this pin is high, the UP subroutine is called from line 64 to increase the R7/R6 word value. The UP subroutine was previously described.

If pin P3.6 is low, the DOWN subroutine (line 402) decreases the previous word sent out by the amount determined from the INC SELECT switch setting.

To insure enough delay to allow the user time to release the START/COUNT button (S5), a delay of 200ms is included at line 66 before jumping to line 27, where another depression of the START/COUNT button is awaited. If S3 (PROGRAM SELECT) is still set to 1 or 9, depression of S5 will cause a jump back to line 52. If another program is selected, the program will jump to the selected routine.

Holding down S5 with PROGRAM SELECT set at position 1 or 9 will cause increasing or decreasing word values to be sent to the SA5775.

Routine 2

PROG2 is the most complex of all these routines. The purpose of this routine is to cause the air core meter deflection to represent the frequency presented at the timer/counter input to the microcontroller. This is done by measuring the period of the input square wave and taking the inverse of the period. The input here must be a square wave because a slow rise and fall time at this input will cause fluctuating readings. To determine the frequency by counting pulses for a time would require a much longer time and, therefore, is impractical.

The MEAS (measure) subroutine is called at line 79 to measure the period of the input waveform and the CALC (calculate) subroutine is called at line 80 to calculate the code to send to the SA5775. The SENDIT subroutine is then called to send the word to the SA5775 and the program jumps back to line 28.

The MEAS subroutine begins at line 83 by being sure the timer is not running and clearing the timer (overflow) flag, then entering zero into both high and low bytes of the timer and the timer register. The carry bit is then cleared (line 90) and the timer started and the timer interrupt enabled.

Lines 93 and 94 form a short loop that waits until either the carry bit is set or until the TO input is low. The carry bit is set when the timer has gone beyond one second. This is done by the timer interrupt subroutine, found at lines 16 through 19. If the TO input never goes low, we know the frequency is at or near zero and the program jumps to GZS (line 108) where R3 is loaded with a 1F (hex) to cause the CALC subroutine to load zero scale into R7/R6.

When (and if) TO is found to be low, the program jumps to line 95 and waits for that input to go high. Time out process is the same as above.

Now that the TO input is found high (if is is before the one second time out), the timer and carry bit are cleared in lines 97 through 100 (R3 is an extension of the timer).

At lines 101 through 107 we wait for one complete cycle at the TO input, with the timer/counter measuring that period, then return to line 80, where the CALC subroutine is called.

The CALC subroutine, starting at line 113, begins by initializing the word to send out (R7/R6) to zero, clearing the carry bit, checking to see if R3 indicates a time above one second, returning to line 81 if it does. Otherwise the program continues at line 26, where the program checks to see if the input frequency is beyond full scale (timer reading above 00 12 88 hex). If it is, R7/R6 is loaded with 12 88 hex (full scale of decimal 1,000). This value was chosen because it is sufficiently far from zero scale that it is easily discerned from zero scale on the display.

If the result is not to be full scale or zero scale, the program continues at line 140 with a shift and subtract divide routine. The dividend would be 1,000,000 (decimal) to convert back to frequency in Hertz (period measurements is in microseconds), but that would provide a maximum count of 200 at 200Hz, only one fifth of the full scale desired of 1,000. So we made the dividend to be 5,000,000 decimal, or 4C 4B 40 hex.

This algorithm is found in lines 156 through 192 and works as follows:

1. Clear a counter.
2. Rotate dividend until the first one is in the second MSB position. Since a code of 4C has already provides that, no shifting is necessary.
3. Rotate the divisor (the period in microseconds in this case) left until the first one is in the second MSB position, but the first byte is LESS THAN the first byte of the dividend. Increment the counter each time the divisor is rotated.
4. Initialize a counter to zero.
5. Rotate the quotient (answer) and dividend one bit left.
6. If first byte of quotient is smaller than the first byte of the quotient, jump to step 8.
7. Add one to the quotient and subtract the divisor from the dividend.
8. Decrement the counter and go to step 5 if it is not zero.

Once the CALC subroutine is completed, the program calls SENDIT from line 81 and jumps, ultimately, to the selected routine.

Routine 3

PROG3, beginning at line 194, measures the input period four times, then calculates the code to display that is the average of these four readings.

It starts by setting a counter for three readings, taking those three readings and storing them in memory, beginning at RAM address 20 hex, using register RO as an index register.

At line 212 the program takes a fourth reading, then adds the three previous readings to it in lines 213 through 227; and divides the

sum by four (rotates right twice) in lines 229 through 239. The word to send out is then calculated from line 240 and sent to the ACMD, after which the program then looks for and jumps to the selected routine.

Routine 4
PROG4 begins at line 243 and displays the average of the current and last three words sent out.

RAM space used is first initialized to zero and a new reading is taken and a new word is calculated and saved. At lines 264 through 284, the new word is added to the last three readings and the average calculated and stored in RAM locations 28 and 29 (hex), and the average word is sent out.

At line 286, the program reads for the program selected and jumps to line 254 if this routine is selected, otherwise it goes to line 28.

Routine 5
PROG5 begins at line 293 and, very simply, send in sequence the codes for 1/8 through full scale in 1/8 scale steps, with 500ms between steps. It then steps down to zero scale in 1/8 scale steps, then returns to line 28.

Routine 6
PROG6 begins at line 314 and does the same as PROG5, but steps in 1/4 scale increments.

Routine 7
PROG7 loads the code for 3/8 scale into R7/R6, sends it, waits 500ms, changes r& for 5/8 scale, sends it, waits for 500ms, then repeats this sequence 9 more times (for a total of ten times), waits 500ms, then returns the output to zero scale and the program jumps to line 28.

Section 5
Development Support Tools

CONTENTS

Development Support Tools 551
Nohau EMUL51-PC — PC-Based In-Circuit Emulator 556
Metalink 561
SDS 8051 Stand-alone Debug Station 563
OM4142 (ASM51, as8051) Cross-Assembler Package 569
OM4144 (plm51, PLMTI51) Compiler Package 571
OM4129 Symbolic Debugging Package XRAY51 574

Section 5
Development Support Tools

DEVELOPMENT SUPPORT TOOLS

Philips Components manufactures support tools and also works closely with many "third-party" vendors who provide support tools for our wide variety of 80C51-based microcontroller derivatives.

Development Systems

In most cases, development systems are available in two versions for ROM and ROMless applications. The ROM emulation products are capable of supporting all versions of a given device type, including EPROM, ROM, and ROMless devices. For example, the SMI–83C451 emulator can support designs based on either the 87C451, 83C451, or 80C451. In contrast, a ROMless emulator can only support applications designed for a ROMless microcontroller. For example, the SMI–80C451 emulator can only support applications using the 80C451.

These development systems are designed to connect to an IBM–PC or compatible personal computer. The development system package includes the emulator hardware, host emulation software, cross-assembler, user's manuals, cables, and power supply.

EPROM Programming Support

Philips Components works closely with major suppliers of EPROM programming equipment to support our family of EPROM microcontrollers. As a result, EPROM programming support is available within the programming facilities of many major distributors.

The following is a list of vendors that offer support for Philips Components 80C51 microcontroller family

DEVELOPMENT SYSTEM CONTACTS

COMPANY	ADDRESS	TELEPHONE
Ashling Microsystems Limited	Plassey Technological Park Limerick, Ireland	(353) 63-334466
Nohau Corp.	51 E. Campbell Ave. Campbell, CA 95008	(408) 866–1820
MetaLink Corp.	325 E. Elliot Road, Suite 23 Chandler, AZ 85225	(602) 926–0797
Philips Components	Corporate Centre Building BAE-2 P.O. Box 218 5600 MD Eindhoven The Netherlands	31-40-724223

EPROM PROGRAMMING SUPPORT CONTACTS

COMPANY	ADDRESS	TELEPHONE
Data I/O Corp.	10525 Willows Road N.E. P.O. Box 97046 Redmond, WA 98073–9746	(206) 867–6899
GTEK, Inc.	P. O. Box 2310 Bay St. Louis, MS 39521–2310	(800) 282–4835
Logical Devices, Inc.	1201 Northwest 65th Place Ft. Lauderdale, FL 33309	(305) 974–0967
Logical Systems	P. O. Box 6184 Syracuse, NY 13217–6184	(315) 478–0722
Needham's Electronics	4535 Orange Grove Ave. Sacramento, CA 95841	(916) 924–8037
North Valley Products	P.O. Box 32899 San Jose, CA 95152	(408) 929–5345
BP Microsystems	10681 Haddington, Suite 190 Houston, TX 77043	(800) 225-2102
Stag Microsystems	528–5 Weddell Drive Sunnyvale, CA 94086	(408) 988–1118

Section 5 – Development Support Tools

SOFTWARE SUPPORT CONTACTS

COMPANY	ADDRESS	TELEPHONE
Franklin Software, Inc.	888 Saratoga Ave. #2 San Jose, CA 95129	(408) 296–8051
Archimedes Software, Inc.	2159 Union St. San Francisco, CA 94123	(415) 567–4010
BSO/Tasking	Tasking Software BV P.O. Box 899 3800 AW Amersfoort The Netherlands BSO Tasking 128 Technology Center P.O. Box 9164 Waltham, MA 02254-9164	31-33-55-85-84 (Telephone) 31-33-55-00-33 (Fax) (617) 894-7800 (Telephone) (617) 894-0551 (Fax) (710) 324-0760 (Telex) (800) 458-8276 (Toll Free)

MICROCONTROLLER DEVELOPMENT SYSTEMS

PRODUCT	DEVICES SUPPORTED	MANUFACTURER	SOFTWARE REV.
EMUL51–PC/E32	12MHz Emulator, 32k emulation memory	Nohau	
EMUL51–PC/E128	12MHz Emulator, 128k emulation memory		
EMUL51–PC/E32–16	16MHz Emulator, 32k emulation memory		
EMUL51–PC/E128–16	16MHz Emulator, 128k emulation memory		
EMUL51–PC/32–20	20MHz Emulator, 32k emulation memory		
EMUL51–PC/128–20	20MHz Emulator, 128k emulation memory		
EMUL51–PC/128–24	24MHz Emulator, 128k emulation memory		
EMUL51–PC/128–30	30MHz Emulator, 128k emulation memory		
EMUL51–PC/128–BS	12MHz Emulator, 128k bankswitched CODE memory		
POD–31	12MHz 8031 pod		
POD–C31	12MHz 80C31 pod		
POD–C31–1	16MHz 80C31 pod		
POD–C31–20	20MHz 80C31 pod		
POD–C31–24	24MHz 80C31 pod		
POD–c31–30	30MHz 80C31 pod		
POD–32	12MHz 8032 pod		
POD–C32	12MHz 80C32 pod		
POD–C32–16	16MHz 80C32 pod		
POD–C652	12MHz 80C652 pod		
POD–C652–16	16MHz 80C652 pod		
POD–C51B	12MHz bondout pod for 8051, 80C51, 83C552, 83C652, 83C654, 83C851, and EPROM or ROMless versions of the above		
POD–C451–DIP	12MHz 80C451 DIP pod		
POD–C451–DIP–16	16MHz 80C451 DIP pod		
POD–C451–PGA	12MHz 80C451 PLCC pod (PGA from pod)		
POD–C451–PGA–16	16MHz 80C451 PLCC pod (PGA from pod)		
POD–C451B–PGA	12MHz bondout pod for 83C451, 87C451, 80C451 PLCC (PGA from pod)		
POD–C552–PGA	12MHz 80C552 PLCC (PGA from pod)		
POD–C552B–PGA	12MHz bondout pod for 83C552, 87C552, 80C552 PLCC (PGA from pod)		
POD–C652B	Order as POD–C51B		
POD–C851B	Order as POD–C51B		
POD–C751	12MHz 83C751, 87C751 pod		
POD–C751–16	16MHz 83C751, 87C751 pod		
POD–C752	12MHz 83C752, 87C752 pod		
POD–C752–16	16MHz 83C752, 87C752 pod		
EMUL51–PC/TR4	12MHz 4k trace buffer option		
EMUL51–PC/TR16	12MHz 16k trace buffer option		
EMUL51–PC/TR4–16	16MHz 4k trace buffer option		
EMUL51–PC/TR4–20	20MHz 4k trace buffer option		
EMUL51–PC/TR16–16	16MHz 16k trace buffer option		
EMUL51–PC/TR16–20	20MHz 16k trace buffer option		
EMUL51–PC/TR16–24	24MHz 16k trace buffer option		
EMUL51–PC/TR16–30	30MHz 16k trace buffer option		

Section 5 – Development Support Tools

MICROCONTROLLER DEVELOPMENT SYSTEMS (continued)

PRODUCT	DEVICES SUPPORTED	MANUFACTURER	SOFTWARE REV.
EMUL51–PC/BOX–S EMUL51–PC/BOX–M EMUL51–PC/BOX–CS	Box with serial port and cable. Box allows operation of emulator external to PC. Box with serial port and modem Serial box with emulator (E128–16) and trace (TR16–16)	Nohau (Continued)	
PRODUCT	**DEVICES SUPPORTED**	**MANUFACTURER**	**SOFTWARE REV.**
MCP–Emulator Pods for MicroICE+: 8031PC 8031–20PC 8032PC 8052–12PC 8052PC 80451PC 80552PC 80652PC 80851PC 83053PC 83451PC 83552PC 83652PC 83654PC 83751PC 83751–16PC 83752PC	MicroICE+ Emulator (see below) 8031, 80C31 8031, 80C31 (20MHz) 8031, 80C31, 8032, 80C32 8031/32/51/52, 80C31/C32/C51/C52, 8751, 87C51/C52 (12MHz) 8031/32/51/52, 80C31/C32/C51/C52, 8751, 87C51/C52 (16MHz) 80C451 80C552 80C652 80C851 83C053, 87C054 83C451, 87C451, 80C451 80C552, 83C552, 87C552 83C652, 87C652, 80C652 83C654, 87C654 83C751, 87C751 (12MHz) 83C751, 87C751 (16MHz) 83C752, 87C752	MetaLink	2.6b
PRODUCT	**DEVICES SUPPORTED**	**MANUFACTURER**	**SOFTWARE REV.**
OM4120 OM1090WP OM1091P OM1091WP OM1092 OM1094P OM1094WP OM4123 OM1079 OM4124 OM4125 OM4129 OM4142 OM4144	8051 Family Stand-Alone Debug System (Does not include probe) 8XC552/562 PLCC Emulation Proge 8XC51/8XC652/8XC654 DIL Emulation Probe 8XC51/8XC652/8XC654 LCC Emulation Probe 8XC851 DIL and LCC Emulation Probe 8XC751 DIL Emulation Probe 8XC751 LCC Emulation Probe 8X451 LCC Emulation Probe 8XCL410 DIL Emulation Probe 68-Pin LCL to 64-Pin DIL Adapter 40-Pin DIL to 44-Pin LCC Adapter XRAY51 Symbolic Debug Package 80C51 Cross Assembler 80C51 PL/M Compiler	Philips Components	

Section 5 – Development Support Tools

EPROM MICROCOMPUTER PROGRAMMING SUPPORT

DEVICE	MANUFACTURER/MODEL	MODULE/ADAPTOR	SOFTWARE VERSION
87C51 DIP	Data I/O Unisite 40 Data I/O Unipak 2b Data I/O Series 1000 Stag PP41, PP42 Logical Devices ALLPRO GTEK 9000, 9800 N. Valley Products SPGM–100	 351B103 SR40 41M200 SAM–51SD	V2.2 V16 V05 (Use Intel 87C51 menu) V1.03 V1.47 V1.0
87C51 PLCC	Data I/O Unisite 40 Data I/O Unipak 2b Logical Devices ALLPRO N. Valley Products SPGM–100	Chipsite 351B103P Required SAM–51ASD	V2.3 V16 V1.47 V1.0
87C52 DIP	Data I/O Unisite 40 N. Valley Products Needham's Electronics	 SAM-52SD	V2.3 V1.0
87C52 PLCC	Data I/O Unisite 40 N. Valley Products Needham's Electronics	Chipsite SAM-52SD	V2.3 V1.0
87C054 DIP	N. Valley Products	SAM-054SD	V1.3
87C451 DIP	Logical Devices ALLPRO N. Valley Products SPGM–100	Required SAM–451SD	V1.47 V1.0
87C451 PLCC	N. Valley Products SPGM–100 Data I/O Unisite	SAM–451ASD Chipsite	V1.0 V2.8
87C528 DIP	Data I/O Unisite 40 N. Valley Products	 SAM-528SD	 V2.3
87C528 PLCC	Data I/O Unisite 40 N. Valley Products	Chipsite SAM-528ASD	V3.0 V2.3
87C550 DIP	Data I/O Unisite 40 N. Valley Products	 SAM-550SD	V3.2 V2.5
87C550 PLCC	Data I/O Unisite 40 N. Valley Products	Chipsite SAM-550ASD	V3.2 V2.5
87C552 PLCC	Data I/O Unisite N. Valley Products SPGM–100	Chipsite SAM–552ASD	V2.8 V2.2
87C654 DIP	Data I/O Unisite 40 N. Valley Products	 SAM-654SD	V2.6 V2.3
87C654 PLCC	Data I/O Unisite 40 N. Valley Products	Chipsite SAM-654ASD	V2.6 V2.3
87C751 DIP	Data I/O Unisite 40 Data I/O 29B, Unipak 2b Logical Devices ALLPRO N. Valley Products SPGM–100 Needham's Electronics MetaLink Logical Systems (Sunshine EW–901)	 351B113D OPTAPC–751 SAM–751SD 752/1 PGMPC PA751	V2.3 29B V6, Unipak 2B V18 V1.47 V1.0 V2.6a (use with MicroICE+)
87C751 PLCC	Data I/O Unisite 40 Logical Devices ALLPRO N. Valley Products SPGM–100	Chipsite Required SAM–751ASD	V2.5 V1.47 V1.0
87C752 DIP	Data I/O Unisite 40 N. Valley Products SPGM–100 Needham's Electronics MetaLink Logical Systems (Sunshine EW–901)	 SAM–752SD 752/1 PGMPC PA751	V2.6 V1.0 (Use Type 751) V2.6a (use with MicroICE+)
87C752 PLCC	N. Valley Products SPGM–100	SAM–752ASD	V1.0 (Use Type 751)

Section 5 – Development Support Tools

ADDITIONAL PROGRAMMING SUPPORT

DEVICE	MANUFACTURER	MODULE/ ADAPTOR	COMMENTS
87C51 PLCC	Logical Systems	PA51–44	Use with any 87C51 40-pin programming site.
87C451 DIP		PA451–64	Use with any 87C51 40-pin programming site.
87C451 PLCC		PA451–68	Use with any 87C51 40-pin programming site.
87C550 DIP		550BASE	Use with any 87C51 40-pin programming site.
87C550 PLCC		PA550–44	Use with any 87C51 40-pin programming site.
87C552		PA552–68	Use with any 8752/C52/C252/C51FA 40-pin programming site.
87C751 PLCC	Signetics	No part number assigned	This adapter allows programming the 87C751 PLCC part in conjunction with any programmer that can already program the DIP version of the part.

MICROCONTROLLER SUPPORT

PRODUCT	DEVICES SUPPORTED	MANUFACTURER	DESCRIPTION
SM8051ASMSD	8051 and derivatives	Signetics/MetaLink	Cross assembler for 8051 family
8051 C Compiler	8051 and derivatives	Franklin Software	C Compiler for 8051 family
8051 C Compiler	8051 and derivatives	Archimedes Software	C Compiler for 8051 family
S87C00KSD	--	Signetics	I^2C Demonstration Board. 87C751 controls various I^2C peripherals. Board has sockets for 87C752, 87C652, and 87C552 also.
--	--	Signetics	The Signetics computer Bulletin Board system has available a microcontroller newsletter, application and demonstration programs for download, and the ability to send messages to microcontroler applications engineers. Access by modem at 2400, 1200, or 300 baud. The telephone numbers are: (800) 451–6644 (in the U.S.) or (408) 991–2406.

nohau
ELEKTRONIK AB

NOHAU EMUL51-PC – PC-based in-circuit emulator

1.0 System Architecture

Features of the Nohau EMUL51-PC in-circuit emulator include:

- Low-cost full real-time emulation
- IBM PC-bus plug-in boards or stand alone Box version with RS232-C connection to IBM PC
- Easy-to-learn user interface with windows and pull-down menus
- Source-level debugging in C, PL/M or Pascal with full support for typed symbols
- 16K frames by 48 bit real-time trace option
- Program Performance Analyzer

The Nohau EMUL51-PC consists of a board which plugs directly into the IBM PC/XT/AT bus for fast file transfer. An optional external box with a serial link is also available. The optional Trace board features an advanced trace function with many trigger capabilities.

The POD, which plugs into the target system, is connected with a 5 ft (1.5 m) ribbon cable to the Emulator board to provide a flexible operating range.

The EMUL51-PC uses no wait states and does not intrude on memory, stack, I/O or interrupt pins.

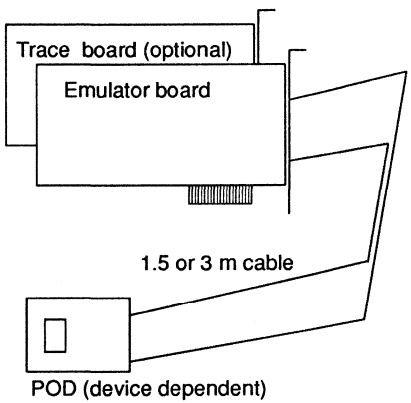


fig 1 EMUL51-PC plug-in board version

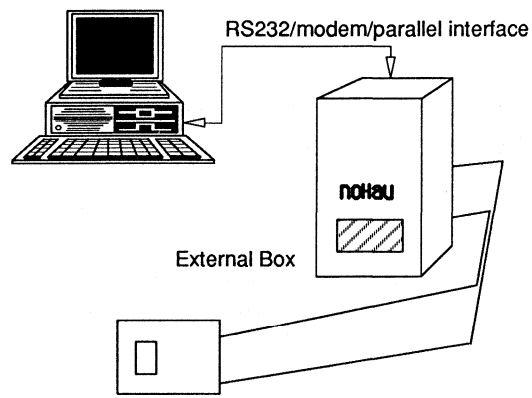


fig 2 External Box version

2.0 User Interface

The user interface is based on MS-DOS software. It incorporates a variety of techniques. The user can select to operate either with short typed commands or using pull-down menus. On-screen features include windows to monitor or alter:

- Assembly or high level language source code
- CPU registers
- Internal data and Special Function Registers
- External data
- Watch variables in C, PL/M or assembly
- Trace setup and display

All commands are supported with context-sensitive help and full on-line manual.

3.0 Specific Features

3.1 Source Level Debugging

Using high level language for code generation is a way to cut development time. Therefore the EMUL51-PC gives full support for debugging directly in C, PL/M or Pascal source code. This eliminates the need for paper listings. Breakpoints can be marked directly in the source code window. The user can single step through the program line by line and follow the program execution on the screen listing.

The trace permits the user to trace his source code in real-time.

All variables can be displayed and altered. This includes full support for typed symbols which permits the user to address such variables as floating-point, arrays and structures — both global and local.

3.2 Emulation Memory

The EMUL51-PC features 64 K of code memory and 64 K of external data memory. The addressable memory can be mapped either to target or to the emulator in 4K pages.

The emulator can load all common file formats and the user can view and alter all registers and memory areas of the microcontroller.

3.3 Breakpoints

The EMUL51-PC permits the user to generate program breakpoints in a number of ways:

- 64K program breakpoints
- 64K data read and 64K data write breakpoints
- Break on external signal
- Break on direct access to internal bit or byte memory
- Break on contents of internal register or memory
- Break on program access out-of-boundary

Using the Trace board it is possible to break on combinations of address, data, control signals, port signals and external signals.

3.4 Macros and debug session logging

Test session automation is made possible using the Macro commands. This permits the user to define his own command sequences using structures like IF/ELSE and REPEAT/WHILE. Such command sequences can be stored to a file which can be loaded automatically when the emulator is invoked.

A complete debug session and all setups can also be recorded to a file.

3.5 Trace Memory

The optional trace board features a trace buffer capable of storing 16K cycles of 48 bit data each. The 48 bits consist of address, data, control signals, port signals and a maximum of 18 external signals.

The trace memory can be displayed, reprogrammed and restarted during emulation.

3.5.1 Trace Filter

The trace filter makes it possible to select what events are stored in the trace buffer. The 20 qualifiers permit the user to define the criteria for which bus cycles are stored. The qualifiers can specify address, data, control signals, port signals and external signals.

3.5.2 Trace Trigger

The trace works much like a logic analyzer. It is therefore possible to trigger the trace on an event and display what happened before or after that event.

The trigger event can be defined using any of the 20 qualifiers. It is possible to trigger on boolean combinations of the qualifiers, or sequential combinations including a loop counter.

The trigger point can be selected anywhere within the 16K trace buffer to give full choice of pre/post trigger alignment.

3.5.3 Trace Display

The trace information can be displayed in high-level language statements, in disassembled form or in binary/hex form. It can also be stored to a file.

3.5.4 Program Performance Analyzer

With the PPA, information can be generated showing which addresses the user program spends its time on. The information can be displayed as statistics or in histogram form.

4.0 General Specifications

Host:	IBM PC/XT/AT/386, PS/2 or compatible with minimum 512K of RAM. Monochrome, color or enhanced graphics display.
External Box:	The emulator boards can be installed in an external box with serial communication to the PC.
Processors supported:	8051, 8052, 8031, 8032, 80C51, 80C52, 80C31, 80C32, 83/80C451, 83/80C552, 80C562, 83/80C652, 83/87C751, 83/87C752, 83/80C851 A switch from one microcontroller to another is made by changing the low cost POD.
File formats supported:	Intel HEX/OBJ/SYM/OMF, Avocet, Archimedes/IAR, HMI, Franklin/Keil, 2500AD, American Automation, Motorola S1
Clock speed:	Allows operation up to 20 MHz in real-time.
Power supply:	The boards are powered from the PC-bus. The Emulator board requires 1.7A/5V and the Trace board 1.3A/5V.

nohau
ELEKTRONIK AB

5.0 Ordering information

EMUL51-PC/E128-16	16 MHz Emulator board with 128K of emulation memory. Includes 5 ft ribbon cable, emulator software and manual
EMUL51-PC/TR16-16	16MHz Trace board with 16K of trace memory
POD31S-xxx	Generic POD for 40 pin microcontrollers with external code memory. With a change of processor it can be configured as:
POD31S-32	12 MHz POD for 8031/8032
POD31S-C32	12 MHz POD for 80C31/80C32
POD31S-C652	12 MHz POD for 80C652
POD31S-C851	12 MHz POD for 80C851
POD51B	12 MHz POD for 80C51, 80C31, 83/80C652, 83/80C851 40 pin DIP devices
POD451B	12 MHz POD for 83/80C451 68 pin PGA
POD451/64-C451	12 MHz POD for 80C451 64 pin DIP
POD451/68-C451	12 MHz POD for 80C451 68 pin PGA
POD552B	12 MHz POD for 83/80C552 68 pin PGA
POD552-C552	12 MHz POD for 80C552 68 pin PGA
POD552-C562	12 MHz POD for 80C562 68 pin PGA
POD751B	12 MHz POD for 83/87C751 24 pin DIP
POD752B	12 MHz POD for 83/87C752 28 pin PLCC
40DIP/40PLCC	Converter from 40 pin DIP to 44 pin PLCC
68PGA/68PLCC	Converter from 68 pin PGA to 68 pin PLCC
EXPBOX/S	External box with RS232-C interface

Note: Minimum configuration must include a EMUL51-PC/E128 emulator board and one POD board. 16 MHz and 20 MHz parts are also available.

MetaLink System Overview

Functional Description

MicroICE+ is an enhanced MicroICE in-circuit emulator which is designed for use in developing, testing and debugging designs based on 8051 single chip microcontrollers. MicroICE+ allows the development of hardware and software designs to occur simultaneously.

The MicroICE+ emulator assists in the following design phases:

- Software Development
- Manufacturing
- Integration of target software and system hardware
- Field Service

Support

The MicroICE+ emulator not only assists the designer in developing, testing and debugging 8051 microcontroller designs, but is also backed by MetaLink with an extensive service and support policy that includes:

- Updates to the MetaLink User's Manual and MetaLink host operating software are provided at no charge for one (1) year period for registered users.
- Toll free phone number (1-800-METAICE) to resolve system hardware and software concerns.
- A 72-hour repair on disabled systems under warranty.

Features Description

Emulator Functions

- •■ Probe Cards supported:
  The MicroICE+ in-circuit emulator supports a wide variety of interchangeable 8051 Family Probe Cards. See the MetaLink Probe Card Directory for the current list of all Probe Cards available.
- ■ Transparency: Operational and electrical
- ■ Performance: Real-time

User Interface

- •■ Advanced menu or single key command driven system
- ■ 11 functional capabilities:

Configure	Macro	Load
Interrogate	Store	Exit
Download	Upload	Help
OS-Escape	Restore	

Mode (Probe Card Dependent)

- ■ Allows user to emulate all the operating modes of:
  Full operation with external address/data bus
  Single-chip operation with NO external address/data bus
- • Watchdog Timer Enabled/Disabled

Interrogation

- ■ Allows the user to :
  Set complex breaks
  Examine the system status
  Set break and trace triggers
  Examine/modify/store data
- ■ Functional capabilities:
  Run
  Single-step
  Reset
  OS-Escape
  Set pass count
  Set simple break and trace triggers
  Set repetition counter
  Set phantom break and trace triggers
  Set trace triggers (start, end, variable & center)
  Turn trace triggers ON/OFF
  View up to 2I of trace buffer
  Dump trace buffer to file
  Examine/modify:
  Special Function Registers
  Internal data memory
  Program memory
  Emulator complex breaks
- • Execution timer

Complex Breaks

- ■ Used to specify complex break and trace triggers
- ■ Uses the if-then construct
- ■ Allowable trigger conditions are:
  PC address
  PC address range
  Opcode value
  Opcode class
  Special function registers
  Direct byte address
  Direct byte address range
  Direct bit address
  Direct bit address range
  Immediate operand value
  Read/write to bit or direct addresses
  External data address
  External data address range
  Logical AND or OR of any of the above
  Pass count overflow
  External input
- ■ Over: 128,000 break triggers
  64,000 trace triggers
- ■ Editor allows user to create or modify complex breaks by:

Edit	Delete
Compile	Store
Load	Specify Opcode Class

- ■ Opcode class is collection of various 8051 instructions that make up a set.
  Set is user-defined
  Opcode class editor allows user to create, delete or edit Opcode classes

Examine/Modify/Store Memory

- ■ Program memory operations:

Disassemble	Disassemble to file
Disassemble to file	Store revised data
Single line assemble	Mapping
Examine/modify raw data	

- ■ Microcontroller internal and external data memory operations:

Dump	Search
Scan and modify	Compare
Fill	Mapping
Move	Dump to file
Examine/modify addressable bits	

Macro

- ■ Repetitive routine
- ■ User created and callable at any time

Performance Analyzer

- •■ Histogram and statistical information of Real-time Program Execution
- •■ 16 groups specified symbolically, PC address ranges, or automatically

High Level Language

- ■ 'C' and PLM support
- ■ Symbols/Variables and Source Line break/trace triggers
- ■ Variables Accessible symbolically
- ■ Single-Step by Source Line number inter-/intra-module

Symbolic Debug

- ■ User and Pre-Defined Symbols
- ■ Source Level Debug
- ■ Supports MetaLink, Enertec, IAR, Franklin, Archimedes, Sysoft, Microtec Research, Keil, or Intel OMF files
- ■ Use a name not address to alter content of:
  bits, bytes, code

Optional Products

- ■ Power Supply
- ■ Memory Expansion

Electrical Specification

Input Power (typical):
1.5 amps @ + 5 volts DC + / - 5%

Mechanical Specification

Emulator dimensions:
1.0" x 7.0" x 5.5" 2.5cm x 17.8cm

Target system cable length:
14.0" 35.6cm

Emulator and target system cable weight:
2.0 lbs. 0.9 kg

Host Specification

An IBM PC, PC XT, PC AT or 100% compatible system with 640K bytes of RAM.
MS-/PC-DOS 2.0 or later.
One (1) floppy disk and one (1) hard disk drive
One (1) RS232C interface card for the PC and cable.

Warranty

One (1) year limited warranty, parts and labor for registered users.

*** Enhanced Features**

MetaLink Corporation 325 E. Elliot Road,
Chandler, Arizona 85225 (602) 926-0797
TELEX: 4998050 MTLNK
FAX: (602) 926-1198

MetaLink System Overview

MetaLink® Corporation **SYSTEM OVERVIEW**

MicroICE™+ In-Circuit Emulator for Microcontrollers

- Enhanced MicroICE In-Circuit Emulator
- High Speed Serially linked to IBM PC or 100% compatible hosts (up to 57.6K baud)
- Advanced menu or single key command driven human interface
- Real-time and transparent emulation up to 20 MHz
- Interchangeable Probe Cards
- Support both modes:
  - Microcontroller
  - Microprocessor
- Full Symbolic Debug capability
- High Level Language support
- Source Level Debug
- 9 probe clips
  - 7 External Events
  - 1 External Trigger Input
  - 1 External Trigger Output
- Examine/Modify/Store Memory capabilities
- Separate Program and Data Memory Mapping in 16 byte blocks
- Emulation Memory:

	Standard	Optional
-Program	16K	64K
-External Data	16K	64K

- 16 Break and Trace Trigger conditions
- Over 128,000 Break Triggers and 64,000 Trace Triggers
- Up to 64K Pass Counts
- Complex Break Editor/Compiler
- Disassembler and Single Line Assembler
- Trace with 2K frames
- Real-Time Performance Analyzer
- Program Execution Timer (greater than 100 years in duration)

SDS 8051

Stand-alone debug station for 80C51/8051-based systems

THREE CONFIGURATIONS TO SUIT YOUR 8051 PROJECTS

For efficient, accurate software and hardware development and integration, there is no substitute for fully transparent, real-time emulation. But development must be cost-effective, and the key to that is to have *one* emulation unit which can fit exactly into *all* your development projects irrespective of their scope and stage of development.

If you are developing with the 8051 family, such a unit is the SDS 8051.* Three configurations suit all your projects without requiring modifications or extras:

Stand-Alone Operation

A VDU Terminal is all that is needed to integrate and debug with the SDS 8051. And because you can do basic emulation without putting demands on computer resources, the smallest user as well as the big development team can take advantage of the SDS 8051.

*SDS 8051: Generic name for Philips Stand-alone Debugging Station for the 8051 family of microcontrollers; for type numbers, see Ordering Information. The 8051 family includes the 80C51/31, 87C51, 8XC451/552/562/652/654/751/851, the 86C410/610 and the 8051/31/52/32. Debugging stations are also available for Philips 8400 microcontroller family and the 5010/5011 digital signal processors.

Connection to a Host Computer

Most development tasks will, however, need the power of a host computer or Microcomputer Development System, and Philips' SDS (Stand-alone Debugging Station) can work with both. The process of writing the software and designing the hardware can be done on the computer, with an SDS used to debug and integrate the software with the hardware. SDS 8051 supports a wide range of hosts. It has its own in-line assembler in firmware, but you may also use the separate MS-DOS cross-assembler.

Connection to a PC

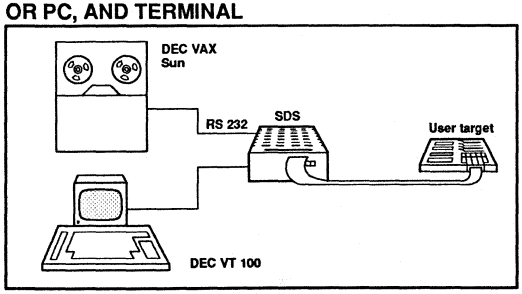
The SDS 8051 operates with an IBM PC or XT, or compatible, e.g., Philips P3100 series, so you can use it in your usual work environment. The SDS can be operated from a PC using readily available terminal emulation software, or the XRAY™51 symbolic debugging package, the latter allowing you to use your own source labels on screen in SDS displays and commands, and having a DOS toggle switch.

If you work in a development team, shared access to files and programs is of course essential to maximize productivity. Programs can be downloaded from any host computer to the SDS memory, using a PC or terminal as a workstation to operate the SDS.

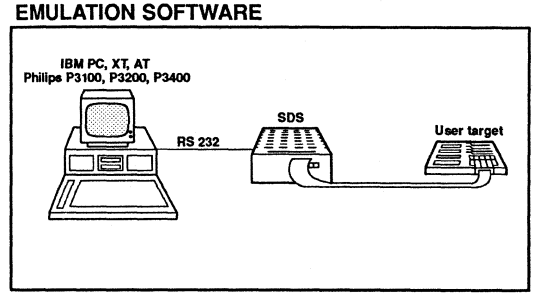
FEATURES

- Real-time, fully transparent emulation
- Works stand-alone from a VDU terminal, PC or other computer
- Full hardware emulation using a dedicated probe—operates according to the exact specifications of the target microcontroller, including maximum speed
- Interfacing signals to external equipment
- Full assembly-level debugging and HLL debugging
- Single-step and breakpoint facilities
- Large trace memory with hardware qualifiers

CONNECTION TO A HOST COMPUTER OR PC, AND TERMINAL



CONNECTION TO A PC WITH TERMINAL EMULATION SOFTWARE



Stand-alone debug station SDS 8051

UNEQUALLED EMULATION

SDS 8051 provides complete, high-quality emulation for development with the 8051 family of microcontrollers. Standard features include:

Fully Transparent Real-Time Emulation

You can develop on the SDS 8051 with the chip running in real time—no stretched clock cycles or wait states to disrupt system timing in the prototype. And emulation is fully transparent—since no resources from the 8051 memory, I/O or interrupt space are used for monitoring emulation, all are available to the target and user program.

In-Circuit Emulation or Simulation

The SDS 8051 is versatile—you can tailor operation to suit the state of the prototype. If no prototype hardware is available, the SDS may be used for simulation, so the software can still be tested. The emulation mode runs the program in the prototype so far as it has been developed, with resources transferred in stages to the prototype. Hardware, software and their integration are fully tested.

Full Real-Time Hardware Emulation and Breakpoint Setting

The SDS 8051 has an 80C51 (or derivative) bond-out chip in the emulation probe. This, and only this, can ensure that hardware emulation is truly real-time, and it also allows you to set hardware breakpoints. These can be set on any combination of addresses, register values or branch instructions, allowing you to investigate program flow in detail and to debug very quickly. When a breakpoint condition is met, the entire CPU is frozen and, besides the full interrupt status, the status of all timers (frozen by a special bond-out chip feature) is displayed.

Alterable Memory and Processor Registers

For really quick fault-finding, the SDS allows you to alter, and to display, memory and CPU register values as required, allowing parts of the program to be repeatedly tested using convenient values.

In addition, the SDS has an emulator-resident in-line assembler which is particularly useful when changing your program—there being no need for repeated up-loading and down-loading.

Disassembly

Of course, there is no need to remember binary code references when developing software with the SDS 8051—instructions are entered in assembly language. In addition, on-board memory can be disassembled into the originally programmed instruction mnemonics.

Trace Memory

SDS 8051 has a 2048-line trace memory for quick checking of the program flow. The display shows the address, opcode/operand, disassembly, instruction status (such as RD OPC, WR) and the contents of microcontroller ports and/or eight user test clips.

Hardware Qualifier Bits

The trace memory has qualifier bits for selecting the information to be captured in the trace memory. These bits enable cycles to be selected, for example, capture only on fetches from emulation memory, or on interrupt acknowledge cycles. Selecting the information before capture overcomes the drawback of software qualifiers which select the information afterwards, relying on luck for it to be in the trace memory!

Debugger Commands

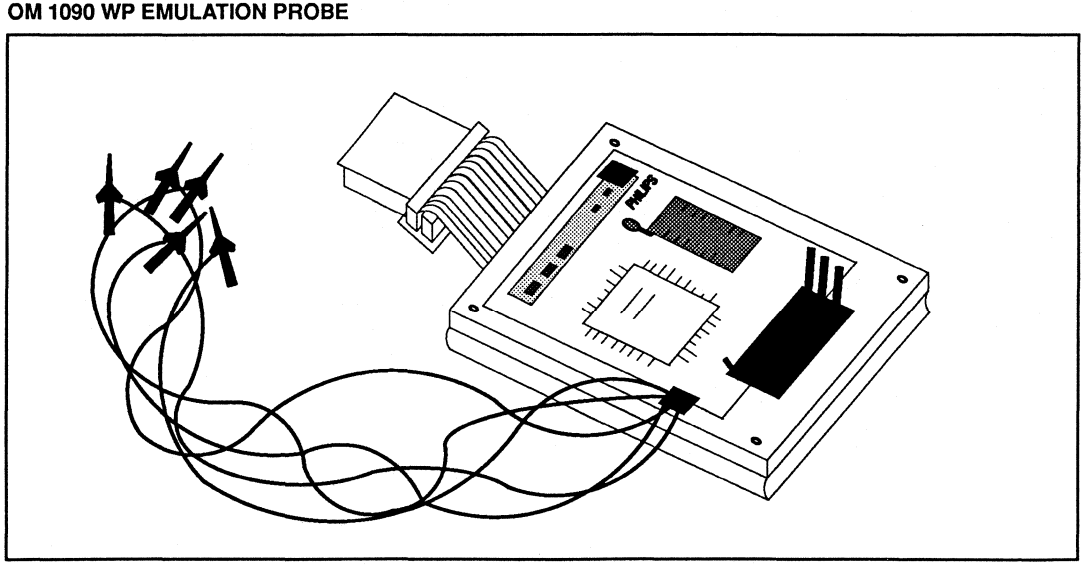
The microprocessor development command language employed by the SDS 8051 has been designed for ease of use. Commands form a subset of, and are similar to, the widely used ICE™ language.

When the XRAY51 symbolic debugging package is used, the commands are the XRAY commands. These and the SDS commands are listed below (at the end of this description).

Software Packages

The XRAY51 high-level debugger, and cross-assemblers and cross-compilers for the SDS 8051 are described on the next page. Each package supports most 8051-derived microcontrollers including the 80C51, 80C451, 80C552, 80C562, 80C652, 80C751, 80C851 and 8XCL410.

OM 1090 WP EMULATION PROBE



Stand-alone debug station SDS 8051

SYMBOLIC DEBUGGING PACKAGE (XRAY51)

The XRAY51 debugger helps to locate programming errors in the source code of PL/M or assembly language programs for the 8051 family. With XRAY51, the user can isolate these errors by controlling and monitoring program execution using the same high-level or assembly level terms, definitions and structures found in the original source program. For example, the user can single-step through the program a specified number of microcontroller instructions or high-level language lines. Variables can be accessed with respect to the source language in which they had been defined. Operating XRAY51 in Assembly mode, the user can manipulate the contents of all processor registers. Only those registers that are part of the selected 8051 derivative are allowed to be accessed.

Macros may be defined that can execute complex user command procedures and provide a variety of complex breakpoints.

When debugging with XRAY51, the user can examine the contents and modify the value of any variable, compute the value of PL/M source language expressions and assembly level address expressions, and define, remove, or display symbols.

Command files can be used to direct XRAY51 to read or write simulated microprocessor input/output from or to a file, allowing easy implementation of automated test sequences. Command files enable scripts of debugger commands to be processed automatically without the need of user interaction.

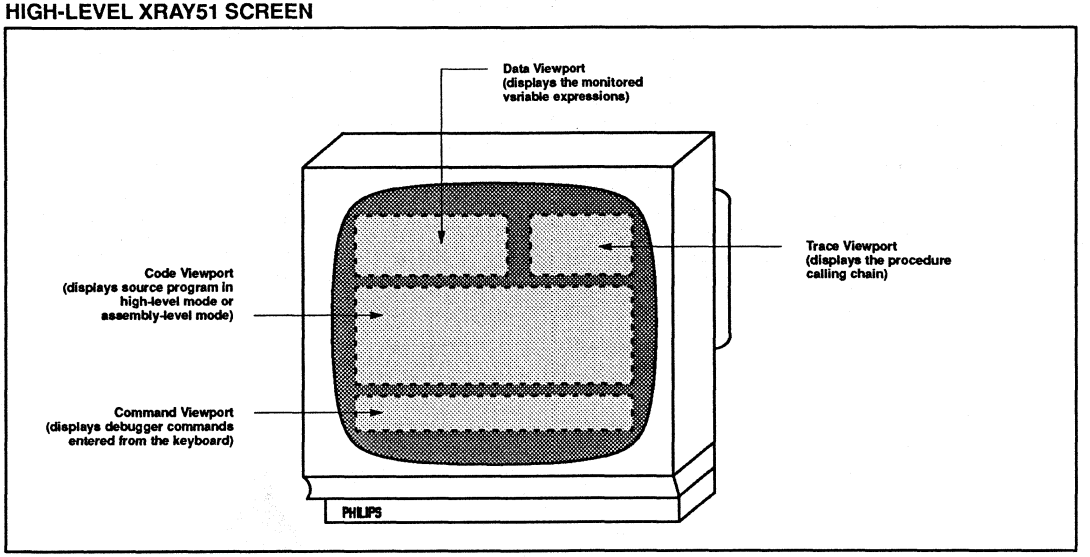
XRAY51 also allows a virtually unlimited number of user-defined windows.

A simulator version of XRAY51 having the same features as the emulator is available.

Features

- Integrated PL/M-51 source-language and assembly-language debugging; toggle by function key at any time
- Window-oriented display with well-organized segregation of debugging information
- Symbolic debugging with PL/M-51 variables and PL/M-51 statements
- Simple and complex breakpoints
- Single-step execution
- User-definable screens and viewports—ability to write selected information
- Command macros
- Breakpoint macros (limited to the number of breakpoints of the SDS 8051)
- Command and breakpoint macros may contain C statements, including: FOR, WHILE DO, PRINTF, and FPRINTF
- On-line context-sensitive help
- Command files
- Output logging
- Fully supports SDS 8051 with the latest firmware version
- High-level trace
- Transparent ICE mode for direct control of SDS 8051

HIGH-LEVEL XRAY51 SCREEN



Stand-alone debug station SDS 8051

CROSS-ASSEMBLERS

Two cross-assemblers* are available for translating 8051 assembly language programs into relocatable object code:

- The TCP (Tasking Compiler Package) ASM51 macroassembler which accepts Intel-compatible assembler source programs and produces relocatable TCP '–.OBJ' object files. An absolute or executable 'A.INT' load-image is obtained using the TCP LINK51 linker.
- The TCP as8051 assembler** which accepts TCP-compatible source language and produces relocatable object files in the TCP '–.o' object format. Here, the required linker is the ld8051 which produces an absolute 'a.out' file whose format is very similar to that of 'A.INT'.

Both assemblers are always supplied together, allowing you to choose the one that suits your requirements best.

Each assembler is a powerful development tool with many directives, so a source file is easily composed, and translation to code is fast.

The assembler directives support features such as the inclusion of ASCII strings, checking module length, forcing a start address, allocating bytes to labels used in current or other modules and module definition.

A very powerful feature is the macroprocessor which includes tools such as macro-substitution, file inclusion and conditional assembly.

TCP ASM51 and TCP as8051 Features

- Produce relocatable object code, listing files and diagnostic messages
- Accept Intel-compatible source programs (TCP ASM51 only)
- Compatible with Tasking's PL/M-51 compilers
- Conversion to IEEE object code format
- Compatible with XRAY51 High-Level/Assembly-Level Debugger
- Include many utilities such as, Librarian, Cross-reference generator, object code-convertors
- Support segment overlay at the Assembly level
- C and MPL (Macro Programming Language) compatible macro preprocessors included
- Separate linking phase

CROSS-COMPILERS

Two cross-compilers* are available for efficiently translating programs written in the PL/M-51 language into 8051 Assembly:

- The TCO PLMTI51 which generates Intel-compatible source, which can be assembled and linked using the TCP ASM51 and LINK51 programs.
- The TCP plm51 which generates Tasking-compatible assembler source, which can be assembled and linked using the TCP as8051, and ld8051 programs.

Both compilers are always supplied together, allowing you to choose the one that suits your requirements best.

TCO plm51 and TCP PLMT151 Features

- Fast, single-pass, memory-based compilers that are fully compatible with the Intel PL/M-51 language definition
- Fully compatible with Tasking's ASM51/LINK51 and as8051/ld8051 cross-assemblers and linkers
- Produce highly optimized code
- Support the XRAY51 high-level debugger
- Support in-line assembly code
- Optional IEEE single-precision floating point package
- Include several utility programs: pr, grep, aar, cpp, scf_i51, ocf_o51, oct_ihex, oct_ieee, oct_srec, ocf_ihex, ocf_srec, makelib and pack
- Utility library included in source code
- Combiner performs optimization at load-file level (ASM51 only)
- Easy tailoring to application and target hardware
- Produce relocatable code and data
- Support optimize(4): intermodule overlay optimization
- Support 15 interrupts
- Automatic installation with self-test
- Available for many hosts besides the IBM PC (MS-DOS), for example: (micro-)VAX, (VMS, Ultrix), HP9000/300 (HP-UX), SUN-3 (Sun-OS); available from Tasking Software B.V.

* Sourced from Tasking Software B.V. Amersfoort, The Netherlands
** Formerly PCP as8051 (Philips' Compiler Package)

Stand-alone debug station SDS 8051

XRAY51 AND SDS COMMANDS

XRAY51 Commands

XRAY51 uses a powerful command language that employs C language expressions. Note, all commands can be issued in an abbreviated form.

Session Control Commands

HOST	Enter the host operating system environment
LOAD	Load an object module for debugging
QUIT	Terminate a debugging session

Execution and Breakpoint Commands

BREAK INSTRUCTION	Set an instruction breakpoint
CLEAR	Clear a breakpoint
GO	Start or continue program execution
GOSTEP	Execute macro after each instruction step
STEP	Execute a specified number of instruction lines
STEPOVER	Step, but execute through procedures

Display Commands

DISASSEMBLE	Display disassembled memory (assembly mode)
DUMP	Display memory contents
EXPAND	Display all local variables of a procedure
FIND	Search for a string
FOPEN	Open a file or device for writing
FPRINTF	Print formatted output to a viewport or file
LIST	Display source code
MONITOR	Monitor variables
NEXT	Find next occurrence of a string
NOMONITOR	Discontinue monitoring variables
PRINTF	Print formatted output to a command viewport
PRINTVALUE	Print the value of a variable

Memory Commands

COMPARE	Compare two blocks of memory
COPY	Copy a memory block
FILL	Fill a memory block with values
SEARCH	Search a memory block for a value
SETMEM	Change the values of memory locations
SETREG	Change the contents of a register
TEST	Examine memory area for invalid values

Port I/O and Interrupt Commands

DIN	Display input port buffer values
DOUT	Display output port buffer values
INPORT	Set or alter input port status
INTERRUPT	Simulate an interrupt
NOINTERRUPT	Cancel pending interrupts
OUTPORT	Set or alter output port status
RIN	Rewind input file associated with input port
ROUT	Rewind output file associated with output port

Symbol Commands

ADD	Create a symbol
DELETE	Delete a symbol from the symbol table
PRINTSYMBOLS	Display symbol, type, and address
SCOPE	Specify current module and procedure scope

Utility Commands

CEXPRESSION	Calculate the value of an expression
ERROR	Set include-file error handling
HELP	Display on-line help screen
INCLUDE	Read in and process a command file
JOURNAL	Record a debugger session in a file
LOG	Record debugger commands and errors in a file
MODE	Select debugger mode (high or assembly)
OPTION	Set debugger options for this session
PAUSE	Pause simulation
RESET	Simulate microprocessor reset
RESTART	Restart program counter to the program starting address
STARTUP	Save the default startup options

Macro Commands

DEFINE	Create a macro
SHOW	Display the macro source

Viewport Commands

VACTIVE	Activate a viewport
VCLEAR	Clear data from a viewport
VCLOSE	Remove a user-defined viewport or screen
VMACRO	Attach a macro to a viewport
VOPEN	Create a screen or viewport or change sizes
VSCREEN	Activate a screen
VSETC	Set the cursor position for a viewport
ZOOM	Increase or decrease the size of a viewport

Function Key Commands

VACTIVE–1	Activate the next viewport (counter clockwise)
VACTIVE+1	Activate the next viewport (clockwise)
MODE	Change debugging mode (assembly/high)
ZOOM	Increase or decrease the size of a viewport
HELP	Access on-line help
VSCREEN	Change the active screen
BACK UP	Back up one command
STEP	Execute one machine instruction or source line
STEP OVER	Step, but execute through procedures

In-Circuit Emulator Commands

ICE	Communicate with in-circuit emulator (ICE)
NOICE	Return to debugger command mode

SDS Commands

Power-Up Commands

RESET	Back to initialization

Program Execution Commands

BRn	Break at given address (n = 0, 1, 2, 3)
BRR	Break within given address range
BRB	Break at branch instruction
BV	Break on internal RAM value
GO (FROM, TILL)	Initiates program execution (specified addresses)
STEP (FROM)	Executes single instruction or instructions (from a specified address)
UD	User-defined memory address on display
TRACE	Display executed instruction flow (real time)
INT	Display interrupt enable, priority and status of all interrupt sources

Memory Access Commands

DBYTE	Displays specified byte from internal data memory
XBYTE	Displays specified byte from external data memory
CBYTE	Displays specified byte from code memory
RBYTE	Displays specified byte from on-chip register memory
RBIT	Displays specified bit from on-chip bit-addressable memory
ASM	Assemble single instruction mnemonic into program memory
DASM	Disassemble memory values into mnemonics

Memory Set Commands

All registers can be set and displayed by commands equal to their names (PSW, SCON, P3, TH1, TL1, TL0, etc.)

