



## HD6301/HD6303 SERIES HANDBOOK

- USER'S MANUALS
- SOFTWARE APPLICATION NOTES
- HARDWARE APPLICATION NOTES
- WIDE TEMPERATURE SPECIFICATIONS



#U07

# HD6301/HD6303 SERIES HANDBOOK

## ■ USER'S MANUALS:

- HD6301V1/HD6303R
- HD63701V
- HD6301X0/HD6303X/HD63701X0
- HD6301Y0/HD6303Y/HD63701Y0

## ■ SOFTWARE APPLICATION NOTES

## ■ HARDWARE APPLICATION NOTES

## ■ C LANGUAGE PROGRAMMING TECHNIQUES

## ■ APPENDIX:

- HD6301V/HD6303R Q and A
- HD6301X0/HD6303X OSCILLATOR CIRCUIT
- WIDE TEMPERATURE RANGE SPECIFICATIONS  
-40°C to +85°C (J VERSION)

When using this document, keep the following in mind:

1. This document may, wholly or partially, be subject to change without notice.
2. All rights are reserved: No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without Hitachi's permission.
3. Hitachi will not be held responsible for any damage to the user that may result from accidents or any other reasons during operation of the user's unit according to this document.
4. Circuitry and other examples described herein are meant merely to indicate the characteristics and performance of Hitachi's semiconductor products. Hitachi assumes no responsibility for any intellectual property claims or other problems that may result from applications based on the examples described herein.
5. No license is granted by implication or otherwise under any patents or other rights of any third party or Hitachi, Ltd.
6. **MEDICAL APPLICATIONS:** Hitachi's products are not authorized for use in **MEDICAL APPLICATIONS** without the written consent of the appropriate officer of Hitachi's sales company. Such use includes, but is not limited to, use in life support systems. Buyers of Hitachi's products are requested to notify the relevant Hitachi sales offices when planning to use the products in **MEDICAL APPLICATIONS**.

# INDEX

Quick Reference Guide

1

Addressing Modes, CPU Architecture, and Instruction Set

2

HD6301V1/HD6303R User's Manual

3

HD63701V User's Manual

4

HD6301X0/HD6303X/HD63701X0 User's Manual

5

HD6301Y0/HD6303Y/HD63701Y0 User's Manual

6

Software Application Notes

7

Hardware Application Notes

8

C Language Programming Techniques

9

APPENDIX:  
HD6301V1/HD6303R Q and A—HD6301X0/HD6303X Oscillation Circuit  
Wide Temperature Range Specifications,  $-40^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$ , (J Version)  
Hitachi Sales Offices

Section 10, Page 1318

10



# TABLE OF CONTENTS

<b>Section 1</b>	
<b>Quick Reference Guide</b>	<b>Page</b>
QUICK REFERENCE GUIDE .....	1
PACKAGE REFERENCE GUIDE .....	4

<b>Section 2</b>	
<b>Addressing Modes, CPU Architecture, and Instruction Set</b>	<b>Page</b>
1. ASSEMBLY LANGUAGE .....	17
1.1 Addressing Modes .....	17
1.2 CPU Registers .....	26
1.3 Instruction Set Details .....	28

<b>Section 3</b>	
<b>HD6301V1/HD6303R User's Manual</b>	<b>Page</b>
Notice on HD6301V1 and HD6303R .....	126
1. OVERVIEW .....	128
1.1 Features of HD6301V1 .....	128
1.2 Block Diagram .....	128
1.3 Functional Pin Description .....	129
2. INTERNAL ARCHITECTURE AND OPERATIONS .....	130
2.1 Mode Selection .....	130
2.2 Memory Map .....	135
2.3 Registers .....	140
2.4 I/O Ports .....	142
2.5 Programmable Timer .....	146
2.6 Serial Communication Interface .....	150
2.7 Interrupts .....	157
2.8 Reset .....	159
2.9 Oscillator .....	161
2.10 Strobe Signals .....	161
2.11 RAM Control Register .....	162
2.12 Low Power Consumption Mode .....	163
2.13 TRAP Function .....	167

3. INSTRUCTIONS .....	168
3.1 Addressing Modes .....	168
3.2 Instruction Set .....	170
3.3 Instruction Execution Cycles .....	175
3.4 System Flowchart .....	181
4. PIN ARRANGEMENT AND PACKAGE INFORMATION .....	183
5. ELECTRICAL CHARACTERISTICS .....	186
6. APPLICATIONS .....	191
6.1 Use of External Expanded Mode .....	191
6.2 Standby Mode .....	195
6.3 Address Trap, OP-Code Trap Application .....	198
6.4 Slow Memory Interface .....	199
6.5 Interface to HN61256 .....	200
6.6 Interface to the Realtime Clock (HD146818) .....	205
6.7 Reference Data of Battery Service Life .....	208
7. PRECAUTIONS .....	209
7.1 Write-Only Register .....	209
7.2 Address Strobe (AS) .....	209
7.3 Mode 0 .....	209
7.4 Trap Interrupt .....	209
7.5 Power-on Reset .....	211
7.6 Precaution to the Board Design of Oscillation Circuit .....	211
7.7 Application Note for High Speed System Design Using the HD6301V1 ..	212
APPENDIX	
I. EPROM on Package HD63P01M1 .....	218
II. Program Development Procedure and Support System .....	226
III. Q & A .....	233

## Section 4

### HD63701V User's Manual

	Page
1. OVERVIEW .....	251
1.1 Features of HD63701V0 .....	251
1.2 Block Diagram .....	251
1.3 Functional Pin Description .....	252

2. INTERNAL ARCHITECTURE .....	253
2.1 Mode Selection .....	253
2.2 Memory Map .....	258
2.3 CPU Registers .....	263
2.4 Ports .....	265
2.5 Timer .....	269
2.6 Serial Communication Interface .....	273
2.7 Interrupts .....	281
2.8 Reset .....	283
2.9 Oscillator .....	285
2.10 Strobe Signals .....	286
2.11 RAM Control Register .....	287
2.12 Low Power Consumption Mode .....	287
2.13 TRAP Function .....	292
3. EPROM (PROM) PROGRAMMING AND TECHNICAL SPECIFICATIONS ...	293
3.1 PROM Mode .....	293
3.2 Programming/Verification .....	294
3.3 Erasure .....	295
3.4 On-Chip PROM Characteristics and Application .....	296
3.5 Instruction Set Overview .....	300
3.6 Addressing Modes .....	300
3.7 Instruction Set .....	302
3.8 Instruction Execution Cycles .....	307
3.9 System Flowchart .....	313
3.10 Pin Arrangement and Package Information .....	315
3.11 Electrical Characteristics .....	316
4. APPLICATIONS .....	322
4.1 Use of External Expanded Mode .....	322
4.2 Standby Mode .....	327
4.3 Address Trap, OP-Code Trap Application .....	330
4.4 Slow Memory Interface .....	331
4.5 Interface to HN61256 .....	332
4.6 Interface to the Realtime Clock (HD146818) .....	337
4.7 Reference Data of Battery Service Life .....	340
5. PRECAUTIONS .....	341
5.1 Write Only Register .....	341
5.2 Address Strobe (AS) .....	341
5.3 Mode 0 .....	341
5.4 Trap Interrupt .....	341



5.5	Precaution to the Board Design of Oscillation Circuit . . . . .	343
5.6	Application Note for High Speed System Design Using the HD63701V0 . . . . .	343
5.7	Differences between HD6301V and HD63701V . . . . .	347

**APPENDIX**

I.	Support Tools . . . . .	353
II.	Q & A . . . . .	355

**Section 5**

**HD6301X0/HD6303X/HD63701X0 User's Manual**

**Page**

1.	OVERVIEW . . . . .	375
1.1	Features . . . . .	375
1.2	Block Diagrams . . . . .	377
1.3	Pin Description . . . . .	380
2.	INTERNAL ARCHITECTURE AND OPERATION . . . . .	383
2.1	Operation Modes . . . . .	383
2.2	Memory Map . . . . .	386
2.3	Function Pin Description . . . . .	388
2.4	Ports . . . . .	393
2.5	RAM/Port 5 Control Register . . . . .	397
3.	CPU FUNCTION . . . . .	399
3.1	CPU Registers . . . . .	399
3.2	Addressing Modes . . . . .	400
3.3	Instruction Set . . . . .	402
3.4	CPU Instruction Flow . . . . .	408
3.5	Low Power Dissipation Modes . . . . .	410
3.6	Trap Function . . . . .	413
3.7	Reset . . . . .	414
3.8	Interrupts . . . . .	415
4.	TIMER 1 . . . . .	418
4.1	Free-Running Counter (FRC) . . . . .	419
4.2	Output Compare Registers (OCR) . . . . .	419
4.3	Input Compare Register (ICR) . . . . .	420
4.4	Timer Control/Status Register 1 (TCSR1) . . . . .	420
4.5	Timer Control/Status Register 2 (TCSR2) . . . . .	421
4.6	Timer Status Flags . . . . .	423
4.7	Precautions on Cleaning the OCF . . . . .	423



5. TIMER 2 .....	425
5.1 Timer 2 Upcounter (T2CNT) .....	425
5.2 Timer Constant Register (TCONR) .....	426
5.3 Timer Control/Status Register 3 (TCSR3) .....	426
5.4 Timer Status Flags .....	428
6. SERIAL COMMUNICATIONS INTERFACE .....	429
6.1 Initialization .....	429
6.2 Asynchronous Mode .....	430
6.3 Clock Synchronous Mode .....	431
6.4 Transmit/Receive Control Status Register (TRCSR) .....	433
6.5 Transmit Rate/Mode Control Register (RMCR) .....	434
6.6 SCI Receiving Margin .....	438
6.7 Timer, SCI Status Flags .....	438
6.8 Precaution for Clock-Synchronous Serial Communication Interface .....	439
7. HD63701X0 PROGRAMMABLE ROM (EPROM) .....	441
7.1 Programming and Verification .....	443
7.2 Erasing (Window Package) .....	444
7.3 Characteristics and Applications .....	444
8. APPLICATIONS .....	447
8.1 HD6301X0 or HD63701X0 in Expanded Mode .....	447
8.2 HD6301X0 or HD63701X0 in Single-Chip Mode .....	448
8.3 Timer Applications .....	448
8.4 SCI Applications .....	452
8.5 Lowering Operating Current .....	454
8.6 Memory Ready Application .....	457
8.7 Halt Application .....	458
8.8 $\overline{RD}$ , $\overline{WR}$ Application .....	459
8.9 LCD-II Interface Application .....	460
8.10 Oscillation Board Design .....	461
APPENDIX I. ELECTRICAL CHARACTERISTICS .....	463
APPENDIX II. INSTRUCTION EXECUTION CYCLES .....	485
APPENDIX III. QUESTIONS AND ANSWERS .....	491
III.1 Parallel Ports .....	491
III.2 Serial Ports .....	492
III.3 Timer/Counter .....	495

III.4	Bus Interface .....	497
III.5	Interrupt Control .....	500
III.6	Oscillation Circuit .....	502
III.7	Reset .....	502
III.8	Low Power Dissipation Mode .....	505
III.9	Software .....	507
III.10	Others .....	508
APPENDIX IV. THE DIFFERENCES BETWEEN HD63701X0 and HD6301X0 ...		509
APPENDIX V. PROGRAM DEVELOPMENT PROCEDURE AND SUPPORT SYSTEM .....		510

## Section 6

### HD6301Y0/HD6303Y/HD63701Y0 User's Manual

	Page
1. OVERVIEW .....	519
1.1 Features .....	519
1.2 Block Diagrams .....	520
1.4 Pin Description .....	523
2. INTERNAL ARCHITECTURE AND OPERATION .....	526
2.1 Operation Modes .....	526
2.2 Memory Map .....	527
2.3 Function Pin Description .....	528
2.4 Ports .....	531
2.5 RAM/Port 5 Control Register .....	543
2.6 Port 6 Control/Status Register .....	546
3. CPU FUNCTION .....	548
3.1 CPU Registers .....	548
3.2 Addressing Modes .....	549
3.3 Instruction Set .....	551
3.4 CPU Instruction Flow .....	557
3.5 Low Power Dissipation Modes .....	559
3.6 Trap Function .....	562
3.7 Reset .....	563
3.8 Interrupts .....	564
4. TIMER 1 .....	567
4.1 Free-Running Counter (FRC) .....	568
4.2 Output Compare Registers (OCR) .....	568

4.3	Input Compare Register (ICR).....	569
4.4	Timer Control/Status Register 1 (TCSR1).....	569
4.5	Timer Control/Status Register 2 (TCSR2).....	570
4.6	Timer Status Flags .....	572
4.7	Precautions on Cleaning the OCF .....	572
5.	TIMER 2.....	574
5.1	Timer 2 Upcounter (T2CNT) .....	574
5.2	Timer Constant Register (TCONR) .....	575
5.3	Timer Control/Status Register 3 (TCSR3).....	575
5.4	Timer Status Flags .....	576
5.5	Precaution for Toggle Pulse Function of HD6301Y0/HD6303Y/ HD63701Y0 Timer 2 .....	577
6.	SERIAL COMMUNICATIONS INTERFACE.....	578
6.1	Initialization .....	578
6.2	Asynchronous Mode .....	579
6.3	Clock Synchronous Mode .....	580
6.4	Transmit/Receive Control Status Register (TRCSR).....	582
6.5	Transmit Rate/Mode Control Register (RMCR).....	583
6.6	SCI Receiving Margin .....	586
6.7	SCI Status Flags .....	587
6.8	Precaution for Clock-Synchronous Serial Communication Interface.....	588
7.	HD63701Y0 Programming ROM (EPROM).....	589
7.1	Programming and Verification.....	590
7.2	Erasing (Window Package) .....	591
7.3	Characteristics and Applications.....	591
8.	APPLICATIONS .....	593
8.1	HD6301Y0 in Expanded Mode .....	593
8.2	HD6301Y0 in Single-Chip Mode .....	594
8.3	Timer Applications .....	594
8.4	SCI Applications .....	598
8.5	Lowering Operating Current .....	600
8.6	Memory Ready Application .....	603
8.7	Halt Application .....	604
8.8	$\overline{RD}$ , $\overline{WR}$ Application.....	605
8.9	LCD-II Interface Application .....	605
8.10	Oscillation Circuit Board Design .....	606

APPENDIX I. ELECTRICAL CHARACTERISTICS .....	608
APPENDIX II. INSTRUCTION EXECUTION CYCLES .....	630
APPENDIX III. QUESTIONS AND ANSWERS .....	636
III.1 Parallel Ports .....	636
III.2 Serial Ports .....	639
III.3 Timer/Counter .....	644
III.4 Bus Interface .....	647
III.5 Interrupt Control .....	650
III.6 Oscillation Circuit .....	653
III.7 Reset .....	653
III.8 Low Power Dissipation Mode .....	657
III.9 Software .....	660
III.10 Others .....	661
APPENDIX IV. THE DIFFERENCES BETWEEN HD63701Y0 and HD6301Y0 .....	663
APPENDIX V. PROGRAM DEVELOPMENT PROCEDURE AND SUPPORT SYSTEM .....	664
V.1 Overview .....	664
V.2 Single Chip Microcomputer ROM Ordering Procedure .....	666

## **Section 7 Software Application Notes**

**Page**

1. HOW TO USE APPLICATION NOTES .....	675
1.1 Formats .....	675
1.1.1 Specification Format (Format 1) .....	677
1.1.2 Description Format (Format 2) .....	683
1.1.3 Flowchart Format (Format 3) .....	686
1.1.4 Program Listing Format (Format 4) .....	688
1.2 How to Execute Programs .....	690
1.3 Symbols .....	692

## **PROGRAM APPLICATION EXAMPLES**

Program Application Table .....	693
---------------------------------	-----

## **MOVING DATA**

1. Filling Constant Values (Fill) .....	694
2. Moving Memory Blocks (Move) .....	698

3. Moving Strings (Moves) .....	703
BRANCHING FROM TABLE	
4. Branching From Table (CCASE) .....	708
HANDLING ASCII	
5. Converting ASCII Lowercase Into Uppercase (TPR) .....	714
6. Converting ASCII Into 1-Byte Hexadecimal (Nibble) .....	719
7. Converting 8-Bit Binary Data Into ASCII (COBYTE) .....	724
BIT MANIPULATION	
8. Counting Number of Logical "1" Bits In 8-Bit Data (HCNT) .....	729
9. Shifting 32-Bit Data (SHR) .....	733
COUNTER	
10. 4-Digit BCD Counter (DECNT) .....	738
COMPARISON	
11. Comparing 32-Bit Binary Data (CMP) .....	743
ARITHMETIC OPERATION	
12. Adding 32-Bit Binary Data (ADD) .....	749
13. Subtracting 32-Bit Binary Data (SUB) .....	755
14. Multiplying 16-Bit Binary Data (MUL) .....	761
15. Dividing 16-Bit Binary Data (DIV) .....	768
16. Adding 8-Digit BCD (ADDD) .....	774
17. Subtracting 8-Digit BCD (SUBD) .....	780
18. 16-Bit Square Root (SQRT) .....	786
CONVERTING BCD INTO HEXADECIMALS	
19. Converting 2-Byte Hexadecimals Into 5-Digit BCD (HEX) .....	791
20. Converting 5-Digit BCD Into 2-Byte Hexadecimals (BCD) .....	796
SORTING	
21. Sorting (SORT) .....	803
 <b>Section 8 Hardware Application Notes</b>	
<b>APPLICATION NOTES GUIDE</b> .....	<b>815</b>
1. Symbols .....	815
2. Application Example Configuration .....	817

3. 1st Section (Hardware) .....	819
4. 2nd Section (Software).....	823
5. 3rd Section (Program Module) .....	825
5.1 Specification .....	826
5.2 Description.....	832
5.3 Flowchart.....	836
6. 4th Section (Subroutine) .....	837
7. 5th Section (Program Listing) .....	840
8. Program Module Execution .....	845
 SYSTEM APPLICATION EXAMPLES .....	 848
System Application Examples .....	848
 I/O PORT APPLICATIONS .....	 849
1. HD61830 (LM200) Graphic Mode .....	849
2. Darlington Transistor Drive (LED Dynamic Display) .....	883
 TIMER APPLICATIONS .....	 896
3. Duty Control of Pulse Output and DA Conversion .....	896
4. Pulse Width Measurement .....	912
5. Input Pulse Count .....	922
6. 8 × 4 Key Matrix .....	932
 INTERRUPT APPLICATION.....	 947
7. A/D Converter (HA16613A) Control .....	947
 PARALLEL HANDSHAKE APPLICATIONS.....	 956
8. Standard Keyboard Interface.....	956
9. Centronics Interface.....	970
 SCI APPLICATIONS .....	 982
10. Data Transfer With Asynchronous SCI .....	982
11. Liquid Crystal Driven (HD61100A) Control.....	998
 EXTERNAL EXPANSION APPLICATIONS .....	 1010
12. External Expansion .....	1010
13. Slow Device Interface .....	1043
 LOW POWER DISSIPATION/FAIL-SAFE APPLICATIONS .....	 1057
14. Low Power Dissipation Mode.....	1057
15. HA1835P Control and Error Detection .....	1074

## Section 9

### C Language Programming Techniques

	Page
1. HOW TO USE APPLICATION NOTES .....	1105
1.1 Hardware Section .....	1107
1.1.1 Function .....	1107
1.1.2 Microcontroller Applications .....	1107
1.1.3 Circuit Diagram .....	1108
1.1.4 Memory Map .....	1109
1.1.5 Pin Functions .....	1110
1.1.6 Hardware Operation .....	1111
1.2 Software Section .....	1112
1.2.1 Program Module Configuration .....	1112
1.2.2 Program Module Functions .....	1113
1.2.3 Program Module Sample Application (Main Program) .....	1114
1.3 Program Module Section .....	1116
1.3.1 Page Heading .....	1117
1.3.2 Function .....	1117
1.3.3 Arguments .....	1117
1.3.4 Libraries Required for Program Execution .....	1118
1.3.5 Specifications .....	1119
1.3.6 Description .....	1120
1.3.7 PAD .....	1123
1.4 Subroutine Section .....	1124
1.4.1 Page Heading .....	1125
1.4.2 Function .....	1125
1.4.3 Basic Operation .....	1125
1.4.4 Program Modules That Use This Function .....	1126
1.4.5 PAD .....	1126
1.5 Program Listing Section .....	1127
1.5.1 Page Heading .....	1127
1.5.2 Function .....	1128
1.5.3 Output Object Listing of C Compiler .....	1131
1.5.4 Linkage Listing .....	1132
1.6 Program Module Use .....	1135



1.7	Pad Symbols Description	1137
1.8	Symbols	1138
2.	DARLINGTON TRANSISTOR DRIVE (LED DYNAMIC DISPLAY)	1139
2.1	Hardware Description	1139
2.1.1	Function	1139
2.1.2	Microcontroller Applications	1139
2.1.3	Circuit Diagram	1139
2.1.4	Pin Functions	1140
2.1.5	Hardware Operation	1141
2.2	Software Description	1142
2.2.1	Program Module Configuration	1142
2.2.2	Program Module Functions	1142
2.2.3	Program Module Sample Application (Main Program)	1143
2.3	Program Module Description	1144
2.4	Program Listing	1152
2.4.1	Main Program Listing	1152
2.4.2	C Source Listing	1153
2.4.3	Output Object Listing of C Compiler	1154
2.4.4	Linkage Listing	1157
3.	8 × 4 KEY MATRIX	1159
3.1	Hardware Description	1159
3.1.1	Function	1159
3.1.2	Microcontroller Applications	1159
3.1.3	Circuit Diagram	1160
3.1.4	Pin Functions	1161
3.1.5	Hardware Operation	1161
3.2	Software Description	1162
3.2.1	Program Module Configuration	1162
3.2.2	Program Module Functions	1162
3.2.3	Program Module Sample Application (Main Program)	1163
3.3	Program Module Description	1165
3.4	Program Listing	1177
3.4.1	Main Program Listing	1177

3.4.1	Main Program Listing .....	1177
3.4.2	C Source Listing .....	1178
3.4.3	Output Object Listing of C Compiler.....	1180
3.4.4	Linkage Listing.....	1185
4.	EXTERNAL EXPANSION .....	1187
4.1	Hardware Description .....	1187
4.1.1	Function.....	1187
4.1.2	Microcontroller Applications .....	1187
4.1.3	Circuit Diagram .....	1187
4.1.4	Memory Map .....	1188
4.1.5	Hardware Operation .....	1189
4.2	Software Description .....	1190
4.2.1	Program Module Configuration.....	1190
4.2.2	Program Module Functions.....	1190
4.2.3	Program Module Sample Application (Main Program).....	1192
4.3	Program Module Description.....	1194
4.4	Program Listing .....	1220
4.4.1	Main Program Listing .....	1220
4.4.2	C Source Listing .....	1222
4.4.3	Output Object Listing of C Compiler.....	1226
4.4.4	Linkage Listing.....	1232
APPENDIX A.		
	C Program and Assembly Program Comparison .....	1235

**Section 10—Appendix**

**HD6301V1/HD6303R Q & A—HD6301X0/HD6303X Oscillation Circuit  
Wide Temperature Range Specifications, -40°C to +85°C, (J Version)**

	<b>Page</b>
1. HD6301V1/HD6303R Q & A .....	1239
(a) Parallel Port	
(1) Process to Use a Port as an Output .....	1241
(b) Serial Port	
(1) Relation between Writing into the FRC and SCI Operation .....	1242
(2) Writing into the FRC during Serial Receive/Transmit .....	1243
(3) RDRF State When SCI Receiving.....	1244
(4) Serial I/O Operation.....	1245
(5) Serial I/O Register Read .....	1246
(6) Detection of the HD6301V1 Serial Start Bit .....	1247

(c)	Timer/Counter	
(1)	Free Running Counter Read .....	1249
(2)	Preset Method of the Free Running Counter .....	1250
(d)	BUS Interface	
(1)	Output of Address Strobe (AS) in the Multiplexed Mode .....	1252
(e)	Interrupt	
(1)	$\overline{IRQ1}$ Acceptance .....	1253
(2)	Timer Interrupt and External Interrupt .....	1254
(3)	$\overline{IRQ1}$ Interrupt and Other Interrupts .....	1255
(4)	CLI Instruction and Interrupt Operation .....	1257
(f)	Oscillator	
(1)	Relation between the External Clock (EXTAL Clock) and Enable Clock (E Clock) .....	1258
(g)	Reset	
(1)	Constants of the Reset Circuit .....	1259
(2)	Schmitt Trigger Circuit of $\overline{RES}$ .....	1260
(3)	I/O Port State on Resetting .....	1261
(4)	SCI (Pin 39) State on Resetting .....	1262
(5)	Port Output after Resetting .....	1263
(h)	Low Power Consumption	
(1)	Schmitt Trigger Circuit of $\overline{STBY}$ .....	1264
(2)	I/O Port State During Standby .....	1265
(3)	Return from Standby Mode .....	1266
(4)	Going into the Standby Mode .....	1267
(5)	Timing for the Standby Mode .....	1268
(i)	EPROM-on-Package	
(1)	Usage of EPROM Socket Pins for the HD63P01M (No. 1) .....	1269
(2)	Usage of EPROM Socket Pins for the HD63P01M (No. 2) .....	1270
(3)	Usage of EPROM Socket Pins for the HD63P01M (No. 3) .....	1271
(j)	Software	
(1)	Usage of Bit Manipulation Instructions .....	1272
(2)	Usage of Bit Manipulation Instructions to the Port .....	1274
(k)	Others	
(1)	RAM Access Disable during Program Execution .....	1275
2.	HD6301X0/HD6303X OSCILLATOR CIRCUIT .....	1276
3.	Wide Temperature Range Specifications $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$ (J Version) .....	1283
	HD6301V1, HD63A01V1, HD63B01V1 .....	1284
	HD6301X0, HD63A01X0, HD63B01X0 .....	1289
	HD6301Y0, HD63A01Y0, HD63B01Y0, HD63C01Y0 .....	1294
	HD6303R, HD63A03R, HD63B03R .....	1300
	HD6303X, HD63A03X, HD63B03X .....	1305
	HD6303Y, HD63A03Y, HD63B03Y, HD63C03Y .....	1310

# HD6301/HD6303 SERIES HANDBOOK

## Section One

- Quick Reference Guide  
and
- Package Reference  
Guide

# 8-BIT SINGLE-CHIP MICROCOMPUTER

## ■ CMOS 8-BIT SINGLE-CHIP MICROCOMPUTER HD6301 SERIES

Type No.		HD6301V1 HD63A01V1 HD63B01V1	HD6301X0 HD63A01X0 HD63B01X0			
LSI Characteristics	Bus Timing (MHz)	1.0 (HD6301V1) 1.5 (HD63A01V1) 2.0 (HD63B01V1)	1.0 (HD6301X0) 1.5 (HD63A01X0) 2.0 (HD63B01X0)			
	Supply Voltage (V)	5.0	5.0			
	Operating Temperature (°C)	0 ~ +70 <sup>*3, *4</sup>	0 ~ +70 <sup>*3</sup>			
	Package †	DP-40, FP-54, CG-40, CP-44, CP-52	DP-64S, FP-80, CP-68			
Functions	Memory	ROM (k byte)	4	4		
		RAM (byte)	128	192		
	I/O Port	I/O Port	29	29	53	24
		Input Port	—	—	—	8
		Output Port	—	—	—	21
	Interrupt	External	2	2	3	3
		Soft	2	2	2	2
		Timer	3	3	4	4
		Serial	1	1	1	1
	Timer	16-bit x 1 (Free running counter x 1 Output compare register x 1 Input capture register x 1)	16-bit x 1 (Free running counter x 1 Output compare register x 2 Input capture register x 1 8-bit x 1 (8-bit up counter x 1 Time constant register x 1)			
	SCI	Asynchronous	Asynchronous/Synchronous			
	External Memory Expansion	65k bytes	65k bytes			
Other Features	<ul style="list-style-type: none"> <li>•Error detection</li> <li>•Low power dissipation modes (sleep and standby)</li> </ul>	<ul style="list-style-type: none"> <li>•Error detection</li> <li>•Low power dissipation modes (sleep and standby)</li> <li>•Slow memory interface</li> <li>•Halt</li> </ul>				
EPROM on Chip Type		HD63701V0C HD637A01V0C HD637B01V0C	HD63701X0C HD637A01X0C HD637B01X0C			
EPROM on the Package Type		HD63P01M1	—			

\*1 Preliminary \*2 Under development \*3 Wide temperature range (-40 ~ +85°C) version is available.

\*4 Wide temperature range (-40 ~ +125°C) version is available.

† DP; Plastic DIP, FP; Plastic Flat Package, CG; Glass-sealed Ceramic Leadless Chip Carrier, CP; Plastic Leaded Chip Carrier (J-bend leads)

HD6301Y0 HD63A01Y0 HD63B01Y0 HD63C01Y0	HD6303R HD63A03R HD63B03R	HD6303X HD63A03X HD63B03X	HD6303Y HD63A03Y HD63B03Y HD63C03Y
1.0 (HD6301Y0) 1.5 (HD63A01Y0) 2.0 (HD63B01Y0) 3.0 (HD63C01Y0)	1.0 (HD6303R) 1.5 (HD63A03R) 2.0 (HD63B03R)	1.0 (HD6303X) 1.5 (HD63A03X) 2.0 (HD63B03X)	1.0 (HD6303Y) 1.5 (HD63A03Y) 2.0 (HD63B03Y) 3.0 (HD63C03Y)
5.0	5.0	5.0	5.0
0 ~ +70**	0 ~ +70**, **	0 ~ +70**	0 ~ +70**
DP-64S, FP-64, FP-64A, CP-68	DP-40, FP-54, CG-40, CP-52	DP-64S, FP-80, CP-68	DP-64S, FP-64, FP-64A, CP-68
16	—	—	—
256	128	192	256
53	13	24	24
48	13	16	24
—	—	8	—
5	—	—	—
3	2	3	3
2	2	2	2
4	3	4	4
1	1	1	1
16-bit x 1 (Free running counter x 1 Output compare register x 2 Input capture register x 1) 8-bit x 1 (8-bit up counter x 1 Time constant register x 1)	16-bit x 1 (Free running counter x 1 Output compare register x 1 Input capture register x 1)	16-bit x 1 (Free running counter x 1 Output compare register x 2 Input capture register x 1) 8-bit x 1 (8-bit up counter x 1 Time constant register x 1)	16-bit x 1 (Free running counter x 1 Output compare register x 2 Input capture register x 1) 8-bit x 1 (8-bit up counter x 1 Time constant register x 1)
Asynchronous/Synchronous	Asynchronous	Asynchronous/Synchronous	Asynchronous/Synchronous
65k bytes	65k bytes	65k bytes	65k bytes
<ul style="list-style-type: none"> <li>•Error detection</li> <li>•Low power dissipation modes (sleep and standby)</li> <li>•Slow memory interface</li> <li>•Halt</li> </ul>	<ul style="list-style-type: none"> <li>•Error detection</li> <li>•Low power dissipation modes (sleep and standby)</li> </ul>	<ul style="list-style-type: none"> <li>•Error detection</li> <li>•Low power dissipation modes (sleep and standby)</li> <li>•Slow memory interface</li> <li>•Halt</li> </ul>	<ul style="list-style-type: none"> <li>•Error detection</li> <li>•Low power dissipation modes (sleep and standby)</li> <li>•Slow memory interface</li> <li>•Halt</li> </ul>
HD63701Y0C HD637A01Y0C HD637B01Y0C	—	—	—
—	—	—	—

# PACKAGE REFERENCE GUIDE

Hitachi microcomputer devices include various types of package which meet a lot of requirements such as ever smaller, thinner and more versatile electric appliances. When selecting a package suitable for the customers' use, please refer to the following for Hitachi microcomputer packages.

multi-function types, applicable to each kind of mounting method. Also, plastic and ceramic materials are offered according to use.

Fig. 1 shows the package classification according to the mounting types on the Printed Circuit Board (PCB) and the materials.

## 1. Package Classification

There are pin insertion types, surface mounting types and

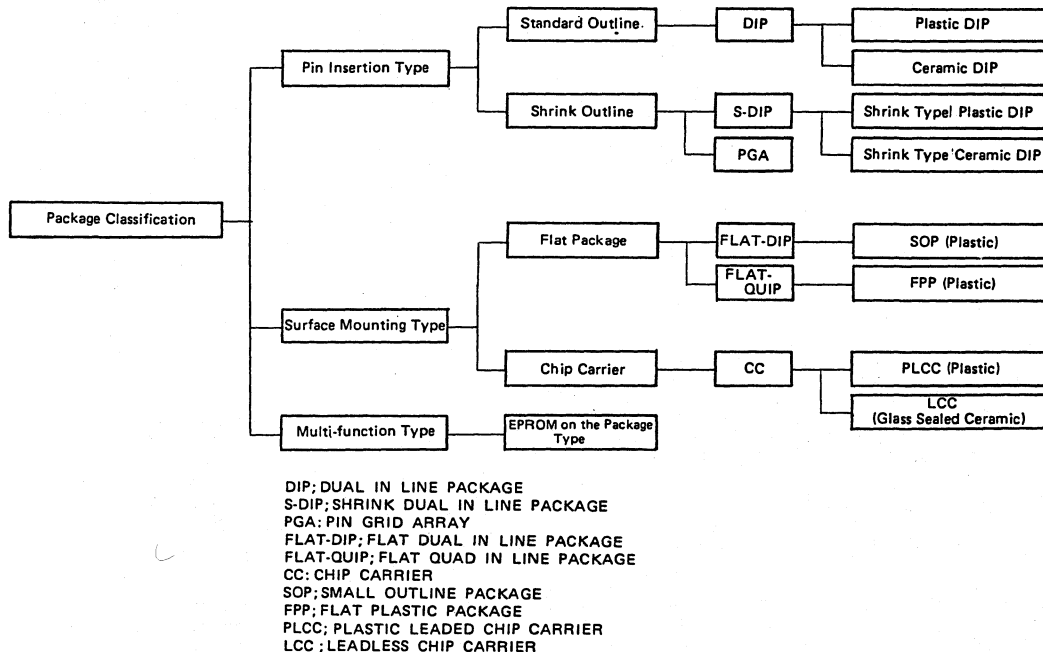


Fig. 1 Package Classification according to the Mounting Type on the Printed Circuit Board and the Materials.



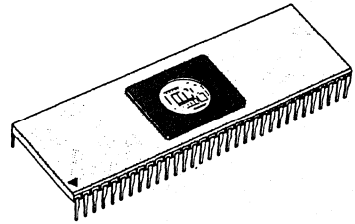
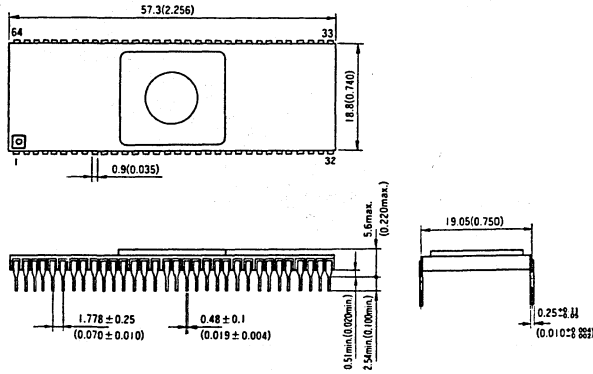


CERAMIC SHRINK TYPE

Unit : mm(inch)

Scale : 1/1

• DC-64S

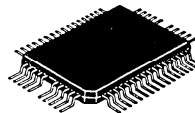
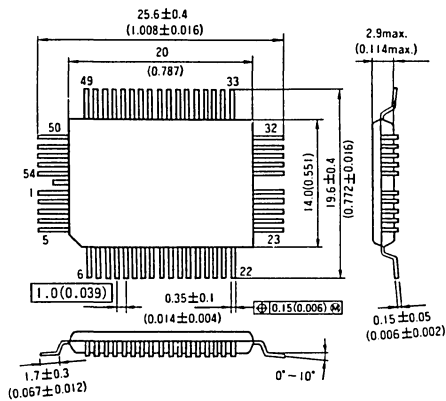


FLAT PACKAGE

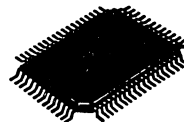
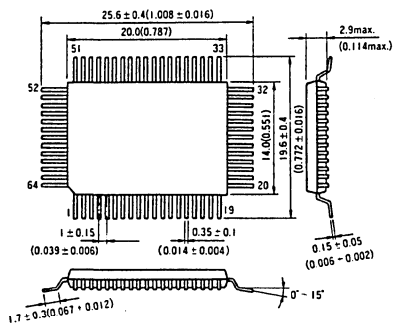
Unit : mm(inch)

Scale : 3/2

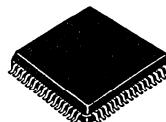
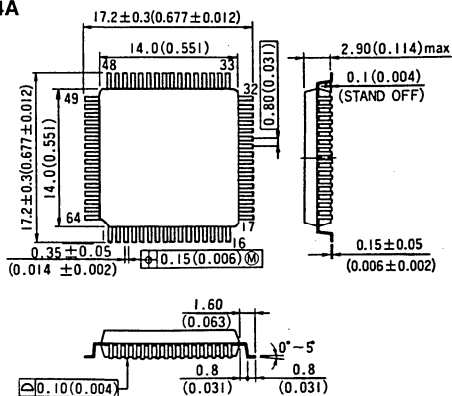
• FP-54



• FP-64

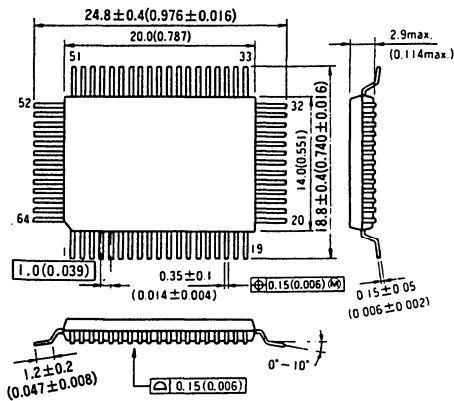


• FP-64A





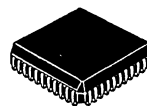
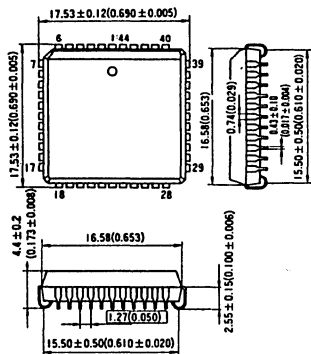
• FP-80B



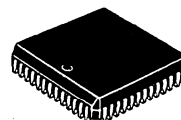
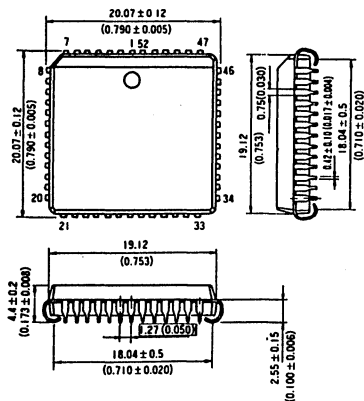
PLASTIC LEADED CHIP CARRIER

Unit : mm(inch)  
Scale : 3/2

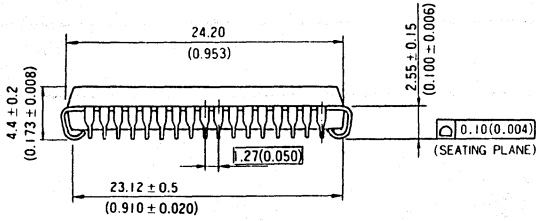
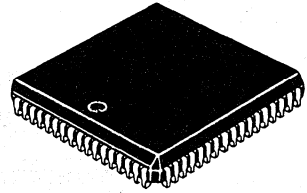
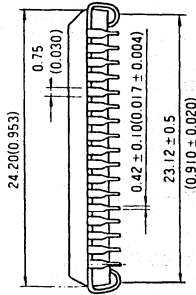
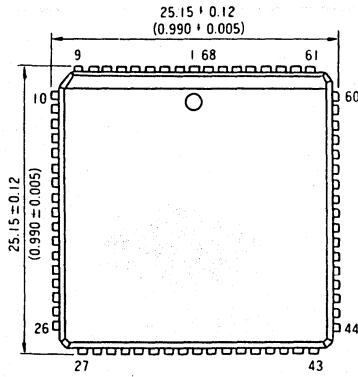
• CP-44



• CP-52



• CP-68

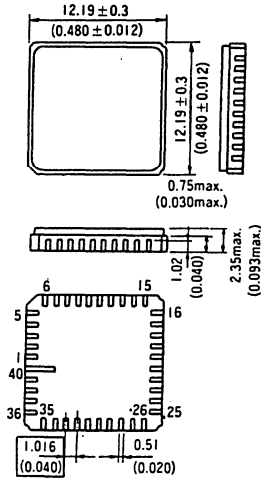


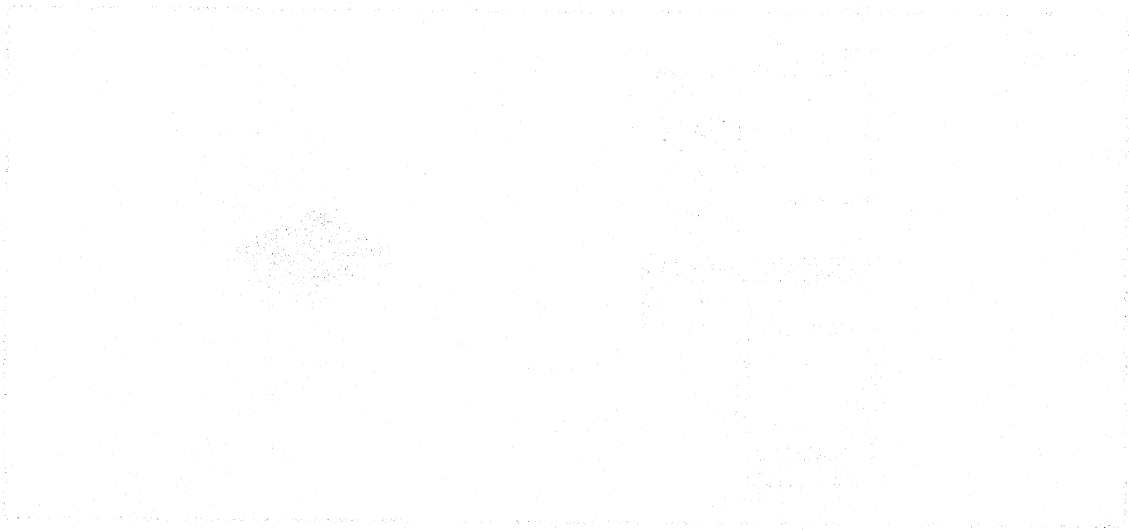
# LEADLESS CHIP CARRIER

Unit : mm(inch)  
Scale : 3/2

1

• CG-40





## Section Two

# Addressing Modes, CPU Architecture, and Instruction Set





**Section 2**  
**Addressing Modes, CPU Architecture, and Instruction Set**  
**Table of Contents**

	<b>Page</b>
1. ASSEMBLY LANGUAGE .....	17
1.1 Addressing Modes .....	17
1.2 CPU Registers .....	26
1.3 Instruction Set Details .....	28



## 1. ASSEMBLY LANGUAGE

### 1.1 Addressing Modes

The assembler determines the addressing mode by referencing the operator and operand fields. There are seven different addressing modes available.

- (1) Accumulator addressing
- (2) Implied addressing
- (3) Immediate addressing
- (4) Direct addressing
- (5) Extended addressing
- (6) Indexed addressing
- (7) Relative addressing

Before going into details about individual addressing modes, we explain the dual operand mode in which an instruction has two operands.

For eight instructions AIM, OIM, EIM, TIM, BCLR, BSET, BTGL and BTST, the operand field requires two operands (the first and second operands). The first operand includes the immediate data (constant) for AIM, OIM and TIM; and the bit number for bit operation for BCLR, BSET, BTGL and BTST. The second operand specifies a memory address in either indexed or direct addressing mode.

#### (1) Accumulator Addressing

Thirteen instructions allow Accumulator A or B as an operand. They are: ASL, ASR, CLR, COM, DEC, INC, LSR, NEG, PSH, PUL, ROL, ROR and TST. In this case, an **SP** or **HT** between the operator and the operand may be omitted. Each accumulator addressing instructions is converted into a one-byte machine code by the assembler.

Example:

Instruction code	Machine code (Hexadecimal)
ASL A or ASLA	48
ASR B or ASRB	57

#### (2) Implied Addressing

In the implied addressing mode, the instruction contained in the operator field permits the address for operation to be clear-cut. The operand is therefore unnecessary.

This implied addressing includes 31 instructions: ABA, ABX, ASLD, CBA, CLC, CLI, CLV, DAA, DES, DEX, INS, INX, LSRL, MUL, NOP, PSHX, PULX, RTI, RTS, SBA, SEC, SEI, SEV, SWI, TAB, TAP, TBA, TPA, TSX, TXS and WAI. Each instruction is converted into a one-byte machine code by the assembler.

(3) Immediate Addressing

There are 16 instructions that allow immediate addressing. They are: ADC, ADD, AND, BIT, CMP, CPX, EOR, LDA, LDS, LDX, ORA, SBC, SUB, LDD, ADDD and SUBD.

The operand field starts with #, followed by numerical data in decimal, hexadecimal, octal or binary, symbols (labels) that will take specific values during assembling, expressions and ASCII constants.

In any case, the assembler converts the immediate data (operand) into an unsigned 8-bit binary, or 16-bit binary for CPX, LDS, LDX, LDD, ADD and SUBD. The resulting immediate data range from 0 to 255, or 0 to 65535 for 16-bit operand instructions.

Example:

Statement	Machine code (Hexadecimal) Label = 100		
	Byte 1	Byte 2	Byte 3
LDA A #25	86	19	-
LDA A #LABEL	86	64	-
LDA A #LABEL + 25	86	7D	-
LDA A #'A	86	41	-
CPX #256	8C	01	00

In this case, the characters following "'" are converted into 7-bit ASCII data. The #'conversion is not generally used with CPX, LDS and LDX instructions. If it is used, however, the converted ASCII data is stored into byte 3. The assembler enables each immediate addressing instruction to be converted into 2 bytes in machine code (3 bytes in the case of CPX, LDS, LDX, LDD, ADDD and SUBD).

Figure 1-1-1 shows how data flows in immediate addressing mode.

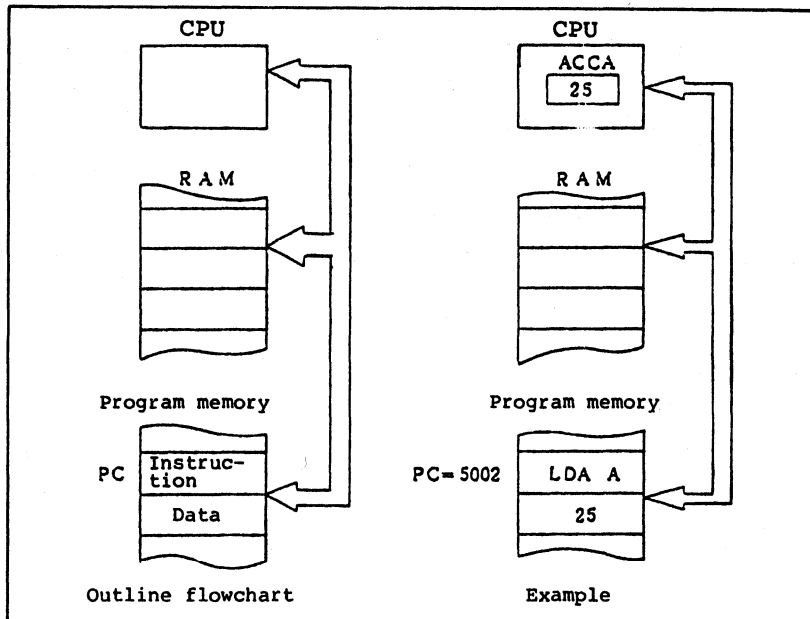


Fig. 1-1-1 Data Activity in Immediate Addressing Mode

(4) Direct Addressing and Extended Addressing

In direct addressing mode, the assembler converts the instruction into 2 bytes of machine code. The second byte, after conversion, includes an unsigned 8-bit binary address.

In extended addressing mode, the assembler converts the instruction into 3 bytes of machine code. The second byte includes the upper 8 bits of the address; and the third byte includes the lower 8 bits. Both of them are unsigned 8-bit in binary notation.

The assembler permits both direct addressing and extended addressing to be translated into absolute addresses.

The assembler automatically selects direct addressing if the address is within 0 - 255; and extended addressing if the address is greater than 255.

Example:

Statement	Machine code (Hexadecimal) Label address = 100		
	Byte 1	Byte 2	Byte 3
LDA A 100	96	64	-
LDA A LABEL	96	64	-
LDA A LABEL + 200	B6	01	2C

Figures 1-1-2 and 1-1-3 show how data flows in direct addressing and extended addressing modes, respectively.

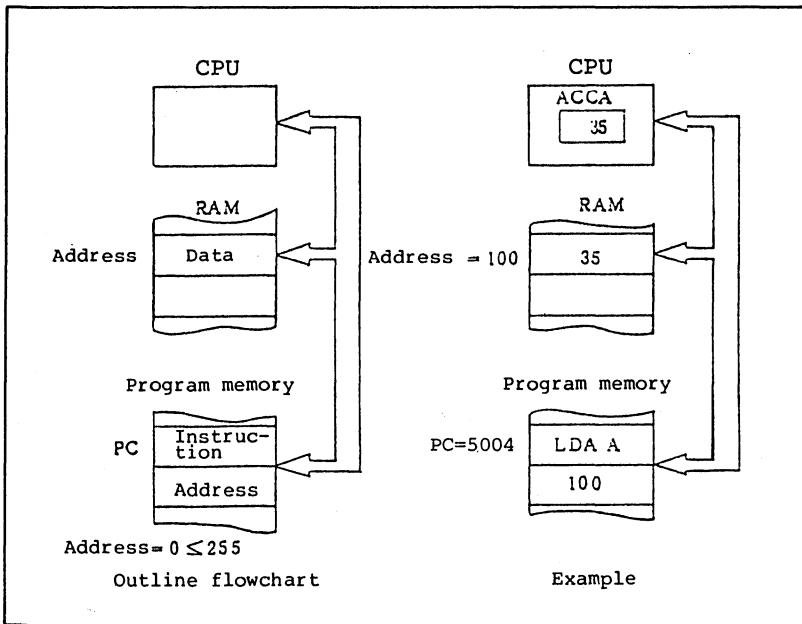


Fig. 1-1-2 Data Activity in Direct Addressing Mode



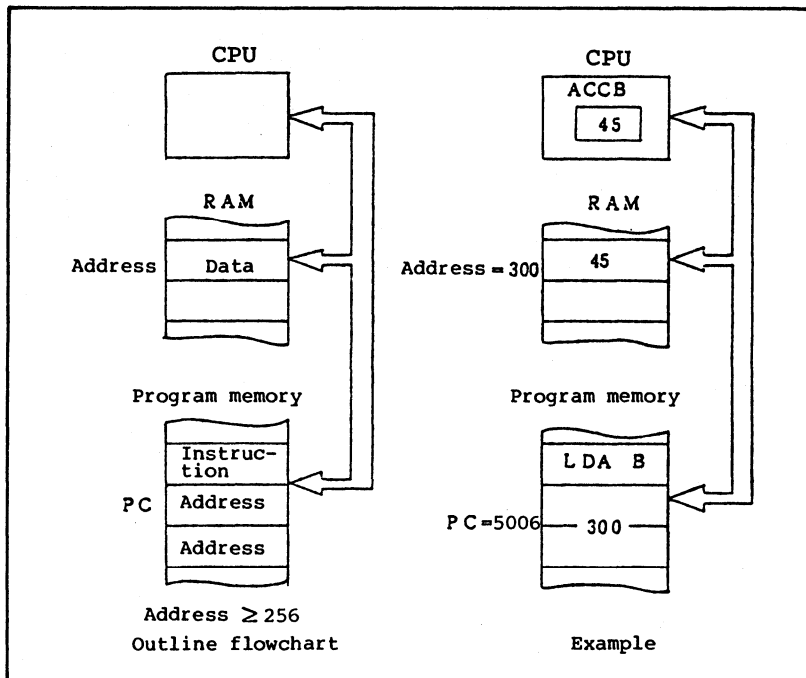


Fig. 1-1-3 Data Activity in Extended Addressing Mode

(5) Indexed Addressing

In Indexed addressing mode, the assembler converts the operand into an unsigned 8-bit displacement "Disp". The displacement "Disp" is added to the contents of the Index Register to determine the effective address M.

$$M = \text{Disp} + (X)$$

As other addressing modes, the operand may contain symbols (labels) and expression that are evaluated during assembling. They must range from 0 to 255.

Example:

Statement	Machine code (Hexadecimal) Label address = 100	
	Byte 1	Byte 2
LDA B X	E6	00
LDA B, X	E6	00
LDA B 5, X	E6	05
LDA B LABEL, X	E6	64
LDA B LABEL + 5, X	E6	69

Figure 1-1-4 shows how data flows in indexed addressing mode.

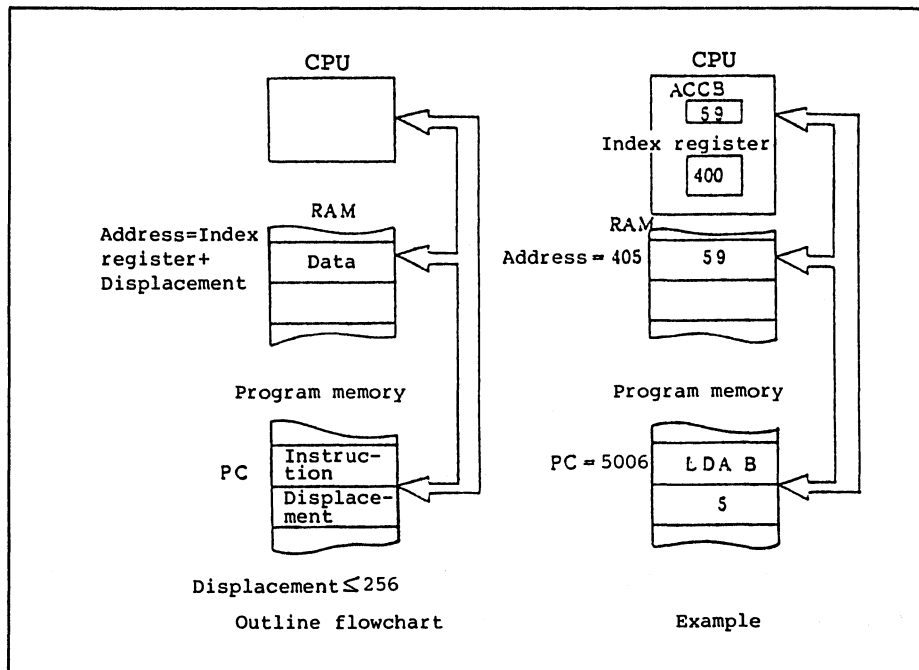


Fig. 1-1-4 Data Activity in Indexed Addressing Mode

## (6) Relative Addressing

This mode is limited to branch instructions.

A relative addressing instruction is converted into 2 bytes of machine code by the assembler. The second byte includes an 8-bit relative address (Rel., used as two's complement). On execution, the relative address (Rel.), the contents of the Program Counter (PC), and 2 are added to obtain the absolute address (D) of the branch destination as follows.

$$D = (PC) + 2 + \text{Rel.}$$

D : absolute address of branch destination

Rel : relative address

Therefore, the branch destination is within -126 and +129 from the OP-code address.

Example:

Statement	Machine code (Hexadecimal) Label address - (PC) - 2 = 100	
	Byte 1	Byte 2
BEQ *+17	27	0F
BEQ LABEL	27	64
BEQ LABEL - 105	27	FB

If, however, the branch destination is more than -126 to +129 away, JMP and JSR instructions can be used as shown below.

Example:

Statement	Machine code (Hexadecimal)		
	Byte 1	Byte 2	Byte 3
JMP 300	7E	01	2C
JSR 300	BD	01	2C

Figure 1-1-5 shows how data flows in relative addressing mode.

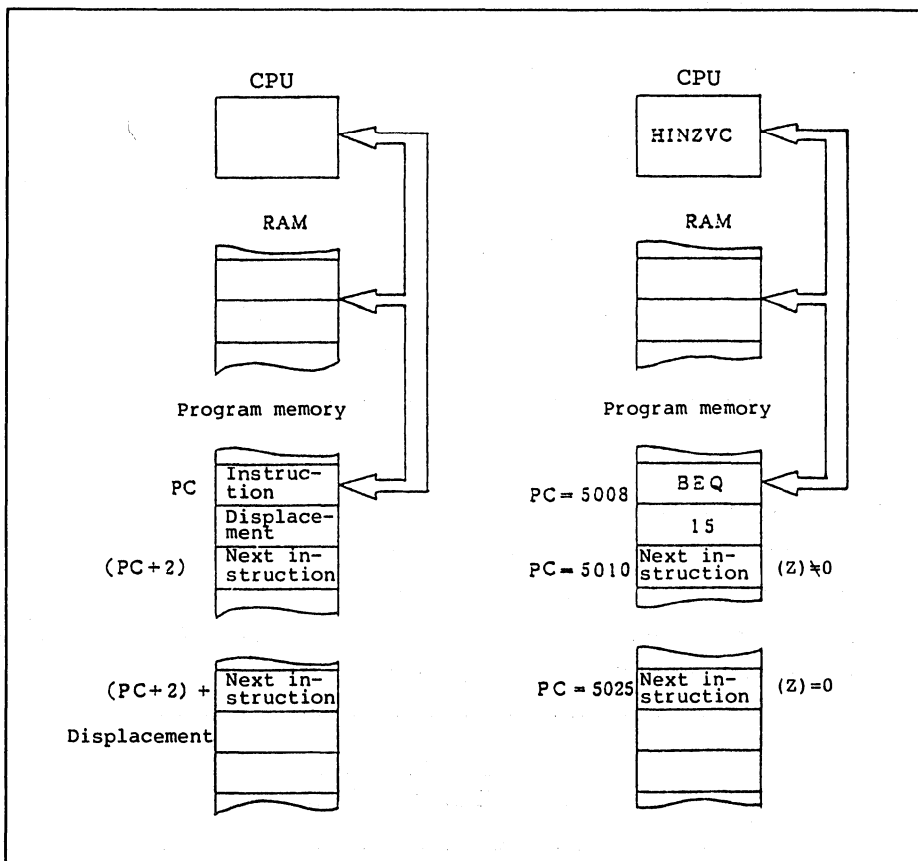


Fig. 1-1-5 Data Activity in Relative Addressing Mode

## 1.2 CPU Registers

The CPU has three 16-bit registers and three 8-bit registers. The register configuration of the CPU is shown in Fig. 1-2-1.

is shown in Fig. 1-2-1.

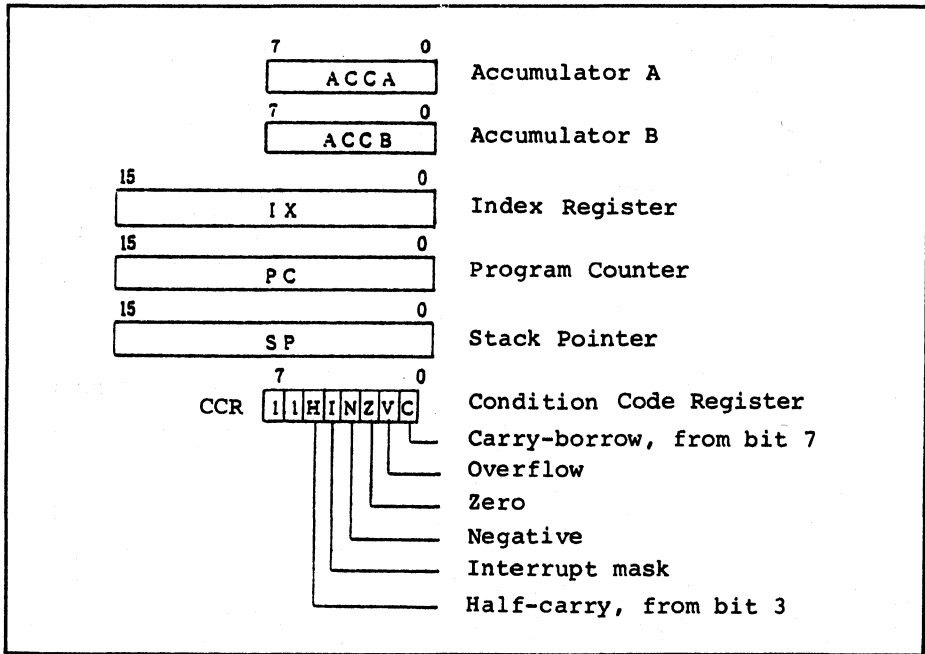


Fig. 1-2-1 CPU Registers

### (1) Accumulators (ACCA & ACCB)

The CPU has two 8-bit accumulators that store the result of arithmetic and logical operation.

If a double accumulator is specified, a pair of registers ACCA and ACCB can be functions as an 16-bit register.

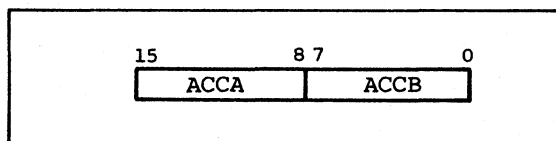


Fig. 1-2-2 ACCAB (Double Accumulator)

### (2) Index register (IX)

The index register is a 2-byte (16-bit) register that stores a 16-bit memory address used in indexed addressing mode or a 16-bit immediate data.

(3) Program counter (PC)

The program counter is a 2-byte (16-bit) register that indicates the address of the instruction being executed by the CPU. After the instruction execution, the program counter is automatically incremented, indicating the address of the next instruction.

(4) Stack pointer (SP)

The stack pointer is a 2-byte (16-bit) register that indicates the next available location in the memory pushdown/popup stacks. Any area of memory may serve as stacks; and random access (read/write) memory is generally used as stacks. In an application system which must hold data in stacks even when power supply is off, the stacks normally use battery-backed CMOS memory.

(5) Condition code register (CCR)

The condition code register indicates the result of arithmetic operation, etc. It consists of six bits: zero (Z), negative (N), overflow (V), carry-borrow from bit 7 (C), half-carry from bit 3 (H), and interrupt mask (I). These bits may be tested by a variety of conditional branch instructions, which is limited to relative addressing. It should be noted that the upper two bits of the condition code register cannot be used.

### 1.3 Instruction Set Details

Meanings of symbols and mnemonics:

#### (1) Operation symbols

( ) = Contents  
← = Direction of data transfer  
↑ = From stack  
↓ = To stack  
· = AND operation  
⊕ = OR operation  
⊕ = Exclusive-OR operation  
~ = NOT operation

#### (2) Registers within MPU

ACCA = Accumulator A  
ACCB = Accumulator B  
ACCX = Accumulator A or B  
ACCD = Double accumulator (ACCA + ACCB)  
CC = Condition-code register  
IX = Index register, 16 bits  
IXH = MSB 8 bits of index register  
IXL = LSB 8 bits of index register  
PC = Program counter, 16 bits  
PCH = MSB 8 bits of program counter  
PCL = LSB 8 bits of program counter  
SP = Stack pointer, 16 bits  
SPH = MSB 8 bits of stack pointer  
SPL = LSB 8 bits of stack pointer

#### (3) Memory and addressing modes

M = Memory address  
MH = MSB 8 bits of memory address  
ML = LSB 8 bits of memory address  
M+1 = Memory address of memory address M + 1  
Imm = Immediate data

ImmH = MSB 8 bits of immediate value  
 ImmL = LSB 8 bits of immediate value  
 Disp = Displacement = M - (IX)  
 Rel = Relative addressing = Branch destination absolute  
       address - (PC) - 2  
 ACCX = Accumulator addressing  
 IMMED = Immediate addressing  
 DIRECT = Direct addressing  
 INDEX = Index addressing  
 EXTEND = Extended addressing  
 RELATIVE = Relative addressing  
 IMPL = Implied addressing

(4) Meaning of bits 0 through 5 of condition-code register

C = Carry and borrow; bit 0  
 V = Overflow for 2's complement; bit 1  
 Z = Zero; bit 2  
 N = Negative; bit 3  
 I = Interrupt mask; bit 4  
 H = Half carry from bit 3 to bit 4; bit 5

(5) Bit status before run of instruction

An = Bit n of ACCA (n = 7, 6, 5, ..., 0)  
 Bn = Bit n of ACCB (n = 7, 6, 5, ..., 0)  
 Dn = Bit n of double accumulator (n = 15, 14, 13, ..., 0)  
 IXn = Bit n of IX (n = 15, 14, 13, ..., 0)  
 IXHn = Bit n of IXH (n = 7, 6, 5, ..., 0)  
 IXLn = Bit n of IXL (n = 7, 6, 5, ..., 0)  
 Mn = Bit n of M (n = 15, 14, 13, ..., 0)  
 SPHn = Bit n of SPH (n = 7, 6, 5, ..., 0)  
 SPLn = Bit n of SPL (n = 7, 6, 5, ..., 0)  
 Xn = Bit n of ACCX (n = 7, 6, 5, ..., 0)

(6) Bit status after run of instruction

Rn = Bit n of result (n = 15, 14, 13, ..., 0)  
 RHn = Bit n of resulting high-order byte  
       (n = 7, 6, 5, ..., 0)  
 RLn = Bit n of resulting low-order byte  
       (n = 7, 6, 5, ..., 0)



Add accumulator B to accumulator A

ABA

Category	Function
Arithmetic operation (Two operands)	$ACCA \leftarrow (ACCA) + (ACCB)$  Adds the contents of ACCB to the contents of ACCA, and stores the result into the ACCA.

Effects on the condition codes

$H = A3 \cdot B3 \oplus B3 \cdot \overline{R3} \oplus \overline{R3} \cdot A3$ : Set if a carry from bit 3 is generated; cleared otherwise.  
 I : Not affected.  
 $N = R7$ : Set if the result's MSB is "1"; cleared otherwise.  
 $Z = \overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise.  
 $V = A7 \cdot B7 \cdot \overline{R7} \oplus \overline{A7} \cdot \overline{B7} \cdot R7$ : Set if the result overflows; cleared otherwise.  
 $C = A7 \cdot B7 \oplus B7 \cdot \overline{R7} \oplus \overline{R7} \cdot A7$ : Set if a carry from the MSB is generated cleared otherwise.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	ABA		1B			1	1

Add accumulator B to index register

ABX

Category	Function
Arithmetic operation	$IX \leftarrow (IX) + (ACCB)$  Adds the unsigned contents of ACCB to the contents of the IX taking into account a carry from the low-order byte of the IX, and stores the result into the IX.

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	ABX		3A			1	1

Category	Function						
Arithmetic operation (Two operands)	$ACCX \leftarrow (ACCX) + (M) + (C)$ Adds the contents of ACCX, memory M, and carry bit C, and stores the result into the ACCX.						
Effects on the condition codes							
<p>H = <math>X3 \cdot M3 \overline{O} M3 \cdot \overline{R3} \overline{O} R3 \cdot X3</math>: Set if a carry from bit 3 is generated; cleared otherwise.</p> <p>I : Not affected.</p> <p>N = <math>R7</math>: Set if the result's MSB is "1"; cleared otherwise.</p> <p>Z = <math>\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}</math>: Set if the result is zero; cleared otherwise.</p> <p>V = <math>X7 \cdot M7 \cdot \overline{R7} \overline{O} X7 \cdot \overline{M7} \cdot R7</math>: Set if the result overflows; cleared otherwise.</p> <p>C = <math>X7 \cdot M7 \overline{O} M7 \cdot \overline{R7} \overline{O} R7 \cdot X7</math>: Set if a carry from the MSB is generated; cleared otherwise.</p>							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMMED	ADC A	#Imm	89	Imm		2	2
DIRECT	ADC A	M	99	M		2	3
EXTND	ADC A	M	B9	MH	ML	3	4
INDEX	ADC A	Disp,X	A9	Disp		2	4
IMMED	ADC B	#Imm	C9	Imm		2	2
DIRECT	ADC B	M	D9	M		2	3
EXTND	ADC B	M	F9	MH	ML	3	4
INDEX	ADC B	Disp,X	E9	Disp		2	4

Category	Function
Arithmetic operation (Two operand)	$ACCX \leftarrow (ACCX) + (M)$ Adds the contents of memory M to the contents of ACCX and stores the result into the ACCX.

## Effects on the condition codes

$H = X3 \cdot M3 \oplus M3 \cdot \overline{R3} \oplus \overline{R3} \cdot X3$ : Set if a carry from bit 3 is generated; cleared otherwise.  
 I : Not affected.  
 N = R7: Set if the result's MSB is "1"; cleared otherwise.  
 $Z = \overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise.  
 $V = X7 \cdot M7 \cdot \overline{R7} \oplus \overline{X7} \cdot \overline{M7} \cdot R7$ : Set if the result overflows; cleared otherwise.  
 $C = X7 \cdot M7 \oplus M7 \cdot \overline{R7} \oplus \overline{R7} \cdot X7$ : Set if a carry from the MSB 16 generated; cleared otherwise.

## Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMMED	ADD A	#Imm	8B	Imm		2	2
DIRECT	ADD A	M	9B	M		2	3
EXTND	ADD A	M	BB	MH	ML	3	4
INDEX	ADD A	Disp,X	AB	Disp		2	4
IMMED	ADD B	#Imm	CB	Imm		2	2
DIRECT	ADD B	M	DB	M		2	3
EXTND	ADD B	M	FB	MH	ML	3	4
INDEX	ADD B	Disp,X	EB	Disp		2	4

Category	Function
Arithmetic operation	<p>ACCD ← (ACCD) + (M:M+1)</p> <p>Adds the contents of memories M and M+1 to the contents of ACCD, and stores the result into the ACCD.</p>

Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N : N=R15; Set if the result's MSB is "1"; cleared otherwise.  
 Z =  $\overline{R15} \cdot \overline{R14} \cdot \overline{R13} \cdot \dots \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise.  
 V =  $AB15 \cdot M15 \cdot \overline{R15} \oplus \overline{AB15} \cdot \overline{M15} \cdot R15$ : Set if the result overflows; cleared otherwise.  
 C =  $AB15 \cdot M15 \oplus M15 \cdot \overline{R15} \oplus \overline{R15} \cdot AB15$ : Set if a carry from the MSB is generated; cleared otherwise.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMMED	ADDD	#Imm	C3	ImmH	ImmL	3	3
DIRECT	ADDD	M	D3	M		2	4
EXTND	ADDD	M	F3	MH	ML	3	5
INDEX	ADDD	Disp,X	E3	Disp		2	5

Category	Function
Logic operation	$M \leftarrow IM \cdot (M)$ <p>ANDs the immediate data and the contents of the memory M, and stores the result into the meory M.</p>

## Effects on the condition codes

H : Not affected.

I : Not affected.

N = R7: Set if the result's MSB is "1"; cleared otherwise.

Z =  $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise.

V = 0: Cleared.

C : Not affected.

## Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
DIRECT	AIM	#Imm,M	71	Imm	M	3	6
INDEX	AIM	#Imm,Disp,X	61	Imm	Disp	3	7

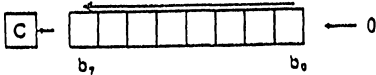
Category	Function
Logic operation	$ACCX \leftarrow (ACCX) \cdot (M)$  ANDs the contents of ACCX and the memory M, and stores the results into the ACCX.

Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N = R7: Set if the result's MSB is "1"; cleared otherwise.  
 Z =  $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise.  
 V = 0: Cleared.  
 C : Not affected.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMMED	AND A	#Imm	84	Imm		2	2
DIRECT	AND A	M	94	M		2	3
EXTND	AND A	M	B4	MH	ML	3	4
INDEX	AND A	Disp,X	A4	Disp		2	4
IMMED	AND B	#Imm	C4	Imm		2	2
DIRECT	AND B	M	D4	M		2	3
EXTND	AND B	M	F4	MH	ML	3	4
INDEX	AND B	Disp,X	E4	Disp		2	4

Category	Function
Shift & rotation	 <p data-bbox="284 336 1065 439">Shifts ACCX or memory M by one bit to the left. Bit 0 takes "0". The original value of bit 7 moves into the carry bit C.</p>

Effects on the condition codes

H : Not affected.

I : Not affected.

N = R7: Set if the result's MSB is "1"; cleared otherwise.

Z =  $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise.

V = N⊕C: Set if either N=1 and C=0 or N=0 and C=1 after the shift operation; cleared otherwise.

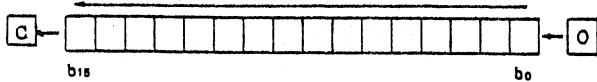
Note: The N and C are those obtained after operation.

C = M7: Set if the MSB of ACCX or the memory is "1" before the shift operation; cleared otherwise.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
ACCX	ASL A		48			1	1
ACCX	ASL B		58			1	1
EXTND	ASL	M	78	MH	ML	3	6
INDEX	ASL	Disp,X	68	Disp		2	6



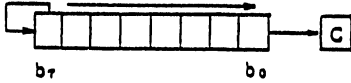
Category	Function
Shift & rotation	 <p data-bbox="385 354 1159 418">Shifts ACCD by one bit to the left. Bit 0 takes "0". The original value of bit 15 moves into carry bit C.</p>

Effects on the condition codes

H : Not affected.  
 I : "  
 N = R15: Set if the result's MSB is "1"; cleared otherwise.  
 Z =  $\overline{R15} \cdot \overline{R14} \cdot \overline{R13} \cdot \dots \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise.  
 V = N⊕C: Set if either N=1 and C=0 or N=0 and C=1 after the shift operation; cleared otherwise.  
 Note: The N and C are those obtained after operation.  
 C = AB15: Set if the MSB of ACCAB is "1" before the shift operation; cleared otherwise.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	ASLD		05			1	1

Category	Function						
Shift & rotation	<div style="text-align: center;">  </div> <p data-bbox="288 352 1083 453">Shifts the contents of ACCX or memory M by one bit to the right. Bit 7 is not affected. The original value of bit 0 moves into carry flag.</p>						
<b>Effects on the condition codes</b>							
<p data-bbox="113 716 355 743"><b>H</b> : Not affected.</p> <p data-bbox="113 753 355 779"><b>I</b> : Not affected.</p> <p data-bbox="113 789 963 815"><b>N = R7</b>: Set if the result's MSB is "1"; cleared otherwise.</p> <p data-bbox="113 826 1040 887"><b>X = <math>\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}</math></b>: Set if the result is zero; cleared otherwise.</p> <p data-bbox="113 897 982 958"><b>V = N⊕C</b>: Set if either N=1 and C=0 or N=0 and C=1 after the shift operation; cleared otherwise.</p> <p data-bbox="168 968 950 994" style="padding-left: 20px;">Note: The N and C are those obtained after operation.</p> <p data-bbox="113 1005 954 1065"><b>C = M0</b>: Set if the LSB is "1" before the shift operation; cleared otherwise.</p>							
<b>Addressing modes and CPU cycles</b>							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
ACCX	ASR A		47			1	1
ACCX	ASR B		57			1	1
EXTND	ASR M		77	MH	ML	3	6
INDEX	ASR	Disp,X	67	Disp		2	6

Branch if Carry Clear

BCC

Category	Function
Conditional branch	<p>PC ← (PC) + 0002 + Rel If (C) = 0</p> <p>Tests the state of carry bit C and causes a branch if C = 0.</p>

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
RELATIVE	BCC	Rel	24	Rel		2	3

Category	Function						
Logic operation	$M_i \leftarrow 0$ Clears bit $i$ ( $i=0$ to $7$ ) of the memory $M$ . Other bits are not affected. * The machine code of this instruction is the same as AIM.						
Effects on the condition codes							
H : Not affected. I : Not affected. N = R7: Set if the result's MSB is "1"; cleared otherwise. Z = $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise. V = 0: Cleared. C : Not affected.							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
DIRECT	BCLR	0,M	71	FE	M	3	6
"	BCLR	1,M	"	FD	"	"	"
"	BCLR	2,M	"	FB	"	"	"
"	BCLR	3,M	"	F7	"	"	"
"	BCLR	4,M	"	EF	"	"	"
"	BCLR	5,M	"	DF	"	"	"
"	BCLR	6,M	"	BF	"	"	"
"	BCLR	7,M	"	7F	"	"	"
INDEX	BCLR	0,Disp,X	61	FE	Disp	3	7
"	BCLR	1,Disp,X	"	FD	"	"	"
"	BCLR	2,Disp,X	"	FB	"	"	"
"	BCLR	3,Disp,X	"	F7	"	"	"
"	BCLR	4,Disp,X	"	EF	"	"	"
"	BCLR	5,Disp,X	"	DF	"	"	"
"	BCLR	6,Disp,X	"	BF	"	"	"
"	BCLR	7,Disp,X	"	7F	"	"	"

Branch if Carry Set

BCS

Category	Function
Conditional branch	<p>PC ← (PC) + 0002 + Rel If (C) = 1</p> <p>Tests the state of carry bit C and causes a branch if C = 1.</p>

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
RELATIVE	BCS	Rel	25	Rel		2	3

Branch if Equal

BEQ

Category	Function
Conditional branch	$PC \leftarrow (PC) + 0002 + Rel$ If $(Z) = 1$ Tests the state of bit Z and causes a branch if Z=1.

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
RELATIVE	BEQ	Rel	27	Rel		2	3

2

Branch if Greater than or Equal to zero

BGE

Category	Function
Conditional branch	<p> <math>PC \leftarrow (PC) + 0002 + Rel</math> if <math>(N) \oplus (V) = 0</math>                      that is, <math>(ACCX) \geq (M)</math>; in the case of two's complement                 </p> <p>                     Branches if <math>N=1</math> and <math>V=1</math> or if <math>N=0</math> and <math>V=0</math>.                      When a BGE instruction is executed immediately after an instruction such as CBA, CMP, SBA or SUB has been executed, a branch occurs if the minuend (ACCX) as a two's complement is greater than, or equal to, the subtracter (M) as a two's complement.                 </p>

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
RELATIVE	BGE	Rel	2C	Rel		2	3

Category	Function
Condi- tional branch	<p> <math>PC \leftarrow (PC) + 0002 + Rel</math> If <math>(Z) \oplus [(N) \oplus (V)] = 0</math>                      that is, <math>(ACCX) &gt; (M)</math>; in the case of two's complement                 </p> <p>                     Branches if <math>Z=0</math> and <math>N \&amp; V=1</math> or if <math>Z=0</math> and <math>N \&amp; V=0</math>.                      When a BGT instruction is executed immediately                      after an instruction such as CBA, CMP, SBA or SUB                      has been executed, a branch occurs if the minuend                      (ACCX) as a two's complement is greater than the                      subtractor (M) as a two's complement.                 </p>

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
RELATIVE	BGT	Rel	2E	Rel		2	3



Category	Function						
Condi- tional branch	<p>PC ← (PC) + 0002 + Rel If (C)⊙(Z) = 0 That is, (ACCX) &gt; (M); in the case of unsigned binary</p> <p>Branches if C=0 and Z=0. When a BHI instruction is executed immediately after an instruction such as CBA, CMP, SBA or SUB has been executed, a branch occurs if the minuend (ACCX) as a unsigned binary is greater than the subtracter (M) as a unsigned binary.</p>						
Effects on the condition codes							
<p>H : Not affected. I : " N : " Z : " V : " C : "</p>							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
RELATIVE	BHI	Rel	22	Rel		2	3

Category	Function
Logic operation	<p>(ACCX) · (M)</p> <p>Performs the logical "AND" operation between the contents of ACCX and those of memory (M). Then, the condition codes reflect the result. The contents of the ACCX and those of memory M remain unchanged.</p>

Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N = R7: Set if the result's MSB is "1"; cleared otherwise.  
 Z =  $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if all bits of the result are zeros; cleared otherwise.  
 V = 0: Cleared.  
 C : Not affected.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMMED	BIT A	#Imm	85	Imm		2	2
DIRECT	BIT A	M	95	M		2	3
EXTND	BIT A	M	B5	MH	ML	3	4
INDEX	BIT A	Disp,X	A5	Disp		2	4
IMMED	BIT B	#Imm	C5	Imm		2	2
DIRECT	BIT B	M	D5	M		2	3
EXTND	BIT B	M	F5	MH	ML	3	4
INDEX	BIT B	Disp, X	E5	Disp		2	4

Branch.if Less than or Equal to zero

BLE

Category	Function
Condi- tional branch	<p>PC <math>\leftarrow</math> (PC) + 0002 + Rel If (Z) <math>\oplus</math> [(N)<math>\oplus</math>(V)] = 1 That is, (ACCX) <math>\leq</math> (M); in the case of two's complement</p> <p>Branches if Z=1 or N=1 &amp; V=0 or N=0 &amp; V=1.</p> <p>When a BLE instruction is executed immediately after an instruction such as CBA, CMP, SBA or SUB has been executed, a branch occurs if the minuend (ACCX) as a two's complement is smaller than, or equal to, the subtracter (M) as a two's complement.</p>

Effects on the condition codes

H : Not affected.  
I : "  
N : "  
Z : "  
V : "  
C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
RELATIVE	BLE	Rel	2F	Rel		2	3

Category	Function
Conditional branch	<p> <math>PC \leftarrow (PC) + 0002 + Rel</math> If <math>(C) \oplus (Z) = 1</math>                          That is, <math>(ACCX) \leq (M)</math>; in the case of unsigned binary                     </p> <p>                         Branches if <math>C=1</math> or <math>Z=1</math>.                     </p> <p>                         When a BLS instruction is executed immediately after an instruction such as CBA, CMP, SBA or SUB has been executed, a branch occurs if the minuend (ACCX) as a unsigned binary is smaller than, or equal to, the subtracter (M) as a unsigned binary.                     </p>

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
RELATIVE	BLS	Rel	23	Rel		2	3

Branch if Less Than zero

BLT

Category	Function
Conditional branch	<p>PC ← (PC) + 0002 + Rel If (N) ⊕ (V) = 1 That is, (ACCX) &lt; (M); in the case of two's complement</p> <p>Branches if N=1 &amp; V=0 or N=0 &amp; V=1.</p> <p>When a BLT instruction is executed immediately after an instruction such as CBA, CMP, SBA or SUB has been executed, a branch occurs if the minuend (ACCX) as a two's complement is smaller than the substracter (M) as a two's complement.</p>

Effects on the condition codes

H : Not affected.  
I : "  
N : "  
Z : "  
V : "  
C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
RELATIVE	BLT	Rel	2D	Rel		2	3

Branch if Minus

BMI

Category	Function
Condi- tional branch	$PC \leftarrow (PC) + 0002 + Rel \quad \text{If } (N) = 1$  Tests the state of negative bit N and causes a branch if N=1.

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
RELATIVE	BNI	Rel	2B	Rel		2	3

2

Branch if Not Equal

BNE

Category	Function
Conditional branch	<p>PC ← (PC) + 0002 + Rel If (Z) = 0</p> <p>Tests the state of zero bit Z and causes a branch if Z=0.</p>

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
RELATIVE	BNE	Rel	26	Rel		2	3



Category	Function
Conditional branch	$PC \leftarrow (PC) + 0002 + Rel \quad \text{If } (N) = 0$ <p>Tests the state of negative bit N and causes a branch if N=0.</p>

Effects on the condition codes

H : Not affected  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
RELATIVE	BPL	Rel	2A	Rel		2	3



Category	Function						
Unconditional branch & jump	$PC \leftarrow (PC) + 0002 + Rel$ <p>Branches unconditionally to the address resulting from the above expression. "Rel" is the relative address stored as a two's complement in the second byte of the machine code of a branch instruction.</p>						
Effects on the condition codes							
H : Not affected. I : " N : " Z : " V : " C : "							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
RELATIVE	BRA	Rel	20	Rel		2	3

Category	Function
Unconditional branch & jump	<p>PC ← (PC) + 0002</p> <p>A two-byte 3-cycle instruction that is equivalent to NOP instruction. As a feature of the HD6301, this instruction provides a function opposite to the BRA instruction.</p> <p>Note: The second byte of the instruction code takes an arbitrary value (0 to \$FF) at which a branch may occur.</p>

Effects on the condition codes	
H : Not affected. I : " N : " Z : " V : " C : "	

Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
RELATIVE	BRN	Rel	21	Rel		2	3

Category	Function						
Logic operation	$M_i \leftarrow 1$ Sets bit $i$ of the memory. ( $i = 0$ to $7$ ) Other bits are not affected. * The machine code of this instruction is the same as OIM.						
Effects on the condition codes							
H : Not affected. I : Not affected. N = R7: Set if the result's MSB is "1"; cleared otherwise. Z = $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise. V = 0: Cleared. C : Not affected.							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
DIRECT	BSET	0,M	72	01	M	3	6
"	BSET	1,M	"	02	"	"	"
"	BSET	2,M	"	04	"	"	"
"	BSET	3,M	"	08	"	"	"
"	BSET	4,M	"	10	"	"	"
"	BSET	5,M	"	20	"	"	"
"	BSET	6,M	"	40	"	"	"
"	BSET	7,M	"	80	"	"	"
INDEX	BSET	0,Disp,X	62	01	Disp	3	7
"	BSET	1,Disp,X	"	02	"	"	"
"	BSET	2,Disp,X	"	04	"	"	"
"	BSET	3,Disp,X	"	08	"	"	"
"	BSET	4,Disp,X	"	10	"	"	"
"	BSET	5,Disp,X	"	20	"	"	"
"	BSET	6,Disp,X	"	40	"	"	"
"	BSET	7,Disp,X	"	80	"	"	"

Category	Function	
Subroutine control	PC ← (PC) + 0002	1. Increments the PC by two.
	↓ (PCL)	2. Saves the low-order byte of the program counter into the stack.
	SP ← (SP) - 0001	3. Decrements the SP by one.
	↓ (PCH)	4. Saves the high-order byte of the PC into the stack.
	SP ← (SP) - 0001	5. Decrements the SP by one.
	PC ← (PC) + Rel	6. Branches to the address indicated by the program.

## Effects on the condition codes

H : Not affected.

I : "

N : "

Z : "

V : "

C : "

## Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
RELATIVE	BSR	Rel	8D	Rel		2	5

Category	Function						
Logic operation	$M_i \leftarrow \overline{M_i}$ Inverts bit $i$ of the memory $M$ . ( $i = 0$ to $7$ ) Other bits are not affected. NOTE) BTGL has the same instruction code as EIM.						
Effects on the condition codes							
H : Not affected. I : Not affected. N : R7: Set if the result's MSB is "1"; cleared otherwise. Z = $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise. V = 0: Cleared. C : Not affected.							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
DIRECT	BTGL	0,M	75	01	M	3	6
"	BTGL	1,M	"	02	"	"	"
"	BTGL	2,M	"	04	"	"	"
"	BTGL	3,M	"	08	"	"	"
"	BTGL	4,M	"	10	"	"	"
"	BTGL	5,M	"	20	"	"	"
"	BTGL	6,M	"	40	"	"	"
"	BTGL	7,M	"	80	"	"	"
INDEX	BTGL	0,Disp,X	65	01	Disp	3	7
"	BTGL	1,Disp,X	"	02	"	"	"
"	BTGL	2,Disp,X	"	04	"	"	"
"	BTGL	3,Disp,X	"	08	"	"	"
"	BTGL	4,Disp,X	"	10	"	"	"
"	BTGL	5,Disp,X	"	20	"	"	"
"	BTGL	6,Disp,X	"	40	"	"	"
"	BTGL	7,Disp,X	"	80	"	"	"

Category	Function						
Logic operation	$M_i \cdot 1$ Performs the logical "AND" operation between bit $i$ ( $i=0$ to $7$ ) of the memory $M$ and "1". Then, the condition codes reflect the result. (NOTE) BTST has the same instruction code as TIM.						
Effects on the condition codes							
H : Not affected. I : Not affected. N = R7: Set if the result's MSB is "1"; cleared otherwise. Z = $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise. V = 0: Cleared. C : Not affected.							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
DIRECT	BTST	0,M	7B	01	M	3	4
"	BTST	1,M	"	02	"	"	"
"	BTST	2,M	"	04	"	"	"
"	BTST	3,M	"	08	"	"	"
"	BTST	4,M	"	10	"	"	"
"	BTST	5,M	"	20	"	"	"
"	BTST	6,M	"	40	"	"	"
"	BTST	7,M	"	80	"	"	"
INDEX	BTST	0,Disp,X	6B	01	Disp	3	5
"	BTST	1,Disp,X	"	02	"	"	"
"	BTST	2,Disp,X	"	04	"	"	"
"	BTST	3,Disp,X	"	08	"	"	"
"	BTST	4,Disp,X	"	10	"	"	"
"	BTST	5,Disp,X	"	20	"	"	"
"	BTST	6,Disp,X	"	40	"	"	"
"	BTST	7,Disp,X	"	80	"	"	"

Category	Function						
Condi- tional branch	$PC \leftarrow (PC) + 0002 + Rel \quad \text{If } (V) = 0$ <p>Tests the state of overflow bit V and causes a branch if V = 0.</p>						
Effects on the condition codes							
<p>H : Not affected.            I : "            N : "            Z : "            V : "            C : "</p>							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
RELATIVE	BVC	Rel	28	Rel		2	3

Category	Function
Condi- tional branch	$PC \leftarrow (PC) + 0002 + Rel \quad \text{If } (V) = 1$  Tests the state of overflow bit V and causes a branch if $V = 1$ .

## Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

## Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
RELATIVE	BVS	Rel	29	Rel		2	3



Compare Accumulators

CBA

Category	Function
Compare & test	<p>(ACCA) - (ACCB)</p> <p>Compares the contents of ACCA to those of ACCB and sets the condition codes according to the result.</p> <p>Used for a conditional branch in arithmetic or logical operation. Both operands are not affected.</p>

Effects on the condition codes

H : Not affected.

I : Not affected.

N = R7: Set if the MSB of the result is "1"; cleared otherwise.

Z =  $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise.

V =  $A7 \cdot \overline{B7} \cdot \overline{R7} \oplus \overline{A7} \cdot B7 \cdot R7$ : Set if the result overflows; cleared otherwise.

C =  $\overline{A7} \cdot B7 \oplus B7 \cdot R7 \oplus R7 \cdot \overline{A7}$ : Set if a borrow is generated; cleared otherwise.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	CBA		11			1	1

Clear Carry

CLC

Category	Function
Bit control	<p>Bit C <math>\leftarrow</math> 0</p> <p>Clears carry bit C.</p>

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C = 0 : Cleared.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	CLC		0C			1	1

Clear Interrupt mask

CLI

Category	Function
Bit control	<p>Bit I ← 0</p> <p>Clears the interrupt mask bit I of the condition code. When an interrupt occurs in response to an interrupt request from a peripheral, this instruction enables the microprocessor to receive the interrupt request.</p>

Effects on the condition codes

H : Not affected.  
 I = 0 : Cleared.  
 N : Not affected.  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	CLI		0E			1	1

Category	Function
Arithmetic operation (One operand)	$ACCX \leftarrow 00$ or $M \leftarrow 00$ Clears the contents of ACCX or those of memory M to zero.

## Effects on the condition codes

H : Not affected.  
 I : "  
 N=0 : Cleared.  
 Z=1 : Set.  
 V=0 : Cleared.  
 C=0 : Cleared.

## Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
ACCX	CLR A		4F			1	1
ACCX	CLR B		5F			1	1
EXTND	CLR M		7F	MH	ML	3	5
INDEX	CLR	Disp,X	6F	Disp		2	5

Clear two's complement overflow bit

CLV

Category	Function
Bit control	Bit V ← 0  Clears the overflow bit V of the condition code.

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V = 0 : Cleared.  
 C : Not affected.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	CLV		0A			1	1

Category	Function
Compare & Test	<p>(ACCX) - (M)</p> <p>Compares the contents of ACCX to those of memory M and changes the condition codes according to the result. The contents of the condition codes may be referenced by the following conditional branch instruction.</p> <p>Both operands are not affected.</p>

## Effects on the condition codes

H : Not affected.

I : Not affected.

N = R7: Set if the MSB of the result is "1"; cleared otherwise.

Z =  $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise.

V =  $X7 \cdot \overline{M7} \cdot \overline{R7} \oplus \overline{X7} \cdot M7 \cdot R7$ : Set if the result overflows; cleared otherwise.

C =  $\overline{X7} \cdot M7 \oplus M7 \cdot R7 \oplus R7 \cdot \overline{X7}$ : Set if the absolute value of the memory is greater than those of the accumulator; cleared otherwise.

## Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMMED	CMP A	#Imm	8l	Imm		2	2
DIRECT	CMP A	M	9l	M		2	3
EXTND	CMP A	M	B1	MH	ML	3	4
INDEX	CMP A	Disp,X	A1	Disp		2	4
IMMED	CMP B	#Imm	C1	Imm		2	2
DIRECT	CMP B	M	D1	M		2	3
EXTND	CMP B	M	F1	MH	ML	3	4
INDEX	CMP B	Disp,X	E1	Disp		2	4

Category	Function						
Logic operation	$ACCX \leftarrow \sim(ACCX) = FF - (ACCX)$ or $M \leftarrow \sim(M) = FF - (M)$  Takes one's complement of each bit in ACCX or memory M.						
Effects on the condition codes							
H : Not affected. I : Not affected. N = R7: Set if the result's MSB is "1"; cleared otherwise. Z = $\overline{R7 \cdot R6 \cdot R5 \cdot R4 \cdot R3 \cdot R2 \cdot R1 \cdot R0}$ : Set if the result is zero; cleared otherwise. V = 0: Cleared. C = 1: Set.							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
ACCX	COM A		43			1	1
ACCX	COM B		53			1	1
EXTND	COM M		73	MH	ML	3	6
INDEX	COM	Disp,X	63	Disp		2	6

Category	Function						
Index register control	(IX) - (M : M + 1)  Compares the contents of the IX to those of memories M and M+1.						
Effects on the condition codes							
H : Not affected. I : Not affected. N = R15: Set if the MSB of the result is "1"; cleared otherwise. Z = ( $\overline{R15} \cdot \overline{R14} \cdot \overline{R13} \cdot \overline{R12} \cdot \overline{R11} \cdot \overline{R10} \cdot \overline{R9} \cdot \overline{R8} \cdot \overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ ) : Set if the result is "0"; cleared otherwise. V = $\overline{IX15} \cdot \overline{M15} \cdot \overline{R15} \oplus \overline{IX15} \cdot M15 \cdot R15$ : Set if the result overflows; and cleared otherwise. C = $\overline{IX15} \cdot M15 \oplus M15 \cdot R15 \oplus R15 \cdot \overline{IX15}$ : Set if the absolute value of the memory is greater than that of the index register; cleared otherwise.							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMMED	CPX	#Imm	8C	ImmH	ImmL	3	3
DIRECT	CPX	M	9C	M		2	4
EXTND	CPX	M	BC	MH	ML	3	5
INDEX	CPX	Disp,X	AC	Disp		2	5



Category	Function									
Bit C before DAA	0	0	0	0	0	0	1	1	1	Adds the hexadecimal data, 00, 06, 60 and 66, to ACCA according to the table.  For BCD (binary-coded decimal) addition by an instruction such as ABA, ADD or ADC, DAA executes this function if the result is in bits C and H of ACCA.
High-order 4 bits (Bits 4 - 7)	0-9	0-8	0-9	A-F	9-F	A-F	0-2	0-2	0-3	
Initial bit H (Half carry)	0	0	1	0	0	1	0	0	1	
Low-order 4 bits (Bits 0 - 3)	0-9	A-F	0-3	0-9	A-F	0-3	0-9	A-F	0-3	
Hex. data added to ACCX by DAA	00	06	06	60	66	66	60	66	66	
Bit c after DAA	0	0	0	1	1	1	1	1	1	

Effects on the condition codes

- H : Not affected.
- I : Not affected.
- N : R7: Set if the MSB of the result is "1"; cleared otherwise.
- Z =  $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ ; Set if the result is zero; cleared otherwise.
- V : Not affected.
- C : Set or cleared as shown in the above table.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	DAA		19			1	2

Category	Function
Arithmetic operation	$ACCX \leftarrow (ACCX) - 01$ or $M \leftarrow (M) - 01$ <p>Subtracts 1 from the contents of ACCX or those of memory M. Bits N, Z and V are set according to the result. Bit C is not affected.</p>

Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N = R7: Set if the MSB of the result is "1"; cleared otherwise.  
 $Z = \overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise.  
 $V = X7 \cdot \overline{X6} \cdot \overline{X5} \cdot \overline{X4} \cdot \overline{X3} \cdot \overline{X2} \cdot \overline{X1} \cdot \overline{X0} = \overline{R7} \cdot R6 \cdot R5 \cdot R4 \cdot R3 \cdot R2 \cdot R1 \cdot R0$   
 : Set if the result overflows; cleared otherwise. An overflow occurs if the contents of ACCX or those of the memory before operation are 80.  
 C : Not affected.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
ACCX	DEC A		4A			1	1
ACCX	DEC B		5A			1	1
EXTND	DEC M		7A	MH	ML	3	6
INDEX	DEC Disp,X		6A	Disp		2	6

DEcrement Stack pointer

DES

Category	Function
Stack pointer control	$SP \leftarrow (SP) - 0001$ <p>Subtracts 1 from the SP.</p>

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	DES		34			1	1

Category	Function
Index register control	$IX \leftarrow (IX) - 0001$ Subtracts 1 from the IX. Bit Z is set or reset according to the result.

Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N : Not affected.  
 $Z = (\overline{RH7} \cdot \overline{RH6} \cdot \overline{RH5} \cdot \overline{RH4} \cdot \overline{RH3} \cdot \overline{RH2} \cdot \overline{RH1} \cdot \overline{RH0}) \cdot (\overline{RL7} \cdot \overline{RL6} \cdot \overline{RL5} \cdot \overline{RL4} \cdot \overline{RL3} \cdot \overline{RL2} \cdot \overline{RL1} \cdot \overline{RL0})$ : Set if the result is zero; cleared otherwise.  
 V : Not affected.  
 C : Not affected.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	DEX		09			1	1

Category	Function						
Logic operation	$M \leftarrow IM \oplus (M)$  Performs the logical exclusive "OR" operation between the immediate data and the contents of memory M, and stores the result into the memory M.						
Effects on the condition codes							
H : Not affected. I : Not affected. N = M7: Set if the M's MSB is "1"; cleared otherwise. Z = $\overline{M7} \cdot \overline{M6} \cdot \overline{M5} \cdot \overline{M4} \cdot \overline{M3} \cdot \overline{M2} \cdot \overline{M1} \cdot \overline{M0}$ : Set if the contents of M is zero, cleared otherwise. V = 0: Cleared. C : Not affected.							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
DIRECT	EIM	#Imm,M	75	Imm	M	3	6
INDEX	EIM	#Imm,Disp,X	65	Imm	Disp	3	7

Category	Function
Logic operation	$\text{ACCX} \leftarrow (\text{ACCX}) \oplus (M)$ <p>Performs the logical exclusive "OR" operation between the contents of ACCX and those of memory M, and stores the result into the ACCX.</p>

## Effects on the condition codes

H : Not affected.

I : Not affected.

N = R7: Set if the MSB of the result is "1"; cleared otherwise.

Z =  $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise.

V = 0: Cleared.

C : Not affected.

## Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMMED	EOR A	#Imm	88	Imm		2	2
DIRECT	EOR A	M	98	M		2	3
EXTND	EOR A	M	B8	MH	ML	3	4
INDEX	EOR A	Disp,X	A8	Disp		2	4
IMMED	EOR B	#Imm	C8	Imm		2	2
DIRECT	EOR B	M	D8	M		2	3
EXTND	EOR B	M	F8	MH	ML	3	4
INDEX	EOR B	Disp,X	E8	Disp		2	4

Category	Function
Arithmetic operation	<p> <math>ACCX \leftarrow (ACCX) + 01</math> or  <math>M \leftarrow (M) + 01</math> </p> <p>                     Adds 1 to the contents of ACCX or those of memory M. Bits N, Z and V are set according to the result. Bit C is not affected.                 </p>

Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N = R7: Set if the MSB of the result is "1"; cleared otherwise.  
 Z =  $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise.  
 V =  $\overline{X7} \cdot X6 \cdot X5 \cdot X4 \cdot X3 \cdot X2 \cdot X1 \cdot X0 = R7 \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$   
 : Set if the result overflows; cleared otherwise. An overflow occurs if the contents of ACCX or those of the memory before operation are 7F.  
 C : Not affected.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
ACCX	INC A		4C			1	1
ACCX	INC B		5C			1	1
EXTND	INC M		7C	MH	ML	3	6
INDEX	INC	Disp,X	6C	Disp		2	6

Category	Function
Stack pointer control	$SP \leftarrow (SP) + 0001$ Adds 1 to the SP.

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	INS		31			1	1



Category	Function						
Index register control	$IX \leftarrow (IX) + 0001$ Adds 1 to the IX. Only bit Z is set or reset according to the result.						
Effects on the condition codes.							
H : Not affected. I : Not affected. N : Not affected. $Z = (\overline{RH7} \cdot \overline{RH6} \cdot \overline{RH5} \cdot \overline{RH4} \cdot \overline{RH3} \cdot \overline{RH2} \cdot \overline{RH1} \cdot \overline{RH0}) \cdot (\overline{RL7} \cdot \overline{RL6} \cdot \overline{RL5} \cdot \overline{RL4} \cdot \overline{RL3} \cdot \overline{RL2} \cdot \overline{RL1} \cdot \overline{RL0})$ : Set if the result is zero; cleared otherwise. V : Not affected. C : Not affected.							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	INX		08			1	1

Category	Function
Unconditional branch & jump	PC ← address  Branches to the instruction at the specified address. The branch destination is computed by using extended addressing or indexed addressing modes.

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
EXTND	JMP	M	7E	MH	ML	3	3
INDEX	JMP	Disp,X	6E	Disp		2	3

Category	Function	
Subroutine control	PC ← (PC)+0003 (EXTND) or PC ← (PC)+0002 (INDEX) ↓ (PCL) SP ← (SP)-0001 ↓ (PCH) SP ← (SP)-0001 PC ← numeric address	Increments the program counter by two or three according to the addressing mode, saves it in the 2-byte stack, and updates the stack pointer. Then branches to the specified address. The branch destination is computed by using extended addressing or indexed addressing.

## Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

## Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
EXTND	JSR	M	BD	MH	ML	3	6
INDEX	JSR	Disp,X	AD	Disp		2	5
DIRECT	JSR	M	9D	M		2	5

Category	Function
Transfer	$ACCX \leftarrow (M)$  Loads the contents of memory M into the ACCX.

## Effects on the condition codes

H : Not affected.

I : Not affected.

N = R7: Set if the MSB of the result is "1"; cleared otherwise.

Z =  $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise.

V = 0: Cleared.

C : Not affected.

## Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMMED	LDA A	#Imm	86	Imm		2	2
DIRECT	LDA A	M	96	M		2	3
EXTND	LDA A	M	B6	MH	ML	3	4
INDEX	LDA A	Disp,X	A6	Disp		2	4
IMMED	LDA B	#Imm	C6	Imm		2	2
DIRECT	LDA B	M	D6	M		2	3
EXTND	LDA B	M	F6	MH	ML	3	4
INDEX	LDA B	Disp,X	E6	Disp		2	4

Category	Function
Load & store	<p>ACCD ← (M:M+1)</p> <p>Loads the 2-byte contents of memories M and M+1 into ACCD.</p>

Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N = R15: Set if the result's MSB is "1"; cleared otherwise.  
 Z =  $\overline{R15} \cdot \overline{R14} \cdot \overline{R13} \dots \overline{R0}$ : Set if the result is zero; cleared otherwise.  
 V = 0: Cleared.  
 C : Not affected.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMMED	LDD	#Imm	CC	ImmH	ImmL	3	3
DIRECT	LDD	M	DC	M		2	4
EXTND	LDD	M	FC	MH	ML	3	5
INDEX	LDD	Disp,X	EC	Disp		2	5

Category	Function
Stack pointer control	<p>SPH ← (M)                      SPL ← (M+1)</p> <p>Loads the contents of memory M into the upper byte of the SP.</p> <p>Then, loads the contents of memory M+1 (which results when memory address M is incremented by one) into the Lower byte of the SP.</p>

Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N = RH7: Set if the MSB of the SP is "1"; cleared otherwise.  
 Z =  $(\overline{RH7} \cdot \overline{RH6} \cdot \overline{RH5} \cdot \overline{RH4} \cdot \overline{RH3} \cdot \overline{RH2} \cdot \overline{RH1} \cdot \overline{RH0}) \cdot (\overline{RL7} \cdot \overline{RL6} \cdot \overline{RL5} \cdot \overline{RL4} \cdot \overline{RL3} \cdot \overline{RL2} \cdot \overline{RL1} \cdot \overline{RL0})$ : Set if the SP contents is zero after the load; cleared otherwise.  
 V = 0: cleared.  
 C : Not affected.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMMED	LDS	#Imm	8E	ImmH	ImmL	3	3
DIRECT	LDS	M	9E	M		2	4
EXTND	LDS	M	BE	MH	ML	3	5
INDEX	LDS	Disp,X	AE	Disp		2	5

Category	Function
Index register control	<p>IXH ← (M) IXL ← (M+1)</p> <p>Loads the contents of memory M into the upper byte of the IX. Then, loads the contents of memory M+1 into the lower byte of the IX.</p>

Effects on the condition codes

H : Not affected.

I : Not affected.

N = RH7: Set if the MSB of the IX is "1"; cleared otherwise.

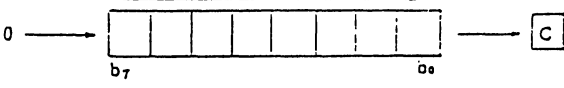
Z =  $(\overline{RH7} \cdot \overline{RH6} \cdot \overline{RH5} \cdot \overline{RH4} \cdot \overline{RH3} \cdot \overline{RH2} \cdot \overline{RH1} \cdot \overline{RH0}) \cdot (\overline{RL7} \cdot \overline{RL6} \cdot \overline{RL5} \cdot \overline{RL4} \cdot \overline{RL3} \cdot \overline{RL2} \cdot \overline{RL1} \cdot \overline{RL0})$ : Set if the IX contents is zero after the load; cleared otherwise.

V = 0: Cleared.

C : Not affected.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMMED	LDX	#Imm	CE	ImmH	ImmL	3	3
DIRECT	LDX	M	DE	M		2	4
EXTND	LDX	M	FE	MH	ML	3	5
INDEX	LDX	Disp,X	EE	Disp		2	5

Category	Function
Shift & rotation	<div style="text-align: center;">  </div> <p>Shifts the contents of ACCX or memory M by one bit to the right.                      Bit 7 takes 0.                      The bit C is loaded from the LSB of ACCX or memory M.</p>

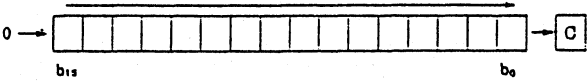
Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N = 0: Cleared.  
 Z =  $\overline{R15} \cdot \overline{R14} \cdot \overline{R13} \dots \overline{R0}$ ; Set if the result is zero; cleared otherwise.  
 V = N⊕C: Set if either N=1 and C=0 or N=0 and C=1; cleared otherwise.  
 Note: The N and C are those obtained after operation.  
 C = AB0: Set if the LSB of ACCX or M is a 1 before the instruction is executed; cleared otherwise.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
ACCX	LSR A		44			1	1
ACCX	LSR B		54			1	1
EXTND	LSR	M	74	MH	ML	3	6
INDEX	LSR	Disp,X	64	Disp		2	6



Category	Function
Shift & rotation	 <p data-bbox="383 340 1134 439">Shifts the contents of ACCD by one bit to the right. Bit 15 takes 0. The bit C is loaded from the LSB of the ACCD.</p>

Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N = 0: Cleared.  
 Z =  $R_{15} \cdot R_{14} \cdot R_{13} \dots \dots \cdot \overline{R_0}$ ; : Set if the result is zero; cleared otherwise.  
 V =  $N \oplus C$ : Set if either  $N=1$  and  $C=0$  or  $N=0$  and  $C=1$ ; cleared otherwise.  
 Note: The N and C are those obtained after operation.  
 C =  $A_{B0}$ : Set if the LSB of ACCD is "1" before the instruction is executed; cleared otherwise.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	LSRD		04			1	1

Category	Function
Arithmetic operation	<p>ACCD ← (ACCA) * (ACCB)</p> <p>Multiplies the contents of ACCA by those of ACCB, and stores the resulting unsigned 16 bits into ACCD. The highest-order byte of the result is stored into the ACCA.</p>

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C = R7: Set if the result's bit 7 is "1"; cleared otherwise.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	MUL		3D			1	7

2

Category	Function						
Arithmetic operation	<p>ACCX <math>\leftarrow</math> <math>-(ACCX) = 00-(ACCX)</math> or  M <math>\leftarrow</math> <math>-(M) = 00-(M)</math></p> <p>Takes two's complement of the contents of ACCX or memory M, and stores the result into ACCX or memory M. No change is caused if the contents of ACCX or memory M is \$80(-128).</p>						
Effects on the condition codes							
<p>H : Not affected.</p> <p>I : Not affected.</p> <p>N = R7: Set if the result's MSB is "1"; cleared otherwise.</p> <p>Z = <math>\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}</math>: Set if the result is zero; cleared otherwise.</p> <p>V = <math>R7 \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}</math>: Set if the result overflows; cleared otherwise. The bit is set only when the contents of ACCX or M is \$80.</p> <p>C = <math>R7 \oplus R6 \oplus R5 \oplus R4 \oplus R3 \oplus R2 \oplus R1 \oplus R0</math>: Set if a borrow is generated cleared otherwise. The bit is set only when the contents of ACCX or M is not zero.</p>							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
ACCX	NEG A		40			1	1
ACCX	NEG B		50			1	1
EXTND	NEG	M	70	MH	ML	3	6
INDEX	NEG	Disp,X	60	Disp		2	6

No Operation

NOP

2

Category	Function
Unconditional branch & jump	Updates the program counter only and has no effect on other registers.

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	NOP		01			1	1

Category	Function						
Logical operation	$M \leftarrow IM \odot (M)$ <p>Ors the immediate data and the contents of memory M, and stores the result into the memory M.</p>						
Effects on the condition codes							
<p>H : Not affected.  I : Not affected.  N = R7" Set if the result's MSB is "1"; cleared otherwise.  Z = <math>\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}</math>: Set if the result is zero; cleared otherwise.  V = 0: Cleared.  C : Not affected.</p>							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
DIRECT	OIM	#Imm,M	72	Imm	M	3	6
INDEX	OIM	#Imm,Disp,X	62	Imm	Disp	3	7

inclusive OR

ORA

Category	Function
Logical operation	$ACCX \leftarrow (ACCX) \oplus (M)$ Performs logical OR between the contents of ACCX and the contents of memory M, and stores the result into the ACCX.

Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N = R7: Set if the result's MSB is "1"; cleared if not.  
 Z =  $\overline{R7 \cdot R6 \cdot R5 \cdot R4 \cdot R3 \cdot R2 \cdot R1 \cdot R0}$ : Set if all the bits of the result are zero's; cleared otherwise.  
 V = 0: Cleared.  
 C : Not affected.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMMED	ORA A	#Imm	8A	Imm		2	2
DIRECT	ORA A	M	9A	M		2	3
EXTND	ORA A	M	BA	MH	ML	3	4
INDEX	ORA A	Disp,X	AA	Disp		2	4
IMMED	ORA B	#Imm	CA	Imm		2	2
DIRECT	ORA B	M	DA	M		2	3
EXTND	ORA B	M	FA	MH	ML	3	4
INDEX	ORA B	Disp,X	EA	Disp		2	4

Category	Function
Transfer	<p>↓ (ACCX)                      SP ← (SP) - 0001</p> <p>Pushes the contents of ACCX onto the stack indicated by the SP. The SP is decremented by one.</p>

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
ACCX	PSH A		36			1	4
ACCX	PSH B		37			1	4

Category	Function
Transfer	<p>↓ (IXL), SP ← (SP) - 0001                      ↓ (IXH), SP ← (SP) - 0001</p> <p>Pushes the contents of the IX onto the stack indicated by the SP. The SP is decremented by two.</p>

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	PSHX		3C			1	5



Category	Function						
Transfer	SP ← (SP) + 0001 ↓ ACCX  Increments the SP by one, and pulls ACCX from the stack.						
Effects on the condition codes							
H : Not affected. I : " N : " Z : " V : " C : "							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
ACCX	PUL A		32			1	3
ACCX	PUL B		33			1	3

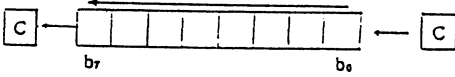
Category	Function
Transfer	<p> <math>SP \leftarrow (SP) + 0001 ; \uparrow IXH</math>  <math>SP \leftarrow (SP) + 0001 ; \uparrow IXL</math> </p> <p>                     Increments the SP by one, and pulls the IX from the stack.                      The SP is incremented by two in total.                 </p>

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	PULX		38			1	4

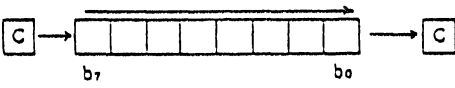
Category	Function
Shift & rotation	 <p>Shifts the contents of ACCX or memory M by one bit to the left. The original value of bit C is moved into b0, and the original value bit b7 to the bit C.</p>

Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N : R7: Set if the MSB of the result is "1"; cleared otherwise.  
 Z =  $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if all bits of the result are zero's; cleared otherwise.  
 V =  $N \oplus C$ : Set if either N=1 and C=0 or N=0 and C=1 after the instruction is executed; cleared otherwise.  
 Note: The N and C are those obtained after operation.  
 C = M7: Set if the MSB of ACCX or M is "1" before the instruction is executed; cleared otherwise.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
ACCX	ROL A		49			1	1
ACCX	ROL B		59			1	1
EXTND	ROL M		79	MH	ML	3	6
INDEX	ROL Disp,X		69	Disp		2	6

Category	Function
Shift & rotation	<div style="text-align: center;">  </div> <p>Shifts the contents of ACCX or memory M by one bit to the right. The original value of bit C is moved into bit 7 and the original value bit 0 to the bit C.</p>

Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N = R7: Set if the MSB of the result is "1"; cleared otherwise.  
 Z =  $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise.  
 V = N⊕C: Set if either N=1 and C=0 or N=0 and C=1 after the instruction is executed; cleared otherwise.  
 Note: The N and C are those obtained after operation.  
 C = M0: Set if the LSB of ACCX or M is "1" before the instruction is executed; cleared otherwise.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
ACCX	ROR A		46			1	1
ACCX	ROR B		56			1	1
EXTND	ROR M		76	MH	ML	3	6
INDEX	ROR	Disp,X	66	Disp		2	6

Category	Function						
Interrupt control	SP ← (SP) + 0001, ↓ CC SP ← (SP) + 0001, ↓ ACCB SP ← (SP) + 0001, ↓ ACCA SP ← (SP) + 0001, ↓ IXH SP ← (SP) + 0001, ↓ IXL SP ← (SP) + 0001, ↓ PCH SP ← (SP) + 0001, ↓ PCL Pulls the CCR, ACCB, ACCA, IXH, IXL, PCH and from the stack sequentially with incrementing SP by one at a time. Note that I=0 results if the interrupt mask bit I of CCR having been saved in the stack is zero.						
Effects on the condition codes							
H : Set or cleared according to the bit pulled from the stack. I :                   " N :                   " Z :                   " V :                   " C :                   "							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	RTI		3B			1	10

Category	Function
Subroutine control	<p> <math>SP \leftarrow (SP) + 0001</math>                      ↓ PCH  <math>SP \leftarrow (SP) + 0001</math>                      ↓ PCL                 </p> <p>                     Increments the SP by one and pulls the upper byte of the PC from the stack. Again increments the SP by one, and pulls the lower byte of the SP from the stack.                 </p>

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	RTS		39			1	5

Category	Function
Arithmetic operation	<p>ACCA ← (ACCA) - (ACCB)</p> <p>Subtracts the contents of ACCB from those of ACCA, and stores the result into the ACCA. The contents of the ACCB remain unchanged.</p>

Effects on the condition codes

- H : Not affected.
- I : Not affected.
- N = R7: Set if the result's MSB is "1"; cleared otherwise.
- Z =  $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise.
- V =  $A7 \cdot \overline{B7} \cdot R7 \oplus \overline{A7} \cdot B7 \cdot R7$ : Set if the result overflows; cleared otherwise.
- C =  $\overline{A7} \cdot B7 \oplus B7 \cdot R7 \oplus R7 \cdot \overline{A7}$ : Set if the absolute value of ACCB is greater than that of ACCA; cleared otherwise.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	SBA		10			1	1

Category	Function						
Arithmetic operation	$ACCX \leftarrow (ACCX) - (M) - (C)$  Subtracts the contents of memory M and the contents of bit C from those of ACCX, and stores the result into the ACCX.						
Effects on the condition codes							
H : Not affected. I : Not affected. N = R7: Set if the result's MSB is "1"; cleared otherwise. Z = $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the result is zero; cleared otherwise. V = $X7 \cdot \overline{M7} \cdot \overline{R7} \oplus X7 \cdot M7 \cdot R7$ : Set if the result overflows; cleared otherwise. C = $\overline{X7} \cdot M7 \oplus M7 \cdot R7 \oplus R7 \cdot \overline{X7}$ : Set if the absolute value of M contents plus C is greater than that of ACCX contents; cleared otherwise.							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMMED	SBC A	#Imm	82	Imm		2	2
DIRECT	SBC A	M	92	M		2	3
EXTND	SBC A	M	B2	MH	ML	3	4
INDEX	SBC A	Disp,X	A2	Disp		2	4
IMMED	SBC B	#Imm	C2	Imm		2	2
DIRECT	SBC B	M	D2	M		2	3
EXTND	SBC B	M	F2	MH	ML	3	4
INDEX	SBC B	Disp,X	E2	Disp		2	4



Category	Function						
Bit control	Bit C ← 1  Sets the carry bit C of the CCR.						
Effects on the condition codes							
H : Not affected. I : " N : " Z : " V : " C = 1 : Set							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	SEC		0D			1	1

SEt Interrupt mask

SEI

2

Category	Function
Bit control	<p>Bit I ← 1</p> <p>Sets the interrupt mask bit of the CCR.</p> <p>When the I bit is set, all maskable interrupts are inhibited and the MPU will recognize only a Non-Maskable Interrupt (NMI) request.</p>

Effects on the condition codes

H : Not affected.  
 I = 1 : Set  
 N : Not affected.  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	SEI		0F			1	1

SEt two's complement oVerflow bit

SEV

Category	Function
Bit control	Bit V ← 1  Sets the overflow bit V of the CCR.

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V = 1 : Set  
 C : Not affected.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	SEV		0B			1	1

Category	Function
Low power dissipation mode	Brings the CPU to a halt. All the internal register states are held as they are. The timer, serial communication interface and interrupt control are not affected by this instruction. If a CPU interrupt request occurs, the SLEEP mode is released. After releasing, following instructions are executed when bit I has been set by a maskable interrupt. When bit I has not been set by either maskable or non-maskable interrupt, the MCU sets bit I and loads the interrupt vectoring address into the program counter to start execution.

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	SLP		1A			1	4

2

Category	Function						
Load & store	<p>M ← (ACCX)</p> <p>Stores the contents of ACCX into the memory M. The contents of the ACCX remains unchanged.</p>						
Effects on the condition codes							
<p>H : Not affected.</p> <p>I : Not affected.</p> <p>N = X7: Set if the MSB of ACCX is "1"; cleared otherwise.</p> <p>Z = <math>\overline{X7 \cdot X6 \cdot X5 \cdot X4 \cdot X3 \cdot X2 \cdot X1 \cdot X0}</math>: Set if the contents of ACCX is zero; cleared otherwise.</p> <p>V = 0: Cleared.</p> <p>C : Not affected.</p>							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
DIRECT	STA A	M	97	M		2	3
EXTND	STA A	M	B7	MH	ML	3	4
INDEX	STA A	Disp,X	A7	Disp		2	4
DIRECT	STA B	M	D7	M		2	3
EXTND	STA B	M	F7	MH	ML	3	4
INDEX	STA B	Disp,X	E7	Disp		2	4

Category	Function
Load & store	<p><math>M:M+1 \leftarrow (ACCD)</math></p> <p>Stores the contents of ACCD into the memories M and M+1. The contents of the ACCD remains unchanged.</p>

Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N =  $\overline{AB15}$ : Set if the MSB of ACCD is "1"; cleared otherwise.  
 Z =  $\overline{AB15} \cdot \overline{AB14} \cdot \overline{AB13} \cdot \dots \cdot \overline{AB0}$ : Set if the contents of ACCD is zero; cleared otherwise.  
 V = 0: Cleared.  
 C = Not affected.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
DIRECT	STD	M	DD	M		2	4
EXTND	STD	M	FD	MH	ML	3	5
INDEX	STD	Disp,X	ED	Disp		2	5

Category	Function						
Stack pointer control	<p> <math>M \leftarrow (SPH)</math>  <math>M+1 \leftarrow (SPL)</math> </p> <p>Stores the upper byte of the SP into the memory M, and then the lower byte of the SP into the memory M+1.</p>						
Effects on the condition codes							
<p>H : Not affected.</p> <p>I : Not affected.</p> <p>N = SPH7: Set if the MSB of the stack pointer is "1"; cleared otherwise.</p> <p>Z = <math>(\overline{SPH7} \cdot \overline{SPH6} \cdot \overline{SPH5} \cdot \overline{SPH4} \cdot \overline{SPH3} \cdot \overline{SPH2} \cdot \overline{SPH1} \cdot \overline{SPH0}) \cdot (\overline{SPL7} \cdot \overline{SPL6} \cdot \overline{SPL5} \cdot \overline{SPL4} \cdot \overline{SPL3} \cdot \overline{SPL2} \cdot \overline{SPL1} \cdot \overline{SPL0})</math>: Set if the contents of the stack pointer is zero; cleared otherwise.</p> <p>V = 0: Cleared.</p> <p>C : Not affected.</p>							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
DIRECT	STS	M	9F	M		2	4
EXTND	STS	M	BF	MH	ML	3	5
INDEX	STS	Disp,X	AF	Disp		2	5

Store index register

STX

Category	Function
Index register control	<p>M ← (IXH) M+1 ← (IXL)</p> <p>Stores the upper byte of the IX into the memory M, then the lower byte of the IX into the memory M+1.</p>

Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N = IXH7: Set if the MSB of the index register is "1"; cleared otherwise.  
 Z = (IXH7·IXH6·IXH5·IXH4·IXH3·IXH2·IXH1·IXH0)·(IXL7·IXL6·IXL5·IXL4·IXL3·IXL2·IXL1·IXL0): Set if the contents of the index register is zero; cleared otherwise.  
 V = 0: Cleared.  
 C : Not affected.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
DIRECT	STX	M	DF	M		2	4
EXTND	STX	M	FF	MH	ML	3	5
INDEX	STX	Disp,X	EF	Disp		2	5

2



## SUBtract

SUB

Category	Function						
Arithmetic operation	<p>ACCX ← (ACCX) - (M)</p> <p>Subtracts the contents of memory M from those of ACCX, and stores the result into the ACCX.</p>						
Effects on the condition codes							
<p>H : Not affected.</p> <p>I : Not affected.</p> <p>N = R7: Set if the result's MSB is "1"; cleared otherwise.</p> <p>Z = <math>\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}</math>: Set if the result's contents is zero; cleared otherwise.</p> <p>V = <math>X7 \cdot \overline{M7} \cdot \overline{R7} \oplus \overline{X7} \cdot M7 \cdot R7</math>: Set if the result overflows; cleared otherwise.</p> <p>C = <math>\overline{X7} \cdot M7 \oplus M7 \cdot R7 \oplus R7 \cdot \overline{X7}</math>: Set if the absolute value of memory contents is greater than that of ACCX contents; cleared otherwise.</p>							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMMED	SUB A	#Imm	80	Imm		2	2
DIRECT	SUB A	M	90	M		2	3
EXTND	SUB A	M	B0	MH	ML	3	4
INDEX	SUB A	Disp,X	A0	Disp		2	4
IMMED	SUB B	#Imm	C0	Imm		2	2
DIRECT	SUB B	M	D0	M		2	3
EXTND	SUB B	M	F0	MH	ML	3	4
INDEX	SUB B	Disp,X	E0	Disp		2	4

Double SUBtract without carry

SUBD

Category	Function
Arithmetic operation	<p>ACCD ← (ACCD) - (M:M+1)</p> <p>Subtracts the contents of memories M: M+1 from the contents of ACCD, and stores the result into the ACCD.</p>

Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N = R15: Set if the Result's MSB is "1"; cleared otherwise.  
 Z =  $\overline{R15} \cdot \overline{R14} \cdot \overline{R13} \cdot \dots \cdot \overline{R0}$ : Set if the result's contents is zero; cleared otherwise.  
 V =  $D15 \cdot \overline{M15} \cdot \overline{R15} \cdot \overline{D15} \cdot M15 \cdot R15$ : Set if the result overflows; cleared otherwise.  
 C =  $\overline{D15} \cdot M15 \cdot M15 \cdot R15 \cdot R15 \cdot \overline{D15}$ : Set if the absolute value of memory contents is greater than that of ACCD contents; cleared otherwise.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMMED	SUBD	#Imm	83	ImmH	ImmL	3	3
DIRECT	SUBD	M	93	M		2	4
EXTND	SUBD	M	B3	MH	ML	3	5
INDEX	SUBD	Disp,X	A3	Disp		2	5

Category	Function	
Interrupt control	$PC \leftarrow (PC) + 0001$	Increments PC by one and pushes it onto the stack in the order of PCL, PCH, IXL, IXH, ACCA, ACCB and CCR. The stack pointer SP is decremented by 1 after each byte of data is stored on the stack. Concerning CCR, transfers bit 0 through bit 5 as they are and bits b6 and b7 as being set. Then sets the interrupt mask bit I, and loads the contents of the memory highest-order address minus 5 (\$FFFA) and those minus 4 (\$FFFB) into the PC.
	$\downarrow(PCL), SP \leftarrow (SP) - 0001$	
	$\downarrow(PCH), SP \leftarrow (SP) - 0001$	
	$\downarrow(IXL), SP \leftarrow (SP) - 0001$	
	$\downarrow(IXH), SP \leftarrow (SP) - 0001$	
	$\downarrow(ACCA), SP \leftarrow (SP) - 0001$	
	$\downarrow(ACCB), SP \leftarrow (SP) - 0001$	
	$\downarrow(CC), SP \leftarrow (SP) - 0001$	
	$I \leftarrow 1$	
	$PCH \leftarrow (\text{Highest-order address} - 0005)$	
	$PCL \leftarrow (\text{Highest-order address} - 0004)$	

## Effects on the condition codes

H : Not affected.  
 I = 1 : Set  
 N : Not affected.  
 Z : "  
 V : "  
 C : "

## Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	SWI		3F			1	12

Transfer from accumulator A to accumulator B

TAB

2

Category	Function
Transfer	<p>ACCB ← (ACCA)</p> <p>Transfers the contents of ACCA into ACCB. The contents of the ACCA remains unchanged.</p>

Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N = R7: Set if the MSB of ACCA is "1"; cleared otherwise.  
 Z =  $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the contents of ACCA is zero; cleared otherwise.  
 V = 0: Cleared.  
 C : Not affected.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	TAB		16			1	1

Category	Function
Transfer	<p>Transfers bits 0 through 5 of ACCA to the corresponding bits of the CCR. The contents of the ACCA remains unchanged.</p>

Effects on the condition codes

- H : Bit 5 of ACCA
- I : Bit 4 of ACCA
- N : Bit 3 of ACCA
- Z : Bit 2 of ACCA
- V : Bit 1 of ACCA
- C : Bit 0 of ACCA

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	TAP		06			1	1

Category	Function
Transfer	<p>ACCA ← (ACCB)</p> <p>Transfers the contents of ACCB into ACCA. The contents of the ACCB remains unchanged.</p>

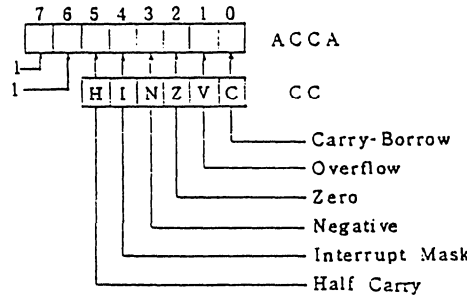
Effects on the condition codes

H : Not affected.  
 I : Not affected.  
 N = R7: Set if the MSB of ACCB is "1"; cleared otherwise.  
 Z =  $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ : Set if the contents of ACCB is zero; cleared otherwise.  
 V = 0: Cleared.  
 C : Not affected.

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	TBA		17			1	1

Category	Function						
Logical operation	<p>IM •(M)</p> <p>ANDs the immediate data and the contents of memory M to change the condition codes.</p> <p>The contents of the CCR can be referenced by the following branch instruction. Both operands remain unchanged.</p>						
Effects on the condition codes							
<p>H : Not affected.</p> <p>I : Not affected.</p> <p>N = R7: Set if the result's MSB is "1"; cleared otherwise.</p> <p>Z = <math>\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}</math>: Set if the result is zero; cleared otherwise.</p> <p>V = Cleared.</p> <p>C : Not affected.</p>							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
DIRECT	TIM	#Imm, M	7B	Imm	M	3	4
INDEX	TIM	#Imm, Disp, X	6B	Imm	Disp	3	5

Category	Function
Transfer	<p>ACCA ← (CC)</p>  <p>Transfers bits 0 through 5 of the CCR to the corresponding bits of ACCA. The contents of CCR remains unchanged.</p>

Effects on the condition codes

- H : Not affected.
- I : "
- N : "
- Z : "
- V : "
- C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	TPA		07			1	1



Category	Function						
Comparison & test	(ACCA) - 00 (M) - 00  Sets bits N and Z of the CCR according to the contents of ACCX or memory M.						
Effects on the condition codes							
H : Not affected. I : Not affected. N : M7: Set if the MSB of ACCX or M is "1"; cleared otherwise. Z = $\overline{M7} \cdot \overline{M6} \cdot \overline{M5} \cdot \overline{M4} \cdot \overline{M3} \cdot \overline{M2} \cdot \overline{M1} \cdot \overline{M0}$ : Set if the contents of ACCX or M is zero; cleared otherwise. V = 0: Cleared. C = 0: Cleared.							
Addressing modes and CPU cycles							
Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
ACCX	TST A		4D			1	1
ACCX	TST B		5D			1	1
EXTND	TST M		7D	MH	ML	3	4
INDEX	TST	Disp,X	6D	Disp		2	4

Transfer from Stack pointer to index register

TSX

Category	Function
Transfer	<p><math>IX \leftarrow (SP) + 0001</math></p> <p>Increments the contents of the SP by one, and loads it into the IX. The contents of the SP remain unchanged.</p>

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	TSX		30			1	1

2

Category	Function
Transfer	<p>SP ← (IX) - 0001</p> <p>Decrements the contents of the IX by one, and loads it into the SP. The contents of the IX remain unchanged.</p>

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	TXS		35			1	1

Category	Function	
Interrupt control	PC ← (PC) + 0001	<p>Increments PC by one and pushes it onto the stack in the order of PCL, PCH, IXL, IXH, ACCA, ACCB and CCR.</p> <p>The SP is decremented by 1 after each byte of data is pushed onto the stack.</p> <p>Concerning CCR, transfers bits 0 through 5 as they are and bits 6 and 7 as being set.</p> <p>The program execution stops temporarily until interrupt from a peripheral device occurs.</p> <p>If bit I is a 0 before an interrupt occurs, the following processings takes place when the interrupt has occurred. That is: sets bit I; and loads the interrupt vectoring address to the PC.</p>
	↓ (PCL), SP ← (SP) - 0001	
	↓ (PCH), SP ← (SP) - 0001	
	↓ (IXL), SP ← (SP) - 0001	
	↓ (IXH), SP ← (SP) - 0001	
	↓ (ACCA), SP ← (SP) - 0001	
	↓ (ACCB), SP ← (SP) - 0001	
↓ (CC), SP ← (SP) - 0001		

Effects on the condition codes

H : Not affected.  
 I : Not affected until an interrupt occurs and set if bit I is a 0 when the interrupt has occurred.  
 N : Not affected.  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	WAI		3E			1	9

Category	Function
Exchange	<p data-bbox="394 204 583 227">IX <math>\rightleftharpoons</math> ACCD</p> <p data-bbox="394 284 1085 307">Exchanges the contents of IX with those of ACCD.</p>

Effects on the condition codes

H : Not affected.  
 I : "  
 N : "  
 Z : "  
 V : "  
 C : "

Addressing modes and CPU cycles

Addressing mode	Mnemonic	Operand format	Instruction code			Bytes of instr. code	CPU cycles
			1st byte	2nd byte	3rd byte		
IMPL	XGDX		18			1	2

# HD6301/HD6303 SERIES HANDBOOK

## Section Three

# HD6301V1/HD6303R User's Manual

3

**Section 3**  
**HD6301V1/HD6303R User's Manual**  
**Table of Contents**

	<b>Page</b>
Notice on HD6301V and HD6303R .....	126
1. OVERVIEW .....	128
1.1 Features of HD6301V1 .....	128
1.2 Block Diagram .....	128
1.3 Functional Pin Description .....	129
2. INTERNAL ARCHITECTURE AND OPERATIONS.....	130
2.1 Mode Selection .....	130
2.2 Memory Map .....	135
2.3 Registers .....	140
2.4 I/O Ports .....	142
2.5 Programmable Timer.....	146
2.6 Serial Communication Interface .....	150
2.7 Interrupts .....	157
2.8 Reset .....	159
2.9 Oscillator .....	161
2.10 Strobe Signals .....	161
2.11 RAM Control Register .....	162
2.12 Low Power Consumption Mode.....	163
2.13 TRAP Function .....	167
3. INSTRUCTIONS .....	168
3.1 Addressing Modes.....	168
3.2 Instruction Set .....	170
3.3 Instruction Execution Cycles.....	175
3.4 System Flowchart .....	181
4. PIN ARRANGEMENT AND PACKAGE INFORMATION.....	183

5. ELECTRICAL CHARACTERISTICS .....	186
6. APPLICATIONS .....	191
6.1 Use of External Expanded Mode .....	191
6.2 Standby Mode .....	195
6.3 Address Trap, OP-Code Trap Application .....	198
6.4 Slow Memory Interface .....	199
6.5 Interface to HN61256 .....	200
6.6 Interface to the Realtime Clock (HD146818) .....	205
6.7 Reference Data of Battery Service Life .....	208
7. PRECAUTIONS .....	209
7.1 Write-Only Register .....	209
7.2 Address Strobe (AS) .....	209
7.3 Mode 0 .....	209
7.4 Trap Interrupt .....	209
7.5 Power-on Reset .....	211
7.6 Precaution to the Board Design of Oscillation Circuit .....	211
7.7 Application Note for High Speed System Design Using the HD6301V1 ..	212
APPENDIX	
I. EPROM on Package HD63P01M1 .....	218
II. Program Development Procedure and Support System .....	226
III. Q & A .....	233



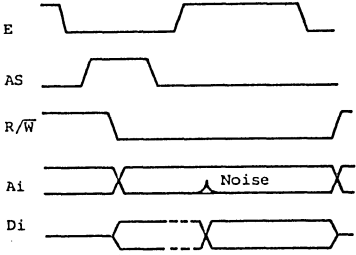
"Notice on HD6301V1"

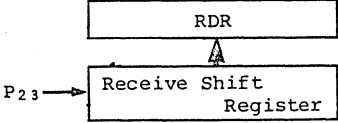
The HD6301V0 (including A and B version) was upgraded to the HD6301V1 series in early 1983.

The spec. deviation between the HD6301V0 series and the HD6301V1 series is as follows. Please refer to the data sheet for detailed spec. of the HD6301V1 series.

Table. Spec. Deviation between the HD6301V0 and the HD6301V1

Items	HD6301V0	HD6301V1
Operating Mode	Mode 2 : Not defined  Mode 3 : Not defined	Mode 2 ; Expanded Multiplexed Mode (equivalent to Mode 4)  Mode 3 ; Not defined
Electrical Characteristics	The electrical characteristics of 2MHz version (B version) are not specified.	The 2MHz version is guaranteed.
Timer	Has problem in output compare function. (Can be avoided by software.)	Fixed

GND Noise	HD6301V1	HD6303R	HD6303R1 HD63P01M1
	 <p>If load capacitance in each data line and GND impedance are large, noise may appear on address bus during MCU write cycle and data won't be written into RAM correctly. The noise is caused by GND impedance which becomes large when large transient current flows into GND at High to Low transition of data line.</p>		

SCI	HD 6301V1	HD6303R HD63P01M1	HD6303R1
		<p>When framing error occurs, receive data is not transferred from the Receive Shift Register to Receive Data Register (RDR).</p> 	<p>Receive data is transferred from Receive Shift Register to RDR even if framing error occurs.</p>

"Notice on HD6303R"

The HD6303R is the same die as the HD6301V1. The on-chip Mask ROM is disabled by mask option; Therefore not all modes of operation are available on the HD6303R. Please note that wherever HD6301V1 is referenced, the information also applies to the HD6303R.

"Notice on HD6303R1"

The HD6303R has been upgraded to HD6303R1. Refer to the following figures for differences between the devices. All other characteristics remain the same.



## 1. OVERVIEW

### 1.1 Features of HD6301V1

The HD6301V1 provides the following features:

- Expanded instruction set of the HD6801 family
- Abundant on-chip functions compatible with the HD6801 family: 4k-byte of ROM, 128-byte of RAM, 29 parallel I/O Lines, 2 data strobe Lines, 16-bit timer, serial communication interface
- Low power consumption mode: sleep/standby mode
- Minimum instruction execution time:  $1\mu\text{s}$  ( $f = 1\text{MHz}$ ),  $0.67\mu\text{s}$  ( $f = 1.5\text{MHz}$ ),  $0.5\mu\text{s}$  ( $f = 2\text{MHz}$ )
- Bit manipulation and bit test instruction
- Error detection: Address trap and operation code trap
- Address space up to 65k words
- Wide operation range:  $V_{CC} = 3$  to  $6\text{V}$  ( $f = 0.1 \sim 0.5\text{MHz}$ ),  $f = 0.1$  to  $2.0\text{MHz}$  ( $V_{CC} = 5\text{V} \pm 10\%$ )
- TTL compatible input/output

### 1.2 Block Diagram

A block diagram of HD6301V1 is given in Fig. 1-2-1

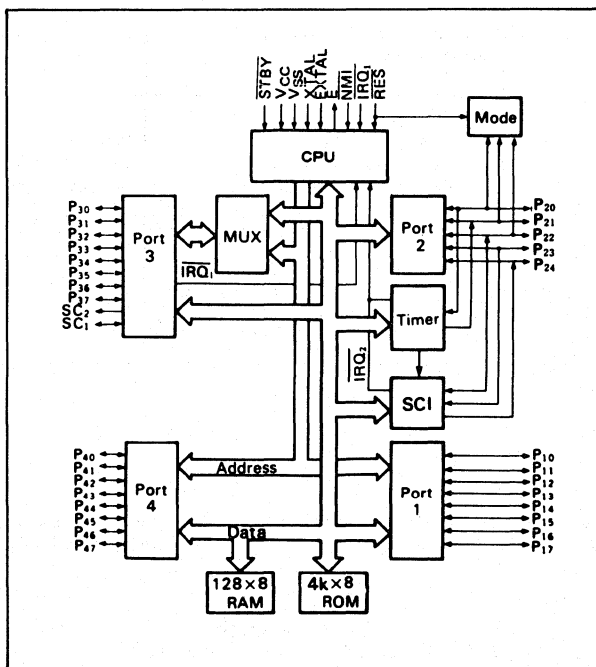


Fig. 1-2-1 Block Diagram of HD6301V1



### 1.3 Functional Pin Description

Table 1-3-1 lists the pin functions. Refer to "2. INTERNAL ARCHITECTURE AND OPERATIONS" for more details.

Table 1-3-1 Pin Functions

Pin	Function				
V <sub>CC</sub> , V <sub>SS</sub>	Power supply and GND pins				
XTAL EXTAL	Crystal connection pin. When external clock is used, input it to EXTAL, and XTAL should be open.				
RES	Reset input pin. When this pin is in "Low" state, MCU is set to reset state.				
STBY	Standby input pin. When this pin is in "Low" state, MCU is set to standby state.				
NMI	Nonmaskable interrupt input pin for edge detection (Negative edge).				
IRQ <sub>1</sub>	Interrupt input pin for level detection (Active Low)				
E	System clock output pin. The frequency is 1/4 of the crystal oscillator frequency.				
P <sub>20</sub> /TIN	5-bit I/O port	Timer input-capture input pin			
P <sub>21</sub> /TOUT		Timer output-compare output pin			
P <sub>22</sub> /SCLK		SCI clock I/O port			
P <sub>23</sub> /RX		SCI receiving pin			
P <sub>24</sub> /TX		SCI transmitting pin			
Following pins function depending on each operation mode					
	Mode 0,2,4	Mode 1	Mode 5	Mode 6	Mode 7
PORT 1	8-bit I/O port	Lower address (A <sub>0</sub> ~A <sub>7</sub> )	8-bit I/O port	←	←
PORT 3	Data (D <sub>0</sub> ~D <sub>7</sub> ) Lower address (A <sub>0</sub> ~A <sub>7</sub> ) Multiplexed Bus	Data Bus D <sub>0</sub> ~D <sub>7</sub>	←	Data (D <sub>0</sub> ~D <sub>7</sub> ) Lower address (A <sub>0</sub> ~A <sub>7</sub> ) Multiplexed Bus	←
PORT 4	Upper address (A <sub>8</sub> ~A <sub>15</sub> )	←	Lower address (A <sub>0</sub> ~A <sub>7</sub> ) or Input-only pin	Upper address (A <sub>8</sub> ~A <sub>15</sub> ) or Input-only pin	8-bit I/o port
SC <sub>1</sub>	Address strobe (AS) output pin	/	I/O strobe (IOS) output pin	Address strobe (AS) output pin	Input strobe (IS3) output pin
SC <sub>2</sub>	Read/write signal (R/W) output pin	←	←	←	Output strobe (OS3) output pin

3

## 2. INTERNAL ARCHITECTURE AND OPERATIONS

This section describes the internal architecture of the HD6301V1 and its operation.

### 2.1 Mode Selection

After the MCU is reset, a user must determine the operation mode of the HD6301V1 by strapping three pins 8, 9 and 10 which are connected by hardware externally. These pins correspond to  $P_{20}$ ,  $P_{21}$  and  $P_{22}$  respectively.

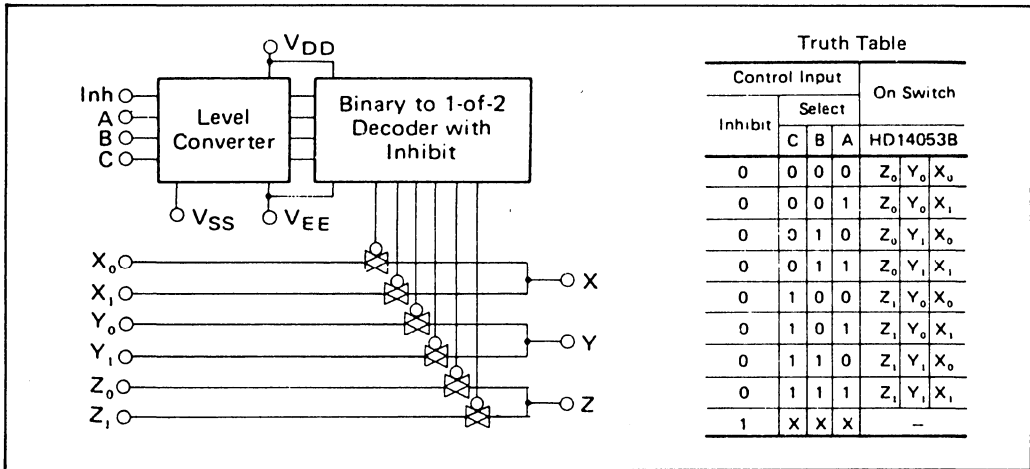
Individual signals on the above three pins are latched and loaded into the program control bits PC0, PC1 and PC2, the most significant three bits of I/O port 2 register, when the  $\overline{RES}$  signal goes "High". The bit assignment of the port 2 data register is shown below.

Port 2 Data Register								
	7	6	5	4	3	2	1	0
\$0003	PC2	PC1	PC0	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0

An example of an external hardware for mode selection is shown in Fig. 2-1-1. The HD14053B may be used to separate the MCU from its peripheral devices during reset (Data confliction should be avoided between the peripheral devices and mode generator circuit). Because bits 5, 6 and 7 of port 2 are for read only, so the operation mode cannot be altered by software. The mode selection in the HD6301V1 is summarized in Table 2-1-1.