Serial I/O Interface Commands

SAVE	Copies data from SDS 8051 program memory to host disk file
LOAD	Transfers file from host to SDS 8051

Stand-alone debug station SDS 8051

SDS 8051 SPECIFICATION

RS232C	– Two ports – Baud rate selectable from 300 to 19200 baud – Download file format: Intel HEX – Recognize Xon/Xoff
Trace memory	– 2048 lines deep, 64 bits wide – Internal/external code memory fetches; data from all ports, lables, user test clips – Selective tracing on cycle type
Emulation memory	– 64 kbytes – No wait states
Clock speed	– Up to 16MHz, real time
Power down	– Supports power-down and idle mode
Signals to external equipment	– Output from SDS: ALE: indicates valid address CLK: indicates opcode read PSENE: indicates byte read EMUL: indicates running user program – Input to SDS: EXTBRK: stop emulation by external pulse
Size	– 300 x 66 x 235 mm (W x H x D)
Weight	– 5 kg (approx.)
Power supply	– 110/220V AC, 50/60 Hz
Cables	– Mains cable; RS232/V24 cable for connection to an IBM PC/XT

ORDERING INFORMATION*

All of the 8051 development tools listed below are available from your local Philips Components sales office. The symbolic debugging package, cross-assembler and compiler are sourced from Tasking Software B.V., Amersfoort, The Netherlands, which holds all intellectual property rights for these products.

Stand-alone debug station (excluding probe) for the 8051 family	OM4120
Symbolic debugging package XRAY51	OM4129
Cross-assembler for MS-DOS (comprises the as8051 and the Intel-compatible ASM51)	OM4142
PL/M-51 compiler for MS-DOS (comprises the plm51 and the Intel-compatible PLMT151)	OM4144
Emulation probes:	
for 8XC556/562 (PLCC interface)	OM1090WP
for 8XC652/654/51:	
(DIL interface)	OM1091P
(PLCC interface)	OM1091WP
for 8XC851 (DIL/PLCC interface)	OM1092
for 8XC751:	
(DIL interface)	OM1094P
(PLCC interface)	OM1094WP
for 8XC451 (PLCC interface)	OM4123
for 8XCL410 (DIL interface)	OM1079

Adapters:**

for 8XC451:	
68-pin PLCC probe to 64-pin DIL socket	OM4124
for 80C51:	
40-pin DIL probe to 44-pin PLCC socket	OM4125

Conversion kits:

for converting the OM1092 to an:	
8XC552/562 probe	OM1095
8XC652/654 probe	OM1096
80C31/80C51 probe	OM1097

TM: ICE is a trademark of Intel Corporation.
XRAY is a trademark of Microtec Research Inc.
MS-DOS is a trademark of Microsoft Corporation.
VAX is a trademark of Digital Equipment Corporation.

* Note: A minimum SDS configuration must include an OM4120 and an emulation probe. A minimum PC configuration must include 256 kbytes system memory running MS-DOS 3.0 (or later releases), one floppy disk drive and a monochrome monitor. However, we recommend using an IBM PC/XT (or compatible) with 640 kbytes RAM, hard disk drive, floppy disk drive and a color monitor. The SDS software can be supplied on $3^1/_2$" or $5^1/_4$" diskettes. Conversion kits require the OM1092.

** Support for QFPs will be available in the near future.

OM4142 (ASM51, as8051)

Cross-assembler package for 80C51/8051-based systems

The package comprises two cross-assemblers for translating 8051 assembly language programs into relocatable object code:

- The ASM51* macroassembler which accepts Intel-compatible assembler source programs and produces relocatable TCP '–.OBJ' object files. An absolute or executable 'A.INT' load-image is obtained using the TCP LINK51 linker.
- The as8051* assembler which accepts TCP-compatible source language and produces relocatable object files in the TCP '–.o' object format. Here, the required linker is the ld8051 which produces an absolute 'a.out' file whose format is very similar to that of 'A.INT'.

Both types of absolute object file can be modified to IEEE format to serve as input to the XRAY51 debugger/simulator.

Both assemblers are always supplied together, allowing the user to choose the one that suits his requirements best. They can be ordered under the Philips type number OM4142.

FEATURES

- Produces relocatable object code, listing files and diagnostic messages
- Accepts Intel-compatible source programs (ASM51)
- Supports most 8051-derived microcontrollers including the 80C51/31, 87C51, 8XC451/552/562/652/654/751/851, 8XCL410 and the 8051/31/52/32.
- Compatible with PL/M-51 compilers
- Conversion to IEEE object code format
- Compatible with XRAY51 High-Level/Assembly-Level Debugger
- Includes many utilities such as, Librarian, Cross-reference generator, object code-convertors
- Supports segment overlay at the Assembly level
- C and MPL (Macro Programming Language) compatible macro preprocessors included
- Separate linking phase.

OPERATION

The input to both assemblers usually comes from a preprocessor which interprets preprocessor directives in the source program to deal with file inclusion, macro definition and replacement, conditional text inclusion, etc. Two preprocessors are available as separate programs, allowing the programmer to use either the C preprocessor directives or Intel's Macro Programming Language, or even a mixture of both.

The output may then be assembled using either the as8051 or ASM51. Depending on the selected member of the 8051 family, the as8051 assembler enables or disables the names of special function registers that are applicable. For the ASM51, this is easily solved by target-dependent inclusion of a specific equate list in the source file.

Both assemblers translate a source program into relocatable object code using three different passes. The program syntax, assembler directives and user-defined symbols are checked and processed during the first pass. In the second pass, all generic forward jumps and calls are optimized. In the third pass, relocatable code is generated.

To obtain a single executable load image, all necessary relocatable objects, including library modules, are linked together using the proper linker (i.e., LINK51 or ld8051). This executable load image may then be converted to an ASCII file that may be downloaded into an EPROM programmer or an emulator.

CHOICE OF ASSEMBLER

Both assemblers are always distributed together—there is no need to specify which you require when ordering. This frees the engineer to select the one that suits his environment and requirements best, without breaking existing code or scrapping existing development tools. Those who are already familiar with the Intel ASM51 will probably stick to the ASM51, while existing Philips (PCP) users might prefer the as8051.

Both assemblers are capable of generating symbolic debug information to accommodate information for the XRAY debugger/simulator. And although their formats differ, there is no functional difference between the object code produced by ASM51 and by as8051.

* Sourced from Tasking Software B.V., Amersfoort, The Netherlands, which holds all intellectual property rights for this software. The as8051 was formerly the PCPas8051 (Philips' Compiler Package).

Cross-assembler package for 80C51/8051-based systems

OM4142 (ASM51, as8051)

ASSEMBLER LIMITATIONS

The number of user-defined symbols and macro parameters is limited only by the available heap space. Save/restore nesting is restricted to 16 levels.

DIFFERENCES BETWEEN PHILIPS' ASM51* AND INTEL'S ASM51

Unlike Intel's ASM51, which restricts the use of register equates to the A and R0–R7 registers, Philips' ASM51 puts no constraints on register name assignment. And it can optimize generic JMP and CALL instructions, even when they contain a forward reference. In such cases, Intel's ASM51 will always produce code for a LJMP and LCALL, which takes 50% more code.

Philips' ASM51 supports four new directives that are not available in the Intel ASM51:

- $listall — generate a listfile in every pass (not only in the final one). This improves diagnostics.
- $(no-)optimize — enable (default) or disable generic JMP/CALL optimization.
- $debuginfo — control symbolic debug information generation.

Since the $gen, $genonly, $nogen, $include and $macro directives are already dealt with during the preprocessing stage, these can be ignored by Philips' ASM51.

Similarly, $(no-)xref and $(no-)symbols directives are ignored—report utilities are available to accomplish the same task. And the $workfiles is no longer useful and is ignored too.

Philips' ASM51 recognizes ?SYMB, ?LINE and ?FILE symbols that are used to pass debug information towards the object module. Philips' ASM51 recognizes the C-like #line directive to adjust the line number and file name. This directive is generated by both preprocessors to synchronize the output line with the original input. This improves the error diagnostics of Philips' ASM51 compared with those of its competitors, since error messages now refer to the proper (include) file and line number.

A powerful addition is the overlay() which gives the programmer full control of the section overlay strategy.

UTILITY PACKAGE

Several utility programs are included for your convenience. Amongst these are a C preprocessor (cpp) and an Intel MPL-compatible preprocessor (mpl) to deal with the various preprocessor directives and macros.

In many cases, the format of the object code produced is not suitable for EPROM-programmers, emulators or debuggers. Therefore, several utilities are included to convert the code to Intel HEX (oct_ihex), Motorola S0–S9 (oct_srec) or IEEE-695 (oct_ieee) or vice-versa (ocf_ihex, ocf_srec).

An optional utility package is available separately from Tasking Software B.V. It contains UNIX-like utilities to obtain a (cross-reference) list of all user-defined symbols in a program ('axref' and 'anm'), and the 'asize' utility to get information about the section sizes.

HARDWARE/SOFTWARE REQUIREMENTS AND INSTALLATION

OM4142 software comes to you on $5^1/_4$" diskettes ($3^1/_2$" diskettes are available on request) together with extensive documentation, including two Assembler Reference Guides, Preprocessor Manuals, Utility and Installation Guide. The software requires an MS-DOS computer with a hard disk and at least 512k byte RAM installed. Software installation is simple, well-documented and can be verified afterwards using an automatic verification program.

RELATED PRODUCTS

The OM4142 assemblers are part of a complete programming and development package for the 8051 family of microcontrollers. For both assemblers, there is an excellent PL/M-51 compiler available from Philips offering the convenience and benefits of high-level language programming, resulting in a dramatic increase of programmer productivity. Testing and debugging can be accelerated using the High-Level/Assembly-Level XRAY51 debugger/simulator. Contact your local Philips Components software sales office for more information. Ask for the following leaflets:

Symbolic debugging package XRAY51 for the SDS 8051 emulator, ordering code 9398 366 10011;

PL/M-51 compiler package for 80C51/8051-based systems, ordering code 9398 366 20011;

Stand-alone debug station for 80C51/8051-based systems, ordering code 9398 366 00011.

SOFTWARE MAINTENANCE AND SUPPORT

After a 90-day warranty period, in which all support will be given free of charge, a software update and support agreement can be taken out with Philips for a modest annual fee. Philips realizes your need for fast and comprehensive support, and with a support license, you get direct access to a development team that can solve your problems.

ORDERING INFORMATION

TYPE NUMBER	DESCRIPTION
OM4142	8051 cross-assembler package comprising the as8051 and ASM51 assemblers, and the ld8051 and LINK51 linkers; MS-DOS
Related Products	
OM4144	PL/M-51 compiler package comprising the plm51 and the PLMTI1 compilers
OM4129	XRAY51 high-level debugger for the SDS 8051

Orders can be placed via your local Philips Components sales representative.

TM: Intel is a trademark of Intel Corporation.
XRAY is a trademark of Microtec Research Inc.
MS-DOS is a trademark of Microsoft Corporation.
UNIX is a trademark of AT&T Bell Laboratories..

OM4144 (plm51, PLMTI51)

PL/M-51 compiler package for 80C51/8051-based systems

The package comprises two cross-compilers (plm51* and PLMTI51*) which efficiently translate programs written in the PL/M-51 language into 8051 Assembly. The plm51 generates Philips-compatible assembler source, which can be assembled and linked using Philips' as8051, and ld8051 programs. The PLMTI51 generates Intel-compatible source, which can be assembled and linked using Philips' ASM51 and LINK51 programs.

The compilers embody the latest techniques. Each is built upon a YACC-based parser. Intermediate code is built using tree structures that are optimized into new trees by the compiler before code-generation starts. An internal peephole optimizer further improves the density of the generated code.

Both compilers are always supplied together, allowing the user to choose the one that suits his requirements best. They can be ordered under the Philips' type number OM4144.

FEATURES

- State-of-the-art compiler: fast, single-pass, memory-based
- Fully compatible with the Intel PL/M-51 language definition
- Fully compatible with Philips' ASM51/LINK51 and as8051/ld8051 cross-assemblers and linkers
- Produces highly optimized code
- Supports most 8051-derived microcontrollers, including the 80C51/31, 87C51, 8XC451/552/562/652/654/751/851, 8XCL410 and the 8051/31/52/32
- Both compilers support the XRAY51 high-level language debugger, available from Philips
- Supports in-line assembly code
- Optional IEEE single-precision floating point package
- Includes several utility programs: pr, grep, aar, cpp, scf_i51, ocf_o51, oct_ihex, oct_ieee, oct_srec, ocf_ihex, ocf_srec, makelib and pack
- Utility library included in source code
- Combiner performs optimization at load-file level (Philips' ASM51 only)
- Easy tailoring to application and target hardware
- Produces relocatable code and data
- Available for many hosts besides the IBM PC (MS-DOS), for example: (micro-)VAX, (VMS, Ultrix), HP9000/300 (HP-UX), SUN-3 (Sun-OS); available from Tasking Software B.V.
- Supports optimize(4): intermodule overlay optimization
- Supports 15 interrupts
- Automatic installation with self-test
- Benchmarks available on request

PL/M-51

PL/M-51 is a structured, high-level programming language derived from the Intel PL/M-80 language and adapted to the specific capabilities of the 8051 family of single-chip microcontrollers. It supports Boolean processing and allows efficient access to all microcontroller hardware functions.

Software development in PL/M-51 combines the ease of programming in a high-level language with access to all of the 8051 I/O and memory—features that are normally only available to assembly language programmers.

The Philips' implementation of PL/M-51 is an extremely fast single-pass optimizing compiler that is fully compatible with the Intel PL/M-51 language definition.

* Sourced from Tasking Software B.V., Amersfoort, The Netherlands, which holds all intellectual property rights for this software.

PL/M-51 compiler package for 80C51/8051-based systems

OM4144 (plm51, PLMTI51)

IMPLEMENTATION-DEPENDENT DATA

Data Types

- Data types BIT, BYTE and WORD are allowed for variables, arrays, structures or combinations thereof.
- Memory types MAIN, IDATA, REGISTER, AUXILIARY and CONSTANT are supported.
- Data can be stored at fixed locations using AT, or stored dynamically using BASED pointer variables.

Procedures

- BIT, BYTE or WORD typed procedures, returning a value upon completion
- Untyped procedures, invoked with a CALL statement
- Optional procedure attributes:
  - USING(n), specifies the register bank (n) to be used by the procedure.
  - INTERRUPT(n), defines an interrupt procedure for interrupt (n).

Statements

- Control statements: do while ... end, do ... to ... by ... end
- Conditional statements: if ... then ... else ...
- Miscellaneous statements: do ... end, do case ... end, call, goto, enable/disable

Library Routines

Internal (Built-In) Procedures:

LENGTH	PROPAGATE
WORD	ROL
EXPAND	SCR
SHR	SIZE
SCL	BOOLEAN
TIME	SHL
LAST	ROR
DOUBLE	TESTCLEAR

PLM51.LIB:
Library routines used by the compiler.

UTIL51.LIB or util51.oa, util51:
Assembler utility libraries for both compiler versions, consisting of procedures for string manipulation. It contains the routines MOV, RMV, CMP, FNDB, FNDW, SKPB, SKPW, SETB and SETW for each memory type and for each register bank.

Re-entrancy

The generated code is not re-entrant, because PL/M-51 is not defined as a language with re-entrant procedures. Because of the limited size of internal RAM, the local and formal parameters of procedures are not put on the stack, but on static areas, which can be overlayed by the compiler using $OPTIMIZE(3).

Compiler Controls

ROM(s), REGISTERBANK(n), OPTIMIZE(n), (NO)INTVECTOR, (END)ASM, INCLUDE, SAVE/RESTORE, (NO)LIST, (NO)SOURCE, (NO)CODE, (NO)DEBUG, (NO)OBJECT, SET/RESET, EJECT and IF/ELSIF/ELSE/ENDIF

Differences Between Philips' and Intel's PL/M-51

- Philips' PL/M-51 allows bit-structures to be ATed at byte-variables in bit-addressable memory
- Philips' PL/M-51 supports floating point
- Invocation and compiler controls are mostly different
- Object modules are not generated
- Error messages are mostly different
- Different assembly language interfacing (only for Philips' plm51)
- Compiler limits always the same or better than Intel's

Code Optimization

Five levels of optimization are supported:

- Level 0: performs folding of constant expressions and of address calculations, in case a constant offset exists.
- Level 1: performs all of level 0 plus elimination of unreachable code, strength reduction of expressions and partial condition evaluation at runtime.
- Level 2: performs all of levels 0 and 1 plus machine code (peephole) optimizations and register history.
- Level 3: performs all of levels 0, 1, and 2 plus automatic overlaying of on-chip RAM variables.
- Level 4: performs all of the other levels plus support of intermodule overlay of on-chip data.

General Optimizations

- Store-copy optimization
- Local constant propagation
- Register allocation
- Peephole optimization
- Dead code elimination
- Constant folding
- Index simplification

Object-code Optimizations

- Branch optimization (sjmp, ajmp, ljmp, acall, lcall)
- Effective address optimization

8051-Specific Optimizations

- Optimal use of the range of address modes of the 8051 family
- Overlay of local data and formal parameters done by $OPTIMIZE(3)/(4)

Easy Adaptation to Target Environment

Cross-compilers are used to develop embedded microprocessor applications, where the hardware environment in which they will run is not fixed beforehand. Adaptation to the target hardware environment takes place via the files headx. These files are used to define all system-dependent set-ups such as the power-on-restart vector, the initial stackpointer value, the definitions of segments needed by the PL/M-51 code and the mapping of those segments to the physical addresses. The files can be modified by the user to match his specific needs.

The PL/M-51 cross-compiler can accommodate different target environments, because:

- The location of the stack is held in a special target set-up file (head_xx file) which can be changed to match your requirements
- This target set-up file also allows general housekeeping tasks to be executed before program start-up
- Simple interfacing to target operating systems and low-level I/O and system routines
- Code and data segments can be placed anywhere in the 64kbyte data and address space of the 8051 processor
- Efficient calls to library routines can be in PL/M or assembler (in-line)
- A user-written interrupt handler can be made by specifying $NOINTVECTOR
- The generated code is ROMable.

Restrictions

The PL/M-51 compiler has a few restrictions listed here for completeness:

- Nesting of all LITERALLY invocations: 8
- Nesting of INCLUDE controls: 8
- Nesting of blocks: 32
- Number of elements in a factored list: no limit
- Number of characters in an input line: 160
- Number of switch names (conditional compilation): 20

PL/M-51 compiler package for 80C51/8051-based systems

OM4144 (plm51, PLMTI51)

- Length of a string constant: 254
- Number of cases in a DO CASE block: 84
- Number of EXTERNAL items: no limit
- Number of non-EXTERNAL procedures in module: no limit
- Number of names in a module: memory dependent

UTILITY PACKAGE

Several utility programs are included for your convenience. Amongst these are a C preprocessor (cpp), an Intel MPL compatible preprocessor (mpl)—both part of the 8051 Cross-Assembler—to deal with the various preprocessor directives and macros. Pagination and pattern search commands 'pr' and 'grep' come with the PL/M-51 compiler package.

In many cases, the format of the produced object code is not suitable for EPROM-programmers, emulators or debuggers. Therefore, several utilities are included (in the 8051 cross-assembler) to convert the code to Intel HEX (oct_ihex), Motorola S0–S9 (oct_srec) or IEEE-695 (oct_ieee) or vice-versa (ocf_ihex, ocf-srec) or to archive object modules (aar).

OPTIONAL SOFTWARE

An optional utility package is available separately from Tasking Software B.V. It contains UNIX-like utilities to obtain a (cross-reference) list of all user-defined symbols in a program ('axref' and 'anm'). Part of this package is the 'asize' utility to get information about the section sizes, 'asort' to sort or merge files, 'astrip' to remove symbols and relocation information, 'adump' to display the contents of an object file.

Also available is an IEEE floating-point library for the 8051 family.

HARDWARE/SOFTWARE REQUIREMENTS AND INSTALLATION

OM4144 software comes to you on $5^1/_4$" diskettes ($3^1/_2$" diskettes are available on request) together with the extensive *PL/M-51 Application Manual*.

The software requires an MS-DOS (Rel. 3.0 or higher) computer with a hard disk and at least 512kbyte RAM installed. Software installation is simple, well-documented and can be verified afterwards using an automatic verification program.

RELATED PRODUCTS

The OM4144 compilers are part of a complete programming and development package for the 8051 family of microcontrollers, a package which includes cross-assemblers and (optimizing) linkers. Testing and debugging can be accelerated using the high-level/assembly-level XRAY-51 debugger in combination with Philips' SDS 8051 emulator, or the 8051 Simulator (available from Tasking software B.V.). Contact your local Philips Components sales office for more information.

Ask for the following leaflets:

Symbolic debugging package XRAY51 for the SDS 8051 emulator, ordering code 9398 366 10011;

Cross-assembler package for 80C51/8051-based systems, ordering code 9398 366 30011;

Stand-alone debug station for 80C51/8051-based systems, ordering code 9398 366 00011.

More information about the PL/M-51 Language can be found in the *PL/M-51 User's Guide*, available from Intel (ordering code: 121966-003).

SOFTWARE MAINTENANCE AND SUPPORT

After a 90-day warranty period, in which all support will be given free of charge, a software update and support agreement can be taken out with Philips for a modest annual fee. Philips realizes your need for fast and comprehensive support, and with a support license, you get direct access to a development team that can solve your problems.

ORDERING INFORMATION

TYPE NUMBER	DESCRIPTION
OM4144	PL/M-51 compiler package comprising the plm51 and the PLMTI1 compilers; MS-DOS
Related Products	
OM4142	8051 cross-assembler package comprising the as8051 and ASM51 assemblers, and the ld8051 and LINK51 linkers
OM4129	XRAY51 high-level language debugger for the SDS 8051

Orders can be placed via your local Philips Components sales representative.

TM: Intel is a trademark of Intel Corporation.
MS-DOS and XENIX are trademarks of Microsoft Corporation.
XRAY is a trademark of Microtec Research Inc.
UNIX is a trademark of AT&T Bell Laboratories.
VAX, microVAX, VMS and Ultrix are trademarks of Digital Equipment Corp.
HP9000 and HP-UX are trademarks of Hewlett Packard Co.
SUN-3 and SunOS are trademarks of Sun Microsystems Inc.
IBM PC is a trademark of International Business Machines Corp.

OM4129

Symbolic debugging package XRAY51 for the SDS 8051 emulator

The XRAY™51 package enables the software developer to monitor and control the execution of a target program using the same high-level or assembly-level terms, definitions and structures found in the original source program. It employs a window-oriented user interface which segregates the debugging information into meaningful areas. Two versions of XRAY51, each with identical user interface, are available:

- An emulator version* using Philips' SDS 8051 emulation hardware as engine
- A simulator version* using a software engine.

The emulator (debugger) facilitates debugging in real-time on a real target; the simulator supports debugging in an 8051 environment, simulated in software on the host system. Both versions control the execution of the program and allow the user to stop, start, and examine the target software by means of a powerful command language that employs C language expressions.

XRAY51 HIGH-LEVEL DEBUGGER

The XRAY51 debugger helps to locate programming errors in the source code of PL/M or assembly language programs for the 8051 family of microcontrollers. The user can isolate these errors by controlling and monitoring program execution. For example, the user can single-step through the program a specified number of microcontroller instructions or high-level language lines. Variables can be accessed with respect to the source language in which they had been defined (i.e., PL/M or Assembly). Operating XRAY51 in Assembly mode, the user can manipulate the contents of all processor registers. Only those registers that are part of the selected 8051 derivative are allowed to be accessed.

Macros may be defined that can execute complex user command procedures and provide a variety of complex breakpoints.

When debugging with XRAY51, the user can examine the contents and modify the value of any variable, compute the value of PL/M source language expressions and assembly level address expressions, and define, remove, or display symbols.

Command files can be used to direct XRAY51 to read or write simulated microprocessor input/output from or to a file, allowing easy implementation of automated test sequences. Command files enable scripts of debugger commands to be processed automatically without the need of user interaction.

XRAY51 also allows a virtually unlimited number of user-defined windows.

FEATURES

The XRAY51 debugger has been designed with the developer of embedded applications in mind. Its principal features are:

- Integrated PL/M-51 source-language and assembly-language debugging; toggle by function key at any time
- Window-oriented display which segregates debugging information into meaningful areas
- Symbolic debugging with PL/M-51 variables and PL/M-51 statements
- Simple and complex breakpoints
- Single-step execution
- User-definable screens and viewports—ability to write selected information
- Command macros
- Breakpoint macros (limited to the number of breakpoints of the SDS 8051)
- Command breakpoint macros may contain C statements, including: FOR, WHILE, DO, PRINTF, and FPRINTF
- On-line context-sensitive help
- Command files
- Output logging
- Fully supports SDS 8051 with the latest firmware version
- High-level trace
- Transparent ICE mode for direct control of SDS 8051
- Supports most 8051-derived microcontrollers, including the 80C51/31, 87C51, 8XC451/552/562/652/654/751/851, 8XCL410 and the 8051/31/52/32

™ XRAY is a trademark of Microtec Research Inc., which holds all intellectual property rights for XRAY51.

* The SDS 8051 emulator version of XRAY51 is available from Philips Components (type number OM4129) and is sourced from Tasking Software B.V., Amersfoort, The Netherlands.
The simulator version is available from Tasking Software B.V.

Symbolic debugging package XRAY51 for the SDS 8051 emulator

OM4129

COMMANDS

Note, all commands can be issued in an abbreviated form.

Session Control Commands

HOST	Enter the host operating system environment
LOAD	Load an object module for debugging
QUIT	Terminate a debugging session

Execution and Breakpoint Commands

BREAKINSTRUCTION	Set an instruction breakpoint
CLEAR	Clear a breakpoint
GO	Start or continue program execution
GOSTEP	Execute macro after each instruction step
STEP	Execute a specified number of instruction lines
STEPOVER	Step, but execute through procedures

Display Commands

DISASSEMBLE	Display disassembled memory (assembly mode)
DUMP	Display memory contents
EXPAND	Display all local variables of a procedure
FIND	Search for a string
FOPEN	Open a file or device for writing
FPRINTF	Print formatted output to a viewport or file
LIST	Display source code
MONITOR	Monitor variables
NEXT	Find next occurrence of a string
NOMONITOR	Discontinue monitoring variables
PRINTF	Print formatted output to command viewport
PRINTVALUE	Print the value of a variable

Memory Commands

COMPARE	Compare two blocks of memory
COPY	Copy a memory block
FILL	Fill a memory block with values
SEARCH	Search a memory block for a value
SETMEM	Change the values of memory locations
SETREG	Change the contents of a register
TEST	Examine memory area for invalid values

Port I/O and Interrupt Commands

DIN	Display input port buffer values
DOUT	Display output port buffer values
INPORT	Set or alter input port status
INTERRUPT	Simulate an interrupt
NOINTERRUPT	Cancel pending interrupts
OUTPORT	Set or alter output port status
RIN	Rewind input file associated with input port
ROUT	Rewind output file associated with output port

Symbol Commands

ADD	Create a symbol
DELETE	Delete a symbol from the symbol table
PRINTSYMBOLS	Display symbol, type, and address
SCOPE	Specify current module and procedure scope

Utility Commands

CEXPRESSION	Calculate the value of an expression
ERROR	Set include-file error handling
HELP	Display On-line help screen
INCLUDE	Read in and process a command file
JOURNAL	Record a debugger session in a file
LOG	Record debugger commands and errors in a file
MODE	Select debugger mode (high or assembly)
OPTION	Set debugger options for this session
PAUSE	Pause simulation
RESET	Simulate microprocessor reset
RESTART	Restart program counter to the program starting address
STARTUP	Save the default start-up options

Macro Commands

DEFINE	Create a macro
SHOW	Display the macro source

Viewport Commands

VACTIVE	Activate a viewport
VCLEAR	Clear data from a viewport
VCLOSE	Remove a user-defined viewport or screen
VMACRO	Attach a macro to a viewport
VOPEN	Create a screen or viewport or change sizes
VSCREEN	Activate a screen
VSETC	Set the cursor position for a viewport
ZOOM	Increase or decrease the size of a viewport

Function Key Commands

VACTIVE–1	Activate the next viewport (counter clockwise)
VACTIVE+1	Activate the next viewport (clockwise)
MODE	Change debugging mode (assembly/high)
ZOOM	Increase or decrease the size of a viewport
HELP	Access on-line help
VSCREEN	Change the active screen
BACK UP	Back up one command
STEP	Execute one machine instruction or source line
STEP OVER	Step, but execute through procedures

In-Circuit Emulator Commands

ICE	Communicate with in-circuit emulator (ICE)
NOICE	Return to debugger command mode

RESTRICTIONS

- Up to 4 instruction breakpoints are allowed simultaneously.
- When an instruction has a breakpoint set, control is returned to XRAY *after* this instruction has been executed. However, the information about the break address returned is correct, so user actions (like macros) are executed as expected.
- In the current release, the TRACE capability of the SDS 8051 is supported by means of the ICE/NOICE command. The SDS 8051 TRACE information can be recorded in a journal file, by means of the JOURNAL command. The next release of XRAY51 will support emulator tracing within the debugger.

Symbolic debugging package XRAY51 for the SDS 8051 emulator

OM4129

HARDWARE/SOFTWARE REQUIREMENTS

XRAY51 software is supplied on $5^1/_4$' diskettes ($3^1/_2$' diskettes are available on request) together with the documentation: the *XRAY51 User's Guide*, *XRAY51 Installation Guide* and the *XRAY51 Reference Manual*. XRAY51 will run on an MS-DOS computer equipped with a hard disk and at least 512kbyte RAM.

To work with XRAY51,you will also need:

- An as8051 relocatable cross-assembler and ld8051 linking loader (version 3.0, or later)

or

- An ASM51 Intel-compatible relocatable cross-assembler and LINK51 Intel-compatible linking loader (version 1.0 or later).

If you want to use the high-level language capabilities of XRAY51, one of the following PL/M-51 compilers is required:

- plm51 (version 2.0 or later) with the as8051 assembler
- PLMTI51 (version 2.0 or later) with the ASM51 assembler

These products can be ordered from Philips Components. See Ordering Information.

Contact your local Philips representative for specific enquiries. Ask for the following leaflets:

Stand-alone debug station for 80C51/8051-based systems, ordering code 9398 366 00011;

PL/M-51 compiler package for 80C51/8051-based systems, ordering code 9398 366 20011;

Cross-assembler package for 80C51/8051-based systems, ordering code 9393 366 30011.

SOFTWARE MAINTENANCE AND SUPPORT

After a 90-day warranty period, in which all support will be given free of charge, a software update and support agreement can be taken out with Philips for a modest annual fee. Philips realizes your need for fast and comprehensive support, and with a support license, you get direct access to a development team that can solve your problems.

ORDERING INFORMATION

TYPE NUMBER	DESCRIPTION
OM4129	XRAY51 high-level language debugger for the SDS 8051; MS-DOS
Related Products	
OM4142	Cross-assembler package comprising the as8051 and the ASM51 assemblers, and the ld8051 and LINK51 linkers
OM4144	Compiler package comprising the plm51 and the PLMTI1

Orders can be placed via your local Philips Components sales representative.

TM: Intel is a trademark of Intel Corporation.
MS-DOS is a trademark of Microsoft Corporation.

Section 6
Additional Microcontroller Group Data Sheets

CONTENTS

SCN8049 Series Microcontrollers 577
8X305 Microcontroller 592
8X401 Microcontroller 617
82C200 Stand-alone CAN Controller 637

Date of Issue	August 26, 1986
Status	Product Specification
Application Specific Product	

SCN8049 Series

SCN8049, SCN8050, SCN8039, SCN8040 single-chip 8-bit microcontroller

DESCRIPTION

The SCN8049 Series Microcontrollers are self-contained, 8-bit processors which contain the system timing, control logic, RAM data memory, ROM program memory (8048/49/50 only), and I/O lines necessary to implement dedicated control functions. All SCN8049 Series devices are pin and program compatible, differing only in the size of the on-board program ROM and data RAM, as follows:

TYPE	RAM SIZE	ROM SIZE
SCN8049	128 x 8	2k x 8
SCN8050	256 x 8	4k x 8
SCN8039	128 x 8	–
SCN8040	256 x 8	–

Program memory can be expanded externally up to a maximum total of 4k bytes without paging. Data memory can also be expanded externally. I/O capabilities can be expanded using standard devices or the 8243 I/O expander.

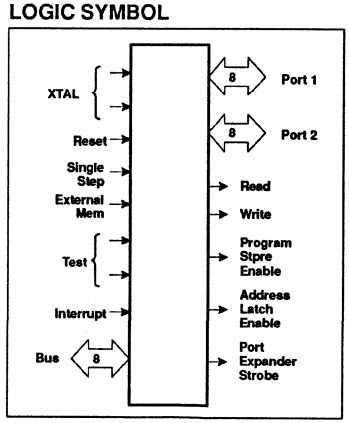
The SCN8049 Series processors are designed to be efficient control processors as well as arithmetic processors. They provide an instruction set which allows the user to directly set and reset individual lines within its I/O ports as well as test individual bits within the accumulator. A large variety of branch and table look-up instructions make these processors very efficient in implementing standard logic functions. Also, special attention has been given to code efficiency. Over 70% of the instructions are a single byte long and all others are only 2 bytes long.

An on-chip 8-bit counter is provided which can count, under program control, either internal clock pulses (with a divide by 32 prescaler) or external events. The counter can be programmed to cause an interrupt on terminal count.

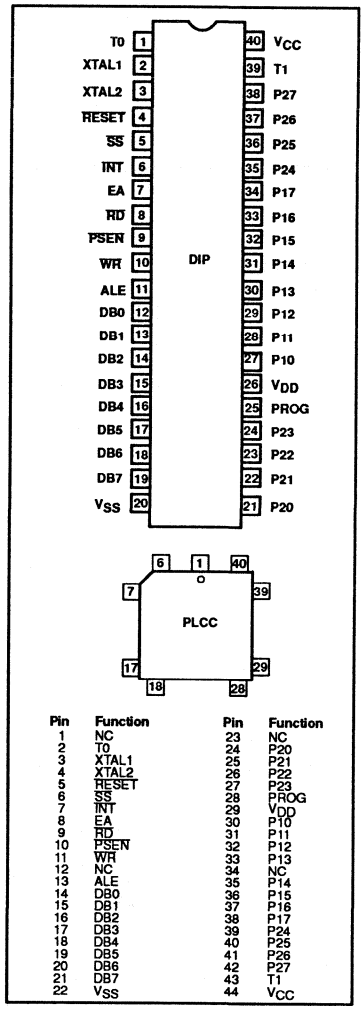
FEATURES

- 8-bit CPU, ROM, RAM, I/O in a 40-pin package
- 24 quasi-bidirectional I/O lines
- Two test inputs
- Internal counter/timer
- Single-level vectored interrupts: external, counter/timer
- Over 90 instructions, 70% single byte
- 1.36µs or 2.5µs instruction cycle, all instructions one or two cycles
- Expandable memory and I/O
- Low voltage standby
- TTL compatible inputs and outputs
- Single +5V power supply

LOGIC SYMBOL



PIN CONFIGURATIONS

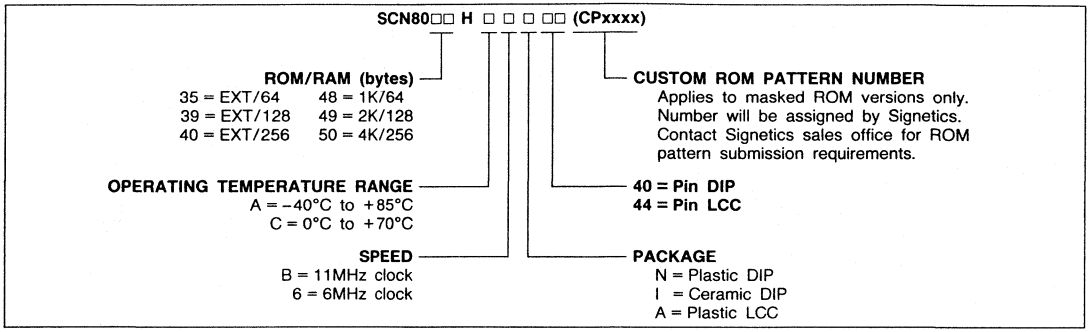


Pin	Function	Pin	Function
1	NC	23	NC
2	T0	24	P20
3	XTAL1	25	P21
4	XTAL2	26	P22
5	RESET	27	P23
6	SS	28	PROG
7	INT	29	VDD
8	EA	30	P10
9	RD	31	P11
10	PSEN	32	P12
11	WR	33	P13
12	NC	34	NC
13	ALE	35	P14
14	DB0	36	P15
15	DB1	37	P16
16	DB2	38	P17
17	DB3	39	P24
18	DB4	40	P25
19	DB5	41	P26
20	DB6	42	P27
21	DB7	43	T1
22	VSS	44	VCC

SCN8049, SCN8050, SCN8039, SCN8040 Single-chip 8-bit microcontroller

SCN8049 Series

ORDERING INFORMATION



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. PLCC	TYPE	NAME AND FUNCTION
V_{SS}	20	22		Circuit ground potential.
V_{DD}	26	29		Low power standby.
V_{CC}	40	44		**Main Power Supply:** +5V during operation.
PROG	25	28	O	Output strobe for 8243 I/O expander.
P10 – P17	27 – 34	30 – 33, 35 – 38	I/O	**Port 1:** 8-bit quasi-bidirectional port.
P20 – P27	21 – 24, 35 – 38	24 – 27, 39 – 42	I/O	**Port 2:** 8-bit quasi-bidirectional port. P20-23 contain the four high-order program counter bits during an external program memory fetch and serve as a 4-bit I/O expander bus for 8243.
DB0 – DB7	12 – 19	14 – 21	I/O	**Data Bus:** True bidirectional port which can be written or read synchronously using the $\overline{RD}$, $\overline{WR}$ strobes. The port can also be statically latched. Contains the eight low-order program counter bits during an external program memory fetch and receives the addressed instruction under the control of $\overline{PSEN}$. Also contains the address and data during an external RAM data store instruction, under control of ALE, $\overline{RD}$ and $\overline{WR}$.
T0	1	2	I	Input pin testable using the conditional transfer instructions JT0 and JNT0. T0 and be designated as a clock output using the ENT0 CLK instruction.
T1	39	43	I	Input pin testable using the JT1 and JNT1 instructions. Can be designated the timer/counter input using the STRT CNT instruction.
XTAL1	2	3	I	**Crystal 1:** One side of the crystal input for internal oscillator. Also input for external source (non-TTL V_{IH}).
XTAL2	3	4	I	**Crystal 2:** Other side of crystal input.
$\overline{INT}$	6	7	I	**Interrupt:** Initiates an interrupt if interrupt is enabled. Interrupt is disabled after a reset. Also testable with conditional jump instruction. Interrupt must remain low for at least three machine cycles for proper operation.
$\overline{RESET}$	4	5	I	**Reset:** Used to initialize the microcomputer. Active low. Internal pullup ~75KΩ. During program verification the address is latched by a "0" to "1" transition on $\overline{RESET}$ and the data at the addressed location is output on BUS.
$\overline{RD}$	8	9	O	**Read:** Output strobe activated during a bus read. Can be used to enable data onto the bus from an external device. Used as a read strobe to external data memory.
$\overline{WR}$	10	11	O	**Write:** Output strobe during a bus write. Used as write strobe to external data memory.
ALE	11	13	O	**Address Latch Enable:** Occurs once during each cycle and is useful as a clock output. The negative edge of ALE strobes address into external data and program memory.
$\overline{PSEN}$	9	10	O	**Program Store Enable:** Output occurs only during a fetch to external program memory.
$\overline{SS}$	5	6	I	**Single Step:** Can be used in conjunction with ALE to "single step" the processor through each instruction.
EA	7	8	I	**External Access:** Forces all program memory fetches to reference external memory. Useful for emulation and debug, and essential for testing and program verification.

NOTE:
Each pin on these ports can be assigned, under program control, to be an input or an output. A pin is designated as an input by writing a logic "1" to the pin. $\overline{RESET}$ sets all pins to the input mode. Each pin has an internal pullup of approximately 50kΩ.

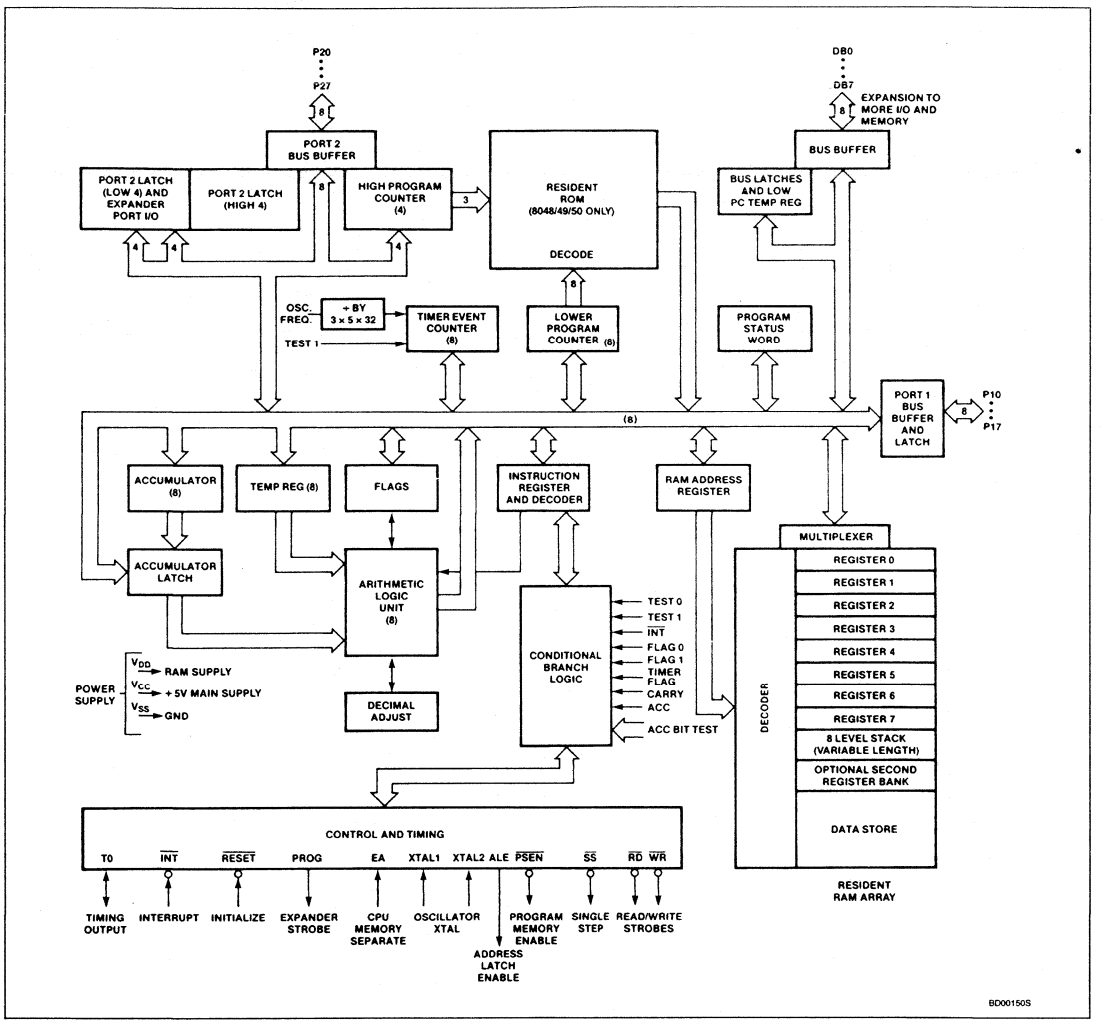
SCN8049, SCN8050, SCN8039, SCN8040 Single-chip 8-bit microcontroller

SCN8049 Series

FUNCTIONAL DESCRIPTION

The following is a general functional description of the SCN8049 Series microcomputers. Refer to the block diagram below.

BLOCK DIAGRAM



PROGRAM MEMORY

Resident program memory consists of up to 4K bytes of ROM. The program memory is divided into pages of 256 bytes each. As shown in the memory map, Figure 1, program memory is also divided into two 2048-byte banks, MB0 and MB1. A total of 4096 bytes can be addressed directly. If more memory is required, an I/O port can be used to address locations over 4095.

There are three locations in program memory of special importance. These locations contain the first instruction to be executed upon the occurrence of one of three events.

LOCATION	EVENT
0	Activation then deactivation of the $\overline{\text{RESET}}$ line.
3	Activation of the $\overline{\text{INT}}$ line when the external interrupt is enabled.
7	An overflow of the timer/counter if the T/C interrupt is enabled.

DATA MEMORY

Resident data memory, as shown in Figure 2, consists of up to 256 bytes of RAM. All locations are indirectly addressable by either of two RAM pointer registers at locations 0 and 1. The first eight locations of RAM (0–7) are designated as working registers and are directly addressable by several instructions.

By selecting register bank 1, RAM locations 24–31 become the working registers, replacing those in register bank 0 (0–7).

RAM locations 8–23 are designated as the stack. Two locations (bytes) are used per CALL, allowing nesting of up to eight subroutines.

If additional RAM is required, up to 256 bytes may be added and addressed directly using the MOVX instructions. If more RAM is required an I/O port can be used to select one (256-byte) bank of external memory at a time.

PROGRAM COUNTER AND STACK

The Program Counter (PC) is a 12-bit counter/register that points to the location from which the next instruction is to be fetched. The 8048 and 8049 will automatically address external memory when the boundary of their internal memory is exceeded. All processors access external memory if EA is high.

An interrupt or CALL to a subroutine causes the contents of the program counter to be stored in one of the 8 register pairs of the program counter stack. The pair to be used is determined by a 3-bit stack pointer which is part of the Program Status Word (PSW). Data RAM locations 8 through 23 are available as stack registers and are used to store the program counter and 4 bits of PSW. The stack pointer, when initialized to 000, points to RAM locations 8 and 9. The first subroutine jump or interrupt results in the program counter contents being transferred to locations 8 and 9 of the RAM array. The stack pointer is then incremented by one to point to locations 10 and 11 in anticipation of another CALL. Nesting of subroutines within subroutines can continue up to eight times without overflowing the stack. If overflow does occur the deepest address stored (location 8 and 9) will be overwritten and lost since the stack pointer overflows from 111 to 000. It also underflows from 000 to 111.

The end of a subroutine, which is signalled by a return instruction (RET or RETR), causes the stack pointer to be decremented and the contents of the resulting register pair to be transferred to the program counter.

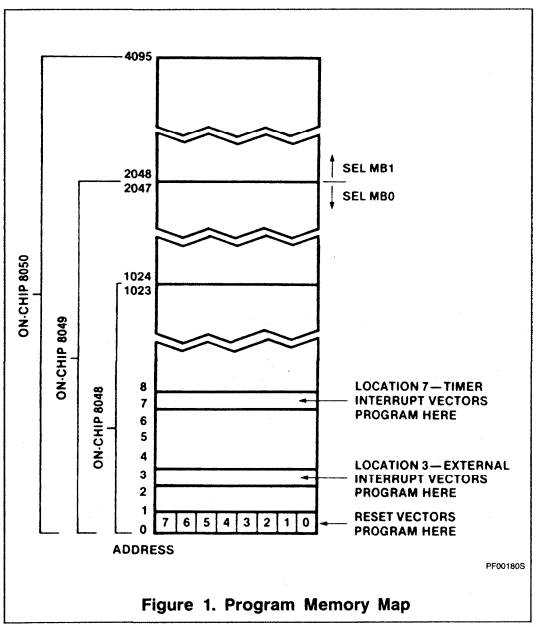


Figure 1. Program Memory Map

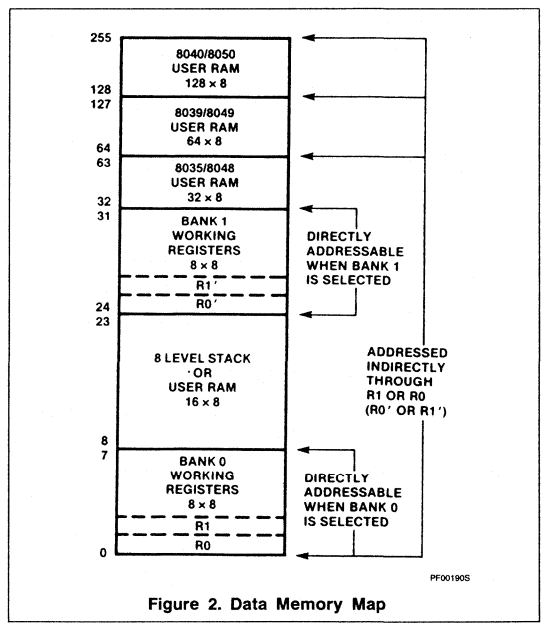


Figure 2. Data Memory Map

OSCILLATOR AND CLOCK

The processor contains its own internal oscillator and clock driver. A crystal, inductor, or external pulse generator may be used to determine the oscillator frequency (see Figure 3). The output of the oscillator is divided by three and can be output on the T0 pin by executing the ENT0 CLK instruction. This CLK signal is divided by 5 to define a machine (instruction) cycle. It is available on Pin 11 as ALE.

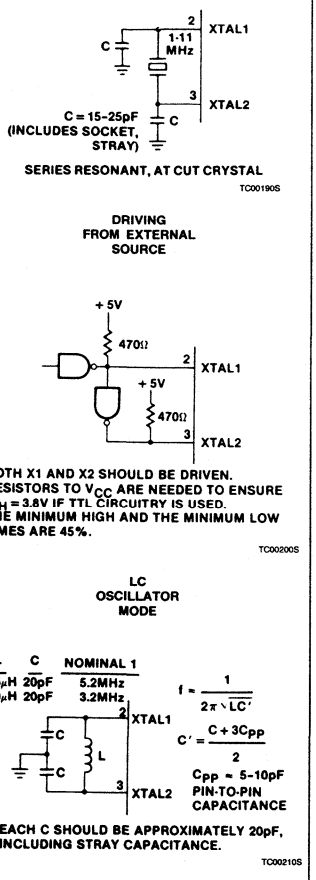


Figure 3. Crystal Oscillator Mode

TIMER/EVENT COUNTER

An internal counter is available which can count either external events or machine cycles (÷ 32). The machine cycles are divided by 32 before they are input to the 8-bit counter. External events are input directly to the counter. The maximum frequency that can be counted is one third of the frequency of the cycle counter. The minimum positive duty cycle that can be detected is 0.2 t_{CY}. The counter is under program control and can be made to generate an interrupt to the processor when it overflows.

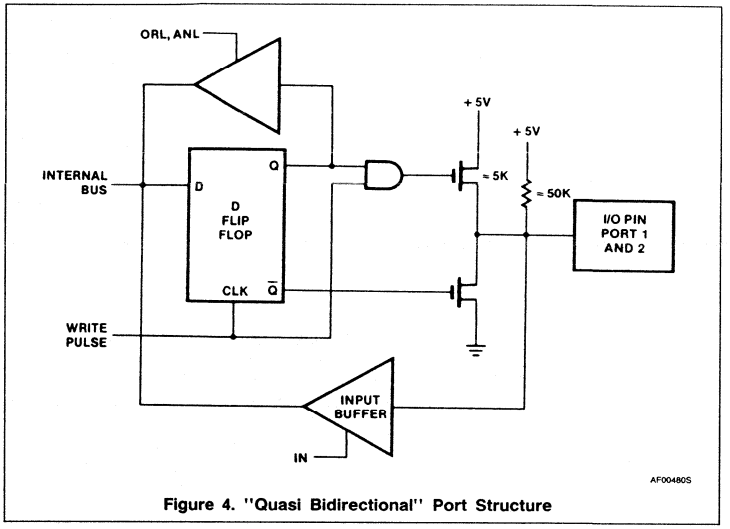


Figure 4. "Quasi Bidirectional" Port Structure

INTERRUPT

An interrupt may be generated by either an external input ($\overline{INT}$, Pin 6) or the overflow of the internal counter, when enabled. In either case, the processor completes execution of the present instruction and then does a CALL to the interrupt service routine. After service, a RETR instruction restores the machine to the state it was prior to the interrupt. The external interrupt has priority over the internal interrupt.

INPUT/OUTPUT

The processor has 27 lines which can be used for input or output functions. These lines are grouped as 3 ports of 8 lines each which serve as either inputs, outputs or bidirectional ports and 3 "test" inputs which can alter program sequences when tested by conditional jump instructions.

Ports 1 and 2

Ports 1 and 2 are each 8 bits wide and have identical characteristics. Data written to these ports is statically latched and remains unchanged until rewritten. As input ports these lines are non-latching; i.e., inputs must be present until read by an input instruction. Inputs are fully TTL compatible and outputs will drive one standard TTL load.

The lines of ports 1 and 2 are called quasi-bidirectional because of a special output circuit structure which allows each line to serve as an input, an output, or both even though outputs are statically latched. Figure 4 shows the circuit configuration. Each line is continuously pulled up to +5V through a resistive device of relatively high impedance (~50K). This pullup is sufficient to provide the source current for a TTL high level yet can be pulled low by a standard TTL gate thus allowing the same pin to be used for both input and output. To provide fast switching times in a "0" to "1" transition a relatively low impedance device (~50KΩ) is switched in momentarily (~500ns) whenever a "1" is written to the line. When a "0" is written to the line, a low impedance (~3000Ω) device overcomes the light pullup and provides TTL current sinking capability.

Since the pulldown transistor is a low impedance device a "1" must first be written to any line which is to be used as an input. Reset initializes all lines to the high impedance "1" state. This structure allows input and output on the same pin and also allows a mix of input lines and output lines on the same port. The quasi-bidirectional port in combination with the ANL and ORL logical instructions provide an efficient means for handling single line inputs and outputs within an 8-bit processor.