The HD6301V1 has three basic operation modes:

- 1) Single chip mode
- 2) Expanded multiplexed mode  
(Compatible Bus with HMCS6800 peripheral LSIs)
- 3) Expanded non-multiplexed mode  
(Compatible Bus with HMCS6800 peripheral LSIs)



HD14053B Multiplexers/De-Multiplexers

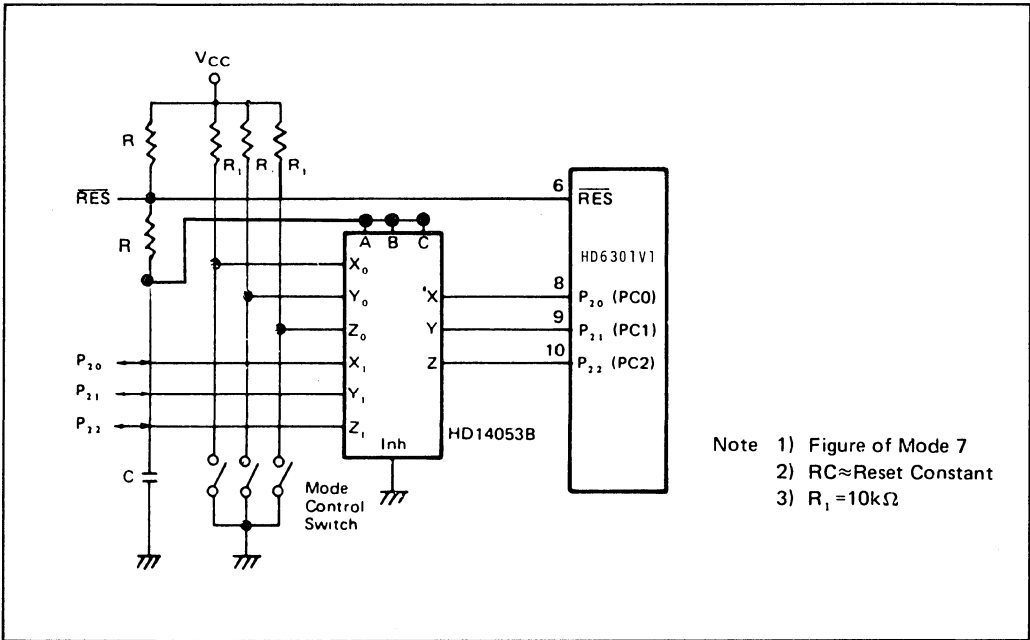


Fig. 2-1-1 Recommended Circuit for Mode Selection

Table 2-1-1 Mode Selection Summary

Mode	P <sub>22</sub> (PC2)	P <sub>21</sub> (PC1)	P <sub>20</sub> (PC0)	ROM	RAM	Interrupt Vectors	Bus Mode	Operating Mode
7 <sup>(4)</sup>	H	H	H	I	I	I	I	Single Chip
6 <sup>(4)</sup>	H	H	L	I	I	I	MUX <sup>3)</sup>	Multiplexed/Partial Decode
5 <sup>(4)</sup>	H	L	H	I	I	I	NMUX <sup>3)</sup>	Non-Multiplexed/Partial Decode
4	H	L	L	E <sup>1)</sup>	I	E	MUX	Multiplexed/RAM
3 <sup>(4)</sup>	L	H	H	—	—	—	—	Not Used
2	L	H	L	E <sup>1)</sup>	I	E	MUX	Multiplexed/RAM
1	L	L	H	E <sup>1)</sup>	I	E	NMUX	Non-Multiplexed
0 <sup>(4)</sup>	L	L	L	I	I	I <sup>2)</sup>	MUX	Multiplexed Test

## LEGEND :

I - Internal  
 E - External  
 MUX - Multiplexed  
 NMUX - Non-Multiplexed  
 L - Logic "0"  
 H - Logic "1"

## (NOTES)

- 1) Internal ROM is disabled.
- 2) Reset vector is external for 3 or 4 cycles after  $\overline{RES}$  goes "high".
- 3) Idle lines of Port 4 address outputs can be assigned to Input Port.
- 4) Not available on HD6303R or HD6303R1

## (1) Single Chip Mode

In the Single Chip Mode, all ports will become I/O. This is shown in figure 2-1-2. In this mode, SC<sub>1</sub>, SC<sub>2</sub> pins are configured for control lines of Port 3 and can be used as input strobe ( $\overline{IS3}$ ) and output strobe ( $\overline{OS3}$ ) for handshaking data.

## (2) Expanded Multiplexed Mode

In this mode, Port 4 is configured for I/O (inputs only) or address lines. The data bus and the lower order address bus are multiplexed in Port 3 and can be separated by an output called Address Strobe.

Port 2 is configured for 5 parallel I/O or Serial I/O, or Timer, or any combination thereof. Port 1 is configured for 8 parallel I/O. In this mode HD6301V1 is expandable to 65k words (See Fig. 2-1-3).

Since the data bus is multiplexed with the lower order address bus in Port 3 in the expanded multiplexed mode, address bits must be latched outside. 74LS373 (Octal-D type transparent latches) is required for address latch. Latch connection of the HD6301V1 is shown in Fig. 2-1-4.

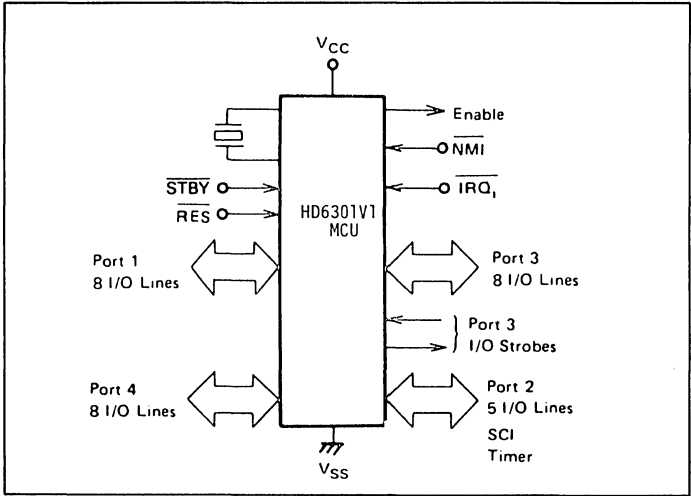


Fig. 2-1-2 HD6301V1 MCU Single-Chip Mode

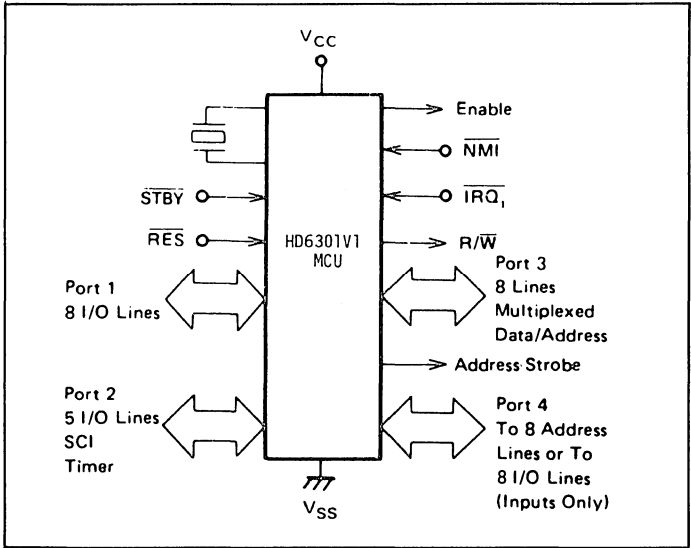


Fig. 2-1-3 HD6301V1 MCU Expanded Multiplexed Mode



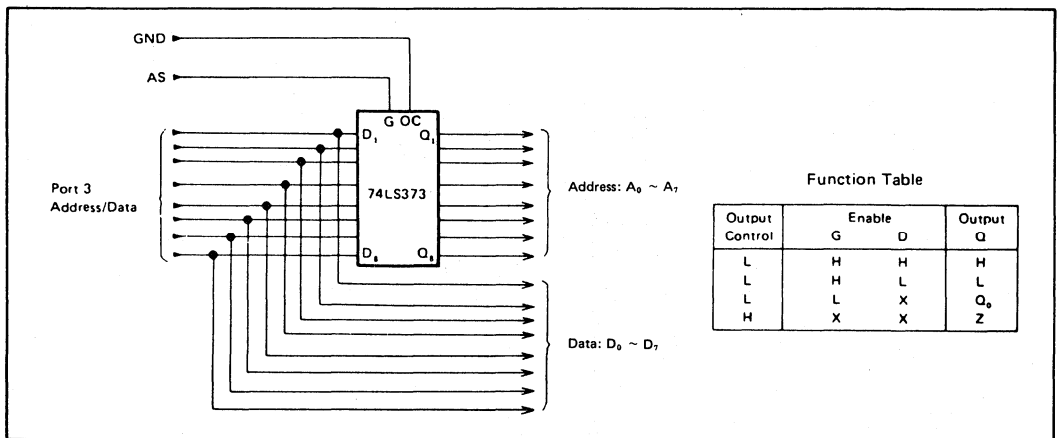


Fig. 2-1-4 Latch Connection

### (3) Expanded Non Multiplexed Mode

In this mode, the HD6301V1 can directly address HMCS6800 peripherals with no address latch. In mode 5, Port 3 becomes a data bus. Port 4 becomes  $A_0$  to  $A_7$  address bus or partial address bus and I/O (inputs only). Port 2 is configured for a parallel I/O, Serial I/O, Timer or any combination. Port 1 is configured as a parallel I/O only.

In this mode, HD6301V1 is expandable to 256 locations. In the application system with fewer addresses, idle pins of Port 4 can be used as I/O lines (inputs only) (See Fig. 2-1-5).

In mode 1, Port 3 becomes a data bus and Port 1 becomes  $A_0$  to  $A_7$  address bus, and Port 4 becomes  $A_8$  to  $A_{15}$  address bus. Port 2 is configured for a parallel I/O, Serial I/O, Timer or any combination. In this mode, the HD6301V1 is expandable to 65k words with no address latch. (See Fig. 2-1-5).

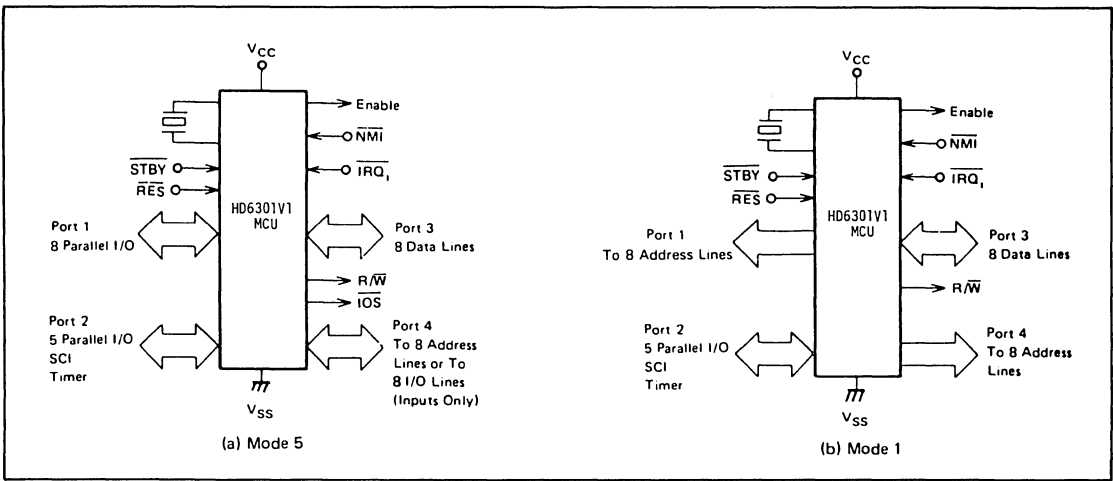


Fig. 2-1-5 HD6301V1 MCU Expanded Non Multiplexed Mode

(4) Mode and Port Summary MCU Signal Description

This section gives a description of the MCU signals for the various modes.  $SC_1$  and  $SC_2$  are signals which vary with the chip mode.

Table 2-1-2 Feature of each mode and Lines

MODE		PORT 1 Eight Lines	PORT 2 Five Lines	PORT 3 Eight Lines	PORT 4 Eight Lines	$SC_1$	$SC_2$
SINGLE CHIP		I/O	I/O	I/O	I/O	$\overline{IS3}$ (I)	$\overline{OS3}$ (O)
EXPANDED MUX		I/O	I/O	ADDRESS BUS ( $A_0 - A_7$ ) DATA BUS ( $D_0 - D_7$ )	ADDRESS BUS* ( $A_8 - A_{15}$ )	AS(O)	R/ $\overline{W}$ (O)
EXPANDED	Mode 5	I/O	I/O	DATA BUS ( $D_0 - D_7$ )	ADDRESS BUS* ( $A_0 - A_7$ )	$\overline{IOS}$ (O)	R/ $\overline{W}$ (O)
NON-MUX	Mode 1	ADDRESS BUS ( $A_0 - A_7$ )	I/O	DATA BUS ( $D_0 - D_7$ )	ADDRESS BUS ( $A_8 - A_{15}$ )	Not Used	R/ $\overline{W}$ (O)

\*These lines can be substituted for I/O (Input Only) starting with the MSB (except Mode 0, 2, 4). When they are not used as address lines.

I = Input       $\overline{IS3}$  = Input Strobe      SC = Strobe Control  
 O = Output     $\overline{OS3}$  = Output Strobe      AS = Address Strobe  
 R/ $\overline{W}$  = Read/Write     $\overline{IOS}$  = I/O Select

2.2 Memory Map

The MCU can address up to 65k bytes depending on the operating mode. Fig. 2-2-1 shows a memory map for each operating mode. The first 32 locations of each map are for the MCU's internal register only, as shown in Table 2-2-1.

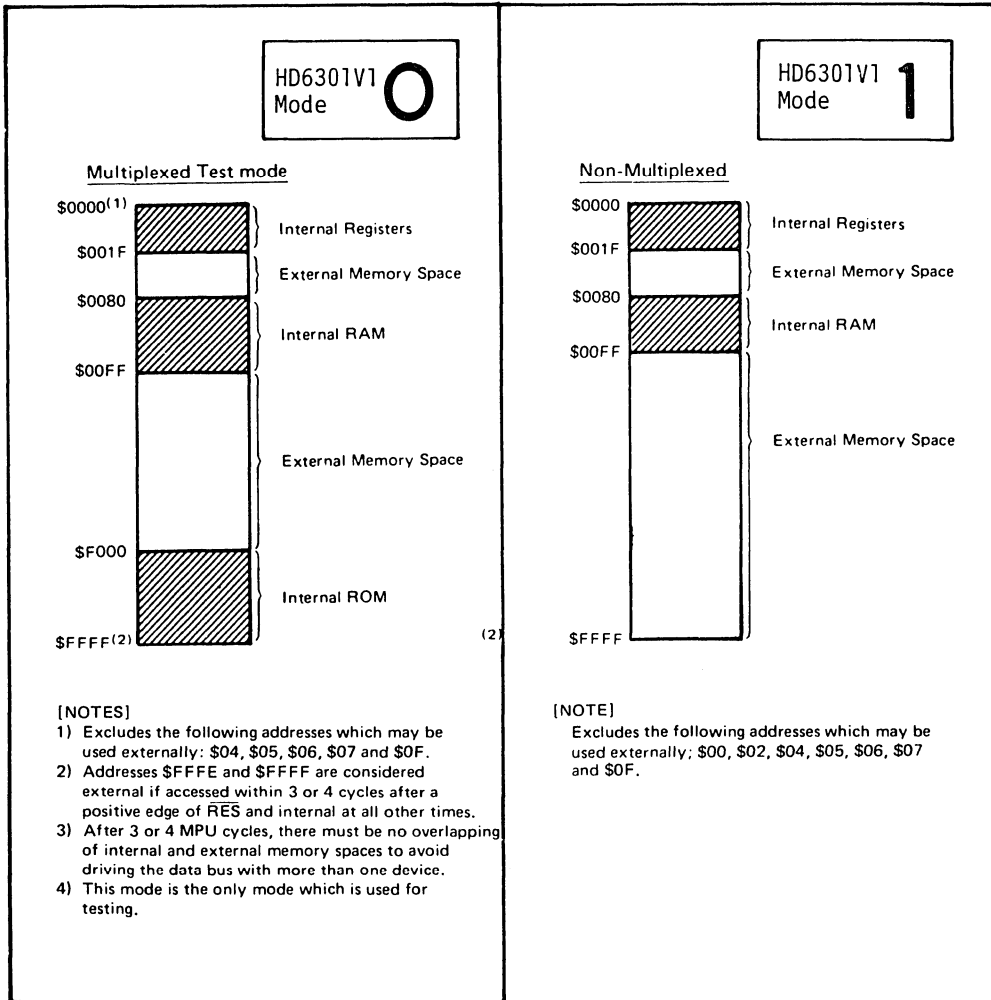
Table 2-2-1 Internal Register Area

Register	Address	R/W*4/Initialize at RESET							
		7	6	5	4	3	2	1	0
Port 1 Data Direction Register	\$00*1	W							
		\$00							
Port 2 Data Direction Register	\$01	W							
		\$00							
Port 1 Data Register	\$02*1	R/W *5							
		Undefined							
Port 2 Data Register/Mode Register	\$03	R *6			R/W *5				
		P <sub>22</sub>	P <sub>21</sub>	P <sub>20</sub>	Undefined				
Port 3 Data Direction Register	\$04*2	W							
		\$00							
Port 4 Data Direction Register	\$05*3	W							
		\$00							
Port 3 Data Register	\$06*2	R/W *5							
		Undefined							
Port 4 Data Register	\$07*3	R/W *5							
		Undefined							
Timer Control and Status Register	\$08	R	R	R	R/W	R/W	R/W	R/W	R/W
		0	0	0	0	0	0	0	0
Counter (High Byte)	\$09	R/W							
		\$00							
Counter (Low Byte)	\$0A	R/W							
		\$00							
Output Compare Register (High Byte)	\$0B	R/W							
		\$FF							
Output Compare Register (Low Byte)	\$0C	R/W							
		\$FF							
Input Capture Register (High Byte)	\$0D	R							
		\$00							
Input Capture Register (Low Byte)	\$0E	R							
		\$00							
Port 3 Control and Status Register	\$0F*2	R	R/W	Un-used	R/W	R/W	Unused		
		0	0	1	0	0	1	1	1
Rate and Mode Control Register	\$10	Unused				W	W	W	W
		1	1	1	1	0	0	0	0
Transmit/Receive Control and Status Register	\$11	R	R	R	R/W	R/W	R/W	R/W	
		0	0	1	0	0	0	0	
Receive Data Register	\$12	R							
		\$00							
Transmit Data Register	\$13	W							
		\$00							
RAM Control Register	\$14	R/W	R/W	Unused					
		*7	1	1	1	1	1	1	1
Reserved	\$15~\$1F	/							

(\*1 through 8 are shown in the next page.)



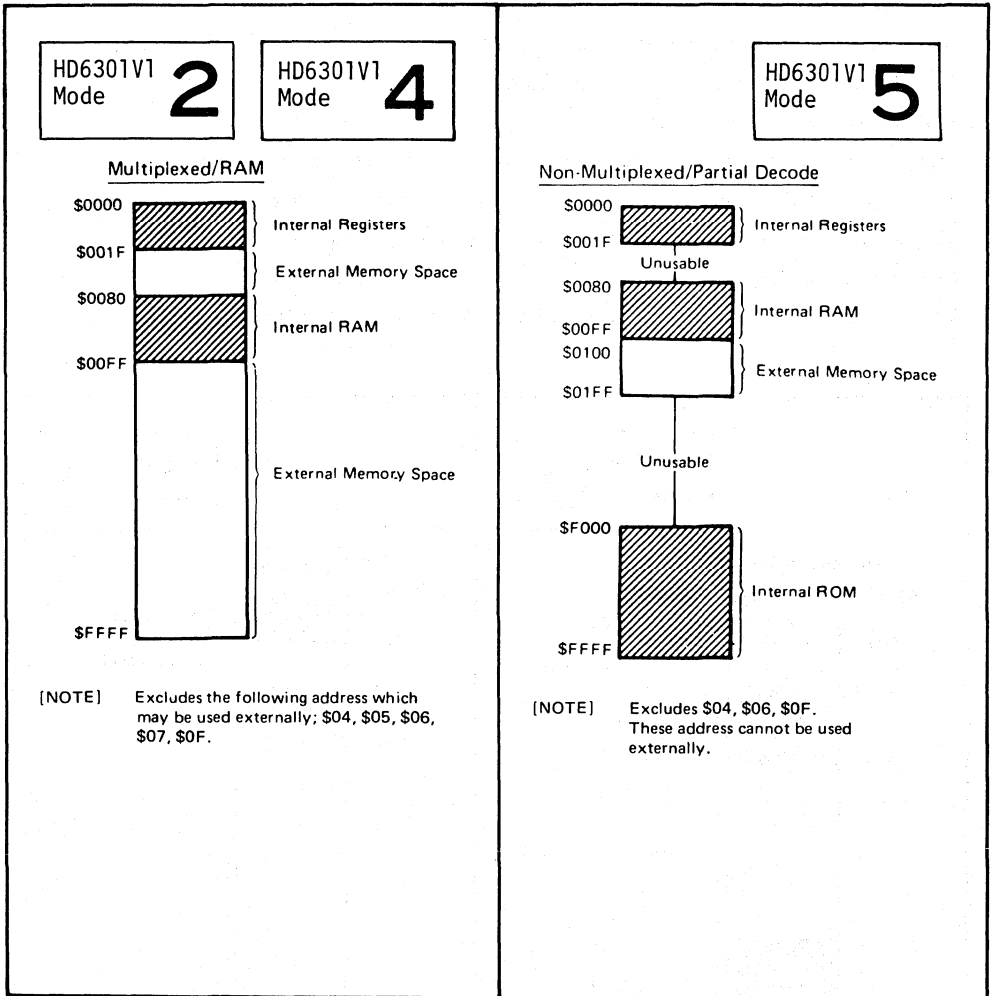
- \*1 External address in mode 1.
- \*2 External address in modes 0, 1, 2, 4, 5, 6; cannot be accessed in mode 5.
- \*3 External address in modes 0, 1, 2, 4.
- \*4 R : Read-only register, W : Write-only register, R/W : Read/Write register.
- \*5 The pin state is read instead of the data of the register when reading Ports. (Refer to "2.4 I/O Ports" for I/O Port 3.)
- \*6 The values of program control bit (PC<sub>0</sub> ~ PC<sub>2</sub>) depend on P<sub>20</sub> ~ P<sub>21</sub> during reset.
- \*7 Refer to "2.12 Low Power Consumption Mode" for standby mode.



(to be continued)

Fig. 2-2-1 HD6301V1 Memory Maps



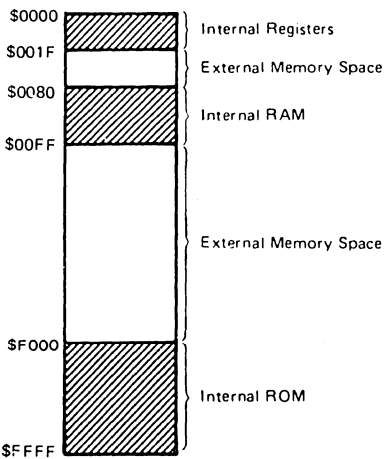


(to be continued)

Fig. 2-2-1 HD6301V1 Memory Maps

HD6301V1  
Mode **6**

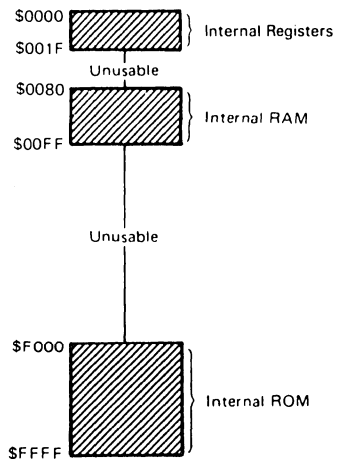
Multiplexed/Partial Decode



[NOTE]  
Excludes the following address which may be used externally: \$04, \$06, \$0F.

HD6301V1  
Mode **7**

Single Chip



**3**

Fig. 2-2-1 HD6301V1 Memory Maps

## 2.3 Registers

The followings describe the HD6301V1 internal architectures and operations.

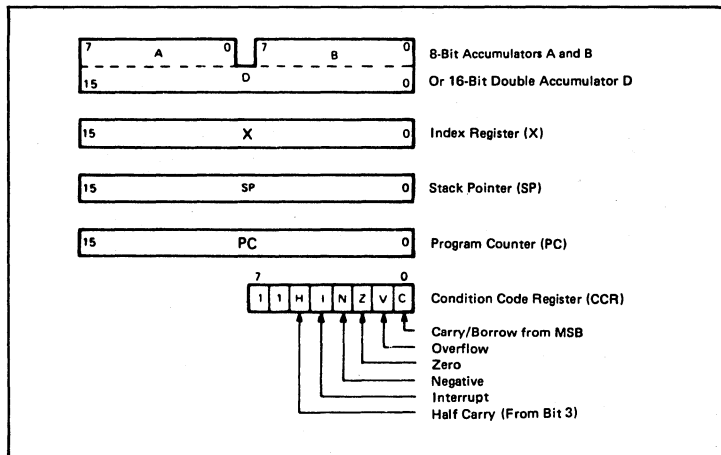


Fig. 2-3-1 Registers of HD6301V1

### (1) Accumulators (A & B, or D)

Two 8-bit registers (ACCA and ACCB) that store the result of arithmetic/logical operation and data. When combined, they make up a 16-bit register (ACCD) used for 16-bit operations. Note that the contents of ACCA and ACCB are destroyed after an ACCD-based operation.

### (2) Index Register (IX)

A 16-bit register that stores either 16-bit data intended for use in indexed addressing mode or ordinary 16-bit data.

### (3) Stack Pointer (SP)

A 16-bit register whose contents indicate the address of a stack operation. This may be used also as a register for ordinary 16-bit data.

### (4) Program Counter (PC)

A 16-bit register whose contents indicate the address of the program being currently executed. Note that software cannot access to this register.

(5) Condition Code Register (CCR)

A register consisting of the following bits: carry (C), overflow (V), zero (Z), negative (N), interrupt mask (I), and half-carry (H). After an instruction is executed, these bits change its states depending on the result of operation and are tested by different conditional branch instructions. The upper 2 bits of this register cannot be used. Individual bits are detailed below. Refer to the following description of each instruction for more details.

(a) Half-carry (H)

This bit is set to "1" if a carry from bit 3 to bit 4 occurs during execution of an ADD, ABA or ADC instruction; it is cleared if no carry takes place.

(b) Interrupt mask (I)

When set at "1", this bit disables any maskable interrupt ( $\overline{IRQ}_1$ ,  $\overline{IRQ}_2$ ).

(c) Negative (N)

After an instruction is executed, this bit is set to "1" if the MSB as the result of operation is "1"; it is cleared if the MSB is "0".

(d) Zero (Z)

After an instruction is executed, this bit is set to "1" if the result of operation is "0"; otherwise, it is cleared.

(e) Overflow (V)

After an instruction is executed, this bit is set if the result of operation shows a 2's complement overflow; it is cleared if no overflow occurs.

(f) Carry (C)

After an instruction is executed, this bit is set to "1" if a carry or a borrow generates from MSB; it is cleared in any other case.



## 2.4 I/O Ports

There are four I/O ports on HD6301V1 MCU (three 8-bit ports and one 5-bit port). 2 control pins are connected to one of the 8-bit port. Each port has an independent write-only data direction register to program individual I/O pins for input or output.\*

When the bit of associated Data Direction Register is "1", I/O pin is programmed for output, if "0", then programmed for an input.

There are four ports; Port 1, Port 2, Port 3, and Port 4. Addresses of each port and associated Data Direction Register are shown in Table 2-4-1.

\* Only one exception is bit 1 of Port 2 which becomes either a data input or a timer output. It cannot be used as an output port.

Table 2-4-1 Port and Data Direction Register Addresses

Ports	Port Address	Data Direction Register Address
I/O Port 1	\$0002	\$0000
I/O Port 2	\$0003	\$0001
I/O Port 3	\$0006	\$0004
I/O Port 4	\$0007	\$0005

### (1) I/O Port 1

This is an 8-bit port, each bit being defined individually as inputs or outputs by associated Data Direction Register. The 8-bit output buffers have three-state capability, maintaining in high impedance state when they are used for input. In order to be read accurately, the voltage on the input lines must be more than 2.0V for logic "1" and less than 0.8V for logic "0".

These are TTL compatible. After the MCU has been reset, all I/O lines are configured as inputs in all modes except mode 1. In all modes other than expanded non multiplexed mode 1, Port 1 is always parallel I/O. In mode 1, Port 1 will be output line for lower order address lines (A<sub>0</sub> to A<sub>7</sub>).

(2) I/O Port 2

This port has five lines, whose I/O direction depends on its data direction register. The 5-bit output buffers have three-state capability, going high impedance state when used as inputs. In order to be read accurately, the voltage on the input pins must be more than 2.0V for logic "1" and less than 0.8V for logic "0". After the MCU has been reset, I/O pins are configured as inputs. These pins on Port 2 (pins 10, 9, 8 of the chip) are used to program the operating mode during reset. The values of these three pins during reset are latched into the upper 3 bits (bit 7, 6 and 5). Refer to "2.1 Mode Selection" for more details.

In all modes, Port 2 can be configured as I/O lines. This port also provides access to the Serial I/O and the Timer. However, note that bit 1 ( $P_{21}$ ) is the only pin restricted to data input or Timer output.

(3) I/O Port 3

This is an 8-bit port which can be configured as I/O lines, a data bus, or an address bus multiplexed with data bus. Its function depends on hardware operation mode programmed by the user using 3 bits of Port 2 during Reset. Port 3 as a data bus is bi-directional. For an input from peripherals, regular TTL level must be supplied, that is greater than 2.0V for a logic "1" and less than 0.8V for a logic "0". This TTL compatible three-state buffer can drive one TTL load and 90pF capacitance. In the expanded Modes, data direction register will be inhibited after Reset and data flow will be dependent on the state of the R/W signal. Function of Port 3 for each mode is explained below.

Single Chip Mode (Mode 7): Parallel Inputs/Outputs as programmed by its corresponding Data Direction Register.

There are two control lines associated with this port in this mode, an input strobe ( $\overline{IS3}$ ) and an output strobe ( $\overline{OS3}$ ), both being used for handshaking. They are

controlled by I/O Port 3 Control/Status Register. Additional 3 characteristics of Port 3 are summarized as follows:

- (1) Port 3 input data can be latched using  $\overline{IS3}$  ( $SC_1$ ) as a control signal.
- (2)  $\overline{OS3}$  ( $SC_2$ ) can be generated by MPU read or write to Port 3's data register.
- (3)  $\overline{IRQ_1}$  interrupt can be generated by an  $\overline{IS3}$  falling edge.

Port 3 strobe and latch timings are shown in Figs. 5-5 and 5-6, respectively.

#### I/O Port 3 Control/Status Register

	7	6	5	4	3	2	1	0
	$\overline{IS3}$	$\overline{IS3}$	X	OSS	LATCH	X	X	X
5000F	FLAG	$\overline{IRQ_1}$ ENABLE			ENABLE			

Bit 0 Not used.

Bit 1 Not used.

Bit 2 Not used.

Bit 3 LATCH ENABLE.

Bit 3 is used to control the input latch of Port 3. If the bit is set to "1", the input data on Port 3 is latched by the falling edge of  $\overline{IS3}$ . The latch is cleared by the MCU read to Port 3; it can be latched again. Bit 3 is cleared by a reset.

Bit 4 OSS (Output Strobe Select)

This bit identifies the cause of output strobe generation: a write operation or read operation to I/O Port 3. When the bit is cleared, the strobe will be generated by a read operation to Port 3. When the bit is not cleared, the strobe will be generated by a write operation. Bit 4 is cleared by reset.

Bit 5 Not used.

Bit 6  $\overline{IS3}$  ENABLE.

If the  $\overline{IS3}$  flag (bit 7) is set with bit 6 set, an interrupt

is enabled. Clearing the flag causes the interrupt to be disabled. The bit is cleared by reset.

Bit 7  $\overline{IS3}$  FLAG.

Bit 7 is a read-only bit which is set by the falling edge of  $\overline{IS3}$  ( $SC_1$ ). It is cleared by a read of the Control/Status Register followed by a read/write of I/O Port 3. The bit is cleared by reset.

Expanded Non Multiplexed Mode (mode 1, 5)

In this mode, Port 3 becomes the data bus. ( $D_0$  to  $D_7$ )

Expanded Multiplexed Mode (mode 0, 2, 4, 6)

Port 3 becomes both the data bus ( $D_0 \sim D_7$ ) and lower 8 bits of the address bus ( $A_0 \sim A_7$ ). An address strobe output is "High" when the address is on the port.

(4) I/O Port 4

This is an 8-bit port that becomes either I/O or address output depending on the operation mode selected. In order to be read accurately, the voltage at the input lines must be greater than 2.0V for a logic "1", and less than 0.8V for a logic "0". For outputs, each line is TTL compatible and can drive one TTL load and 90pF capacitance. After reset, this port becomes inputs. To use these pins as addresses, they should be programmed as outputs.

Function of Port 4 for each mode is explained below.

Single Chip Mode (Mode 7): Parallel Inputs/Outputs as programmed by its associated data direction register.

Expanded Non Multiplexed Mode (Mode 5): In this mode, Port 4 becomes the lower address lines ( $A_0$  to  $A_7$ ) by writing "1"s on the data direction register.

When all of the eight bits are not required as addresses, the remaining lines can be used as I/O lines (Inputs only).

Expanded Non Multiplexed Mode (Mode 1): In this mode, Port 4 becomes output for upper order address lines ( $A_8$  to  $A_{15}$ ).

Expanded Multiplexed Mode (Mode 0, 2, 4): In this mode, Port 4 becomes output for upper order address lines (A<sub>8</sub> to A<sub>15</sub>) regardless of the value of data direction register.

The relation between each mode and I/O port 1 to 4 is summarized in Table 2-1-2.

## 2.5 Programmable Timer

The HD6301V1 provides 16-bit programmable timer which can measure input waveform and generate an output waveform. The pulse width of both input/output may vary from microseconds to seconds. microseconds to many seconds.

The timer hardware consists of

- an 8-bit control and status register
- a 16-bit free running counter
- a 16-bit output compare register, and
- a 16-bit input capture register

A block diagram of the timer is shown in Figure 2-5-1.

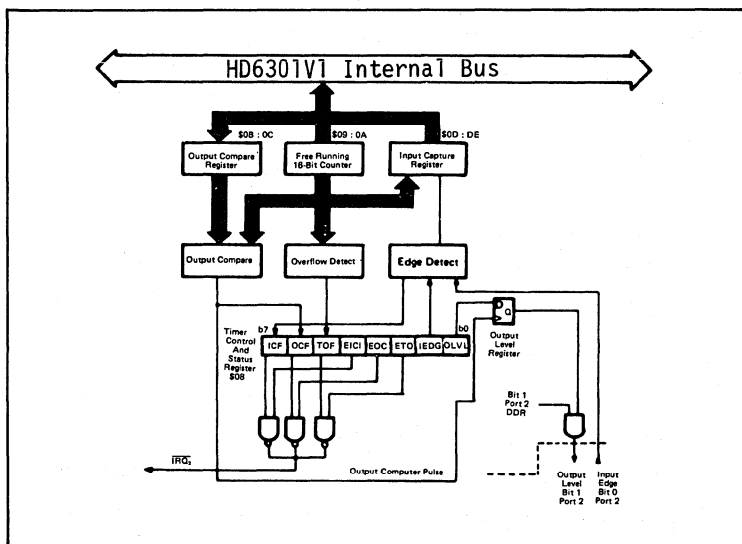


Fig. 2-5-1 Programmable Timer Block Diagram

(1) Free Running Counter (\$0009:000A)

The key timer element is a 16-bit free running counter, that is driven by an E (Enable) clock to increment its values. The counter value is readable by the MPU software at any time with no effects on the counter. The counter is cleared during reset.

When writing to the upper byte (\$09), the CPU writes the preset value (\$FF8) into the counter (address \$09, \$0A) regardless of the write data value. But when writing to the lower byte (\$0A) after the upper byte writing, the CPU writes not only the lower byte data into lower 8 bit, but also the upper byte data into higher 8 bit of the FRC. The counter value written to the counter using the double store instruction is shown in Figure 2-5-2.

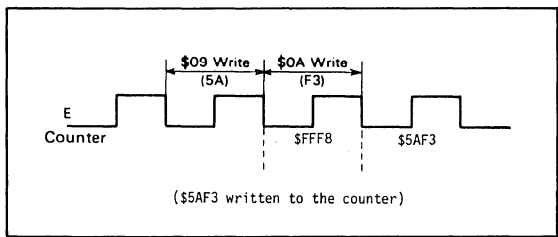


Fig. 2-5-2 Counter Write Timing

\* To write to the counter can disturb serial operations, so it should be inhibited during using the SCI.

(2) Output Compare Register (\$000B:\$000C)

This is a 16-bit read/write register which is used to control an output waveform. The data of this register is constantly compared with the free running counter.

When the data matches, a flag (OCF) in the timer control/status register (TCSR) is set and the current value of an output level Bit (OLVL) in the TCSR is transferred to Port 2 bit 1. When bit 1 of the Port 2 data direction register is "1" (output) the OLVL value will appear on the bit 1 of Port 2. Then, the value of Output Compare Register and Output level bit should be changed to control an output level again on the next compare values.

The output compare register is set to \$FFFF during reset. The compare function is inhibited at the cycle of writing to the upper bytes of the output compare register and at the cycle just after that. It is also inhibited in same manner at the cycle of writing to the free running counter.

- \* For the data write to The OCR (Output Compare Register), 2-byte transfer instructions such as STD, STX are available.

(3) Input Capture Register (\$000D:\$000E)

The input capture register is a 16-bit read-only register used to store the FRC's value obtained when the proper transition of an external input signal occurs.

The input transition change required to trigger the counter transfer is controlled by the input Edge bit (IEDG).

To allow the external input signal to gate in the edge detector, the bit of the Data Direction Register corresponding to bit 0 of Port 2 must have been cleared (to zero).

To insure input capture in all cases, the width of an input pulse requires at least 2 Enable cycles.

(4) Timer Control/Status Register (TCSR) (\$0008)

This is an 8-bit register. All 8 bits are readable and the lower 5 bits may be written. The upper 3 bits are read-only, indicating the timer status information below.

- (a) Defined transition of the timer input signal causes the counter to transfer its data to the ICR.
- (b) A match has been found between the value in the free running counter and the output compare register (OCF).
- (c) The counter value reached to \$0000 as a result of counting-up (TOF).

Each flag may contain an individual enable bit in TCSR which controls whether or not an interrupt request may be

output to internal interrupt signal ( $\overline{IRQ}_2$ ). If the I-bit in Condition Code Register has been cleared, a priority vectored address occurs corresponding to each flag being set. Each bit is described as follows.

Timer Control/Status Register

7	6	5	4	3	2	1	0	
ICF	OCF	TOF	EICI	EOCI	ETOI	IEDG	OLVL	\$0008

- Bit 0 OLVL (Output Level); When a match is found in the value between the counter and the output compare register, this bit is transferred to the Port 2 bit 1. If the DDR corresponding to Port 2 bit 1 is set to "1", the value will appear on the output pin of Port 2 bit 1.
- Bit 1 IEDG (Input Edge); This bit control which transition of an input of Port 2 bit 0 will trigger the data transfer from the counter to the input capture register. The DDR corresponding to Port 2 bit 0 must be cleared in advance of using this function. When IEDG = 0, trigger takes place on a falling edge ("High"-to-"Low" transition). When IEDG = 1, trigger takes place on a rising edge ("Low"-to-"High" transition).
- Bit 2 ETOI (Enable Timer Overflow Interrupt); When set, this bit enables TOF interrupt to generate the interrupt request ( $\overline{IRQ}_2$ ) but when cleared, the interrupt is inhibited.
- Bit 3 EOCI (Enable Output Compare Interrupt); When set, this bit enables OCF interrupt to generate the interrupt request ( $\overline{IRQ}_2$ ), when cleared, the interrupt is inhibited.



Bit 4 EICI (Enable Input Capture Interrupt); When set, this bit enables ICF interrupt to generate the interrupt request ( $\overline{IRQ_2}$ ) but when cleared, the interrupt is inhibited.

Bit 5 TOF (Timer Overflow Flag); This read-only bit is set when the counter value is \$0000. It is cleared by CPU read of TCSR (with TOF set) followed by an CPU read of the counter (\$0009).

Bit 6 OCF (Output Compare Flag); This read-only bit is set when a match is found in the value between the output compare register and the counter. It is cleared by a read of TCSR (with OCF set) followed by an CPU write to the output compare register (\$000B or \$000C).

Bit 7 ICF (Input Capture Flag); The read-only bit is set when an input signal to edge detector makes a transition as defined by IEDG, and is cleared by a read of TCSR (with ICF set) followed by an CPU read of Input Capture Register (\$000D).

Each bit of Timer Control and Status Register is cleared during reset.

## 2.6 Serial Communication Interface

The HD6301V1 contains a full-duplex asynchronous Serial Communication Interface (SCI). SCI may select the several kinds of the data transmit rate and comprises a transmitter and a receiver which operate independently on each other but at the same data transmit rate. Both of transmitter and receiver communicate with the CPU by the data bus, and with the outside through Port 2 bit 2, 3 and 4. Description of hardware, software register is as follows.

### (1) Wake-Up Function

In typical multiprocessor applications the software protocol has the destination address at the initial byte of the message. The purpose of Wake-Up function is to have the non-selected MCU neglect the remainder of the message. Thus the non-selected MCU can inhibit

the all further interrupt process until the next message begins.

Wake-Up function is triggered by a ten consecutive "1"s which indicates an idle transmit line. Therefore software protocol needs an idle period between the messages. With this hardware feature, the non-selected MPU is re-enabled (or "wakes-up") for the appearing next message.

## (2) Programmable Option

The HD6301V1 has the following optional features provided for its Serial I/O. They are all programmable.

- . data format; standard mark/space (NRZ) start bit + 8 bit data + 1 stop bit
- . Clock source; external or internal
- . baud rate; one of 4 rates per given MCU E clock frequency or 1/8 of external clock
- . wake-up function; enabled or disabled
- . Interrupt requests; enabled or masked individually for transmitter and receive data registers
- . Clock Output; internal clock enabled or disabled to Port 2 bit 2
- . Port 2 (bits 3,4); dedicated or not dedicated to serial I/O individually for receiver and transmitter

## (3) Serial Communication Hardware

The serial communications hardware is controlled by 4 registers as shown in Figure 2-6-1. The registers include:

- . an 8-bit control/status register
- . a 4-bit rate/mode control register (write-only)
- . an 8-bit read-only receive data register
- . an 8-bit write-only transmit data register

Besides these 4 registers, Serial I/O utilizes Port 2 bit 3 (input) and bit 4 (output). Port 2 bit 2 can be used when an option is selected for the internal-clock-out or the external-clock-in.

## (4) Transmit/Receive Control Status Register (TRCSR)

TRCS Register consists of 8 bits which all may be read

while only bits 0 to 4 may be written. The register is initialized to \$20 on  $\overline{RES}$ . The bits of the TRCS register are defined as follows.

Transmit/Receive Control Status Register								
7	6	5	4	3	2	1	0	
RDRF	ORFE	TDRE	RIE	RE	TIE	TE	WU	
								ADDR: \$0011

- Bit 0 WU (Wake Up); Set by software and cleared by hardware on receipt of ten consecutive "1"s. It should be noted that RE flag has already set in advance of WU flag's set.
- Bit 1 TE (Transmit Enable); Set to produce preamble of ten consecutive "1"s and to enable the data of transmitter to output subsequently to the Port 2 bit 4 independently of its corresponding DDR value. When cleared, serial I/O affects nothing on Port 2 bit 4.
- Bit 2 TIE (Transmit Interrupt Enable); When this bit is set with TDRE (bit 5) set, it will permit an  $\overline{IRQ}_2$  interrupt. When cleared, TDRE interrupt is masked.
- Bit 3 RE (Receive Enable); When set, gates Port 2 bit 3 to input of receiver regardless of DDR value for this bit. When cleared, the serial I/O affects nothing on Port 2 bit 3.
- Bit 4 RIE (Receive Interrupt Enable); When this bit is set with bit 7 (RDRF) or a bit 6 (ORFE) set, it will permit an  $\overline{IRQ}_2$ . When cleared,  $\overline{IRQ}_2$  interrupt is masked.
- Bit 5 TDRE (Transmit Data Register Empty); When the data transfer is made from the Transmit Data Register to Output Shift Register, it is set by hardware. The bit is cleared by reading the status register (with TDRE set) and followed by writing the next new data into the Transmit Data Register. TDRE is initialized to 1 by  $\overline{RES}$ .
- Bit 6 ORFE (Over Run Framing Error); When overrun or framing error occurs (receive only), it is set by hardware. Over Run Error occurs if the attempt is made to transfer the new byte to the receive data

register with the RDRF set. Framing Error occurs when the bit counters are not synchronized with the boundary of the byte in the bit stream. The bit is cleared by reading the status register (with ORFE set) followed by reading the receive data register, or by  $\overline{RES}$ .

Bit 7 RDRF (Receive Data Register Full); It is set by hardware when the data transfer is made from the receive shift register to the receive data register. It is cleared by reading the status register (with RDRF set) and followed by reading the receive data register, or by  $\overline{RES}$ .

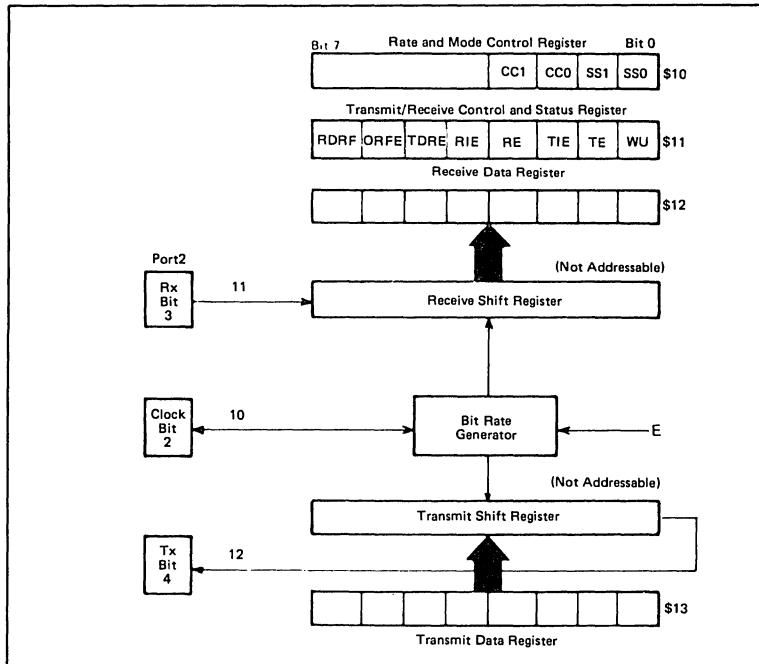


Fig. 2-6-1 SCI Register

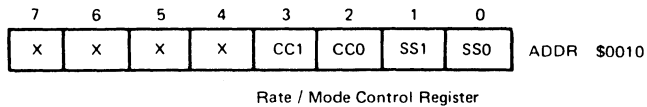


Table 2-6-1 SCI Bit Times and Transfer Rates

SS1 : SS0	XTAL	2.4576 MHz	4.0 MHz	4.9152MHz
	E	614.4 kHz	1.0 MHz	1.2288MHz
0 0	$E \div 16$	26 $\mu$ s/38,400 Baud	16 $\mu$ s/62,500 Baud	13 $\mu$ s / 76,800Baud
0 1	$E \div 128$	208 $\mu$ s/4,800 Baud	128 $\mu$ s/7812.5 Baud	104.2 $\mu$ s / 9,600Baud
1 0	$E \div 1024$	1.67ms/600 Baud	1.024ms/976.6 Baud	833.3 $\mu$ s / 1,200Baud
1 1	$E \div 4096$	6.67ms/150 Baud	4.096ms/244.1 Baud	3.333ms / 300Baud

Table 2-6-2 SCI Format and Clock Source Control

CC1, CC0	Format	Clock Source	Port 2 Bit 2	Port 2 Bit 3	Port 2 Bit 4
00	—	—	—	—	—
01	NRZ	Internal	Not Used***	**	**
10	NRZ	Internal	Output*	**	**
11	NRZ	External	Input	**	**

\* Clock output is available regardless of values for bits RE and TE.

\*\* Bit 3 is used for serial input if RE = "1" in TRCS.

Bit 4 is used for serial output if TE = "1" in TRCS.

\*\*\*It can be used as I/O port.

(5) Transmit Rate/Mode Control Register (RMCR)

The register controls the following serial I/O variables:

- Baud rate
- clock source
- Port 2, bit 2 function

It is 4-bit write-only register, cleared by  $\overline{RES}$ . The 4 bits are considered as a pair of 2-bit fields. The lower 2 bits control the bit rate of internal clock while the upper 2 bits control the clock select logic.

Bit 0 SS0 }  
 Bit 1 SS1 } Speed Select

These bits select the Baud rate for the internal clock. The selectable 4 rates are function of E clock frequency within the MCU. Table 2-6-1 lists the available Baud Rates.

Bit 2 CC0 }  
 Bit 3 CC1 } Clock Control/Format Select

These bits control the clock select logic. Table 2-6-2 defines the bit field.

(6) Internally Generated Clock

When using the internal clock for the SCI externally, the followings should be noted.

- The values of RE and TE have no effect.
- CC1, CC0 must be set to "10".
- The maximum clock rate is E/16.
- The clock is once the bit rate.

(7) Externally Generated Clock

When supplying an external clock for the SCI, the followings should be noted.

- The CC1, CC0, in the Rate and Mode Control Register must be set to "11" (See Table 2-6-2).
- The external clock frequency must be set to 8 times the desired baud rate.
- The maximum external clock frequency is half of E clock.

#### (8) Serial Operations

The serial I/O hardware must be initialized by the HD6301V1 software prior to operation. The sequence is normally as follows.

- Writing the desired operation control bits to the Rate and Mode Control Register.
- Writing the desired operation control bits to the TRCS Register.

If using Port 2 bit 3, 4 for serial I/O exclusively, TE, RE bits may be preserved set. When TE, RE bit cleared during SCl operation, and subsequently set again, it should be noted that the setting of TE, RE must refrain for at least one bit time of the current baud rate. If set within one bit time, there may be the case where the initializing of internal function for transmit and receive does not take place.

#### (9) Transmit Operation

Data transmission is enabled by the TE bit in the TRCS register. When set, outputs the data of the serial transmit shift register to Port 2 bit 4 which is unconditionally configured as an output irrespectively of corresponding DDR value.

Following  $\overline{\text{RES}}$ , the user should configure both the RMC register and the TRCS register for desired operation. Setting the TE bit during this procedure causes a transmission of ten-bit preamble of "1"s. Following the preamble, internal synchronization is established and the transmitter section is ready to operate. Then either of the followings operates.

- (a) If the transmit data register is empty (TDRE = 1), the consecutive "1"s are transmitted indicating an idle lines.
- (b) If the data has been loaded into the Transmit Data Register (TDRE = 0), it is transferred to the output shift register and data transmission begins.

During the data transfer, the "0" start bit is first transferred. Next the 8-bit data (beginning at bit 0) and the "1" stop bit. When the transmit data register has been empty, the hardware sets the TDRE flag bit: If the CPU fails to respond to the flag within the proper time, TDRE is preserved set and then a "1" will be sent (instead of a "0" at start bit time) and more "1"s will be set consecutively until the data is supplied to the data register. While the TDRE remains a "1", no "0" will be sent.

#### (10) Receive Operation

The receive operation is enabled by the RE bit. The serial input is connected with Port 2 bit 3. The receive operation is determined by the contents of the TRCS and RMC register. The received bit stream is synchronized by the first "0" (space). During 10-bit time, the data is strobed approximately at the center of each bit. If the tenth bit is not "1" (stop bit), the system assumes a framing error and the ORFE is set. (RDRF is not set.)

If the tenth bit is "1", the data is transferred to the receive data register, with the interrupt flag set (RDRF). If the tenth bit of the next data is received, however, still RDRF is preserved set, then ORFE is set indicating that an overrun error has occurred.

After the CPU read of the status register as a response to RDRF flag or ORFE flag followed by the CPU read of the receive data register, RDRF or ORFE will be cleared.

#### (11) Timer, SCI Status Flag

The set and reset condition of each status flag of timer and SCI is shown in Table 2-7.

## 2.7 Interrupts

The HD6301V1 has two external interrupt pins ( $\overline{\text{NMI}}$ ,  $\overline{\text{IRQ}}_1$ ) and 8 internal interrupt source (Soft-TRAP, SWI, Timer-ICF, OCF, TOF, SCI-RDRF, ORFE, TDRE). The features of these interrupt are detailed in the following paragraphs.

### (1) Non maskable Interrupt ( $\overline{\text{NMI}}$ )

When the input signal of this pin is recognized to fall, NMI sequence starts. The current instruction is continued to the last if  $\overline{\text{NMI}}$  signal is detected as well as the following  $\overline{\text{IRQ}}_1$  interrupt. Interrupt mask bit in Condition Code Register has no effect on NMI interrupt. In response to NMI interrupt, the contents of Program Counter, Index Register Accumulators, and Condition Code Register are stored on the stack. On completion of this sequence, vectoring address \$FFFC and \$FFFD will occur to load the contents to the program counter and branch to a non maskable interrupt service routine. Inputs  $\overline{\text{IRQ}}_1$ , and  $\overline{\text{NMI}}$  are hardware interrupt lines sampled by internal clock. After the execution of instructions, start the interrupt routine in synchronization with E.

### (2) Interrupt Request ( $\overline{\text{IRQ}}_1$ )

This is the level-sensitive pin which requests an internal interrupt sequence to the CPU. At interrupt request, the CPU will complete the current instruction before the acceptance of the request. Unless the interrupt mask in the condition code register is set, the CPU starts an interrupt sequence; if set, the interrupt request will be ignored.

When the sequence starts, the contents of Program Counter, Index Register, Accumulator, Condition Code Register are stored on the stack. Then the CPU sets the interrupt mask bit and will not acknowledge the maskable request.

At the end of the cycle, the CPU generates 16 bit vectoring addresses indicating memory addresses \$FFF8 and \$FFF9, and locates the contents in Program Counter to branch to an interrupt service routine.




### (3) Internal interrupts

For an internal interrupt requested from the timer or SCI, an internal interrupt signal  $\overline{IRQ}_2$  is activated. This interrupt is identical to  $\overline{IRQ}_1$  except that it uses vector addresses \$FFF0 through \$FFF7. The  $\overline{IRQ}_1$  has the priority to the  $\overline{IRQ}_2$  when interrupt requests have taken place at the same time. When the interrupt mask bit in the condition code register is set, both interrupts are inhibited.

The SWI is an instruction which requests an interrupt by software. The state of CCR mask bit doesn't influence the SWI. If an address error or operation code error (see "2.13 Error Processing") occurs, TRAP takes place whose priority is next to the reset. In the case of TRAP, CPU starts the interrupt sequence regardless of the state of the mask bit. The vectors corresponding to this interrupt are \$FFEE and \$FFEF. The memory map for interrupt vectors is shown in Table 2-7-1 and the interrupt sequence are shown in Fig. 2-7-1. Fig. 2-7-2 shows the logic of the interrupt circuit.

Table 2-7-1 Interrupt Vectoring Memory Map

	Vector		Interrupt
	MSB	LSB	
Highest Priority  Lowest Priority	FFFE	FFFF	$\overline{RES}$
	FFEE	FFEF	TRAP
	FFFC	FFFD	$\overline{NMI}$
	FFFA	FFFB	Software Interrupt (SWI)
	FFF8	FFF9	$\overline{IRQ}_1$ (or $\overline{IS3}$ )
	FFF6	FFF7	ICF (Timer Input Capture)
	FFF4	FFF5	OCF (Timer Output Compare)
	FFF2	FFF3	TOF (Timer Overflow)
	FFF0	FFF1	SCI (RDRF+ORFE+TDRE)

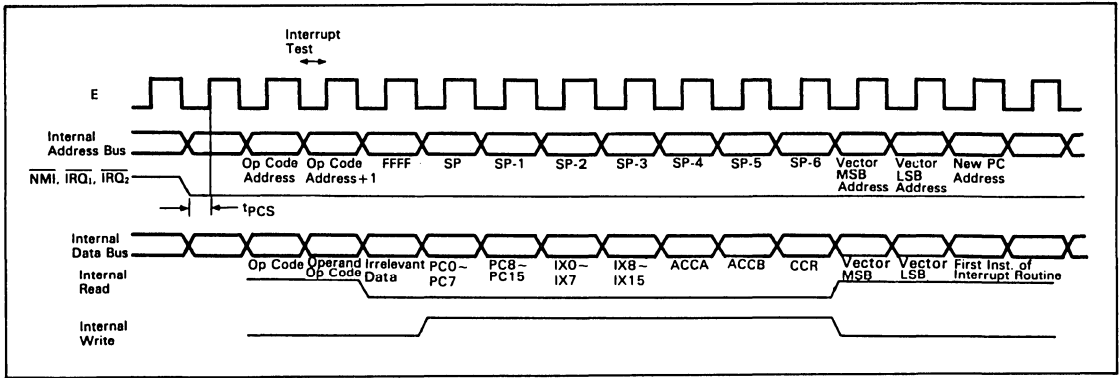


Fig. 2-7-1 Interrupt Sequence

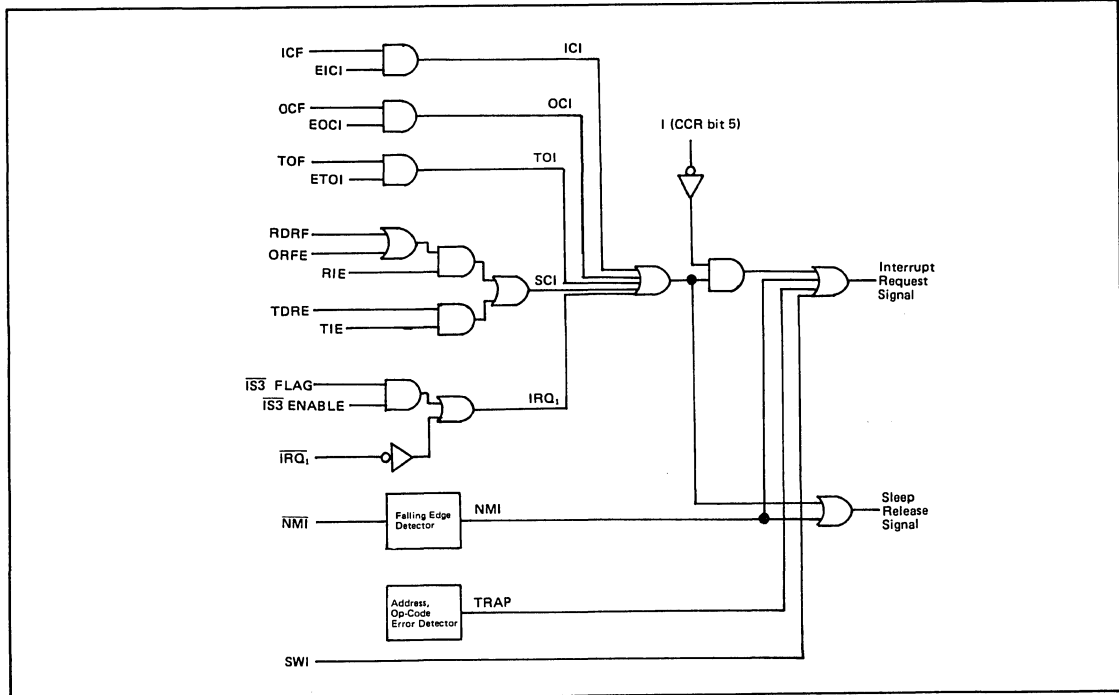


Fig. 2-7-2 Logic of Interrupt Circuit

## 2.8 Reset ( $\overline{\text{RES}}$ )

This input is used to reset the MCU and start it from a power-off condition. During Power-on,  $\overline{\text{RES}}$  pin must be held "Low" for at least 20ms. To reset the MCU during system operation, it must be held "Low" at least 3 system clock cycles. At the 3rd cycle during "Low" level, all address buses become "High impedance" while  $\overline{\text{RES}}$  is "Low". Detecting "High" level, MCU operates as followings.

- (1) Latch I/O Port 2 bits 2, 1, 0 into bits PC2, PC1, PC0 of mode register.
- (2) Put the contents (=start address) of the last two addresses (\$FFFE, \$FFFF) into the program counter and start the program from this address. (Refer to Table 2-7-1)
- (3) Set the interrupt mask bit. For the CPU to recognize the maskable interrupts  $\overline{IRQ}_1$  and  $\overline{IRQ}_2$ , this bit should be cleared in advance. Fig. 2-8-1 shows the reset timing, and Table 2-8-1 shows the pin condition during reset.

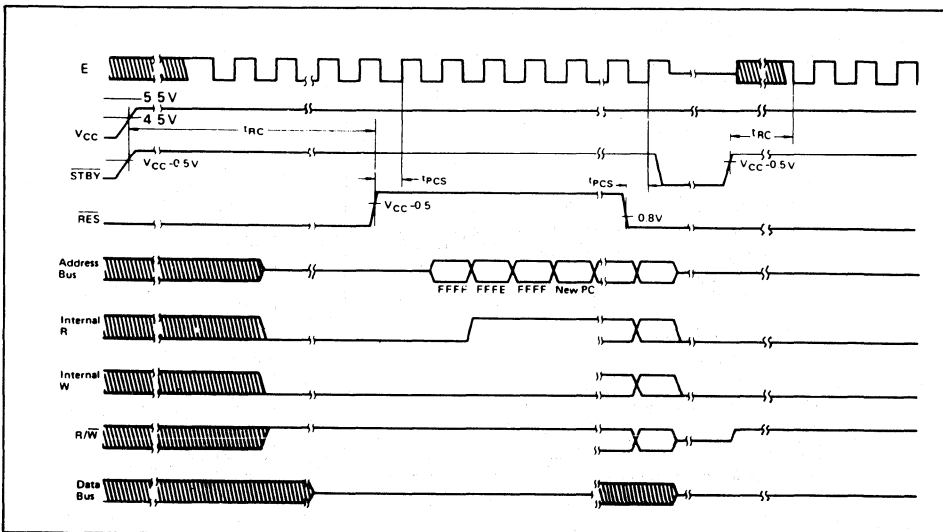


Fig. 2-8-1 Reset Timing

Table 2-8-1 Pin Condition during RESET

Mode Pin	0	1	2,4	5	6	7
Port 1 P <sub>10</sub> ~P <sub>17</sub>	High impedance (input)	←	←	←	←	←
Port 2 P <sub>20</sub> ~P <sub>24</sub>	High impedance (input)	←	←	←	←	←
Port 3 P <sub>30</sub> ~P <sub>37</sub>	$\overline{E}$ : "1" output E: "1" output (High impedance)	High impedance	$\overline{E}$ : "1" output E: "1" output (High impedance)	High impedance	$\overline{E}$ : "1" output E: "1" output (High impedance)	High impedance (input)
Port 4 P <sub>40</sub> ~P <sub>47</sub>	High impedance (input)	←	←	←	←	←
SC <sub>2</sub>	"1" output (READ)	←	←	←	←	"1" output
SC <sub>1</sub>	$\overline{E}$ : "1" output E: High impedance	←	←	"1" output	$\overline{E}$ : "1" output E: High impedance	High impedance (input)

## 2.9 Oscillator

XTAL, EXTAL pins interface with an AT-cut parallel resonant crystal. Divide-by-four circuit is on chip, so if 4 MHz crystal oscillator is used, the system clock is 1 MHz for example. EXTAL pin can be driven by the external clock with 45% to 55% duty. The system clock which is one fourth frequency of the external clock is generated in the LSI. When using the external clock, XTAL pin should be open. Fig. 2-9-1 shows examples of connection circuit.

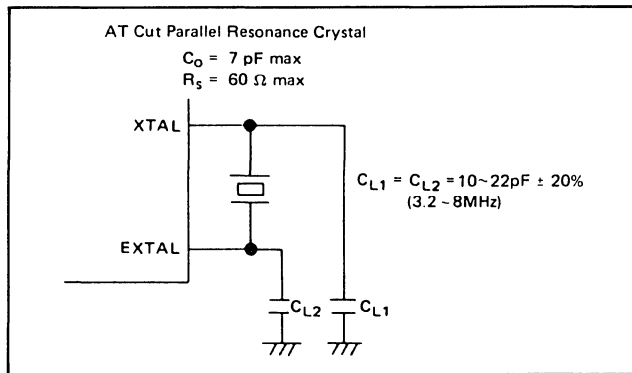


Fig. 2-9-1 Crystal Interface

## 2.10 Strobe Signals

Two pins,  $SC_1$  (39 pin) and  $SC_2$  (38 pin) are used as the strobe signals in each mode. Followings are applied only to Port 3 in single chip mode.

### (1) Input Strobe ( $\overline{IS3}$ ) ( $SC_1$ )

This signal controls  $\overline{IS3}$  interrupt and the latch of Port 3. When the falling edge of this signal is detected, the flag of Port 3 Control Status Register is set.

For respective bits of Port 3 Control Status Register, see the "2.4 I/O Ports" section.

### (2) Output Strobe ( $\overline{OS3}$ ) ( $SC_2$ )

This signal is used by the processor to strobe an external device, indicating effective data is on the I/O pins. The timing chart for Output Strobe are shown in figure 5-5.

The following pins are available for Expanded Modes.

(3) Read/Write ( $R/\overline{W}$ ) ( $SC_2$ )

This TTL compatible output signal indicates peripheral and memory devices whether the CPU is in Read ("High"), or in Write ("Low"). The normal stand-by state of this signal is Read ("High"). This output can drive one TTL load and 90pF capacitance.

(4) I/O Strobe ( $\overline{IOS}$ ) ( $SC_1$ )

In expanded non multiplexed mode 5 of operation,  $\overline{IOS}$  becomes "Low" when  $A_9$  through  $A_{15}$  are "0"s and  $A_8$  is "1". This allows external access of 256 addresses from \$0100 to \$01FF in memory. The timing chart is shown in Figure 5-2.

(5) Address Strobe (AS) ( $SC_1$ )

In the expanded multiplexed mode of operation, address strobe is output to this pin. This signal is used to latch the lower 8 bits addresses multiplexed with data at Port 3 and to control the 8-bit latch by address strobe as shown in Figure 2-1-4. Thereby, I/O Port 3 can become data bus during E pulse. The timing chart of this signal is shown in Figure 5-1.

## 2.11 RAM Control Register

The register assigned to the address \$0014 gives a status information about standby RAM.

RAM Control Register

	7	6	5	4	3	2	1	0
\$0014	STBY PWR	RAME	x	x	x	x	x	x

- Bit 0 Not used.
- Bit 1 Not used.
- Bit 2 Not used.
- Bit 3 Not used.
- Bit 4 Not used.
- Bit 5 Not used.
- Bit 6 RAM Enable.

Using this control bit, the user can disable the RAM. RAM Enable bit is set on the positive edge of  $\overline{RES}$  and RAM is enabled. With the program control, it is capable of writing "1" or "0". With the disabled RAM (logic "0"),

the RAM address becomes external address and the CPU may read the data from the outside memory.

#### Bit 7 Standby Bit

This bit is cleared when VCC is not provided in standby mode. This bit is a read/write status flag that user can read. If this bit is preserved set, indicating that VCC voltage is applied and the data in the RAM is valid.

### 2.12 Low Power Consumption Mode

The HD6301V1 has two low power consumption modes; sleep and standby.

#### (1) Sleep Mode

On execution of SLP instruction, the CPU is brought to the sleep mode. In the sleep mode, the CPU stops its operation, but the contents of the register in the CPU are retained. In this mode, the peripherals of MPU will remain operational. So the operations such as transmit and receive of the SCI data and counter may keep on functioning. In this mode, the power consumption is one-sixth that of operating condition.

The MCU returns from this mode by interrupt,  $\overline{\text{RES}}$ ,  $\overline{\text{STBY}}$ . The  $\overline{\text{RES}}$  resets the MCU and the  $\overline{\text{STBY}}$  brings it into the standby mode (This will be mentioned later). When the CPU acknowledges an interrupt request, it cancels the sleep mode, returns to the operation mode and branches to the interrupt routine. When the CPU masks the interrupt, it cancels the sleep mode and executes the next instruction. However, for example, if the timer 1 or 2 prohibits a timer interrupt, the CPU doesn't cancel the sleep mode because of no interrupt request.

This sleep mode is available to reduce the power consumption for a system with no need of the HD6301V1's consecutive operation.

Please refer to Table 2-12-1 for other pins except VCC, clock pin, input-only pin, E clock pin (their function are the same as operating condition).

## (2) Standby Mode

The HD6301V1 stops all the clocks and goes into the reset state with  $\overline{STBY}$  "Low". In this mode, the power consumption is reduced conspicuously.

In the standby mode, the power is supplied to the HD6301V1, so the contents of RAM are retained. The standby mode should escape by bringing  $\overline{STBY}$  "High".

Transitions among the active mode, sleep mode, standby mode and reset are shown in Fig. 2-12-2.

Table 2-12-1 Pin Condition in Sleep Mode

Pin \ Mode		0	1	2,4	5	6	7
Port 1 P <sub>10</sub> ~P <sub>17</sub>	Function	I/O Port	Lower Address Bus	I/O Port	←	←	←
	Condition	Keep the condition just before sleep	Output "1"	Keep the condition just before sleep	←	←	←
Port 2 P <sub>20</sub> ~P <sub>24</sub>	Function	I/O Port	←	←	←	←	←
	Condition	Keep the condition just before sleep	←	←	←	←	←
Port 3 P <sub>30</sub> ~P <sub>37</sub>	Function	$\overline{E}$ :Lower Address Bus E:Data Bus	Data Bus	$\overline{E}$ :Lower Address Bus E:Data Bus	Data Bus	$\overline{E}$ :Lower Address Bus E:Data Bus	I/O Port
	Condition	$\overline{E}$ :Output "1" E:High impedance	High impedance	$\overline{E}$ :Output "1" E:High impedance	High impedance	$\overline{E}$ :Output "1" E:High impedance	Keep the condition just before sleep
Port 4 P <sub>40</sub> ~P <sub>47</sub>	Function	Upper Address	←	←	Lower Address Bus or Input Port	Upper Address Bus or Input Port	I/O Port
	Condition	Output "1"	←	←	Address Bus: Output "1" Port:Keep the condition just before sleep	←	Keep the condition just before sleep
SC <sub>2</sub>		Output "1" (Read Condition)	←	←	←	←	Output "1"
SC <sub>1</sub>		Output Address Strobe	←	←	Output "1"	Output Address Strobe	Input Pin

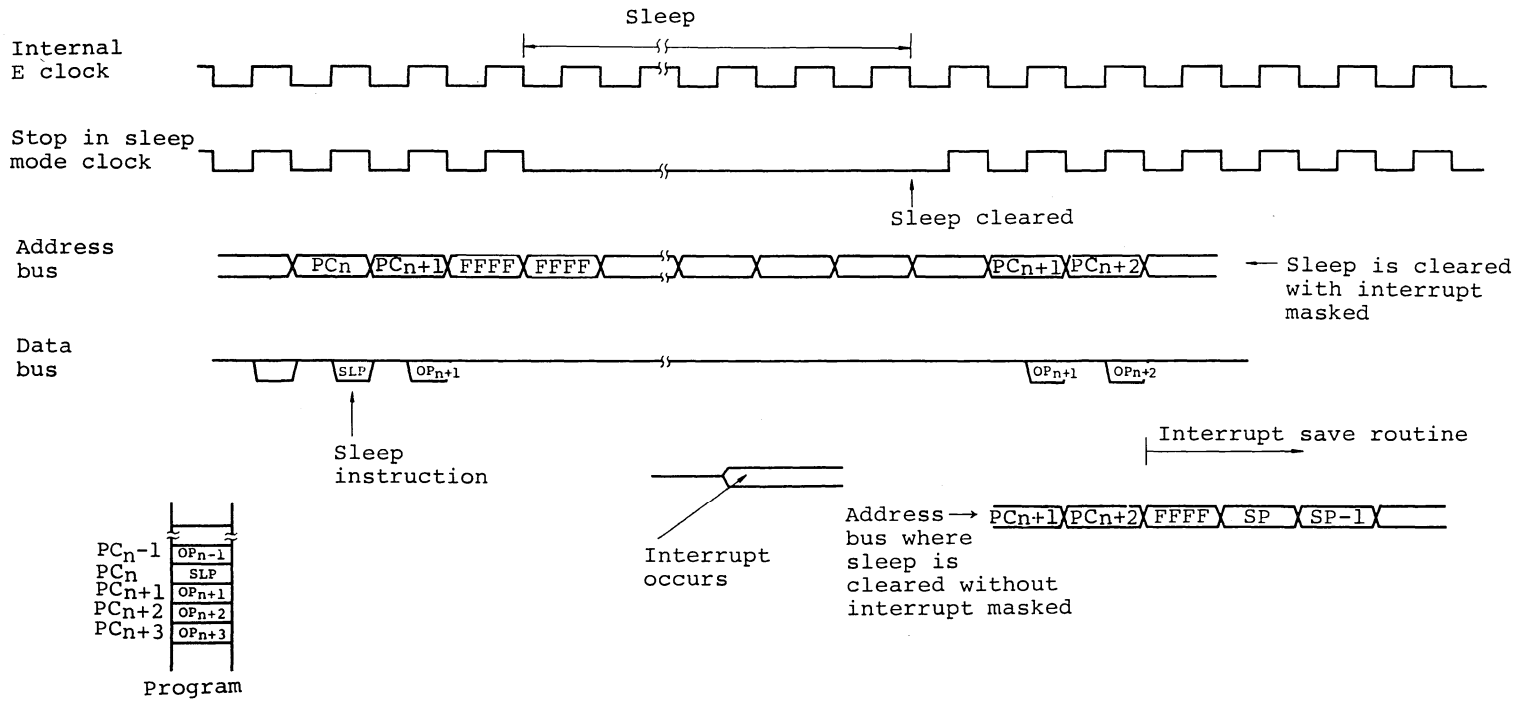


Fig. 2-12-1 Sleep Instruction Timing Chart





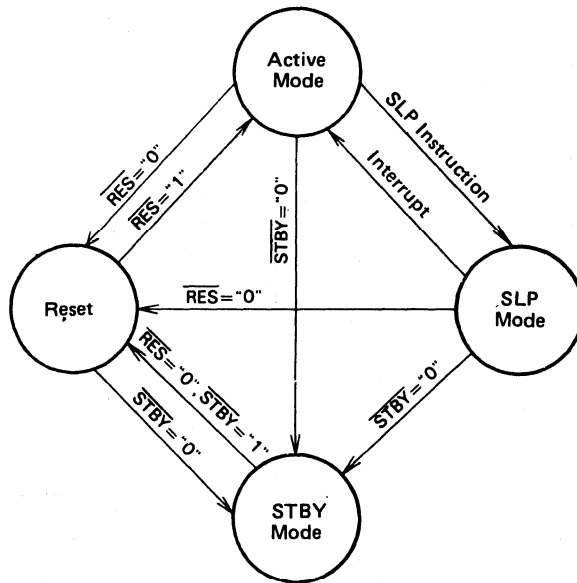


Fig. 2-12-2 Transitions among Active Mode, Standby Mode Sleep Mode, and Reset

## 2.13 TRAP Function

The CPU generates an interrupt with the highest priority (TRAP) when fetching an undefined instruction or an instruction from non-memory space. The TRAP prevents the system-burst caused by noise or a program error.

### (1) Op-Code Error

When fetching an undefined op-code, the CPU saves register as well as a normal interrupt and branches to the TRAP (\$FFEE, \$FFEF). This has the priority next to reset.

### (2) Address Error

When an instruction is fetched from excluding internal ROM, RAM, or an external memory area, the MCU generates the same interrupt as op-code error. If the instruction is fetched from external memory area without memory devices, this function is not applicable.

Table 2-13-1 shows addresses where an address error occurs to each mode. This function is available only for the instruction fetch, and is not applicable to the access of normal data read/write.

Table 2-13-1 Addresses Applicable to Address Errors

Mode	0	1	4	5	6	7
Address	\$0000	\$0000	\$0000	\$0000	\$0000	\$0000
	}	}	}	}	}	}
	\$001F	\$001F	\$001F	\$007F	\$001F	\$007F
				\$0200		\$0100
				}		}
			\$EFFF		\$EFFF	

### 3. INSTRUCTIONS

The HD6301V1 Provides object code upward. Besides having object code compatible with the HD6801 series, the HD6301V1 the predecessor with additional instructions; enhances bit control instructions (AIM, EIM, OIM, TIM), index/accumulator exchange instruction (XGDX), and sleep instruction (SLP). These new instructions improve programming efficiency.

#### 3.1 Addressing Modes

The HD6301V1 provides seven addressing modes. The adequate selection of these addressing mode will permit to implement an efficient and easy programming.

The addressing mode is determined by an instruction type and code. The addressing mode for each instruction is shown in Table 3-2-1 to 3-2-4 with execution time counted by the machine cycles. When the clock frequency is 4 MHz, the machine cycle time will be microseconds.

##### Accumulator (ACCX) Addressing

Only an accumulator is addressed. Either accumulator A or B is selected. This is a one-byte instruction.

##### Immediate Addressing

In this mode, the data is stored in the second byte of the instruction except that LDS and LDX, store a data in the second and the third byte exceptionally. These are two or three-byte instructions.

##### Direct Addressing

In this mode, the second byte of an instruction indicates the address where the data is stored. 256 bytes (\$0 through \$255) can be addressed directly. Execution times can be reduced by storing data in these locations. In configurating system, it is recommended that these locations should be RAM for users' data area. These are two-byte instructions, while the AIM, OIM, EIM and TIM are three-byte.

### Extended Addressing

In this mode, the second byte indicates the upper 8 bits addresses where the data is stored, and the third byte indicates the lower 8 bits. These are three-byte instructions.

### Indexed Addressing

In this mode, the contents of the second byte and the lower 8 bits in the Index Register are added. As for AIM, OIM, EIM and TIM instructions, the contents of the third byte and the lower 8 bits in the Index Register are added. In addition, this carry is added to the upper 8 bits in the Index Register. The result is used for addressing memory. The modified address is held in the Temporary Address Register, so there is no change to the contents of the Index Register. These are two-byte instructions, while AIM, OIM, EIM and TIM are three-byte.

### Implied Addressing

In this mode, the instruction itself gives the address. That is, the instruction addresses an accumulator, stack pointer, index register, etc. This is a one-byte instruction.

### Relative Addressing

In this mode, the contents of the second byte and the lower 8 bits in the program counter are added. The carry or borrow is added to the upper 8 bits. So addressing from -126 to +129 bytes of the current instruction is enabled. These are two-byte instructions.

### 3.2 Instruction Set

The HD6301V1 has an upward object code compatible with the HD6801 to utilize all instruction sets of the HMCS6800. The execution time of the key instruction is reduced to increase the system through-put. In addition, the bit manipulation instruction, the exchange instruction of the index and the accumulator, the sleep instruction are added. The followings are described here.

- Accumulator and memory manipulation instructions (See Table 3-2-1).
- Additional instructions.
- Index register and stack manipulation instructions (See Table 3-2-2).
- Jump and branch instructions (See Table 3-2-3).
- Condition code register manipulation instructions (See Table 3-2-4).
- Op-code map (See Table 3-2-5).

Table 3-2-1 Accumulator, Memory Manipulation Instructions

Operations	Mnemonic	Addressing Modes												Boolean/ Arithmetic Operation	Condition Code Register																		
		IMMED			DIRECT			INDEX			EXTEND				IMPLIED			5	4	3	2	1	0										
		OP	~	#	OP	~	#	OP	~	#	OP	~	#		OP	~	#	H	I	N	Z	V	C										
Add	ADDA	8B	2	2	9B	3	2	AB	4	2	BB	4	3											A + M → A	↑	•	↑	↑	↑	↑			
	ADDB	CB	2	2	DB	3	2	EB	4	2	FB	4	3												B + M → B	↑	•	↑	↑	↑	↑		
Add Double	ADDD	C3	3	3	D3	4	2	E3	5	2	F3	5	3												A : B + M : M + 1 → A : B	•	•	↑	↑	↑	↑		
Add Accumulators	ABA													1B	1	1									A + B → A	↑	•	↑	↑	↑	↑		
Add With Carry	ADCA	89	2	2	99	3	2	A9	4	2	B9	4	3												A + M + C → A	↑	•	↑	↑	↑	↑		
	ADCB	C9	2	2	D9	3	2	E9	4	2	F9	4	3												B + M + C → B	↑	•	↑	↑	↑	↑		
AND	ANDA	84	2	2	94	3	2	A4	4	2	B4	4	3												A · M → A	•	•	↑	↑	↑	R	•	
	ANDB	C4	2	2	D4	3	2	E4	4	2	F4	4	3												B · M → B	•	•	↑	↑	↑	R	•	
Bit Test	BIT A	85	2	2	95	3	2	A5	4	2	B5	4	3												A · M	•	•	↑	↑	↑	R	•	
	BIT B	C5	2	2	D5	3	2	E5	4	2	F5	4	3												B · M	•	•	↑	↑	↑	R	•	
Clear	CLR							6F	5	2	7F	5	3												00 → M	•	•	↑	↑	↑	R	R	
	CLRA													4F	1	1									00 → A	•	•	↑	↑	↑	R	R	
	CLRB													5F	1	1									00 → B	•	•	↑	↑	↑	R	R	
Compare	CMPA	81	2	2	91	3	2	A1	4	2	B1	4	3												A - M	•	•	↑	↑	↑	↑	↑	
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3												B - M	•	•	↑	↑	↑	↑	↑	
Compare Accumulators	CBA													11	1	1									A - B	•	•	↑	↑	↑	↑	↑	
Complement, 1's	COM							63	6	2	73	6	3												M → M	•	•	↑	↑	↑	R	S	
	COMA													43	1	1									A → A	•	•	↑	↑	↑	R	S	
	COMB													53	1	1									B → B	•	•	↑	↑	↑	R	S	
Complement, 2's (Negate)	NEG							60	6	2	70	6	3												00 - M → M	•	•	↑	↑	↑	(1)	(2)	
	NEGA													40	1	1									00 - A → A	•	•	↑	↑	↑	(1)	(2)	
	NEGB													50	1	1									00 - B → B	•	•	↑	↑	↑	(1)	(2)	
Decimal Adjust, A	DAA													19	2	1								Converts binary add of BCD characters into BCD format	•	•	↑	↑	↑	↑	(3)		
Decrement	DEC							6A	6	2	7A	6	3												M - 1 → M	•	•	↑	↑	↑	(4)	•	
	DECA													4A	1	1									A - 1 → A	•	•	↑	↑	↑	(4)	•	
	DECB													5A	1	1									B - 1 → B	•	•	↑	↑	↑	(4)	•	
Exclusive OR	EORA	88	2	2	98	3	2	A8	4	2	B8	4	3												A ⊕ M → A	•	•	↑	↑	↑	↑	R	•
	EORB	C8	2	2	D8	3	2	E8	4	2	F8	4	3												B ⊕ M → B	•	•	↑	↑	↑	↑	R	•
Increment	INC							6C	6	2	7C	6	3												M + 1 → M	•	•	↑	↑	↑	(5)	•	
	INCA													4C	1	1									A + 1 → A	•	•	↑	↑	↑	(5)	•	
	INCB													5C	1	1									B + 1 → B	•	•	↑	↑	↑	(5)	•	
Load Accumulator	LDAA	86	2	2	96	3	2	A6	4	2	B6	4	3												M → A	•	•	↑	↑	↑	R	•	
	LDAB	C6	2	2	D6	3	2	E6	4	2	F6	4	3												M → B	•	•	↑	↑	↑	R	•	
Load Double Accumulator	LDD	CC	3	3	DC	4	2	EC	5	2	FC	5	3												M + 1 → B, M → A	•	•	↑	↑	↑	R	•	
Multiply Unsigned	MUL													3D	7	1									A × B → A : B	•	•	•	•	•	(1)		
OR, Inclusive	ORAA	8A	2	2	9A	3	2	AA	4	2	BA	4	3												A + M → A	•	•	↑	↑	↑	R	•	
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3												B + M → B	•	•	↑	↑	↑	R	•	
Push Data	PSHA													36	4	1										A → Msp, SP - 1 → SP	•	•	•	•	•	•	
	PSHB													37	4	1										B → Msp, SP - 1 → SP	•	•	•	•	•	•	
Pull Data	PULA													32	3	1										SP + 1 → SP, Msp → A	•	•	•	•	•	•	
	PULB													33	3	1										SP + 1 → SP, Msp → B	•	•	•	•	•	•	
Rotate Left	ROL							69	6	2	79	6	3													M	•	•	↑	↑	↑	(6)	↑
	ROLA													49	1	1										A	•	•	↑	↑	↑	(6)	↑
	ROLB													59	1	1										B	•	•	↑	↑	↑	(6)	↑
Rotate Right	ROR							66	6	2	76	6	3													M	•	•	↑	↑	↑	(6)	↑
	RORA													46	1	1										A	•	•	↑	↑	↑	(6)	↑
	RORB													56	1	1										B	•	•	↑	↑	↑	(6)	↑

Note) Condition Code Register will be explained in Note of Table 3-2-4.

(to be continued)

Table 3-2-1 Accumulator, Memory Manipulation Instructions

Operations	Mnemonic	Addressing Modes												Boolean/ Arithmetic Operation	Condition Code Register									
		IMMED			DIRECT			INDEX			EXTEND				IMPLIED			5	4	3	2	1	0	
		OP	~	#	OP	~	#	OP	~	#	OP	~	#		OP	~	#	H	I	N	Z	V	C	
Shift Left Arithmetic	ASL								68	6	2	78	6	3		•	•	•	•	Ⓢ	•			
	ASLA													48	1	1		•	•	•	•	Ⓢ	•	
	ASLB													58	1	1		•	•	•	•	Ⓢ	•	
Double Shift Left, Arithmetic	ASLD													05	1	1		•	•	•	•	Ⓢ	•	
Shift Right Arithmetic	ASR							67	6	2	77	6	3		•	•	•	•	Ⓢ	•				
	ASRA													47	1	1		•	•	•	•	Ⓢ	•	
	ASRB													57	1	1		•	•	•	•	Ⓢ	•	
Shift Right Logical	LSR							64	6	2	74	6	3		•	•	•	•	R	Ⓢ	•			
	LSRA													44	1	1		•	•	•	•	R	Ⓢ	•
	LSRB													54	1	1		•	•	•	•	R	Ⓢ	•
Double Shift Right Logical	LSRD													04	1	1		•	•	•	•	R	Ⓢ	•
Store Accumulator	STAA						97	3	2	A7	4	2	B7	4	3	A → M	•	•	•	•	R	•		
	STAB						D7	3	2	E7	4	2	F7	4	3	B → M	•	•	•	•	R	•		
Store Double Accumulator	STD						DD	4	2	ED	5	2	FD	5	3	A → M B → M + 1	•	•	•	•	R	•		
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3	A - M → A	•	•	•	•	•	•	•			
	SUBB	C0	2	2	D0	3	2	E0	4	2	F0	4	3	B - M → B	•	•	•	•	•	•	•			
Double Subtract	SUBD	83	3	3	93	4	2	A3	5	2	B3	5	3	A : B - M : M + 1 → A : B	•	•	•	•	•	•	•			
Subtract Accumulators	SBA													10	1	1	A - B → A	•	•	•	•	•	•	
Subtract With Carry	SBCA	B2	2	2	92	3	2	A2	4	2	B2	4	3	A - M - C → A	•	•	•	•	•	•	•			
	SBCB	C2	2	2	D2	3	2	E2	4	2	F2	4	3	B - M - C → B	•	•	•	•	•	•	•			
Transfer Accumulators	TAB													16	1	1	A → B	•	•	•	•	R	•	
	TBA													17	1	1	B → A	•	•	•	•	R	•	
Test Zero or Minus	TST							6D	4	2	7D	4	3	M - 00	•	•	•	•	•	•	R	R		
	TSTA													4D	1	1	A - 00	•	•	•	•	•	R	R
	TSTB													5D	1	1	B - 00	•	•	•	•	•	R	R
And Immediate	AIM						71	6	3	61	7	3	M-IMM → M	•	•	•	•	•	•	R	•			
OR Immediate	OIM						72	6	3	62	7	3	M+IMM → M	•	•	•	•	•	•	R	•			
EOR Immediate	EIM						75	6	3	65	7	3	M ⊕ IMM → M	•	•	•	•	•	•	R	•			
Test Immediate	TIM						7B	4	3	6B	5	3	M-IMM	•	•	•	•	•	•	R	•			

Note) Condition Code Register will be explained in Note of Table 3-2-4.

Additional Instructions

In addition to the HD6801 Instruction Set, the HD6301V1 has the following new instructions:

AIM --- (M) • (IMM) → (M)

Executes "AND" operation to the immediate data and the memory contents and stores the result in the memory.

OIM --- (M) + (IMM) → (M)

Executes "OR" operation to the immediate data and the memory contents and stores the result in the memory.

EIM --- (M) ⊕ (IMM) → (M)

Executes "EOR" operation to the immediate data and the memory contents and stores the result in the memory.










Table 3-2-5 OP-Code Map

OP CODE						ACC A	ACC B	IND	EXT DIR*	ACCA or SP				ACCB or X					
HI	LO	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111		
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
0000	0	/	SBA	BRA	TSX	NEG				SUB								0	
0001	1	NOP	CBA	BRN	INS	AIM				CMP								1	
0010	2	/	/	BHI	PULA	OIM				SBC								2	
0011	3	/	/	BLS	PULB	COM				SUBD		ADD							3
0100	4	LSRD	/	BCC	DES	LSR				AND								4	
0101	5	ASLD	/	BCS	TXS	EIM				BIT								5	
0110	6	TAP	TAB	BNE	PSHA	ROR				LDA								6	
0111	7	TPA	TBA	BEQ	PSHB	ASR				STA		STA							7
1000	8	INX	XGDX	BVC	PULX	ASL				EOR								8	
1001	9	DEX	DAA	BVS	RTS	ROL				ADC								9	
1010	A	CLV	SLP	BPL	ABX	DEC				ORA								A	
1011	B	SEV	ABA	BMI	RTI	TIM				ADD								B	
1100	C	CLC	/	BGE	PSHX	INC				CPX		LDD							C
1101	D	SEC	/	BLT	MUL	TST				BSR	JSR		STD						D
1110	E	CLI	/	BGT	WAI	JMP				LDS		LDX							E
1111	F	SEI	/	BLE	SWI	CLR				STS		STX							F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		

UNDEFINED OP CODE   
 \* Only each instructions of AIM, OIM, EIM, TIM

3

### 3.3 Instruction Execution Cycles

In the HMCS6800 series, the execution cycle of each instruction is counted from the start of the op-code fetch.

The HD6301V1 employs a mechanism of the pipeline control for the instruction fetch and the subsequent instruction fetch is performed during the current instruction being executed.

Therefore, the method to count instruction cycles used in the HMCS6800 series cannot be applied to the instruction cycles such as MULT, PULL, DAA and XGDX in the HD6301V1.

Table 3-3-1, provides the information about the relationship among each data on the Address Bus, Data Bus, and R/ $\bar{W}$  status in cycle by cycle basis during the execution of each instruction.

Table 3-3-1 Cycle by Cycle Operation

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/ $\bar{W}$	Data Bus
-----------------------------------	--------	------------	-------------	--------------	----------

## IMPLIED

ADC ADD	2	1	Op Code Address + 1	1	Operand Data
AND BIT		2	Op Code Address + 2	1	Next Op Code
CMP EOR					
LDA ORA					
SBC SUB					
ADDD CPX	3	1	Op Code Address + 1	1	Operand Data (MSB)
LDD LDS		2	Op Code Address + 2	1	Operand Data (LSB)
LDX SUBD		3	Op Code Address + 3	1	Next Op Code

## DIRECT

ADC ADD	3	1	Op Code Address + 1	1	Address of Operand (LSB)
AND BIT		2	Address of Operand	1	Operand Data
CMP EOR		3	Op Code Address + 2	1	Next Op Code
LDA ORA					
SBC SUB					
STA	3	1	Op Code Address + 1	1	Destination Address
		2	Destination Address	0	Accumulator Data
		3	Op Code Address + 2	1	Next Op Code
ADDD CPX	4	1	Op Code Address + 1	1	Address of Operand (LSB)
LDD LDS		2	Address of Operand	1	Operand Data (MSB)
LDX SUBD		3	Address of Operand + 1	1	Operand Data (LSB)
		4	Op Code Address + 2	1	Next Op Code
STD STS	4	1	Op Code Address + 1	1	Destination Address (LSB)
STX		2	Destination Address	0	Register Data (MSB)
		3	Destination Address + 1	0	Register Data (LSB)
		4	Op Code Address + 2	1	Next Op Code
JSR	5	1	Op Code Address + 1	1	Jump Address (LSB)
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer - 1	0	Return Address (MSB)
		5	Jump Address	1	First Subroutine Op Code
TIM	4	1	Op Code Address + 1	1	Immediate Data
		2	Op Code Address + 2	1	Address of Operand (LSB)
		3	Address of Operand	1	Operand Data
		4	Op Code Address + 3	1	Next Op Code
AIM EIM	6	1	Op Code Address + 1	1	Immediate Data
OIM		2	Op Code Address + 2	1	Address of Operand (LSB)
		3	Address of Operand	1	Operand Data
		4	FFFF	1	Restart Address (LSB)
		5	Address of Operand	0	New Operand Data
		6	Op Code Address + 3	1	Next Op Code

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	Data Bus
-----------------------------	--------	---------	-------------	-----	----------

INDEXED

JMP	3	1 2 3	Op Code Address + 1 FFFF Jump Address	1 1 1	Offset Restart Address (LSB) First Op Code of Jump Routine
ADC ADD AND BIT CMP EOR LDA ORA SBC SUB TST	4	1 2 3 4	Op Code Address + 1 FFFF IX + Offset Op Code Address + 2	1 1 1 1	Offset Restart Address (LSB) Operand Data Next Op Code
STA	4	1 2 3 4	Op Code Address + 1 FFFF IX + Offset Op Code Address + 2	1 1 0 1	Offset Restart Address (LSB) Accumulator Data Next Op Code
ADDD CPX LDD LDS LDX SUBD	5	1 2 3 4 5	Op Code Address + 1 FFFF IX + Offset IX + Offset + 1 Op Code Address + 2	1 1 1 1 1	Offset Restart Address (LSB) Operand Data (MSB) Operand Data (LSB) Next Op Code
STD STS STX	5	1 2 3 4 5	Op Code Address + 1 FFFF IX + Offset IX + Offset + 1 Op Code Address + 2	1 1 0 0 1	Offset Restart Address (LSB) Register Data (MSB) Register Data (LSB) Next Op Code
JSR	5	1 2 3 4 5	Op Code Address + 1 FFFF Stack Pointer Stack Pointer - 1 IX + Offset	1 1 0 0 1	Offset Restart Address (LSB) Return Address (LSB) Return Address (MSB) First Subroutine Op Code
ASL ASR COM DEC INC LSR NEG ROL ROR	6	1 2 3 4 5 6	Op Code Address + 1 FFFF IX + Offset FFFF IX + Offset Op Code Address + 2	1 1 1 1 0 1	Offset Restart Address (LSB) Operand Data Restart Address (LSB) New Operand Data Next Op Code
TIM	5	1 2 3 4 5	Op Code Address + 1 Op Code Address + 2 FFFF IX + Offset Op Code Address + 3	1 1 1 1 1	Immediate Data Offset Restart Address (LSB) Operand Data Next Op Code
CLR	5	1 2 3 4 5	Op Code Address + 1 FFFF IX + Offset IX + Offset Op Code Address + 2	1 1 1 0 1	Offset Restart Address (LSB) Operand Data 00 Next Op Code
AIM EIM OIM	7	1 2 3 4 5 6 7	Op Code Address + 1 Op Code Address + 2 FFFF IX + Offset FFFF IX + Offset Op Code Address + 3	1 1 1 1 1 0 1	Immediate Data Offset Restart Address (LSB) Operand Data Restart Address (LSB) New Operand Data next Op Code

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	Data Bus
-----------------------------	--------	---------	-------------	-----	----------

EXTEND

JMP	3	1 2 3	Op Code Address + 1 Op Code Address + 2 Jump Address	1 1 1	Jump Address (MSB) Jump Address (LSB) Next Op Code
ADC ADD TST AND BIT CMP EOR LDA ORA SBC SUB	4	1 2 3 4	Op Code Address + 1 Op Code Address + 2 Address of Operand Op Code Address + 3	1 1 1 1	Address of Operand (MSB) Address of Operand (LSB) Operand Data Next Op Code
STA	4	1 2 3 4	Op Code Address + 1 Op Code Address + 2 Destination Address Op Code Address + 3	1 1 0 1	Destination Address (MSB) Destination Address (LSB) Accumulator Data Next Op Code
ADDD CPX LDD LDS LDX SUBD	5	1 2 3 4 5	Op Code Address + 1 Op Code Address + 2 Address of Operand Address of Operand + 1 Op Code Address + 3	1 1 1 1 1	Address of Operand (MSB) Address of Operand (LSB) Operand Data (MSB) Operand Data (LSB) Next Op Code
STD STS STX	5	1 2 3 4 5	Op Code Address + 1 Op Code Address + 2 Destination Address Destination Address + 1 Op Code Address + 3	1 1 0 0 1	Destination Address (MSB) Destination Address (LSB) Register Data (MSB) Register Data (LSB) Next Op Code
JSR	6	1 2 3 4 5 6	Op Code Address + 1 Op Code Address + 2 FFFF Stack Pointer Stack Pointer - 1 Jump Address	1 1 1 0 0 1	Jump Address (MSB) Jump Address (LSB) Restart Address (LSB) Return Address (LSB) Return Address (MSB) First Subroutine Op Code
ASL ASR COM DEC INC LSR NGE ROL ROR	6	1 2 3 4 5 6	Op Code Address + 1 Op Code Address + 2 Address of Operand FFFF Address of Operand Op Code Address + 3	1 1 1 1 0 1	Address of Operand (MSB) Address of Operand (LSB) Operand Data Restart Address (LSB) New Operand Data Next Op Code
CLR	5	1 2 3 4 5	Op Code Address + 1 Op Code Address + 2 Address of Operand Address of Operand Op Code Address + 3	1 1 1 0 1	Address of Operand (MSB) Address of Operand (LSB) Operand Data 00 Next Op Code

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	Data Bus
-----------------------------------	--------	------------	-------------	-----	----------

IMPLIED

ABA ABX ASL ASLD ASR CBA CLC CLI CLR CLV COM DEC DES DEX INC INS INX LSR LSRD ROL ROR NOP SBA SEC SEI SEV TAB TAP TBA TPA TST TSX TXS	1	1	Op Code Address + 1	1	Next Op Code
DAA XGDX	2	1 2	Op Code Address + 1 FFFF	1 1	Next Op Code Restart Address (LSB)
PULA PULB	3	1 2 3	Op Code Address + 1 FFFF Stack Pointer + 1	1 1 1	Next Op Code Restart Address (LSB) Data from Stack
PSHA PSHB	4	1 2 3 4	Op Code Address + 1 FFFF Stack Pointer Op Code Address + 1	1 1 0 1	Next Op Code Restart Address (LSB) Accumulator Data Next Op Code
PULX	4	1 2 3 4	Op Code Address + 1 FFFF Stack Pointer + 1 Stack Pointer + 2	1 1 1 1	Next Op Code Restart Address (LSB) Data from Stack (MSB) Data from Stack (LSB)
PSHX	5	1 2 3 4 5	Op Code Address + 1 FFFF Stack Pointer Stack Pointer - 1 Op Code Address + 1	1 1 0 0 1	Next Op Code Restart Address (LSB) Index Register (LSB) Index Register (MSB) Next Op Code
RTS	5	1 2 3 4 5	Op Code Address + 1 FFFF Stack Pointer + 1 Stack Pointer + 2 Return Address	1 1 1 1 1	Next Op Code Restart Address (LSB) Return Address (MSB) Return Address (LSB) First Op Code of Return Routine
MUL	7	1 2 3 4 5 6 7	Op Code Address + 1 FFFF FFFF FFFF FFFF FFFF FFFF	1 1 1 1 1 1 1	Next Op Code Restart Address (LSB) Restart Address (LSB) Restart Address (LSB) Restart Address (LSB) Restart Address (LSB) Restart Address (LSB)

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/ $\overline{W}$	Data Bus
-----------------------------	--------	---------	-------------	-------------------	----------

IMPLIED

WAI	9	1	Op Code Address + 1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer - 1	0	Return Address (MSB)
		5	Stack Pointer - 2	0	Index Register (LSB)
		6	Stack Pointer - 3	0	Index Register (MSB)
		7	Stack Pointer - 4	0	Accumulator A
		8	Stack Pointer - 5	0	Accumulator B
		9	Stack Pointer - 6	0	Conditional Code Register
RTI	10	1	Op Code Address + 1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer + 1	1	Conditional Code Register
		4	Stack Pointer + 2	1	Accumulator B
		5	Stack Pointer + 3	1	Accumulator A
		6	Stack Pointer + 4	1	Index Register (MSB)
		7	Stack Pointer + 5	1	Index Register (LSB)
		8	Stack Pointer + 6	1	Return Address (MSB)
		9	Stack Pointer + 7	1	Return Address (LSB)
		10	Return Address	1	First Op Code of Return Routine
SWI	12	1	Op Code Address + 1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer - 1	0	Return Address (MSB)
		5	Stack Pointer - 2	0	Index Register (LSB)
		6	Stack Pointer - 3	0	Index Register (MSB)
		7	Stack Pointer - 4	0	Accumulator A
		8	Stack Pointer - 5	0	Accumulator B
		9	Stack Pointer - 6	0	Conditional Code Register
		10	Vector Address FFFA	1	Address of SWI Routine (MSB)
		11	Vector Address FFFB	1	Address of SWI Routine (LSB)
		12	Address of SWI Routine	1	First Op Code of SWI Routine
SLP	4	1	Op Code Address + 1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		↑	FFFF		High Impedance - Non MPX Mode
		↓	FFFF		Address Bus - MPX Mode
		3	FFFF		Restart Address (LSB)
		4	Op Code Address + 1		Next Op Code

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	Data Bus
-----------------------------	--------	---------	-------------	-----	----------

RELATIVE

BCC BCS	3	1	Op Code Address + 1	1	Branch Offset
BEQ BGE		2	FFFF	1	Restart Address (LSB)
BGT BHI		3	{ Branch Address Test = "1" Op Code Address Test = "0"	1	First Op Code of Branch Routine
BLE BLS				1	Next Op Code
BLT BMT					
BNE BPL					
BRA BRN					
BVC BVS					
BSR	5	1	Op Code Address + 1	1	Offset
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer - 1	0	Return Address (MSB)
		5	Branch Address	1	First Op Code of Subroutine

3

### 3.4 System Flowchart

A system flow of the HD6301V1 is given in Fig. 3-4-1.



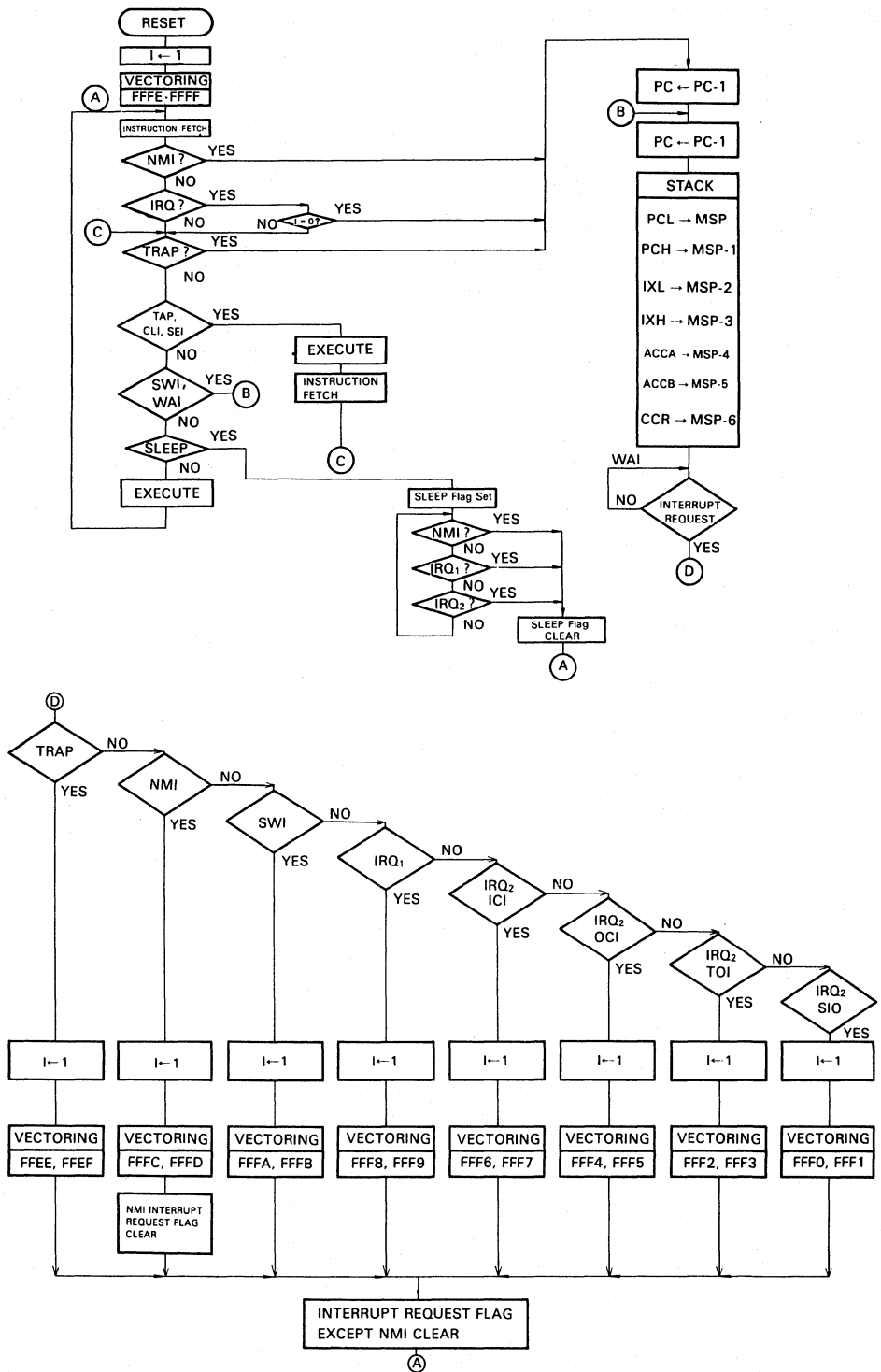
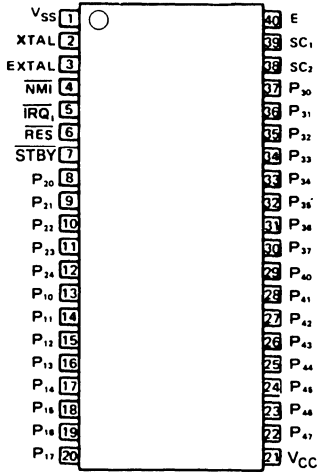


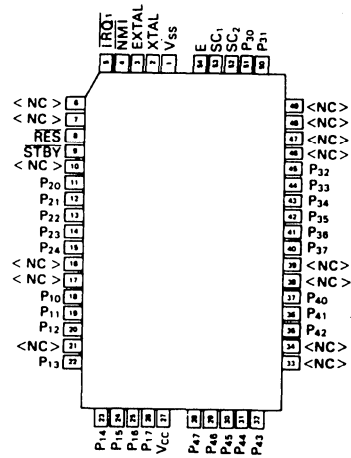
Fig. 3-4-1 HD6301V1 System Flowchart



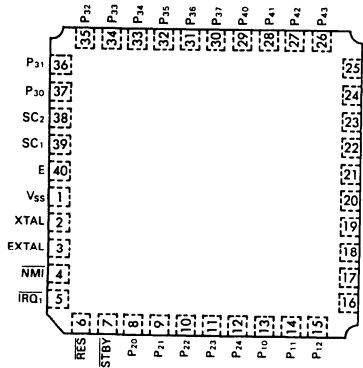
● HD6301V1P, HD63A01V1P  
HD63B01V1P (DP-40)



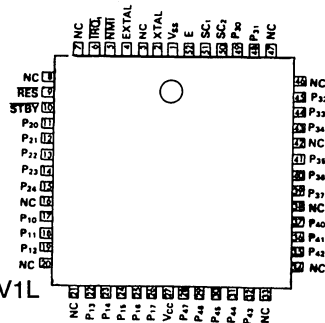
● HD6301V1FG, HD63A01V1F  
HD63B01V1FG (FP-54)



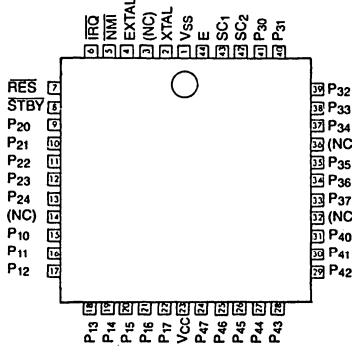
● HD6301V1CG, HD63A01V1CG  
HD63B01V1CG (CG-40)



● HD6301V1CP, HD63A01V1CP  
HD63B01V1CP (CP-52)



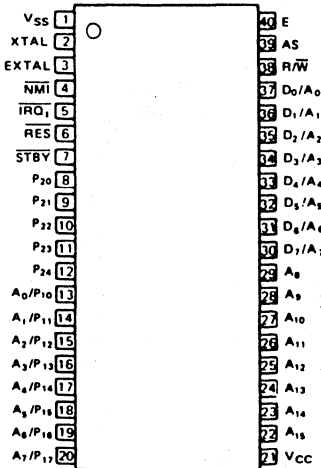
● HD6301V1L, HD63A01V1L  
HD63B01V1L (CP-44)



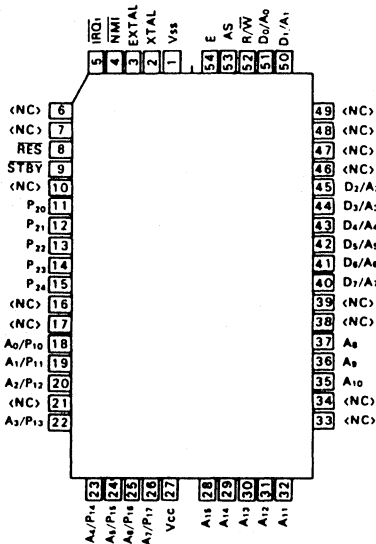
(Top View)

Fig. 4-1-1 Pin Arrangement (Top View)

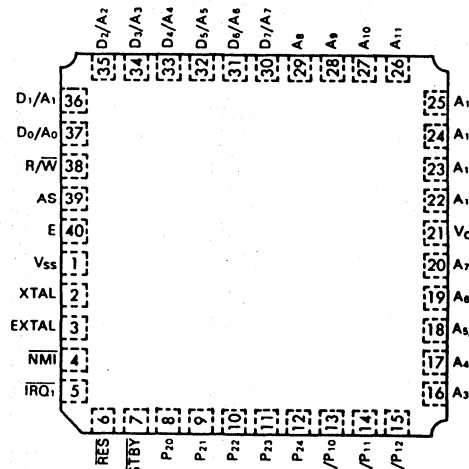
● HD6303RP, HD63A03RP, HD63B03RP (DP-40)



● HD6303RF, HD63A03RF, HD63B03RF (FP-54)



● HD6303RCG, HD63A03RCG, HD63B03RCG (CG-40)



● HD6303RCP, HD63A03RCP, HD63B03RCP (CP-52)

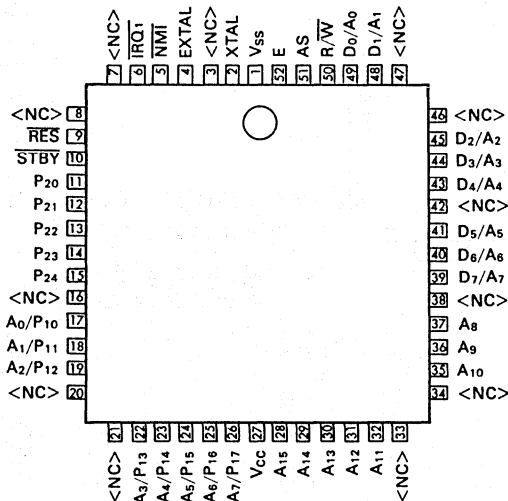
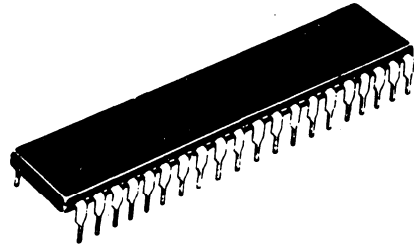
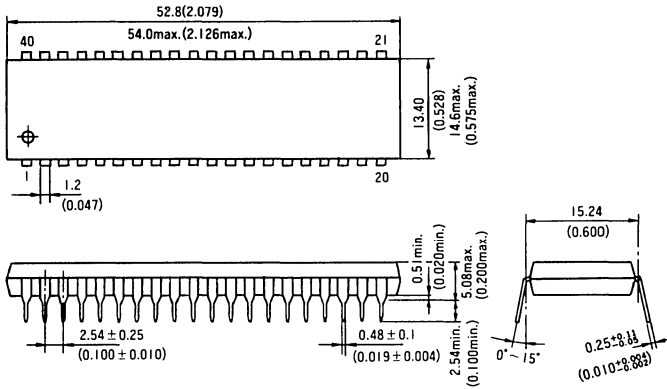


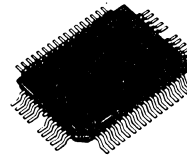
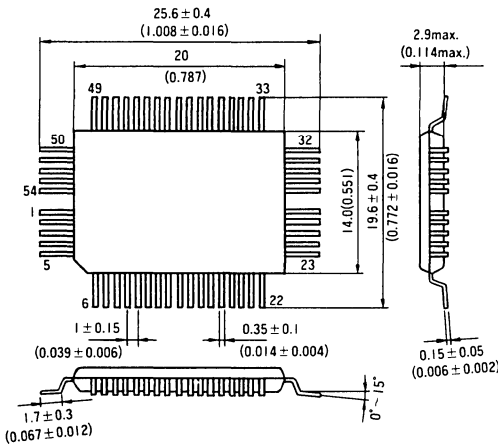
Fig. 4-1-2 Pin Arrangement (Top View)



● DP-40



● FP-54



● CG-40

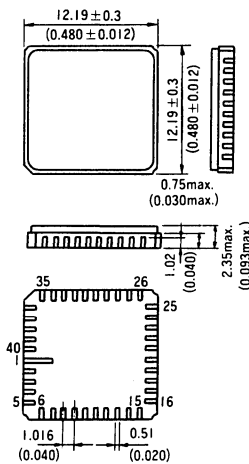
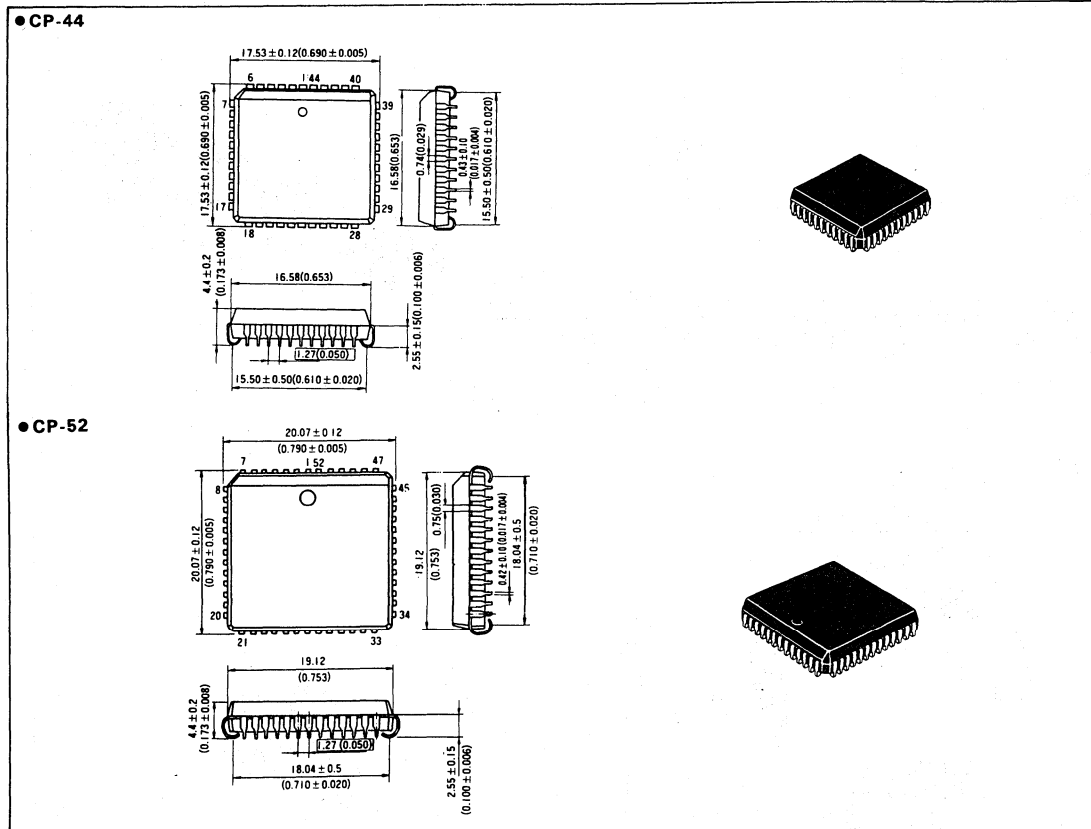


Fig. 4-2 Package Information

(to be continued)





## 5. ELECTRICAL CHARACTERISTICS

### ■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	$V_{CC}$	-0.3 ~ +7.0	V
Input Voltage	$V_{in}$	-0.3 ~ $V_{CC}+0.3$	V
Operating Temperature	$T_{opr}$	0 ~ +70	°C
Storage Temperature	$T_{stg}$	-55 ~ +150	°C

(NOTE) This product has protection circuits in input pin from high static electricity voltage and high electric field. But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend  $V_{in}, V_{out} : V_{SS} \leq (V_{in} \text{ or } V_{out}) \leq V_{CC}$ .

## ■ ELECTRICAL CHARACTERISTICS (HD6301V1 and HD6303R)

- DC CHARACTERISTICS ( $V_{CC} = 5.0V \pm 10\%$ ,  $f = 0.1 \sim 2.0$  MHz,  $V_{SS} = 0V$ ,  $T_a = 0 \sim +70^\circ C$ , unless otherwise noted.)

Item	Symbol	Test Condition	min	typ	max	Unit	
Input "High" Voltage	$V_{IH}$		$V_{CC}-0.5$	—	$V_{CC} + 0.3$	V	
			$V_{CC} \times 0.7$	—			
			2.0	—			
Input "Low" Voltage	$V_{IL}$		-0.3	—	0.8	V	
Input Leakage Current	$I_{in}$	$V_{in} = 0.5 \sim V_{CC}-0.5$	—	—	1.0	$\mu A$	
Three State (off-state) Leakage Current	$I_{TSI}$	$V_{in} = 0.5 \sim V_{CC}-0.5$	—	—	1.0	$\mu A$	
Output "High" Voltage	$V_{OH}$		$I_{OH} = -200\mu A$	2.4	—	V	
			$I_{OH} = -10\mu A$	$V_{CC}-0.7$	—	V	
Output "Low" Voltage	$V_{OL}$		—	—	0.55	V	
Input Capacitance	$C_{in}$	$V_{in} = 0V, f = 1.0$ MHz, $T_a = 25^\circ C$	—	—	12.5	pF	
Standby Current	$I_{CC}$	$V_{IL}(\overline{STBY}) = 0 \sim 0.6V$ $V_{IH}(\overline{RES}) = V_{CC}-0.5 \sim V_{CC} V$ $V_{IL}(\overline{RES}) = 0 \sim 0.6V$	—	2.0	15.0	$\mu A$	
			Operating ( $f = 1$ MHz)**	—	6.0		10.0
Current Dissipation*	$I_{CC}$		Sleeping ( $f = 1$ MHz)**	—	1.0	2.0	mA
RAM Stand-By Voltage	$V_{RAM}$		2.0	—	—	V	

\*  $V_{IH}$  min =  $V_{CC} - 1.0V$ ,  $V_{IL}$  max =  $0.8V$  A11 output pins have no load.

\*\* Current Dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ. or max. values about Current Dissipations at x MHz operation are decided according to the following formulas.

$$\begin{aligned} \text{typ. value (f = x MHz)} &= \text{typ. value (f = 1 MHz)} \times x \\ \text{max. value (f = x MHz)} &= \text{max. value (f = 1 MHz)} \times x \\ &\text{(both the sleeping and operating)} \end{aligned}$$

## PERIPHERAL PORT TIMING

Item	Symbol	Test Condition	HD6301V1/ HD6303R			HD63A01V1/ HD63A03R			HD63B01V1/ HD63B03R			Unit	
			min	typ	max	min	typ	max	min	typ	max		
Peripheral Data Set-up Time	Port 1, 2, 3, 4	$t_{PDSU}$	Fig. 5-3	200	—	—	200	—	—	200	—	—	ns
Peripheral Data Hold Time	Port 1, 2, 3, 4	$t_{PDH}$	Fig. 5-3	200	—	—	200	—	—	200	—	—	ns
Delay Time, Enable Positive Transition to $\overline{OS3}$ Negative Transition		$t_{OSD1}$	Fig. 5-5	—	—	300	—	—	300	—	—	300	ns
Delay Time, Enable Positive Transition to $\overline{OS3}$ Positive Transition		$t_{OSD2}$	Fig. 5-5	—	—	300	—	—	300	—	—	300	ns
Delay Time, Enable Negative Transition to Peripheral Data Valid	Port 1, 2*, 3, 4	$t_{PWD}$	Fig. 5-4	—	—	300	—	—	300	—	—	300	ns
Input Strobe Pulse Width		$t_{PWIS}$	Fig. 5-6	200	—	—	200	—	—	200	—	—	ns
Input Data Hold Time	Port 3	$t_{IH}$	Fig. 5-6	150	—	—	150	—	—	150	—	—	ns
Input Data Setup Time	Port 3	$t_{IS}$	Fig. 5-6	0	—	—	0	—	—	0	—	—	ns

\* Except P<sub>21</sub>

- AC CHARACTERISTICS ( $V_{CC} = 5.0V \pm 10\%$ ,  $f = 0.1 \sim 2.0$  MHz,  $V_{SS} = 0V$ ,  $T_a = 0 \sim +70^\circ C$ , unless otherwise noted.)

## BUS TIMING

Item	Symbol	Test Condition	HD6301V1/ HD6303R			HD63A01V1/ HD63A03R			HD63B01V1/ HD63B03R			Unit	
			min	typ	max	min	typ	max	min	typ	max		
Cycle Time	$t_{cyc}$	Fig. 5-1, Fig. 5-2	1	—	10	0.666	—	10	0.5	—	10	$\mu s$	
Address Strobe Pulse Width "High"	$PW_{ASH}$		220	—	—	150	—	—	110	—	—	ns	
Address Strobe Rise Time	$t_{ASr}$		—	—	20	—	—	20	—	—	20	ns	
Address Strobe Fall Time	$t_{ASf}$		—	—	20	—	—	20	—	—	20	ns	
Address Strobe Delay Time	$t_{ASD}$		60	—	—	40	—	—	20	—	—	ns	
Enable Rise Time	$t_{Er}$		—	—	20	—	—	20	—	—	20	ns	
Enable Fall Time	$t_{Ef}$		—	—	20	—	—	20	—	—	20	ns	
Enable Pulse Width "High" Level	$PW_{EH}$		450	—	—	300	—	—	220	—	—	ns	
Enable Pulse Width "Low" Level	$PW_{EL}$		450	—	—	300	—	—	220	—	—	ns	
Address Strobe to Enable Delay Time	$t_{ASED}$		60	—	—	40	—	—	20	—	—	ns	
Address Delay Time	$t_{AD1}$		—	—	250	—	—	190	—	—	160	ns	
	$t_{AD2}$		—	—	250	—	—	190	—	—	160	ns	
Address Delay Time for Latch	$t_{ADL}$		—	—	250	—	—	190	—	—	160	ns	
Data Set-up Time	Write		$t_{DSW}$	230	—	—	150	—	—	100	—	—	ns
	Read		$t_{DSR}$	80	—	—	60	—	—	50	—	—	ns
Data Hold Time	Read		$t_{HR}$	0	—	—	0	—	—	0	—	—	ns
	Write		$t_{HW}$	20	—	—	20	—	—	20	—	—	ns
Address Set-up Time for Latch	$t_{ASL}$		60	—	—	40	—	—	20	—	—	ns	
Address Hold Time for Latch	$t_{AHL}$		20	—	—	20	—	—	20	—	—	ns	
Address Hold Time	$t_{AH}$		20	—	—	20	—	—	20	—	—	ns	
$A_0 \sim A_7$ Set-up Time Before E	$t_{ASM}$		200	—	—	110	—	—	60	—	—	ns	
Peripheral Read Access Time	Non-Multiplexed Bus		$(t_{ACCN})$	—	—	650	—	—	395	—	—	270	ns
	Multiplexed Bus	$(t_{ACCM})$	—	—	650	—	—	395	—	—	270	ns	
Oscillator stabilization Time	$t_{RC}$	Fig. 2-7-1, Fig. 2-8-1	20	—	—	20	—	—	20	—	—	ms	
Processor Control Set-up Time	$t_{PCS}$		200	—	—	200	—	—	200	—	—	ns	

## TIMER, SCI TIMING

Item	Symbol	Test Condition	HD6301V1/ HD6303R			HD63A01V1/ HD63A03R			HD63B01V1/ HD63B03R			Unit
			min	typ	max	min	typ	max	min	typ	max	
Timer Input Pulse Width	$t_{PWT}$		2.0	—	—	2.0	—	—	2.0	—	—	$t_{cyc}$
Delay Time, Enable Positive Transition to Timer Out	$t_{TOD}$	Fig. 5-7	—	—	400	—	—	400	—	—	400	ns
SCI Input Clock Cycle	$t_{Scyc}$		2.0	—	—	2.0	—	—	2.0	—	—	$t_{cyc}$
SCI Input Clock Pulse Width	$t_{PWCK}$		0.4	—	0.6	0.4	—	0.6	0.4	—	0.6	$t_{Scyc}$

## MODE PROGRAMMING

Item	Symbol	Test Condition	HD6301V1/ HD6303R			HD63A01V1/ HD63A03R			HD63B01V1/ HD63B03R			Unit
			min	typ	max	min	typ	max	min	typ	max	
RES "Low" Pulse Width	$PW_{RSTL}$	Fig. 5-8	3	—	—	3	—	—	3	—	—	$t_{cyc}$
Mode Programming Set-up Time	$t_{MPS}$		2	—	—	2	—	—	2	—	—	$t_{cyc}$
Mode Programming Hold Time	$t_{MPH}$		150	—	—	150	—	—	150	—	—	ns

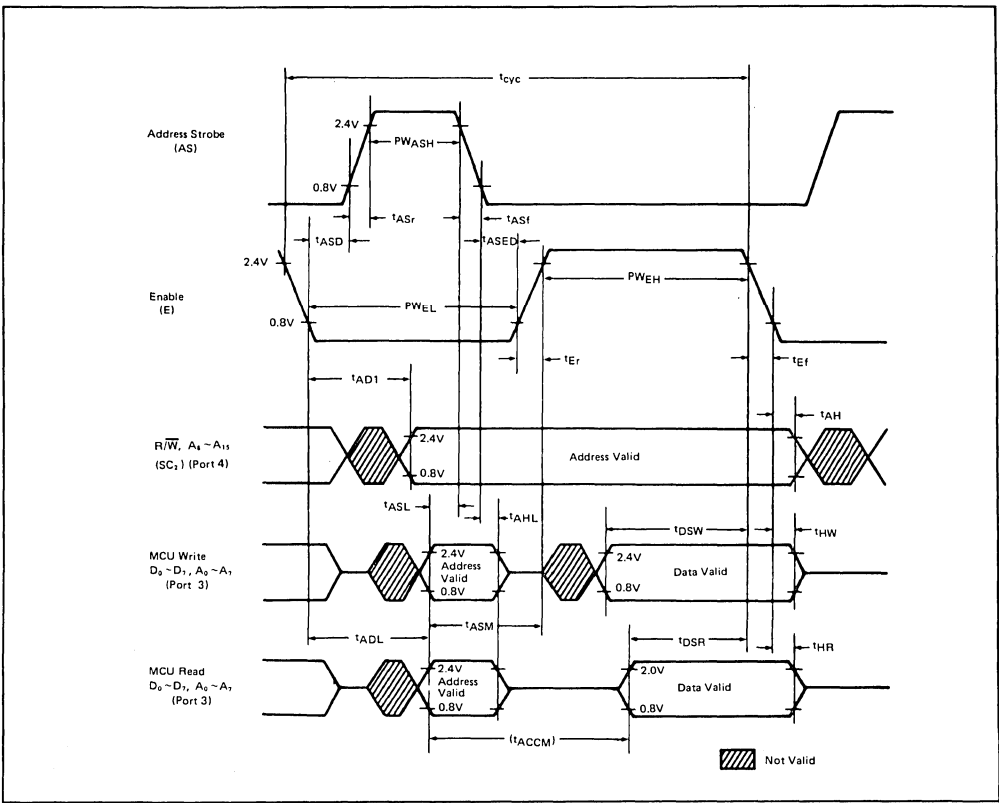


Fig. 5-1 Expanded Multiplexed Bus Timing

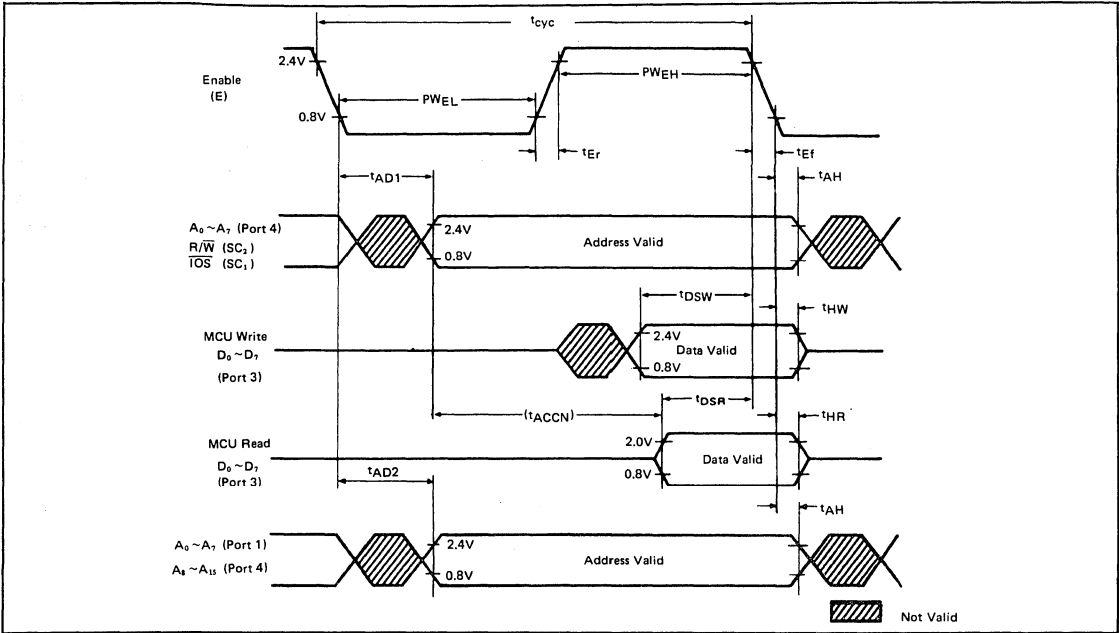


Fig. 5-2 Expanded Non-Multiplexed Bus Timing



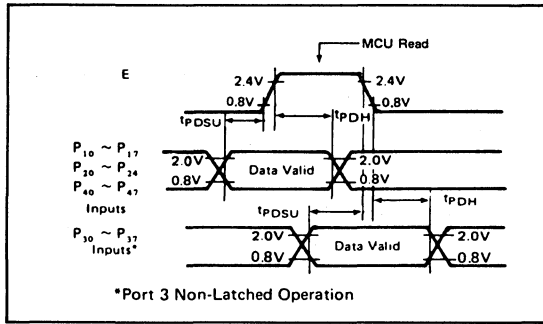


Fig. 5-3 Port Data Set-up and Hold Times (MCU Read)

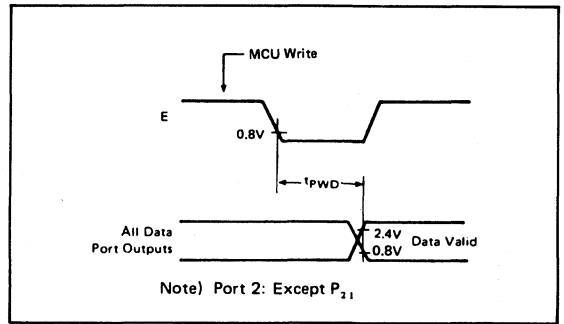


Fig. 5-4 Port Data Delay Times (MCU Write)

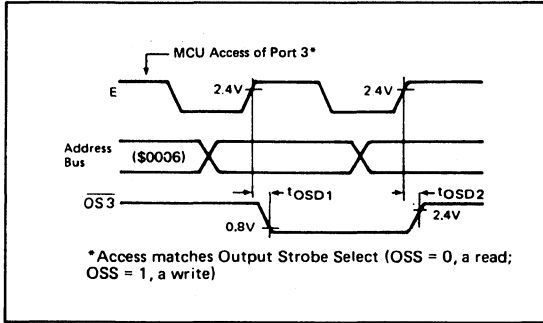


Fig. 5-5 Port 3 Output Strobe Timing (Single Chip Mode)

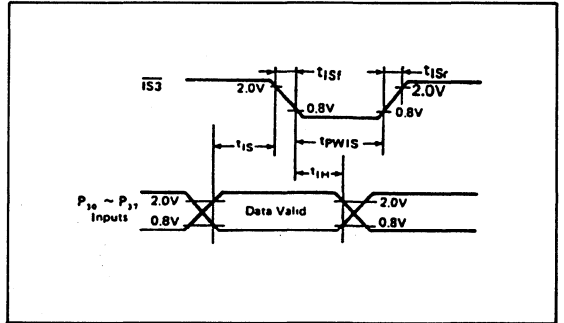


Fig. 5-6 Port 3 Latch Timing (Single Chip Mode)

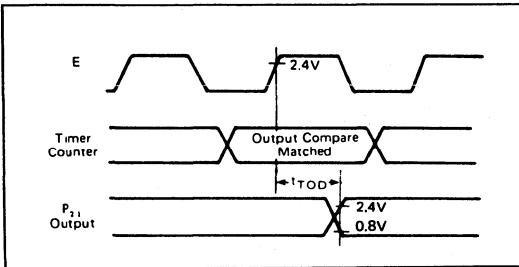


Fig. 5-7 Timer Output Timing

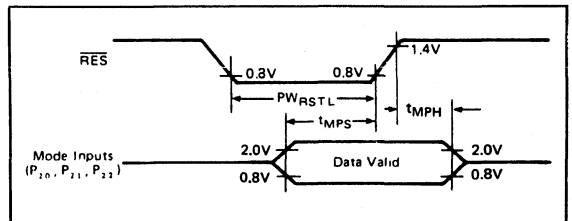


Fig. 5-8 Mode Programming Timing

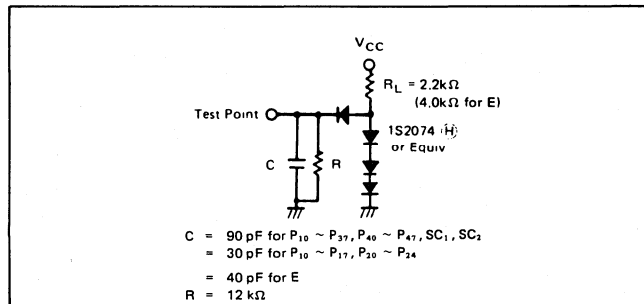


Fig. 5-9 Bus Timing Test Loads (TTL Load)

## 6. APPLICATIONS

### 6.1 Use of External Expanded Mode

The HD6301V1 supports five operation modes 1, 2, 4, 5 and 6 as external expanded modes. Usage of these modes is detailed in the following paragraphs.

#### (1) Non-multiplexed modes

##### (a) Mode 1 (New Mode)

In this mode, port 3 works as data bus, port 1 as lower address bus ( $A_0 - A_7$ ), and port 4 as upper address bus ( $A_8 - A_{15}$ ). Since 16-bit addresses are sent out in parallel, the HD6301V1 can access to a 65k memory space with no address latch externally under this mode.

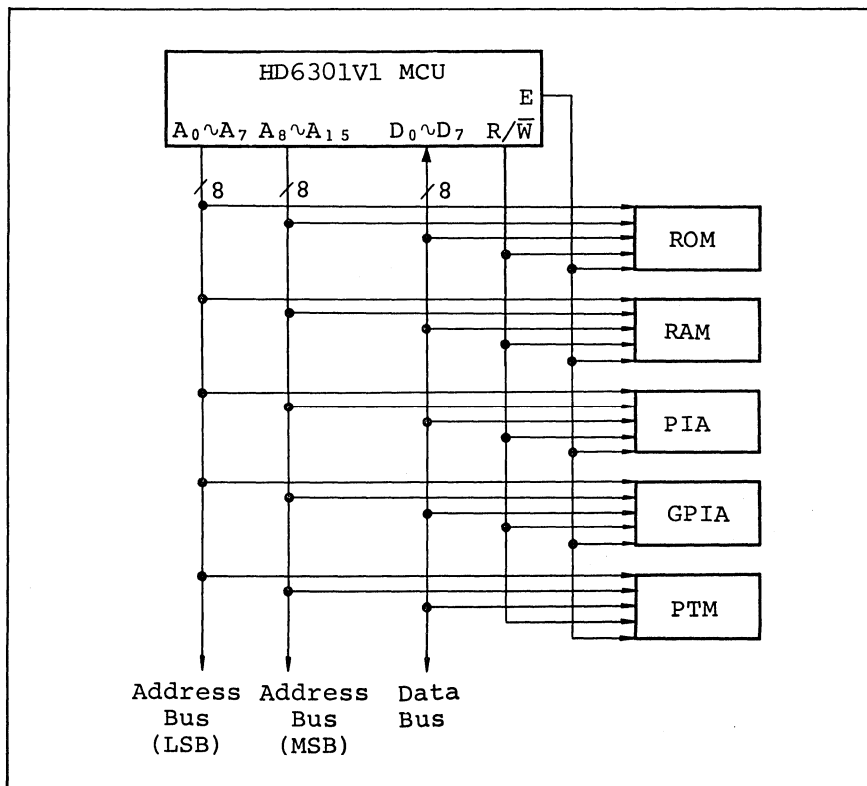


Fig. 6-1-1 HD6301V1, Mode 1

In the case when a write operation is performed to the internal memory including I/O and registers, the same data is also written into the external memory located by the same address if a memory exists.

In the case when a read operation is performed to the internal memory, however, only a data of the internal memory is read and no external data pointed by the same address is read. Read/write operation to the internal/external memory with the internal memory address range is also applied to the mode 2, 4, 5 and 6. Under this mode, the internal mask ROM of which location is \$F000 through \$FFFF to address is no more accessible and an external memory can be accessed with this address range.

After reset, Port 1 is a lower address bus ( $A_0 - A_7$ ), Port 4 is a upper address bus ( $A_8 - A_{15}$ ).

(b) Mode 5 (Equivalent to Mode 5 of HD6801V)

Port 3 works as data bus; and port 4 as address bus ( $A_0 - A_7$ ) or input pin by DDR. In this mode, pin 39 provides the result of the following decoding:

$$\overline{A_{15}} \cdot \overline{A_{14}} \cdot \overline{A_{13}} \cdot \overline{A_{12}} \cdot \overline{A_{11}} \cdot \overline{A_{10}} \cdot \overline{A_9} \cdot A_8$$

This output signal may be used as a chip select or chip enable signal permits to access an external memory up to 256 byte locations (\$0100 - \$01FF). The pin function of Port 4 can be changed from an address line to an input port in the case that the system does not need all of the 8 address lines by writing zero into the corresponding bit of Port 4 DDR.

An example of connection with PIA (HD6821, HD6321) is shown in Fig. 6-1-2.

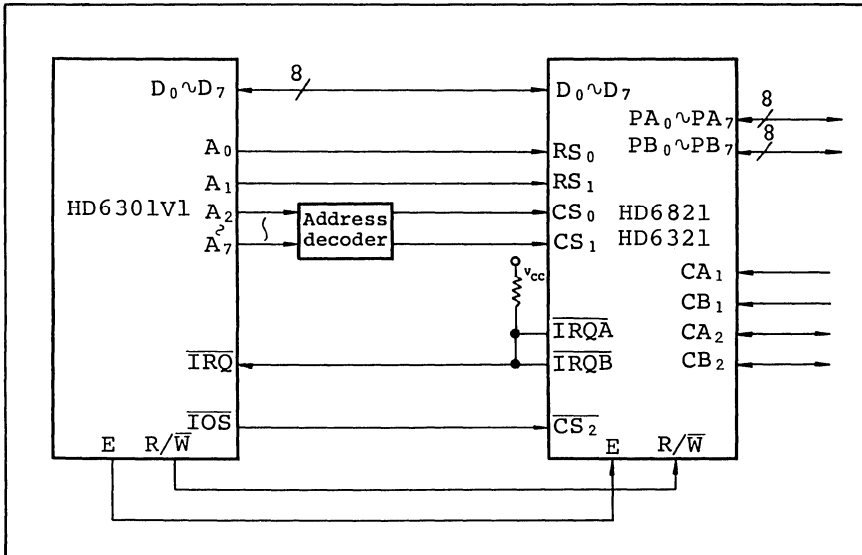


Fig. 6-1-2 Connection of HD6301V1 with PIA

(2) Multiplex Modes (Modes 2, 4 & 6)

Any multiplex mode provides a time multiplexed address and data on port 3. Therefore, an address latch is required externally. AS (Pin 39) signal is used for an address latch strobe. An example is illustrated in Fig. 6-1-3 to show how CMOS latch is used with the HD6301V1. It should be noted, however, that the output address from this latch is delayed.

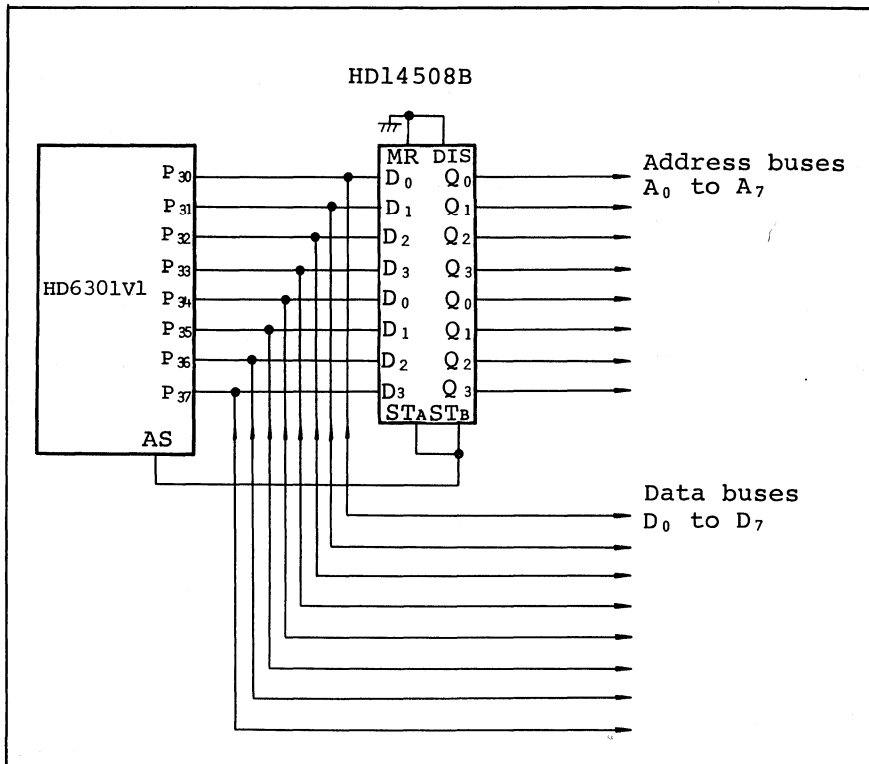


Fig. 6-1-3 CMOS Latch

For high-speed operation, 74LS373 or high speed CMOS latch (74HC373) is desirable to minimize the delay time.

(a) Mode 2, 4 (Equivalent to Mode 2 of HD6801V)

In this mode, the internal mask ROM (\$F000 through \$FFFF) is disabled and external memory becomes valid instead. Port 4 works as the upper address bus.

(b) Mode 6 (Equivalent to Mode 6 of HD6801V)

In this mode, the internal mask ROM is enabled. Port 4 works as address bus (A<sub>8</sub> - A<sub>15</sub>) input. Since Port 4 becomes input mode after reset, "1" must be written into DDR by program if it is required to use the port as address buses.