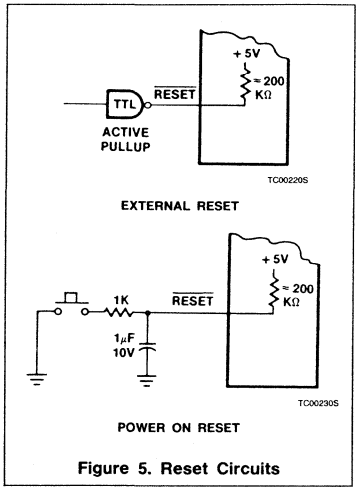


Figure 5. Reset Circuits

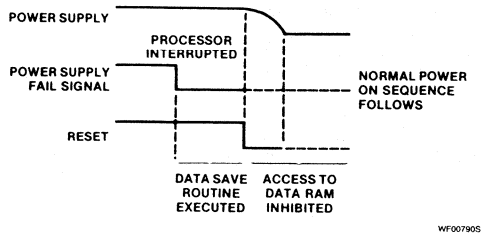


Figure 6. Power Down Sequence

BUS

BUS is also an 8-bit port which is a true bidirectional port with associated input and output strobes. If the bidirectional feature is not needed, BUS can serve as either a statically latched output port or non-latching input port. Input and output lines on this port cannot be mixed.

As a static port, data is written and latched using the OUTL instruction and input using the INS instruction. The INS and OUTL instructions generate pulses on the corresponding $\overline{RD}$ and $\overline{WR}$ output strobe lines; however, in the static port mode they are generally not used. As a bidirectional port, the MOVX instructions are used to read and write to the port. A write to the port generates a pulse on the $\overline{WR}$ output line and output data is valid at the trailing edge of $\overline{WR}$. A read of the port generates a pulse on the $\overline{RD}$ output line and input data must be valid at the trailing edge of $\overline{RD}$. When not being written or read, the BUS lines are in a high impedance state.

Test and INT inputs

Three pins serve as inputs and are testable with the conditional jump instruction. These are T0, T1, and $\overline{INT}$. These pins allow inputs to cause program branches without the necessity to load an input port into the accumulator. The T0, T1, and $\overline{INT}$ pins have other possible functions as well.

RESET INPUT

The reset input provides a means for initialization for the processor. This Schmitt-trigger input has an internal pullup resistor which in combination with an external 1μF capacitor provides an internal reset pulse of sufficient length to guarantee all circuitry is reset. If the reset pulse is generated externally, the reset pin must be held at ground (0.5V) for at least 10 milliseconds after the power supply is within tolerance. Only five machine cycles (12.5μs @ 6MHz) are required if power is already on and the oscillator has stabilized. Typical circuitry is shown in Figure 5.

SINGLE STEP

By proper control of the $\overline{SS}$ line, the mircrocomputer can be made to execute one instruction and then pause or wait until the single step switch is activated again.

POWER DOWN MODE

The SCN8049 Series devices permit power to be removed from all but the data RAM array for low power standby operation. In the power down mode the contents of data RAM can be maintained while drawing typically 5% of normal operating power.

V_{CC} serves as the 5V supply pin for the bulk of the circuitry while the V_{DD} pin supplies only the RAM array. In normal operation both pins are at + 5V. In standby, V_{CC} is at ground and only V_{DD} is maintained at its specified voltage. Applying $\overline{RESET}$ to the processor through the $\overline{RESET}$ pin inhibits any access to the RAM by the processor and guarantees that RAM cannot be inadvertently altered as power is removed from V_{CC}.

A typical power down sequence occurs as shown in Figure 6.

INSTRUCTION SET

The SCN8049 Series instruction set consists of over 90 one and two byte instructions (see Table 1). Program code efficiency is high because: (1) working registers and program variables are stored in RAM, which require only one byte to address and (2) program memory is divided into pages of 256 bytes each, which means that branch destination addresses require one byte.

The instruction set efficiently manipulates and tests bits in addition to performing logical and arithmetic operations upon and the testing of bytes. A set of move instructions operates indirectly upon either RAM or ROM, which permits efficient access of pointers and data tables. The indirect jump instruction performs a multi (up to 256) way branch upon the content of the accumulator to addresses stored in a lookup table. The "decrement register and jump if not zero" instruction saves a byte every time it is used versus using separate increment and test instructions.

The on-chip counter enables either external events or time to be counted off-line from the main program. The processor can either test the counter (under program control) or cause its overflow to generate an interrupt. These features are highly desirable for real time applications. See Table 2 for instruction timing.

ABSOLUTE MAXIMUM RATINGS [1]

SYMBOL	PARAMETER	RATING	UNIT
T_A	Operating ambient temperature[2] range SCN80xxHC SCN80xxHA	 0 to +70 −40 to +85	 °C °C
T_{STG}	Storage temperature range	−65 to +150	°C
V_{IN}	Input voltages with respect to V_{SS}[3]	−0.5 to +7	V
P_D	Power dissipation	1.5	W

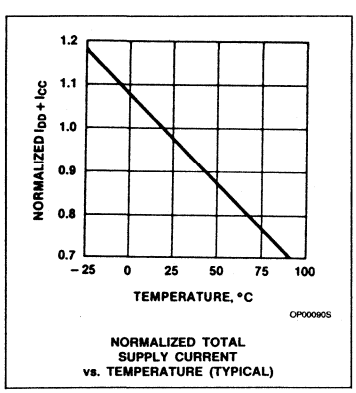
DC ELECTRICAL CHARACTERISTICS T_A = 0°C to 70°C, $V_{CC} = V_{DD}$ = 5V ± 10%, V_{SS} = 0V[4, 5, 6]

SYMBOL	PARAMETER	TEST CONDITIONS[7]	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low-voltage All except XTAL1, XTAL2		−0.5		0.8	V
V_{IL1}	XTAL1, XTAL2		−0.5		0.6	V
V_{IH}	Input high voltage All except $\overline{RESET}$, XTAL1, XTAL2		2.0		V_{CC}	V
V_{IH1}	$\overline{RESET}$, XTAL1, XTAL2		3.8		V_{CC}	V
V_{OL}	Output low-voltage	I_{OL} = 2.0mA			0.4	V
V_{OH}	Output high-voltage All except BUS	$I_{OH} = -125\mu A$	2.4			V
	BUS	$I_{OH} = -400\mu A$	2.4			V
I_{LI1}	Port1, Port2, EA, $\overline{SS}$	$V_{SS} + 0.45 \leq V_{IN} \leq V_{CC}$			−500	μA
I_{L1}	T1, $\overline{Int}$	$V_{SS} + 0.45 \leq V_{IN} \leq V_{CC}$			±10	μA
I_{LI2}	$\overline{RESET}$	$V_{SS} + 0.45 \leq V_{IN} \leq V_{CC}$	−10		−300	μA
I_{OL}	Output leakage current BUS, T0 (high inpedance state)	$V_{SS} + 0.45 \leq V_{IN} \leq V_{CC}$			±10	μA
I_{DD}	Standby supply current 8035/8048 8039/8049 8040/8050	$\overline{RESET} \leq V_{IL}$ All inputs = 0V V_{CC} = 0V			 2.5 4.5 8.5	 mA mA mA
$I_{DD} + I_{CC}$	Total supply current 8035/8048 8039/8049 8040/8050	$\overline{RESET} \leq V_{IL}$		 45 50 60	 80 95 110	 mA mA mA
V_{DD}	Standby power supply		2.5			V
T_A = −40 to 85°C, Automotive temperature range[8]						
V_{IH}	Input high voltage All except XTAL1 and XTAL2		2.2			V
V_{IH1}	$\overline{RESET}$, XTAL1, XTAL2		4.0			V
	Input leakage current					
I_{IL1}	Port1, Port2, EA, $\overline{SS}$	$V_{SS} + .45 \leq V_{IN} \leq V_{CC}$			−750	μA
I_{LI2}	$\overline{RESET}$	$V_{SS} + .45 \leq V_{IN} \leq V_{CC}$	−5		−300	μA
I_{DD}	Standby supply current 8035/8048 8039/8049 8040/8050	$\overline{RESET} \leq V_{IL}$ All inputs = 0V V_{CC} = 0V			 3.75 6.75 12.75	 mA mA mA
$I_{CC} + I_{DD}$	Total supply current 8035/8048 8039/8049 8040/8050	$\overline{RESET} \leq V_{IL}$			 90 105 120	 mA mA mA

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $70°C$, $V_{CC} = V_{DD} = 5V \pm 10\%$, $V_{SS} = 0V$[4, 5, 6]

SYMBOL	PARAMETER	TEST CONDITIONS[7]	11 MHz VERSIONS		6 MHz VERSIONS		UNIT
			Min	Max	Min	Max	
(Refer to Figures 7, 8 and 9)							
t_{LL}	ALE pulse width		150		400		ns
t_{AL}	Address setup to ALE		70		150		ns
t_{LA}	Address hold from ALE		50		80		ns
t_{CC}	Control pulse width ($\overline{PSEN}$, $\overline{RD}$, $\overline{WR}$)		300		700		ns
t_{DW}	Data setup before $\overline{WR}$		250		500		ns
t_{WD}	Data hold after $\overline{WR}$		40		120		ns
t_{CY}	Cycle time		1.36	3.75	2.5	15.0	μs
t_{DR}	Data hold		0	100	0	200	ns
t_{RD}	$\overline{PSEN}$, $\overline{RD}$ to data in			200		500	ns
t_{AW}	Address setup to $\overline{WR}$		200		230		ns
t_{AD}	Address setup to data in			400		950	ns
t_{AFC}	Address float to $\overline{RD}$, $\overline{PSEN}$		−10		0		ns
t_{CA}	Control pulse to ALE		10		10		ns
(Refer to Figure 10)							
t_{CP}	Port control setup before falling edge of PROG		100		110		ns
t_{PC}	Port control hold after falling edge of PROG		60		130		ns
t_{PR}	PROG to time P2 input must be valid			650		810	ns
t_{DP}	Output data setup time		200		250		ns
t_{PD}	Output data hold time		20		65		ns
t_{PF}	Input data hold time		0	150	0	150	ns
t_{PP}	PROG pulse width		700		1200		ns
t_{PL}	Port 2 I/O data setup		250		350		ns
t_{LP}	Port 2 I/O data hold		20		150		ns

NOTES:
1. Stresses above those listed under absolute maximum ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to aviod applying any voltages larger than the rated maximum.
4. Parameters are valid over operating temperature range unless otherwise specified.
5. All voltage measurements are referenced to ground (V_{SS}). For testing, all input signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages 0.8V and 2.0V as appropriate.
6. Typical values are at +25°C, typical ssupply voltages and typical processing parameters.
7. Control outputs: $C_L = 80pF$
   Bus outputs: $C_L = 150pF$
   $t_{CY} = 1.36\mu s$ for 11 MHz versions
   $t_{CY} = 2.5\mu s$ for 6 MHz versions
8. Where no specification is shown, the commercial temperature range specification applies.



NORMALIZED TOTAL SUPPLY CURRENT vs. TEMPERATURE (TYPICAL)

TIMING DIAGRAMS

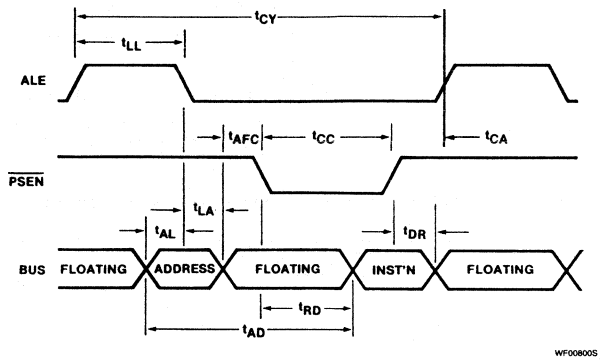


Figure 7. Instruction Fetch From External Program Memory

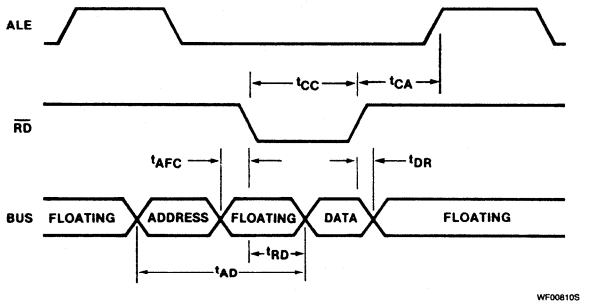


Figure 8. Read From External Data Memory

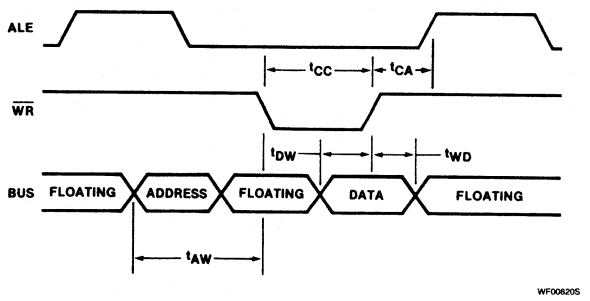


Figure 9. Write to External Data Memory

TIMING DIAGRAMS (Continued)

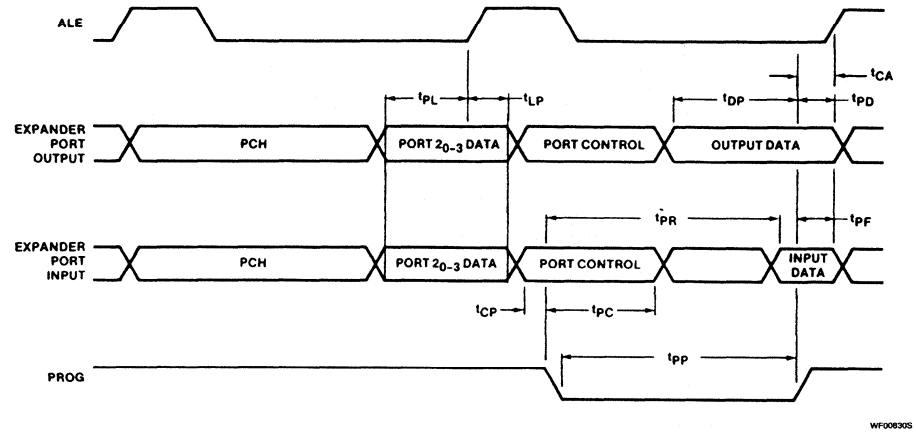


Figure 10. Port 2 Timing

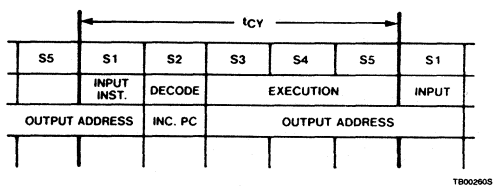


Figure 11. Instruction Cycle

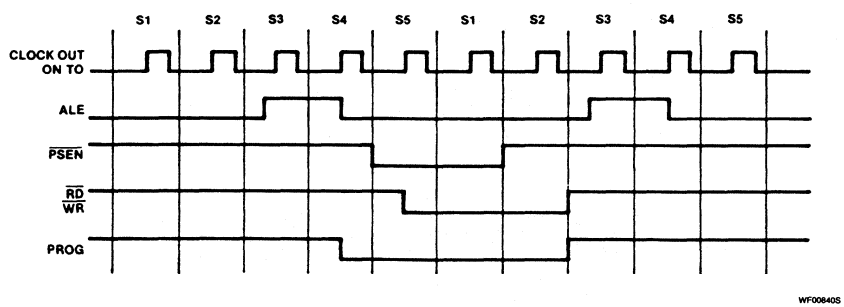


Figure 12. Instruction Cycle Timing

Table 1. Instruction Set

MNEMONIC	FUNCTION	DESCRIPTION	INSTRUCTION CODE D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	CYCLES	BYTES	FLAGS C	AC	F0	F1	F2
Accumulator										
ADD A, # data	(A) ← (A) + data	Add immediate the specified data to the accumulator.	0 0 0 0 0 0 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2	•	•			
ADD A, Rr	(A) ← (A) + (Rr) for r = 0 – 7	Add contents of designated register to the accumulator.	0 1 1 0 1 r r r	1	1	•	•			
ADD A, @ Rr	(A) ← (A) + ((Rr)) for r = 0 – 1	Add indirect the contents the data memory location to the accumulator.	0 1 1 0 0 0 0 r	1	1	•	•			
ADDC A, # data	(A) ← (A) + (C) + data	Add immediate with carry the specified data to the accumulator.	0 0 0 1 0 0 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2	•	•			
ADDC A, Rr	(A) ← (A) + (C) + (Rr) for r = 0 – 7	Add with carry the contents of the designated register to the accumulator.	0 1 1 1 1 r r r	1	1	•	•			
ADDC A, @ Rr	(A) ← (A) + (C) + ((Rr)) for r = 0 – 1	Add indirect with carry the contents of data memory location to the accumulator.	0 1 1 1 0 0 0 r	1	1	•	•			
ANL A, # data	(A) ← (A) AND data	Logical AND specified immediate data with accumulator.	0 1 0 1 0 0 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
ANL A, Rr	(A) ← (A) AND (Rr) for r = 0 – 7	Logical AND contents of designated register with accumulator.	0 1 0 1 1 r r r	1	1					
ANL A, @ Rr	(A) ← (A) AND ((Rr)) for r = 0 – 1	Logical AND indirect the contents of data memory with accumulator.	0 1 0 1 0 0 0 r	1	1					
CPL A	(A) ← NOT (A)	Complement the contents of the accumulator.	0 0 1 1 0 1 1 1	1	1					
CLR A	(A) ← 0	Clear the contents of the accumulator.	0 0 1 0 0 1 1 1	1	1					
DA A		Decimal adjust the contents of the accumulator.	0 1 0 1 0 1 1 1	1	1	•	•			
DEC A	(A) ← (A) – 1	Decrement the accumulator's contents by 1.	0 0 0 0 0 1 1 1	1	1					
INC A	(A) ← (A) + 1	Increment the accumulator's contents by 1.	0 0 0 1 0 1 1 1	1	1					
ORL A, # data	(A) ← (A) OR data	Logical OR specified immediate data with accumulator.	0 1 0 0 0 0 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
ORL A, Rr	(A) ← (A) OR (Rr) for r = 0 – 7	Logical OR contents of designated register with accumulator.	0 1 0 0 1 r r r	1	1					
ORL A, @ Rr	(A) ← (A) OR ((Rr)) for r = 0 – 1	Logical OR indirect the contents of data memory location with accumulator.	0 1 0 0 0 0 0 r	1	1					
RL A	(An + 1) ← (An) (A_0) ← (A_7) for N = 0 ← 6	Rotate accumulator left by 1-bit without carry.	1 1 1 0 0 1 1 1	1	1					
RLC A	(An + 1) ← (An); n = 0 – 6 (A_0) ← (C) (C) ← (A_7)	Rotate accumulator left by 1-bit through carry.	1 1 1 1 0 1 1 1	1	1	•				
RR A	(An) ← (An + 1); n = 0 – 6 (A_7) ← (A_0)	Rotate accumulator right by 1-bit without carry.	0 1 1 1 0 1 1 1	1	1					
RRC A	(An) ← (An + 1); n = 0 – 6 (A_7) ← (C) (C) ← (A_0)	Rotate accumulator right by 1-bit through carry.	0 1 1 0 0 1 1 1	1	1	•				
SWAP A	(A_{4-7}) ← (A_0 – 3)	Swap the 2 4-bit nibbles in the accumulator.	0 1 0 0 0 1 1 1	1	1					
XRL A, # data	(A) ← (A) XOR data	Logical XOR specified immediate data with accumulator.	1 1 0 1 0 0 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
XRL A, Rr	(A) ← (A) XOR (Rr) for r = 0 – 7	Logical XOR contents of designated register with accumulator.	1 1 0 1 1 r r r	1	1					
XRL A, @ Rr	(A) ← (A) XOR ((Rr)) for r = 0 – 1	Logical XOR indirect the contents of data memory location with accumulator.	1 1 0 1 0 0 0 r	1	1					

Table 1. Instruction Set (Continued)

MNEMONIC	FUNCTION	DESCRIPTION	INSTRUCTION CODE D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	CYCLES	BYTES	FLAGS C	AC	F0	F1	F2
Branch										
DJNZ Rr, addr	(Rr) ← (Rr) – 1; r = 0 – 7 if (Rr) ≠ 0: (PC 0 – 7) ← addr	Decrement the specified register and test contents.	1 1 1 0 1 r r r a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JBb addr	(PC 0 – 7) ← addr if Bb = 1 (PC) ← (PC) + 2 if Bb = 0	Jump to specified address if accumulator bit is set.	b_2 b_1 b_0 1 0 0 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JC addr	(PC 0 – 7) ← addr if C = 1 (PC) ← (PC) + 2 if C = 0	Jump to specified address if carry flag is set.	1 1 1 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JF0 addr	(PC 0 – 7) ← addr if F0 = 1 (PC) ← (PC) + 2 if F0 = 0	Jump to specified address if flag F0 is set.	1 0 1 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JF1 addr	(PC 0 – 7) ← addr if F1 = 1 (PC) ← (PC) + 2 if F1 = 0	Jump to specified address if flag F1 is set.	0 1 1 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JMP addr	(PC 8 – 10) ← addr 8 – 10 (PC 0 – 7) ← addr 0 – 7 (PC 11) ← (DBF)	Direct jump to specified address within the 2K address block.	a_{10} a_9 a_8 0 0 1 0 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JMPP @ A	(PC 0 – 7) ← ((A))	Jump indirect to specified address within address page.	1 0 1 1 0 0 1 1	2	1					
JNC addr	(PC 0 – 7) ← addr if C = 0 (PC) ← (PC) + 2 if C = 1	Jump to specified address if carry flag is low.	1 1 1 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JNI	(PC 0 – 7) ← addr if $\overline{INT}$ = 0 (PC) ← (PC) + 2 if $\overline{INT}$ = 1	Jump to specified address if $\overline{INT}$ input is low.	1 0 0 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JNT0 addr	(PC 0 – 7) ← addr if T0 = 0 (PC) ← (PC) + 2 if T0 = 1	Jump to specified address if test 0 is low.	0 0 1 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JNT1 addr	(PC 0 – 7) ← addr if T1 = 0 (PC) ← (PC) + 2 if T1 = 1	Jump to specified address if test 1 is low.	0 1 0 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JNZ addr	(PC 0 – 7) ← addr if A = 0 (PC) ← (PC) + 2 if A = 0	Jump to specified address if accumulator is non-zero.	1 0 0 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JTF addr	(PC 0 – 7) ← addr if TF = 1 (PC) ← (PC) + 2 if TF = 0	Jump to specified address if timer flag is set to 1.	0 0 0 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JT0 addr	(PC 0 – 7) ← addr if T0 = 1 (PC) ← (PC) + 2 if T0 = 0	Jump to specified address if test 0 is a 1.	0 0 1 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JT1 addr	(PC 0 – 7) ← addr if T1 = 1 (PC) ← (PC) + 2 if T1 = 0	Jump to specified address if test 1 is a 1.	0 1 0 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JZ addr	(PC 0 – 7) ← addr if A = 0 (PC) ← (PC) + 2 if A ≠ 0	Jump to specified address if accumulator is 0.	1 1 0 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
Control										
EN I		Enable the external ($\overline{INT}$) interrupt.	0 0 0 0 0 1 0 1	1	1					
DIS I		Disable the external ($\overline{INT}$) interrupt.	0 0 0 1 0 1 0 1	1	1					
SEL RB0	(BS) ← 0	Select bank 0 (locations 0 – 7) of data memory.	1 1 0 0 0 1 0 1	1	1					•

Table 1. Instruction Set (Continued)

MNEMONIC	FUNCTION	DESCRIPTION	INSTRUCTION CODE	CYCLES	BYTES	FLAGS				
			D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0			C	AC	F0	F1	F2
Control (Cont.)										
SEL RB1	(BS) ← 1	Select bank 1 (locations 24 – 31) of data memory.	1 1 0 1 0 1 0 1	1	1					•
SEL MB0	(DBF) ← 0	Select program memory bank 0, addresses 0 – 2047.	1 1 1 0 0 1 0 1	1	1					
SEL MB1	(DBF) ← 1	Select program memory bank 1, addresses 2048 – 4095	1 1 1 1 0 1 0 1	1	1					
ENT0 CLK		Enable clock output on T0 pin.	0 1 1 1 0 1 0 1	1	1					
Data moves										
MOV A, # data	(A) ← data	Move immediate the specified data into the accumulator.	0 0 1 0 0 0 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
MOV A, Rr	(A) ← (Rr); r = 0 – 7	Move the contents of the designated register into the accumulator.	1 1 1 1 1 r r r	1	1					
MOV A, @ Rr	(A) ← ((Rr)); r = 0 – 1	Move indirect the contents of data memory location into the accumulator.	1 1 1 1 0 0 0 r	1	1					
MOV A, PSW	(A) ← (PSW)	Move contents of the program status word into the accumulator.	1 1 0 0 0 1 1 1	1	1					
MOV Rr, # data	(Rr) ← data; r = 0 – 7	Move immediate the specified data into the designated register.	1 0 1 1 1 r r r d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
MOV Rr, A	(Rr) ← (A); r = 0 – 7	Move accumulator contents into the designated register.	1 0 1 0 1 r r r	1	1					
MOV @ Rr, A	((Rr)) ← (A); r = 0 – 1	Move indirect accumulator contents into data memory location.	1 0 1 0 0 0 0 r	1	1					
MOV @ Rr, # data	((Rr)) ← data; r = 0 – 1	Move indirect the specified data into data memory.	1 0 1 1 0 0 0 r d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
MOV PSW, A	(PSW) ← (A)	Move contents of accumulator into the program status word.	1 1 0 1 0 1 1 1	1	1	•	•	•		•
MOVP A, @ A	(A) ← ((A))	Move data in the current page into the accumulator.	1 0 1 0 0 0 1 1	2	1					
MOVP3 A, @ A	(A) ← ((A)) in page 3	Move data in page 3 into the accumulator.	1 1 1 0 0 0 1 1	2	1					
MOVX A, @ Rr	(A) ← ((Rr)); r = 0 – 1	Move indirect the contents of external memory location into the accumulator.	1 0 0 0 0 0 0 r	2	1					
MOVX @ Rr, A	((Rr)) ← (A); r = 0 – 1	Move indirect the contents of the accumulator into external memory.	1 0 0 1 0 0 0 r	2	1					
XCH A, Rr	(A) ↔ (Rr); r = 0 – 7	Exchange the accumulator and designated register's contents.	0 0 1 0 1 r r r	1	1					
XCH A, @ Rr	(A) ↔ ((Rr)); r = 0 – 1	Exchange indirect contents of accumulator and location in data memory.	0 0 1 0 0 0 0 r	1	1					
XCHD A, @ Rr	(A 0 – 3) ↔ (Rr)(0 – 3) r = 0 – 1	Exchange indirect 4-bit contents of accumulator and data memory.	0 0 1 1 0 0 0 r	1	1					
Flags										
CPL C	(C) ← NOT (C)	Complement content of carry bit.	1 0 1 0 0 1 1 1	1	1	•				
CPL F0	(F0) ← NOT (F0)	Complement content of flag F0.	1 0 0 1 0 1 0 1	1	1			•		
CPL F1	(F1) ← NOT (F1)	Complement content of flag F1.	1 0 1 1 0 1 0 1	1	1				•	
CLR C	(C) ← 0	Clear content of carry bit to 0.	1 0 0 1 0 1 1 1	1	1	•				
CLR F0	(F0) ← 0	Clear content of flag 0 to 0.	1 0 0 0 0 1 0 1	1	1			•		
CLR F1	(F1) ← 0	Clear content of flag 1 to 0.	1 0 1 0 0 1 0 1	1	1					•
Input/output										
ANL BUS, # data	(BUS) ← (BUS) AND data	Logical AND immediate specified data with BUS.	1 0 0 1 1 0 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
ANL Pp, # data	(Pp) ← (Pp) AND data p = 1 – 2	Logical AND immediate specified data with designated port (1 or 2).	1 0 0 1 1 0 p p d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
ANLD Pp, A	(Pp) ← (Pp) AND (A 0 – 3) p = 4 – 7	Logical AND contents of accumulator with designated port (4 – 7).	1 0 0 1 1 1 p p	2	1					

Table 1. Instruction Set (Continued)

MNEMONIC	FUNCTION	DESCRIPTION	INSTRUCTION CODE D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	CYCLES	BYTES	FLAGS C	AC	F0	F1	F2
Input/output (Cont.)										
IN A, Pp	(A) ← (Pp); p = 1 – 2	Input data from designated port (1 – 2) into accumulator.	0 0 0 0 1 0 p p	2	1					
INS A, BUS	(A) ← (BUS)	Input strobed BUS data into accumulator.	0 0 0 0 1 0 0 0	1	2					
MOVD A, Pp	(A 0 – 3) ← (Pp); p = 4 – 7 (A 4 – 7) ← 0	Move contents of designated port (4 – 7) into accumulator.	0 0 0 0 1 1 p p	2	1					
MOVD Pp, A	(Pp) ← A 0 – 3; p = 4 – 7	Move contents of accumulator to designated port (4 – 7).	0 0 1 1 1 1 p p	1	1					
ORLD Pp, A	(Pp) ← (Pp) OR (A 0 – 3) p = 4 – 7	Logical OR contents of accumulator with designated port (4 – 7).	1 0 0 0 1 1 p p	1	1					
ORL BUS, # data	(BUS) ← (BUS) OR data	Logical OR immediate specified data with BUS.	1 0 0 0 1 0 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
ORL Pp, # data	(Pp) ← (Pp) OR data p = 1 – 2	Logical OR immediate specified data with designated port (1 – 2).	1 0 0 0 1 0 p p d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
OUTL BUS, A	(BUS) ← (A)	Output contents of accumulator onto BUS.	0 0 0 0 0 0 1 0	1	2					
OUTL Pp, A	(Pp) ← (A); p = 1 – 2	Output contents of accumulator to designated port (1 – 2).	0 0 1 1 1 0 p p	1	1					
Registers										
DEC Rr	(Rr) ← (Rr) – 1; r = 0 – 7	Decrement contents of designated register by 1.	1 1 0 0 1 r r r	1	1					
INC Rr	(Rr) ← (Rr) + 1; r = 0 – 7	Increment contents of designated register by 1.	0 0 0 1 1 r r r	1	1					
INC @ Rr	((Rr)) ← ((Rr)) + 1; r = 0 – 1	Increment indirect the contents of data memory location by 1.	0 0 0 1 0 0 0 r	1	1					
Subroutine										
CALL addr	((SP)) ← (PC), (PSW 4 – 7) (SP) ← (SP) + 1 (PC 8 – 10) ← addr 8 – 10 (PC 0 – 7) ← addr 0 – 7 (PC 11) ← DBF	Call designated subroutine.	a_{10} a_9 a_8 1 0 1 0 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
RET	(SP) ← (SP) – 1 (PC) ← ((SP))	Return from subroutine without restoring program status word.	1 0 0 0 0 0 1 1	2	1					
RETR	(SP) ← (SP) – 1 (PC) ← ((SP)) (PSW 4 – 7) ← ((SP))	Return from subroutine restoring program status word.	1 0 0 1 0 0 1 1	2	1					
Timer/counter										
EN TCNTI		Enable timer/counter interrupt.	0 0 1 0 0 1 0 1	1	1					
DIS TCNTI		Disable timer/counter interrupt.	0 0 1 1 0 1 0 1	1	1					
MOV A, T	(A) ← (T)	Move contents of timer/counter into accumulator.	0 1 0 0 0 0 1 0	1	1					
MOV T, A	(T) ← (A)	Move contents of accumulator into timer/counter.	0 1 1 0 0 0 1 0	1	1					
STOP TCNT		Stop count for event counter or timer.	0 1 1 0 0 1 0 1	1	1					
STRT CNT		Start count for event counter.	0 1 0 0 0 1 0 1	1	1					
STRT T		Start count for timer.	0 1 0 1 0 1 0 1	1	1					
Miscellaneous										
NOP		No operation performed	0 0 0 0 0 0 0 0	1	1					

NOTES:
1. Instruction code designations r and p form the binary representation of the registers and ports involved.
2. The dot under the appropriate flag bit indicates that its content is subject to change by the instruction in which it appears.
3. Numerical subscripts appearing in the FUNCTION column reference the specific bits affected.

SYMBOL DEFINITIONS

SYMBOL	DESCRIPTION
A	The accumulator
AC	The auxiliary carry flag
addr	Program memory address (11 bits)
Bb	Bit designator (b = 0 – 7)
BS	The bank switch
C	Carry flag
CLK	Clock signal
CNT	Event counter
D	Nibble designator (4 bits)
DBF	Program memory bank flip-flop
data	Number or expression (8 bits)
F_0, F_1	Flags 0, 1
I	Interrupt
$\overline{INT}$	External interrupt

P	"In-Page" operation designator
P_p	Port designator (p = 1, 2 or 4 – 7)
PSW	Program status word
Rr	Register designator (r = 0, 1 or 0 – 7)
SP	Stack pointer
T	Timer
TF	Timer flag
T0, T1	Testable inputs 0, 1
#	Prefix for immediate data
@	Prefix for indirect address
$	Program counter's current value
←	Replaced by
↔	Exchanged with

Table 2. Instruction Timing**

	CYCLE 1					CYCLE 2				
INSTRUCTION	**S1**	**S2**	**S3**	**S4**	**S5**	**S1**	**S2**	**S3**	**S4**	**S5**
IN A,P	Fetch Instruction	Increment Program Counter	—	Increment Timer	—	—	Read Port	* —	—	—
OUTL P,A	Fetch Instruction	Increment Program Counter	—	Increment Timer	Output To Port	—	—	* —	—	—
ANL P, # data	Fetch Instruction	* Increment Program Counter	—	Increment Timer	Read Port	Fetch Immediate Data	—	* Increment Program Counter	Output To Port	—
ORL P, # data	Fetch Instruction	* Increment Program Counter	—	Increment Timer	Read Port	Fetch Immediate Data	—	* Increment Program Counter	Output To Port	—
INS A, BUS	Fetch Instruction	Increment Program Counter	—	Increment Timer	—	—	Read Port	* —	—	—
OUTL BUS, A	Fetch Instruction	Increment Program Counter	—	Increment Timer	Output To Port	—	—	* —	—	—
ANL BUS, # data	Fetch Instruction	* Increment Program Counter	—	Increment Timer	Read Port	Fetch Immediate Data	—	* Increment Program Counter	Output To Port	—
ORL BUS, # data	Fetch Instruction	* Increment Program Counter	—	Increment Timer	Read Port	Fetch Immediate Data	—	* Increment Program Counter	Output To Port	—
MOVX @R,A	Fetch Instruction	Increment Program Counter	Output RAM Address	Increment Timer	Output Data to RAM	—	—	* —	—	—
MOVX A,@R	Fetch Instruction	Increment Program Counter	Output RAM Address	Increment Timer	—	—	Read Data	* —	—	—
MOVD A, P_i	Fetch Instruction	Increment Program Counter	Output Opcode/Address	Increment Timer	—	—	Read P2 Lower	* —	—	—
MOVD P_h, A	Fetch Instruction	Increment Program Counter	Output Opcode/Address	Increment Timer	Output Data TO P2 Lower	—	—	* —	—	—
ANLD P, A	Fetch Instruction	Increment Program Counter	Output Opcode/Address	Increment Timer	Output Data	—	—	* —	—	—
ORLD P, A	Fetch Instruction	Increment Program Counter	Output Opcode/Address	Increment Timer	Output Data	—	—	* —	—	—
J (CONDITIONAL)	Fetch Instruction	* Increment Program Counter	Sample Condition	Increment Timer	—	Fetch Immediate Data	—	* Update Program Counter	—	—
START CNT/STRT T	Fetch Instruction	* Increment Program Counter	—	—	Start Counter					
STOP TCNT	Fetch Instruction	* Increment Program Counter	—	—	Stop Counter					
EN I	Fetch Instruction	* Increment Program Counter	—	Enable Interrupt	—					
DIS I	Fetch Instruction	* Increment Program Counter	—	Disable Interrupt	—					
ENT0 CLK	Fetch Instruction	* Increment Program Counter	—	Enable Clock	—					

NOTES:
*Valid instruction address are output at this time if external program memory is being accessed.
**See figures 11 and 12 for instruction cycle and cycle timing.

Date of Issue	December 17, 1986
Status	Product Specification
Application Specific Product	

8X305
Microcontroller

FEATURES

- Fetch, Decode, and Execute a 16-bit instruction in a minimum of 200ns (one machine cycle)
- Bit-oriented instruction set (addressable single-or-multiple bit subfields)
- Separate buses for Instruction, Instruction Address and Three-State I/O
- Thirteen 8-bit general-purpose working registers
- Source/destination architecture
- Bipolar low-power Schottky technology/TTL inputs and outputs
- On-chip oscillator and timing generation
- Single +5V supply
- 0.9-in. 50-pin DIP
- 68-pin PLCC

DESCRIPTION

The 8X305 Microcontroller (Figure 1) is a high-speed bipolar microprocessor implemented with low-power Schottky technology. In a single chip, the 8X305 combines speed, flexibility, and a bit-oriented instruction set. These features and other basic characteristics of the chip combine to provide cost-effective solutions for a broad range of applications. The 8X305 is particularly useful in systems that require high-speed bit manipulations—sophisticated controllers, data communications, very fast interface control, and other applications of a similar nature.

The 8X305 can fetch, decode, and execute a 16-bit instruction in a minimum of 200ns. Within one instruction cycle, the 8-bit data-processing path can be programmed to rotate, mask, shift, and/or merge single or multiple bit subfields and, in addition, perform an ALU operation. In the same instruction, an external data field can be input, processed, and output to a specified destination—likewise, single or multiple bit data fields can be internally moved from a given source to a given destination. To summarize, fixed or variable-length data fields can be fetched, processed, operated on by the ALU, and moved to a different location—all in a timeframe of 200ns. To interface with I/O and program memory, the 8X305 uses a 13-bit instruction address bus, a 16-bit instruction bus, an 8-bit bidirectional multiplexed I/O data/address bus and a 5-bit I/O control bus.

A wide selection of I/O devices, interface chips, and special-purpose parts are available for systems use. In most applications, the more powerful 8X305 is functionally interchangeable with its predecessor—the 8X300.

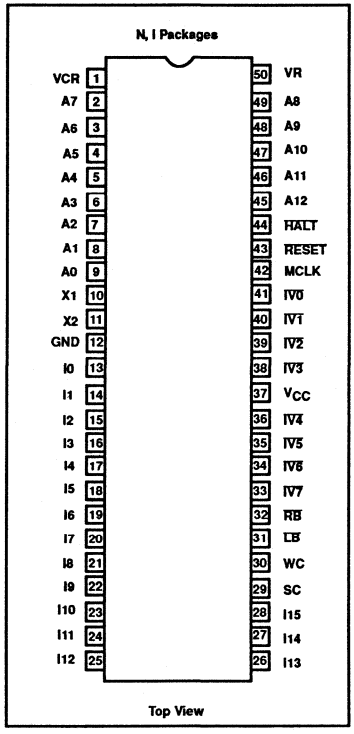
ASSOCIATED DOCUMENTATION

Other documents directly relating to *design* and *applications use* of the 8X305 Microcontroller are:

- Product Capabilities Manual
- 8X305 Users Manual

These documents and other current literature (Data Sheets, Product Bulletins, Applications Notes, etc.) are available at all Signetics Sales and Service Offices—see rear cover of this data sheet for the office in your locality.

PIN CONFIGURATIONS



ORDERING INFORMATION

DESCRIPTION	ORDER CODE
50-Pin plastic DIP	N8X305N
50-Pin ceramic DIP	N8X305I
68-Pin PLCC	N8X305A

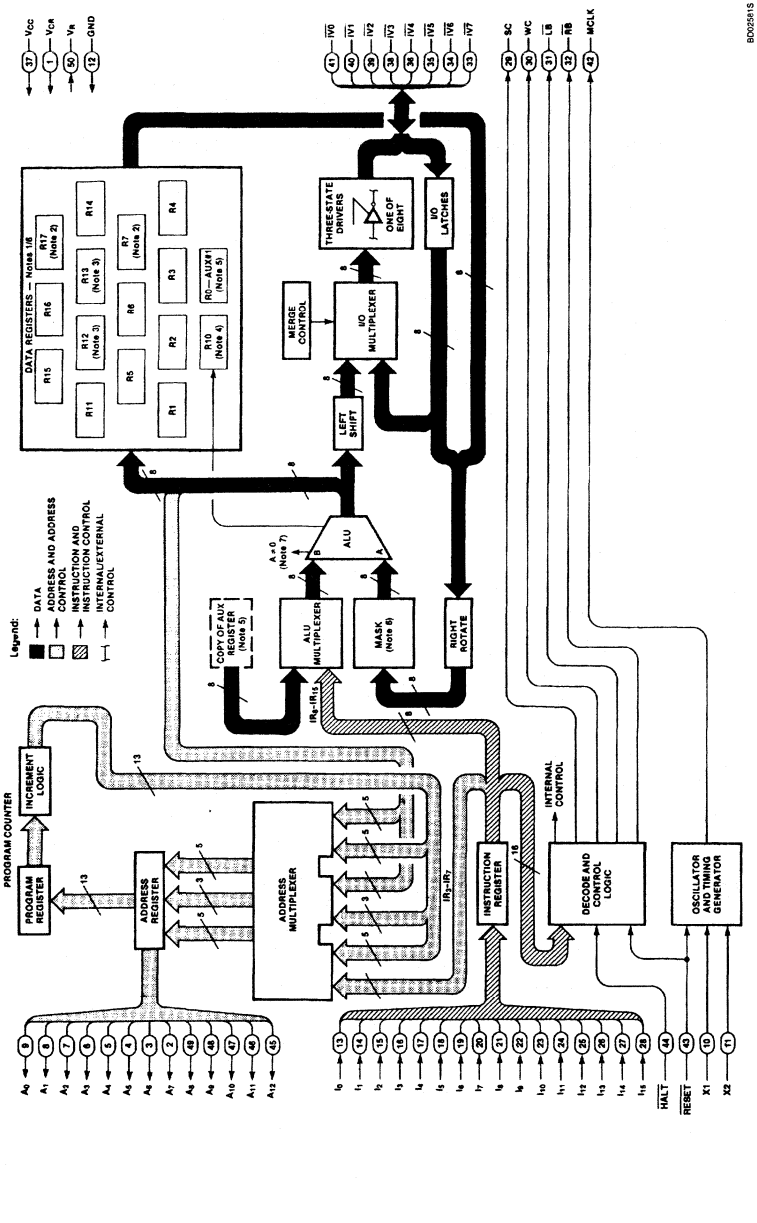
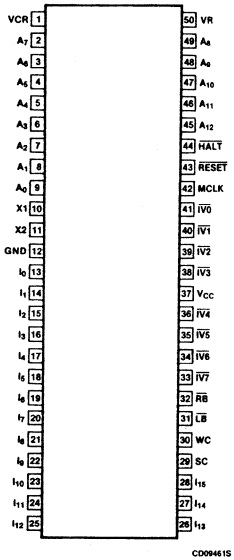
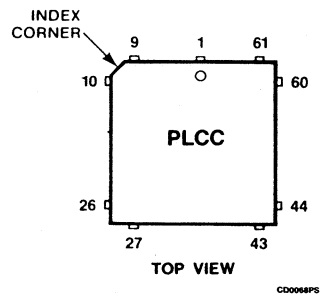


Figure 1. Architecture and Pin Designations for 8X305 Microcontroller
Pin Numbers Shown are for DIP Packaging

NOTES:
1. Registers R1 – R6, R11 and R14 – R16 are general-purpose working registers.
2. In any instruction where R7 (IVL) or R17 (IVR) is specified as the destination, the 8-bit value is output on the IV bus as an IV device enable address (SC = High) — R7 = left bank and R17 = right bank; the results are also stored into the specified internal register and may later be accessed as source data.
3. R12 and R13 are general-purpose working registers for all operations except transmit (XMIT).
4. The least significant bit of register R10 (OVF) is used to reflect the carryout status resulting from the most recent ADD operation.
5. Auxiliary register R0 #1 is a general-purpose working register that holds the implied operand for Arithmetic and Logical operations; the content of this register is repeated in AUX #2 (shown dotted). The duplicate register is physically part of the ALU and is shown separate only for layout convenience.
6. Internal working registers cannot be operated on by the MASK logic.
7. During NZT instructions the ALU tests for all bits equal to "0" (Transfer if A ≠ 0) — refer to BASIC OPERATIONS that follow.





PLCC PIN NO.	DIP PIN NO.	IDENTIFIER	FUNCTION
1, 68	1	VCR	Regulated voltage input from series-pass transistor (2N5320 or equivalent).
4-11, 62-66	2-9, 45-49	$A_0 - A_{12}$	**Program Address Lines:** These active-high outputs permit direct addressing of up to 8192 words of program storage; A_{12} is least significant bit.
12, 13	10, 11	X1, X2	Timing generator connections for a capacitor, a series resonant crystal, or an external clock source with complementary outputs.
2,3, 14-16	12	GND	Ground.
17-23, 28-36	13-28	$I_0 - I_{15}$	**Instruction Lines:** These active-high input lines receive 16-bit instructions from program storage; I_{15} is least significant bit.
37	29	SC	**Select Command:** When high (binary 1), an address is being output on pins $\overline{IV0}$ through $\overline{IV7}$.
38	30	WC	**Write Command:** When high (binary 1), data is being output on pins $\overline{IV0}$ through $\overline{IV7}$.
39	31	$\overline{LB}$	**Left Bank Control:** When low (binary 0), devices connected to the Left Bank are accessed. (Note: *Typically, the $\overline{LB}$ signal is tied to the $\overline{ME}$ input pin of I/O peripherals*).
45	32	RB	**Right Bank Control:** When low (binary 0), devices connected to the Right Bank are accessed (Note. *Typically, the $\overline{RB}$ signal is tied to the $\overline{ME}$ input pin of I/O peripherals*).

PLCC PIN NO.	DIP PIN NO.	IDENTIFIER	FUNCTION
46-49, 55-58	33-36, 38-41	$\overline{IV0} - \overline{IV7}$	**Interface Vector** (Input/Output Bus) — these bidirectional active-low three-state lines communicate data and/or addresses to I/O devices and memory locations. A low voltage level equals a binary "1"; $\overline{IV7}$ is Least Significant Bit.
50-52	37	V_{CC}	+5V power supply.
59	42	MCLK	**Master Clock:** This active-high output signal is used for clocking I/O devices and/or synchronization of external logic.
60	43	$\overline{RESET}$	When **$\overline{RESET}$** input is low (binary 0), the 8X305 is initialized — sets Program Counter/Address Register to zero and inhibits MCLK. For the period of time $\overline{RESET}$ is low, the Left Bank/Right Bank ($\overline{LB}/\overline{RB}$) signals are forced high asynchronously.
61	44	$\overline{HALT}$	When **$\overline{HALT}$** input is low (binary 0), internal operation of the 8X305 stops at the start of next instruction; MCLK is not inhibited nor is any internal register affected. However, both the Left Bank/Right Bank ($\overline{LB}/\overline{RB}$) signals are synchronously driven high during the first quarter of the instruction cycle time and remain high during the time HALT is low.
67	50	VR	Internally-generated reference output voltage for external series-pass regulator transistor.
24-27, 40-44, 53, 54	–	No Connect	

NOTE:
Multiple V_{CC}, GND, and V_{CR} pins must be externally connected.

Figure 2. Designations and Descriptions for Pins of 8X305 Microcontroller.

FUNCTIONAL OPERATION

Typical System Configuration

Although the system hookup shown in Figure 3 is of the simplest form, it provides a fundamental look at the 8X305 Microcontroller and peripheral relationships. As indicated, the 8X305 can directly address up to 8K words of program storage — either ROM or PROM. The user interface ($\overline{IV0}$ through $\overline{IV7}$) is capable of uniquely addressing 256 Input/Output locations and, with additional bank bits ($\overline{LB}$, $\overline{RB}$), this number is expanded to 512 — each bank comprising 256 addressable locations. The addressable locations of each bank can be used in a variety of ways; a simple method of implementation is shown in Figure 3. When $\overline{LB}$ is active low, the left bank is enabled and any one of 256 locations within the RAM memory can be accessed for input/output operations. A similar set of "enable/access" conditions are applicable to the right bank when $\overline{RB}$ is active low.

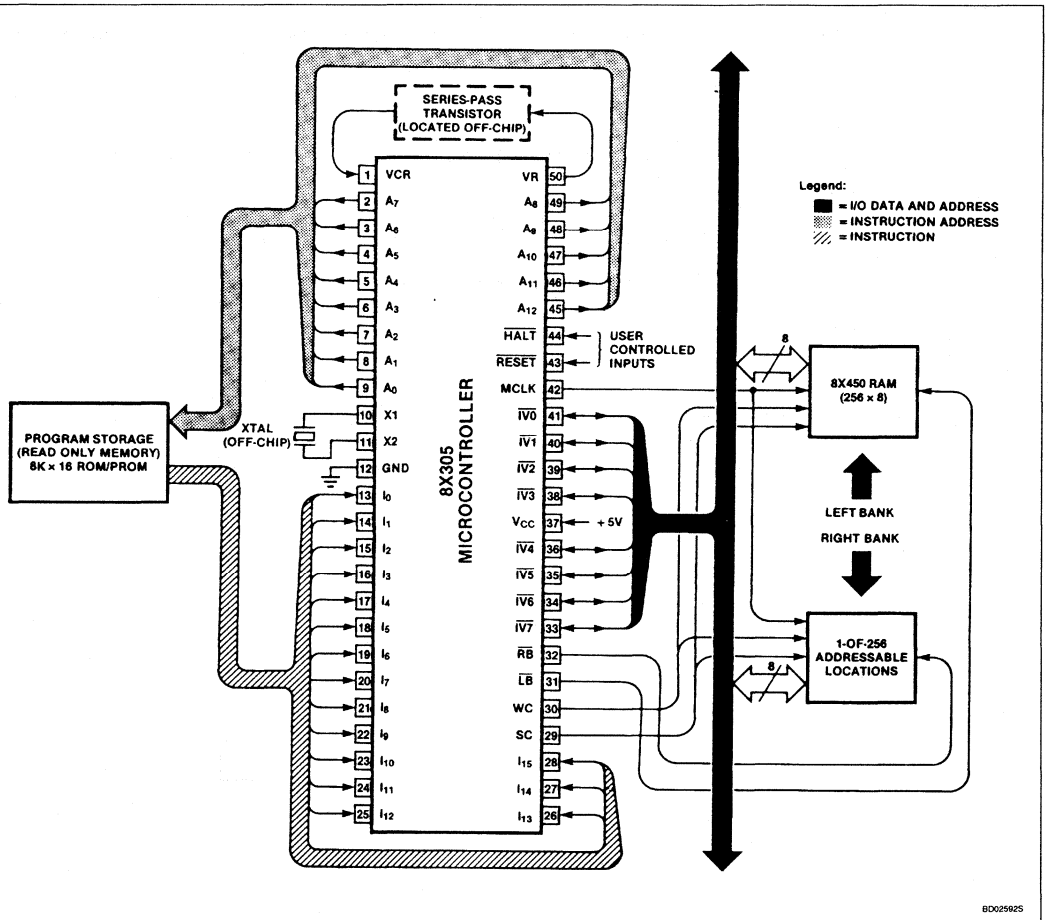


Figure 3. Typical 8X305 System Hookup

BASIC OPERATIONS OF 8X305

Refer to a later discussion of "Instruction Fields" for a detailed examination of all operand fields and subdivisions thereof — "S" (S_0, S_1), "D" (D_0, D_1), "R", "L", "J", and "A".

MOVE OPERATIONS

REGISTER-TO-REGISTER

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
R0-R17 as specified by "S" field of instruction.	Right rotate as specified by "R" field of instruction.	No operation.	No operation.	R0-R7, R11-R17 as specified by "D" field of instruction.

REGISTER-TO-$\overline{IV}$ BUS

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
R0-R17 as specified by "S" field of instruction.	No operation.	No operation.	Shift and merge as specified by "D_0" and "L" fields of instruction.	Variable length field of $\overline{IV}$ bus — Left Bank ($\overline{LB}$) or Right Bank ($\overline{RB}$) as specified by "D_0" and "D_1" fields of instruction.

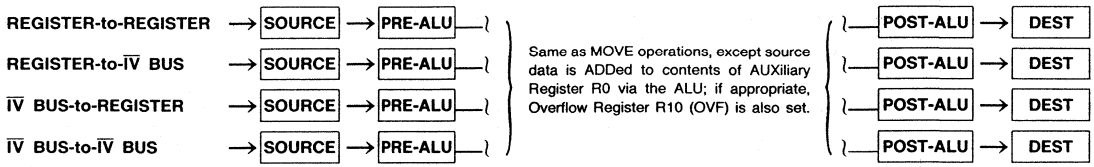
$\overline{IV}$ BUS-TO-REGISTER

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
Variable length field of $\overline{IV}$ bus — Left Bank ($\overline{LB}$) or Right Bank ($\overline{RB}$) as specified by "S_1" and "S_0" fields of instruction.	Right rotate and mask as specified by "S_0" and "L" fields of instruction.	No operation.	No operation.	R0-R7, R11-R17 as specified by "D" field of instruction.

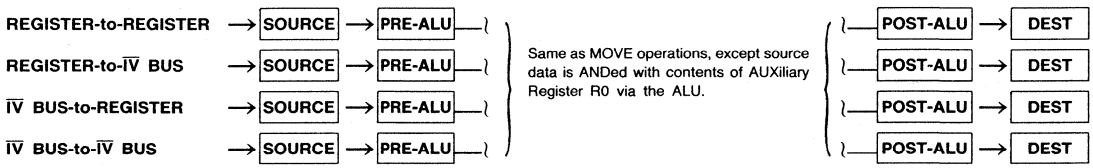
$\overline{IV}$ BUS-TO-$\overline{IV}$ BUS

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
Variable length field of $\overline{IV}$ bus — Left Bank ($\overline{LB}$) or Right Bank ($\overline{RB}$) as specified by "S_1" and "S_0" fields of instruction.	Right rotate and mask as specified by "S_0" and "L" fields of instruction.	No operation.	Shift and merge as specified by "D_0" and "L" fields of instruction.	Variable length field of $\overline{IV}$ bus — Left Bank ($\overline{LB}$) or Right Bank ($\overline{RB}$) as specified by "D_0" and "D_1" fields of instruction.