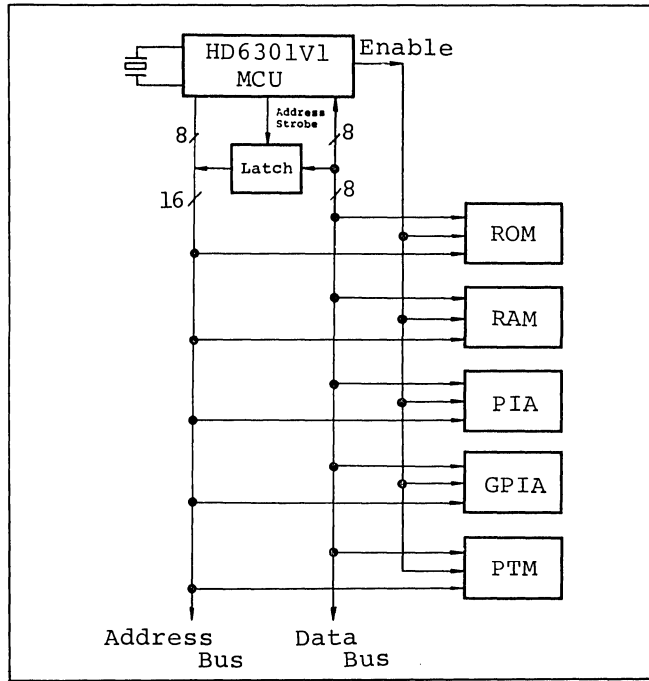


Fig. 6-1-4 HD6301V1 MCU Expanded Multiplex Mode

## 6.2 Standby Mode

Bringing  $\overline{STBY}$  "Low", the HD6301V1 goes into the Standby mode. In this mode, the CPU becomes reset and all clocks of the HD6301V1 become inactive.

The contents of the internal RAM is retained as long as  $V_{CC}$  is supplied ( $V_{CC} \geq 2V$ ). Under Standby Mode, memory back-up is possible with only a few  $\mu A$  of leakage current. With "1" level at  $\overline{STBY}$  pin, the MCU exits from Standby Mode. When "1" level is detected at  $\overline{STBY}$  pin, a clock generator begins to oscillate and the internal reset condition is released. At this time,  $\overline{RES}$  signal should be set at "0" level for at least OSC stabilization time ( $t_{RC}$ ) before the CPU operation restarts. Otherwise, the normal operation is not guaranteed.

A typical flowchart to use a Standby Mode is shown in Fig. 6-2-1.

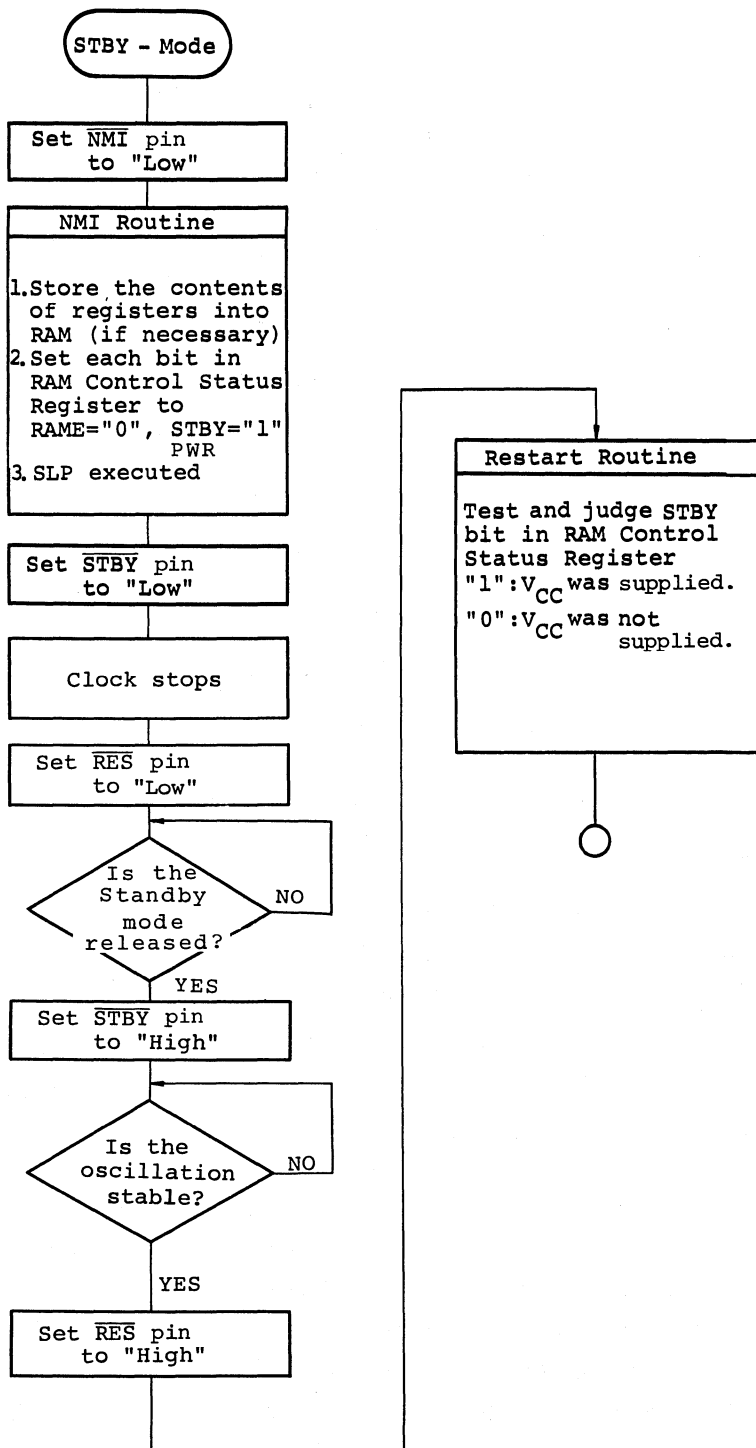


Fig. 6-2-1 Flowchart of Standby Mode Application

The timing relationship shown in Fig. 6-2-2 must be satisfied.

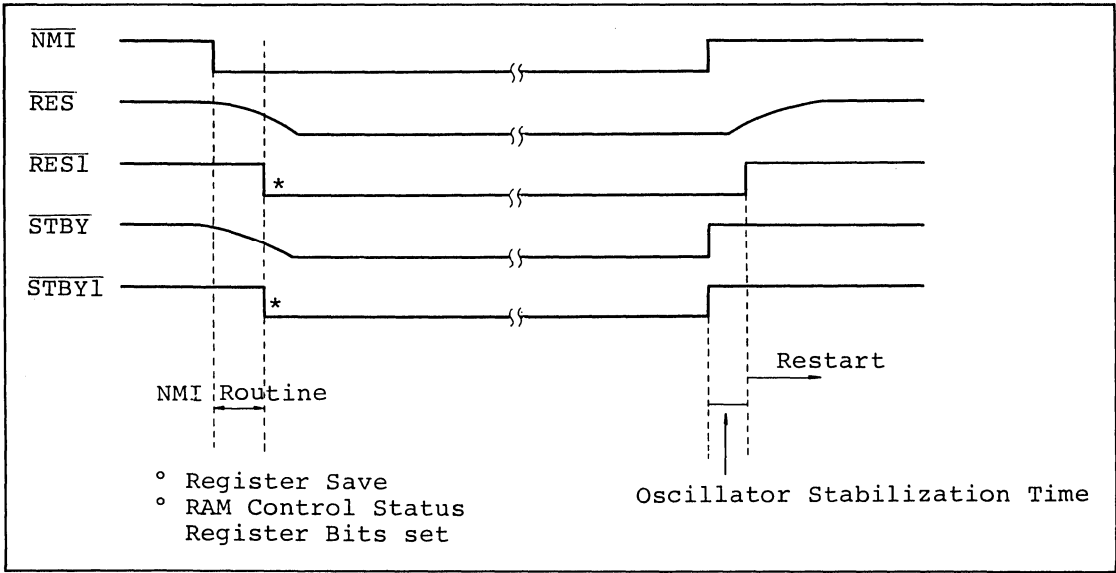


Fig. 6-2-2 Timing Chart of Each Signal

\* Either RES1 or STBY1 can become "0" level as long as the execution time of NMI routine is guaranteed.

Fig. 6-2-3 shows an example of a circuit to implement the timing sequence shown in the Fig. 6-2-2.

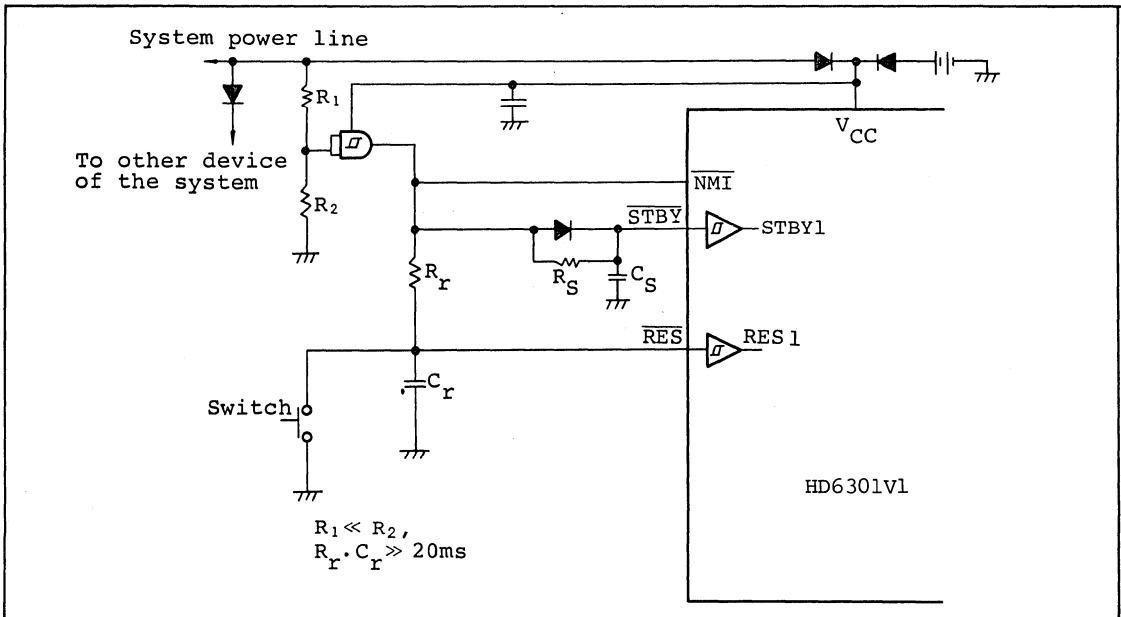


Fig. 6-2-3 Example of Circuit Diagram for a Standby Operation



<Precaution for using Standby Power bit>

The Standby power bit in the RAM control status register detects that V<sub>CC</sub> is supplied or not. When the V<sub>CC</sub> rise time is equal or less than 100 $\mu$ s, the Standby power bit may not be cleared. To avoid this, the V<sub>CC</sub> rise time should be more than 100 $\mu$ s, for example, by using the larger bypass capacitor.

### 6.3 Address Trap, OP-Code Trap Application

The HD6301V1 facilitates two trap functions, the operation code trap and the address trap, to protect the HD6301V1 to proceed an erroneous operation. The operation code trap is a trap when an operation code currently fetched is illegal or undefined. Therefore, when undefined codes listed below are fetched, a trap is caused and the HD6301V1 avoids further erroneous operation. The priority level of the interrupt caused by this operation code trap is next to the RESET. Undefined codes of the HD6301V1 are: \$00, \$02, \$03, \$12, \$13, \$14, \$15, \$1C, \$1D, \$1E, \$1F, \$41, \$42, \$45, \$4B, \$4E, \$51, \$52, \$55, \$5B, \$5E, \$87, \$8F, \$C7, \$CD and \$CF.

The address trap is a TRAP when an operation code is fetched from the memory area shown in Table 2-3-1. It should be noted, however, this function works only under op-code fetch (not for data access). Under the support of error processing program in trap service routine, the user can protect the system from further erroneous operation. If RTI instruction is executed at the end of the trap service routine, the program control returns to the location where the trap is caused previously and then another trap may be caused again. So, please take special care when a programmer uses this trap function.

## 6.4 Slow Memory Interface

Here described is the example of clock width controll circuit and its timing chart, where E-clock high time is extended to assure enough access time.

The expanded enable high pulse width ( $PW'_{EH}$ ), which is implemented by using the circuit shown below, is calculated as follows:

$$PW'_{EH} = (n+1) \cdot t_{4\phi cyc} + PW_{EH} \leq 10 - PW_{EL} (\mu s)$$

where  $n$  : Integer part of  $[t_W / t_{4\phi cyc}]$

$t_{4\phi cyc}$  :  $4\phi$  clock cycle time ( $\mu s$ )

$PW_{EH}$  : Enable High pulse width ( $\mu s$ )

$PW_{EL}$  : Enable Low pulse width ( $\mu s$ )

$t_W$  : approx.  $0.45 \cdot C_{ext} (pF) \cdot R_{ext} (k\Omega) \times 10^{-3} (\mu s)$

The circuit shown is for a reference purpose. It is assumed that users will refine it for actual design.

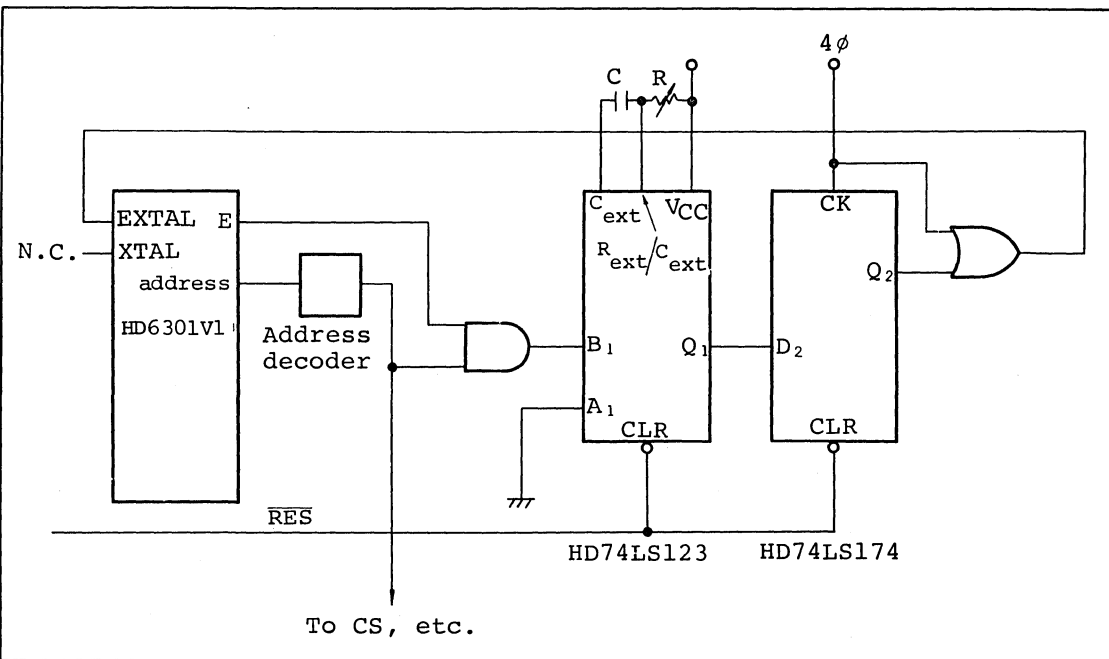


Fig. 6-4-1 Clock Control Circuit

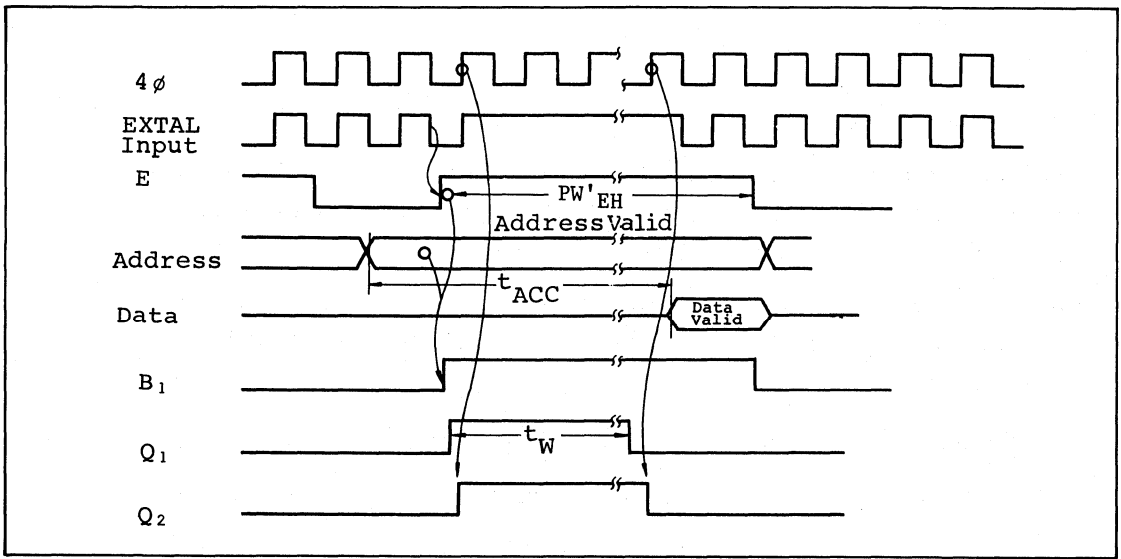


Fig. 6-4-2 Clock Timing

### 6.5 Interface to HN61256

The examples of the interface to a slow memory device, HN61256 (CMOS 256k bit Mask programmable ROM), is described here.

The AC characteristics and the access timing of the HN61256 is shown in Fig. 6-5-1.

Item	Symbol	min	max	Unit
Read Cycle Time	$t_{RC}$	4.0	-	$\mu s$
Address Access Time	$t_{AACC}$	-	3.5	$\mu s$
Chip Enable Access Time	$t_{EACC}$	-	3.0	$\mu s$
Data Hold Time from Address	$t_{DF}$	0.05	0.5	$\mu s$
Address Set-up Time	$t_{AS}$	0.5	-	$\mu s$
Address Hold Time	$t_{AH}$	0	-	$\mu s$
Chip Enable ON Time	$t_{\overline{CE}}$	3.0	-	$\mu s$
Chip Enable OFF Time	$t_{CE}$	0.5	-	$\mu s$

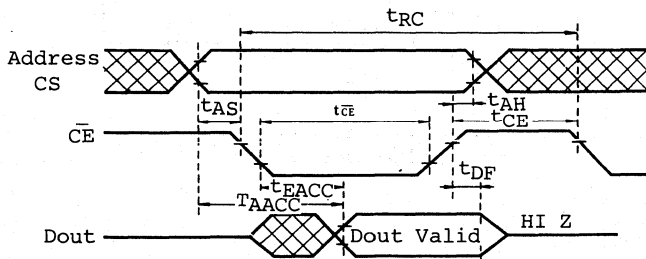


Fig. 6-5-1 AC Characteristics and Access Timing of HN61256



### 6.5.1 Use of Two Latches

The two HD14508B are used in order to latch 16 bit address. An example of the program and its access timing chart are shown in Table 6-5-1 and Fig. 6-5-3, respectively.

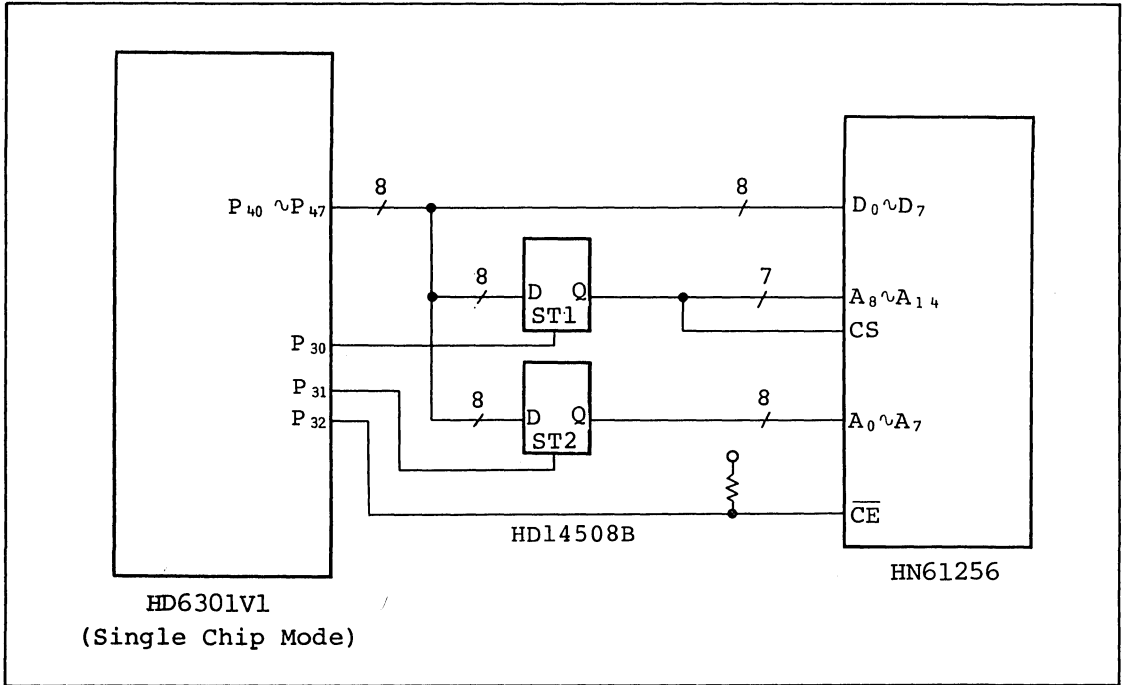


Fig. 6-5-2 Using Two Latches

Table 6-5-1 An Example of the Program

<u>Mnemonic</u>		<u>Cycles</u>	
LDAA	#\$FF	2	
STAA	P4DDR	3	PORT 4 is the output port.
LDD	#\$ADDRS1	3	Data that is the address's upper 8 bits including CS signal and changes ST1 into high and ST2 into low.
STD	PORT3	4	Enables ST1, disables ST2, and moves the address's upper 8 bits into PORT 4.
LDD	#\$ADDRS2	3	Data that is the address's lower 8 bits and changes ST1 into low and ST2 into high.
STD	PORT3	4	Disables ST1, enables ST2, and stores the address's lower 8 bits into PORT 4.
LDAA	#IMM1	2	Data that changes ST1 and ST2 into low and $\overline{CE}$ into active.
STAA	PORT3	3	Disables ST1 and ST2 and enables $\overline{CE}$ .
LDAB	#\$00	2	
STAB	P4DDR	3	PORT 4 becomes the input port.
LDAA	PORT4		Reads data.

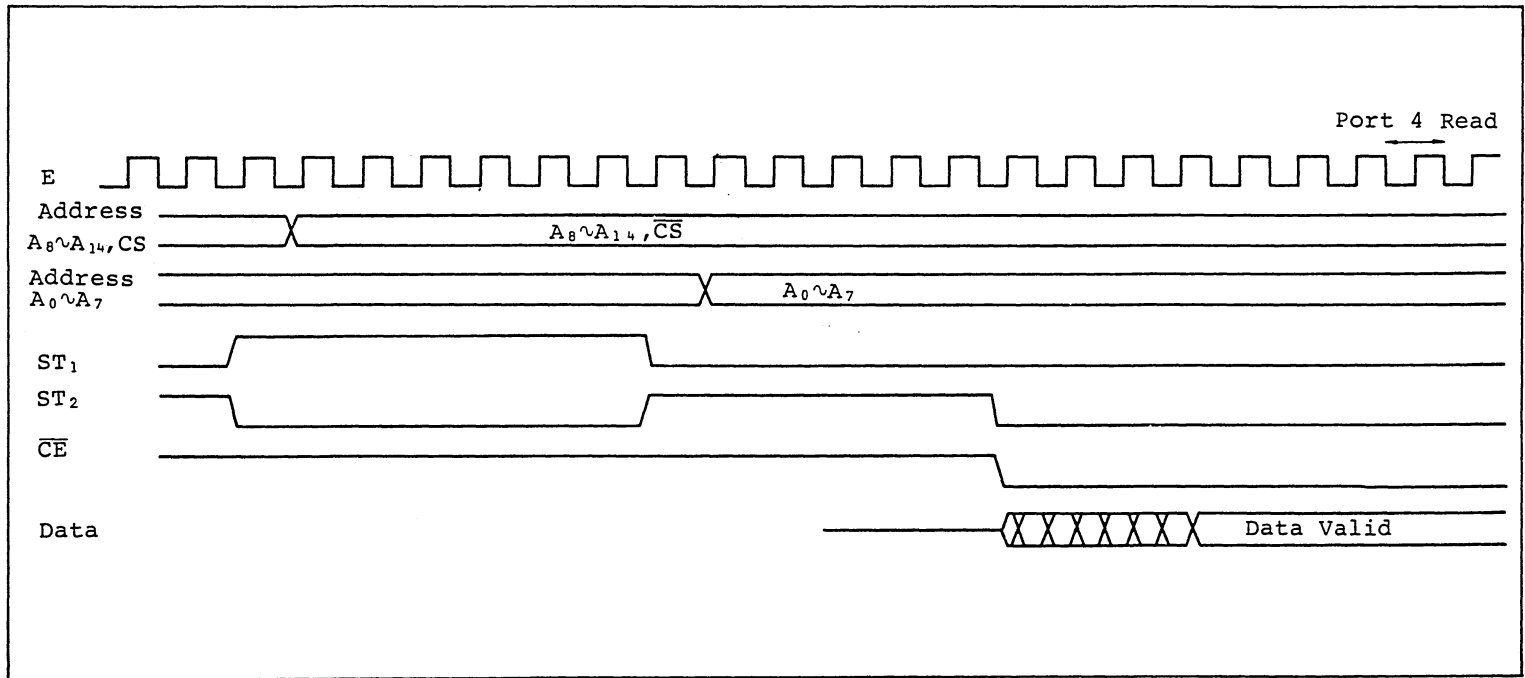


Fig. 6-5-3 Access Timing Chart

### 6.5.2 Stretch of E clock

Fig. 6-5-4 is an example circuitry to show how the E clock is stretched.

The operation Mode of the HD6301V1 is in Mode 6; and the clock frequency of  $4\phi$  is 4 MHz.

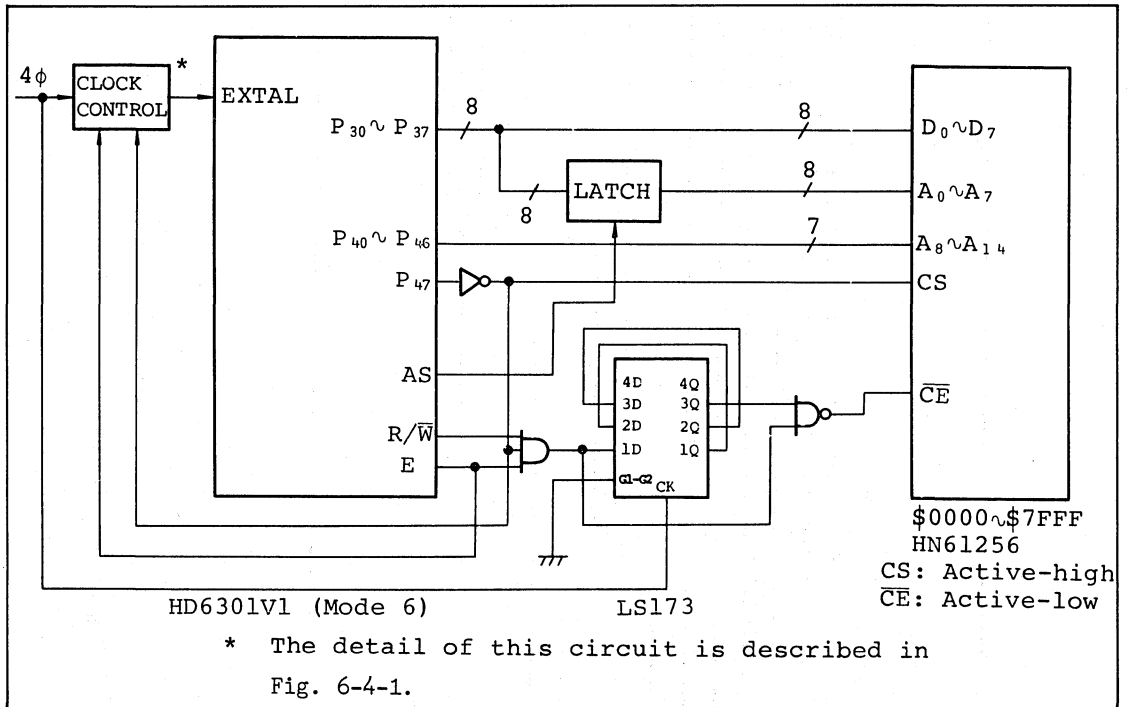


Fig. 6-5-4

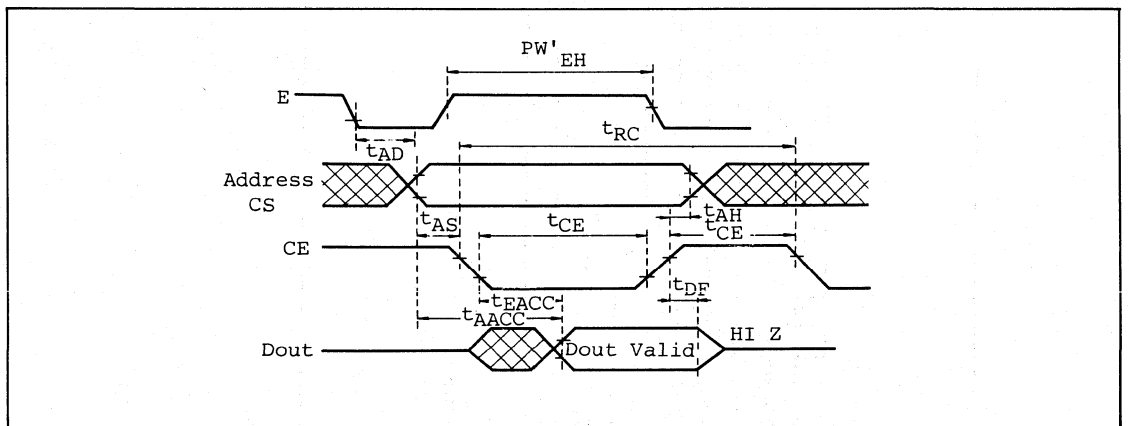


Fig. 6-5-5 HN61256 Read Timing



In this example,  $PW_{EH}$  of which timing is extended by using the clock control circuit (Fig. 6-4-1) must be at least 4  $\mu$ s. The LS173 is to assure enough address set up time ( $t_{AS}$ ) of HN61256.

## 6.6 Interface to the Realtime Clock (HD146818)

The HD146818 (realtime clock + RAM : RTC) is a CMOS micro-computer peripheral LSI that incorporates the clock and calendar functions to compute year, month, day, day of week, and time. When this HD146818 is interfaced to the HD6301V1, this LSI provides a real time clock information to be displayed.

In addition to the real time clock function of the HD146818, this device also be utilized as a system interval timer and a square waves generator. An example of the interface between the HD146818 and the HD6301V1 is shown in Fig. 6-6-1. It can be interfaced under the expanded multiplexed mode (mode 4 or 6) of the HD6301V1.



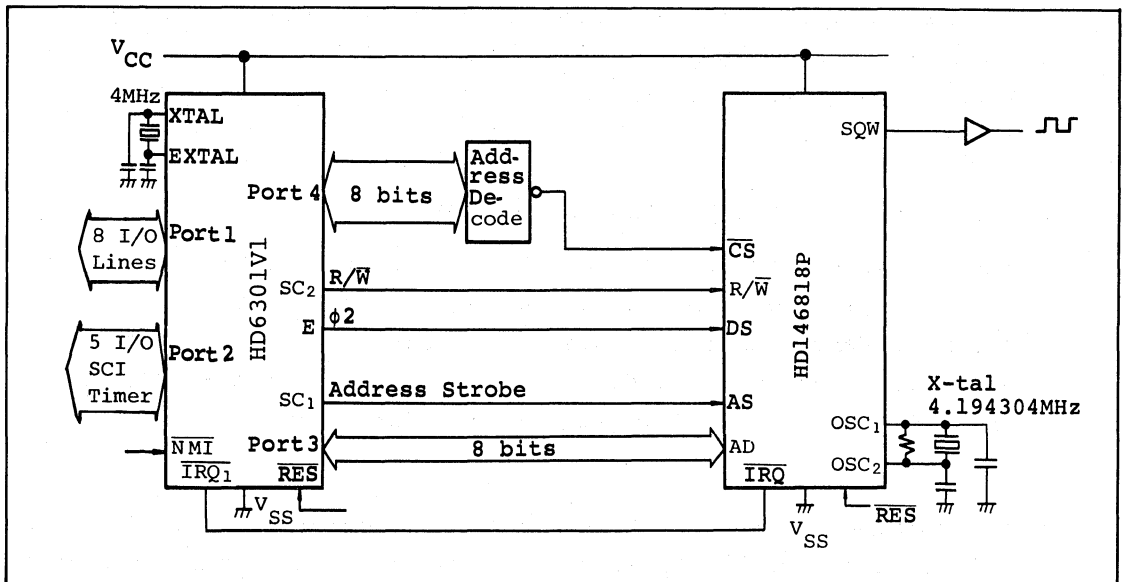


Fig. 6-6-1 HD6301V1 MCU Expanded Multiplexed Mode Interface

The calendar and clock display functions of HD146818 are shown below.

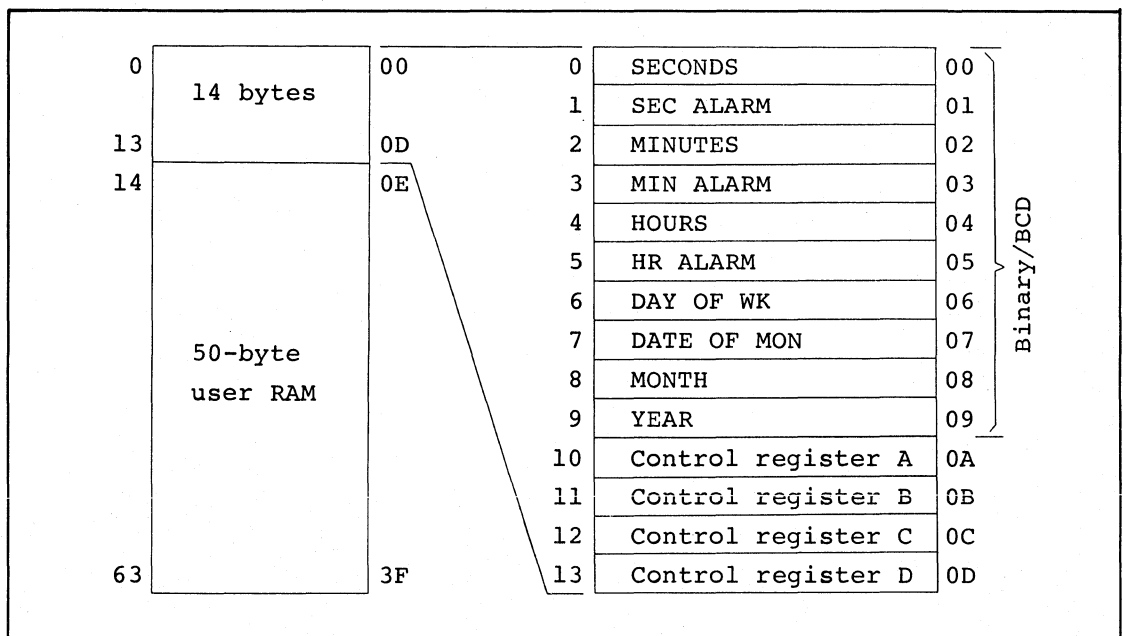


Fig. 6-6-2 HD146818 Built-in RAM Address Map

Table 6-6-1. HD146818 Time, Calendar, & Alarm Data Display

Address	Function		Data range (Decimal)	Data range (Hexadecimal)	
				Binary data mode	BCD data mode
0	SECONDS		0 to 59	00 to 3B	00 to 59
1	SECONDS ALARM		0 to 59	00 to 3B	00 to 59
2	MINUTES		0 to 59	00 to 3B	00 to 59
3	MINUTES ALARM		0 to 59	00 to 3B	00 to 59
4	HOURS	12-hour mode	1 to 12	01 to 0C/ 81 to 8C*	01 to 12/ 81 to 92*
		24-hour mode	0 to 23	00 to 17	00 to 23
5	HOURS ALARM	12-hour mode	1 to 12	01 to 0C/ 81 to 8C*	01 to 12/ 81 to 92*
		24-hour mode	0 to 23	00 to 17	00 to 23
6	DAY OF THE WEEK		1 to 7**	01 to 07	01 to 07
7	DAY OF THE MONTH		1 to 31	01 to 1F	01 to 31
8	MONTH		1 to 12	01 to 0C	01 to 12
9	YEAR		0 to 99***	00 to 63	00 to 99

[Notes]

\*: The most significant bit differentiates between AM and PM. That is, 0 = AM and 1 = PM.

\*\* : 1 = Sunday, 2 = Monday, 3 = Tuesday, 4 = Wednesday, 5 = Thursday, 6 = Friday, and 7 = Saturday

\*\*\*: This takes the lower two digits of the calendar year.

The information of the calendar and the time are stored on the built-in RAM and updated every second. The built-in RAM includes not only the display RAM but also 50-byte user RAM which stores data necessary for the system.

The HD6301V1 gets the calendar and time information by reading the built-in RAM of the HD146818. The HD146818 generates three different types of interrupts, update interrupt, alarm interrupt and periodic interrupt, to the HD6301V1. The HD6301V1 proceeds a service for each of these interrupt requests by a software control.

Such a combination of the HD6301V1 and the HD146818 easily implements a compact real time system with reduced power dissipation.

Note: For details of the HD146818, refer to "HD146818 Data Sheet".

### 6.7 Reference Data of Battery Service Life

Fig. 6-7-1 shows the battery service life taken from a silver oxide battery: SR44W (by Hitachi Maxell).

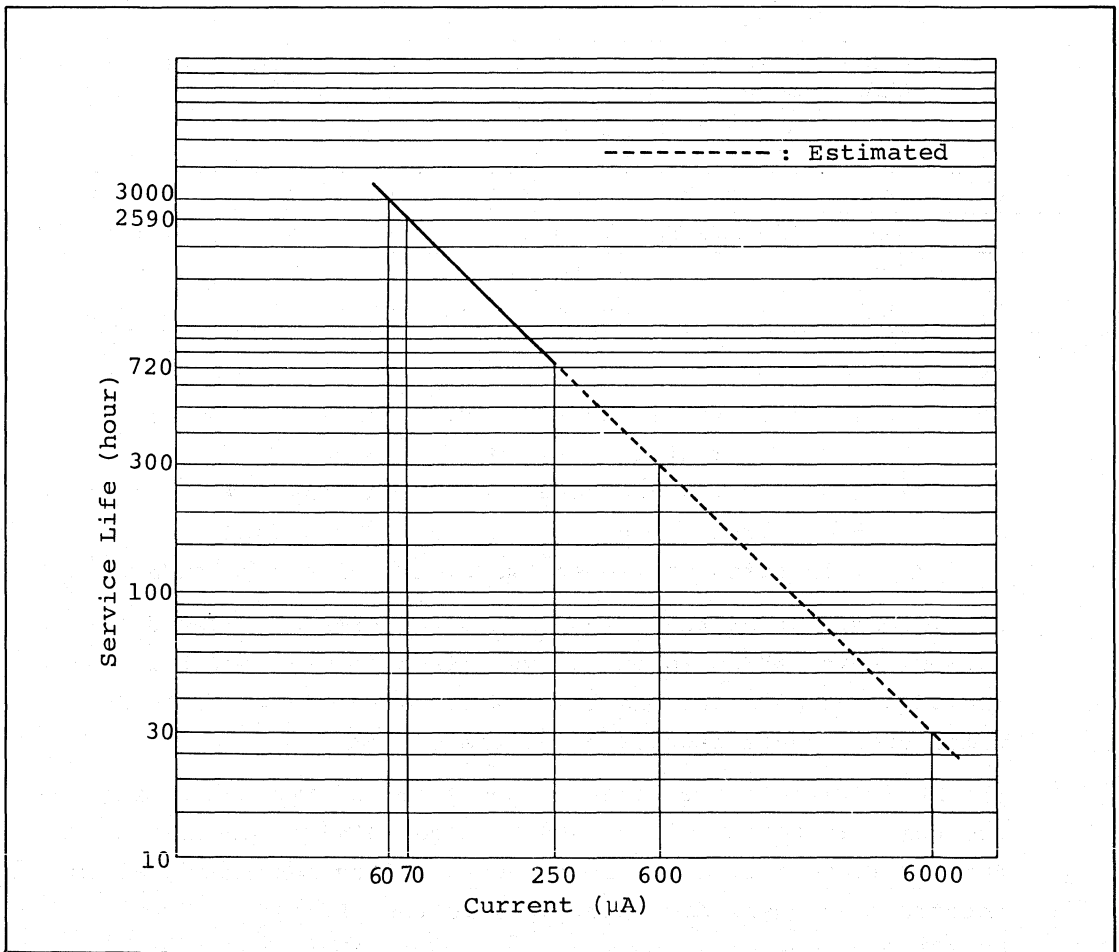


Fig. 6-7-1 Battery Service Life (Maxell SR44W)

## 7. PRECAUTIONS

### 7.1 Write-Only Register

When a write-only register such as the DDR of the port is read by the MPU, "\$FF" always appears on the data bus. Note that when an instruction which reads the memory contents and does some arithmetic operation on the contents of the write-only register, it always gets \$FF as the arithmetic and logical results. AIM, OIM and EIM instructions are unable to apply especially for the bit manipulation of the DDR of the I/O port.

### 7.2 Address Strobe (AS)

The AS signal is used as an address latch strobe and is always accompanied with the E-clock. This means the AS is available in both Operation and Sleep Mode whenever the E-clock is generated. The AS signal is disabled in Mode 5, 7 or under Standby Mode and the Pin 39 is used for other purposes in these cases.

### 7.3 Mode 0

This mode is used for the test purpose only. It is not recommended to use this mode for the other purposes.

### 7.4 Trap Interrupt

When executing an RTI instruction at the end of the interrupt routine, trap interrupt different from other interrupts returns to the address where the trap interrupt was generated. Attention is necessary when using several trap interrupts in the program. See Fig. 7-4-1 and 7-4-2 for details.

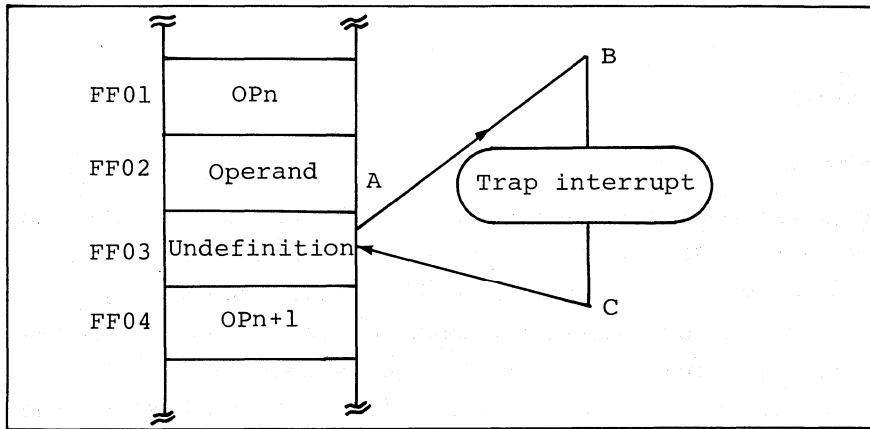


Fig. 7-4-1 Fetching an Undefined Op-code

After executing OPn instruction, the HD6301V1 fetches and decodes an undefined op-code inside to generate a trap interrupt. When RTI instruction is executed in this trap interrupt servicing routine, the HD6301V1 will set \$FF03 in PC, fetch the undefined code again, generate a trap interrupt and repeat ABC endless-loop.

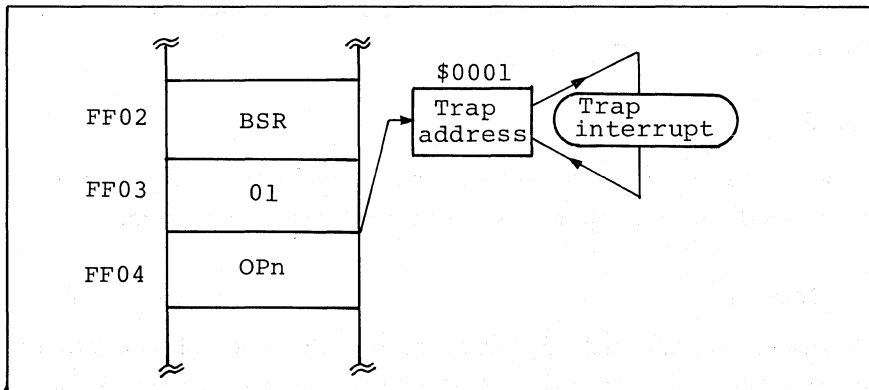


Fig. 7-4-2 Fetching Erroneously

After performing BSR instruction, the branch destination address is output on an address bus to fetch the first op-code of a subroutine. If \$0001 is output as an address by some mistake the HD6301V1 decodes it inside and generates a trap interrupt. When RTI instruction is performed in this trap interrupt servicing routine, the HD6301V1 will set \$0001 in PC and start from this address, which causes a trap interrupt again and repeat this endless-loop.

## 7.5 Power-on Reset

At power-on it is necessary to hold  $\overline{\text{RES}}$  "low" to reset the internal state of the device and to provide sufficient time for the oscillator to stabilize. Pay attention to the following.

- \* Just after power-on, the MPU doesn't enter reset state until the oscillation starts. This is because the reset signal is input internally, with the clocked synchronization as shown below.

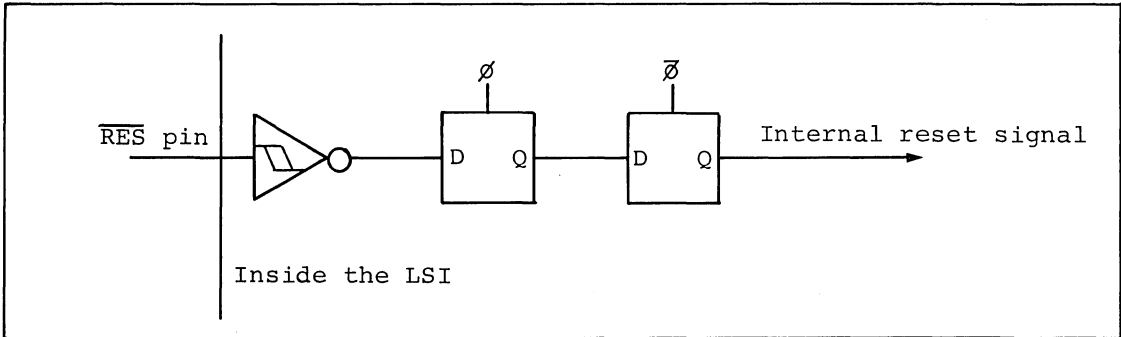


Fig. 7-5-1 Reset Circuit

Thus, just after power-on the LSI state (I/O port, mode condition etc.) is unstable until the oscillation starts. If it is necessary to inform the LSI state to the external devices during this period, it needs to be done by the external circuits.

## 7.6 Precaution to the Board Design of Oscillation Circuit

As shown in Fig. 7-6-1 there is a case that the crosstalk disturbs the normal oscillation if signal lines are put near the oscillation circuit. When designing a board, pay attention to this. Crystal and  $C_L$  must be put as near the HD6301V1 as possible.

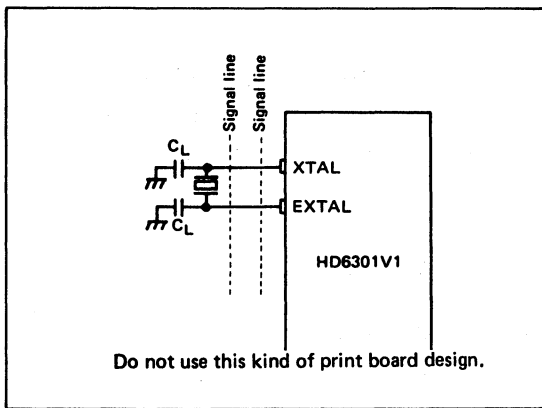


Fig. 7-6-1 Precaution to the Board Design of Oscillation Circuit

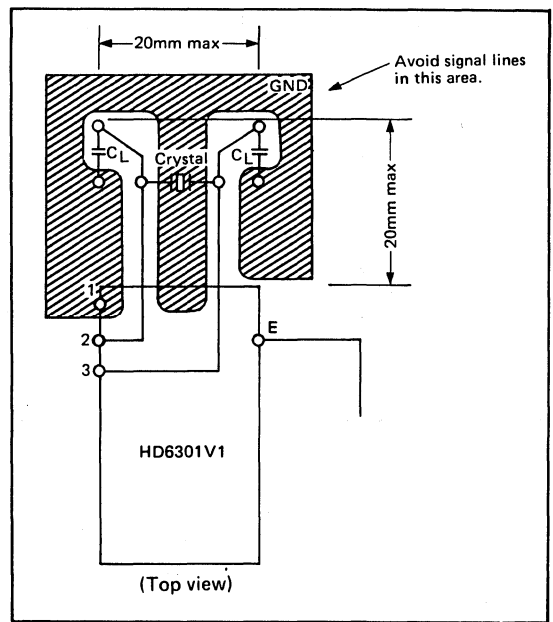


Fig. 7-6-2 Example of Oscillation Circuits in Board Design

## 7.7 Application Note for High Speed System Design Using the HD6301V1

This note describes the solutions of the potential problem caused by noise generation in the system using the HD6301V1.

The CMOS ICs and LSIs featured by low power consumption and high noise immunity are generally considered to be enough with simply designed power source and the GND line.

But this does not apply to the applications configured of high speed system or of high speed parts. Such high speed system may have a chance to work incorrectly because of the noise by the transient current generated during switching. The noise generation owing to the over current (Sometimes it may be several hundreds mA for peak level.) during switching may cause data write error.

This noise problem may be observed only at the Expanded Mode (Mode 1, 2, 4, 5 and 6) of the HD6301V1. The Single Chip Mode (Mode 7) of the HD6301V1 has no such a problem.

Assuming the HD6301V1 is used as CPU in a system.

### 7.7.1 Noise Occurrence

If the HD6301V1 is connected to high speed RAM, a write error

may occur. As shown in Fig. 7-7-1 the noise is generated in address bus during write cycle and data is written into an unexpected address from the HD6301V1. This phenomenon causes random failures in systems whose data bus load capacitance exceeds the specification value (90 pF max.) and/or the impedance of the GND line is high.

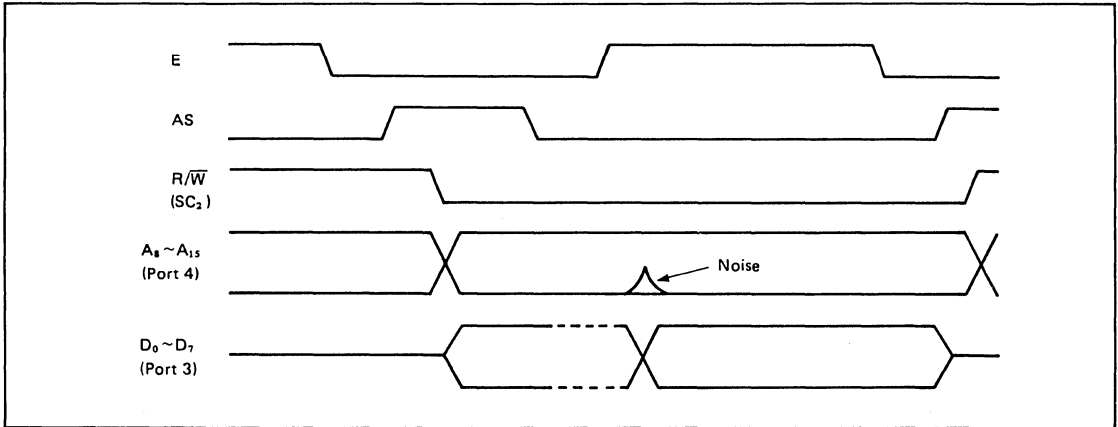


Fig. 7-7-1 Noise Occurrence in Address Bus During Write Cycle

If the data bus  $D_0 \sim D_7$  changes from "FF" to "00", extremely large transient current flows through the GND line. Then the noise is generated on the LSI's  $V_{SS}$  pins proportioning to the transient current and to the impedance ( $Z_g$ ) of the GND line.

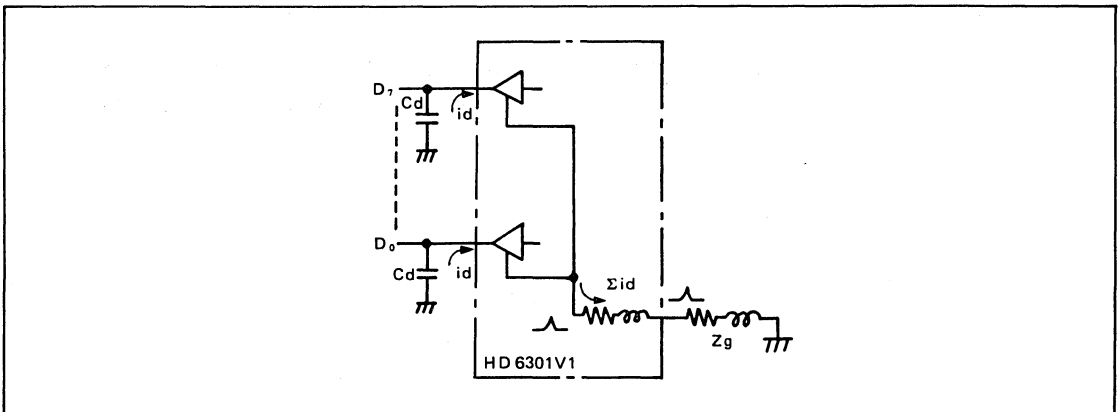


Fig. 7-7-2 Noise Source

This noise level,  $V_n$ , appears on all output pins on the LSI including the address bus.

Fig. 7-7-3 shows the dependency of the noise voltage on the each parameter.



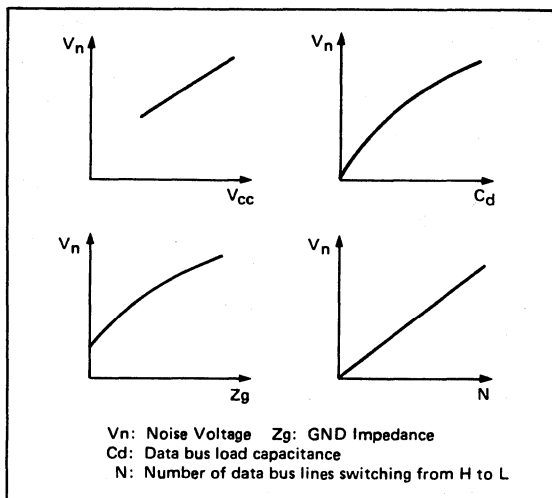


Fig. 7-7-3 Dependency of the Noise Voltage on each Parameter

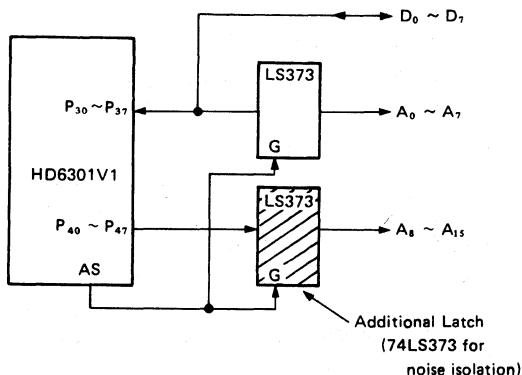
### 7.7.2 Noise Protection

To avoid the noise on the address bus during the system operation mentioned before, there are two solutions as follows:

The one method is to isolate the HD6301V1 from peripheral devices so that peripherals are not affected by the noise. The other is to reduce noise level to the extent of not affecting peripherals using analog method.

#### (1) Noise Isolation

Addresses should be latched at the negative edge of the AS signal or at the positive edge of the E signal. The 74LS373 is often used in this case.



## 2. Noise Reduction

As the noise level depends on each parameter such  $C_d$ ,  $V_{CC}$ ,  $Z_g$ , the noise level can be reduced to the allowable level by controlling those analog parameters.

### (a) Transient Current Reduction

- (i) Reduce the data bus load capacitance. If large load capacitance is expected, a bus buffer should be inserted.
- (ii) Lower the power supply voltage  $V_{CC}$  within specification.
- (iii) Increase a time constant at transient state by inserting a resistor ( $100 \sim 200\Omega$ ) to Data Buses in series to keep noise level down.

Table 8-1 shows the relationship between a series resistors and noise level or a resistor and DC/AC characteristics.

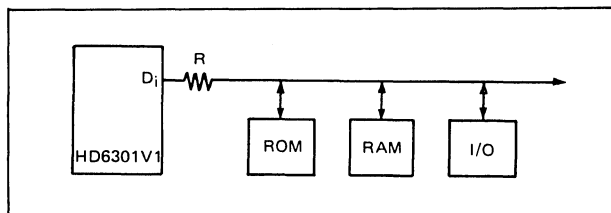


Table 7-1

Item		Resistor	No	100 $\Omega$	200 $\Omega$
Noise Voltage Level			See Fig. 36		
DC Characteristics		$I_{OL}$	1.6 mA	1.6 mA	1.0 mA
AC Characteristics	$f = 1$ MHz		No change		
	$f = 1.5$ MHz	$t_{ADL}$	190 ns	190 ns	210 ns
		$t_{ACCM}$	395 ns	395 ns	375 ns
	$f = 2$ MHz	$t_{ADL}$	160 ns	180 ns	200 ns
		$t_{ASL}$	20 ns	20 ns	0 ns
		$t_{ACCM}$	270 ns	250 ns	230 ns

Fig. 7-7-4 shows an example of the dependency of the noise voltage on the load capacitance of the data bus.\*

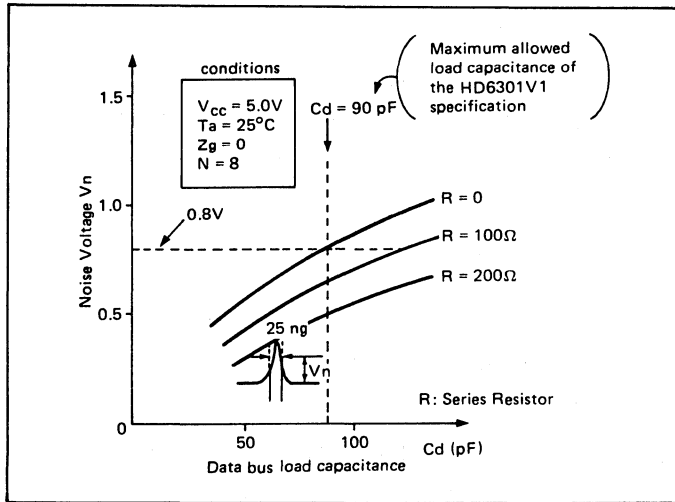


Fig. 7-7-4

\*Note: The value of series resistor should be carefully selected because it heavily depends on each parameter of actual application system.

Fig. 7-7-5 shows the typical wave form of the noise.

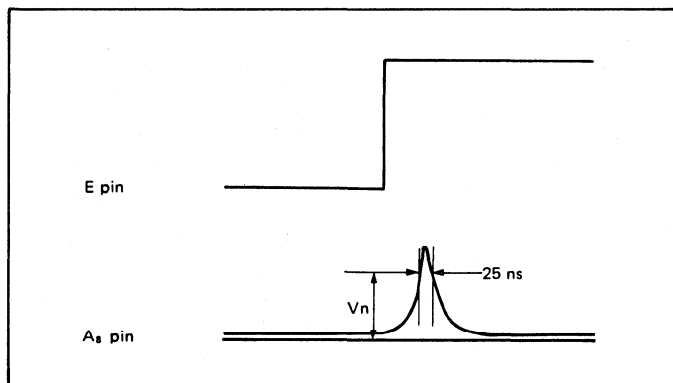


Fig. 7-7-5

(b) Reduction of GND line impedance

- (i) Widen the GND line width on the PC board.
- (ii) Place the HD6301V1 close by power source.
- (iii) Insert a bypass capacitor between the  $V_{CC}$  line and the GND of the HD6301V1. A tantalum capacitor (about  $0.1\mu\text{F}$ ) is effective on the reduction.

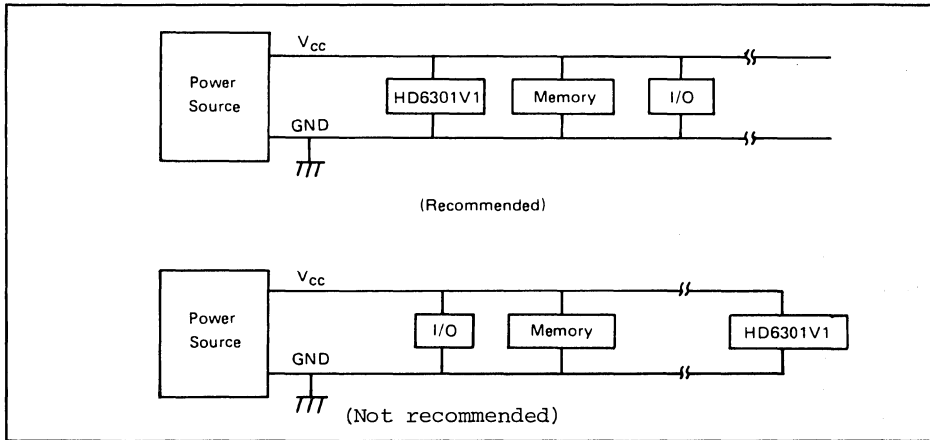


Fig. 7-7-6 Layout of the HD6301V1 on the PC Board

## I. EPROM ON PACKAGE HD63P01M1

## 1. Overview

The HD63P01M1 is an 8-bit CMOS single-chip microcomputer unit, which can use 4k bytes or 8k bytes of EPROM on the package instead of internal ROM. The HD63P01M1 can be used to debug or emulate the HD6301V1 for software development. And also it can be used in low-volume production.

## (1) Features

- Pin Compatible with HD6301V1
- On Chip Function Compatible with HD6301V1
  - 128 Bytes of RAM
  - 29 Parallel I/O
  - 2 Lines of Data Strobe
  - 16 Bit Programmable Timer
  - Serial Communication Interface
  - 2 Interrupt Pins
- Low Power Consumption Mode
  - Sleep Mode, Standby Mode
- Minimum Instruction cycle Time
  - 1 $\mu$ s (f=1MHz)
- Bit Manipulation, Bit Test Instruction
- Protection from System Upset
  - Address Trap, Op-Code Trap
- Applicable to 4k or 8k Bytes of EPROM
  - 4096 Bytes : HN482732A
  - 8192 Bytes : HN482764, HN27C64

## II. 1.5 MHz &amp; 2 MHz Operation in Single Chip Mode of HD63P01M1

HD63P01M1 now in mass production is guaranteed to be operated in 1 MHz. But if it satisfies the conditions below, it can be operated in up to 2 MHz.

Note (1) Only single chip mode (mode 7) is available.

Note (2) The access time is limited when the operating frequency is more than 1 MHz. So, use the EPROM which satisfies the condition below.

While operating in 1.5 MHz, the access time must be less than or equal to 400 ns.

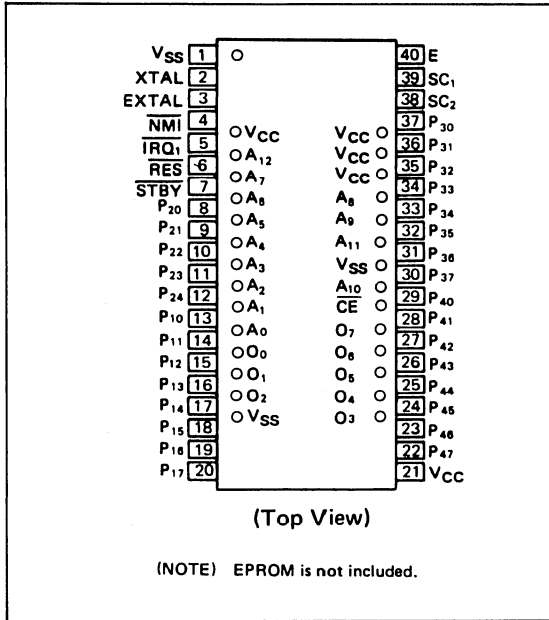
While operating in 2 MHz, the access time must be less than or equal to 250 ns.

Note (3) Temperature Range : Ta=0°C-70°C

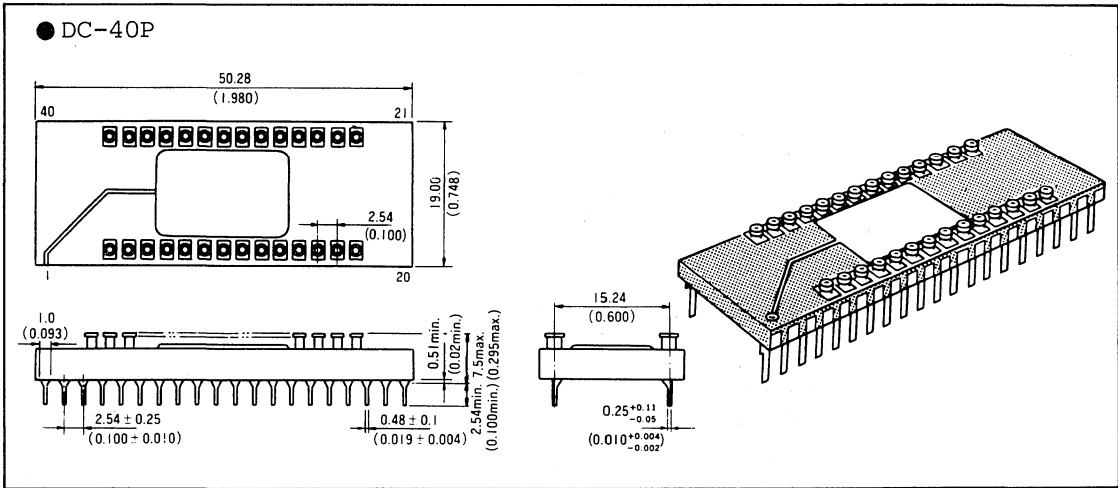
Operating Voltage : V<sub>CC</sub>=5V±10%

Note (4) This data is only for reference, and does not guarantee this characteristic.

(2) Pin Arrangement

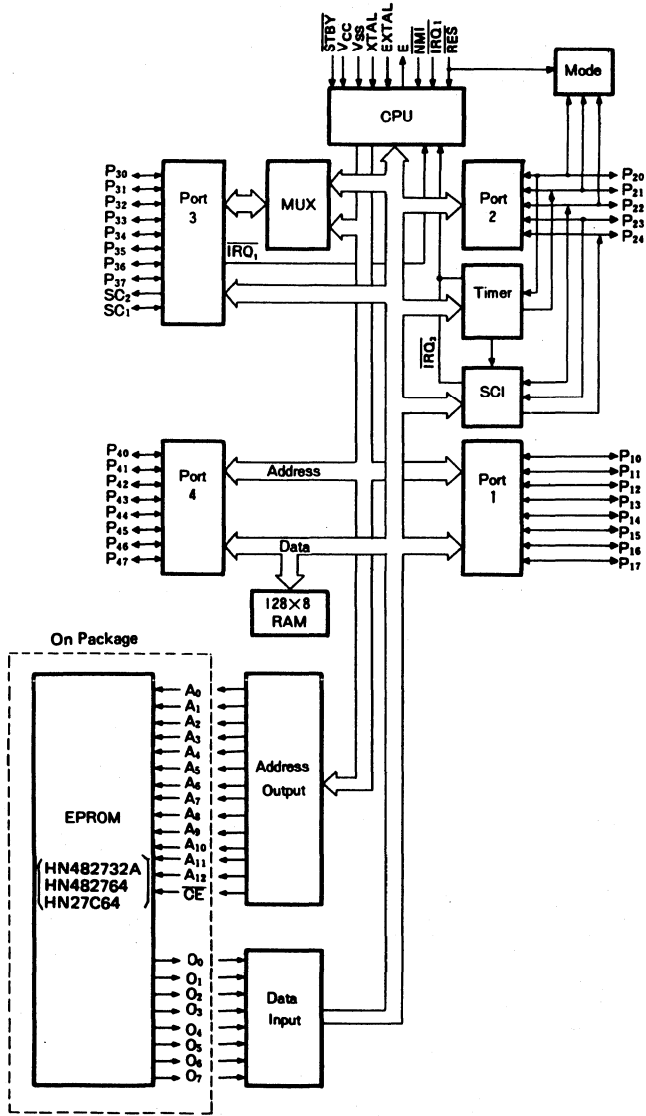


(3) Dimensional Outline



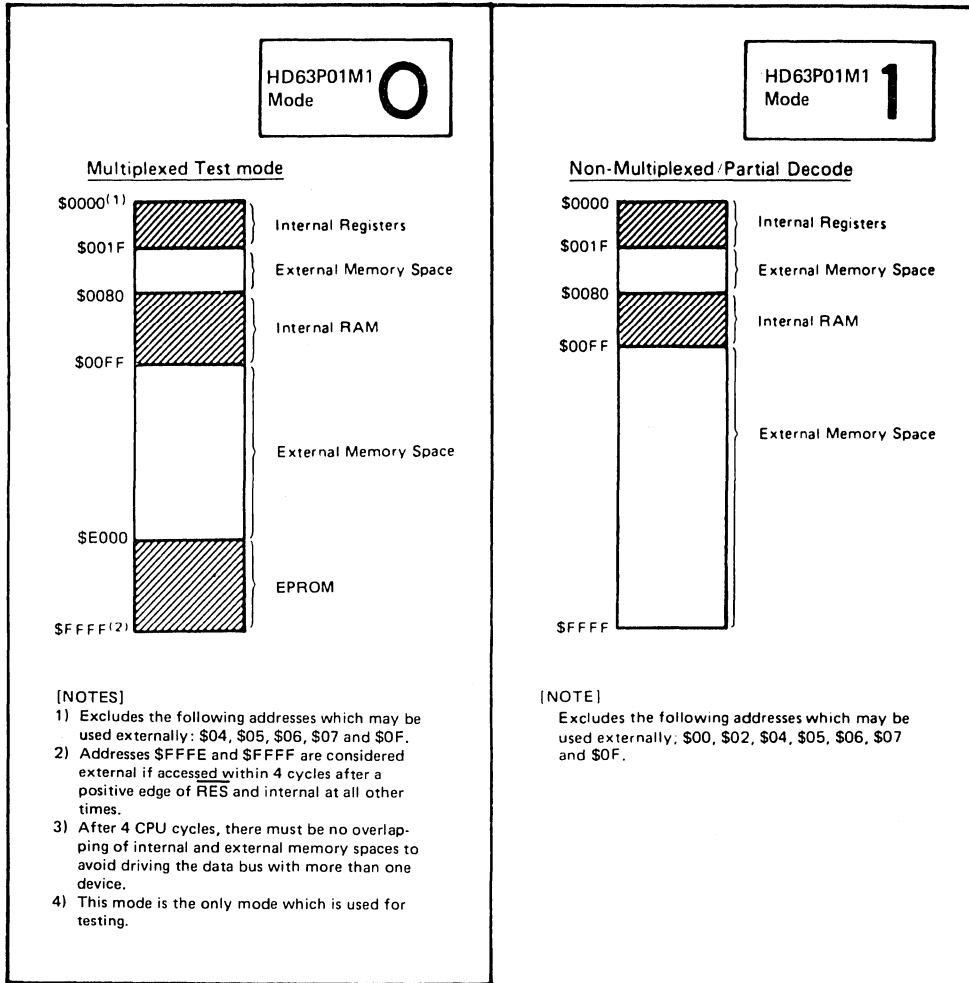
Note) Inch value indicated for your reference

(4) Block Diagram



(5) Memory Map and Operation Mode

The operation mode of the HD63P01M1 is similar to the HD6301V1. As for the memory map, EPROM address space is 8k Bytes (\$E000 to \$FFFF) in the HD63P01M1, while ROM address space is 4k Bytes (\$F000 to \$FFFF) in the HD6301V1.





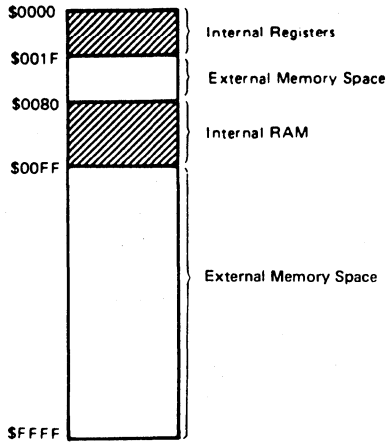
HD63P01M1  
Mode

2

HD63P01M1  
Mode

4

Multiplexed/RAM

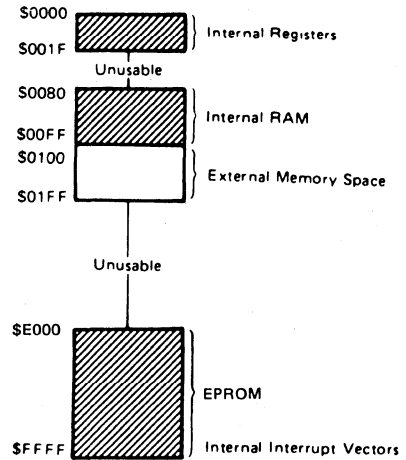


[NOTE] Excludes the following address which may be used externally: \$04, \$05, \$06, \$07, \$0F.

HD63P01M1  
Mode

5

Non-Multiplexed/Partial Decode

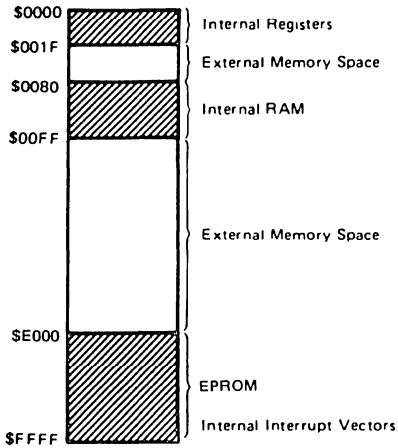


[NOTE] Excludes \$04, \$06, \$0F. These address cannot be used externally.

(to be continued)

HD63P01M1  
Mode **6**

Multiplexed/Partial Decode

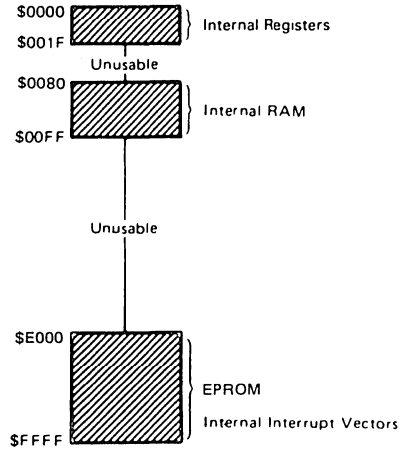


[NOTE]

Excludes the following address which may be used externally: \$04, \$06, \$0F.

HD63P01M1  
Mode **7**

Single Chip



## 2. Precautions to Use the HD63P01M1

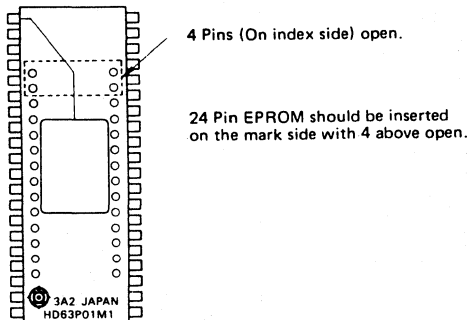
### (1) Precaution to Emulate the HD6301V1 by HD63P01M1

Please use 4k bytes of EPROM address space located from \$F000 through \$FFFF. But do not use 4k bytes from \$E000 through \$EFFF because these addresses are internal for the HD63P01M1, while these are external for the HD6301V1.

### (2) Precaution to Use the EPROM On-Package 8-bit Single Chip Microcomputer

Please pay attention to the followings, since this MCU has special structure with pin socket on the package.

- (a) Don't apply high static voltage or surge voltage over MAXIMUM RATINGS to the socket pins as well as the LSI pins. If not, that may cause permanent damage to the device.
- (b) When using 32k EPROM (24 pin), insert it on the mark side and let the four above pins open.



- (C) When using this in production like mask ROM type single chip microcomputer, pay attention to the followings to keep the good contact between the EPROM pins and socket pins.

- (i) When soldering the LSI on a print circuit board, the recommended condition is

Temperature : lower than 250°C

Time : within 10 sec.

- (ii) Note that the detergent or coating will not get in the socket during flux washing or board coating after soldering, because that may cause bad effect on socket contact.

(iii) Avoid permanent application of this under the condition of vibratory place and system.

(iv) The socket, inserted and pulled repeatedly loses its contactability. It is recommended to use new one when applied in production.

Ask our sales agent about anything unclear.

## II PROGRAM DEVELOPMENT PROCEDURE AND SUPPORT SYSTEM

### 1. Overview

The cross assembler and the hardware emulator using various types of computer are prepared by the company as supporting systems to develop user's programs. User's programs are mask programmed into the ROM and delivered as the LSI by the company.

Fig. II-1 shows the typical program design procedure and Table II-1 shows the system development support tool for the HD6301V1 which are used in these processes.

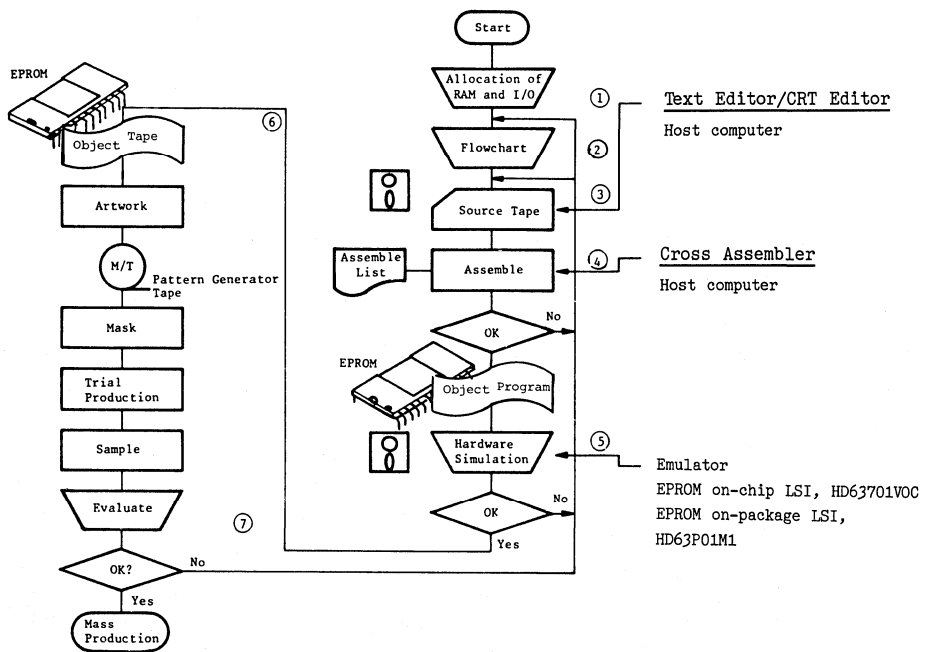


Fig. II-1 Program Design Procedure

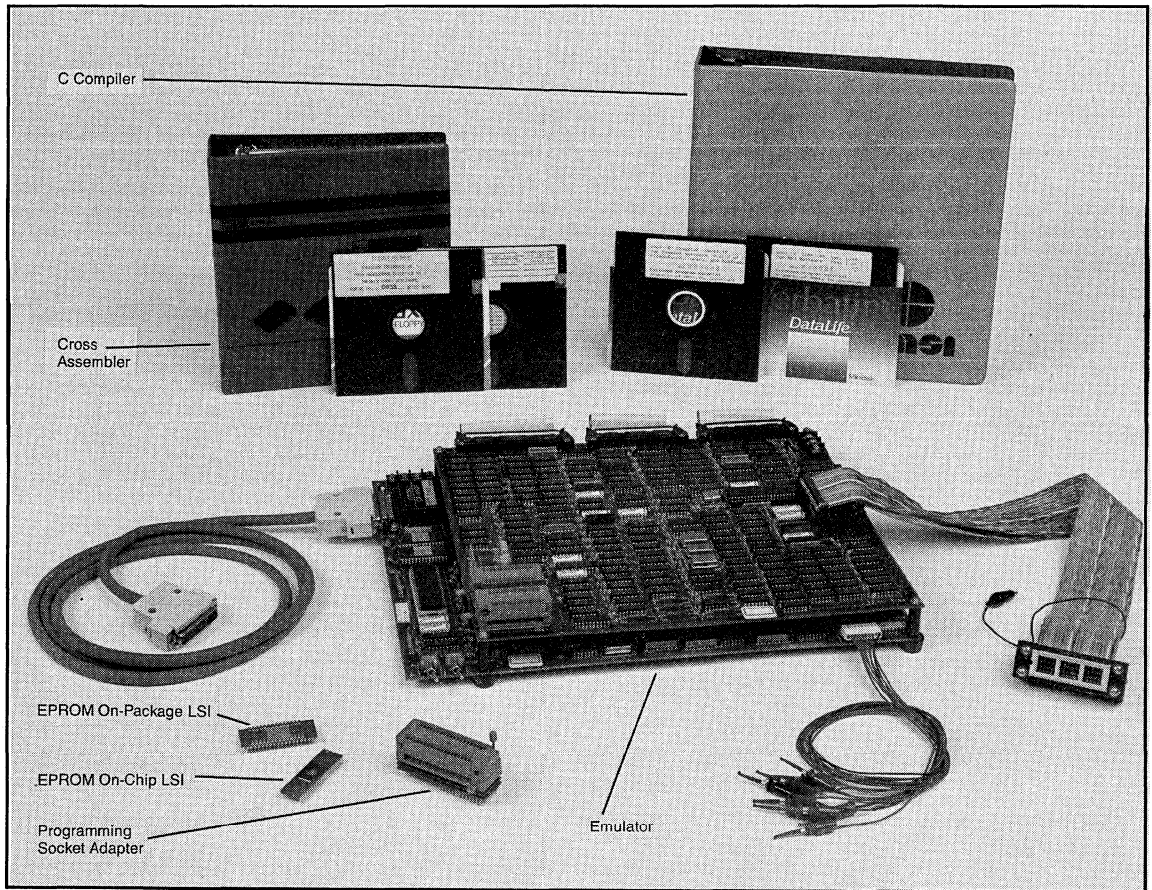
#### (Explanation)

- ① When the user programs the system using the HD6301V1 series, a functional assignment of each I/O pin and an allocation of RAM area should be specified adjusting to designed system before actual programming.
- ② A flowchart is designed to implement the functions and it is coded by using the HD6301V1 mnemonic code.

- ③ Write the software coded according to the flowchart on a floppy disk to make a source program.
  
- ④ Assemble the source program to generate an object program using a computer. Assembly errors are also detected.
  
- ⑤ Verify the program through hardware emulation with an emulator, EPROM on-chip or EPROM on-package type microcomputer.
  
- ⑥ Send the completed program to the company in the form of EPROM. Send "Single-chip microcomputer order specification" and "Mask option list" at that time.
  
- ⑦ ROM and mask option are masked by the company. LSI is testatively produced and the sample is handed in to the user. If a user doesn't see any problem in programming, mass production can be started.

Table II-1 Support Tools

Part No.	Emulator	EPROM on-chip LSI	EPROM on-chip LSI Programming Socket Adapter	EPROM on-package LSI	IBM PC cross assembler	IBM PC C Compiler
HD6301V1, HD6303R, HD6303R1	H31MIX4 (HS31VEML04H)	HD63701VOC	H31VSA01A	HD63P01M1	S31IBMPC	US31PCLI1SF

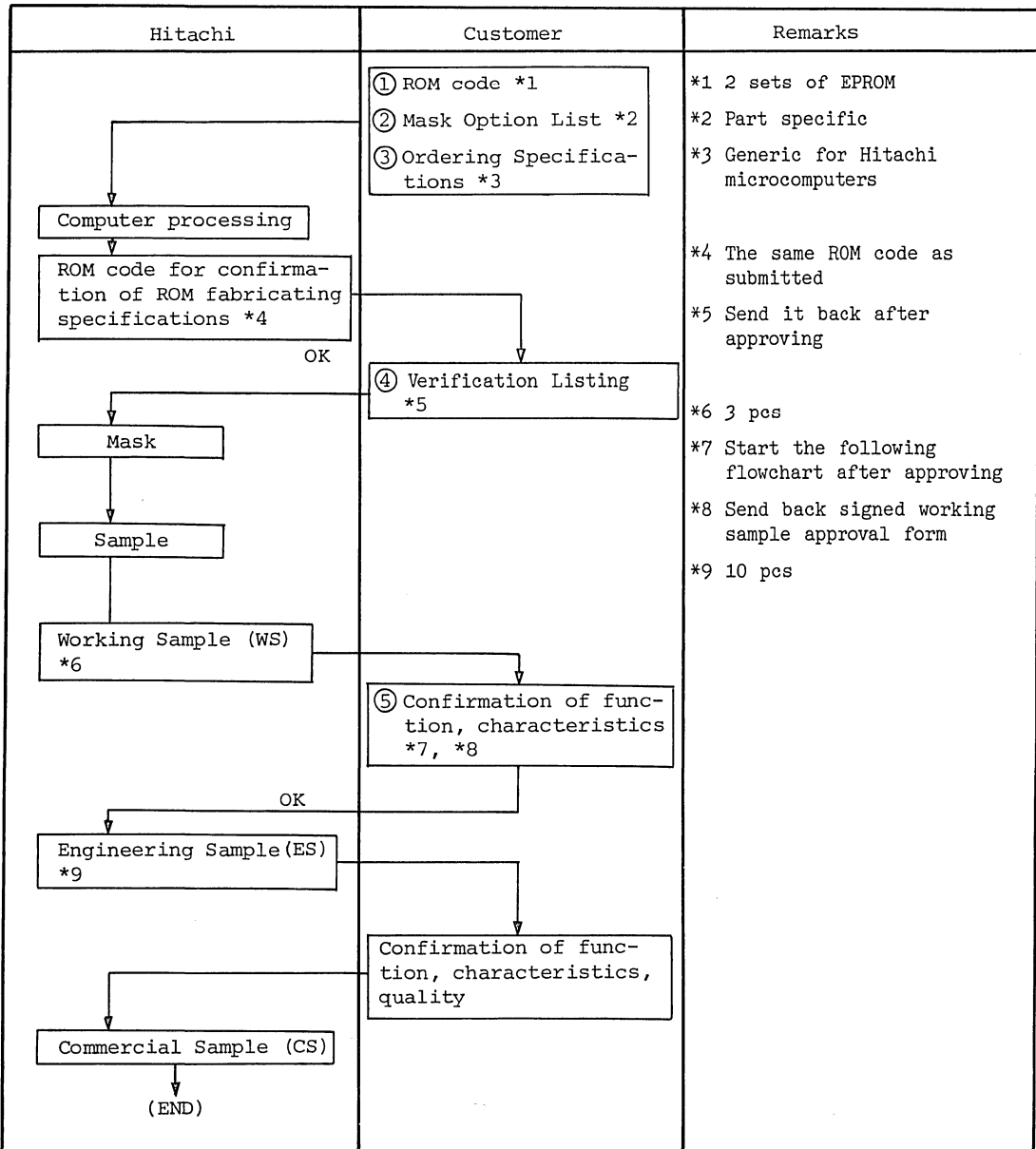


HD6301V1 and HD6303R Development Tools

## 2. Single Chip Microcomputer ROM Ordering Procedure

### (1) Development Flowchart

Single chip microcomputer device is developed according to the following flowchart after program development.



(Note) Please send in ①, ②, and ③ at ROM ordering, and send back ④, ⑤ after approving.

Device Development Flowchart





(2) Data you send and precautions

- (a) Ordering specifications ----- Common style for all Hitachi single chip microcomputer devices. Please enter as for the followings. The format is shown in the next page.

- Basic ITEM
- Environment Check List
- Check List of attached data
- Customer

- (b) ROM code ----- Please send in the ordering ROM code by 2 sets of EPROM the same contents are written. Enter ROM code No. in them. It is desirable to send in program list for easy confirmation of the program contents.

(3) Change of ROM code

Note that if you change the ROM code once send in or other specification, the ROM must be developed from the beginning. The cost of mask charge should be provided again in this case.

(4) Samples and Mass production

(Working Sample) ----- Sample for confirmation of the contents of ROM code and that of mask option. Normally 3 samples are sent, but not guaranteed as for reliability. Please evaluate and approve immediately because the following sample making and mass production are set about after obtaining your evaluation.

(Engineering Sample) ----- Sample for evaluating also reliability. 10 pcs are included in mask charge.

(Commercial Sample) ----- Samples for pre-production which  
maybe purchased separately.

(Mass Product) ----- Products for actual mass produc-  
tion. Please enter the plan of  
mass production in full.

**HD6301V1**  
**ORDERING SPECIFICATIONS**

**(1) GENERAL CHARACTERISTICS** (Fill in blank space or check appropriate box )

Customer		Package Outline (See page 183.)	<input type="checkbox"/> DP-40 <input type="checkbox"/> CP-44
Device Type			<input type="checkbox"/> FP-54 <input type="checkbox"/> CP-52
Application (be specific)			<input type="checkbox"/> CG-40
Customer ROM Code ID		Options/Remarks:	
ZTAT™ Conversion	<input type="checkbox"/> Yes <input type="checkbox"/> No		
ROM Code Media	<input type="checkbox"/> EPROM <input type="checkbox"/> ZTAT™	Must Specify: Customer Programmed Start Address _____ Customer Programmed Stop Address _____	
Operating Temperature	<input type="checkbox"/> Standard <input type="checkbox"/> J (-40° C to +85° C) version if offered		
Remark	<input type="checkbox"/> Yes <input type="checkbox"/> No      Previous Hitachi P/N _____		

**(2) OPERATING CHARACTERISTICS** (Fill in blank space or appropriate box )

LSI Ambient Temperature	Typical	°C	Target Level Of Reliability	<input type="checkbox"/> 1000 Fit	<input type="checkbox"/> (____)
	Range	°C- °C		<input type="checkbox"/> 500 Fit	
LSI Ambient Humidity	Typical	%	Acceptable Quality Level	Electrical	<input type="checkbox"/> 0.25% <input type="checkbox"/> (____)
	Range	%- %		Major Visual	<input type="checkbox"/> 0.65% <input type="checkbox"/> (____)
Power On Duration	Typical	Hours/Day	LSI Operating Speed (Specify MHz or KHz)		
Maximum Applied Voltage To LSI	Power Supply	Max.	Remarks:		
	I/O	Max.			

**(3) ELECTRICAL CHARACTERISTICS** (Fill in blank space or check appropriate box )

<input type="checkbox"/> Purchasing Specifications	<input type="checkbox"/> Hitachi's Standard Specifications
Refer To Data Sheet: _____	

**For Hitachi Use Only**

**(4) CUSTOMER APPROVAL**

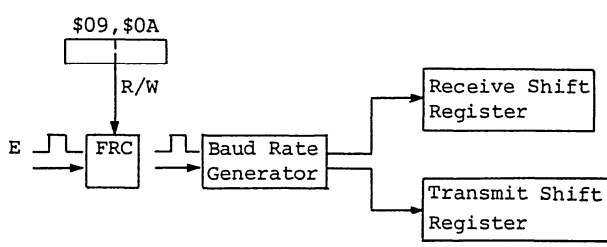
Customer Name _____
PO# _____
Approved By (print) _____
Approved By (signature) _____
Date _____

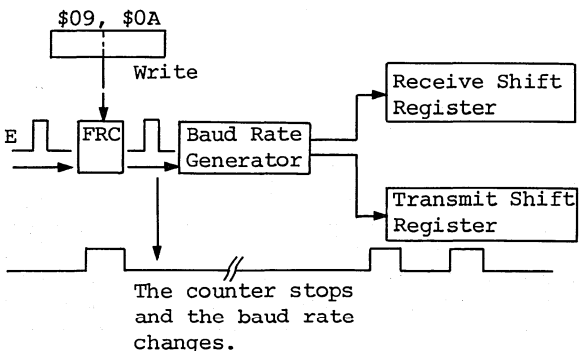
**(5) ROM CODE VERIFICATION**

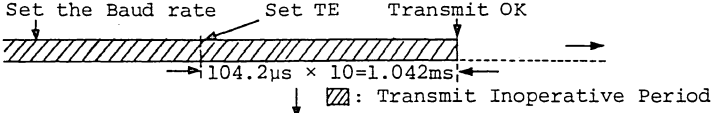
LSI Type No.	
Shipping Date of ROM To Customer	
Approved Date of ROM From Customer	

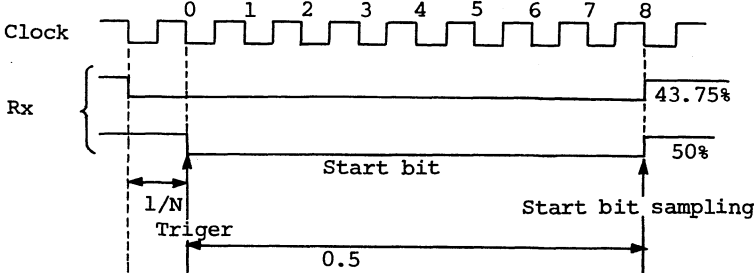


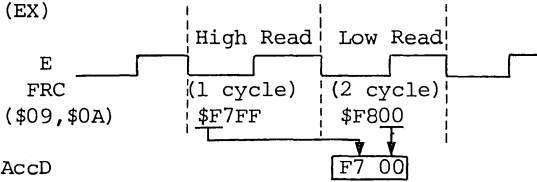
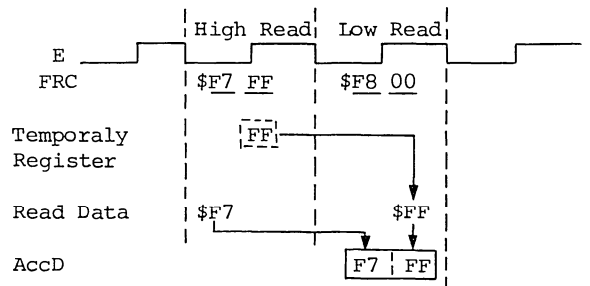
III Q & A

Theme	Question	Answer
Process to Use a Port as an Outputs	When using an I/O port as an output, is the data stored to the Data Register or is the Data Direction Register (DDR) set at first?	Store the data to the Data Register at first and then set DDR (DDR=1); if not, unknown data is output from the port.
Relation between Writing into the FRC and SCI Operation	How are writing into the timer Free Running Counter(FRC) and the Serial Communication Interface(SCI) related?	<p>The source of the clock input to the SCI Shift Registers is the timer FRC. Therefore, if new data is written into the FRC, SCI operations are disturbed. See the following diagram.</p>  <p>* A write into the FRC is prohibited during SCI operations.</p>
Writing into the FRC during Serial Receive/Transmit	Is it prohibited to write data into the Free Running Counter(FRC) during serial receive/transmit?	Yes. If data is written into the FRC during serial receive/transmit, the FRC stops counting up and the baud rate changes. In condition other than serial receive/transmit, it's possible to write.

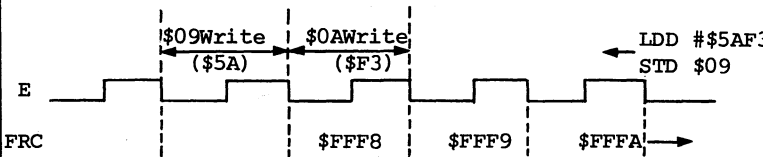
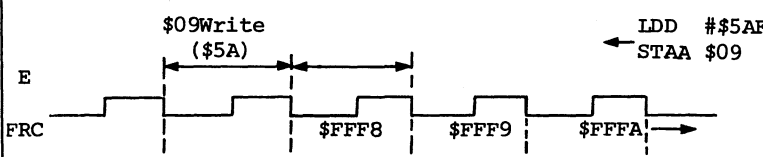
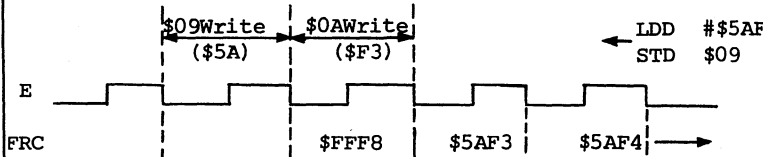
Theme	Question	Answer																											
		 <p>The counter stops and the baud rate changes.</p>																											
RDRF State When SCI Receiving	<p>What is the state of the Receive Data Register Full (RDRF) flag when the HD6301V1/HD6303R SCI can receive signals (RE=1) and the wake-up flag (WU bit) is set?</p> <p>TRCSR</p> <table border="1" data-bbox="381 740 946 843"> <tr> <td></td> <td>7</td> <td>6</td> <td>5</td> <td>4</td> <td>3</td> <td>2</td> <td>1</td> <td>0</td> </tr> <tr> <td>\$0011</td> <td>RDRF</td> <td>ORFE</td> <td>TDRE</td> <td>RIE</td> <td>RE</td> <td>TIE</td> <td>TE</td> <td>WU</td> </tr> <tr> <td></td> <td>?</td> <td></td> <td></td> <td></td> <td>1</td> <td></td> <td></td> <td>1</td> </tr> </table>		7	6	5	4	3	2	1	0	\$0011	RDRF	ORFE	TDRE	RIE	RE	TIE	TE	WU		?				1			1	<p>When the wake-up flag is set (WU=1) the RDRF flag cannot be set. (RDRF=0)</p>
	7	6	5	4	3	2	1	0																					
\$0011	RDRF	ORFE	TDRE	RIE	RE	TIE	TE	WU																					
	?				1			1																					

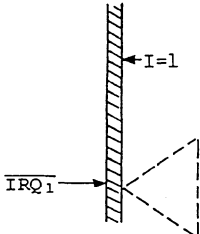
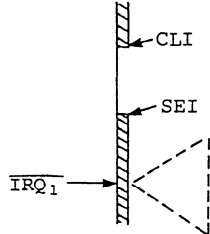
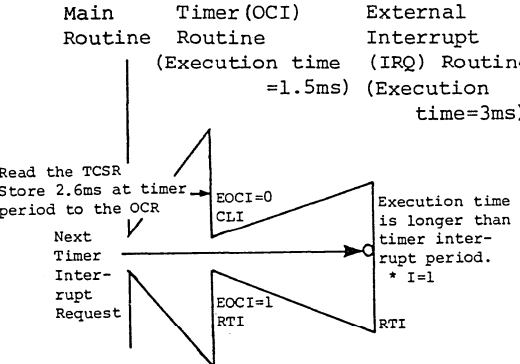
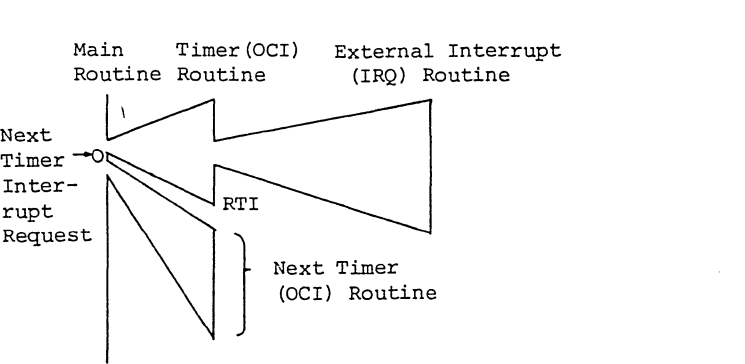
Theme	Question	Answer
<p>Serial I/O Operation</p>	<p>The serial I/O does not operate satisfactorily. Initialization does not seem to be wrong, but the data is not transmitted. What is wrong?</p> <p>Initialize by User Program</p> <ol style="list-style-type: none"> <li>(1) Set the Rate/Mode Control Register (RMCR) to the desired operation.</li> <li>(2) Set the Transmit/Receive Control Status Register (TRCSR) to the desired operation.</li> </ol>	<p>Just after the initialization of serial I/O, the data transmit is not operative during 10 cycles of Baud Rate after setting the TE. The reason is as follows. Setting the transmit enable bit (TE bit) causes ten consecutive "1" of preamble and makes the transmitter section operative. In other words, the transmitter section gets ready after one frame (10 bits) transmitting time according to the Baud rate.</p> <p>(ex.) When the Baud rate is set to 9600 Baud (104.2μs at 1 bit),</p>  <p>1.042ms after setting the TE, the transmitter section is operative.</p>
<p>Serial I/O Register Read</p>	<p>When transmitting the data, is reading the Transmit/Receive Control Register (TRCSR) required?</p> <p>When the transfer interval is long enough compared with the Baud rate, Transmit Data Register Empty (TDRE) will be set. In that case, are there any problems when transmitting data without checking the TDRE flag in the TRCSR?</p>	<p>The TDRE flag shows if the TDRE register is empty or not. When writing a data to the TDR with TDRE=1, it's not necessary to check the TDRE. But reading the TDRE flag tells us the contents of TDR. For example, when new data is written to the TDR with TDRE "0" (TDR already has a data); the old data will be erased. When the transfer interval is long enough compared with the Baud rate, there's no problem. However, check TRCSR if possible.</p>

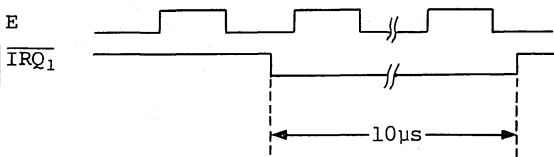
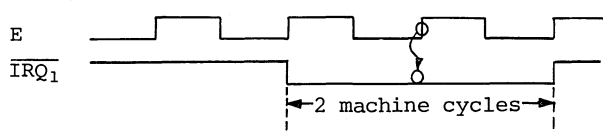
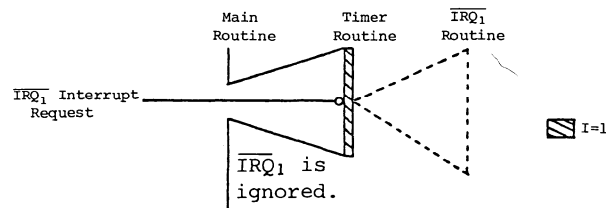
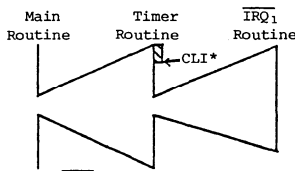
Theme	Question	Answer												
Detection of the HD6301V1 Serial Start Bit	(1) What is the relation between the HD6301V1 serial sampling clock frequency and the baud rate ? (2) What does "Sampling error" mean ?	<p>(1) The serial sampling clock frequency is eight times the baud rate.</p> <p>(2) "Sampling error" means receive margin at the serial operation time.</p> <p><u>Receive margin:</u>            The HD6301V1 detects the start bit and samples the data bit using the falling edge of the sampling clock. The general equation is shown as follows.</p> <p>1.) General equation  <math display="block">M = [ (0.5 - 1/N) - (D - 0.5)/N - (L - 0.5)F ] \times 100 (\%)</math>           M: Receive margin            N: Ratio of baud rate to sampling clock (0 to 0.5)            D: Duty of the longer sampling clock of "H", and "L"            L: Frame length (7 to 12 bits)            F: Absolute value of deviation of sampling clock frequency</p> <p>2.) Abbreviated equation  <math display="block">M = (0.5 - 1/N) \times 100 (\%)</math>           Conditions: D = 0.5, F = 0</p> <table border="1" data-bbox="906 819 1647 902"> <thead> <tr> <th>N</th> <th>8</th> <th>16</th> <th>32</th> <th>64</th> <th>Note</th> </tr> </thead> <tbody> <tr> <td>M (%)</td> <td>37.5</td> <td>43.75 (Fig.1)</td> <td>46.875</td> <td>48.4375</td> <td>In the HD6301V1, N=8.</td> </tr> </tbody> </table> <p>Figure 1  </p>	N	8	16	32	64	Note	M (%)	37.5	43.75 (Fig.1)	46.875	48.4375	In the HD6301V1, N=8.
N	8	16	32	64	Note									
M (%)	37.5	43.75 (Fig.1)	46.875	48.4375	In the HD6301V1, N=8.									

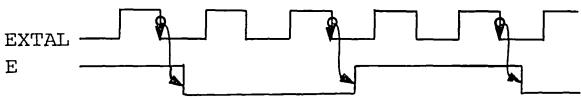
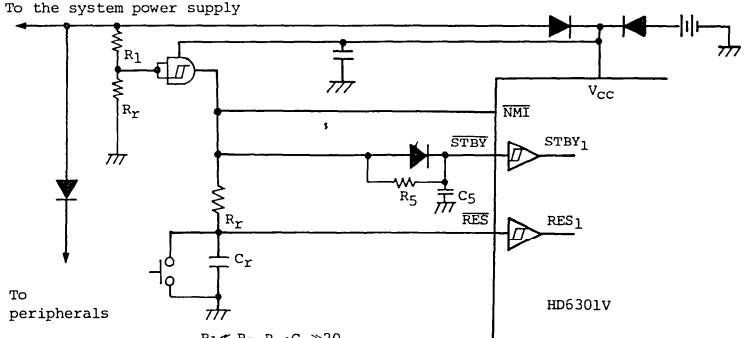
Theme	Question	Answer						
<p>Free Running Counter Read</p>	<p>When the FRC of the HD6301V1/HD6303R is read with the double byte load instructions (2 cycle execution for FRC reading), is it read correctly? Double byte load instructions require two cycles to be executed and the cycle to read the low byte of FRC becomes the next cycle of the high byte. Is it OK?</p> <p>(EX)</p>  <p>(When reading \$F7FF from the counter)</p>	<p>The FRC of the HD6301V1/HD6303R contains a parallel temporary register. When the high byte of the FRC is read, the low byte is set in the temporary register. The Low byte data in the temporary register is set to the AccD at the next cycle. Therefore, it is possible to read the FRC correctly.</p>  <p>(When reading \$F7FF from the counter)</p>						
<p>Preset Method of the Free Running Counter</p>	<p>What is the difference between the HD6801V and HD6301V1 in writing data into the free running counter ?</p>	<p>The FRC preset method of the HD6801V is different from the HD6301V1.</p> <table border="1" data-bbox="911 766 1645 1014"> <thead> <tr> <th>Type</th> <th>Preset Method</th> </tr> </thead> <tbody> <tr> <td>HD6801V</td> <td>The FRC is always preset to "\$FFF8".</td> </tr> <tr> <td>HD6301V1</td> <td> <ol style="list-style-type: none"> <li>1. Writing to the high byte presets the FRC to \$FFF8.</li> <li>2. The FRC is set to desirable data by a double byte store instruction.</li> </ol> </td> </tr> </tbody> </table>	Type	Preset Method	HD6801V	The FRC is always preset to "\$FFF8".	HD6301V1	<ol style="list-style-type: none"> <li>1. Writing to the high byte presets the FRC to \$FFF8.</li> <li>2. The FRC is set to desirable data by a double byte store instruction.</li> </ol>
Type	Preset Method							
HD6801V	The FRC is always preset to "\$FFF8".							
HD6301V1	<ol style="list-style-type: none"> <li>1. Writing to the high byte presets the FRC to \$FFF8.</li> <li>2. The FRC is set to desirable data by a double byte store instruction.</li> </ol>							



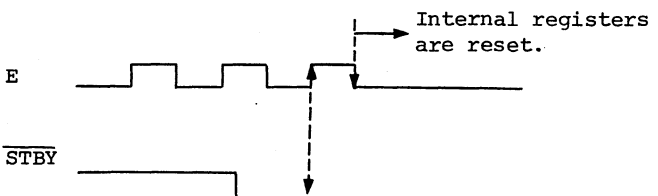
Theme	Question	Answer
		<p>(1) The HD6801V Preset Method</p>  <p>The FRC is always preset to \$FFF8.</p> <p>(2) The HD6301V1 Preset Method</p> <p>1. \$FFF8</p>  <p>Writing to the high byte presets the FRC to \$FFF8.</p> <p>2. Optional valve (In this case \$5AF3)</p>  <p>The FRC is set to desirable data (\$5AF3) by a double byte store instruction.</p>
Output of Address Strobes (AS) in the Multiplexed Mode	Is AS always output when using the HD6301V1 in the expanded multiplexed mode (mode 2, 4, 6)?	Yes. AS is always output in the expanded multiplexed mode, even when the MPU accesses the internal RAM, ROM, etc.

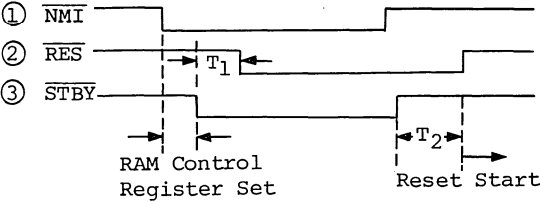
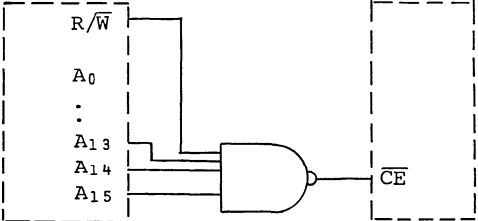
Theme	Question	Answer
$\overline{IRQ}_1$ Acceptance	(1) Is $\overline{IRQ}_1$ ignored when the Condition Code Register I mask is set? (2) After the I mask is reset, will the interrupt sequence start by the interrupt request flag having been latched?	(1) If the Condition Code Register I mask is set, $\overline{IRQ}_1$ is completely ignored. (2) With the I mask set, the interrupt request flag will not be latched. <div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: center;"> <p>(1) Reset starts</p>  <p><math>\overline{IRQ}_1</math> is ignored.</p> </div> <div style="text-align: center;"> <p>(2) Reset starts</p>  <p><math>\overline{IRQ}_1</math> is ignored.</p> </div> </div>
Timer Interrupt and External Interrupt	<p>In the routine below, when is the next timer interrupt accepted?</p> <div style="display: flex; justify-content: space-around; text-align: center;"> <div> <p>Main Routine</p> </div> <div> <p>Timer (OCI) Routine (Execution time =1.5ms)</p> </div> <div> <p>External Interrupt (IRQ) Routine (Execution time=3ms)</p> </div> </div> 	<p>The next timer interrupt is accepted in the main routine just after RTI instruction execution.</p> <div style="display: flex; justify-content: space-around; text-align: center;"> <div> <p>Main Routine</p> </div> <div> <p>Timer (OCI) Routine</p> </div> <div> <p>External Interrupt (IRQ) Routine</p> </div> </div> 

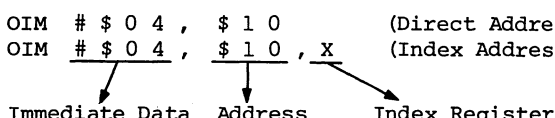
Theme	Question	Answer
<p><math>\overline{IRQ_1}</math> Interrupt and Other Interrupts</p>	<p><math>\overline{IRQ_1}</math> pin (pin 5) is held at low for <math>10\mu s</math> but an interrupt does not occur. What should be done to generate an interrupt sequence?</p> 	<p>(1) <math>\overline{IRQ_1}</math> is a level sensitive interrupt pin which needs a minimum of 2 machine cycles (<math>2\mu s</math> at <math>1MHz</math>) to accept an interrupt. However, if another interrupt has been already generated, no interrupt request is accepted with <math>\overline{IRQ_1}</math> at low for <math>10\mu s</math>. In such a case, <math>\overline{IRQ_1}</math> should be held at low until the request is accepted.</p>  <p>(2) In this case, as a timer interrupt is executed the interrupt mask is automatically set. So <math>\overline{IRQ_1}</math> is ignored. See the followings for the illustration of <math>\overline{IRQ_1}</math> and other interrupts and a countermeasure.</p> <p><math>\overline{IRQ_1}</math> and Other Interrupts</p>  <p>Countermeasure</p> <p>Clear the I mask at the beginning of the timer interrupt routine.</p>  <p><math>\overline{IRQ_1}</math> is acceptable.</p> <p>*CLI : Clears the interrupt mask (I=0).</p> <p>With this method, note the following ;</p> <ol style="list-style-type: none"> <li><math>\overline{IRQ_1}</math> may be ignored when the request occurs during timer interrupt vectoring.</li> <li>Interrupts from <math>NMI</math> or <math>SWI</math> are excluded.</li> </ol>

Theme	Question	Answer				
<p>CLI Instruction and Interrupt Operation</p>	<p>In the HD6301V1, a timer interrupt is not accepted in the following program. Is there any problem?</p> <pre style="border: 1px dashed black; padding: 10px; margin: 10px auto; width: fit-content;"> Main Routine L01 CLI       NOP       SEI       :       :       BRA L01                     </pre>	<p>To accept an interrupt, two machine cycles are necessary between CLI and SEI. That is, in this program, two NOP instructions are necessary. The same thing can be said when using TAP for CLI and SEI.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%;">Using CLI</th> <th style="width: 50%;">Using TAP</th> </tr> </thead> <tbody> <tr> <td style="border: 1px dashed black; padding: 5px;"> <pre> L01 CLI       NOP       NOP       SEI       :       :       BRA L01                     </pre> </td> <td style="border: 1px dashed black; padding: 5px;"> <pre> TAP (Clears the I mask) NOP NOP TAP (Sets the I mask) : : :                     </pre> </td> </tr> </tbody> </table>	Using CLI	Using TAP	<pre> L01 CLI       NOP       NOP       SEI       :       :       BRA L01                     </pre>	<pre> TAP (Clears the I mask) NOP NOP TAP (Sets the I mask) : : :                     </pre>
Using CLI	Using TAP					
<pre> L01 CLI       NOP       NOP       SEI       :       :       BRA L01                     </pre>	<pre> TAP (Clears the I mask) NOP NOP TAP (Sets the I mask) : : :                     </pre>					
<p>Relation between the External Clock (EXTAL Clock) and Enable Clock (E Clock)</p>	<p>With which edges of the EXTAL clock does the E clock change synchronously, rising edge (↑) or falling edge (↓)?</p>	<p>It changes synchronously with the falling edge (↓) of the EXTAL clock.</p> 				
<p>Constants of the Reset Circuit</p>	<p>Does the capacitor of the recommended reset circuit in the HD6303R (HD6301V1) have an upper limit?</p>	<p>Capacitor Cr does not have upper limit because of the Schmitt trigger circuit provided with the RES.</p> <p>Available if <math>R_r \cdot C_r \gg 20\text{ms}</math></p>  <p style="text-align: center;"><math>R_1 \ll R_2, R_r \cdot C_r \gg 20\text{ms}</math></p>				



Theme	Question	Answer
Port Output After Resetting	What data does a port output when the Data Direction Register (DDR)=1 after resetting?	After resetting, since the Data Register of a port is undefined, undefined data is output when the DDR=1. Input definite data by programming in the Data Register before setting the DDR=1.
Schmitt Trigger Circuit of $\overline{STBY}$	Is the Schmitt trigger circuit provided with the HD6301V1 $\overline{STBY}$ ?	Yes.
Return from Standby Mode	What occurs when returning from the standby mode without using $\overline{RES}$ ?	The CPU does not operate normally because the contents of each register are not definite. Therefore, always use the $\overline{RES}$ when returning from the standby mode.
Going into the Standby Mode	Does the CPU go into the standby mode after current instruction execution is completed?	<p>No. Because there is no connection between the instruction execution sequence and the standby mode. That is, when the <math>\overline{STBY}</math> pin goes into "Low", the state is latched at the next rising edge of E clock. Then the internal registers are reset at the next falling edge.</p> 

Theme	Question	Answer
<p>Timing for the Standby Mode</p>	<p>The timing for the standby mode is shown in the HD6301V user's manual. <math>T_1</math> is not defined. How long is <math>T_1</math>?</p>  <p>① <math>\overline{\text{NMI}}</math>          ② <math>\overline{\text{RES}}</math>          ③ <math>\overline{\text{STBY}}</math></p> <p>RAM Control Register Set      Reset Start</p> <p><math>T_1</math>      <math>T_2</math></p> <p><math>T_2</math> : Oscillation Stabilization Time</p>	<p>After the RAM Control Register is set in the NMI routine, either <math>\overline{\text{STBY}}</math> or <math>\overline{\text{RES}}</math> can be in the low state with no priority.</p>
<p>Usage of EPROM Socket Pins for the HD63P01M (No.1)</p>	<p>Are the data buses of the EPROM socket pins for the HD63P01M bi-directional in order to access not only the EPROM but the RAM?</p>	<p>The data bus output from EPROM socket pins for the HD63P01M is Read only.</p>
<p>Usage of EPROM Socket Pins for the HD63P01M (No.2)</p>	<p>In EPROM socket pins for the HD63P01M, what is <math>\overline{\text{CE}}</math> composed of?</p>	<p><math>\overline{\text{CE}}</math> is a NAND circuit of the address bus (<math>A_{13}</math> to <math>A_{15}</math>) and the MCU internal <math>R/\overline{W}</math> signal.          (Refer below.)          Therefore, <math>\overline{\text{CE}}</math> does not output in the dummy cycle.          (When not accessing EPROM of HD63P01M)</p> 

Theme	Question	Answer																	
Usage of EPROM Socket Pins for the HD63P01M (No.3)	With EPROM socket pins for the HD63P01M, (1) Can pins drive one TTL load or more? (2) If not, what can pins drive?	(1) The current of each pin is too little to drive one TTL load. (2) Each pin can drive one NMOS load.																	
Usage of Bit Manipulator Instructions	How the bit manipulation instructions of the HD6301V should be written?	<p>They are written as follows;</p> <p>OIM # \$ 0 4 , \$ 1 0 (Direct Addressing)            OIM # \$ 0 4 , \$ 1 0 , X (Index Addressing)</p> <p style="text-align: center;">  </p> <p>This is an example of OR operation of the immediate data and the memory and storing the result in the memory. The HD6301V has the following bit manipulation instructions.</p> <p>OIM .... (IMM) · (M) → (M)            AIM .... (IMM) + (M) → (M)            EIM .... (IMM) ⊕ (M) → (M)            TIM .... (IMM) - (M)</p> <p>These instructions are written in the same way.</p> <p>The following bit manipulations have different mnemonics in the same OP code.</p> <table border="1" data-bbox="899 895 1675 1210"> <thead> <tr> <th colspan="2" rowspan="2">OP code</th> <th colspan="2">Bit Manipulation Instruction</th> <th rowspan="2">Function</th> </tr> <tr> <th>Mnemonics</th> <th></th> </tr> </thead> <tbody> <tr> <td>71</td> <td>61</td> <td>A I M</td> <td>B C L R</td> <td>0 → Mi The memory bit i (i=0 to 7) is cleared and the other bits don't change.</td> </tr> <tr> <td>72</td> <td>62</td> <td>O I M</td> <td>B S E T</td> <td>1 → Mi The memory bit i (i=0 to 7) is set and the other bits don't change.</td> </tr> </tbody> </table>	OP code		Bit Manipulation Instruction		Function	Mnemonics		71	61	A I M	B C L R	0 → Mi The memory bit i (i=0 to 7) is cleared and the other bits don't change.	72	62	O I M	B S E T	1 → Mi The memory bit i (i=0 to 7) is set and the other bits don't change.
OP code		Bit Manipulation Instruction			Function														
		Mnemonics																	
71	61	A I M	B C L R	0 → Mi The memory bit i (i=0 to 7) is cleared and the other bits don't change.															
72	62	O I M	B S E T	1 → Mi The memory bit i (i=0 to 7) is set and the other bits don't change.															

Theme	Question	Answer				
		75	65	E I M	B T G L	$M_i \rightarrow \overline{M_i}$ The memory bit $i$ ( $i=0$ to $7$ ) is inverted and the other bits don't change.
		7B	6B	T I M	B T S T	$1 \cdot M_i$ AND operation test of the memory bit $i$ ( $i=0$ to $7$ ) and "1" is executed and its corresponding condition code is changed.

↑ Direct Addressing      ↘ Index Addressing

The mnemonics mentioned above can be written as follows.

BCLR 3, \$10 ↔ AIM #F7, \$10 (Direct Addressing)

BCLR 3, \$10, X ↔ AIM #F7, \$10, X (Index Addressing)

BSET 3, \$10 ↔ OIM #08, \$10 (Direct Addressing)

BSET 3, \$10, X ↔ OIM #08, \$10, X (Index Addressing)

↓ Bit Address      ↘ Index Register



Theme	Question	Answer
Usage of Bit Manipulation Instructions to the Port	Are the bit manipulation instructions (AIM, OIM, EIM, TIM) executable when a port is in the output state (DDR=1)?	<p>It can be used if the port is in the output state (DDR=1). However, the bit manipulation instruction is executed as follows ;</p> <ol style="list-style-type: none"> <li>1 Reads specified address.</li> <li>2 Executes logical operation.</li> <li>3 Writes the result into the specified address.</li> </ol> <p>Since the specified address(1) reads the pin state of the port, the data is influenced by the pins even if any data is output from the port.</p>
RAM Access Disable during Program Execution	<p>When executing a program with the RAME bit of the RAM Control Register disabled,</p> <ol style="list-style-type: none"> <li>(1) What occurs if the internal RAM address is accessed?</li> <li>(2) What occurs if the interrupt requests are generated?</li> </ol>	<ol style="list-style-type: none"> <li>(1) The external RAM can be accessed; the internal RAM is neither readable nor writable when the RAME bit is disabled.</li> <li>(2) If there is no stacking area other than the internal RAM, the MPU will burst when returning from the interrupt sequence.</li> </ol>

# HD6301/HD6303 SERIES HANDBOOK

## Section Four

# HD63701V User's Manual



**Section 4**  
**HD63701V User's Manual**  
**Table of Contents**

	<b>Page</b>
1. OVERVIEW .....	251
1.1 Features of HD63701V0 .....	251
1.2 Block Diagram .....	251
1.3 Functional Pin Description .....	252
2. INTERNAL ARCHITECTURE .....	253
2.1 Mode Selection .....	253
2.2 Memory Map .....	258
2.3 CPU Registers .....	263
2.4 Ports .....	265
2.5 Timer .....	269
2.6 Serial Communication Interface .....	273
2.7 Interrupts .....	281
2.8 Reset .....	283
2.9 Oscillator .....	285
2.10 Strobe Signals .....	286
2.11 RAM Control Register .....	287
2.12 Low Power Consumption Mode .....	287
2.13 TRAP Function .....	292
3. EPROM (PROM) PROGRAMMING AND TECHNICAL SPECIFICATIONS ...	293
3.1 PROM Mode .....	293
3.2 Programming/Verification .....	294
3.3 Erasure .....	295
3.4 On-Chip PROM Characteristics and Application .....	296
3.5 Instruction Set Overview .....	300
3.6 Addressing Modes .....	300
3.7 Instruction Set .....	302
3.8 Instruction Execution Cycles .....	307
3.9 System Flowchart .....	313
3.10 Pin Arrangement and Package Information .....	315
3.11 Electrical Characteristics .....	316

4.	APPLICATIONS .....	322
4.1	Use of External Expanded Mode .....	322
4.2	Standby Mode .....	327
4.3	Address Trap, OP-Code Trap Application .....	330
4.4	Slow Memory Interface .....	331
4.5	Interface to HN61256 .....	332
4.6	Interface to the Realtime Clock (HD146818) .....	337
4.7	Reference Data of Battery Service Life .....	340
5.	PRECAUTIONS .....	341
5.1	Write Only Register .....	341
5.2	Address Strobe (AS) .....	341
5.3	Mode 0 .....	341
5.4	Trap Interrupt .....	341
5.5	Precaution to the Board Design of Oscillation Circuit .....	343
5.6	Application Note for High Speed System Design Using the HD63701V0 ..	343
5.7	Differences between HD6301V and HD63701V .....	347
APPENDIX		
I.	Support Tools .....	353
II.	Q & A .....	355

## 1. OVERVIEW

### 1.1 Features of HD63701V0

The HD63701V0 provides the following features:

- Compatible with the HD6301V
- Expanded instruction set of the HD6801 family
- Abundant on-chip functions compatible with the HD6801/HD6301 family: 4k-byte of EPROM, 192-byte of RAM, 29 parallel I/O Lines, 2 data strobe Lines, 16-bit timer, serial communication interface
- Low power consumption mode: sleep/standby mode
- Minimum instruction execution time: 1 $\mu$ s (f = 1MHz), 0.67 $\mu$ s (f=1.5MHz), 0.5 $\mu$ s (f=2MHz)
- Bit manipulation and bit test instruction
- Error detection: Address trap and op-code trap
- Address space up to 65k words
- Wide operation range:  
f = 0.1 to 2.0MHz (V<sub>CC</sub> = 5V  $\pm$  10%)
- TTL compatible input/output

### 1.2 Block Diagram

A block diagram of HD63701V0 is given in Fig. 1-2-1.

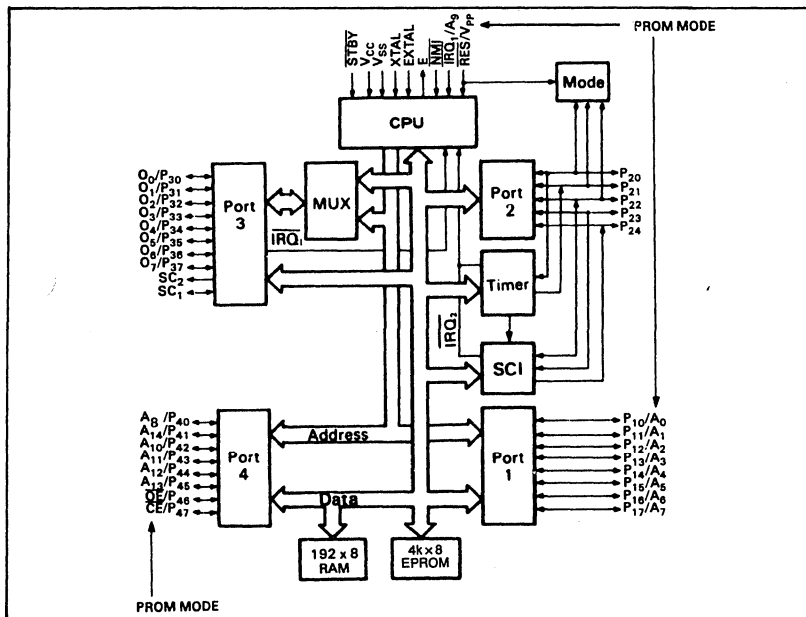


Fig. 1-2-1 HD63701V0 Block Diagram

### 1.3 Functional Pin Description

Table 1-3-1 lists the pin functions. Refer to "2. INTERNAL ARCHITECTURE" for more details.

Table 1-3-1 Pin Functions

Pin	Function				
VCC, VSS	Power supply and GND pins				
XTAL EXTAL	Crystal connection pin. When external clock is used, input it to EXTAL, and XTAL should be open.				
RES	Reset input pin. When this pin is asserted "Low", MCU is set to reset state.				
STBY	Standby input pin. When this pin is asserted "Low", MCU is set to standby state.				
NMI	Edge sensitive (negative edge) non-maskable interrupt input pin.				
IRQ1	Level sensitive maskable interrupt input pin (active Low).				
E	System clock output pin. The frequency is 1/4 of the crystal oscillator frequency.				
P <sub>20</sub> /TIN	5-bit I/O port	Timer input-capture input pin			
P <sub>21</sub> /TOUT		Timer output-compare output pin			
P <sub>22</sub> /SCLK		SCI clock I/O port			
P <sub>23</sub> /RX		SCI receiving pin			
P <sub>24</sub> /TX		SCI transmitting pin			
Following pins function depending on each operation mode					
	Mode 0,2	Mode 1	Mode 5	Mode 6	Mode 7
PORT 1	8-bit I/O port	Lower address (A <sub>0</sub> ~A <sub>7</sub> )	8-bit I/O port	←	←
PORT 3	Data (D <sub>0</sub> ~D <sub>7</sub> ) Lower address (A <sub>0</sub> ~A <sub>7</sub> ) Multiplexed Bus	Data Bus D <sub>0</sub> ~D <sub>7</sub>	←	Data (D <sub>0</sub> ~D <sub>7</sub> ) Lower address (A <sub>0</sub> ~A <sub>7</sub> ) Multiplexed Bus	←
PORT 4	Upper address (A <sub>8</sub> ~A <sub>15</sub> )	←	Lower address (A <sub>0</sub> ~A <sub>7</sub> ) or Input-only pin	Upper address (A <sub>8</sub> ~A <sub>15</sub> ) or Input-only pin	8-bit I/o port
SC <sub>1</sub>	Address strobe (AS) output pin	/	I/O strobe (IOS) output pin	Address strobe (AS) output pin	Input strobe (IS3) output pin
SC <sub>2</sub>	Read/write signal (R/W) output pin	←	←	←	Output strobe (OS3) output pin

## 2. INTERNAL ARCHITECTURE

This section describes the HD63701V0 internal architecture.

### 2.1 Mode Selection

After the MCU is reset, a user must determine the operation mode of the HD63701V0 by strapping three pins and which are connected by hardware externally.

Individual signals on the above three pins are latched into the program control bits PC2, PC1 and PC0 of I/O port 2 Data register, when the  $\overline{\text{RES}}$  signal goes "High". The bit assignment of the Port 2 Data Register is shown below.

Port 2 Data Register

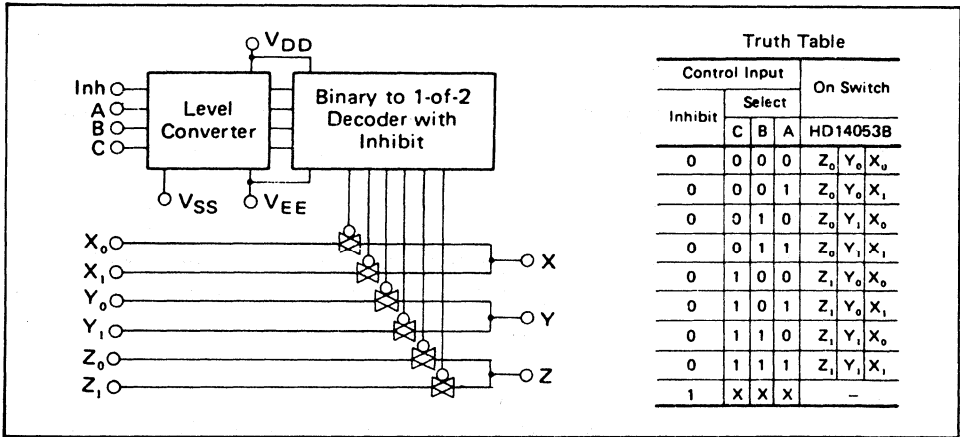
\$0003	7	6	5	4	3	2	1	0
	PC2	PC1	PC0	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0

An example of an external circuit for mode selection is shown in Fig. 2-1-1. The HD14053B may be used to separate the MCU from its peripheral devices during reset (Data confliction should be avoided between the peripheral devices and mode selection circuit). Because bits 5, 6 and 7 of port 2 are for read only, so the operation mode cannot be altered by software. The mode selection in the HD63701V0 is summarized in Table 2-1-1.

The HD63701V0 has three basic operation modes:

- 1) Single chip mode
- 2) Expanded multiplexed mode  
(Bus Compatible with HMCS6800 peripheral LSIs)
- 3) Expanded non-multiplexed mode  
(Bus Compatible with HMCS6800 peripheral LSIs)





HD14053B Multiplexers/De-Multiplexers

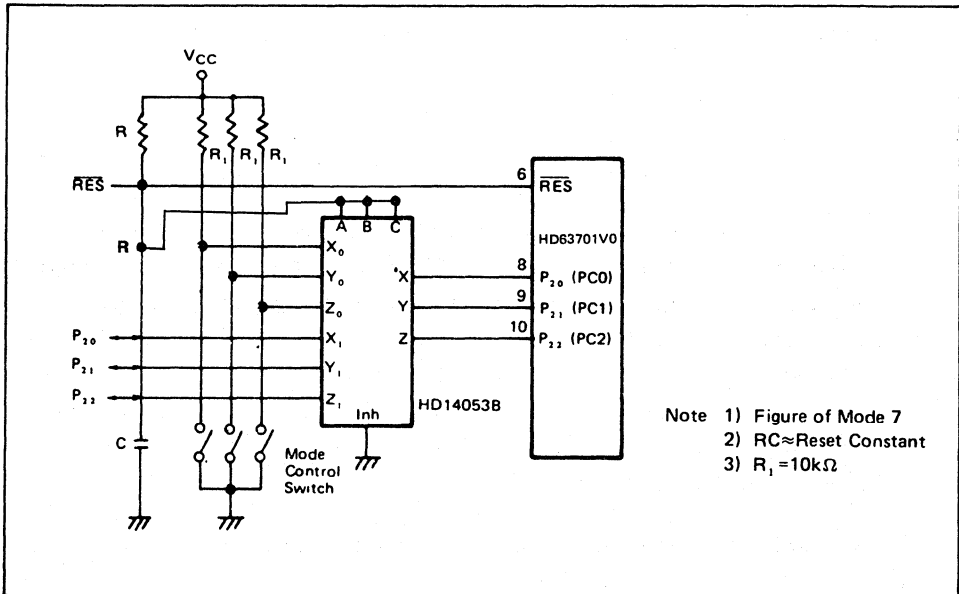


Fig. 2-1-1 Recommended Circuit for Mode Selection

Table 2-1-1 Mode Selection Summary

Mode	P <sub>22</sub> (PC2)	P <sub>21</sub> (PC1)	P <sub>20</sub> (PC0)	ROM	RAM	Interrupt Vectors	Bus Mode	Operating Mode
7	H	H	H	I	I	I	I	Single Chip
6	H	H	L	I	I	I	MUX <sup>3)</sup>	Multiplexed/Partial Decode
5	H	L	H	I	I	I	NMUX <sup>3)</sup>	Non-Multiplexed/Partial Decode
4	H	L	L	—	—	—	—	Not Used
3	L	H	H	—	—	—	—	Not Used
2	L	H	L	E <sup>1)</sup>	I	E	MUX	Multiplexed/RAM
1	L	L	H	E <sup>1)</sup>	I	E	NMUX	Non-Multiplexed
0	L	L	L	I	I	I <sup>2)</sup>	MUX	Multiplexed Test

LEGEND :

I — Internal  
 E — External  
 MUX — Multiplexed  
 NMUX — Non-Multiplexed  
 L — Logic "0"  
 H — Logic "1"

(NOTES)

1) Internal ROM is disabled.  
 2) Reset vector is external for 3 or 4 cycles after  $\overline{RES}$  goes "high".  
 3) Idle lines of Port 4 address outputs can be assigned to Input Port.

(1) Single Chip Mode

In the Single Chip Mode, all ports will function as I/O. This is shown in figure 2-1-2. In this mode, SC<sub>1</sub>, SC<sub>2</sub> pins are configured as Port 3 control lines and functions as input strobe ( $\overline{IS3}$ ) and output strobe ( $\overline{OS3}$ ) for handshaking data respectively.

(2) Expanded Multiplexed Mode

In this mode, Port 4 is configured as I/O (inputs only) Port or address lines. Port 3 functions as multiplexed lower address/data bus and Address Strobe (AS) selects the function of Port 3.

Port 2 is configured as a 5-bit parallel I/O port or Serial I/O, or Timer, or any combination thereof. Port 1 is configured as an 8-bit parallel I/O port. In this mode HD63701V0 is expandable to 65k words (See Fig. 2-1-3).

Since the data bus is multiplexed with the lower address bus in Port 3 in the expanded multiplexed mode, address bits must be latched outside. 74LS373 (Octal-D type transparent latches) is required for address latch.

Latch connection to the HD63701V0 is shown in Fig. 2-1-4.

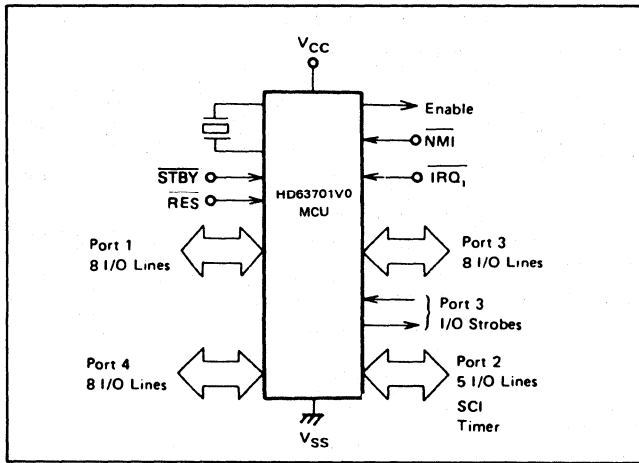


Fig. 2-1-2 HD63701V0 MCU Single-Chip Mode

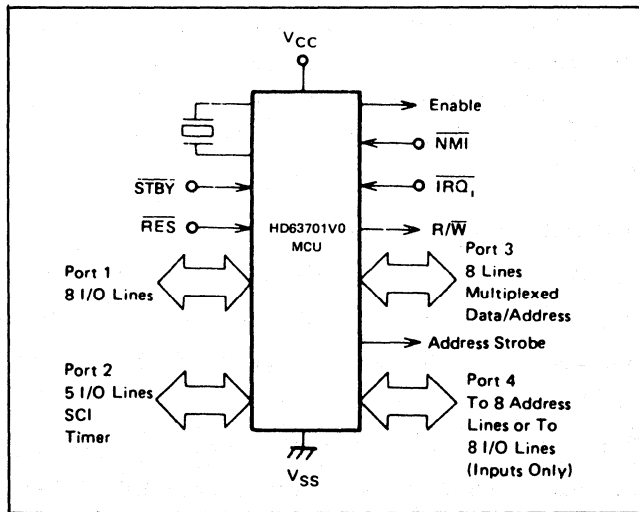


Fig. 2-1-3 HD63701V0 MCU Expanded Multiplexed Mode

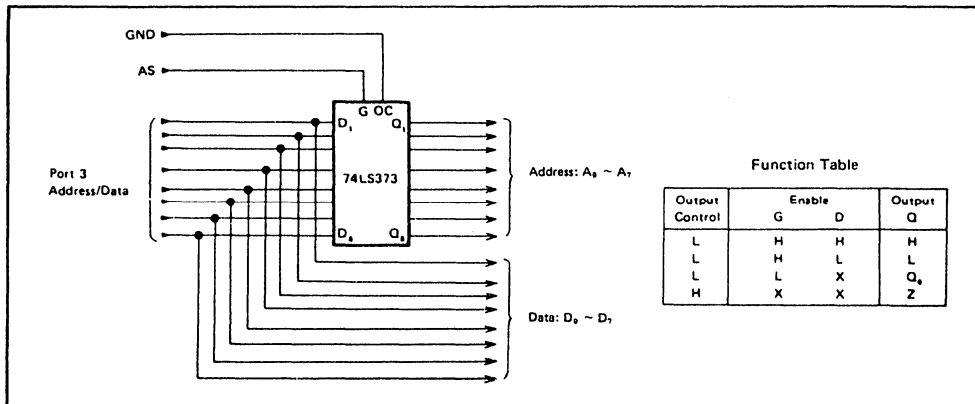


Fig. 2-1-4 Latch Connection

### (3) Expanded Non Multiplexed Mode

In this mode, the HD63701V0 can directly address HMCS6800 peripherals with no address latch. In mode 5, Port 3 functions as a data bus. Port 4 is configured as  $A_0$  to  $A_7$  address bus or partial address bus and I/O (inputs only) port. Port 2 is configured as a parallel I/O port, Serial I/O port, Timer or any combination. Port 1 is configured as a parallel I/O port only.

In this mode, the HD63701V0 can access up to 256 bytes of external address space. In the application system with fewer addresses, idle pins of Port 4 can be used as I/O lines (inputs only) (See Fig. 2-1-5).

In mode 1, Port 3 functions as a data bus, Port 1 functions as  $A_0$  to  $A_7$  address bus, and Port 4 is configured as  $A_8$  to  $A_{15}$  address bus. Port 2 is configured as a parallel I/O port, Serial I/O port, Timer or any combination. In this mode, the HD63701V0 is expandable up to 65k words with no address latch. (See Fig. 2-1-5).

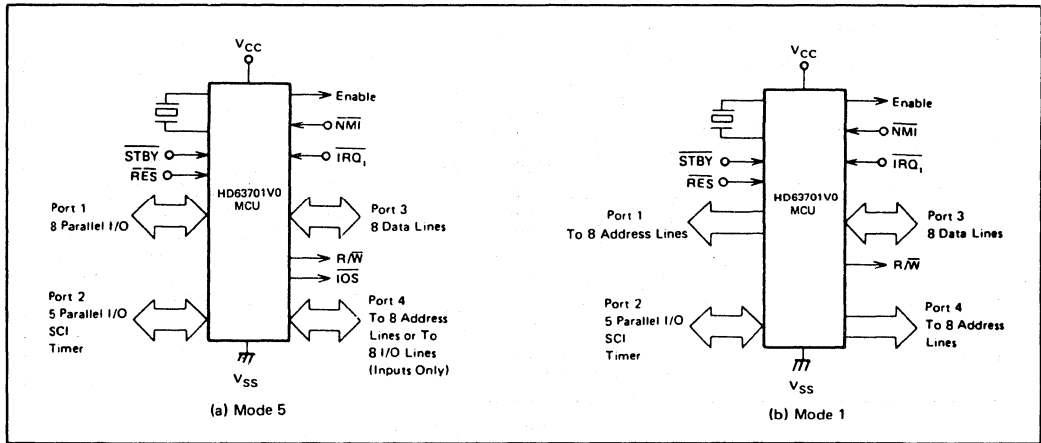


Fig. 2-1-5 HD63701V0 MCU Expanded Non Multiplexed Mode

(4) Mode and Port Summary MCU Signal Description

This section gives a description of the MCU signals for the various modes. SC<sub>1</sub> and SC<sub>2</sub> function depending on the operating mode.

Table 2-1-2 Feature of each mode and Lines

MODE		PORT 1 Eight Lines	PORT 2 Five Lines	PORT 3 Eight Lines	PORT 4 Eight Lines	SC <sub>1</sub>	SC <sub>2</sub>
SINGLE CHIP		I/O	I/O	I/O	I/O	$\overline{IS3}$ (I)	$\overline{OS3}$ (O)
EXPANDED MUX		I/O	I/O	ADDRESS BUS (A <sub>0</sub> - A <sub>7</sub> ) DATA BUS (D <sub>0</sub> - D <sub>7</sub> )	ADDRESS BUS* (A <sub>8</sub> - A <sub>15</sub> )	AS(O)	R/ $\overline{W}$ (O)
EXPANDED	Mode 5	I/O	I/O	DATA BUS (D <sub>0</sub> - D <sub>7</sub> )	ADDRESS BUS* (A <sub>0</sub> - A <sub>7</sub> )	$\overline{IOS}$ (O)	R/ $\overline{W}$ (O)
NON-MUX	Mode 1	ADDRESS BUS (A <sub>0</sub> - A <sub>7</sub> )	I/O	DATA BUS (D <sub>0</sub> - D <sub>7</sub> )	ADDRESS BUS (A <sub>8</sub> - A <sub>15</sub> )	Not Used	R/ $\overline{W}$ (O)

\*These lines can be substituted for I/O (Input Only) starting with the MSB (except Mode 0, 2, 4). When they are not used as address lines.

I = Input                     $\overline{IS3}$  = Input Strobe                    SC = Strobe Control  
 O = Output                 $\overline{OS3}$  = Output Strobe                    AS = Address Strobe  
 R/ $\overline{W}$  = Read/Write         $\overline{IOS}$  = I/O Select

2.2 Memory Map

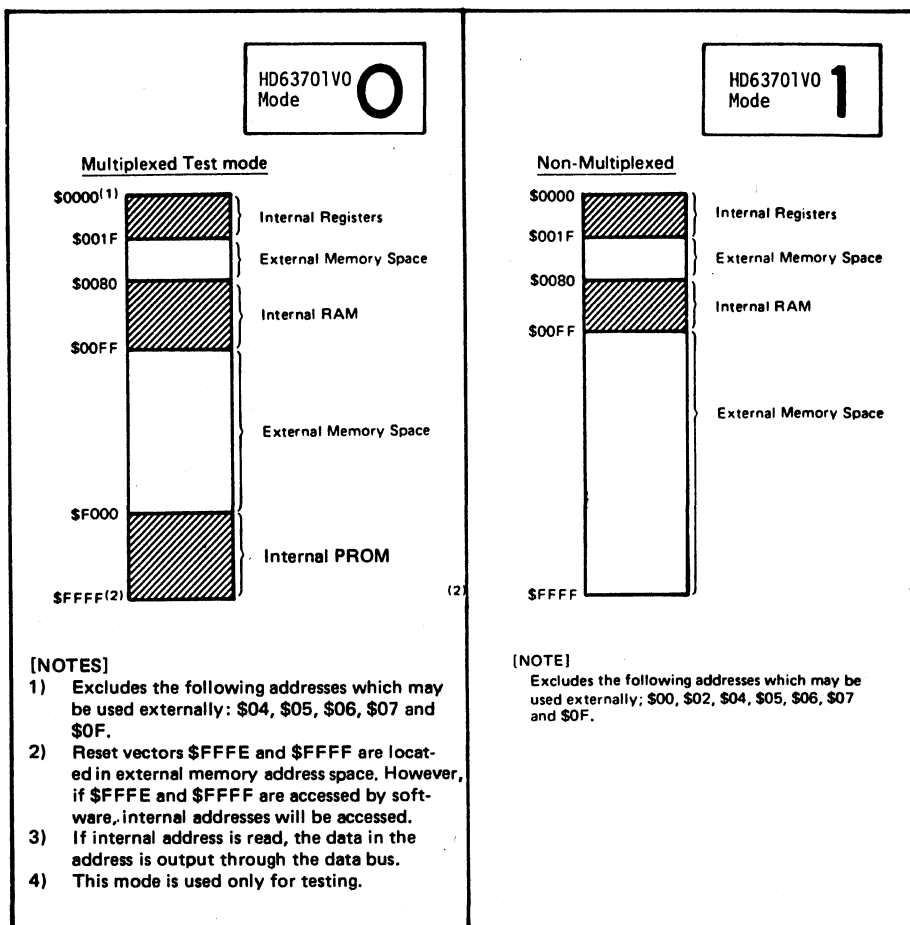
The MCU can address up to 65k bytes depending on the operating mode. Fig. 2-2-1 shows a memory map for each operating mode. The first 32 locations of each map are reserved for the MCU's internal register as shown in Table 2-2-1.