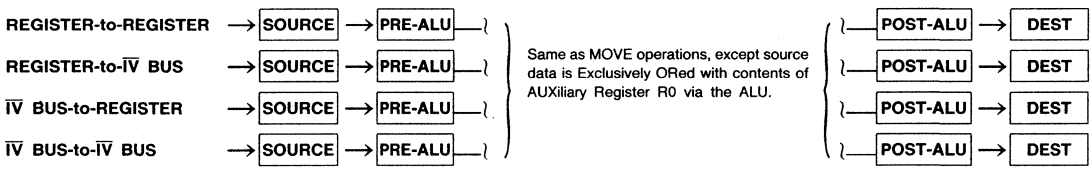
ADD OPERATIONS



AND OPERATIONS



EXCLUSIVE OR (XOR) OPERATIONS



EXECUTE (XEC) OPERATIONS

XEC, REGISTER

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
R0 – R17 as specified by "S" field of instruction.	No operation.	Add source data to 8-bit field specified by instruction literal ($0 \leqslant J \leqslant 377_8$).	No operation.	Replace 8 LSB of Address Register* with 8-bit sum from ALU. *PGM CTR unchanged.

XEC, $\overline{IV}$ BUS

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
Left or Right Bank of $\overline{IV}$ bus as specified by "S" field of instruction.	Rotate and mask as specified by "S_0" and "L" fields of instruction.	Add masked field of source data to 5-bit literal specified by "J" field of instruction ($0 \leqslant J \leqslant 37_8$).	No operation.	Replace 5 LSB of Address Register* with 5-bit sum from ALU. *PGM CTR unchanged.

NON-ZERO TRANSFER (NZT) OPERATIONS

NZT, REGISTER

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DESTINATION
R0-R17 as specified by "S" field of instruction.	No operation.	Test contents of source register for all zeroes.	If S = 0, increment PC by 1; if S ≠ 0, replace 8 LSB of AR and PC with literal specified by "J" field of instruction.

NZT, $\overline{IV}$ BUS

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DESTINATION
Left or Right Bank of $\overline{IV}$ bus as specified by "S" field of instruction.	Rotate and mask as specified by "S_0" and "L" fields of instruction.	Test contents of masked field for all zeroes.	If S = 0, increment PC by 1; if S ≠ 0, replace 5 LSB of AR and PC with literal specified by "J" field of instruction.

TRANSMIT (XMIT) OPERATIONS

XMIT, REGISTER

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
$0 \leqslant J \leqslant 377_8$ – value specified by "J" field of instruction.	No operation.	No operation.	No operation.	R0-R7, R11, R14-R17. Load 8-bit integer specified by "J" field into register specified by "D" field.

XMIT 8-BIT IMMEDIATE, $\overline{IV}$ BUS

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
$0 \leqslant J \leqslant 377_8$ – value specified by "J" field of instruction.	No operation.	No operation.	No operation.	Left (R12) or Right (R13) Bank of $\overline{IV}$ bus as specified by "D" field of instruction.

XMIT VARIABLE-BIT FIELD IMMEDIATE $\overline{IV}$ BUS

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
$0 \leqslant J \leqslant 37_8$ – value specified by "J" field of instruction.	No operation.	No operation.	Shift and merge source data as specified by "D_0" and "L" fields of instruction.	Left or Right Bank of $\overline{IV}$ bus as specified by "D" field of instruction.

Table 1.

$\overline{LB}$	$\overline{RB}$	FUNCTION
Low	Low	This state is not generated by the 8X305.
Low	High	Enable left bank devices.
High	Low	Enable right bank devices.
High	High	Disable all devices; $\overline{IV}$ bus is 3-State.

Table 2.

$\overline{LB}/\overline{RB}$	SC	WC	FUNCTION
High	Low	Low	The $\overline{IV}$ bus is 3-State and not looking for input data.
Low	Low	Low	The $\overline{IV}$ bus is reading input data.
Low	Low	High	Data is being output.
Low	High	Low	Address is being output.
X	High	High	This condition is never generated.

Program Storage Interface

As shown in Figure 3, program storage is connected to output address lines A_0 through A_{12} (A_{12} = LSB) and input instruction lines I_0 through I_{15}. An address output on A_0/A_{12} identifies one 16-bit instruction word in program storage. The instruction word is subsequently input on I_0/I_{15} and defines the Micro-Controller operation which is to follow — one instruction word equals one completed operation. Any TTL-compatible memory can be used for program storage provided the worst-case access time is compatible with the instruction cycle time used for the application — see timing section for appropriate calculations.

I/O Interface and Control

An 8-bit bidirectional I/O bus, referred to as the Interface Vector ($\overline{IV}$) bus, provides a communication link between the Microcontroller and the two banks of I/O devices. The $\overline{LB}$ (Left Bank) and $\overline{RB}$ (Right Bank) control signals identify which bank is enabled; when both $\overline{LB}$ and $\overline{RB}$ are high (inactive), neither bank is enabled and the $\overline{IV}$ bus is inactive (three-state). A functional analysis of the Left and Right Bank signals is shown in Table 1.

Both data and I/O address information are multiplexed on the $\overline{IV}$ bus. The SC (Select Command) and WC (Write Command) signals distinguish between data and I/O address information as shown in Table 2.

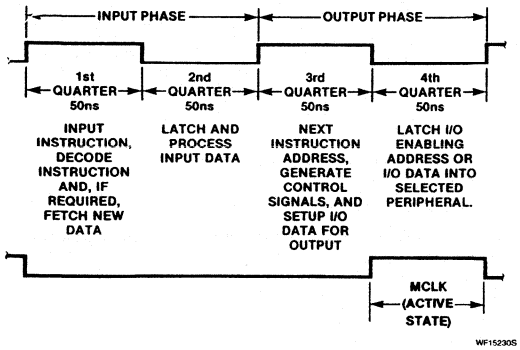
Data Processing

Basically, the data processing path of the 8X305 consists of the Rotate/Mask logic, the Arithmetic Logic Unit (ALU), the Shift/Merge functions, on-chip memory (sixteen 8-bit registers), and the bidirectional $\overline{IV}$ bus interface with its associated driver circuits and internal latches. The on-board memory and the $\overline{IV}$ bus are connected to both inputs and outputs of the ALU via internal 8-bit data paths — see Figure 1. Inputs to the ALU are preceded by right-rotate and data-mask functions; the ALU output is followed by the left-shift and merge operations. Depending on the desired operation, any one or all of the functions (Rotate/Mask/Shift/Merge) can operate on 8 bits of data in a single instruction cycle. For a summary of all data-processing capabilities, refer to BASIC OPERATIONS OF THE 8X305 described earlier in this data sheet.

Instruction Cycle

Each operation of the 8X305 is executed in a single instruction cycle. The instruction cycle is internally divided into four equal parts — each part being as short as 50ns. Figure 4 shows the general functions that occur during each quarter cycle; specifics regarding minimum/maximum timing and other critical values are described later in this data sheet. During the first quarter cycle, a new instruction from program storage is input via $I_0 - I_{15}$ and decoded. If an I/O operation is indicated, new data is fetched from a specified internal register or via the $\overline{IV}$ bus. At the end of the first quarter cycle, the new instruction is latched into the instruction register.

In the second quarter cycle, the I/O input data stabilizes and preliminary processing is completed. At the end of this quarter, the $\overline{IV}$ latches close and final processing can be accomplished, thus completing the input phase of the instruction cycle. During the third quarter cycle, the address for the next instruction is output to the instruction address bus, $\overline{IV}$ control signals are generated, and both data and destination are setup for the remainder of the output phase. During the fourth quarter cycle, a master clock signal (MCLK) generated by the 8X305 is used to latch either the I/O-enabling address or the I/O data into peripheral devices connected to the $\overline{IV}$ bus. MCLK can also be used to synchronize any external logic with timing circuits of the 8X305. To summarize the action, the first half of the instruction cycle deals primarily with input functions and the second half is mostly concerned with output functions.



NOTES:
1. New instruction must be accepted and latched at end of first quarter cycle.
2. The I/O data latches are open for the first two quarter cycles, that is, for 100ns.
3. The address changes during third quarter cycle.
4. $\overline{IV}$ bus drivers are active (turned on) during third and fourth quarter cycles.

Figure 4. Instruction Cycle and MCLK with: Crystal = 10MHz and Cycle Time = 200ns.

Table 3.

BIT POSITIONS →	↓ MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB ↓ 15
	OPCODE			OPERAND(S)												

Table 4.

- **Data Control —**
  - **ADD, AND, XOR** } **Arithmetic and Logic Operations**
  - **MOVE, XMIT** } **Movement of Data and Constants**
- **Program Control —**
  - **XEC, NZT, JMP** } **Branch or Test**

INSTRUCTION SET

General Format and Operating Principles

The 16-bit instruction word (I_0 through I_{15}) from program storage is input to the instruction register (Figure 1) and is subsequently decoded to implement the events to occur during the current instruction cycle.

The general format for each instruction word is shown in Table 3.

The 3-bit operation code (OPCODE) define any one of eight classes of instructions; variations within each class are specified by the remaining thirteen operand bits. The eight instruction classes can be separated into two control areas — *data* and *program*; general functions within these areas are as shown in Table 4.

Instruction Fields

As shown in Table 5, each instruction word consists of an operation code (OPCODE) field and from one to three operand fields. The possible operand fields are: Source (S), Destination (D), Rotate/Length (R/L), Literal (J), and Address (A). The OPCODE and operand fields are described in the paragraphs that follow the table.

Table 5. Functional Description of Instruction Set

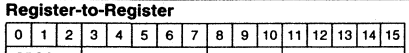
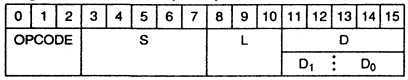
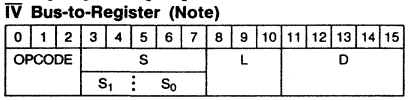
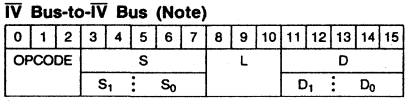
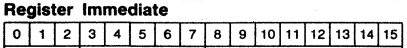
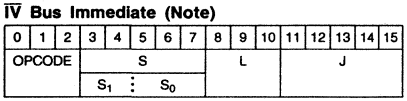
INSTRUCTION WORD	DESCRIPTION	STATE OF CONTROL SIGNAL DURING INSTRUCTION CYCLE — SEE FIGURE 4		
		CONTROL SIGNAL	INPUT PHASE	OUTPUT PHASE
CLASS = **MOVE** OPCODE = **0** OPERATION = **(S) → D**				
Register-to-Register  $S = 00_8 - 17_8$ $D = 00_8 - 07_8$, $11_8 - 17_8$	Move content of internal register specified by S-field to internal register specified by D-field. Prior to the "MOVE" operation, right-rotate contents of internal source register by octal value (0 through 7) defined by the R-field.	SC	L	H if $D = 07_8$, 17_8
		WC	L	L
		$\overline{LB}$	H	L if $D = 07_8$
		$\overline{RB}$	H	L if $D = 17_8$
Register-to-$\overline{IV}$ Bus (Note)  $S = 00_8 - 17_8$ $D = 20_8 - 37_8$	Move contents of internal register specified by the S-field to the $\overline{IV}$ bus. Before outputting on $\overline{IV}$ bus, data is shifted as specified by the least significant octal digit of the D-field and the bits specified by the L-field are merged with the latched I/O data.	SC	L	L
		WC	L	H
		$\overline{LB}$	L if $D = 20_8 - 27_8$	L if $D = 20_8 - 27_8$
		$\overline{RB}$	L if $D = 30_8 - 37_8$	L if $D = 30_8 - 37_8$
$\overline{IV}$ Bus-to-Register (Note)  $S = 20_8 - 37_8$ $D = 00_8 - 07_8$, $11_8 - 17_8$	Move right-rotated $\overline{IV}$ bus (source) data specified by the S-field to internal register specified by the D-field. The L-field specifies the length of source data starting from the LSB-position and, if less than 8 bits, the remaining bits are filled with zeros.	SC	L	H if $D = 07_8$, 17_8
		WC	L	L
		$\overline{LB}$	L if $S = 20_8 - 27_8$	L if $D = 07_8$
		$\overline{RB}$	L if $S = 30_8 - 37_8$	L if $D = 17_8$
$\overline{IV}$ Bus-to-$\overline{IV}$ Bus (Note)  $S = 20_8 - 37_8$ $D = 20_8 - 37_8$	Move right-rotated $\overline{IV}$ bus (source) data specified by the S-field to the I/O latches. Before outputting on $\overline{IV}$ bus, shift data as specified by the D-field; then merge source and latched I/O data as specified by the L (length) field.	SC	L	L
		WC	L	H
		$\overline{LB}$	L if $S = 20_8 - 27_8$	L if $D = 20_8 - 27_8$
		$\overline{RB}$	L if $S = 30_8 - 37_8$	L if $D = 30_8 - 37_8$
CLASS = **ADD** OPCODE = **1** OPERATION = **(S) + (AUX) → D**				
Same as MOVE instruction class	Same as MOVE instruction class except that contents of AUX (R0) register are ADDed to the source data. If there is a "carry" from MSB, then R10 (OVF) = 1 (overflow), otherwise OVF = 0.	Same as MOVE instruction class		
CLASS = **AND** OPCODE = **2** OPERATION = **(S) ∧ (AUX) → D**				
Same as MOVE instruction class	Same as MOVE instruction class except that contents of AUX (R0) register are ANDed with source data.	Same as MOVE instruction class		
CLASS = **XOR** OPCODE = **3** OPERATION = **(S) ⊕ (AUX) → D**				
Same as MOVE instruction class	Same as MOVE instruction class except that contents of AUX (R0) register are Exclusively ORed with source data.	Same as MOVE instruction class		
CLASS = **XEC** OPCODE = **4** OPERATION = **Refer to Description**				
Register Immediate  $S = 00_8 - 17_8$ $J = 000_8 - 377_8$	Execute instruction at current page address offset by J (literal) + (S). Return to normal instruction flow unless a branch is encountered. Execute instruction at an address determined by replacing the low-order 8 bits of the Address Register with the following derived sum: Value of literal (J-field) plus contents of internal register specified by S-field The PC is not incremented and the overflow status (OVF) is not changed.	SC	L	L
		WC	L	L
		$\overline{LB}$	H	H
		$\overline{RB}$	H	H
$\overline{IV}$ Bus Immediate (Note)  $S = 20_8 - 37_8$ $J = 00_8 - 37_8$	Execute instruction at an address determined by replacing the low-order 5 bits of Address Register with the following derived sum: 5-bit value of literal (J-field) plus value of rotated source data specified by S-field. The L-field specifies the length of source data starting from the LSB position and, if less than 8 bits, the remaining bits are filled with zeros; the Program Counter is not incremented and the overflow status (OVF) is not changed.	SC	L	L
		WC	L	L
		$\overline{LB}$	L if $S = 20_8 - 27_8$	H
		$\overline{RB}$	L if $S = 30_8 - 37_8$	H

Table 5. Functional Description of Instruction Set (Continued)

INSTRUCTION WORD	DESCRIPTION	STATE OF CONTROL SIGNAL DURING INSTRUCTION CYCLE — SEE FIGURE 4		
		CONTROL SIGNAL	INPUT PHASE	OUTPUT PHASE
CLASS = **NZT** OPCODE = **5** OPERATION = **Refer to Description**				
Register Immediate 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | S | J $S = 00_8 - 17_8$ $J = 000_8 - 377_8$	If data specified by the S-field is not equal to zero, jump to current page address offset by value of J-field; otherwise, increment the Program Counter. If contents of internal register specified by S-field is non-zero, transfer to address determined by replacing the low-order 8 bits of Address Register and Program Counter with "J", otherwise, increment PC.	SC	L	L
		WC	L	L
		$\overline{LB}$	H	H
		$\overline{RB}$	H	H
$\overline{IV}$ Bus Immediate (Note) 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | S | L | J S_1 ⋮ S_0 $S = 20_8 - 37_8$ $J = 00_8 - 37_8$	If right-rotated and masked IV bus is non-zero, transfer to address determined by replacing low-order 5 bits of Address Register and Program Counter with "J", otherwise, increment PC. (The L-field specifies the length of source I/O data starting from the LSB-position and, if less than 8 bits, the remaining bits are filled with zeros.)	SC	L	L
		WC	L	L
		$\overline{LB}$	L if $S = 20_8 - 27_8$	H
		$\overline{RB}$	L if $S = 30_8 - 37_8$	H
CLASS = **XMIT** OPCODE = **6** OPERATION = **J → D**				
XMIT, Register 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | D | J $D = 00_8 - 06_8$, 11_8, $14_8 - 16_8$ $J = 000_8 - 377_8$	Store 8-bit value specified by "J" into register specified by "D".	SC	L	L
		WC	L	L
		$\overline{LB}$	H	H
		$\overline{RB}$	H	H
XMIT, $\overline{IV}$ Bus Address 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | D | J $D = 07_8$, 17_8 $J = 000_8 - 377_8$	Enable I/O device on the bank specified by "D", whose address is the 8-bit integer specified by "J". Address "J" is stored in register "D".	SC	L	H
		WC	L	L
		$\overline{LB}$	H	L if $D = 07_8$
		$\overline{RB}$	H	L if $D = 17_8$
XMIT 8 Bits Immediate, $\overline{IV}$ Bus (Note) 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | D | J $D = 12_8 - 13_8$ $J = 000_8 - 377_8$	Store value of 8-bit integer in the previously enabled I/O port, at the bank destination ($\overline{LB}$ or $\overline{RB}$) specified by "D". Contents of R12 or R13 remain unchanged.	SC	L	L
		WC	L	H
		$\overline{LB}$	H	L if $D = 12_8$
		$\overline{RB}$	H	L if $D = 13_8$
XMIT Variable Bit Field Immediate, $\overline{IV}$ Bus (Note) 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | D | L | J D_1 ⋮ D_0 $D = 20_8 - 37_8$ $J = 00_8 - 37_8$	Transmit Least Significant "L" bits of "J" field to "L-bit" field of $\overline{IV}$ bus specified by "D"; if "L" is greater than 5 bits, the MSB bits of destination field is filled with zeros.	SC	L	L
		WC	L	H
		$\overline{LB}$	L if $D = 20_8 - 27_8$	L if $D = 20_8 - 27_8$
		$\overline{RB}$	L if $D = 30_8 - 37_8$	L if $D = 30_8 - 37_8$
CLASS = **JMP** OPCODE = **7** OPERATION = **Refer to Description**				
Address Immediate 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | A $A = 00000_8 - 17777_8$	Jump to address in program storage specified by A-field; this address is loaded into the Address Register and the Program Counter.	SC	L	L
		WC	L	L
		$\overline{LB}$	H	H
		$\overline{RB}$	H	H

NOTE:

S_0 specifies the LSB of rotated input data field

S_1 specifies the bank of $\overline{IV}$ bus from which source data will be input

D_0 specifies bit position in I/O device with which LSB of processed data will be aligned, and

D_1 specifies the bank of $\overline{IV}$ bus which will be the destination.

Table 6. Octal Addresses and Source/Destination Fields for 8X305 Registers

ADDRESS	REGISTER DESIGNATION	SOURCE	DESTI-NATION	ADDRESS	REGISTER DESIGNATION	SOURCE	DESTI-NATION
00_8	R0 (AUX) — General purpose register	X	X	10_8	R10 (OVF — Overflow register)	X	
01_8	R1 — General purpose register	X	X	11_8	R11 — General purpose register	X	X
02_8	R2 — General purpose register	X	X	12_8	R12 — General purpose register (Note)	X	X
03_8	R3 — General purpose register	X	X	13_8	R13 — General purpose register (Note)	X	X
04_8	R4 — General purpose register	X	X	14_8	R14 — General purpose register	X	X
05_8	R5 — General purpose register	X	X	15_8	R15 — General purpose register	X	X
06_8	R6 — General purpose register	X	X	16_8	R16 — General purpose register	X	X
07_8	R7 — Special purpose register (refer to next paragraph)	X	X	17_8	R17 — Special purpose register (refer to next paragraph)	X	X

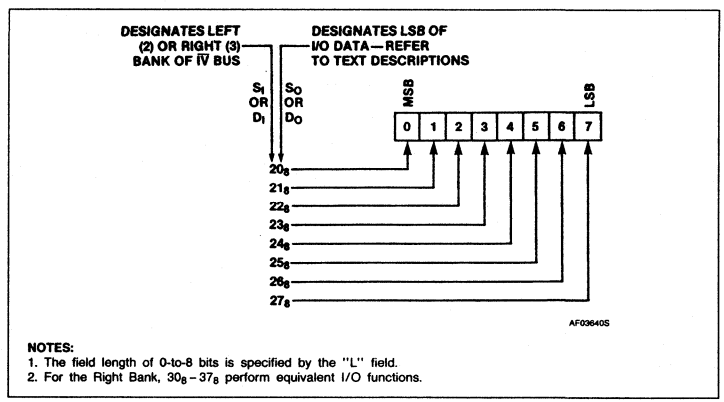
NOTE:
R12 and R13 function as general purpose working registers for all operations except transmit (XMIT). During a transmit instruction where R12 or R13 is the destination, the 8-bit "J" field is immediately transferred to the $\overline{IV}$ bus; for this operation, the contents of the designated register remain unchanged.

Operations Code Field. The 3-bit OPCODE field specifies one of eight classes of 8X305 instructions; octal designations for this field and operands for each instruction class are shown in Table 5.

Source (S) and Destination (D) Fields. The 5-bit "S" and "D" fields specify the source and destination, respectively, for whatever operation is defined by the OPeration CODE. The "S" and/or "D" fields can specify an internal 8X305 register or any one-to-eight bit field within an I/O device; octal values and source/destination field assignments for all internal registers are shown in Table 6.

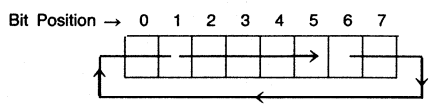
In instructions where $R7_8$ (IVL) or $R17_8$ (IVR) is specified as the destination, the 8-bit value is output on the $\overline{IV}$ bus as an I/O device address or memory location; register R7 selects the Left Bank and register R17 selects the Right Bank. The results are also stored into the specified internal register ($R7_8$ or $R17_8$) and may later be accessed as source data. When the $\overline{IV}$ bus is specified as a source and/or destination, the "S" and "D" fields are split into two parts, that is,

- Source (S) = S_1, S_0 and Destination (D) = D_1, D_0 where,
  S_0 specifies the LSB of rotated input data field
  S_1 specifies the bank of $\overline{IV}$ bus from which source data will be input
  D_0 specifies bit position in I/O device with which LSB of processed data will be aligned and
  D_1 specifies the bank of $\overline{IV}$ bus which will be the destination.



NOTES:
1. The field length of 0-to-8 bits is specified by the "L" field.
2. For the Right Bank, $30_8 - 37_8$ perform equivalent I/O functions.

RIGHT-ROTATE FUNCTION



Rotate (R) and Length (L) Field. The 3-bit R/L field performs one of two functions, specifying either the field length (L) for I/O operations or a right-rotate (R) for internal operations. For a given instruction, the specified function depends upon the contents of the Source (S) and Destination (D) fields.

When an internal register is specified by both the source and destination fields, the "R" field is invoked and it specifies a right-rotate of the data specified in the "S" field (see accompanying diagram.) The source-register data (up to 8 bits) is right-rotated during the "input phase" of the instruction cycle (Figure 4). This function is always performed prior to any ALU operation. (*Note: The right-rotate function is implemented on the bus and not in the source register.*)

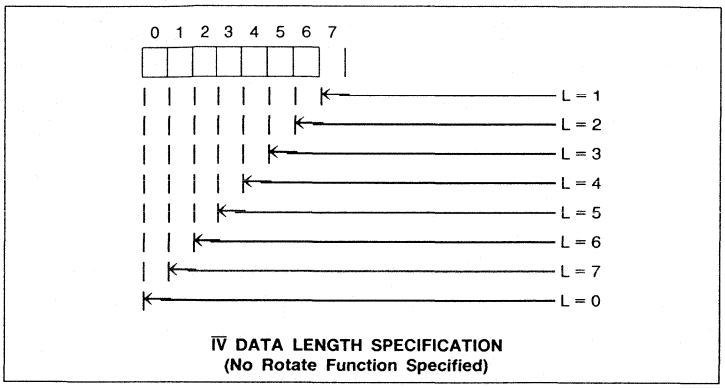
When either or both of the source and destination fields specify a variable-length I/O

data field, the "L" field specifies the length of the I/O data field (see following diagram). If the source field specifies an $\overline{IV}$ address ($20_8 - 37_8$) and the destination field specifies an internal register ($00_8 - 07_8$, $11_8 - 17_8$), the "L" field specifies the length of source data; the source data is formed by right-rotating the $\overline{IV}$ bus data according to the source address and then masking result as specified by the "L" field. If length is less than 8 bits, all remaining bits are set to zero prior to processing data in the ALU. If the source field specifies an internal register ($00_8 - 17_8$) and the destination field specifies $\overline{IV}$ bus data ($20_8 - 37_8$), the "L" field specifies the length of the destination data. To form the destination data, the ALU output is left-shifted according to the destination address and then masked to the required length (see $\overline{IV}$ DATA LENGTH SPECIFICATION). The destination data is merged with data in the I/O latches to finalize the $\overline{IV}$ bus data. Hence, a one-to-eight bit destination data field can be inserted into the existing 8-bit I/O port without modifying surrounding bits. If both the source and destination fields specify $\overline{IV}$ bus data ($20_8 - 37_8$), the "L" field specifies the length of both the source and destination data.

To form the source data, the $\overline{IV}$ bus input data is right-rotated according to the source address and then masked to the required length — see $\overline{IV}$ DATA LENGTH SPECIFICATION. If length is less than 8 bits, all remaining bits are set to zero before processing in the ALU. To form the destination data, the ALU output is left-shifted according to the destination address and masked to the required length specification. The destination data is then merged into the $\overline{IV}$ bus data that was used to obtain the source; thus, if the source and destination addresses are on the same bank, the $\overline{IV}$ bus data written to the destination I/O Port appears unmodified, except for bits changed during the shift-and-mask operations. If the source and destination addresses refer to different banks, the destination I/O Port is changed to contain the contents of the source I/O Port in those bit positions not affected by the destination data.

J Field. The 5-bit or 8-bit "J" field is used to load a literal value (contained in the instruction) into a register, into a variable I/O data field, or to modify the low-order bits of the Program Counter. The bit length of the "J" field is implied by the "S" and "L" fields in the XEC, NZT, and XMIT instructions, based on the following conditions:

- When the Source (S) field specifies an internal register, the literal value of the "J" field is an 8-bit binary number.
- When the Source (S) field specifies a variable I/O data field, the literal value of the "J" field is a 5-bit binary number.



$\overline{IV}$ DATA LENGTH SPECIFICATION
(No Rotate Function Specified)

A Field. The 13-bit "A" field is an address field which allows the 8X305 to directly branch to any of the 8192 locations in Program Storage memory.

Formation of Instruction Address

The Address Register and Program Counter are used to generate addresses for accessing an instruction from program storage. The instruction address is formed in one of the following ways:

- For all except the JMP, XEC, and a "satisfied" NZT instruction, the Program Counter is incremented by one and placed in the Address Register.
- For the JMP instruction, the 13-bit "A" field contained in the JMP instruction word replaces the contents of both the Address Register and the Program Counter.
- For the XEC instruction, the Address Register is loaded with bits from the Program Counter modified as follows:
  XEC using $\overline{IV}$ Bus Data — low-order 5 bits of ALU output replaces counterpart bits in Address Register.

  XEC using Data from Internal Register — low-order 8 bits of ALU output replaces counterpart bits in Address Register.

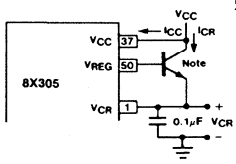
The Program Counter is not modified for either of the above conditions.

- For a "satisfied" NZT instruction, the low-order 5 bits (NZT source is $\overline{IV}$ bus data) or low-order 8 bits (NZT source is an internal register) of both the Address Register and Program Counter are loaded with the literal value specified by the "J" field of instruction word.

Data Addressing

The source and/or destination addresses of the data to be operated upon are specified as part of the instruction word. As shown earlier, source/destination addresses are specified using a 5-bit code ($00_8 - 37_8$). When the most significant octal digit is a "0" or "1", the source and/or destination address is an internal register; if the most significant digit is a 2 or 3, an $\overline{IV}$ bus operation is indicated — 2 specifying a Left-Bank ($\overline{LB}$) operation and 3 specifying a Right-Bank ($\overline{RB}$) operation. The least significant octal digit (0 through 7) indicates either a specific internal register address or positioning information for the least significant bit when specifying $\overline{IV}$ bus data. Referring to Table 5, AUXiliary register R0 (00_8) is the implied source of the second argument for the ADD, AND, and XOR operations. IVL register R7 and IVR register R17 (destination addresses 07_8 and 17_8, respectively) provide a means of routing enabling address information to I/O peripherals. With IVL or IVR specified as the destination address, data is placed on the $\overline{IV}$ bus during the output phase of the instruction cycle; simultaneously, a Select Command (SC) is generated to inform all I/O devices that information on the $\overline{IV}$ bus is to be considered as an I/O address. Since the contents of IVL and IVR are preserved, either register may later be accessed as a source of data.

Control outputs $\overline{LB}$ and $\overline{RB}$ are used to partition I/O bus devices into two fields of 256 addresses. With $\overline{LB}$ in the active-low state and a source address of $20_8 - 27_8$, the left bank of I/O devices are enabled during the input phase of the instruction cycle. With $\overline{RB}$ in the active-low state and a source address of $30_8 - 37_8$, the right bank of devices are enabled. During the output phase, $\overline{LB}$ is low if the destination address is 07_8 or $20_8 - 27_8$, whereas $\overline{RB}$ is low if the destination address is 17_8 or $30_8 - 37_8$. Each address field ($\overline{LB}$ and $\overline{RB}$) can have a different I/O device selected, that is, data can be transferred from a device in one bank to a device in the other in one instruction cycle.



PARAMETER	CONDITIONS	LIMITS
h_{fe}	V_{CE} = 2V; 100mA < I_C < 500mA	> 50
VBE_{ON}	V_{CE} = 5V; I_C = 500mA	< 1V
V_{CESAT}	I_C = 500mA; I_B = 50mA	< 0.5V
BV_{CEO}		> 15V
f_t		> 30MHz

LD05661S

NOTE:
Typical approved parts — 2N5320, 2N5337

Voltage Regulator

DESIGN PARAMETERS

Hardware design of an 8X305-based system largely consists of the following operations:

- Selecting and interfacing a Program Storage device — ROM, PROM, etc.
- Selecting and interfacing input/output devices — RAM, Ports, and other 8-bit addressable I/O devices.
- Choosing and implementing System Clock — Capacitor-Controlled, Crystal-Controlled, or Externally-Driven.
- Selection of an off-chip series-pass transistor.

VOLTAGE REGULATOR

All internal logic of the 8X305 is powered by an on-chip voltage regulator that requires an external series-pass transistor. Electrical specifications for the off-chip power transistor and a typical hook-up are shown in the accompanying diagram. To minimize lead inductance, the transistor should be as close as possible to the 8X305 package and the emitter should be AC-grounded via a 0.1μf ceramic capacitor.

All information required for easy implementation of these design requirements is provided under the following captions:

- Ordering Information
- Voltage Regulator
- DC Characteristics
- AC Characteristics
- Timing Considerations
- Clock Considerations
- $\overline{\text{HALT}}/\overline{\text{RESET}}$ Logic

ABSOLUTE MAXIMUM RATINGS Storage Temperature (T_{STG}) rating are from −65°C to +150°C

SYMBOL	PIN	DESCRIPTION	RATING	UNIT
V_{CC}	V_{CC}	Supply voltage	+7.0	V
	X1, X2	Crystal input voltage	2.0	V
	All other pins	Logic input voltage	5.5	V

DC ELECTRICAL CHARACTERISTICS (Commercial Part) $4.75V \leqslant V_{CC} \leqslant 5.25V$, $0°C \leqslant T_A \leqslant 70°C$

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT	COMMENTS
			Min	Typ	Max		
V_{CC}	Supply voltage		4.75	5	5.25	V	
V_{IH}	High-level input voltage		0.9 2		2 5.5	V	X1 and X2 All other pins
V_{IL}	Low-level input voltage				0.5 0.8	V	X1 and X2 All other pins
V_{OH}	High-level output voltage	V_{CC} = min; I_{OH} = −3mA	2.4			V	
V_{OL}	Low-level output voltage	V_{CC} = min; I_{OL} = 6mA V_{CC} = min; I_{OL} = 16mA			0.55 0.55	V	A_0 through A_{12} All other outputs
V_{CR}	Regulator voltage	V_{CC} = 5V		3.1 2.9		V	T_A = 0°C T_A = 70°C
V_{IC}	Input clamp voltage	V_{CC} = min; I_{IN} = −10mA			−1.5	V	Crystal inputs X1 and X2 do not have internal clamp diodes
I_{IH}	High-level input current	V_{CC} = max V_{IH} = 0.9V V_{IH} = 4.5V			4 50	mA μA	X1 and X2 All other pins
I_{IL}	Low-level input current	V_{CC} = max; V_{IL} = 0.4V			−3 −0.2 −1.6 −0.4	mA	X1 and X2 $\overline{IV0} - \overline{IV7}$ I0 − I15 $\overline{HALT}$ and $\overline{RESET}$
I_{OS}	Short circuit output current	V_{CC} = max; (Note: At any time, no more than one output should be connected to ground.)	−30		−140	mA	All output pins
I_{CC}	Supply current	V_{CC} = max			180 195	mA	T_A = 70°C T_A = 0°C
I_{REG}	Regulator control	V_{CC} = 5.0V	−10		−25	mA	Max available base drive for series-pass transistor
I_{CR}	Regulator current	V_{CC} = max			200 230	mA	T_A = 70°C T_A = 0°C

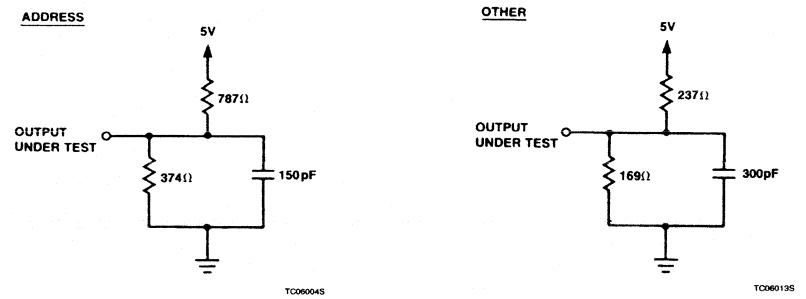
NOTES:
1. Operating temperature ranges are guaranteed after thermal equilibrium has been reached.
2. All voltages measured with respect to ground terminal.

AC ELECTRICAL CHARACTERISTICS (Commercial Part) Conditions: $4.75V \leqslant V_{CC} \leqslant 5.25V$; $0°C \leqslant T_A \leqslant 70°C$
Loading: (See test circuits)

SYMBOL	PARAMETER (NOTE 1)	LIMITS (INSTRUCTION CYCLE TIME = 200ns)			LIMITS (INSTRUCTION CYCLE TIME > 200ns)			UNITS	COMMENTS
		Min	Typ	Max	Min	Typ	Max		
t_{PC}	Processor cycle time	200			200			ns	
t_{CP}	X1 clock period	100			100			ns	
t_{CH}	X1 clock high time	50			50			ns	
t_{CL}	X1 clock low time	50			50			ns	
t_{MCL}	MCLK low delay	15		40	15		40	ns	
t_W	MCLK pulse width	35		55	$T_{4Q} - 15$		$T_{4Q} + 5$	ns	Note 2
t_{DD}	Input data to output data	70		105	70		105	ns	
t_{MHS}	MCLK falling edge to $\overline{HALT}$ falling edge			30			$T_{1Q} - 20$	ns	Note 2
t_{MHH}	$\overline{HALT}$ hold time (MCLK falling edge)	65			$T_{1Q} + 15$			ns	Note 2
t_{ACC}	Program storage access time			60				ns	
t_{IO}	I/O port output enable time ($\overline{LR}/\overline{RB}$ to valide $\overline{IV}$ data input)			30				ns	
t_{MAS}	MCLK falling edge to address stable			140			$T_{1Q} + T_{2Q} + 40$	ns	Notes 2, 3, & 4
t_{IA}	Instruction to address			140			$T_{2Q} + 90$	ns	Notes 2, 3 & 5
t_{IVA}	Input data to address			85			85	ns	Notes 3 & 6
t_{MIS}	MCLK falling edge to instruction stable			25			$T_{1Q} - 25$	ns	Notes 2 & 7
t_{MIH}	Instruction hold time (MCLK falling edge)	55			$T_{1Q} + 5$			ns	Notes 2 & 8
t_{MWH}	MCLK falling edge to SC/$\overline{WC}$ rising edge	105		125	$T_{1Q} + T_{2Q} + 5$		$T_{1Q} + T_{2Q} + 25$	ns	Note 2
t_{MWL}	MCLK falling edge to SC/$\overline{WC}$ falling edge	2		15	2		15	ns	
t_{MIBS}	MCLK falling edge to $\overline{LB}/\overline{RB}$ (Input phase)	5		25	5		25	ns	
t_{IIBS}	Instruction to $\overline{LB}/\overline{RB}$ (Input phase)			25			25	ns	
t_{MOBS}	MCLK falling edge to $\overline{LB}/\overline{RB}$ (Output phase)	115		145	$T_{1Q} + T_{2Q} + 15$		$T_{1Q} + T_{2Q} + 45$	ns	Note 2
t_{MIDS}	MCLK falling edge to input data stable			55			$T_{1Q} + T_{2Q} - 45$	ns	Note 2
t_{MIDH}	Input data hold time (MCLK falling edge)	115			$T_{1Q} + T_{2Q} + 15$			ns	Note 2
t_{MODH}	Output data hold time (MCLK falling edge)	11			11			ns	
t_{MODS}	Output data stable (MCLK falling edge)	123		150	$T_{1Q} + T_{2Q} + 23$		$T_{1Q} + T_{2Q} + 50$	ns	Note 2
t_{ODSM}	Output data stable (MCLK rising edge)	10			$t_{3Q} - 40$			ns	Note 2

NOTES:
1. X1 and X2 inputs are driven by an external pulse generator with an amplitude of 1.5 volts; all timing parameters are measured at this voltage level.
2. Respectively, T_{1Q}, T_{2Q}, T_{3Q}, and T_{4Q} represent time intervals for the first, second, third, and fourth quarter cycles.
3. Capacitive loading for the address bus is 150 picofarads.
4. T_{MAS} is obtained by forcing a valid instruction and an I/O bus input to occur earlier than the specified minimum set up time.
5. T_{IA} is obtained by forcing a valid instruction input to occur earlier than the minimum set up time.
6. T_{IVA} is obtained by forcing a valid I/O bus input to meet the minimum set up time.
7. T_{MIS} represents the setup time required by internal latches of the 8X305. In system applications, the instruction input may have to be valid before the worst-case set up time in order for the system to respond with a valid I/O bus input that meets the I/O bus input set up time (T_{IDS} and T_{MIDS}).
8. T_{MIH} represents the hold time required by internal latches of the 8X305. To generate proper $\overline{LB}/\overline{RB}$ signals, the instruction must be held valid until the address bus changes.

AC TEST CIRCUITS



NOTE:
Load capacitance includes Test Jig and Probe Capacitance

TIMING CONSIDERATIONS (Commercial Part)

As shown in the AC CHARACTERISTICS table for the commercial part, the minimum instruction cycle time is 200ns; whereas, the maximum is determined by the on-chip oscillator frequency and can be any value the user chooses. With an instruction cycle time of 200ns, the part can be characterized in terms of absolute values; these are shown in the first "LIMITS" column of the table. When the instruction cycle time is greater than 200ns, certain parameters are cycle-time dependent; thus, these parameters are specified in terms of the four quarter cycles (T_{1Q}, T_{2Q}, T_{3Q}, and T_{4Q}) that make up one instruction cycle — see 8X305 TIMING DIAGRAM. As the time interval for each instruction cycle increases (becomes greater than 200ns), the delay for all parameters that are cycle-time dependent is likewise increased. In some cases, these delays have a significant impact on timing relationships and other areas of systems design; subsequent paragraphs describe these timing parameters and reliable methods of calculation.

Timing parameters for the 8X305 are normally measured with reference to MCLK.

System determinants for the instruction cycle time are:

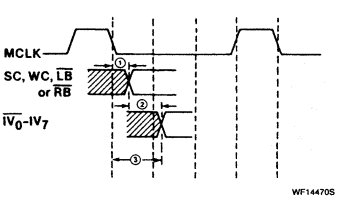
- Propagation delays within the 8X305
- Access time of Program Storage
- Enable time of the I/O port

Normally, the instruction cycle time is constrained by one or more of the following conditions:

Condition 1 — Instruction or MCLK to $\overline{LB}/\overline{RB}$ (input phase) plus I/O port access time (TIO) $\leqslant \overline{IV}$ data setup time (Figure 5a).

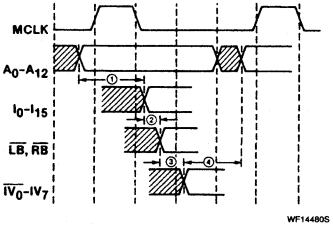
Condition 2 — Program storage access time (TACC) plus instruction to $\overline{LB}/\overline{RB}$ (input phase) plus I/O port access time (TIO) plus $\overline{IV}$ data (input phase) to address $\leqslant$ instruction cycle time (Figure 5b).

Condition 3 — Program storage access time plus instruction to address $\leqslant$ instruction cycle time (Figure 5c).



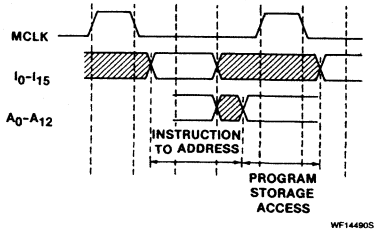
NOTES:
① MCLK to $\overline{LB}/\overline{RB}$ (input phase) or instruction to $\overline{LB}/\overline{RB}$ (input phase).
② I/O port access (TIO).
③ $\overline{IV}$ data setup time (referenced to MCLK).

a. Condition #1



NOTES:
① Program storage access time.
② MCLK to $\overline{LB}/\overline{RB}$ (input phase) or instruction to $\overline{LB}/\overline{RB}$ (input phase).
③ I/O port access (TIO).
④ $\overline{IV}$ data (input phase to address.)

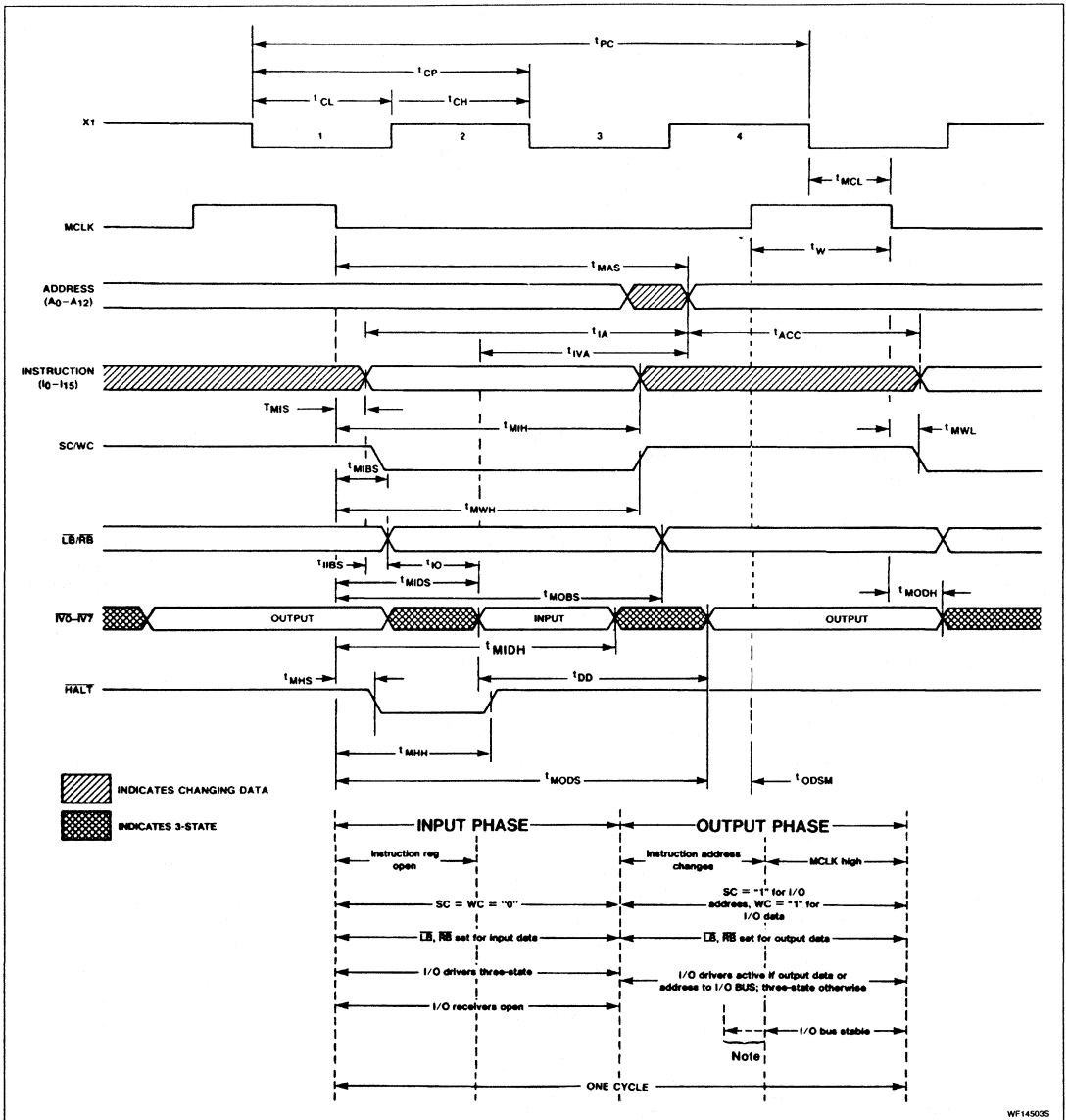
b. Condition #2



c. Condition #3

Figure 5. Constraints of 8X305 Instruction Cycle Time

8X305 TIMING DIAGRAM



NOTE:
For an instruction cycle time greater than 200ns, the I/O bus can be stable sometime within the third quarter (T_{3Q}) cycle.

From condition #1 and with an instruction cycle time of 200ns, the I/O port access time (TIO) can be calculated as follows:

$$\begin{aligned} & TMIBS + TIO \leqslant TMIDS \\ \text{transposing,}\ & TIO \leqslant TMIDS = TMIBS \\ \text{substituting,}\ & TIO \leqslant 55ns - 25ns \\ \text{result,}\ & TIO \leqslant 30ns \end{aligned}$$

Using 30ns for TIO, the constraint imposed by condition #1 can also be used to calculate the minimum cycle time:

$$\begin{aligned} & TMIBS + TIO \leqslant TMIDS \\ \text{thus,}\ & 25ns + 30ns \leqslant T_{1Q} + T_{2Q} - 45 \\ & 25ns + 30ns \leqslant 1/2\ \text{cycle} - 45 \end{aligned}$$

therefore, the worst-case instruction cycle time is 200ns. With subject parameters referenced to X1, the same calculations are valid:

$$\begin{aligned} & TIBS + TIO + TIDS \leqslant 1/2\ \text{cycle} \\ \text{thus,}\ & 45ns + 30ns + 25ns \leqslant 1/2\ \text{cycle} \end{aligned}$$

therefore, the worst-case instruction cycle time is again 200ns. From condition #2 and with an instruction cycle time of 200ns, the program storage access time can be calculated:

$$\begin{aligned} & TACC + TIIBS + TIO + TIVA \\ & \leqslant 200ns \\ \text{transposing,}\ & TACC \leqslant 200ns - TIIBS - TIO \\ & - TIVA \\ \text{substituting,}\ & TACC \leqslant 200ns - 25ns - 30ns \\ & - 85ns \\ \text{thus,}\ & TACC \leqslant 60ns \end{aligned}$$

hence, for instruction cycle time of 200ns, a program storage access time of 60ns is implied. The constraint imposed by condition #3 can be used to verify the maximum program storage access time:

$$\begin{aligned} & TIA + TACC \leqslant \text{Instruction Cycle} \\ \text{thus,}\ & TACC \leqslant 200ns - 140ns \\ \text{and,}\ & TACC \leqslant 60ns \end{aligned}$$

confirming that a program storage access time of 60ns is satisfactory.

For an instruction cycle time of 200ns and a program storage access time of 60ns (Condition #2/Figure 5b), the instruction should be valid at the falling edge of MCLK. This relationship can be derived by the following equation:

$$\begin{aligned} & 200ns - TMAS - TACC \\ = & 200ns - 140ns - 60ns \\ = & 0ns \end{aligned}$$

It is important to note that, during the input phase, the beginning of a valid $\overline{LB}/\overline{RB}$ signal is determined by either the instruction to $\overline{LB}/\overline{RB}$ delay (TIIBS) or the delay from the falling edge of MCLK to $\overline{LB}/\overline{RB}$ (TMIBS). Assuming the instruction is valid at the falling edge of MCLK and adding the instruction-to-$\overline{LB}/\overline{RB}$ delay (TIIBS = 25ns), the $\overline{LB}/\overline{RB}$ signal will be valid 25ns after the falling edge of MCLK. With a fast program storage memory and with a valid instruction before the falling edge of MCLK — the $\overline{LB}/\overline{RB}$ signal will, due to the TMIBS delay, still be valid 25ns after the falling edge of MCLK. Using a worst-case instruction cycle time of 200ns, the user cannot gain a speed advantage by selecting a memory with faster access time. Under the same conditions, a speed advantage cannot be obtained by using an I/O port with fast access time (TIO) because the address bus will be stable 55ns (TAS) after the beginning of the third quarter cycle — no matter how early the $\overline{IV}$ data input is valid.

CLOCK CONSIDERATIONS

The on-chip oscillator and timing-generation circuits of the 8X305 can be controlled by any one of the following methods:

Capacitor — if timing is not critical
Crystal — if precise timing is required
External Drive — if application requires that the 8X305 be driven from a system clock

Capacitor Timing. A non-polarized ceramic or mica capacitor with a working voltage equal to or greater than 25V is recommended. The lead lengths of capacitor should be approximately the same and as short as possible; also, the timing circuits should not be in close proximity to external sources of noise. For various capacitor (C_X) values, the cycle time can be approximated as:

C_X (in pF)	APPROXIMATE CYCLE TIME
100	300ns
200	500ns
500	1.1μs
1000	2.0μs

Crystal Timing. When a crystal is used, the on-chip oscillator operates at the resonant frequency (f_o) of the crystal. The series-resonant quartz crystal connects to the 8X305 via pins 10 (X1) and 11 (X2). The lead lengths of the crystal should be approximately equal and as short as possible. Also, the timing circuits should not be in close proximity to external sources of noise. The crystal should be hermetically sealed (HC type can) and have the following electrical characteristics:

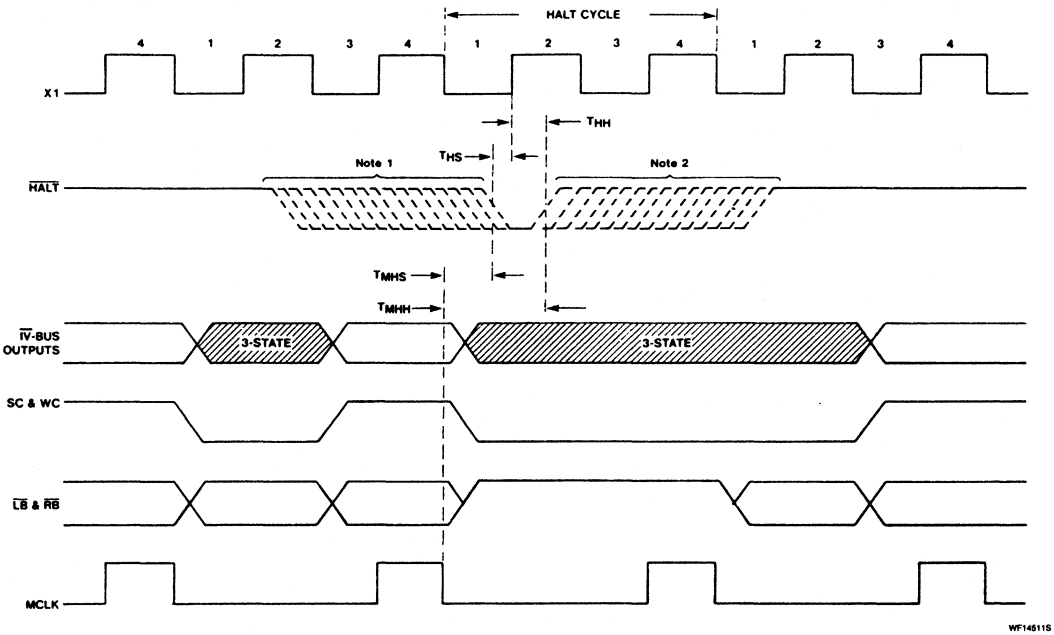
Type — Fundamental mode, series resonant
Impedance at Fundamental — 35Ω max.
Impedance at Harmonics and Spurs — 50Ω min.

The resonant frequency (f_o) of the crystal is related to the desired cycle time (T) by the equation: $f_o = 2/T$; thus, for a cycle time of 200ns, f_o = 10MHz.

$\overline{HALT}$ Logic

The $\overline{HALT}$ signal is sampled via internal chip logic at the end of the first internal quarter of each instruction cycle. If, when sampled, the $\overline{HALT}$ signal is active-low, a halt is immediately executed and the current instruction cycle is terminated. However, the halt cycle does not inhibit MCLK nor does it affect any internal registers of the 8X305. As long as the $\overline{HALT}$ line is active-low, the SC and WC lines are low (inactive), the Left Bank ($\overline{LB}$)/Right Bank ($\overline{RB}$) signals are high (inactive), and the $\overline{IV}$ bus remains in the 3-State mode of operation. Normal operation resumes at the next cycle in which $\overline{HALT}$ is high when sampled (see $\overline{HALT}$ TIMING DIAGRAM.)

HALT TIMING DIAGRAM



NOTES:
1. The HALT signal can switch from High to Low at any time during this interval.
2. The HALT signal can switch from Low to High at any time during this interval.

TIMING DESCRIPTIONS:
T_{HS} — setup time from HALT to X1 (independent of instruction cycle time)
T_{HH} — hold time from X1 to HALT (independent of instruction cycle time)
T_{MHS} — setup time from MCLK to HALT (dependent upon instruction cycle time)
T_{MHH} — hold time from MCLK to HALT (dependent upon instruction cycle time)

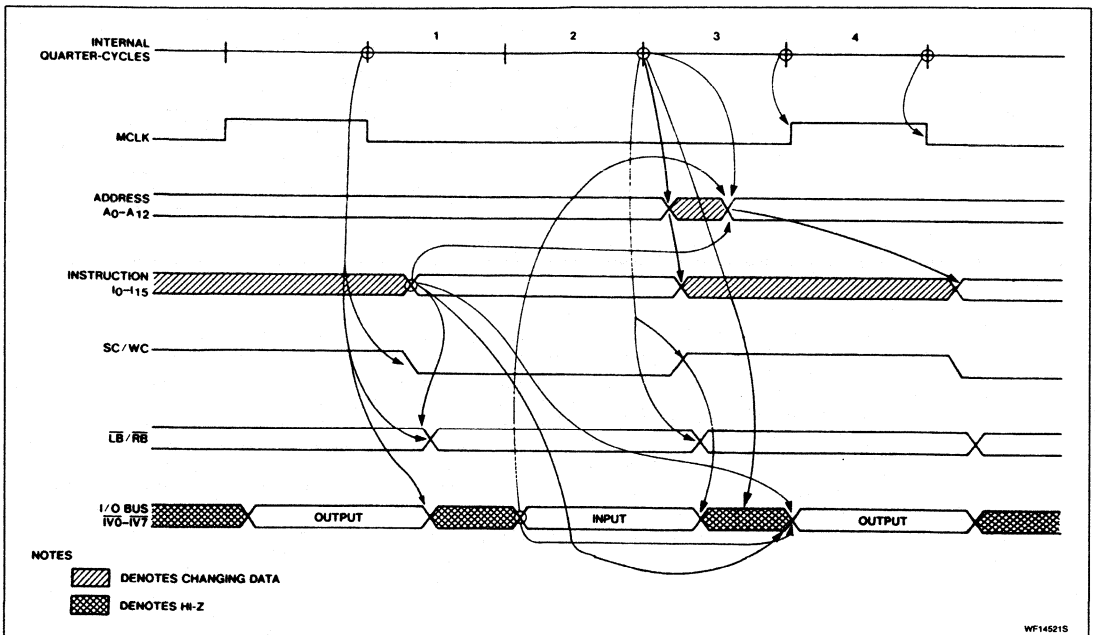


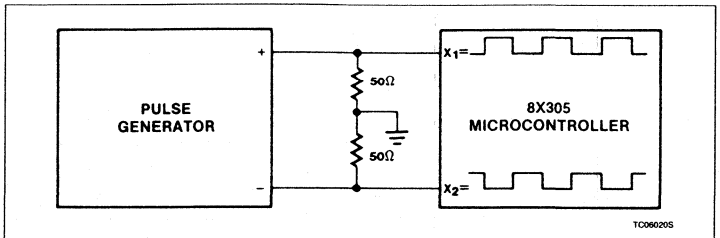
Figure 7. Timing Relationships of 8X305 I/O Signals

Using an External Clock. The 8X305 can be synchronized with an external clock by simply connecting appropriate drive circuits to the X1/X2 inputs. Figure 8 shows how the on-chip oscillator can be driven from the complementary outputs of a pulse generator. In applications where the Microcontroller must be driven from a master clock, the X1/X2 lines can be interfaced to TTL logic as shown in Figure 9.

$\overline{\text{RESET}}$ Logic

$\overline{\text{RESET}}$ (pin 43) can be driven from a high (inactive) state to a low (active) state at any time with respect to the system clock, that is, the reset function is asynchronous. To ensure proper operation, $\overline{\text{RESET}}$ must be held low (active) for one full instruction time. When the line is driven from a high state to an active-low state, several events occur — the precise instant of occurrence is basically a function of the propagation delay for that particular event. As shown in the $\overline{\text{RESET}}$ TIMING DIAGRAM, these events are:

- The Program Counter and Address Register are set to address zero and remain in that state as long as the $\overline{\text{RESET}}$ line is low. Other than PC and AR, $\overline{\text{RESET}}$ does not affect other internal registers.



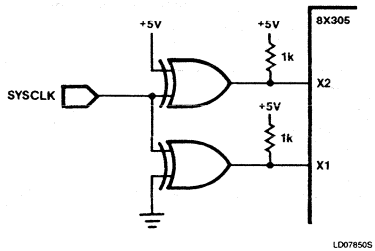
PULSE GENERATOR CHARACTERISTICS:
Z_{OUT} = 50Ω
RISETIME ≤ 10ns
V_{OUT} = 0 to 1.5V
Skew ≤ 10ns

Figure 8. Clocking With a Pulse Generator

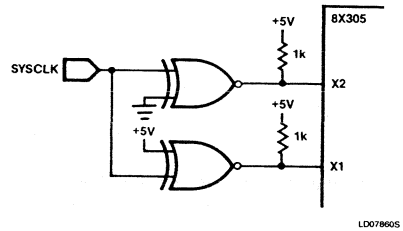
- The input/output ($\overline{\text{IV}}$) bus goes 3-State and remains in that condition as long as the $\overline{\text{RESET}}$ line is low.
- The Select Command and Write Command signals are driven low and remain low as long as the $\overline{\text{RESET}}$ line is low.
- The Left Bank/Right Bank ($\overline{\text{LB}}$/$\overline{\text{RB}}$) signals are forced high asynchronously for the period in which the $\overline{\text{RESET}}$ line is low.

During the time $\overline{\text{RESET}}$ is active-low, MCLK is inhibited. Moreover, if the $\overline{\text{RESET}}$ line is driven low during the last two quarter cycles, MCLK may be shortened for that particular machine cycle. When $\overline{\text{RESET}}$ line is driven high (inactive) — one quarter to one full instruction cycle later, MCLK appears just before normal operation is resumed. The $\overline{\text{RESET}}$/MCLK relationship is clearly shown by "B" in the timing diagram. As long as the $\overline{\text{RESET}}$ line is active-low, the $\overline{\text{HALT}}$ signal (described next) is not sampled by internal logic of the 8X305.

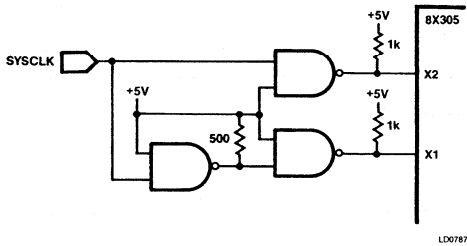
1. Using a 74LS136 Quad Execlusive-OR gate (open collector).



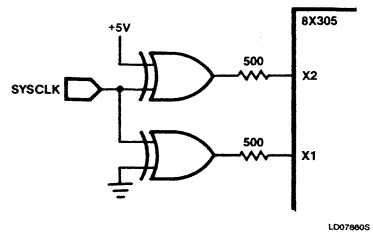
2. Using a 74LS266 Quad Exclusive-NOR gate (open collector).



3. Using a 74F38* Quad NAND gate (open collector).



4. Using a 74LS286 Quad Exclusive-OR gate.

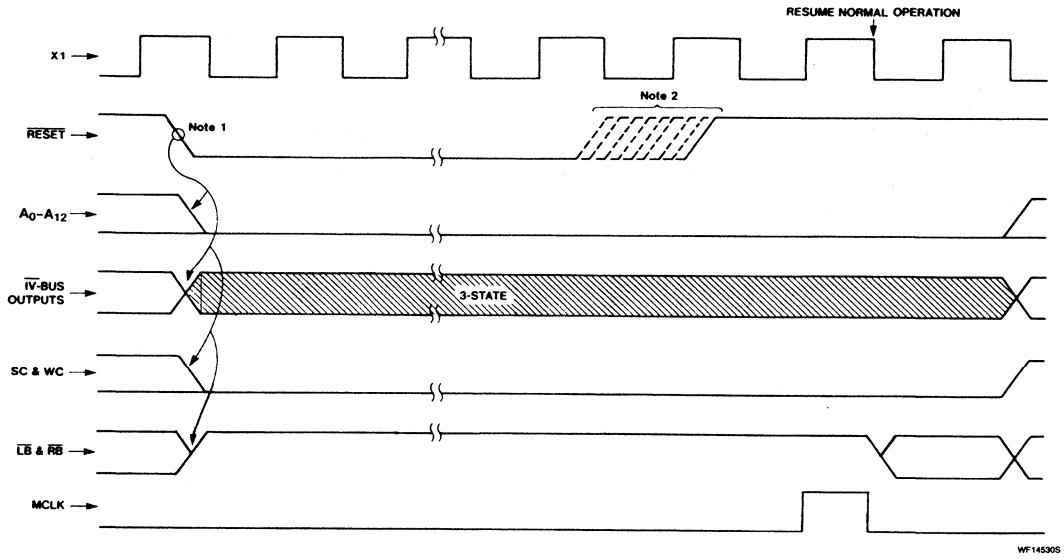


TTL DRIVER CHARACTERISTICS:
Fall Time ⩽ 10ns
Skew Between Complementary Outputs ⩽ 10ns

NOTES:
1. All circuits, as drawn, preserve the phase relationship of SYSCLK to X1.
2. The resistor values of 1k ohms for open-collector pull-ups and 500 ohms for active pull-up series resistors are calculated to be optimum for all opening conditions and silicon varations.
3. The 8X305 clock may be driven by circuits other than those shown here. The circuits shown however, have been tested under conditions in excess of those that will be found in a normal system. Exclusive-OR/NOR type gates were selected to minimize the skew between the X1 and X2 inputs.
4. 74LS38 and 74S38 gates were tested in addition to the 74F38 chip. These were found to be the least robust configurations of all those tested, although they did work over normal operating conditions abd beyond. When failure occured it was due to excessive skew between X1 and X2 caused by the inverting gate in the X1 leg.

Figure 9. Clocking With TTL

RESET TIMING DIAGRAM



NOTES:
1. A High to Low transition of the RESET signal will force the Address Bus to an all-zero configuration.
2. The RESET signal can switch from Low to High at any point within this time interval and, in all cases, MCLK will occur at least one-quarter cycle time later as shown.

Date of Issue	December 17, 1986
Status	Product Specification
Application Specific Product	

8X401
Microcontroller

DESCRIPTION

The 8X401 Microcontroller is a very high-speed bipolar microprocessor implemented with internal ECL technology. The 8X401 combines speed, flexibility, and a bit-oriented instruction set to accommodate many sophisticated applications. It excels in systems that require high-speed bit or byte manipulations, such as high-speed controllers and data communications.

The 8X401 can fetch, execute, and generate the next instruction address for a 20-bit instruction in a minimum of 150ns. Within one instruction cycle, the 8X401 can be programmed to input, right-rotate and mask single or multiple bit subfields, perform an ALU operation, left-rotate, merge the subfield into the destination, and output.

To interface with program memory, the 8X401 uses a 13-bit address bus and a 20-bit instruction bus. An 8-bit bidirectional data/address bus, and an I/O control and timing bus is used to access external peripheral devices.

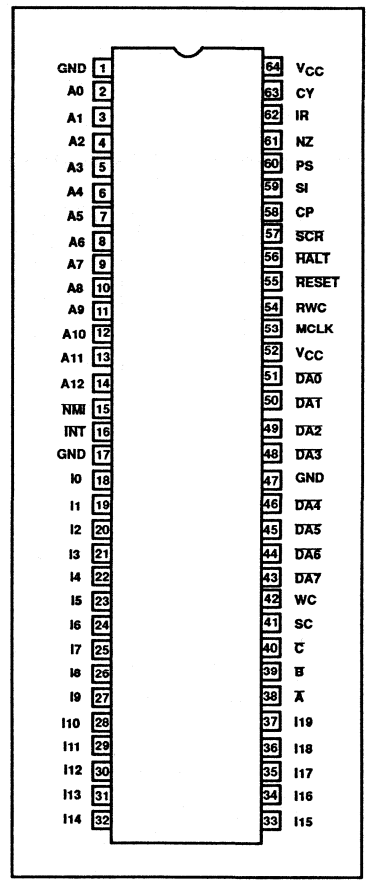
ORDERING INFORMATION

DESCRIPTION	ORDER CODE
64-Pin Ceramic DIP – 900 Mil Wide	N8X401I
64-Pin Ceramic DIP – 900 Mil Wide	N8X401IN

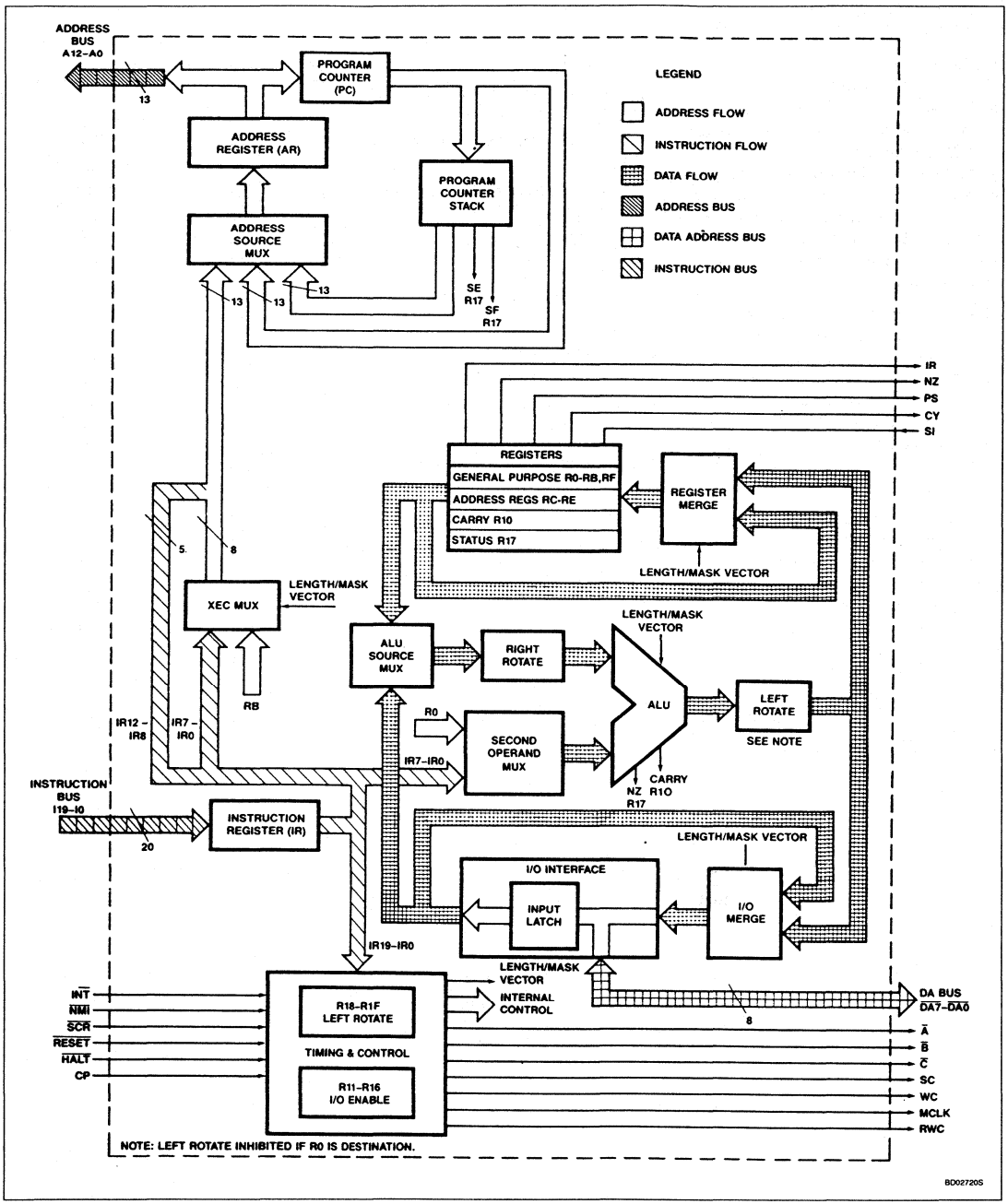
FEATURES

- Fetches and executes all instructions in a minimum of 150ns
- Bit manipulation-oriented instruction set
- Separate buses for instruction, instruction address and I/O
- Sixteen 8-bit registers
- On-chip interrupt control
- TTL compatible I/O
- Single +5V supply
- 0.9-inch 64-pin DIP, 68-pin plastic leaded chip carrier
- Two user-definable status flags
- Single TTL clock input
- Three independent I/O banks
- On-board control sequencer
- On-chip subroutine capabilities
- Fixed instruction set—32 instructions
- Complete development support

PIN CONFIGURATION



BLOCK DIAGRAM OF THE 8X401



PIN DESCRIPTION

PIN NO.	IDENTIFIER	FUNCTION
1, 17, 47	GND	Ground.
2 – 14	A12 – A0	**Program Address Lines:** These active-high outputs permit direct addressing of up to 8192 locations of program storage; A0 is LSB.
15	$\overline{NMI}$	**Non-Maskable Interrupt:** The falling edge of this active-low input pin generates a non-maskable interrupt.
16	$\overline{INT}$	**Interrupt:** This active-low input pin is tested during the fourth quarter of each instruction cycle. If an interrupt is indicated and if interrupts are enabled, the address of the next instruction that was to be executed is stored onto the program counter stack before the interrupt is serviced.
18 – 37	I19 – I0	**Instruction Lines:** These active-high input lines receive 20-bit instructions from program storage; I0 is LSB.
38	$\overline{A}$	**Bank A:** When low, devices connected to bank A are accessed. (Note: Typically, the A signal is tied to the $\overline{ME}$ input pin of I/O peripherals.)
39	$\overline{B}$	**Bank B:** When low, devices connected to bank B are accessed. (Note: Typically, the B signal is tied to the $\overline{ME}$ input pin of I/O peripherals.)
40	$\overline{C}$	**Bank C:** When low, devices connected to bank C are accessed. (Note: Typically, the C signal is tied to the $\overline{ME}$ input pin of I/O peripherals.)
41	SC	**Select Control:** When high, an address is being output on pins $\overline{DA7}$ through $\overline{DA0}$.
42	WC	**Write Control:** When high, data is being output on pins $\overline{DA7}$ through $\overline{DA0}$.
43 – 46 48 – 51	$\overline{DA7}$ – $\overline{DA0}$	**Data Address Bus:** These active-low, bidirectional, three-state lines are used for I/O; $\overline{DA0}$ is LSB.
52, 64	V_{CC}	+5V power supply.
53	MCLK	**Master Clock:** This active-high output signal is used to strobe data into data peripherals for clocking I/O devices and/or synchronization of external logic. MCLK is active-high in the fourth quarter cycle.
54	RWC	**Read/Write Clock:** This active-high output signal is used for synchronization of external logic and is active-high during the third and fourth quarter cycles.
55	$\overline{RESET}$	**Reset:** The $\overline{RESET}$ input pin is used to initialize the 8X401.
56	$\overline{HALT}$	**Halt:** The $\overline{HALT}$ input is sampled during the first quarter cycle of each instruction cycle. When the $\overline{HALT}$ input is low, the instruction cycle is not executed.
57	$\overline{SCR}$	**Slow Clock Request:** This active-low control input is sampled during the first quarter cycle of each instruction. When $\overline{SCR}$ is asserted, it will cause the current instruction to be executed at half of the normal clock rate. This control input is necessary to accommodate I/O devices that cannot operate at the 8X401's full speed, without having to continuously run the 8X401 at half speed.
58	CP	**Clock Pulse:** Each 8X401 quarter cycle will correspond to one full cycle of the clock pulse.
59	SI	**Status Input:** The value of the SI pin during the fourth quarter cycle is transferred to SI bit in the status register.
60	PS	**Programmable Status:** The programmable status pin is controlled entirely by the user program.
61	NZ	**Non-Zero:** The NZ bit of the status register is reflected on this pin.
62	IR	**Interrupt Receivable:** The IR pin indicates whether an interrupt applied at any point in time will be serviced. Interrupts are receivable when the interrupt mask (status register, bit 0) is clear and the stack is not full (IM = 0 and SF = 0).
63	CY	**Carry:** Carry bit from R10 is output on this pin.

SMALL SYSTEM CONFIGURATION

The system hookup shown on the next page, although of the simplest form, provides a fundamental example of the 8X401 Microcontroller and compatible peripheral relationships. As shown, the 8X401 can directly address up to 8K locations of program storage.

Each of the three bank pins ($\overline{A}$, $\overline{B}$, or $\overline{C}$) are capable of uniquely addressing 256 input/output locations via the Data Address bus ($\overline{DA7}$ – $\overline{DA0}$).

The addressable locations for each bank can be used in a variety of ways. The hookup shown below is just one method of implementation.

When a particular bank signal is asserted, that bank is enabled and any one of 256 locations on that bank can be accessed for input/output operations.

PROGRAM STORAGE INTERFACE

As shown in the 8X401 small system hookup, program memory is connected to output address lines A12 through A0 (A0 = LSB) and input instruction lines I19 through I0 (I0 = LSB). An address output on A12 – A0 identifies one 20-bit instruction word in program memory. The program memory outputs an instruction word on I19 – I0 which defines the microcontroller operation which is to follow. One instruction word equals one completed operation. Any TTL-compatible memory can be used for program storage, provided the worst-case access time is compatible with the instruction cycle time used for the application. See timing section for appropriate calculations.

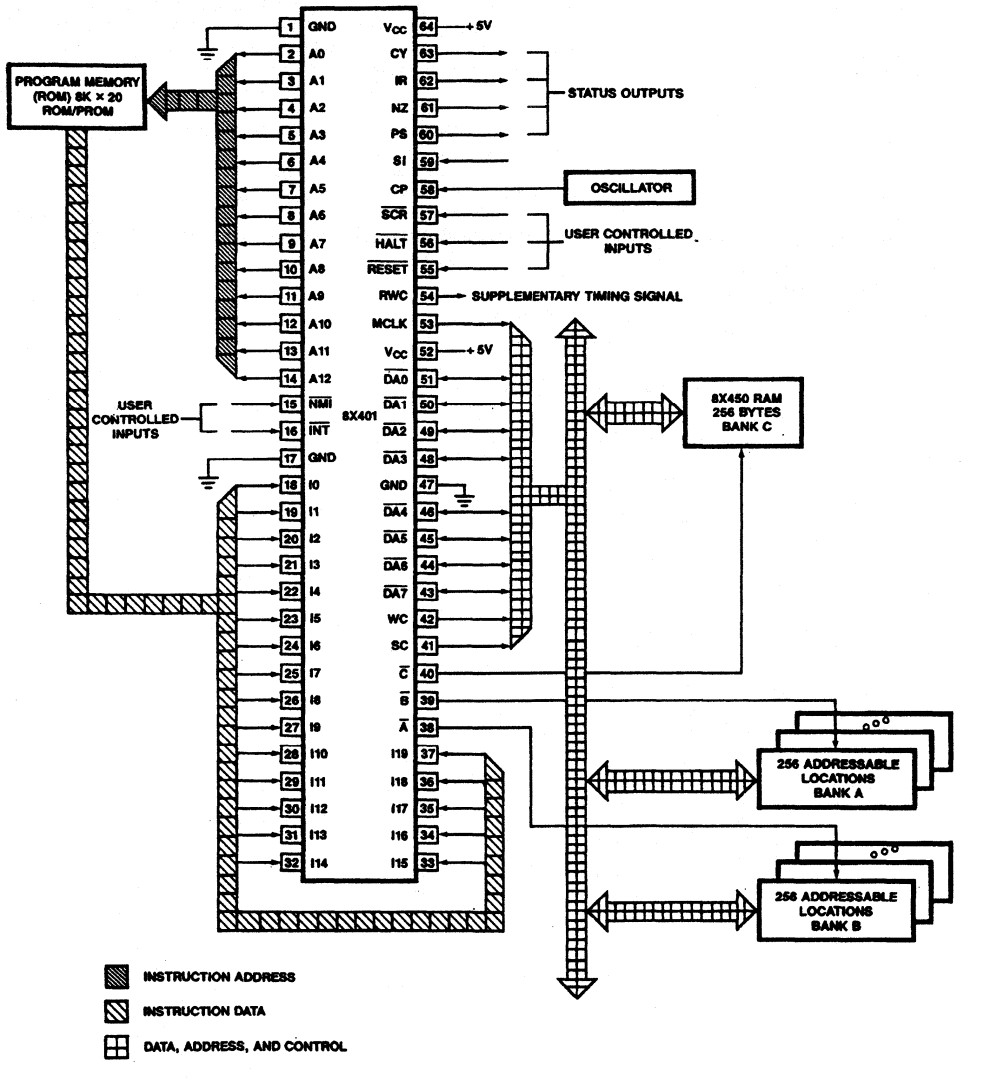
I/O INTERFACE AND CONTROL

The Data Address (DA) bus is an 8-bit bidirectional I/O bus which provides a communication link between the 8X401 and the three banks of the I/O devices. The $\overline{A}$ (A bank), $\overline{B}$ (B bank), and $\overline{C}$ (C bank) control signals identify which bank is enabled. When all three banks go high (inactive), neither bank is enabled and the DA bus is inactive (three-state). A functional analysis of the three bank signals is shown below:

A	B	C	FUNCTION
Low	Low	Low	This state is not generated by the 8X401.
Low	High	High	Enable A bank devices.
High	Low	High	Enable B bank devices.
High	High	Low	Enable C bank devices.
High	High	High	Disable all devices; DA bus is three-state

Both data and I/O address information are multiplexed on the DA bus. The SC (Select Command) and WC (Write Command) signals distinguish between data and I/O address information as shown in the table below. Although the table shows bank A only, the same conditions apply to banks B and C.

BANK A	SC	WC	FUNCTION
High	Low	Low	DA bus is three-state and not looking for input data.
Low	Low	Low	The DA bus is reading input data.
Low	Low	High	Data is being output.
Low	High	Low	Address is being output.
X	High	High	This condition is never generated.



8X401 Small System Hookup

DATA PROCESSING

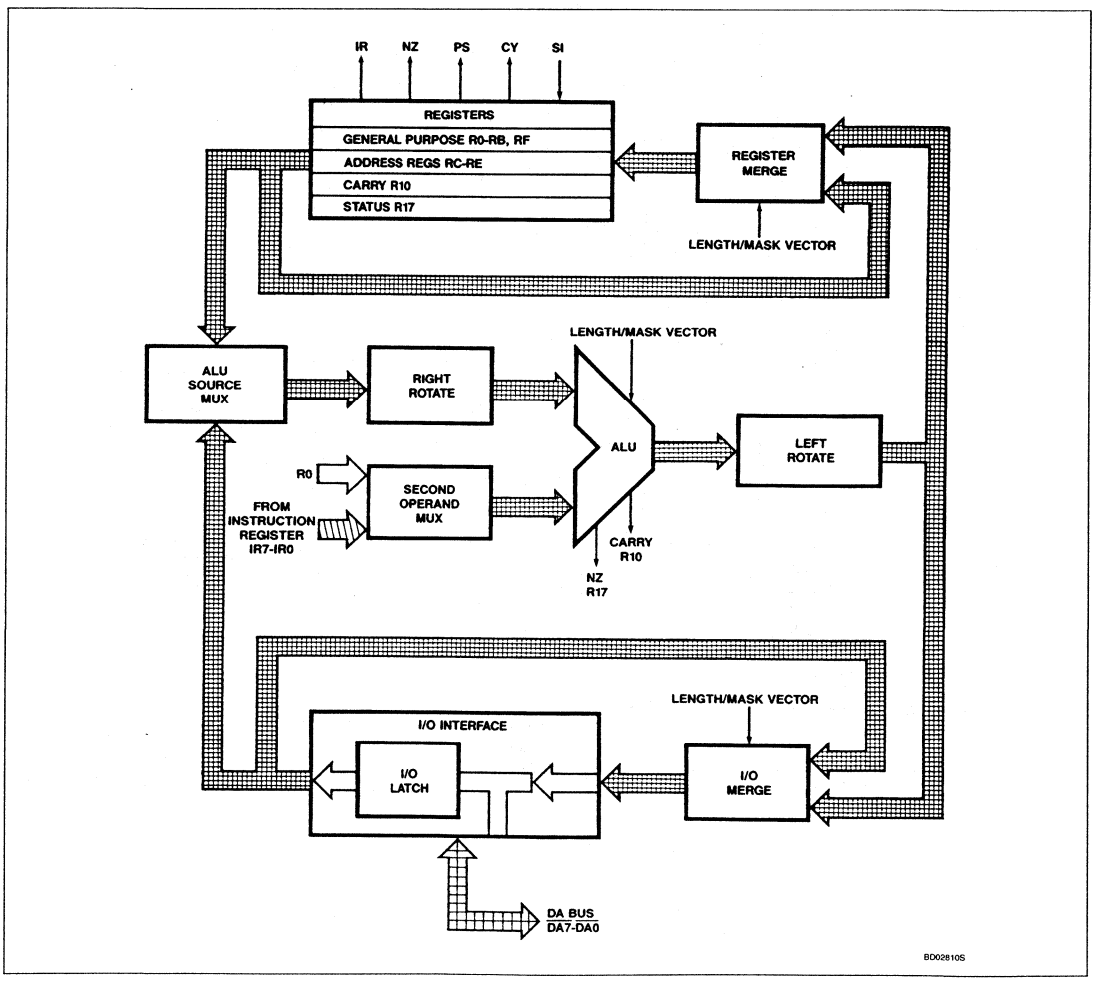
The data processing section of the 8X401 consists of a number of logical subsections. In order of processing, the data sees the right rotator, the ALU, the left rotator, and the merge circuits. Data sources and destinations can be various on-chip registers, the bidirectional DA bus, or immediate subfields. The data processing paths are shown below.

DATA REGISTERS

General-Purpose Storage

There are 13 source/destination general-purpose registers available on the 8X401. Three of these registers, specifically Registers 0, B, and F, have other special functions in addition to being general purpose. A summary of the registers is listed below:

- **R0 (Auxiliary Register)** — Register 0 is also used as the implied second operand for two operand instructions (ADD, ADD with CARRY, XOR, AND). The primary operand is specified in the source field of the instruction word and the AUX register is the implied second operand. Prior to performing arithmetic or logical operations (other than the IMMEDIATE operation), it is assumed that R0 contains the appropriate data. In order to reduce the possibility of erroneous results and to minimize the number of instructions required to transfer a right-justified second operand into the AUX, the left-rotate and merge functions are inhibited when specifying the AUX as a destination address. This allows subfields from any internal register or I/O bank to be transferred to the AUX with the subfield LSB right-justified and unspecified bits set to zero.
- **R1 through RA** — These 10 addresses specify general-purpose, on-chip storage registers.
- **RB** — Register B is also used as the implied source for the XEC instruction.
- **RF** — Register F is also used as the implied destination for the XOR IMMEDIATE and AND IMMEDIATE instruction classes.



Enabled I/O Addresses

These three registers (RC RD, and RE) always contain the address of the most recently enabled I/O device for each of the three I/O banks. When register C, D, or E is the specified destination address, the destination data is sent to both the on-chip register and the corresponding bank on the DA bus. The relationship between the register addresses and I/O banks is shown below:

REGISTER	BANK
C	A
D	B
E	C

When these registers are specified as a destination address, the L field must be set to 0 (full 8-bit operation). Also, note that registers C, D, and E may not be used with the XMIT 5 or ADD IMMEDIATE 5 instructions.

Carry

Register 10 contains the Carry bit. Bit position 0 (LSB) is the Carry bit, and positions one through seven are always zero. The Carry bit is updated each time an ADD, ADD IMMEDIATE, or ADD WITH CARRY instruction is performed. When specifying address 10 as a destination, only bit 0 (the Carry bit) will be written to. Data written to the Carry bit will be the LSB of the right-rotated data after any specified operation.

When the Carry register is the explicit destination of any ADD instruction, it will contain the carry resulting from the add operation rather than the LSB of the sum. Carry can also be affected via the Return and Set Carry or Return and Clear Carry instructions.

Status Register

This address specifies the current condition of the 8X401 system. The status register may be either a source or destination; however, certain bits in the status register are read-only. Four status outputs are available on 8X401 pins. They are NZ (Not Equal to Zero), PS (Programmable Status), IR (Interrupts Receivable), and Carry (R10, bit 0). The IR pin goes high when the interrupt mask is clear and the stack is not full. The IR output is updated during the 4th quarter cycle. The following descriptions define the bits within the status register.

Bit 0: (IM) — This bit represents the Interrupt Mask control. When IM is set, the interrupt is inhibited. This bit is set automatically by a response from a standard or non-maskable interrupt, or RESET. IM can also be set or cleared by a write to the status register.

Bit 1: (NZ) — This bit is set whenever the ALU output data is not equal to zero after any of the following instructions: MOVE, ADD, AND, XOR, ADD IMMEDIATE, AND IMMEDIATE, XOR IMMEDIATE, or ADD WITH CARRY. NZ can also be written to directly when specified in the destination field. This operation will negate and take priority over the normal setting by the ALU output. NZ is not affected after an XMIT instruction, or after a write to R17.

Bit 2: (PS) — This is the Programmable Status bit. The contents are reflected on the PS output pin. This status is controlled entirely by the user program.

Bits 3 and 4: (UF0, UF1) — These two bits represent user flags and have no assigned functions. They can be used as 1-bit internal flags and are entirely under control of the user program.

Bit 5: (SI) — This bit reflects the state of the status input pin. This read-only bit is updated during the 4th quarter cycle.

Bits 6 and 7: (SE, SF) — These read-only bits indicate Stack Empty and Stack Full, respectively. The bits are updated during the 3rd quarter cycle within the instruction that alters the stack status.

INSTRUCTION WORD (See Table 3)

Operations Code Field — The 4-bit opcode specifies one of 16 classes of instructions. Some instructions require two additional subopcode fields, X and XS. Variations and interpretations are displayed in Table 2.

Source (S) and Destination (D) Fields — The 5-bit "S" and "D" fields specify the source and destination, respectively, for the operation that is defined by the opcode. The "S" and/or "D" fields specify an internal 8X401 register or a variable length field from an I/O device. Hexadecimal values and source/destination field assignments for all internal registers are shown in Table 1.

When RC-RE (banks A, B, or C, respectively) are specified as the destination, the data is output onto the DA bus using the specified bank. The data is also stored in the specified register and may be later accessed as source data.

Rotate (R) and Length (L) Fields — The R field is used in conjunction with the L field to define the desired data within a register or I/O device. The source data is right-rotated prior to ALU operations, such that the bit specified by the R field is right-justified. The L field specifies the number of bits of data to be used for the operation. After the ALU operation, the data is left-rotated back to the original position prior to merging the data in the destination register.

When the L field specification is 0 (indicating a full 8-bit operation), the left-rotate is suppressed. This allows byte rotate operations to be performed. The left-rotate is also suppressed when the destination is register 0. This is the AUX register and is used as the implied second operand in certain instructions.

It should also be noted that subfields are defined at the ends of a register; for example, bit positions 1, 0, 7, and 6 constitute a contiguous 4-bit subfield.

Data Field — The data field holds data that can be processed directly from the instruction word.

LEFT-ROTATE OVERRIDE BLOCK

Register addresses 18-1F are destination only and are used to independently control left rotation of data prior to storage in the destination. Specifying 18-1F as a destination causes the data to be returned to the source address.

In order to move a processed subfield within the same register but in different bit positions (the LSB of the contiguous subfield can vary), it is necessary to independently specify the LSB for both the source and destination. The order of operation is as follows:

- Register or I/O source data is right-rotated as specified by the "R" field. Along with the "L" field, the subfield data is defined.
- Subfield data is processed via the ALU.
- Data is left-rotated 0 – 7 bits, depending on the corresponding register addresses 18-1F as specified in the destination field rather than using the "R" field.
- After left-rotation the specified subfield is merged into those bits of the original source data. The unspecified bits of the original source data remain unchanged.
- Result is stored in the register address specified by the source field.

Note that the left-rotate is always inhibited if the "L" field is zero. Also, addresses 18-1F may not be used in the destination field for the XMIT or ADD IMMEDIATE instructions. The destination addresses and corresponding left-rotate values are shown in Table 2.

DA BUS CONTROL BLOCK

Register addresses 11 – 16 are used by the 8X401 to access I/O devices for either a source or destination specified within the instruction. Register addresses 13, 15, and 16 specify banks A, B, and C, respectively, whereas addresses 11, 12, and 14 specify bank pairs AB, CA, and BC, respectively (Table 4). One bank of each pair is known as the preferred bank. The preferred banks for

pairs AB, CA, and BC are banks A, C, and B, respectively. The first letter from each bank pair can serve as a mnemonic aid as to which bank is the preferred bank. Having a preferred bank is simply a method of determining which bank to read when an instruction would otherwise indicate that two banks should be read at once.

When used as a source, the appropriate I/O bank is enabled and data is read from the activated I/O device on that bank. The I/O device may have been activated by a previous address select instruction where registers C-E were the specified destination. If a bank pair (addresses 11, 12, or 14) is specified as a source, only data from the preferred bank of that pair will be read in.

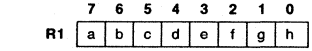
When addresses 11 – 16 are specified as the destination address, the destination data is sent to the DA bus. The Write Control (WC) signal goes high, indicating data (as opposed to an address) is on the DA bus and is to be written to the activated I/O device on the selected bank(s).

When addresses 11 – 16 are specified as the destination and the "L" field is not zero, the following statements apply: If the source is a register and the destination is a single bank, the bank will be read (or the preferred bank of a bank pair will be read) to obtain the data required to perform the merge operation. The result is that processed data from the specified subfield of the source register is returned to that selected field of the destination bank and any bits outside of the specified subfield will be loaded with unprocessed data from the I/O device just read. If the destination is a bank pair, the data from the procedure just described is sent to both banks. Below is an example of the outcome of an ADD instruction with Length = 4, Rotate = 2, Source = R1, and Destination = R11 (bank pair AB).

Example 1:

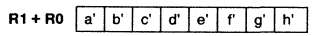
OPCODE	L	S	R	D
ADD	4	R1	2	R11

Bank Pair AB

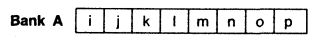


Specified Subfield

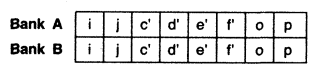
Source register R1. Note specified subfield from L = 4 and R = 2.



Add R1 with R0 (AUX). Result after left-rotate.



Read preferred bank A of AB.



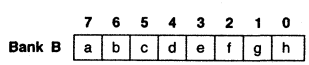
Result after merge. Data put out on both banks A and B.

If, however, the specified source is a bank or bank pair, any unspecified bits will contain unprocessed data from the source I/O device. Below is an example of the outcome with Source = R14 (bank pair BC) and Destination = R13 (bank A).

Example 2:

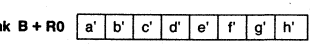
OPCODE	L	S	R	D
ADD	4	R14	2	R13

Bank Pair BC — Bank A

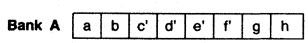


Specified Subfield

Read Preferred bank B of BC. Note specified subfield from L = 4 and R = 2.



Add bank B data with R0 (AUX). Result after left-rotate.



Result after merge. Unspecified bits contain original source data.

PROGRAM COUNTER STACK

The 8X401 stack is capable of saving up to four return addresses for subroutines and interrupts. Addresses are pushed onto the stack as a result of a call or a maskable or non-maskable interrupt. Addresses are popped from the stack as a result of an unconditional RETURN, a satisfied conditional RETURN, or a Pop Stack and Jump instruction. The status of the stack (whether empty, full, or neither) is available from the SE and SF flags in the status register and the Interrupts Receivable (IR) output pin.

INTERRUPTS

Interrupt ($\overline{INT}$)

The interrupt input is tested once each instruction cycle, during the fourth quarter cycle (see Figure 1). When the interrupt input is taken low and is enabled, the address of the next instruction is pushed onto the program counter stack.

Program flow is transferred to address 2 for the start of the service routine (Figure 2). This is accomplished by inserting a dummy instruction cycle after the interrupt is accepted.

The interrupt mask bit (R17, bit 0) is set automatically as part of the interrupt response.

The Interrupts Receivable (IR) pin indicates whether an interrupt applied at any point in time will be serviced. Interrupts are receivable when the interrupt mask is clear and the stack is not full (IM = 0 and SF = 0).

Non-Maskable Interrupt ($\overline{NMI}$)

The function of the non-maskable interrupt is similar to the standard interrupt, except that the interrupt receivable status has no effect on its operation and the address jumped to is 1 rather than 2 (Figure 2). Address 1 should contain an unconditional JUMP to the start of the NMI service routine. An NMI is triggered by a falling edge on the $\overline{NMI}$ input. The interrupt mask is set to prevent normal interrupts from interfering with the NMI service routine. Note that it may not always be possible to recover from an NMI, since the condition of the interrupt mask prior to the NMI is not known, and the NMI response may overflow the stack.

SLOW CLOCK REQUEST ($\overline{SCR}$)

This control input is sampled during the first quarter cycle of each instruction along with the instruction data. If the input is low, it will cause the current instruction to be executed at half of the normal clock rate. The purpose of this function is to facilitate accesses to I/O devices that cannot operate at the 8X401's

full speed, without the need to run the 8X401 continuously at half speed.

HALT

The HALT input is sampled during the first quarter cycle of each instruction. If the HALT input is low, the instruction cycle is not executed. The MCLK continues to operate normally (high every fourth quarter cycle), even though program execution has ceased. When the HALT input goes high, program execution will resume at the next falling edge of MCLK. The DA bus is also inactive during a HALT operation. Like MCLK, RWC continues to operate during a HALT operation.

RESET LOGIC

The RESET pin is used to initialize the 8X401. When RESET is low, the address outputs (A12 – A0) are high impedance, the stack pointer is set to the top of the stack (empty), MCLK is inhibited, RWC is low, and the interrupt mask (bit 0, register 17) is set.

When RESET is released, the address outputs all go low (program address 0). A dummy instruction cycle occurs to allow time to fetch the first instruction from program storage at address 0. Only MCLK, RWC, and the address bus are in operation during the dummy cycle. The first active instruction cycle will begin following the first MCLK after RESET is released. The instruction at address 0 should be an unconditional jump to the beginning of the main program (which may be proceeded by a power-up sequence to initialize the system (Figure 2).

If RESET is applied during program execution, its effect is immediate. That is, if MCLK is high, it may be prematurely terminated by RESET.

Table 1. Hexadecimal Addresses and Source/Destination Specification

ADDRESS	DESIGNATION	S	D	ADDRESS	DESTINATION	S	D
00	R0 (AUX) — General Purpose[1]	X	X	10	R10 — Carry[6]	X	X
01	R1 — General Purpose	X	X	11	R11 — Bank Access Command (Bank Pair AB)	X	x
02	R2 — General Purpose	X	X	12	R12 — Bank Access Command (Bank Pair CA)	X	x
03	R3 — General Purpose	X	X	13	R13 — Bank Access Command (Bank A)	X	x
04	R4 — General Purpose	X	X	14	R14 — Bank Access Command (Bank Pair BC)	X	x
05	R5 — General Purpose	X	X	15	R15 — Bank Access Command (Bank B)	X	x
06	R6 — General Purpose	X	X	16	R16 — Bank Access Command (Bank C)	X	x
07	R7 — General Purpose	X	X	17	R17 — Status[4]	X	X
08	R8 — General Purpose	X	X	18	R18 — Suppress Left-Rotate[5]		X
09	R9 — General Purpose	X	X	19	R19 — Left-Rotate 1 Place[5]		X
0A	RA — General Purpose	X	X	1A	R1A — Left-Rotate 2 Places[5]		X
0B	RB — General Purpose[2]	X	X	1B	R1B — Left-Rotate 3 Places[5]		X
0C	RC — Address Reg (Bank A)	X	X	1C	R1C — Left-Rotate 4 Places[5]		X
0D	RD — Address Reg (Bank B)	X	X	1D	R1D — Left-Rotate 5 Places[5]		X
0E	RE — Address Reg (Bank C)	X	X	1E	R1E — Left-Rotate 6 Places[5]		X
0F	RF — General Purpose[3]	X	X	1F	R1F — Left-Rotate 7 Places[5]		X

NOTES:
1. Also used as implied second operand for two operand instructions.
2. Also used as implied source for XEC instructions.
3. Also used as implied destination for XOR IMMEDIATE and AND IMMEDIATE instructions.
4. Certain bits in the status register are read-only. (See Status Register within text.)
5. The result is returned to the register address specified by the source field.
6. Carry register, bit 0 is the carry bit. Bits 1 – 7 are always set to zero.

Table 2. Various of Instruction Types

INSTRUCTION TYPE	VARIATION	OPCODE	X	XS	DESCRIPTION
MOVE	MOV	0000	–	–	MOVE
ADD	ADD	0001	–	–	ADD
	ADC	0101	–	–	ADD with CARRY
	AD8	1000	–	–	ADD IMMEDIATE 8
	AD5	1001	–	–	ADD IMMEDIATE 5
AND	AND	0010	–	–	AND
	AN8	1010	–	–	AND IMMEDIATE 8
	AN5	1011	–	–	AND IMMEDIATE 5
XOR	XOR	0011	–	–	Exclusive-OR
	XR8	1100	–	–	Exclusive-OR IMMEDIATE 8
	XR5	1101	–	–	Exclusive-OR IMMEDIATE 5
XEC	XEC	0100	–	–	EXECUTE
XMIT	XT8	0110	–	–	Transmit IMMEDIATE 8
	XT5	0111	–	–	Transmit IMMEDIATE 5
RETURN	RIF NS	1110	000	–	RETURN IF SI = 0
	RIF S	1110	001	–	RETURN IF SI = 1
	RIF NC	1110	010	–	RETURN IF CARRY = 0
	RIF C	1110	011	–	RETURN IF CARRY = 1
	RIF Z	1110	100	–	RETURN IF ALU = 0
	RIF NZ	1110	101	–	RETURN IF ALU ≠ 0
	PSJ	1110	110	–	POP STACK and JUMP
	RTN	1110	111	00	RETURN
	RCC	1110	111	10	RETURN and CLEAR CARRY
	RSC	1110	111	11	RETURN and SET CARRY
JUMP	JIF NS	1111	000	–	JUMP IF SI = 0
	JIF S	1111	001	–	JUMP IF SI = 1
	JIF NC	1111	010	–	JUMP IF CARRY = 0
	JIF C	1111	011	–	JUMP IF CARRY = 1
	JIF Z	1111	100	–	JUMP IF ALU = 0
	JIF NZ	1111	101	–	JUMP IF ALU ≠ 0
	JSR	1111	110	–	JUMP to SUBROUTINE
	JMP	1111	111	–	JUMP

Instruction Set Overview: The 8X401 instruction set is summarized in Table 2. Subsets of each instruction type are grouped together showing the variations of each instruction type. The hardware and software descriptions can be found in the data operations section.

Table 3. 8X401 Instruction Formats

1. Format for the MOVE, ADD, XOR, and ADD with Carry Instructions:

OPCODE	L	SOURCE	R	DESTINATION

2. Format for the XMIT 8 and ADD Immediate 8 Instruction:

OPCODE	L	DESTINATION	DATA8

3. Format for the XMIT 5 and ADD Immediate 5 Instruction:

OPCODE	L	DESTINATION	R	DATA5

4. Format for the AND Immediate 8, and XOR Immediate 8 Instruction:

OPCODE	L	SOURCE	DATA8

5. Format for the AND Immediate 5 and XOR Immediate 5 Instruction:

OPCODE	L	SOURCE	R	DATA5

6. Format for the JUMP, Subroutine Jump, Conditional Jump, and Conditional Return Instruction:

OPCODE	X	ADDRESS

7. Format for the Unconditional Return, Return and Set Carry, and Return and Clear Carry Instruction:

OPCODE	X	XS	(UNUSED

8. Format for the XEC Instruction:

OPCODE	L	ADDRESS

Table 4. I/O Acess Register

REGISTER	INPUT BANK	OUTPUT BANK
11	A	A & B
12	C	C & A
13	A	A
14	B	B & C
15	B	B
16	C	C

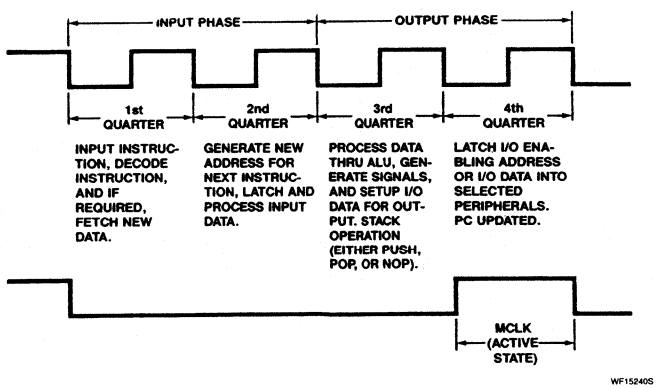


Figure 1. Instruction Cycle and MCLK with: Clock Input = 26.67MHz and Cycle Time = 150ns

DATA OPERATIONS OF THE 8X401 (See Tables 2 and 3)

MOVE, ADD, AND, XOR, ADD with CARRY

SOURCE	PRE-ALU	ALU	POST-ALU	DESTINATION
R0 – R17 as specified by "S" field of instruction.	Right rotate as specified by "R" field of instruction.	Perform appropriate ALU operation. (Note 1)	Left-rotate and mask as specified by the "R" and "L" fields of instruction.	R0 – R17 as specified by the "Destination" field of instruction. (Notes 2 & 3)

MOVE, ADD, AND, XOR, ADD with CARRY (Using left-rotate override R18 – R1F)

SOURCE	PRE-ALU	ALU	POST-ALU	DESTINATION
R0 – R17 as specified by "S" field of instruction.	Right-rotate as specified by "R" field of instruction.	Perform appropriate ALU operation. (Note 1)	Left-rotate and mask as specified by the "D" and "L" fields of instruction. (Note 4)	When R18 – R1F are specified in the "D" field, data is returned to the address specified in the "Source" field.

XMIT 8 (XT8), ADD IMMMEDIATE 8 (AD8), AND IMMEDIATE 8 (AN8), XOR IMMEDIATE 8 (XR8)

SOURCE	PRE-ALU	ALU	POST-ALU	DESTINATION
One to eight bit constants from data field of instruction and register/ address specified by either the source or destination field.	NOP.	Perform appropriate ALU operation. (Note 5)	Mask as specified by the "L" field.	**XT8, AD8:** Results are sent to register/address, starting from LSB, specified by the destination field of the instruction word. **AN8, XR8:** Results are sent to RF, starting from LSB.

XMIT 5 (XT5), ADD IMMMEDIATE 5 (AD5), AND IMMEDIATE 5 (AN5), XOR IMMEDIATE 5 (XR5)

SOURCE	PRE-ALU	ALU	POST-ALU	DESTINATION
One to five bit constants from data field of instruction and register/ address specified by either the source or destination.	Right-rotate as specified by "R" field, the data defined in either the source or destination field.	Perform appropriate ALU operation. (Note 6)	Left-rotate and mask as specified by the "R" and "L" fields.	**XT5, AD5:** Subfield is merged into register/ address specified by destination field. LSB of data field is sent to bit position of destination defined by "R" field. **AN5, XR5:** Subfield is merged into register/ address specified by source field. The "R" field specifies the LSB position of source and RF. Results are returned to RF.

ALL CONDITIONAL AND UNCONDITIONAL RETURNS

ADDRESS REGISTER (AR)	PROGRAM COUNTER (PC)	STACK
Conditional Returns: If condition is true. AR = address from top of stack, else load address from instruction word. **Unconditional Returns:** AR = address from top of stack.	PC = AR.	**Conditional Returns:** If condition is true. POP stack, else NOP. **Unconditional Returns:** POP stack.

ALL CONDITIONAL JUMPS, POP STACK AND JUMP (PSJ), and JUMP (JMP)

ADDRESS REGISTER	PROGRAM COUNTER	STACK
Conditional Jumps: If condition is true. AR = instruction word address, else AR = PC + 1. **PSJ and JMP:** AR = instruction word address.	PC = AR.	**Conditional Jumps:** NOP. **PSJ:** POP stack. **JMP:** NOP.

DATA OPERATIONS OF THE 8X401 (Continued)

CALL (JSR), INTERRUPT ($\overline{INT}$), NON-MASKABLE INTERRUPT ($\overline{NMI}$)

ADDRESS REGISTER	STACK	PROGRAM COUNTER
JSR: AR = instruction address. **$\overline{INT}$:** AR = address 2. **$\overline{NMI}$:** AR = address 1.	PUSH PC + 1. (Note 6)	PC = AR.

XEC

ADDRESS REGISTER	PROGRAM COUNTER	STACK
Right-most L bits of RB merged into corresponding bits in instruction address	PC not updated.	NOP.