Table 2-2-1 Internal Register Area

Register	Address	R/W*4/Initialize at RESET							
		7	6	5	4	3	2	1	0
Port 1 Data Direction Register	\$00*1	W							
		\$00							
Port 2 Data Direction Register	\$01	W							
		\$00							
Port 1 Data Register	\$02*1	R/W *5							
		Undefined							
Port 2 Data Register/Mode Register	\$03	R	*6			R/W *5			
		P <sub>22</sub>	P <sub>21</sub>	P <sub>20</sub>	Undefined				
Port 3 Data Direction Register	\$04*2	W							
		\$00							
Port 4 Data Direction Register	\$05*3	W							
		\$00							
Port 3 Data Register	\$06*2	R/W *5							
		Undefined							
Port 4 Data Register	\$07*3	R/W *5							
		Undefined							
Timer Control and Status Register	\$08	R	R	R	R/W	R/W	R/W	R/W	R/W
		0	0	0	0	0	0	0	0
Counter (High Byte)	\$09	R/W							
		\$00							
Counter (Low Byte)	\$0A	R/W							
		\$00							
Output Compare Register (High Byte)	\$0B	R/W							
		\$FF							
Output Compare Register (Low Byte)	\$0C	R/W							
		\$FF							
Input Capture Register (High Byte)	\$0D	R							
		\$00							
Input Capture Register (Low Byte)	\$0E	R							
		\$00							
Port 3 Control and Status Register	\$0F*2	R	R/W	Un-used	R/W	R/W	Unused		
		0	0	1	0	0	1	1	1
Rate and Mode Control Register	\$10	Unused				W	W	W	W
		1	1	1	1	0	0	0	0
Transmit/Receive Control and Status Register	\$11	R	R	R	R/W	R/W	R/W	R/W	
		0	0	1	0	0	0	0	
Receive Data Register	\$12	R							
		\$00							
Transmit Data Register	\$13	W							
		\$00							
RAM Control Register	\$14	R/W	R/W	Unused					
		*7	1	1	1	1	1	1	1
Reserved	\$15~\$1F	/							

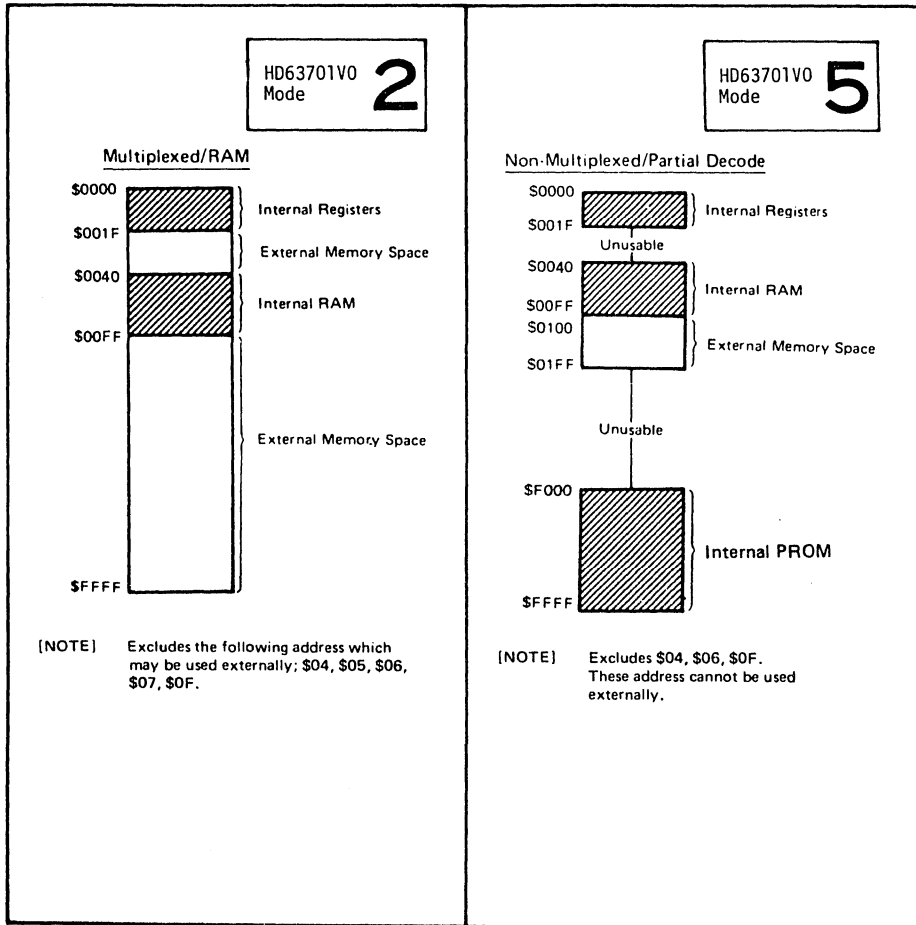
(\*1 through 8 are shown in the next page.)

- \*1 External address in mode 1.
- \*2 External address in modes 0, 1, 2, 5, 6; cannot be accessed in mode 5.
- \*3 External address in modes 0, 1, 2,
- \*4 R : Read-only register, W : Write-only register, R/W : Read/Write register.
- \*5 The pin state is read instead of the data of the register when reading Ports. (Refer to "2.4 I/O Ports" for I/O Port 3.)
- \*6 The values of program control bit (PC<sub>0</sub> ~ PC<sub>2</sub>) depend on P<sub>20</sub> ~ P<sub>21</sub> during reset.
- \*7 Refer to "2.12 Low Power Consumption Mode" for standby mode.



(to be continued)

Fig. 2-2-1 HD63701V0 Memory Maps



(to be continued)

Fig. 2-2-1 HD63701V0 Memory Maps



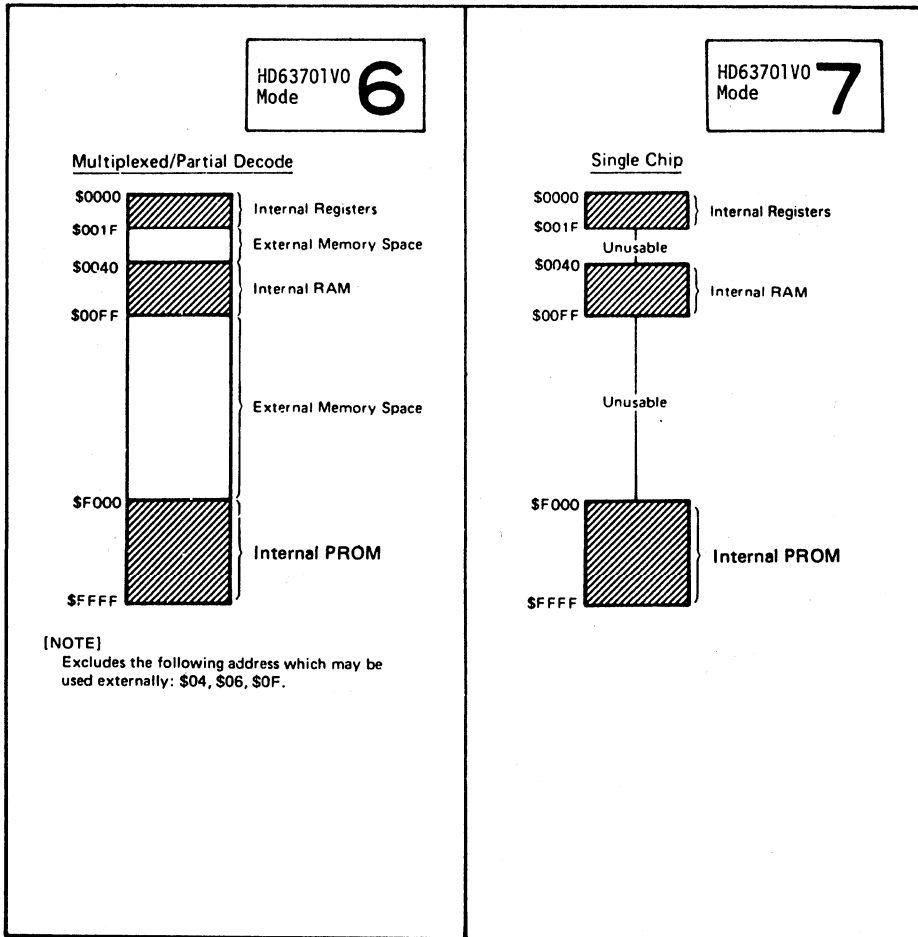


Fig. 2-2-1 HD63701V0 Memory Maps

## 2.3 CPU Registers

The followings describe the HD63701V0 internal architectures and operations.

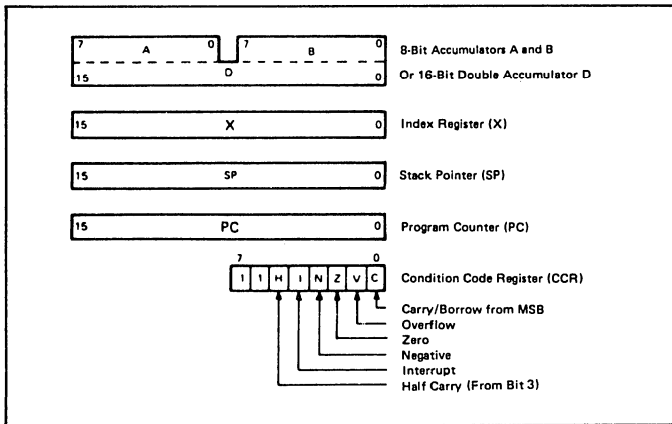


Fig. 2-3-1 HD63701V0 CPU Registers

### (1) Accumulators (A & B, or D)

Two 8-bit registers (ACCA and ACCB) store the result of arithmetic/logical operation and data. When combined, they make up a 16-bit register (ACCD) used for 16-bit operations. Note that the contents of ACCA and ACCB are modified after an ACCD-based operation.

### (2) Index Register (IX)

The 16-bit register IX stores a 16-bit data for use in indexed addressing mode or for general purpose.

### (3) Stack Pointer (SP)

The 16-bit register SP indicates the address of the next available location in the stack. This can also be used as a general purpose register.

### (4) Program Counter (PC)

The 16-bit register PC indicates the address of the instruction being currently executed. Note that the PC cannot be accessed by software.

## (5) Condition Code Register (CCR)

The CCR consists of the following bits: carry (C), overflow (V), zero (Z), negative (N), interrupt mask (I), and half-carry (H). After an instruction is executed, these bits reflect the result of operation. They can be tested by different conditional branch instructions. The upper 2 bits of this register cannot be used. Individual bits are detailed below. Refer to the following description of each instruction for more details.

### (a) Half-carry (H)

This bit is set to "1" if a carry occurs between bit 3 and bit 4 during execution of an ADD, ABA or ADC instruction; cleared otherwise.

### (b) Interrupt mask (I)

When set to "1", this bit disables any maskable interrupt ( $\overline{IRQ}_1$ ,  $\overline{IRQ}_2$ ).

### (c) Negative (N)

After an instruction is executed, this bit is set to "1" if the MSB of the result is "1"; cleared otherwise.

### (d) Zero (Z)

After an instruction is executed, this bit is set to "1" if the result is "0"; cleared otherwise.

### (e) Overflow (V)

After an instruction is executed, this bit is set if the result of operation shows a 2's complement overflow; cleared otherwise.

### (f) Carry (C)

After an instruction is executed, this bit is set to "1" if a carry or a borrow generates from MSB; it is cleared otherwise.

## 2.4 Ports

The HD63701V0 has four I/O ports (three 8-bit ports and one 5-bit port). 2 control pins are connected to one of the 8-bit port. Each port has an independent write-only Data Direction Register to program individual I/O pins for input or output.\*

When the bit of associated Data Direction Register is "1", I/O pin is programmed for output, if "0", then programmed for an input.

Addresses of each port and associated Data Direction Register are shown in Table 2-4-1.

\* Only one exception is bit 1 of Port 2 which becomes either a data input or a timer output. It cannot be used as an output.

Table 2-4-1 Port and Data Direction Register Addresses

Ports	Port Address	Data Direction Register Address
Port 1	\$0002	\$0000
Port 2	\$0003	\$0001
Port 3	\$0006	\$0004
Port 4	\$0007	\$0005

### (1) Port 1

Port 1 is an 8-bit I/O port, each bit being individually defined as inputs or outputs by the Port 1 Data Direction Register. The 8-bit output buffers have three-state capability, maintaining in high impedance state when they are used for input.

These are TTL compatible and can drive one TTL load and 90pF capacitance. After the MCU has been reset, all I/O pins are configured as inputs in all modes except mode 1. In all modes other than expanded non multiplexed mode 1, Port 1 is always parallel I/O. In mode 1, Port 1 is configured as output lines for lower order address lines (A<sub>0</sub> to A<sub>7</sub>).

## (2) Port 2

Port 2 has five lines, whose I/O direction depend on its Data Direction Register. The 5-bit output buffers have three-state capability, going high impedance state when used as inputs. After the MCU has been reset, Port 2 I/O pins are configured as inputs. P<sub>20</sub> - P<sub>22</sub> (pins 10 - 8) are used to program the operating mode during reset. The values of P<sub>20</sub> - P<sub>22</sub> during reset are latched into the upper 3 bits (bit 7, 6 and 5) of Port 2 Data Direction Register. Refer to "2.1 Mode Selection" for more details.

In all modes, Port 2 can be configured as I/O lines. This port can also function as I/O pins for the SCI and the Timer. However, note that bit 1 (P<sub>21</sub>) is the only pin restricted to data input or Timer output.

## (3) Port 3

Port 3 is an 8-bit port which can be configured as I/O Port, a data bus, or an address bus multiplexed with data bus. Its function depends on operation mode, determined by user using 3 bits of Port 2 during Reset. (Refer to 2.1 Mode Selection.) Port 3 as a data bus is bi-directional. This TTL compatible three-state buffer can drive one TTL load and 90pF capacitance. In the expanded Modes, Data Direction Register is inhibited after Reset and data flow is controlled by the state of the R/W signal. Function of Port 3 for each mode is explained below.

- (a) Single Chip Mode (Mode 7): Parallel I/O Port, whose I/O direction are programmed by the Port 3 Data Direction Register.

There are two control lines associated with this port in this mode, an input strobe ( $\overline{IS3}$ ) and an output strobe ( $\overline{OS3}$ ), both being used for handshaking. They are

controlled by I/O Port 3 Control/Status Register.  
 Additional 3 characteristics of Port 3 are summarized as follows:

- (1) Port 3 input data can be latched using  $\overline{IS3}$  ( $SC_1$ ) as a control signal.
- (2)  $\overline{OS3}$  ( $SC_2$ ) can be generated by MPU read or write to Port 3's data register.
- (3)  $\overline{IRQ_1}$  interrupt can be generated by an  $\overline{IS3}$  falling edge.

Port 3 strobe and latch timings are shown in Figs. 5-5 and 5-6, respectively.

I/O Port 3 Control/Status Register

	7	6	5	4	3	2	1	0
\$000F	$\overline{IS3}$ FLAG	$\overline{IS3}$ $\overline{IRQ_1}$ ENABLE	X	OSS	LATCH ENABLE	X	X	X

Bit 0 Not used.

Bit 1 Not used.

Bit 2 Not used.

Bit 3 LATCH ENABLE.

Controls the input latch of Port 3. If the bit is set, the input data on Port 3 is latched by the falling edge of  $\overline{IS3}$ . The latch is cleared by the MCU read to Port 3; it can be latched again. Bit 3 is cleared by a reset.

Bit 4 OSS (Output Strobe Select)

Determines the cause of output strobe generation: a write operation or read operation to I/O Port 3. When the bit is cleared, the strobe will be generated by a read operation to Port 3. When the bit is set, the strobe will be generated by a write operation. Bit 4 is cleared by reset.

Bit 5 Not used.

Bit 6  $\overline{IS3}$  ENABLE.

If the  $\overline{IS3}$  flag (bit 7) is set with bit 6 set, an interrupt is enabled. Clearing the flag causes the interrupt to be disabled. The bit is cleared by reset.

Bit 7  $\overline{IS3}$  FLAG.

Bit 7 is a read-only bit which is set by the falling edge of  $\overline{IS3}$  ( $SC_1$ ). It is cleared by a read of the Control/Status Register followed by a read/write of I/O Port 3. The bit is cleared by reset.

(b) Expanded Non Multiplexed Mode (mode 1, 5)

In this mode, Port 3 is configured as data bus. ( $D_0$  to  $D_7$ )

(c) Expanded Multiplexed Mode (mode 0, 2, 6)

Port 3 is configured as either data bus ( $D_0$  to  $D_7$ ) or lower 8 bits of the address bus ( $A_0$  to  $A_7$ ). An address strobe output is "High" when the address is on the port.

#### (4) Port 4

Port 4 is an 8-bit port that becomes either I/O Port or address bus depending on the operation mode selected. Each line is TTL compatible and can drive one TTL load and 90pF capacitance. After reset, this port becomes inputs. To use Port 4 as address bus, Port 4 pins should be programmed as outputs.

Function of Port 4 for each mode is explained below.

(a) Single Chip Mode (Mode 7): Parallel I/O Port, whose I/O direction is programmed by the Port 4 Data Direction Register.

(b) Expanded Non Multiplexed Mode (Mode 5): In this mode, Port 4 becomes the lower address lines ( $A_0$  to  $A_7$ ) by writing "1"s on the Data Direction Register.

When all of the eight bits are not required as addresses,

the remaining lines can be used as I/O lines (Inputs only).

(c) Expanded Non Multiplexed Mode (Mode 1): In this mode, Port 4 becomes output for upper address lines (A<sub>8</sub> to A<sub>15</sub>).

(d) Expanded Multiplexed Mode (Mode 0, 2): In this mode, Port 4 becomes output for upper address lines (A<sub>8</sub> to A<sub>15</sub>) regardless of the value of Data Direction Register.

The relation between each mode and ports 1 to 4 is summarized in Table 2-1-2.

## 2.5 Timer

The HD63701V0 provides 16-bit programmable timer which can measure input waveform and generate an output waveform. The pulse widths of both input/output can vary from microseconds to seconds. microseconds to many seconds.

The timer hardware consists of

- an 8-bit Control and Status Register
- a 16-bit Free Running Counter
- a 16-bit Output Compare Register, and
- a 16-bit Input Capture Register

A block diagram of the timer is shown in Figure 2-5-1.

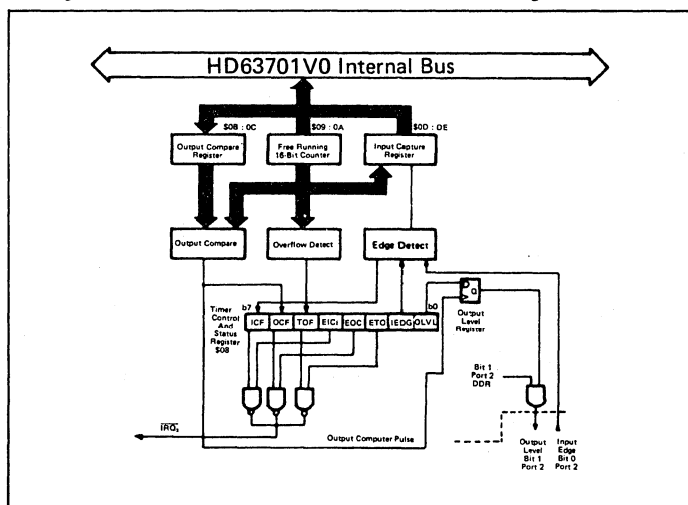


Fig. 2-5-1 Programmable Timer Block Diagram



(1) Free Running Counter (FRC) (\$0009:\$000A)

The key timer element is a 16-bit Free Running Counter (FRC), that is driven by E (Enable) clock to increment its values. The FRC value is readable by software at any time with no effects on the FRC. The FRC is cleared during reset.

When writing to the upper byte of the FRC (\$09), the CPU writes the preset value (\$FF8) into the FRC (address \$09, \$0A) regardless of the write data. But when writing to the lower byte (\$0A) after the upper byte writing, the CPU writes not only the lower byte data into lower byte of the FRC, but also the upper byte data into upper byte of the FRC.

The FRC value written by the double store instruction is shown in Figure 2-5-2.

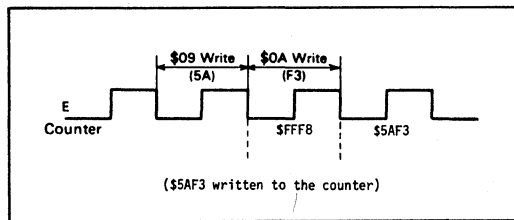


Fig. 2-5-2 Counter Write Timing

\* A write to the counter can disturb serial operations, so it should be inhibited during the SCI operation.

(2) Output Compare Register (OCR) (\$000B:\$000C)

The Output Compare Register (OCR) is a 16-bit read/write register which is used to control an output waveform. The data of the OCR is constantly compared with the FRC.

When the data matches, a flag (OCF) in the Timer Control/Status Register (TCSR) is set and the current value of an output level Bit (OLVL) in the TCSR is transferred to Port 2 bit 1. When bit 1 of the Port 2 Data Direction Register is "1" (output) the OLVL value will appear on the bit 1 of Port 2. Then, the value of the OCR and Output level bit should be changed to control an output level again on the next compare values.

The OCR is initialized to \$FFFF during reset. The compare function is inhibited at the cycle of writing to the upper bytes of the OCR and at the cycle just after that. It is also inhibited at the cycle of writing to the Free Running Counter.

\* For the data write to the OCR, 2-byte transfer instructions such as STD, STX should be used.

(3) Input Capture Register (\$000D:\$000E)

The Input Capture Register (ICR) is a 16-bit read-only register used to store the FRC's value when the proper transition of an external input signal occurs as defined by the input edge bit (IED1) in the TCSR.

The input transition change required to trigger the counter transfer is controlled by the input Edge bit (IEDG).

To allow the external input signal to gate in the edge detector, the bit of the Port 2 Data Direction Register corresponding to bit 0 of Port 2 must have been cleared (to zero).

To insure input capture in all cases, the width of an input pulse requires at least 2 E clock cycles.

(4) Timer Control/Status Register (TCSR) (\$0008)

This is an 8-bit register. All 8 bits are readable and the lower 5 bits can be written. The upper 3 bits are read-only, indicating the timer status information below.

- (a) Defined transition of the timer input signal causes the counter to transfer its data to the ICR.
- (b) A match has occurred between the value in the FRC and the OCF.
- (c) The counter value reached to \$0000 by counting-up (TOF).

Each of the upper three flags can generate an  $\overline{\text{IRQ}}_2$  interrupt and is controlled by an corresponding enable bit in the TCSR. If the I-bit in the CCR has been cleared, a prioritized vectored address occurs corresponding to each flag being set. Each bit is described as follows.

Timer Control/Status Register (TCSR)

7	6	5	4	3	2	1	0	
ICF	OCF	TOF	EICI	EOCI	ETOI	IEDG	OLVL	\$0008

- Bit 0 OLVL (Output Level): When a match occurs between the FRC and the OCR, this bit is transferred to the Port 2 bit 1. If the DDR corresponding to Port 2 bit 1 is set to "1", the value will appear on the output pin of Port 2 bit 1.
- Bit 1 IEDG (Input Edge): This bit control which transition of Port 2 bit 0 input ( $P_{20}$ ) will trigger the data transfer from the FRC to the ICR. The DDR corresponding to Port 2 bit 0 must be cleared in advance of using this function. When IEDG = 0, data transfer is triggered on a falling edge ("High"-to-"Low" transition) of  $P_{20}$ . When IEDG = 1, data transfer is triggered on a rising edge ("Low"-to-"High" transition) of  $P_{20}$ .
- Bit 2 ETOI (Enable Timer Overflow Interrupt): When set, this bit enables TOF interrupt to generate the interrupt request ( $\overline{\text{IRQ}}_2$ ); when cleared, the interrupt is inhibited.
- Bit 3 EOCI (Enable Output Compare Interrupt): When set, this bit enables OCF interrupt to generate the interrupt request ( $\overline{\text{IRQ}}_2$ ); when cleared, the interrupt is inhibited.

Bit 4 EICI (Enable Input Capture Interrupt): When set, this bit enables ICF interrupt to generate the interrupt request ( $\overline{IRQ}_2$ ); when cleared, the interrupt is inhibited.

Bit 5 TOF (Timer Overflow Flag): This read-only bit is set when the counter value is \$0000. It is cleared by CPU read of TCSR (with TOF set) followed by CPU read of the counter (\$0009).

Bit 6 OCF (Output Compare Flag): This read-only bit is set when a match occurs between the OCR and the FRC. It is cleared by read of TCSR (with OCF set) followed by CPU write to the OCR (\$000B or \$000C).

Bit 7 ICF (Input Capture Flag): The read-only bit is set when an input signal to edge detector makes a transition as defined by IEDG. It is cleared by read of TCSR (with ICF set) followed by CPU read of the ICR (\$000D).

Each bit of TCSR is cleared during reset.

## 2.6 Serial Communication Interface (SCI)

The HD63701V0 contains a full-duplex asynchronous Serial Communication Interface (SCI). The SCI can select the several kinds of the data transmit rate and comprises a transmitter and a receiver which operate independently on each other but at the same data transmit rate. Both of transmitter and receiver communicate with the CPU by the data bus, and with the outside through Port 2 bit 2, 3 and 4. Descriptions of hardware, software, and the SCI registers are as follows.

### (1) Wake-Up Function

In typical multiprocessor applications the software protocol has the destination address at the initial byte of the message. The purpose of Wake-Up function is to have the non-selected MCU ignore the remainder of the message. Thus the non-selected MCU can inhibit

the all further interrupt process until the next message begins.

Wake-Up function is triggered by a ten consecutive "1"s which indicates an idle transmit line. Therefore software protocol needs an idle period between the messages. With this hardware feature, the non-selected MPU is re-enabled (or "wakes-up") for the appearing next message.

## (2) Programmable Options

The HD63701V0 SCI has the following programmable features.

- . data format; standard mark/space (NRZ) start bit + 8 bit data + 1 stop bit
- . Clock source; external or internal
- . baud rate; one of 4 rates per given MCU E clock frequency or 1/8 of external clock
- . wake-up function; enabled or disabled
- Interrupt requests; enabled or masked individually for transmitter and receive data registers
- Clock Output; internal clock enabled or disabled to Port 2 bit 2
- Port 2 (bits 3,4); dedicated or not dedicated to serial I/O individually for receiver and transmitter

## (3) SCI Hardware

The SCI hardware is controlled by 4 registers as shown in Figure 2-6-1. The registers include:

- an 8-bit Transmit/Receive Control Status Register (TRCSR)
- a 4-bit write-only Transmit Rate/Mode Control Register (RMCR)
- an 8-bit read-only Receive Data Register (RDR)
- an 8-bit write-only Transmit Data Register (TDR)

Besides these 4 registers, the SCI utilizes Port 2 bit 3 (input) and bit 4 (output). Port 2 bit 2 can be used when an option is selected for the internal-clock-out or the external-clock-in.

## (4) Transmit/Receive Control Status Register (TRCSR)

TRCSR consists of 8 bits which all may be read

while only bits 0 to 4 may be written. The register is initialized to \$20 during  $\overline{RES}$ . The bits of the TRCSR are defined as follows.

Transmit/Receive Control Status Register (TRCSR)

7	6	5	4	3	2	1	0	ADDR: \$0011
RDRF	ORFE	TDRE	RIE	RE	TIE	TE	WU	

- Bit 0 WU (Wake Up): Set by software and cleared by hardware on receipt of ten consecutive "1"s. It should be noted that RE flag has already set in advance of WU flag's set.
- Bit 1 TE (Transmit Enable): Set to produce preamble of ten consecutive "1"s and to enable the data of transmitter to output subsequently to the Port 2 bit 4 independently of its corresponding DDR value. When cleared, the SCI affects nothing on Port 2 bit 4.
- Bit 2 TIE (Transmit Interrupt Enable): When set with TDRE (bit 5) set, an  $\overline{IRQ_2}$  interrupt is enabled. When cleared,  $\overline{IRQ_2}$  interrupt is masked.
- Bit 3 RE (Receive Enable); When set, gates Port 2 bit 3 to input of receiver regardless of DDR value for this bit. When cleared, the SCI affects nothing on Port 2 bit 3.
- Bit 4 RIE (Receive Interrupt Enable): When set with bit 7 (RDRF) or bit 6 (ORFE) set,  $\overline{IRQ_2}$  interrupt is enabled. When cleared,  $\overline{IRQ_2}$  interrupt is masked.
- Bit 5 TDRE (Transmit Data Register Empty): When the data is transmitted from the Transmit Data Register to Output Shift Register, it is set by hardware. The bit is cleared by reading the TRCSR (with TDRE set) and followed by writing the next new data into the Transmit Data Register. TDRE is initialized to 1 during  $\overline{RES}$ .
- Bit 6 ORFE (Over Run Framing Error): When overrun or framing error occurs (receive only), it is set by hardware. Over Run Error occurs if the attempt is made to transmit the new byte to the receive data

register with the RDRF still set. Framing Error occurs when the bit counters are not synchronized to the byte boundaries of the bit stream. The bit is cleared by reading the TRCSR (with ORFE set) followed by reading the RDR, or by  $\overline{\text{RES}}$ .

Bit 7 RDRF (Receive Data Register Full): It is set by hardware when the data is transmitted from the RSR to the RDR. It is cleared by reading the TRCSR (with RDRF set) and followed by reading the RDR, or by  $\overline{\text{RES}}$ .

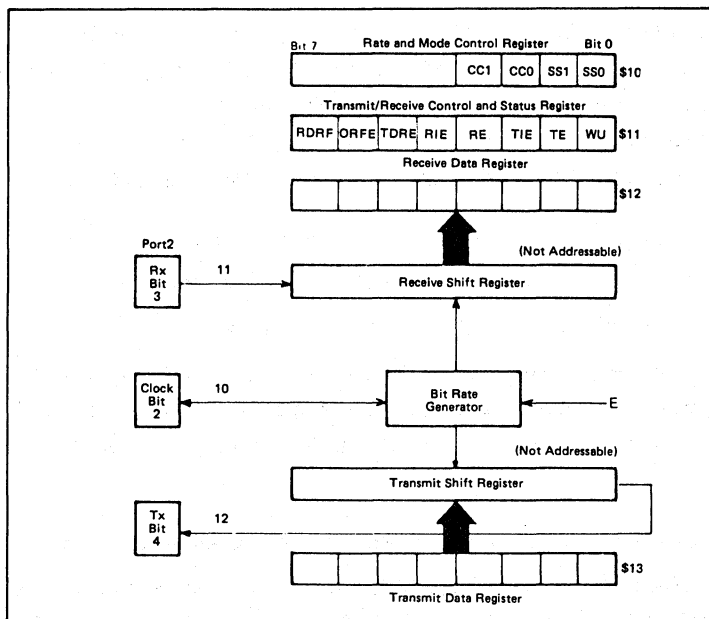


Fig. 2-6-1 SCI Register

7	6	5	4	3	2	1	0	
X	X	X	X	CC1	CC0	SS1	SS0	ADDR: \$0010
Rate / Mode Control Register								

Table 2-6-1 SCI Bit Times and Transfer Rates

SS1 : SS0	XTAL	2.4576 MHz	4.0 MHz	4.9152MHz
	E	614.4 kHz	1.0 MHz	1.2288MHz
0 0	E ÷ 16	26 $\mu$ s/38,400 Baud	16 $\mu$ s/62,500 Baud	13 $\mu$ s / 76,800Baud
0 1	E ÷ 128	208 $\mu$ s/4,800 Baud	128 $\mu$ s/7812.5 Baud	104.2 $\mu$ s / 9,600Baud
1 0	E ÷ 1024	1.67ms/600 Baud	1.024ms/976.6 Baud	833.3 $\mu$ s / 1,200Baud
1 1	E ÷ 4096	6.67ms/150 Baud	4.096ms/244.1 Baud	3.333ms / 300Baud

Table 2-6-2 SCI Format and Clock Source Control

CC1, CC0	Format	Clock Source	Port 2 Bit 2	Port 2 Bit 3	Port 2 Bit 4
00	-	-	-	-	-
01	NRZ	Internal	Not Used***	..	..
10	NRZ	Internal	Output*	..	..
11	NRZ	External	Input	..	..

\* Clock output is available regardless of values for bits RE and TE.

\*\* Bit 3 is used for serial input if RE = "1" in TRCS.

Bit 4 is used for serial output if TE = "1" in TRCS.

\*\*\*It can be used as I/O port.

(5) Transmit Rate/Mode Control Register (RMCR)

Transmit Rate/Mode Control Register (RMCR) controls the following SCI variables:

- Baud rate
- clock source
- Port 2 bit 2 function

The RMCR is a 4-bit write-only register, cleared by  $\overline{RES}$ . The 4 bits are considered as a pair of 2-bit fields. The lower 2 bits control the bit rate of internal clock while the upper 2 bits control the clock select logic.

Bit 0 SS0 }  
Bit 1 SS1 } Clock Speed Select

These bits select the Baud rate for the internal clock. The selectable 4 rates are function of E clock frequency within the MCU. Table 2-6-1 lists the available Baud Rates.

Bit 2 CC0 }  
Bit 3 CC1 } Clock Control/Format Select

These bits control the clock select logic. Table 2-6-2 defines the bit field.

(6) Internally Generated Clock

When using the SCI internal clock for external devices, the followings should be noted.

- The values of RE and TE have no effect.
- CC1, CC0 must be set to "10".
- The maximum clock rate is E/16.
- The clock is equal to the bit rate.

(7) Externally Generated Clock

When supplying an external clock for the SCI, the followings should be noted.



- The CCl, and CC0 in the RMCR must be set to "11"  
(See Table 2-6-2).
- The external clock frequency must be set to 8 times the desired baud rate.
- The maximum external clock frequency is half of E clock.

#### (8) Serial Operations

The SCI hardware must be initialized by the HD63701V0 software prior to operation. The sequence is normally as follows.

- Writing the desired operation control bits to the RMCR.
- Writing the desired operation control bits to the TRCSR.

If using Port 2 bits 3 and 4 for the SCI exclusively, TE and RE bits may be preserved set. When TE and RE bits are cleared during the SCI operation, and subsequently set again, it should be noted that the setting of TE, RE must refrain for at least one bit time of the current baud rate. If set within one bit time, internal function for transmit and receive may not occur.

#### (9) Transmit Operation

Data transmission is enabled by the TE bit in the TRCSR. When TE is set, outputs the data of the SCI Transmit Shift Register to Port 2 bit 4 which is unconditionally configured as an output irrespectively of the corresponding DDR value.

Following  $\overline{\text{RES}}$ , the user should program both the RMCR and the TRCSR for desired operation. Setting the TE bit during this procedure causes a transmission of ten-bit preamble of "1"s. Following the preamble, internal synchronization is established and the transmitter section is ready to operate. Then either of the followings operates.

(a) If the TDR is empty (TDRE = 1), the consecutive "1"s are transmitted indicating an idle lines.

(b) If the data has been loaded into the TDR (TDRE = 0), it is transferred to the Transmit Shift Register and data transmission begins.

During the data transfer, the "0" start bit is first transferred. Next the 8-bit data (beginning at bit 0) and the "1" stop bit. When the TDR has been empty, the TDRE flag bit is set.

If the CPU fails to respond to the flag within the proper time, TDRE is preserved set and then a "1" will be sent (instead of a "0" at start bit time) consecutively until the data is supplied to the Transmit Data Register. While the TDRE remains a "1", no "0" will be sent.

#### (10) Receive Operation

The receive operation is enabled by the RE bit. The serial input is connected with Port 2 bit 3. The receive operation is determined by the contents of the TRCSR RMCR. The received bit stream is synchronized by the first "0" (space). During 10-bit time, the data is strobed approximately at the center of each bit.

If the tenth bit is not "1" (stop bit), the system assumes a framing error and the ORFE is set. (RDRF is not set.)

If the tenth bit is "1", the data is transferred to the RDR, with the interrupt flag set (RDRF). If the tenth bit of the next data is received, however, still RDRF is preserved set, then ORFE is set indicating that an overrun error has occurred.

After the CPU read of the TRCSR as a response to RDRF flag or ORFE flag followed by the CPU read of the RDR, RDRF or ORFE will be cleared.

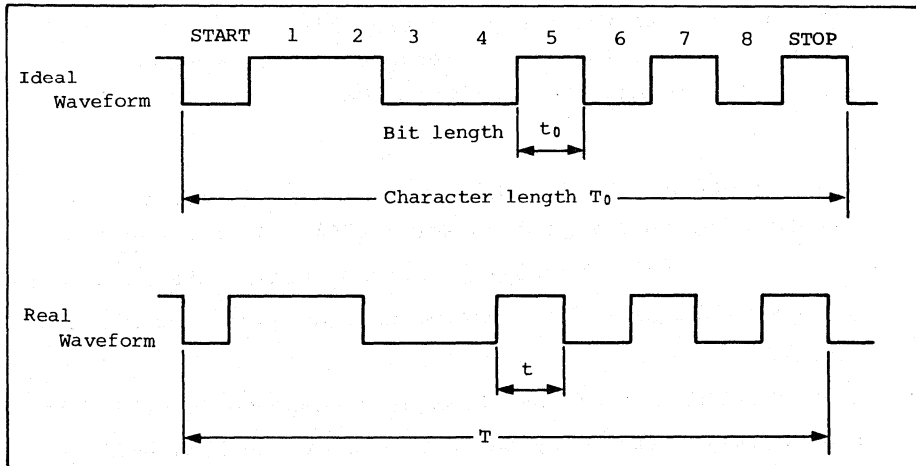
#### (11) Timer, SCI Status Flag

The set and reset condition of each status flag of timer and SCI is shown in Table 2-6-3.

Table 2-6-3 Timer and SCI Status Flag

Status Flag	Set Condition	Reset Condition	
Timer	ICF	FRC → ICR by edge input to P20.	<ol style="list-style-type: none"> <li>1. Read the TCSR then ICRH, when ICF=1</li> <li>2. <math>\overline{\text{RES}}=0</math></li> </ol>
	OCF	OCR=FRC	<ol style="list-style-type: none"> <li>1. Read the TCSR then write to the OCRH or OCRL, when OCF=1</li> <li>2. <math>\overline{\text{RES}}=0</math></li> </ol>
	TOF	FRC=\$FFFF+1 cycle	<ol style="list-style-type: none"> <li>1. Read the TCSR then FRCH, when TOF=1</li> <li>2. <math>\overline{\text{RES}}=0</math></li> </ol>
SCI	RDRF	Receive Shift Register → RDR	<ol style="list-style-type: none"> <li>1. Read the TRCSR then RDR, when RDRF=1</li> <li>2. <math>\overline{\text{RES}}=0</math></li> </ol>
	ORFE	<ol style="list-style-type: none"> <li>1. Framing Error (Asynchronous Mode) Stop Bit = 0</li> <li>2. Overrun Error (Asynchronous Mode) Receive Shift Register → RDR when RDRF=1</li> </ol>	<ol style="list-style-type: none"> <li>1. Read the TRCSR then RDR, when ORFE=1</li> <li>2. <math>\overline{\text{RES}}=0</math></li> </ol>
	TDRE	<ol style="list-style-type: none"> <li>1. TDR → Transmit Shift Register</li> <li>2. <math>\overline{\text{RES}}=0</math></li> </ol>	Read the TRCSR then write to the TDR, when TDRE=1

The receive margin of the HD63701V0 SCI is shown below.



Bit distortion tolerance $(t-t_0)/t_0$	Character distortion tolerance $(T-T_0)/T_0$
±37.5%	+3.75% -2.5%

## 2.7 Interrupts

The HD63701V0 has two external interrupt terminals ( $\overline{\text{NMI}}$ ,  $\overline{\text{IRQ}}_1$ ) and 8 internal interrupt sources (Soft-TRAP, SWI, Timer-ICF, OCF, TOF, SCI-RDRF, ORFE, TDRE). The features of these interrupt are detailed in the following paragraphs.

### (1) Non maskable Interrupt ( $\overline{\text{NMI}}$ )

When the high-to low transition of  $\overline{\text{NMI}}$  is detected,  $\overline{\text{NMI}}$  acknowledge sequence starts. The current instruction is completed if  $\overline{\text{NMI}}$  signal is detected as well as the following  $\overline{\text{IRQ}}_1$  interrupt. Interrupt mask bit in the Condition Code Register has no effect on  $\overline{\text{NMI}}$  interrupt. In response to  $\overline{\text{NMI}}$  interrupt, the contents of Program Counter, Index Register Accumulators, and Condition Code Register are pushed onto the stack. On completion of this sequence, the CPU generates vector addresses \$FFFC and \$FFFD, and loads the contents into the Program Counter and branch to a non maskable interrupt service routine.

### (2) Interrupt Request ( $\overline{\text{IRQ}}_1$ )

$\overline{\text{IRQ}}_1$  is the level-sensitive pin which requests an  $\overline{\text{IRO}}_1$  interrupt to the CPU. When  $\overline{\text{IRQ}}_1$  interrupt request occurs, the CPU will complete the current instruction before the acceptance of the request. If the I bit of the Condition Code Register is cleared, the CPU starts an interrupt acknowledge sequence; if set, the interrupt request will be ignored.

When the  $\overline{\text{IRQ}}_1$  acknowledge sequence starts, the contents of Program Counter, Index Register, Accumulator, and Condition Code Register are pushed onto the stack. Then the CPU sets the I bit to ignore another  $\overline{\text{IRQ}}_1$  or  $\overline{\text{IRQ}}_2$  interrupt request.

At the end of the cycle, the CPU generates 16-bit vector addresses \$FFF8 and \$FFF9, and loads the contents into the Program Counter to branch to an interrupt service routine.

(3) Internal interrupts ( $\overline{\text{IRQ}}_2$ )

When an internal interrupt is requested from the timer or SCI, an internal interrupt signal  $\overline{\text{IRQ}}_2$  is activated. This interrupt is identical to  $\overline{\text{IRQ}}_1$  except that it uses vector addresses \$FFF0 through \$FFF7. The  $\overline{\text{IRQ}}_1$  has the priority to the  $\overline{\text{IRQ}}_2$  when interrupt requests occurs at the same time. When the interrupt mask bit in the condition code register is set, both interrupts are inhibited.

The SWI is an instruction which requests an interrupt by software. The state of CCR mask bit doesn't influence the SWI. If an address error or op-code error (see "2.13 Error Processing") occurs, TRAP occurs whose priority is next to the reset. In the case of TRAP, CPU starts the interrupt sequence regardless of the state of the mask bit. The vector for the TRAP interrupt are \$FFFE and \$FFEF. The interrupt sources and their corresponding vector are listed in Table 2-7-1 and the interrupt sequence are shown in Fig. 2-7-1. Fig. 2-7-2 shows the interrupt generation circuit.

Table 2-7-1 Interrupt Vector

Vector		Interrupt Source
Upper Byte	Lower Byte	
FFFE	FFFF	$\overline{\text{RES}}$
FFEE	FFEF	TRAP
FFFC	FFFD	$\overline{\text{NMI}}$
FFFA	FFFB	Software Interrupt (SWI)
FFF8	FFF9	$\overline{\text{IRQ}}_1$ (or $\overline{\text{IS}}_3$ )
FFF6	FFF7	ICF (Timer Input Capture)
FFF4	FFF5	OCF (Timer Output Compare)
FFF2	FFF3	TOF (Timer Overflow)
FFF0	FFF1	SCI (RDRF+ORFE+TDRE)

Highest Priority  
 ↑  
 ↓  
 Lowest Priority

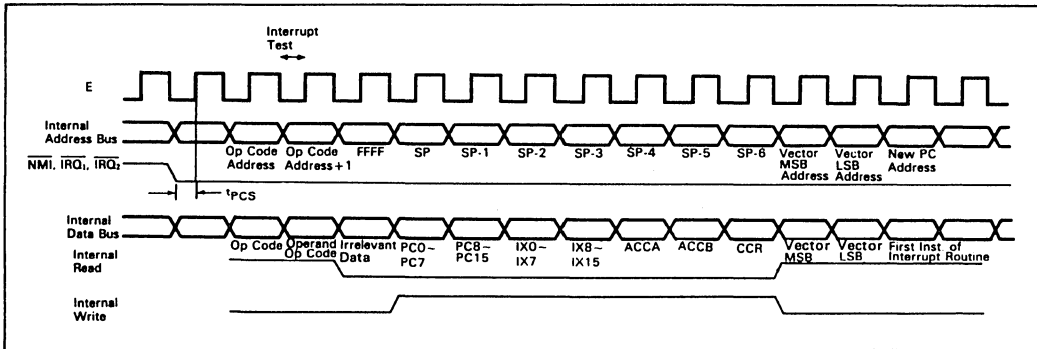


Fig. 2-7-1 Interrupt Sequence

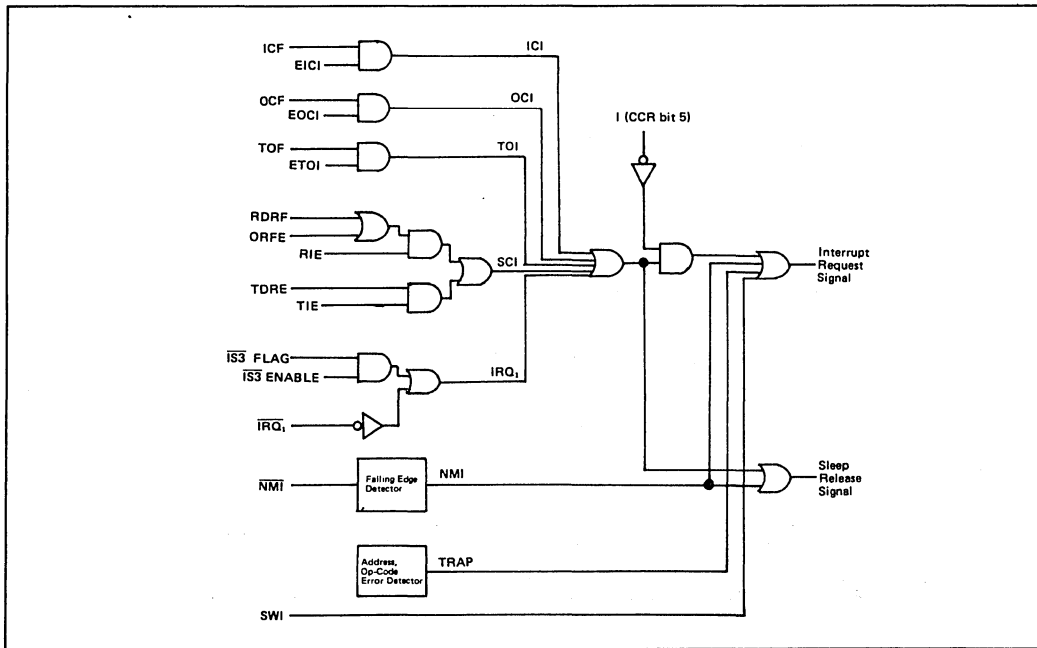


Fig. 2-7-2 Interrupt Generation Circuit

## 2.8 Reset ( $\overline{RES}$ )

This input is used to reset the MCU and start it from a power-off condition. During Power-on,  $\overline{RES}$  pin must be held "Low" for at least 20ms. To reset the MCU during system operation,  $\overline{RES}$  must be held "Low" for at least 3 system clock cycles. All address buses become "High impedance" state while  $\overline{RES}$  is "Low". When  $\overline{RES}$  is brought "high" again, the MCU operates as followings.

- (1) Latch I/O Port 2 bits 2, 1, and 0 into bits PC2, PC1, and PC0 of the Port 2 Data Register.
- (2) Load the contents (=start address) of the last two addresses (\$FFFE, \$FFFF) into the program counter and start the program from this address. (Refer to Table 2-7-1)
- (3) Set the interrupt mask bit. For the CPU to recognize the maskable interrupts  $\overline{IRQ}_1$  and  $\overline{IRQ}_2$ , this bit should be cleared in advance. Fig. 2-8-1 shows the reset timing, and Table 2-8-1 shows the pin condition during reset.

When  $\overline{RES}$  is asserted "Low", all I/O pins enters into reset mode (high impedance state) asynchronously to the E clock while the MCU enters into the reset mode synchronously to the E clock.

Both the MCU and I/O pins recover from reset mode synchronously to the E clock.

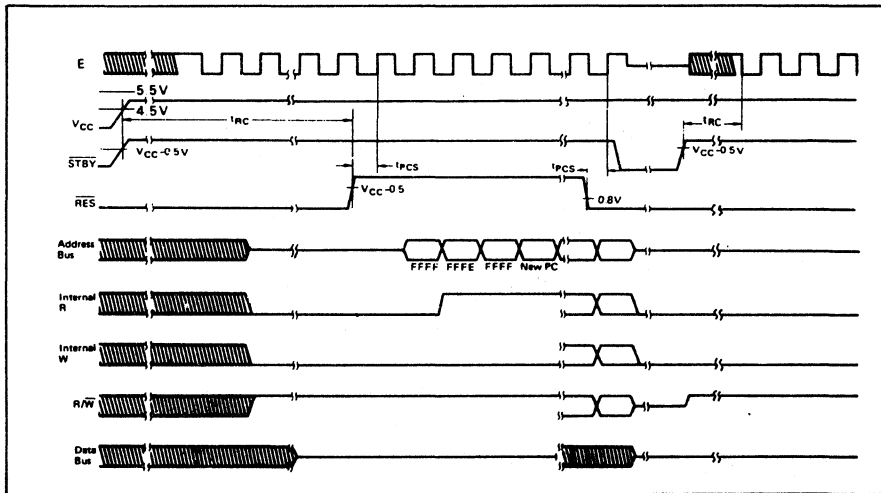


Fig. 2-8-1 Reset Timing

Table 2-8-1 Pin Condition during RESET

Mode Pin	0	1	2	5	6	7
Port 1 P <sub>10</sub> ~P <sub>17</sub>	High impedance (input)	←	←	←	←	←
Port 2 P <sub>20</sub> ~P <sub>24</sub>	High impedance (input)	←	←	←	←	←
Port 3 P <sub>30</sub> ~P <sub>37</sub>	$\bar{E}$ : "1" output E: "1" output (High impedance)	High impedance	$\bar{E}$ : "1" output E: "1" output (High impedance)	High impedance	$\bar{E}$ : "1" output E: "1" output (High impedance)	High impedance (input)
Port 4 P <sub>40</sub> ~P <sub>47</sub>	High impedance (input)	←	←	←	←	←
SC <sub>2</sub>	"1" output (READ)	←	←	←	←	"1" output
SC <sub>1</sub>	$\bar{E}$ : "1" output / E: High impedance	←	←	"1" output	$\bar{E}$ : "1" output E: High impedance	High impedance (input)

## 2.9 Oscillator

XTAL and EXTAL pins interface with either an AT-cut parallel resonant crystal or external clock. The on-chip divide-by-four circuit internally divides the input frequency by four to produce the system clock. For example, 4MHz of a crystal or external clock input corresponds to 1MHz of system clock.

When using a crystal, a 10-22 pF ± 20% capacitor should be tied from each crystal pin to ground. Alternately, EXTAL can be driven by external clock with 45% to 55% duty cycle. When external clock is input to EXTAL, XTAL should be left open.

Fig. 2-9-1 shows an example of crystal interface and crystal specification.



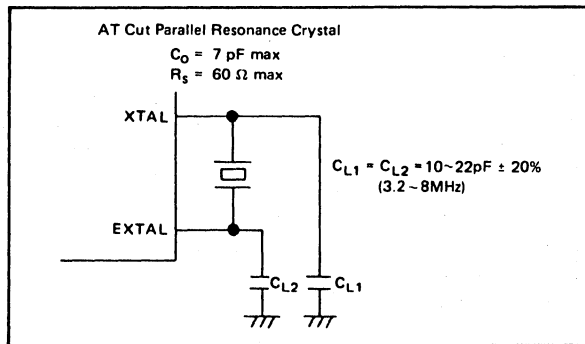


Fig. 2-9-1 Crystal Interface

## 2.10 Strobe Signals

Two pins,  $SC_1$  (39 pin) and  $SC_2$  (38 pin) are used as the strobe signals in each mode. Followings are applied only to Port 3 in single chip mode.

### (1) Input Strobe ( $\overline{IS3}$ ) ( $SC_1$ )

This signal controls  $\overline{IS3}$  interrupt and the latch of Port 3. When the falling edge of this signal is detected, the  $\overline{IS3}$  flag of Port 3 Control Status Register is set.

For respective bits of Port 3 Control Status Register, see the "2.4 Ports" section.

### (2) Output Strobe ( $\overline{OS3}$ ) ( $SC_2$ )

This signal is used by the processor to strobe an external device, indicating effective data is on the I/O pins. The timing chart for Output Strobe are shown in figure 6-5.

The following pins are available for Expanded Modes.

### (3) Read/Write ( $R/\overline{W}$ ) ( $SC_2$ )

This TTL compatible output signal indicates peripheral and memory devices whether the CPU is in Read ("High"), or in Write ("Low"). The normal stand-by state of this signal is Read ("High"). This output can drive one TTL load and 90pF capacitance.

### (4) I/O Strobe ( $\overline{IOS}$ ) ( $SC_1$ )

In expanded non multiplexed mode 5  $\overline{IOS}$  becomes "Low" when  $A_9$  through  $A_{15}$  are "0"s and  $A_8$  is "1". This allows external device access located in \$0100 through \$01FF in memory. The timing chart is shown in Figure 6-2.

(5) Address Strobe (AS) (SC<sub>1</sub>)

In the expanded multiplexed mode, address strobe is output to this pin. This signal is used to latch the lower 8 bits addresses multiplexed with data at Port 3. The 8-bit latch is controlled by address strobe as shown in Figure 2-1-4. Thereby, I/O Port 3 can functions as data bus while E is "high". The timing chart of this signal is shown in Figure 6-1.

2.11 RAM Control Register

The register located in the memory address \$0014 gives a status information about standby RAM.

RAM Control Register

	7	6	5	4	3	2	1	0
\$0014	STBY PWR	RAME	×	×	×	×	×	×

- Bit 0 Not used.
- Bit 1 Not used.
- Bit 2 Not used.
- Bit 3 Not used.
- Bit 4 Not used.
- Bit 5 Not used.

Bit 6 RAM Enable.

Using this control bit, the user can disable the RAM. RAME bit is set on the positive edge of  $\overline{\text{RES}}$  and RAM is enabled. RAME can be program to "1" or "0". If RAME is cleared (logic "0"), any access to a RAM address is external.

Bit 7 Standby Bit

This bit is cleared when VCC is not provided in standby mode. This bit is a read/write status flag that user can read. If this bit is preserved set, indicating that VCC voltage is applied and the data in the standby RAM is valid.

2.12 Low Power Consumption Mode

The HD63701V0 has two low power consumption modes; sleep and standby.

## (1) Sleep Mode

On execution of SLP instruction, the CPU is brought to the sleep mode. In the sleep mode, the CPU stops its operation, but the contents of the register in the CPU are retained. In this mode, on-chip peripherals such as the SCI and Timer continue their operations. In this mode, the power consumption is one-sixth that of normal operation mode.

The MCU returns from this mode by interrupt,  $\overline{\text{RES}}$ , or  $\overline{\text{STBY}}$ . The  $\overline{\text{RES}}$  resets the MCU and the  $\overline{\text{STBY}}$  brings it into the standby mode (This will be mentioned later). When the CPU acknowledges an interrupt request, it cancels the sleep mode, returns to the operation mode and branches to the interrupt routine. When the CPU masks the interrupt, it cancels the sleep mode and executes the next instruction. However, for example, if the timer 1 or 2 prohibits a timer interrupt, the CPU doesn't cancel the sleep mode because of no interrupt request.

The sleep mode is available to reduce the power consumption for a system with no need of the HD63701V0's consecutive operation.

Please refer to Table 2-12-1 for other pins except  $V_{CC}$ , clock pin, input-only pin, E clock pin (their function are the same as normal operating condition).

## (2) Standby Mode

The HD63701V0 stops all the clocks and goes into the reset state with  $\overline{\text{STBY}}$  "Low". In this mode, the power consumption is reduced conspicuously.

In the standby mode, the power is supplied to the HD63701V0, so the contents of RAM are retained. The MCU returns from the standby mode by bringing  $\overline{\text{STBY}}$  "High" and restarts execution at the same restart address as reset.

If external clock is used during standby mode, EXTAL must be brought "High" not increase standby current by 5-10 $\mu$ A. If the increase of standby current does not

affect the MCU, EXTAL can be held either "Low" or "High".  
 Transitions among the active mode, sleep mode, standby mode  
 and reset are shown in Fig. 2-12-2.

Table 2-12-1 Pin Condition in Sleep Mode

Pin \ Mode		Mode					
		0	1	2	5	6	7
Port 1 P <sub>10</sub> ~P <sub>17</sub>	Function	I/O Port	Lower Address Bus	I/O Port	←	←	←
	Condition	Keep the condition just before sleep	Output "1"	Keep the condition just before sleep	←	←	←
Port 2 P <sub>20</sub> ~P <sub>24</sub>	Function	I/O Port	←	←	←	←	←
	Condition	Keep the condition just before sleep	←	←	←	←	←
Port 3 P <sub>30</sub> ~P <sub>37</sub>	Function	$\bar{E}$ :Lower Address Bus E:Data Bus	Data Bus	$\bar{E}$ :Lower Address Bus E:Data Bus	Data Bus	$\bar{E}$ :Lower Address Bus E:Data Bus	I/O Port
	Condition	$\bar{E}$ :Output "1" E:High impedance	High impedance	$\bar{E}$ :Output "1" E:High impedance	High impedance	$\bar{E}$ :Output "1" E:High impedance	Keep the condition just before sleep
Port 4 P <sub>40</sub> ~P <sub>47</sub>	Function	Upper Address	←	←	Lower Address Bus or Input Port	Upper Address Bus or Input Port	I/O Port
	Condition	Output "1"	←	←	Address Bus: Output "1" Port:Keep the condition just before sleep	←	Keep the condition just before sleep
SC <sub>2</sub>		Output "1" (Read Condition)	←	←	←	←	Output "1"
SC <sub>1</sub>		Output Address Strobe	←	←	Output "1"	Output Address Strobe	Input Pin

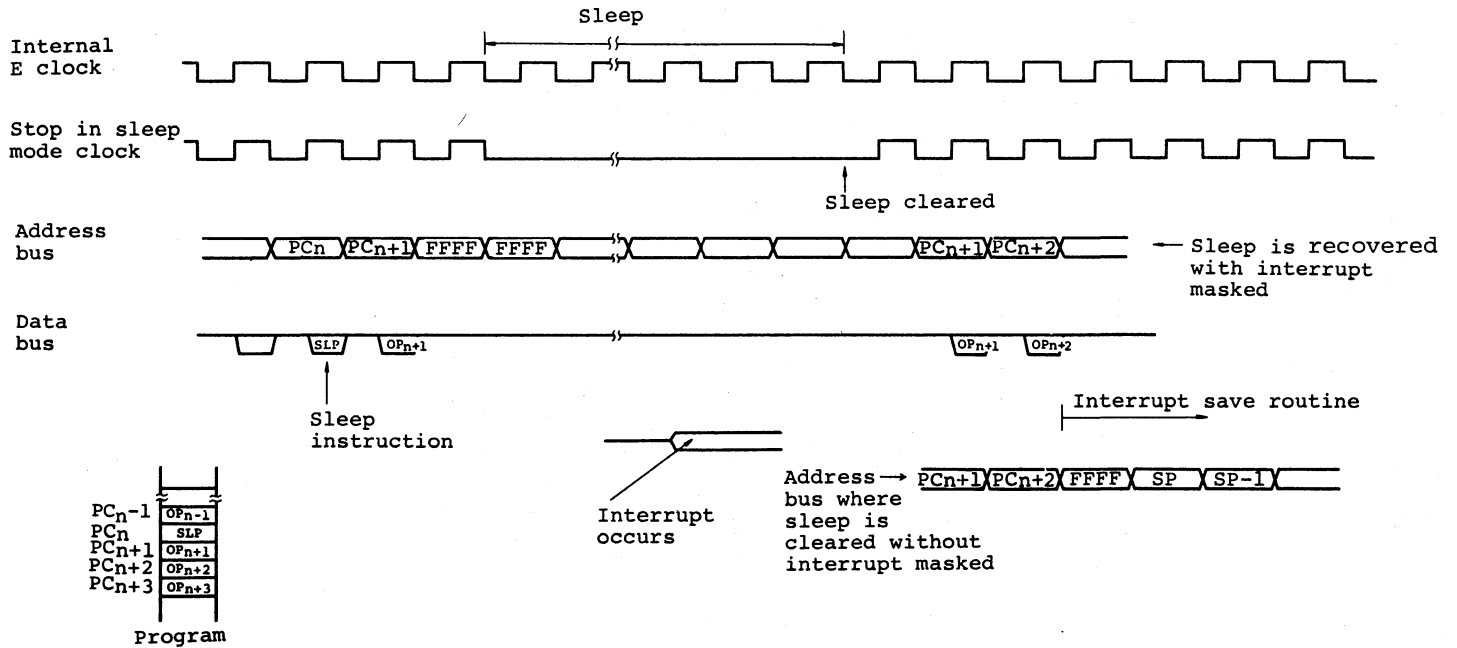


Fig. 2-12-1 Sleep Instruction Timing Chart

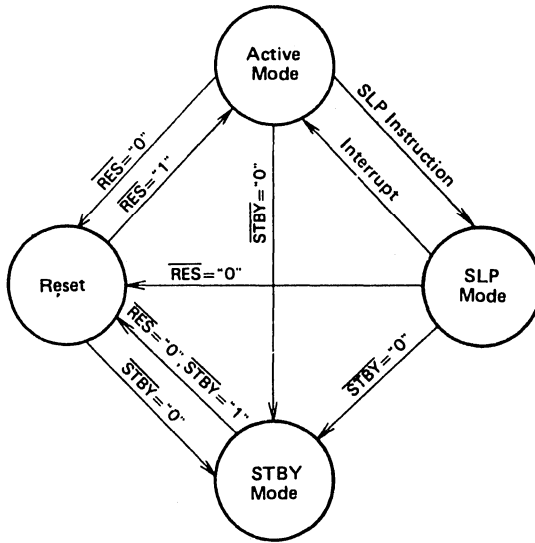


Fig. 2-12-2 Transitions among Active Mode, Standby Mode Sleep Mode, and Reset

## 2.13 TRAP Function

The CPU generates a TRAP interrupt with the highest priority when fetching an undefined op-code or an instruction from non-memory space. The TRAP prevents the system-burst caused by noise or a program error.

### (1) Op-Code Error

When fetching an undefined op-code, the CPU saves register as well as a normal interrupt and branches to the TRAP service routine (vector address=\$FFEE, \$FFEF). This has the priority next to reset.

### (2) Address Error

When an instruction is fetched from excluding internal ROM, RAM, or an external memory area, the MCU generates a TRAP interrupt as op-code error. If the instruction is fetched from external memory area without memory devices, this function is not applicable.

Table 2-13-1 shows addresses where an address error occurs to each mode. This function is available only for the instruction fetch, and is not applicable to the normal data read/write.

Table 2-13-1 Address Applicable to Address Errors

Mode	0	1	2	5	6	7
Address	\$0000	\$0000	\$0000	\$0000	\$0000	\$0000
	}	}	}	}	}	}
	\$001F	\$001F	\$001F	\$007F	\$001F	\$007F
				\$0200		\$0100
				}		}
			\$EFFF		\$EFFF	

### 3. EPROM (PROM) PROGRAMMING AND TECHNICAL SPECIFICATIONS

#### 3.1 PROM Mode

In PROM mode, on-chip EPROM can be programmed while other MCU functions stop operating.

The HD63701V0 can be configured in the PROM mode by connecting P<sub>20</sub> to V<sub>CC</sub>, P<sub>21</sub> to V<sub>SS</sub>, P<sub>22</sub> to GND, XTAL, STBY and  $\overline{\text{NMI}}$  to GND, and EXTAL to V<sub>CC</sub> respectively. See Figure 3-2.

The on-chip EPROM can be programmed and read in the same way as the 27256. Therefore, general purpose PROM programmer can perform programming the on-chip PROM. At this time, a socket adaptor which changes the number of pins from 40-pin to 28-pin is necessary. Note that the address range must be \$0000 through \$00FF because the on-chip EPROM is 4k bytes. Fill remainder of EPROM area with FFFF for PROM programmer to correctly verify. The Memory map in PROM mode is shown in Figure 3-3.

GND	1	40	NC
GND	2	39	NC
V <sub>CC</sub>	3	38	NC
GND	4	37	O <sub>0</sub>
A <sub>9</sub>	5	36	O <sub>1</sub>
V <sub>PP</sub>	6	35	O <sub>2</sub>
GND	7	34	O <sub>3</sub>
V <sub>CC</sub>	8	33	O <sub>4</sub>
V <sub>CC</sub>	9	32	O <sub>5</sub>
GND	10	31	O <sub>6</sub>
NC	11	HD63701V0	O <sub>7</sub>
NC	12	29	A <sub>8</sub>
A <sub>0</sub>	13	28	A <sub>14</sub>
A <sub>1</sub>	14	27	A <sub>10</sub>
A <sub>2</sub>	15	26	A <sub>11</sub>
A <sub>3</sub>	16	25	A <sub>12</sub>
A <sub>4</sub>	17	24	A <sub>13</sub>
A <sub>5</sub>	18	23	$\overline{\text{OE}}$
A <sub>6</sub>	19	22	$\overline{\text{CE}}$
A <sub>7</sub>	20	21	V <sub>CC</sub>

Fig. 3-1 HD63701V0 Pin Assignment in PROM Mode

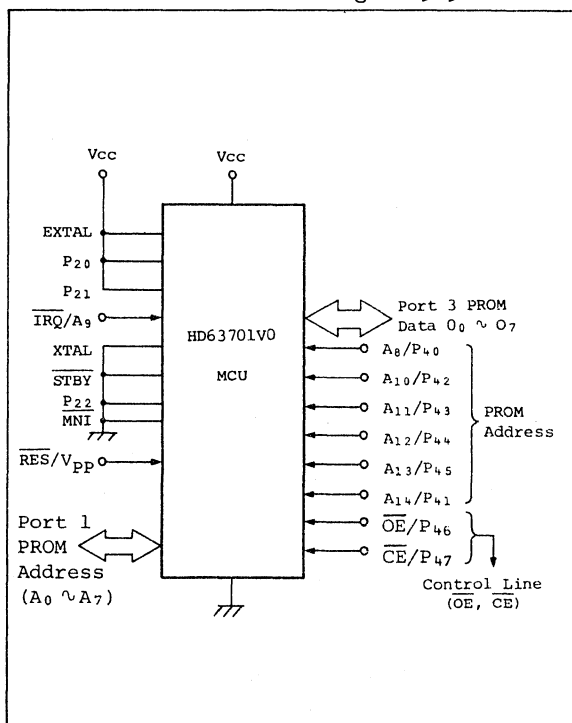


Fig. 3-2 Symbolical Pin Configuration in PROM Mode



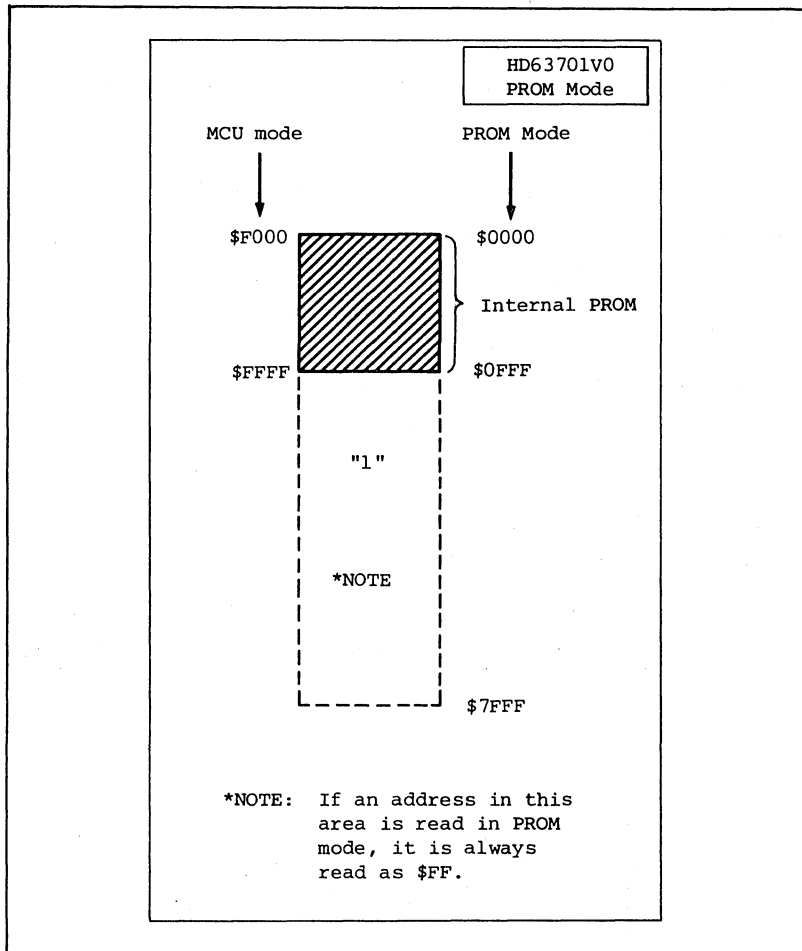


Fig. 3-3 Memory Map in PROM Mode

### 3.2 Programming/Verification

The on-chip EPROM can be programmed in high speed programming scheme, which allows the on-chip EPROM to be programmed effectively without damaging the device by voltage and provides high reliability.

The basic programming flow is shown in Figure 3-4.

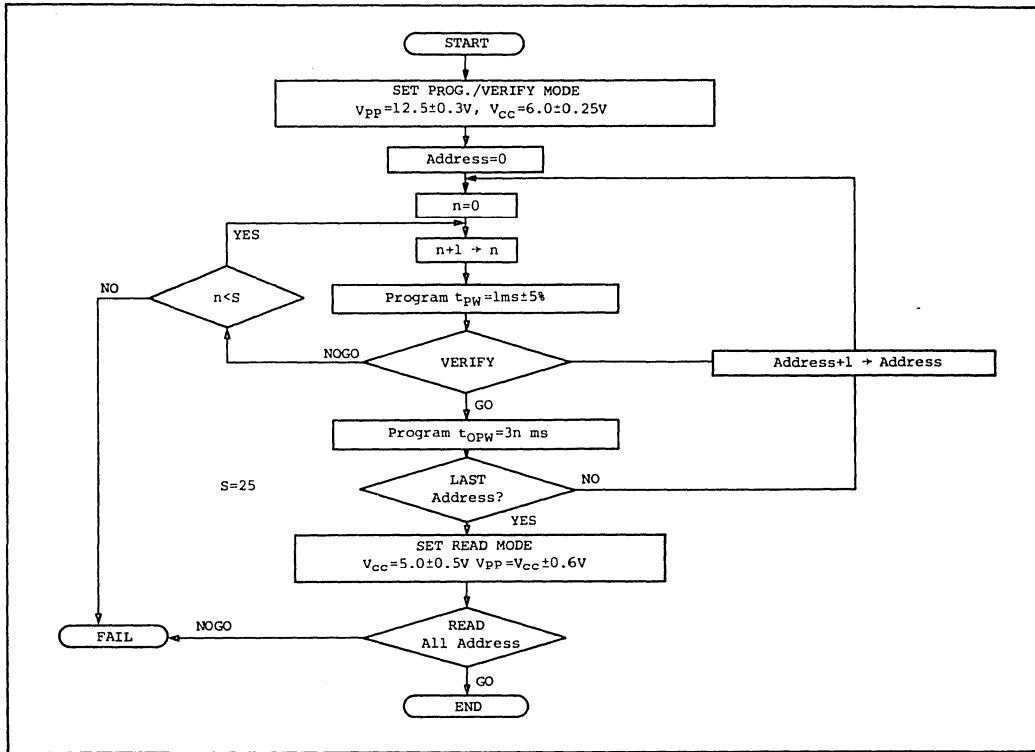


Fig. 3-4 Programming Flow

Table 3-1 Mode Selection

Mode	Pin	$\overline{CE}$	$\overline{OE}$	V <sub>PP</sub>	O <sub>0</sub> ~ O <sub>7</sub>
Read		L	L	V <sub>CC</sub>	D out
Output Disable		L	H	V <sub>CC</sub>	High Z
High performance Program		L	H	V <sub>PP</sub>	D in
Verify		H	L	V <sub>PP</sub>	D out
Program Inhibit		H	H	V <sub>PP</sub>	High Z

### 3.3 Erasure (with window package type)

The EPROM is erased by exposing an LSI to ultraviolet light. All erased bits are in the "1" state.

The conditions for erasing are: ultraviolet light with wavelength of 2573Å, and a minimum irradiation of 15W·sec/cm<sup>2</sup>. These conditions are satisfied by exposing the LSI to an ultraviolet lamp rated at 12,000µW/cm<sup>2</sup> for 20 minutes, at a distance of about one



inch. Dust of the cap must be removed by a solvent which does not damage the package, because the dust reduces the transmittance of the ultraviolet light.

### 3.4 On-chip PROM Characteristics and Application

#### (1) Principles of programming/erasure

The HD63701V0's memory cells are the same as the EPROM's. Therefore, they are programmed by applying high voltage to control gates and drains, which injects hot electrons into the floating gate. The condensed electrons in the floating gate are stable, surrounded by an energy barrier of SiO<sub>2</sub> film, and the proper bit becomes "0" due to the memory threshold voltage change.

Electrons in memory devices decrease as time goes by. This is caused by the following:

- ① Ultraviolet light, discharged by photo emitting electrons (erasure principle);
- ② Heat, discharged by thermal emitting electrons;
- ③ High voltage; discharged by a high electric field, control gate or drain.

If the oxide film covering a floating gate is defective, the erasure becomes great. Normally electron erasing does not occur because defective devices are removed.

The proper bit for a memory device whose floating gate does not condense electrons is "1".

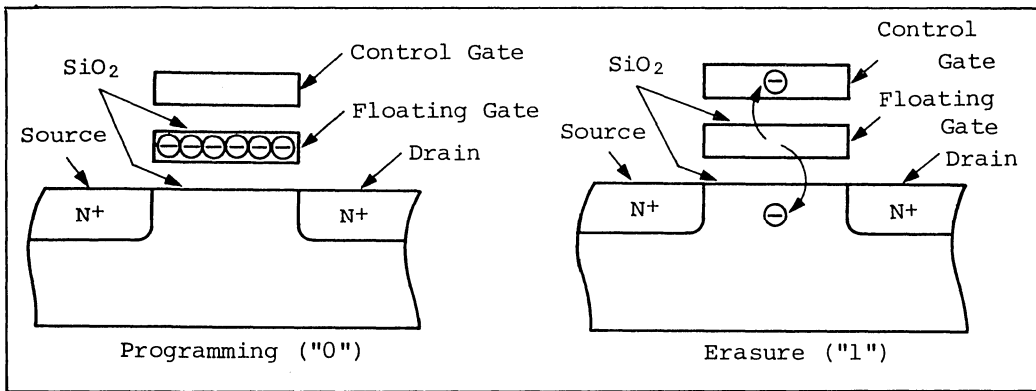


Fig. 3-5 Cross Section of EPROM Memory Cell

(2) Precautions on programming

The higher the program voltage  $V_{pp}$  or the longer program pulse width ( $t_{pw}$ ), better the programming because many electrons flow. However, data should be programmed under specified voltage and timing conditions. If over-voltage is applied to  $V_{pp}$ , the p-n junction may be damaged and permanent damage may occur. Pay particularly attention to PROM programmer overshoot. Minus voltage noise causes a parasitic transistor effect, which may cause apparent breakdown voltage.

The HD63701V0 is connected electrically to the PROM programmer through a socket adapter, so also pay attention to the followings:

- ① Confirm before programming that the socket adapter is firmly fixed on the PROM writer.
- ② Do not touch the socket adapter or the LSI during programming because writing malfunction may occur from bad contacts.

- (3) Precautions for using the window package type after programming

(NOTES)

- ① Transient current may cause permanent damage to the device if the socket adaptor and the device are not installed in the PROM programmer correctly.

Care must be taken to install the socket adaptor and the device in the PROM programmer correctly before programming the PROM.

- ② Note that the HD63701V0 programming voltage  $V_{pp}$  must be 12.5V, not 21V. If the  $V_{pp}$  is set to 21V, it may cause permanent damage to the device. Select the Intel 27256 mode in the PROM programmer to apply 12.5V to the  $V_{pp}$ .

- (a) Glass window for erasure

If the glass window comes in contact with plastic or something else with a static charge, the LSI may malfunction due to electrostatic charge on the surface of the window.

If this occurs, exposing the LSI to ultraviolet light for a few minutes neutralizes the charge and restores it to normal function. However, charge weight stored in the floating gate decreases at the same time, so re-programming is recommended.

Electrostatic charge buildup on the window is a fundamental cause of malfunction. Measurement for its prevention are the same as those for preventing electrostatic break-down:

- ① Operators should be grounded when handling equipment.
- ② Do not rub the glass window with plastics.
- ③ Be careful of coolant sprays, which may include a few ions.
- ④ The ultraviolet shading label (which includes conductive material) is effective for neutralizing the charge.

(b) Precautions after programming the EPROM

If the device is exposed to fluorescent light or sun light, its memory contents may be reversed because they include a small quantity of ultraviolet light. In strong light the MCU may malfunction under the influence of photo-current. To prevent these problems, it is recommended that the device be used with the glass window for erasure covered with the ultraviolet shading label after programming.

A special label is on the market for this purpose. Labels made with metal are effective because they absorb ultraviolet light.

Note the following when selecting a shading label.

- ① Adhesion (mechanical intensity) ---- Adhesion is reduced with re-use or dust. When peeling the label, static electricity may occur. As a result, erasure and rewriting by ultraviolet light are recommended after peeling. (Sticking a new label above the old one can be done to change labels.)
- ② Allowable temperature range ---- The allowable temperature range and environment temperature of the shading label should be noted. If it is used under conditions exceeding this range, the paste may stiffen or adhere to the label, causing paste to remain on the window after peeling.
- ③ Moisture resistance ---- The allowable moisture range and environment conditions of the label should be noted. It is difficult to find a shade label applicable to all allowable environmental conditions of the MCU. The proper label should be selected depending on use.

### 3.5 INSTRUCTION SET OVERVIEW

Besides having object code compatible with the HD6801 series, the HD63701V0 adds 6 new instructions; enhances bit control instructions (AIM, EIM, OIM, TIM), index/accumulator exchange instruction (XGDX), and sleep instruction (SLP). These new instructions improve programming efficiency.

### 3.6 Addressing Modes

The HD63701V0 provides seven addressing modes. The adequate selection of these addressing mode will permit to implement an efficient and easy programming.

The addressing mode is determined by an instruction type and code. The addressing mode for each instruction is shown in Table 3-7 to 3-7-4 with execution time counted by the machine cycles. When the clock frequency is 4 MHz, the machine cycle time will be microseconds.

#### (1) Accumulator (ACCX) Addressing

Operand is contained in either accumulator A or B.

#### (2) Immediate Addressing

Operand is contained in the second byte of the instruction except LDS and LDX which contain operand in the second and the third bytes. These are two or three-byte instructions.

#### (3) Direct Addressing

The second byte of an instruction contains an effective address of operand. 256 byte area \$0 through \$255 can be addressed directly. Execution times can be reduced by storing operand in these locations. In configuring system, it is recommended that these locations should be RAM for users' data area. These are two-byte instructions, while the AIM, OIM, EIM and TIM are three-byte instructions.

(4) Extended Addressing

The second and third bytes of the instruction contain the effective address of the operand. These are three-byte instructions.

(5) Indexed Addressing

The effective address of the operand is the sum of the contents of the second byte and the lower byte of the Index Register. As for AIM, OIM, EIM and TIM instructions, the effective address is calculated by adding the contents of the third byte and the lower byte of the Index Register. The effective address is held in the Temporary Address Register, so the contents of the Index Register is not changed. These are two-byte instructions, while AIM, OIM, EIM and TIM are three-byte.

(6) Implied Addressing

The instruction itself gives the address. That is, the instruction addresses an Accumulator, Stack Pointer, Index Register, etc. This is a one-byte instruction.

(7) Relative Addressing

Relative addressing mode is only used in branch instructions. The branch address is calculated by adding the contents of the second byte and the lower byte of the Program Counter. At this time, a carry or borrow is added to the upper byte of the Program Counter. The span of relative addressing is from -126 to +129 from the op-code address. These are two-byte instructions.



### 3.7 Instruction Set

The HD63701V0 has an upward object code compatible with the HD6801 to utilize all instruction sets of the HMCS6800. The execution time of the key instruction is reduced to increase the system through-put. In addition, the bit manipulation instruction, the exchange instruction of the index and the accumulator, the sleep instruction are added. The followings are described here.

- Accumulator and memory manipulation instructions (See Table 3-7).
- Additional instructions.
- Index register and stack manipulation instructions (See Table 3-7-2).
- Jump and branch instructions (See Table 3-7-3).
- Condition code register manipulation instructions (See Table 3-7-4).
- Op-code map (See Table 3-7-5).

Table 3-7 Accumulator, Memory Manipulation Instructions

Operations	Mnemonic	Addressing Modes												Boolean/ Arithmetic Operation	Condition Code Register								
		IMMED			DIRECT			INDEX			EXTEND				IMPLIED			5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#		OP	~	#	H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	4	2	BB	4	3			A + M → A	!	!	!	!	!	!	
	ADDB	CB	2	2	DB	3	2	EB	4	2	FB	4	3			B + M → B	!	!	!	!	!	!	
Add Double	ADDD	C3	3	3	D3	4	2	E3	5	2	F3	5	3			A : B + M : M + 1 → A : B	!	!	!	!	!	!	
Add Accumulators	ABA													1B	1	1	A + B → A	!	!	!	!	!	
Add With Carry	ADCA	89	2	2	99	3	2	A9	4	2	B9	4	3			A + M + C → A	!	!	!	!	!	!	
	ADCB	C9	2	2	D9	3	2	E9	4	2	F9	4	3			B + M + C → B	!	!	!	!	!	!	
AND	ANDA	84	2	2	94	3	2	A4	4	2	B4	4	3			A · M → A	!	!	!	!	R	!	
	ANDB	C4	2	2	D4	3	2	E4	4	2	F4	4	3			B · M → B	!	!	!	!	R	!	
Bit Test	BIT A	85	2	2	95	3	2	A5	4	2	B5	4	3			A · M	!	!	!	!	R	!	
	BIT B	C5	2	2	D5	3	2	E5	4	2	F5	4	3			B · M	!	!	!	!	R	!	
Clear	CLR							5F	5	2	7F	5	3			00 → M	!	!	R	S	R	R	
	CLRA													4F	1	1	00 → A	!	!	R	S	R	R
	CLRB													5F	1	1	00 → B	!	!	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	4	2	B1	4	3			A - M	!	!	!	!	!	!	
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3			B - M	!	!	!	!	!	!	
Compare Accumulators	CBA													11	1	1	A - B	!	!	!	!	!	!
	COM							63	6	2	73	6	3			M - M	!	!	!	!	R	S	
	COMA													43	1	1	A - A	!	!	!	!	R	S
Complement, 1's	COMB													53	1	1	B - B	!	!	!	!	R	S
	COMB																00 - M → M	!	!	!	!	①	②
	NEG							60	6	2	70	6	3				00 - A → A	!	!	!	!	①	②
Complement, 2's (Negate)	NEGA													40	1	1	00 - A → A	!	!	!	!	①	②
	NEGB													50	1	1	00 - B → B	!	!	!	!	①	②
Decimal Adjust, A	DAA													19	2	1	Converts binary add of BCD characters into BCD format	!	!	!	!	!	③
Decrement	DEC							6A	6	2	7A	6	3			M - 1 → M	!	!	!	!	!	!	
	DECA													4A	1	1	A - 1 → A	!	!	!	!	!	!
	DECB													5A	1	1	B - 1 → B	!	!	!	!	!	!
Exclusive OR	EORA	88	2	2	98	3	2	A8	4	2	B8	4	3			A ⊕ M → A	!	!	!	!	!	!	
	EORB	C8	2	2	D8	3	2	E8	4	2	F8	4	3			B ⊕ M → B	!	!	!	!	!	!	
Increment	INC							6C	6	2	7C	6	3			M + 1 → M	!	!	!	!	!	!	
	INCA													4C	1	1	A + 1 → A	!	!	!	!	!	!
	INCB													5C	1	1	B + 1 → B	!	!	!	!	!	!
Load Accumulator	LDAA	86	2	2	96	3	2	A6	4	2	B6	4	3			M → A	!	!	!	!	!	!	
	LDAB	C6	2	2	D6	3	2	E6	4	2	F6	4	3			M → B	!	!	!	!	!	!	
Load Double Accumulator	LDD	CC	3	3	DC	4	2	EC	5	2	FC	5	3			M + 1 → B, M → A	!	!	!	!	!	!	
Multiply Unsigned	MUL													3D	7	1	A × B → A : B	!	!	!	!	!	!
OR, Inclusive	ORAA	8A	2	2	9A	3	2	AA	4	2	BA	4	3			A + M → A	!	!	!	!	!	!	
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3			B + M → B	!	!	!	!	!	!	
Push Data	PSHA													36	4	1	A → M <sub>sp</sub> , SP - 1 → SP	!	!	!	!	!	!
	PSHB													37	4	1	B → M <sub>sp</sub> , SP - 1 → SP	!	!	!	!	!	!
Pull Data	PULA													32	3	1	SP + 1 → SP, M <sub>sp</sub> → A	!	!	!	!	!	!
	PULB													33	3	1	SP + 1 → SP, M <sub>sp</sub> → B	!	!	!	!	!	!
Rotate Left	ROL							69	6	2	79	6	3			M <sub>A</sub> → C, C → b <sub>7</sub> , b <sub>0</sub> → M <sub>A</sub>	!	!	!	!	!	!	
	ROLA													49	1	1	M <sub>A</sub> → C, C → b <sub>7</sub> , b <sub>0</sub> → M <sub>A</sub>	!	!	!	!	!	!
	ROLB													59	1	1	M <sub>B</sub> → C, C → b <sub>7</sub> , b <sub>0</sub> → M <sub>B</sub>	!	!	!	!	!	!
Rotate Right	ROR							66	6	2	76	6	3			M <sub>A</sub> → C, C → b <sub>7</sub> , b <sub>0</sub> → M <sub>A</sub>	!	!	!	!	!	!	
	RORA													46	1	1	M <sub>A</sub> → C, C → b <sub>7</sub> , b <sub>0</sub> → M <sub>A</sub>	!	!	!	!	!	!
	RORB													56	1	1	M <sub>B</sub> → C, C → b <sub>7</sub> , b <sub>0</sub> → M <sub>B</sub>	!	!	!	!	!	!

4

Note) Condition Code Register will be explained in Note of Table 3-7-4.

(to be continued)

Table 3-7-1 Accumulator, Memory Manipulation Instructions

Operations	Mnemonic	Addressing Modes												Boolean/ Arithmetic Operation	Condition Code Register																			
		IMMED			DIRECT			INDEX			EXTEND				IMPLIED			H	I	N	Z	V	C											
		OP	~	#	OP	~	#	OP	~	#	OP	~	#		OP	~	#																	
Shift Left Arithmetic	ASL							68	6	2	78	6	3											M		•	•	•	•	•	•			
	ASLA													48	1	1								A		•	•	•	•	•	•			
	ASLB														58	1	1								B		•	•	•	•	•	•		
Double Shift Left, Arithmetic	ASLD															05	1	1								C		•	•	•	•	•	•	
Shift Right Arithmetic	ASR							67	6	2	77	6	3											M		•	•	•	•	•	•			
	ASRA													47	1	1								A		•	•	•	•	•	•			
	ASRB														57	1	1								B		•	•	•	•	•	•		
Shift Right Logical	LSR							64	6	2	74	6	3											M		•	•	•	•	•	•			
	LSRA													44	1	1								A		•	•	•	•	•	•			
	LSRB														54	1	1								B		•	•	•	•	•	•		
Double Shift Right Logical	LSRD															04	1	1								C		•	•	•	•	•	•	
Store Accumulator	STAA				97	3	2	A7	4	2	B7	4	3												A → M	•	•	•	•	•	•			
	STAB				D7	3	2	E7	4	2	F7	4	3												B → M	•	•	•	•	•	•			
Store Double Accumulator	STD				DD	4	2	ED	5	2	FD	5	3												A → M B → M + 1	•	•	•	•	•	•			
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3												A - M → A	•	•	•	•	•	•			
	SUBB	C0	2	2	D0	3	2	E0	4	2	F0	4	3												B - M → B	•	•	•	•	•	•			
Double Subtract	SUBD	83	3	3	93	4	2	A3	5	2	B3	5	3												A : B - M : M + 1 → A : B	•	•	•	•	•	•			
Subtract Accumulators	SBA																10	1	1									A - B → A	•	•	•	•	•	•
Subtract With Carry	SBCA	82	2	2	92	3	2	A2	4	2	B2	4	3												A - M - C → A	•	•	•	•	•	•			
	SBCB	C2	2	2	D2	3	2	E2	4	2	F2	4	3												B - M - C → B	•	•	•	•	•	•			
Transfer Accumulators	TAB													16	1	1									A → B	•	•	•	•	•	•			
	TBA														17	1	1									B → A	•	•	•	•	•	•		
Test Zero or Minus	TST							6D	4	2	7D	4	3												M - 00	•	•	•	•	•	•			
	TSTA													4D	1	1									A - 00	•	•	•	•	•	•			
	TSTB														5D	1	1									B - 00	•	•	•	•	•	•		
And Immediate	AIM				71	6	3	61	7	3														M - IMM → M	•	•	•	•	•	•				
OR Immediate	OIM				72	6	3	62	7	3														M + IMM → M	•	•	•	•	•	•				
EOR Immediate	EIM				75	6	3	65	7	3														M ⊕ IMM → M	•	•	•	•	•	•				
Test Immediate	TIM				7B	4	3	6B	5	3														M - IMM	•	•	•	•	•	•				

Note) Condition Code Register will be explained in Note of Table 3-7-4.

Additional Instructions

In addition to the HD6801 Instruction Set, the HD63701V0 has the following new instructions:

AIM --- (M) · (IMM) → (M)

Executes "AND" operation between the immediate data and the memory contents, and stores the result in the memory.

OIM --- (M) + (IMM) → (M)

Executes "OR" operation between the immediate data and the memory contents, and stores the result in the memory.

EIM --- (M) ⊕ (IMM) → (M)

Executes "EOR" operation between the immediate data and the memory contents, and stores the result in the memory.

TIM --- (M) · (IMM)

Executes "AND" operation between the immediate data and the memory contents, and changes the flag of associated condition code register.

AIM, OIM, EIM and TIM are three bytes instructions; the first byte is op-code, the second is immediate data, and the third is address modifier.

XGDX --- (ACCD) ↔ (IX)

Exchanges the contents of accumulator and the index register.

SLP --- The MCU goes to the sleep mode. Refer to "Low Power Dissipation Mode" for more details of the sleep mode.

Table 3-7-2 Index Register, Stack Manipulation Instructions

Pointer Operations	Mnemonic	Addressing Modes										Boolean/ Arithmetic Operation	Condition Code Register							
		IMMED.		DIRECT		INDEX		EXTEND		IMPLIED			5	4	3	2	1	0		
		OP	~ #	OP	~ #	OP	~ #	OP	~ #	OP	~ #		H	I	N	Z	V	C		
Compare Index Reg	CPX	8C	3 3	9C	4 2	AC	5 2	BC	5 3					X - M.M + 1	•	•	!	!	!	!
Decrement Index Reg	DEX									09	1 1			X - 1 - X	•	•	•	!	•	•
Decrement Stack Pntr	DES									34	1 1			SP - 1 - SP	•	•	•	•	•	•
Increment Index Reg	INX									08	1 1			X + 1 - X	•	•	•	!	•	•
Increment Stack Pntr	INS									31	1 1			SP + 1 - SP	•	•	•	•	•	•
Load Index Reg	LDX	CE	3 3	DE	4 2	EE	5 2	FE	5 3					M → X <sub>H</sub> , (M + 1) → X <sub>L</sub>	•	•	!	!	R	•
Load Stack Pntr	LDS	8E	3 3	9E	4 2	AE	5 2	BE	5 3					M → SP <sub>H</sub> , (M + 1) → SP <sub>L</sub>	•	•	!	!	R	•
Store Index Reg	STX			DF	4 2	EF	5 2	FF	5 3					X <sub>H</sub> → M, X <sub>L</sub> → (M + 1)	•	•	!	!	R	•
Store Stack Pntr	STS			9F	4 2	AF	5 2	BF	5 3					SP <sub>H</sub> → M, SP <sub>L</sub> → (M + 1)	•	•	!	!	R	•
Index Reg → Stack Pntr	TXS									35	1 1			X - 1 - SP	•	•	•	•	•	•
Stack Pntr → Index Reg	TSX									30	1 1			SP + 1 - X	•	•	•	•	•	•
Add	ABX									3A	1 1			B + X - X	•	•	•	•	•	•
Push Data	PSHX									3C	5 1			X <sub>L</sub> → M <sub>sp</sub> , SP - 1 - SP X <sub>H</sub> → M <sub>sp</sub> , SP - 1 - SP	•	•	•	•	•	•
Pull Data	PULX									38	4 1			SP + 1 - SP, M <sub>sp</sub> → X <sub>H</sub> SP + 1 - SP, M <sub>sp</sub> → X <sub>L</sub>	•	•	•	•	•	•
Exchange	XGDX									18	2 1			ACCD ← IX	•	•	•	•	•	•

Note) Condition Code Register will be explained in Note of Table 3-7-4.

Table 3-7-3 Jump, Branch Instruction

Operations	Mnemonic	Addressing Modes												Branch Test	Condition Code Register								
		RELATIVE		DIRECT		INDEX		EXTEND		IMPLIED		5	4		3	2	1	0					
		OP	~ #	OP	~ #	OP	~ #	OP	~ #	OP	~ #								H	I	N	Z	V
Branch Always	BRA	20	3	2													None	•	•	•	•	•	•
Branch Never	BRN	21	3	2													None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	3	2													C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	3	2													C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	3	2													Z = 1	•	•	•	•	•	•
Branch If > Zero	BGE	2C	3	2													$N \oplus V = 0$	•	•	•	•	•	•
Branch If > Zero	BGT	2E	3	2													$Z + (N \oplus V) = 0$	•	•	•	•	•	•
Branch If Higher	BHI	22	3	2													$C + Z = 0$	•	•	•	•	•	•
Branch If < Zero	BLE	2F	3	2													$Z + (N \oplus V) = 1$	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	3	2													$C + Z = 1$	•	•	•	•	•	•
Branch If < Zero	BLT	2D	3	2													$N \oplus V = 1$	•	•	•	•	•	•
Branch If Minus	BMI	2B	3	2													N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	3	2													Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	3	2													V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	3	2													V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	3	2													N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	5	2														•	•	•	•	•	•
Jump	JMP							6E	3	2	7E	3	3					•	•	•	•	•	•
Jump To Subroutine	JSR				9D	5	2	AD	5	2	BD	6	3					•	•	•	•	•	•
No Operation	NOP													01	1	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI													3B	10	1		•	•	•	•	•	•
Return From Subroutine	RTS													39	5	1		•	•	•	•	•	•
Software Interrupt	SWI													3F	12	1		•	S	•	•	•	•
Wait for Interrupt*	WAI													3E	9	1		•	Ⓣ	•	•	•	•
Sleep	SLP													1A	4	1		•	•	•	•	•	•

Note) \*WAI puts R/W high; Address Bus goes to FFFF; Data Bus goes to the three state level. Condition Code Register will be explained in Note of Table 3-7-4.

Table 3-7-4 Condition Code Register Manipulation Instructions


Operations	Mnemonic	Addressing Modes		Boolean Operation	Condition Code Register							
		IMPLIED			5	4	3	2	1	0		
OP	~ #	H	I	N							Z	V
Clear Carry	CLC	0C	1	1	0 → C	•	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	1	1	0 → I	•	R	•	•	•	•	•
Clear Overflow	CLV	0A	1	1	0 → V	•	•	•	•	•	R	•
Set Carry	SEC	0D	1	1	1 → C	•	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	1	1	1 → I	•	S	•	•	•	•	•
Set Overflow	SEV	0B	1	1	1 → V	•	•	•	•	•	•	S
Accumulator A → CCR	TAP	06	1	1	A → CCR	Ⓣ						
CCR → Accumulator A	TPA	07	1	1	CCR → A	•	•	•	•	•	•	•

[NOTE] Condition Code Register Notes: (Bit set if test is true and cleared otherwise)

- ① (Bit V) Test: Result = 10000000?
- ② (Bit C) Test: Result = 00000000?
- ③ (Bit C) Test: BCD Character of high-order byte greater than 10? (Not cleared if previously set)
- ④ (Bit V) Test: Operand = 10000000 prior to execution?
- ⑤ (Bit V) Test: Operand = 01111111 prior to execution?
- ⑥ (Bit V) Test: Set equal to NeC=1 after the execution of instructions
- ⑦ (Bit N) Test: Result less than zero? (Bit 15=1)
- ⑧ (All) Load Condition Code Register from Stack.
- ⑨ (Bit I) Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exist the wait state.
- ⑩ (All Bit) Set according to the contents of Accumulator A.
- ⑪ (Bit C) Result of Multiplication Bit 7=1? (ACCB)

Table 3-7-5 OP-Code Map

OP CODE					ACC A	ACC B	IND	EXT DIR	ACCA or SP				ACCB or X				
	HI	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
LO	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000	0	/	SBA	BRA	TSX	/	/	NEG	/	/	/	/	/	SUB	/	/	0
0001	1	NOP	CBA	BRN	INS	/	/	AIM	/	/	/	/	/	CMP	/	/	1
0010	2	/	/	BHI	PULA	/	/	OIM	/	/	/	/	/	SBC	/	/	2
0011	3	/	/	BLS	PULB	/	/	COM	/	/	SUBD	/	/	ADDD	/	/	3
0100	4	LSRD	/	BCC	DES	/	/	LSR	/	/	/	/	/	AND	/	/	4
0101	5	ASLD	/	BCS	TXS	/	/	EIM	/	/	/	/	/	BIT	/	/	5
0110	6	TAP	TAB	BNE	PSHA	/	/	ROR	/	/	/	/	/	LDA	/	/	6
0111	7	TPA	TBA	BEQ	PSHB	/	/	ASR	/	/	STA	/	/	STA	/	/	7
1000	8	INX	XGDX	BVC	PULX	/	/	ASL	/	/	/	/	/	EOR	/	/	8
1001	9	DEX	DAA	BVS	RTS	/	/	ROL	/	/	/	/	/	ADC	/	/	9
1010	A	CLV	SLP	BPL	ABX	/	/	DEC	/	/	/	/	/	ORA	/	/	A
1011	B	SEV	ABA	BMI	RTI	/	/	TIM	/	/	/	/	/	ADD	/	/	B
1100	C	CLC	/	BGE	PSHX	/	/	INC	/	/	CPX	/	/	LDD	/	/	C
1101	D	SEC	/	BLT	MUL	/	/	TST	/	BSR	/	JSR	/	STD	/	/	D
1110	E	CLI	/	BGT	WAI	/	/	JMP	/	/	LDS	/	/	LDX	/	/	E
1111	F	SEI	/	BLE	SWI	/	/	CLR	/	/	STS	/	/	STX	/	/	F

UNDEFINED OP CODE 

\* Only each instructions of AIM, OIM, EIM, TIM

3.8 Instruction Execution Cycles

In the HMCS6800 series, the execution cycle of each instruction is counted from the start of the op-code fetch.

The HD63701V0 employs a mechanism of the pipeline control for the instruction fetch, so the subsequent instruction is prefetched during the current instruction being executed.

Therefore, the method to count instruction cycles used in the HMCS6800 series cannot be applied to the instruction cycles such as MULT, PULL, DAA and XGDX in the HD63701V0.

Table 3-7-6, provides the information about the Address Bus, Data Bus, and R/W status in cycle by cycle basis during each instruction execution.

Table 3-7-6 Cycle by Cycle Operation

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	Data Bus
<b>IMMEDIATE</b>					
ADC ADD	2	1	Op Code Address + 1	1	Operand Data
AND BIT		2	Op Code Address + 2	1	Next Op Code
CMP EOR					
LDA ORA					
SBC SUB					
ADDD CPX	3	1	Op Code Address + 1	1	Operand Data (MSB)
LDD LDS		2	Op Code Address + 2	1	Operand Data (LSB)
LDX SUBD		3	Op Code Address + 3	1	Next Op Code
<b>DIRECT</b>					
ADC ADD	3	1	Op Code Address + 1	1	Address of Operand (LSB)
AND BIT		2	Address of Operand	1	Operand Data
CMP EOR		3	Op Code Address + 2	1	Next Op Code
LDA ORA					
SBC SUB					
STA	3	1	Op Code Address + 1	1	Destination Address
		2	Destination Address	0	Accumulator Data
		3	Op Code Address + 2	1	Next Op Code
ADDD CPX	4	1	Op Code Address + 1	1	Address of Operand (LSB)
LDD LDS		2	Address of Operand	1	Operand Data (MSB)
LDX SUBD		3	Address of Operand + 1	1	Operand Data (LSB)
		4	Op Code Address + 2	1	Next Op Code
STD STS	4	1	Op Code Address + 1	1	Destination Address (LSB)
STX		2	Destination Address	0	Register Data (MSB)
		3	Destination Address + 1	0	Register Data (LSB)
		4	Op Code Address + 2	1	Next Op Code
JSR	5	1	Op Code Address + 1	1	Jump Address (LSB)
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer - 1	0	Return Address (MSB)
		5	Jump Address	1	First Subroutine Op Code
TIM	4	1	Op Code Address + 1	1	Immediate Data
		2	Op Code Address + 2	1	Address of Operand (LSB)
		3	Address of Operand	1	Operand Data
		4	Op Code Address + 3	1	Next Op Code
AIM EIM	6	1	Op Code Address + 1	1	Immediate Data
OIM		2	Op Code Address + 2	1	Address of Operand (LSB)
		3	Address of Operand	1	Operand Data
		4	FFFF	1	Restart Address (LSB)
		5	Address of Operand	0	New Operand Data
		6	Op Code Address + 3	1	Next Op Code

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	Data Bus
-----------------------------	--------	---------	-------------	-----	----------

INDEXED

JMP	3	1 2 3	Op Code Address + 1 FFFF Jump Address	1 1 1	Offset Restart Address (LSB) First Op Code of Jump Routine
ADC ADD AND BIT CMP EOR LDA ORA SBC SUB TST	4	1 2 3 4	Op Code Address + 1 FFFF IX + Offset Op Code Address + 2	1 1 1 1	Offset Restart Address (LSB) Operand Data Next Op Code
STA	4	1 2 3 4	Op Code Address + 1 FFFF IX + Offset Op Code Address + 2	1 1 0 1	Offset Restart Address (LSB) Accumulator Data Next Op Code
ADDD LDD CPX LDX LDS LDX SUBD	5	1 2 3 4 5	Op Code Address + 1 FFFF IX + Offset IX + Offset + 1 Op Code Address + 2	1 1 1 1 1	Offset Restart Address (LSB) Operand Data (MSB) Operand Data (LSB) Next Op Code
STD STS STX	5	1 2 3 4 5	Op Code Address + 1 FFFF IX + Offset IX + Offset + 1 Op Code Address + 2	1 1 0 0 1	Offset Restart Address (LSB) Register Data (MSB) Register Data (LSB) Next Op Code
JSR	5	1 2 3 4 5	Op Code Address + 1 FFFF Stack Pointer Stack Pointer - 1 IX + Offset	1 1 0 0 1	Offset Restart Address (LSB) Return Address (LSB) Return Address (MSB) First Subroutine Op Code
ASL ASR COM DEC INC LSR NEG ROL ROR	6	1 2 3 4 5 6	Op Code Address + 1 FFFF IX + Offset FFFF IX + Offset Op Code Address + 2	1 1 1 1 0 1	Offset Restart Address (LSB) Operand Data Restart Address (LSB) New Operand Data Next Op Code
TIM	5	1 2 3 4 5	Op Code Address + 1 Op Code Address + 2 FFFF IX + Offset Op Code Address + 3	1 1 1 1 1	Immediate Data Offset Restart Address (LSB) Operand Data Next Op Code
CLR	5	1 2 3 4 5	Op Code Address + 1 FFFF IX + Offset IX + Offset Op Code Address + 2	1 1 1 0 1	Offset Restart Address (LSB) Operand Data 00 Next Op Code
AIM EIM OIM	7	1 2 3 4 5 6 7	Op Code Address + 1 Op Code Address + 2 FFFF IX + Offset FFFF IX + Offset Op Code Address + 3	1 1 1 1 1 0 1	Immediate Data Offset Restart Address (LSB) Operand Data Restart Address (LSB) New Operand Data next Op Code

(to be continued)



Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	Data Bus
EXTEND					
JMP	3	1 2 3	Op Code Address + 1 Op Code Address + 2 Jump Address	1 1 1	Jump Address (MSB) Jump Address (LSB) Next Op Code
ADC ADD TST AND BIT CMP EOR LDA ORA SBC SUB	4	1 2 3 4	Op Code Address + 1 Op Code Address + 2 Address of Operand Op Code Address + 3	1 1 1 1	Address of Operand (MSB) Address of Operand (LSB) Operand Data Next Op Code
STA	4	1 2 3 4	Op Code Address + 1 Op Code Address + 2 Destination Address Op Code Address + 3	1 1 0 1	Destination Address (MSB) Destination Address (LSB) Accumulator Data Next Op Code
ADDD LDD CPX LDD LDS LDX SUBD	5	1 2 3 4 5	Op Code Address + 1 Op Code Address + 2 Address of Operand Address of Operand + 1 Op Code Address + 3	1 1 1 1 1	Address of Operand (MSB) Address of Operand (LSB) Operand Data (MSB) Operand Data (LSB) Next Op Code
STD STS STX	5	1 2 3 4 5	Op Code Address + 1 Op Code Address + 2 Destination Address Destination Address + 1 Op Code Address + 3	1 1 0 0 1	Destination Address (MSB) Destination Address (LSB) Register Data (MSB) Register Data (LSB) Next Op Code
JSR	6	1 2 3 4 5 6	Op Code Address + 1 Op Code Address + 2 FFFF Stack Pointer Stack Pointer - 1 Jump Address	1 1 1 0 0 1	Jump Address (MSB) Jump Address (LSB) Restart Address (LSB) Return Address (LSB) Return Address (MSB) First Subroutine Op Code
ASL ASR COM DEC INC LSR NGE ROL ROR	6	1 2 3 4 5 6	Op Code Address + 1 Op Code Address + 2 Address of Operand FFFF Address of Operand Op Code Address + 3	1 1 1 1 0 1	Address of Operand (MSB) Address of Operand (LSB) Operand Data Restart Address (LSB) New Operand Data Next Op Code
CLR	5	1 2 3 4 5	Op Code Address + 1 Op Code Address + 2 Address of Operand Address of Operand Op Code Address + 3	1 1 1 0 1	Address of Operand (MSB) Address of Operand (LSB) Operand Data 00 Next Op Code

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	Data Bus
-----------------------------	--------	---------	-------------	-----	----------

IMPLIED

ABA ABX ASL ASLD ASR CBA CLC CLI CLR CLV COM DEC DES DEX INC INS INX LSR LSRD ROL ROR NOP SBA SEC SEI SEV TAB TAP TBA TPA TST TSX TXS	1	1	Op Code Address + 1	1	Next Op Code
DAA XGDY	2	1 2	Op Code Address + 1 FFFF	1 1	Next Op Code Restart Address (LSB)
PULA PULB	3	1 2 3	Op Code Address + 1 FFFF Stack Pointer + 1	1 1 1	Next Op Code Restart Address (LSB) Data from Stack
PSHA PSHB	4	1 2 3 4	Op Code Address + 1 FFFF Stack Pointer Op Code Address + 1	1 1 0 1	Next Op Code Restart Address (LSB) Accumulator Data Next Op Code
PULX	4	1 2 3 4	Op Code Address + 1 FFFF Stack Pointer + 1 Stack Pointer + 2	1 1 1 1	Next Op Code Restart Address (LSB) Data from Stack (MSB) Data from Stack (LSB)
PSHX	5	1 2 3 4 5	Op Code Address + 1 FFFF Stack Pointer Stack Pointer - 1 Op Code Address + 1	1 1 0 0 1	Next Op Code Restart Address (LSB) Index Register (LSB) Index Register (MSB) Next Op Code
RTS	5	1 2 3 4 5	Op Code Address + 1 FFFF Stack Pointer + 1 Stack Pointer + 2 Return Address	1 1 1 1 1	Next Op Code Restart Address (LSB) Return Address (MSB) Return Address (LSB) First Op Code of Return Routine
MUL	7	1 2 3 4 5 6 7	Op Code Address + 1 FFFF FFFF FFFF FFFF FFFF FFFF	1 1 1 1 1 1 1	Next Op Code Restart Address (LSB) Restart Address (LSB) Restart Address (LSB) Restart Address (LSB) Restart Address (LSB) Restart Address (LSB)

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	Data Bus
-----------------------------	--------	---------	-------------	-----	----------

IMPLIED

WAI	9	1	Op Code Address + 1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer - 1	0	Return Address (MSB)
		5	Stack Pointer - 2	0	Index Register (LSB)
		6	Stack Pointer - 3	0	Index Register (MSB)
		7	Stack Pointer - 4	0	Accumulator A
		8	Stack Pointer - 5	0	Accumulator B
		9	Stack Pointer - 6	0	Conditional Code Register
RTI	10	1	Op Code Address + 1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer + 1	1	Conditional Code Register
		4	Stack Pointer + 2	1	Accumulator B
		5	Stack Pointer + 3	1	Accumulator A
		6	Stack Pointer + 4	1	Index Register (MSB)
		7	Stack Pointer + 5	1	Index Register (LSB)
		8	Stack Pointer + 6	1	Return Address (MSB)
		9	Stack Pointer + 7	1	Return Address (LSB)
		10	Return Address	1	First Op Code of Return Routine
SWI	12	1	Op Code Address + 1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer - 1	0	Return Address (MSB)
		5	Stack Pointer - 2	0	Index Register (LSB)
		6	Stack Pointer - 3	0	Index Register (MSB)
		7	Stack Pointer - 4	0	Accumulator A
		8	Stack Pointer - 5	0	Accumulator B
		9	Stack Pointer - 6	0	Conditional Code Register
		10	Vector Address FFFA	1	Address of SWI Routine (MSB)
		11	Vector Address FFFB	1	Address of SWI Routine (LSB)
		12	Address of SWI Routine	1	First Op Code of SWI Routine
SLP	4	1	Op Code Address + 1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		↑ Sleep ↓	FFFF		High Impedance - Non MPX Mode Address Bus - MPX Mode
		3	FFFF		↓ Restart Address (LSB)
4	Op Code Address + 1		Next Op Code		

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	Data Bus
-----------------------------	--------	---------	-------------	-----	----------

RELATIVE

BCC BCS	3	1	Op Code Address + 1	1	Branch Offset
BEQ BGE		2	FFFF	1	Restart Address (LSB)
BGT BHI		3	{ Branch Address Test = "1" Op Code Address Test = "0"	1	First Op Code of Branch Routine
BLE BLS					Next Op Code
BLT BMT					
BNE BPL					
BRA BRN					
BVC BVS					
BSR	5	1	Op Code Address + 1	1	Offset
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer - 1	0	Return Address (MSB)
		5	Branch Address	1	First Op Code of Subroutine

### 3.9 System Flowchart

A system flow of the HD63701V0 is given in Fig. 3-7-7.

4

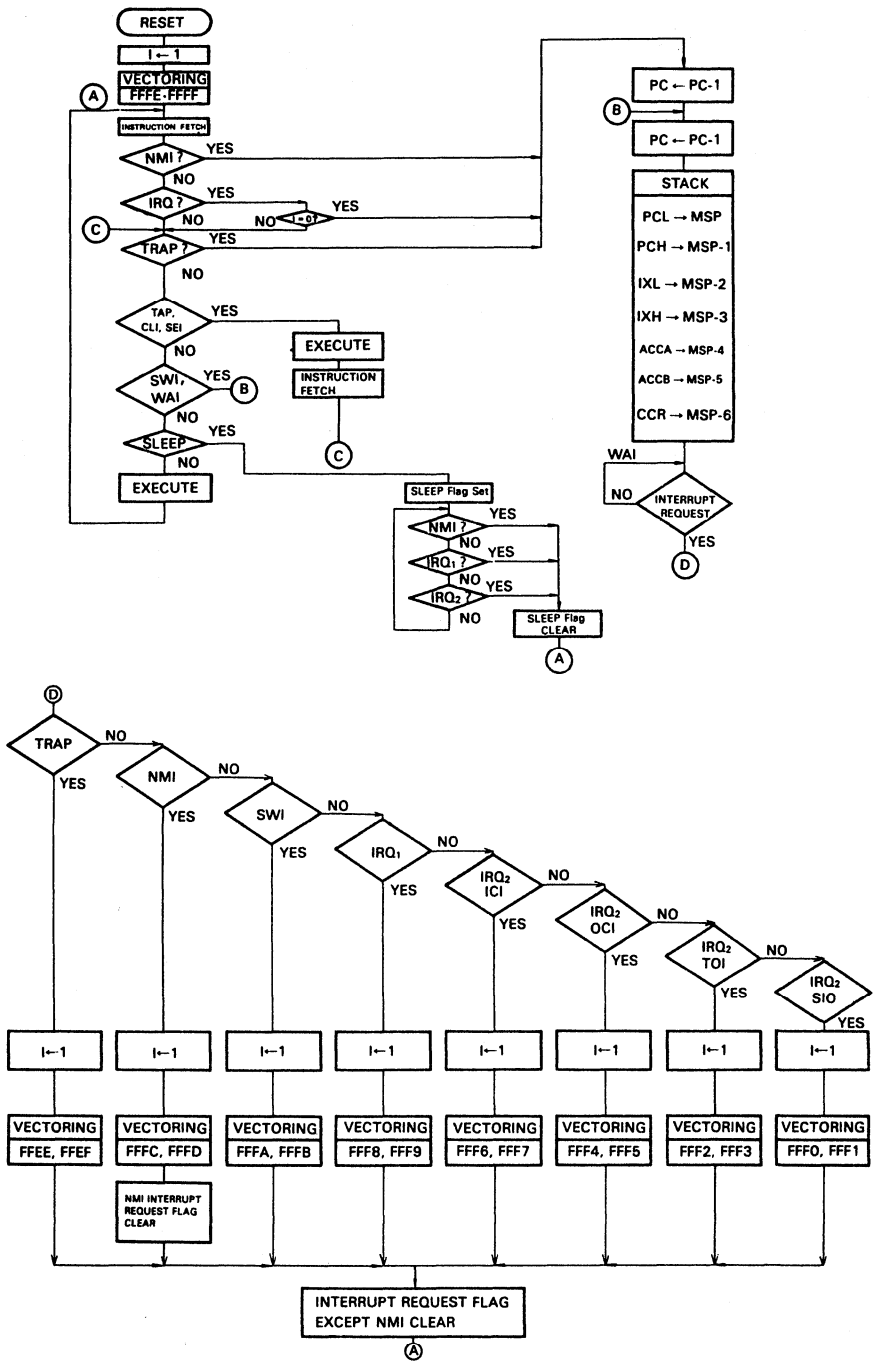


Fig. 3-7-7 HD63701V0 System Flowchart

### 3.10 PIN ARRANGEMENT AND PACKAGE INFORMATION

- HD63701VOC, HD637A01VOC, HD637B01VOC (DC-40)

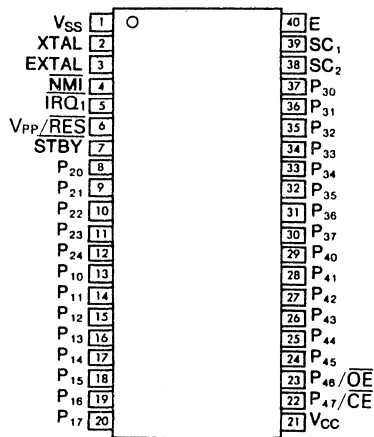


Fig. 3-10-1 Pin Arrangement

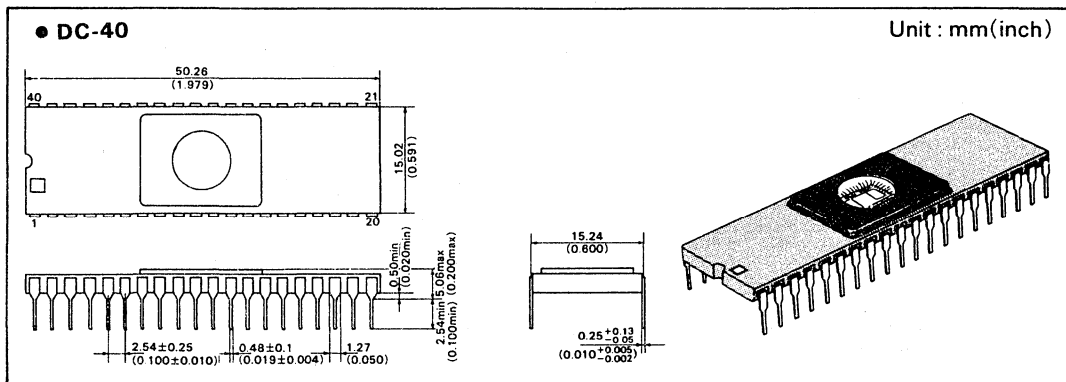


Fig. 3-10-2 Package Information

### 3.11 ELECTRICAL CHARACTERISTICS

#### ■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	$V_{CC}$	-0.3 ~ +7.0	V
Input Voltage	$V_{in}$	-0.3 ~ $V_{CC}+0.3$	V
Operating Temperature	$T_{opr}$	0 ~ +70	°C
Storage Temperature	$T_{stg}$	-55 ~ +125	°C

(NOTE) This product has protection circuits in input pin from high static electricity voltage and high electric field. But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend  $V_{in}, V_{out} : V_{SS} \leq (V_{in} \text{ or } V_{out}) \leq V_{CC}$ .

#### ■ MCU ELECTRICAL CHARACTERISTICS

- DC CHARACTERISTICS ( $V_{CC} = 5.0V \pm 10\%$ ,  $f = 0.1 \sim 2.0 \text{ MHz}$ ,  $V_{SS} = 0V$ ,  $T_a = 0 \sim +70^\circ\text{C}$ , unless otherwise noted.)

Item	Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$V_{IH}$		$V_{CC}-0.5$	-	$V_{CC}+0.3$	V
			$V_{CC} \times 0.7$	-		
			2.0	-		
Input "Low" Voltage	$V_{IL}$		-0.3	-	0.8	V
Input Leakage Current	$ I_{in} $	$V_{in} = 0.5 \sim V_{CC} - 0.5V$	-	-	10.0	$\mu\text{A}$
			-	-	1.0	
Three State (off-state) Leakage Current	$ I_{TS} $	$V_{in} = 0.5 \sim V_{CC} - 0.5V$	-	-	1.0	$\mu\text{A}$
Output "High" Voltage	$V_{OH}$		$I_{OH} = -200\mu\text{A}$	2.4	-	V
			$I_{OH} = -10\mu\text{A}$	$V_{CC}-0.7$	-	V
Output "Low" Voltage	$V_{OL}$		$I_{OL} = 1.6 \text{ mA}$	-	0.55	V
Input Capacitance	$C_{in}$	$V_{in} = 0V, f = 1.0 \text{ MHz}$ $T_a = 25^\circ\text{C}$	-	-	12.5	pF
Standby Current	$I_{CC}$	Non Operation	-	2.0	15.0	$\mu\text{A}$
Current Dissipation*	$I_{CC}$	Operating (f=1MHz)**	-	5.0	10.0	mA
		Sleeping (f=1MHz)**	-	1.0	2.0	
RAM Stand-by Voltage	$V_{RAM}$		2.0	-	-	V

\*  $V_{IH} \text{ min} = V_{CC} - 0.8V, V_{IL} \text{ max} = 0.8V$  A11 output pins have no load.

\*\* Current Dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ. or max. values about Current Dissipations at  $\chi$  MHz operation are decided according to the following formulas.

typ. value (f =  $\chi$  MHz) = typ. value (f = 1MHz)  $\times \chi$   
 max. value (f =  $\chi$  MHz) = max. value (f = 1MHz)  $\times \chi$   
 (both the sleeping and operating)

● AC CHARACTERISTICS (V<sub>CC</sub> = 5.0V ± 10% ... f=0.1~2.0 MHz, V<sub>SS</sub> = 0V, Ta = 0~+70°C, unless otherwise noted.)

**BUS TIMING**

Item	Symbol	Test Condition	HD63701V0		HD637A01V0		HD637B01V0		Unit	
			min	max	min	max	min	max		
Cycle Time	t <sub>cyc</sub>	Fig. 6-1, Fig. 6-2	1	10	0.666	10	0.5	10	μs	
Address Strobe Pulse Width "High"	PWASH*		220	—	150	—	110	—	ns	
Address Strobe Rise Time	t <sub>ASr</sub>		—	25	—	25	—	25	ns	
Address Strobe Fall Time	t <sub>ASf</sub>		—	25	—	25	—	25	ns	
Address Strobe Delay Time	t <sub>ASD</sub> *		60	—	40	—	20	—	ns	
Enable Rise Time	t <sub>Er</sub>		—	20	—	20	—	20	ns	
Enable Fall Time	t <sub>Ef</sub>		—	20	—	20	—	20	ns	
Enable Pulse Width "High" Level	PWEH*		450	—	300	—	220	—	ns	
Enable Pulse Width "Low" Level	PWEL*		450	—	300	—	220	—	ns	
Address Strobe to Enable Delay Time	t <sub>ASED</sub> *		60	—	40	—	20	—	ns	
Address Delay Time	t <sub>AD1</sub> *		—	250	—	190	—	160	ns	
	t <sub>AD2</sub> *		—	250	—	190	—	160	ns	
Address Delay Time for Latch	t <sub>ADL</sub> *		—	250	—	190	—	160	ns	
Data Set-up Time	Write		t <sub>DSW</sub> *	230	—	150	—	100	—	ns
	Read		t <sub>DSR</sub>	80	—	60	—	50	—	ns
Data Hold Time	Read		t <sub>HR</sub>	0	—	0	—	0	—	ns
	Write		t <sub>HW</sub> *	60	—	40	—	30	—	ns
Address Set-up Time for Latch	t <sub>ASL</sub> *		60	—	40	—	20	—	ns	
Address Hold Time for Latch	t <sub>AHL</sub>		30	—	20	—	20	—	ns	
Address Hold Time	t <sub>AH</sub> *		60	—	40	—	30	—	ns	
A <sub>0</sub> ~ A <sub>7</sub> Set-up Time Before E	t <sub>ASM</sub> *	200	—	110	—	60	—	ns		
Peripheral Read Access Time	Non-Multiplexed Bus	(t <sub>ACCN</sub> )*	—	650	—	395	—	270	ns	
	Multiplexed Bus	(t <sub>ACCM</sub> )*	—	650	—	395	—	270	ns	
Oscillator Stabilization Time	t <sub>RC</sub>	Fig. 2-7-1,	20	—	20	—	20	—	ms	
Processor Control Set-up Time	t <sub>PCS</sub>	Fig. 2-8-1	200	—	200	—	200	—	ns	

\*These timings change depending on the t<sub>cyc</sub>. The values in the table are those when the t<sub>cyc</sub> is minimum.

**PERIPHERAL PORT TIMING**

Item	Symbol	Test Condition	HD63701V0		HD637A01V0		HD637B01V0		Unit	
			min	max	min	max	min	max		
Peripheral Data Set-up Time	Port 1, 2, 3, 4	t <sub>PDUS</sub>	Fig. 6-3	200	—	200	—	200	—	ns
Peripheral Data Hold Time	Port 1, 2, 3, 4	t <sub>PDH</sub>	Fig. 6-3	200	—	200	—	200	—	ns
Delay Time, Enable Positive Transition to OS3 Negative Transition		t <sub>OSD1</sub>	Fig. 6-5	—	300	—	300	—	300	ns
Delay Time, Enable Positive Transition to OS3 Positive Transition		t <sub>OSD2</sub>	Fig. 6-5	—	300	—	300	—	300	ns
Delay Time, Enable Negative Transition to Peripheral Data Valid	Port 1, 2*, 3, 4	t <sub>PWD</sub>	Fig. 6-4	—	300	—	300	—	300	ns
Input Strobe Pulse Width		t <sub>PWIS</sub>	Fig. 6-6	200	—	200	—	200	—	ns
Input Strobe Rise Time		t <sub>ISr</sub>	Fig. 6-6	—	50	—	50	—	50	ns
Input Strobe Fall Time		t <sub>ISf</sub>	Fig. 6-6	—	50	—	50	—	50	ns
Input Data Hold Time	Port 3	t <sub>IH</sub>	Fig. 6-6	150	—	150	—	150	—	ns
Input Data Setup Time	Port 3	t <sub>IS</sub>	Fig. 6-6	0	—	0	—	0	—	ns

\* Except P21





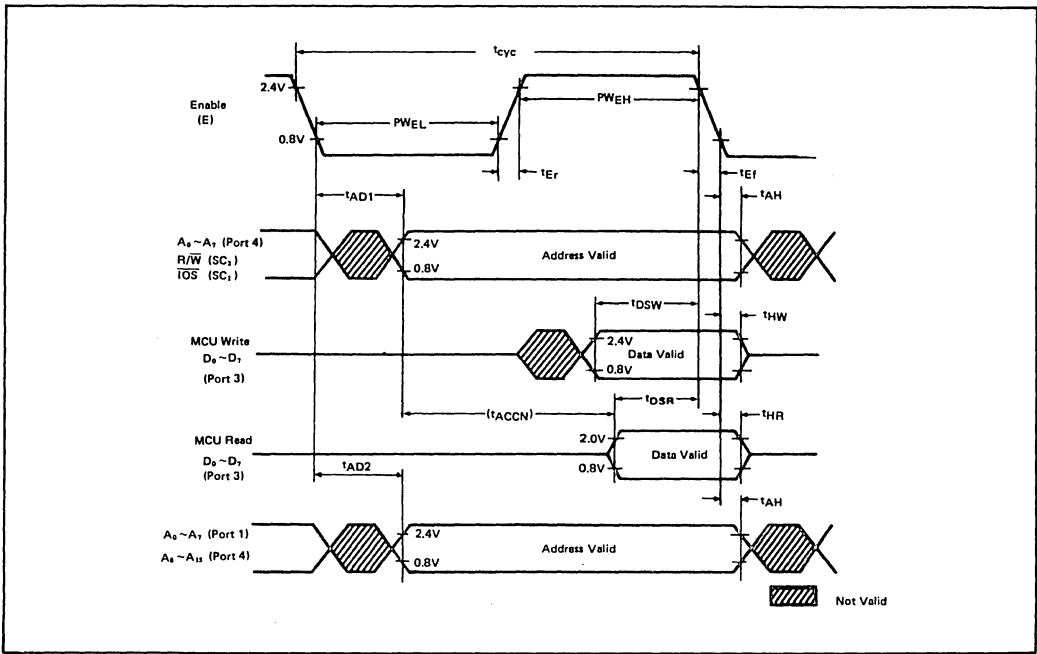


Fig. 3-11-2 Expanded Non-Multiplexed Bus Timing

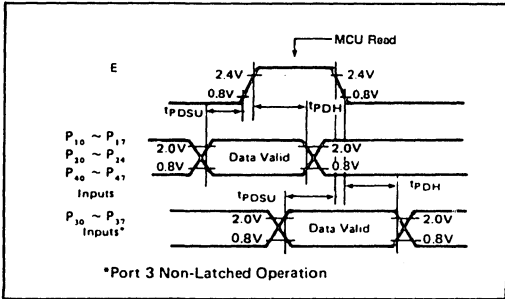


Fig. 3-11-3 Port Data Set-up and Hold Times (MCU Read)

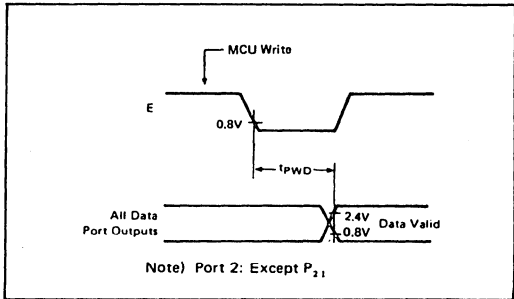


Fig. 3-11-4 Port Data Delay Times (MCU Write)

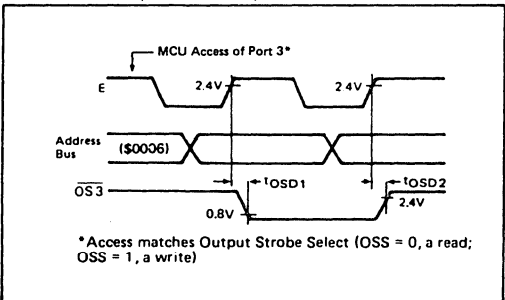


Fig. 3-11-5 Port 3 Output Strobe Timing (Single Chip Mode)

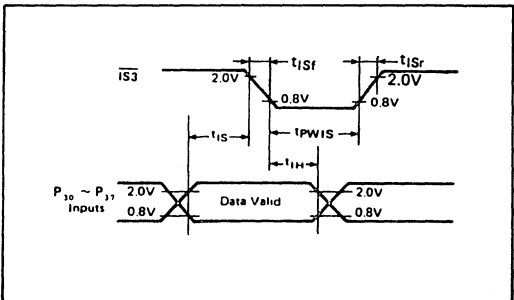


Fig. 3-11-6 Port 3 Latch Timing (Single Chip Mode)

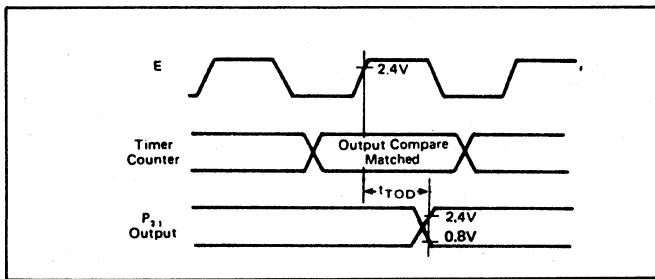


Fig. 3-11-7 Timer Output Timing

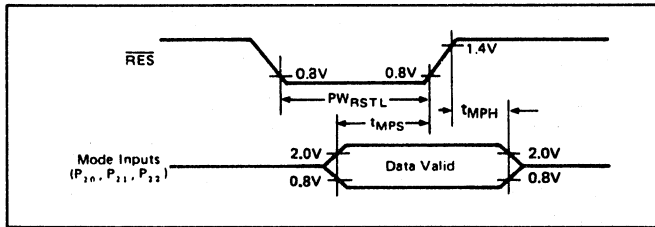


Fig. 3-11-8 Mode Programming Timing

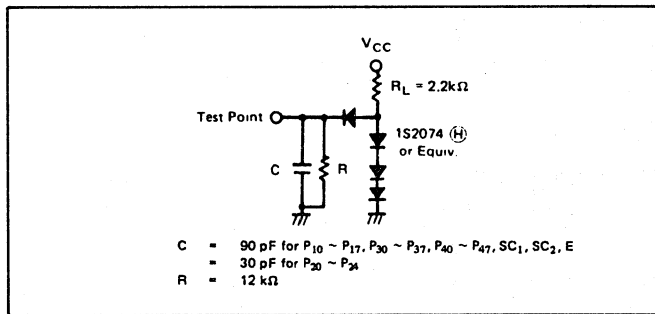


Fig. 3-11-9 Bus Timing Test Loads (TTL Load)

■ PROGRAMMING ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS ( $V_{CC} = 6.0V \pm 0.25V$ ,  $V_{PP} = 12.5V \pm 0.3V$ ,  $V_{SS} = 0V$ ,  $T_a = 25^\circ C \pm 5^\circ C$  unless otherwise noted.)

Item	Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$O_0 \sim O_7, A_0 \sim A_{14}, \overline{OE}, \overline{CE}$	$V_{IH}$	2.2	—	$V_{CC} + 0.3$	V
Input "Low" Voltage	$O_0 \sim O_7, A_0 \sim A_{14}, \overline{OE}, \overline{CE}$	$V_{IL}$	-0.3	—	0.8	V
Output "High" Voltage	$O_0 \sim O_7$	$V_{OH}$	$I_{OH} = -200\mu A$	—	—	V
Output "Low" Voltage	$O_0 \sim O_7$	$V_{OL}$	$I_{OL} = 1.6mA$	—	0.45	V
Input Leakage Current	$O_0 \sim O_7, A_0 \sim A_{14}, \overline{OE}, \overline{CE}$	$I_{LI}$	$V_{IN} = 5.25V/0.5V$	—	2	$\mu A$
$V_{CC}$ Current		$I_{CC}$	—	—	30	mA
$V_{PP}$ Current		$I_{PP}$	—	—	30	mA

● AC CHARACTERISTICS ( $V_{CC} = 6.0V \pm 0.25V$ ,  $V_{PP} = 12.5V \pm 0.3V$ ,  $V_{SS} = 0V$ ,  $T_a = 25^\circ C \pm 5^\circ C$  unless otherwise noted.)

Item	Symbol	Test condition	min	ty	max	Unit
Address Set-up Time	$t_{AS}$	Fig.6-10*1	2	—	—	$\mu s$
$\overline{CE}$ Set-up Time	$t_{OES}$		2	—	—	$\mu s$
Data Set-up Time	$t_{DS}$		2	—	—	$\mu s$
Address Hold Time	$t_{AH}$		0	—	—	$\mu s$
Data Hold Time	$t_{DH}$		2	—	—	$\mu s$
Data Output Disable Time	$t_{DF}$		—	—	130	ns
$V_{PP}$ Set-up Time	$t_{VPS}$		2	—	—	$\mu s$
Program Pulse Width (High Speed Writing)	$t_{PW}$		0.95	1.0	1.05	ms
Program Pulse Width	$t_{OPW}$		2.85	—	78.75	ms
$V_{CC}$ Set-up Time	$t_{VCS}$		2	—	—	$\mu s$
Data Output Delay Time	$t_{OE}$		0	—	150	ns

\*1 Input Pulse Level = 0.8~2.2V  
 Input Rise Time/Fall Time  $\leq 20ns$ .  
 Tuning Reference Level Input: 1.0V, 2.0V.  
 Output: 0.8V, 2.0V.

4

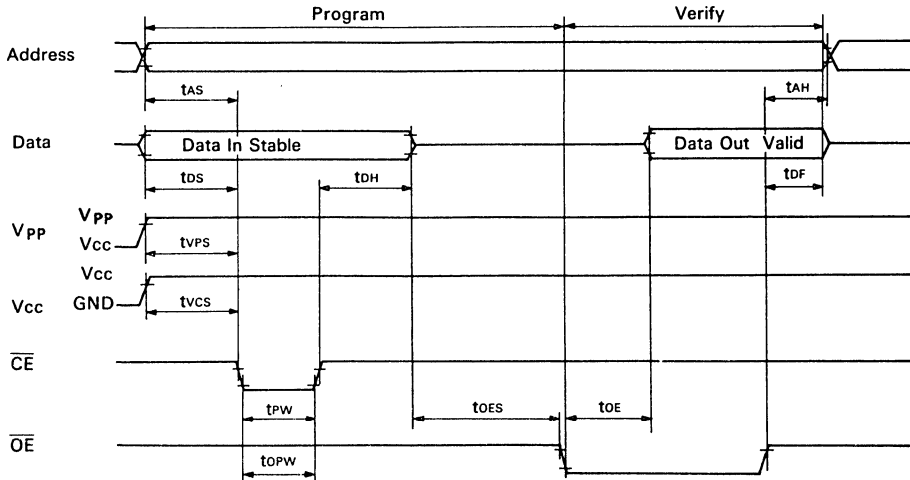


Fig. 3-11-10 PROM Program/Verify Timing

## 4. APPLICATIONS

### 4.1 Use of External Expanded Mode

The HD63701V0 supports four operation modes 1, 2, 5 and 6 as external expanded modes. Usage of these modes is detailed in the following paragraphs.

#### (1) Non-multiplexed modes

##### (a) Mode 1 (New Mode)

In this mode, Port 3 functions as data bus, Port 1 as lower address bus ( $A_0 - A_7$ ), and Port 4 as upper address bus ( $A_8 - A_{15}$ ). Since 16-bit addresses are sent out in parallel, the HD63701V0 can access up to 65k memory space with no address latch externally in this mode.

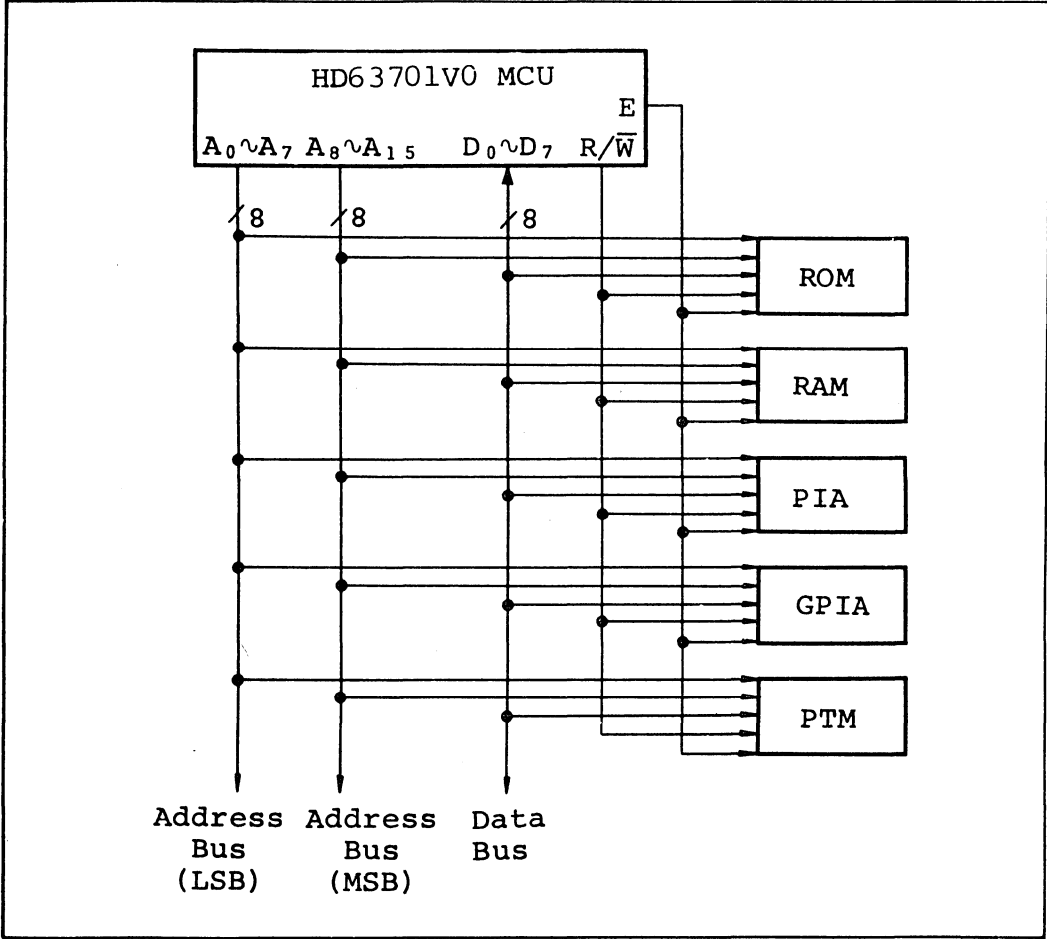


Fig. 4-1-1 HD63701V0, Mode 1

If an internal memory address and an external memory address are overlapped, the memory address can be accessed as follows. When writing, the same data can be written into both internal and external memories simultaneously.

When reading, only internal memory can be read while the external memory can not be read.

The same operations can be applied to modes 2, 5, and 6.

In mode 1, external memory addresses range from \$00FF to \$FFFF and so internal mask ROM located in \$F000 through \$FFFF can not be accessed.

After reset, Port 1 functions as lower address bus ( $A_0 - A_7$ ), Port 4 is a upper address bus ( $A_8 - A_{15}$ ).

(b) Mode 5 (Equivalent to Mode 5 of HD6801V)

Port 3 works as data bus; and Port 4 as address bus ( $A_0 - A_7$ ) or input pin by programming the DDR. In this mode, pin 39 provides the result of the following decoding:

$$\overline{A_{15}} \cdot \overline{A_{14}} \cdot \overline{A_{13}} \cdot \overline{A_{12}} \cdot \overline{A_{11}} \cdot \overline{A_{10}} \cdot \overline{A_9} \cdot A_8$$

This output signal may be used as a chip select or chip enable signal permitting to access an external memory up to 256 byte locations (\$0100 - \$01FF). The pin function of Port 4 can be changed from an address line to an input port if the system does not need all of the 8 address lines by writing zero into the corresponding bit of Port 4 DDR.

An example of connection with PIA (HD6821, HD6321) is shown in Fig. 4-1-2.

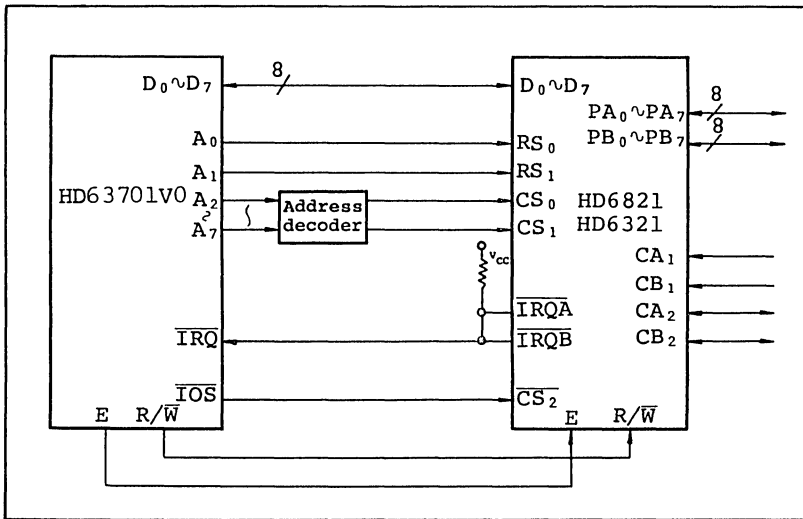


Fig. 4-1-2 Connection of HD63701V0 with PIA

(2) Multiplex Modes (Modes 2 & 6)

Any multiplex mode provides a time multiplexed address and data on port 3. Therefore, an address latch is required externally to access external devices. As (Pin 39) signal is used for an address latch strobe. An example of HD63701V0 and CMOS latch interface is shown in Fig. 4-1-3.

It should be noted, however, that the HD63701V0 can not operate at more than 500 kHz when interfaced with the latch.



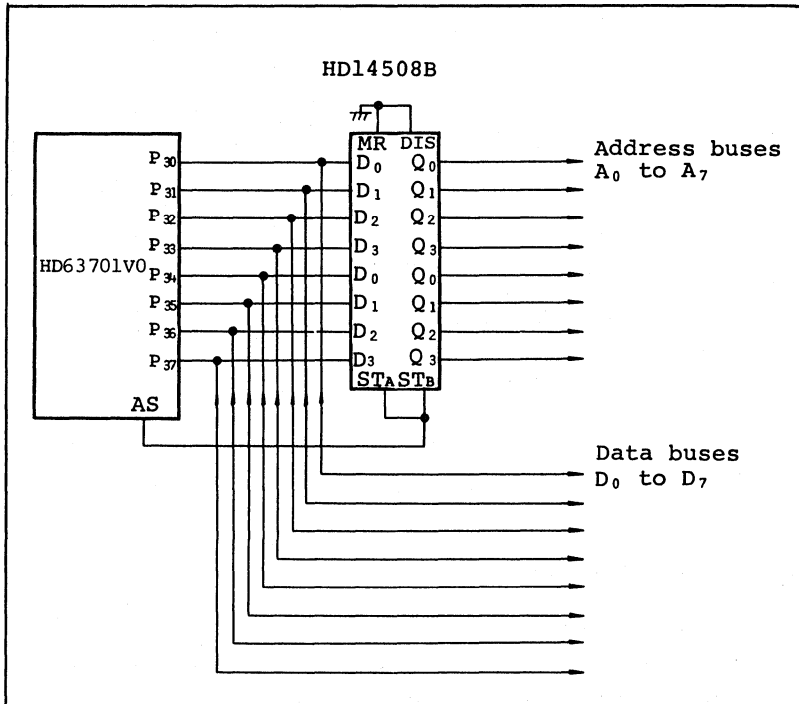


Fig. 4-1-3 CMOS Latch

For high-speed operation, 74LS373 or high speed CMOS latch (74HC373) is desirable to minimize the delay time.

(a) Mode 2 (Equivalent to HD6801V Mode 2)

In this mode, the internal mask ROM (\$F000 through \$FFFF) is disabled and external memory becomes valid instead. Port 4 functions as the upper address bus.

(b) Mode 6 (Equivalent to HD6801V Mode 6)

In this mode, the internal mask ROM is enabled. Port 4 functions as address bus ( $A_8 - A_{15}$ ). Since Port 4 becomes input mode after reset, the DDR must be programmed to "1" to use the port as address bus.

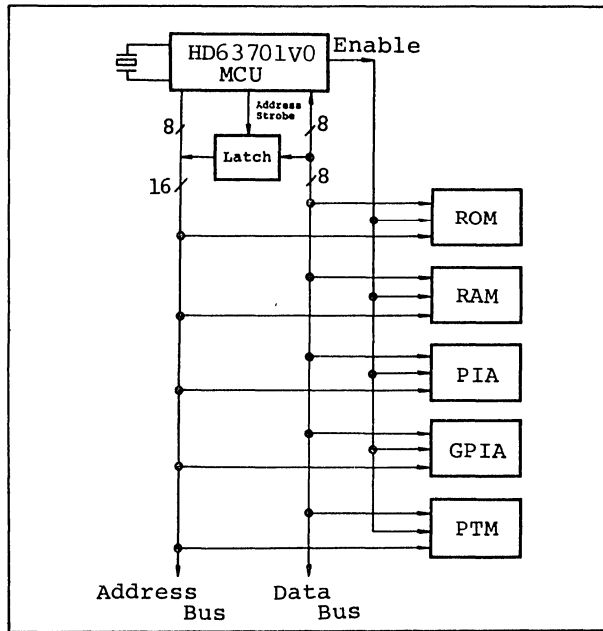


Fig. 4-1-4 HD63701V0 MCU Expanded Multiplexed Mode

#### 4.2 Standby Mode

Bringing  $\overline{STBY}$  "Low", the HD63701V0 goes into the Standby mode. In this mode, the CPU becomes reset and all clocks of the HD63701V0 become inactive.

The contents of the internal RAM is retained as long as  $V_{CC}$  is supplied ( $V_{CC} \geq 2V$ ). Under Standby Mode, memory back-up is possible with only a few  $\mu A$  of leakage current. When  $\overline{STBY}$  is brought "High", the MCU exits from Standby Mode. When "1" level is detected at  $\overline{STBY}$  pin, a clock generator begins to oscillate and the internal reset condition is released. At this time,  $\overline{RES}$  signal must be held "Low" for at least OSC stabilization time ( $t_{RC}$ ) before the CPU operation restarts. Otherwise, the normal operation is not guaranteed.

Fig. 4-2-1.

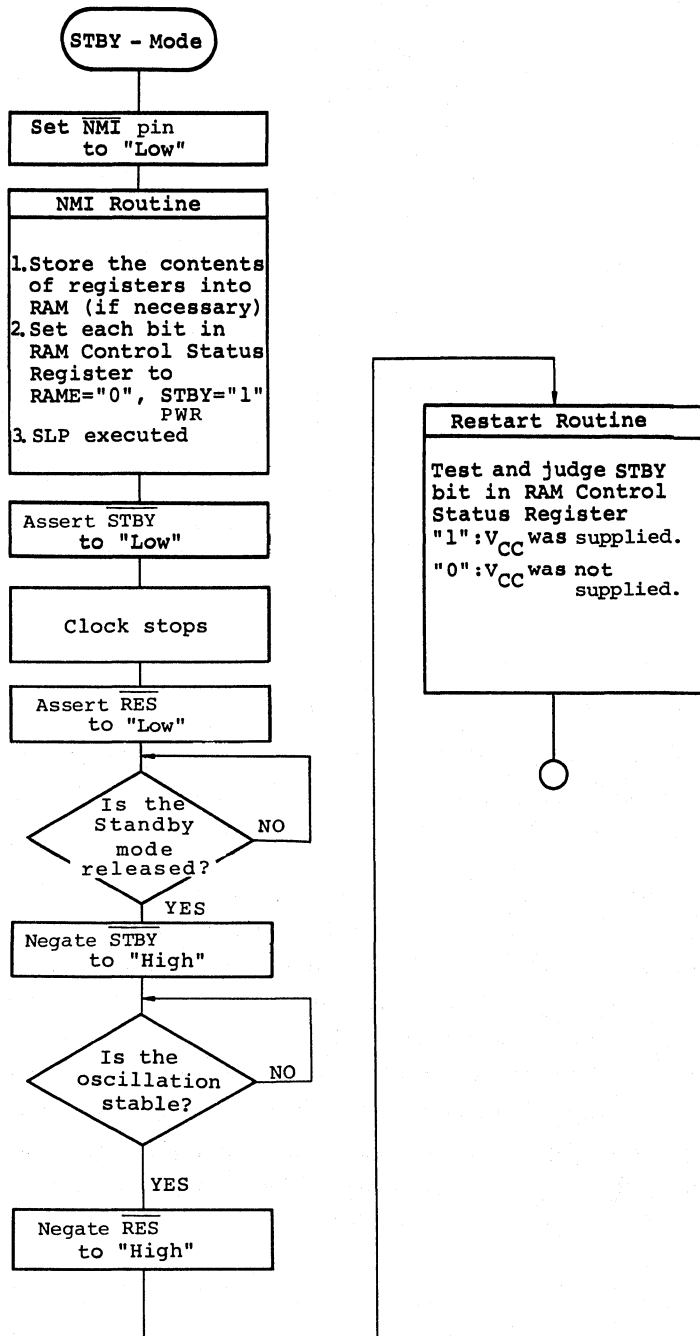


Fig. 4-2-1 Flowchart of Standby Mode Application

The timing relationship shown in Fig. 4-2-2 must be satisfied.

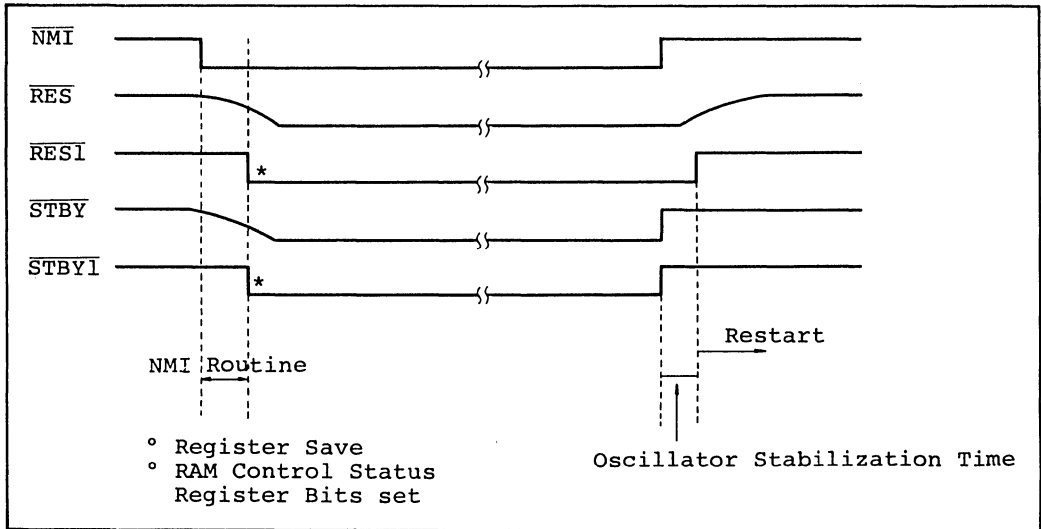


Fig. 4-2-2 Timing Chart of Each Signal

\* Either  $\overline{\text{RES1}}$  or  $\overline{\text{STBY1}}$  can become "0" level as long as the execution time of NMI routine is guaranteed.

Fig. 4-2-3 shows an example of a circuit to implement the timing sequence shown in the Fig. 4-2-2.

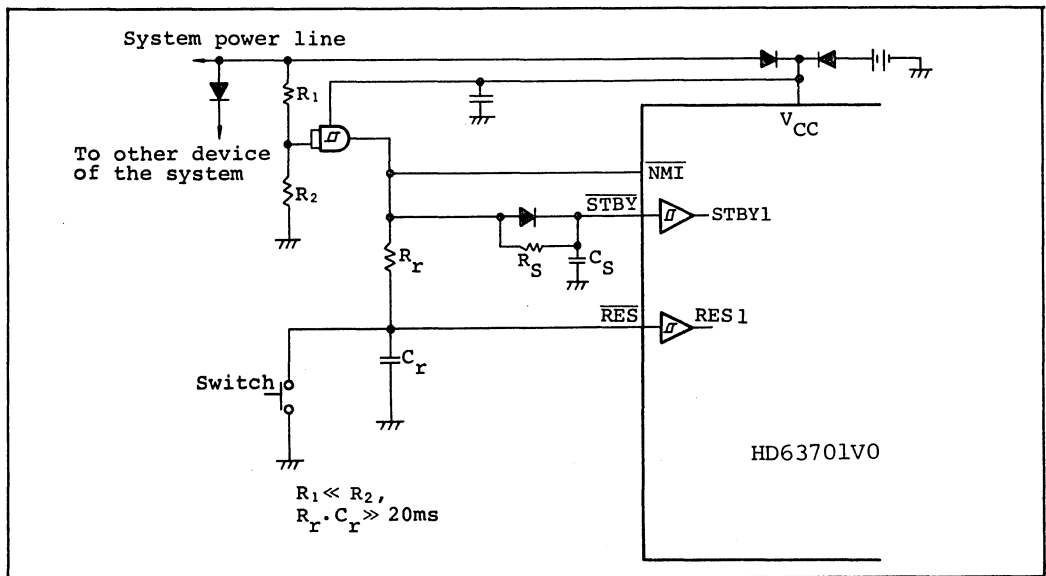


Fig. 4-2-3 Example of Circuit Diagram for a Standby Operation

<Precaution for using Standby Power bit>

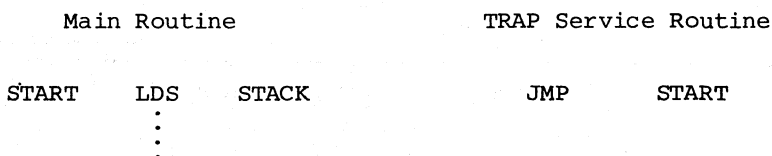
The Standby power bit in the RAM control status register detects that V<sub>CC</sub> is supplied or not. When the V<sub>CC</sub> rise time is equal or less than 100μs, the Standby power bit may not be cleared. To avoid this, the V<sub>CC</sub> rise time should be more than 100μs, for example, by using the larger bypass capacitor.

#### 4.3 Address Trap, OP-Code Trap Application

The HD63701V0 facilitates two trap functions, the op-code trap and the address trap, to protect the HD63701V0 to proceed an erroneous operation. The op-code trap is generated when an illegal or undefined op-code is fetched. Therefore, when undefined codes listed below are fetched, a trap is caused and the HD63701V0 avoids further erroneous operation. The priority level of the op-code trap interrupt is next to the RESET. Undefined codes of the HD63701V0 are: \$00, \$02, \$03, \$12, \$13, \$14, \$15, \$1C, \$1D, \$1E, \$1F, \$41, \$42, \$45, \$4B, \$4E, \$51, \$52, \$55, \$5B, \$5E, \$87, \$8F, \$C7, \$CD and \$CF.

The address trap is generated when an op-code is fetched from the memory area shown in Table 2-3-1. It should be noted, however, this function works only under op-code fetch (not for data access). Under the support of error processing program in trap service routine, the user can realize the proper error processing for the application system. An example of restarting from the trap service routine is shown below. If RTI instruction is executed at the end of the trap service routine, the program is restarted at the location where the trap interrupt is generated and then another trap may occur again.

So, special care must be taken when a programmer uses trap function.



#### 4.4 Slow Memory Interface

Here described is the example of clock width control circuit and its timing chart, where E-clock high time is extended to assure enough access time.

The expanded enable high pulse width ( $PW'_{EH}$ ), which is implemented by using the circuit shown below, is calculated as follows:

$$PW'_{EH} = (n+1) \cdot t_{4\phi cyc} + PW_{EH} \leq 10 - PW_{EL} (\mu s)$$

where  $n$  : Integer part of  $[t_W/t_{4\phi cyc}]$

$t_{4\phi cyc}$  :  $4\phi$  clock cycle time ( $\mu s$ )

$PW_{EH}$  : Enable High pulse width ( $\mu s$ )

$PW_{EL}$  : Enable Low pulse width ( $\mu s$ )

$t_W$  : approx.  $0.45 \cdot C_{ext} (pF) \cdot R_{ext} (k\Omega) \times 10^{-3} (\mu s)$

The circuit shown is for a reference purpose. It is assumed that users will refine it for actual design.

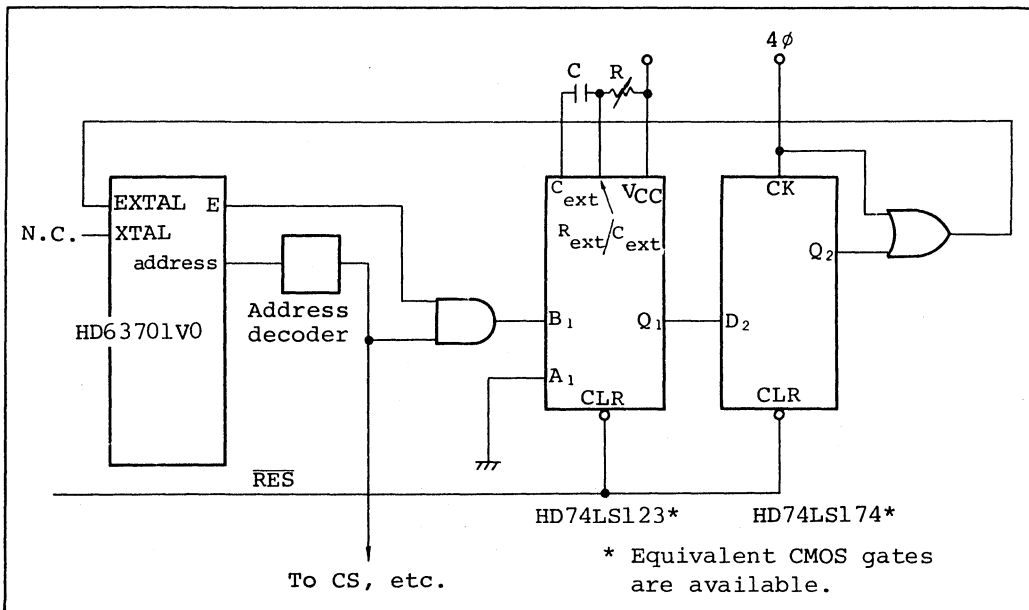


Fig. 4-4-1 Clock Control Circuit

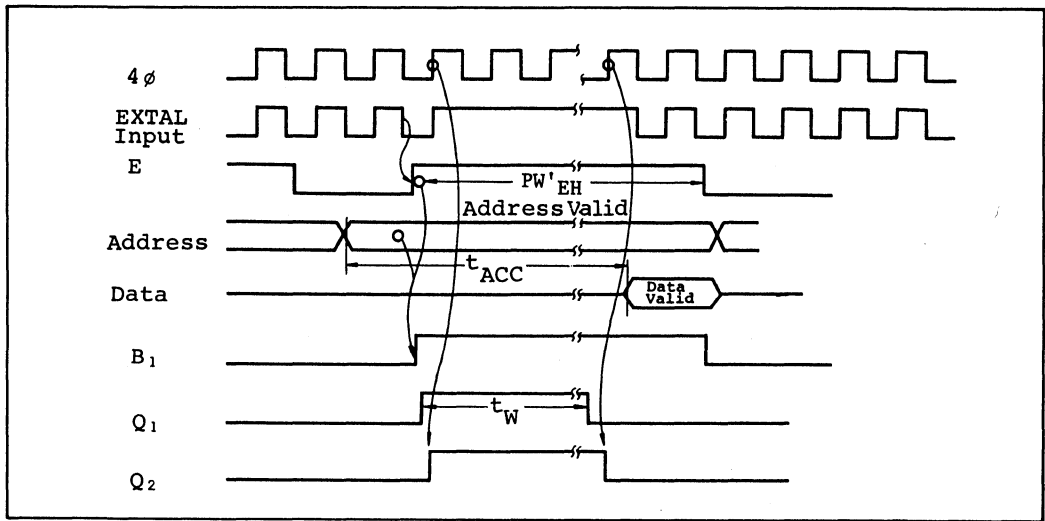


Fig. 4-4-2 Clock Timing

#### 4.5 Interface to HN61256

An examples of the interface to a slow memory device, HN61256 (CMOS 256k bit Mask programmable ROM), is described here.

The AC characteristics and the access timing of the HN61256 is shown in Fig. 3-5-1.

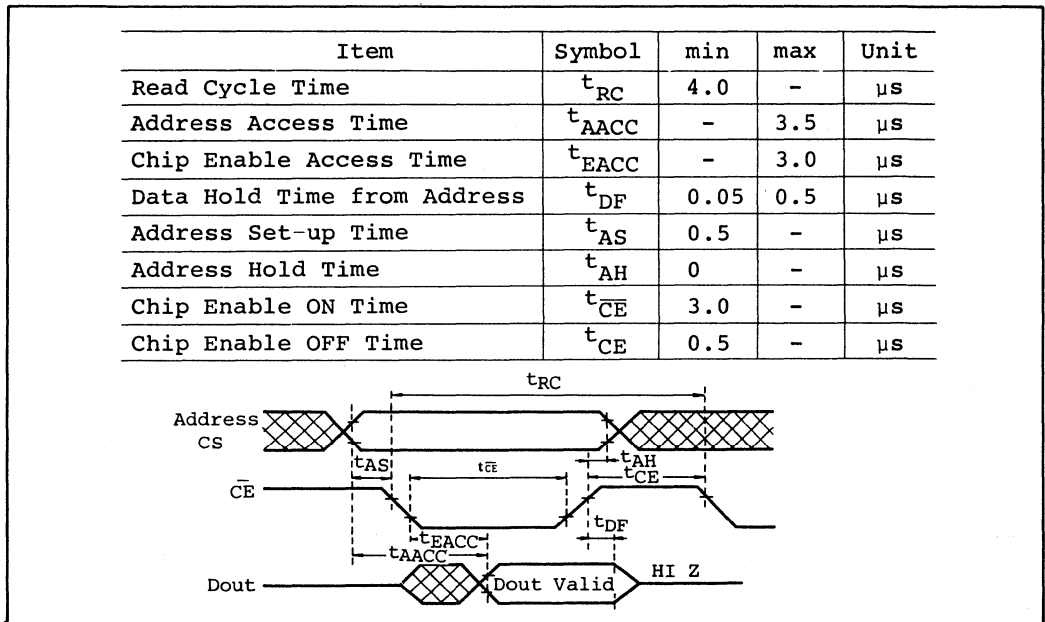


Fig. 4-5-1 AC Characteristics and Access Timing of HN61256



#### 4.5.1 Use of Two Latches

Two HD14508Bs are used in order to latch 16 bit address. An example of the program and its access timing are shown in Table 4-5-1 and Fig. 4-5-3, respectively.

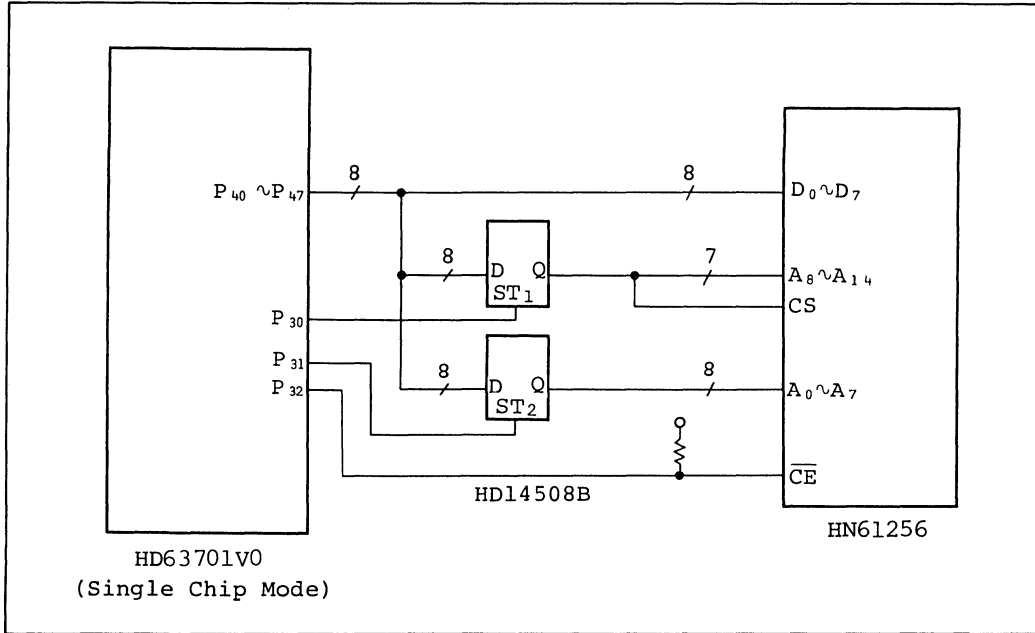


Fig. 4-5-2 HD63701V0 and HN61256 Interface by Two Latches



Table 4-5-1 An Example of the Program

<u>Mnemonic</u>		<u>Cycles</u>	
LDAA	#\$FF	2	
STAA	P4DDR	3	Port 4 is the output port.
LDD	#\$ADDRS1	3	Data that is the address's upper 8 bits including CS signal and changes ST1 into high and ST2 into low.
STD	PORT3	4	Enables ST1, disables ST2, and moves the address's upper 8 bits into Port 4.
LDD	#\$ADDRS2	3	Data that is the address's lower 8 bits and changes ST1 into low and ST2 into high.
STD	PORT3	4	Disables ST1, enables ST2, and stores the address's lower 8 bits into Port 4.
LDAA	#IMM1	2	Data that changes ST1 and ST2 into low and $\overline{CE}$ into active.
STAA	PORT3	3	Disables ST1 and ST2 and enables $\overline{CE}$ .
LDAB	#\$00	2	
STAB	P4DDR	3	Port 4 becomes the input port.
LDAA	PORT4		Reads data.

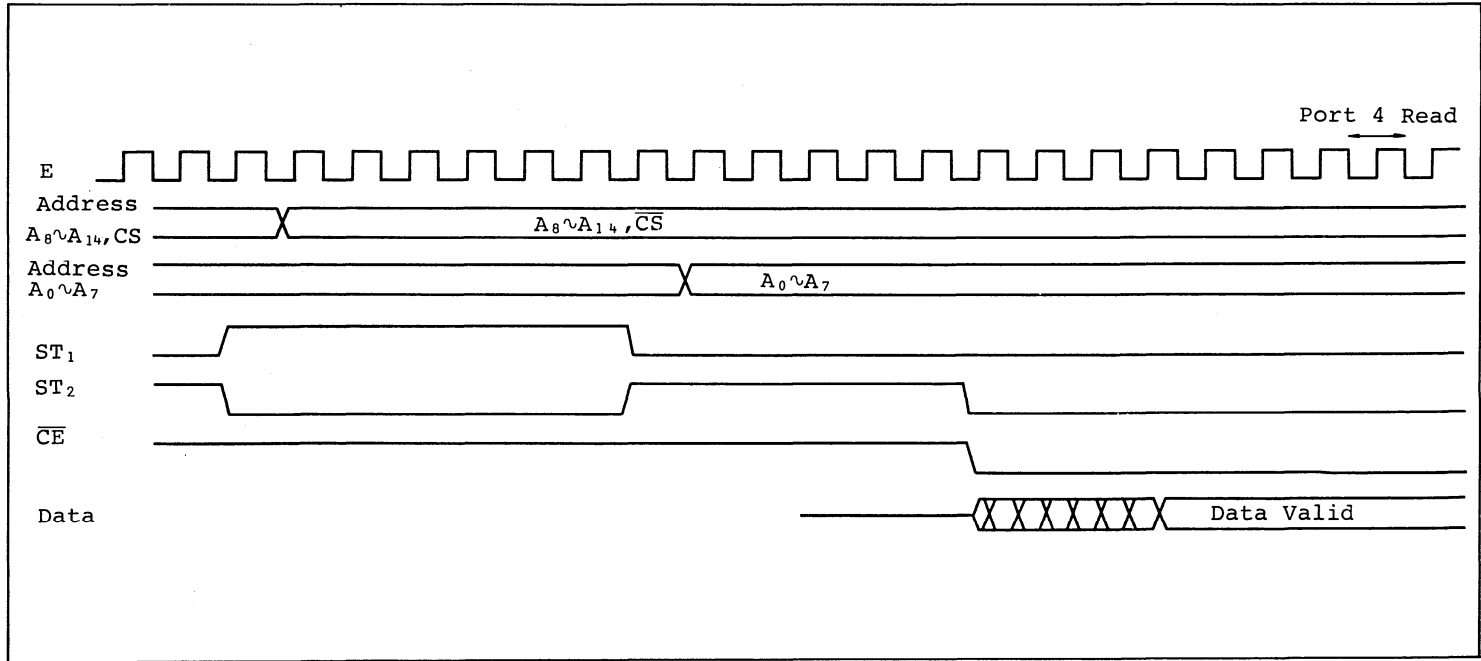


Fig. 4-5-3 Access Timing

### 4.5.2 Extending E clock

Fig. 4-5-4 is an example circuitry to extend the E clock.

The operation mode of the HD63701V0 is in mode 6; and the clock frequency of  $4\phi$  is 4 MHz.

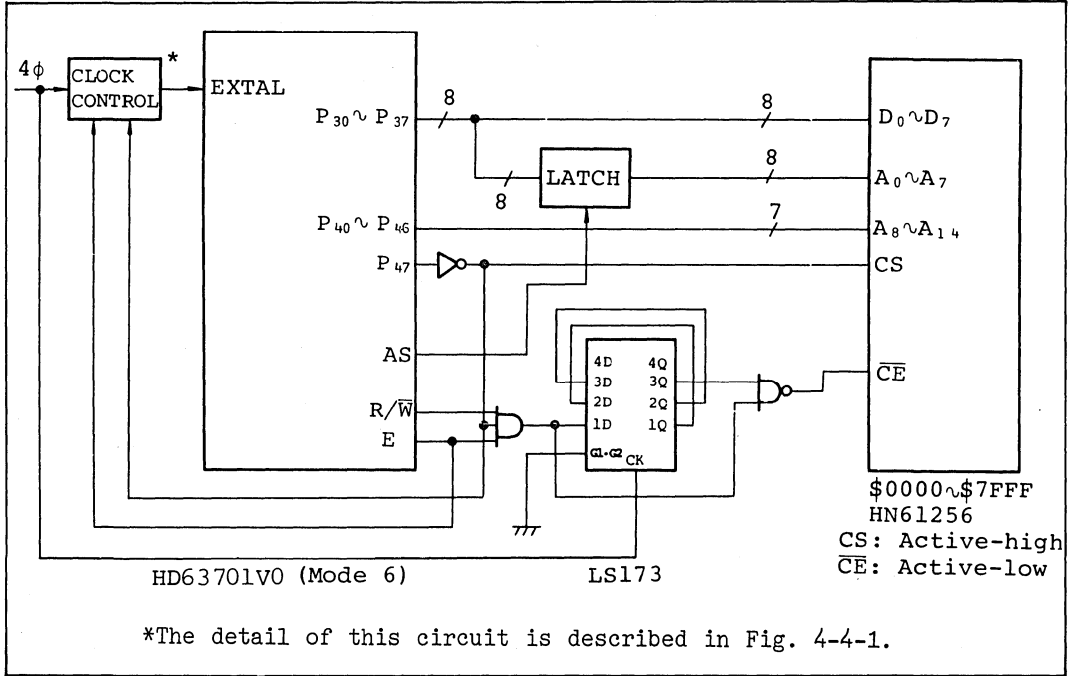


Fig. 4-5-4 HD63701V0 and HN61256 Interface by extending E clock cycle

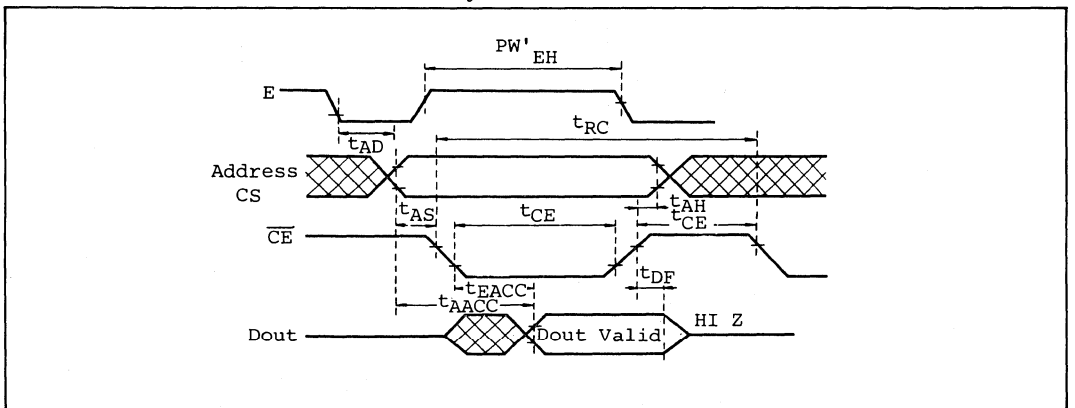


Fig. 4-5-5 HN61256 Read Timing

In this example,  $PW_{EH}$  of which timing is extended by using the clock control circuit (Fig. 4-4-1) must be at least 4  $\mu$ s. The LS173 is to assure enough address set up time ( $t_{AS}$ ) of HN61256.

#### 4.6 Interface to the Realtime Clock (HD146818)

The HD146818 (realtime clock + RAM : RTC) is a CMOS micro-computer peripheral LSI that incorporates the clock and calendar functions to compute year, month, day, day of week, and time. When this HD146818 is interfaced to the HD63701V0, this LSI provides a real time clock information to be displayed.

In addition, the HD146818 can be also utilized as a system interval timer and a square waves generator. An example of the HD146818 and the HD63701V0 interface is shown in Fig 4-6-1. It can be interfaced under the expanded multiplexed mode (mode 6) of the HD63701V0.

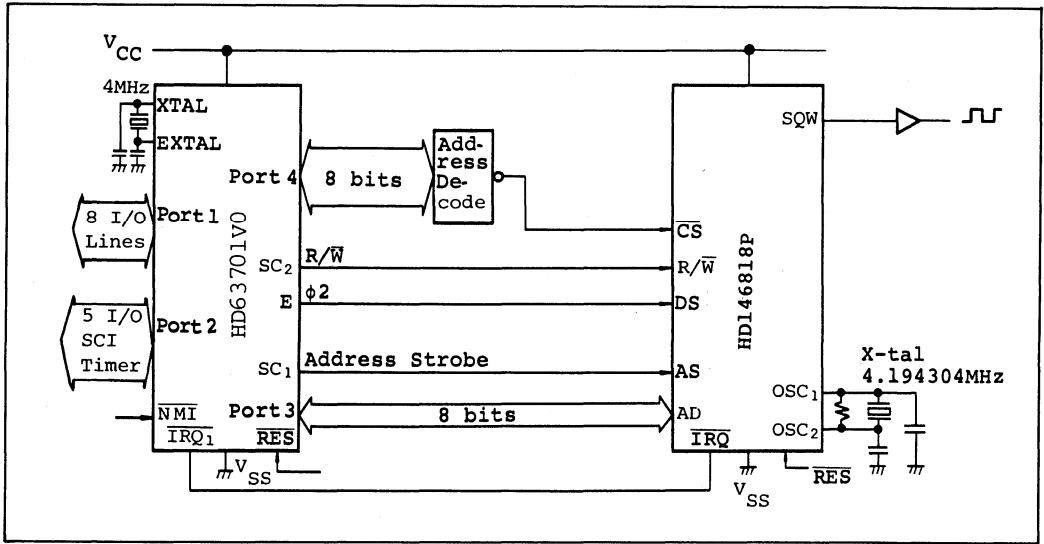


Fig. 4-6-1 HD63701V0 and HD146818P Interface  
(MCU Expanded Multiplexed Mode)

The calendar and clock display functions of HD146818 are shown below.

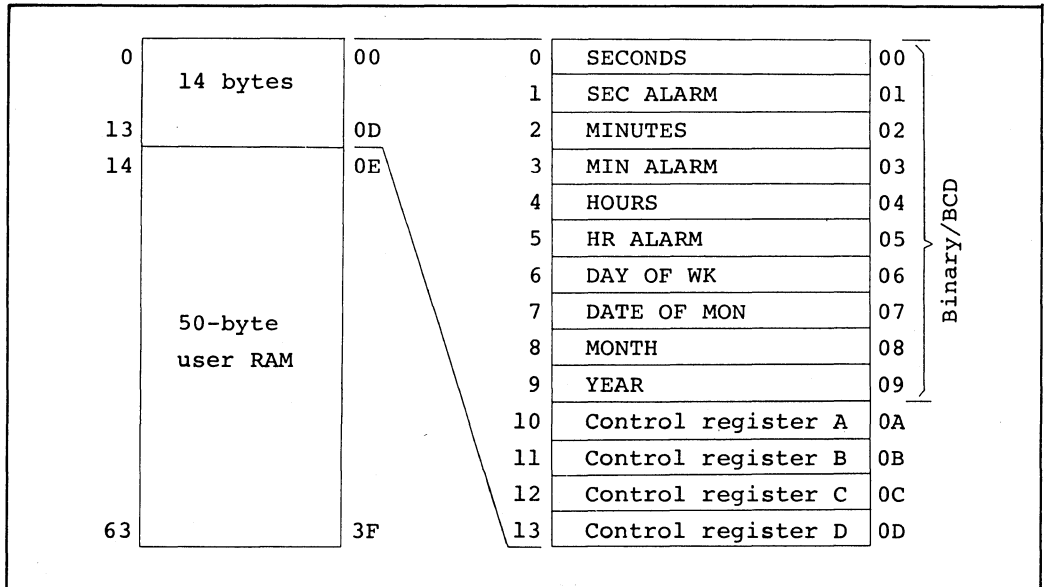


Fig. 4-6-2 HD146818 On-chip RAM Address Map

Table 4-6-1 HD146818 Time, Calendar, &amp; Alarm Data Display

Address	Function		Data range (Decimal)	Data range (Hexadecimal)	
				Binary data mode	BCD data mode
0	SECONDS		0 to 59	00 to 3B	00 to 59
1	SECONDS ALARM		0 to 59	00 to 3B	00 to 59
2	MINUTES		0 to 59	00 to 3B	00 to 59
3	MINUTES ALARM		0 to 59	00 to 3B	00 to 59
4	HOURS	12-hour mode	1 to 12	01 to 0C/ 81 to 8C*	01 to 12/ 81 to 92*
		24-hour mode	0 to 23	00 to 17	00 to 23
5	HOURS ALARM	12-hour mode	1 to 12	01 to 0C/ 81 to 8C*	01 to 12/ 81 to 92*
		24-hour mode	0 to 23	00 to 17	00 to 23
6	DAY OF THE WEEK		1 to 7**	01 to 07	01 to 07
7	DAY OF THE MONTH		1 to 31	01 to 1F	01 to 31
8	MONTH		1 to 12	01 to 0C	01 to 12
9	YEAR		0 to 99***	00 to 63	00 to 99

## [Notes]

- \*: The most significant bit differentiates between AM and PM. That is, 0 = AM and 1 = PM.
- \*\* : 1 = Sunday, 2 = Monday, 3 = Tuesday, 4 = Wednesday, 5 = Thursday, 6 = Friday, and 7 = Saturday
- \*\*\*: This takes the lower two digits of the calendar year.

The information of the calendar and the time are stored on the on-chip RAM and updated every second. The on-chip RAM includes not only the display RAM but also 50-byte user RAM which stores data necessary for the system.

The HD63701V0 gets the calendar and time information by reading the on-chip RAM of the HD146818. The HD146818 generates three interrupts, update interrupt, alarm interrupt, and periodic interrupt, to the HD63701V0. The HD63701V0 can service proper routine for the application system by accepting the HD146818 three interrupts.

Such a combination of the HD63701V0 and the HD146818 easily implements a compact real time system with reduced power dissipation.

Note: Refer to "HD146818 Data Sheet" for details.

#### 4.7 Reference Data of Battery Service Life

Fig. 4-7-1 shows the battery service life taken from a silver oxide battery: SR44W (by Hitachi Maxell).

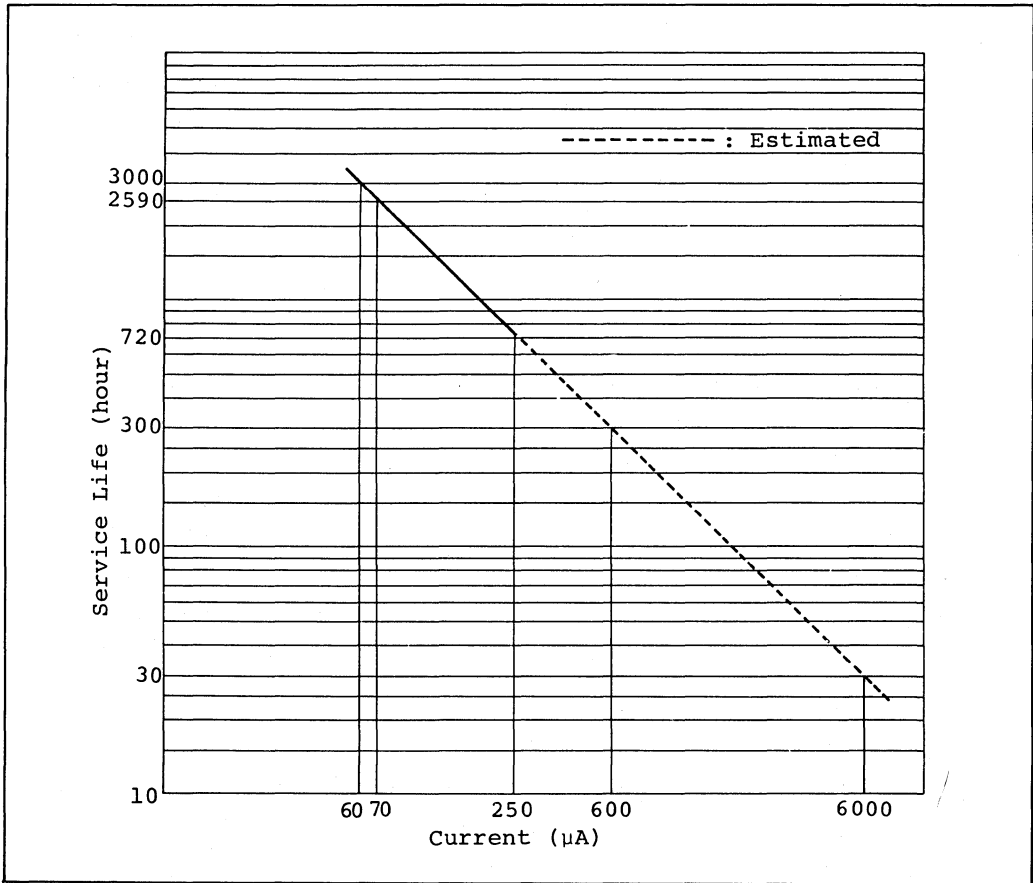


Fig. 4-7-1 Battery Service Life (Maxell SR44W)

## 5. PRECAUTIONS

### 5.1 Write-Only Register

When a write-only register such as the DDR of the port is read by the MPU, "\$FF" always appears on the data bus. Note that when an instruction which reads the memory contents and does some arithmetic operation on the contents of the write-only register, it always gets \$FF as the arithmetic and logical results. AIM, OIM and EIM instructions can not be used for the bit manipulation of the DDR of the I/O port.

### 5.2 Address Strobe (AS)

The AS signal is used as an address latch strobe and is always accompanied with the E-clock. This means the AS is available in both Normal Operation Mode and Sleep Mode whenever the E clock is generated.

### 5.3 Mode 0

This mode is used for the test purpose only. It is not recommended to use this mode for the other purposes.

### 5.4 Trap Interrupt

When executing an RTI instruction at the end of the interrupt routine, trap interrupt different from other interrupts returns to the address where the trap interrupt was generated. Care must be taken when using trap interrupts in the program. See Fig. 5-4-1 and 5-4-2 for details.



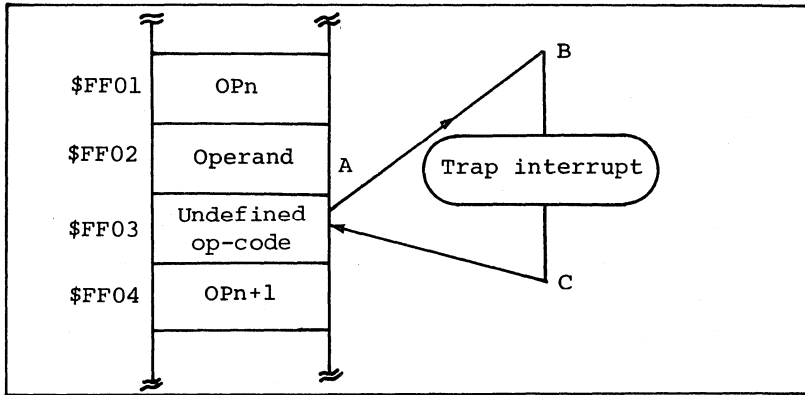


Fig. 5-4-1 Undefined Op-code Trap

After executing OPn instruction, the HD63701V0 fetches and decodes an undefined op-code to generate a trap interrupt. When RTI instruction is executed at the end of the trap interrupt service routine, the HD63701V0 will set \$FF03 in PC, fetch the undefined code again, generate a trap interrupt and repeat ABC endless-loop.

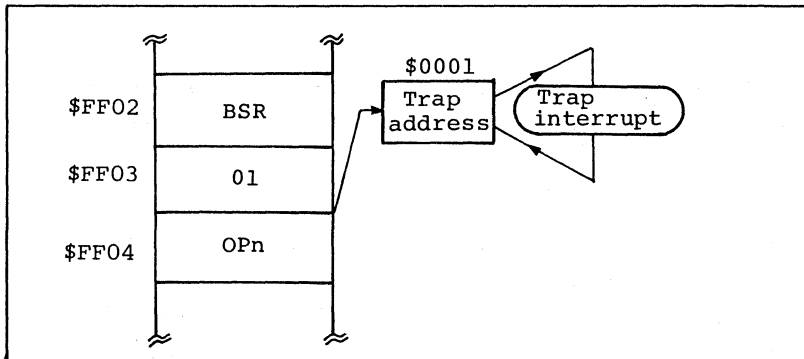


Fig. 5-4-2 Address Trap

After performing BSR instruction, the branch destination address is output on an address bus to fetch the first op-code of a subroutine. If \$0001 is output as an address by some mistake, the HD63701V0 internally decodes it executed at the end of this trap interrupt servicing routine, the HD63701V0 will set \$0001 in PC and restart from this address, which causes a trap interrupt again and repeat this endless-loop.

## 5.5 Precaution on the Board Design of Oscillation Circuit

As shown in Fig. 5-5-1, the cross talk disturbs the normal oscillation if signal lines are put near the oscillation circuit. When designing a board, pay attention not to do that. In addition, crystal and  $C_L$  must be put as close to the HD63701V0 as possible.

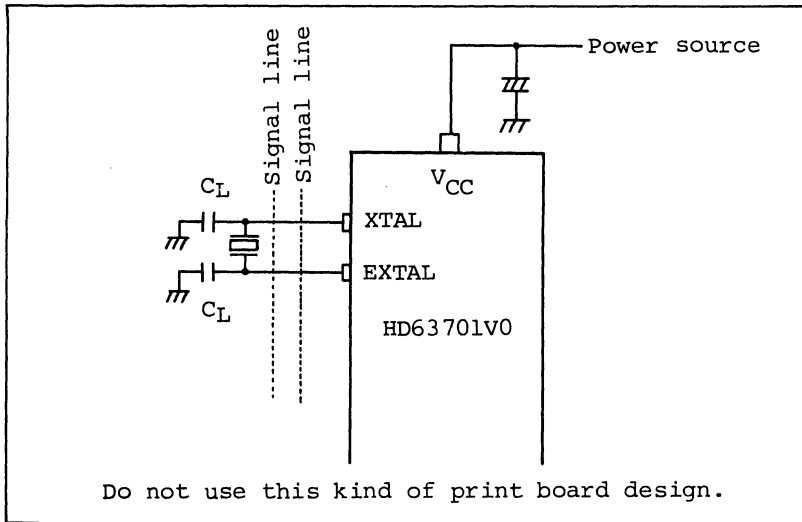


Fig. 5-5-1 Precaution to the Board Design of Oscillation Circuit

## 5.6 Application Note for High Speed System Design Using the HD63701V0

When interfacing the HD63701V0 to the high speed memory (ex. HM6264) in expanded multiplexed mode, noise may appear on the address bus. Therefore, the following countermeasure must be taken to prevent this noise from occurring. However, when using the HD63701V0 in single chip mode, no problem of this sort occur in the bus.

### 5.6.1 Problem

If load capacitance of the data bus exceeds the specification

and the GND impedance is high in HD63701V0 application system, noise may appear on the address bus during the write cycle and a write error may occur. The timing is shown in Fig. 5-6-1.

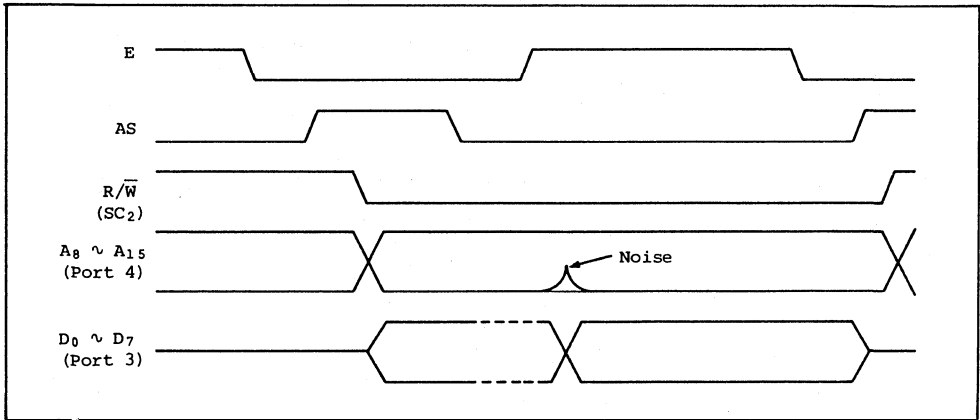


Fig. 5-6-1 Noise Occurrence in Address Bus During Write Cycle

### 5.6.2 Cause

If the data bus changes from "High" to "Low" (from FFH to 00H), extremely large transient current flows through the GND and noise may appear on the GND because of the GND impedance. This noise level appears on all outputs including address bus. (See Fig. 5-6-2.)

Fig. 5-6-3 shows the dependency of the noise level on each parameter.

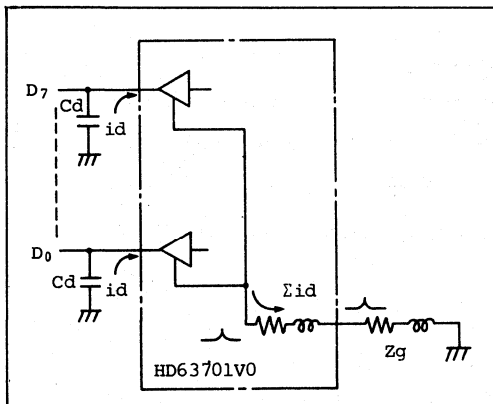


Fig. 5-6-2 Noise Source

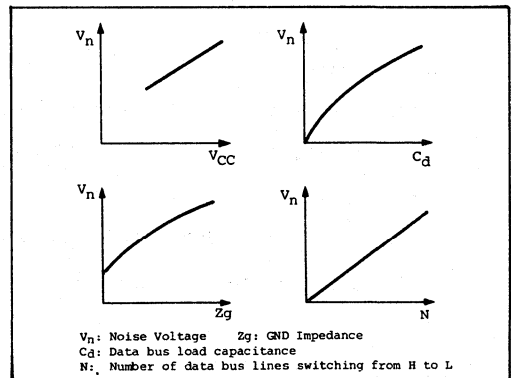


Fig. 5-6-3 Dependency of the Noise Voltage on each Parameter

### 5.6.3 Countermeasure

One of the following three countermeasures must be taken to prevent noise from occurring on the bus.

#### (1) Noise Isolation

The address must be latched at the falling edge of AS (E clock). An example circuit for this countermeasure is shown in Fig. 5-6-4.

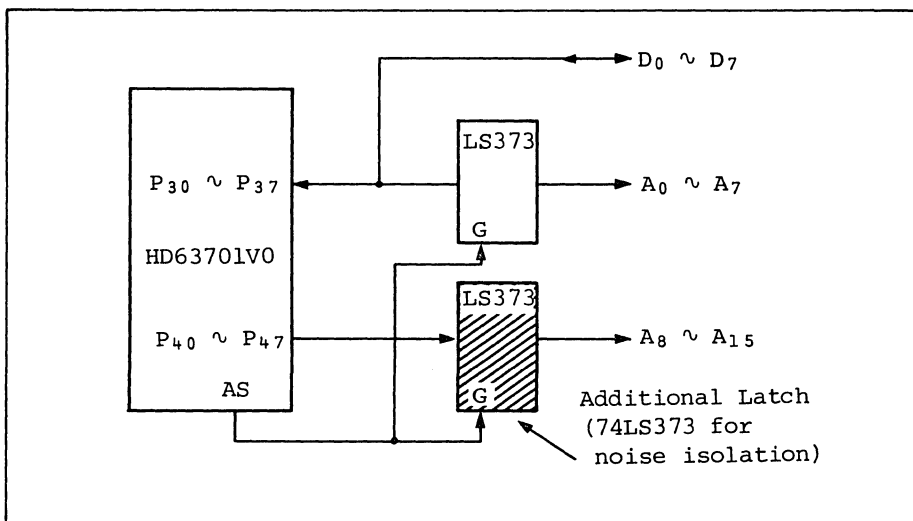


Fig. 5-6-4 Circuit for Countermeasure

#### (2) Transient Current Reduction

The transient current must be reduced by reducing a load capacitance of the data bus. If data bus load capacitance becomes large, a bus buffer must be connected to the data bus.

#### (3) GND Impedance Reduction

Since the noise level in the LSI is max.  $V_{OL}$  (0.55V max.) or less, GND impedance must be reduced as much as possible to lower the noise level to where it will not cause any problems.

#### 5.6.4 Notes on Printed Circuit Board Design

Generally, PC boards based on low speed COMS LSIs can be designed without  $V_{CC}$  or GND impedance problem. However, when designing PC boards based on high speed CMOS LSIs such as the HD63701V0,  $V_{CC}$  and GND lines must be carefully distributed because large transient current flows through the LSI on switching.

When designing PC boards, the following countermeasures must be taken against this large current:

- (1) Widen the GND line width on the PC board.
- (2) Place the HD63701V0 as close to the power source as possible.
- (3) Connect the bypass capacitor between GND and  $V_{CC}$ . (An electrolytic capacitor (0.1 $\mu$ F) and a tantalum capacitor (about 10 $\mu$ F) are connected in parallel in the bypass capacitor.)

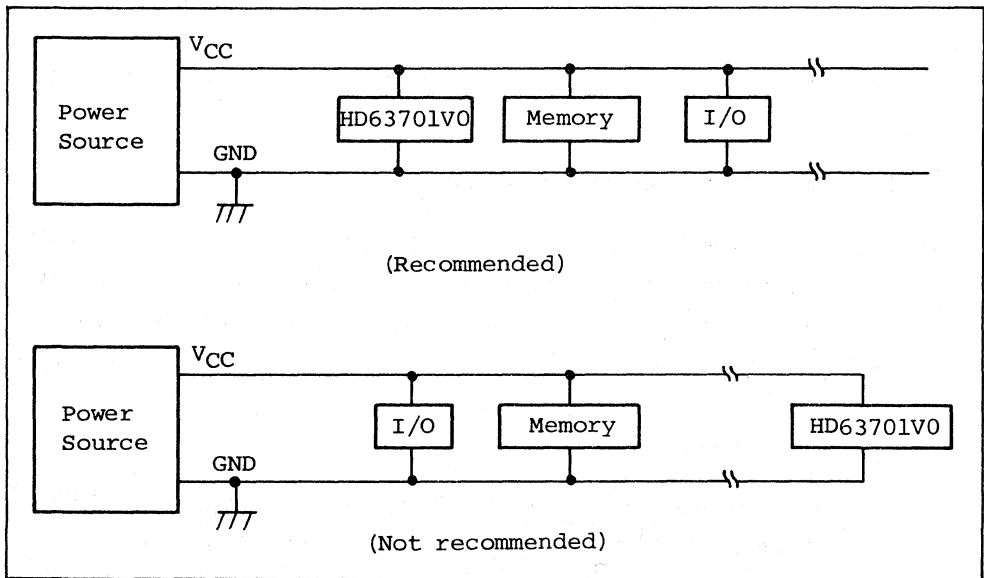
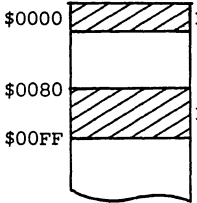
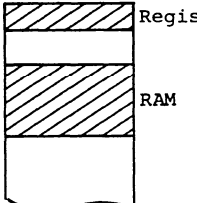
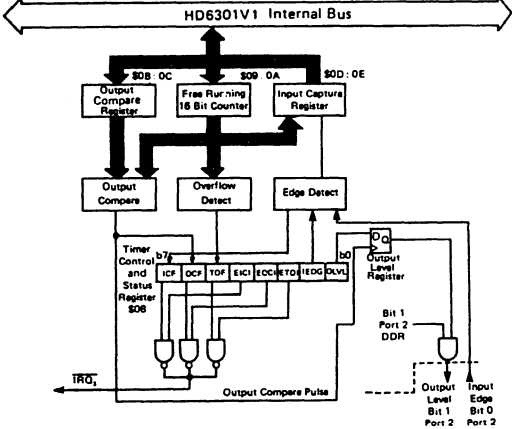
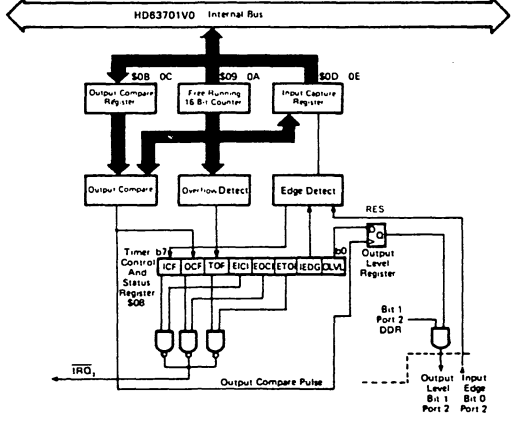
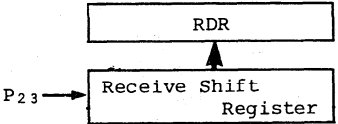
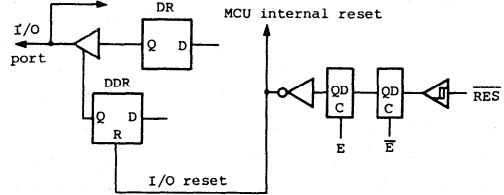
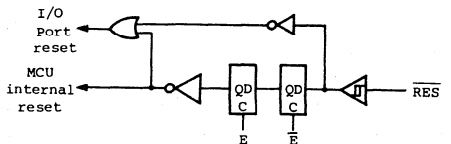
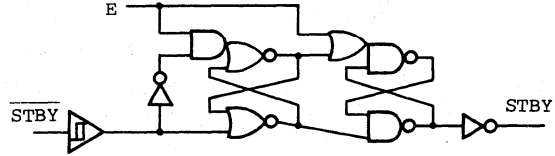
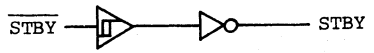
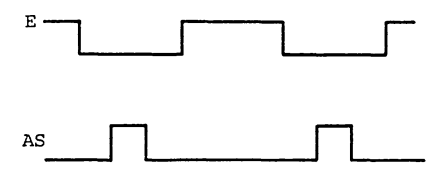
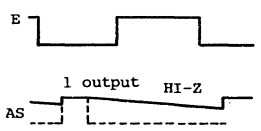
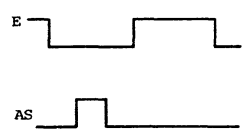


Fig. 5-6-5 Layout of the HD63701V0 on the PC Board

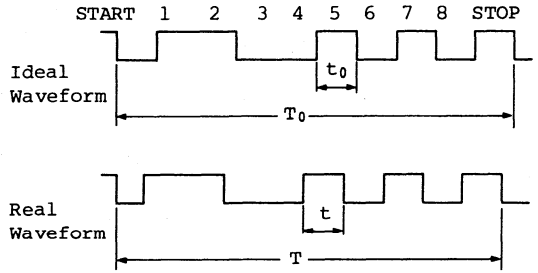
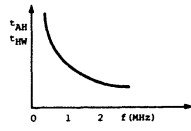
5.7 Differences between HD6301V Series and HD63701V0

Item	HD6301V	HD63701V0
RAM	<p>RAM Size: 128-byte Address : \$0080-\$00FF</p> 	<p>RAM Size: 192-byte Address : \$0040-\$00FF</p> 
Operation Mode	Mode 4: Expanded Multiplexed Mode = Mode 2	HD63701V0 does not have Mode 4
Function	<p>After providing supply voltage, output level is undefined (0 or 1) unless the contents of the Output Compare Register matches with those of the Free Running Counter. The Output Level Register is not initialized by reset.</p>  <p>Fig. 20 Programmable Timer Block Diagram</p>	<p>The Output Level Register is initialized to 0 by reset.</p>  <p>Fig. 20 Programmable Timer Block Diagram</p>

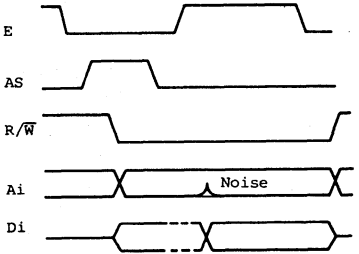
Item	HD6301V	HD63701V0	
Function	HD 6301V1 HD6303R HD63P01M1	HD6303R1	Receive data is transferred from Receive Shift Register to RDR even if framing error occurs.
	When framing error occurs, receive data is not transferred from the Receive Shift Register to Receive Data Register (RDR).	Receive data is transferred from Receive Shift Register to RDR even if framing error occurs.	
			
Port Reset	The DDR of port is reset synchronously with E clock. I/O state is undefined from providing power supply till oscillation start (max. 20ms).	The DDR of port is reset asynchronously with E clock. CPU enters into high impedance state (input state) by bringing RES Low. Reset release and MCU internal reset is performed synchronously with E clock.	
			
Standby Mode	STBY signal is latched synchronously with E clock.	STBY signal is latched asynchronously with E clock. CPU enters into standby state by bringing STBY Low.	
			

Item		HD6301V		HD63701V0
Function	AS (Address) (Strobe)	HD63P01M1	HD6301V1 HD6303R      HD6303R1	 <p>During reset, AS functions normally.</p>
		 <p>In Expanded Multiplexed Mode (mode 0, 2, 4 or 6), AS becomes high impedance state for a half E clock cycle during reset. Therefore, I/O Port 3 functions as data bus during reset.</p>	 <p>During reset, AS functions normally.</p>	



Item	HD6301V		HD63701V0												
Function	HD6301V1 HD6303R      HD6303R1		HD63P01M1												
	The SCI receive margin is shown below.		The SCI receive margin is shown below.												
	<table border="1"> <tr> <td>Bit distortion tolerance (<math>t-t_0</math>)/<math>t_0</math></td> <td>±37.5%</td> </tr> <tr> <td>Character distortion tolerance (<math>T-T_0</math>)/<math>T_0</math></td> <td>+3.75% -2.5%</td> </tr> </table>	Bit distortion tolerance ( $t-t_0$ )/ $t_0$	±37.5%	Character distortion tolerance ( $T-T_0$ )/ $T_0$	+3.75% -2.5%	<table border="1"> <tr> <td>Bit distortion tolerance (<math>t-t_0</math>)/<math>t_0</math></td> <td>±25%</td> </tr> <tr> <td>Character distortion tolerance (<math>T-T_0</math>)/<math>T_0</math></td> <td>±3.75%</td> </tr> </table>	Bit distortion tolerance ( $t-t_0$ )/ $t_0$	±25%	Character distortion tolerance ( $T-T_0$ )/ $T_0$	±3.75%	The SCI receive margin is shown below.   <table border="1" style="margin-top: 20px;"> <tr> <td>Bit distortion tolerance (<math>t-t_0</math>)/<math>t_0</math></td> <td>±37.5%</td> </tr> <tr> <td>Character distortion tolerance (<math>T-T_0</math>)/<math>T_0</math></td> <td>±3.75%</td> </tr> </table>		Bit distortion tolerance ( $t-t_0$ )/ $t_0$	±37.5%	Character distortion tolerance ( $T-T_0$ )/ $T_0$
Bit distortion tolerance ( $t-t_0$ )/ $t_0$	±37.5%														
Character distortion tolerance ( $T-T_0$ )/ $T_0$	+3.75% -2.5%														
Bit distortion tolerance ( $t-t_0$ )/ $t_0$	±25%														
Character distortion tolerance ( $T-T_0$ )/ $T_0$	±3.75%														
Bit distortion tolerance ( $t-t_0$ )/ $t_0$	±37.5%														
Character distortion tolerance ( $T-T_0$ )/ $T_0$	±3.75%														
Supply Voltage	HD6301V1 HD6303R      HD6303R1		HD63P01M1												
	$V_{CC}=5V\pm 10\%$ ( $f=0.1\sim 2\text{MHz}$ ) $=3\sim 6V$ ( $f=0.1\sim 0.5\text{MHz}$ )		$V_{CC}=5V\pm 10\%$ ( $f=0.1\sim 1\text{MHz}$ )												
Address/Data Hold Time ( $t_{AH}, t_{HW}$ )	$t_{AH} = 20 \text{ ns min}$ $t_{HW} = 20 \text{ ns min}$ $t_{AH}$ and $t_{HW}$ are constant independently of operating frequency.		$t_{AH}, t_{HW} = 60 \text{ ns}$ ( $f=1\text{MHz}$ ) $= 40 \text{ ns}$ ( $=1.5\text{MHz}$ ) $= 30 \text{ ns}$ ( $=2\text{MHz}$ ) $t_{AH}$ and $t_{HW}$ are proportion to $1/f$ . ( $f$ = operating frequency) 												

Item		HD6301V	HD63701V0										
Spec.	Address Delay Time	<p>(1) <math>t_{AD1}</math> and <math>t_{AD2}</math> are constant independently of operating frequency. In HD63B01V (B version of HD6301V), <math>t_{AD1}</math> and <math>t_{AD2}</math> are 160 ns max. at 0.1MHz through 2MHz operation.</p> <p>(2) <math>t_{ADL}</math> is related to operating frequency. (<math>t_{ADL}</math> is in proportion to <math>1/f</math>. <math>f</math>: operating frequency)</p>	<p><math>t_{AD1}</math>, <math>t_{AD2}</math> and <math>t_{ADL}</math> are related to operating frequency (They are in proportion to <math>1/f</math>. <math>f</math>: operating frequency). Therefore, if HD63701V0 operates at lower operating frequency, <math>t_{AD1}</math>, <math>t_{AD2}</math> and <math>t_{ADL}</math> will become 160 ns or more. <math>t_{AD1}</math>, <math>t_{AD2}</math> and <math>t_{ADL}</math> are calculated as follows.</p> $t_{AD} (f \text{ MHz}) \doteq 250 \text{ ns} (1 \text{ MHz}) \times \frac{1}{f} (\text{MHz})$										
	$I_{in}$ and $C_{in}$ of $\overline{RES}$	$I_{in} = 1.0\mu\text{A max.}$ $C_{in} = 12.5\text{pF max.}$	$I_{in} = 10\mu\text{A max.}$ $C_{in} = 50\text{pF max.}$ Since $\overline{RES}$ is multiplexed with $V_{pp}$ , $C_{in}$ and $I_{in}$ are larger than those of HD6301V.										
	Load Capacitance of E	2 - LSTTL + 40pF $I_{OL} = 0.8 \text{ mA}$ $I_{OH} = -200\mu\text{A}$	1 - TTL + 90pF $I_{OL} = 1.6\text{mA}$ $I_{OH} = -200\mu\text{A}$										
	Load Capacitance of Port 1	1 - TTL + 30pF	1 - TTL + 90pF										
	Spec. of Crystal Oscillator	Spec. $R_s = 60\Omega \text{ max.}$	Spec. <table border="1"> <thead> <tr> <th>Clock frequency (MHz)</th> <th>2.5</th> <th>4.0</th> <th>6.0</th> <th>8.0</th> </tr> </thead> <tbody> <tr> <td><math>R_s \text{ max. } (\Omega)</math></td> <td>500</td> <td>120</td> <td>80</td> <td>60</td> </tr> </tbody> </table>	Clock frequency (MHz)	2.5	4.0	6.0	8.0	$R_s \text{ max. } (\Omega)$	500	120	80	60
	Clock frequency (MHz)	2.5	4.0	6.0	8.0								
$R_s \text{ max. } (\Omega)$	500	120	80	60									
Storage Temperature	$T_{stg} = -55 - +150^\circ\text{C}$	$T_{stg} = -55 - +125^\circ\text{C}$											

Item		HD6301V			HD63701V0
Function	GND Noise	HD6301V1	HD6303R	HD6303R1 HD63P01M1	
		 <p>Noise is reduced by 33%.</p> <p>If load capacitance in each data line and GND impedance are large, noise may appear on address bus during MCU write cycle and data won't be written into RAM correctly. The noise is caused by GND impedance which becomes large when large transient current flows into GND at High to Low transition of data line.</p>			Noise is reduced by 50%.
	Miscellaneous	<p>Chip design and manufacturing process of the HD6301V differ from those of the HD63701V0. Therefore, actual spec. and margin are different between the HD6301V and the HD63701V0. Please carefully examine your system before applying HD6301V or HD63701V0 to your system.</p>			

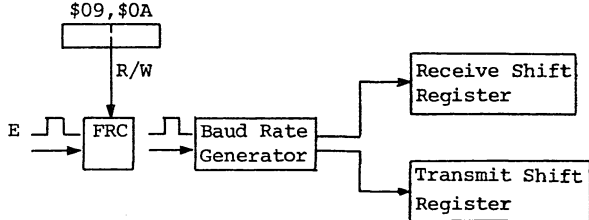
## APPENDIX

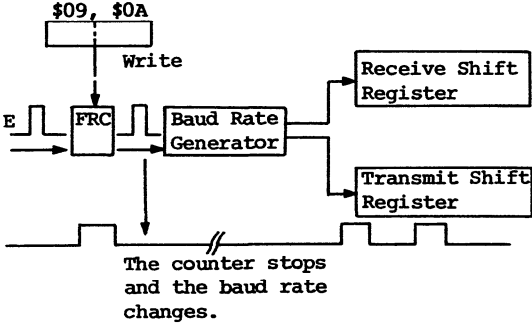
<PROM Programmer and Socket Adaptor>

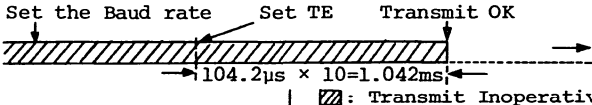
General purpose PROM programmer corresponding to the 27256 can perform programming to the HD63701V0. When programming, a socket adaptor which changes the number of pins, 40 pins to 28 pins, is necessary.

PROM Programmer and Socket Adaptor		
Products name	PROM Programmer	Socket Adaptor
HD63701V0	PROM Programmers for 27256	H31VSA01A

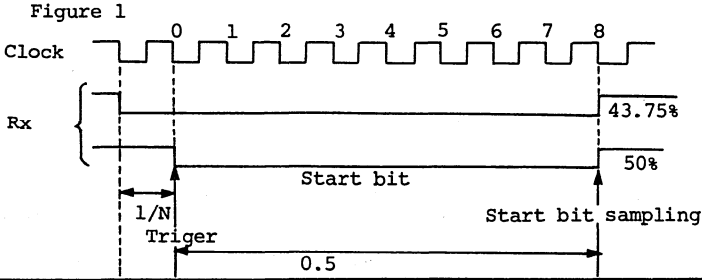
II Q & A

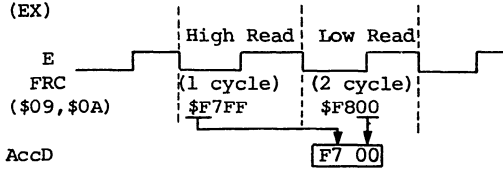
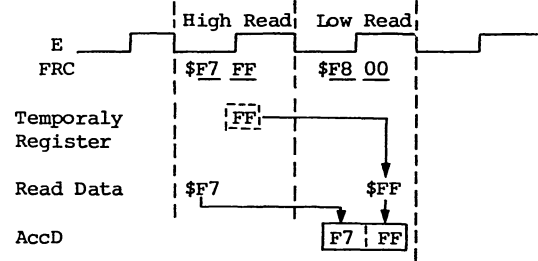
Category	Question	Answer
Process to Use a Port as an Outputs	When using an I/O port as an output, is the data stored to the Data Register or is the Data Direction Register (DDR) set at first?	Store the data to the Data Register at first and then set DDR (DDR=1); if not, unknown data is output from the port.
Relation between Writing into the FRC and SCI Operation	How are writing into the timer Free Running Counter(FRC) and the Serial Communication Interface(SCI) related?	<p>The source of the clock input to the SCI Shift Registers is the timer FRC. Therefore, if new data is written into the FRC, SCI operations are disturbed. See the following diagram.</p>  <pre> graph TD     E[E] --&gt; FRC[FRC]     FRC --&gt; BRG[Baud Rate Generator]     BRG --&gt; RSR[Receive Shift Register]     BRG --&gt; TSR[Transmit Shift Register]     FRC --- RW[R/W]     RW --- RA["\$09, \$0A"]     </pre> <p>* A write into the FRC is prohibited during SCI operations.</p>
Writing into the FRC during Serial Receive/Transmit	Is it prohibited to write data into the Free Running Counter(FRC) during serial receive/transmit?	Yes. If data is written into the FRC during serial receive/transmit, the FRC stops counting up and the baud rate changes. In condition other than serial receive/transmit, it's possible to write.

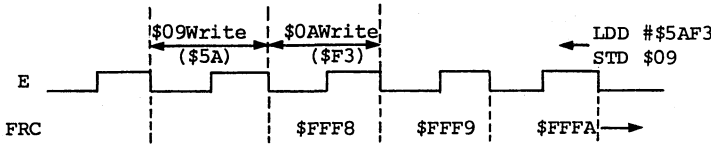
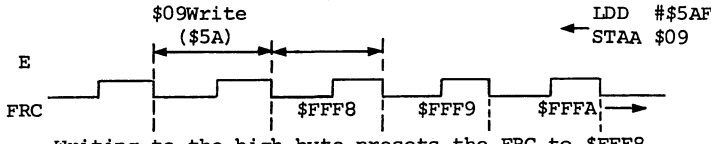
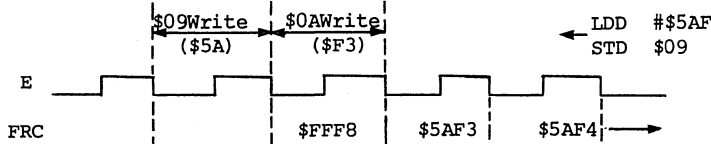
Category	Question	Answer																																				
		 <p>The counter stops and the baud rate changes.</p>																																				
RDRF State When SCI Receiving	<p>What is the state of the Receive Data Register Full (RDRF) flag when the HD63701V0 SCI can receive signals (RE=1) and the wake-up flag (WU bit) is set?</p> <p>TRCSR</p> <table border="1" data-bbox="430 740 951 830"> <tr> <td></td> <td>7</td> <td>6</td> <td>5</td> <td>4</td> <td>3</td> <td>2</td> <td>1</td> <td>0</td> </tr> <tr> <td>\$0011</td> <td>RDRF</td> <td>ORFE</td> <td>TDRE</td> <td>RIE</td> <td>RE</td> <td>TIE</td> <td>TE</td> <td>WU</td> </tr> <tr> <td></td> <td>↓</td> <td></td> <td></td> <td></td> <td>↓</td> <td></td> <td></td> <td>↓</td> </tr> <tr> <td></td> <td>?</td> <td></td> <td></td> <td></td> <td>1</td> <td></td> <td></td> <td>1</td> </tr> </table>		7	6	5	4	3	2	1	0	\$0011	RDRF	ORFE	TDRE	RIE	RE	TIE	TE	WU		↓				↓			↓		?				1			1	<p>When the wake-up flag is set (WU=1) the RDRF flag cannot be set. (RDRF=0)</p>
	7	6	5	4	3	2	1	0																														
\$0011	RDRF	ORFE	TDRE	RIE	RE	TIE	TE	WU																														
	↓				↓			↓																														
	?				1			1																														

Category	Question	Answer
<p>Serial I/O Operation</p>	<p>The serial I/O does not operate satisfactorily. Initialization does not seem to be wrong, but the data is not transmitted. What is wrong?</p> <p>Initialize by User Program</p> <ol style="list-style-type: none"> <li>(1) Set the Rate/Mode Control Register (RMCR) to the desired operation.</li> <li>(2) Set the Transmit/Receive Control Status Register (TRCSR) to the desired operation.</li> </ol>	<p>Just after the initialization of serial I/O, the data transmit is not operative during 10 cycles of Baud Rate after setting the TE. The reason is as follows. Setting the transmit enable bit (TE bit) causes ten consecutive "1" of preamble and makes the transmitter section operative. In other words, the transmitter section gets ready after one frame (10 bits) transmitting time according to the Baud rate.</p> <p>(ex.) When the Baud rate is set to 9600 Baud (104.2<math>\mu</math>s at 1 bit),</p>  <p>1.042ms after setting the TE, the transmitter section is operative.</p>
<p>Serial I/O Register Read</p>	<p>When transmitting the data, is reading the Transmit/Receive Control Register (TRCSR) required?</p> <p>When the transfer interval is long enough compared with the Baud rate, Transmit Data Register Empty (TDRE) will be set. In that case, are there any problems when transmitting data without checking the TDRE flag in the TRCSR?</p>	<p>The TDRE flag shows if the TDRE register is empty or not. When writing a data to the TDR with TDRE=1, it's not necessary to check the TDRE. But reading the TDRE flag tells us the contents of TDR. For example, when new data is written to the TDR with TDRE "0" (TDR already has a data), the old data will be erased. When the transfer interval is long enough compared with the Baud rate, there's no problem. However, check TRCSR if possible.</p>

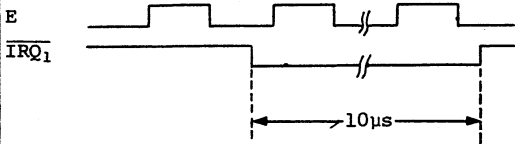
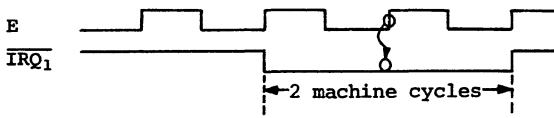
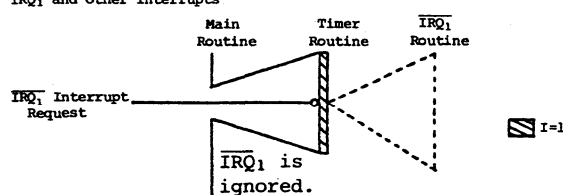
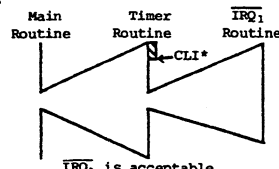


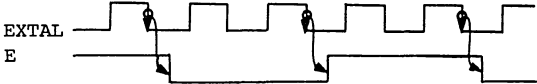
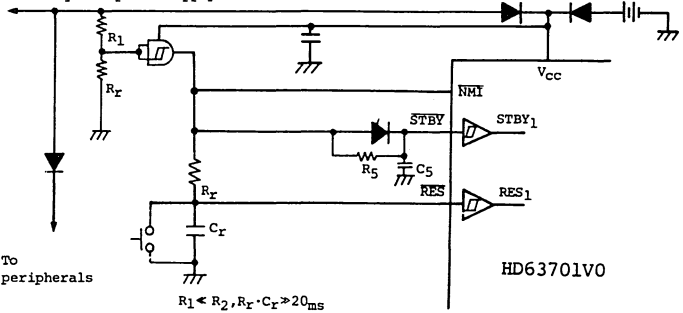
Category	Question	Answer												
Detection of the HD63701V0 Serial Start Bit	(1) What is the relation between the HD63701V0 serial sampling clock frequency and the baud rate ? (2) What does "Sampling error" mean ?	(1) The serial sampling clock frequency is eight times the baud rate. (2) "Sampling error" means receive margin at the serial operation time.  <u>Receive margin:</u> The HD63701V0 detects the start bit and samples the data bit using the falling edge of the sampling clock. The general equation is shown as follows.  1.) General equation $M = [ (0.5 - 1/N) - (D - 0.5)/N - (L - 0.5)F ] \times 100 (\%)$ M: Receive margin N: Ratio of baud rate to sampling clock (0 to 0.5) D: Duty of the longer sampling clock of "H", and "L" L: Frame length (7 to 12 bits) F: Absolute value of deviation of sampling clock frequency  2.) Abbreviated equation $M = (0.5 - 1/N) \times 100 (\%)$ Conditions: D = 0.5, F = 0  <table border="1" data-bbox="991 808 1680 890"> <thead> <tr> <th>N</th> <th>8</th> <th>16</th> <th>32</th> <th>64</th> <th>Note</th> </tr> </thead> <tbody> <tr> <td>M (%)</td> <td>37.5</td> <td>43.75 (Fig.1)</td> <td>46.875</td> <td>48.4375</td> <td>In the HD63701V0, N=8.</td> </tr> </tbody> </table> Figure 1 	N	8	16	32	64	Note	M (%)	37.5	43.75 (Fig.1)	46.875	48.4375	In the HD63701V0, N=8.
N	8	16	32	64	Note									
M (%)	37.5	43.75 (Fig.1)	46.875	48.4375	In the HD63701V0, N=8.									

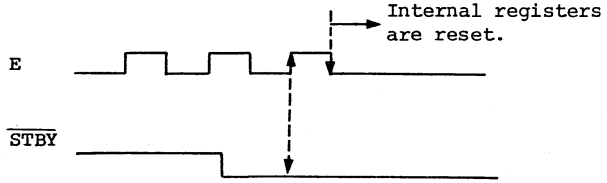
Category	Question	Answer						
<p>Free Running Counter Read</p>	<p>When the FRC of the HD63701V0 is read with the double byte load instructions (2 cycle execution for FRC reading), is it read correctly? Double byte load instructions require two cycles to be executed and the cycle to read the low byte of FRC becomes the next cycle of the high byte. Is it OK ?</p> <p>(EX)</p>  <p>(When reading \$F7FF from the counter)</p>	<p>The FRC of the HD63701V0 contains a parallel temporary register. When the high byte of the FRC is read, the low byte is set in the temporary register. The Low byte data in the temporary register is set to the AccD at the next cycle. Therefore, it is possible to read the FRC correctly.</p>  <p>(When reading \$F7FF from the counter)</p>						
<p>Preset Method of the Free Running Counter</p>	<p>What is the difference between the HD6801V and HD63701V0 in writing data into the free running counter ?</p>	<p>The FRC preset method of the HD6801V is different from the HD63701V0.</p> <table border="1" data-bbox="980 772 1663 1004"> <thead> <tr> <th>Type</th> <th>Preset Method</th> </tr> </thead> <tbody> <tr> <td>HD6801V</td> <td>The FRC is always preset to "\$FFF8".</td> </tr> <tr> <td>HD63701V0</td> <td> <ol style="list-style-type: none"> <li>1. Writing to the high byte presets the FRC to \$FFF8.</li> <li>2. The FRC is set to desirable data by a double byte store instruction.</li> </ol> </td> </tr> </tbody> </table>	Type	Preset Method	HD6801V	The FRC is always preset to "\$FFF8".	HD63701V0	<ol style="list-style-type: none"> <li>1. Writing to the high byte presets the FRC to \$FFF8.</li> <li>2. The FRC is set to desirable data by a double byte store instruction.</li> </ol>
Type	Preset Method							
HD6801V	The FRC is always preset to "\$FFF8".							
HD63701V0	<ol style="list-style-type: none"> <li>1. Writing to the high byte presets the FRC to \$FFF8.</li> <li>2. The FRC is set to desirable data by a double byte store instruction.</li> </ol>							

Category	Question	Answer
		<p>(1) The HD6801V Preset Method</p>  <p>The FRC is always preset to \$FFF8.</p> <p>(2) The HD63701V0 Preset Method</p> <p>1. \$FFF8</p>  <p>Writing to the high byte presets the FRC to \$FFF8.</p> <p>2. Optional valve (In this case \$5AF3)</p>  <p>The FRC is set to desirable data (\$5AF3) by a double byte store instruction.</p>
Output of Address Strobes(AS) in the Multiplexed Mode	Is AS always output when using the HD63701V0 in the expanded multiplexed mode (mode 2, 4, 6)?	Yes. AS is always output in the expanded multiplexed mode, even when the MPU accesses the internal RAM, ROM, etc.

Category	Question	Answer
<p><math>\overline{IRQ}_1</math> Acceptance</p>	<p>(1) Is <math>\overline{IRQ}_1</math> ignored when the Condition Code Register I mask is set?</p> <p>(2) After the I mask is reset, will the interrupt sequence start by the interrupt request flag having been latched?</p>	<p>(1) If the Condition Code Register I mask is set, <math>\overline{IRQ}_1</math> is completely ignored.</p> <p>(2) With the I mask set, the interrupt request flag will not be latched.</p> <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> <p>(1) Reset starts</p> <p><math>\overline{IRQ}_1</math> is ignored.</p> </div> <div style="text-align: center;"> <p>(2) Reset starts</p> <p><math>\overline{IRQ}_1</math> is ignored.</p> </div> </div>
<p>Timer Interrupt and External Interrupt</p>	<p>In the routine below, when is the next timer interrupt accepted?</p> <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> <p>Main Routine</p> </div> <div style="text-align: center;"> <p>Timer (OCI) Routine (Execution time =1.5ms)</p> </div> <div style="text-align: center;"> <p>External Interrupt (IRQ) Routine (Execution time=3ms)</p> </div> </div>	<p>The next timer interrupt is accepted in the main routine just after RTI instruction execution.</p>

Category	Question	Answer
<p><math>\overline{\text{IRQ}}_1</math> Interrupt and Other Interrupts</p>	<p><math>\overline{\text{IRQ}}_1</math> pin (pin 5) is held at low for 10<math>\mu\text{s}</math> but an interrupt does not occur. What should be done to generate an interrupt sequence?</p> 	<p>(1) <math>\overline{\text{IRQ}}_1</math> is a level sensitive interrupt pin which needs a minimum of 2 machine cycles (2<math>\mu\text{s}</math> at 1MHz) to accept an interrupt. However, if another interrupt has been already generated, no interrupt request is accepted with <math>\overline{\text{IRQ}}_1</math> at low for 10<math>\mu\text{s}</math>. In such a case, <math>\overline{\text{IRQ}}_1</math> should be held at low until the request is accepted.</p>  <p>(2) In this case, as a timer interrupt is executed the interrupt mask is automatically set. So <math>\overline{\text{IRQ}}_1</math> is ignored. See the followings for the illustration of <math>\overline{\text{IRQ}}_1</math> and other interrupts and a countermeasure.</p> <p><math>\overline{\text{IRQ}}_1</math> and Other Interrupts</p>  <p>Countermeasure</p> <p>Clear the I mask at the beginning of the timer interrupt routine.</p>  <p><math>\overline{\text{IRQ}}_1</math> is acceptable.</p> <p>*CLI : Clears the interrupt mask (I=0).</p> <p>With this method, note the following ;</p> <p>(1) <math>\overline{\text{IRQ}}_1</math> may be ignored when the request occurs during timer interrupt vectoring.</p> <p>(2) Interrupts from NMI or SWI are excluded.</p>

Category	Question	Answer				
<p>CLI Instruction and Interrupt Operation</p>	<p>In the HD63701V0, a timer interrupt is not accepted in the following program. Is there any problem?</p> <pre style="border: 1px dashed black; padding: 10px; width: fit-content; margin: auto;"> Main Routine L01 CLI       NOP       SEI       :       :       :       BRA L01                     </pre>	<p>To accept an interrupt, two machine cycles are necessary between CLI and SEI. That is, in this program, two NOP instructions are necessary. The same thing can be said when using TAP for CLI and SEI.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%;">Using CLI</th> <th style="width: 50%;">Using TAP</th> </tr> </thead> <tbody> <tr> <td style="border: 1px dashed black; padding: 5px;"> <pre> L01 CLI       NOP       NOP       SEI       :       :       :       BRA L01                     </pre> </td> <td style="border: 1px dashed black; padding: 5px;"> <pre> TAP (Clears the I mask) NOP NOP TAP (Sets the I mask) : : :                     </pre> </td> </tr> </tbody> </table>	Using CLI	Using TAP	<pre> L01 CLI       NOP       NOP       SEI       :       :       :       BRA L01                     </pre>	<pre> TAP (Clears the I mask) NOP NOP TAP (Sets the I mask) : : :                     </pre>
Using CLI	Using TAP					
<pre> L01 CLI       NOP       NOP       SEI       :       :       :       BRA L01                     </pre>	<pre> TAP (Clears the I mask) NOP NOP TAP (Sets the I mask) : : :                     </pre>					
<p>Relation between the External Clock (EXTAL Clock) and Enable Clock (E Clock)</p>	<p>With which edges of the EXTAL clock does the E clock change synchronously, rising edge (↑) or falling edge (↓)?</p>	<p>It changes synchronously with the falling edge (↓) of the EXTAL clock.</p> 				
<p>Constants of the Reset Circuit</p>	<p>Does the capacitor of the recommended reset circuit in the HD63701V0 have an upper limit?</p>	<p>Capacitor <math>C_r</math> does not have upper limit because of the Schmitt trigger circuit provided with the RES. Available if <math>R_r \cdot C_r \gg 20\text{ms}</math></p> <p>To the system power supply</p>  <p style="text-align: center;"><math>R_1 \ll R_2, R_r \cdot C_r \gg 20\text{ms}</math></p>				

Category	Question	Answer
Port Output After Resetting	What data does a port output when the Data Direction Register (DDR)=1 after resetting?	After resetting, since the Data Register of a port is undefined, undefined data is output when the DDR=1. Input definite data by programming in the Data Register before setting the DDR=1.
Schmitt Trigger Circuit of $\overline{STBY}$	Is the Schmitt trigger circuit provided with the HD63701V0 $\overline{STBY}$ ?	Yes.
Return from Standby Mode	What occurs when returning from the standby mode without using $\overline{RES}$ ?	The CPU does not operate normally because the contents of each register are not definite. Therefore, always use the $\overline{RES}$ when returning from the standby mode.
Going into the Standby Mode	Does the CPU go into the standby mode after current instruction execution is completed?	<p>No. Because there is no connection between the instruction execution sequence and the standby mode. That is, when the <math>\overline{STBY}</math> pin goes into "Low", the state is latched at the next rising edge of E clock. Then the internal registers are reset at the next falling edge.</p> 

Category	Question	Answer
<p>Timing for the Standby Mode</p>	<p>The timing for the standby mode is shown in the HD63701V0 user's manual. <math>T_1</math> is not defined. How long is <math>T_1</math>?</p> <p>① NMI ② RES ③ STBY</p> <p>RAM Control Register Set</p> <p>Reset Start</p> <p><math>T_1</math> : Oscillation Stabilization Time</p>	<p>After the RAM Control Register is set in the NMI routine, either <math>\overline{STBY}</math> or <math>\overline{RES}</math> can be in the low state with no priority.</p>
<p>Usage of Bit Manipulator Instructions</p>	<p>How the bit manipulation instructions of the HD63701V0 should be written?</p>	<p>They are written as follows;</p> <p>OIM # \$ 0 4 , \$ 1 0 (Direct Addressing)  OIM # \$ 0 4 , \$ 1 0 , X (Index Addressing)</p> <p style="margin-left: 40px;"> ↓                      ↓                      ↓  Immediate Data    Address                      Index Register</p> <p>This is an example of OR operation of the immediate data and the memory and storing the result in the memory. The HD63701V0 has the following bit manipulation instructions.</p> <p>OIM .... (IMM) · (M) → (M)  AIM .... (IMM) + (M) → (M)  EIM .... (IMM) ⊕ (M) → (M)  TIM .... (IMM) ^ (M)</p> <p>These instructions are written in the same way.</p> <p>The following bit manipulations have different mnemonics in the same OP code.</p>



Category	Question	Answer																																		
		<table border="1" data-bbox="956 252 1680 828"> <thead> <tr> <th colspan="2" data-bbox="956 278 1072 310">OP code</th> <th colspan="3" data-bbox="1072 278 1680 310">Bit Manipulation Instruction</th> </tr> <tr> <th colspan="2" data-bbox="956 310 1072 349"></th> <th colspan="2" data-bbox="1072 310 1291 349">Mnumonics</th> <th data-bbox="1291 310 1680 349">Function</th> </tr> </thead> <tbody> <tr> <td data-bbox="956 349 1017 432">71</td> <td data-bbox="1017 349 1072 432">61</td> <td data-bbox="1072 349 1170 432">A I M</td> <td data-bbox="1170 349 1291 432">B C L R</td> <td data-bbox="1291 349 1680 432">           0 <math>\rightarrow</math> Mi            The memory bit i(i=0 to 7) is cleared and the other bits don't change.         </td> </tr> <tr> <td data-bbox="956 432 1017 544">72</td> <td data-bbox="1017 432 1072 544">62</td> <td data-bbox="1072 432 1170 544">O I M</td> <td data-bbox="1170 432 1291 544">B S E T</td> <td data-bbox="1291 432 1680 544">           1 <math>\rightarrow</math> Mi            The memory bit i(i=0 to 7) is set and the other bits don't change.         </td> </tr> <tr> <td data-bbox="956 544 1017 656">75</td> <td data-bbox="1017 544 1072 656">65</td> <td data-bbox="1072 544 1170 656">E I M</td> <td data-bbox="1170 544 1291 656">B T G L</td> <td data-bbox="1291 544 1680 656">           Mi <math>\rightarrow</math> <math>\overline{M_i}</math>            The memory bit i(i=0 to 7) is inverted and the other bits don't change.         </td> </tr> <tr> <td data-bbox="956 656 1017 828">7B</td> <td data-bbox="1017 656 1072 828">6B</td> <td data-bbox="1072 656 1170 828">T I M</td> <td data-bbox="1170 656 1291 828">B T S T</td> <td data-bbox="1291 656 1680 828">           1 <math>\cdot</math> Mi            AND operation test of the memory bit i(i=0 to 7) and "1" is executed and its corresponding condition code is changed.         </td> </tr> </tbody> </table> <p data-bbox="956 828 1239 896"> <math>\uparrow</math> Direct Addressing      <math>\swarrow</math> Index Addressing         </p> <p data-bbox="956 935 1633 956">The mnunonics mentioned above can be written as follows.</p> <p data-bbox="956 982 1607 1004">BCLR 3,\$10 <math>\leftrightarrow</math> AIM #\$F7, \$10 (Direct Addressing)</p> <p data-bbox="956 1005 1607 1027">BCLR 3,\$10,X <math>\leftrightarrow</math> AIM #\$F7, \$10,X (Index Addressing)</p> <p data-bbox="956 1053 1607 1075">BSET 3,\$10 <math>\leftrightarrow</math> OIM #\$08, \$10 (Direct Addressing)</p> <p data-bbox="956 1076 1607 1098">BSET 3,\$10,X <math>\leftrightarrow</math> OIM #\$08, \$10,X (Index Addressing)</p> <p data-bbox="1017 1098 1355 1150"> <math>\downarrow</math> Bit Address      <math>\swarrow</math> Index Register         </p>					OP code		Bit Manipulation Instruction					Mnumonics		Function	71	61	A I M	B C L R	0 $\rightarrow$ Mi The memory bit i(i=0 to 7) is cleared and the other bits don't change.	72	62	O I M	B S E T	1 $\rightarrow$ Mi The memory bit i(i=0 to 7) is set and the other bits don't change.	75	65	E I M	B T G L	Mi $\rightarrow$ $\overline{M_i}$ The memory bit i(i=0 to 7) is inverted and the other bits don't change.	7B	6B	T I M	B T S T	1 $\cdot$ Mi AND operation test of the memory bit i(i=0 to 7) and "1" is executed and its corresponding condition code is changed.
OP code		Bit Manipulation Instruction																																		
		Mnumonics		Function																																
71	61	A I M	B C L R	0 $\rightarrow$ Mi The memory bit i(i=0 to 7) is cleared and the other bits don't change.																																
72	62	O I M	B S E T	1 $\rightarrow$ Mi The memory bit i(i=0 to 7) is set and the other bits don't change.																																
75	65	E I M	B T G L	Mi $\rightarrow$ $\overline{M_i}$ The memory bit i(i=0 to 7) is inverted and the other bits don't change.																																
7B	6B	T I M	B T S T	1 $\cdot$ Mi AND operation test of the memory bit i(i=0 to 7) and "1" is executed and its corresponding condition code is changed.																																

Category	Question	Answer
Usage of Bit Manipulation Instructions to the Port	Are the bit manipulation instructions (AIM, OIM, EIM, TIM) executable when a port is in the output state (DDR=1)?	<p>It can be used if the port is in the output state (DDR=1). However, the bit manipulation instruction is executed as follows ;</p> <ol style="list-style-type: none"> <li>① Reads specified address.</li> <li>② Executes logical operation.</li> <li>③ Writes the result into the specified address.</li> </ol> <p>Since the specified address ① reads the pin state of the port, the data is influenced by the pins even if any data is output from the port.</p>
RAM Access Disable during Program Execution	<p>When executing a program with the RAME bit of the RAM Control Register disabled,</p> <ol style="list-style-type: none"> <li>(1) What occurs if the internal RAM address is accessed?</li> <li>(2) What occurs if the interrupt requests are generated?</li> </ol>	<ol style="list-style-type: none"> <li>(1) The external RAM can be accessed; the internal RAM is neither readable nor writable when the RAME bit is disabled.</li> <li>(2) If there is no stacking area other than the internal RAM, the MPU will burst when returning from the interrupt sequence.</li> </ol>



# HD6301/HD6303 SERIES HANDBOOK

## Section Five

# HD6301X0/ HD6303X/ HD63701X0 User's Manual

5



**Section 5**  
**HD6301X0/HD6303X/HD63701X0 User's Manual**  
**Table of Contents**

	<b>Page</b>
1. OVERVIEW .....	375
1.1 Features .....	375
1.2 Block Diagrams .....	377
1.3 Pin Description .....	380
2. INTERNAL ARCHITECTURE AND OPERATION .....	383
2.1 Operation Modes .....	383
2.2 Memory Map .....	386
2.3 Function Pin Description .....	388
2.4 Ports .....	393
2.5 RAM/Port 5 Control Register .....	397
3. CPU FUNCTION .....	399
3.1 CPU Registers .....	399
3.2 Addressing Modes .....	400
3.3 Instruction Set .....	402
3.4 CPU Instruction Flow .....	408
3.5 Low Power Dissipation Modes .....	410
3.6 Trap Function .....	413
3.7 Reset .....	414
3.8 Interrupts .....	415
4. TIMER 1 .....	418
4.1 Free-Running Counter (FRC) .....	419
4.2 Output Compare Registers (OCR) .....	419
4.3 Input Compare Register (ICR) .....	420
4.4 Timer Control/Status Register 1 (TCSR1) .....	420
4.5 Timer Control/Status Register 2 (TCSR2) .....	421
4.6 Timer Status Flags .....	423
4.7 Precautions on Cleaning the OCF .....	423
5. TIMER 2 .....	425
5.1 Timer 2 Upcounter (T2CNT) .....	425
5.2 Timer Constant Register (TCONR) .....	426
5.3 Timer Control/Status Register 3 (TCSR3) .....	426
5.4 Timer Status Flags .....	428

6. SERIAL COMMUNICATIONS INTERFACE .....	429
6.1 Initialization .....	429
6.2 Asynchronous Mode .....	430
6.3 Clock Synchronous Mode .....	431
6.4 Transmit/Receive Control Status Register (TRCSR) .....	433
6.5 Transmit Rate/Mode Control Register (RMCR) .....	434
6.6 SCI Receiving Margin .....	438
6.7 Timer, SCI Status Flags .....	438
6.8 Precaution for Clock-Synchronous Serial Communication Interface .....	439
7. HD63701X0 PROGRAMMABLE ROM (EPROM) .....	441
7.1 Programming and Verification .....	443
7.2 Erasing (Window Package) .....	444
7.3 Characteristics and Applications .....	444
8. APPLICATIONS .....	447
8.1 HD6301X0 or HD63701X0 in Expanded Mode .....	447
8.2 HD6301X0 or HD63701X0 in Single-Chip Mode .....	448
8.3 Timer Applications .....	448
8.4 SCI Applications .....	452
8.5 Lowering Operating Current .....	454
8.6 Memory Ready Application .....	457
8.7 Halt Application .....	458
8.8 RD, WR Application .....	459
8.9 LCD-II Interface Application .....	460
8.10 Oscillation Board Design .....	461
APPENDIX I. ELECTRICAL CHARACTERISTICS .....	463
APPENDIX II. INSTRUCTION EXECUTION CYCLES .....	485

APPENDIX III. QUESTIONS AND ANSWERS .....	491
III.1 Parallel Ports .....	491
III.2 Serial Ports .....	492
III.3 Timer/Counter .....	493
III.4 Bus Interface .....	497
III.5 Interrupt Control .....	500
III.6 Oscillation Circuit .....	502
III.7 Reset .....	502
III.8 Low Power Dissipation Mode .....	505
III.9 Software .....	507
III.10 Others .....	508
APPENDIX IV. THE DIFFERENCES BETWEEN HD63701X0 and HD6301X0 ...	509
APPENDIX V. PROGRAM DEVELOPMENT PROCEDURE AND SUPPORT SYSTEM .....	510





# Section 1. Overview

The HD6301X0, HD6303X, and the HD63701X0 are high-performance CMOS, 8-bit, single-chip microcomputer units (MCU s) which are object-code compatible with the HD6301V.

In addition to the CPU, these MPUs contain 192 bytes of RAM, a 16-bit 4-function timer, an 8-bit reloadable timer, a serial communications interface (SCI), and 53 parallel lines. The HD6301X0 has 4k bytes of masked ROM. The HD63701X0 has 4k bytes of EPROM sometimes referred to as programmable ROM or PROM in this handbook. The HD6303X has no ROM. The MPUs' halt and memory ready functions enable them to release external buses and perform low-speed external memory access.

The HD63701X0's programmable ROM is programmed by the same method as the standard 2732A EPROM. It is available in ceramic packages. The ceramic package with window is erasable for use in the debugging development stage.

## 1.1 Features

The HD6301X0, HD6303X, and HD63701X0 provide the following features.

- Instruction set compatible with the HD6301V1
- On-board ROM
  - 4k bytes programmable (HD63701X0)
  - 4k bytes masked (HD6301X0)
- 192 bytes RAM
- 53 parallel I/O lines
  - 24 common I/O lines (ports 2, 3, and 6)
  - 21 output only lines (ports 1, 4, and 7)
  - 8 input only lines (port 5)
- Darlington transistor direct drive lines (ports 2 and 6)
- 16-bit programmable timer
  - 1 input capture register
  - 1 free-running counter
  - 2 output compare registers
- 8-bit reloadable counter
  - External event counter
  - Square-wave generator

- Serial communications interface (SCI)
  - Asynchronous mode/clocked synchronous mode
  - 3 transmit formats (asynchronous mode)
  - 6 clock sources
- Memory-ready function for low-speed memory access
- Halt function
- Error detection function (address trap, opcode trap)
- Interrupts
  - 3 external
  - 7 internal
- MCU operation modes
  - Mode 1: expanded mode (internal ROM inhibited)
  - Mode 2: expanded mode (internal ROM valid)
  - Mode 3: single chip mode
- PROM mode (HD63701X0)
- Address space up to 65k bytes
- Low power modes
  - Sleep mode
  - Standby mode
- Minimum instruction time 0.5  $\mu$ s ( $f = 2.0$  MHz)

## 1.2 Block Diagrams

Figures 1-1, 1-2, and 1-3 are block diagrams for HD6301X0, HD6303X, and HD63701X0 respectively.

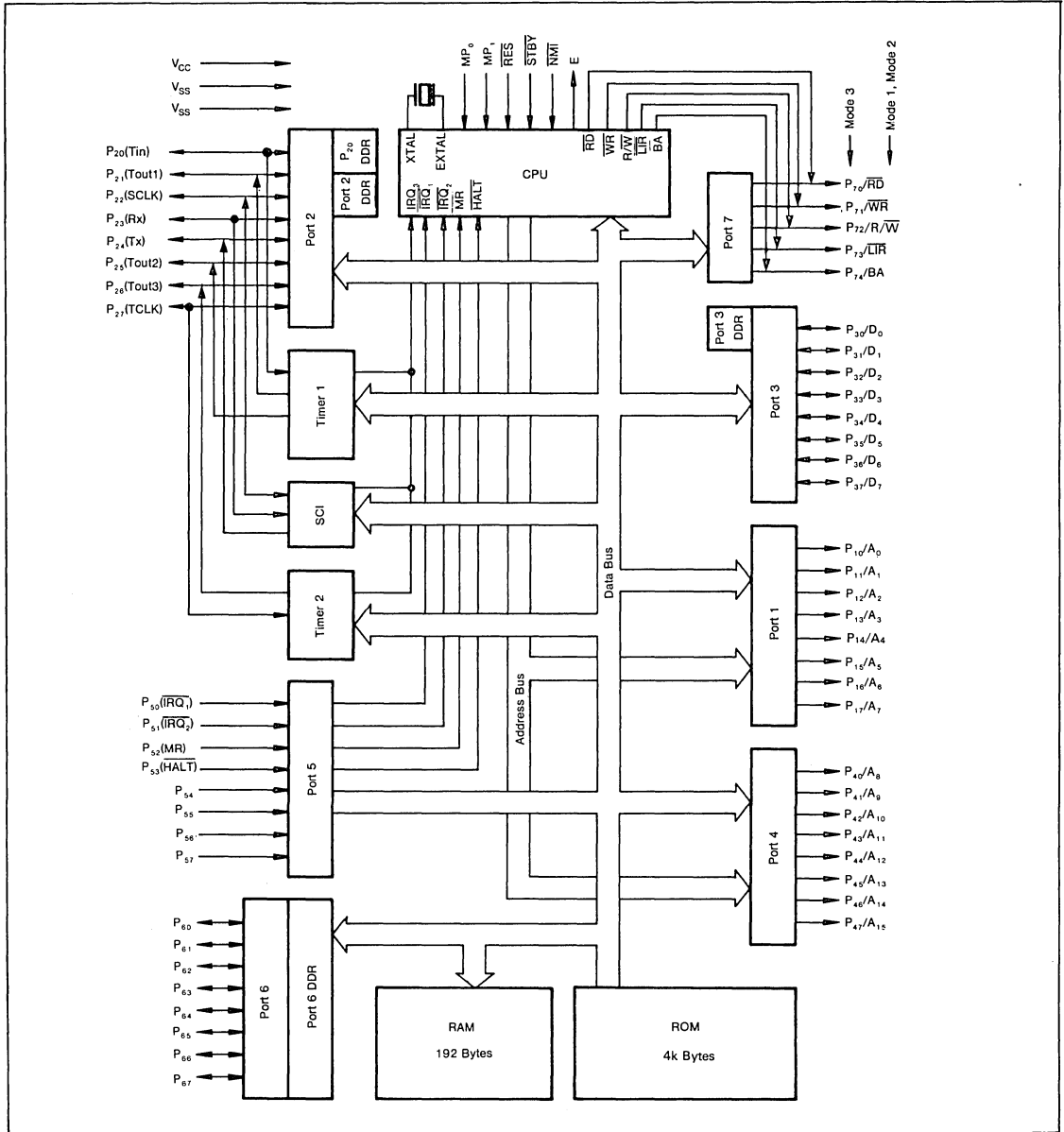


Figure 1-1. HD6301X0 Block Diagram

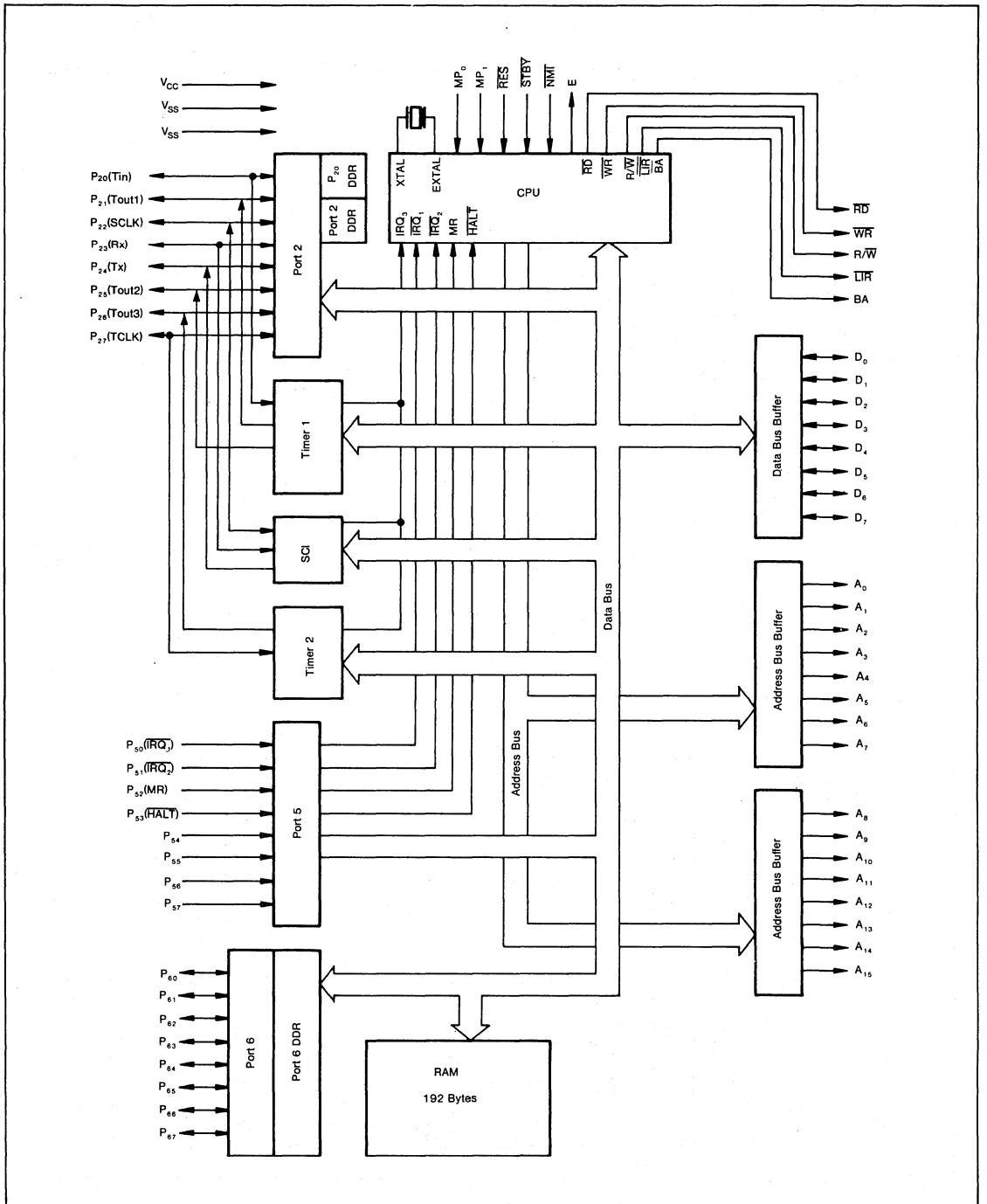


Figure 1-2. HD6303X Block Diagram

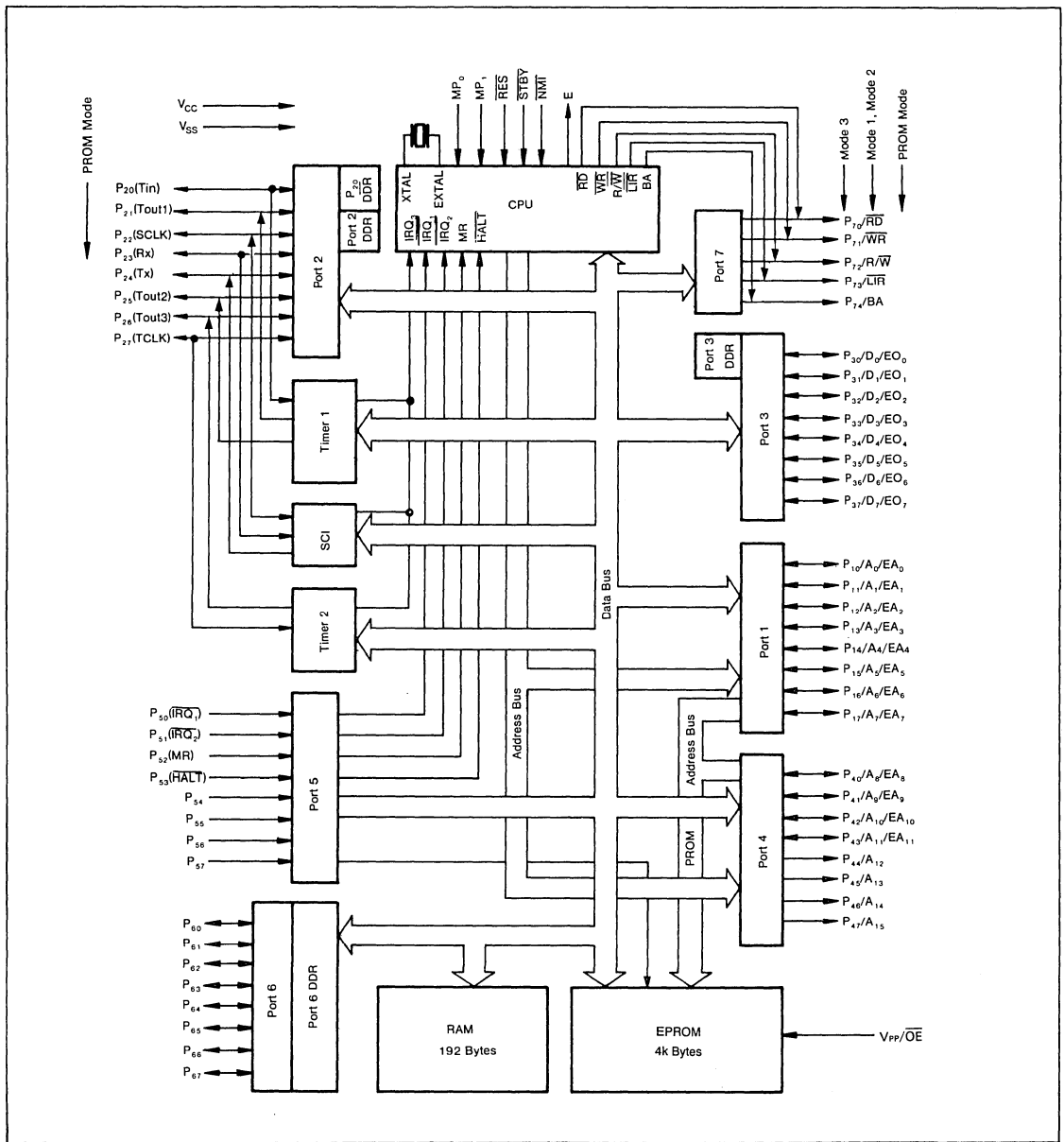


Figure 1-3. HD63701X0 Block Diagram

### 1.3 Pin Description

Figure 1-4 shows the pin arrangements for the various packages.

Table 1-1 lists pin functions for the HD6301X0, HD6303X, and the HD63701X0 in modes 1, 2, and 3.

Table 1-2 lists pin functions for the HD63701X0 in PROM mode. For further pin description, see 2.3 Functional Pin Description, and 2.4 Ports.

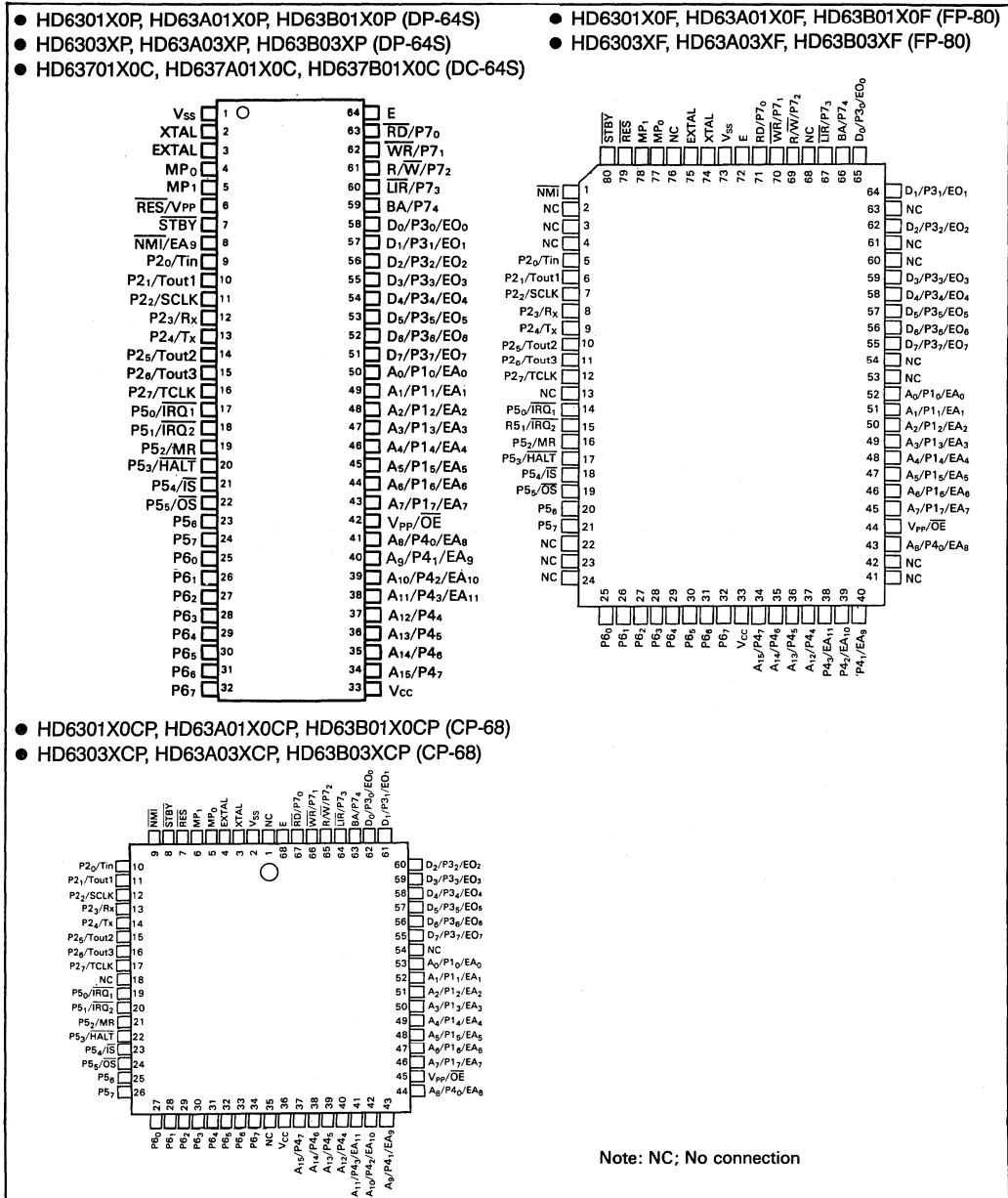


Figure 1-4. Pin Arrangement



Table 1-1. Pin Functions

Number			Name	Function	
DP-64S	FP-80	CP-68			
1	73	2	V <sub>SS</sub>	Ground	
2, 3	74,75	3,4	XTAL, EXTAL	Crystal connections. Connect external clock to EXTAL	
4, 5	77,78	5,6	MP <sub>0</sub> , MP <sub>1</sub>	Operation mode	
6	79	7	$\overline{\text{RES}}$	Reset input	
7	80	8	$\overline{\text{STBY}}$	Standby input	
8	1	9	$\overline{\text{NMI}}$	Nonmaskable interrupt	
9	5	10	Tin/P2 <sub>0</sub> *	Timer 1 capture input	Port 2
10	6	11	Tout1/P2 <sub>1</sub> *	Timer 1 OCR1 output	
11	7	12	SCLK/P2 <sub>2</sub> *	SCI clock input or output	
12	8	13	Rx/P2 <sub>3</sub> *	SCI receive input	
13	9	14	Tx/P2 <sub>4</sub> *	SCI transmit input	
14	10	15	Tout2/P2 <sub>5</sub> *	Timer 1 OCR2 output	
15	11	16	Tout3/P2 <sub>6</sub> *	Timer 2 output	
16	12	17	TCLK/P2 <sub>7</sub> *	Timer 2 external clock input	
17,18	14,15	19,20	$\overline{\text{IRQ}}_1/\text{P5}_0$ , $\overline{\text{IRQ}}_2/\text{P5}_1$	Level-detect interrupt inputs	Port 5
19	16	21	MR/P5 <sub>2</sub> *	Memory ready input	
20	17	22	$\overline{\text{HALT}}/\text{P5}_3$ *	Halt request input	
21-24	18-21	23-26	P5 <sub>4</sub> -P5 <sub>7</sub>		
25-32	25-32	27-34	P6 <sub>0</sub> -P6 <sub>7</sub>	Port 6	
33	33	36	V <sub>CC</sub>	Power supply	
34-41	34-40, 37-44 43		A <sub>15</sub> /P4 <sub>7</sub> -* A <sub>8</sub> /P4 <sub>0</sub> *	Address bus, bits15-8	Port 4
42	44	45	$\overline{\text{V}}_{\text{SS}}$ *	Ground	

\*Mode 1 or Mode 2/Mode 3



Table 1-1. Pin Functions (continued)

Number			Name	Function	
DP-64S	FP-80	CP-68			
43-50	45-52	46-53	A <sub>7</sub> /P <sub>17</sub> * A <sub>0</sub> /P <sub>10</sub> *	Address bus, bits 7-0	Port 1
51-58	55-59, 55-62, 62,64, 65		D <sub>7</sub> /P <sub>37</sub> * D <sub>0</sub> /P <sub>30</sub> *	Data bus	Port 3
59	66	63	BA/P <sub>74</sub> *	Bus available output	Port 7
60	67	64	$\overline{\text{LIR}}/P_{73}$ *	Opcode fetch cycle output	
61	69	65	R/ $\overline{\text{W}}$ /P <sub>72</sub> *	Read/write output	
62	70	66	$\overline{\text{WR}}/P_{71}$ *	Write cycle output	
63	71	67	$\overline{\text{RD}}/P_{70}$ *	Read cycle output	
64	72	68	E	External clock output	

\*Mode 1 or Mode 2/Mode 3

Table 1-2. Pin Functions for HD63701X0 PROM Mode

Number	Name	Function
DP-64S		
7	$\overline{\text{STBY}}$	PROM mode input
38-41	EA <sub>11</sub> -EA <sub>8</sub>	Address input bus, bits 11-8
42	V <sub>pp</sub> / $\overline{\text{OE}}$	Programming power supply
43-50	EA <sub>7</sub> -EA <sub>0</sub>	Address input bus, bits 7-0
51-58	EO <sub>7</sub> -EO <sub>0</sub>	Data input bus

Note: Ground all other HD63701X0 pins in PROM mode.

Table 1-3. Relationship of HD6301X0, HD6303X, and HD63701X0 Operating Modes

Device Type	Mode			EPROM
	1	2	3	
HD6301X0	X	X	X	
HD6303X	X			
HD63701X0	X	X	X	X

# Section 2. Internal Architecture and Operation

## 2.1 Operation Modes

The HD6301X0 and HD63701X0 operate in three MCU modes. The HD63701X0 also operates in PROM mode. The HD6303X only operates in MCU mode 1. The mode program pins MP<sub>0</sub> and MP<sub>1</sub>, and the  $\overline{\text{STBY}}$  pin select the mode (table 2-1).

- MCU 1 (expanded): external memory access enabled, internal ROM disabled
- MCU 2 (expanded): external memory access enabled, internal ROM enabled
- MCU 3 (single-chip): external memory access disabled
- PROM programming: MCU disabled, PROM programming enabled

Table 2-1. Mode Selection

MP <sub>1</sub>	MP <sub>0</sub>	$\overline{\text{STBY}}$	ROM	RAM	Interrupt Vector	Operation Mode
Low	High	X	External	Internal	External	MCU 1 (expanded)
High	Low	X	Internal	Internal	Internal	MCU 2 (expanded)
High	High	X	Internal	Internal	Internal	MCU 3 (single-chip)
Low	Low	Low	Internal	X	X	PROM programming

Note: X = Don't care

### 2.1.1 MCU Mode 1 (Expanded)

In MCU mode 1, port 3 is the data bus, port 1 is the lower address bus, and port 4 is the upper address bus. They can directly interface with HD6800 buses. Port 7 supplies signals such as R/W. See table 2-2. In mode 1, the ROM is disabled and the external address space is 65k bytes (figure 2-1). Since the HD6303X has no internal ROM, it only operates in mode 1.

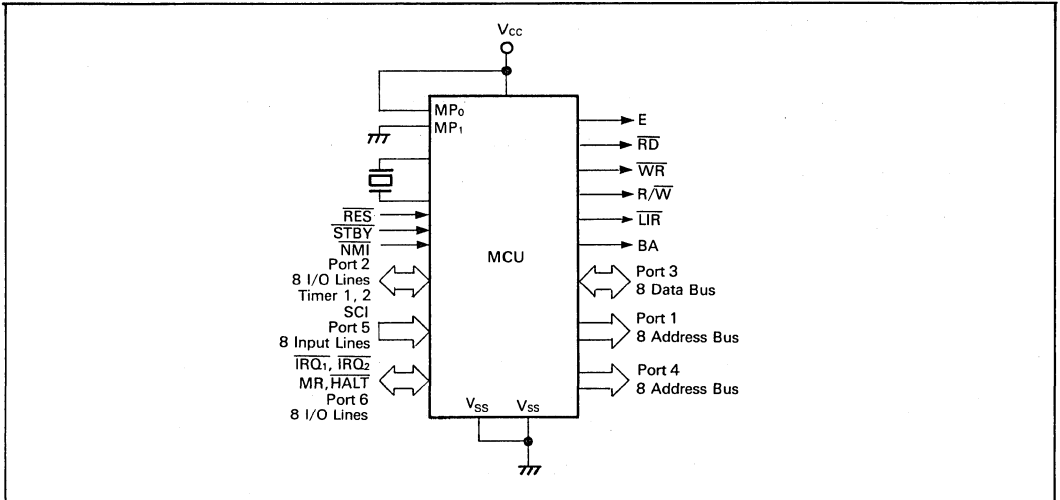


Figure 2-1. MCU Mode 1

### 2.1.2 MCU Mode 2 (Expanded)

MCU mode 2 is the same as mode 1, except that the ROM is enabled. The external address space is 61k bytes (figure 2-2).

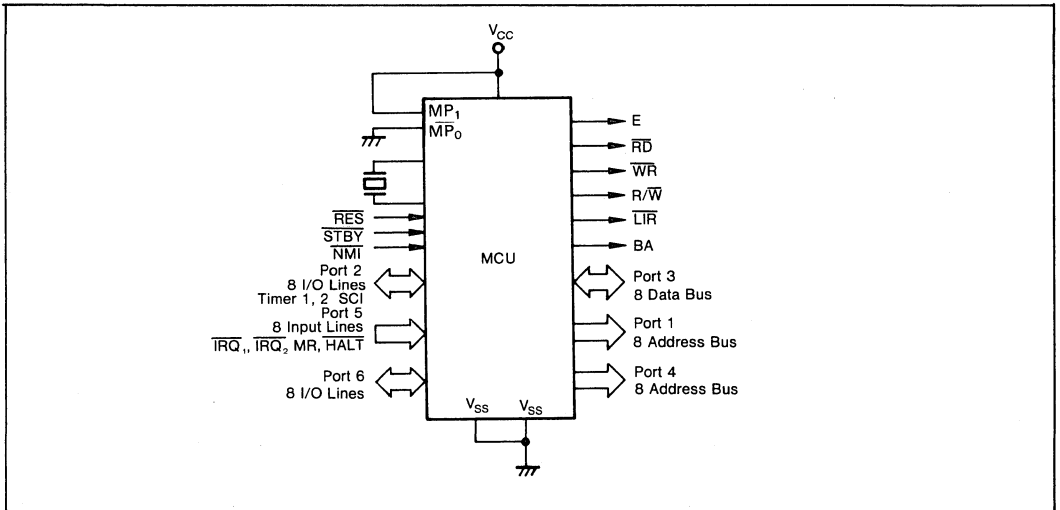


Figure 2-2. MCU Mode 2

### 2.1.3 MCU Mode 3 (Single-Chip)

In MCU mode 3, all ports are I/O ports. There is no interface to external buses (figure 2-3).

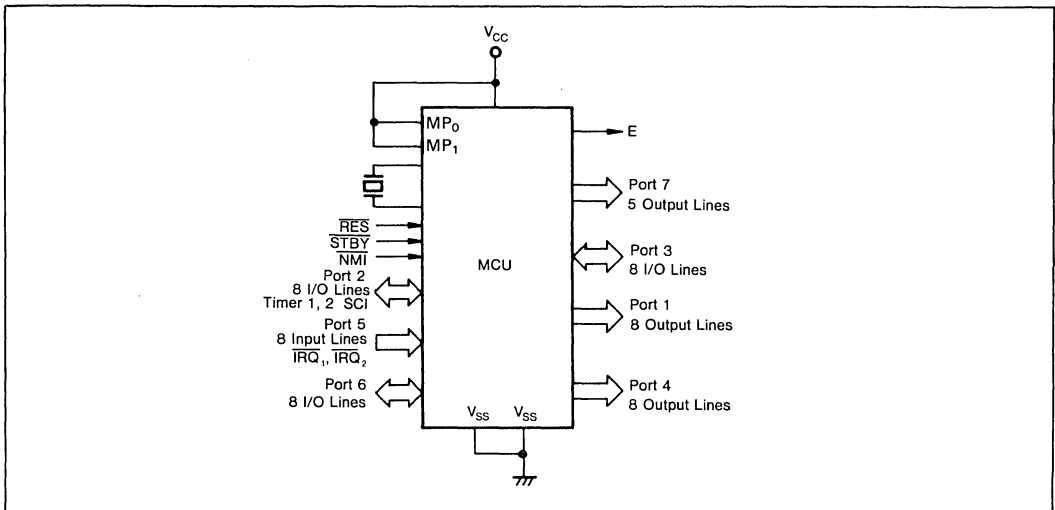


Figure 2-3. MCU Mode 3

### 2.1.4 PROM Mode

In PROM mode, the HD63701X0's EPROM can be programmed (figure 2-4, table 2-2). Refer to Section 7, Programmable ROM, for details.

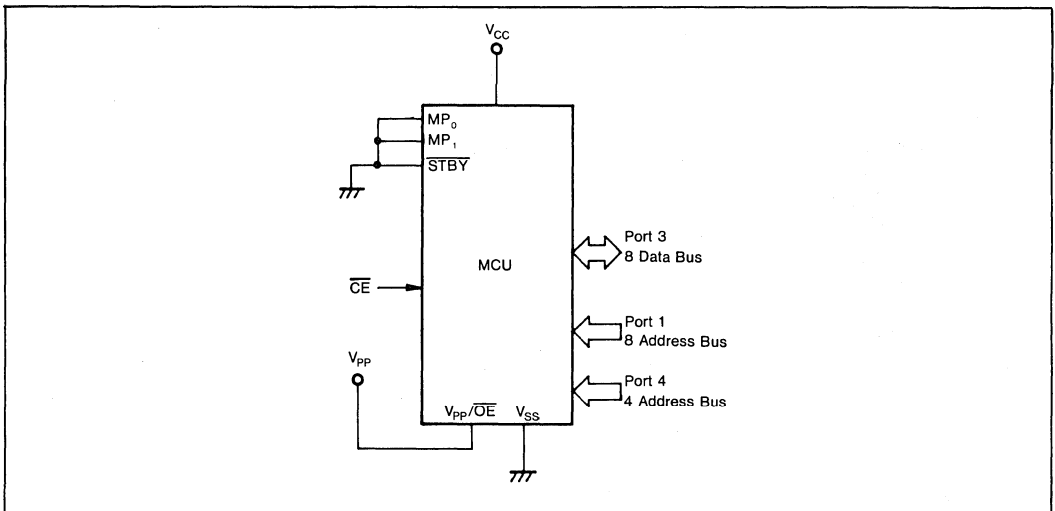


Figure 2-4. PROM Programming Mode

Table 2-2. Port Signals

Port	MCU Mode 1	MCU Mode 2	MCU Mode 3	PROM Mode
1	Address bus (A <sub>0</sub> -A <sub>7</sub> )	Address bus ( A <sub>0</sub> -A <sub>7</sub> )	Output port	Address bus (EA <sub>0</sub> -EA <sub>7</sub> )
2	I/O port	I/O port	I/O port	Connect to ground
3	Data bus (D <sub>7</sub> -D <sub>0</sub> ).	Data bus (D <sub>7</sub> -D <sub>0</sub> )	I/O port	Data bus (EO <sub>7</sub> -EO <sub>0</sub> )
4	Address bus (A <sub>8</sub> -A <sub>15</sub> )	Address bus ( A <sub>8</sub> -A <sub>15</sub> )	Output port	Address bus (EA <sub>8</sub> -EA <sub>11</sub> , pins P4 <sub>0</sub> -P4 <sub>3</sub> only)
5	Input port	Input port	Input port	CĒ (P5 <sub>7</sub> only)
6	I/O port	I/O port	I/O port	Connect to ground
7	$\overline{RD}$ , $\overline{WR}$ , $\overline{R/W}$ , $\overline{LIR}$ , BA	$\overline{RD}$ , $\overline{WR}$ , $\overline{R/W}$ , $\overline{LIR}$ , BA	Output port	Connect to ground

## 2.2 Memory Map

The HD6301X0, HD6303X, and HD63701X0 can access up to 65k bytes of external memory, depending on the operating mode. Figure 2-5 shows a memory map for each mode. The first 32 locations of each map, from \$00 to \$1F, are reserved for the MCU's internal register area (table 2-3).

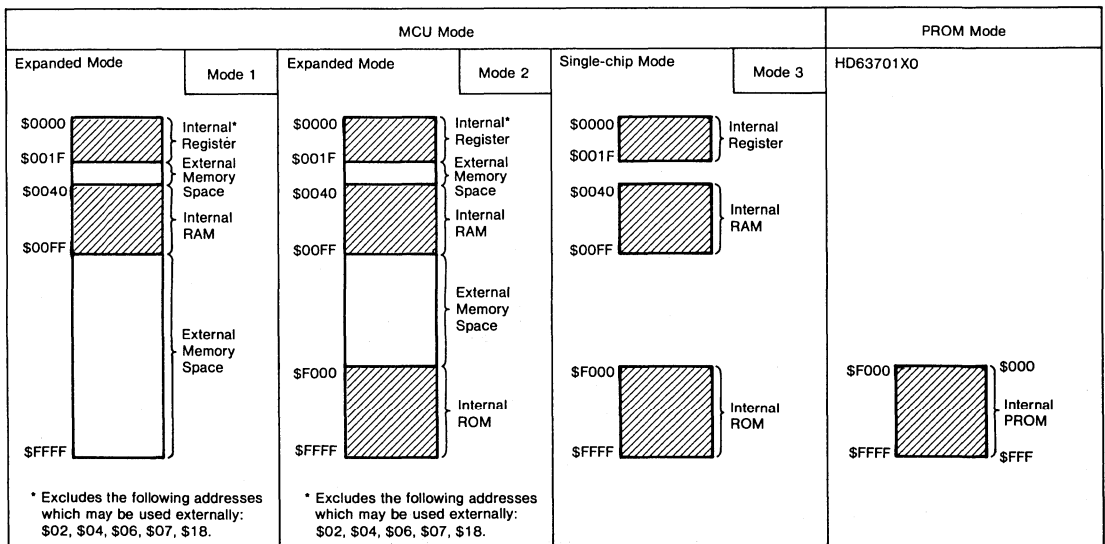


Figure 2-5. Memory Maps

Table 2-3. Internal Register Area

Address	Register	R/W	State at RESET
00			
01	Port 2 data direction register	W	\$FC
02	Port 1	R/W	Undefined
03	Port 2	R/W	Undefined
04	Port 3 data direction register	W	\$FE
05			
06	Port 3	R/W	Undefined
07	Port 4	R/W	Undefined
08	Timing control/status register 1	R/W	\$00
09	Free-running counter (upper byte)	R/W	\$00
0A	Free-running counter (lower byte)	R/W	\$00
0B	Output compare register 1 (upper byte)	R/W	\$FF
0C	Output compare register 1 (lower byte)	R/W	\$FF
0D	Input capture register (upper byte)	R	\$00
0E	Input capture register (lower byte)	R	\$00
0F	Timer control/status register 2	R/W	\$10
10	Rate, mode control register	R/W	\$00
11	Tx/Rx control status register	R/W	\$20
12	Receive data register	R	\$00
13	Transmit data register	W	\$00
14	RAM/port 5 control register	R/W	\$7C or \$FC
15	Port 5	R	
16	Port 6 data direction register	W	\$00

Table 2-3. Internal Register Area (continued)

Address	Register	R/W	State at RESET
17	Port 6	R/W	Undefined
18	Port 7	R/W	Undefined
19	Output capture register 2 (upper byte)	R/W	\$FF
1A	Output capture register 2 (lower byte)	R/W	\$FF
1B	Timer control/status register 3	R/W	\$20
1C	Timer constant register	W	\$FF
1D	Timer 2 upcounter	R/W	\$00
1E	Reserved		
1F	Reserved		

## 2.3 Functional Pin Description

### 2.3.1 Power ( $V_{CC}$ , $V_{SS}$ )

$V_{CC}$  and  $V_{SS}$  are the power supply pins. Apply  $+5\text{ V} \pm 10\%$  to  $V_{CC}$ . Tie  $V_{SS}$  to ground.

### 2.3.2 Clock (XTAL, EXTAL)

XTAL and EXTAL connect to an AT-cut parallel resonant crystal. The chip has a divide-by-four circuit. For example, if a 4 MHz crystal is used, the system clock will be 1 MHz.

Figure 2-6 is an example of the crystal oscillator connection. The crystal and  $C_{L1}$  and  $C_{L2}$  should be located as close as possible to the XTAL and EXTAL pins. No line must cross the lines between the crystal oscillator and the XTAL and EXTAL pins.

The EXTAL pin can be driven by an external clock with a 45% to 55% duty cycle. The LSI divides the external clock frequency by four. The external clock should therefore be less than four times the maximum clock frequency. When using an external clock, the XTAL pin should be left open.

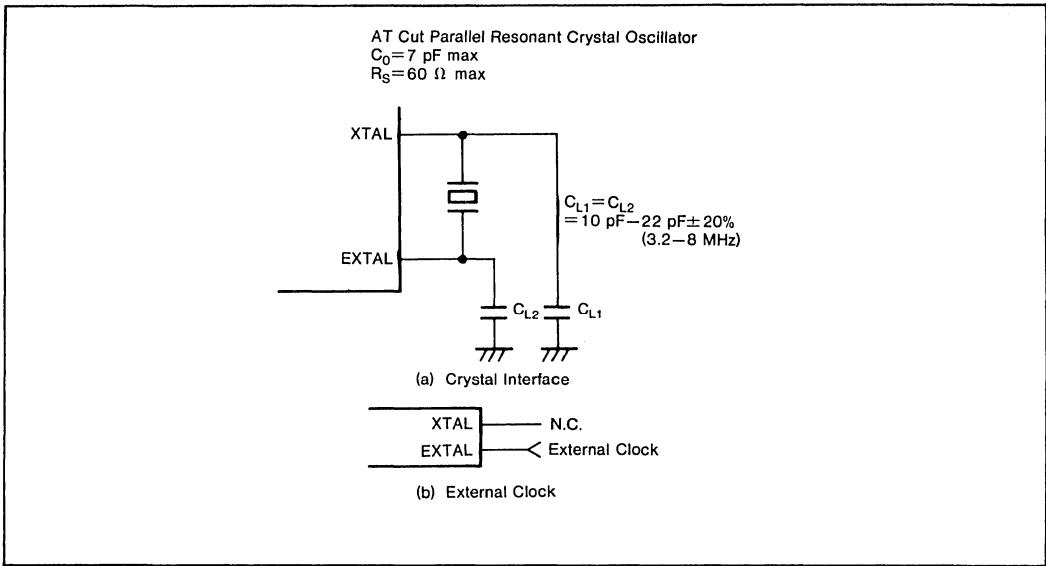


Figure 2-6. Recommended Crystal Oscillator Connection

### 2.3.3 Standby ( $\overline{\text{STBY}}$ )

The  $\overline{\text{STBY}}$  pin puts the MCU in standby mode. When  $\overline{\text{STBY}}$  is low, the oscillation stops, and the internal clock is stabilized to put the MCU in a reset condition. To retain the contents of RAM during standby, write 0 to the RAM enable bit (RAME). RAME is bit 6 of the RAM/port 5 control register at address \$0014. RAM is disabled, and its contents are sustained. Refer to 3.5 Low Power Dissipation Mode for details on the standby mode.

When  $\overline{\text{STBY}}$ ,  $\text{MP}_0$ , and  $\text{MP}_1$  are low, the MCU is in PROM mode. Refer to Section 7, Programmable ROM for details.

### 2.3.4 Reset ( $\overline{\text{RES}}$ )

This pin resets the MCU's internal state and provides a startup procedure. The  $\overline{\text{RES}}$  input must be held low for at least 20 ms during power-on.

The CPU registers accumulator, index register, stack pointer, condition code register except for mask bit, RAM, and the data registers of the ports are not initialized during reset, so their contents are undefined.

### 2.3.5 External Clock (E)

E provides a TTL-compatible system clock to external circuits. Its frequency is one-fourth that of the crystal oscillator or external clock. E can drive one TTL load and 90 pF.



### 2.3.6 Nonmaskable Interrupt ( $\overline{\text{NMI}}$ )

When CPU detects a falling edge at the  $\overline{\text{NMI}}$  input, it begins the internal nonmaskable interrupt sequence. The instruction being executed when the  $\overline{\text{NMI}}$  is detected will proceed to completion. The interrupt mask bit of the condition code register does not affect the nonmaskable interrupt.

In response to an  $\overline{\text{NMI}}$  interrupt, the contents of the program counter, index register, accumulators, and condition code register will be saved onto the stack. After they are saved, a vector is fetched from \$FFFC and \$FFFD to the program counter, and the nonmaskable interrupt service routine starts.

Note: After reset, the stack pointer should be initialized to an appropriate memory location before any  $\overline{\text{NMI}}$  input.

### 2.3.7 Interrupt Requests ( $\overline{\text{IRQ}}_1$ , $\overline{\text{IRQ}}_2$ )

The interrupt requests are level-sensitive inputs which request an internal interrupt sequence from the CPU.

### 2.3.8 Mode Program ( $\text{MP}_0$ , $\text{MP}_1$ )

These pins determine the operation mode. Refer to 2.1 Operation Mode for details.

Note: The following signals,  $\overline{\text{RD}}$ ,  $\overline{\text{WR}}$ ,  $\text{R}/\overline{\text{W}}$ ,  $\overline{\text{LIR}}$ ,  $\text{MR}$ ,  $\overline{\text{HALT}}$ , and  $\text{BA}$ , are only used in modes 1 and 2.

### 2.3.9 Read/Write ( $\text{R}/\overline{\text{W}}$ ; $\text{P7}_2$ )

The read/write signal shows whether the MCU is in read ( $\text{R}/\overline{\text{W}}$  high) or write ( $\text{R}/\overline{\text{W}}$  low) state to the peripheral or memory devices. It is usually high, in read state.  $\text{R}/\overline{\text{W}}$  can drive one TTL load and 30 pF.

### 2.3.10 Read and Write ( $\overline{\text{RD}}$ ; $\text{P7}_0$ , $\overline{\text{WR}}$ ; $\text{P7}_1$ )

The read and write outputs show active low outputs to peripherals or memories when the CPU is reading or writing. This enables the CPU to access LSI peripherals with  $\overline{\text{RD}}$  and  $\overline{\text{WR}}$  inputs easily. These pins can drive one TTL load and 30 pF.

### 2.3.11 Load Instruction Register ( $\overline{\text{LIR}}$ ; $\text{P7}_3$ )

The  $\overline{\text{LIR}}$  output low shows that the instruction opcode is on the data bus.  $\overline{\text{LIR}}$  can drive one TTL load and 30 pF.

### 2.3.12 Memory Ready (MR; P5<sub>2</sub>)

The memory ready control input lengthens the system clock's high period to allow access to low-speed memory. When MR is high, the system clock operates normally. But when MR is low, the high period will be lengthened depending on its low time in integral multiples of its cycle time. It can be lengthened up to 9  $\mu$ s.

During internal address or invalid memory access, MR is prohibited internally from decreasing operation speed. Even in the halt state, MR can lengthen the high period of the system clock to allow peripheral devices to access low-speed memories. MR is also used as P5<sub>2</sub>. The function is chosen by the enable bit in the RAM/port 5 control register (bit 2) at \$0014. See 2.5 RAM/Port 5 Control Register for details.

### 2.3.13 Halt ( $\overline{\text{HALT}}$ ; P5<sub>3</sub>)

The halt control input stops instruction execution and releases the buses. When  $\overline{\text{HALT}}$  switches low, the CPU finishes the current instruction, then stops and enters the halt state. When entering the halt state, the CPU sets BA (P7<sub>4</sub>) high, and sets the address bus, data bus,  $\overline{\text{RD}}$ ,  $\overline{\text{WR}}$ , and R/W to high impedance. When an interrupt occurs in the halt state, the CPU cancels the halt, and executes the interrupt service routine.

Note: When the CPU is in the interrupt wait state, executing the WAI instruction,  $\overline{\text{HALT}}$  should be held high. If halt turns low, the CPU may fetch the incorrect vector after releasing the halt state (figure 2-7). If a halt is expected, a loop should be used instead of WAI (figure 2-8).