NOTES:

1. **ALU Descriptions:**
   - MOVE: • No operation
   - ADD: • Source data ADDed to contents of auxiliary register (R0 – AUX). Carry bit set if carry is generated at MSB of selected data field. NZ status bit set if specified bits are not zero after ALU add.
   - AND: • Source data ANDed to contents of AUX register. NZ status bit updated accordingly.
   - XOR: • Source data Exclusive-ORed with contents of AUX register. NZ status bit set accordingly.
   - ADD with CARRY: • Sum is formed from source data. AUX register, and carry bit (register 10, bit 0). Carry and NZ status bits are set when appropriate.
2. Left-rotate is suppressed when destination is R0 (AUX).
3. When address registers RC, RD and RE are specified in the destination, source data will also go out on banks A, B, C, respectively. The L-field should be zero (a full 8-bit operation) to ensure duplication of the two outputs.
4. A left-rotate of 0 – 7 bits will correspond to R18 – R1F as specified in the "Destination" field of instruction word.
5. **ALU Descriptions:**
   - XMIT: • Input constants from the instruction word to specified destination. NZ flag is not updated when an XMIT is performed: however, NZ can be written to by an XMIT if R17 bit 1 is within the destination field.
   - ADD IMMEDIATE: • Instruction word data is ADDed to data specified by destination fied. The carry bit is set if a carry is generated at the MSB of the selected data field. NZ status bit is updated to reflect the value of "L" bits of data after the addition.
   - AND IMMEDIATE: • Instruction word data is ANDed to data specified by source field. Returning the destination data to RF allows the operation to be performed without destroying the original data field. This will facilitate testing of data for certain pre-defined values while still preserving the original data for other uses. NZ status bit updated accordingly. Unspecified bits in RF remain unchanged.
   - XOR IMMEDIATE: • Same as AND IMMEDIATE, except the logical operation performed is Exclusive-OR.
6. Note that the stack operation is show before the PC in the CALL and INTERRUPT formats. This is because the stack is actually in operation in cycle 3, and the PC is updated in cycle 4 (see Figure 1). In fact, for the Call (JSR) instruction and interrupt servicing, cycle order is important for the user to understand the current status of the PC. The other instructions are in reverse order for visual simplicity in keeping with block diagram flow, and cycle order is irrelevant.

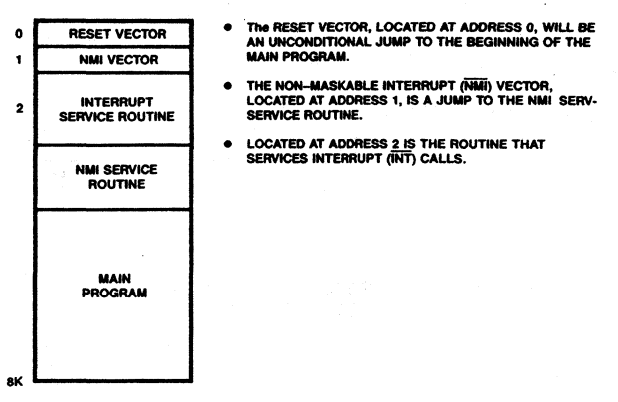
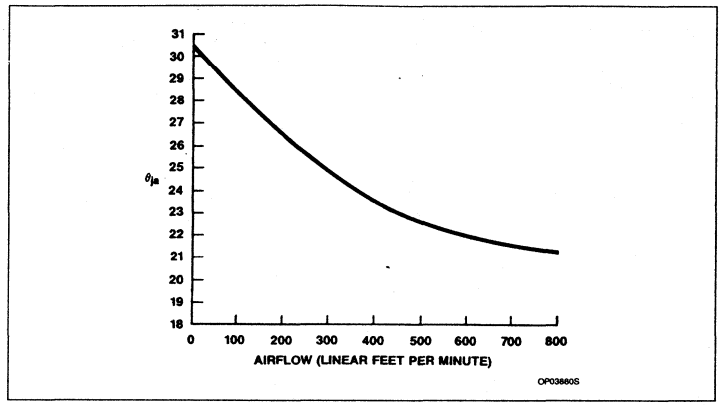


Figure 2. Typical 8X401 System Memory Map

Thermal Junction Temperature vs Airflow

The ceramic package used for the 8X401 has no heat sink and a θ_{ja} rating of 30.5°C/W in still air. Currently, to ensure operation at 150ns, the junction temperature of the devices must be kept below 115°C. The maximum power dissipation at that junction temperature will be 2.4W so that airflow will be required for full commercial range operation. The θ_{ja} versus Airflow curve is drawn here:



DC ELECTRICAL CHARACTERISTICS Commercial Part $4.75V \leqslant V_{CC} \leqslant 5.25V$, $0°C \leqslant T_A \leqslant 70°C$[1]

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT	COMMENTS
			Min	Typ	Max		
V_{CC}	Supply voltage		4.75	5	5.25	V	
V_{IH}	High-level input voltage		2			V	
V_{IL}	Low-level input voltage				0.8	V	
V_{OH}	High-level output voltage	V_{CC} = Min, I_{OH} = −3mA	2.4			V	$\overline{DA0}$ through $\overline{DA7}$, MCLK, SC, WC, AB, BB, CB
		V_{CC} = Min, I_{OH} = −400μA	2.4			V	All others
V_{OL}	Low-level output voltage	V_{CC} = Min, I_{OL} = 16mA			0.5	V	$\overline{DA0}$ through $\overline{DA7}$, MCLK, RWC, SC, WC, $\overline{A}$, $\overline{B}$, $\overline{C}$
		V_{CC} = Min, I_{OL} = 8mA			0.5	V	A0 through A12, PS, NZ, CY, IR
V_{IC}	Input clamp voltage	V_{CC} = Min, I_{IN} = −10mA			−1.5	V	
I_{IH}	High-level input current	V_{CC} = Max, V_I = 2.7			20	μA	
I_{IL}	Low-level input current	V_{CC} = Max, V_I = 0.4V			−400	μA	
I_{OZH}	Off-state output current, high-level voltage applied	V_{CC} = Max, V_O = 2.7V			50	μA	$\overline{DA0}$ through $\overline{DA7}$
I_{OZL}	Off-state output current low-level voltage applied	V_{CC} = Max, V_O = 0.4V			−400	μA	$\overline{DA0}$ through $\overline{DA7}$
I_{OS}	Short circuit output current[2]	V_{CC} = Max, V_O = 0V	−30		−140	mA	
I_{CC}	Supply Current	V_{CC} = Max			500	mA	T_A = 0°C; Cold start[3]
					430	mA	T_J = 115°C

NOTES:
1. 64-pin CDIP, airflow required for commercial operation. (The plastic 64-pin DIP with internal heatsink does not have this requirement.) See above for thermal characteristics.
2. Not more than one output should be tested at a time.
3. Guaranteed by operation to Icc measured at 25°C.

AC ELECTRICAL CHARACTERISTICS (V_{CC} = 5V +5%, o°C ≤ T_A ≤ 70°C)[1]

SYMBOL	PARAMETER	150ns CYCLE			> 150ns CYCLE			UNIT	COMMENTS
		Min	Typ	Max	Min	Typ	Max		
t_{PC}	Processor cycle time	150						ns	
t_{CP}	Clock pulse period	37.5						ns	
t_{CH}	Clock pulse high time	15						ns	See Note 2
t_{CL}	Clock pulse low time	15						ns	
t_{MCL}	CP1 to MCLK low			30				ns	See Note 2
t_{MCH}	CP4 to MCLK high			40				ns	
t_W	MCLK pulse width	25	31	38	T4Q – 12		T4Q	ns	
t_{RWCL}	CP1 to RWC low		34	40				ns	
t_{RWCH}	CP3 to RWC high			40				ns	
t_{RWCW}	RWC pulse width	65	75	80	T3Q + T4Q – 10		T3Q + T4Q + 5	ns	
t_{AS}	CP2 to address stable			52				ns	
t_{MAS}	MCLK to address stable			62			T1Q + 24	ns	
t_{IS}	Instruction setup to CP1	0						ns	See Note 3
t_{MIS}	Instruction setup to MCLK	25						ns	
t_{IH}	Instruction hold from CP2	20						ns	
t_{MIH}	Instruction hold from MLCK	25			T1Q – 12			ns	
t_{SCH}	CP3 to SC rising edge			45				ns	
t_{MSCH}	MCLK to SC rising edge			95			T1Q + T2Q + 20	ns	
t_{WCH}	CP3 to WC rising edge			55				ns	
t_{MWCH}	MCLK to WC rising edge			105			T1Q + T2Q + 20	ns	
t_{WL}	CP1 to SC/WC falling edge			35				ns	See Note 4
t_{MWL}	MCLK to SC/WC falling edge	0						ns	
t_{IBSL}	CP1 to input phase bank signal falling edge			60				ns	
t_{MIBSL}	MCLK to input phase bank signal falling edge			33				ns	
t_{IBSH}	CP3 to input phase bank signal rising edge			45				ns	
t_{MIBSH}	MCLK to input phase bank signal rising edge			95			T1Q + T2Q + 20	ns	
t_{OBSL}	CP3 to output phase bank signal falling edge			53				ns	
t_{MOBSL}	MCLK to output phase bank signal falling edge			105			T1Q + T2Q + 30	ns	
t_{OBSH}	CP1 to output phase bank signal rising edge			46				ns	
t_{MOBSH}	MCLK to output phase bank signal rising edge	0		20				ns	
t_{IDS}	Input data setup to CP3	–3						ns	
t_{MIDS}	Input data setup to MCLK	–50			25 – T1Q – T2Q			ns	
t_{IDH}	Input data hold from CP3	28						ns	
t_{MIDH}	Input data hold from MCLK	78			T1Q + T2Q + 3			ns	
t_{ODH}	Output data hold from CP1	35		55				ns	
t_{MODH}	Output data hold from MCLK	10		25				ns	

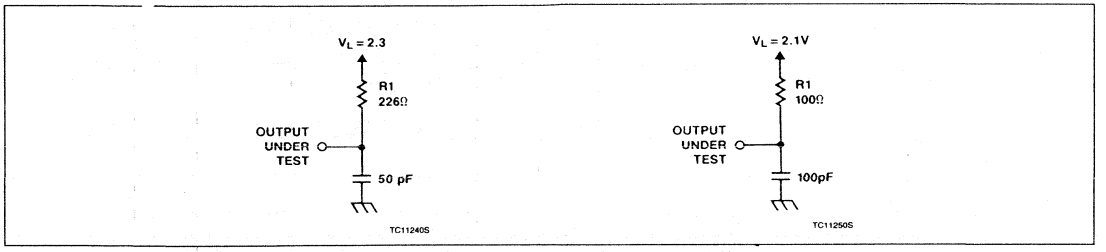
AC ELECTRICAL CHARACTERISTICS (Continued)

SYMBOL	PARAMETER	150ns CYCLE			> 150ns CYCLE			UNIT	COMMENTS
		Min	Typ	Max	Min	Typ	Max		
t_{ODS}	CP3 to output data stable			70				ns	
t_{MODS}	MCLK to output data stable			120			T1Q + T2Q + 45	ns	
t_{DI}	SC/WC rising edge to output driver turn on	18						ns	
t_{HS}	Halt setup to CP2	0						ns	
t_{MHS}	Halt setup to MCLK	−10						ns	
t_{HH}	Halt hold from CP2	50						ns	
t_{MHH}	Halt hold from MCLK	60			T1Q + 22			ns	
t_{SIS}	Status input setup to CP1	10						ns	
t_{MSIS}	Status input setup to MCLK	40						ns	
t_{SIH}	Status input hold from CP1	20						ns	
t_{MSIH}	Status input hold from MCLK	0						ns	
t_{SCRS}	SCR setup to CP1	0						ns	
t_{MSCRS}	SCR setup to MCLK	25						ns	
t_{SCRH}	SCR hold from CP2 (Slow CP2)	20						ns	See Diagram for CP2
t_{MSCRH}	SCR hold from MCLK	63			ST1Q − 12			ns	Slow T1Q
t_{INTS}	INT setup to CP1	10						ns	
t_{MINTS}	INT setup to MCLK	40						ns	
t_{INTH}	INT hold from CP1	10						ns	
t_{MINTH}	INT hold from MCLK	−10						ns	
t_{CYU}	CP4 to CY update			60				ns	
t_{MCYU}	MCLK to CY update			−10			28 − T4Q	ns	
t_{NZU}	CP4 to NZ update			60				ns	
t_{MNZU}	MCLK to NZ update			−5			33 − T4Q	ns	
t_{IRU}	CP4 to IR update			75				ns	
t_{MIRU}	MCLK to IR update			20			58 − T4Q	ns	
t_{PSU}	CP4 to PS update			60				ns	
t_{MPSU}	MCLK to PS update			−10			28 − T4Q	ns	
t_{ACC}	Program memory access time (address stable to valid instruction)			60		T2Q + T3Q + T4Q − 52			ns
t_{IO}	I/O port output enable time (bank signal to valid data on bus)			24			T1Q + T2Q − 51	ns	
t_{RW}	Reset pulse width	150			t_{PC}			ns	
t_{NMIW}	NMI pulse width	50						ns	
t_{NMIS}	NMI setup to CP2	15						ns	See Note 5
t_{MNMIS}	NMI setup to MCLK	10						ns	See Note 5

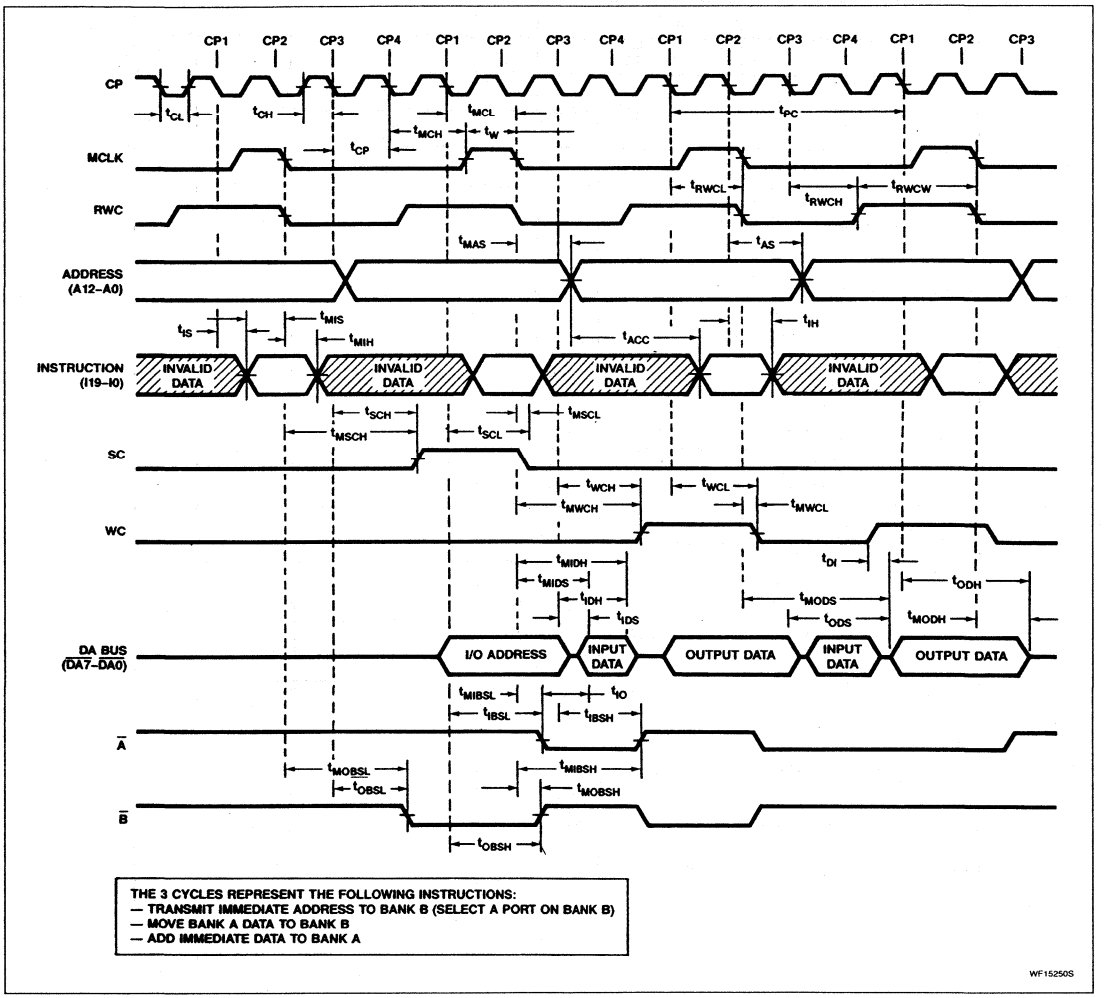
NOTES:

1. Inputs swing between 0V and 3V. All outputs are measured at 1.5V with loading as specified in the test circuits.
2. CP1, CP2, CP3, and CP4 refer to the clock pulse that causes the first, second, third, and fourth 8X401 quarter cycles, respectively. Parameters referenced to MCLK, CP1, CP2, CP3, and CP4 are measured to the falling edge of those signals. T1Q, T2Q, T3Q, and T4Q represent time intervals for the first, second, third, and fourth 8X401 quarter cycles, respectively. Duty cycle can be from 40% to 60%.
3. Instructions must be setup before CP1.
4. t_{WL} represents t_{SCL} and t_{WCL}. t_{MWL} represents t_{MSCL} and t_{MWCL}.
5. This guarantees NMI is serviced in the current cycle.

TEST CIRCUIT



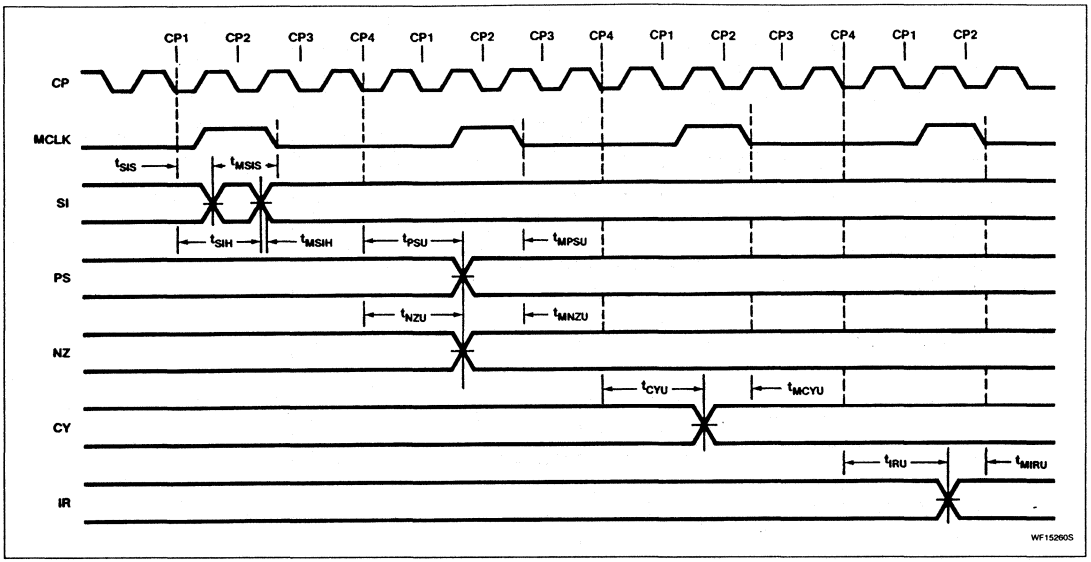
MAIN TIMING DIAGRAM



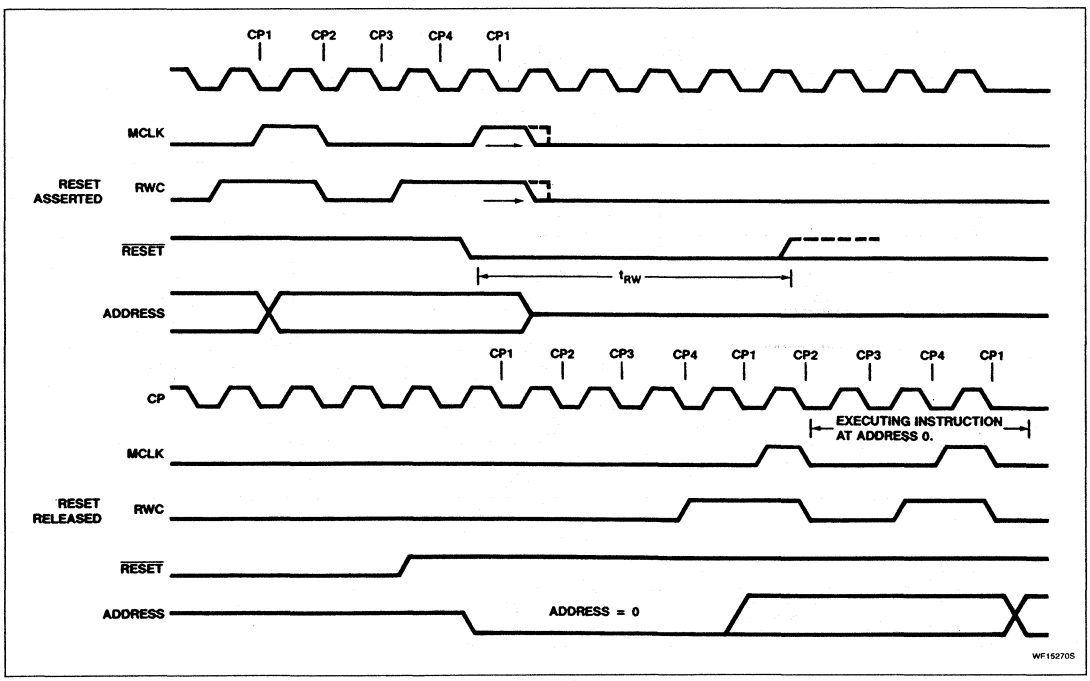
THE 3 CYCLES REPRESENT THE FOLLOWING INSTRUCTIONS:
— TRANSMIT IMMEDIATE ADDRESS TO BANK B (SELECT A PORT ON BANK B)
— MOVE BANK A DATA TO BANK B
— ADD IMMEDIATE DATA TO BANK A

WF15250S

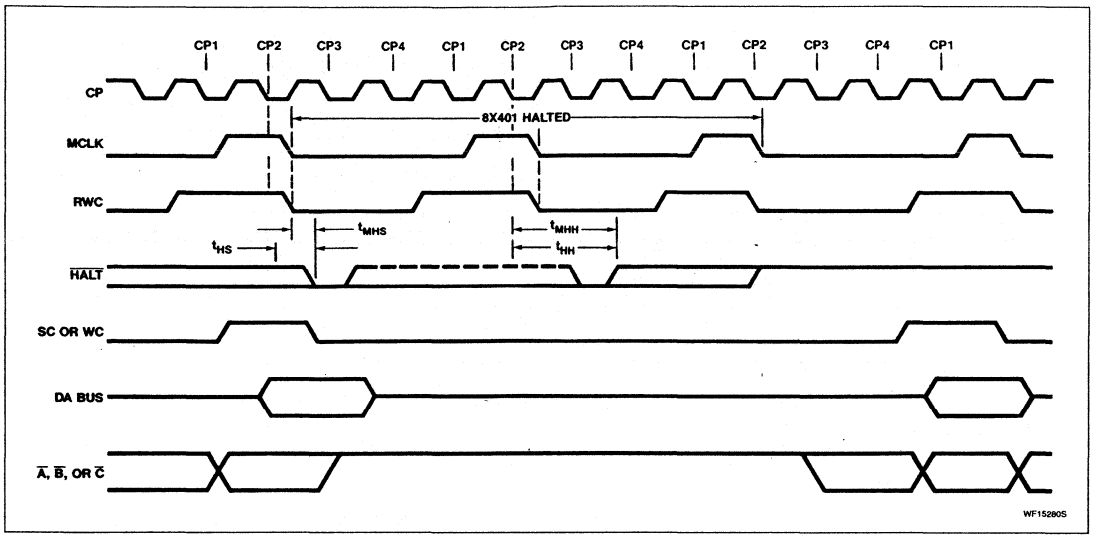
TIMING SUPPLEMENT: STATUS



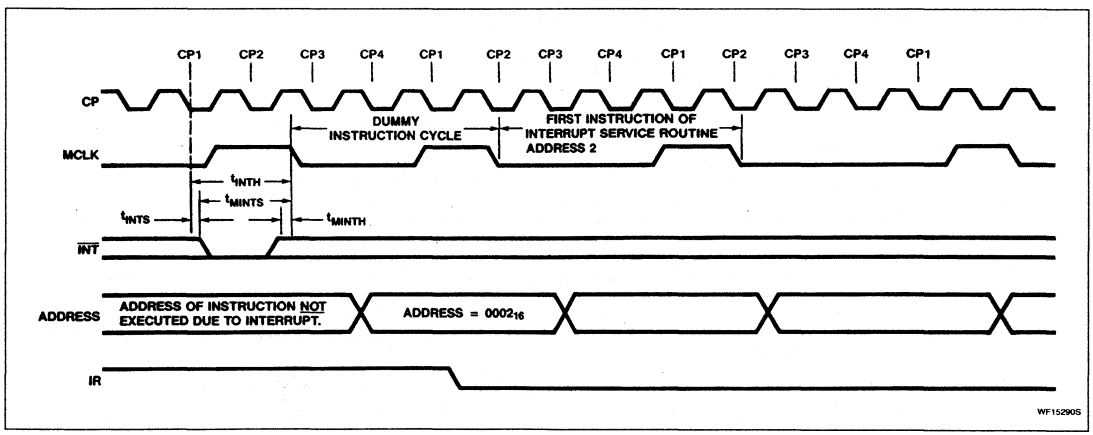
TIMING SUPPLEMENT: $\overline{\text{RESET}}$



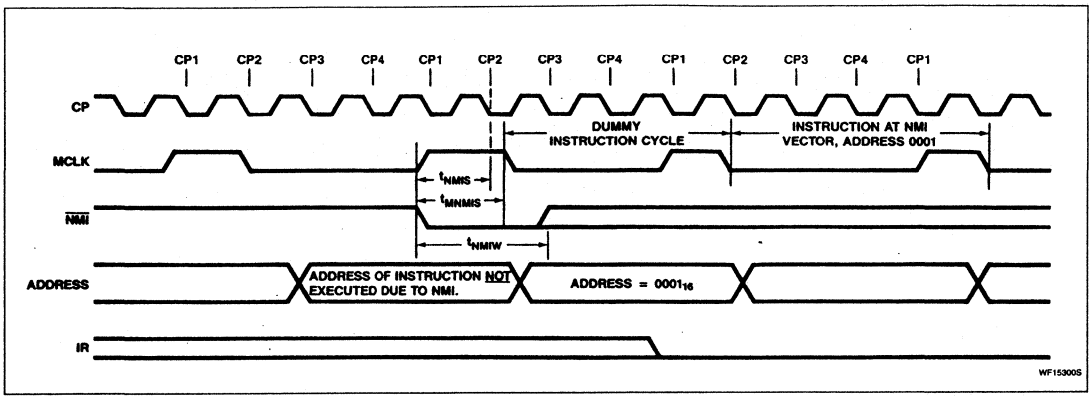
TIMING SUPPLEMENT: $\overline{\text{HALT}}$



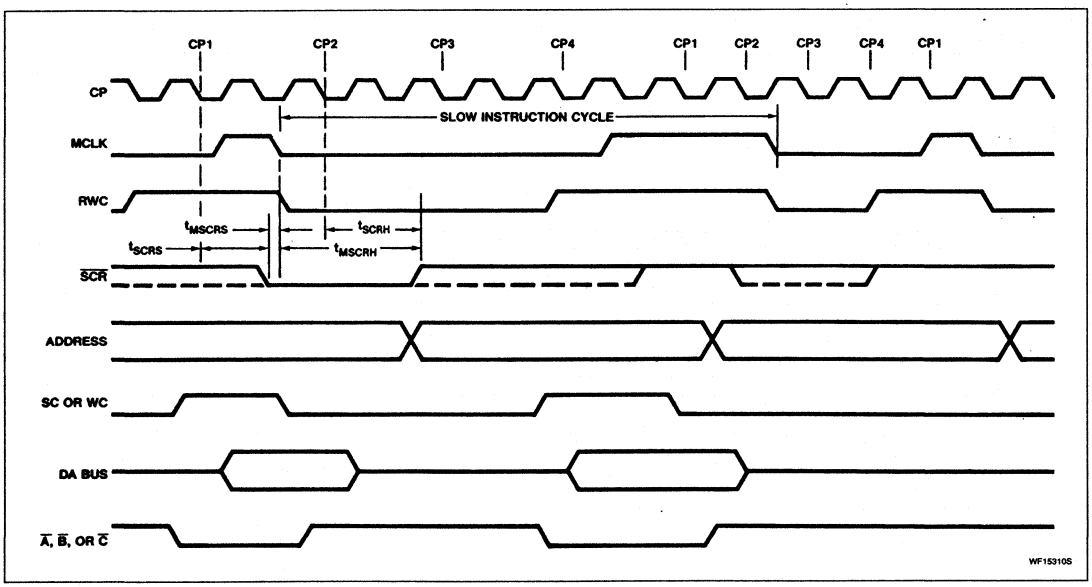
TIMING SUPPLEMENT: $\overline{\text{INT}}$



TIMING SUPPLEMENT: $\overline{NMI}$



TIMING SUPPLEMENT: $\overline{SCR}$



Philips Components

Date of Issue	July 5, 1990
Status	Preliminary Specification
Application Specific Product	

82C200

Stand-alone CAN controller

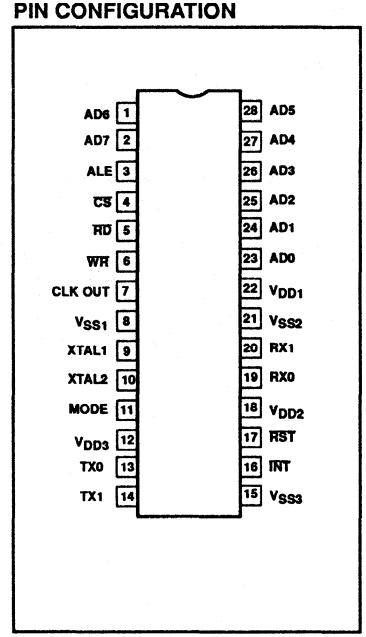
DESCRIPTION

The 82C200 is a highly integrated stand-alone controller for the controller area network (CAN) used within automotive and general industrial environments. The 82C200 contains all necessary features required to implement a high-performance communication protocol. The 82C200 with a simple bus line connection performs all the functions of the physical and data-link layers. The application layer of an electronic control unit (ECU) is provided by a microcontroller, to which the 82C200 provides a versatile interface. The use of the 82C200 in an automotive or industrial environment results in a reduced wiring harness and an enhanced diagnostic and supervisory capability.

FEATURES

- Multi-master architecture
- Interfaces with a large variety of microcontrollers (Intel, Motorola or Intel and Motorola compatible)
- Bus access priority (determined by the message identifier)
- 2032 message identifiers
- Guaranteed latency time for high-priority messages
- Powerful error handling capability
- Data length from 0 up to 8 bytes
- Configurable bus interface
- Multicast and Programmable clock output
- Broadcast message facility
- Nondestructive bit-wise arbitration
- Non-return-to-zero (NRZ) coding/decoding with bit-stuffing
- Programmable transfer rate (up to 1 Mbit/s)
- Programmable output driver configuration
- Suitable for use in a wide range of networks, including the SAE networks Class A, B, and C
- Frequency range: 1, 2 to 16MHz
- Temperature range: –40 to +125°C

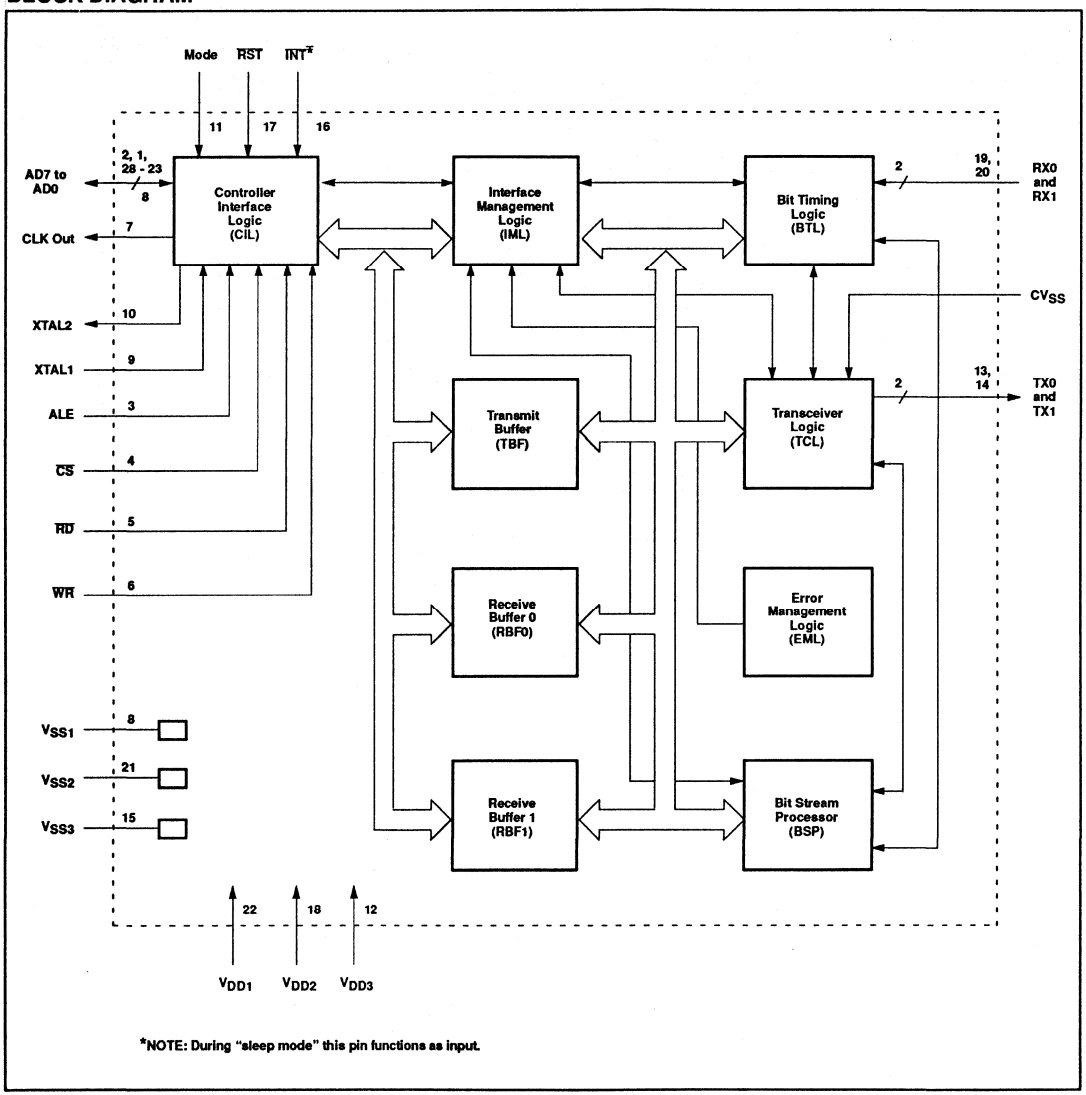
PIN CONFIGURATION



PART NUMBER SELECTION

EXTENDED TYPE NUMBER	PACKAGE PINS	PIN ARRANGEMENT	MATERIAL	CODE
PCA82C200P	28	DIL	Plastic	SOT117
PCA82C200T	28	SO28	Plastic	SOT136

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	DESCRIPTION
AD7 – AD0	2, 1, 28 – 23		Multiplexed address/data bus.
ALE	3		ALE signal (Intel mode) or AS input signal (Motorola mode).
$\overline{CS}$	4		Chip select input. Low level allows access to the PCA82C200.
$\overline{RD}$	5		$\overline{RD}$ signal (Intel mode) or E enable signal (Motorola mode) from the microcontroller.
$\overline{WR}$	6		$\overline{WR}$ signal (Intel mode) or RD/WR signal (Motorola mode) from the microcontroller.
CLK OUT	7		Clock output signal produced by the PCA82C200 for the microcontroller. The clock signal is derived from the built-in oscillator, via the programmable divider. This output is capable of driving one CMOS or NMOS load.
V_{SS1}	8		Ground potential for the logic circuits.
XTAL1	9		Input to the oscillator's amplifier. External oscillator signal is input via this pin.
XTAL2	10		Output from the oscillator's amplifier. Output must be left open when an external oscillator signal is used.
MODE	11		Mode select input: Connected to V_{DD} selects Intel mode; connected V_{SS} selects Motorola mode.
V_{DD3}	12		5V power supply for the output driver.
TX0	13		Output from the output-driver 0 to the physical bus line.
TX1	14		Output from the output-driver 1 to the physical bus line.
V_{SS3}	15		Ground potential for the output driver.
$\overline{INT}$	16		Interrupt output, used to interrupt the microcontroller. $\overline{INT}$ is active if the Interrupt Register contains a logic High bit (present). $\overline{INT}$ is an open-drain output and is designed to be wired-OR with other $\overline{INT}$ outputs within the system. A Low level on this pin will reactivate the IC from the sleep mode.
$\overline{RST}$	17		Reset input, used to reset the CAN interface (Low level). A schmitt trigger at the input is used for noise rejection. Automatics power-on reset can be obtained by connecting $\overline{RST}$ via a capacitor to V_{SS} and via a resistor to V_{DD}.
V_{DD2}	18		5V power supply for the input comparator.
RX0 – RX1	19, 20		Input from the physical bus line to the input comparator of the PCA82C200. A dominant level will wake up the PCA82C200. A recessive level is read if RX0 is higher than RX1 and vice versa for the dominant level.
V_{SS2}	21		Ground potential for the input comparator.
V_{DD1}	22		5V power supply for the logic circuits.

FUNCTIONAL DESCRIPTION

Functional Overview

The 82C200 includes all hardware modules necessary to implement the transfer layer which represents the kernel of the CAN protocol and a microcontroller interface with clock generator, as shown in the block diagram (above).

Interface Management Logic (IML)

The IML interprets the commands from the microcontroller, controls the allocation of the message buffers Transmit Buffer (TBF), Receive Buffer 0 (RBF0), and Receive Buffer 1 (RBF1), and provides interrupts and status information to the microcontroller via the Control Interface Logic (CIL).

Transmit Buffer (TBF)

The TBF is an interface between the microcontroller and the Bit Stream Processor (BSP) and is able to store a complete message. The buffer is written by the microcontroller and read by the BSP. On a microcontroller's Transmission Request, the BSP has the exclusive access to the TBF. The TBF is "released" to the microcontroller by BSP after the message transfer has been completed or aborted.

The TBF is 10 bytes long to hold the Descriptor (2 bytes) and the Data-Field (up to 8 bytes) of the message. The buffer is implemented as a single ported RAM with mutually exclusive access by the microcontroller and BSP.

Receive Buffer (RBF0, RBF1)

The RBF is an interface between the BSP and the microcontroller and stores a message received from the bus line. Once filled by BSP and allocated to the microcontroller by the IML, the buffer cannot be used to store subsequently received messages until the microcontroller has (read and) released the buffer. Thus, unless the microcontroller releases an RBF within a protocol-defined time, messages to be received may be lost.

To reduce the requirements on the microcontroller, two receive buffers (RBF0, RBF1) are implemented. While one RBF is allocated to the microcontroller, the BSP may write to the other one. Both RBF0 and RBF1 are 10 bytes long to hold the Descriptor (2 bytes) and the Data-Field (up to 8 bytes) of the message. The buffers are implemented as single ported RAM with mutually exclusive access from the microcontroller and BSP. The BSP only writes into a receive buffer when the message being received has an Identifier which passes the Acceptance Filter.

Bit Stream Processor (BSP)

This is a sequencer controlling the data stream between transmit and receive buffers (parallel data) and the bus line (serial data). The BSP contains the Acceptance Filter and also controls the Transceiver Logic (TCL) and the Error Management Logic (EML) such that the processes of reception, arbitration, transmission, and error signalling are performed according to the protocol. The BSP provides signals to the IML indicating when a message has got acceptance, when a receive buffer contains a valid message, and also when the transmit butter is no longer required after a successful transmission. The automatic retransmission of messages in error cases is effectively handled by the BSP.

Bit Timing Logic (BTL)

This block monitors the bus line using the (built-in) Input Comparator and handles the busline-related bit timing.

The BTL synchronizes on a "recessive" to "dominant" bus line transition at the beginning of a message (hard synchronization) and resynchronizes on further transitions during the reception of a message (soft synchronization). A microcontroller programmable control bit (Speed Mode) determines which edges are used for resynchronization.

The BTL also provides programmable time segments to compensate for the propagation delay times and phase shifts (e.g., due to oscillator drifts) and to define the sampling time and the number of samples (one or three) to be taken within a bit time.

Transceiver Logic (TCL)

This is a generic term for a group of logic elements consisting of a programmable Output Driver, bit stuff logic, CRC logic, and the Transmit Shift Register. The coordination of these components is done by the BSP.

Error Management Logic (EML)

The EML is responsible for the error confinement of the transfer-layer modules. The EML gets error announcements from BSP and then informs the BSP, TCL, and IML about error statistics.

Controller Interface Logic (CIL)

The CIL links the transfer-layer modules to external microcontrollers. It can directly interface with most commonly used microcontrollers. The CIL includes the oscillator and a programmable clock divider to provide the Clockout signal for the microcontroller. The oscillator is a high gain, parallel resonance circuit with a frequency range of up to 24MHz.

The CIL receives the various signals for wakeup/sleep inhibit from the rest of the circuit and the "go to sleep" signal from the IML and is responsible for ensuring the 15-bit time overrun of the Clockout signal from entering sleep mode. This is to enable the attached microcontroller to execute its halt instruction after setting the Goto Sleep control bit to "sleep" and to ensure that the bus is really idle.

The BSP, TCL BTL, and EML are denoted with "Bus Line Related Logic," whereas IML together with CIL, TBF, RBF0, and RBF1 are denoted with "Microcontroller Related Logic."

Description of the Control Segment and the Message Buffers

For communication, the 82C200 has to be configured by the microcontroller in an initialization download after power-up. This is done by programming the control segment (see Figure 1). On programming the Clock Divider Register, the microcontroller may define an appropriate external clock signal, offered at the 82C200 Clockout pin (see the Block Diagram, above). Transmission is carried out by writing the respective message into the Transmit Buffer and setting Transmission Request to "present" from a microcontroller's point of view. A successful transmission is detected by an 82C200's Transmit Interrupt and/or on polling the Transmission Complete Status bit. On reception of a message, the microcontroller gets informed from the 82C200 by a Receive Interrupt and/or on polling the Status Register. The microcontroller then may read out the message from the Receive Buffer and afterwards should release the Receive Buffer for further use by BSP.

The 82C200 appears to a microcontroller as a memory-mapped I/O device due to the on-chip RAM, guaranteeing the independent operation of both devices.

Address Allocation

The address area of the 82C200 consists of the Control Segment and the message buffers. The Control Segment is programmed during an initialization download in order to configure communication parameters (e.g., bit timing). Communication over the CAN bus is also controlled via this segment by the microcontroller. During initialization the Clock Out signal may be programmed to a value determined by the microcontroller. A message which is to be transmitted must be written to the Transmit Buffer. After a successful reception the microcontroller may read the message from the Receive Buffer and then release it for further use.

Control-Segment Layout

The interchange of commands, status and controls signals between the microcontroller and the 82C200 takes place via the control segment. The layout of this segment is shown in Figure 1.

It is not foreseen that during normal operation of the 82C200 the contents of the Acceptance Code Register, the Acceptance Mask Register, the Bus Timing Register0, the Bus Timing Register1, and the Output Control Register will be changed after the initial download. Thus, these registers are only accessible when Reset Request in the Control Register is set to "present".

Control Register (CR)

MSB 7	6	5	4	3	2	1	LSB 0
TM	S	–	OIE	EIE	TIE	RIE	RR

The contents of the Control Register are used to change the state and behavior of the 82C200. Control bits may be set and reset by the attached microcontroller only. The 82C200's Control Register is implemented as a "read/write memory" from the microcontroller's point of view.

RR (Reset Request) CR.0

- High: The 82C200 aborts the current transmission or reception of a message and enters the reset state.
- Low: Normal operation.

During External Reset (low level on the pin $\overline{RST}$) or when the Bus Status changes to Off-Bus, the IML forces Reset Request to High. During an External Reset, the microcontroller is not able to program Reset Request to Low. Therefore, after having set Reset Request to Low, the microcontroller has to check this bit to ensure that External Reset is not still holding it High.

After Reset Request is Low, the 82C200 will wait for Bus Free in the cases of an External Reset or a microcontroller-initiated reset and for 128 occurrences of Bus Free after an "off-bus" initiated reset before going "on-bus".

When Reset Request is set High (present), for whatever reason, the control, command, status, and interrupt bits are affected. See the table below.

When Reset Request is set High (present), the registers at addresses 4 to 8 are accessible but the TBF is not.

Effects of Setting the Reset Request Bit High (Present)

TYPE	BIT	EFFECT
Control	Test Mode	Low (disabled)
Command	Goto Sleep	Low (wake-up)
	Clear Overrun Status	High (clear)
	Release Receive Buffer	High (released)
	Abort Transmission	Low (absent)
	Transmission Request	Low (absent)
Status	Bus Status	Low (bus ON), only after external power-up reset
	Error Status	Low (no error), only after external power-up reset
	Transmit Status	Low (idle)
	Receive Status	Low (idle)
	Transmission Complete Status	High (complete)
	Transmit Buffer Access	High (released)
	Data Overrun	Low (absent)
	Receive Buffer Status	Low (empty)
Interrupt	Overrun Interrupt	Low (reset)
	Transmit Interrupt	Low (reset)
	Receive Interrupt	Low (reset)

RIE (Receive Interrupt Enable) CR.1

- High: Microcontroller gets an interrupt if a message has been received free of errors.
- Low: Microcontroller gets no interrupt of a reception.

TIE (Transmit Interrupt Enable) CR.2

- High: Microcontroller gets an interrupt when a message has been successfully transmitted and theTransmit Buffer is accessible again.
- Low: Microcontroller gets no transmit interrupt.

EIE (Error Interrupt Enable) CR.3

- High: Microcontroller gets an interrupt if Error Status or Bus Status changes.
- Low: Microcontroller gets no error interrupt.

OIE (Overrun Interrupt Enable) CR.4

- High: Microcontroller gets an interrupt if Data Overrun gets set.
- Low: Microcontroller gets no overrun interrupt.

– (Reserved) CR.5

SM (Speed Mode) CR.6

- High: Slow: Bus line transition from recessive to dominant and vice versa are used for resynchronization.
- Low: Fast: Only transitions from recessive to dominant are used for resynchronization.

TM (Test Mode) CR.7

- High: The 82C200 enters test mode.
- Low: Normal operation.

Command Register (CMR)

MSB 7	6	5	4	3	2	1	LSB 0
–	–	–	GTS	COS	RRB	AT	TR

A command bit initiates an action within the transfer layer of the 82C200.

From a microcontroller's point of view, this register is implemented as write only memory. If a read access is performed to this address, the byte 11111111 bin is returned.

TR (Transmission Request) CMR.0

- High: A message shall be transmitted.
- Low: No action.

Note that if in a previous command a transmission was requested (the TR bit was set high), then this request will not be cancelled by writing "no action" (setting the TR bit low) into this command bit. To cancel a Transmission Request, use the Abort Transmission command bit.

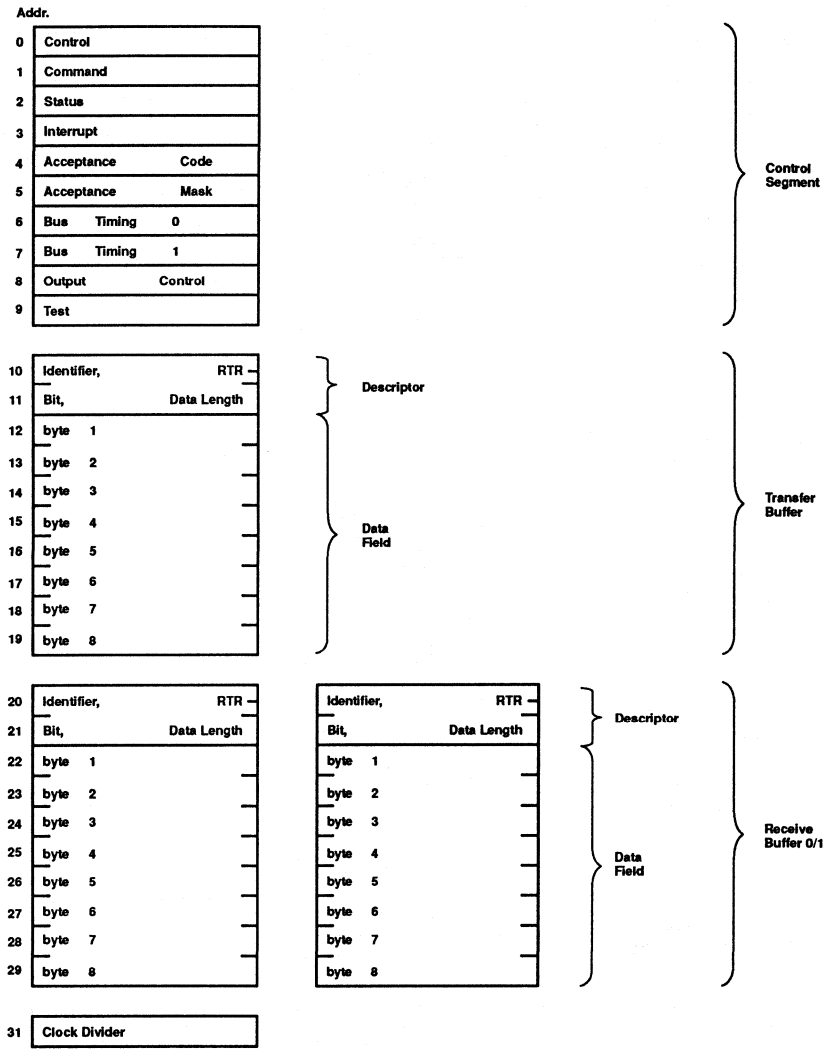


Figure 1. Address Allocation of the 82C200

AT (Abort Transmission) CMR.1

High A pending transmission request is cancelled if it is not al- ready in progress.

Low No action.

The Abort Transmission command is used when the microcontroller wants to suspend the previously requested transmission, for instance, to transmit a more urgent message. A transmission already in progress is not aborted. To know whether the original message has been sent anyway, the status bit Transmission Complete Status should be checked.

RRB (Release Receive Buffer) CMR.2

High The Receive Buffer attached to the microcontroller is released.

Low No action.

After having read the content of the Receive Buffer which is attached to the microcontroller, the microcontroller has to release this buffer by setting Release Receive Buffer to "released".

COS (Clear Overrun) CMR.3

High The Data Overrun status bit is cleared.

Low No action.

This command is used to acknowledge the Data Overrun condition signalled by the Data Overrun status bit. It may be given at the same time as a Release Receive Buffer command.

GTS (Goto Sleep) CMR.4

High The chip will go into sleep mode if the interrupt pin ($\overline{INT}$) is high (no interrupt from the 82C200 or external sources pending) and there is no bus activity.

Low The 82C200 can function normally and the oscillator and Clockout will run.

The 82C200 will wake up if either Goto Sleep is set to Low or there is bus activity, or the interrupt pin is driven low (active). On wakeup, the oscillator is started and a wakeup interrupt is generated.

An 82C200 which is sleeping and is awakened by bus activity will not be able to receive the current or following messages until it detects Bus Free.

– (Reserved) CMR.5

– (Reserved) CMR.6

– (Reserved) CMR.7

Status Register (SR)

MSB 7	6	5	4	3	2	1	LSB 0
BS	ES	TS	RS	TCS	TBS	DO	RBS

The status bits reflect the actual status of the 82C200 bus controller.

The status register is read by the microcontroller and modified by IML only. This register is implemented as a read only memory from the microcontroller's point of view.

RBS (Receive Buffer Status) SR.0

High Full: This bit is set by IML when a new message is available.

Low Empty: No message has become available since the last Release Receive Buffer command.

If the command bit Release Receive Buffer is set to released by the microcontroller, Receive Buffer Status is set to empty by IML. When a new message is stored in any receive buffer, the Receive Buffer Status is set to full again.

DO (Data Overrun) SR.1

High This bit is set to overrun when both receive buffers are full and the first byte of another message has got acceptance and thus should be stored.

Low When the microcontroller sets Clear Overrun to clear, IML sets this bit to absent.

If overrun is signalled, the currently received message is dropped.

A transmitted message, granted acceptance, is also stored in a receive buffer. This occurs because it is not known if the 82C200 will lose arbitration and become a receiver of the message. If no receive buffer is available, Data Overrun is signalled.

TBS (Transmit Buffer Access) SR.2

High The transmit buffer may be written by the microcontroller.

Low The microcontroller cannot access the transmit buffer. A message is either waiting for transmision or is in the process of being transmitted.

If the microcontroller tries to write to the transmit buffer when Transmit Buffer Access is locked, the written bytes will not be accepted and will be lost without this being signalled.

TCS (Transmission Complete Status) SR.3

High Last requested transmission has been successfully completed.

Low Previously requested transmission is not yet completed.

RS (Receive Status) SR.4

High The 82C200 is receiving a message.

Low After a Bus Free, the 82C200 entered receive-idle or transmit mode.

TS (Transmit Status) SR.5

High The 82C200 started to transmit a message.

Low No message is being transmitted.

The CAN bus is idle if both Receive Status and Transmit Status are idle.

ES (Error Status) SR.6

High At least one of the error counters of the EML has reached the microcontroller Warning Limit.

Low Neither the receive-error counter nor the transmit-error counter of the EML has reached the microcontroller Warning Limit (96 error points each).

BS (Bus Status) SR.7

High Off-bus: The 82C200 does not take part in bus activities.

Low On-bus: The 82C200 takes part in bus activities.