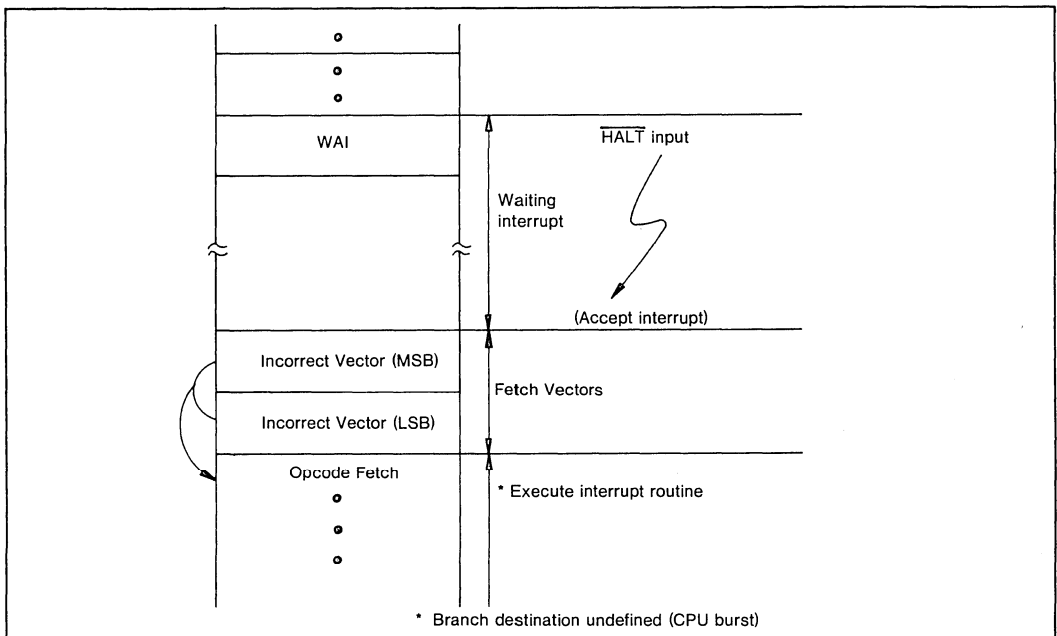


Figure 2-7.  $\overline{\text{HALT}}$  After WAI



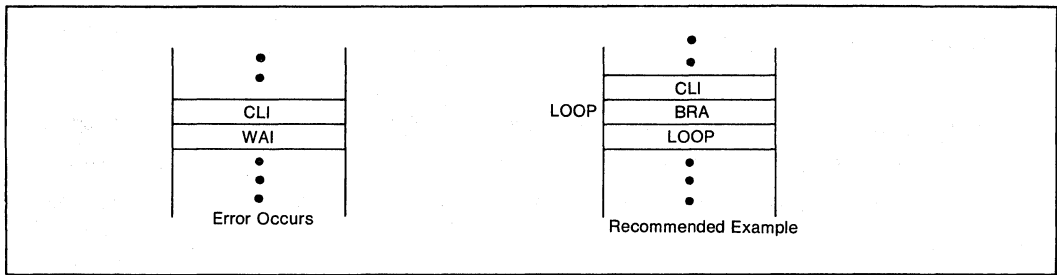


Figure 2-8. Branch Replacement for WAI

### 2.3.14 Bus Available (BA; P7<sub>4</sub>)

The bus available output control signal goes high when the CPU accepts  $\overline{\text{HALT}}$  and releases the buses. It is normally low. The HD6800 and HD6802 bring BA high and release the buses at WAI execution, but the HD6301X0 and HD63701X0 don't. But if  $\overline{\text{HALT}}$  goes low when the CPU is in the interrupt wait state after having executed a WAI, the CPU sets BA high and releases the buses. When  $\overline{\text{HALT}}$  goes high, the CPU returns to the interrupt wait state.

The following signals,  $\overline{\text{CE}}$  and  $V_{\text{PP}}/\overline{\text{OE}}$ , are only used in the HD63701X0 PROM programming mode.

### 2.3.15 Chip Enable ( $\overline{\text{CE}}$ ; P5<sub>7</sub>)

The chip enable input enables PROM programming and verifying. When this signal is low, the PROM is enabled. The PROM cannot be programmed or verified with  $\overline{\text{CE}}$  high.

### 2.3.16 Program Voltage/Output Enable ( $V_{\text{PP}}/\overline{\text{OE}}$ )

The program voltage/output enable pin is the input for the program voltage for programming the PROM, and the control for data verification output.

To program data from port 3 (EO<sub>0</sub>-EO<sub>7</sub>) into the PROM, apply 21 V ± 0.5 V to  $V_{\text{PP}}$  while holding  $\overline{\text{CE}}$  low. Set the PROM address on port 1 and 4 (EA<sub>0</sub>-EA<sub>11</sub>). To verify, bring the  $\overline{\text{OE}}$  pin low. The data addressed by EA<sub>0</sub>-EA<sub>11</sub> will be output at EO<sub>0</sub>-EO<sub>7</sub>. When  $\overline{\text{OE}}$  is high, port 3 will be high impedance. In the MCU modes, connect this pin to  $V_{\text{SS}}$ .

## 2.4 Ports

The HD63701X0 provides seven ports (six 8-bit ports and a 5-bit port). Some pins have other uses, as shown in table 2-2. Table 2-5 shows the addresses of the ports and their data direction registers. Figure 2-9 shows block diagrams of each port. Table 2-6 shows the state of each port at reset.

Table 2-5. Port and Data Direction Register Address

Port	Port Address	Data Direction Register
1	\$0002	
2	\$0003	\$0001
3	\$0006	\$0004
4	\$0007	
5	\$0015	
6	\$0017	\$0016
7	\$0018	

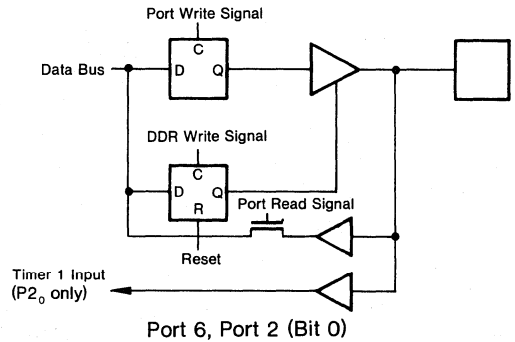
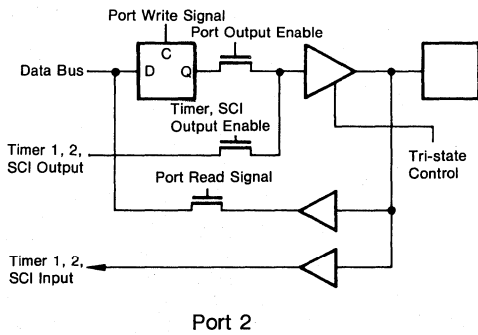
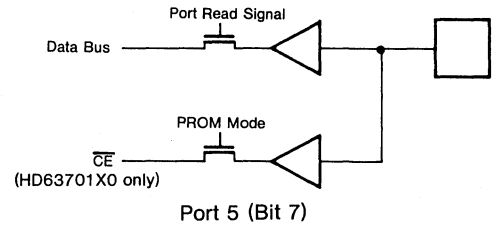
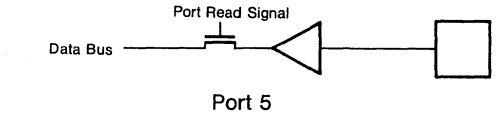
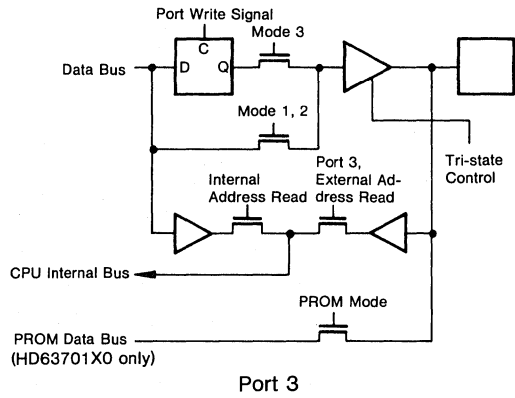
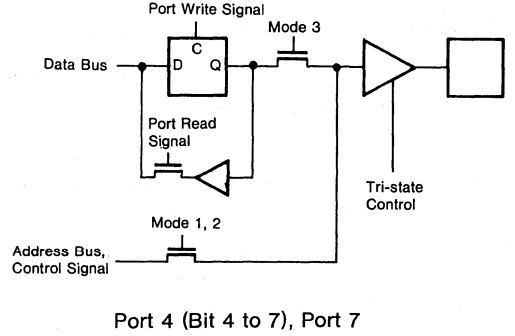
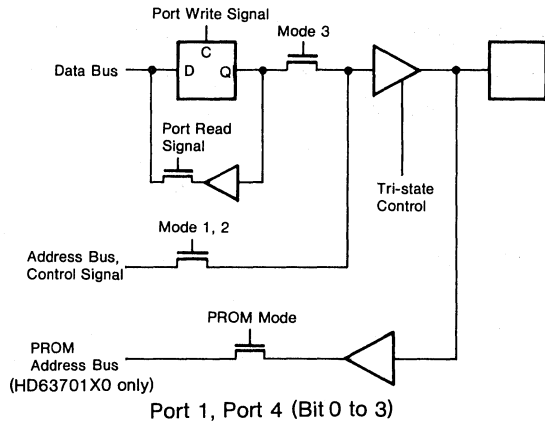


Figure 2-9. Port Block Diagrams

Table 2-6. Port at Reset (Modes 1 and 2)

Port	State at Reset
1 (A <sub>0</sub> -A <sub>7</sub> )	High
2	High impedance
3 (D <sub>0</sub> -D <sub>7</sub> )	High impedance
4 (A <sub>8</sub> -A <sub>15</sub> )	High
5	High impedance
6	High impedance
7	$\overline{RD}$ , $\overline{WR}$ , $R/W$ , $\overline{LIR}$ = High BA = Low

Note: All ports are high impedance after reset in mode 3.

### 2.4.1 Port 1

In the MCU modes, port 1 is an 8-bit output port. In mode 3 (single-chip), port 1 is high impedance during reset, and stays high impedance after reset is released. When the CPU writes to the port 1 data register, the data written will appear at port 1. Once port 1 is in the output state, it operates as an output until reset. The CPU can read the port 1 data register for bit manipulation instructions.

In modes 1 and 2, port 1 is used for the lower byte of the address bus. Port 1 can drive 1 TTL load and 30 pF.

In the PROM mode, port 1 is the lower byte of the PROM address (EA<sub>0</sub>-EA<sub>7</sub>).

### 2.4.2 Port 2

Port 2 is an 8-bit input/output port. The port 2 data direction register (DDR) controls the I/O state (figure 2-10). Bit 0 controls the I/O direction of P2<sub>0</sub>, and bit 1 controls the direction of P2<sub>1</sub>-P2<sub>7</sub>. A 1 specifies input, 0 specifies output.

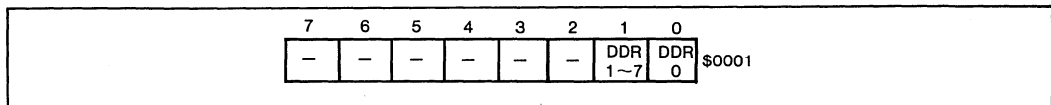


Figure 2-10. Port 2 Data Direction Register

Port 2 is also used as I/O pins for the timers and SCI. In this case, port 2 pins except P2<sub>0</sub> automatically become inputs or outputs regardless of the data direction register's value.

A reset clears the port 2 DDR and configures port 2 as an input port. Port 2 can drive 1 TTL load and 30 pF. In addition, it can produce 1 mA at  $V_{OUT} = 1.5\text{ V}$  to directly drive the base of Darlington transistors.

When a write-only register like a DDR is read by the MCU, \$FF always appears on the internal data bus. Whenever the MCU performs an arithmetic or logic operation between memory, and a write-only register, the result will be \$FF. AIM, OIM, and EIM instructions cannot be applied to the DDR.

### 2.4.3 Port 3

Port 3 is an 8-bit I/O port. The port 3 DDR controls its direction. If bit 0 of the DDR is 1, port 3 is an input port. If it is 0, port 3 is an output (figure 2-11). The DDR is cleared during reset. In modes 1 and 2, port 3 is the data bus (D<sub>0</sub>-D<sub>7</sub>). In the HD63701X0 PROM mode, port 3 is the PROM data bus (EO<sub>0</sub>-EO<sub>7</sub>). In the PROM mode, port 3's direction is controlled by  $\overline{OE}$ , not the DDR. Port 3 can drive 1 TTL load and 90 pF.

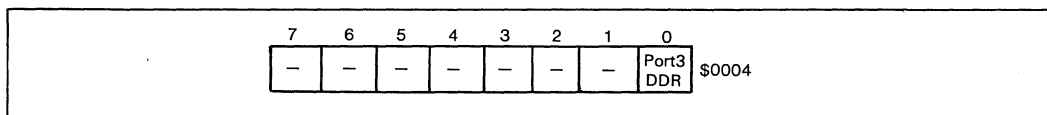


Figure 2-11. Port 3 Data Direction Register

### 2.4.4 Port 4

Port 4 is an 8-bit output-only port like port 1. In modes 1 and 2, it outputs the upper byte of the address (A<sub>8</sub>-A<sub>15</sub>). In the HD63701X0 PROM mode, P<sub>40</sub>-P<sub>43</sub> are used as the upper PROM address bits (EA<sub>8</sub>-EA<sub>11</sub>).

### 2.4.5 Port 5

Port 5 is an 8-bit input-only port. The lower four bits (P<sub>50</sub>-P<sub>53</sub>) are also used for interrupt, MR and  $\overline{HALT}$  input. In the HD63701X0 PROM mode, P<sub>57</sub> is used as  $\overline{CE}$  to control the PROM.

### 2.4.6 Port 6

Port 6 is an 8-bit I/O port. Each bit in the port 6 data direction register controls the direction of the corresponding bit of port 6. A 1 specifies input, 0 specifies output. Port 6 can drive 1 TTL load and 30 pF. In addition, it can produce 1 mA at  $V_{OUT} = 1.5\text{ V}$  to directly drive the base of Darlington transistors. A reset clears the port 6 DDR.

## 2.4.7 Port 7

Port 7 is a 5-bit output port. In mode 3, port 7 is high impedance during and after reset. When the CPU writes to the port 7 register, the data will appear at port 7. Once port 7 is in the output state, it will be an output until reset. The CPU can read the port 7 data register for bit manipulation instructions. Bits 5-7 will be read as 1.

In modes 1 and 2, port 7 is used for control signals ( $\overline{RD}$ ,  $\overline{WR}$ ,  $R\overline{W}$ ,  $\overline{LIR}$ , and BA). Port 7 can drive 1 TTL load and 30 pF.

## 2.5 RAM/Port 5 Control Register

The control register (figure 2-15) located at \$0014 controls on-chip RAM and port 5.

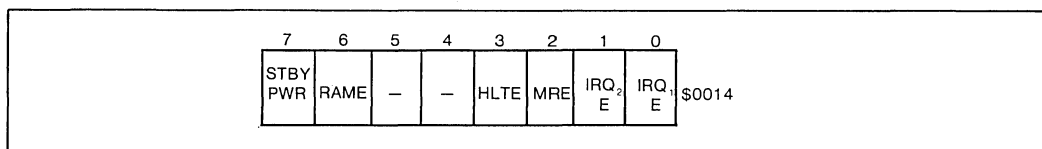


Figure 2-15. RAM/Port 5 Control Register

### 2.5.1 $\overline{IRQ}$ Enable (IRQ<sub>1</sub>E, IRQ<sub>2</sub>E)

When IRQ<sub>1</sub>E and IRQ<sub>2</sub>E are 1, P5<sub>0</sub> and P5<sub>1</sub> are interrupt pins  $\overline{IRQ}_1$  and  $\overline{IRQ}_2$ . When these bits are 0, the CPU doesn't accept external interrupts. External interrupts won't cancel the sleep state. These bits are 0 after reset. Bits 0, 1.

### 2.5.2 Memory Read Enable (MRE)

When MRE is 1, P5<sub>2</sub> is used as the memory ready (MR) signal. When it is 0, the MR signal is inhibited. In mode 3, the MR signal is inhibited regardless of MRE. MRE becomes 1 after reset. Bit 2.

### 2.5.3 Halt Enable (HLTE)

When HLTE is 1, P5<sub>3</sub> is used as the HALT input. When 0, the halt function is inhibited. In mode 3, the HALT signal is inhibited regardless of the value of HLTE. HLTE becomes 1 after reset.

**Note:** When using P5<sub>2</sub> and P5<sub>3</sub> for port bits in modes 1 and 2, clear MRE and HLTE after reset. If P5<sub>2</sub> or P5<sub>3</sub> is brought low before MRE or HLTE are cleared, a memory ready or halt will be accepted. Bit 3.



#### **2.5.4 RAM Enable (RAME)**

RAME controls on-chip RAM. When RAME is 0, on-chip RAM is disabled, and the CPU can read from external memory at addresses \$0040-\$00FF in modes 1 and 2. RAME is 1 after reset and on-chip RAM is enabled. RAME should be set to 0 at the beginning of standby mode to protect on-chip RAM. Bit 6.

#### **2.5.5 Standby Power (STBY PWR)**

When  $V_{CC}$  is not provided in standby mode, STBY PWR is cleared. The STBY PWR flag can be read and written by software. If it is set to 1 before standby mode and remains set after returning from standby mode,  $V_{CC}$  has been provided during standby, and on-chip data is valid. Refer to 3.5 Low Power Dissipation Mode. Bit 7.

# Section 3. CPU Function

## 3.1 CPU Registers

The CPU has three 16-bit registers and three 8-bit registers (figure 3-1).

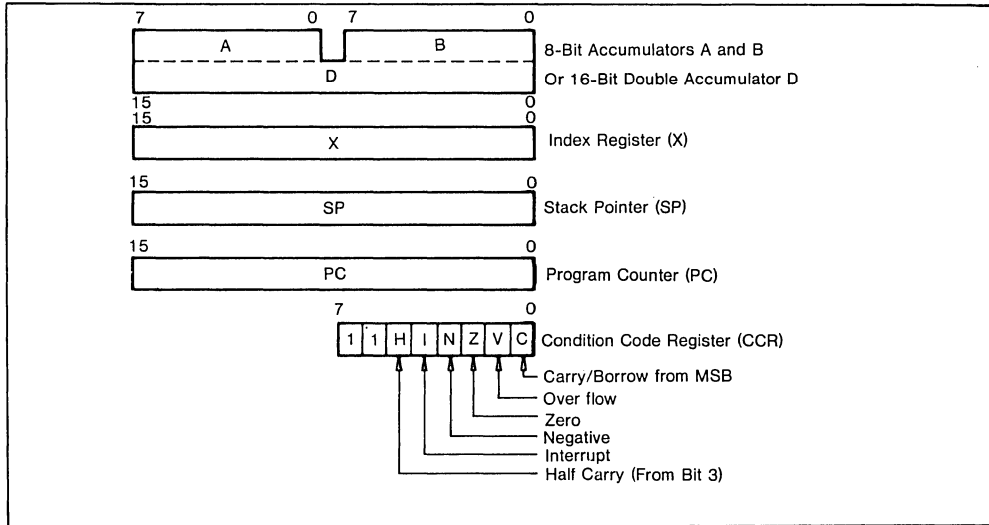


Figure 3-1. CPU Registers

5

### 3.1.1 Accumulators (ACCA, ACCB, ACCD)

Two 8-bit accumulators, ACCA and ACCB, store the result of arithmetic/logic operations and data. When combined, these make up the 16-bit accumulator ACCD used for 16-bit operations. Note that the contents of ACCA and ACCB are destroyed by an ACCD operation.

### 3.1.2 Index Register (IX)

The 16-bit register IX stores 16-bit data for use in indexed addressing or for general purposes.

### 3.1.3 Stack Pointer (SP)

The contents of the 16-bit register SP indicate the address of a stack. SP can also be used as a general-purpose register.

### 3.1.4 Program Counter (PC)

The contents of the 16-bit PC indicate the address of the instruction being executed. Note that software cannot access this register.

### 3.1.5 Condition Code Register (CCR)

The CCR register consists of the carry (C), overflow (V), zero (Z), negative (N), interrupt mask (I), and half-carry (H) bits. After an instruction is executed, the CCR bits change state depending on the result of the operation. They can be tested by conditional branch instructions. The upper two bits of this register are not used.

**Half-Carry (H):** H is set to 1 if a carry at bit 3 or bit 4 occurs during an ADD, ABA, or ADC instruction. It is cleared if no carry occurs.

**Interrupt Mask (I):** When I is set to 1, it disables all maskable interrupts ( $\overline{IRQ_1}$ ,  $\overline{IRQ_2}$ , and  $IRQ_3$ ).

**Negative (N):** N is set to 1 if the MSB of the result of an operation is 1. N is cleared if it is 0.

**Zero (Z):** Z is set to 1 if the result of an operation is zero. Z is cleared if it is not zero.

**Overflow (V):** V is set to 1 if the result of an operation shows a two's complement overflow. It is cleared if there is no overflow.

**Carry (C):** C is set to 1 if a carry or borrow is generated from the MSB. If there is no carry or borrow, it is cleared.

## 3.2 Addressing Modes

The HD6301X0, HD6303X, and HD63701X0 instructions have seven addressing modes.

### 3.2.1 Accumulator Addressing (ACCX)

The instruction addresses an accumulator and ACCA or ACCB is selected. Accumulator addressing instructions take one byte.

### 3.2.2 Immediate Addressing

Immediate addressing places the data in the second byte of an instruction, except LDS and LDX, which use the second and third bytes. An immediate instruction causes the CPU to address this operand. Immediate instructions take 2 or 3 bytes.

### 3.2.3 Direct Addressing

In direct addressing, the second byte of an instruction holds the address where the data is stored. 256

bytes (\$00-\$FF) can be addressed directly. Storing data in this area reduces instruction time, so configuring \$00-\$FF as user's RAM is recommended. Direct addressing instructions take 2 bytes, or 3 bytes for AIM, OIM, EIM, or TIM.

### 3.2.4 Extended Addressing

In extended addressing, the second byte of an instruction holds the upper eight bits of the absolute address of the stored data, and the third byte holds the lower eight bits. Extended addressing instructions take 3 bytes.

### 3.2.5 Indexed Addressing

In indexed addressing, the second byte of the instruction (third byte for AIM, OIM, EIM, or TIM instructions) is added to the lower eight bits of the index register. The carry is added to the upper eight bits of the index register, and the 16-bit sum is the memory location of the data. The modified address is held in the temporary address register, so the index register doesn't change. Indexed addressing instructions take 2 bytes, or 3 bytes for AIM, OIM, EIM, or TIM.

### 3.2.6 Implied Addressing

In implied addressing, the instruction itself specifies the address. For example, the instruction addresses the stack pointer or index register. Implied addressing instructions take 1 byte.

### 3.2.7 Relative Addressing

In relative addressing, the second byte of the instruction and the lower eight bits of the program counter are added. The carry or borrow is added to the upper eight bits of the program counter. Locations from -126 to +129 bytes from the current location can be addressed. Relative addressing instructions take 2 bytes.

### 3.3 Instruction Set

The HD6301X0, HD6303X, and HD63701X0 are object-code upwardly compatible with the HD6801 to use all instructions of the HMCS6800. The instruction time of key instructions has been reduced, improving throughput.

#### 3.3.1 Additional Instructions

Bit manipulation, index register and accumulator exchange, and sleep instructions have also been added to the HD6801 instruction set. AIM, OIM, EOM, and TIM are 3 byte instructions. The first byte is the opcode, second byte is the immediate data, and the third byte is the address modifier.

**AIM:** ANDs the immediate data with the memory contents and stores the result in memory. (M) AND (IMM) → (M).

**OIM:** ORs the immediate data with the memory contents and stores the result in memory. (M) OR (IMM) → (M).

**EIM:** EORs the immediate data with the memory contents and stores the result in memory. (M) EOR (IMM) → (M).

**TIM:** ANDs the immediate data with the memory contents and changes the related flag in the condition code register. (M) AND (IMM).

**XGDX:** Exchanges the contents of the accumulator with the contents of the index register. (ACCD) ↔ (IX).

**SLP:** Puts the MCU into sleep mode. Refer to 3.5 Low Power Dissipation Mode for details.

#### 3.3.2 Instruction Set Summary

Tables 3-1 to 3-5 summarize the instruction set.

- Accumulator and memory manipulation instructions: table 3-1
- Index register and stack manipulation instructions: table 3-2
- Jump and branch instructions: table 3-3
- Condition code register manipulation: table 3-4
- Opcode map: table 3-5

Table 3-1. Accumulator and Memory Manipulation Instructions

Operations	Mnemonic	Addressing Modes												Boolean/ Arithmetic Operation	Condition Code Register								
		IMMED			DIRECT			INDEX			EXTEND				IMPLIED			5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#		OP	~	#	H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	4	2	BB	4	3			A + M → A	!	●	!	!	!	!	
	ADDB	CB	2	2	DB	3	2	EB	4	2	FB	4	3			B + M → B	!	●	!	!	!	!	
Add Double	ADDD	C3	3	3	D3	4	2	E3	5	2	F3	5	3			A : B + M : M + 1 → A : B	●	●	!	!	!	!	
Add Accumulators	ABA													1B	1	1	A + B → A	!	●	!	!	!	
Add With Carry	ADCA	89	2	2	99	3	2	A9	4	2	B9	4	3			A + M + C → A	!	●	!	!	!	!	
	ADCB	C9	2	2	D9	3	2	E9	4	2	F9	4	3			B + M + C → B	!	●	!	!	!	!	
AND	ANDA	84	2	2	94	3	2	A4	4	2	B4	4	3			A · M → B	●	●	!	!	R	●	
	ANDB	C4	2	2	D4	3	2	E4	4	2	F4	4	3			B · M → B	●	●	!	!	R	●	
Bit Test	BIT A	85	2	2	95	3	2	A5	4	2	B5	4	3			A · M	●	●	!	!	R	●	
	BIT B	C5	2	2	D5	3	2	E5	4	2	F5	4	3			B · M	●	●	!	!	R	●	
Clear	CLR							6F	5	2	7F	5	3			00 → M	●	●	R	S	R	R	
	CLRA													4F	1	1	00 → A	●	●	R	S	R	R
	CLRB													5F	1	1	00 → B	●	●	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	4	2	B1	4	3			A - M	●	●	!	!	!	!	
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3			B - M	●	●	!	!	!	!	
Compare Accumulators	CBA													11	1	1	A - B	●	●	!	!	!	!
Complement, 1's	COM							63	6	2	73	6	3			M → M	●	●	!	!	R	S	
	COMA													43	1	1	A → A	●	●	!	!	R	S
	COMB													53	1	1	B → B	●	●	!	!	R	S
Complement, 2's (Negate)	NEG							60	6	2	70	6	3			00 → M → M	●	●	!	!	①	②	
	NEGA													40	1	1	00 → A → A	●	●	!	!	①	②
	NEGB													50	1	1	00 → B → B	●	●	!	!	①	②
Decimal Adjust, A	DAA													19	2	1	Converts binary add of BCD characters into BCD format	●	●	!	!	!	③
Decrement	DEC							6A	6	2	7A	6	3			M - 1 → M	●	●	!	!	④	●	
	DECA													4A	1	1	A - 1 → A	●	●	!	!	④	●
	DECB													5A	1	1	B - 1 → B	●	●	!	!	④	●
Exclusive OR	EORA	88	2	2	98	3	2	A8	4	2	B8	4	3			A ⊕ M → A	●	●	!	!	R	●	
	EORB	C8	2	2	D8	3	2	E8	4	2	F8	4	3			B ⊕ M → B	●	●	!	!	R	●	
Increment	INC							6C	6	2	7C	6	3			M + 1 → M	●	●	!	!	⑤	●	
	INCA													4C	1	1	A + 1 → A	●	●	!	!	⑤	●
	INCB													5C	1	1	B + 1 → B	●	●	!	!	⑤	●
Load Accumulator	LDAA	86	2	2	96	3	2	A6	4	2	B6	4	3			M → A	●	●	!	!	R	●	
	LDAB	C6	2	2	D6	3	2	E6	4	2	F6	4	3			M → B	●	●	!	!	R	●	
Load Double Accumulator	LDD	CC	3	3	DC	4	2	EC	5	2	FC	5	3			M + 1 → B, M → A	●	●	!	!	R	●	
Multiply Unsigned	MUL													3D	7	1	A × B → A : B	●	●	●	●	●	⑥
OR, Inclusive	ORAA	8A	2	2	9A	3	2	AA	4	2	BA	4	3			A + M → A	●	●	!	!	R	●	
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3			B + M → B	●	●	!	!	R	●	
Push Data	PSHA													36	4	1	A → M <sub>SP</sub> , SP - 1 → SP	●	●	●	●	●	●
	PSHB													37	4	1	B → M <sub>SP</sub> , SP - 1 → SP	●	●	●	●	●	●

Note: Condition Code Register will be explained in Note of table 3-4.

Table 3-1. Accumulator and Memory Manipulation Instructions (Cont.)

Operations	Mnemonic	Addressing Modes												Boolean/ Arithmetic Operation	Condition Code Register								
		IMMED			DIRECT			INDEX			EXTEND				IMPLIED			5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#		OP	~	#	H	I	N	Z	V	C
Pull Data	PULA													32	3	1	SP+1→SP, Msp→A	●	●	●	●	●	●
	PULB													33	3	1	SP+1→SP, Msp→B	●	●	●	●	●	●
Rotate Left	ROL							69	6	2	79	6	3				*1	●	●	●	●	●	●
	ROLA													49	1	1		●	●	●	●	●	●
	ROLB													59	1	1		●	●	●	●	●	●
Rotate Right	ROR							66	6	2	76	6	3				*2	●	●	●	●	●	●
	RORA													46	1	1		●	●	●	●	●	●
	RORB													56	1	1		●	●	●	●	●	●
Shift Left Arithmetic	ASL							68	6	2	78	6	3				*3	●	●	●	●	●	●
	ASLA													48	1	1		●	●	●	●	●	●
	ASLB													58	1	1		●	●	●	●	●	●
Double Shift Left, Arithmetic	ASLD													05	1	1	*4	●	●	●	●	●	●
Shift Right Arithmetic	ASR							67	6	2	77	6	3				*5	●	●	●	●	●	●
	ASRA													47	1	1		●	●	●	●	●	●
	ASRB													57	1	1		●	●	●	●	●	●
Shift Right Logical	LSR							64	6	2	74	6	3				*6	●	●	●	●	●	●
	LSRA													44	1	1		●	●	●	●	●	●
	LSRB													54	1	1		●	●	●	●	●	●
Double Shift Right Logical	LSRD													04	1	1	*7	●	●	●	●	●	●
Store Accumulator	STAA							97	3	2	A7	4	2	B7	4	3	A→M	●	●	●	●	●	●
	STAB							D7	3	2	E7	4	2	F7	4	3	B→M	●	●	●	●	●	●
Store Double Accumulator	STD							DD	4	2	ED	5	2	FD	5	3	A→M B→M+1	●	●	●	●	●	●
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3				A-M→A	●	●	●	●	●	●
	SUBB	C0	2	2	D0	3	2	E0	4	2	F0	4	3				B-M→B	●	●	●	●	●	●
Double Subtract	SUBD	83	3	3	93	4	2	A3	5	2	B3	5	3				A : B-M : M+1→ A : B	●	●	●	●	●	●
Subtract Accumulators	SBA													10	1	1	A-B→A	●	●	●	●	●	●
Subtract With Carry	SBCA	82	2	2	92	3	2	A2	4	2	B2	4	3				A-M-C→A	●	●	●	●	●	●
	SBCB	C2	2	2	D2	3	2	E2	4	2	F2	4	3				B-M-C→B	●	●	●	●	●	●
Transfer Accumulators	TAB													16	1	1	A→B	●	●	●	●	●	●
	TBA													17	1	1	B→A	●	●	●	●	●	●
Test Zero or Minus	TST							6D	4	2	7D	4	3				M-00	●	●	●	●	●	●
	TSTA													4D	1	1	A-00	●	●	●	●	●	●
	TSTB													5D	1	1	B-00	●	●	●	●	●	●
And Immediate	AIM							71	6	3	61	7	3				M·IMM→M	●	●	●	●	●	●
OR Immediate	OIM							72	6	3	62	7	3				M+IMM→M	●	●	●	●	●	●
EOR Immediate	EIM							75	6	3	65	7	3				M⊕IMM→M	●	●	●	●	●	●
Test Immediate	TIM							7B	4	3	6B	5	3				M·IMM	●	●	●	●	●	●

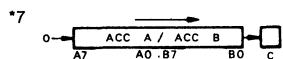
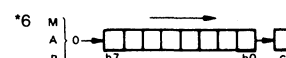
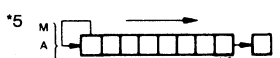
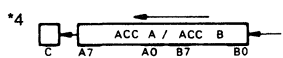
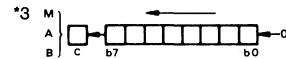
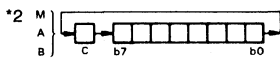
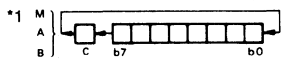


Table 3-2. Index Register and Stack Manipulation Instructions

Pointer Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register							
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0		
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C		
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	5	2	BC	5	3						X-M:M+1	•	•	1	1	1	1
Decrement Index Reg	DEX													09	1	1			X-1-X	•	•	•	1	•	•
Decrement Stack Pntr	DES													34	1	1			SP-1-SP	•	•	•	•	•	•
Increment Index Reg	INX													08	1	1			X+1-X	•	•	•	1	•	•
Increment Stack Pntr	INS													31	1	1			SP+1-SP	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	5	2	FE	5	3						M $\rightarrow$ X <sub>H</sub> , (M+1) $\rightarrow$ X <sub>L</sub>	•	•	⑦	1	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	5	2	BE	5	3						M $\rightarrow$ SP <sub>H</sub> , (M+1) $\rightarrow$ SP <sub>L</sub>	•	•	⑦	1	R	•
Store Index Reg	STX				DF	4	2	EF	5	2	FF	5	3						X <sub>H</sub> $\rightarrow$ M, X <sub>L</sub> $\rightarrow$ (M+1)	•	•	⑦	1	R	•
Store Stack Pntr	STS				9F	4	2	AF	5	2	BF	5	3						SP <sub>H</sub> $\rightarrow$ M, SP <sub>L</sub> $\rightarrow$ (M+1)	•	•	⑦	1	R	•
Index Reg $\rightarrow$ Stack Pntr	TXS													35	1	1			X-1-SP	•	•	•	•	•	•
Stack Pntr $\rightarrow$ Index Reg	TSX													30	1	1			SP+1-X	•	•	•	•	•	•
Add	ABX													3A	1	1			B+X-X	•	•	•	•	•	•
Push Data	PSHX													3C	5	1			X <sub>L</sub> $\rightarrow$ Msp, SP-1-SP X <sub>H</sub> $\rightarrow$ Msp, SP-1-SP	•	•	•	•	•	•
Pull Data	PULX													38	4	1			SP+1-SP, Msp $\rightarrow$ X <sub>H</sub> SP+1-SP, Msp $\rightarrow$ X <sub>L</sub>	•	•	•	•	•	•
Exchange	XGDX													18	2	1			ACCD $\rightarrow$ IX	•	•	•	•	•	•

Note: Condition Code Register will be explained in Note of table 3-4.



Table 3-3. Jump and Branch Instructions

Operations	Mnemonic	Addressing Modes															Branch Test	Condition Code Register					
		RELATIVE			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Branch Always	BRA	20	3	2												None	•	•	•	•	•	•	
Branch Never	BRN	21	3	2												None	•	•	•	•	•	•	
Branch if Carry Clear	BCC	24	3	2												C=0	•	•	•	•	•	•	
Branch if Carry Set	BCS	25	3	2												C=1	•	•	•	•	•	•	
Branch if =Zero	BEQ	27	3	2												Z=1	•	•	•	•	•	•	
Branch if ≥Zero	BGE	2C	3	2												$N \oplus V = 0$	•	•	•	•	•	•	
Branch if >Zero	BGT	2E	3	2												$Z + (N \oplus V) = 0$	•	•	•	•	•	•	
Branch if Higher	BHI	22	3	2												$C + Z = 0$	•	•	•	•	•	•	
Branch if ≤Zero	BLE	2F	3	2												$Z + (N \oplus V) = 1$	•	•	•	•	•	•	
Branch if Lower Or Same	BLS	23	3	2												$C + Z = 1$	•	•	•	•	•	•	
Branch if <Zero	BLT	2D	3	2												$N \oplus V = 1$	•	•	•	•	•	•	
Branch if Minus	BMI	2B	3	2												N=1	•	•	•	•	•	•	
Branch if Not Equal Zero	BNE	26	3	2												Z=0	•	•	•	•	•	•	
Branch if Overflow Clear	BVC	28	3	2												V=0	•	•	•	•	•	•	
Branch if Overflow Set	BVS	29	3	2												V=1	•	•	•	•	•	•	
Branch if Plus	BPL	2A	3	2												N=0	•	•	•	•	•	•	
Branch To Subroutine	BSR	8D	5	2													•	•	•	•	•	•	
Jump	JMP							6E	3	2	7E	3	3				•	•	•	•	•	•	
Jump To Subroutine	JSR				9D	5	2	AD	5	2	BD	6	3				•	•	•	•	•	•	
No Operation	NOP													01	1	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return from Interrupt	RTI													3B	10	1		Ⓢ					
Return From Subroutine	RTS													39	5	1		•	•	•	•	•	•
Software Interrupt	SWI													3F	12	1		•	5	•	•	•	•
Wait for Interrupt*	WAI													3E	9	1		•	Ⓣ	•	•	•	•
Sleep	SLP													1A	4	1		•	•	•	•	•	•

Note: \* WAI puts R/W high; Address Bus goes to FFFF; Data Bus goes to the three state. Condition Code Register will be explained in Note of table 3-4.

Table 3-4. Condition Code Register Manipulation Instructions

Operations	Mnemonic	Addressing Modes			Boolean Operation	Condition Code Register						
		Implied				5	4	3	2	1	0	
		OP	~	#		H	I	N	Z	V	C	
Clear Carry	CLC	0C	1	1	0 · C	●	●	●	●	●	R	
Clear Interrupt Mask	CLI	0E	1	1	0 · I	●	R	●	●	●	●	
Clear Overflow	CLV	0A	1	1	0 · V	●	●	●	●	●	R	●
Sat Carry	SEC	0D	1	1	1 · C	●	●	●	●	●	●	S
Set Interrupt Mask	SEI	0F	1	1	1 · I	●	S	●	●	●	●	●
Set Overflow	SEV	0B	1	1	1 · V	●	●	●	●	●	S	●
Accumulator A ← CCR	TAP	06	1	1	A · CCR	⑩						
CCR ← Accumulator A	TPA	07	1	1	CCR · A	●	●	●	●	●	●	●

Legend

- OP Operation Code (Hexadecimal)
- ~ Number of MCU Cycles
- M<sub>SP</sub> Contents of memory location pointed to by Stack Pointer
- # Number of Program Bytes
- + Arithmetic Plus
- Arithmetic Minus
- Boolean AND
- ⊕ Boolean Inclusive OR
- ⊗ Boolean Exclusive OR
- M Complement of M
- Transfer into
- 0 Bit = Zero
- 00 Byte = Zero

Condition Code Symbols

- H Half-carry from bit 3 to bit 4
- I Interrupt mask
- N Negative (sign bit)
- Z Zero (byte)
- V Overflow, 2's complement
- C Carry/Borrow from/to bit 7
- R Reset Always
- S Set Always
- ↓ Set if true after test or clear
- Not Affected

Note: Condition Code Register Notes: (Bit set if test is true and cleared otherwise)

- ① (Bit V) Test: Result = 10000000?
- ② (Bit C) Test: Result = 00000000?
- ③ (Bit C) Test: BCD Character of high-order byte greater than 10? (Not cleared if previously set)
- ④ (Bit V) Test: Operand = 10000000 prior to execution?
- ⑤ (Bit V) Test: Operand = 01111111 prior to execution?
- ⑥ (Bit V) Test: Set equal to N + C = 1 after the execution of instructions
- ⑦ (Bit N) Test: Result less than zero? (Bit 15=1)
- ⑧ (All Bit) Load condition code register from stack.
- ⑨ (Bit I) Set when interrupt occurs. If previous set, a non-maskable interrupt is required to exist the wait state.
- ⑩ (All Bit) Set according to the contents of accumulator A.
- ⑪ (Bit C) Result of multiplication bit 7=1? (ACCB)

Table 3-5. Memory Map

OP CODE					ACC A	ACC B	IND	EXT/DIR*	ACCA or SP				ACCB or X									
	HI	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111					
LO	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F						
0000	0	↘	SBA	BRA	TSX	NEG			SUB				0									
0001	1	NOP	CBA	BRN	INS	↘			AIM				CMP				1					
0010	2	↘	↘	BHI	PULA	↘			OIM				SBC				2					
0011	3	↘	↘	BLS	PULB	COM			SUBD				ADD				3					
0100	4	LSRD	↘	BCC	DES	LSR			AND				4									
0101	5	ASLD	↘	BCS	TXS	↘			EIM				BIT				5					
0110	6	TAP	TAB	BNE	PSHA	ROR			LDA				6									
0111	7	TPA	TBA	BEQ	PSHB	ASR			↘				STA	↘				STA	7			
1000	8	INX	XGDX	BVC	PULX	ASL			EOR				8									
1001	9	DEX	DAA	BVS	RTS	ROL			ADC				9									
1010	A	CLV	SLP	BPL	ABX	DEC			ORA				A									
1011	B	SEV	ABA	BMI	RTI	↘			TIM				ADD				B					
1100	C	CLC	↘	BGE	PSHX	INC			CPX				LDD				C					
1101	D	SEC	↘	BLT	MUL	TST			BSR	JSR				↘				STD	D			
1110	E	CLI	↘	BGT	WAI	↘			JMP				LDS				LDX				E	
1111	F	SEI	↘	BLE	SWI	CLR			↘				STS				↘				STX	F

☐ UNDEFINED OP CODE

\* Only AIM, OIM, EIM, TIM instructions



### 3.4 CPU Instruction Flow

When operating, the CPU fetches an instruction from memory and executes the required function. This sequence starts from  $\overline{\text{RES}}$  high, and repeats itself continuously if not affected by a special instruction or control signal. SWI, RTI, WAI, and SLP instructions change this operation, and  $\overline{\text{NMI}}$ ,  $\overline{\text{IRQ}}_1$ ,  $\overline{\text{IRQ}}_2$ ,  $\overline{\text{IRQ}}_3$ ,  $\overline{\text{HALT}}$ , and  $\overline{\text{STBY}}$  control it. Figure 3-2 shows the CPU mode transitions, and figure 3-3 is the CPU system flowchart. Table 3-6 shows the CPU operating states and port states.

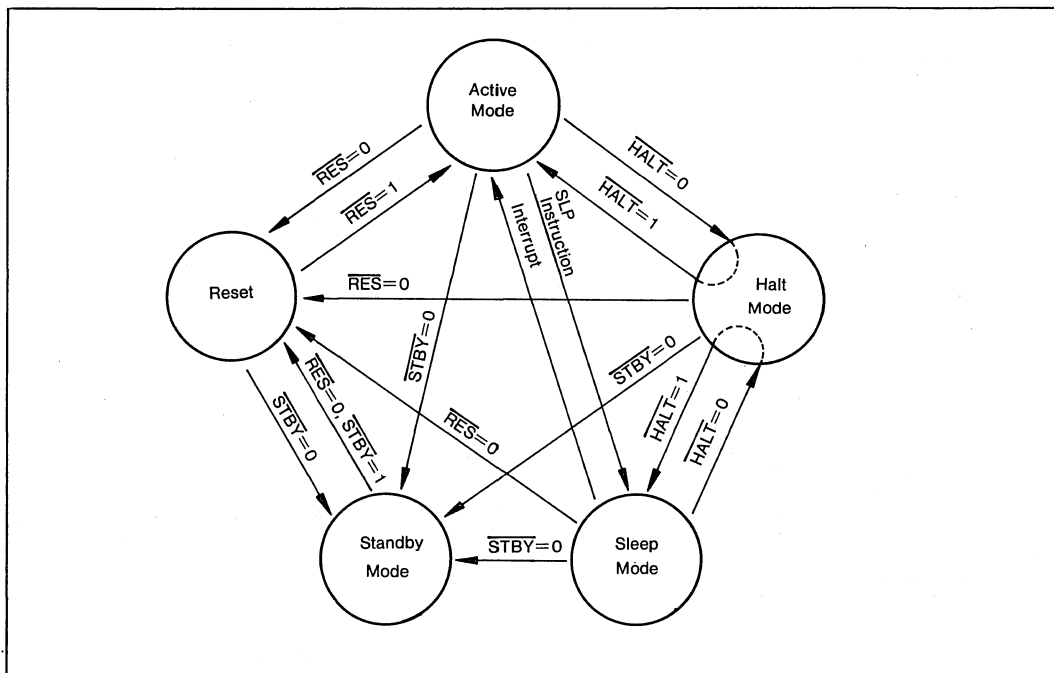
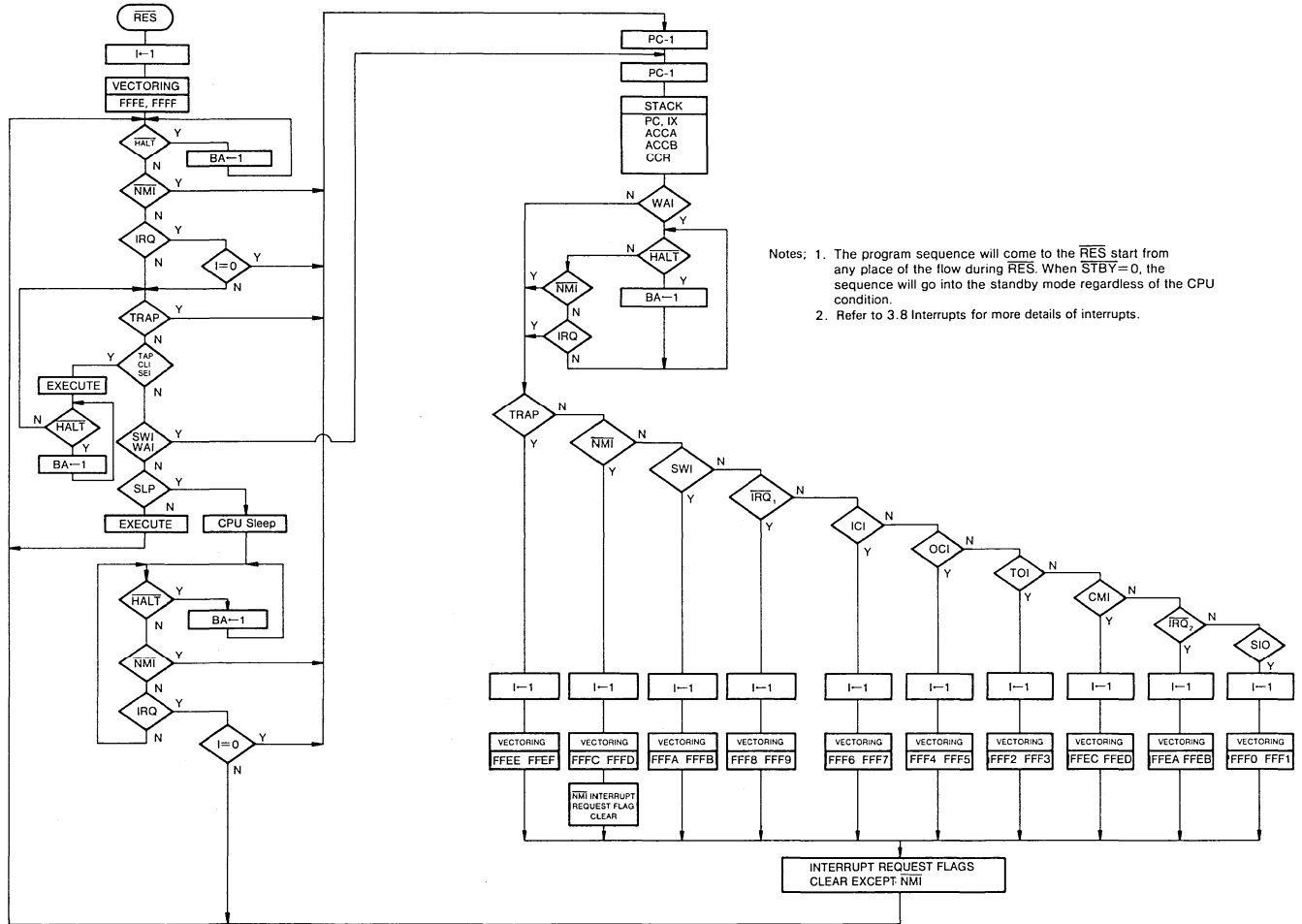


Figure 3-2. CPU Operation Mode Transitions



Notes: 1. The program sequence will come to the  $\overline{\text{RES}}$  start from any place of the flow during RES. When STBY=0, the sequence will go into the standby mode regardless of the CPU condition.  
 2. Refer to 3.8 Interrupts for more details of interrupts.

Figure 3-3. System Flowchart

Table 3-6. CPU Operating States and Port States

Port	Mode	Reset	Standby <sup>3</sup>	Halt <sup>4</sup>	Sleep
1 (A <sub>0</sub> -A <sub>7</sub> )	1, 2	High	High impedance	High impedance	High
	3	High impedance	High impedance		Keep
2	1, 2	High impedance	High impedance	Keep	Keep
	3	High impedance	High impedance		Keep
3 (D <sub>0</sub> -D <sub>7</sub> )	1, 2	High impedance	High impedance	High impedance	High impedance
	3	High impedance	High impedance		Keep
4 (A <sub>8</sub> -A <sub>15</sub> )	1, 2	High	High impedance	High impedance	High
	3	High impedance	High impedance		Keep
5	1, 2	High impedance	High impedance	High impedance	High impedance
	3	High impedance	High impedance		High impedance
6	1, 2	High impedance	High impedance	Keep	Keep
	3	High impedance	High impedance		Keep
7	1, 2	Note 1	High impedance	Note 2	Note 1
	3	High impedance	High impedance		Keep

Notes:

1.  $\overline{RD}$ ,  $\overline{WR}$ ,  $R\overline{W}$ ,  $\overline{LIR}$  = high; BA = low
2.  $\overline{RD}$ ,  $\overline{WR}$ ,  $R\overline{W}$  = high impedance;  $\overline{LIR}$ , BA = high
3. E is high impedance in standby state.
4.  $\overline{HALT}$  cannot be accepted in mode 3.

### 3.5 Low Power Dissipation Modes

The MCU has two low power dissipation modes, sleep and standby. Table 3-7 shows the MCU state in sleep and standby modes.

Table 3-7. Sleep and Standby Modes

	Sleep Mode	Standby Mode
Oscillation circuits	Continue operation	Stop
CPU	Stop	Stop
CPU registers	Hold	Undefined
RAM	Hold	Hold
I/O pins	Hold	High impedance
Timers	Continue operation	Stop
SCI	Continue operation	Stop
Internal Registers	Hold	Reset
How to release	Interrupt STBY = low Reset start	$\overline{\text{STBY}}$ = high before reset start (Hold RES low after $\overline{\text{STBY}}$ high until oscillator stabilizes, 20 ms min)

### 3.5.1 Sleep Mode

The MCU goes into sleep mode when the SLP instruction is executed. In the sleep mode, the CPU stops operation while maintaining the registers' contents. Peripherals such as the timers and the SCI continue their functions. One-fifth as much power is dissipated in sleep mode as in the operating mode.

The sleep mode is terminated by an interrupt, or a  $\overline{\text{RES}}$  or  $\overline{\text{STBY}}$  signal.  $\overline{\text{RES}}$  causes the MCU to reset,  $\overline{\text{STBY}}$  causes it to go into standby mode. When the CPU receives an interrupt request, it returns to operating mode. If the interrupts are enabled, it branches to the interrupt service routine. If they are masked, it executes the next instruction. However, if timer 1 or 2 prohibits a timer interrupt, the CPU won't cancel the sleep mode because there is no interrupt request to the CPU.

The sleep mode reduces power dissipation for a system that doesn't need the CPU's continuous operation. Figure 3-4 is the sleep instruction timing chart.

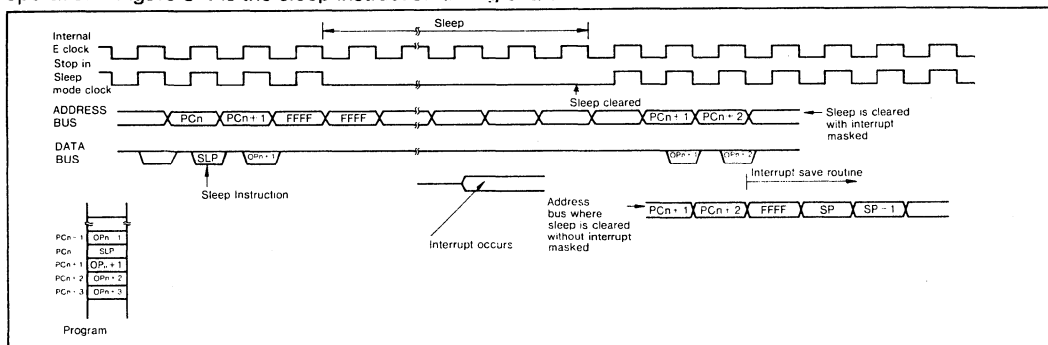


Figure 3-4. Sleep Instruction Timing

### 3.5.2 Standby Mode

When the  $\overline{\text{STBY}}$  input goes low, the MPU stops all clocks and goes to the reset state. In this mode, power dissipation is greatly reduced. All pins except  $V_{\text{CC}}$ ,  $V_{\text{SS}}$ ,  $\overline{\text{STBY}}$ , and XTAL (outputs 0) are detached from the MCU internally, and go to high impedance.

In standby mode, power is supplied to the MCU, so that the contents of RAM are retained. The MCU returns from this mode with a reset.

An example of the use of this mode follows. First, save the CPU state and SP contents in RAM by an NMI routine. Then disable the RAME bit in the RAM control register and set the STBY PWR bit to go to standby mode. If the STBY PWR bit is still set after reset start, power has been supplied to the MCU and the RAM contents have been retained properly. The system can restore itself by returning the pre-standby information to the SP and registers. Figure 3-5 shows the timing at the  $\overline{\text{NMI}}$ ,  $\overline{\text{RES}}$  and  $\overline{\text{STBY}}$  pins.

Note: In standby mode, the mode program pins,  $\text{MP}_0$  and  $\text{MP}_1$ , should be held according to the operation mode. If they are opened, the standby current will increase over the specified value.

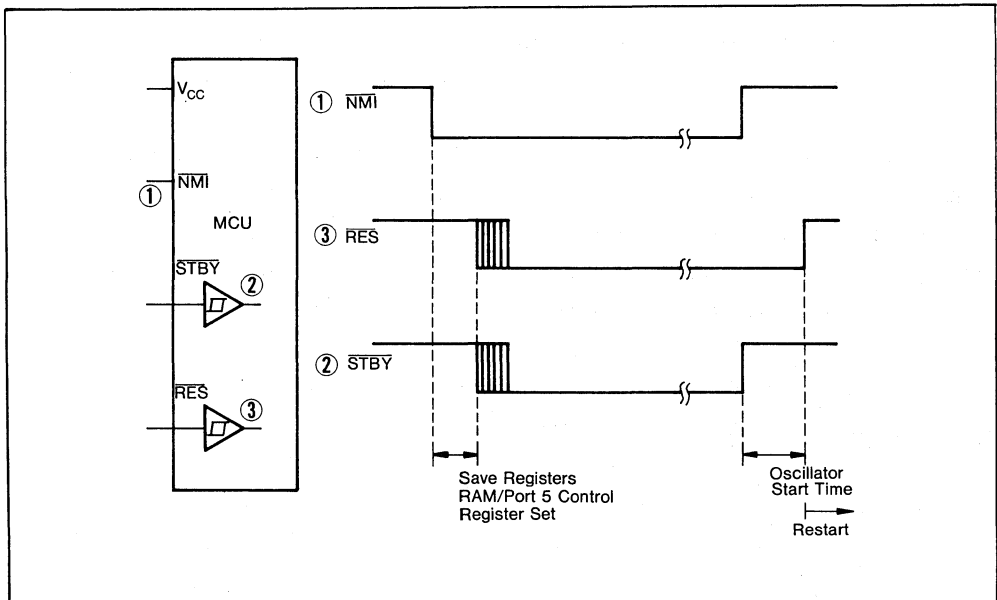


Figure 3-5. Standby Mode Timing

## 3.6 Trap Function

The CPU generates an interrupt with the highest priority (TRAP) when it fetches an undefined instruction or an instruction from outside of memory space. The trap function prevents system malfunctions caused by noise or program error.

### 3.6.1 Opcode Error

When the CPU fetches an undefined opcode, it saves the CPU registers as well as performing the normal interrupt procedure and branches to TRAP (\$FFEE, \$FFEF). This has the highest priority next to reset.

### 3.6.2 Address Error

When an instruction is fetched from outside the internal ROM, RAM, and external memory area, the MCU generates an address interrupt as well as an opcode error. But on a system with no external memory, a trap is not generated if an instruction is fetched from the external memory area. Table 3-8 shows the addresses where an address error occurs in each mode. This function is available only for an instruction fetch, and does not apply to data read/write.

Table 3-8. Address Error Addresses

Mode	Address
1	\$0000-\$001F
2	\$0000-\$001F
3	\$0000-\$003F, \$0100-\$EFFF

### 3.6.3 Caution

The trap function has a retry function other interrupts do not have. The program flow returns to the address where the trap occurred when RTI returns the CPU to the main routine from the TRAP routine. The retry can prevent problems caused by noise, etc. However, if another trap occurs, the program can repeat the retry/TRAP cycle forever. Consideration is necessary in programming.

In figure 3-6, after executing instruction OPn, the MPU fetches and decodes an undefined opcode and generates a trap interrupt. When the RTI is executed in the trap interrupt servicing routine, the MPU will put \$FF03 in the PC, fetch the same opcode, and generate the trap again. The MPU will endlessly repeat loop ABC.



In figure 3-7, after executing the BSR, the branch destination address is output to the address bus to fetch the first instruction of the subroutine. If \$0001 is erroneously output as the address, the MPU will decode it and generate a trap interrupt. When the RTI is executed in the trap interrupt servicing routine, the MPU will put \$0001 in the PC, and start from this address. This will generate another trap, in an endless loop.

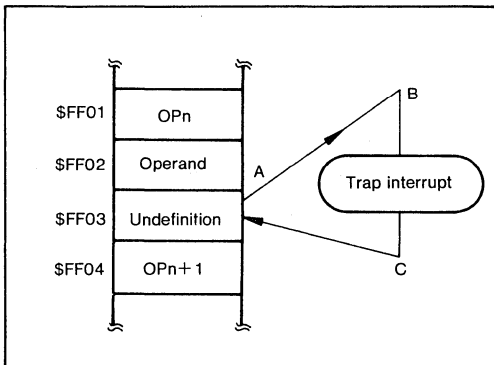


Figure 3-6. Executing an Undefined Opcode

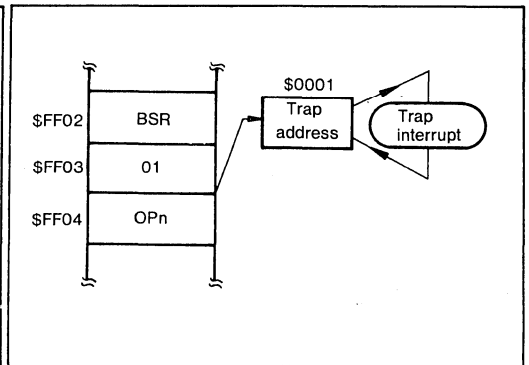


Figure 3-7. Erroneous Fetch

### 3.7 Reset

To reset the MCU during operation, hold  $\overline{\text{RES}}$  low for at least 3 system-clock cycles. At the third cycle, when the clock signal is low, all the address buses become high. While  $\overline{\text{RES}}$  is low, the buses remain high. When  $\overline{\text{RES}}$  goes high, the MCU starts the following operations.

1. Latches the value of the mode program pins,  $\text{MP}_1$  and  $\text{MP}_0$ .
2. Initializes the internal registers (see table 2-3).
3. Sets the interrupt mask bit. For the CPU to recognize the maskable interrupts  $\overline{\text{IRQ}}_1$ ,  $\overline{\text{IRQ}}_2$ , and  $\text{IRQ}_3$ , this bit should be cleared in advance.
4. Puts the contents (= start address) of the last two addresses (\$FFFE, \$FFFF) into the program counter and starts the program from this address. See table 2-4.

The MCU cannot accept a reset input until the clock oscillation is stable after power-on (20 ms maximum). This is because the reset signal is internally synchronized to the clock as shown in figure 3-8. Until oscillation starts, the MCU and I/O pins are undefined. External devices that need to know the MPU's state during this period must be informed by external circuits. Refer to 2.4 Ports for the state of the ports during reset. Figure 3-9 shows reset timing.

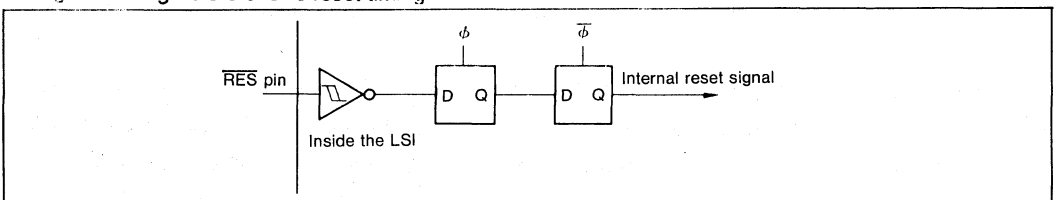


Figure 3-8 Reset Circuit

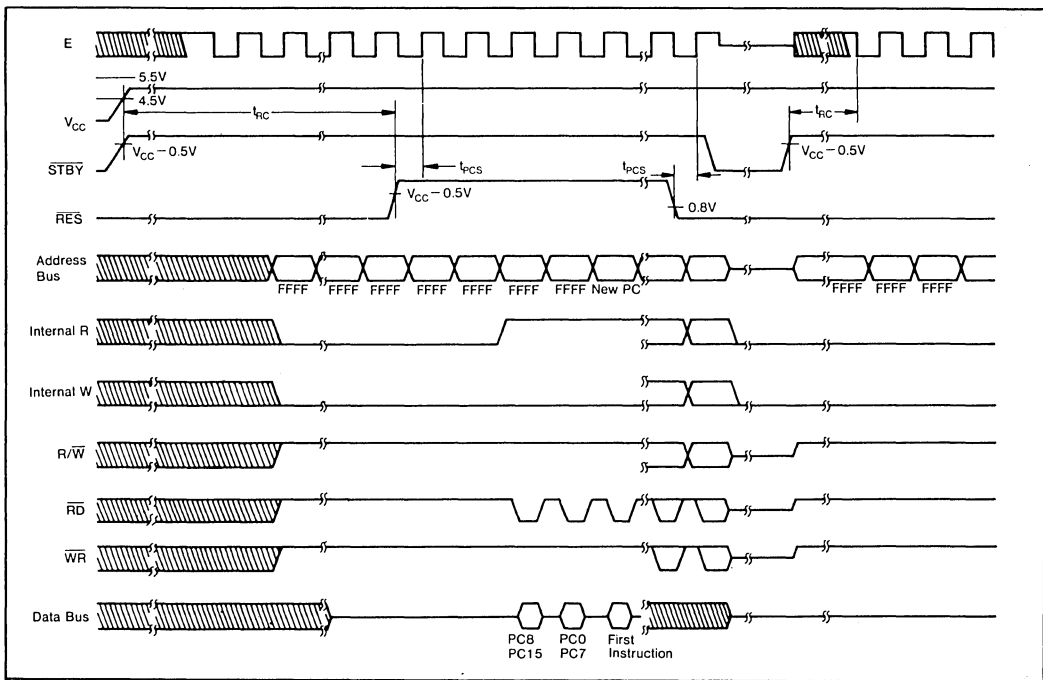


Figure 3-9 Reset Timing

### 3.8 Interrupts

The CPU will complete the current instruction before accepting the request. If the interrupt mask bit in the condition code register is set, the request will be ignored. When the interrupt sequence starts, the contents of the program counter, index register, accumulators, and condition code register will be saved onto the stack. Then the CPU sets the interrupt mask bit and will not respond to further maskable interrupt requests. In the last cycle of the interrupt, the CPU fetches the vectors shown in table 3-9, transfers their contents to the program counter and branches to the interrupt service routine.

The external interrupt pins  $\overline{IRQ}_1$  and  $\overline{IRQ}_2$  are also used as P5<sub>0</sub> and P5<sub>1</sub>. The function is chosen by the enable bits in the RAM/port 5 control register (bits 0 and 1) at \$0014. See 2.5 RAM/Port 5 Control Register for details.

When one of the internal interrupts, ICI, OCI, TOI, CMI, or SIO is generated, the CPU produces the internal interrupt signal, IRQ<sub>3</sub>. IRQ<sub>3</sub> functions just the same as  $\overline{IRQ}_1$  or  $\overline{IRQ}_2$ , except for its vector address. Table 3-9 is an interrupt vector map, figure 3-10 is the interrupt sequence, and figure 3-11 is the interrupt circuit block diagram.

Table 3-9. Interrupt Vector Memory Map

Priority	Vector Location		Interrupt
	MSB	LSB	
Highest ↑ ↓ Lowest	FFFE	FFFF	RES
	FFEE	FFEF	TRAP
	FFFC	FFFD	NMI
	FFFA	FFFB	SWI (Software interrupt)
	FFF8	FFF9	IRQ <sub>1</sub>
	FFF6	FFF7	ICI (Timer 1 input capture)
	FFF4	FFF5	OCI (Timer 1 output compare 1, 2)
	FFF2	FFF3	TOI (Timer 1 overflow)
	FFEC	FFED	CMI (Timer 2 counter match)
	FFEA	FFEB	IRQ <sub>2</sub>
	Lowest	FFF0	FFF1

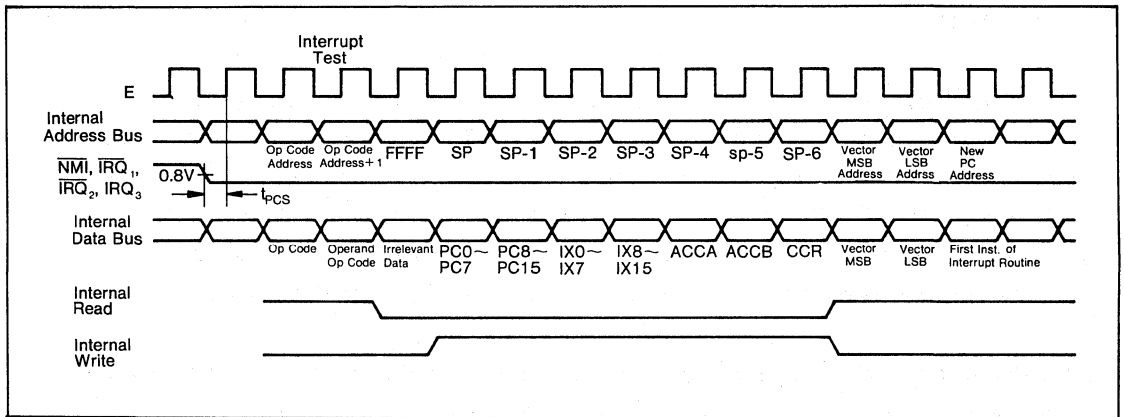


Figure 3-10. Interrupt Sequence



## Section 4. Timer 1

The 16-bit programmable timer, timer 1, can measure an input waveform and independently generate two independent waveforms. The pulse widths of the input and output waveforms can vary from microseconds to seconds.

Timer 1 has the following components (figure 4-1).

- Control/status register 1 (8 bits)
- Control/status register 2 (7 bits)
- Free-running counter (16 bits)
- Output compare register 1 (16 bits)
- Output compare register 2 (16 bits)
- Input capture register (16 bits)

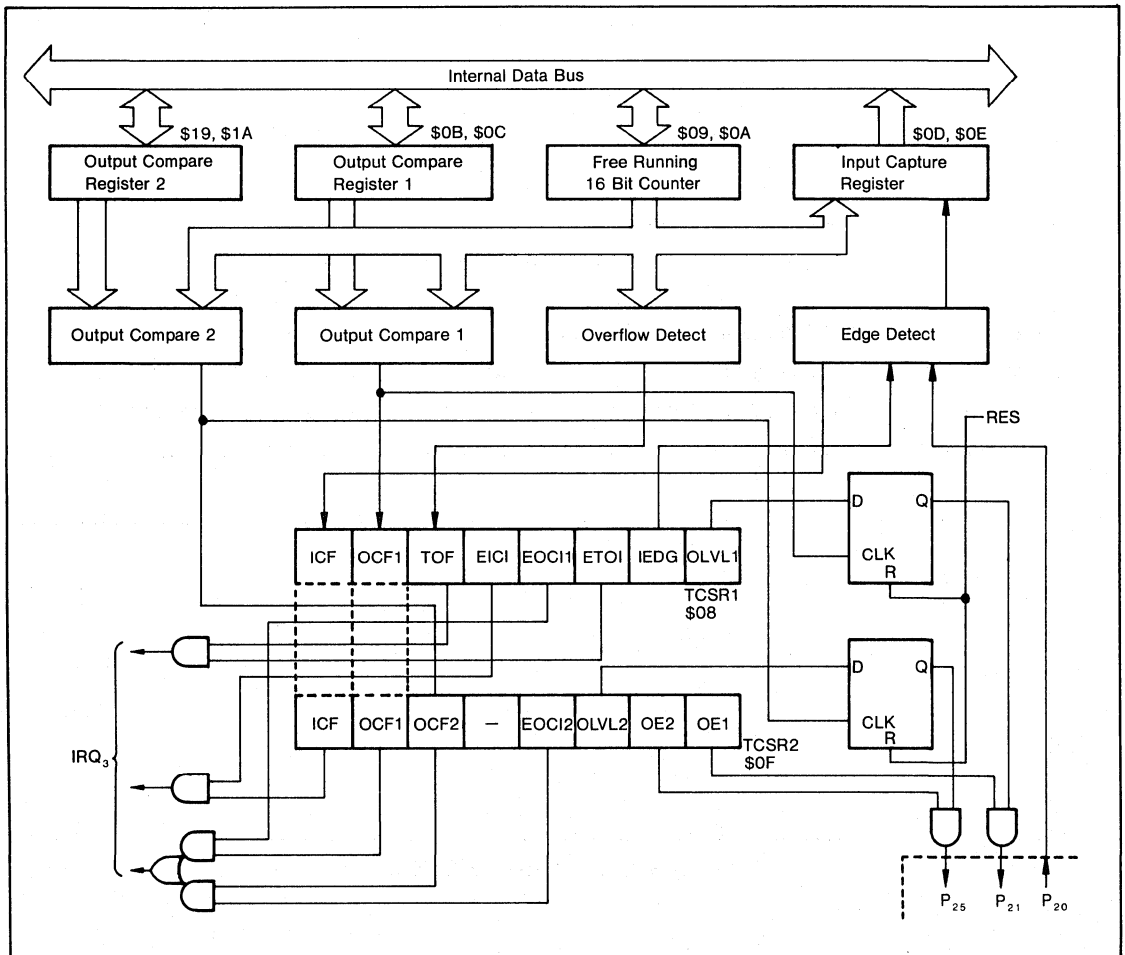


Figure 4-1. Timer 1 Block Diagram



## 4.1 Free-Running Counter (FRC)

The key element of timer 1 is the 16-bit free-running counter. It is incremented by the system clock. The counter value can be read by software without affecting the counter. Reset clears the counter.

The free-running counter is located at addresses \$0009 and \$000A. When the CPU writes to the high byte of the FRC (\$0009), a preset value (\$FFF8) is actually written to both bytes of the counter, regardless of the write data value. When the CPU writes to the low byte (\$000A) after the high byte, both the low and high byte of the write data value are written to the FRC. See figure 4-2. The counter operates this way when written to by double-byte store instructions (STD, STX, etc).

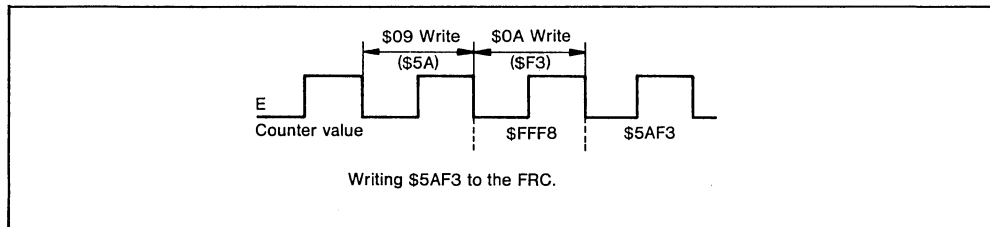


Figure 4-2. Counter Write Timing

## 4.2 Output Compare Registers (OCR)

The output compare registers are 16-bit read/write registers that control the output waveforms. They are located at \$000B, \$000C (OCR1) and \$0019, \$001A (OCR2).

The OCR's are constantly compared to the FRC. When the data matches, the output compare flag (OCF) in the timer control/status register (TCSR) is set. If an output enable bit (OE) in TCSR2 is set to 1, an output level bit (OLVL) will be output to bit 1 (Tout1) and bit 5 (Tout2) of port 2. To determine the output level for the next compare match, change OCR and OLVL.

The OCR is set to \$FFFF after reset. The compare function is inhibited for a cycle just after a write to the OCR or the upper byte of the FRC. This is so that the 16-bit value will be valid in the OCR, and because \$FFF8 is set after the FRC's upper byte is written.

To write to the OCR, use a 2-byte transfer instruction, such as STX.

### 4.3 Input Capture Register (ICR)

The input capture register is a 16-bit read-only register located at \$000D, \$000E. It stores the FRC's value when an external input signal transition at P2<sub>0</sub> generates an input capture pulse. Which transition generates the pulse is defined by the input edge bit (IEDG) in TCSR1.

To input an edge bit to the edge detector, clear bit 0 of port 2's DDR. When an input transition occurs at the next cycle of the CPU's ICR upper-byte read, the input capture pulse will be delayed one cycle. To ensure input capture, the CPU must read the ICR with a 2-byte transfer instruction. The ICR is cleared to all zeros during reset.

### 4.4 Timer Control/Status Register 1 (TCSR1)

The timer control/status register 1 is an 8-bit register located at \$0008 (figure 4-3). All of the bits can be read and the lower 5 can be written to. The 3 upper read-only bits indicate the timer status.

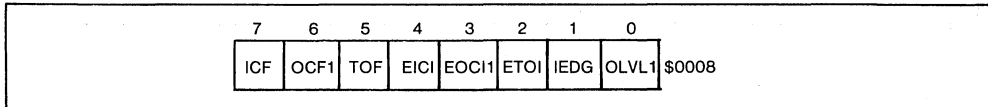


Figure 4-3. Timer Control/Status Register 1

#### 4.4.1 Output Level 1 (OLVL1)

OLVL1 is transferred to port 2, bit 1 when a match occurs between the counter and OCR1. If OE1, bit 0 of TCSR2 is set to 1, OLVL1 will be output at port 2 bit 1. Bit 0.

#### 4.4.2 Input Edge (IEDG)

IEDG determines whether the rising edge or the falling edge of P2<sub>0</sub> will trigger data transfer from the counter to the ICR. IEDG = 0 specifies a falling edge (high to low); IEDG = 1 specifies a rising edge (low to high). Bit 0 of port 2's DDR must be cleared for this function to operate. Bit 1.

#### 4.4.3 Enable Timer Overflow Interrupt (ETOI)

Setting ETOI to 1 enables timer overflow interrupt (TOI) to trigger an internal interrupt (IRQ<sub>3</sub>). When ETOI is cleared, the interrupt is inhibited. Bit 2.

#### 4.4.4 Enable Output Compare Interrupt 1 (EOCI1)

Setting EOCI1 to 1 enables output compare interrupt 1 (OCI1) to trigger an internal interrupt (IRQ<sub>3</sub>). When EOCI1 is cleared, the interrupt is inhibited. Bit 3.

#### 4.4.5 Enable Input Capture Interrupt (EICI)

Setting EICI to 1 enables input capture interrupt (IC!) to trigger an internal interrupt (IRQ<sub>3</sub>). When EICI is cleared, the interrupt is inhibited: Bit 4.

#### 4.4.6 Timer Overflow Flag (TOF)

TOF is set when the counter value increments from \$FFF to \$0000. TOF is cleared when CPU reads the TCSR1, then the counter's upper byte (at \$0009). Bit 5, read only.

#### 4.4.7 Output Compare Flag 1 (OCF1)

OCF1 is set when a match has occurred between the FCR and OCR1. Writing to OCR1 (\$000B or \$000C) after reading the TCSR1 or TCSR2 clears OCF1. Bit 6, read only.

#### 4.4.8 Input Capture Flag (ICF)

ICF is set when the transition of the P2<sub>0</sub> input signal selected by IEDG causes the counter to transfer its data to the ICR. Reading the high byte of the ICR (\$000D) after reading TCSR1 or TCSR2 clears ICF. Bit 7, read only.

### 4.5 Timer Control/Status Register 2 (TCSR2)

The timer control/status register 2 is a 7-bit register located at \$000F (figure 4-4). All of the bits can be read and the lower 4 can be written to. The 3 upper read-only bits indicate the timer status.

Both TCSR1 and TCSR2 are cleared during reset.

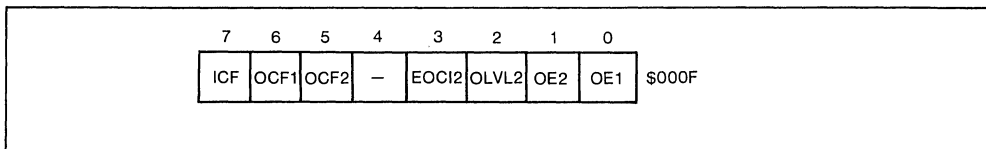


Figure 4-4. Timer Control/Status Register 2

#### 4.5.1 Output Enable 1 (OE1)

Setting OE1 to 1 enables OLVL1 to appear at P2<sub>1</sub> when a match has occurred between the counter and the output compare register 1 (OCR1). Clearing OE1 makes P2<sub>1</sub> an I/O port. Bit 0.



#### 4.5.2 Output Enable 2 (OE2)

Setting OE2 to 1 enables OLVL2 to appear at P2<sub>5</sub> when a match occurs between the counter and the output compare register 2 (OCR2). Clearing OE2 makes P2<sub>5</sub> an I/O port. Bit 1.

Note: If OE1 or OE2 is set to 1 before the first output compare match after reset, P2<sub>1</sub> or P2<sub>5</sub> will output 0.

#### 4.5.3 Output Level 2 (OLVL2)

OLVL2 is transferred to P2<sub>5</sub> when a match occurs between the counter and OCR2. If OE2 (bit 1 of TCSR2) is set to 1, OVLV2 will be output at P2<sub>5</sub>. Bit 2.

#### 4.5.4 Enable Output Compare Interrupt 2 (EOCI2)

Setting EOCI2 to 1 enables output compare interrupt 2 (OCI2) to trigger an internal interrupt (IRQ<sub>3</sub>). When EOCI2 is cleared, the interrupt is inhibited. Bit 3.

#### 4.5.5 Output Compare Flag 2 (OCF2)

OCF2 is set when a match has occurred between the FCR and OCR2. Writing to OCR2 (\$0019 or \$001A) after reading TCSR2 clears OCF1. Bit 6, read only.

#### 4.5.6 Output Compare Flag 1 (OCF1) and Input Capture Flag (ICF)

The OCF1 and ICF addresses are partially decoded. The CPU reading TCSR1/TCSR2 makes it possible to read OCF1 and ICF into bits 6 and 7.

## 4.6 Timer Status Flags

Table 4-1 shows set and clear conditions of each status flag in timer 1.

If flag set and clear conditions occur at the same time, timer 1 flags will be set.

Table 4-1 Timer 1 Status Flags

Flag	Set Condition	Clear Condition
Timer 1 ICF	· FRC → ICR at edge of P2 <sub>0</sub>	· Read TRCSR1 or TRCSR2, then ICR <sub>H</sub> · $\overline{RES} = 0$
OCF1	· OCR1 = FRC	· Read TRCSR1 or TRCSR2, then write OCR1 <sub>H</sub> or OCR1 <sub>L</sub> · $\overline{RES} = 0$
OCF2	· OCR2 = FRC	· Read TRCSR2, then write OCR2 <sub>H</sub> or OCR2 <sub>L</sub> · $\overline{RES} = 0$
TOF	· FRC = \$FFFF + 1 cycle	· Read TRCSR, then FRC <sub>H</sub> · $\overline{RES} = 0$

## 4.7 Precautions on Clearing OCF

Writing to the OCR after reading the TCSR when the OCF is 1 clears the OCF. However, the OCF is not cleared under the following conditions.

1. A compare match is found before the CPU writes to the OCR after reading the TCSR with OCF = 0.
2. A compare match is found at the same time as the CPU writes to the OCR after reading the TCSR with OCF = 1.

See figure 4-5.

The OCF will always be cleared if you assure that a compare match does not occur between the TCSR read and the OCR write. In example 1, figure 4-6, the OCR is loaded with the contents of the free-running counter (FRC) before the TCSR is read. A compare match will not occur until the FRC is counted up. In example 2, an OCR write cycle is executed immediately before and after TCSR read. A compare match will not occur until a match occurs between the contents of the FRC and the OCR write data.

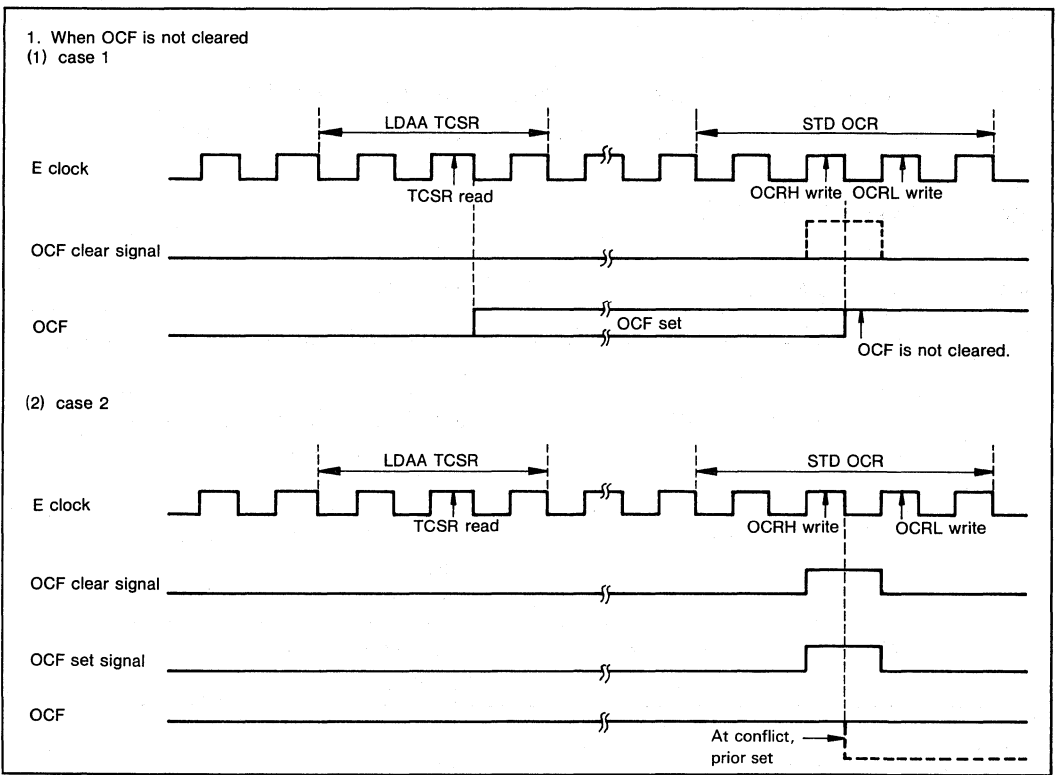


Figure 4-5. OCF Clearing Problems

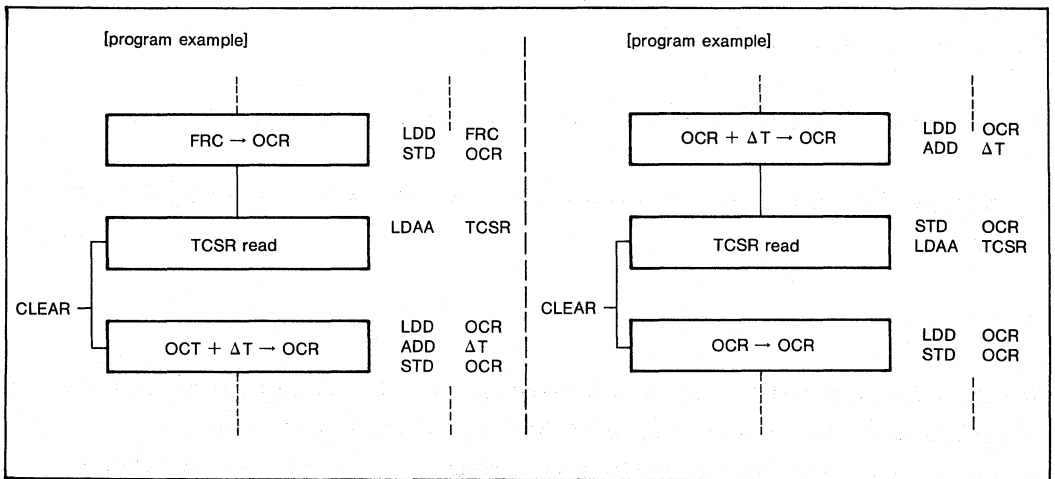


Figure 4-6 Clearing the OCF

## Section 5. Timer 2

In addition to timer 1, the HD6301X0, HD6303X, and HD63701X0 have an 8-bit reloadable timer for counting external events, timer 2. Timer 2 has a timer output, so the MCU can generate three independent waveforms.

Timer 2 has the following components (figure 5-1).

- Control/status register 3 (7 bits)
- Upcounter (8 bits)
- Time constant register (8 bits)

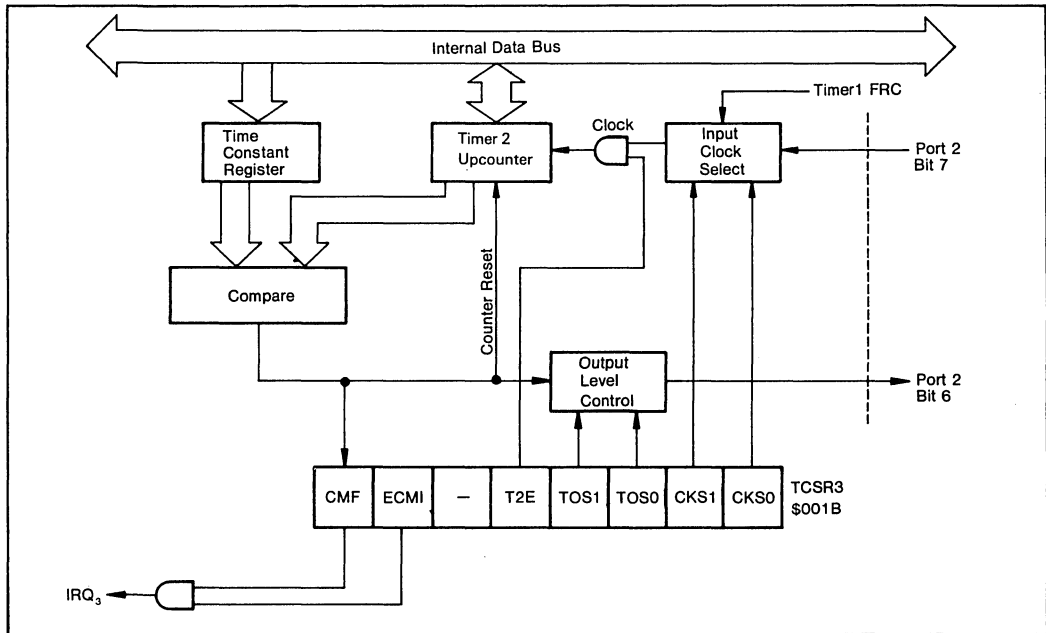


Figure 5-1. Timer 2 Block Diagram

### 5.1 Timer 2 Upcounter (T2CNT)

The 8-bit upcounter is located at \$001D. It operates from the clock input selected by CKS0 and CKS1 of TCSR3. The counter can always be read without being affected. In addition, it can be written to at any time, even during counting.

The counter is cleared when the counter value matches the time constant register (TCONR) value, or during reset.

If the CPU writes to the counter during a cycle when it is being cleared, it will not be cleared, but will take the value written by the CPU.

## 5.2 Time Constant Register (TCNR)

The 8-bit write-only time constant register is located at \$001C. It is always being compared to the upcounter.

When it matches, the counter match flag (CMF) of the timer control/status register 3 (TCSR3) is set. P2<sub>6</sub> will then output the value selected by TOS0 and TOS1 of the TCSR3. When the CMF is set, the counter will be cleared simultaneously and start counting from \$00. This enables regular interrupts and waveform output without any software attention. TCONR is set to \$FF during reset.

When a write-only register like TCONR is read by the MCU, \$FF always appears on the data bus. Whenever the MCU performs an arithmetic or logic operation between memory, and a write-only register, the result will be \$FF.

## 5.3 Timer Control/Status Register 3 (TCSR3)

The 7-bit timer control/status register is located at \$001B (figure 5-2). All bits can be read and all bits can be written except CMF (bit 7). TCSR3 is cleared at reset.

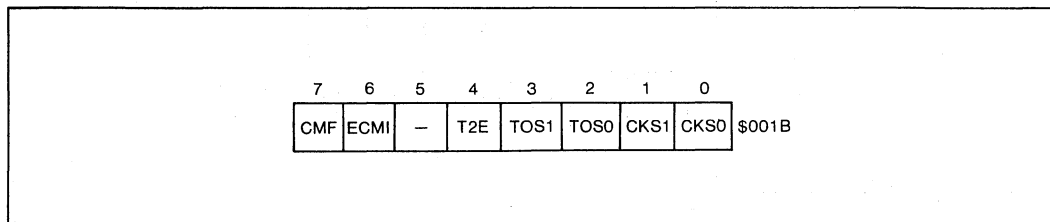


Figure 5-2. Timer Control/Status Register 3

### 5.3.1 Input Clock Select 0 and 1 (CKS0, CKS1)

CKS0 and CKS1 select the counter clock as shown in table 5-1. When the external clock is selected, the rising edge of P2<sub>7</sub> increments the counter. The external clock's frequency can be up to one-half the system clock frequency. If the E clock divided by 8 or 128 is selected, the clock comes from timer 1, so do not write to the FRC. Bits 0 and 1.

Table 5-1. Input Clock Select

CKS1	CKS0	Input Clock
0	0	E clock
0	1	E/8
1	0	E/128
1	1	External clock (P2 <sub>7</sub> )

### 5.3.2 Timer Output Select 0 and 1 (TOS0, TOS1)

When the upcounter matches TCONR, timer 2 will output to P2<sub>6</sub> as selected by TOS0 and TOS1 (table 5-2). When TOS0 and TOS1 are 0, P2<sub>6</sub> will be an I/O port. When toggle output is selected, the P2<sub>6</sub> output level reverses each time the upcounter and TCONR match. This produces a 50% duty cycle square wave at P2<sub>6</sub> without software support. Bits 2 and 3.

Table 5-2. Timer 2 Output Select

TOS1	TOS0	Timer Output
0	0	Timer output inhibited
0	1	Toggle output
1	0	Output 0
1	1	Output 1

### 5.3.3 Timer 2 Enable (T2E)

When T2E is cleared to 0, the clock input to the upcounter is inhibited, and the upcounter stops. When T2E is set, the clock selected by CKS0 and CKS1 is input to the upcounter. Bit 4.

Note: P2<sub>6</sub> outputs 0 when T2E bit is 0 and timer 2 is enabled by TOS0 and TOS1. It also outputs 0 when T2E is 1 and timer 2 is output enabled before the first match occurs.

### 5.3.4 Enable Counter Match Interrupt (ECMI)

Setting ECMI to 1 enables CMI to trigger an internal interrupt (IRQ<sub>3</sub>). When ECMI is cleared, the interrupt is inhibited. Bit 6.

### 5.3.5 Counter Match Flag (CMF)

The read-only CMF bit is set when the upcounter matches the TCONR. It is cleared by writing a zero to it. Bit 7.

### 5.4 Timer Status Flag

Table 5-1 shows set and clear condition of each status flag in timer 2.

If flag set and clear condition occurs at the same time, timer 2 flag will be set.

Table 5-1. Timer 2 Status Flag

Flag	Set Condition	Clear Condition
CMF	· T2CNT = TCONR	· Write 0 to CMF · RES = 0

# Section 6. Serial Communications Interface

The serial communications interface (SCI) has two operating modes: asynchronous with NRZ format, and clock synchronous. The synchronous mode transfers data synchronized with the serial clock.

The SCI has the following components (figure 6-1).

- Transmit/receive control/status register (TRCSR)
- Rate/mode control register (RMCR)
- Receive data register (RDR)
- Receive data shift register (RDSR)
- Transmit data register (TDR)
- Transmit data shift register (TDSR)

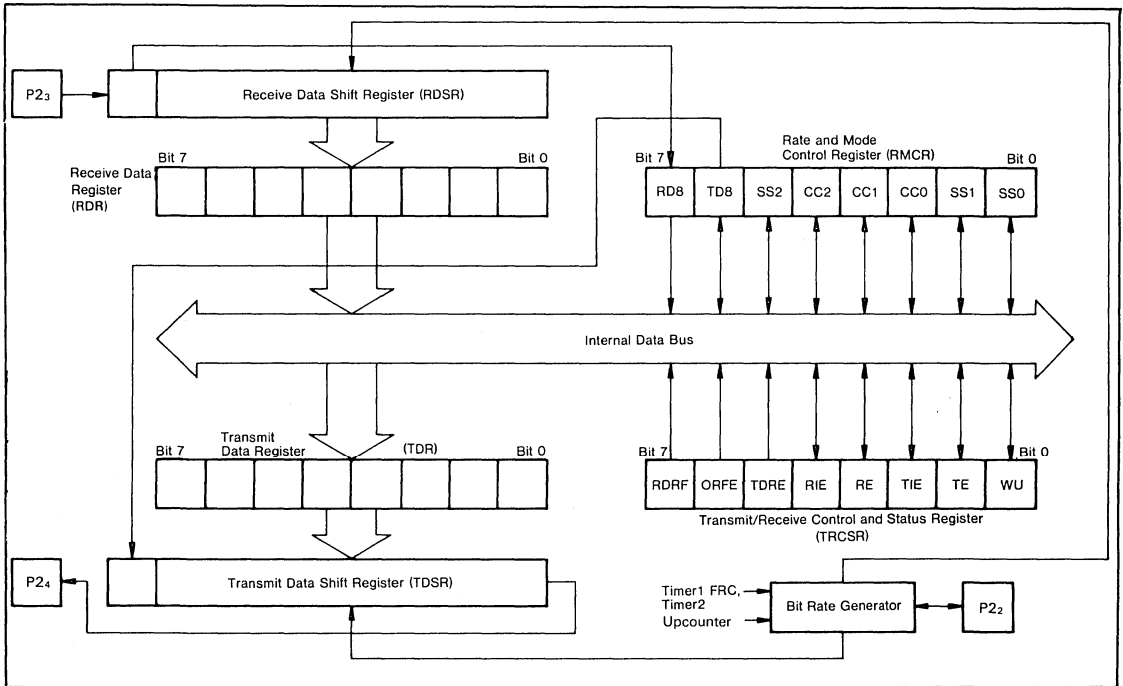


Figure 6-1. SCI Block Diagram

## 6.1 Initialization

The serial I/O hardware must be initialized by the software for operation. The usual procedure follows.

1. Write the desired operating mode to the RMCR.
2. Write the desired operating mode to the TRCSR.



The TE and RE bits may only be set when P2<sub>3</sub> and P2<sub>4</sub> are used for serial I/O only. But TE and RE should be 0 when you set the baud rate and operating mode. Clearing and setting TE and RE again must take more than one cycle at the current baud rate. If they are set within less than 1 cycle, transmit/receive initialization may fail.

## 6.2 Asynchronous Mode

The asynchronous mode has two data formats:

- 1 start bit + 8 data bits + 1 stop bit
- 1 start bit + 9 data bits + 1 stop bit

In addition, the ninth bit can be set to 1 in the 9-bit format to form a third format:

- 1 start bit + 8 data bits + 2 stop bits

### 6.2.1 Asynchronous Transmission

Setting TE in the TRCSR enables transmission. P2<sub>4</sub> becomes the serial output port regardless of the state of bit 4 of the DDR.

Both RMCR and TRCSR should be set to the desired operating conditions. When TE is set, a 10-bit preamble (8-bit format) or 11-bit preamble (9-bit format) will be sent. When it is being sent, internal synchronization will stabilize, and the transmitter will become ready to send.

At this point, if the TDR is empty (TDRE = 1), all 1's will be output, to indicate the idle state. If the TDR contains data (TDRE = 0), the data is sent to the transmit data shift register, and transmission begins.

During transmission, first a 0 start bit is sent. Then 8 or 9 bits of data, starting at bit 0, are transmitted, followed by a stop bit of 1.

When the TDR is empty, hardware sets the TDRE flag bit. If the CPU doesn't respond to the TDRE flag before the next normal transfer should start, the transmitter sends 1's (instead of the 0 start bit) until data is provided to the data register. While the TDRE is set, the transmitter will not send a 0.

### 6.2.2 Asynchronous Reception

Setting the RE bit enables reception. P2<sub>3</sub> becomes the serial input port, regardless of the state of bit 3 of the DDR. The contents of TRCSR and RMCR select the data receive operating mode. The first 0 (space) synchronizes the receive bit flow. Each bit of the following data will be strobed in the middle.

If the stop bit is not 1, the receiver assumes a framing error, and sets ORFE. When a framing error occurs, the receiver transfers the data to the receive data register and the CPU can read the data that caused the error. This makes it possible to detect line breaks.

If the stop bit is 1, the data is transferred to the receive data register and the interrupt flag RDRF is set. If the RDRF is still set when the stop bit of the next data is received, the receiver sets ORFE to indicate an overrun.

When the CPU reads the RDR in response to the RDRF or ORFE in the TRCSR, the RDRF or ORFE bit is cleared to 0.

### 6.2.3 Asynchronous Clock Source

When using an internal clock for asynchronous serial I/O, keep the following in mind:

- Set CC1 and CC0 to 1 and 0, respectively (table 6-3).
- A clock will be generated regardless of the value of TE and RE.
- The maximum clock rate is  $E/16$ .
- The output clock rate is the same as the bit rate.

When using an external clock, keep the following in mind:

- Set CC1 and CC0 to 1 and 1, respectively.
- The external clock frequency should be set to 16 times the baud rate.
- Maximum clock frequency is that of the system clock

### 6.3 Clock Synchronous Mode

In the clock synchronous mode, data transmission is synchronized with a clock pulse. The SCI has a fully independent transmitter and receiver, which make full duplex asynchronous operation possible.

Therefore, in synchronous mode, the only clock I/O pin is P2<sub>2</sub>, so simultaneous transmit and receive is not available. In synchronous mode, TE and RE should not be set to 1 at the same time. Figure 6-2 is the clock and data format for synchronous mode.

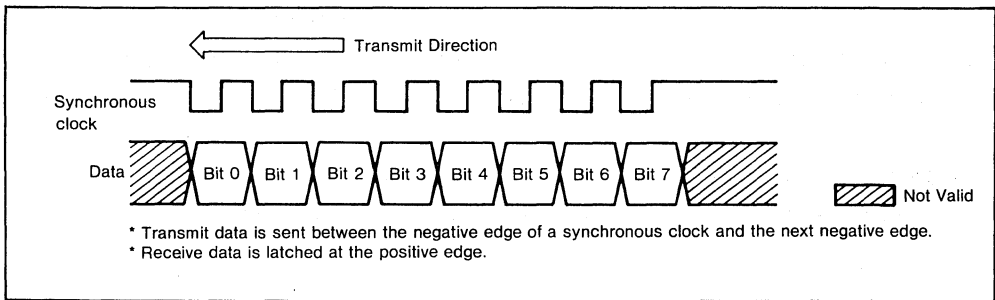


Figure 6-2. Clock Synchronous Mode

### 6.3.1 Synchronous Transmission

Setting the TE bit in the TRCSR enables transmission. P2<sub>4</sub> becomes the serial output port regardless of bit 4 in the DDR. Both the TRCSR and RMCR should be set to the desired operating conditions for transmission.

When external clock input is selected, data is transmitted under the TDRE flag 0 from P2<sub>4</sub>, synchronized with 8 clock pulses input to P2<sub>2</sub>. Data is transmitted from bit 0, and TDRE is set when the transmit data shift register is empty. More than 8 external clock pulses are ignored.

When the transmitter is selected to output the clock, the SCI outputs the clock and synchronous data when the TDRE flag is cleared.

### 6.3.2 Synchronous Reception

Setting the RE bit enables data reception. P2<sub>3</sub> becomes the serial input port regardless of bit 3 in the DDR. TRCSR and RMCR select the data reception operating mode.

If external clock input is selected, the RE bit should be set while the clock signal at P2<sub>2</sub> is high. After the RE bit is set, 8 external clock pulses and synchronized bits of receive data are input at P2<sub>2</sub> and P2<sub>3</sub> respectively. The SCI puts a bit of data into the receive data shift register at every clock pulse, and sets the RDRF flag after 8 bits have been received. More than 8 pulses are ignored. When the CPU reads the received data, RDRF is cleared, and the SCI starts receiving the next data. Clear RDRF when P2<sub>2</sub> is high.

When the receiver is selected to output the clock, 8 clocks are output to P2<sub>2</sub> when the RE bit is set. The receive data should appear at P2<sub>3</sub> synchronously with this clock. When the first byte of data is received, the SCI sets the RDRF flag. To receive the next byte, clear the RDRF flag to start the clock and start receiving.

## 6.4 Transmit/Receive Control Status Register (TRCSR)

The TRCSR is located at \$0011 (figure 6-3). All 8 bits can be read, and bits 0-4 can be written to. TRCSR is initialized to \$20 during reset.

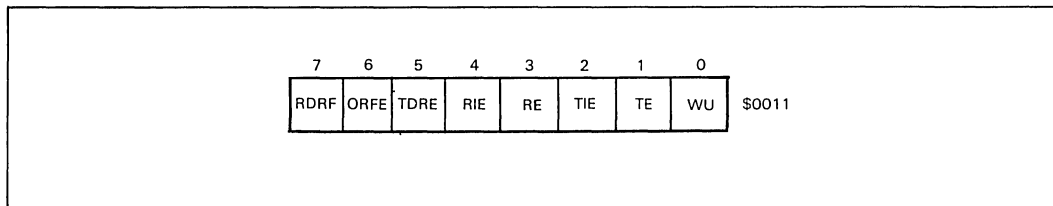


Figure 6-3. Transmit/Receive Control Status Register

### 6.4.1 Wake-Up (WU)

In a typical multiprocessor configuration, the software protocol provides the destination address as the first byte of a message. The wake-up function allows uninterested MCU's to ignore the rest of the message. When the WU bit is set, the SCI stops receiving data until the next message.

The wake-up function is triggered by one frame length of consecutive 1's (10 bits for 8-bit data, 11 bits for 9-bit data). This function is only available in asynchronous mode. Do not set WU in clock synchronous mode. Receiving these consecutive 1's wakes up the SCI and clears WU. The SCI starts receiving data. The RE flag should be set before WU is set. Bit 0.

### 6.4.2 Transmit Enable (TE)

When TE is set, transmit data will appear at P2<sub>4</sub> after a 1-frame preamble in asynchronous transmission, or immediately in clock synchronous transmission. P2<sub>4</sub> will be the serial output regardless of the state of bit 4 of port 2's DDR. If TE is cleared, serial I/O doesn't affect P2<sub>4</sub>. Bit 1.

### 6.4.3 Transmit Interrupt Enable (TIE)

Setting TIE enables TDRE to trigger an internal interrupt (IRQ<sub>3</sub>). Clearing TIE inhibits the interrupt. Bit 2.

### 6.4.4 Receive Enable (RE)

Setting RE inputs the signal at P2<sub>3</sub> regardless of the state of bit 3 of port 2's DDR. When RE is cleared, serial I/O doesn't affect P2<sub>3</sub>. Bit 3.

#### 6.4.5 Receive Interrupt Enable (RIE)

Setting RIE enables RDRF or ORFE (TRCSR bit 6 or 7) to trigger an internal interrupt (IRQ<sub>3</sub>). Clearing RIE inhibits the interrupt. Bit 4.

#### 6.4.6 Transmit Data Register Empty (TDRE)

In asynchronous mode, the SCI sets TDRE when the TDR is transferred to the TDSR. In the clock synchronous mode, SCI sets TDRE when the TDSR is empty. TDRE is reset by reading the TRCSR and writing new transmit data to the transmit data register. TDRE is set to 1 at reset. Bit 5, read only.

#### 6.5.7 Overrun/Framing Error (ORFE)

The SCI sets ORFE when an overrun or framing error is generated during data receive. An overrun error occurs when new receive data is ready to be transferred to the RDR, and RDRF is still set. A framing error occurs when a stop bit is not 0. ORFE is only affected in asynchronous mode. Reading the RDR after reading the TRCSR clears the ORFE. It is cleared at reset. Bit 6, read only.

#### 6.4.8 Receive Data Register Full (RDRF)

RDRF is set when the RDSR is transferred to the RDR. Reading the RDR after reading the TRCSR clears the RDRF. It is cleared at reset. Bit 7, read only.

Note: When more than 1 of bits 5, 6, and 7 are set, one TRCSR read will clear them all. It is not necessary to read the TRCSR once for each bit.

### 6.5 Transmit Rate/Mode Control Register (RMCR)

The RMCR (figure 6-4) controls the following for serial I/O:

- Baud rate
- Clock source
- Data format
- P2<sub>2</sub> function

In addition, if the 9-bit asynchronous format is used, RMCR holds the ninth bit. All bits can be read, and all bits can be written to, except bit 7 (RD8).

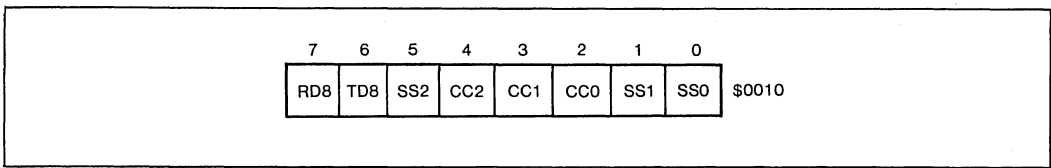


Figure 6-4. Transfer Rate/Mode Control Register

### 6.5.1 Speed Select (SS0, SS1, SS2)

SS0-SS2 control the baud rate used for the SCI. Table 6-1 lists the available baud rates. The timer 1 FRC (SS2 = 0) and the timer 2 upcounter (SS2 = 1) provide the internal clock to the SCI. When SS2 is set, timer 2 functions as the baud rate generator. Timer 2 generates a baud rate dependent on TCONR as shown in table 6-2. Bits 0, 1, and 5.

Table 6-1. SCI Bit Times and Transfer Rates

#### Asynchronous

SS0	SS1	SS2	XTAL	2.4576 MHz	4.0 MHz	4.9152 MHz
			E	614.4 kHz	1.0 MHz	1.2288 MHz
0	0	0	E/16	26 μs/38400 baud	16 μs/62500 baud	13 μs/76800 baud
0	0	1	E/128	208 μs/4800 baud	128 μs/7812.5 baud	104.2 μs/9600 baud
0	1	0	E/1024	1.67 ms/600 baud	1.024 ms/976.6 baud	833.3 ms/1200 baud
0	1	1	E/4096	6.67 ms/150 baud	4.096 ms/244.1 baud	3.333 ms/300 baud
1	X	X		Note 1	Note 1	Note 1

**Note:**

- When SS2 = 1, timer 2 is the SCI clock. The baud rate is as follows:  
 Baud rate =  $f/[32(TCONR + 1)]$   
 Where:  
 f = timer 2 input clock frequency  
 TCONR = contents of timer constant register, 0-255

Table 6-1. SCI Bit Times and Transfer Rates (cont.)

## Clock Synchronous (Note 1)

SS2	SS1	SS0	XTAL	4.0 MHz	6.0 MHz	8.0 MHz
			E	1.0 MHz	1.5 MHz	2.0 MHz
0	0	0	E/2	2 μs/bit	1.33 μs/bit	1 μs/bit
0	0	1	E/16	16 μs/bit	10.7 μs/bit	8 μs/bit
0	1	0	E/128	128 μs/bit	85.3 μs/bit	64 μs/bit
0	1	1	E/512	512 μs/bit	341 μs/bit	256 μs/bit
1	X	X		Note 2	Note 2	Note 2

## Notes:

- Bit rates for internal clock operation. External clock can operate from DC to 1/2 system clock frequency.
- When SS2 is 1, timer 2 is the SCI clock. The bit rate is as follows:

$$\text{Bit rate } (\mu\text{s/bit}) = 4(\text{TCONR} + 1)/f$$

Where:

f = timer 2 input clock frequency

TCONR = contents of timer constant register, 0-255

Table 6-2. Baud Rate and Time Constant Register Example

Baud Rate	XTAL Frequency				
	2.4576 MHz	3.6864 MHz	4.0 MHz	4.9152 MHz	8.0 MHz
110 (note 1)	21	32	35	43	70
150	127	191	207	255	51
300	63	95	103	127	207
600	31	47	51	63	103
1200	15	23	25	31	51
2400	7	11	12	15	25
4800	3	5		7	12
9600	1	2		3	
19200	0			1	
38400				0	

## Note:

- E/8 is used as the clock for 110 baud, E is used for all other baud rates.

### 6.5.2 Clock Control/Format Select (CC0, CC1, CC2)

CC0, CC1, and CC2 control the clock source and data format (table 6-3). They are cleared during reset, so the MCU will be in clock synchronous mode with external clock. Therefore, P2<sub>2</sub> starts out as a clock input. To use P2<sub>2</sub> as an output port, set bit 2 of the port 2 DDR to 1 and set CC1 and CC0 to 0, 1. Bits 2, 3, and 4.

Table 6-3. SCI Format and Clock Source Control

CC2	CC1	CC0	Format	Mode	Clock Source	P2 <sub>2</sub>
0	0	0	8-bit data	Clock synchronous	External	Clock input
0	0	1	8-bit data	Asynchronous	Internal	Not used
0	1	0	8-bit data	Asynchronous	Internal	Clock output (note 1)
0	1	1	8-bit data	Asynchronous	External	Clock input
1	0	0	8-bit data	Clock synchronous	Internal	Clock output

Table 6-3. SCI Format and Clock Source Control (continued)

CC2	CC1	CC0	Format	Mode	Clock Source	P2 <sub>2</sub>
1	0	1	9-bit data	Asynchronous	Internal	Not used
1	1	0	9-bit data	Asynchronous	Internal	Clock output (note 1)
1	1	1	9-bit data	Asynchronous	External	Clock input

Note:

1. Clock output regardless of bits TE and RE of TRCSR.

### 6.5.3 Transmit Data Bit 8 (TD8)

When the SCI transmits asynchronous 9-bit data, TD8 is the ninth bit. Write this bit first, then write the eight bits to the transmit data register. Bit 6.

### 6.5.4 Receive Data Bit 8 (RD8)

When the SCI receives asynchronous 9-bit data, RD8 stores the ninth bit. Read this bit first, then read the receive data register. Bit 7.



## 6.6 SCI Receiving Margin

The receiving margin for the SCI is as follows.

Allowable deviation of bit error  $(t - t_0)/t_0 = \pm 43.7\%$

Allowable deviation of character error  $(T - T_0)/T_0 = \pm 4.37\%$

T, T<sub>0</sub>, t, and t<sub>0</sub> are defined in figure 6-5. When a modem is used for communication, waveform distortion may exceed the allowable value, depending on the modem and channel.

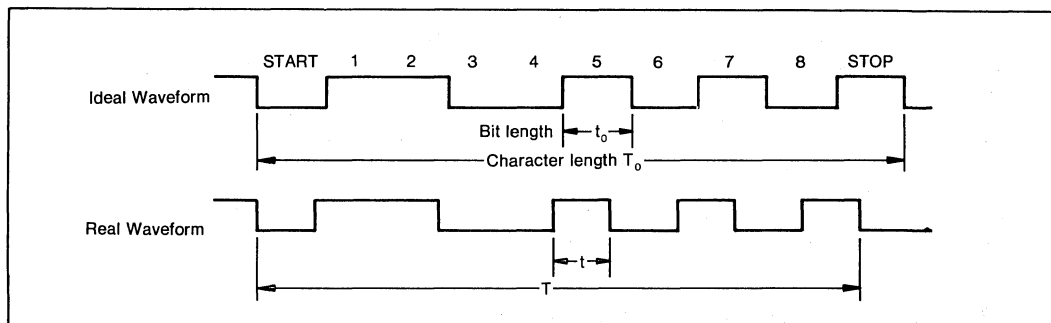


Figure 6-5. Bit and Character Error

## 6.7 SCI Status Flags

Table 6-5 shows set and clear conditions of each status flag in the SCI.

If flag set and clear conditions occur at the same time, the SCI flags will be cleared.

Table 6-5. SCI Status Flags

Flag	Set Condition	Clear Condition
SCI	RDRF · RDSR → RDR	· Read TRCSR, then RDR · $\overline{RES} = 0$
ORFF	· Framing error (async mode). Stop bit = 0 · Overrun error (async mode). RDSR → RDR when RDRF = 1	· Read TRCSR, then RDR · $\overline{RES} = 0$
TDRE	· TDR → TDSR (async mode) · TDSR is empty (clock sync mode)	· Read TRCSR, then write to TDR

## 6.8 Precaution for clock-synchronous serial communication interface

When transmitting through clock-synchronous serial communication interface, TE bit should not be cleared with TDRE of TRCSR (\$11) is "0".

The TDRE set and clear conditions of SCI are shown as follows.

	Set Condition	Clear Condition
TDRE	1. TDR → transmit shift register (asynchronous)	When writing to TDR after TRCSR read, with TDRE = 1, TDRE is cleared.
	2. Transmit shift register is empty. (clock-synchronous)	
	3. $\overline{RES} = 0$ )	

If transmit data is written to TDR, and then TE bit is cleared with TDRE = 0 to stop transmitting, TDRE remains "0".

In this case, even if TE bit is set and transmit data is written again, the TDR data is not transmitted.

Please note that TE bit must be cleared after the last data has been transmitted.

(This caution is not applied to asynchronous serial communication interface.)-



# Section 7. HD63701X0 Programmable ROM (EPROM)

The HD63701X0's on-chip EPROM is programmed in the chip's PROM mode. When  $MP_0$ ,  $MP_1$ , and  $\overline{STBY}$  are low (table 2-2), the HD63701X0 doesn't operate as an MCU. It can be programmed by the same procedure as a standard 2732A EPROM. In the PROM mode, P30-P37 are the data bus, P10-P17 and P40-P43 are the address bus, and P57 is the  $\overline{CE}$  input. See figures 7-1 and 7-2, table 7-1.

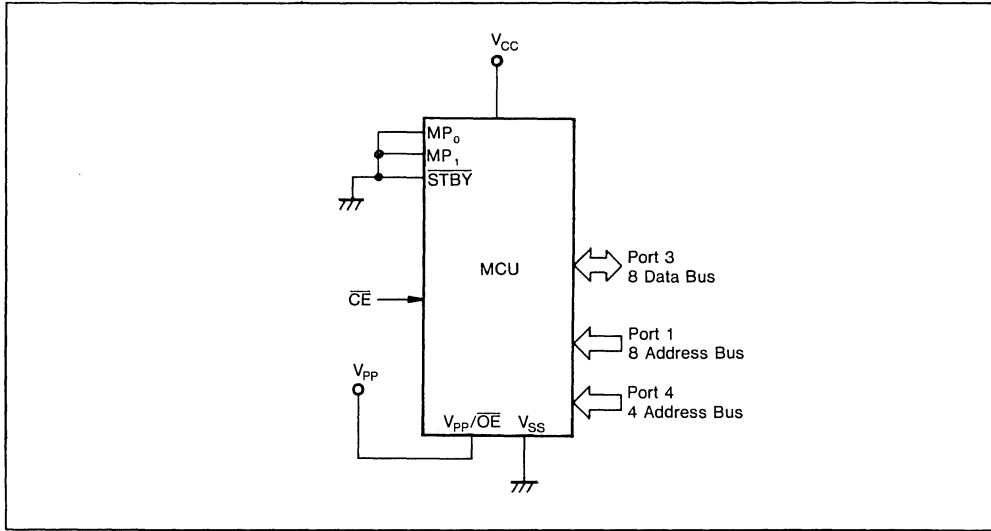
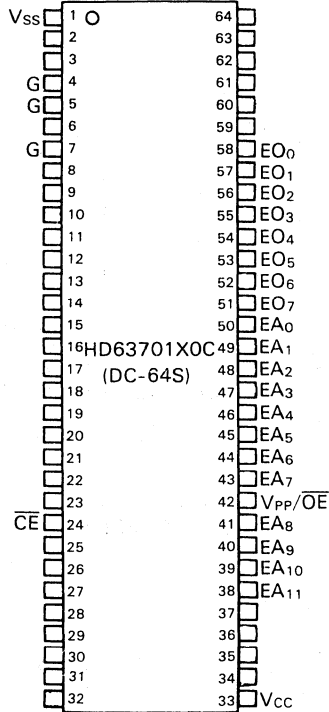


Figure 7-1. PROM Mode

Table 7-1. Pin Conditions in PROM Mode

Pin Name	Pin No.	Programming	Verification	PROM Inhibit
$V_{CC}$	33	+5 V	+5 V	+5 V
$V_{SS}$	1	GND	GND	GND
$V_{PP}/\overline{OE}$	42	$V_{pp}$	Low	Don't care
$\overline{CE}$	24	Low	Low	High
P30-P37	51-58	Data input	Data output	High impedance
P10-P17 P40-P43	43-50 38-41	Address input	Address input	Don't care
$MP_0$ , $MP_1$ , $\overline{STBY}$	4, 5, 7	Low	Low	Low
Other pins		GND	GND	GND



G: Ground (V<sub>SS</sub> level)

Figure 7-2. PROM Mode Pin Arrangement

Table 7-2 shows the recommended combinations of PROM programmers and socket adapters for programming the HD63701X0. The socket adapter converts the pin assignment of the necessary 24 pins to the same assignment as the standard EPROM.

Table 7-2. PROM Programmers and Socket Adapters

Programmer		Socket Adapter	
Data I/O	121A/121B, 22A/22B, 29A/29B	Data I/O	HD63701X0 (for 29A/29B)
		Hitachi	H67PWA01A

## 7.1 Programming and Verification

When the  $\overline{CE}$  pin is held low after the programming voltage ( $V_{PP}$ ) is applied, data can be programmed in PROM one byte at a time through port 3. To verify the data, hold the  $V_{PP}/\overline{OE}$  and  $\overline{CE}$  pins low after programming, and the programmed data will be output from port 3.

When  $\overline{CE}$  is returned high, port 3 will be high impedance, and PROM programming/verification will be inhibited.

Programming precautions: The PROM memory cells should be programmed under specific voltage and timing conditions. The higher the program voltage and the longer the program pulse is applied, the more electrons will be injected into the floating gate. However, if an overvoltage is applied to  $V_{PP}$ , the p-n junction may be permanently damaged. Pay particular attention to PROM programmer overshoot. Negative voltage noise will cause a parasitic transistor effect, which may reduce breakdown voltage.

The HD63701X0 is connected electrically to the PROM programmer through a socket adapter. Therefore, pay attention to the following:

1. Confirm that the socket adapter is firmly fixed on the PROM programmer.
2. Do not touch the socket adapter or the LSI during programming. Mis-programming can be caused by poor contacts.

## 7.2 Erasing (Window Package)

The EPROM is erased by exposing the LSI to ultraviolet light. All erased bits are in 1's.

The conditions for erasing are: ultraviolet light with wavelength of  $2537 \text{ \AA}$ , and a minimum irradiation of  $15 \text{ W} \cdot \text{s}/\text{cm}^2$ . These conditions are satisfied by exposing the LSI to an ultraviolet light rated at  $12,000 \mu\text{W}/\text{cm}^2$  for 15-20 minutes, at a distance of 1 inch.

## 7.3 Characteristics and Applications

### 7.3.1 Principles of Programming/Erasing

The HD63701X0's memory cells are the same as an EPROM's. Therefore they are programmed by applying high voltage to control gates and drains, which injects hot electrons into the floating gate (figure 7-3). The condensed electrons in the floating gate are stable, surrounded by an energy barrier of  $\text{SiO}_2$  film. Such a cell becomes a 0 bit due to the memory threshold voltage change. A cell with no condensed electrons at its floating gate appears as a 1 bit.

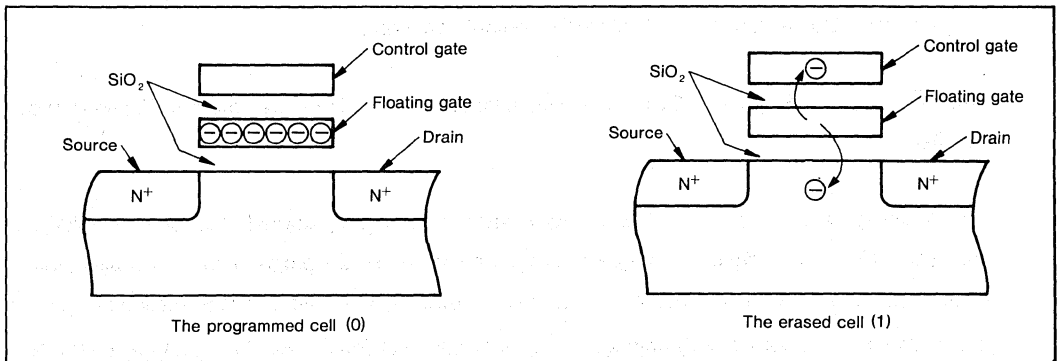


Figure 7-3. Cross-Section of EPROM Memory Cell

The electron charge in memory cells may decrease as time goes by. This can be caused by:

1. Ultraviolet light, discharged by photo-emitting electrons (erasure principle)
2. Heat, discharged by thermal emitting electrons
3. High voltage, discharged by a high electric field at the control gate or drain

If the oxide film covering a floating gate is defective, the erasure rate is great. Normally, electron erasure does not occur, because such defective devices are found and removed during testing.

### 7.3.2 Window-Type Package Precautions

**Glass Erasure Window:** If the glass window comes in contact with plastic or anything with a static charge, the LSI may malfunction due to the electrostatic charge on the surface of the window. If this occurs, exposing the LSI to ultraviolet light for a few minutes neutralizes the charge, and restores the LSI to normal operation. However, charge stored in the floating gate decreases at the same time, so reprogramming is recommended.

Electrostatic charge buildup on the window is a fundamental cause of malfunctions. Measures for its prevention are the same as those for preventing electrostatic breakdown:

1. Operators should be grounded when handling equipment.
2. Do not rub the glass window with plastics.
3. Be careful of coolant sprays, which may contain a few ions.
4. The ultraviolet shading label (which includes conductive material) effectively neutralizes charge.



**Ultraviolet Shading Label:** If the LSI is exposed to fluorescent light or sunlight, its memory contents may be erased by the small quantity of ultraviolet light in these sources. In strong light, the MCU may fail under the influence of photocurrent. To prevent these problems, it is recommended that the device be used with an ultraviolet shading label covering the erasure window after programming.

Special labels are sold for this purpose. They contain metal to absorb ultraviolet light. When choosing a label, note the following:

1. Adhesion (mechanical intensity)—Re-use and dust reduce adhesion. Peeling off a label may cause static electricity. Therefore, erasing and rewriting is recommended after peeling. Sticking a new label over the old one is better than replacing a label.
2. Allowable temperature range—The allowable environmental temperature range of the label should be noted. If it is used under conditions outside this range, the paste may stiffen or adhere to the label, causing paste to remain on the window when the label is removed.
3. Moisture resistance—The allowable moisture range and environmental conditions of the label should be noted. It is difficult to find a shade label applicable to all conditions. The proper label should be selected depending on the intended use of the MCU.

# Section 8. Applications

## 8.1 HD6301X0 or HD63701X0 in Expanded Mode

Figure 8-1 shows a microcomputer system using all CMOS peripheral LSI's as an application example of the HD6301X0 or HD63701X0 in the expanded mode (modes 1,2).

Ports 1 and 4 are used for address output, and port 3 is used for data I/O. The system is controlled by directly connecting  $\overline{RD}$  and  $\overline{WR}$  as memory control signals and  $R/\overline{W}$  and  $E$  as peripheral controls.

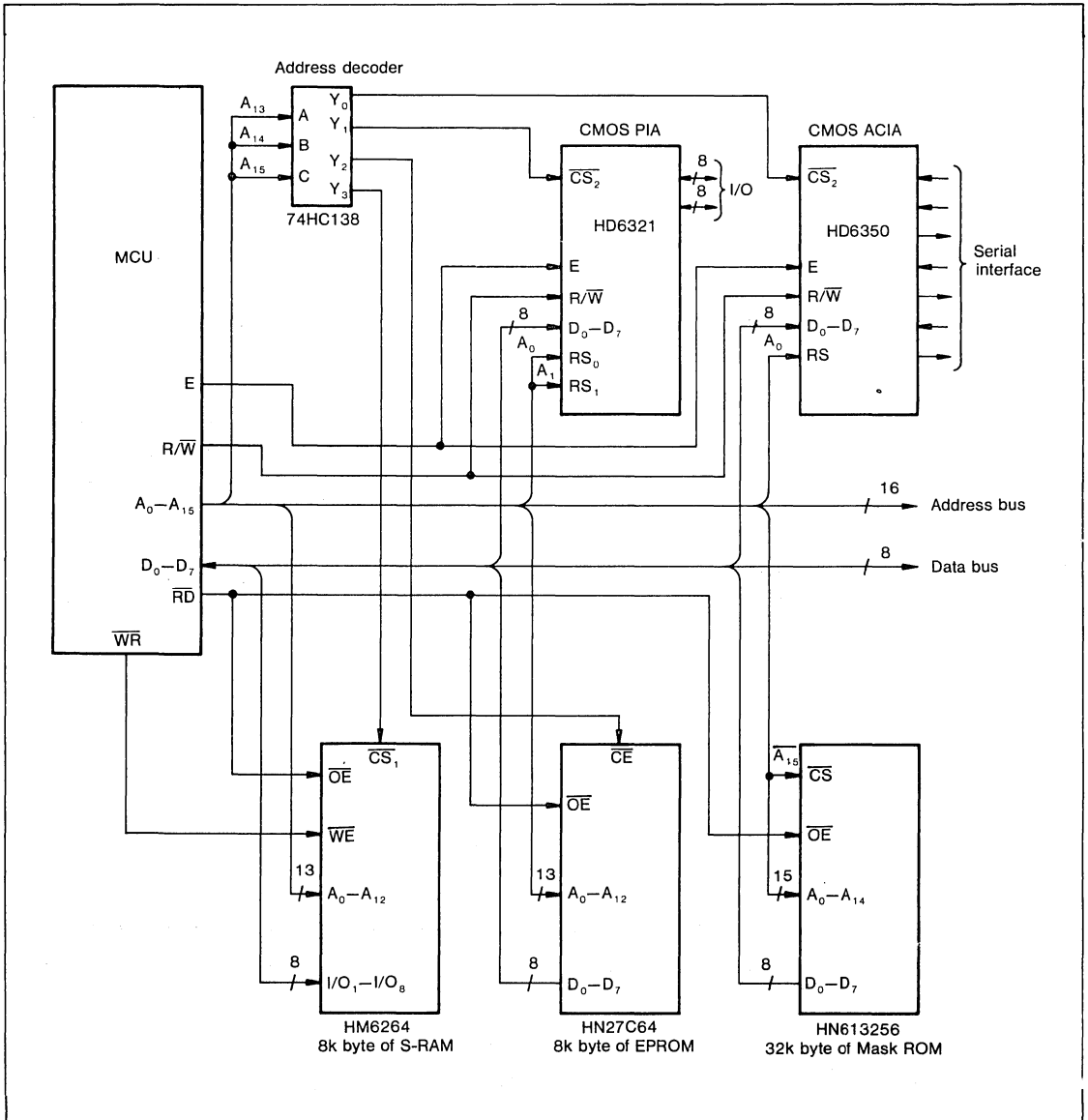


Figure 8-1. All CMOS Microcomputer System



## 8.2 HD6301X0 or HD63701X0 in Single-Chip Mode

Figure 8-2 shows a printer controller using the HD6301X0 or HD63701X0 in the single-chip mode (mode 3).

The HD6301X0 or HD63701X0 controls a 16-dot printer using I/O lines as its ports. Data from the host is transferred to the MCU through the serial interface or through a Centronics interface at port 3.

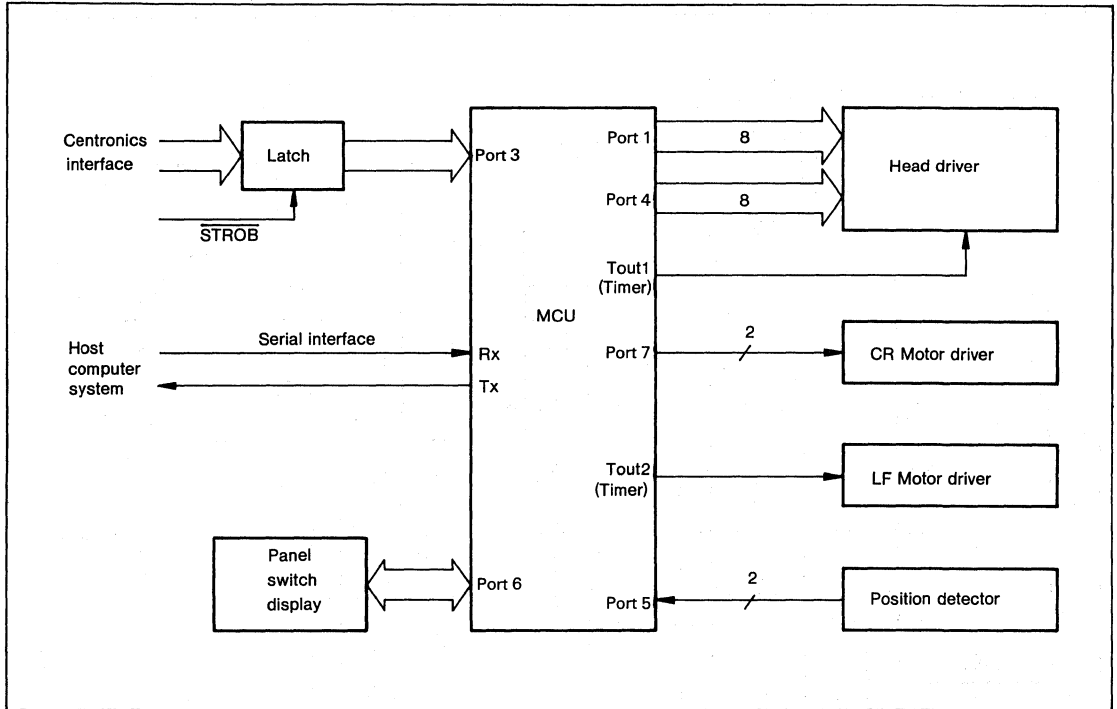


Figure 8-2. Printer Controller

## 8.3 Timer Applications

### 8.3.1 Timer 1

Timer 1 is a 16-bit programmable timer with the same architecture as the timer on the HD6301V1, but with an output compare register added. Timer 1 can perform the following four operations:

1. Waveform generation or interval timing using output compare register 1 (OCR1)
2. Waveform generation or interval timing using output compare register 2 (OCR2)
3. Pulse width or pulse cycle measurement using the input capture register
4. Interval timing with overflow interrupt

**Waveform Generation.** The values of the output compare registers (OCR1, OCR2) are compared with the free-running counter (FRC) at every E cycle. When a match occurs, an output compare flag (OCF1, OCF2) is set. When an output enable bit (OE1E, OE2E) is set, the value of the output level bit (OLVL1, OLVL2) is output at port 2 (Tout1: P2<sub>1</sub>, Tout2: P2<sub>5</sub>). Figure 9-3 is a flowchart for OCR1 waveform generation.

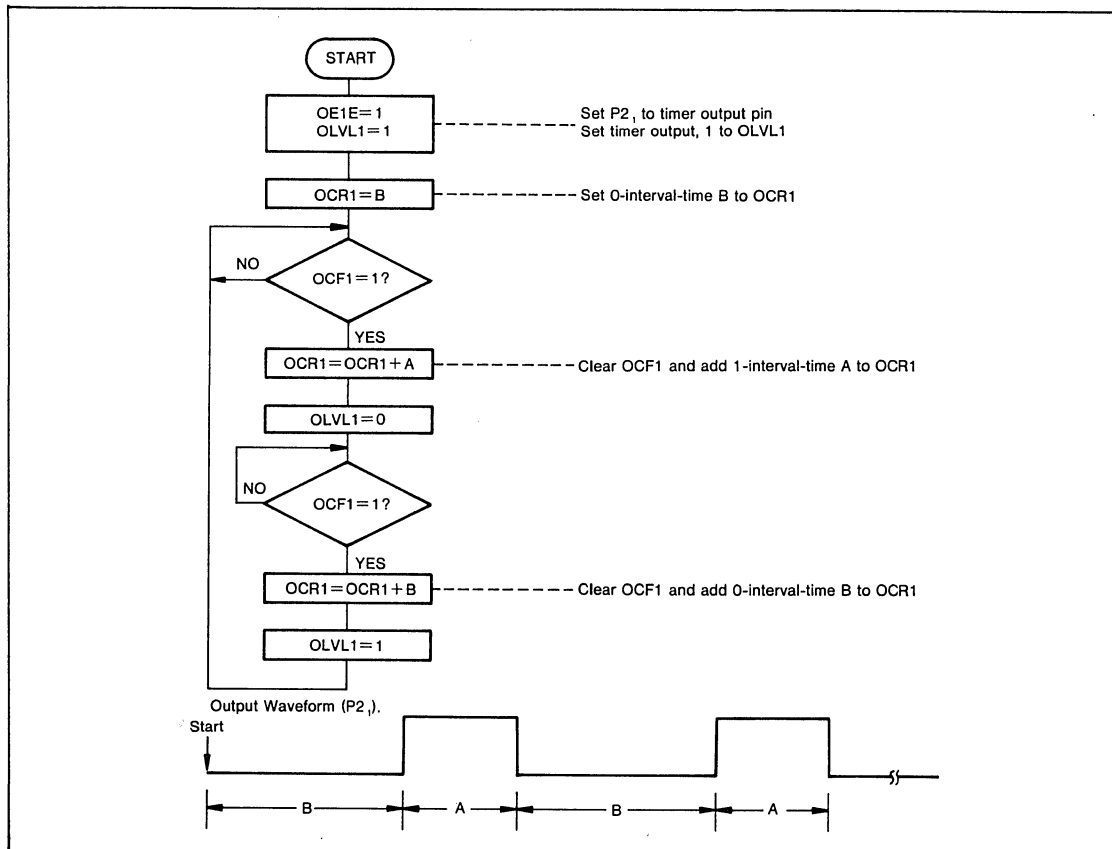


Figure 8-3. OCR1 Waveform Generation

**Pulse Width Measurement.** The input capture register (ICR) latches the free-running counter value at the transition of the external input signal, measuring the pulse width or cycle. Figure 8-4 is a flowchart of pulse width measurement.

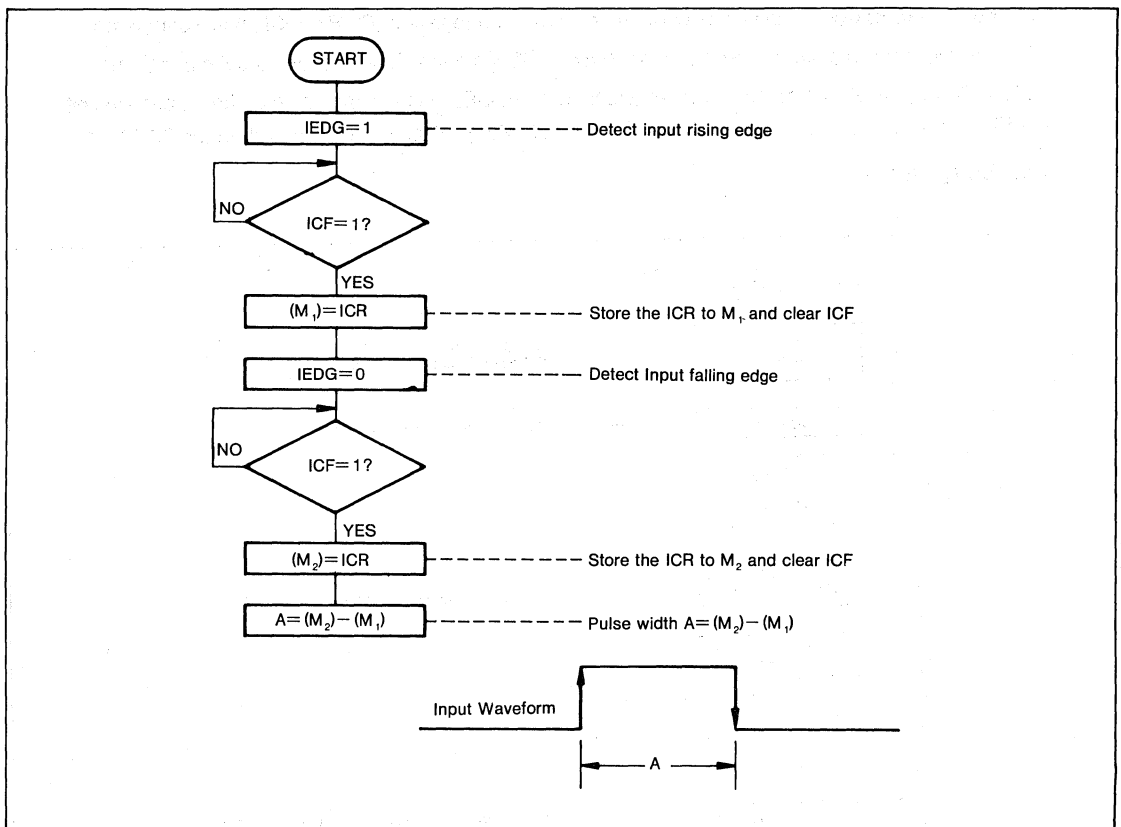


Figure 8-4. ICR Pulse Width Measurement

### 8.3.2 Timer 2

The 8-bit reloadable timer provides such functions as an external event counter, interval timer, waveform generator, and SCI baud rate generator.

**External Event Counter.** Operate timer 2 as an external event counter by setting input clock select, CKS0 and CKS1, to external clock and writing 1 into T2E. The timer 2 upcounter is incremented by the external clock's rising edge. Figure 9-5 shows the routine that generates an interrupt after N external events occur (where N is an integer between 1 and 256).

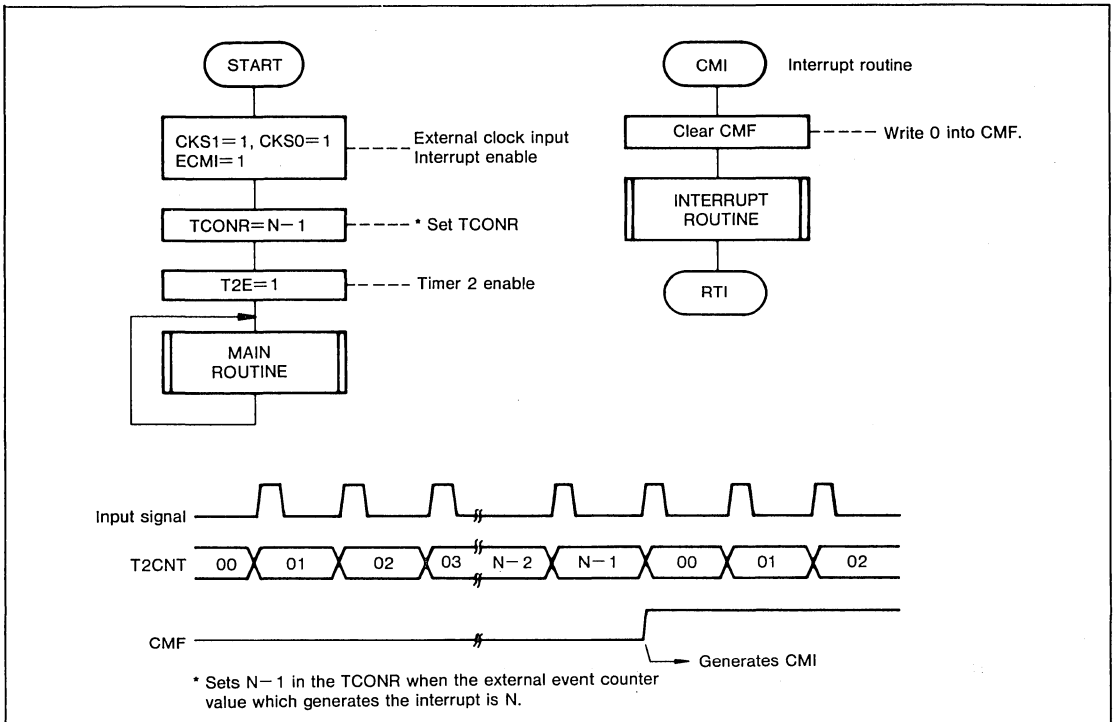


Figure 8-5. External Event Counter

5

**Square-Wave Generator.** Timer 2 can generate a continuous square wave without software supervision. Figure 8-6 shows this routine.

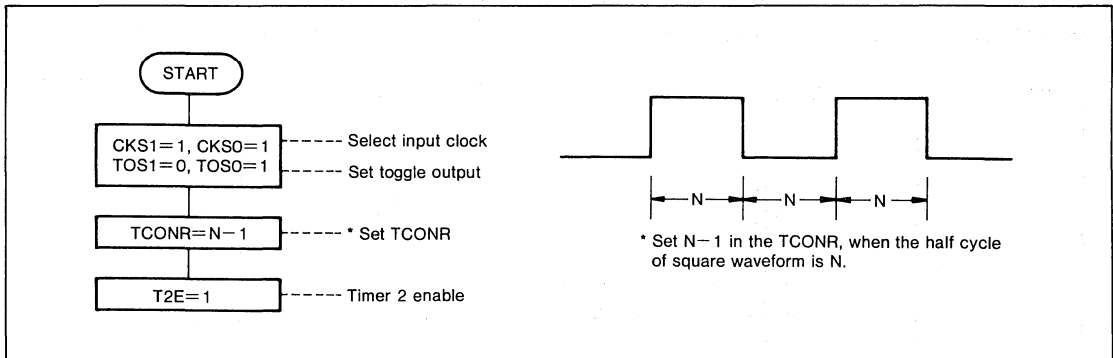


Figure 8-6. Square-Wave Generator

## 8.4 SCI Applications

### 8.4.1 Timer 2 Baud Rate Generator

The SCI can use six kinds of clock source: timer 1's FRC (four kinds), timer 2, and an external clock. The timer 1 baud rate clocks are not adjustable, but timer 2 can provide any baud rate. Figure 8-7 shows how time 2 can provide the baud rate.

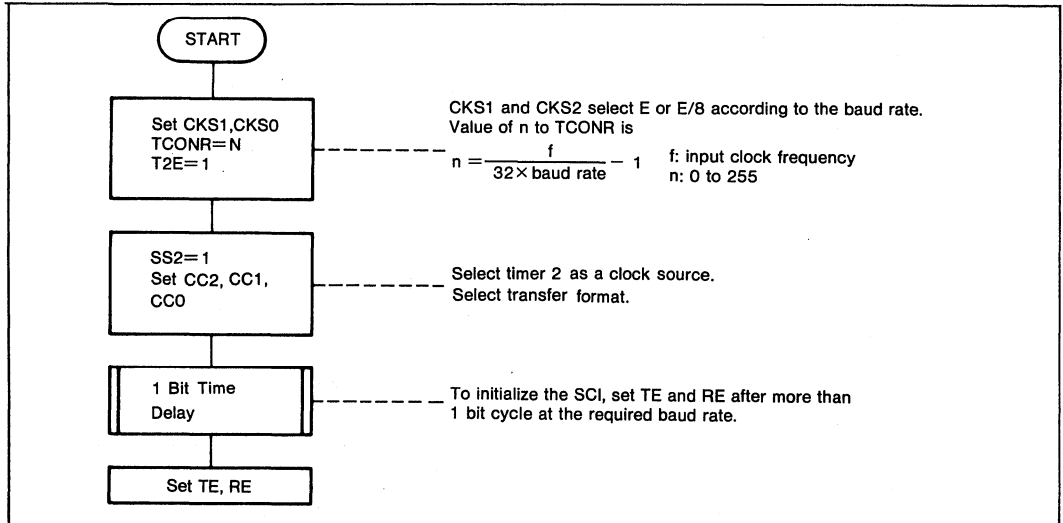


Figure 8-7. Timer 2 as Baud Rate Generator

### 8.4.2 Interface between HD6301X0/HD63701X0 and HD6305X0

An HD6301X0/HD63701X0 can interface to an HD6305X0 in the clock synchronous mode. This gives 99 I/O lines, suitable for systems requiring many I/O lines. Figure 9-8 shows an example of this interface.

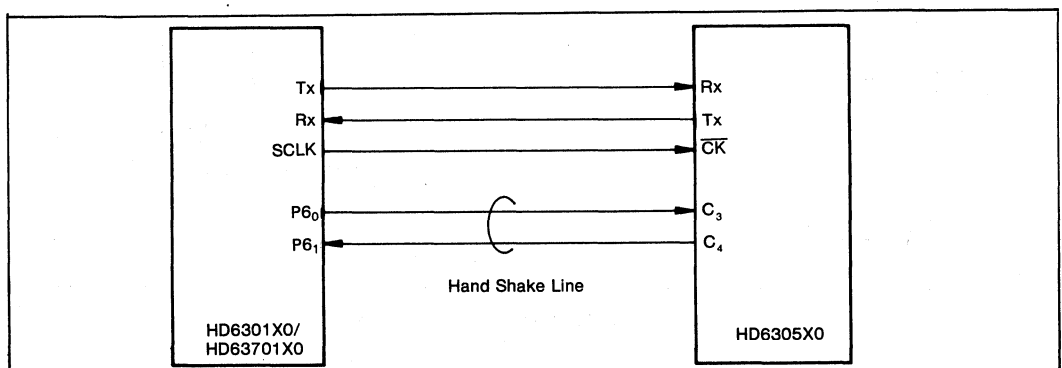


Figure 8-8. HD6301X0/HD63701X0 to HD6305X0 Interface

Employing the clock synchronous mode enables the HD6301X0/HD63701X0 to interface easily to peripheral devices (A/D converter, real-time clock, etc) which use a clock synchronous interface, as well as to the HD6305X0.

### 8.4.3 I/O Expansion

The SCI can be used in the clock synchronous mode to supplement the available parallel I/O ports. Use an external shift register to perform the serial-to-parallel conversion. Figure 8-9 shows this kind of I/O expansion.

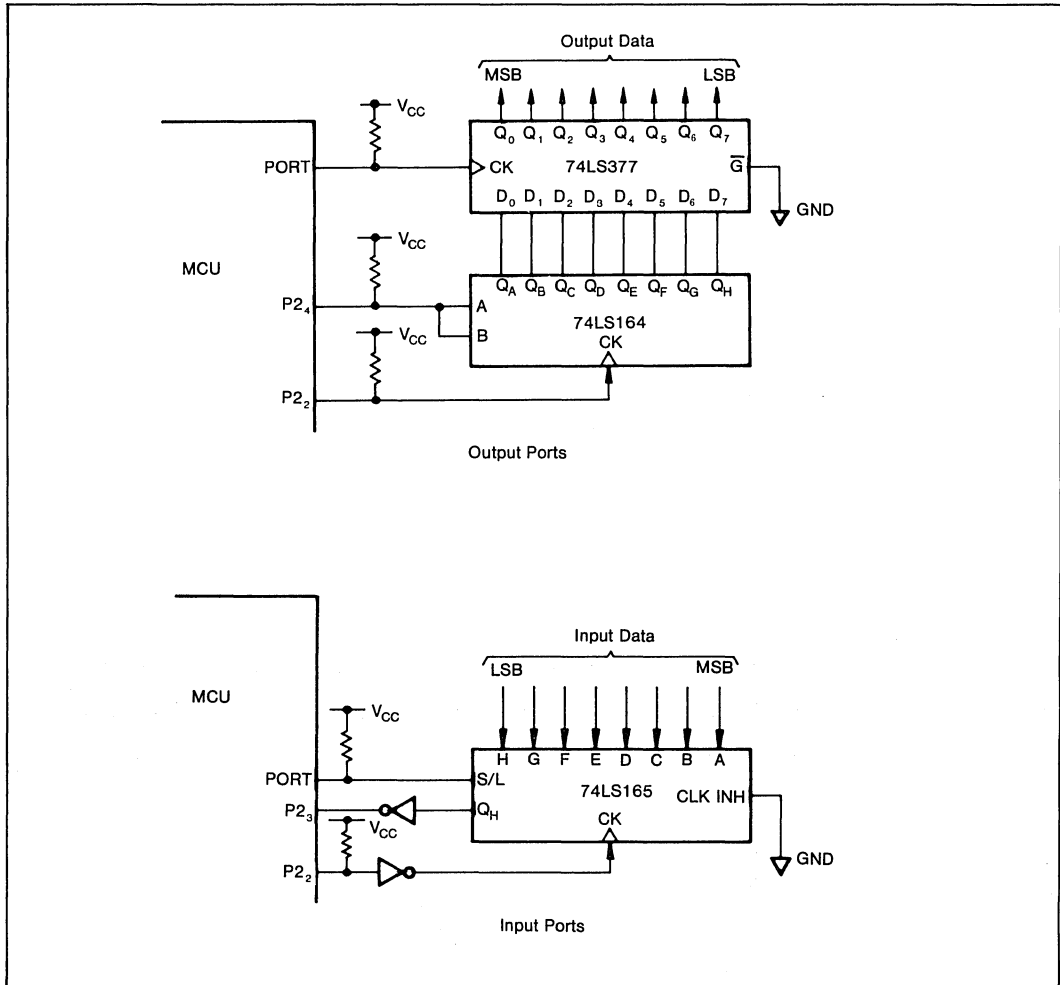


Figure 8.9 I/O Expansion in Clock Synchronous Mode

### 8.4.4 SCI Multiplexer

Use an analog multiplexer as shown in figure 8-10 to use the SCI with both an asynchronous and a clock synchronous device, such as an HD6305X0 and an RS-232C.



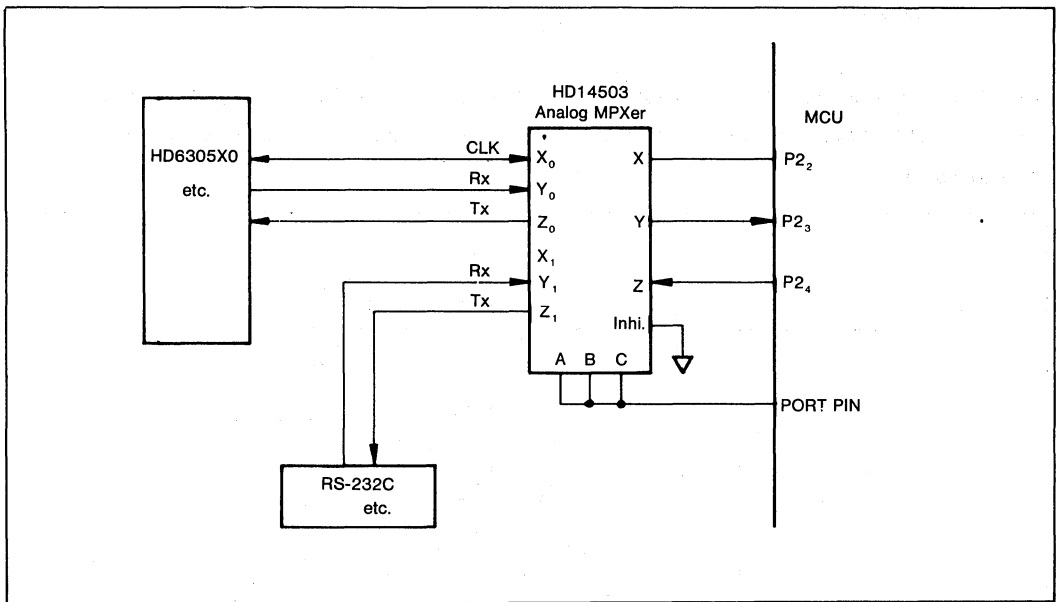


Figure 8-10. Multiplexed SCI

## 8.5 Lowering Operating Current

### 8.5.1 Lowering Operating Frequency

The HD6301X0/HD6303X/HD63701X0 operating current is approximately proportional to the operating frequency (figure 8-11). Therefore, if the system does not require a high-speed MCU, power can be reduced by lowering the operating frequency.

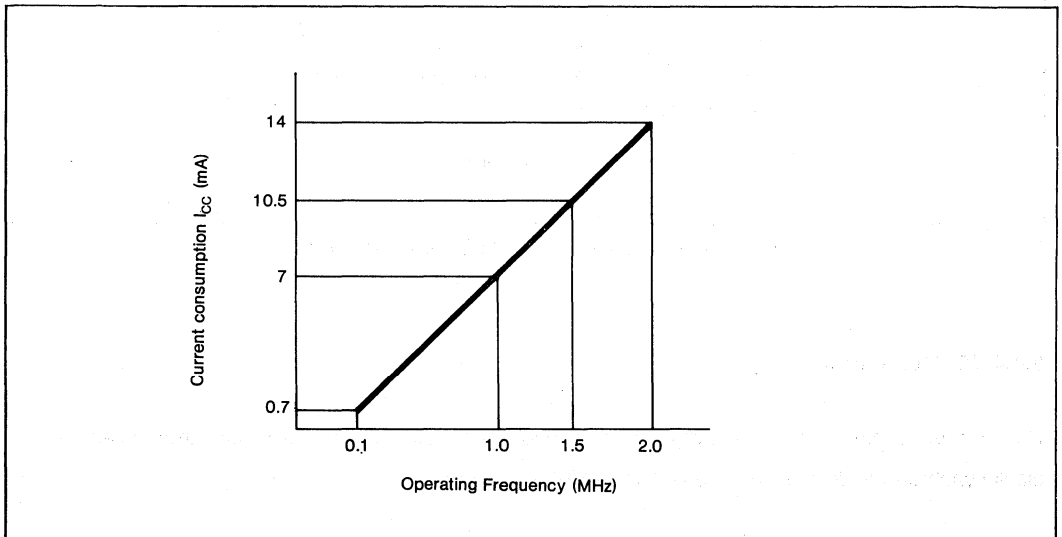


Figure 8-11. Operating Frequency and Current (Typical)

## 8.5.2 Sleep Mode

The SLP instruction puts the MCU into the sleep mode. In the sleep mode, current consumption is reduced to one-fourth to one-fifth of that in the operating state. When the CPU acknowledges an interrupt request, it cancels the sleep mode. The average power consumption can be reduced by putting the CPU in sleep mode whenever it doesn't actually execute any instructions, such as in interrupt wait state or polling. Figure 8-12 shows a routine which wakes the CPU up every 65 ms, using the overflow interrupt of the timer 1 FRC.

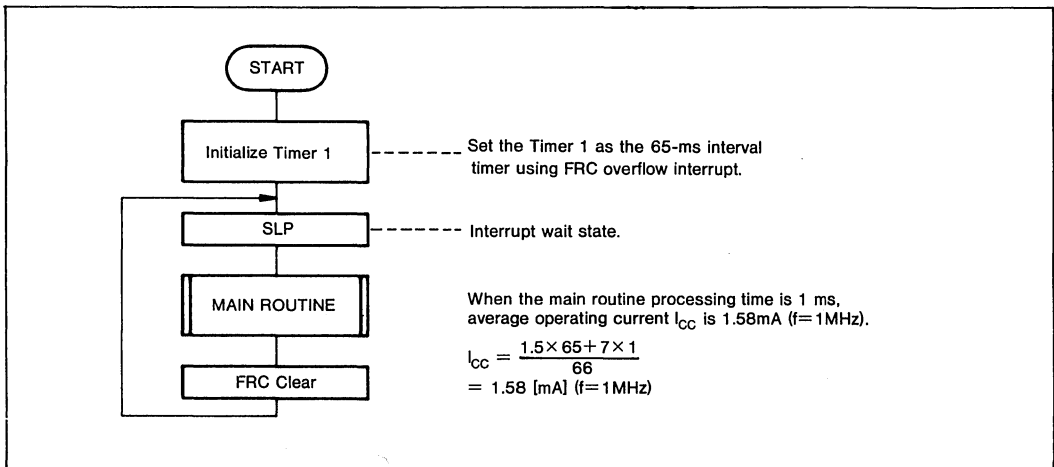


Figure 8-12. Low Power Consumption Using the Sleep Mode

## 8.5.3 Standby Mode

Bringing  $\overline{\text{STBY}}$  (pin 7) low puts the MCU into standby mode. In standby mode, the oscillator stops and the MCU goes into the reset state. The contents of RAM are maintained as long as  $V_{CC}$  is greater than or equal to 2 V. In standby mode, current consumption is reduced to a few  $\mu\text{A}$ . RAM can be maintained by battery.

Bringing  $\overline{\text{STBY}}$  high cancels standby mode. The MCU releases the reset state and starts oscillation.  $\overline{\text{RES}}$  (pin 6) should be held low for at least the oscillation stabilization time ( $t_{RC}$ ) after  $\overline{\text{STBY}}$  high. Figure 8-13 gives an example of a circuit that sets standby from software. Figure 8-14 shows the timing for this circuit, and figure 8-15 is an operating flowchart.

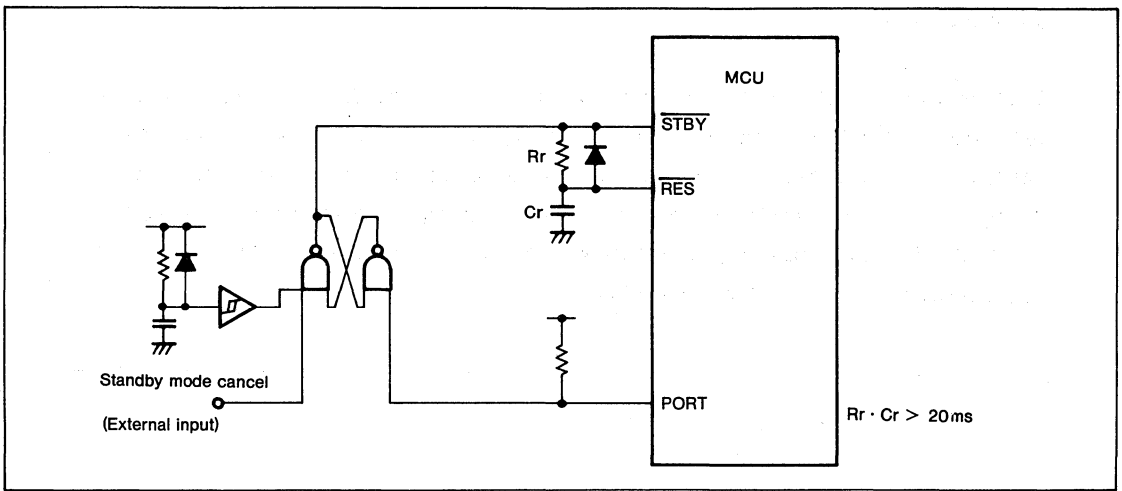


Figure 8-13. Standby Circuit Example

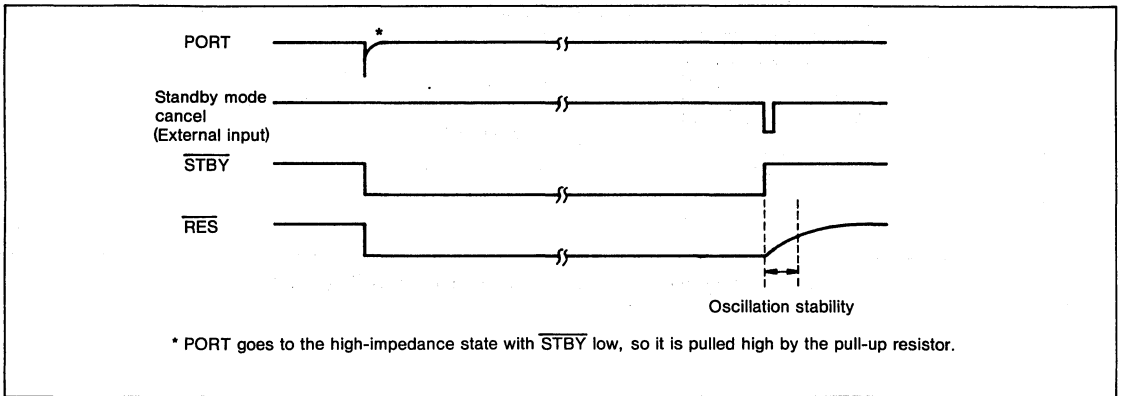


Figure 8-14. Standby Timing

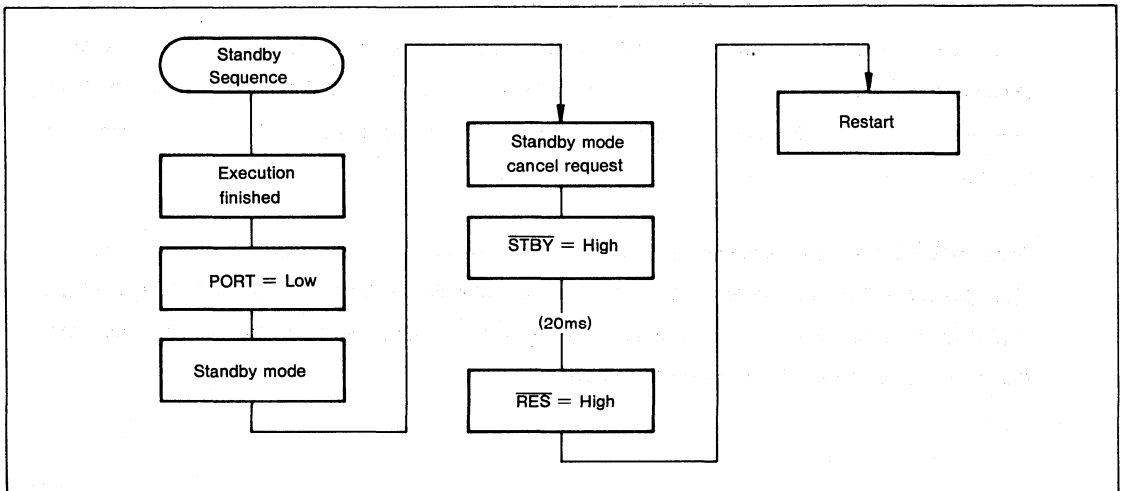


Figure 8-15. Standby Circuit Flowchart

## 8.6 Memory Ready Application

The memory ready function allows the MCU to access low-speed memories or low-speed devices.

Figure 8-16 shows a circuit example, and figure 8-17 is its timing chart.

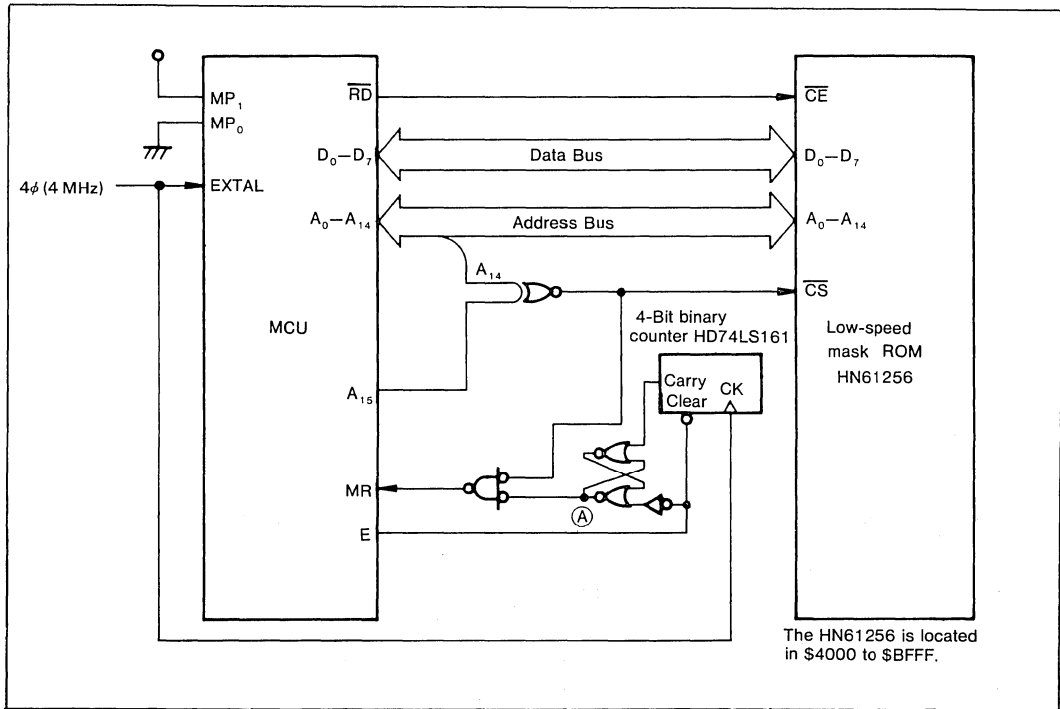


Figure 8-16. Low-Speed Memory Access Circuit

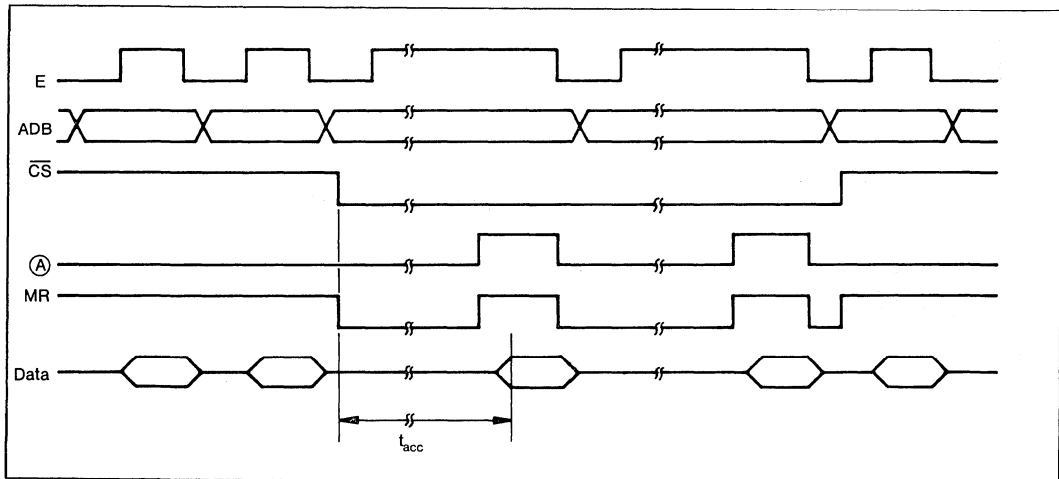


Figure 8-17. Memory Ready Bus Timing

## 8.7 Halt Application

The halt function enables the MCU in the expanded mode to interface with a DMAC (HD6844) and execute DMA (figure 8-18).

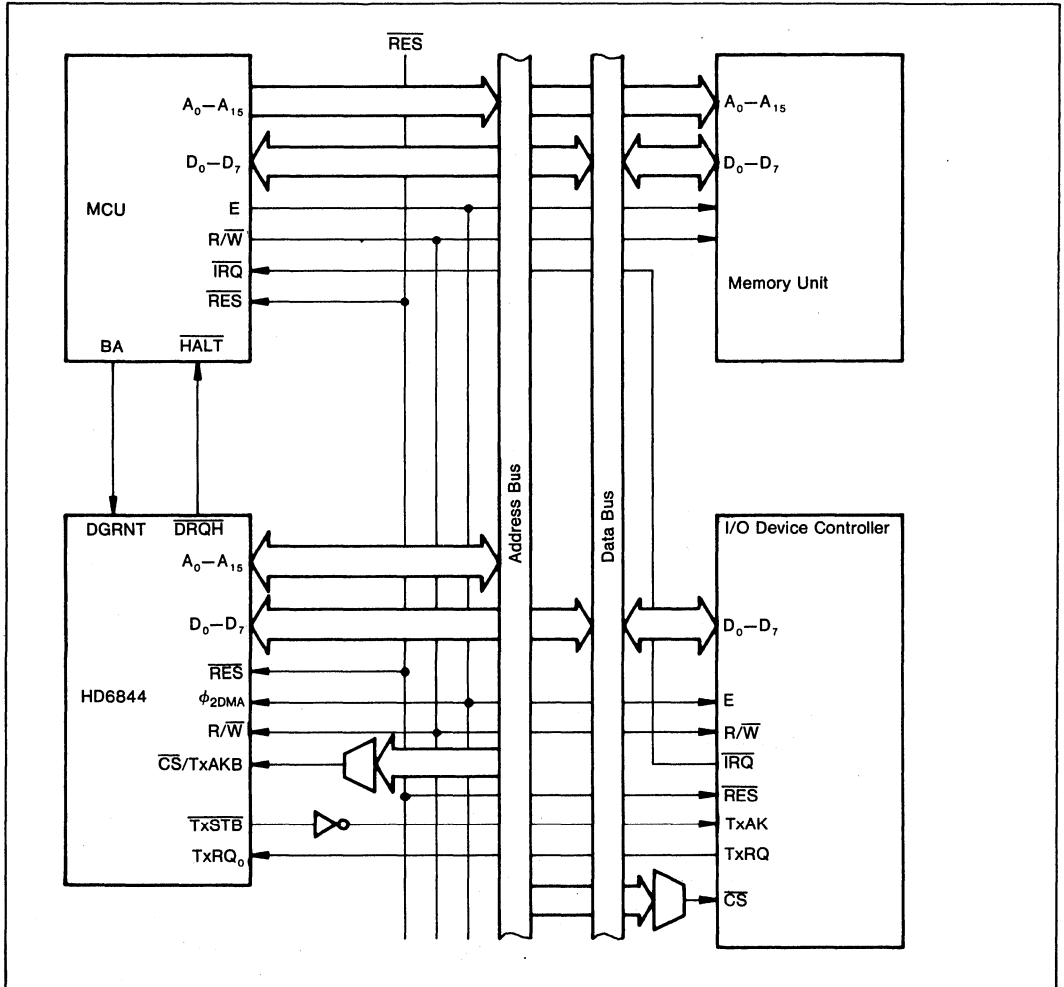


Figure 8-18. One-Channel DMAC Interface Example

## 8.8 $\overline{RD}$ , $\overline{WR}$ Application

$\overline{RD}$  and  $\overline{WR}$ , as well as E and  $R/\overline{W}$ , can act as external interface signals.  $\overline{RD}$  and  $\overline{WR}$  allow the MCU to easily interface with the 80xx family peripherals as well as with the 6800 series. Figure 8-19 shows an example of an interface between the MCU and an 8255.

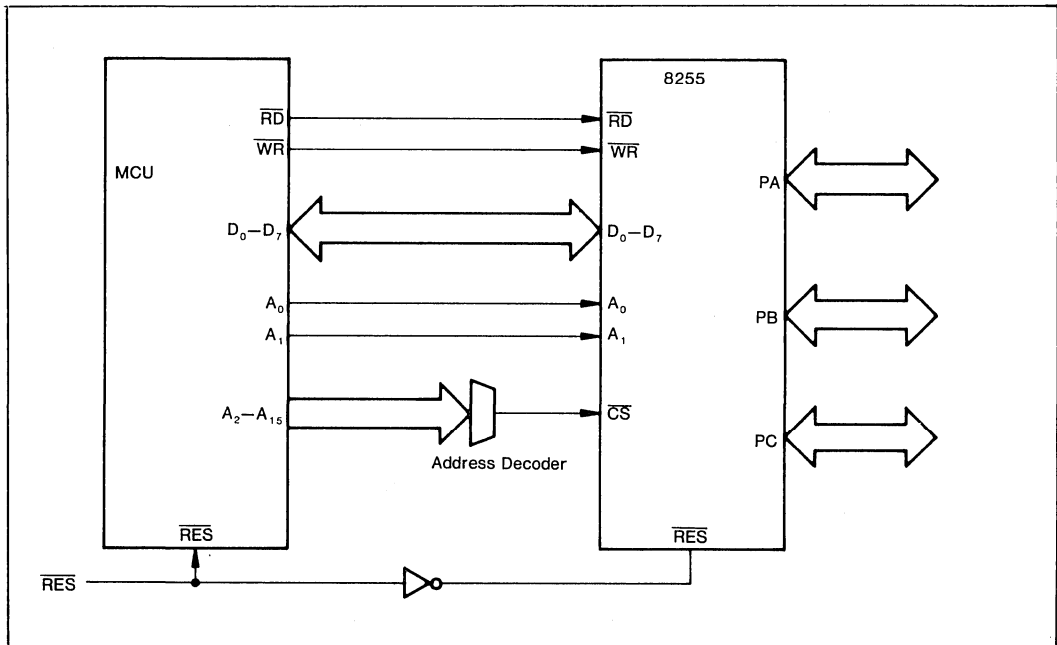


Figure 8-19. HD6301X0/HD63701X0 and 8255 Interface

## 8.9 LCD-II Interface Application

Figure 8-20 and 8-21 show examples of interfaces between an HD6301X0/HD63701X0 and a liquid crystal driver (LCD-II). The interface lines are TTL compatible. The HD6301X0/HD63701X0 in the expanded mode in figure 8-20 interfaces with the LCD-II directly through the external bus lines. Port 3 connects to the LCD-II data bus,  $R/\overline{W}$  connects to  $R/\overline{W}$ ,  $A_0$  connects to RS, and the rest of the address bus is decoded and ANDed with E to connect with E on the LCD-II.

The HD6301X0/HD63701X0 in the single-chip mode in figure 8-21 interfaces with the LCD-II through the I/O port. The read/write operation should be performed with care for the timing of the LCD-II E signal and others.

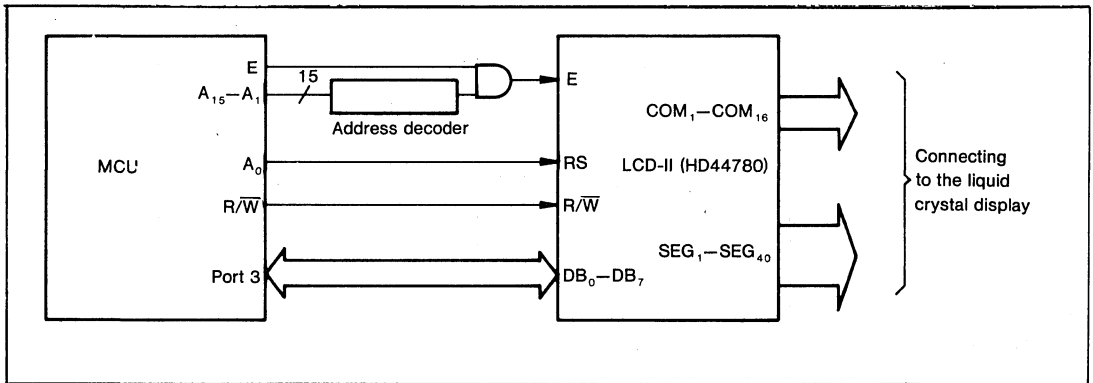


Figure 8-20. LCD-II Interface, Expanded Mode

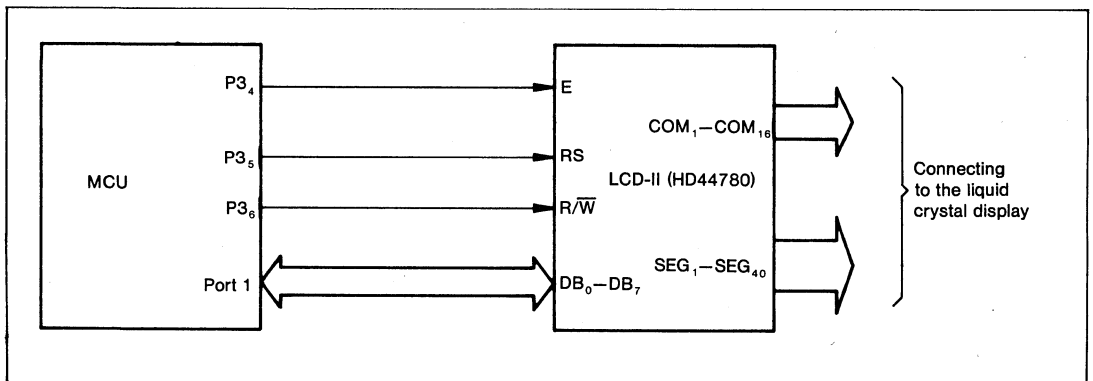


Figure 8-21. LCD-II Interface, Single-Chip Mode

## 8.10 Oscillation Circuit Board Design

Keep the following rules in mind when designing the circuit to connect the crystal resonator with the XTAL and EXTAL pins (figures 8-22, 8-23).

1. The crystal and load capacitors should be as close to the LSI as possible. External noise at the XTAL and EXTAL pins will disturb normal oscillation.
2. Keep the lines from XTAL and E as far apart as possible. Avoid parallel wiring. Interference from E to XTAL will disturb normal oscillation.
3. Do not allow signal or power lines to cross or run closely parallel to the oscillator lines (signals A, B, C in figure 8-22). They will disturb normal oscillation. Keep the resistance between XTAL and EXTAL pins and the next nearest pins greater than  $10\text{ M}\Omega$ .

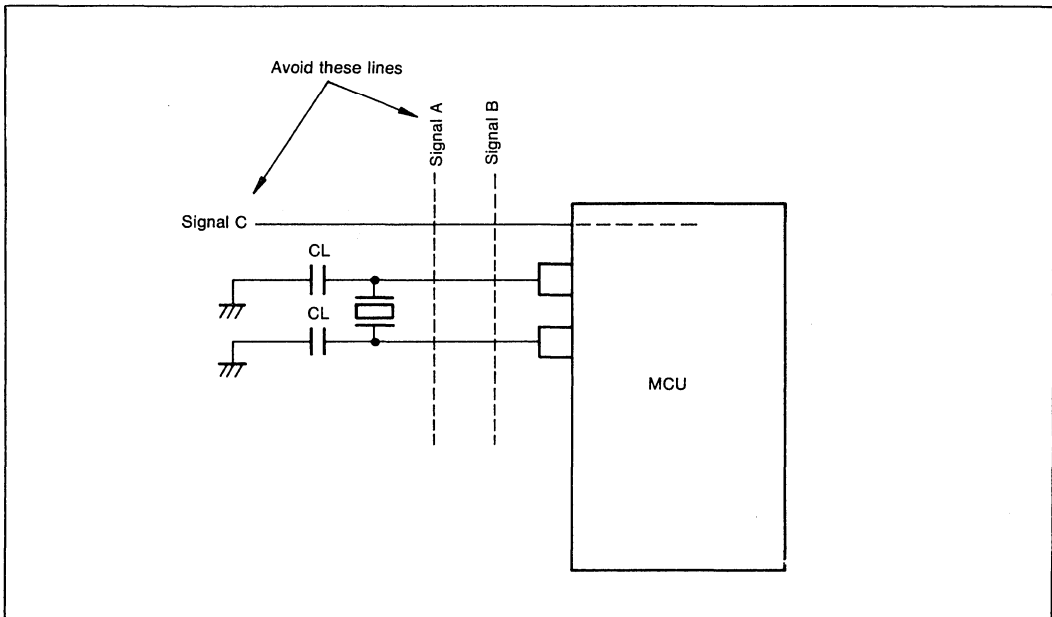


Figure 8-22. Oscillation Circuit Precautions



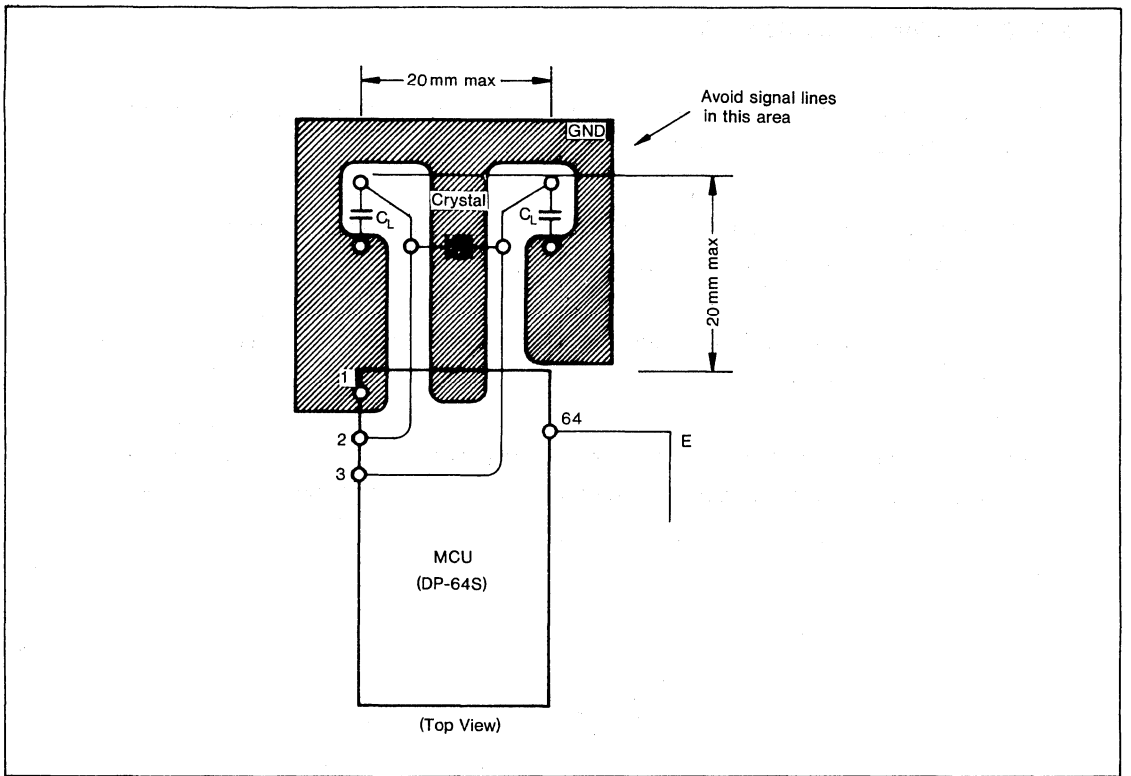


Figure 9-23. Oscillation Circuit Board Design Example

# Appendix I. Electrical Characteristics

## I.1 HD6301X0, HD63A01X0, HD63B01X0 Electrical Characteristics

### Absolute Maximum Ratings

Item	Symbol	Value	Unit
Supply voltage	$V_{CC}$	-0.3 to +7.0	V
Input voltage	$V_{in}$	-0.3 to $V_{CC}+0.3$	V
Operating temperature	$T_{opr}$	0 to +70	°C
Storage temperature	$T_{stg}$	-55 to +150	°C

Note: This product has protection circuits in input terminal from high static electricity voltage and high electric field.  
But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend  $V_{in}, V_{out}: V_{SS} \leq (V_{in} \text{ or } V_{out}) \leq V_{CC}$ .

### Electrical Characteristics

#### DC Characteristics

( $V_{CC}=5.0\text{ V} \pm 10\%$ ,  $f=0.1$  to 2.0 MHz,  $V_{SS}=0\text{ V}$ ,  $T_a=0$  to +70°C, unless otherwise noted.)

Item		Symbol	Min	Typ	Max	Unit	Test Condition
Input high voltage	RES, STBY,	$V_{IH}$	$V_{CC}-0.5$		$V_{CC}+0.3$	V	
	EXTAL		$V_{CC} \times 0.7$		$V_{CC}+0.3$	V	
	Other inputs		2.0		$V_{CC}+0.3$	V	
Input low voltage	All other inputs	$V_{IL}$	-0.3		0.8	V	
Input leakage current	RES, Port5 NMI, STBY, MP0, MP1	$ I_{in} $			1.0	$\mu\text{A}$	$V_{in}=0.5$ to $V_{CC}-0.5\text{ V}$
Three state leakage current	Ports 1, 2, 3, 4, 6, 7	$ I_{TSI} $			1.0	$\mu\text{A}$	$V_{in}=0.5$ to $V_{CC}-0.5\text{ V}$
Output high voltage		$V_{OH}$	2.4			V	$I_{OH}=-200\ \mu\text{A}$
			$V_{CC}-0.7$			V	$I_{OH}=-10\ \mu\text{A}$
Output low voltage		$V_{OL}$			0.4	V	$I_{OL}=1.6\text{ mA}$
Darlington drive current	Ports 2, 6	$-I_{OH}$	1.0		10.0	mA	$V_{out}=1.5\text{ V}$
Input capacitance	All other inputs	$C_{in}$			12.5	pF	$V_{in}=0\text{ V}$ , $f=1\text{ MHz}$ , $T_a=25^\circ\text{C}$
Standby current	Not operating	$I_{STB}$		3.0	15.0	$\mu\text{A}$	
Current dissipation <sup>1</sup>		$I_{SLP}$		1.5	3.0	mA	Sleeping ( $f=1\text{ MHz}^2$ )
				2.3	4.5	mA	Sleeping ( $f=1.5\text{ MHz}^2$ )
				3.0	6.0	mA	Sleeping ( $f=2\text{ MHz}^2$ )
				7.0	10.0	mA	Operating ( $f=1\text{ MHz}^2$ )
				10.5	15.0	mA	Operating ( $f=1.5\text{ MHz}^2$ )
				14.0	20.0	mA	Operating ( $f=2\text{ MHz}^2$ )
RAM standby voltage		$V_{RAM}$	2.0			V	

#### Notes:

- $V_{IH\ min}=V_{CC}-1.0\text{V}$ ,  $V_{IH\ max}=0.8\text{V}$  (All output terminals are at no load.)
- Current dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ. or max. values about current dissipations at x MHz operation are decided according to the following formula:  
 typ. value ( $f=x\text{ MHz}$ ) = typ. value ( $f=1\text{ MHz}$ )  $\times x$   
 max. value ( $f=x\text{ MHz}$ ) = max. value ( $f=1\text{ MHz}$ )  $\times x$   
 (both the sleeping and operating)

## AC Characteristics

( $V_{CC}=5.0\text{ V} \pm 10\%$ ,  $f=0.1$  to  $2.0\text{ MHz}$ ,  $V_{SS}=0\text{ V}$ ,  $T_a=0$  to  $+70\text{ }^\circ\text{C}$ , unless otherwise noted.)

### Bus Timing

Item	Symbol	HD6301X0			HD63A01X0			HD63B01X0			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Cycle time	$t_{cyc}$	1		10	0.666		10	0.5		10	$\mu\text{s}$	Fig. I-1
Enable rise time	$t_{Er}$			25			25			25	ns	
Enable fall time	$t_{Ef}$			25			25			25	ns	
Enable pulse width high level <sup>1</sup>	$PW_{EH}$	450			300			220			ns	
Enable pulse width low level <sup>1</sup>	$PW_{EL}$	450			300			220			ns	
Address, R/ $\bar{W}$ delay time <sup>1</sup>	$t_{AD}$			250			190			160	ns	
Data delay time (Write)	$t_{DDW}$			200			160			120	ns	
Data set-up time (Read)	$t_{DSR}$	80			70			70			ns	
Address, R/ $\bar{W}$ hold time <sup>1</sup>	$t_{AH}$	80			50			35			ns	
Data hold time (Write) <sup>1</sup>	$t_{HW}$	80			50			40			ns	
Data hold time (Read)	$t_{HR}$	0			0			0			ns	
$\overline{RD}$ , $\overline{WR}$ pulse width <sup>1</sup>	$PWRW$	450			300			220			ns	
$\overline{RD}$ , $\overline{WR}$ delay time	$t_{RWD}$			40			40			40	ns	
$\overline{RD}$ , $\overline{WR}$ hold time	$t_{HRW}$			30			30			25	ns	
$\overline{LIR}$ delay time	$t_{DLR}$			200			160			120	ns	
$\overline{LIR}$ hold time	$t_{HLR}$	10			10			10			ns	
MR set-up time <sup>1</sup>	$t_{SMR}$	400			280			230			ns	Fig. I-2
MR hold time <sup>1</sup>	$t_{HMR}$			90			40			0	ns	
E clock pulse width at MR	$PW_{EMR}$			9			9			9	$\mu\text{s}$	
Processor control set-up time	$t_{PCS}$	200			200			200			ns	Figs. I-3, I-11, I-12
Processor control rise time	$t_{PCr}$			100			100			100	ns	Figs. I-2, I-3
Processor control fall time	$t_{PCf}$			100			100			100	ns	
BA delay time	$t_{BA}$			250			190			160	ns	Fig. I-3
Oscillator stabilization time	$t_{RC}$	20			20			20			ms	Fig. I-12
Reset pulse width	$PW_{RST}$	3			3			3			$t_{cyc}$	

Note: 1. These timings change in approximate proportion to  $t_{cyc}$ . The figures in this characteristics represent those when  $t_{cyc}$  is minimum (=in the highest speed operation).

## Peripheral Port Timing

Item	Symbol	HD6301X0			HD63A01X0			HD63B01X0			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Peripheral data set-up time	(Ports 2, 3, 5, 6) tPDSU	200			200			200			ns	Fig. I-5
Peripheral data hold time	(Ports 2, 3, 5, 6) tPDH	200			200			200			ns	
Delay time (From enable fall edge to peripheral output)	(Ports 1, 2, 3, 4, 6, 7) tPWD			300			300			300	ns	Fig. I-6

## Timer, SCI Timing

Item	Symbol	HD6301X0			HD63A01X0			HD63B01X0			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Timer 1 input pulse width	tPWT	2.0			2.0			2.0			t <sub>cyc</sub>	Fig. I-9
Delay time (enable positive transition to timer output)	tTOD			400			400			400	ns	Figs. I-7, I-8
SCI input clock cycle	(Async. mode) t <sub>Scyc</sub>	1.0			1.0			1.0			t <sub>cyc</sub>	Fig. I-9
	(Clock sync.)	2.0			2.0			2.0			t <sub>cyc</sub>	Fig. I-4
SCI transmit data delay time (Clock sync. mode)	tTXD			200			200			200	ns	Fig. I-4
SCI receive data set-up time (Clock sync. mode)	tSRX	290			290			290			ns	
SCI receive data hold time (Clock sync. mode)	tHRX	100			100			100			ns	
SCI input clock pulse width	tPWCK	0.4	0.6		0.4	0.6		0.4	0.6		t <sub>Scyc</sub>	Fig. I-9
Timer 2 input clock cycle	t <sub>cyc</sub>	2.0			2.0			2.0			t <sub>cyc</sub>	
Timer 2 input clock pulse width	tPWCK	200			200			200			ns	
Timer 1 - 2, SCI input clock rise time	tCKr			100			100			100	ns	
Timer 1 - 2, SCI input clock fall time	tCKf			100			100			100	ns	

5

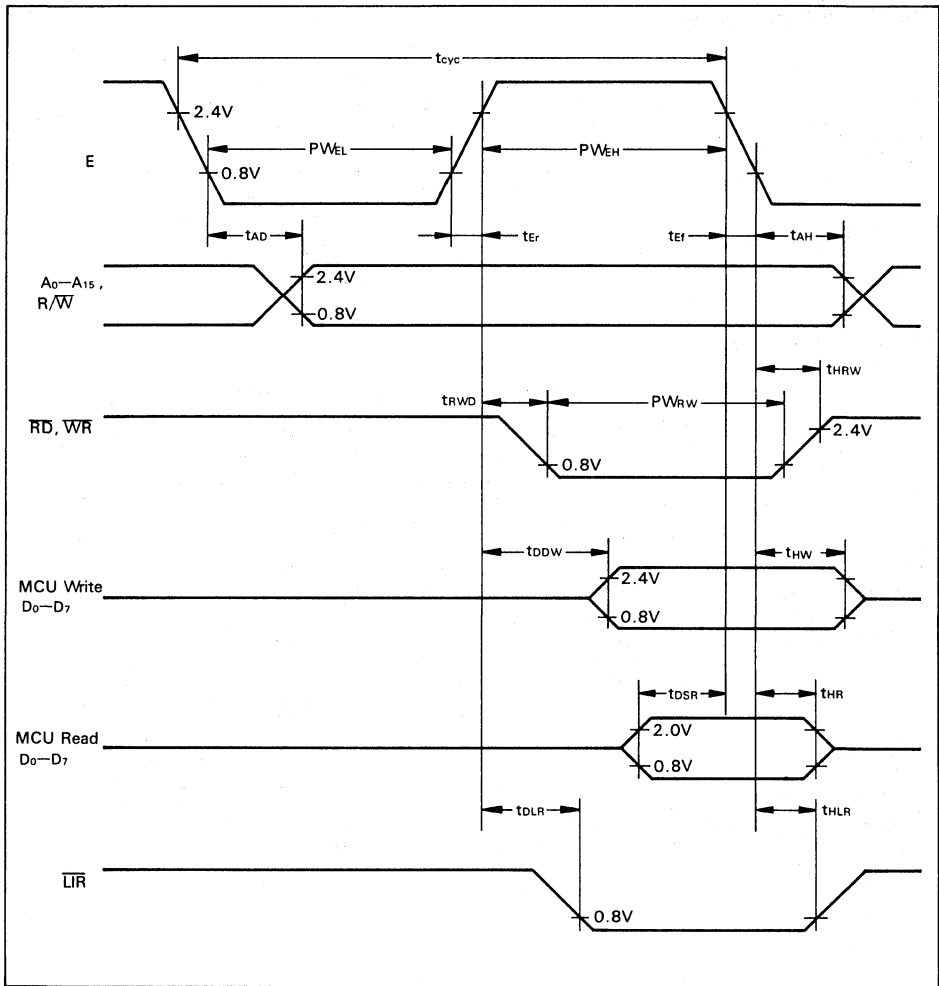


Figure I-1. Mode 1, Mode 2 Bus Timing

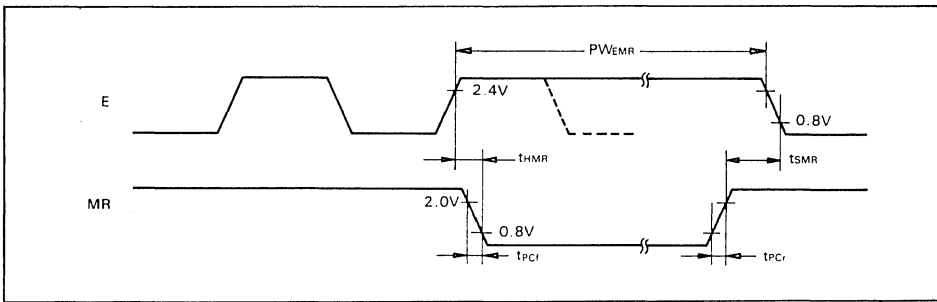


Figure I-2. Memory Ready and E Clock Timing

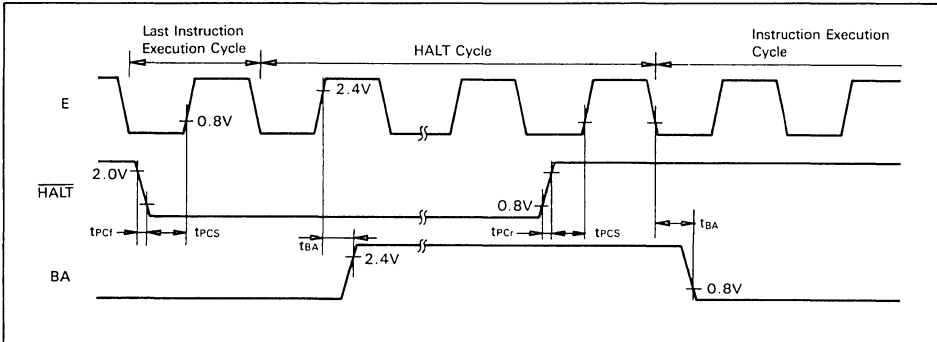


Figure I-3.  $\overline{\text{HALT}}$  and BA Timing

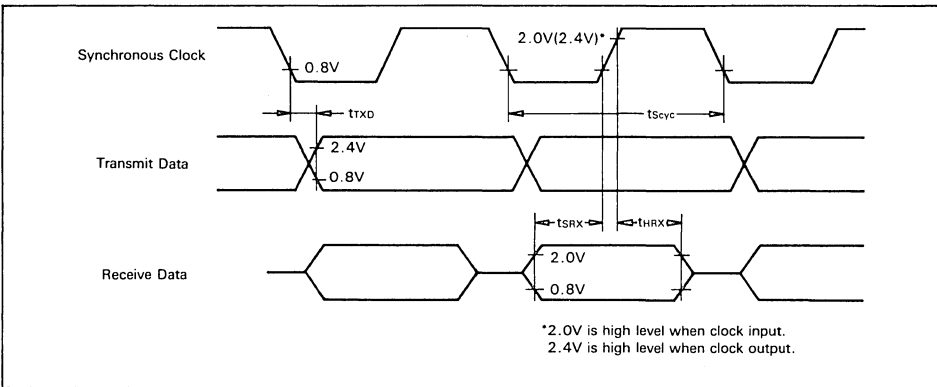


Figure I-4. SCI Clocked Synchronous Timing

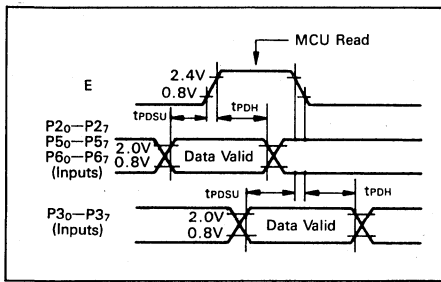


Figure I-5. Port Data Set-up and Hold Times (MCU Read)

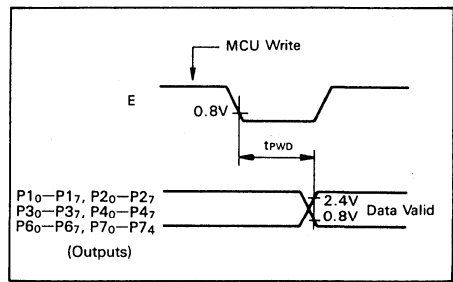


Figure I-6. Port Data Delay Times (MCU Write)

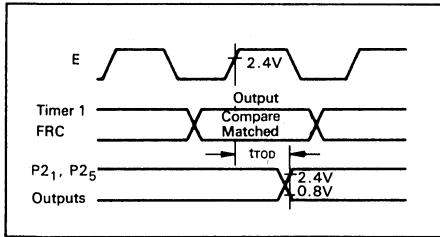


Figure I-7. Timer 1 Output Timing

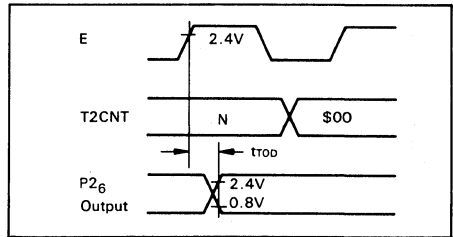


Figure I-8. Timer 2 Output Timing

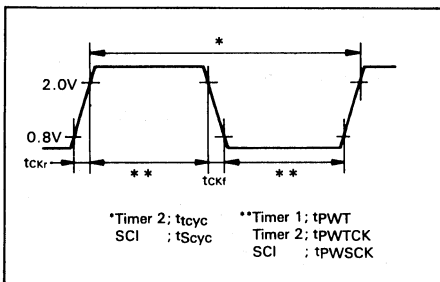


Figure I-9. Timer 1-2, SCI Input Clock Timing

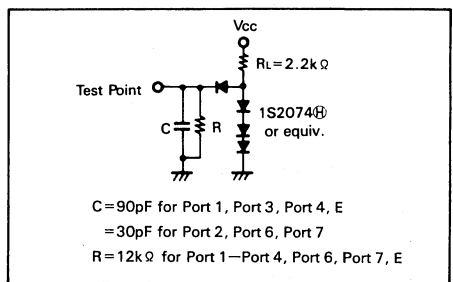


Figure I-10. Bus Timing Test Loads (TTL Load)

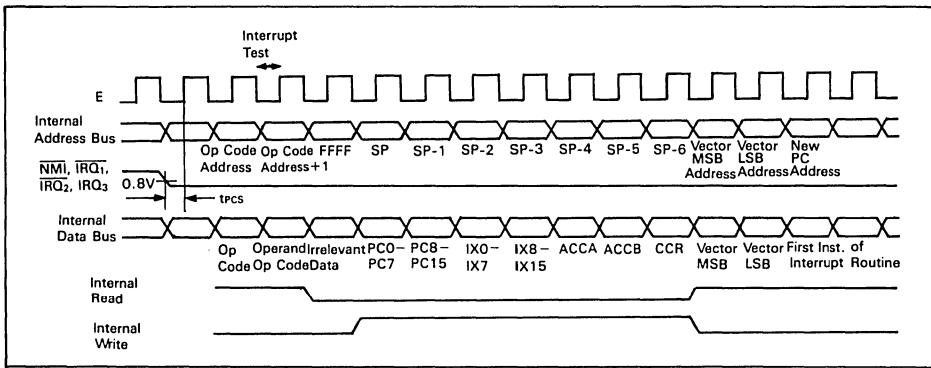


Figure I-11. Interrupt Sequence

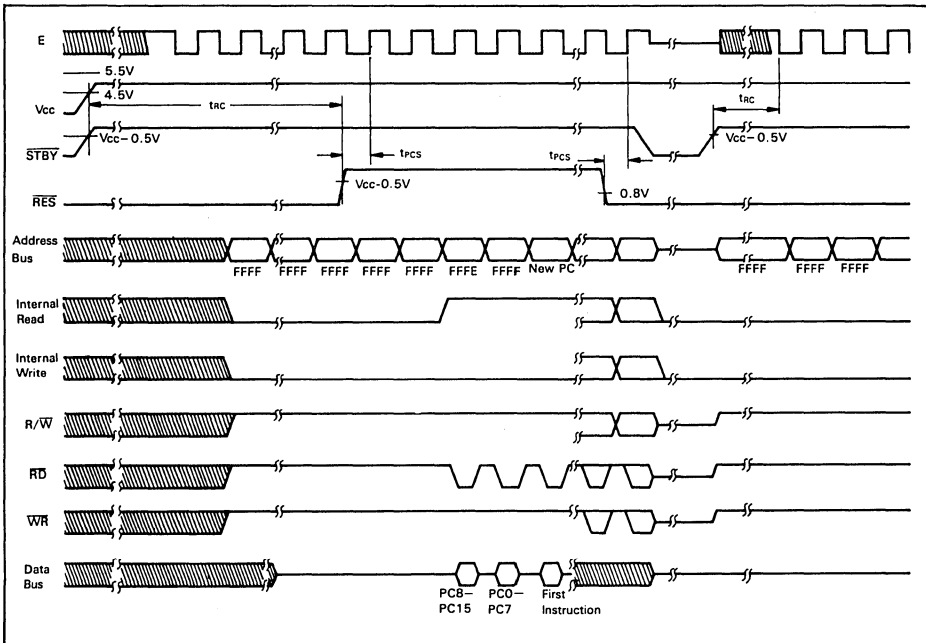


Figure I-12. Reset Timing



## I.2 HD6303X, HD63A03X, HD63B03X Electrical Characteristics

### Absolute Maximum Ratings

Item	Symbol	Value	Unit
Supply voltage	$V_{CC}$	-0.3 to +7.0	V
Input voltage	$V_{in}$	-0.3 to $V_{CC}+0.3$	V
Operating temperature	$T_{opr}$	0 to +70	°C
Storage temperature	$T_{stg}$	-55 to +150	°C

Note: This product has protection circuits in input terminal from high static electricity voltage and high electric field. But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend  $V_{in}$ ,  $V_{out}$ :  $V_{SS} \leq (V_{in} \text{ or } V_{out}) \leq V_{CC}$ .

### Electrical Characteristics

#### DC Characteristics

( $V_{CC}=5.0\text{ V} \pm 10\%$ ,  $f=0.1$  to 2.0 MHz,  $V_{SS}=0\text{ V}$ ,  $T_a=0$  to +70°C, unless otherwise noted.)

Item	Symbol	Min	Typ	Max	Unit	Test Condition
Input high voltage	RES, STBY	$V_{IH}$	$V_{CC}-0.5$	$V_{CC}+0.3$	V	
	EXTAL		$V_{CC} \times 0.7$	$V_{CC}+0.3$	V	
	Other inputs		2.0	$V_{CC}+0.3$	V	
Input low voltage	All other inputs	$V_{IL}$	-0.3	0.8	V	
Input leakage current	RES, Port5	$ I_{in} $		1.0	$\mu\text{A}$	$V_{in}=0.5$ to $V_{CC}-0.5\text{ V}$
	NMI, STBY, MP0, MP1					
Three state leakage current	A <sub>0</sub> -A <sub>15</sub> , D <sub>0</sub> -D <sub>7</sub> , RD, WR, R/W, Ports 2,6	$ I_{TSI} $		1.0	$\mu\text{A}$	$V_{in}=0.5$ to $V_{CC}-0.5\text{ V}$
Output high voltage	$V_{OH}$	2.4			V	$I_{OH}=-200\ \mu\text{A}$
		$V_{CC}-0.7$			V	$I_{OH}=-10\ \mu\text{A}$
Output low voltage	$V_{OL}$		0.4		V	$I_{OL}=1.6\text{ mA}$
Darlington drive current	Ports 2, 6	$-I_{OH}$	1.0	10.0	mA	$V_{out}=1.5\text{ V}$
Input capacitance	All other inputs	$C_{in}$		12.5	pF	$V_{in}=0\text{V}$ , $f=1\text{ MHz}$ $T_a=25^\circ\text{C}$
Standby current	Not operating	$I_{STB}$	3.0	15.0	$\mu\text{A}$	
Current dissipation <sup>1</sup>	$I_{SLP}$		1.5	3.0	mA	Sleeping ( $f=1\text{ MHz}^2$ )
			2.3	4.5	mA	Sleeping ( $f=1.5\text{ MHz}^2$ )
			3.0	6.0	mA	Sleeping ( $f=2\text{ MHz}^2$ )
			7.0	10.0	mA	Operating ( $f=1\text{ MHz}^2$ )
			10.5	15.0	mA	Operating ( $f=1.5\text{ MHz}^2$ )
			14.0	20.0	mA	Operating ( $f=2\text{ MHz}^2$ )
RAM standby voltage	$V_{RAM}$	2.0			V	

#### Notes:

- $V_{in\ min}=V_{CC}-1.0\text{V}$ ,  $V_{in\ max}=0.8\text{V}$  (All output terminals are at no load.)
- Current dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ. or max. values about current dissipations at x MHz operation are decided according to the following formula:  
 typ. value ( $f=x\text{ MHz}$ ) = typ. value ( $f=1\text{ MHz}$ )  $\times x$   
 max. value ( $f=x\text{ MHz}$ ) = max. value ( $f=1\text{ MHz}$ )  $\times x$   
 (both the sleeping and operating)

## AC Characteristics

( $V_{CC}=5.0\text{ V} \pm 10\%$ ,  $f=0.1$  to  $2.0\text{ MHz}$ ,  $V_{SS}=0\text{ V}$ ,  $T_a=0$  to  $+70\text{ }^\circ\text{C}$ , unless otherwise noted.)

### Bus Timing

Item	Symbol	HD6303X			HD63A03X			HD63B03X			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Cycle time	$t_{cyc}$	1		10	0.666		10	0.5		10	$\mu\text{s}$	Fig. I-13
Enable rise time	$t_{Er}$			25			25			25	ns	
Enable fall time	$t_{Ef}$			25			25			25	ns	
Enable pulse width high level <sup>1</sup>	$PW_{EH}$	450			300			220			ns	
Enable pulse width low level <sup>1</sup>	$PW_{EL}$	450			300			220			ns	
Address, $R/\overline{W}$ delay time <sup>1</sup>	$t_{AD}$			250			190			160	ns	
Data delay time (Write)	$t_{DDW}$			200			160			120	ns	
Data set-up time (Read)	$t_{DSR}$	70			70			70			ns	
Address, $R/\overline{W}$ hold time <sup>1</sup>	$t_{AH}$	80			50			35			ns	
Data hold time (Write) <sup>1</sup>	$t_{HW}$	80			50			40			ns	
Data hold time (Read)	$t_{HR}$	0			0			0			ns	
$\overline{RD}$ , $\overline{WR}$ pulse width <sup>1</sup>	$PW_{RW}$	450			300			220			ns	
$\overline{RD}$ , $\overline{WR}$ delay time	$t_{rWD}$			40			40			40	ns	
$\overline{RD}$ , $\overline{WR}$ hold time	$t_{HRW}$			30			30			25	ns	
$\overline{LIR}$ delay time	$t_{DLR}$			200			160			120	ns	
$\overline{LIR}$ hold time	$t_{HLR}$	10			10			10			ns	
MR set-up time <sup>1</sup>	$t_{SMR}$	400			280			230			ns	Fig. I-14
MR hold time <sup>1</sup>	$t_{HMR}$			90			40			0	ns	
E clock pulse width at MR	$PW_{EMR}$			9			9			9	$\mu\text{s}$	
Processor control set-up time	$t_{PCS}$	200			200			200			ns	Figs. I-15, I-23, I-24
Processor control rise time	$t_{PCr}$			100			100			100	ns	Figs. I-14, I-15
Processor control fall time	$t_{PCf}$			100			100			100	ns	
BA delay time	$t_{BA}$			250			190			160	ns	Fig. I-15
Oscillator stabilization time	$t_{RC}$	20			20			20			ms	Fig. I-24
Reset pulse width	$PW_{RST}$	3			3			3			$t_{cyc}$	

Note: 1. These timings change in approximate proportion to  $t_{cyc}$ . The figures in this characteristics represent those when  $t_{cyc}$  is minimum (= in the highest speed operation).

## Peripheral Port Timing

Item	Symbol	HD6303X			HD63A03X			HD63B03X			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Peripheral data set-up time (Ports 2, 5, 6)	tPDSU	200			200			200			ns	Fig. I-17
Peripheral data hold time (Ports 2, 5, 6)	tPDH	200			200			200			ns	
Delay time (From enable fall edge to peripheral output) (Ports 2, 6)	tPWD			300				300			300	ns Fig. I-18

## Timer, SCI Timing

Item	Symbol	HD6303X			HD63A03X			HD63B03X			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Timer 1 input pulse width	tPWT	2.0			2.0			2.0			t <sub>cyc</sub>	Fig. I-21
Delay time (enable positive transition to timer output)	tTOD			400			400			400	ns	Figs. I-19, I-20
SCI input clock cycle	t <sub>scyc</sub>	(Async. mode)			1.0			1.0			t <sub>cyc</sub>	Fig. I-21
		(Clock sync.)	2.0			2.0			2.0		t <sub>cyc</sub>	Fig. I-16
SCI transmit data delay time (Clock sync. mode)	tTXD			200			200			200	ns	Fig. I-16
SCI receive data set-up time (Clock sync. mode)	tSRX	290			290			290			ns	
SCI receive data hold time (Clock sync. mode)	tHRX	100			100			100			ns	
SCI input clock pulse width	tPW <sub>SCK</sub>	0.4		0.6	0.4		0.6	0.4		0.6	t <sub>scyc</sub>	Fig. I-21
Timer 2 input clock cycle	t <sub>cyc</sub>	2.0			2.0			2.0			t <sub>cyc</sub>	
Timer 2 input clock pulse width	tPW <sub>TCK</sub>	200			200			200			ns	
Timer 1 - 2, SCI input clock rise time	tCKr			100			100			100	ns	
Timer 1 - 2, SCI input clock fall time	tCKf			100			100			100	ns	

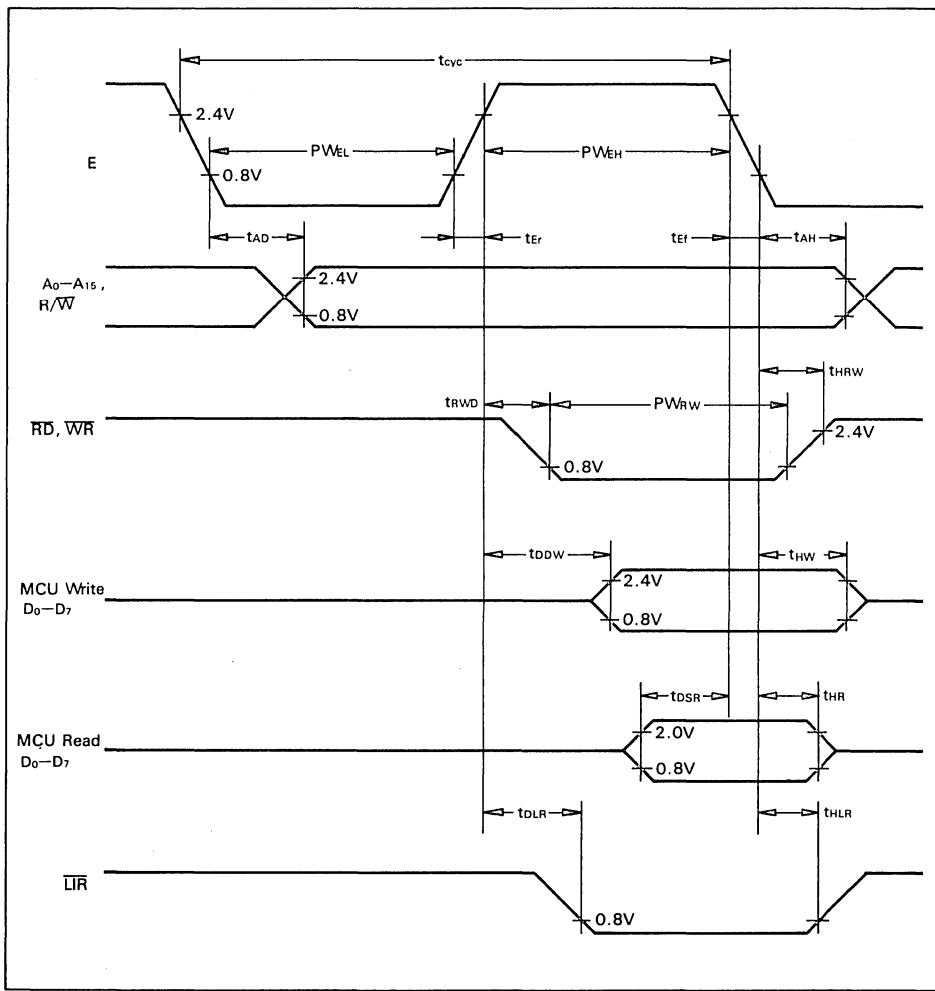


Figure I-13. Bus Timing

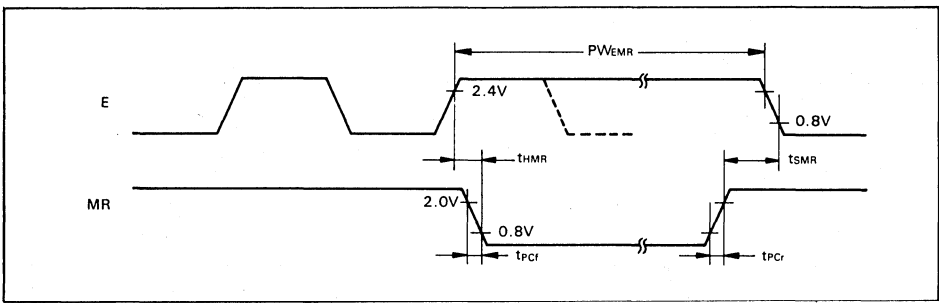


Figure I-14. Memory Ready and E Clock Timing

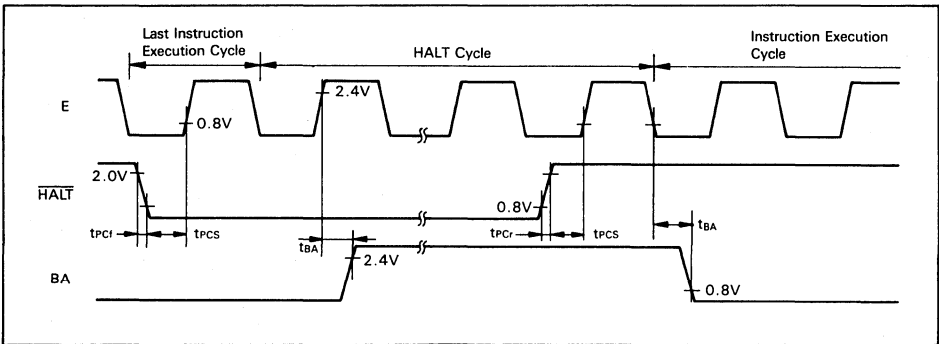


Figure I-15.  $\overline{HALT}$  and BA Timing

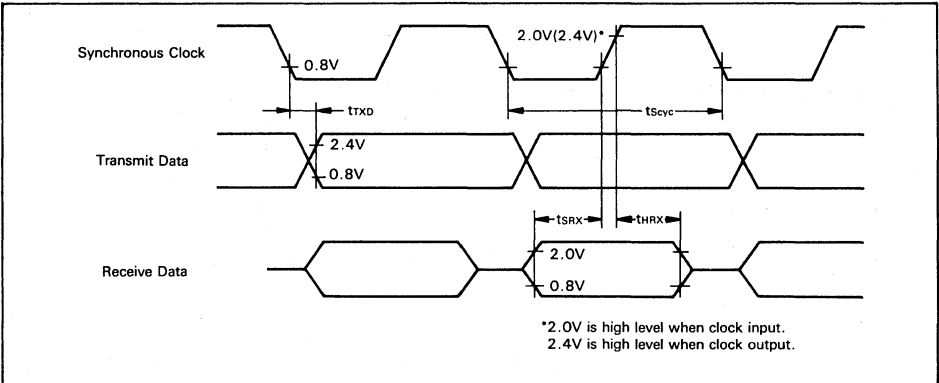


Figure I-16. SCI Clocked Synchronous Timing

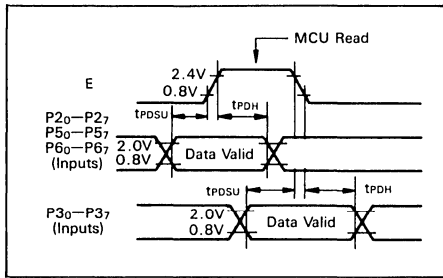


Figure I-17. Port Data Set-up and Hold Times (MCU Read)

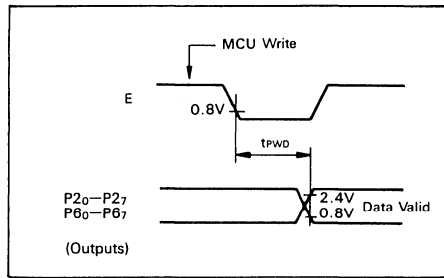


Figure I-18. Port Data Delay Times (MCU Write)

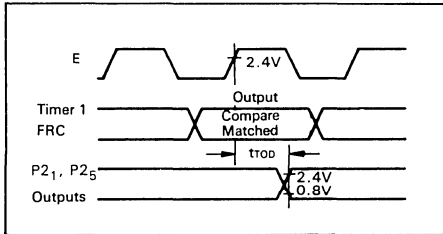


Figure I-19. Timer 1 Output Timing

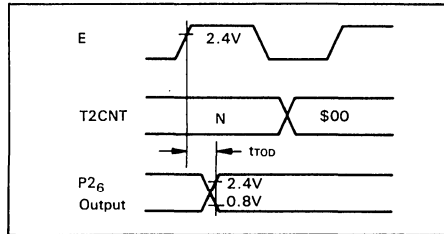


Figure I-20. Timer 2 Output Timing

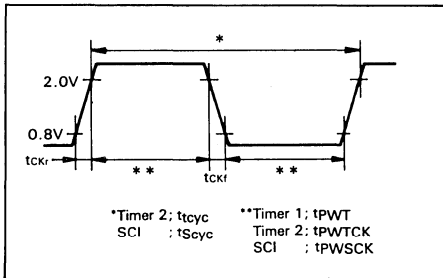


Figure I-21. Timer 1·2, SCI Input Clock Timing

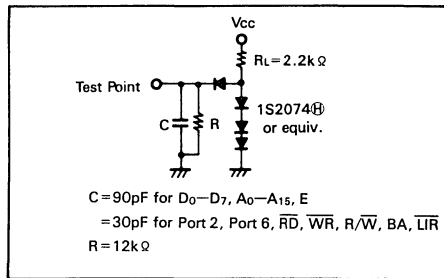


Figure I-22. Bus Timing Test Loads (TTL Load)

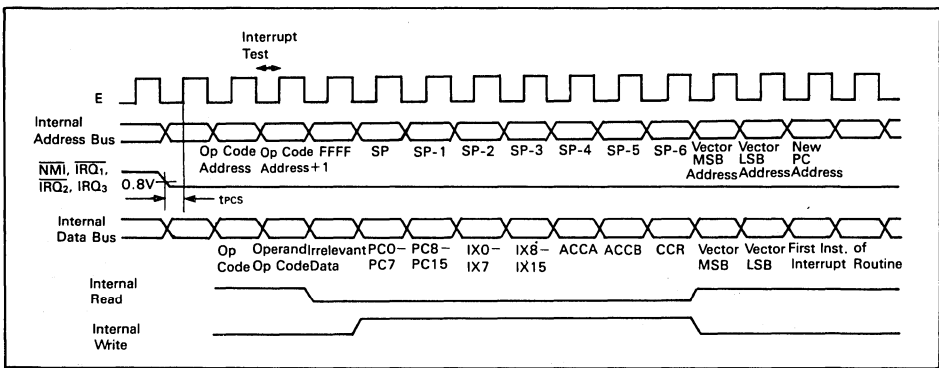


Figure I-23. Interrupt Sequence

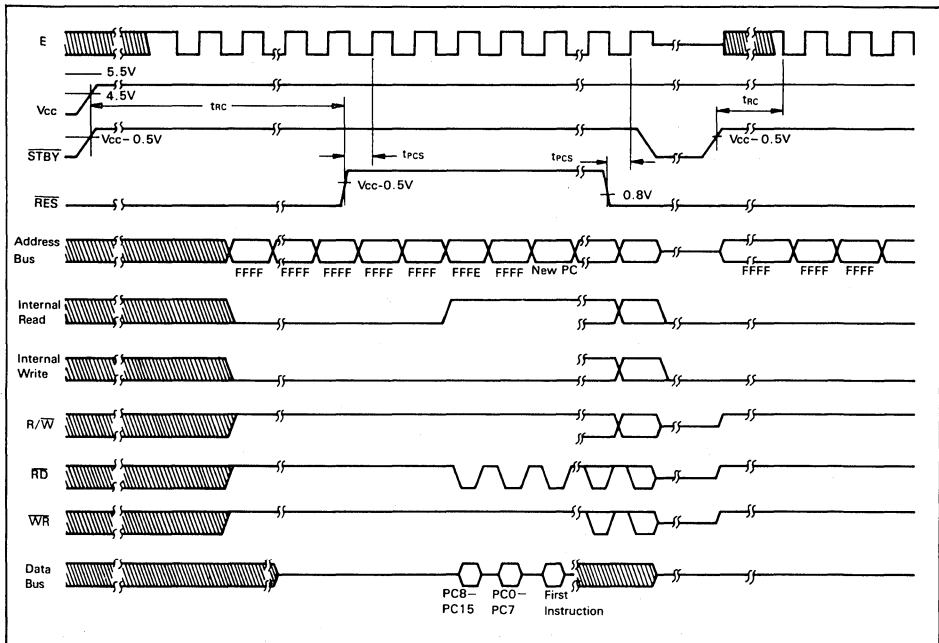


Figure I-24. Reset Timing

# I.3 HD63701X0, HD637A01X0, HD637B01X0 Electrical Characteristics

## Absolute Maximum Ratings

Item	Symbol	Value	Unit
Supply voltage	$V_{CC}$	-0.3 to +7.0	V
$V_{PP}$ voltage	$V_{PP}$	-0.3 to +22	V
Input voltage	$V_{in}$	-0.3 to $V_{CC}+0.3$	V
Operating temperature	$T_{opr}$	0 to +70	°C
Storage temperature	$T_{stg}$	-55 to +125	°C

Note: This product has protection circuits in input terminal from high static electricity voltage and high electric field. But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend  $V_{in}$ ,  $V_{out}$ :  $V_{SS} \leq (V_{in} \text{ or } V_{out}) \leq V_{CC}$ .

## Electrical Characteristics

### DC Characteristics

( $V_{CC}=5.0\text{ V} \pm 10\%$ ,  $f=0.1$  to 2.0 MHz,  $V_{SS}=0\text{ V}$ ,  $T_a=-20$  to  $+70^\circ\text{C}$ , unless otherwise noted.)

Item		Symbol	Min	Typ	Max	Unit	Test Condition
Input high voltage	RES, STBY, MP0, MP1	$V_{IH}$	$V_{CC}-0.5$		$V_{CC}+0.3$	V	
	EXTAL		$V_{CC} \times 0.7$		$V_{CC}+0.3$	V	
	P22(SCLK) <sup>3</sup>		2.4		$V_{CC}+0.3$	V	
	Other inputs		2.0		$V_{CC}+0.3$	V	
Input low voltage	All other inputs	$V_{IL}$	-0.3		0.8	V	
Input leakage current	RES, Port 5 NMI, STBY, MP0, MP1	$ I_{in} $			1.0	$\mu\text{A}$	$V_{in}=0.5$ to $V_{CC}-0.5\text{ V}$
Three state leakage current	Ports 1, 2, 3, 4, 6, 7	$ I_{TSI} $			1.0	$\mu\text{A}$	$V_{in}=0.5$ to $V_{CC}-0.5\text{ V}$
Output high voltage		$V_{OH}$	2.4			V	$I_{OH}=-200\ \mu\text{A}$
			$V_{CC}-0.7$			V	$I_{OH}=-10\ \mu\text{A}$
Output low voltage	Ports 2, 6	$V_{OL}$			0.5	V	$I_{OL}=1.6\text{ mA}$
	Other outputs				0.4	V	
Darlington drive current	Ports 2, 6	$-I_{OH}$	1.0		10.0	mA	$V_{out}=1.5\text{ V}$
Input capacitance	All inputs (except $V_{PP}/\overline{OE}$ )	$C_{in}$			6.5	pF	$V_{in}=0\text{ V}$ , $f=1\text{ MHz}$ , $T_a=25^\circ\text{C}$
	$V_{PP}/\overline{OE}$				12.5	pF	
Standby current	Not operating	$I_{STB}$		3.0	15.0	$\mu\text{A}$	
Current dissipation <sup>1</sup>		$I_{SLP}$		1.5	3.0	mA	Sleeping ( $f=1\text{ MHz}^2$ )
				2.3	4.5	mA	Sleeping ( $f=1.5\text{ MHz}^2$ )
				3.0	6.0	mA	Sleeping ( $f=2\text{ MHz}^2$ )
		$I_{CC}$		7.0	10.0	mA	Operating ( $f=1\text{ MHz}^2$ )
				10.5	15.0	mA	Operating ( $f=1.5\text{ MHz}^2$ )
				14.0	20.0	mA	Operating ( $f=2\text{ MHz}^2$ )
RAM standby voltage		$V_{RAM}$	2.0			V	

Notes :

- $V_{IH}$  min= $V_{CC}-1.0\text{V}$ ,  $V_{IH}$  max= $0.8\text{V}$  (All output terminals are at no load.)
- Current dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ. or max. values about current dissipations at x MHz operation are decided according to the following formula :  
 typ. value ( $f=x\text{ MHz}$ ) = typ. value ( $f=1\text{ MHz}$ )  $\times$  x  
 max. value ( $f=x\text{ MHz}$ ) = max. value ( $f=1\text{ MHz}$ )  $\times$  x  
 (both the sleeping and operating)
- Only serial clock use.



## AC Characteristics

( $V_{CC}=5.0\text{ V} \pm 10\%$ ,  $f=0.1$  to  $2.0\text{ MHz}$ ,  $V_{SS}=0\text{ V}$ ,  $T_a=-20$  to  $+70\text{ }^\circ\text{C}$ , unless otherwise noted.)

### Bus Timing

Item	Symbol	HD63701Y0			HD637A01Y0			HD637B01Y0			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Cycle time	$t_{cyc}$	1		10	0.666		10	0.5		10	$\mu\text{s}$	Fig. I-25
Enable rise time	$t_{Er}$			25			25			25	ns	
Enable fall time	$t_{Ef}$			25			25			25	ns	
Enable pulse width high level <sup>1</sup>	$PW_{EH}$	450			300			220			ns	
Enable pulse width low level <sup>1</sup>	$PW_{EL}$	450			300			220			ns	
Address, $R/\bar{W}$ delay time <sup>1</sup>	$t_{AD}$			250			190			160	ns	
Data delay time (Write)	$t_{DDW}$			200			160			120	ns	
Data set-up time (Read)	$t_{DSR}$	80			70			70			ns	
Address, $R/\bar{W}$ hold time <sup>1</sup>	$t_{AH}$	70			45			30			ns	
Data hold time (Write) <sup>1</sup>	$t_{HW}$	70			50			35			ns	
Data hold time (Read)	$t_{HR}$	0			0			0			ns	
$\bar{RD}$ , $\bar{WR}$ pulse width <sup>1</sup>	$PW_{RW}$	450			300			220			ns	
$\bar{RD}$ , $\bar{WR}$ delay time	$t_{RWD}$			40			40			40	ns	
$\bar{RD}$ , $\bar{WR}$ hold time	$t_{HRW}$			30			30			25	ns	
$\bar{LIR}$ delay time	$t_{DLR}$			200			160			120	ns	
$\bar{LIR}$ hold time	$t_{HLR}$	30			30			25			ns	
MR set-up time <sup>1</sup>	$t_{SMR}$	400			280			230			ns	Fig. I-26
MR hold time <sup>1</sup>	$t_{HMR}$			90			40			0	ns	
E clock pulse width at MR	$PW_{EMR}$			9			9			9	$\mu\text{s}$	
Processor control set-up time	$t_{PCS}$	200			200			200			ns	Figs. I-27, I-35, I-36
Processor control rise time	$t_{PCr}$			100			100			100	ns	Figs. I-26, I-27
Processor control fall time	$t_{PCf}$			100			100			100	ns	
BA delay time	$t_{BA}$			250			190			160	ns	Fig. I-27
Oscillator stabilization time	$t_{RC}$	20			20			20			ms	Fig. I-36
Reset pulse width	$PW_{RST}$	3			3			3			$t_{cyc}$	

Note: 1. These timings change in approximate proportion to  $t_{cyc}$ . The figures in this characteristics represent those when  $t_{cyc}$  is minimum (=in the highest speed operation).

## Peripheral Port Timing

Item	Symbol	HD63701X0			HD637A01X0			HD637B01X0			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Peripheral data set-up time	(Ports 2, 3, 5, 6) tPDSU	200			200			200			ns	Fig. I-29
Peripheral data hold time	(Ports 2, 3, 5, 6) tPDH	200			200			200			ns	
Delay time (From enable fall edge to peripheral output)	(Ports 1, 2, 3, 4, 6, 7) tPWD			300				300			300	ns Fig. I-30

## Timer, SCI Timing

Item	Symbol	HD63701X0			HD637A01X0			HD637B01X0			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Timer 1 input pulse width	tPWT	2.0			2.0			2.0			t <sub>cyc</sub>	Fig. I-33
Delay time (enable positive transition to timer output)	tTOD			400			400			400	ns	Figs. I-31 I-32
SCI input clock cycle	(Async. mode) tScyc	1.0			1.0			1.0			t <sub>cyc</sub>	Fig. I-33
	(Clock sync.)	2.0			2.0			2.0			t <sub>cyc</sub>	Fig. I-28
SCI transmit data delay time (Clock sync. mode)	tTXD			200			200			200	ns	Fig. I-28
SCI receive data set-up time (Clock sync. mode)	tSRX	290			290			290			ns	
SCI receive data hold time (Clock sync. mode)	tHRX	100			100			100			ns	
SCI input clock pulse width	tPWSCK	0.4		0.6	0.4		0.6	0.4		0.6	t <sub>Scyc</sub>	Fig. I-33
Timer 2 input clock cycle	tCyc	2.0			2.0			2.0			t <sub>cyc</sub>	
Timer 2 input clock pulse width	tPWTCX	200			200			200			ns	
Timer 1 + 2, SCI input clock rise time	tCKr			100			100			100	ns	
Timer 1 + 2, SCI input clock fall time	tCKf			100			100			100	ns	

5

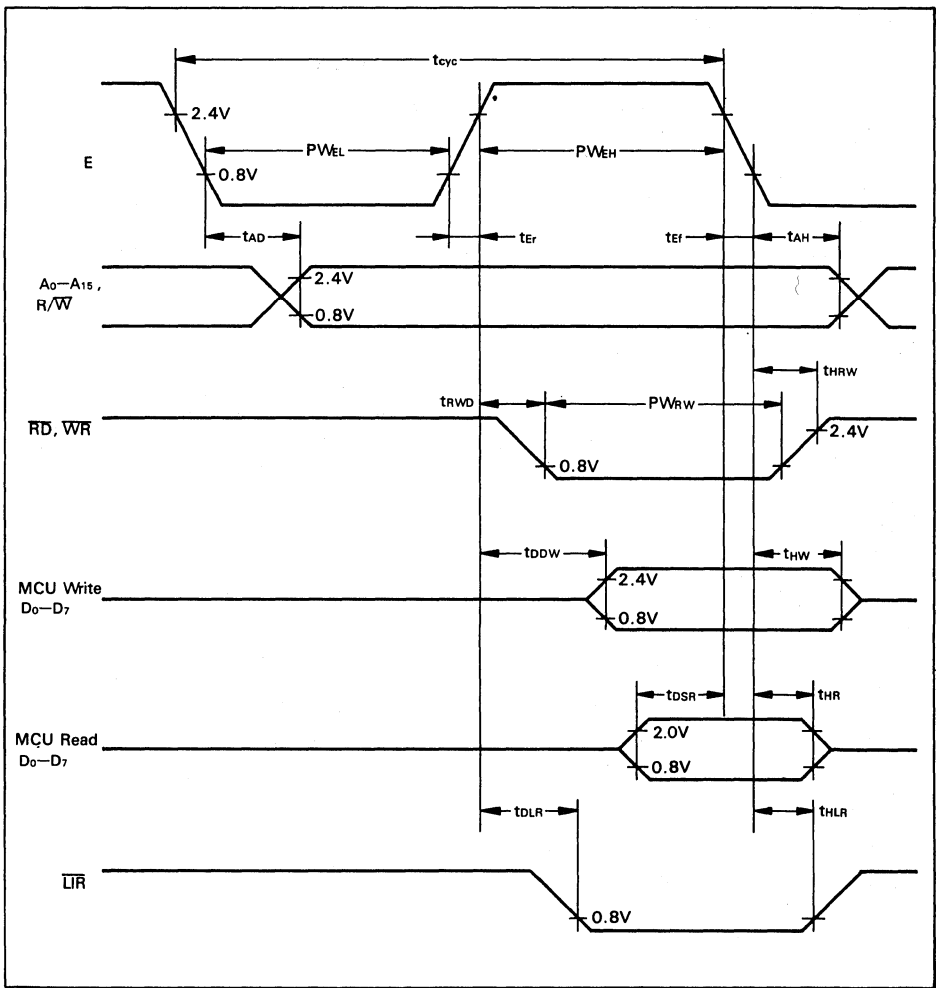


Figure I-25. Mode 1, Mode 2 Bus Timing

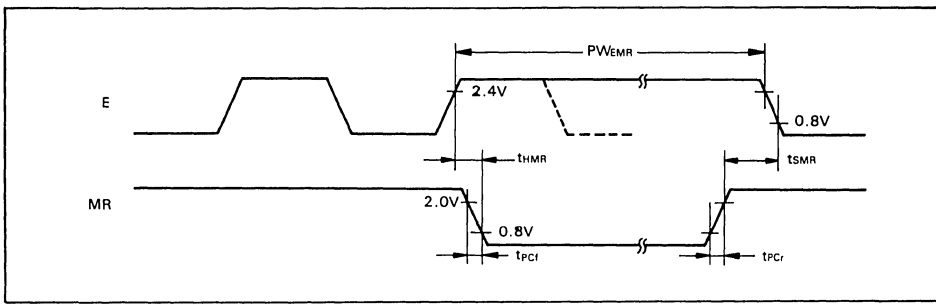


Figure I-26. Memory Ready and E Clock Timing

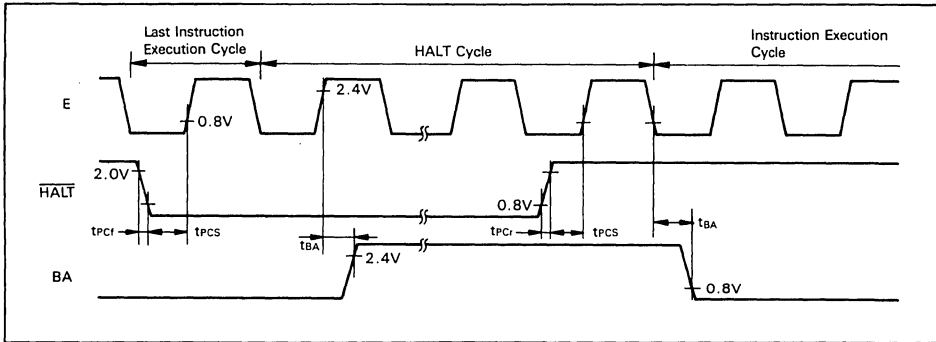


Figure I-27.  $\overline{\text{HALT}}$  and BA Timing

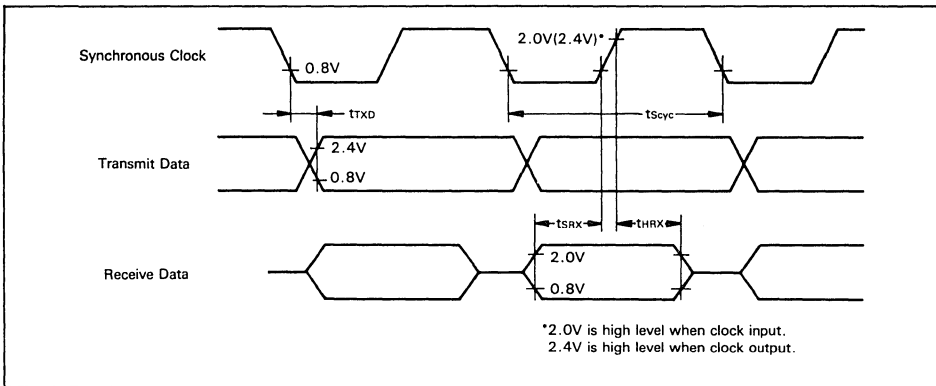


Figure I-28. SCI Clocked Synchronous Timing

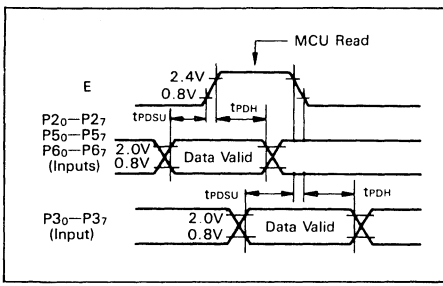


Figure I-29. Port Data Set-up and Hold Times (MCU Read)

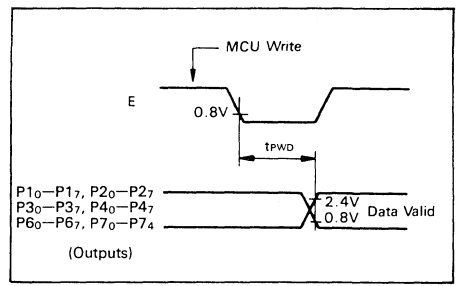


Figure I-30. Port Data Delay Times (MCU Write)

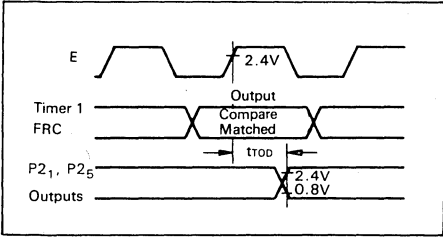


Figure I-31. Timer 1 Output Timing

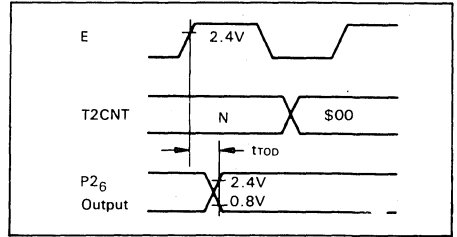


Figure I-32. Timer 2 Output Timing

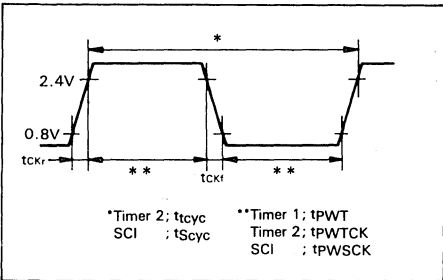


Figure I-33. Timer 1/2, SCI Input Clock Timing

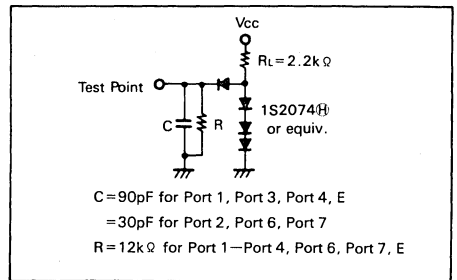


Figure I-34. Bus Timing Test Loads (TTL Load)

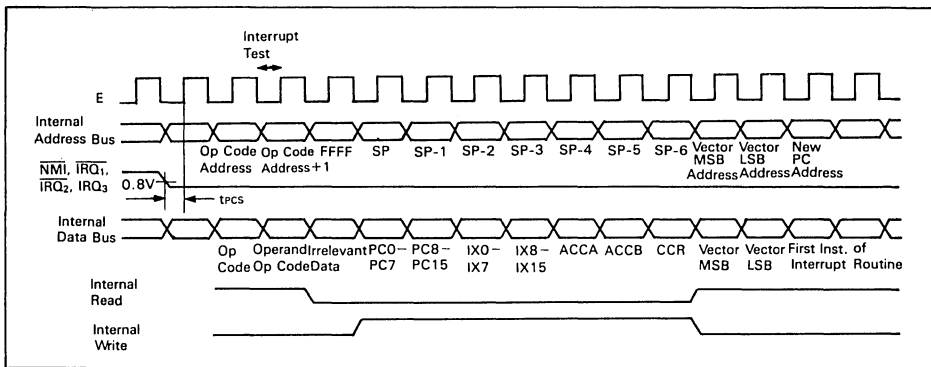


Figure I-35. Interrupt Sequence

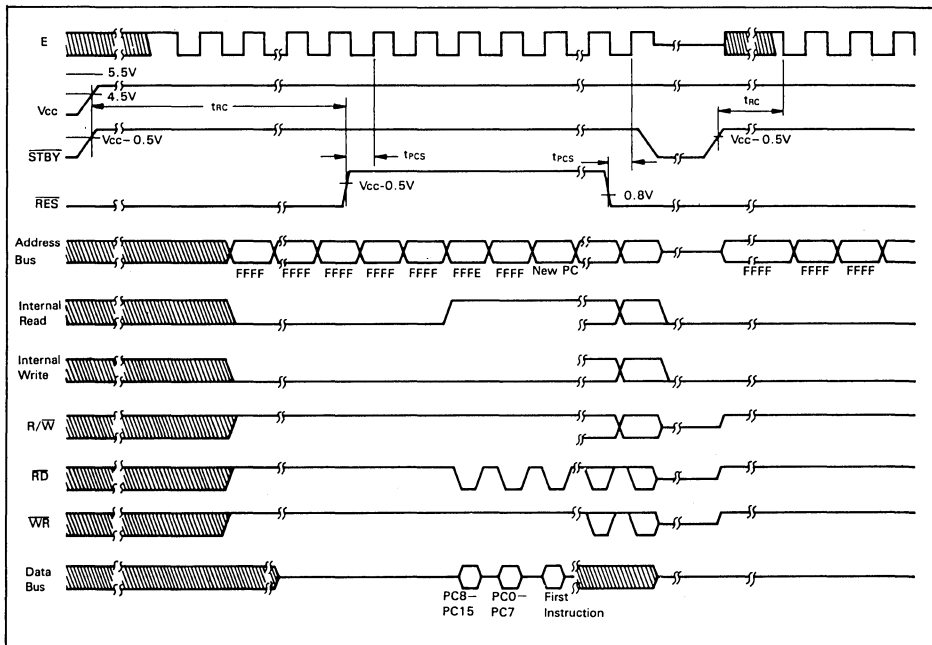


Figure I-36. Reset Timing

## Programming Electrical Characteristics

### DC Characteristics

( $V_{CC}=5\text{ V} \pm 5\%$ ,  $V_{PP}=21\text{ V} \pm 0.5\text{ V}$ ,  $V_{SS}=0\text{ V}$ ,  $T_a=25^\circ\text{C} \pm 5^\circ\text{C}$ , unless otherwise notes.)

Item	Symbol	Min	Typ	Max	Unit	Test Condition
Input high voltage	$V_{IH}$	2.2	—	$V_{CC}+1.0$	V	
Input low voltage	$V_{IL}$	-0.1	—	0.8	V	
Output high voltage	$V_{OH}$	2.0	—	—	V	$I_{OH}=-200\mu\text{A}$
Output low voltage	$V_{OL}$	—	—	0.45	V	$I_{OL}+1.6\text{mA}$
Input leakage current	$ I_{LI} $	—	—	10	$\mu\text{A}$	$V_{in}=5.25\text{V}/0.4\text{V}$
$V_{PP}$ voltage	$V_{PP}$	20.5	21	21.5	mA	
$V_{PP}$ current	$I_{PP}$	—	—	30	mA	$V_{PP}=21\text{V}$ , $\overline{CE}=V_{IV}$

### AC Characteristics

( $V_{CC}=5\text{ V} \pm 5\%$ ,  $V_{PP}=12.5\text{ V} \pm 0.3\text{ V}$ ,  $T_a=25^\circ\text{C} \pm 5^\circ\text{C}$ , unless otherwise noted.)

Item	Symbol	Min	Typ	Max	Unit	Test Condition
Address set-up time	$t_{AS}$	2	—	—	$\mu\text{s}$	
$\overline{OE}$ set-up time	$t_{OES}$	2	—	—	$\mu\text{s}$	
$\overline{OE}$ hold time	$t_{OEH}$	2	—	—	$\mu\text{s}$	
Data set-up time	$t_{DS}$	2	—	—	$\mu\text{s}$	
Address hold time	$t_{AH}$	0	—	—	$\mu\text{s}$	
Data hold time	$t_{DH}$	2	—	—	$\mu\text{s}$	
Output disable delay time	$t_{DF}$	0	—	130	ns	
Data Valid from $\overline{CE}$	$t_{DV}$	—	—	1	$\mu\text{s}$	$\overline{CE}=V_{IL}$ , $\overline{OE}=V_{IL}$
$\overline{CE}$ pulse width	$t_{PW}$	45	50	55	ms	
$\overline{OE}$ pulse rise time	$t_{PRT}$	50	—	—	ns	
$V_{PP}$ recovery time	$t_{VR}$	2	—	—	$\mu\text{s}$	

Note:  $t_{DF}$  is defined when output becomes open because output level can not be referred.

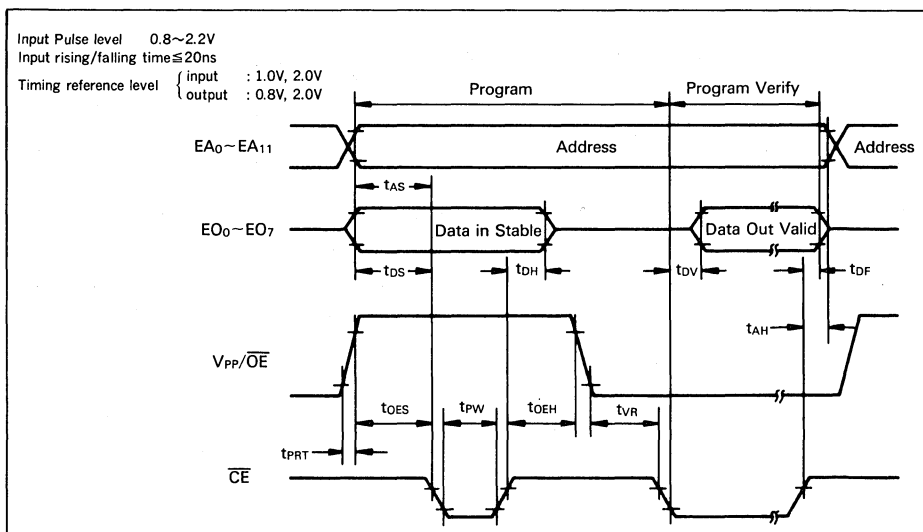


Figure I-37. PROM Programming/Verify Timing

# Appendix II. Instruction Execution Cycles

## II.1 Instruction Execution Cycles

By the pipeline control of the HD63701X0, MULT, PUL, DAA and XGDX instructions etc. prefetch the next instruction. So attention is necessary to the counting of the instruction cycles because it is different from the existent one ..... op-code fetch to the next instruction op-code.

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	$\overline{RD}$	WR	$\overline{LIR}$	Data Bus
-----------------------------	--------	---------	-------------	-----	-----------------	----	------------------	----------

### IMMEDIATE

ADC ADD	2	1	Op Code Address+1	1	0	1	1	Operand Data
AND BIT		2	Op Code Address+2	1	0	1	0	Next Op Code
CMP EOR								
LDA ORA								
SBC SUB								
ADDD CPX	3	1	Op Code Address+1	1	0	1	1	Operand Data (MSB)
LDD LDS		2	Op Code Address+2	1	0	1	1	Operand Data (LSB)
LDX SUBD		3	Op Code Address+3	1	0	1	0	Next Op Code

### DIRECT

ADC ADD	3	1	Op Code Address+1	1	0	1	1	Address of Operand (LSB)
AND BIT		2	Address of Operand	1	0	1	1	Operand Data
CMP EOR		3	Op Code Address+2	1	0	1	0	Next Op Code
LDA ORA								
SBC SUB								
STA	3	1	Op Code Address+1	1	0	1	1	Destination Address
		2	Destination Address	0	1	0	1	Accumulator Data
		3	Op Code Address+2	1	0	1	0	Next Op Code
ADDD CPX	4	1	Op Code Address+1	1	0	1	1	Address of Operand (LSB)
LDD LDS		2	Address of Operand	1	0	1	1	Operand Data (MSB)
LDX SUBD		3	Address of Operand+1	1	0	1	1	Operand Data (LSB)
		4	Op Code Address+2	1	0	1	0	Next Op Code
STD STS	4	1	Op Code Address+1	1	0	1	1	Destination Address (LSB)
STX		2	Destination Address	0	1	0	1	Register Data (MSB)
		3	Destination Address+1	0	1	0	1	Register Data (LSB)
		4	Op Code Address+2	1	0	1	0	Next Op Code
JSR	5	1	Op Code Address+1	1	0	1	1	Jump Address (LSB)
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer-1	0	1	0	1	Return Address (MSB)
		5	Jump Address	1	0	1	0	First Subroutine Op Code
TIM	4	1	Op Code Address+1	1	0	1	1	Immediate Data
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	Op Code Address+3	1	0	1	0	Next Op Code
AIM EIM	6	1	Op Code Address+1	1	0	1	1	Immediate Data
OIM		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	FFFF	1	1	1	1	Restart Address (LSB)
		5	Address of Operand	0	1	0	1	New Operand Data
		6	Op Code Address+3	1	0	1	0	Next Op Code

(continued)



Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
-----------------------------	--------	---------	-------------	-----	-----------------	-----------------	------------------	----------

**INDEXED**

JMP	3	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Jump Address	1	0	1	0	First Op Code of Jump Routine
ADC ADD AND BIT CMP EOR LDA ORA SBC SUB TST	4	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	1	0	1	1	Operand Data
		4	Op Code Address+2	1	0	1	0	Next Op Code
STA	4	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	0	1	0	1	Accumulator Data
		4	Op Code Address+2	1	0	1	0	Next Op Code
ADDD CPX LDD LDS LDX SUBD ADD	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	1	0	1	1	Operand Data (MSB)
		4	IX+Offset+1	1	0	1	1	Operand Data (LSB)
		5	Op Code Address+2	1	0	1	0	Next Op Code
STD STS STX	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	0	1	0	1	Register Data (MSB)
		4	IX+Offset+1	0	1	0	1	Register Data (LSB)
		5	Op Code Address+2	1	0	1	0	Next Op Code
JSR	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer-1	0	1	0	1	Return Address (MSB)
		5	IX+Offset	1	0	1	0	First Subroutine Op Code
ASL ASR COM DEC INC LSR NEG ROL ROR	6	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	1	0	1	1	Operand Data
		4	FFFF	1	1	1	1	Restart Address (LSB)
		5	IX+Offset	0	1	0	1	New Operand Data
		6	Op Code Address+2	1	0	1	0	Next Op Code
TIM	5	1	Op Code Address+1	1	0	1	1	Immediate Data
		2	Op Code Address+2	1	0	1	1	Offset
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	IX+Offset	1	0	1	1	Operand Data
		5	Op Code Address+3	1	0	1	0	Next Op Code
CLR	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	1	0	1	1	Operand Data
		4	IX+Offset	0	1	0	1	00
		5	Op Code Address+2	1	0	1	0	Next Op Code
AIM EIM OIM	7	1	Op Code Address+1	1	0	1	1	Immediate Data
		2	Op Code Address+2	1	0	1	1	Offset
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	IX+Offset	1	0	1	1	Operand Data
		5	FFFF	1	1	1	1	Restart Address (LSB)
		6	IX+Offset	0	1	0	1	New Operand Data
		7	Op Code Address+3	1	0	1	0	Next Op Code

(continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
-----------------------------	--------	---------	-------------	-----	-----------------	-----------------	------------------	----------

**EXTEND**

JMP	3	1	Op Code Address+1	1	0	1	1	Jump Address (MSB)
		2	Op Code Address+2	1	0	1	1	Jump Address (LSB)
		3	Jump Address	1	0	1	0	Next Op Code
ADC ADD TST AND BIT CMP EOR LDA ORA SBC SUB	4	1	Op Code Address+1	1	0	1	1	Address of Operand (MSB)
2		Op Code Address+2	1	0	1	1	Address of Operand (LSB)	
3		Address of Operand	1	0	1	1	Operand Data	
4		Op Code Address+3	1	0	1	0	Next Op Code	
STA	4	1	Op Code Address+1	1	0	1	1	Destination Address (MSB)
		2	Op Code Address+2	1	0	1	1	Destination Address (LSB)
		3	Destination Address	0	1	0	1	Accumulator Data
		4	Op Code Address+3	1	0	1	0	Next Op Code
ADDD CPX LDD LDS LDX SUBD	5	1	Op Code Address+1	1	0	1	1	Address of Operand (MSB)
2		Op Code Address+2	1	0	1	1	Address of Operand (LSB)	
3		Address of Operand	1	0	1	1	Operand Data (MSB)	
4		Address of Operand+1	1	0	1	1	Operand Data (LSB)	
5		Op Code Address+3	1	0	1	0	Next Op Code	
STD STS STX	5	1	Op Code Address+1	1	0	1	1	Destination Address (MSB)
2		Op Code Address+2	1	0	1	1	Destination Address (LSB)	
3		Destination Address	0	1	0	1	Register Data (MSB)	
4		Destination Address+1	0	1	0	1	Register Data (LSB)	
5		Op Code Address+3	1	0	1	0	Next Op Code	
JSR	6	1	Op Code Address+1	1	0	1	1	Jump Address (MSB)
		2	Op Code Address+2	1	0	1	1	Jump Address (LSB)
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	Stack Pointer	0	1	0	1	Return Address (LSB)
		5	Stack Pointer-1	0	1	0	1	Return Address (MSB)
		6	Jump Address	1	0	1	0	First Subroutine Op Code
ASL ASR COM DEC INC LSR NEG ROL ROR	6	1	Op Code Address+1	1	0	1	1	Address of Operand (MSB)
2		Op Code Address+2	1	0	1	1	Address of Operand (LSB)	
3		Address of Operand	1	0	1	1	Operand Data	
4		FFFF	1	1	1	1	Restart Address (LSB)	
5		Address of Operand	0	1	0	1	New Operand Data	
6		Op Code Address+3	1	0	1	0	Next Op Code	
CLR	5	1	Op Code Address+1	1	0	1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	Address of Operand	0	1	0	1	00
		5	Op Code Address+3	1	0	1	0	Next Op Code

(continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	RD	WR	LIR	Data Bus
-----------------------------	--------	---------	-------------	-----	----	----	-----	----------

**IMPLIED**

ABA ABX ASL ASLD ASR CBA CLC CLI CLR CLV COM DEC DES DEX INC INS INX LSR LSRD ROL ROR NOP SBA SEC SEI SEV TAB TAP TBA TPA TST TSX TXS	1	1	Op Code Address+1	1	0	1	0	Next Op Code
DAA XGDY	2	1 2	Op Code Address+1 FFFF	1 1	0 1	1 1	0 1	Next Op Code Restart Address (LSB)
PULA PULB	3	1 2 3	Op Code Address+1 FFFF Stack Pointer+1	1 1 1	0 1 0	1 1 1	0 1 1	Next Op Code Restart Address (LSB) Data from Stack
PSHA PSHB	4	1 2 3 4	Op Code Address+1 FFFF Stack Pointer Op Code Address+1	1 1 0 1	0 1 1 0	1 1 0 1	1 1 1 0	Next Op Code Restart Address (LSB) Accumulator Data Next Op Code
PULX	4	1 2 3 4	Op Code Address+1 FFFF Stack Pointer+1 Stack Pointer+2	1 1 1 1	0 1 0 0	1 1 1 1	0 1 1 1	Next Op Code Restart Address (LSB) Data from Stack (MSB) Data from Stack (LSB)
PSHX	5	1 2 3 4 5	Op Code Address+1 FFFF Stack Pointer Stack Pointer-1 Op Code Address+1	1 1 0 0 1	0 1 1 1 0	1 1 0 0 1	1 1 1 1 0	Next Op Code Restart Address (LSB) Index Register (LSB) Index Register (MSB) Next Op Code
RTS	5	1 2 3 4 5	Op Code Address+1 FFFF Stack Pointer+1 Stack Pointer+2 Return Address	1 1 1 1 1	0 1 0 0 0	1 1 1 1 1	1 1 1 1 0	Next Op Code Restart Address (LSB) Return Address (MSB) Return Address (LSB) First Op Code of Return Routine
MUL	7	1 2 3 4 5 6 7	Op Code Address+1 FFFF FFFF FFFF FFFF FFFF FFFF	1 1 1 1 1 1 1	0 1 1 1 1 1 1	1 1 1 1 1 1 1	0 1 1 1 1 1 1	Next Op Code Restart Address (LSB) Restart Address (LSB) Restart Address (LSB) Restart Address (LSB) Restart Address (LSB) Restart Address (LSB) Restart Address (LSB)

(continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	RD	WR	LIR	Data Bus
-----------------------------	--------	---------	-------------	-----	----	----	-----	----------

**IMPLIED**

WAI	9	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer-1	0	1	0	1	Return Address (MSB)
		5	Stack Pointer-2	0	1	0	1	Index Register (LSB)
		6	Stack Pointer-3	0	1	0	1	Index Register (MSB)
		7	Stack Pointer-4	0	1	0	1	Accumulator A
		8	Stack Pointer-5	0	1	0	1	Accumulator B
		9	Stack Pointer-6	0	1	0	1	Conditional Code Register
RTI	10	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer+1	1	0	1	1	Conditional Code Register
		4	Stack Pointer+2	1	0	1	1	Accumulator A
		5	Stack Pointer+3	1	0	1	1	Accumulator B
		6	Stack Pointer+4	1	0	1	1	Index Register (MSB)
		7	Stack Pointer+5	1	0	1	1	Index Register (LSB)
		8	Stack Pointer+6	1	0	1	1	Return Address (MSB)
		9	Stack Pointer+7	1	0	1	1	Return Address (LSB)
		10	Return Address	1	0	1	0	First Op Code of Return Routine
SWI	12	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer-1	0	1	0	1	Return Address (MSB)
		5	Stack Pointer-2	0	1	0	1	Index Register (LSB)
		6	Stack Pointer-3	0	1	0	1	Index Register (MSB)
		7	Stack Pointer-4	0	1	0	1	Accumulator A
		8	Stack Pointer-5	0	1	0	1	Accumulator B
		9	Stack Pointer-6	0	1	0	1	Conditional Code Register
		10	Vector Address FFFA	1	0	1	1	Address of SWI Routine (MSB)
		11	Vector Address FFFB	1	0	1	1	Address of SWI Routine (LSB)
		12	Address of SWI Routine	1	0	1	0	First Op Code of SWI Routine
SLP	4	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		↓						
		↓						
		3	FFFF	1	1	1	1	Restart Address (LSB)
4	Op Code Address+1	1	0	1	0	Next Op Code		

(continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
-----------------------------	--------	---------	-------------	-----	-----------------	-----------------	------------------	----------

**RELATIVE**

BCC BCS	3	1	Op Code Address+1	1	0	1	1	Branch Offset	
BEQ BGE		2	FFFF	1	1	1	1	Restart Address (LSB)	
BGT BHI		3	3	{ Branch Address...Test="1" Op Code Address+2...Test="0"	1	0	1	0	First Op Code of Branch Routine Next Op Code
BLE BLS									
BLT BMT									
BNE BPL									
BRA BRN									
BVC BVS									
BSR	5	1	Op Code Address+1	1	0	1	1	Offset	
		2	FFFF	1	1	1	1	Restart Address (LSB)	
		3	Stack Pointer	0	1	0	1	Return Address (LSB)	
		4	Stack Pointer-1	0	1	0	1	Return Address (MSB)	
		5	Branch Address	1	0	1	0	First Op Code of Subroutine	

# Appendix III. Questions and Answers

This appendix contains some frequently asked questions about the HD6301X0, HD63701X0, and HD6303X.

## III.1 Parallel Ports

### III.1.1 DDR and Data Register

**Question:** Should the data or DDR (data direction register) be set first, when an I/O port functions as an output port?

**Answer:** Output data should be stored in the data register first, then DDR should be set (DDR = 1). If DDR is set first, unknown data will be output from the port.

**Supplement:** DDR (data direction register)

DDR programs I/O port as an input or output.

DDR = 1: output

DDR = 0: input

DDR is initialized to 0 during reset.



### III.1.2 Port 7 Upper Bits

**Question:** What is the state of the upper 3 bits in port 7 (5-bit output port) when reading port 7 in mode 3 (single chip mode)?

**Answer:** The upper 3 bits in port 7 are all set to 1. Port 7 DDR can read the contents of the data register by using the bit manipulation instruction.

**Supplement:** Ports 1 and 4 can also be read with bit manipulation instructions.

### III.1.3 SCLK/P2<sub>2</sub> Pin

**Question:** How do you use the P2<sub>2</sub> (SCLK/P2<sub>2</sub> multiplexed pin) as an I/O port?

**Answer:** To use the P2<sub>2</sub> as an I/O port, set bit 1 in the port 2 DDR (data direction register), and CC0, CC1, and CC2 in the RMCR (rate/mode control register) as in table III-1.

Table III-1. P2<sub>2</sub> I/O Settings

Bit	Setting
Bit 1 of port 2 DDR (Note1)	0 (Input port) 1 (Output port)
CC0 (Note 2)	1
CC1	0
CC2	0 or 1

Notes:

1. Bit 1 of the port 2 DDR selects the direction of 7 bits P2<sub>1</sub> - P2<sub>7</sub>.
2. During reset, CC0, CC1 and CC2 are cleared to 0 and the P2<sub>2</sub> functions as SCLK pin.

**Supplement:** The CC0, CC1, and CC2 (clock control format select) program the SCI data format and the SCI clock direction.

The DDR (data direction register) programs the direction of the I/O port.

DDR = 0: Input

DDR = 1: Output

### III.1.4 P5<sub>3</sub>/ $\overline{\text{HALT}}$ Pin

**Question:** How can P5<sub>3</sub> (P5<sub>3</sub>/ $\overline{\text{HALT}}$  multiplexed pin) be used as an input-only port in expanded mode (modes 1 and 2)?

**Answer:** In expanded mode, P5<sub>3</sub> functions as  $\overline{\text{HALT}}$  pin with HLTE bit = 1 during reset. To use P5<sub>3</sub> as an input port, hold it high until 0 is written in the HLTE after reset, inhibiting  $\overline{\text{HALT}}$  input.

## III.2 Serial Port

### II.2.1 RDRF in Wake-Up Mode

**Question:** When using the SCI in the asynchronous mode with the receive enable bit (RE) of the transmit/receive control status register (TRCSR) = 1 and wake-up bit (WU) = 1, what is the state of the receive data register full bit (RDRF)?

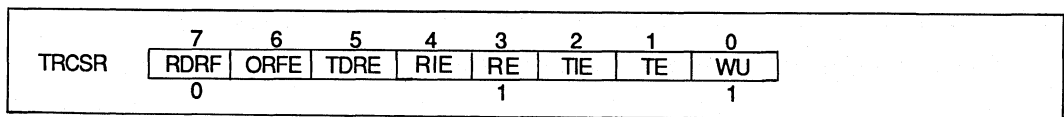


Figure III-1. Transmit/Receive Control Status Register in Wake-Up Mode



**Answer:** When the wake-up flag is set (WU = 1), the RDRF flag is not set (RDRF = 0).

### III.2.2 SCLK Direction and DDR

**Question:** When using the P2<sub>2</sub> (SCLK/bit 2 of I/O port 2) as the SCI clock I/O, is the clock direction determined by CC0, CC1 and CC2 (clock control/form select) in the RMCR (rate/mode control register) regardless of bit 2 of the port 2 DDR?

**Answer:** Yes, it is determined by CC0, CC1 and CC2 independently of the port 2 DDR. When used as an I/O port, its I/O direction is determined by bit 2 of the port 2 DDR. In this case, CC0, CC1 and CC2 should be set to a mode where P2<sub>2</sub> is not used as SCI clock (CC0, CC1, and CC2 set to 101, or 100). CC0, CC1, and CC2 are cleared to 0 at reset (table III-2).

Table III-2. P2<sub>2</sub> Direction

	P2 <sub>2</sub>	SCLK
Port 2 DDR	Input or output	No effect
CC0	1	CC0, CC1, CC2 determine
CC1	0	clock form, direction
CC2	0 or 1	

**Supplement:** The CC0, CC1, and CC2 (clock control format select) program the SCI data format and the SCI clock direction.

The DDR (data direction register) programs the direction of the I/O port.

DDR = 0: Input

DDR = 1: Output

### III.2.3 Receive Sampling Clock

**Question:** What is the relation between the receive data sampling clock at the SCI receive, and the data transfer rate?

**Answer:** The sampling clock is sixteen times as fast as the transfer rate.



### III.2.4 Sampling Error

**Question:** What does "sampling error" mean?

**Answer:** "Sampling error" means receive margin in SCI operation. The HD63701X0 detects a start bit at the negative edge of the sampling clock, and samples the start bit and data bit at the positive edge of the sampling clock.

The general equation of the receive margin is shown as follows (figure III.2).

$$M = \{(0.5 - 1/2N) - (D - 0.5)/N - (L - 0.5)F\} \times 100 (\%)$$

M: Receive margin

N: Baud rate ratio to sampling clock

D: Duty of the longer sampling clock of high and low (0.5 - 1)

L: Frame Length (7 - 12)

F: Absolute value of deviation of sampling clock frequency

An abbreviated version is:

$$M = (0.5 - 1/2N) \times 100 (\%) \quad (\text{Condition: } D = 0.5, F = 0)$$

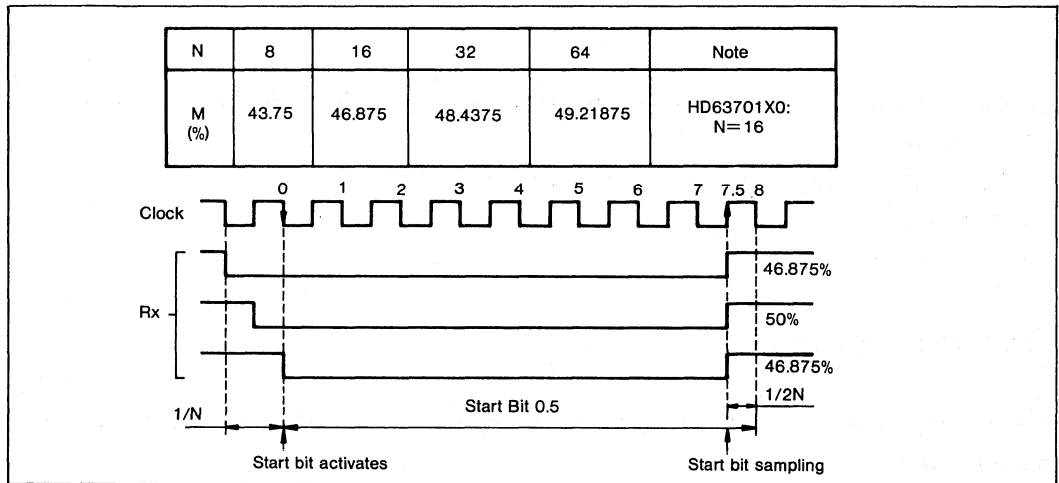


Figure III-2. Sampling Error

### III.3 Timer/Counter

#### III.3.1 Reading the FRC

**Question:** When you read the free-running counter (FRC) of the timer 1 by a double-byte load instruction, is the read value correct?

**Answer:** It is correct. In the first cycle, the high byte of the FRC is read, when the low byte is set in a temporary register. At the next cycle, the data stored in the temporary register is read (figure III-3).

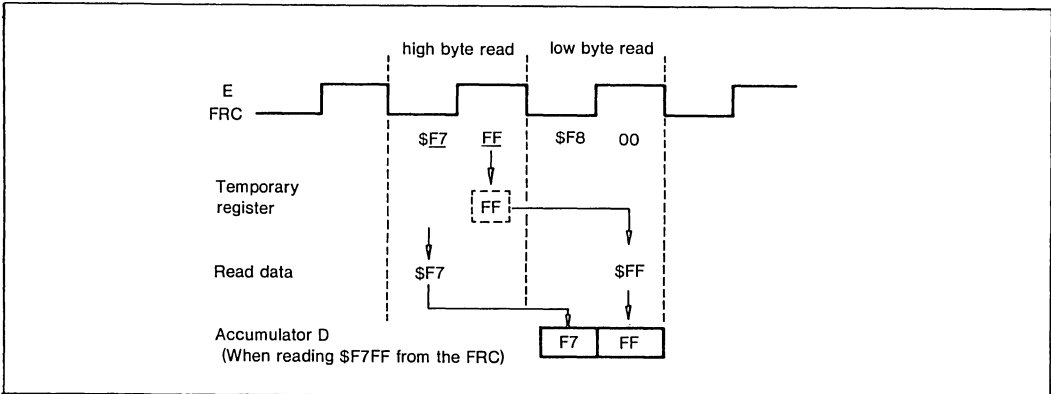


Figure III-3. FRC Double-Byte Read

**Supplement:** To read the timer FRC correctly, use double-byte load instructions (LDD,LDX).

#### II.3.2 Reading the FRC in the HD6801V

**Question:** How is FRC writing in the HD6301X0, HD6303X, and HD63701X0 different from the HD6801V?

**Answer:** The difference is shown in table III-3.

Table III-3. HD6301X0/HD6303X/HD63701X0 and HD6801V Write Differences

Type	How to Write (Preset)
HD6801V	The FRC is always preset to \$FFF8.
HD6301X0, HD6303X, HD63701X0	Writing to the high byte presets the FRC to \$FFF8. Data is always set in the FRC by a double-byte store instruction.

See figure III-4 for an example.

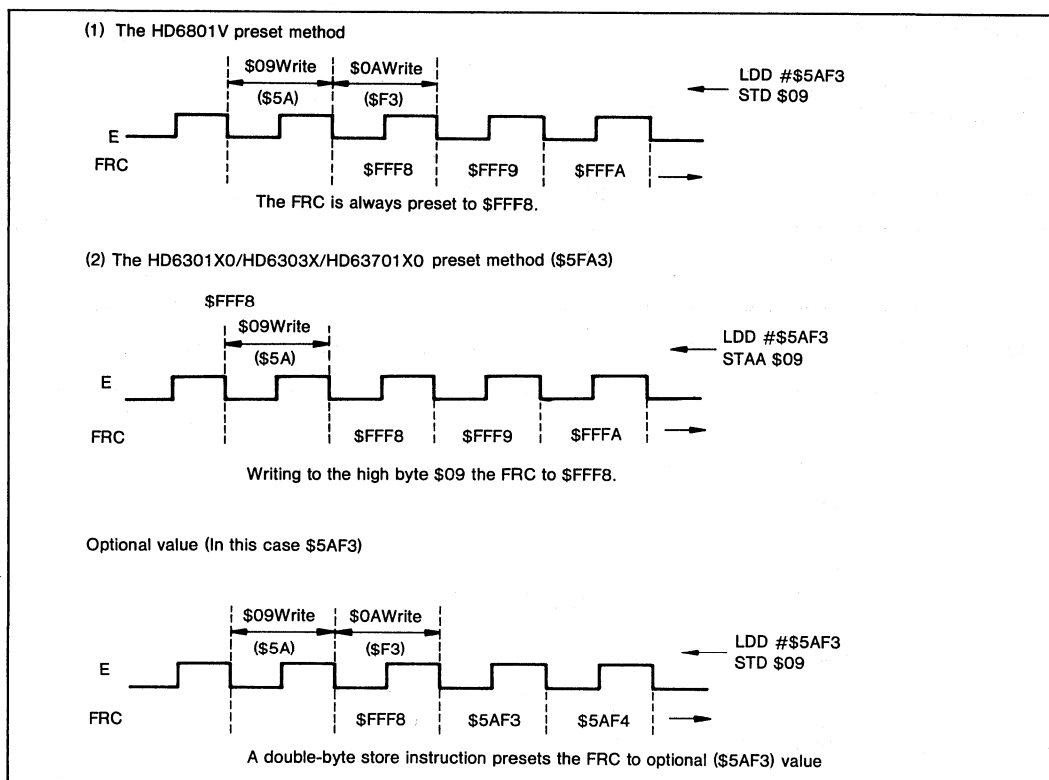


Figure III-4. FRC Writing for HD6301X0/HD6303X/HD63701X0 and HD6801V

### III.3.3 ECMI Interrupt

**Question:** Timer 2 is used by writing 0 to enable counter match interrupt (ECMI) of the timer control/status register 3 (TCSR3). When a counter match flag (CMF) of TCSR3 becomes 1, 1 is written to ECMI. Does this generate an interrupt?

**Answer:** Yes. When the time constant register (TCONR) matches the timer 2 counter, the CMF is set to 1 and kept at 1 unless 0 is written in by software. An interrupt will occur if ECMI = 1 after CMF = 1.

**Supplement:** A timer 2 interrupt is generated with CMF = 1 and ECMI = 1.

ECMI defines internal interrupt (IRQ<sub>3</sub>) enable/disable.

ECMI = 0: disable

ECMI = 1: enable

### III.3.4 SCI and Writing to Timers

**Question:** When the SCI is operating, can data be written into the timer 1 FRC or timer 2 T2CNT?

**Answer:** If the SCI is using an external clock, the timer 1 FRC and the timer 2 T2CNT can be written

into. In the case of an internal clock, either the FRC or the T2CNT is used as a clock-source counter (note 1). No clock-source counter can be written to. Note that there are some restrictions, as follows:

1. External clock operation
  - a. Timer 1 FRC can be written to
  - b. Timer 2 T2CNT can be written to
2. Internal clock operation
  - a. Using timer 1 FRC as an internal clock
    - Don't write to the timer 1 FRC during SCI operation.
    - Timer 2 T2CNT can be written to.
  - b. Using timer 2 T2CNT as an internal clock
    - The timer 1 FRC can be written to, except when input clock to T2CNT is E/8 or E/128. E/8, E/128 come from the timer 1 FRC. If these clocks are selected as T2CNT input clocks, writing to the FRC will delay them.
    - Don't write to timer 2 T2CNT during SCI operation.

**Supplement:** When an internal clock is operating the SCI, writing to the clock-source counter will delay the SCI transfer rate.

### III.4 Bus Interface

#### III.4.1 E and Memory Ready

**Question:** What is the internal E clock state when the CPU uses the memory ready function?

**Answer:** Internal E clock operates at normal frequency (figure III-5). Since the timer count and the SCI transfer rate are set by the internal E clock, they are not also affected by the memory ready function.

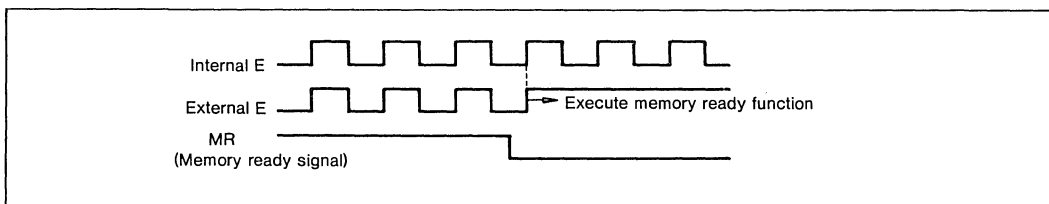


Figure III-5. Internal and External E Clocks

**Supplement:** It is impossible to examine the internal E clock from an external pin when using the memory ready function.

### III.4.2 Memory Ready and Halt After Reset

**Question:** After reset, are memory ready and halt functions enabled or disabled?

**Answer:** Both are enabled. MR and  $\overline{\text{HALT}}$  in three operating modes is shown in table III-4.

Table III-4. Operating Modes

Operating Mode	Memory Ready	Halt
Expanded mode	1 Enabled (note)	Enabled
	2 Enabled (note)	Enabled
Single-chip mode	No memory ready function	No halt function

Note: Invalid when accessing internal address space

**Supplement:** In the expanded mode (modes 1, 2), the memory ready bit (MRE) and halt enable bit (HLTE) of the RAM/port 5 control register are set to 1 during reset, enabling memory ready and halt functions.

### III.4.3 Buses at Internal Address Access

**Question:** When you access internal memory space, what states are the address buses, data buses, and control lines in?

**Answer:** Address buses and control lines ( $\overline{\text{RD}}$ ,  $\overline{\text{WR}}$ ,  $\text{R}/\overline{\text{W}}$ ) are always output regardless of internal or external address space accessing. During writes to the internal address space, the same data is output from the data bus. During reads, the data buses become high impedance.

### III.4.4 External Access to Register Addresses

**Question:** When using external memory at the addresses shown below in expanded modes (modes 1, 2), some addresses overlap internal registers and RAM addresses (figure III-6). In such a case, are there any problems?

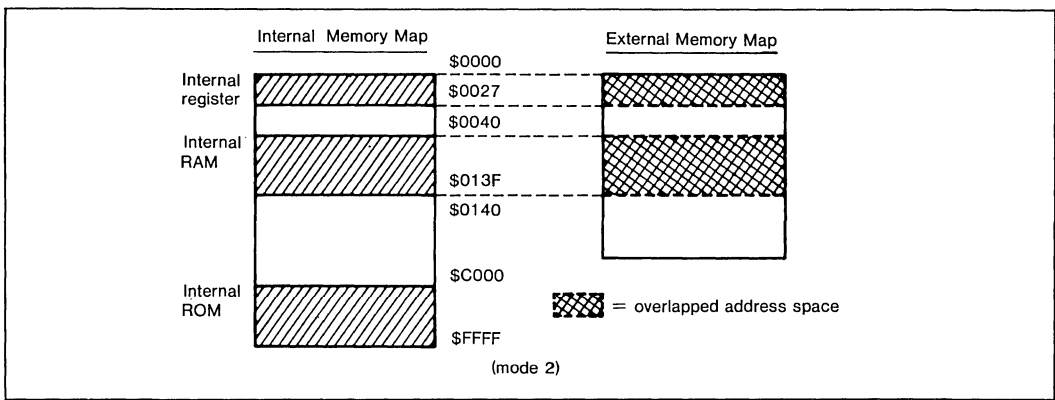


Figure III-6. Overlapping Addresses

**Answer:** There are no problems, but the overlapped addresses in the external memory space should not be used. When writing to the overlapping addresses, the same data is written into the internal and external address space. When reading, data is read from the internal, and the external address data is ignored.

**Supplement:** If the RAM enable bit (RAME) of the RAM/port 5 control register is 0, a read/write from/to the internal RAM space is invalid, and both operations are executed to the overlapped external address space.

5

### III.4.5 Buses During WAI

**Question:** What states are address buses, data buses, and control lines in after WAI instruction execution?

**Answer:** They are as in table III-5.

Table III-5. WAI State

Line	State
Address bus	FFFF (High)
Data bus	High impedance
$\overline{R/W}$	High
$\overline{RD}$	High
$\overline{WR}$	High

### III.5 Interrupt Control

#### III.5.1 $\overline{\text{IRQ}}_1$ During Standby

**Question:** When the CPU is returning from standby mode ( $\overline{\text{RES}} = \text{low}$ ,  $\overline{\text{STBY}} = \text{low}$ ) with  $\overline{\text{IRQ}}_1$  low, can the interrupt be accepted if  $\overline{\text{IRQ}}_1$  low continues after return?

**Answer:** It cannot. Interrupts can be accepted when  $\text{IRQ1E} = 1$  and  $I = 0$ . After the CPU returns from standby, it has  $\text{IRQ1E} = 0$  and  $I = 1$ . To accept the interrupt, the software should make  $\text{IRQ1E} = 1$ ,  $I = 0$  after resetting.

**Supplement:**  $\text{IRQ1E}$  is the  $\overline{\text{IRQ}}_1$  interrupt enable bit of the RAM/port 5 control register. When  $\text{IRQ1E} = 1$ ,  $P5_0$  can be used as an interrupt pin.  $I$  is the interrupt mask bit. When  $I = 0$ , the CPU accepts interrupts.

#### III.5.2 Trap Interrupt

**Question:** How does the trap interrupt differ from other interrupts ( $\overline{\text{NMI}}$ ,  $\overline{\text{IRQ}}_1$ ,  $\overline{\text{IRQ}}_2$  and  $\text{IRQ}_3$ )?

**Answer:** The differences are:

- Return address (figure III-7)
- Interrupt sequence (figure III-8)

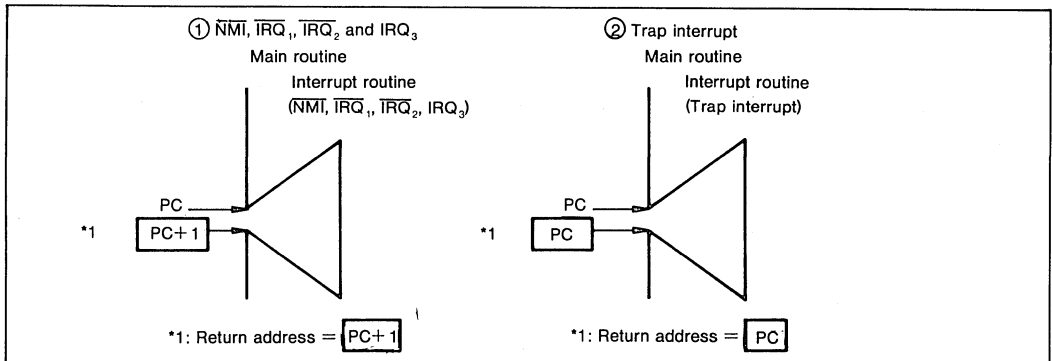


Figure III-7. Return Address

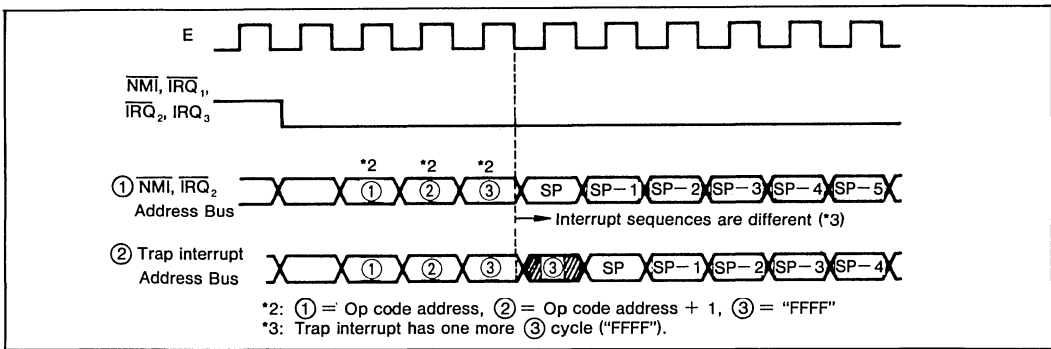


Figure III-8. Interrupt Sequence

### III.5.3 LIR During Interrupt

**Question:** What is the output state of the load instruction register ( $\overline{\text{LIR}}$ ) bit in the interrupt sequence?

**Answer:** The output state of  $\overline{\text{LIR}}$  is low in the following cycles:

1. Prefetch cycle of the last instruction cycle opcode just before interrupt sequence
2. Fetch cycle of the first opcode of the interrupt routine

The output state of  $\overline{\text{LIR}}$  in the interrupt sequence is shown below.

1. Last instruction execution cycle just before the interrupt sequence (figure III-9).

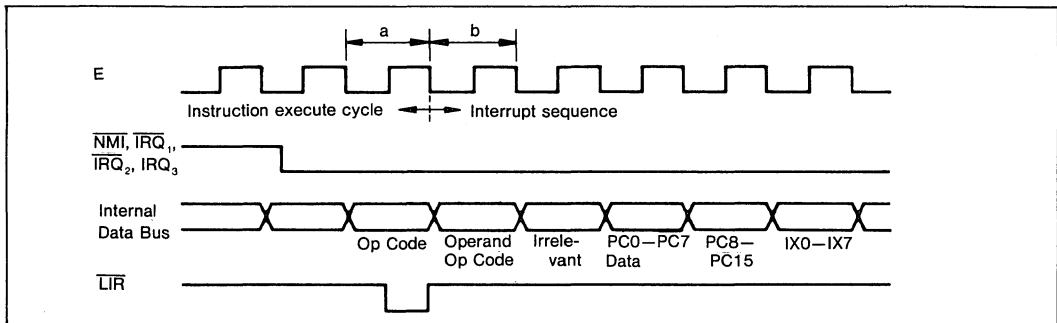


Figure III-9. Last Cycle Before Interrupt

- a.  $\overline{\text{LIR}}$  output is low at the last instruction execution cycle just before interrupt sequence is opcode prefetch.
- b. The first cycle of the interrupt sequence (2 in the above figure) is a dummy fetch cycle. In this cycle, there are two cases; an operand is on the data bus, or an opcode is on the bus. In both cases,  $\overline{\text{LIR}}$  output is low.



z. First opcode fetch cycle in the interrupt routine (figure III-10).

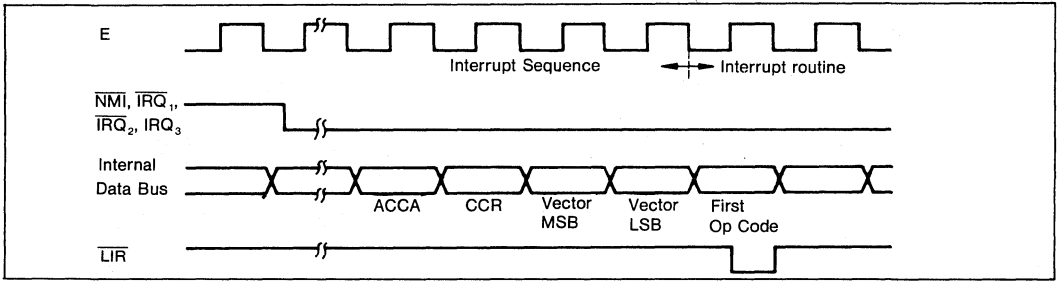


Figure III-10. First Cycle in Interrupt

$\overline{\text{LIR}}$  output is low when the first opcode of the interrupt routine is fetched.

**Supplement:** Load instruction register ( $\overline{\text{LIR}}$ ) low shows that instruction opcode is on the data bus.

### III.6 Oscillation Circuit

#### III.6.1 E Clock Triggering

**Question:** With which edge of the EXTAL clock does the E clock change, the rising or falling edge?

**Answer:** It changes synchronously with the falling edge (figure III-11).

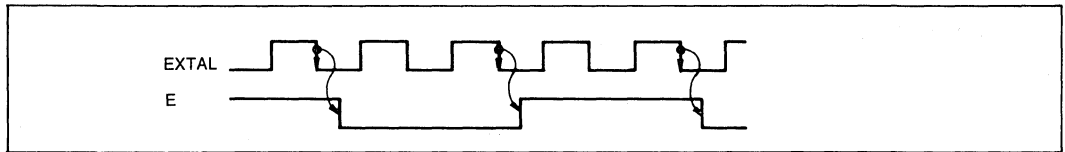


Figure III-11. E Clock Timing

### III.7 Reset

#### III.7.1 Ports at Reset

**Question:** What is the state of each port at reset?

**Answer:** It is as shown in table III-6.

Table III-6. Port State at Reset

Port	Mode	Reset
1 (A <sub>0</sub> -A <sub>7</sub> )	1, 2	High
	3	High impedance
2	1, 2	High impedance
	3	High impedance
3 (D <sub>0</sub> -D <sub>7</sub> )	1, 2	High impedance
	3	High impedance
4 (A <sub>8</sub> -A <sub>15</sub> )	1, 2	High
	3	High impedance
5	1, 2	High impedance
	3	High impedance
6	1, 2	High impedance
	3	High impedance
7	1, 2	Note 1
	3	High impedance

Note :

1.  $\overline{RD}$ ,  $\overline{WR}$ ,  $R/\overline{W}$ ,  $\overline{LIR}$  = high; BA = low

**Supplement:** E clock at reset is output at normal frequency after oscillation stabilization time.

### III.7.2 I/O Port Output After Reset

**Question:** What data does an I/O port output when the data direction register (DDR) = 1 after reset?

**Answer:** After reset, undefined data is output from the I/O port , since the data register of an I/O port is undefined. For the output state, put data in the data register before setting the DDR = 1.

### III.7.3 $\overline{\text{RES}}$ Schmitt Trigger

**Question:** Is a Schmitt trigger circuit provided with  $\overline{\text{RES}}$ ?

**Answer:** Yes (figure III-12).

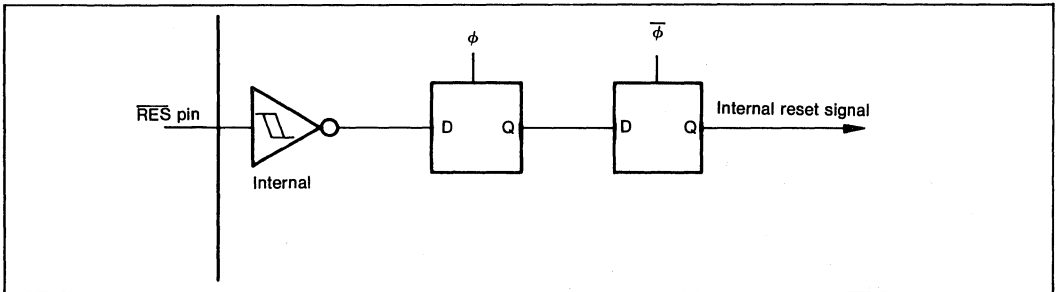


Figure III-12. Reset Circuit

### III.7.4 Reset Circuit Capacitance

**Question:** Does  $C_r$  in the reset circuit shown in figure III-13 ( $R_r \times C_r > 20 \text{ ms}$ ), have an upper limit?