When the Bus Status goes to off-bus, the 82C200 will set Reset Request to High. It will stay in this state until the microcontroller sets Reset Request to Low.

Interrupt Register (IR)

MSB 7	6	5	4	3	2	1	LSB 0
–	–	–	WUI	OI	EI	TI	RI

The 82C200's interrupt register makes it possible to identify the source of an interrupt without checking status bits and the corresponding interrupt enable bits. When the IML sets one or more bits of this register, an interrupt is given to the microcontroller by activating the INT pin. All bits are reset by the 82C200 after this register is read by the microcontroller.

The interrupt register is read by the microcontroller and modified by IML only. It is implemented as a read only memory from the microcontroller's point of view.

RI (Receive Interrupt) IR.0

High This bit is set by IML, when a new message is available in the Receive Buffer and Receive Interrupt Enable is enabled.

Low Receive Interrupt is reset automatically by a read access of the microcontroller to the interrupt register.

Note: Receive Interrupt (if enabled) and Receive Buffer Status are set in coincidence.

TI (Transmit Interrupt) IR.1

High This bit is set whenever Transmit Buffer Access is set to released and Transmit Interrupt Enable is set to enabled.

Low Transmit Interrupt will be reset after a read access of the microcontroller to the interrupt register.

EI (Error Interrupt) IR.2

High When Error Interrupt Enable is set to enabled, this bit is set on a change of Error Status or Bus Status.

Low Error Interrupt is reset by a read access of the microcontroller to the interrupt register

OI (Overrun Interrupt) IR.3

High This bit is set whenever both receive buffers contain a valid message and the first byte of another message should be stored.

Low Overrun Interrupt is reset by a read access of the microcontroller to the interrupt register.

WUI (Wakeup Interrupt) ER.4

High When the 82C200 is setting Goto Sleep to wakeup after it had been previously set to sleep by the microcontroller, this bit is set.

Low Wakeup Interrupt is reset by a read access of the microcontroller to the interrupt register

– (Reserved) ER.5

– (Reserved) ER.6

– (Reserved) ER.7

Overrun Interrupt and Data Overrun are set in coincidence.

To clear the overrun condition (Data Overrun is Low), use the Clear Overrun command.

Acceptance Code Register (ACR)

MSB 7	6	5	4	3	2	1	LSB 0
AC.7	AC.6	AC.5	AC.4	AC.3	AC.2	AC.1	AC.0

The acceptance code register in conjunction with the acceptance mask register determines whether a message correctly received over the bus line will be accepted or dropped. When a message is received which passes the accepting condition described below and there is a free receive buffer, then the respective Descriptor and the Data Field (see Figure 1) are sequentially stored in this buffer; otherwise, in the case of no empty receive buffer being available, the Data Overrun bit will be set.

On acceptance, Receive Buffer Status is set to High, and if Receive Interrupt Enable is set to enabled, Receive Interrupt is set to High.

The condition for acceptance is that the acceptance code bits (AC.7 – AC.0) and the eight most significant bits of the message's Identifier (ID.10 – ID.3) are equal at least on those bit positions being marked relevant by the acceptance mask bits (AM.7 – AM.0). If the following equation is satisfied, acceptance is given:

((ID.10 – ID.3) .equal.
(AC.7 – AC.0)) .or. .not.
(AM.7 – AM.0) = '11111111' bin.

Acceptance Mask Register (AMR)

MSB 7	6	5	4	3	2	1	LSB 0
AM.7	AM.6	AM.5	AM.4	AM.3	AM.2	AM.1	AM.0

The Acceptance Mask Register qualifies which of the corresponding bits of the acceptance code are relevant (Low) or do not care (High) for acceptance filtering.

AM.7 – AM.0

High This bit position is do not care (High) for the acceptance of a message.

Low This bit position is relevant (Low) for acceptance filtering.

Bus Timing Register 0 (BTR0)

MSB 7	6	5	4	3	2	1	LSB 0
SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0

By Bus Timing Register 0 the Baud Rate Prescaler and the Synchronization Jump Width are programmed.
BRP0–BRP5 (Baud Rate Prescaler)

The length of the system clock cycle time T_{SCL}, which is used to build up the individual bit timing, is programmable via the Baud Rate Prescaler.

t_{SCL} = (32BRP.5 + 16BRP.4 + 8BRP.3 + 4BRP.2 + 2BRP.1 + BRP.0 + 1) $2t_{OSC}$

t_{OSC}: period of the 82C200 oscillator

SJW0–SJW1 (Synchronization Jump Width)

To compensate phase shifts between clock oscillators of different bus controllers, any bus controller has to resynchronize on any relevant signal edge of the current transmitter.

The Synchronization Jump Width defines the maximum number of clock cycles a bit time may be shortened or lengthened by one resynchronization.

t_{SJW} = (2SJW.1 + SJW.0 + 1) t_{SCL}

Bus Timing Register 1 (BTR1)

MSB	7	SAM
	6	TSEG2.2
	5	TSEG2.1
	4	TSEG2.0
	3	TSEG1.3
	2	TSEG1.2
	1	TSEG1.1
LSB	0	TSEG1.0

By Bus Timing Register 1 the propagation delay times (to be compensated), the sample point, and the 82C200 internal transmit point, all within a bit time, are programmed.

SAM (Sampling)

SAM determines whether the bus line is sampled directly within the BTL or whether the bus line is filtered by a majority logic:

SAM = 0: bus sampled once

SAM = 1: three samples taken

SAM = 0 is recommended at high speed busses, while with SAM = 1, spikes on the bus line can be filtered out.

TSEG2.2–TSEG1.0 (Time Segments)

These are time segments within the bit time to fix the number of clock cycles per bit time and the location of the sample point.

t_{TSEG1} = (8TSEG1.3 + 4TSEG1.2 + 2TSEG1.1 + TSEG1.0 + 1) t_{SCL}

t_{TSEG2} = (4TSEG2.2 + 2TSEG2.1 + TSEG2.0 + 1) t_{SCL}

Output Control Register (OCR)

MSB	7	OCTP1
	6	OCTN1
	5	OCPOL1
	4	OCTP0
	3	OCTN0
	2	OCPOL0
	1	OCMODE1
LSB	0	OCMODE0

The output control register allows you to set up different output driver configurations of the 82C200 under software control. The mnicro-controller writes into this register and the IML reads it.

If the 82C200 is in the sleep mode (Goto Sleep = High), a recessive level is output on the TX0 and TX1 pins. If the 82C200 is in the reset state (Reset Request = High), the output drivers are floating.

Normal Mode
The voltage levels on the output driver pins TX1, TX0, assigned to dominant and recessive states, depend on both the driver characteristic programmed by OCTPx, OCTNx (pull-up, pull-down, push-pull) and the output polarity programmed by OCPOLx. In contrast to biphase mode the bit representation is time invariant and not toggled.

Biphase Mode
If the bus controllers are galvanically decoupled from the bus line by a transformer, the bit stream has to be coded in such a way that there is no resulting DC component. During recessive bits, all outputs are deactivated. Dominant bits are sent alternatingly on TX0 and TX1; i.e., the first dominant bit is sent on TX0, the second is sent on TX1, and the third one is sent on TX0 again, and so on.

Test Mode
Within the bit timing, only the sum of the propagation delays of input comparator and output driver has to be taken into account. To simplify production testing in test mode, the output of the input comparator (COMPOUT) is directly driven to the input of the output driver TX1, whereas TX0 outputs the bit sequence (txd).

Tables 1 and 2 show the relation between the bits of the Output Control Register and the two serial output pins TX0 and TX1 of the 82C200, connected to the serial bus (see the Block Diagram, above).

Transmit Buffer Layout

The global layout of the transmit buffer is shown in Figure 1. This buffer serves to store a message from the microcontroller and to be transmitted by the 82C200. It is subdivided into Descriptor and Data Field. The Transmit Buffer is written by the microcontroller only and read by the IML.

Descriptor Byte1 (DSCR1)

MSB 7	6	5	4	3	2	1	LSB 0
ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3

Descriptor Byte2 (DSCR2)

MSB 7	6	5	4	3	2	1	LSB 0
ID.2	ID.1	ID.0	RTR	DLC3	DLC2	DLC1	DLC0

ID0–ID10 (Identifier)

The identifier consists of 11 bits). ID.10 is the most significant bit, which is transmitted first on the bus during the arbitration process. The identifier both is the message's name used in a receiver for acceptance filtering and also determines bus access priority during the arbitration process, where the priority is defined to be highest for the smallest binary number of the identifier. This is due to the greater number of leading dominant bits during arbitration.

RTR (Remote Transmission Request)

- High — Remote Frame for requesting data is transmitted by the 82C200.
- Low — Data Frame for offering of data is transmitted by the 82C200.

DLC0–DLC2 (Data Length Code)

The number of data bytes (Data Byte Count) in the Data Field of a message is coded by the Data Length Code. At the transmission of a Remote Frame, Data Byte Count of Data Length Code shall not be regarded (due to the RTR bit), forcing the number of transmitted data bytes to be 0.

The range of the Data Byte Count is 0 to 8 bytes and coded as follows:

Data Byte Count = 8DLC.3 + 4DLC.2 + 2DLC.1 + DLC.0

For reasons of compatibility, no other Data Byte Counts than 0, 1, 2, ..., 8 should be used.

Data Field
The number of relevant data bytes is determined by the Data Length Code. The bit to be transmitted first is the most significant bit of data byte 1 at the address 12.

Table 1. Programmable Output Functions

OCMODE0	OCMODE1	FUNCTION
0	1	Normal Mode; TX0, TX1: bit sequence (TXD)
1	1	Normal Mode; TX0: bit sequence, TX1: bus clock (TXCLK)
0	0	Biphase Mode
1	0	Test Mode; TX0: bit sequence, TX1: Compout

Table 2. TX0, TX1 Output Driver Configuration

MODE	OCTPx	OCTNx	OCPOLx	TXD	TPx[1]	TNx[2]	TXi Output Level[3]
Float	0	0	0	0	Off	Off	Float
	0	0	0	1	Off	Off	Float
	0	0	1	0	Off	Off	Float
	0	0	1	1	Off	Off	Float
Pull Down	0	1	0	0	Off	On	Low
	0	1	0	1	Off	Off	Float
	0	1	1	0	Off	Off	Float
	0	1	1	1	Off	On	Low
Pull Up	1	0	0	0	Off	Off	Float
	1	0	0	1	On	Off	High
	1	0	1	0	On	Off	High
	1	0	1	1	Off	Off	Float
Push Pull	1	1	0	0	Off	On	Low
	1	1	0	1	On	Off	High
	1	1	1	0	On	Off	High
	1	1	1	1	Off	On	Low

NOTES:
1. TPx: On-chip output transistor x, connected to V_{DD}; x = 0, 1
2. TNx: On-chip output transistor x, connected to V_{SS}; x = 0, 1
3. TXx: Serial output on pin TX0 or TX1. The output level on the CAN bus is dominant when TXD = 0 and recessive when TXD = 1.

Receive Buffer Layout

The layout of the Receive Buffer as well as the layout of the individual bytes corresponds to the definitions given for the Transmit Buffer layout, except that the addresses start at 20 instead of 10 (see Figure 1).

Clock Divider Register (CDR)

MSB 7	6	5	4	3	2	1	LSB 0
-	-	-	-	-	CD2	CD1	CD0

The Clock Divider Register controls the Clockout frequency for the microcontroller (see the Block Diagram, above). It is read/writeable by the microcontroller. Its value is preset by external reset to divide by 12 when the Mode pin is tied to ground and to divide by 2 when tied to V_{DD}. Values from 0 to 7 may be written into this register and will result in the following Clockout frequencies:

CLOCK-OUT FREQUENCIES

CD.2	CD.1	CD.1	CLOCK OUT FREQUENCY
0	0	0	$f_{OSC}/2$
0	0	1	$f_{OSC}/4$
0	1	0	$f_{OSC}/6$
0	1	1	$f_{OSC}/8$
1	0	0	$f_{OSC}/10$
1	0	1	$f_{OSC}/12$
1	1	0	$f_{OSC}/14$
1	1	1	f_{OSC}

Bit Timing

A bit time is subdivided into a number of BTL cycles (identical to the system clock cycles. This number results from the addition of the programmable segments SJW1, TSEG1, TSEG2, and SJW2 plus the general segment $SYNCSEG_1$ (see Figure 2).

$SYNCSEG_1$

The incoming edge of a bit is expected during this state; this state corresponds to one BTL cycle.

SJW1, SJW2

Both segments determine the maximum synchronization jump width for resynchronization, programmable from 1 to 4 BTL cycles. The width of SJW1 is increased but maximum doubled to lengthen the bit time during resynchronization. The width of SJW2 is reduced or canceled to shorten the bit time during resynchronization.

TSEG1

This determines the sampling point based on the number of BTL cycles programmed by TSEG1 (4 bits). The sampling point is located at the end of TSEG1 (SAM = 0). TSEG1 is used to compensate delay times on the bus and to have some reserved time to tolerate one or more missynchronization pulses caused by spikes on the bus line. TSEG1 is programmable from 1 to 16 BTL cycles.

TSEG2

This defines the time between the sampling point and the end of the bit time; it is programmable from 1 to 8 BTL cycles (3 bits). This segment is necessary to tolerate one or more missynchronizations caused by spikes on the bus line. Also TSEG2 is necessary to guarantee sufficient time for BSP to generate an internal transmit signal dependent on the sampled bus level. This transmit point is determined internally in such a way that with zero delay the internally generated transmit signal will appear within the INSYNC state at the TX output pins (see the Block Diagram). For example, no transmit signal would be generated if an arbitration was lost. This guarantees that the transmit logic immediately stops to continue a message transfer and immediately will pass into the receive mode.

Synchronization

Synchronization is performed by a state machine which compares the incoming edge with its actual bit timing and adapts the bit timing by hard- or resynchronization.

Hard Synchronization

This type of synchronization occurs only at the beginning of a message. The 82C200 synchronizes to the first incoming recessive to dominant edge of a message (being the leading edge of a message's Start of Frame bit.

Resynchronization

Resynchronization occurs during the message bit stream to compensate differences in the oscillator frequencies of individual 82C200s as well as changes introduced by switching from one to another transmitter (e.g., during arbitration). SJW1 and SJW2 define the maximum number of BTL cycles a bit time may be lengthened or shortened by one resynchronization:

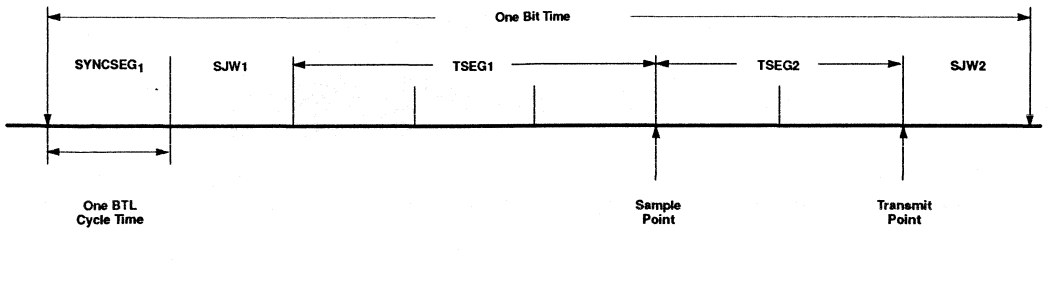


Figure 2. Bit Time Segments

As a result of resynchronization, t_{TSEG1} may be increased or t_{TSEG2} may be decreased to a maximum of t_{SJW}:

$t_{TSEG1} \leq t_{SCL}$ ((TSEG1 + 1) + (SJW + 1))
$t_{TSEG1} \geq t_{SCL}$ ((TSEG1 + 1) – (SJW + 1))

Resynchronization can be performed on both edges—recessive to dominant and dominant to recessive—or on the recessive to dominant edge only, depending on the programmed bit of Speed Mode.

Resynchronization is done once during a bit time.

The phase error (e) of an edge is given by the position of the edge relative to SYNCSEG, measured in system clock cycles (t_{SCL}). The value of the phase error is defined as:

e = 0, if the edge occurs within SYNCSEG
e > 0, if the edge occurs within TSEG1
e < 0, if the edge occurs within TSEG2

The effect of resynchronization is:

- The same as that of a hard synchronization, if the magnitude of the phase error (e) is less than or equal to the programmed value of t_{SJW}.
- To increase a bit period by the amount of t_{SJW}, if the phase error is positive and the magnitude of the phase error is larger than t_{SJW}.
- To decrease a bit period by the amount of t_{SJW}, if the phase error is negative and the magnitude of the phase error is larger than t_{SJW}.

Synchronization Rules
The synchronization rules are as follows:

- Only one synchronization within one bit time is used.
- An edge is used for synchronization only if the value detected at the previous sample point differs from the bus value immediately after the edge.
- Hard synchronization is performed whenever there is a recessive-to-dominant edge during Bus-Idle.
- All other edges (recessive-to-dominant and optionally dominant-to-recessive edges if the Synch bit is set High) which are candidates for resynchronization will be used, with the following exception:

  A transmitting 82C200 will not perform a resynchronization as a result of a recessive-to-dominant edge with positive phase error, if only these edges are used for resynchronization. This ensures that the delay times of the output driver and input comparator do not cause a permanent increase in the bit time.

Note: The 82C200, transmitting a dominant bit, does not lengthen (synchronize) the bit time if Speed Mode is set to Low (i.e., resynchronize on a recessive to dominant edge only).

Frame Types

Data Frame
A Data Frame carries data from a transmitting 82C200 to one or more receiving 82C200s. A data Frame (see Figure 3) is composed of seven different bit fields:

- Start of Frame
- Arbitration Field
- Control Field
- Data Field
- CRC Field
- ACK Field
- End of Frame

The Data Field can be of length zero.

Start of Frame
Start of Frame signals the start of a data frame or a remote frame. It consists of a single dominant bit used for hard synchronization of receiving 82C200s.

Arbitration Field
This field consists of the message Identifier and the RTR bit. In the event of simultaneous message transmission start of two or more 82C200s, the bus access conflict is solved by a bitwise arbitration mechanism active during the transmission of the Arbitration Field.

Identifier
This 11-bit field is dedicated to provide information about each individual message as well as a specific bus access priority for each message.

An identifier does not define a physical 82C200 address to determine the particular receiver of a frame. Within a CAN-based communication network, there exists no discrimination between a point-to-point, multicast, or broadcast communication.

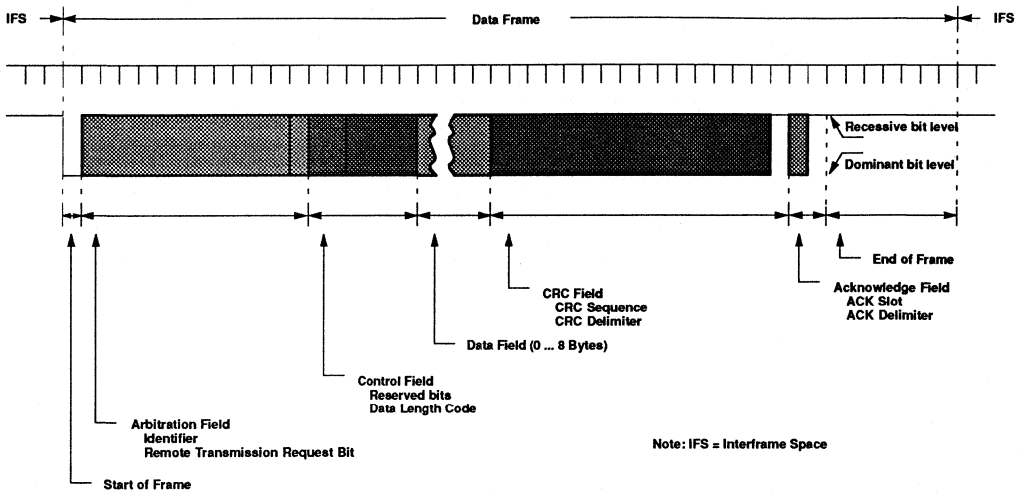


Figure 3. Data Frame

RTR Bit
An 82C200, acting as a receiver for certain information, may initiate the transmission of the respective data by transmission of a Remote Frame to the network, addressing the data source via the identifier and setting the RTR Bit to remote (recessive bus level). If, in coincidence, the data source transmits a Data Frame, containing the requested data, it uses the same identifier, and bus access conflict is prevented by the RTR Bit only, set to data (dominant bus level) in a Data Frame.

Control Field
This field consists of six bits. It includes two reserved bits (for future expansions of CAN), transmitted with a dominant bus level, and is followed by the Data Length Code (4 bits). A 0 bit (1 bit) in the data length code is transmitted as a dominant (recessive) bus level, respectively.

Data Field
The data, stored within the Data Field of the transmit buffer, are transmitted according to the data length code. On the other hand, data of a received Data Frame will be stored in the Data Field within a receive buffer. Data is stored bytewise both for transmission by the microcontroller and on reception by the 82C200. The most significant bit of the first data byte (lowest address) is transmitted/received first.

CRC Field
This field contains the CRC Sequence (15 bits) followed by the CRC Delimiter (one recessive bit). The Cyclic Redundancy Code (CRC) encloses the (destuffed bit stream of) Start of Frame, Arbitration Field, Control Field, Data Field, and CRC sequence. The most significant bit of the CRC Sequence is transmitted/received first. This frame check sequence, implemented in the 82C200, is derived from a cyclic redundancy code best suited for frames with a total bit count of less than 127 bits (BCH Code). With Start of Frame (dominant bit) included in the code word, any rotation of the code word can be detected by the absence of the CRC Delimiter (recessive bit).

Acknowledge Field
The ACK Field consists of two bits, the ACK Slot and the ACK Delimiter, which are sent with a recessive level by the transmitter. Any receiving 82C200 bus controller acknowledges to the transmitting one a match of the completely and correctly received CRC sequence with the ACK Slot by superscribing the transmitter's recessive bit with a dominant bit. Thereby, a transmitter, still monitoring the bus level, at this point recognizes that at least one receiver within the network has received a complete and correct message (i.e., no error was found). The ACK Delimiter (recessive bit) is the second bit of the ACK Field. As a result, the ACK Slot is surrounded by two recessive bits—the CRC Delimiter and the ACK Delimiter.

End of Frame
Each data or remote frame is delimited by the End of Frame bit sequence, which consists of seven recessive bits (exceeds the bit stuff width by two bits). By this mechanism a receiver detects the end of a frame independently of a previous transmission error because the receiver expects all bits up to the end of the CRC field to be coded by the method of bit stuffing. The bit stuff logic during the End of Frame is turned off.

Remote Frame
An 82C200 acting as a Receiver for certain data can initiate the transmission of the respective data by sending a Remote Frame to the source 82C200.

A Remote Frame is composed of six different bit fields:

- Start of Frame
- Arbitration Field
- Control Field
- CRC Field
- ACK Field
- End of Frame

Contrary to data frames, the RTR bit of Remote Frames is recessive and the Data Byte Count is 0, independent of the Data Length Code held in the Descriptor of the message buffer (i.e., no Data Field is being transmitted).

Error Frame
The Error Frame contains a sequence of variable length holding only dominant bits. It is accomplished by superposition of Error Flags contributed from different 82C200s. This sequence is followed by an Error Delimiter.

Error Flag:
The Error Flag consists of six consecutive dominant bits.

An error-active 82C200 detecting an error condition signals this by transmission of an Error Flag. An Error Flag violates the law of bit stuffing applied to all fields from Start of Frame to CRC Delimiter, or destroys the fixed form of ACK Field or End of Frame. As a consequence, all other 82C200s detect an error condition too and on their part start transmission of an Error Flag. So the sequence of dominant bits, which actually can be monitored on the bus, results from a superposition of different Error Flags transmitted by individual 82C200s. The total length of this sequence varies between a minimum of 6 and a maximum of 12 bits.

An error-passive 82C200 detecting an error condition tries to signal this by transmission of six recessive bits instead of an Error Flag. If this 82C200 is not able to send these six recessive bits (since other 82C200s send dominant ones), the error-passive 82C200 waits for six consecutive bits of equal polarity.

Error Delimiter:
The Error Delimiter consists of eight recessive bits.

The Error Delimiter has the same format as End of Frame and Overload Delimiter and violates the rules of bit stuffing. After transmission of an Error Flag, each 82C200 monitors the bus line until it detects a transition from a dominant to recessive bit level. At this point of time, every 82C200 has finished sending its Error Flag, and all 82C200s start transmission of six more recessive bits (seven plus the recessive bit at transition time gives a total of eight recessive bits) in coincidence; i.e., all 82C200s in the network are synchronized again.

If a detected error is signalled during transmission of a Data Frame or Remote Frame, this procedure associates an Error Frame to the corresponding message and initiates a retransmission of the spoiled message.

If an 82C200 monitors any deviation of the Error Frame, it will start transmitting an Error Frame again, but if this occurs several times in a sequence the 82C200 may become error passive and does not block the network even in this case.

In order to terminate an Error Flag correctly, an error passive CAN as controller requires the bus to be Bus-Idle for at least three bit periods. Therefore, a CAN bus should not be 100% permanently loaded.

Overload Frame
The Overload Frame consists of two bit fields, the Overload Flag and the Overload Delimiter.

According to the CAN protocol, there are two conditions both leading to the transmission of an Overload Flag:

1. Internal conditions of the receiver circuitry, which require a delay time before receiving the next frame (receiver not ready).
2. Detection of a dominant bit during Intermission.

The transmission of an Overload Frame due to condition 1 is only allowed to be started at the first bit time of an expected Intermission, whereas Overload Frames due to condition 2 start one bit time after detecting the dominant bit during Intermission.

The 82C200 never will initiate transmission of an Overload Frame (condition 1) but only will react on a transmitted Overload Frame (condition 2). No more than two overload frames are generated to delay a Data Frame or a Remote Frame.

Overload Flag:
The Overload Flag consists of six dominant bits; the overall form corresponds to that of an Error Flag.

The Overload Flag's form destroys the fixed form Intermission field. As a consequence, all other 82C200s also detect an overload condition and on their part start transmission of an Overload Flag, too.

Overload Delimiter:
The Overload Delimiter consists of eight recessive bits and is of the same form as End of Frame and Error Delimiter.

After transmission of an Overload Flag, each 82C200 monitors the bus until a transition from dominant to a recessive bit level occurs. At this point of time every 82C200 has finished sending its Overload Flag and all 82C200s start transmission of seven more recessive bits in coincidence.

Interframe Space
Data Frames and Remote Frames are separated from preceding frames of whatever type they are (Data, Remote, Error, or Overload Frame) by an Interframe Space, consisting of an Intermission field followed by a possible Bus Idle. In contrast, Overload Frames and Error Frames are not preceded by an Interframe Space.

Intermission:
The Intermission bit field consists of three recessive bits. During intermission time, no 82C200 bus controller will start a transmission of a frame. Intermission is required to have a fixed time period for the 82C200 to execute some internal processes prior to the next receive or transmit task. For example, control bits are being updated only if no error condition up to the last bit of the just transfered message has occurred, i.e., after message transfer time.

Bus Idle:
The Bus Idle time may be of arbitrary length (minimum of 0 bit). The bus is recognized to be free and any 82C200 having something to transmit may access the bus.

The detection of a dominant bit level during Bus Idle on the bus is interpreted as Start of Frame.

Bus Access
82C200s are only allowed to start transmission during Bus Idle state. All 82C200s have to synchronize on the leading edge of Start of Frame (hard synchronization).

Arbitration
If two or more 82C200 bus controllers concurrently start transmission, the bus access conflict is solved by a bitwise arbitration mechanism during transmission of the Arbitration Field.

During arbitration every transmitting 82C200 82C200 compares its transmitted bit level with the monitored bus level. Any 82C200 which sends a recessive bit and monitors a dominant bus level immediately becomes receiver of the current higher prioritized message on the bus without destroying any information on the bus.

Each message is attached with a unique Identifier and an RTR bit characterizing the type of data within the message. Moreover, the Identifier together with the RTR bit implicitly defines the message's bus access priority. During arbitration, the most significant bit of the Identifier is transmitted first and the RTR bit at last, giving a message the higher the priority the smaller the binary value of Identifier and RTR bit is. Since a Data Frame's RTR bit is dominant, it has a higher priority than the matching Remote Frame (same Identifiers) owing a recessive RTR bit.

For any Data Frame, there is a unique transmitter.

For reasons of compatibility to other CAN bus controllers, never use the Identifier bit pattern ID = '1111111xxxx'bit (x being bits of arbitrary level). Thus the total number of available different Identifiers amounts to 2032 ($2^{11} - 2^4$).

Coding/Decoding
The following bit fields are coded by the method of bit stuffing:

- Start of Frame
- Arbitration Field
- Control Field
- Data Field
- CRC Sequence

Whenever a transmitting 82C200 detects five consecutive bits of identical level to be transmitted, it automatically inserts a complementary stuff bit in the transmitted bit stream.

Whenever a receiving 82C200 has monitored five identical consecutive bit levels in the received bit streams of the above described bit fields, it automatically deletes the next received (stuff) bit. The level of the deleted stuff bit has to be the complement of the previous bits; otherwise a stuff-bit error will be detected and signalled.

The remaining bit fields of frames are of fixed form and are not coded or decoded by the method of bit stuffing.

The bit stream in a message is coded according to the Non-Return-to-Zero (NRZ) method. This means that during a bit time, the bit level is held constant, either recessive or dominant.

Error Signalling
An error-active 82C200 detecting an error condition signals this by transmitting a recessive Error Flag.

Whenever a Bit Error, Stuff Error, Form Error, or an Acknowledgment Error is detected, transmission of an Error Flag is started at the next bit. Whenever a CRC Error is detected, transmission of an Error Flag starts at the bit following the Acknowledge Delimiter, unless an Error Flag for another error condition has already started. An Error Flag violates the bit-stuffing law or corrupts the fixed form bit fields. A violation of the bit-stuffing law affects any 82C200 which detects the error condition. These devices will also transmit an Error Flag.

A recessive Error Flag is not able to break a running message at different 82C200s, but may be destroyed by other 82C200s. After having detected an error condition, an error-passive 82C200 will wait for six identical consecutive bits and, when monitoring them, interpret them as an Error Flag.

After the transmission of an Error Flag, each 82C200 monitors the bus line until it detects a transition from a dominant to a recessive bit level. At this point of time, every 82C200 has finished sending its Error Flag, and all 82C200s start transmission of six more recessive bits of the Error Delimiter in coincidence.

The message format of a Data Frame or Remote Frame is defined in such a way that all detectable errors can be signalled within the message transmission time, and thus it is very simple to associate an Error Frame to the corresponding message and to initiate retransmission of the spoiled message.

If a 82C200 monitors any deviation of the fixed form of the Error Frame, it starts transmission of a further Error Frame.

Overload Signalling
By sending of one or more Overload Frame(s) any 82C200 is able to delay the transmission of the next Data Frame or Remote Frame. The transmission of an Overload Frame has to start during the first bit of an expected Intermission. On the other hand, Overload Frames which are reactions on a dominant bit during an expected Intermission start one bit after this event.

Though the formats of Overload Frame and Error Frame are identical, they are treated differently. Thereby, sending an Overload Frame during Intermission does not initiate retransmission of any previous Data Frame or Remote Frame.

If the 82C200 sending an Overload Frame monitors any deviation of its fixed form, it starts transmission of an Error Frame.

Bit Error
A transmitting 82C200 bus controller monitors the bus by a bit-by-bit basis. If the bit level monitored is different from the transmitted one, a Bit Error is signaled.

Exceptions:

- During Arbitration Field, a recessive bit can be overwritten by a dominant bit. In this case, the 82C200 interprets this as a loss of arbitration.
- During the ACK Slot, only the receiving 82C200s are able to recognize a Bit Error.

Stuff Error
The following bit fields are coded by the method of bit stuffing:

- Start of Frame
- Arbitration Field
- Control Field
- Data Field
- CRC Sequence

There are two possibilities to generate a Stuff Error:

- The disturbance generates more than the allowed five identical consecutive bits. These errors are detected by all 82C200s.
- A disturbance falsifies one or more of the five bits preceding the stuff bit. This error situation is not recognized as a Stuff Error by the receivers. Therefore, other error detection mechanisms like CRC Check and format violation at the receiver's site or Bit Error detection by the transmitting 82C200 may detect this error condition.

CRC Error
To ensure the validity of a transmitted message, all receivers perform a CRC Check. Therefore, in addition to the (destuffed) information digits (Start of Frame up to Data Field), any code word (information plus control digits) includes some control digits (CRC Sequence, generated by the transmitting 82C200 of the respective message) used for error detection.

The code used for the 82C200 bus controller is a (shortened) BCH code, extended by a parity check and has the following attributes:

- 127 bits as maximum length of the code word
- 112 bits as maximum number of information digits (a maximum of 83 bits are used by the 82C200)
- Length of the CRC Sequence amounts to 15 bits
- Hamming distance d = 6

As a result, d-1 random errors are detectable (some exceptions exist).

The CRC Sequence is determined by the request that the code word, if interpreted as polynomial with coefficients 0 or 1 is divisible (modulo 2) by the generator polynomial:

$$f(x) = (x^{14} + x^9 + x^8 + x^6 + x^5 + x^4 + x^2 + x + 1)(x + 1)$$
$$= 1100010110011001 \text{ bin}$$

Burst errors are detected up to a length of 15 (degree of f(x)). Multiple errors (number of disturbed bits at least d = 6) are not detected with a residual error probability of $215 = 3*10^{-5}$ (by CRC Check only).

Form Error
Form Errors result from violation of the fixed form of the following bit fields:

- End of Frame
- Intermission
- ACK Delimiter
- CRC Delimiter

During the transmission of these bit fields, an error condition is recognized if a dominant bit level instead of a recessive one is detected.

Error Detection by an Error Flag of Another 82C200
If there is not sent an acknowledge by any receiver, the transmitter recognizes that its transmission has failed.

The detection of an error is signalled by transmitting an Error Flag, if the error detecting 82C200 is error active. An Error Flag causes a Stuff Error, a Bit Error, or a Form Error at all other 82C200s.

Error Detection Capabilities
Errors which occur at all 82C200s (global errors) are detected by 100%.

For local errors, i.e., for errors occurring at some 82C200s only, the shortened BCH code, extended by a parity check, has the following error detection capabilities:

- Up to five single Bit Errors are detected by 100%, even if they are distributed randomly within the code word.
- All single Bit Errors are detected if their total number (within the code word) is odd.
- The residual error probability of the CRC check amounts to $3*10^{-5}$. As an error may be detected not only by CRC check but also by other detection mechanisms described above, the residual error probability is significantly less than $3*10^{-5}$.

Error Confinement

Bus Off
An 82C200 bus controller which is transmitter of too many unsuccessful transmissions relative to the successful ones will pass into the stat Bus Off. This means that it neither sends nor receives messages, until it has got a Reset from the microcontroller (Reset Request = 'absent') and a predefined time of standstill has passed.

ACK
An 82C200 which has correctly received a valid message reports this to the transmitter by sending a dominant bit level on the bus during the ACK Slot bit (i.e., sends 'ACK'), independent of accepting the message or not.

Error Active
An Error Active 82C200 is in its usual state, able to receive and/or transmit normally, and is able to send a dominant Error Flag.

Error Passive
An Error Passive 82C200 may send or receive messages normally, but is not able in the case of a detected error condition to signal this by transmitting a dominant but only recessive Error Flag. Thus an Error Passive 82C200 may not hang up all bus activities due to a failure in its transmit logic.

Suspend Transmission
After an Error Passive 82C200 having transmitted a message, it sends seven recessive bits following Intermission, before starting to transmit a further message. If meanwhile another transmission starts, the Error Passive 82C200 will become receiver of this message.

Start Up
A 82C200 which was switched off or in the state Off Bus has to run through a Start Up routine in order to:

- Synchronize with already available 82C200s before starting to transmit. Synchronizing is achieved when 11 recessive bits, equivalent to Ack Delimiter + End of Frame + Intermission, have been detected (Bus Free).
- Wait for other 82C200s without passing into Off Bus (because of a missing acknowledge), if there is no other 82C200 available yet.

Aims of Error Confinement

Distinction of short and long lasting disturbances:

The microcontroller should be informed when there are long lasting disturbances and when bus activities are undisturbed again. In cases of long lasting disturbances, an 82C200 bus controller enters the Off Bus state and the microcontroller has to work with default values.

Minor disturbances of bus activities should not affect an 82C200 bus controller, in particular not drive it into Off Bus or let it inform the microcontroller.

Detection and localization of disturbances and defects of the hardware:

Disturbances and defects of the hardware should affect bus activities as little as possible. When the hardware of a 82C200 is defective, this 82C200 should stop sending (especially dominant bits) as soon as possible (Off Bus state) or stop affecting bus activities in another way (Error Passive state), whereas the other 82C200s should not pass into Off Bus state.

After the end of long lasting disturbances, a restart must be possible.

Strategies to Gain Error Confinement
Any 82C200 bus controller is supplied with a Transmit Error Count and a Receive Error Count, the first of which registers errors during the transmission and the other one registers errors during the reception of messages.

If messages are sent or received correctly, the counts are decreased. In case of errors, the counts are increased more than they are decreased in absence of errors. This leads to an increase of the counts in the elapse of time, even if there are fewer disturbed messages than undisturbed ones. The levels of the Error Counts reflect the relative frequency of disturbances. The ratio of increase/decrease depends on the acceptable ratio of invalid/valid messages on the bus.

There is a microcontroller Warning Limit (96 error points) the exceeding of which by at least one of the counts means a non-negligible accumulation of error conditions and leads to an information to the microcontroller.

The exceeding of 127 error points by at least one of the counts leads to the Error Passive mode. Otherwise an 82C200 is in its Error Active mode.

The exceeding of 255 error points by the Transmit Error Count leads to the Off Bus state.

The rules for error confinement are chosen in such a way that a defective 82C200 detects its malfunction and resulting errors in any relevant case and furthermore reacts on these errors by increasing one of its Error Counts, whereas the intact 82C200s do not or do not always react on these errors. Thus it is possible to identify the defective 82C200 due to the statistically quicker increase of its Error Counts.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Supply voltage range	4.5 to + 5.5	V
Power dissipation (based on package heat transfer limitations, not device power consumption)	1	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_A = –40°C to +125°C, V_{CC} = 5V ±10%, V_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{IL1}	Input low voltage, all except XTAL1, RX0, RX1		–0.5	0.8	V
V_{IL2}	Input low voltage, XTAL1			$0.2V_{CC}$	V
V_{IH1}	Input high voltage, all except XTAL1, $\overline{RST}$, RX0, RX1		2.8	V_{CC}+0.5	V
V_{IH2}	Input high voltage, XTAL1		$0.7V_{CC}$		V
V_{IH3}	Input high voltage, $\overline{RST}$		3.3	V_{CC}+0.5	V
V_{OL}	Output low voltage, all outputs	I_{OL} = 1.6mA		0.45	V
V_{OH1}	Output high voltage, all outputs except TX0, TX1, INTN, and CLKOUT	I_{OH} = 80µA	2.4		V
V_{OH2}	Output high voltage, CLKOUT	I_{OH} = 80µA	$0.8V_{CC}$		V
I_{LK}	Input leakage current, all except XTAL1, RX0, RX1	$V_{SS} < V_{IN} < V_{CC}$		±10	µA
I_{CC}	Supply current	See note 1		15	mA
I_{SM}[1]	Sleep mode supply current	Oscillator stopped See note 2		40	µA
Input Comparator T_A = –40°C to +125°C, V_{CC} = 5V ±10%, V_{SS} = 0V					
V_I	Input voltage		–0.5	V_{CC}+0.5	V
V_{ICOM}	Comon mode range		1.5	V_{CC}–1.5	V
I_I	Input current			±1	µA
V_{OFFSET}	Input offset voltage	See note 3		±10	mV
V_{HYST}	Hysteresis voltage	See note 3	8	18	mV
Output Driver T_A = –40°C to +125°C, V_{CC} = 5V ±10%, V_{SS} = 0V					
$-I_O$	Source current (each TX0, TX1)	$V_{OUT} = V_{CC} - 1.0V$		–10	mA
I_O	Sink current (each TX0, TX1)	V_{OUT} = 1.0V		10	mA

NOTES:
1. AD0 – AD7 = ALE = $\overline{RD}$ = $\overline{WR}$ = $\overline{INT}$ = $\overline{RST}$ = $\overline{CS}$ = V_{CC}; RX0 = 2.6V, RX1 = 2.4V; XTAL1 = 0.5V/V_{DD} – 0.5V, all outputs unloaded.
2. AD0 – AD7 = ALE = $\overline{RD}$ = $\overline{WR}$ = $\overline{INT}$ = $\overline{RST}$ = $\overline{CS}$ = Mode = RX0 = V_{CC}; RX1 = V_{CC} –V_V; XTAL1 = V_{SS}; all outputs unloaded.
3. Hysteresis and offset voltages are not tested.

AC ELECTRICAL CHARACTERISTICS

T_A = −40°C to +125°C, V_{CC} = 5V ±10%, V_{SS} = 0V, load capacitance on output pins: 100pF

SYMBOL	PARAMETER	LIMITS		UNIT
		MIN	MAX	
f_{XTAL}	Oscillator frequency	3	16	MHz
t_{SU1}	Address setup to ALE/AS low	20		ns
t_{HD1}	Address hold time	28		ns
t_{PW1}	ALE/AS pulse width	85		ns
t_{VD1}	$\overline{RD}$ low to valid data output		148	ns
t_{VD2}	E high to valid data out		148	ns
t_{DF1}	Data float after $\overline{RD}$ high	0	55	ns
t_{DF2}	Data float after E low	0	55	ns
t_{SU2}	Input data set up to $\overline{WR}$ high	30		ns
t_{HD2}	Input data hold after $\overline{WR}$ high	10		ns
t_{SU3}	Input data set up to E low	30		ns
t_{HD3}	Input data hold after E low	10		ns
t_{LL1}	ALE low to $\overline{WR}$ low	50		ns
t_{LL2}	ALE low to $\overline{RD}$ low	50		ns
t_{LH1}	AS low to E high	50		ns
t_{SU4}	Set-up time of RD/$\overline{WR}$ to E high	50		ns
t_{PW2}	$\overline{WR}$ pulse width	210		ns
t_{PW3}	$\overline{RD}$ pulse width	210		ns
t_{PW4}	E pulse width	210		ns
t_{LL3}	$\overline{CS}$ low to $\overline{WR}$ low	0		ns
t_{LL4}	$\overline{CS}$ low to RD low	0		ns
t_{LH2}	$\overline{CS}$ low to E high	0		ns
t_{ed}	Input comparator to output driver		70	ns

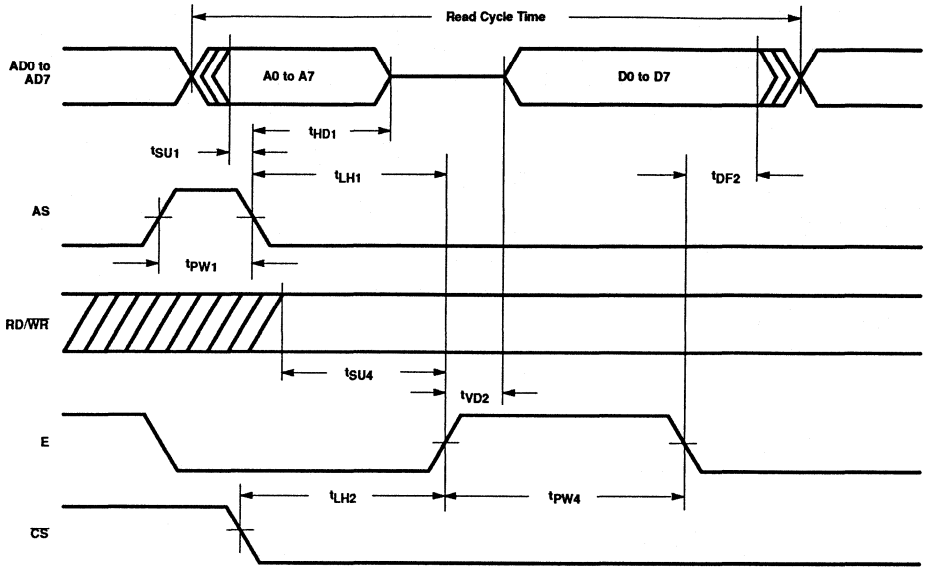


Figure 4.

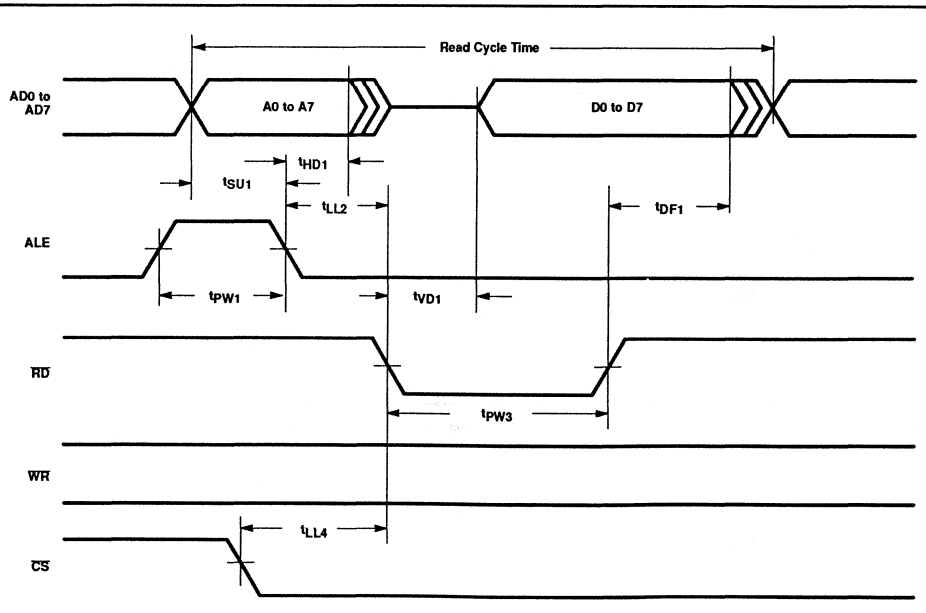


Figure 5.

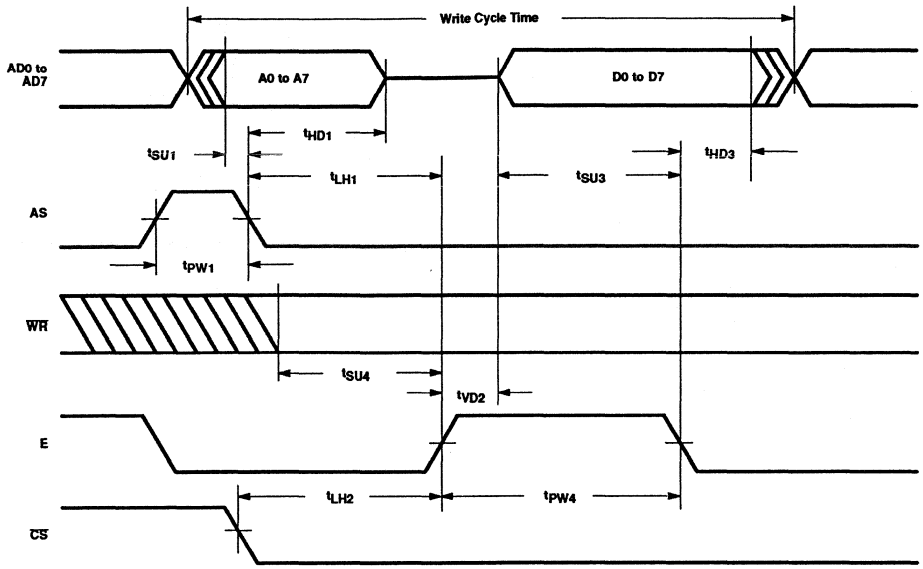


Figure 6.

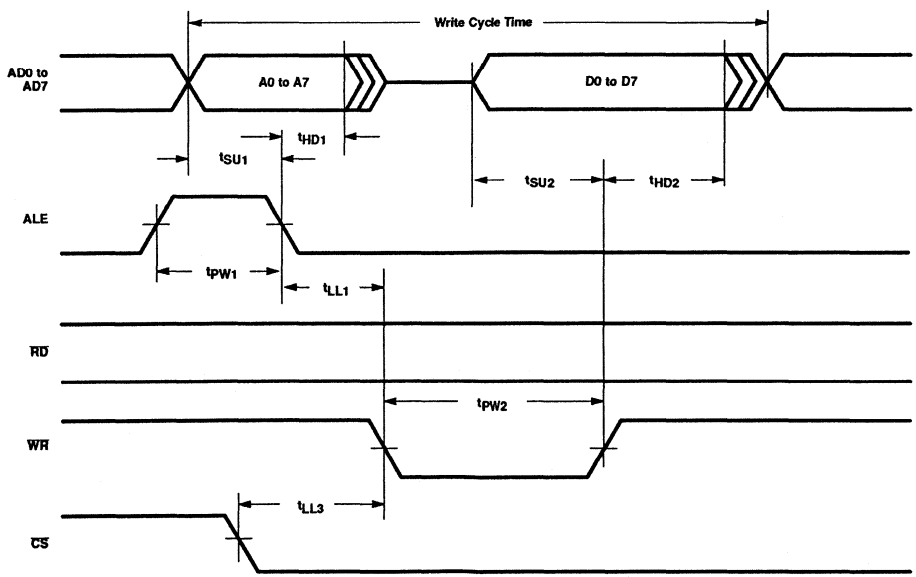


Figure 7.

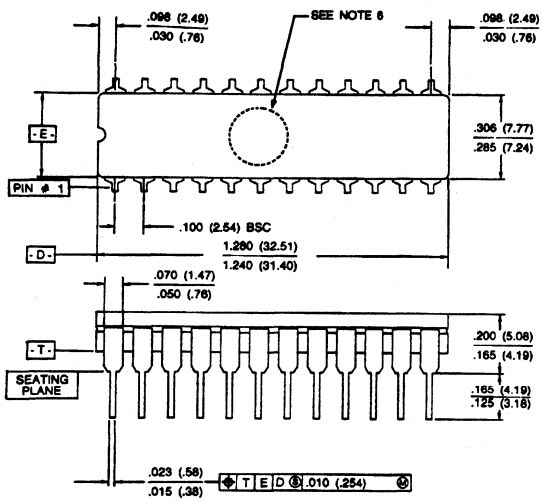
Section 7
Package Outlines

CONTENTS

24-Pin Hermetic CERDIP (300 mil wide) 656
24-Pin Plastic Dual In-Line (300 mil wide) 657
28-Pin Ceramic Dual In-Line 658
28-Pin Plastic Dual In-Line (600 mil wide) 658
28-Pin Plastic Leaded Chip Carrier 659
40-Pin Ceramic Dual In-Line 659
40-Pin Plastic Dual In-Line 660
44-Pin Plastic Leaded Chip Carrier 661
44-Pin Square Ceramic Leadless Chip Carrier 661
44-Pin Square Ceramic Leaded Chip Carrier 662
64-Pin Plastic Dual In-Line 663
68-Pin Plastic Leaded Chip Carrier 663
68-Pin Square Ceramic Leadless Chip Carrier 664
84-Pin Square Plastic Leaded Chip Carrier 665
84-Pin Square Ceramic Leadless Chip Carrier 665
40-Lead Mini-Pack; Plastic (VSO40; SOT158A) 666
44-Pin Quad Flat-Pack 667
80-Lead Quad Flat-Pack; Plastic (SOT219) 668

Section 7 – Package outlines

24-PIN HERMETIC CERDIP (300 mil wide)



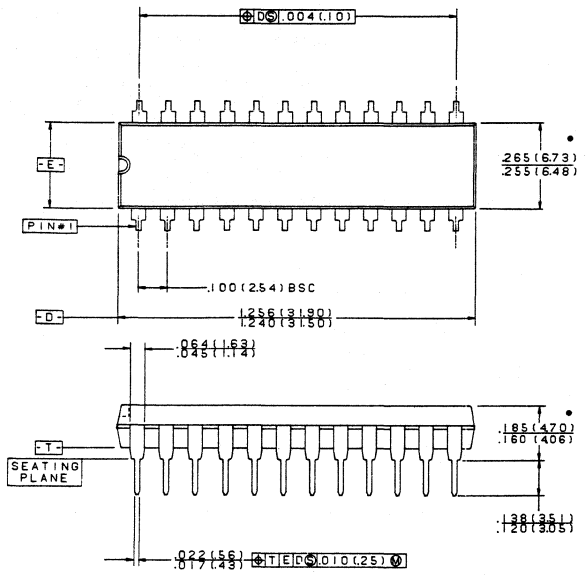
NOTES:
1. Controlling dimension: inches. Millimeters are shown in parentheses.
2. Dimensions and tolerancing per ANSI Y14.5M - 1982.
3. "T", "D", and "E" are reference datums on the body and include allowance for glass overrun and meniscus on the seal line, and lid to base mismatch.
4. These dimensions measured with the leads constrained to be perpendicular to plane T.
5. Pin numbers start with pin #1 and continue counterclockwise to pin #24 when viewed from the top.
6. Denotes window location for EPROM products (see FA package).

853-0586 84000

FN1

Section 7 – Package outlines

24-PIN PLASTIC DUAL IN-LINE (300 mil wide)



NOTES

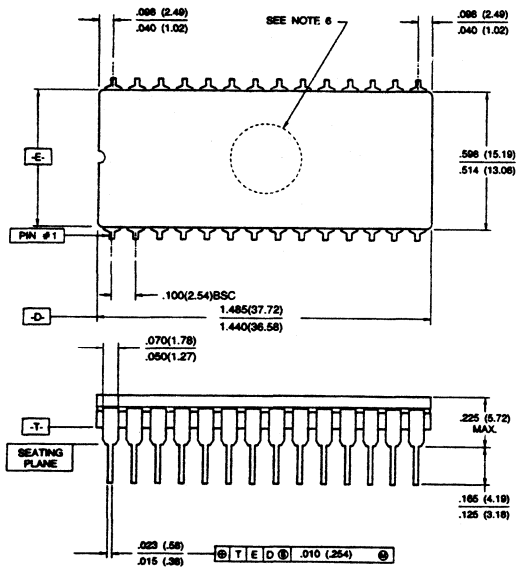
1. CONTROLLING DIMENSION: INCHES METRIC ARE SHOWN IN PARENTHESES.
2. PACKAGE DIMENSIONS CONFORM TO JEDEC SPECIFICATION MS-001-AF FOR STANDARD DUAL IN-LINE (DIP) PACKAGE .300 INCH ROW SPACING (PLASTIC) 24 LEADS (ISSUE B, 7/85)
3. DIMENSION AND TOLERANCING PER ANSI Y14.5M-1982.
4. "T", "D" AND "E" ARE REFERENCE DATEMS ON THE MOLDED BODY AND DO NOT INCLUDE MOLD FLASH OR PROTRUSIONS. MOLD FLASH OR PROTRUSIONS SHALL NOT EXCEED .010 INCH (.25MM) ON ANY SIDE.
5. THESE DIMENSIONS MEASURED WITH THE LEADS CONSTRAINED TO BE PERPENDICULAR TO PLANE T.
6. PIN NUMBERS START WITH PIN#1 AND CONTINUE COUNTERCLOCKWISE TO PIN#24 WHEN VIEWED FROM THE TOP.

853-0410 98727

NN1

Section 7 – Package outlines

28-PIN CERAMIC DUAL IN-LINE

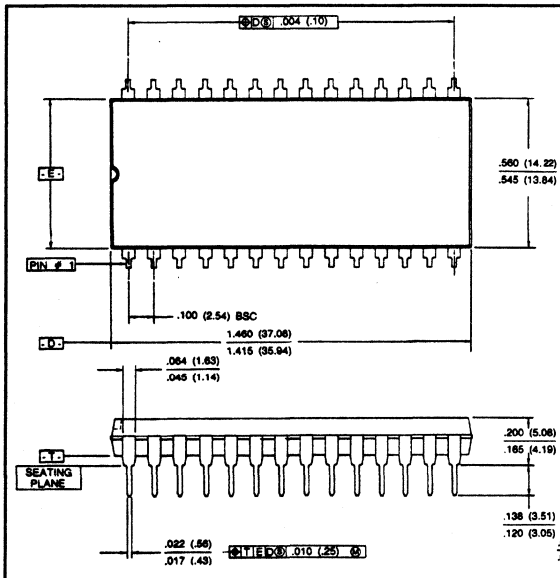


NOTES:
1. Controlling dimension: inches. Millimeters are shown in parentheses.
2. Dimensions and tolerancing per ANSI Y14.5M - 1982.
3. "T", "D", and "E" are reference datums on the body and include allowance for glass overrun and meniscus on the seal line, and lid to base mismatch.
4. These dimensions measured with the leads constrained to be perpendicular to plane T.
5. Pin numbers start with pin #1 and continue counterclockwise to pin #28 when viewed from the top.
6. Denotes window location for EPROM products.

FQ3 853-0589 84000

FQ

28-PIN PLASTIC DUAL IN-LINE (600 mil wide)



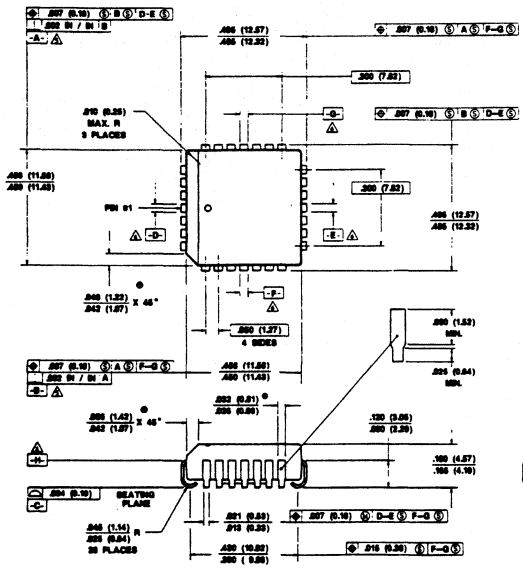
NOTES:
1. Controlling dimension: inches. Metric are shown in parentheses.
2. Package dimensions conform to JEDEC specification MS-011-AB for standard dual in-line (DIP) package .600 inch row spacing (PLASTIC) 28 leads (issue B. 7/85)
3. Dimensions and tolerancing per ANSI Y14. 5M-1982.
4. "T", "D" and "E" are reference datums on the molded body and do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010 inch (.25mm) on any side.
5. These dimensions measured with the leads constrained to be perpendicular to plane T.
6. Pin numbers start with pin #1 and continue counterclockwise to pin #28 when viewed from the top.

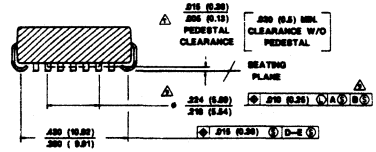
853-0413 84099

NQ3

Section 7 – Package outlines

28-PIN PLASTIC LEADED CHIP CARRIER





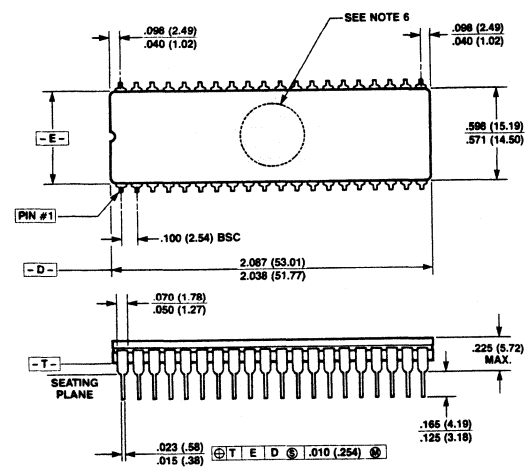
NOTES:

1. Package dimensions conform to JEDEC specification MO-047-AB for plastic leaded chip carrier 28 leads, .050 inch lead spacing, square. (issue A, 10/31/84).
2. Controlling dimensions: inches. Metric dimensions in mm are shown in parentheses.
3. Dimensioning and tolerancing per ANSI Y14.5M-1982.
4. Datum plane "H" located at the top of mold parting line and coincident with top of lead. Where lead exits plastic body.
5. Location to datum "A" and "B" to be determined at plane "H". These Datums do not include mold flash. Mold flash protrusion shall not exceed .006" (0.15 mm) on any side.
6. Datum "D-E" and "F-G" are determined where these center leads exit from the body at plane "H".
7. Pin numbers continue counterclockwise to pin #28 (top view).
8. Signetics order code for product packaged in a PLCC is the suffix "A" after the product number.
9. Applicable to packages with pedestal only.

AQ1 853-0401 93885

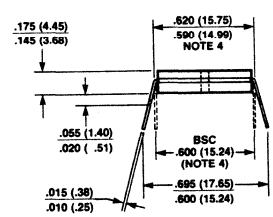
AQ1

40-PIN CERAMIC DUAL IN-LINE



NOTES

1. Controlling dimension: inches. Metric are shown in parenthesis.
2. Dimension and tolerancing per ANSI Y14, 5M-1982.
3. "T", "D", and "E" are reference datums on the body and include allowance for glass overrun and meniscus on the seal line, and lid to base mismatch.
4. These dimensions measured with the leads constrained to be perpendicular to plane T.
5. Pin numbers start with pin #1 and continue counterclockwise to pin #40 when viewed from the top.
6. Denotes window location for EPROM products.

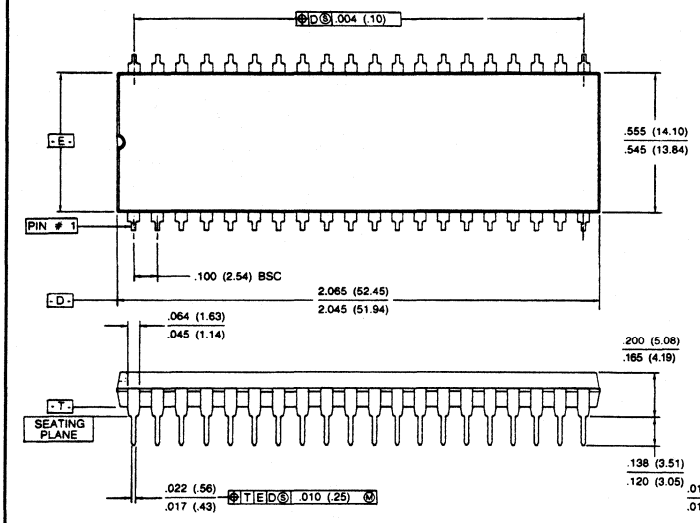


853-0590 95528

FW3

Section 7 – Package outlines

40-PIN PLASTIC DUAL IN-LINE



NOTES:

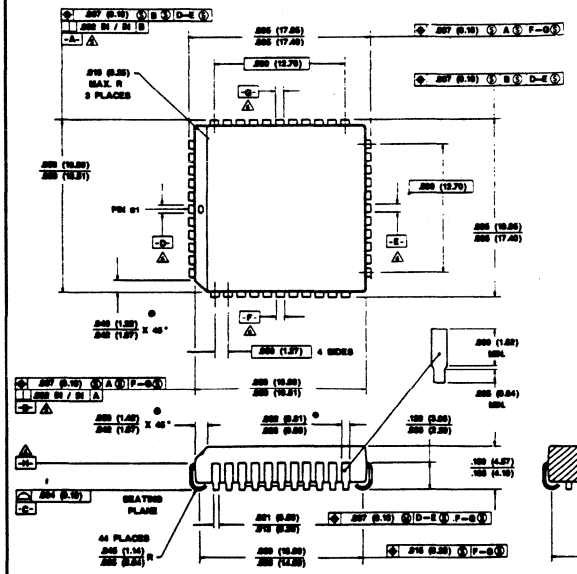
1. Controlling dimension: inches. Metric are shown in parentheses.
2. Package dimensions conform to JEDEC specification MS-011-AC for standard dual in-line (DIP) package .600 inch row spacing (PLASTIC) 40 leads (issue B. 7/85)
3. Dimensions and tolerancing per ANSI Y14. 5M-1982.
4. "T", "D" and "E" are reference datums on the molded body and do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010 inch (.25mm) on any side.
5. These dimensions measured with the leads constrained to be perpendicular to plane T.
6. Pin numbers start with pin #1 and continue counterclockwise to pin #40 when viewed from the top.

853-0415 84971

NW3

Section 7 – Package outlines

44-PIN PLASTIC LEADED CHIP CARRIER

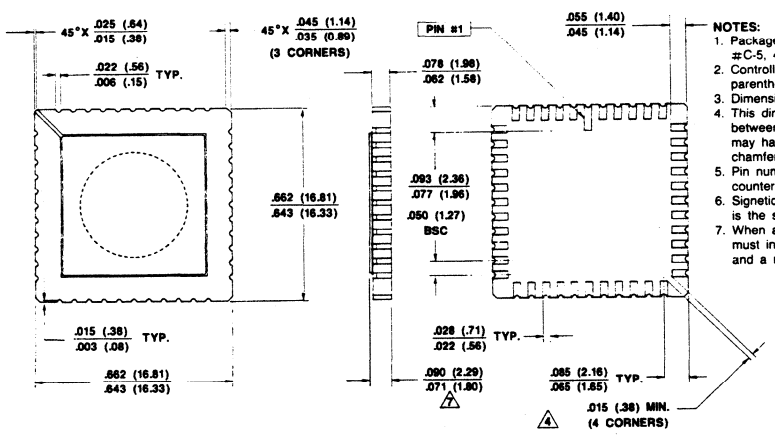


NOTES:
1. Package dimensions conform to JEDEC specification MO-047-AL for plastic leaded chip carrier 44 leads, .050 inch lead spacing, square. (issue A, 10/31/84).
2. Controlling dimensions: inches. Metric dimensions in mm are shown in parentheses.
3. Dimensioning and tolerancing per ANSI Y14.5M-1982.
4. Datum plane "H" located at the top of mold parting line and coincident with top of lead, where lead exits plastic body.
5. Location to datum "A" and "B" to be determined at plane "H". These Datums do not include mold flash. Mold flash protrusion shall not exceed .006" (0.15 mm) on any side.
6. Datum "D-E" and "F-G" are determined where these center leads exit from the body at plane "H".
7. Pin numbers continue counterclockwise to pin #44 (top view).
8. Signetics order code for product packaged in a PLCC is the suffix "A" after the product number.
9. Applicable to packages with pedestal only.

AX1 853-0403 93885

AX1

44-PIN SQUARE CERAMIC LEADLESS CHIP CARRIER



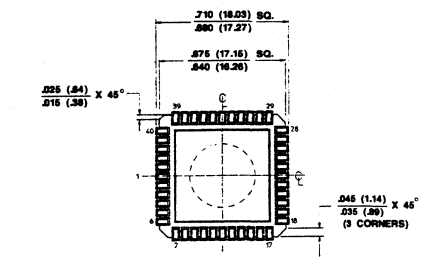
NOTES:
1. Package dimensions conform to MIL-M-38510, outline #C-5, 44 leads, square ceramic leadless chip carrier.
2. Controlling dimension: inches millimeters are shown in parenthesis.
3. Dimension and tolerancing per ANSI Y14.5M - 1982.
4. This dimension represents the minimum spacing between the corner contact pads. These corner pads may have a .020 inch by 45 degree maximum chamfer to accomplish the 0.15 minimum spacing.
5. Pin numbers start with pin #1 and continue counterclockwise to pin #44 when viewed from top.
6. Signetics order code for product packaged in a CCCL is the suffix "G" after the product number.
7. When a window lid is used, the overall thickness must increase by a minimum of .010 inch (0.254mm) and a maximum of .040 (1.020mm).

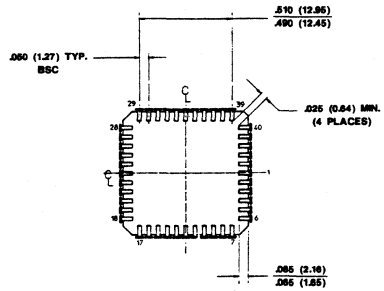
853-0803 95005

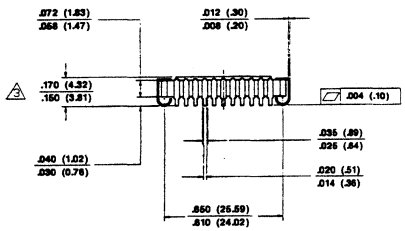
GX1

Section 7 – Package outlines

44-PIN SQUARE CERAMIC LEADED CHIP CARRIER







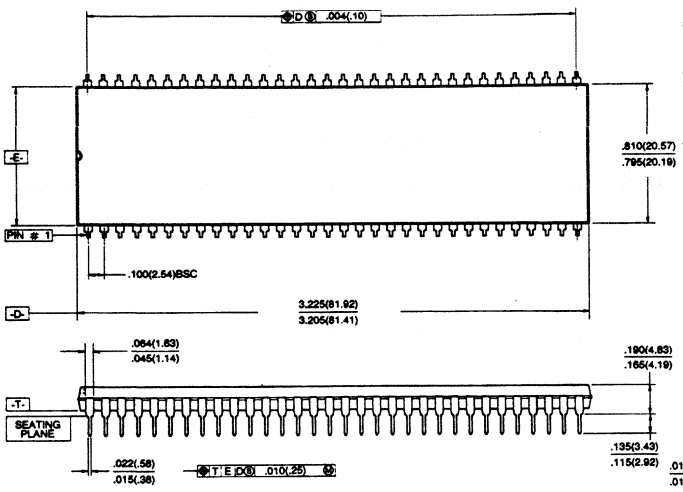
NOTES:
1. Controlling dimensions: inches. Millimeters are shown in parenthesis.
2. Dimension and toleracing per ANSI Y14.5M-1982.
3. When a window lid is used, the overall package thickness must increase by a minimum of .010 inch (0.254mm) and a maximum of .040 inch (1.020mm).

853-1268 95007

LX1

Section 7 – Package outlines

64-PIN PLAST'C DUAL IN-LINE

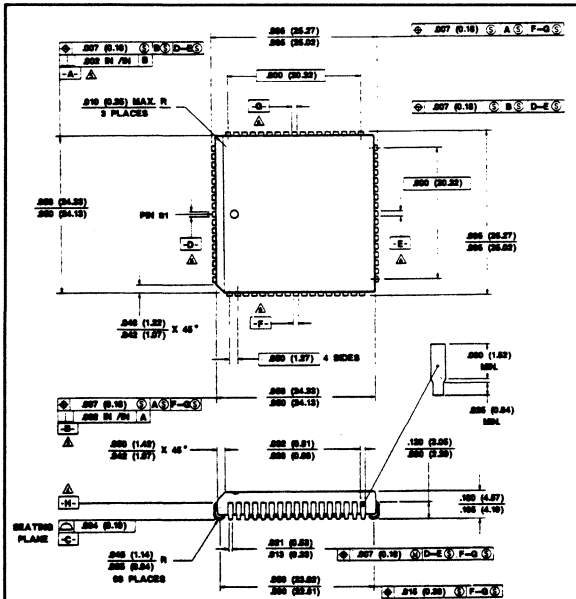


NOTES:
1. Controlling dimension: inches. Metric are shown in parentheses.
2. Dimensions and tolerancing per ANSI Y14.5M – 1982.
3. "T", "D", and "E" are reference datums on the molded body and do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010 inch (.25mm) on any side.
4. These dimensions measured with the leads constrained to be perpendicular to plane T.
5. Pin numbers start with pin #1 and continue counterclockwise to pin #64 when viewed from the top.

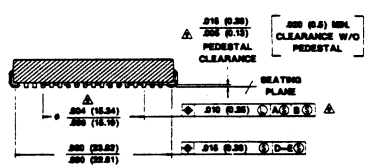
853-0414 81240

NS4

68-PIN PLASTIC LEADED CHIP CARRIER



NOTES:
1. Package dimensions conform to JEDEC specification MO-047-AE for plastic leaded chip carrier 68 leads, .050 inch lead spacing, square. (issue A, 10/31/84).
2. Controlling dimensions: inches. Metric dimensions in mm are shown in parentheses.
3. Dimensioning and tolerancing per ANSI Y14.5M-1982.
4. Datum plane "H" located at the top of mold parting line and coincident with top of lead, where lead exits plastic body.
5. Location to datum "A" and "B" to be determined at plane "H". These Datums do not include mold flash. Mold flash protrusion shall not exceed .006" (0.15 mm) on any side.
6. Datum "D-E" and "F-G" are determined where these center leads exit from the body at plane "H".
7. Pin numbers continue counterclockwise to pin #68 (top view).
8. Signetics order code for product packaged in a PLCC is the suffix "A" after the product number.
9. Applicable to packages with pedestal only.

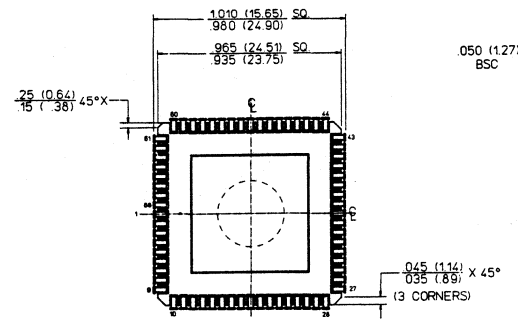


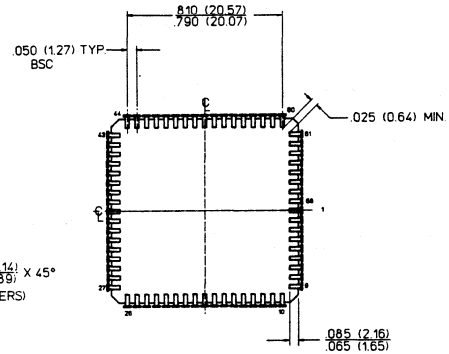
AB1 853-0398 92630

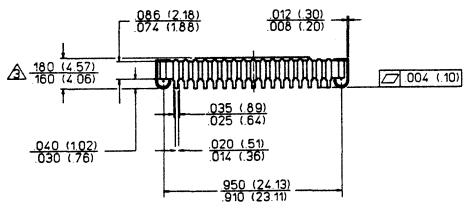
AB1

Section 7 – Package outlines

68-PIN SQUARE CERAMIC LEADLESS CHIP CARRIER







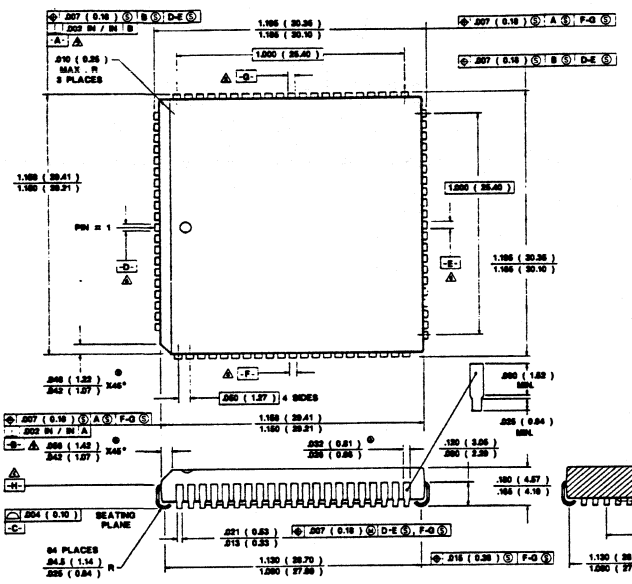
NOTES:

1. CONTROLLING DIMENSIONS: INCHES. MILLIMETERS ARE SHOWN IN PARENTHESIS.
2. DIMENSION AND TOLERANCING PER ANSI Y14.5M-1982.
3. WHEN A WINDOW LID IS USED, THE OVERALL PACKAGE THICKNESS MUST INCREASED BY A MINIMUM OF .010 INCH (0.25mm) AND A MAXIMUM OF .040 INCH (1.020mm).

853-1240 95007 LB

Section 7 – Package outlines

84-PIN SQUARE PLASTIC LEADED CHIP CARRIER

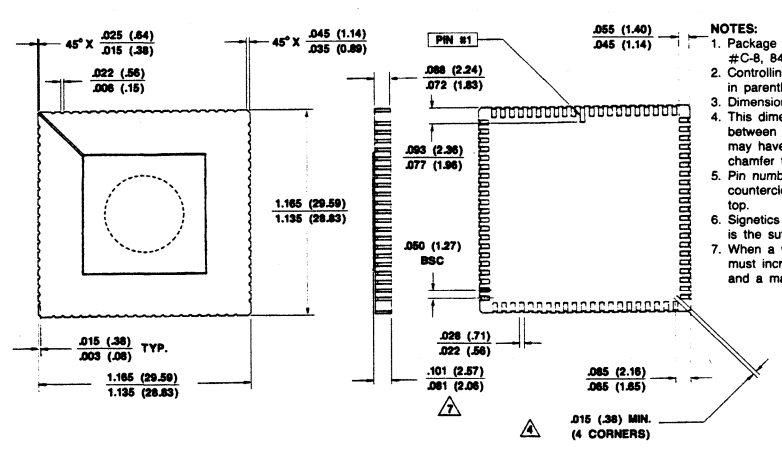


NOTES:
1. Package dimensions conform to JEDEC specification MO-047-AF for plastic leaded chip carrier 84 leads, .050 inch lead spacing, square. (issue A, 10/31/84).
2. Controlling dimensions: inches. Metric dimensions in mm are shown in parentheses.
3. Dimensioning and tolerancing per ANSI Y14.5M-1982.
4. Datum plane "H" located at the top of mold parting line and coincident with top of lead, where lead exits plastic body.
5. Location to datum "A" and "B" to be determined at plane "H". These Datums do not include mold flash. Mold flash protrusion shall not exceed .006" (0.15 mm) on any side.
6. Datum "D-E" and "F-G" are determined where these center leads exit from the body at plane "H".
7. Pin numbers continue counterclockwise to pin #84 (top view).
8. Signetics order code for product packaged in a PLCC is the suffix "A" after the product number.
9. Applicable to packages with pedestal only.

AC1 853-0399 93885

AC1

84-PIN SQUARE CERAMIC LEADLESS CHIP CARRIER



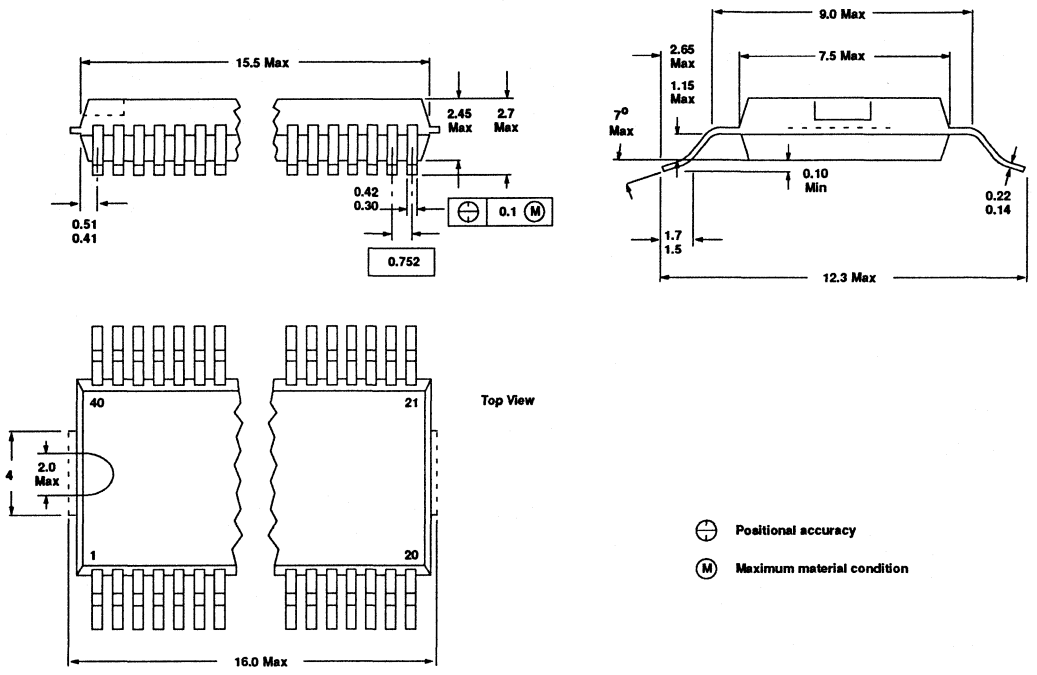
NOTES:
1. Package dimensions conform to MIL-M-38510, outline #C-8, 84 leads, square ceramic leadless chip carrier.
2. Controlling dimension: inches. Millimeters are shown in parenthesis.
3. Dimensioning and tolerancing per ANSI Y14.5M-1982.
4. This dimension represents the minimum spacing between the corner contact pads. These corner pads may have a .020 inch by 45 degree maximum chamfer to accomplish the .015 minimum spacing.
5. Pin numbers start with pin #1 and continue counterclockwise to pin #84 when viewed from the top.
6. Signetics order code for product packaged in a CCCL is the suffix "G" after the product number.
7. When a window lid is used, the overall thickness must increase by a minimum of .010 inch (0.254mm) and a maximum of .040 (1.020mm).

GC1 853-0805 95005

GC1

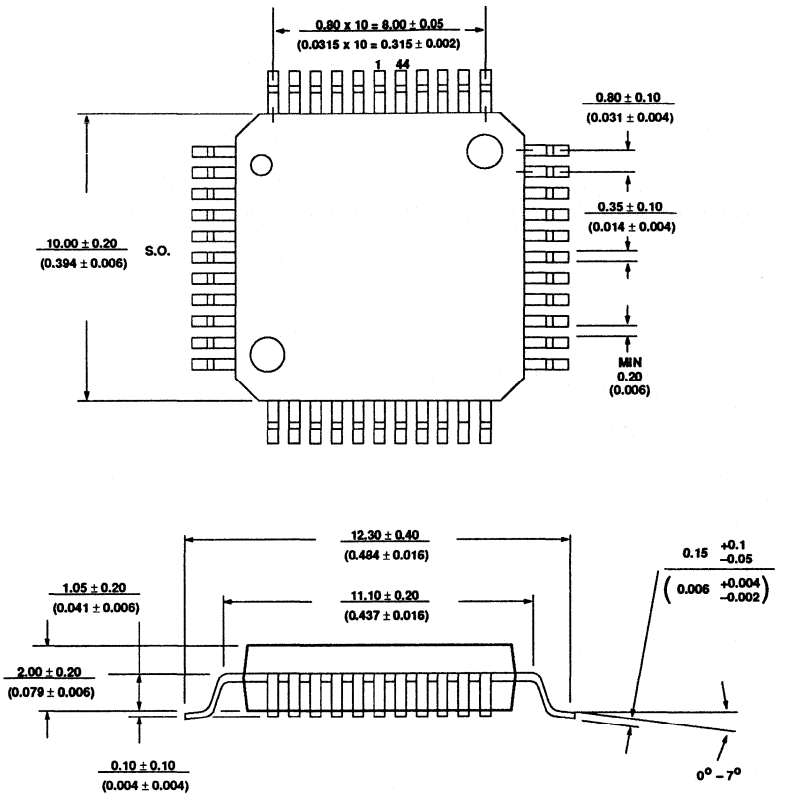
Section 7 – Package outlines

40-LEAD MINI-PACK; PLASTIC (VSO40; SOT158A)



Section 7 – Package outlines

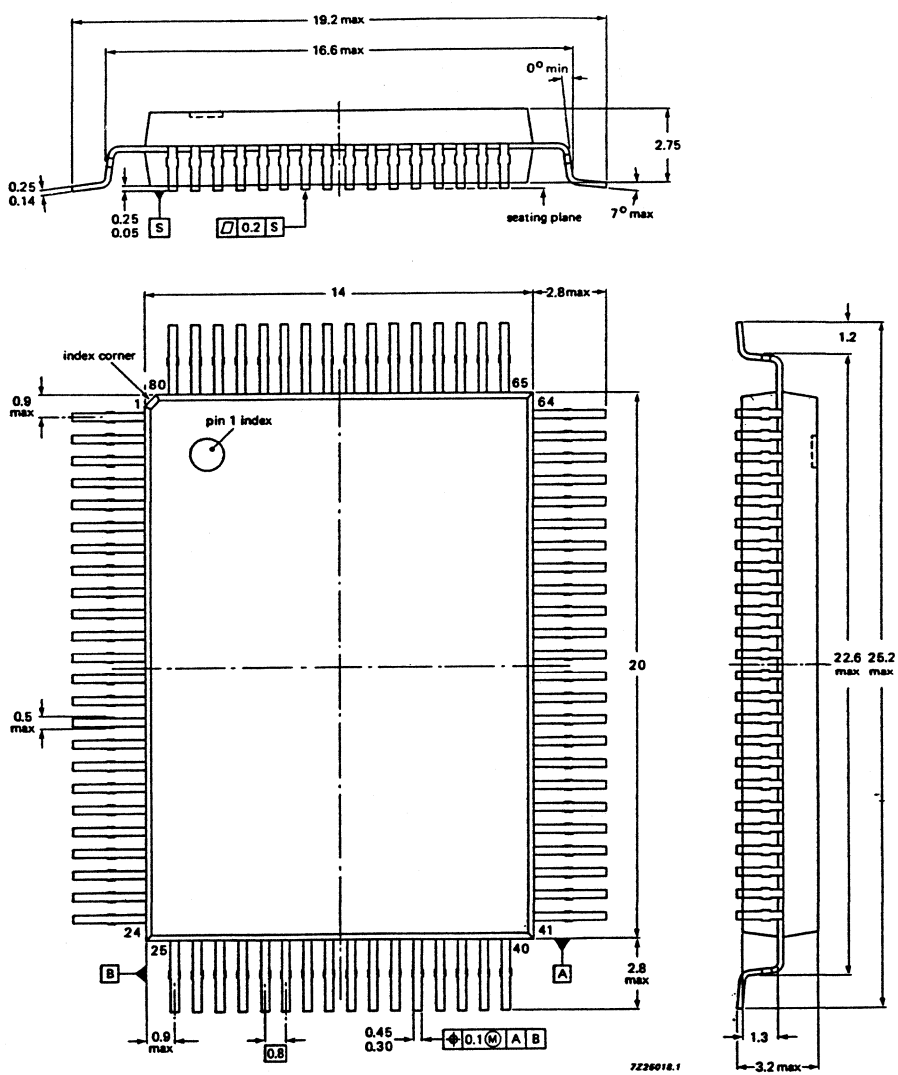
44-PIN QUAD FLAT-PACK



NOTE: All dimensions in mm.

Section 7 – Package outlines

80-LEAD QUAD FLAT-PACK; PLASTIC (SOT219)



Dimensions in mm

⊕ Positional accuracy.

Ⓜ Maximum Material Condition.

DATA HANDBOOK SYSTEM

DATA HANDBOOK SYSTEM

Our Data Handbook System comprises more than 60 books with specifications on electronic components, subassemblies and materials. It is made up of seven series of handbooks:

INTEGRATED CIRCUITS

DISCRETE SEMICONDUCTORS

DISPLAY COMPONENTS

PASSIVE COMPONENTS*

PROFESSIONAL COMPONENTS**

MAGNETIC PRODUCTS*

LIQUID CRYSTAL DISPLAYS

The contents of each series are listed on pages III to IX.

The data handbooks contain all pertinent data available at the time of publication, and each is revised and reissued periodically.

Where application is given it is advisory and does not form part of the product specification.

Condensed data on the preferred products of Philips Components is given in our Preferred Type Range catalogue (issued annually).

Information on current Data Handbooks and how to obtain a subscription for future issues is available from any of the Organizations listed on the back cover.
Product specialists are at your service and enquiries will be answered promptly.

* Will replace the Components and materials (green) series of handbooks.
** Will replace the Electron tubes (blue) series of handbooks.

INTEGRATED CIRCUITS

This series of handbooks comprises:

code	handbook title
IC01	**Radio, audio and associated systems** Bipolar, MOS
IC02a/b	**Video and associated systems** Bipolar, MOS
IC03	**ICs for Telecom** Bipolar, MOS Subscriber sets, Cordless Telephones
IC04	**HE4000B logic family** CMOS
IC05	**Advanced Low-power Schottky (ALS) Logic Series**
IC06	**High-speed CMOS; PC74HC/HCT/HCU** Logic family
IC07	**Advanced CMOS logic (ACL)**
IC08	**10/100K ECL Logic/Memory/PLD**
IC09	**TTL logic series**
IC10	**Memories** MOS, TTL, ECL
IC11	**Linear Products**
IC12	**I^2C-bus compatible ICs**
IC13	**Semi-custom** Programmable Logic Devices (PLD)
IC14	**Microcontrollers** NMOS, CMOS
IC15	**FAST TTL logic series**
Supplement to IC15	**FAST TTL logic series**
IC16	**CMOS integrated circuits for clocks and watches**
IC17	**ICs for Telecom** Bipolar, MOS Radio pagers Mobile telephones ISDN
IC18	**Microprocessors and peripherals**
IC19	**Data communication products**
IC20	**8051-based 16-bit microcontrollers**
IC23	**Advanced BiCMOS interface logic**

DISCRETE SEMICONDUCTORS

This series of data handbooks comprises:

current code	new code	handbook title
S1	**SC01**	**Diodes** High-voltage tripler units
S2a	**SC02**	**Power diodes**
S2b	**SC03**	**Thyristors and triacs**
S3	**SC04**	**Small-signal transistors**
S4a	**SC05**	**Low-frequency power transistors and hybrid IC power modules**
S4b	**SC06**	**High-voltage and switching power transistors**
S5	**SC07**	**Small-signal field-effect transistors**
S6	**SC08a***	**RF power bipolar transistors**
	SC08b	**RF power MOS transistors**
	SC09	**RF power modules**
S7	**SC10**	**Surface mounted semiconductors**
S8b	**SC12**	**Optocouplers**
S9	**SC13***	**PowerMOS transistors**
S10	**SC14**	**Wideband transistors and wideband hybrid IC modules**
S11	**SC15**	**Microwave transistors**
S15**	**SC16**	**Laser diodes**
S13	**SC17**	**Semiconductor sensors**

* Not yet issued with the new code in this series of handbooks.
** New handbook in this series; will be issued shortly.

DISPLAY COMPONENTS

This series of data handbooks comprises:

code	handbook title
DC01	**Colour display components** Colour TV Picture Tubes and Assemblies Colour Monitor Tube Assemblies
DC02	**Monochrome monitor tubes and deflection units**
DC03	**Television tuners, coaxial aerial input assemblies**
DC04	**Loudspeakers**
DC05	**Flyback transformers, mains transformers and general-purpose FXC assemblies**

PASSIVE COMPONENTS

This series of data handbooks comprises:

current code	new code	handbook title
C14	**PA01**	**Electrolytic capacitors; solid and non-solid**
C11	**PA02**	**Varistors, thermistors and sensors**
C12	**PA03**	**Potentiometers and switches**
C7	**PA04**	**Variable capacitors**
C22	**PA05***	**Film capacitors**
C15	**PA06**	**Ceramic capacitors**
C9	**PA07***	**Piezoelectric quartz devices**
C13	**PA08**	**Fixed resistors**

* Not yet issued with the new code in this series of handbooks.

PROFESSIONAL COMPONENTS

This series of data handbooks comprises:

current code	new code	handbook title
T3	PC01	High-power klystrons and accessories
T5	PC02*	Cathode-ray tubes
T6	PC03*	Geiger-Müller tubes
T9	PC04	Photo multipliers
T10	PC05	Plumbicon camera tubes and accessories
T11	PC06	Circulators and Isolators
T12	PC07	Vidicon and Newvicon camera tubes and deflection units
T13	PC08	Image intensifiers
T15	PC09	Dry-reed switches
	PC11**	Solid state image sensors and peripherals integrated circuits
T9	PC12*	Electron multipliers

* Not yet issued with the new code in this series of handbooks.
** Will be issued as IC23 in the future.

MAGNETIC PRODUCTS

This series of data handbooks comprises:

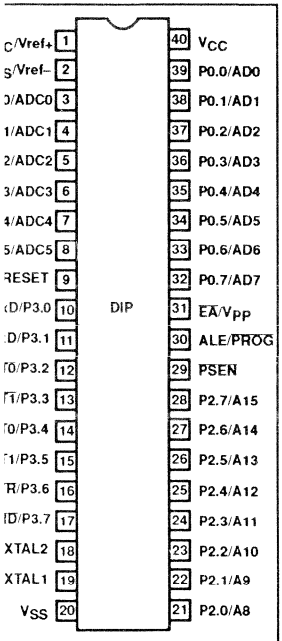
current code	new code	handbook title
C4 C5	MA01	**Soft Ferrites**
C16	MA02*	**Permanent magnet materials**
C19	MA03*	**Piezoelectric ceramics**

* Not yet issued with the new code in this series of handbooks.

LIQUID CRYSTAL DISPLAYS

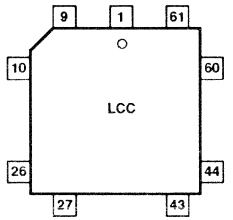
current code	new code	handbook title
S14	**LCD01**	**Liquid Crystal Displays and driver ICs for LCDs**

80/83/87C550



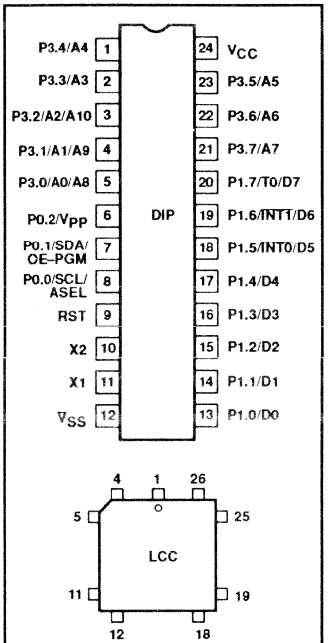
in	Function	Pin	Function
1	AV_{CC}	23	XTAL1
2	Vref+	24	V_{SS}
3	Vref−	25	P2.0/A8
4	AV_{SS}	26	P2.1/A9
5	P1.0/ADC0	27	P2.2/A10
6	P1.1/ADC1	28	P2.3/A11
7	P1.2/ADC2	29	P2.4/A12
8	P1.3/ADC3	30	P2.5/A13
9	P1.4/ADC4	31	P2.6/A14
0	P1.5/ADC5	32	P2.7/A15
1	P1.6/ADC6	33	PSEN
2	P1.7/ADC7	34	ALE/PROG
3	RESET	35	EA/V_{PP}
4	P3.0/RxD	36	P0.7/AD7
5	P3.1/TxD	37	P0.6/AD6
6	P3.2/INT0	38	P0.5/AD5
7	P3.3/INT1	39	P0.4/AD4
8	P3.4/T0	40	P0.3/AD3
9	P3.5/T1	41	P0.2/AD2
0	P3.6/WR	42	P0.1/AD1
1	P3.7/RD	43	P0.0/AD0
2	XTAL2	44	V_{CC}

80/83/87C552*, 80/83C562



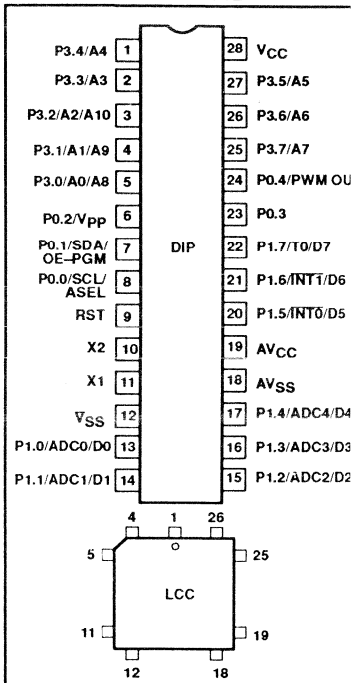
Pin	Function	Pin	Function
1	P5.0/ADC0	35	XTAL1
2	V_{DD}	36	V_{SS}
3	STADC	37	V_{SS}
4	PWM0	38	NC
5	PWM1	39	P2.0/A08
6	EW	40	P2.1/A09
7	P4.0/CMSR0	41	P2.2/A10
8	P4.1/CMRS1	42	P2.3/A11
9	P4.2CMSR2	43	P2.4/A12
10	P4.3/CMSR3	44	P2.5/A13
11	P4.4/CMSR4	45	P2.6/A14
12	P4.5/CMSR5	45	P2.7/A15
13	P4.6/CMT0	47	PSEN
14	P4.7/CMT1	48	ALE/PROG
15	RST	49	EA/V_{PP}
16	P1.0/CT0I	50	P0.7/AD7
17	P1.1/CT1I	51	P0.6/AD6
18	P1.2/CT2I	52	P0.5/AD5
19	P1.3/CT3I	53	P0.4/AD4
20	P1.4/T2	54	P0.3/AD3
21	P1.5/RT2	55	P0.2/AD2
22	P1.6	56	P0.1/AD1
23	P1.7	57	P0.0/AD0
24	P3.0/RxD	58	AVref−
25	P3.1TxD	59	AVref+
26	P3.2/INT0	60	AV_{SS}
27	P3.3/INT1	61	AV_{DD}
28	P3.4/T0	62	P5.7/ADC7
29	P3.5/T1	63	P5.6/ADC6
30	P3.6/WR	64	P5.5/ADC5
31	P3.7/RD	65	P5.4/ADC4
32	NC	66	P5.3/ADC3
33	NC	67	P5.2/ADC2
34	XTAL2	68	P5.1/ADC1

83/87C751



Pin	Function	Pin	Function
1	P3.4/A4	15	P1.0/D0
2	P3.3/A3	16	P1.1/D1
3	P3.2/A2/A10	17	P1.2/D2
4	P3.1/A1/A9	18	P1.3/D3
5	N.C.	19	P1.4/D4
6	P3.0/A0/A8	20	P1.5/INT0/D5
7	$P0.2/V_{PP}$	21	N.C.
8	P0.1/SDA/OE–PGM	22	N.C.
9	P0.0/SCL/ASEL	23	P1.6/INT1/D6
10	N.C.	24	P1.7/T0/D7
11	RST	25	P3.7/A7
12	X2	26	P3.6/A6
13	X1	27	P3.5/A5
14	V_{SS}	28	V_{CC}

83/87C752



Pin	Function	Pin	Function
1	P3.4/A4	15	P1.2/ADC2/D2
2	P3.3/A3	16	P1.3/ADC3/D3
3	P3.2/A2/A10	17	P1.4/ADC4/D4
4	P3.1/A1/A9	18	AV_{SS}
5	P3.0/A0/A8	19	AV_{CC}
6	$P0.2/V_{PP}$	20	P1.5/INT0/D5
7	P0.1/SDA/OE–PGM	21	P1.6/INT1/D6
8	P0.0/SCL/ASEL	22	P1.7/T0/D7
9	RST	23	P0.3
10	X2	24	P0.4/PWM OUT
11	X1	25	P3.7/A7
12	V_{SS}	26	P3.6/A6
13	P1.0/ADC0/D0	27	P3.5/A5
14	P1.1/ADC1/D1	28	V_{CC}

1.6 and P1.7 have the alternate functions SCL and SDA, respectively,
n the 80/83/87C652, 80/83/87C654, 80/83/87C52I, and 80/83/87C556.

Philips Components – a worldwide company

Argentina: PHILIPS ARGENTINA S.A., Div. Philips Components, Vedia 3892, 1430 BUENOS AIRES, Tel. (01) 541-4261.
Australia: PHILIPS COMPONENTS PTY Ltd, 11 Waltham Street, ARTARMON, N.S.W. 2064, Tel. (02) 439 3322.
Austria: ÖSTERREICHISCHE PHILIPS INDUSTRIE G.m.b.H., UB Bauelemente, Triester Str. 64, 1101 WIEN, Tel. (0222) 60 101-820.
Belgium: N.V PHILIPS PROF. SYSTEMS – Components Div., 80 Rue Des Deux Gares, B-1070 BRUXELLES, Tel. (02) 52 56 111.
Brazil: PHILIPS COMPONENTS (Active Devices & LCD) Av. das Nacoes Unidas, 12495-SAO PAULO-SP, CEP 04578, P.O. Box 7383, Tel. (011) 534-2211.
PHILIPS COMPONENTS (Passive Devices & Materials) Av. Francisco Monteiro 702, RIBEIRAO PIRES-SP, CEP 09400, Tel. (011) 459-8211.
Canada: PHILIPS ELECTRONICS LTD., Philips Components, 601 Milner Ave., SCARBOROUGH, Ontario, M1B 1M8, Tel. (416) 292-5161.
(IC Products) PHILIPS COMPONENTS – Signetics Canada LTD., 1 Eva Road, Suite 411, ETOBICOKE, Ontario, M9C 4Z5, Tel. (416) 626-6676.
Chile: PHILIPS CHILENA S.A., Av. Santa Maria 0760, SANTIAGO, Tel. (02) 77 38 16.
Colombia: IPRELENSO LTDA., Carrera 21 No. 56-17, BOGOTA, D.E., P.O. Box 77621, Tel. (01) 2 49 76 24.
Denmark: PHILIPS COMPONENTS A/S, Prags Boulevard 80, PB 1919, DK-2300 COPENHAGEN S, Tel. 01-54 11 33.
Finland: PHILIPS COMPONENTS, Sinikalliontie 3, SF-2630 ESPOO, Tel. 358-0-50 261.
France: PHILIPS COMPOSANTS, 117 Quai du Président Roosevelt, 92134 ISSY-LES-MOULINEAUX Cedex, Tel. (01) 40 93 80 00.
Germany: PHILIPS COMPONENTS UB der Philips G.m.b.H., Valvo Haus, Burchardstrasse 19, D-2 HAMBURG 1, Tel. (040) 3296-0.
Greece: PHILIPS HELLENIQUE S.A., Components Division, No. 15, 25th March Street, GR 17778 TAVROS, Tel. (01) 4894 339/4894 911.
Hong Kong: PHILIPS HONG KONG LTD., Components Div., 15/F Philips Ind. Bldg., 24-28 Kung Yip St., KWAI CHUNG, Tel. (0)-42 45 121.
India: PEICO ELECTRONICS & ELECTRICALS LTD., Components Dept., Shivsagar Estate 'A' Block, P.O. Box 6598, 254-D Dr. Annie Besant Rd., BOMBAY – 40 018, Tel. (022) 49 21 500-49 21 515.
Indonesia: P.T. PHILIPS-RALIN ELECTRONICS, Components Div., Setiabudi II Building, 6th Fl., Jalan H.R. Rasuna Said (P.O. Box 223/KBY) Kuningan, JAKARTA 12910, Tel. (021) 51 79 95.
Ireland: PHILIPS ELECTRONICS (IRELAND) LTD., Components Division, Newstead, Clonskeagh, DUBLIN 14, Tel. (01) 69 33 55.
Italy: PHILIPS S.p.A., Philips Components, Piazza IV Novembre 3, I-20124 MILANO, Tel. (02) 6752.1.
Japan: PHILIPS JAPAN LTD., Components Division, Philips Bldg 13-37, Kohnan 2-chome, Minato-ku, TOKYO 108, Tel. (03) 740-5028.
Korea (Republic of): PHILIPS ELECTRONICS (KOREA) LTD. Components Division, Philips House, 260-199 Itaewon-dong, Yongsan-ku, SEOUL, Tel. (02) 794-5011.
Malaysia: PHILIPS MALAYSIA SDN BHD, Components Div., 3 Jalan SS15/2A SUBANG, 47500 PETALING JAYA, Tel. (03) 73 45 511.
Mexico: PHILIPS COMPONENTS, Paseo Triunfo de la Republica, No 215 Local 5, Cd Juarez CHI HUA HUA 32340 MEXICO Tel. (16) 18-67-01/02.
Netherlands: PHILIPS NEDERLAND B.V., Marktgroep Philips Components, Postbus 90050, 5600 PB EINDHOVEN, Tel. (040) 78 37 49.
New Zealand: PHILIPS NEW ZEALAND LTD., Components Division, 110 Mt. Eden Road, C.P.O. Box 1041, AUCKLAND, Tel. (09) 605-914.
Norway: NORSK A/S PHILIPS, Philips Components, Box 1, Manglerud 0612, OSLO, Tel. (02) 68 02 00.
Pakistan: PHILIPS ELECTRICAL CO. OF PAKISTAN LTD., Philips Markaz, M.A. Jinnah Rd., KARACHI-3, Tel. (021) 72 57 72.
Peru: CADESA, Carretera Central 6.500, LIMA 3, Apartado 5612, Tel. 51-14-35 00 59.
Philippines: PHILIPS ELECTRICAL LAMPS INC. Components Div., 106 Valero St. Salcedo Village, P.O. Box 911, MAKATI, Metro MANILA, Tel. (63-2) 810-0161.
Portugal: PHILIPS PORTUGUESA S.A.R.L., Av. Eng. Duarte Pacheco 6, 1009 LISBOA Codex, Tel. (019) 68 31 21.
Singapore: PHILIPS SINGAPORE, PTE LTD., Components Div., Lorong 1, Toa Payoh, SINGAPORE 1231, Tel. 35 02 000.
South Africa: S.A. PHILIPS PTY LTD., Components Division, JOHANNESBURG 2000, P.O. Box 7430.
Spain: PHILIPS COMPONENTS, Balmes 22, 08007 BARCELONA, Tel. (03) 301 63 12.
Sweden: PHILIPS COMPONENTS, A.B., Tegeluddsvägen 1, S-11584 STOCKHOLM, Tel. (0)8-782 1 000.
Switzerland: PHILIPS A.G., Components Dept., Allmendstrasse 140-142, CH-8027 ZÜRICH, Tel. (01) 488 22 11.
Taiwan: PHILIPS TAIWAN LTD., 581 Min Sheng East Road, P.O. Box 22978, TAIPEI 10446, Taiwan, Tel. 886-2-5005899.
Thailand: PHILIPS ELECTRICAL CO. OF THAILAND LTD., 283 Silom Road, P.O. Box 961, BANGKOK, Tel. (02) 233-6330-9.
Turkey: TÜRK PHILIPS TICARET A.S., Philips Components, Talatpasa Cad. No. 5, 80640 LEVENT/ISTANBUL, Tel. (01) 179 27 70.
United Kingdom: PHILIPS COMPONENTS LTD., Mullard House, Torrington Place, LONDON WC1E 7HD, Tel. (071) 580 6633.
United States: (Colour picture tubes – Monochrome & Colour Display Tubes) PHILIPS DISPLAY COMPONENTS COMPANY, 1600 Huron Parkway, P.O. Box 963, ANN ARBOR, Michigan 48106, Tel. 313/996-9400.
(IC Products) PHILIPS COMPONENTS – Signetics, 811 East Arques Avenue, SUNNYVALE, CA 94088-3409, Tel. (408) 991-2000.
(Passive Components, Discrete Semiconductors, Materials and Professional Components & LCD) PHILIPS COMPONENTS, Discrete Products Division, 2001 West Blue Heron Blvd., P.O. Box 10330, RIVIERA BEACH, Florida 33404, Tel. (407) 881-3200.
Uruguay: PHILIPS COMPONENTS, Coronel Mora 433, MONTEVIDEO, Tel. (02) 70-40 44.
Venezuela: MAGNETICA S.A., Calle 6, Ed. Las Tres Jotas, CARACAS 1074A, App. Post. 78117, Tel. (02) 241 75 09.
Zimbabwe: PHILIPS ELECTRICAL (PVT) LTD., 62 Mutare Road, HARARE, P.O. Box 994, Tel. 47211.

For all other countries apply to: Philips Components Division, Strategic Accounts and International Sales, P.O. Box 218, 5600 MD EINDHOVEN, The Netherlands, Telex 35000 phtcnl, Fax. +31-40-723753

AS83

Printed in The Netherlands

9398 173 30011

Philips Components

PHILIPS

PHILIPS