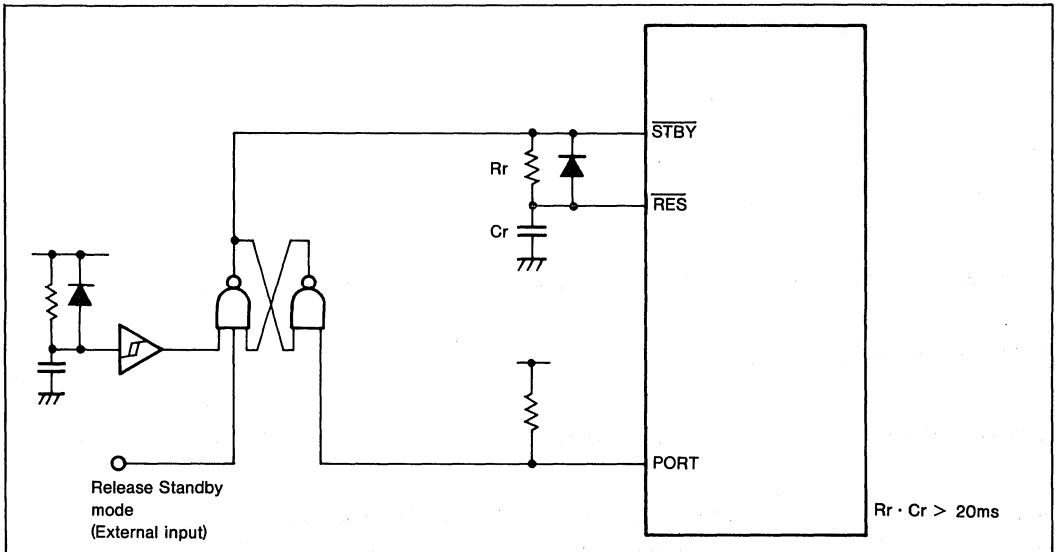


Figure III-13. Reset Input Circuit

**Answer:** No, because  $\overline{\text{RES}}$  is provided with a Schmitt trigger circuit (figure III-12).

### III.8 Low Power Dissipation Mode

#### III.8.1 Standby During Instruction Execution

**Question:** Does the CPU wait until the current instruction is executed to enter the standby mode?

**Answer:** No. The CPU enters standby mode regardless of the current instruction; the CPU goes into reset condition and the oscillator stops with  $\overline{STBY}$  low (figure III-14).

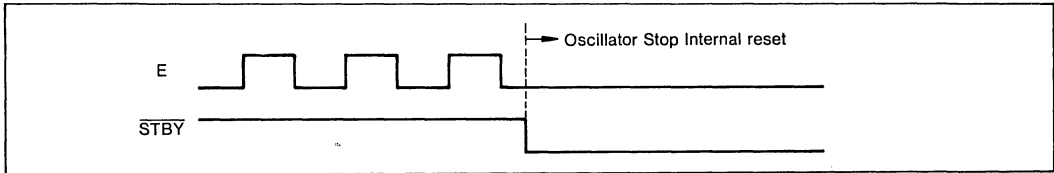


Figure III-14. E During Standby

#### III.8.2 Standby Timing

**Question:** The timing for the standby mode is shown in figure III-15 (see also figure 3-5). Is  $T_1$  in the figure defined?

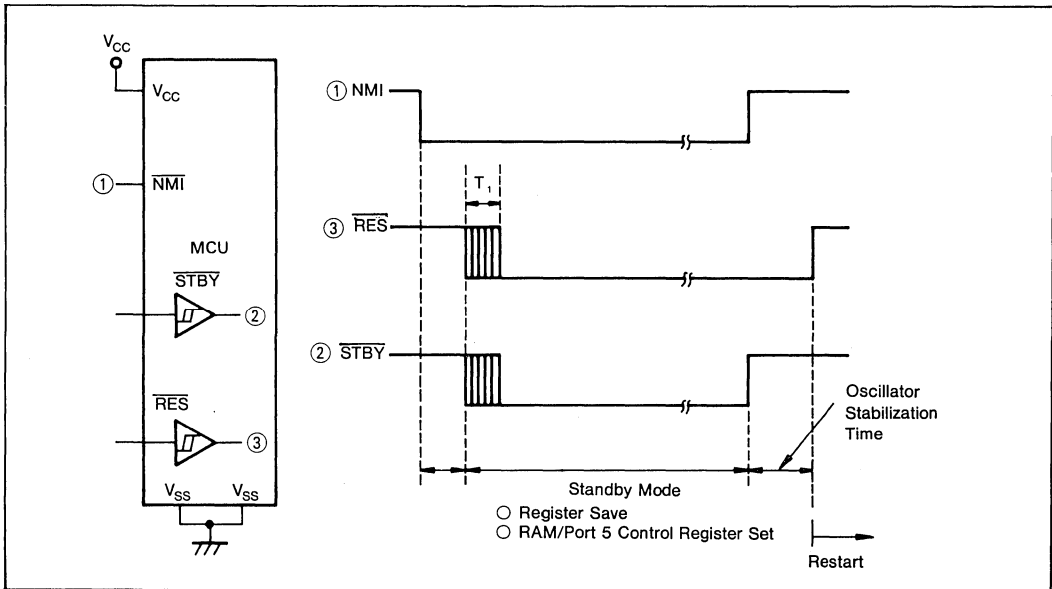


Figure III-15. Standby Mode Timing

**Answer:** It is not, but if the time for nonmaskable interrupt ( $\overline{NMI}$ ) is guaranteed, either  $\overline{RES}$  or  $\overline{STBY}$  can go low with no priority.

**Supplement:** The CPU goes to the standby mode independently of instruction execution sequence. Use the  $\overline{NMI}$  routine before entering standby mode.

### III.8.3 Ports at Standby

**Question:** What is the state of each I/O port during standby?

**Answer:** Each I/O port and E pin during standby is high impedance.

### III.8.4 Return from Standby Without Reset

**Question:** What occurs when the CPU returns from the standby mode without using reset start?

**Answer:** The CPU does not operate normally because the contents of each register are not defined. Therefore, always use the reset start when returning from the standby mode.

### III.8.5 Sleep and Standby Internal States

**Question:** What are the internal states in the sleep or standby mode?

**Answer:** They are as shown in table III-7.

Table III-7. Sleep and Standby Mode States

	Sleep Mode	Standby Mode
Oscillation circuit	Continues	Stops
CPU (register)	Stops (retained)	Stops (undefined)
RAM	Retained	Retained
I/O	Retained	High impedance
Timer	Continues	Stops
Serial communications	Continues	Stops
Internal registers	Retained	Reset
Cancel	Interrupt $\overline{\text{STBY}} = \text{low}$ Reset start	Reset start after $\overline{\text{STBY}} = \text{high}$

**Supplement:** Internal states in the standby mode are the same as those in reset start. Use the reset start when returning from the standby mode. In this case  $\overline{\text{RES}}$  should be kept low from  $\overline{\text{STBY}} = \text{high}$  during oscillation stabilization time (20 ms minimum).

### III.9 Software

#### III.9.1 Bit Manipulation Instructions

**Question:** How should the bit manipulation instructions of the HD6301X0, HD6303X, and HD63701X0 be written?

**Answer:** They are written as shown in figure III-16.

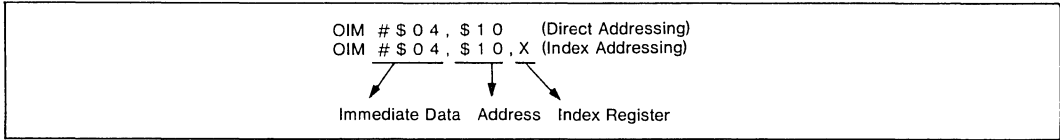


Figure III-16. OIM Example

This is an example of OR operation between the immediate data and the memory which stores the result in the memory. The AIM, EIM, and TIM instructions are written in the same way.

The bit manipulations in table III-8 have different mnemonics with the same opcode.

Table III-8. Shared Opcodes

Bit Manipulation Instruction	Instructions Having the Same Opcode	
	Mnemonic	Function
AIM	BCLR	0 AND Mi The memory bit i (i = 0 to 7) is cleared and the other bits don't change
OIM	BSET	1 OR Mi The memory bit i (i = 0 to 7) is set and the other bits don't change
EIM	BTGL	Mi EOR Mi The memory bit i (i = 0 to 7) is inverted and the other bits don't change
TIM	BTST	1 AND Mi AND operation test of the memory bit i (i = 0 to 7) and 1 is executed and its corresponding condition code is changed.

The mnemonics mentioned above can be written as in figure III-17.

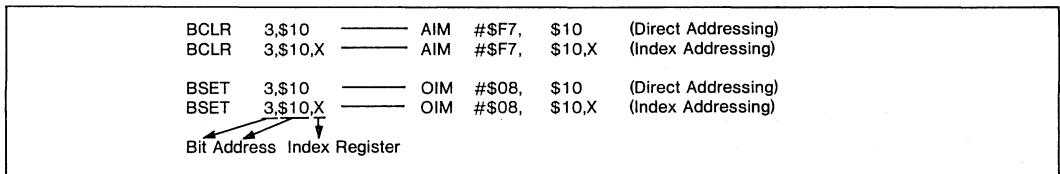


Figure III-17. Shared Opcode Instruction Format



### III.10 Others

#### III.10.1 RAME Disabled

**Question:** When executing a program with the RAM enable bit (RAME) of the RAM/port 5 control register disabled (RAME = 0),

1. What occurs if the internal RAM address is accessed?
2. What occurs if interrupt requests are generated?

**Answer:**

1. The internal RAM cannot be accessed. It is neither readable nor writable with RAME = 0, so in mode 1 or 2, the external memory is read/written into.
2. Interrupts are accepted, but the CPU will burst when returning from the interrupt with no stacking area other than the internal RAM.

**Supplement:**

1. RAME = 0; internal RAM is invalid. In modes 1 or 2, data can be read from the external memory.
2. RAME = 1; internal RAM is enabled.

#### III.10.2 RAME at Reset

**Question:** Is the RAM enable bit (RAME) set to 1 on reset at  $\overline{\text{RES}}$  low or the rising edge of  $\overline{\text{RES}}$ ?

**Answer:** It is set at the rising edge of  $\overline{\text{RES}}$  (figure III-18).

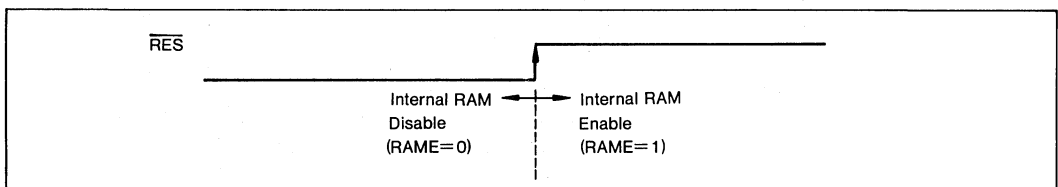


Figure III-18. RAME at Reset

**Supplement:** RAME is set/cleared by the software.

1. RAME = 0; Internal RAM is invalid. In mode 1 or 2, data can be read from the external memory.
2. RAME = 1; Internal RAM is enabled.

# Appendix IV. The Differences Between HD63701X0 and HD6301X0

## IV.1 The Differences Between HD63701X0 and HD6301X0

Item	HD63701X0	HD6301X0																								
$V_{pp}/\overline{OE}$ pin function	$V_{pp}/\overline{OE}$ MCU mode; Connected to $V_{SS}$ voltage PROM mode; Input for the programming voltage	$V_{SS}$ Connected to $V_{SS}$ Voltage																								
Input capacitance	$V_{pp}/\overline{OE}$ 25pF max All other inputs 12.5pF max	All inputs 12.5pF max																								
Input high voltage of $MP_0, MP_1$	$V_{IH} = V_{SS} - 0.5V$ min	$V_{IH} = 2.0V$ min																								
Bus Timing · Address, R/W, hold timing · Data hold timing	<table border="1"> <thead> <tr> <th></th> <th>HD63701X0</th> <th>HD637A01X0</th> <th>HD637B01X0</th> </tr> </thead> <tbody> <tr> <td><math>t_{AH}</math></td> <td>70</td> <td>45</td> <td>30</td> </tr> <tr> <td><math>t_{HW}</math></td> <td>70</td> <td>50</td> <td>35</td> </tr> </tbody> </table> <p style="text-align: right;">Unit: ns</p>		HD63701X0	HD637A01X0	HD637B01X0	$t_{AH}$	70	45	30	$t_{HW}$	70	50	35	<table border="1"> <thead> <tr> <th></th> <th>HD6301X0</th> <th>HD63A01X0</th> <th>HD63B01X0</th> </tr> </thead> <tbody> <tr> <td><math>t_{AH}</math></td> <td>80</td> <td>50</td> <td>35</td> </tr> <tr> <td><math>t_{HW}</math></td> <td>80</td> <td>50</td> <td>40</td> </tr> </tbody> </table> <p style="text-align: right;">Unit: ns</p>		HD6301X0	HD63A01X0	HD63B01X0	$t_{AH}$	80	50	35	$t_{HW}$	80	50	40
		HD63701X0	HD637A01X0	HD637B01X0																						
$t_{AH}$	70	45	30																							
$t_{HW}$	70	50	35																							
	HD6301X0	HD63A01X0	HD63B01X0																							
$t_{AH}$	80	50	35																							
$t_{HW}$	80	50	40																							
Crystal oscillator characteristics	Internal resistance of crystal oscillator $R_S$	Internal resistance of crystal oscillator $R_S$ $R_S = 60\Omega$ max																								
Storage temperature	$T_{stg} = -55$ to $125^\circ C$	$T_{stg} = -55$ to $150^\circ C$																								
Caution	The HD63701X0 differs from HD6301X0 in chip design and manufacturing process. When applying the HD63701X0 system to HD6301X0, and HD6301X0 system to HD63701X0, note that characteristic values are not exactly the same even if guaranteed values are the same.																									



# Appendix V. Program Development Procedure and Support System

## V.1 Overview

The cross assembler and the hardware emulator using various types of computer are prepared by the company as supporting systems to develop user's programs. User's programs are mask programmed into the ROM and delivered as the LSI by the company.

Figure V-1 shows the typical program design procedure and Table V-1 shows the system development support tool for HD6301X0 and HD6303X which are used in these processes.

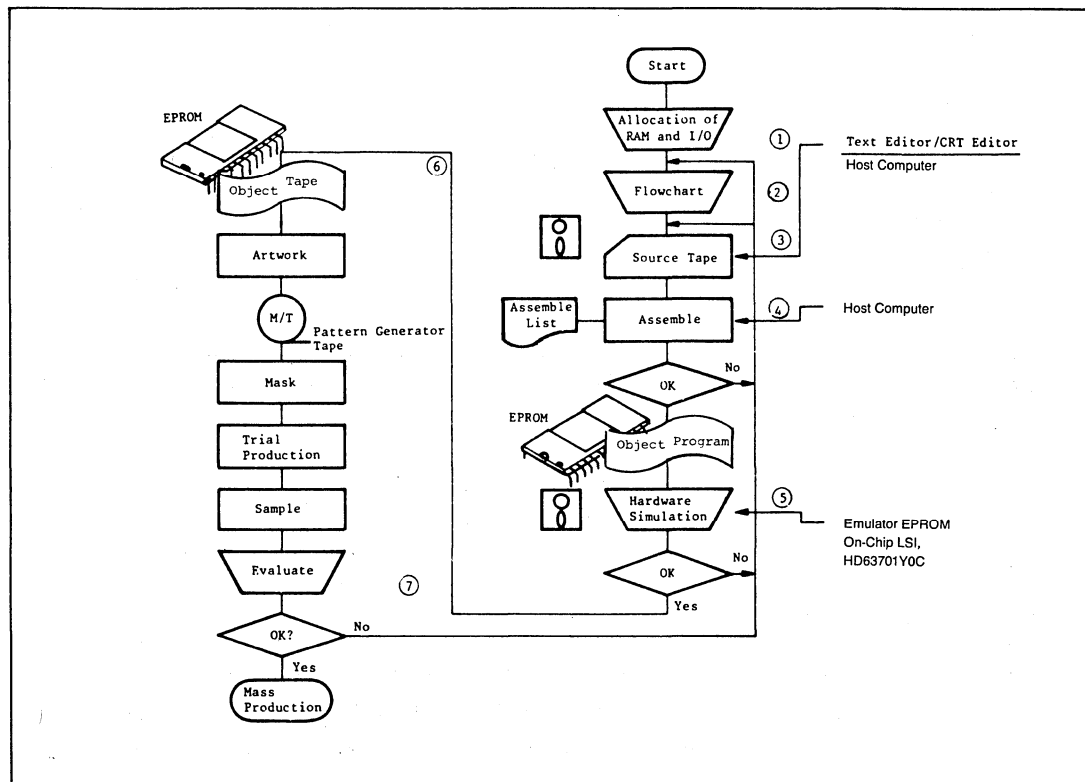


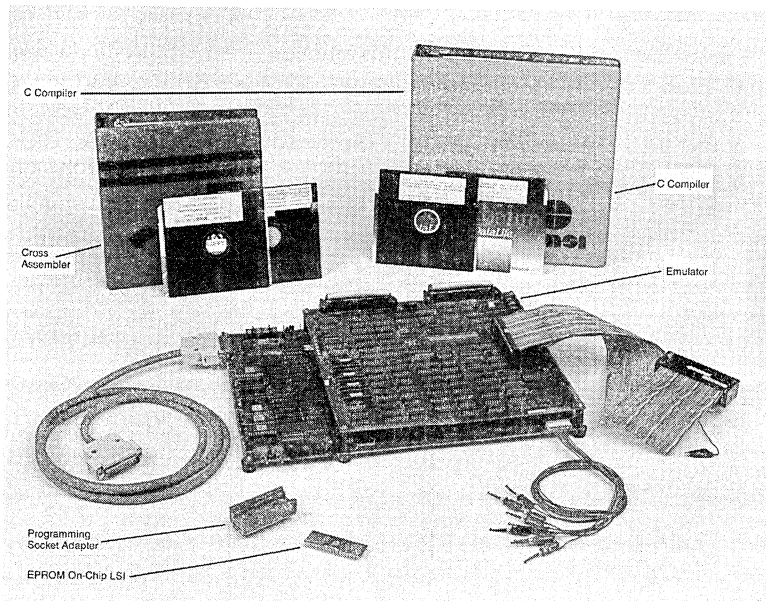
Figure V-1. Program Design Procedure

**(Explanation)**

1. When the user programs the system using the HD6301X0 series, a functional assignment of each I/O pin and an allocation of RAM area should be specified adjusting to designed system before actual programming.
2. A flowchart is designed to implement the functions and it is coded by using the HD6301X0 mnemonic code.
3. Write the software coded according to the flowchart on a floppy disk to make a source program.
4. Assemble the source program to generate an object program using a computer. Assembly errors are also detected.
5. Verify the program through hardware emulation with an emulator or EPROM on-chip type microcomputer.
6. Send the completed program to the company in the form of EPROM. Send Single-chip microcomputer order specification and Mask option list at that time.
7. ROM and mask option are masked by the company. LSI is testatively produced and the sample is handed in to the user. If a user doesn't see any problem in programming, mass production can be started.

Table V-1 Support Tools

Part No.	Emulator	EPROM on-chip LSI	EPROM on-chip LSI Programming Socket Adapter	IBM PC Cross Assembler	IBM PC C Compiler
HD6301X0, HD6303X	H31MIX2 (HS31XEMLO2H)	HD63701X0C	H67PWA01	S31IBMPC	US31PCLI1SF



**HD6301X0 and HD6303X Development Tools**

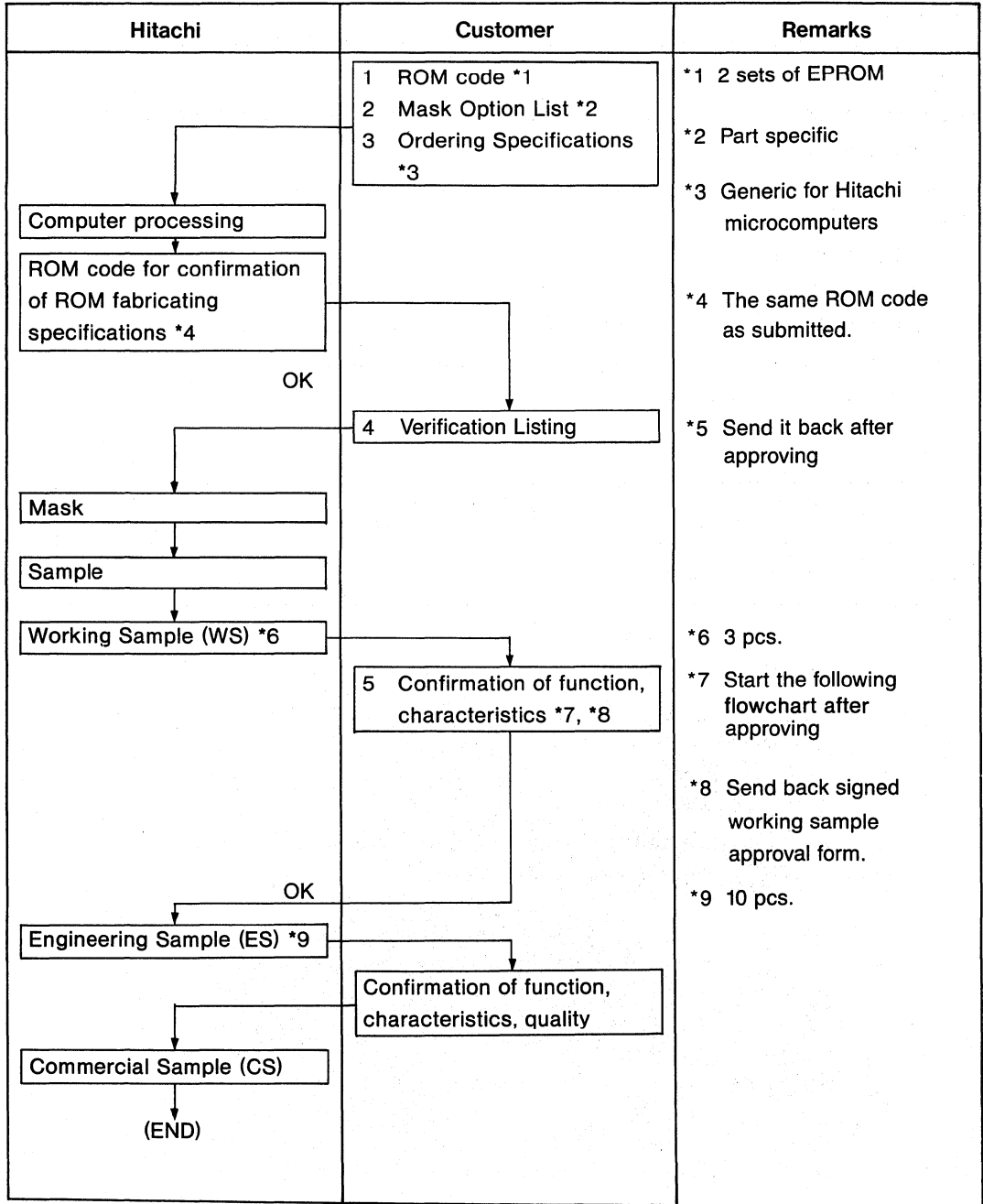


## V.2 Single Chip Microcomputer ROM Ordering Procedure

### V.2.1 Development Flowchart

Single chip microcomputer device is developed according to the following flowchart after program development.

Device Development Flowchart



Note: Please send in 1, 2, and 3 at ROM ordering, and send back 4, 5 after approving.



## V.2.2 Data you send and precautions

(a) Ordering specifications..... Common style for all Hitachi single chip microcomputer devices. Please enter as for the followings. The format is shown in the next page.

- Basic ITEM
- Environment Check List
- Check List of attached data
- Customer

(b) ROM code ..... Please send in the ordering ROM code by 2 sets of EPROM the same contents are written. Enter ROM code No. in them. It is desirable to send in program list for easy confirmation of the program contents.

## V.2.3 Change of ROM code

Note that if you change the ROM code once sended in or other specification, the ROM must be developed from the beginning. The cost of mask charge should be provided again in this case.

## V.2.4 Samples and Mass production

(Working Sample) ..... Sample for confirmation of the contents of ROM code and that of mask option. Normally samples are sent but not guaranteed as for reliability. Please evaluate and approve immediately because the following sample making and mass production are set about after obtaining your evaluation.

(Engineering Sample) ..... Sample for evaluating also reliability. 10 pcs are included in mask charge.

(Commercial Sample) ..... Samples for pre-production which may be purchased separately.

(Mass Product) ..... Products for actual mass production. Please enter the plan of mass production in full.

**HD6301X0**  
**ORDERING SPECIFICATIONS**

**(1) GENERAL CHARACTERISTICS** (Fill in blank space or check appropriate box )

Customer		Package Outline (See page 380.)	<input type="checkbox"/> DP-64S <input type="checkbox"/> FP-80 <input type="checkbox"/> CP-68
Device Type		Options/Remarks:	
Application (be specific)			
Customer ROM Code ID			
ZTAT™ Conversion	<input type="checkbox"/> Yes <input type="checkbox"/> No		
ROM Code Media	<input type="checkbox"/> EPROM      Must Specify: Customer Programmed Start Address _____ <input type="checkbox"/> ZTAT™      Customer Programmed Stop Address _____		
Operating Temperature	<input type="checkbox"/> Standard <input type="checkbox"/> J (-40° C to +85° C) version if offered		
Remask	<input type="checkbox"/> Yes <input type="checkbox"/> No      Previous Hitachi P/N _____		

**(2) OPERATING CHARACTERISTICS** (Fill in blank space or check appropriate box )

LSI Ambient Temperature	Typical	°C	Target Level Of Reliability	<input type="checkbox"/> 1000 Fit	<input type="checkbox"/> (_____)
	Range	°C- °C		<input type="checkbox"/> 500 Fit	
LSI Ambient Humidity	Typical	%	Acceptable Quality Level	Electrical	<input type="checkbox"/> 0.25% <input type="checkbox"/> (_____)
	Range	%- %		Major Visual	<input type="checkbox"/> 0.65% <input type="checkbox"/> (_____)
Power On Duration	Typical	Hours/Day	LSI Operating Speed (Specify MHz or KHz)		
Maximum Applied Voltage To LSI	Power Supply	Max.	Remarks:		
	I/O	Max.			

**(3) ELECTRICAL CHARACTERISTICS** (Fill in blank space or check appropriate box )

<input type="checkbox"/> Purchasing Specifications	<input type="checkbox"/> Hitachi's Standard Specifications
_____	Refer To Data Sheet: _____

**For Hitachi Use Only**

**(4) CUSTOMER APPROVAL**

Customer Name _____
PO# _____
Approved By (print) _____
Approved By (signature) _____
Date _____

**(5) ROM CODE VERIFICATION**

LSI Type No.	
Shipping Date of ROM To Customer	
Approved Date of ROM From Customer	



# HD6301/HD6303 SERIES HANDBOOK

## Section Six

# HD6301Y0/ HD6303Y/HD63701Y0 User's Manual

6

**Section 6**  
**HD6301Y0/HD6303Y/HD63701Y0 User's Manual**  
**Table of Contents**

	<b>Page</b>
1. OVERVIEW .....	519
1.1 Features .....	519
1.2 Block Diagrams .....	520
1.3 Pin Description .....	523
2. INTERNAL ARCHITECTURE AND OPERATION .....	526
2.1 Operation Modes .....	526
2.2 Memory Map .....	527
2.3 Function Pin Description .....	528
2.4 Ports .....	531
2.5 RAM/Port 5 Control Register .....	543
2.6 Port 6 Control/Status Register .....	546
3. CPU FUNCTION .....	548
3.1 CPU Registers .....	548
3.2 Addressing Modes .....	549
3.3 Instruction Set .....	551
3.4 CPU Instruction Flow .....	557
3.5 Low Power Dissipation Modes .....	559
3.6 Trap Function .....	562
3.7 Reset .....	563
3.8 Interrupts .....	564
4. TIMER 1 .....	567
4.1 Free-Running Counter (FRC) .....	568
4.2 Output Compare Registers (OCR) .....	568
4.3 Input Compare Register (ICR) .....	569
4.4 Timer Control/Status Register 1 (TCSR1) .....	569
4.5 Timer Control/Status Register 2 (TCSR2) .....	570
4.6 Timer Status Flags .....	572
4.7 Precautions on Cleaning the OCF .....	572
5. TIMER 2 .....	574
5.1 Timer 2 Upcounter (T2CNT) .....	574
5.2 Timer Constant Register (TCONR) .....	575

5.3	Timer Control/Status Register 3 (TCSR3) . . . . .	575
5.4	Timer Status Flags . . . . .	576
5.5	Precaution for Toggle Pulse Function of HD6301Y0/HD6303Y/HD63701Y0 Timer 2 . . . . .	577
6.	SERIAL COMMUNICATIONS INTERFACE. . . . .	578
6.1	Initialization . . . . .	578
6.2	Asynchronous Mode . . . . .	579
6.3	Clock Synchronous Mode . . . . .	580
6.4	Transmit/Receive Control Status Register (TRCSR). . . . .	582
6.5	Transmit Rate/Mode Control Register (RMCR). . . . .	583
6.6	SCI Receiving Margin . . . . .	586
6.7	SCI Status Flags . . . . .	587
6.8	Precaution for Clock-Synchronous Serial Communication Interface. . . . .	588
7.	HD63701Y0 Programmable ROM (EPROM). . . . .	589
7.1	Programming and Verification. . . . .	590
7.2	Erasing (Window Package) . . . . .	591
7.3	Characteristics and Applications. . . . .	591
8.	APPLICATIONS . . . . .	593
8.1	HD6301Y0 in Expanded Mode . . . . .	593
8.2	HD6301Y0 in Single-Chip Mode . . . . .	594
8.3	Timer Applications. . . . .	594
8.4	SCI Applications . . . . .	598
8.5	Lowering Operating Current . . . . .	600
8.6	Memory Ready Application . . . . .	603
8.7	Halt Application . . . . .	604
8.8	$\overline{RD}$ , $\overline{WR}$ Application. . . . .	605
8.9	LCD-II Interface Application . . . . .	605
8.10	Oscillation Circuit Board Design. . . . .	606
	APPENDIX I. ELECTRICAL CHARACTERISTICS . . . . .	608
	APPENDIX II. INSTRUCTION EXECUTION CYCLES . . . . .	630





APPENDIX III. QUESTIONS AND ANSWERS .....	636
III.1 Parallel Ports .....	636
III.2 Serial Ports .....	639
III.3 Timer/Counter .....	644
III.4 Bus Interface .....	647
III.5 Interrupt Control .....	650
III.6 Oscillation Circuit .....	653
III.7 Reset .....	653
III.8 Low Power Dissipation Mode .....	657
III.9 Software .....	660
III.10 Others .....	661
APPENDIX IV. THE DIFFERENCES BETWEEN HD63701Y0 AND HD6301Y0. . .	663
APPENDIX V. PROGRAM DEVELOPMENT PROCEDURE AND SUPPORT SYSTEM. ....	664
V.1 Overview .....	664
V.2 Single Chip Microcomputer ROM Ordering Procedure .....	666

# Section 1. Overview

The HD6301Y0, HD6303Y, and HD63701Y0 are high-performance CMOS, 8-bit, single-chip microcomputer units (MCUs) which are object-code compatible with the HD6301V.

In addition to the CPU, these MPUs contain 256 bytes of RMA, a 16-bit 4-function timer, an 8-bit reloadable timer, a serial communications interface (SCI), and 53 parallel lines. The HD6301Y0 has 16k bytes of masked ROM. The HD6303Y has no ROM. The HD63701Y0 has 16k bytes of EPROM. The MPU's halt and memory ready functions enable them to release external buses and perform low-speed external memory access.

The HD63701Y0's programmable ROM is programmed by the same method as the standard 27256 EPROM. It is available in ceramic packages. The ceramic package with window is erasable for use in the debugging development stage.

## 1.1 Features

The HD6301Y0, HD6303Y, and HD63701Y0 provide the following features.

- Instruction set compatible with the HD6301V1
- On-board ROM
  - 16k bytes programmable (HD63701Y0)
  - 16k bytes masked (HD6301Y0)
- 256 bytes RAM
- 53 parallel I/O lines
  - 48 common I/O lines (ports 1, 2, 3, 4, 5, 6)
  - 5 output only lines (port 7)
- Darlington transistor direct drive lines (ports 2 and 6)
- 16-bit programmable timer
  - 1 input capture register
  - 1 free-running counter
  - 2 output compare register
- 8-bit reloadable counter
  - External event counter
  - Square-wave generator
  - 2 output compare registers
- Serial communications interface (SCI)
  - Asynchronous mode/clocked synchronous mode
  - 3 transmit formats (asynchronous mode)
  - 6 clock sources
- Memory-ready function for low-speed memory access
- Halt function
- Error detection function (address trap, opcode trap)
- Interrupts
  - 3 external
  - 7 internal
- MCU operation modes
  - Mode 1: expanded mode (internal ROM inhibited)
  - Mode 2: expanded mode (internal ROM valid)
  - Mode 3: single-chip mode
- Address space up to 65k bytes
- Low power modes
  - Sleep mode
  - Standby mode
- Minimum instruction time  $0.33\mu$  ( $f = 3.0\text{MHz}$ )
- Wide operating range
  - $V_{CC} = 3$  to  $5.5\text{V}$  ( $f = 0.1$  to  $0.5\text{MHz}$ )
  - $V_{CC} = 5\text{V} \pm 10\%$  : HD6301Y0 ( $f = 0.1$  to  $1.0\text{MHz}$ )  
HD63A01Y0 ( $f = 0.1$  to  $1.5\text{MHz}$ )  
HD63B01Y0 ( $f = 0.1$  to  $2.0\text{MHz}$ )  
HD63C01Y0 ( $f = 0.1$  to  $3.0\text{MHz}$ )

## 1.2 Block Diagrams

Figures 1-1, 1-2, and 1-3, are block diagrams for HD6301Y0, HD6303Y, and HD63701Y0, respectively.

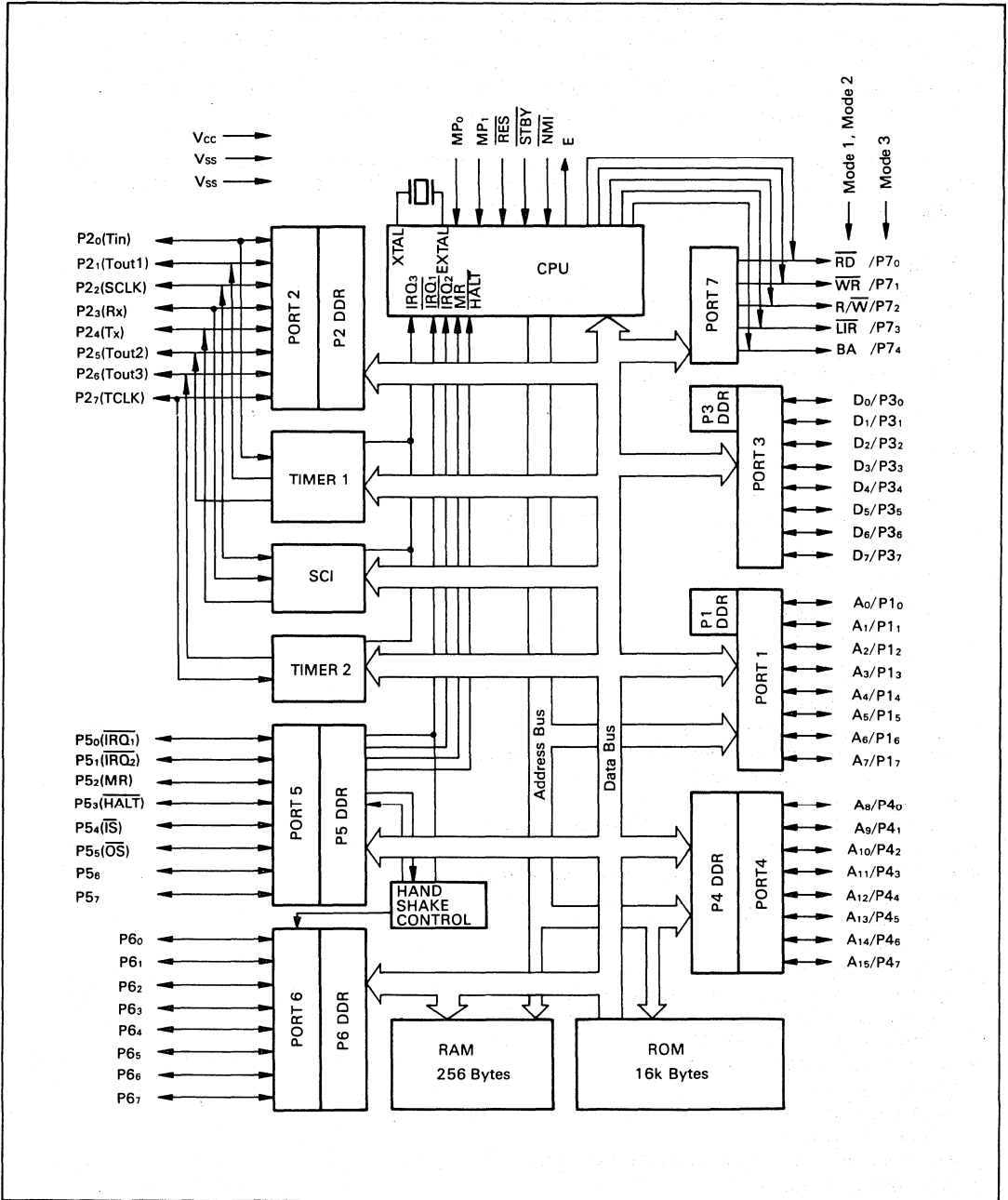


Figure 1-1. HD6301Y0 Block Diagram

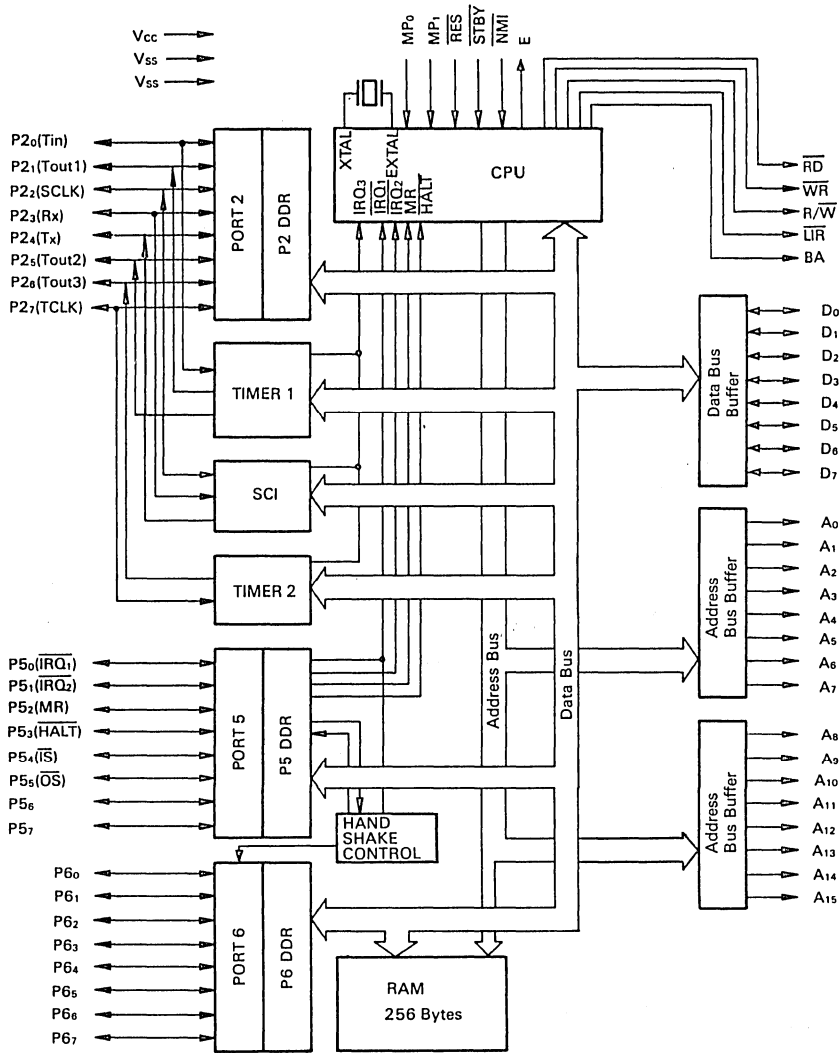


Figure 1-2. HD6303Y Block Diagram

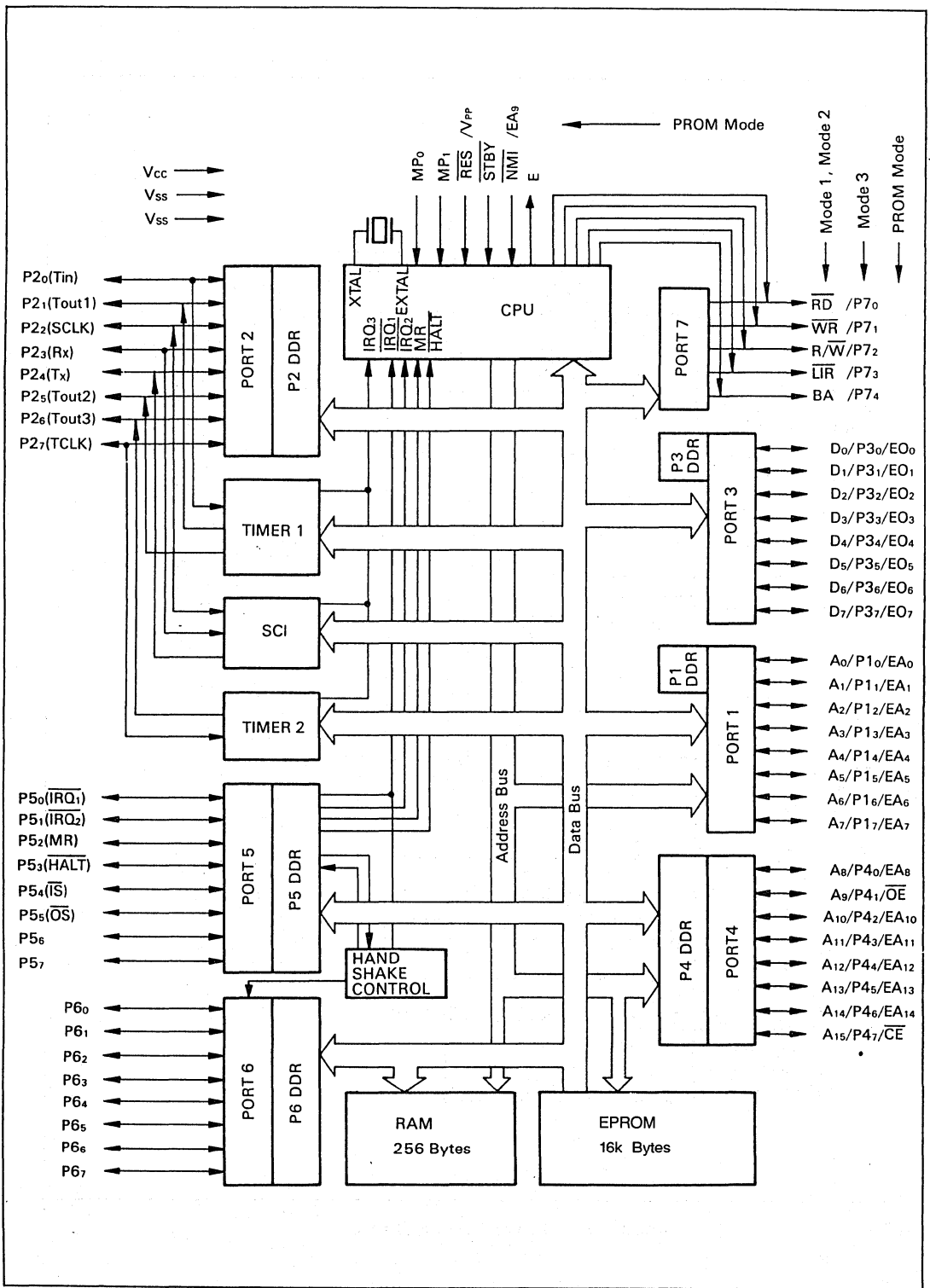


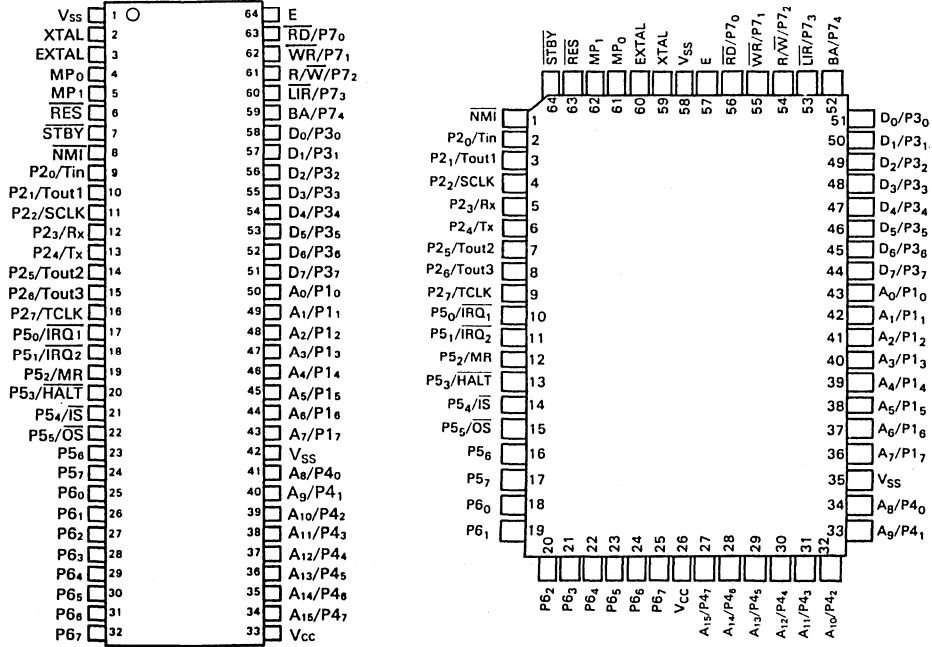
Figure1-3. HD63701Y0 Block Diagram



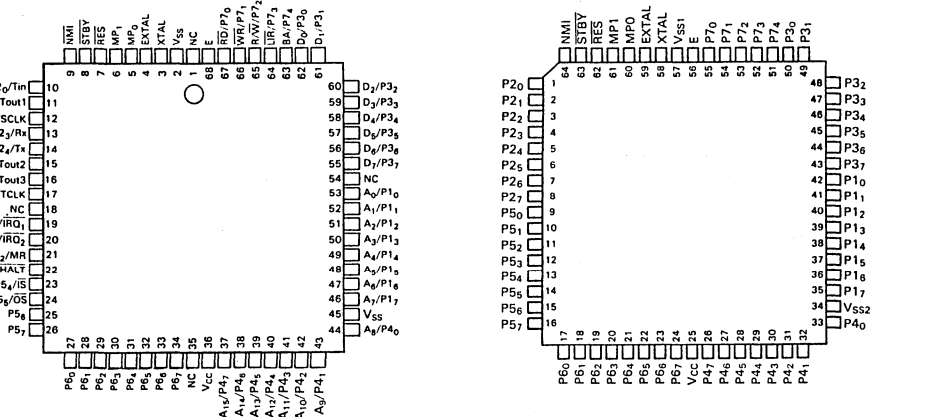
### 1.3 Pin Description

Figure 1-4 shows the pin arrangements for the various packages. Table 1-1 lists pin functions for the HD6301Y0, HD6303Y, and HD63701Y0 in modes 1, 2, and 3. For further pin description, see 2.3 Functional Pin Description, and 2.4 Ports.

- HD6301Y0P, HD63A01Y0P, HD63B01Y0P, HD63C01Y0P (DP-64S)
- HD6303YP, HD63A03YP, HD63B03YP, HD63C03YP (DP-64S)
- HD63701Y0C, HD637A01Y0C, HD637B01Y0C (DC-64S)
- HD6301Y0F, HD63A01Y0F, HD63B01Y0F, HD63C01Y0F (FP-64)
- HD6303YF, HD63A03YF, HD63B03YF, HD63C03YF (FP-64)



- HD6301Y0CP, HD63A01Y0CP, HD63B01Y0CP, HD63C01Y0CP (CP-68)
- HD6303YCP, HD63A03YCP, HD63B03YCP, HD63C03YCP (CP-68)
- HD6301Y0H, HD63A01Y0H, HD63B01Y0H, HD63C01Y0H (FP-64A)
- HD6303YH, HD63A03YH, HD63B03YH, HD63C03YH (FP-64A)



Note: NC; No connection

Figure1-4. Pin Arrangement



Table 1-1. Pin Functions

Pin No.				Mode 1, Mode 2		Mode 3		PROM Mode	
DP-64S	FP-64	FP-64A	CP-68	Pin Name	Function	Pin Name	Function	Pin Name	Function
1	58	57	2	VSS	Ground	VSS	Ground	VSS	Ground
2	59	58	3	XTAL	Crystal connection	XTAL	Crystal connection		
3	60	59	4	EXTAL	Crystal or external clock connection	EXTAL	Crystal or external clock connection		
4	61	60	5	MP <sub>0</sub>	Mode select inputs	MP <sub>0</sub>	Mode select inputs		
5	62	61	6	MP <sub>1</sub>		MP <sub>1</sub>		V <sub>pp</sub>	PROM Programming voltage
6	63	62	7	RES	Reset input	RES	Reset input		
7	64	63	8	STBY	Standby mode input	STBY	Standby mode input	EA <sub>9</sub>	Address bus, bit 9
8	1	64	9	NMI	Nonmaskable interrupt	NMI	Nonmaskable interrupt		
9	2	1	10	P2 <sub>0</sub> /T <sub>in</sub>	Port 2, bit 0/ Timer 1 input	P2 <sub>0</sub> /T <sub>in</sub>	Port 2, bit 0/ Timer 1 input		
10	3	2	11	P2 <sub>1</sub> /T <sub>out1</sub>	Port 2, bit 1/ Timer 1 output 1	P2 <sub>1</sub> /T <sub>out1</sub>	Port 2, bit 1/ Timer 1 output 1		
11	4	3	12	P2 <sub>2</sub> /SCLK	Port 2, bit 2/ SCI clock	P2 <sub>2</sub> /SCLK	Port 2, bit 2/ SCI clock		
12	5	4	13	P2 <sub>3</sub> /R <sub>X</sub>	Port 2, bit 3/ SCI receive input	P2 <sub>3</sub> /R <sub>X</sub>	Port 2, bit 3/ SCI receive input		
13	6	5	14	P2 <sub>4</sub> /T <sub>X</sub>	Port 2, bit 4/ SCI transmit output	P2 <sub>4</sub> /T <sub>X</sub>	Port 2, bit 4/ SCI transmit output		
14	7	6	15	P2 <sub>5</sub> /T <sub>out2</sub>	Port 2, bit 5/ Timer 1 output 2	P2 <sub>5</sub> /T <sub>out2</sub>	Port 2, bit 5/ Timer 1 output 2		
15	8	7	16	P2 <sub>6</sub> /T <sub>out3</sub>	Port 2, bit 6/ Timer 2 output 3	P2 <sub>6</sub> /T <sub>out3</sub>	Port 2, bit 6/ Timer 2 output 3		
16	9	8	17	P2 <sub>7</sub> /TCLK	Port 2, bit 7/ Timer 2 clock	P2 <sub>7</sub> /TCLK	Port 2, bit 7/ Timer 2 clock		
17	10	9	19	P5 <sub>0</sub> /IRQ <sub>1</sub>	Port 5, bit 0/ Interrupt input 1	P5 <sub>0</sub> /IRQ <sub>1</sub>	Port 5, bit 0/ Interrupt input 1		
18	11	10	20	P5 <sub>1</sub> /IRQ <sub>2</sub>	Port 5, bit 1/ Interrupt input 2	P5 <sub>1</sub> /IRQ <sub>2</sub>	Port 5, bit 1/ Interrupt input 2		
19	12	11	21	P5 <sub>2</sub> /MR	Port 5, bit 2/ Memory ready input	P5 <sub>2</sub>	Port 5, bit 2		
20	13	12	22	P5 <sub>3</sub> /HALT	Port 5, bit 3/ Halt input	P5 <sub>3</sub>	Port 5, bit 3		
21	14	13	23	P5 <sub>4</sub> /IS	Port 5, bit 4/ Input strobe	P5 <sub>4</sub> /IS	Port 5, bit 4/ Input strobe		
22	15	14	24	P5 <sub>5</sub> /OS	Port 5, bit 5/ Output strobe	P5 <sub>5</sub> /OS	Port 5, bit 5/ Output strobe		
23	16	15	25	P5 <sub>6</sub>	Port 5, bits 6 and 7	P5 <sub>6</sub>	Port 5, bits 6 and 7		
24	17	16	26	P5 <sub>7</sub>		P5 <sub>7</sub>			

Table 1-2. Relationship of HD6301Y0, HD6303Y, and HD63701Y0 Operating Modes

(continued)

Device Type	Mode			EPROM
	1	2	3	
HD6301Y0	X	X	X	
HD6303Y	X			
HD63701Y0	X	X	X	X

Table 1-1. Pin Functions (continued)

Pin No.				Mode 1, Mode 2		Mode 3		PROM Mode	
DP-64S	FP-64	FP-64A	CP-68	Pin Name	Function	Pin Name	Function	Pin Name	Function
25	18	17	27	P60	Port 6, bits 0-7	P60	Port 6, bits 0-7		
26	19	18	28	P61		P61			
27	20	19	29	P62		P62			
28	21	20	30	P63		P63			
29	22	21	31	P64		P64			
30	23	22	32	P65		P65			
31	24	23	33	P66		P66			
32	25	24	34	P67		P67			
33	26	25	36	V <sub>CC</sub>	+5V power supply	V <sub>CC</sub>	+5V power supply	V <sub>CC</sub>	+5 V power supply
34	27	26	37	A <sub>15</sub>	Address bus, bits 15-8	P47	Port 4, bits 7-0	$\overline{CE}$	Chip enable
35	28	27	38	A <sub>14</sub>		P46		EA <sub>14</sub>	Address bus.
36	29	28	39	A <sub>13</sub>		P45		EA <sub>13</sub>	bits 14-10
37	30	29	40	A <sub>12</sub>		P44		EA <sub>12</sub>	
38	31	30	41	A <sub>11</sub>		P43		EA <sub>11</sub>	
39	32	31	42	A <sub>10</sub>		P42		EA <sub>10</sub>	
40	33	32	43	A <sub>9</sub>		P41		$\overline{OE}$	Output enable
41	34	33	44	A <sub>8</sub>		P40		EA <sub>8</sub>	Address bus.
42	35	34	45	V <sub>SS</sub>	Ground	V <sub>SS</sub>	Ground	V <sub>SS</sub>	Ground
43	36	35	46	A <sub>7</sub>	Address bus, bits 7-0	P17	Port 1, bits 7-0	EA <sub>7</sub>	Address bus, 8 bit bits 7-0
44	37	36	47	A <sub>6</sub>		P16		EA <sub>6</sub>	
45	38	37	48	A <sub>5</sub>		P15		EA <sub>5</sub>	
46	39	38	49	A <sub>4</sub>		P14		EA <sub>4</sub>	
47	40	39	50	A <sub>3</sub>		P13		EA <sub>3</sub>	
48	41	40	51	A <sub>2</sub>		P12		EA <sub>2</sub>	
49	42	41	52	A <sub>1</sub>		P11		EA <sub>1</sub>	
50	43	42	53	A <sub>0</sub>		P10		EA <sub>0</sub>	
51	44	43	55	D <sub>7</sub>	Data bus, bits 7-0	P37	Port 3, bits 7-0	EO <sub>7</sub>	Data bus, bits 7-0
52	45	44	56	D <sub>6</sub>		P36		EO <sub>6</sub>	
53	46	45	57	D <sub>5</sub>		P35		EO <sub>5</sub>	
54	47	46	58	D <sub>4</sub>		P34		EO <sub>4</sub>	
55	48	47	59	D <sub>3</sub>		P33		EO <sub>3</sub>	
56	49	48	60	D <sub>2</sub>		P32		EO <sub>2</sub>	
57	50	49	61	D <sub>1</sub>		P31		EO <sub>1</sub>	
58	51	50	62	D <sub>0</sub>		P30		EO <sub>0</sub>	
59	52	51	63	BA	Bus available output	P74	Port 7, bits 4-0		
60	53	52	64	$\overline{LIR}$	Load instruction register output	P73			
61	54	53	65	R/W	Read/Write output	P72			
62	55	54	66	$\overline{WR}$	Write output	P71			
63	56	55	67	$\overline{RD}$	Read output	P70			
64	57	56	68	E	External clock output	E		External clock output	



# Section 2. Internal Architecture and Operation

## 2.1 Operation Modes

The HD6301Y0 and HD63701Y0 operate in three MCU modes. The HD6303Y only operates in MCU mode 1. The mode program pins  $MP_0$  and  $MP_1$ , and the  $\overline{STBY}$  pin select the mode (table 2-1).

- MCU 1 (expanded): external memory access enabled, internal ROM disabled
- MCU 2 (expanded): external memory access enabled, internal ROM enabled
- MCU 3 (single-chip): external memory access disabled

Table 2-1. Mode Selection

$MP_1$	$MP_0$	$\overline{STBY}$	ROM	RAM	Interrupt Vector	Operation Mode
Low	High	X	External	Internal	External	MCU 1 (expanded)
High	Low	X	Internal	Internal	Internal	MCU 2 (expanded)
High	High	X	Internal	Internal	Internal	MCU 3 (single-chip)

Note: X = Don't care

### 2.1.1 MCU Mode 1 (Expanded)

In MCU mode 1, port 3 is the data bus, port 1 is the lower address bus, and port 4 is the upper address bus. They can directly interface with HD6800 buses. Port 7 supplies signals such as  $R/\overline{W}$ . See table 2-2. In mode 1, the ROM is disabled and the external address space is 65k bytes (figure 2-1). Since the HD6303Y has no internal ROM, it only operates in mode 1.

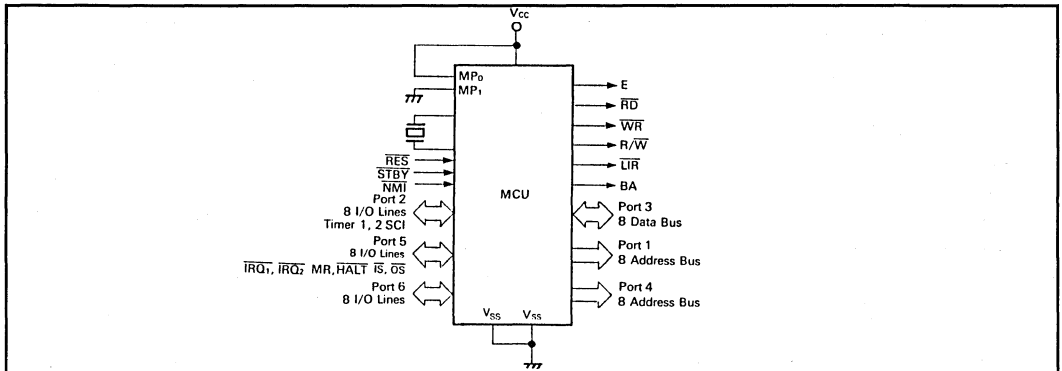


Figure 2-1. MCU Mode 1

### 2.1.2 MCU Mode 2 (Expanded)

MCU mode 2 is the same as mode 1, except that the ROM is enabled. The external address space is 48k bytes (figure 2-2).

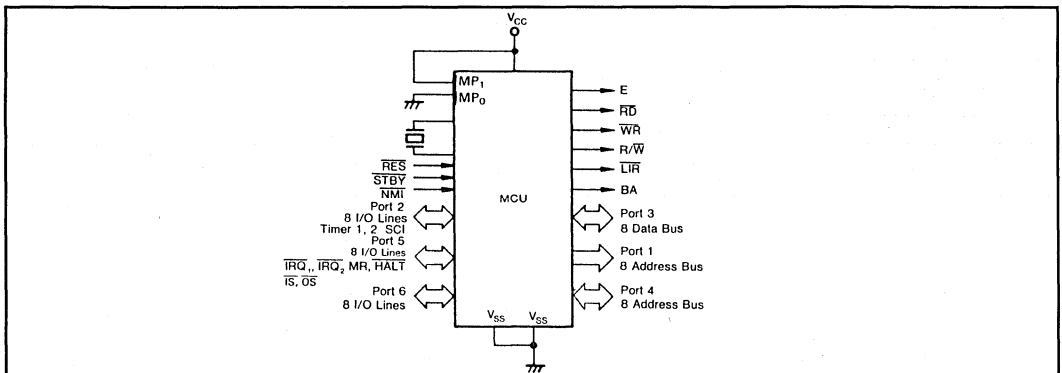


Figure 2-2. MCU Mode 2

### 2.1.3 MCU Mode 3 (Single-Chip)

In MCU mode 3, all ports are I/O ports. There is no interface to external buses (figure 2-3).

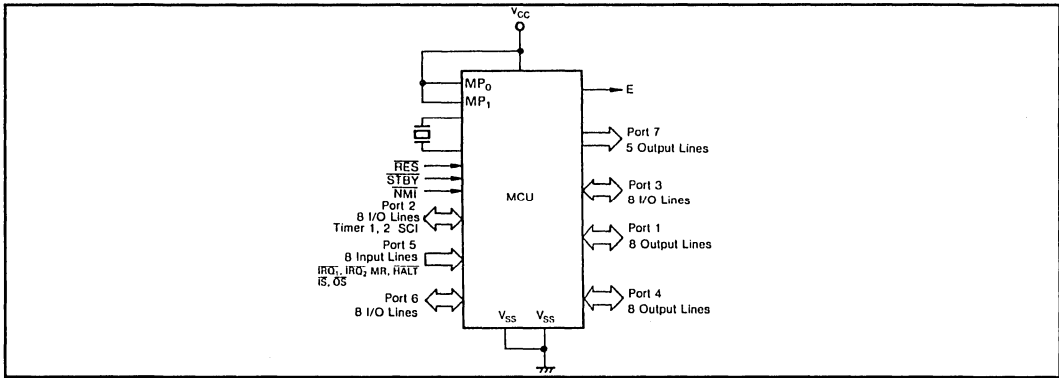


Figure 2-3. MCU Mode 3

Table 2-2. Port Signals

Port	MCU Mode 1	MCU Mode 2	MCU Mode 3
1	Address bus ( $A_0$ - $A_7$ )	Address bus ( $A_0$ - $A_7$ )	I/O port
2	I/O port	I/O port	I/O port
3	Data bus ( $D_7$ - $D_0$ )	Data bus ( $D_7$ - $D_0$ )	I/O port
4	Address bus ( $A_8$ - $A_{15}$ )	Address bus ( $A_8$ - $A_{15}$ )	I/O port
5	I/O port	I/O port	I/O port
6	I/O port	I/O port	I/O port
7	$\overline{RD}$ , $\overline{WR}$ , $R/\overline{W}$ , $\overline{LIR}$ , BA	$\overline{RD}$ , $\overline{WR}$ , $R/\overline{W}$ , $\overline{LIR}$ , BA	Output port

## 2.2 Memory Map

The HD6301Y0, HD6303Y, and HD63701Y0 can access up to 65k bytes of external memory, depending on the operating mode. Figure 2-4 shows a memory map for each mode. The first 40 locations of each map, from \$00 to \$27, are reserved for the MCU's internal register area (table 2-3).

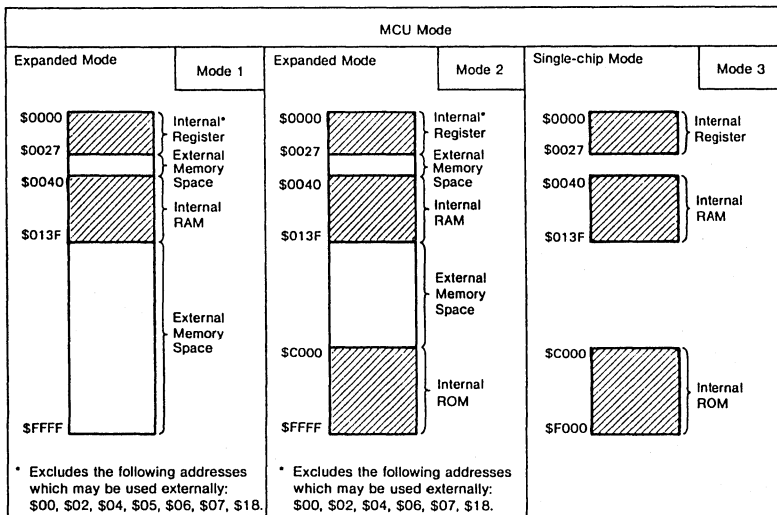


Figure 2-4. Memory Maps

**Table 2-3. Internal Register Area**

Address	Register	R/W	State at RESET
00	Port 1 data direction register	W	\$FE
01	Port 2 data direction register	W	\$00
02	Port 1	R/W	Undefined
03	Port 2	R/W	Undefined
04	Port 3 data direction register	W	\$FE
05	Port 4 data direction register	W	\$00
06	Port 3	R/W	Undefined
07	Port 4	R/W	Undefined
08	Timing control/status register 1	R/W	\$00
09	Free-running counter (upper byte)	R/W	\$00
0A	Free-running counter (lower byte)	R/W	\$00
0B	Output compare register 1 (upper byte)	R/W	\$FF
0C	Output compare register 1 (lower byte)	R/W	\$FF
0D	Input capture register (upper byte)	R	\$00
0E	Input capture register (lower byte)	R	\$00
0F	Timer control/status register 2	R/W	\$10
10	Rate, mode control register	R/W	\$C0
11	Tx/Rx control status register 1	R/W	\$20
12	Receive data register	R	\$00
13	Transmit data register	W	\$00
14	RAM/port 5 control register	R/W	\$F8 or \$78
15	Port 5	R	Undefined
16	Port 6 data direction register	W	\$00
17	Port 6	R/W	Undefined
18	Port 7	R/W	Undefined
19	Output capture register 2 (upper byte)	R/W	\$FF
1A	Output capture register 2 (lower byte)	R/W	\$FF
1B	Timer control/status register 3	R/W	\$20
1C	Timer constant register	W	\$FF
1D	Timer 2 upcounter	R/W	\$00
1E	Tx/Rx control status register 2	R/W	\$28
1F	Test register		
20	Port 5 data direction register	W	\$00
21	Port 6 control/status register	R/W	\$07

## 2.3 Functional Pin Description

### 2.3.1 Power ( $V_{CC}$ , $V_{SS}$ )

$V_{CC}$ ,  $V_{SS}$  are the power supply pins. Apply  $+5V \pm 10\%$  to  $V_{CC}$ . Tie  $V_{SS}$  to ground.

### 2.3.2 Clock (XTAL, EXTAL)

XTAL and EXTAL connect to an AT-cut parallel resonant crystal. The chip has a divide-by-four circuit. For example, if a 4 MHz crystal is used, the system clock will be 1 MHz.

Figure 2-5 is an example of the crystal oscillator connection. The crystal and  $C_{L1}$  and  $C_{L2}$  should be located as close as possible to the XTAL and EXTAL pins. No line must cross the lines between the crystal oscillator and the XTAL and EXTAL pins.

The EXTAL pin can be driven by an external clock with a 45% to 55% duty cycle. The LSI divides the external clock frequency by four. The external clock should therefore be less than four times the maximum clock frequency. When using an external clock, the XTAL pin should be left open.

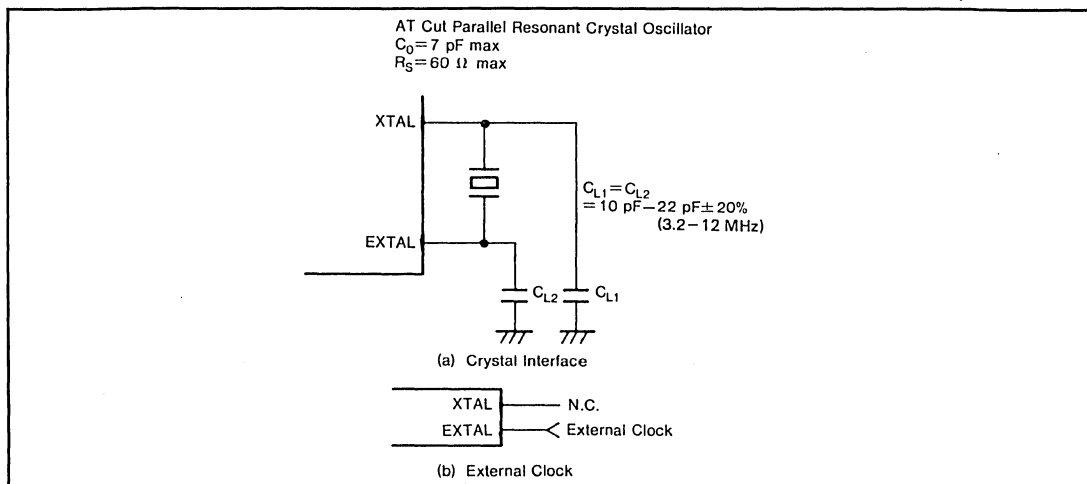


Figure 2-5. Recommended Crystal Oscillator Connection

### 2.3.3 Standby ( $\overline{STBY}$ )

The  $\overline{STBY}$  pin puts the MCU in standby mode. When  $\overline{STBY}$  is low, the oscillation stops, and the internal clock is stabilized to put the MCU in a reset condition. To retain the contents of RAM during standby, write 0 to the RAM enable bit (RAME). RAME is bit 6 of the RAM/port 5 control register at address \$0014. RAM is disabled, and its contents are sustained. Refer to 3.5 Low Power Dissipation Mode for details on the standby mode.

### 2.3.4 Reset ( $\overline{RES}$ )

This pin resets the MCU's internal state and provides a startup procedure. The  $\overline{RES}$  input must be held low for at least 20 ms during power-on.

The CPU registers accumulator, index register, stack pointer, condition code register except for mask bit, RAM, and the data registers of the ports are not initialized during reset, so their contents are undefined.

### 2.3.5 External Clock (E)

E provides a TTL-compatible system clock to external circuits. Its frequency is one-fourth that of the crystal oscillator or external clock. E can drive one TTL load and 90 pF.

### 2.3.6 Nonmaskable Interrupt ( $\overline{NMI}$ )

When CPU detects a falling edge at the  $\overline{NMI}$  input, it begins the internal nonmaskable interrupt sequence. The instruction being executed when the  $\overline{NMI}$  is detected will proceed to completion. The interrupt mask bit of the condition code register does not affect the nonmaskable interrupt.

In response to an  $\overline{NMI}$  interrupt, the contents of the program counter, index register, accumulators, and condition code register will be saved onto the stack. After they are saved, a vector is fetched from \$FFFC and \$FFFD to the program counter, and the nonmaskable interrupt service routine starts.

Note: After reset, the stack pointer should be initialized to an appropriate memory location before any  $\overline{NMI}$  input.

### 2.3.7 Interrupt Requests ( $\overline{IRQ_1}$ , $\overline{IRQ_2}$ )

The interrupt requests are level-sensitive inputs which request an internal interrupt sequence from the CPU.

### 2.3.8 Mode Program (MP<sub>0</sub>, MP<sub>1</sub>)

These pins determine the operation mode. Refer to 2.1 Operation Mode for details.

Note: The following signals  $\overline{RD}$ ,  $\overline{WR}$ ,  $R/\overline{W}$ ,  $\overline{LIR}$ , MR,  $\overline{HALT}$ , and BA, are only used in modes 1 and 2.

### 2.3.9 Read/Write (R/ $\overline{W}$ ; P7<sub>2</sub>)

The read/write signal shows whether the MCU is in read (R/ $\overline{W}$  high) or write (R/ $\overline{W}$  low) state to the peripheral or memory devices. It is usually high, in read state. R/ $\overline{W}$  can drive one TTL load and 30 pF.

### 2.3.10 Read and Write ( $\overline{RD}$ ; P7<sub>0</sub>, $\overline{WR}$ ; P7<sub>1</sub>)

The read and write outputs show active low outputs to peripherals or memories when the CPU is reading or writing. This enables the CPU to access LSI peripherals with  $\overline{RD}$  and  $\overline{WR}$  inputs easily. These pins can drive one TTL load and 30 pF.

### 2.3.11 Load Instruction Register ( $\overline{LIR}$ ; P7<sub>3</sub>)

The  $\overline{LIR}$  output low shows that the instruction opcode is on the data bus.  $\overline{LIR}$  can drive one TTL load and 30 pF.

### 2.3.12 Memory Ready (MR; P5<sub>2</sub>)

The memory ready control input lengthens the system clock's high period to allow access to low-speed memory. When MR is high, the system clock operates normally. But when MR is low, the high period will be lengthened depending on its low time in integral multiples of its cycle time. It can be lengthened up to 9  $\mu$ s.

During internal address or invalid memory access, MR is prohibited internally from decreasing operation speed. Even in the halt state, MR can lengthen the high period of the system clock to allow peripheral devices to access low-speed memories. MR is also used as P5<sub>2</sub>. The function is chosen by the enable bit in the RAM/port 5 control register (bit 2) at \$0014. See 2.5 RAM/Port 5 Control Register for details.

### 2.3.13 Halt ( $\overline{HALT}$ ; P5<sub>3</sub>)

The halt control input stops instruction execution and releases the buses. When  $\overline{HALT}$  switches low, the CPU finishes the current instruction, then stops and enters the halt state. When entering the halt state, the CPU sets BA (P7<sub>4</sub>) high, and sets the address bus, data bus,  $\overline{RD}$ ,  $\overline{WR}$ , and R/ $\overline{W}$  to high impedance. When an interrupt occurs in the halt state, the CPU cancels the halt, and executes the interrupt service routine.

Note: When the CPU is in the interrupt wait state, executing the WAI instruction,  $\overline{HALT}$  should be held high. If halt turns low, the CPU may fetch the incorrect vector after releasing the halt state (figure 2-6). If a halt is expected, a loop should be used instead of WAI (figure 2-7).

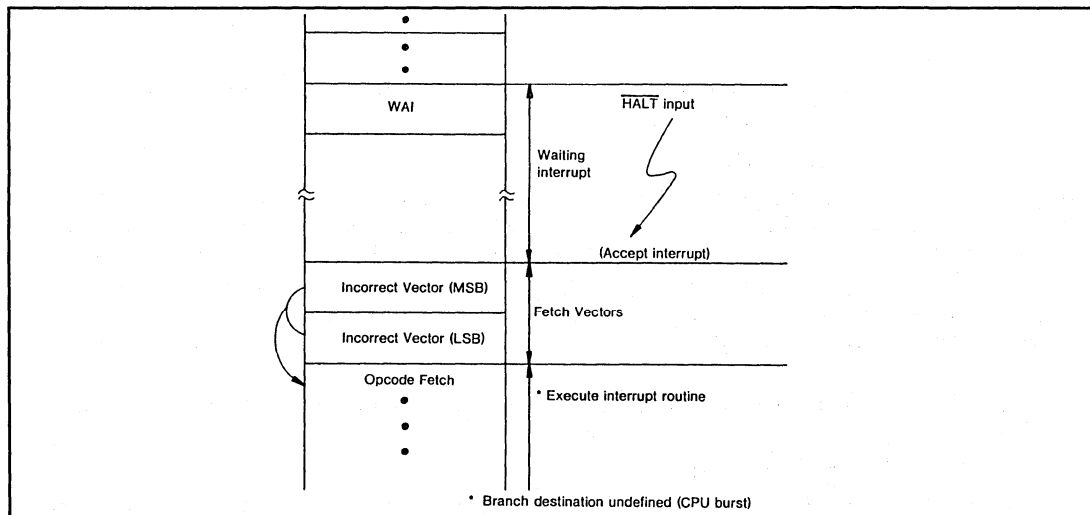


Figure 2-6  $\overline{HALT}$  WAI



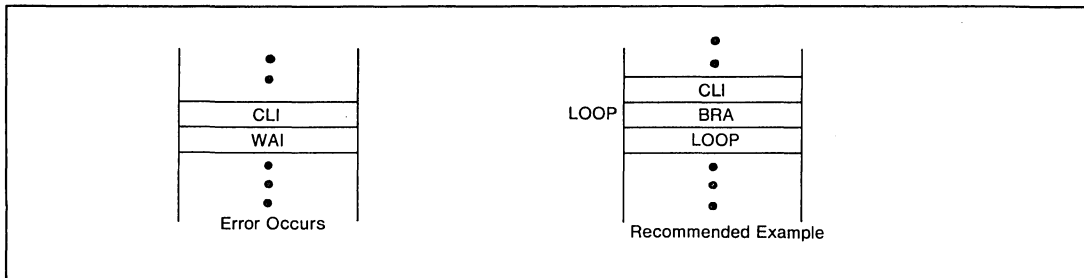


Figure 2-7. Branch Replacement for WAI

### 2.3.14 Bus Available (BA; P7<sub>4</sub>)

The bus available output control signal goes high when the CPU accepts  $\overline{\text{HALT}}$  and releases the buses. It is normally low. The HD6800 and HD6802 bring BA high and release the buses at WAI execution, but the HD6301Y0, HD6303Y, and HD63701Y0, don't. But if  $\overline{\text{HALT}}$  goes low when the CPU is in the interrupt wait state after having executed a WAI, the CPU sets BA high and releases the buses. When  $\overline{\text{HALT}}$  goes high, the CPU returns to the interrupt wait state.

## 2.4 Ports

The HD6301Y0, HD6303Y, and HD63701Y0 provides seven ports (six 8-bit ports and a 5-bit port). Some pins have other uses, as shown in table 2-2. Table 2-4 shows the addresses of the ports and their data direction registers. Figure 2-9 shows block diagrams of each port. Table 2-5 shows the state of each port at reset.

Table 2-4. Port and Data Direction Register Address

Port	Port Address	Data Direction Register
1	\$0002	\$0000
2	\$0003	\$0001
3	\$0006	\$0004
4	\$0007	\$0005
5	\$0015	\$0020
6	\$0017	\$0016
7	\$0018	

Table 2-5. Port at Reset (Modes 1 and 2)

Port	State at Reset
1 (A <sub>0</sub> -A <sub>7</sub> )	High
2	High impedance
3 (D <sub>0</sub> -D <sub>7</sub> )	High impedance
4 (A <sub>8</sub> -A <sub>15</sub> )	High (Mode 1 only)
5	High impedance
6	High impedance
7	$\overline{RD}$ , $\overline{WR}$ , $\overline{R/W}$ , $\overline{LIR}$ = High BA = Low

Note: Port 4 is high impedance in Mode 2.  
All ports are high impedance after reset in Mode 3.

### 2.4.1 Port 1

Port 1 is an 8-bit I/O port (figure 2-8). The LSB of the DDR (\$0000) selects the data direction of the whole port (figure 2-9). In the expanded modes (1 and 2) port 1 is the lower address bus (A<sub>7</sub>-A<sub>0</sub>). Port 1 can drive one TTL load and 90 pF capacitance.

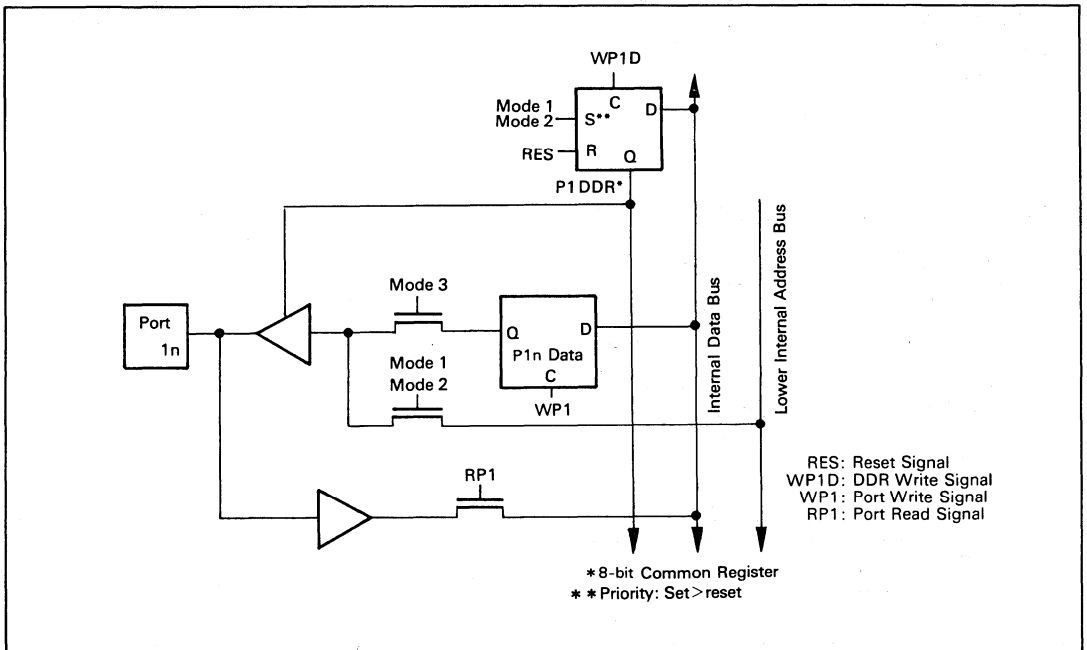


Figure 2-8. Port 1 Block Diagram



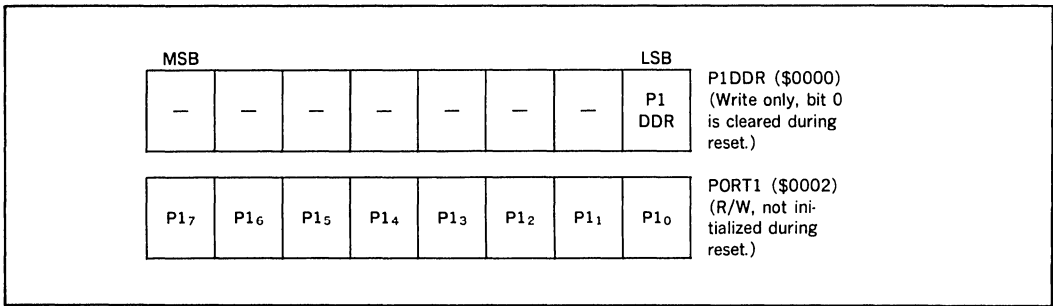


Figure 2-9. Port 1 Register and Data Direction Register

### 2.4.2 Port 2

Port 2 is an 8-bit I/O port (figure 2-10). Each bit of the DDR (\$0001) defines the data direction of the corresponding bit of port 2 (figure 2-11). Port 2 can drive one TTL load and 30 pF capacitance. It can produce 1 mA when  $V_{OUT} = 1.5$  V to directly drive the base of a Darlington transistor.

Port 2 pins are also used as I/O pins by timers 1, 2, and the SCI (table 2-6). The pin functions are controlled by registers in timers 1, 2, and the SCI.



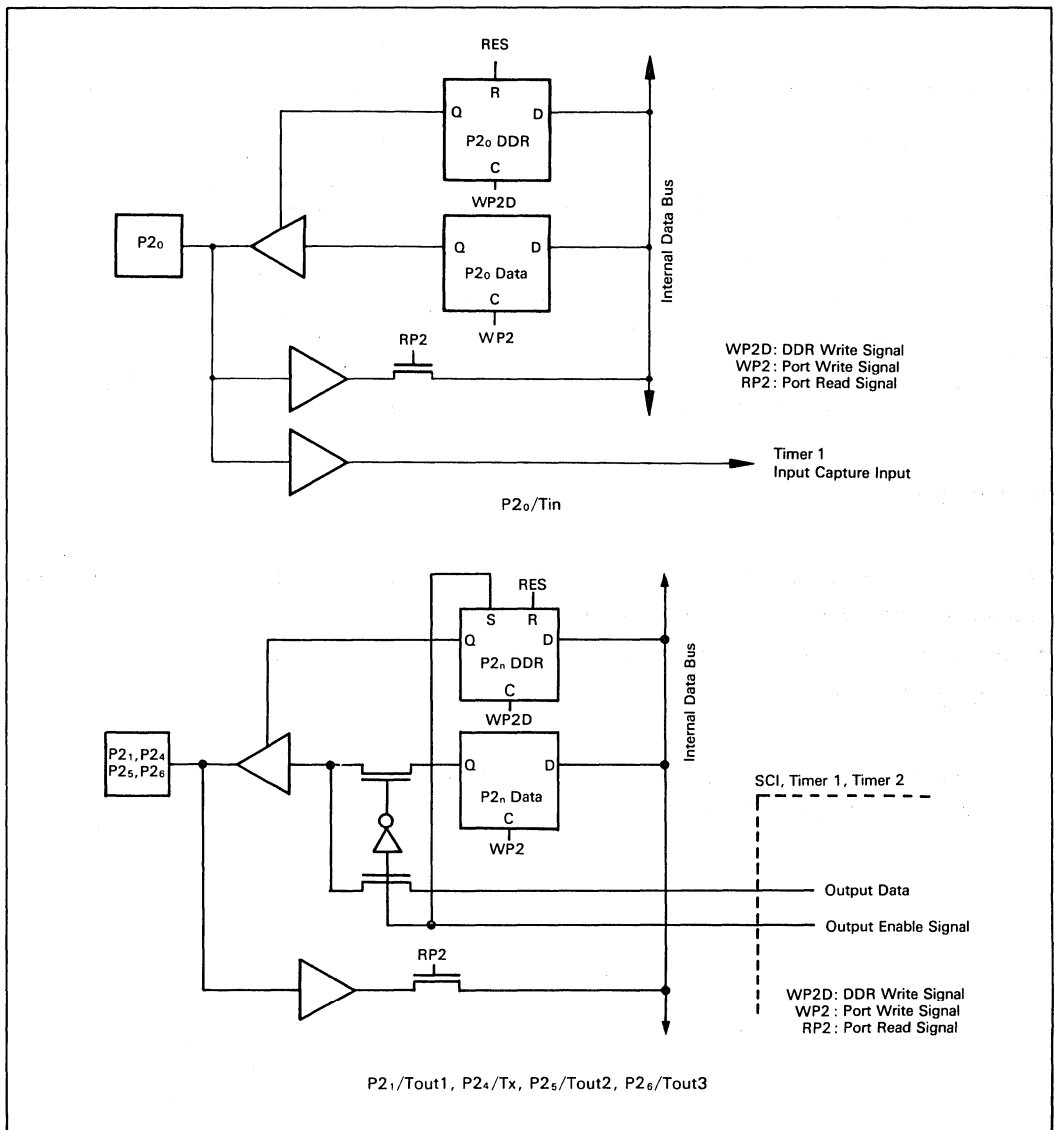


Figure 2-10. Port 2 Block Diagram

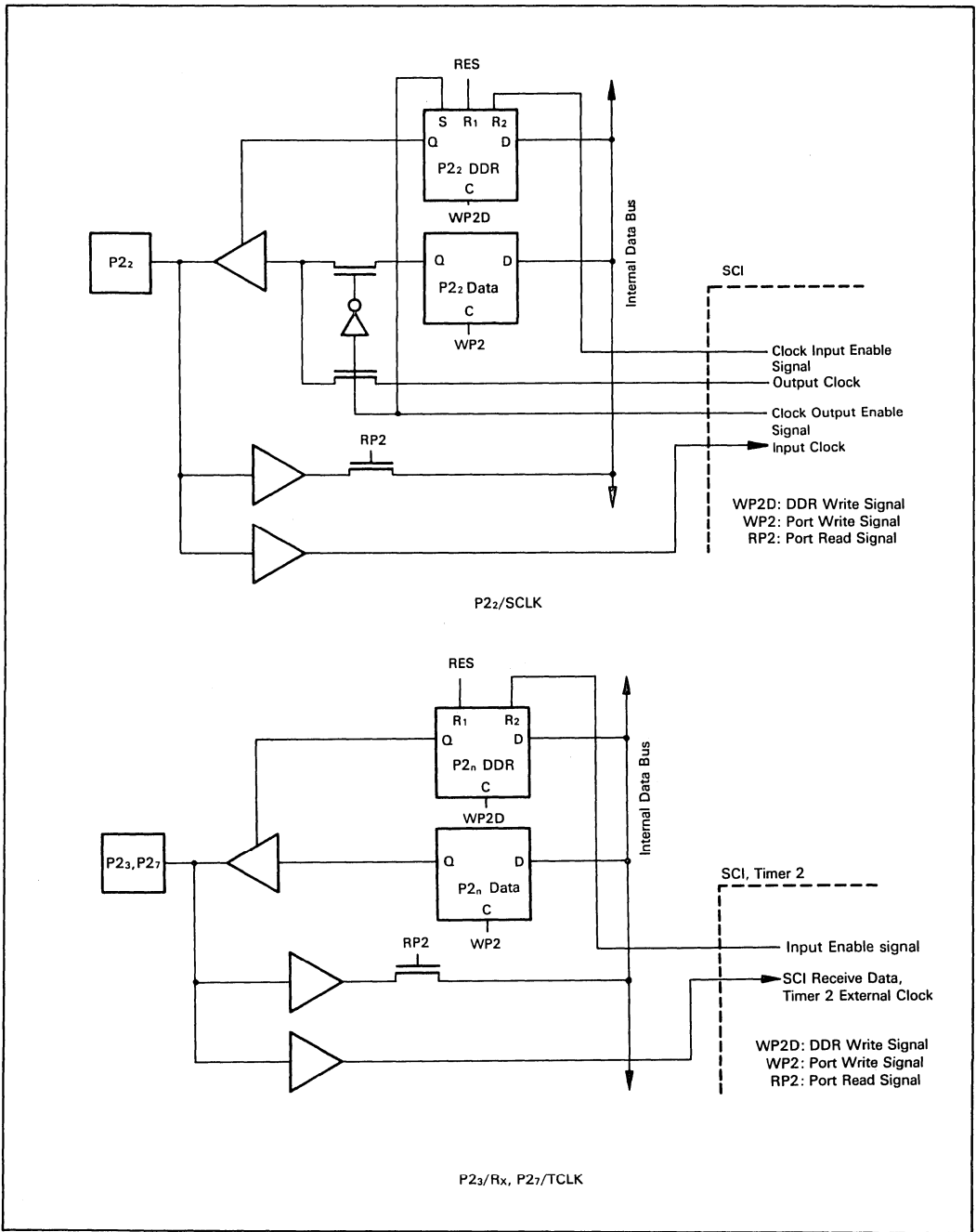


Figure 2-10. Port 2 Block Diagram (Cont)

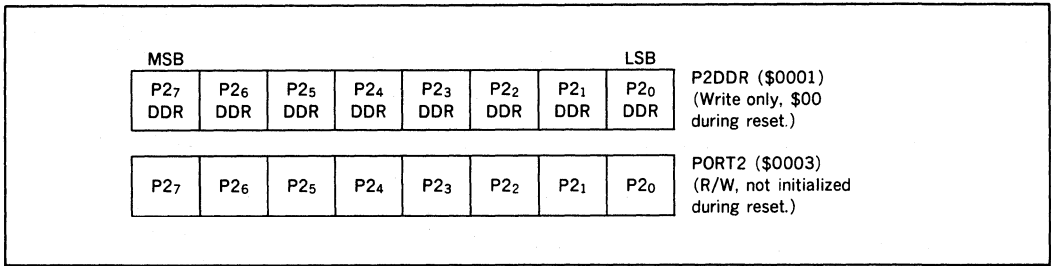


Figure 2-11. Port 2 Register and Data Direction Register

Table 2-6. Port 2 Pin Functions

Port 2 Pin	Alternate Function	Description
P20	Tin	Timer 1 input
P21	Tout1	Timer 1 output 1
P22	SCLK	SCI clock
P23	R <sub>x</sub>	SCI receive input
P24	T <sub>x</sub>	SCI transmit output
P25	Tout2	Timer 1 output 2
P26	Tout3	Timer 2 output 3
P27	TCLK	Timer 2 clock



## 2.4.4 Port 4

Port 4 is an 8-bit I/O port (figure 2-14). Each bit of the DDR (\$0005) defines the data direction of the corresponding bit of port 4 (figure 2-15). In the expanded modes (1 and 2), port 4 is the upper address bus (A<sub>15</sub>-A<sub>8</sub>). In mode 1 (expanded mode with no external ROM), the DDR is set automatically and port 4 outputs addresses. In mode 2 (expanded mode with external ROM), the DDR must be set to 1 for port 4 to function as the address bus. Pins that are not needed for the address bus can be used as input pins. Port 4 can drive one TTL load and 90 pF capacitance.

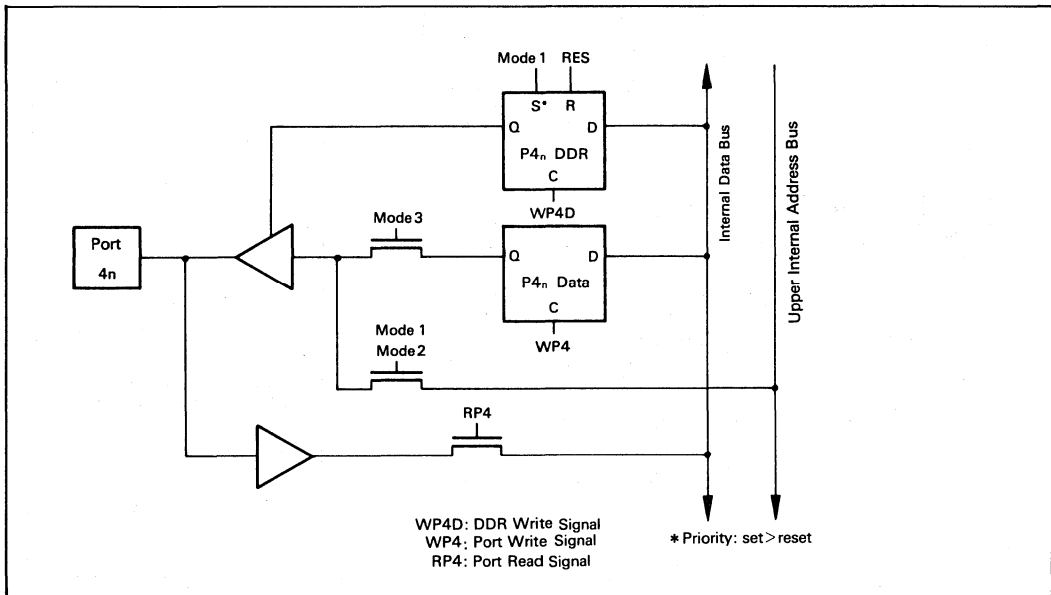


Figure 2-14. Port 4 Block Diagram

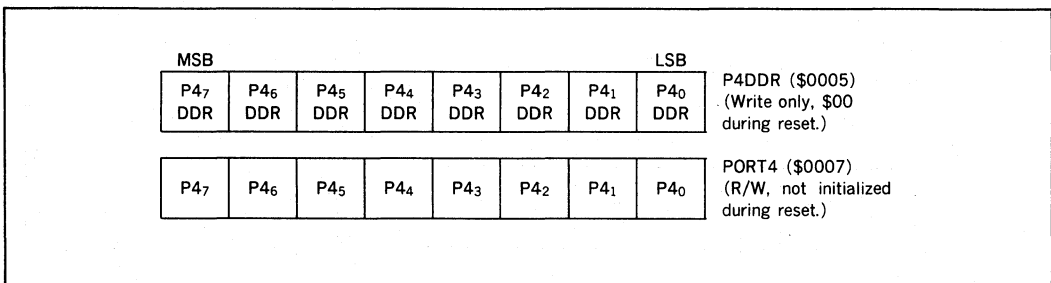


Figure 2-15. Port 4 Register and Data Direction Register



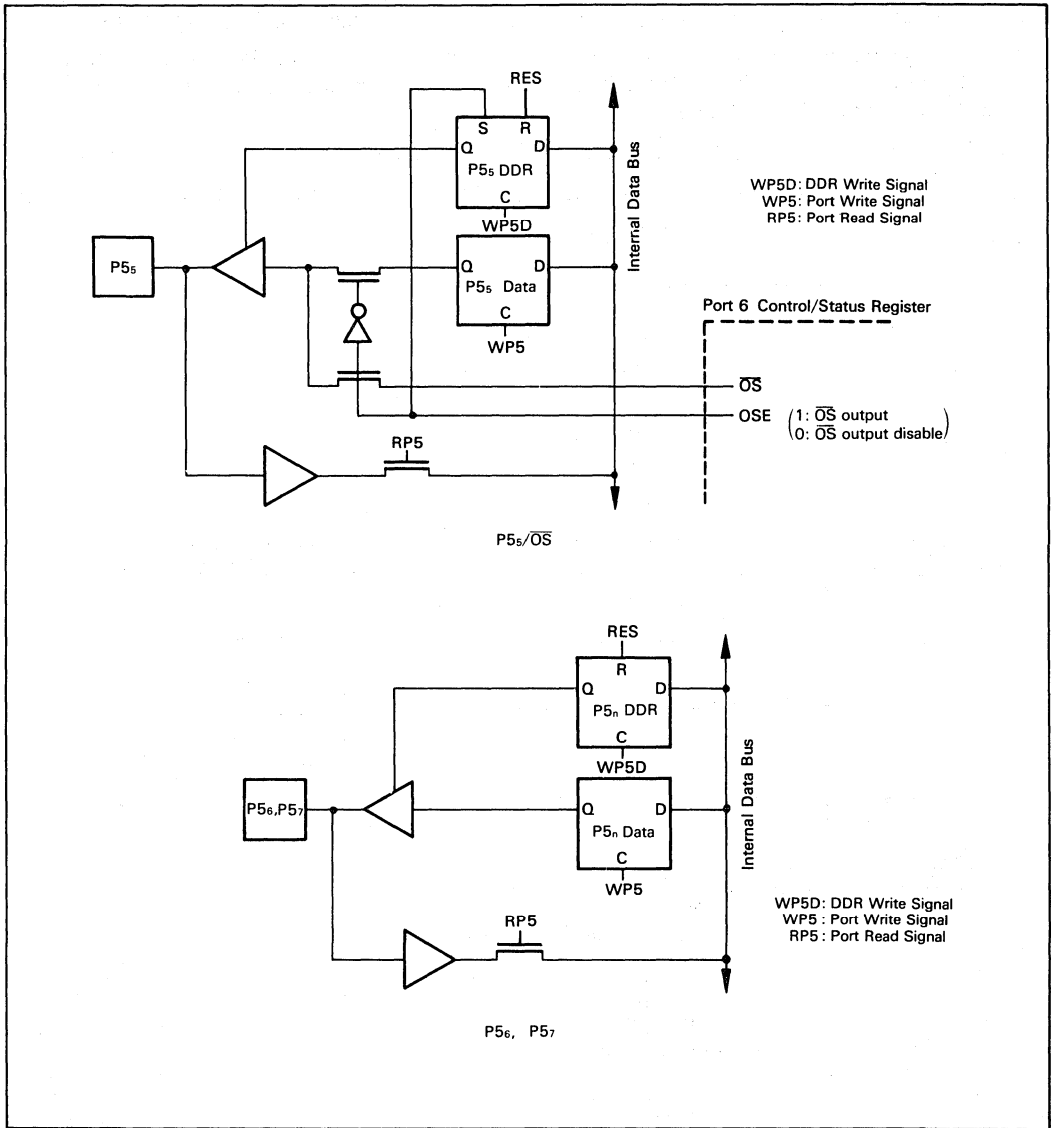


Figure 2-16. Port 5 Block Diagram (Cont)

## 2.4.6 Port 6

Port 6 is an 8-bit I/O port (figure 2-17). Each bit of the DDR (\$0016) defines the data direction of the corresponding bit of port 6 (figure 2-18). Port 5 can drive one TTL load and 30 pF capacitance. In addition, it can drive the base of Darlington transistors directly.

Port 6 can function as a parallel handshake interface under the control of the port 6 control/status register (P6CSR: \$0021). Port 6 has a data latch for input data (IS LATCH).

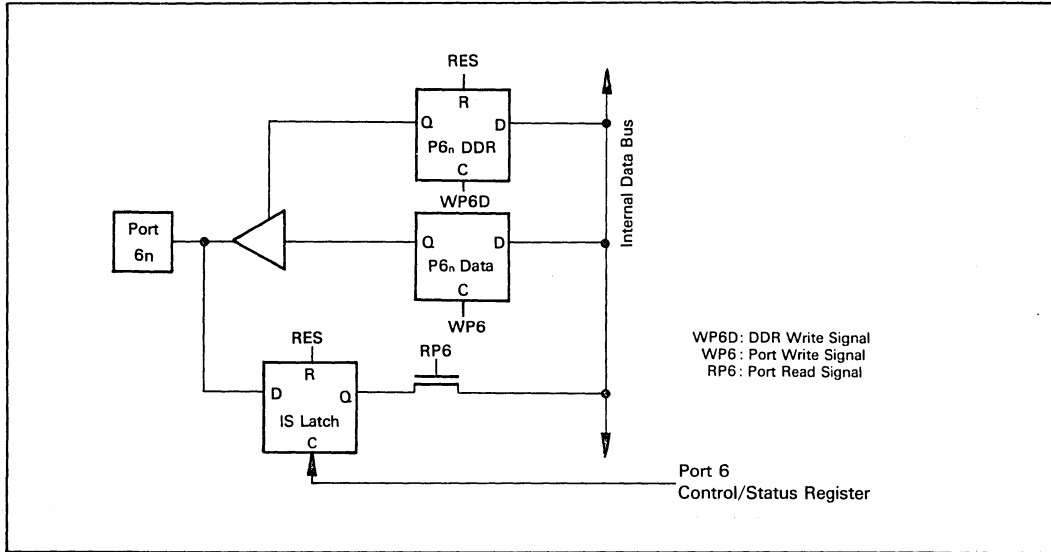


Figure 2-17. Port 6 Block Diagram

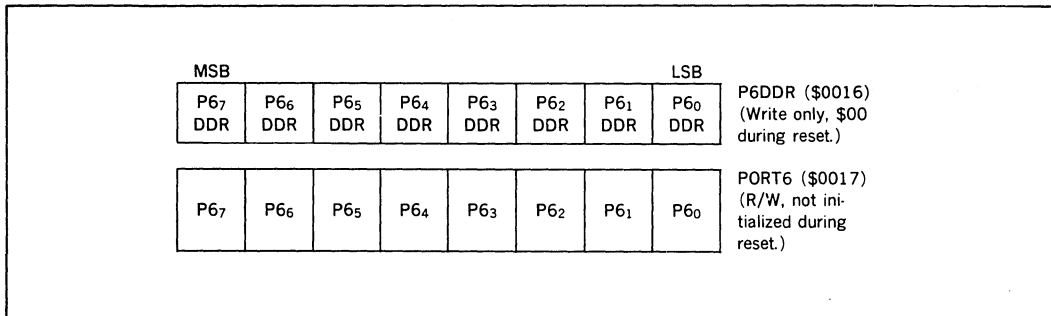


Figure 2-18. Port 6 Register and Data Direction Register



## 2.4.7 Port 7

Port 7 is a 5-bit output only port (figures 2-19, 2-20). In the expanded modes (1 and 2), port 7 outputs control signals from the CPU. Port 7 goes to high-impedance state during reset. Port 7 can drive one TTL load and 30 pF capacitance.

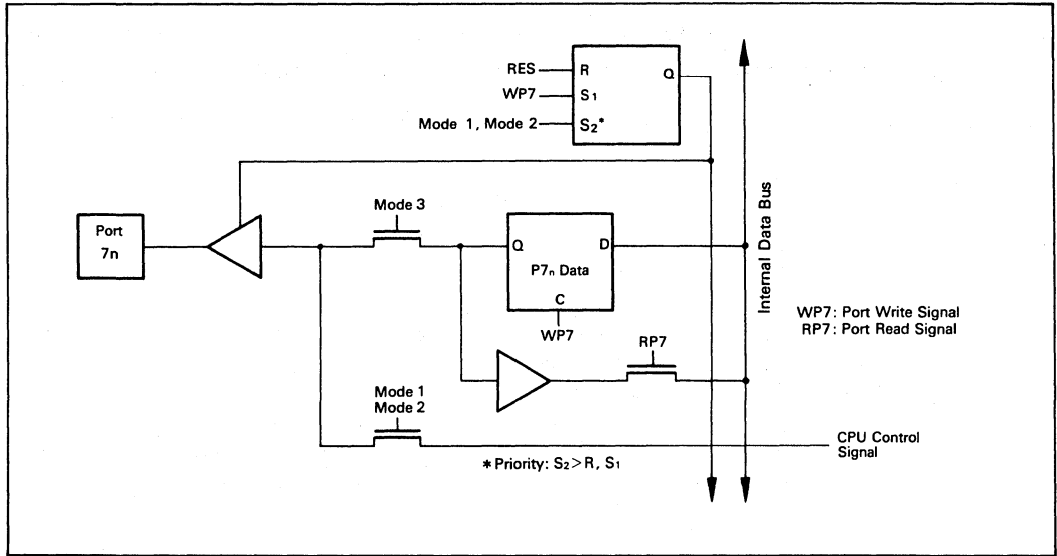


Figure 2-19. Port 7 Block Diagram

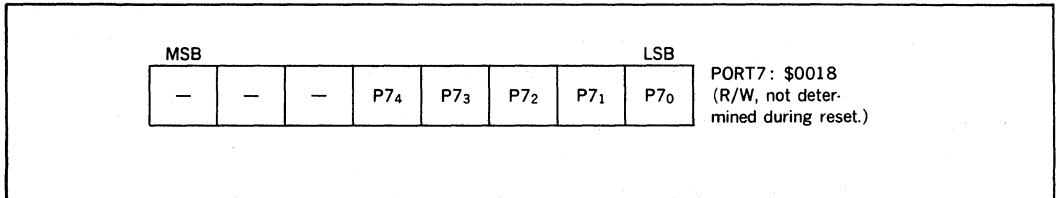


Figure 2-20. Port 7 Register

## 2.5 RAM/Port 5 Control Register

The RAM port 5 control register (RP5CR:\$0014) controls onchip RAM and port 5 (figure 2-22).

### 2.5.1 IRQ<sub>1</sub>E, IRQ<sub>2</sub>E

Setting IRQ<sub>1</sub>E and IRQ<sub>2</sub>E to 1 selects P5<sub>0</sub> and P5<sub>1</sub> as the  $\overline{\text{IRQ}}_1$  and  $\overline{\text{IRQ}}_2$  interrupt inputs. These bits are cleared at reset.

### 2.5.2 MRE, AMRE

When MRE or AMRE is set to 1, P5<sub>2</sub> becomes the MR input. When both are 0, memory ready is inhibited (table 2-8). In mode 3, memory ready is always inhibited, regardless of these bits. MRE is cleared at reset. AMRE is set to 1.

### 2.5.3 HLTE

When HLTE is set to 1, P5<sub>3</sub> becomes the  $\overline{\text{HALT}}$  input. When 0, HALT is inhibited. In mode 3,  $\overline{\text{HALT}}$  is always inhibited, regardless of HLTE. This bit is set to 1 at reset.

### 2.5.4 STBY FLAG

Clearing STBY FLAG by software puts the MCU into standby mode. This flag is set to 1 at reset, so reset cancels the standby mode. If the STBY pin is low, this flag cannot be cleared.

### 2.5.5 RAME

When RAME is set to 1, on-chip RAM is enabled. When 0, it is disabled. RAME is set to 1 at reset. This bit should be set to 0 before going into standby state to protect on-chip RAM data.

### 2.5.6 STBY PWR

When V<sub>CC</sub> is not provided in standby mode, STBY PWR is cleared. If STBY PWR is set before the MCU goes to standby, and remains set after standby V<sub>CC</sub> was continuously supplied, and the contents of on-chip RAM are valid.

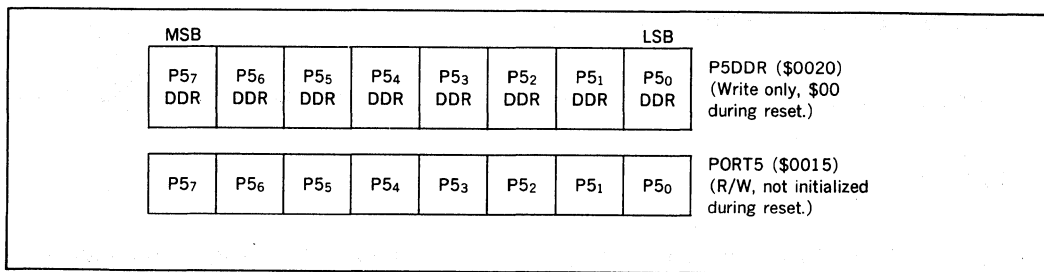


Figure 2-21. Port 5 Register and Data Direction Register

Table 2-7. Port 5 Pin Functions

Port 5 Pin	Alternate Function	Description
P5 <sub>0</sub>	$\overline{\text{IRQ}}_1$	Interrupt input 1
P5 <sub>1</sub>	$\overline{\text{IRQ}}_2$	Interrupt input 2
P5 <sub>2</sub>	MR	Memory ready input
P5 <sub>3</sub>	$\overline{\text{HALT}}$	Halt input
P5 <sub>4</sub>	$\overline{\text{IS}}$	Input Strobe
P5 <sub>5</sub>	$\overline{\text{OS}}$	Output strobe

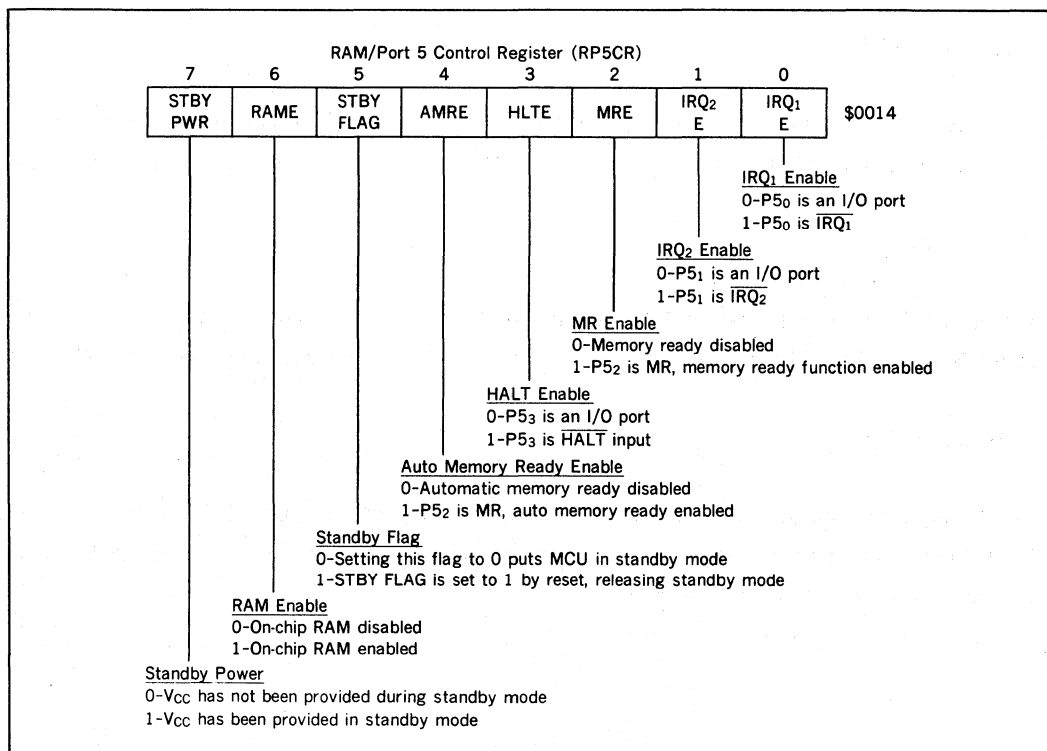


Figure 2-22. RAM/Port 5 Control Register



Table 2-8. Memory Ready Function

MRE	AMRE	Function
0	0	Memory ready inhibited.
0	1	Auto memory ready. When the CPU accesses the external address regardless of MR, E clock automatically stays high one-cycle longer. This state is retained during reset.
1	0	Memory ready. MR pin controls E clock high time.
1	1	When the CPU accesses the external address space with the P5 <sub>2</sub> (MR) pin low the auto memory ready operates. This function useful if there is both high-speed memory and slow memory outside. Input $\overline{CS}$ signal of slow memory to MR pin.

## 2.6 Port 6 Control/Status Register

The port 6 control/status register (P6CSR: \$0021) controls and holds the status of the port 6 handshake interface (figure 2-23). The handshake interface functions as follows.

- Latches the data input at port 6 on the falling edge of  $\overline{IS}$  (P54).
- Outputs  $\overline{OS}$  (P55) when reading or writing to port 6.
- When IS FLAG is set by the falling edge of IS, an interrupt occurs (figure 2-24).

### 2.6.1 LATCH ENABLE

The LATCH ENABLE bit controls the port 6 input latch (IS LATCH). When it is set, the input data at port 6 will be latched in at the falling edge of  $\overline{IS}$  (P54). Reading port 6 clears the latch. If LATCH ENABLE is 0, the input latch is disabled, and P54 acts as an ordinary I/O port. LATCH ENABLE is cleared at reset.

### 2.6.2 OSS

When OSS is set, writing to port 6 initiates an output strobe signal ( $\overline{OS}$ /P55). When OSS is cleared, reading port 6 initiates an  $\overline{OS}$ . OSS is cleared at reset.

### 2.6.3 OSE

When OSE is set, P55 is the output strobe,  $\overline{OS}$ . When cleared, it is a normal I/O port.

### 2.6.4 IS IRQ<sub>1</sub> ENABLE

When IS IRQ<sub>1</sub> ENABLE is set, IS FLAG set causes an  $\overline{IRQ_1}$  interrupt. When cleared, IS FLAG does not cause an interrupt. This bit is cleared during reset.

### 2.6.5 IS FLAG

The IS FLAG is set by the falling edge of  $\overline{IS}$ . It is a read-only flag. It is cleared by reading or writing to port 6 after reading the P6CSR. IS FLAG is cleared during reset.

Table 2-9 shows the conditions that set and reset the port 6 control/status register flags.

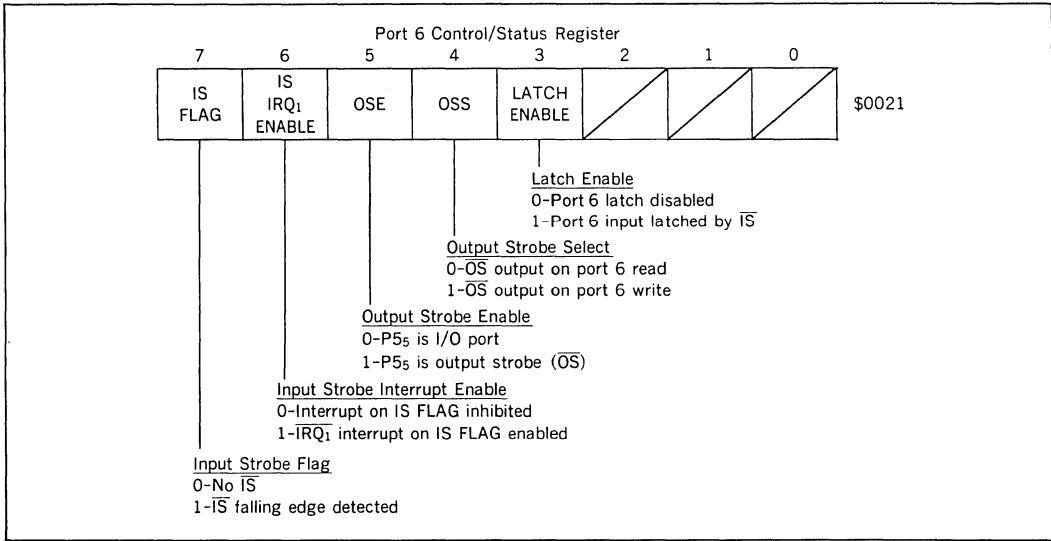


Figure 2-23. Port 6 Control/Status Register

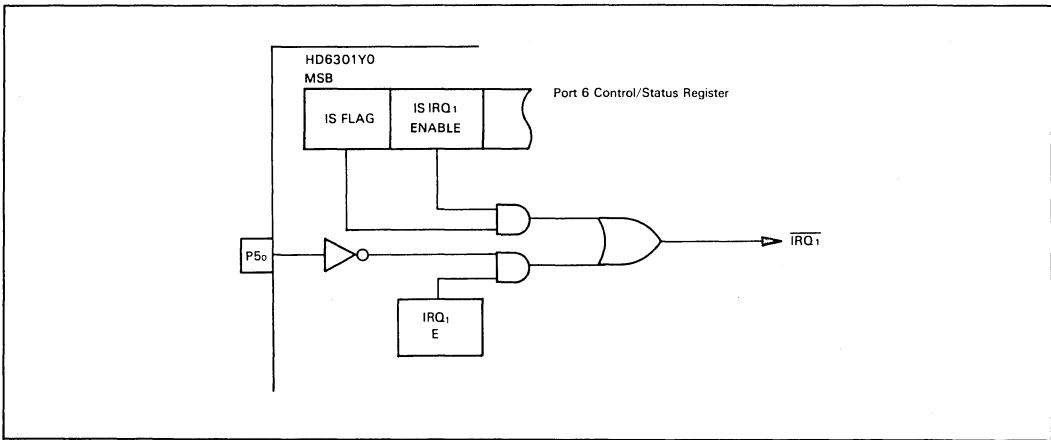


Figure 2-24. Input Strobe Interrupt Block Diagram

Table 2-9. Port 6 Control Status Register Status Flags Set and Reset Conditions

Flag	Set Condition	Clear Condition
IS FLAG	Falling edge input to P5 <sub>4</sub> ( $\overline{IS}$ )	<ul style="list-style-type: none"> <li>• Read the P6CSR then read or write the port 6, when IS FLAG = 1</li> <li>• <math>\overline{RES} = 0</math></li> </ul>
ICF	FRC → ICR by rising or falling edge input to P2 <sub>0</sub> . (Selected by IEDG)	<ul style="list-style-type: none"> <li>• Read the TCSR1 or TCSR2 then ICRH, when ICF = 1</li> <li>• RES = 0</li> </ul>

# Section 3. CPU Function

## 3.1 CPU Registers

The CPU has three 16-bit registers and three 8-bit registers (figure 3-1).

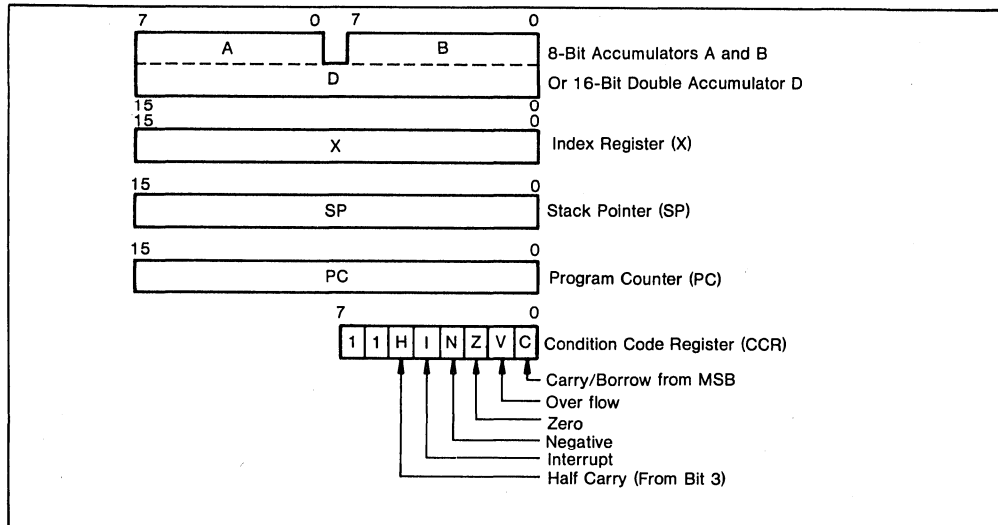


Figure 3-1. CPU Registers

### 3.1.1 Accumulators (ACCA, ACCB, ACCD)

Two 8-bit accumulators, ACCA and ACCB, store the result of arithmetic/logic operations and data. When combined, these make up the 16-bit accumulator ACCD used for 16-bit operations. Note that the contents of ACCA and ACCB are destroyed by an ACCD operation.

### 3.1.2 Index Register (IX)

The 16-bit register IX stores 16-bit data for use in indexed addressing or for general purposes.

### 3.1.3 Stack Pointer (SP)

The contents of the 16-bit register SP indicate the address of a stack. SP can also be used as a general-purpose register.

### 3.1.4 Program Counter (PC)

The contents of the 16-bit PC indicate the address of the instruction being executed. Note that software cannot access this register.

### 3.1.5 Condition Code Register (CCR)

The CCR register consists of the carry (C), overflow (V), zero (Z), negative (N), interrupt mask (I), and half-carry (H) bits. After an instruction is executed, the CCR bits change state depending on the result of the operation. They can be tested by conditional branch instructions. The upper two bits of this register are not used.

**Half-Carry (H):** H is set to 1 if a carry at bit 3 or bit 4 occurs during an ADD, ABA, or ADC instruction. It is cleared if no carry occurs.

**Interrupt Mask (I):** When I is set to 1, it disables all maskable interrupts ( $\overline{IRQ}_1$ ,  $\overline{IRQ}_2$ , and  $IRQ_3$ ).

**Negative (N):** N is set to 1 if the MSB of the result of an operation is 1. N is cleared if it is 0.

**Zero (Z):** Z is set to 1 if the result of an operation is zero. Z is cleared if it is not zero.

**Overflow (V):** V is set to 1 if the result of an operation shows a two's complement overflow. It is cleared if there is no overflow.

**Carry (C):** C is set to 1 if a carry or borrow is generated from the MSB. If there is no carry or borrow, it is cleared.

## 3.2 Addressing Modes

The HD6301Y0, HD6303Y and HD63701Y0 instructions have seven addressing modes.

### 3.2.1 Accumulator Addressing (ACCX)

The instruction addresses an accumulator and ACCA or ACCB is selected. Accumulator addressing instructions take one byte.

### 3.2.2 Immediate Addressing

Immediate addressing places the data in the second byte of an instruction, except LDS and LDX, which use the second and third bytes. An immediate instruction causes the CPU to address this operand. Immediate instructions take 2 or 3 bytes.

### 3.2.3 Direct Addressing

In direct addressing, the second byte of an instruction holds the address where the data is stored. 256



bytes (\$00-\$FF) can be addressed directly. Storing data in this area reduces instruction time, so configuring \$00-\$FF as user's RAM is recommended. Direct addressing instructions take 2 bytes, or 3 bytes for AIM, OIM, EIM, or TIM.

### **3.2.4 Extended Addressing**

In extended addressing, the second byte of an instruction holds the upper eight bits of the absolute address of the stored data, and the third byte holds the lower eight bits. Extended addressing instructions take 3 bytes.

### **3.2.5 Indexed Addressing**

In indexed addressing, the second byte of the instruction (third byte for AIM, OIM, EIM, or TIM instructions) is added to the lower eight bits of the index register. The carry is added to the upper eight bits of the index register, and the 16-bit sum is the memory location of the data. The modified address is held in the temporary address register, so the index register doesn't change. Indexed addressing instructions take 2 bytes, or 3 bytes for AIM, OIM, EIM, or TIM.

### **3.2.6 Implied Addressing**

In implied addressing, the instruction itself specifies the address. For example, the instruction addresses the stack pointer or index register. Implied addressing instructions take 1 byte.

### **3.2.7 Relative Addressing**

In relative addressing, the second byte of the instruction and the lower eight bits of the program counter are added. The carry or borrow is added to the upper eight bits of the program counter. Locations from -126 to +129 bytes from the current location can be addressed. Relative addressing instructions take 2 bytes.

### 3.3 Instruction Set

The HD6301Y0, HD6303Y, and HD63701Y0 are object-code upwardly compatible with the HD6801 to use all instructions of the HMCS6800. The instruction time of key instructions has been reduced improving throughput.

#### 3.3.1 Additional Instructions

Bit manipulation, index register and accumulator exchange, and sleep instructions have also been added to the HD6801 instruction set. AIM, OIM, EOM, and TIM are 3 byte instructions. The first byte is the opcode, second byte is the immediate data, and the third byte is the address modifier.

**AIM:** ANDs the immediate data with the memory contents and stores the result in memory. (M) AND (IMM) → (M).

**OIM:** ORs the immediate data with the memory contents and stores the result in memory. (M) OR (IMM) → (M).

**EIM:** EORs the immediate data with the memory contents and stores the result in memory. (M) EOR (IMM) → (M).

**TIM:** ANDs the immediate data with the memory contents and changes the related flag in the condition code register. (M) AND (IMM).

**XGDx:** Exchanges the contents of the accumulator with the contents of the index register. (ACCD) ↔ (IX).

**SLP:** Puts the MCU into sleep mode. Refer to 3.5 Low Power Dissipation Mode for details.

#### 3.3.2 Instruction Set Summary

Tables 3-1 to 3-5 summarize the instruction set.

- Accumulator and memory manipulation instructions: table 3-1
- Index register and stack manipulation instructions: table 3-2
- Jump and branch instructions: table 3-3
- Condition code register manipulation: table 3-4
- Opcode map: table 3-5

Table 3-1. Accumulator and Memory Manipulation Instructions

Operations	Mnemonic	Addressing Modes												Boolean/ Arithmetic Operation	Condition Code Register								
		IMMED			DIRECT			INDEX			EXTEND				IMPLIED			5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#		OP	~	#	H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	4	2	BB	4	3			A + M → A	:	•	:	:	:	:	
	ADDB	CB	2	2	DB	3	2	EB	4	2	FB	4	3			B + M → B	:	•	:	:	:	:	
Add Double	ADDD	C3	3	3	D3	4	2	E3	5	2	F3	5	3			A : B + M : M + 1 → A : B	:	•	:	:	I	:	
Add Accumulators	ABA													1B	1	1	A + B → A	:	•	:	:	:	
Add With Carry	ADCA	89	2	2	99	3	2	A9	4	2	B9	4	3			A + M + C → A	:	•	:	:	:		
	ADCB	C9	2	2	D9	3	2	E9	4	2	F9	4	3			B + M + C → B	:	•	:	:	:		
AND	ANDA	84	2	2	94	3	2	A4	4	2	B4	4	3			A · M → B	•	•	:	:	R	•	
	ANDB	C4	2	2	D4	3	2	E4	4	2	F4	4	3			B · M → B	•	•	:	:	R	•	
Bit Test	BIT A	85	2	2	95	3	2	A5	4	2	B5	4	3			A · M	•	•	:	:	R	•	
	BIT B	C5	2	2	D5	3	2	E5	4	2	F5	4	3			B · M	•	•	:	:	R	•	
Clear	CLR							6F	5	2	7F	5	3			00 → M	•	•	R	S	R	R	
	CLRA													4F	1	1	00 → A	•	•	R	S	R	R
	CLRB													5F	1	1	00 → B	•	•	P	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	4	2	B1	4	3			A - M	•	•	:	:	:	:	
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3			B - M	•	•	:	:	:	:	
Compare Accumulators	CBA													11	1	1	A - B	•	•	:	:	:	:
Complement, 1's	COM							63	6	2	73	6	3			$\overline{M} \rightarrow M$	•	•	:	:	R	S	
	COMA													43	1	1	$\overline{A} \rightarrow A$	•	•	:	:	R	S
	COMB													53	1	1	$\overline{B} \rightarrow B$	•	•	:	:	R	S
Complement, 2's (Negate)	NEG							60	6	2	70	6	3			00 - M → M	•	•	:	:	①	②	
	NEGA													40	1	1	00 - A → A	•	•	:	:	①	②
	NEGB													50	1	1	00 - B → B	•	•	:	:	①	②
Decimal Adjust, A	DAA													19	2	1	Converts binary add of BCD characters into BCD format	•	•	:	:	:	③
Decrement	DEC							6A	6	2	7A	6	3			M - 1 → M	•	•	:	:	④	•	
	DECA													4A	1	1	A - 1 → A	•	•	:	:	④	•
	DECB													5A	1	1	B - 1 → B	•	•	:	:	④	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	4	2	B8	4	3			A ⊕ M → A	•	•	:	:	R	•	
	EORB	C8	2	2	D8	3	2	E8	4	2	F8	4	3			B ⊕ M → B	•	•	:	:	R	•	
Increment	INC							6C	6	2	7C	6	3			M + 1 → M	•	•	:	:	⑤	•	
	INCA													4C	1	1	A + 1 → A	•	•	:	:	⑤	•
	INCB													5C	1	1	B + 1 → B	•	•	:	:	⑤	•
Load Accumulator	LDAA	86	2	2	96	3	2	A6	4	2	B6	4	3			M → A	•	•	:	:	R	•	
	LDAB	C6	2	2	D6	3	2	E6	4	2	F6	4	3			M → B	•	•	:	:	R	•	
Load Double Accumulator	LDD	CC	3	3	DC	4	2	EC	5	2	FC	5	3			M + 1 → B, M → A	•	•	:	:	R	•	
Multiply Unsigned	MUL													3D	7	1	A × B → A : B	•	•	•	•	•	⑥
OR, Inclusive	ORAA	8A	2	2	9A	3	2	AA	4	2	BA	4	3			A ∨ M → A	•	•	:	:	R	•	
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3			B ∨ M → B	•	•	:	:	R	•	
Push Data	PSHA													36	4	1	A → Msp, SP - 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → Msp, SP - 1 → SP	•	•	•	•	•	•

Note: Condition Code Register will be explained in Note of table 3-4.

Table 3-1. Accumulator and Memory Manipulation Instructions (Cont.)

Operations	Mnemonic	Addressing Modes												Boolean/ Arithmetic Operation	Condition Code Register								
		IMMED			DIRECT			INDEX			EXTEND				IMPLIED			5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#		OP	~	#	H	I	N	Z	V	C
Pull Data	PULA													32	3	1	SP+1→SP, Msp→A	•	•	•	•	•	•
	PULB													33	3	1	SP+1→SP, Msp→B	•	•	•	•	•	•
Rotate Left	ROL						69	6	2	79	6	3					*1	•	•	•	•	•	•
	ROLA												49	1	1	•		•	•	•	•	•	
	ROLB												59	1	1	•		•	•	•	•	•	
Rotate Right	ROR						66	6	2	76	6	3					*2	•	•	•	•	•	•
	RORA												46	1	1	•		•	•	•	•	•	
	RORB												56	1	1	•		•	•	•	•	•	
Shift Left Arithmetic	ASL						68	6	2	78	6	3					*3	•	•	•	•	•	•
	ASLA												48	1	1	•		•	•	•	•	•	
	ASLB												58	1	1	•		•	•	•	•	•	
Double Shift Left, Arithmetic	ASLD												05	1	1	*4	•	•	•	•	•	•	
Shift Right Arithmetic	ASR						67	6	2	77	6	3					*5	•	•	•	•	•	•
	ASRA												47	1	1	•		•	•	•	•	•	
	ASRB												57	1	1	•		•	•	•	•	•	
Shift Right Logical	LSR						64	6	2	74	6	3					*6	•	•	R	•	•	•
	LSRA												44	1	1	•		•	R	•	•	•	
	LSRB												54	1	1	•		•	R	•	•	•	
Double Shift Right Logical	LSRD												04	1	1	*7	•	•	R	•	•	•	
Store Accumulator	STAA				97	3	2	A7	4	2	B7	4	3				A→M	•	•	•	•	R	•
	STAB				D7	3	2	E7	4	2	F7	4	3				B→M	•	•	•	•	R	•
Store Double Accumulator	STD				DD	4	2	ED	5	2	FD	5	3				A→M B→M+1	•	•	•	•	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3				A→M→A	•	•	•	•	•	•
	SUBB	C0	2	2	D0	3	2	E0	4	2	F0	4	3				B→M→B	•	•	•	•	•	•
Double Subtract	SUBD	83	3	3	93	4	2	A3	5	2	B3	5	3				A : B→M : M+1→ A : B	•	•	•	•	•	•
Subtract Accumulators	SBA												10	1	1	A→B→A	•	•	•	•	•	•	
Subtract With Carry	SBCA	82	2	2	92	3	2	A2	4	2	B2	4	3				A→M→C→A	•	•	•	•	•	•
	SBCB	C2	2	2	D2	3	2	E2	4	2	F2	4	3				B→M→C→B	•	•	•	•	•	•
Transfer Accumulators	TAB												16	1	1	A→B	•	•	•	•	R	•	
	TBA												17	1	1	B→A	•	•	•	•	R	•	
Test Zero or Minus	TST						6D	4	2	7D	4	3					M→00	•	•	•	•	R	R
	TSTA												4D	1	1	A→00	•	•	•	•	R	R	
	TSTB												5D	1	1	B→00	•	•	•	•	R	R	
And Immediate	AIM				71	6	3	61	7	3							M·IMM→M	•	•	•	•	R	•
OR Immediate	OIM				72	6	3	62	7	3							M∨IMM→M	•	•	•	•	R	•
EOR Immediate	EIM				75	6	3	65	7	3							M⊕IMM→M	•	•	•	•	R	•
Test Immediate	TIM				7B	4	3	6B	5	3							M·IMM	•	•	•	•	R	•

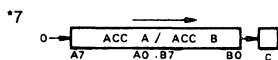
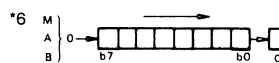
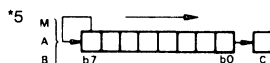
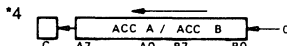
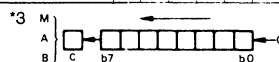
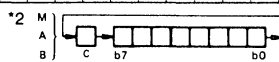
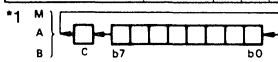


Table 3-2. Index Register and Stack Manipulation Instructions

Pointer Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	5	2	BC	5	3			X ← M:M+1	●	●	1	1	1	1	
Decrement Index Reg	DEX													09	1	1	X ← X	●	●	●	1	●	●
Decrement Stack Pntr	DES													34	1	1	SP ← SP	●	●	●	●	●	●
Increment Index Reg	INX													08	1	1	X ← X	●	●	●	1	●	●
Increment Stack Pntr	INS													31	1	1	SP ← SP	●	●	●	●	●	●
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	5	2	FE	5	3			M ← X <sub>H</sub> , (M+1) ← X <sub>L</sub>	●	●	⑦	1	R	●	
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	5	2	BE	5	3			M ← SP <sub>H</sub> , (M+1) ← SP <sub>L</sub>	●	●	⑦	1	R	●	
Store Index Reg	STX				DF	4	2	EF	5	2	FF	5	3			X <sub>H</sub> ← M, X <sub>L</sub> ← (M+1)	●	●	⑦	1	R	●	
Store Stack Pntr	STS				9F	4	2	AF	5	2	BF	5	3			SP <sub>H</sub> ← M, SP <sub>L</sub> ← (M+1)	●	●	⑦	1	R	●	
Index Reg → Stack Pntr	TXS													35	1	1	X ← SP	●	●	●	●	●	●
Stack Pntr → Index Reg	TSX													30	1	1	SP ← X	●	●	●	●	●	●
Add	ABX													3A	1	1	B ← X ← X	●	●	●	●	●	●
Push Data	PSHX													3C	5	1	X <sub>L</sub> ← Msp, SP ← SP X <sub>H</sub> ← Msp, SP ← SP	●	●	●	●	●	●
Pull Data	PULX													38	4	1	SP ← SP, Msp ← X <sub>H</sub> SP ← SP, Msp ← X <sub>L</sub>	●	●	●	●	●	●
Exchange	XGDX													18	2	1	ACCD ← IX	●	●	●	●	●	●

Note: Condition Code Register will be explained in Note of table 3-4.

Table 3-3. Jump and Branch Instructions

Operations	Mnemonic	Addressing Modes															Branch Test	Condition Code Register					
		RELATIVE			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Branch Always	BRA	20	3	2												None	•	•	•	•	•	•	
Branch Never	BRN	21	3	2												None	•	•	•	•	•	•	
Branch if Carry Clear	BCC	24	3	2												C=0	•	•	•	•	•	•	
Branch if Carry Set	BCS	25	3	2												C=1	•	•	•	•	•	•	
Branch if = Zero	BEQ	27	3	2												Z=1	•	•	•	•	•	•	
Branch if ≥ Zero	BGE	2C	3	2												$N\oplus V=0$	•	•	•	•	•	•	
Branch if > Zero	BGT	2E	3	2												$Z+(N\oplus V)=0$	•	•	•	•	•	•	
Branch if Higher	BHI	22	3	2												$C+Z=0$	•	•	•	•	•	•	
Branch if ≤ Zero	BLE	2F	3	2												$Z+(N\oplus V)=1$	•	•	•	•	•	•	
Branch if Lower Or Same	BLS	23	3	2												$C+Z=1$	•	•	•	•	•	•	
Branch if < Zero	BLT	2D	3	2												$N\oplus V=1$	•	•	•	•	•	•	
Branch if Minus	BMI	2B	3	2												N=1	•	•	•	•	•	•	
Branch if Not Equal Zero	BNE	26	3	2												Z=0	•	•	•	•	•	•	
Branch if Overflow Clear	BVC	28	3	2												V=0	•	•	•	•	•	•	
Branch if Overflow Set	BVS	29	3	2												V=1	•	•	•	•	•	•	
Branch if Plus	BPL	2A	3	2												N=0	•	•	•	•	•	•	
Branch To Subroutine	BSR	8D	5	2													•	•	•	•	•	•	
Jump	JMP							6E	3	2	7E	3	3				•	•	•	•	•	•	
Jump To Subroutine	JSR				9D	5	2	AD	5	2	BD	6	3				•	•	•	•	•	•	
No Operation	NOP													01	1	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI													3B	10	1		⑧					
Return From Subroutine	RTS													39	5	1		•	•	•	•	•	•
Software Interrupt	SWI													3F	12	1		•	S	•	•	•	•
Wait for Interrupt*	WAI													3E	9	1		•	⑨	•	•	•	•
Sleep	SLP													1A	4	1		•	•	•	•	•	•

Note: \* WAI puts R/W high; Address Bus goes to FFFF; Data Bus goes to the three state. Condition Code Register will be explained in Note of table 3-4.

6

**Table 3-4. Condition Code Register Manipulation Instructions**

Operations	Mnemonic	Addressing Modes			Boolean Operation	Condition Code Register					
		Implied				H	I	N	Z	V	C
		OP	~	#							
Clear Carry	CLC	0C	1	1	0→C	●	●	●	●	●	R
Clear Interrupt Mask	CLI	0E	1	1	0→I	●	R	●	●	●	●
Clear Overflow	CLV	0A	1	1	0→V	●	●	●	●	R	●
Sat Carry	SEC	0D	1	1	1→C	●	●	●	●	●	S
Set Interrupt Mask	SEI	0F	1	1	1→I	●	S	●	●	●	●
Set Overflow	SEV	0B	1	1	1→V	●	●	●	●	S	●
Accumulator A→CCR	TAP	06	1	1	A→CCR	⑩					
CCR→Accumulator A	TPA	07	1	1	CCR→A	●	●	●	●	●	●

**Legend**

- OP Operation Code (Hexadecimal)
- ~ Number of MCU Cycles
- M<sub>SP</sub> Contents of memory location pointed to by Stack Pointer
- # Number of Program Bytes
- + Arithmetic Plus
- Arithmetic Minus
- Boolean AND
- ⊕ Boolean Inclusive OR
- ⊕ Boolean Exclusive OR
- M Complement of M
- Transfer into
- 0 Bit = Zero
- 00 Byte = Zero

**Condition Code Symbols**

- H Half-carry from bit 3 to bit 4
- I Interrupt mask
- N Negative (sign bit)
- Z Zero (byte)
- V Overflow, 2's complement
- C Carry/Borrow from/to bit 7
- R Reset Always
- S Set Always
- ↑ Set if true after test or clear
- Not Affected

Note: Condition Code Register Notes: (Bit set if test is true and cleared otherwise)

- ① (Bit V) Test: Result = 10000000?
- ② (Bit C) Test: Result = 00000000?
- ③ (Bit C) Test: BCD Character of high-order byte greater than 10? (Not cleared if previously set)
- ④ (Bit V) Test: Operand = 10000000 prior to execution?
- ⑤ (Bit V) Test: Operand = 01111111 prior to execution?
- ⑥ (Bit V) Test: Set equal to N + C = 1 after the execution of instructions
- ⑦ (Bit N) Test: Result less than zero? (Bit 15=1)
- ⑧ (All Bit) Load condition code register from stack.
- ⑨ (Bit I) Set when interrupt occurs. If previous set, a non-maskable interrupt is required to exist the wait state.
- ⑩ (AI Bit) Set according to the contents of accumulator A.
- ⑪ (Bit C) Result of multiplication bit 7=1? (ACCB)

**Table 3-5. Memory Map**

OP CODE					ACC A	ACC B	IND	EXT/DIR*	ACCA or SP				ACCB or X					
	HI	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
LO	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
0000	0	SBA	BRA	TSX	NEG				SUB				0					
0001	1	NOP	CBA	BRN	INS	AIM				CMP				1				
0010	2			BHI	PULA	OIM				SBC				2				
0011	3			BLS	PULB	COM				SUBD				ADDD				3
0100	4	LSRD		BCC	DES	LSR				AND				4				
0101	5	ASLD		BCS	TXS	EIM				BIT				5				
0110	6	TAP	TAB	BNE	PSHA	ROR				LDA				6				
0111	7	TPA	TBA	BEQ	PSHB	ASR				STA				STA				7
1000	8	INX	XGDX	BVC	PULX	ASL				EOR				8				
1001	9	DEX	DAA	BVS	RTS	ROL				ADC				9				
1010	A	CLV	SLP	BPL	ABX	DEC				ORA				A				
1011	B	SEV	ABA	BMI	RTI	TIM				ADD				B				
1100	C	CLC		BGE	PSHX	INC				CPX				LDD				C
1101	D	SEC		BGT	MUL	TST				BSR	JSR			STD				D
1110	E	CLI		BGL	WAI	JMP				LDS				LDX				E
1111	F	SEI		BLE	SWI	CLR				STS				STX				F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

☐ UNDEFINED OP CODE

\* Only AIM, OIM, EIM, TIM instructions



### 3.4 CPU Instruction Flow

When operating, the CPU fetches an instruction from memory and executes the required function. This sequence starts from  $\overline{\text{RES}}$  high, and repeats itself continuously if not affected by a special instruction or control signal. SWI, RTI, WAI, and SLP instructions change this operation, and  $\overline{\text{NMI}}$ ,  $\overline{\text{IRQ}}_1$ ,  $\overline{\text{IRQ}}_2$ ,  $\text{IRQ}_3$ ,  $\overline{\text{HALT}}$ , and  $\overline{\text{STBY}}$  control it. Figure 3-2 shows the CPU mode transitions, and figure 3-3 is the CPU system flowchart. Table 3-6 shows the CPU operating states and port states.

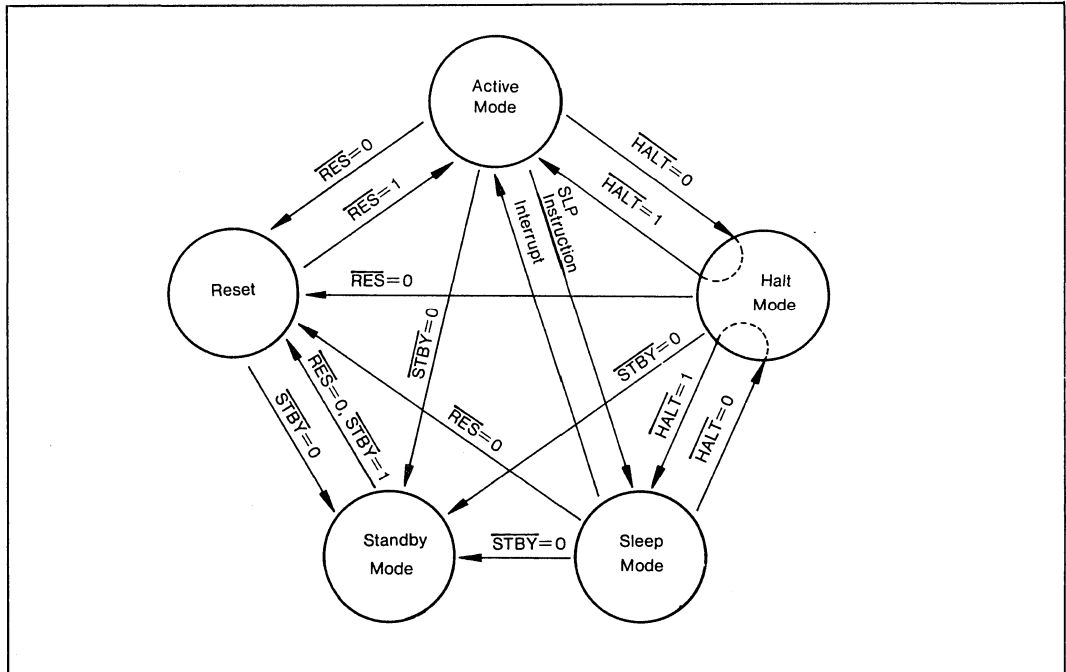
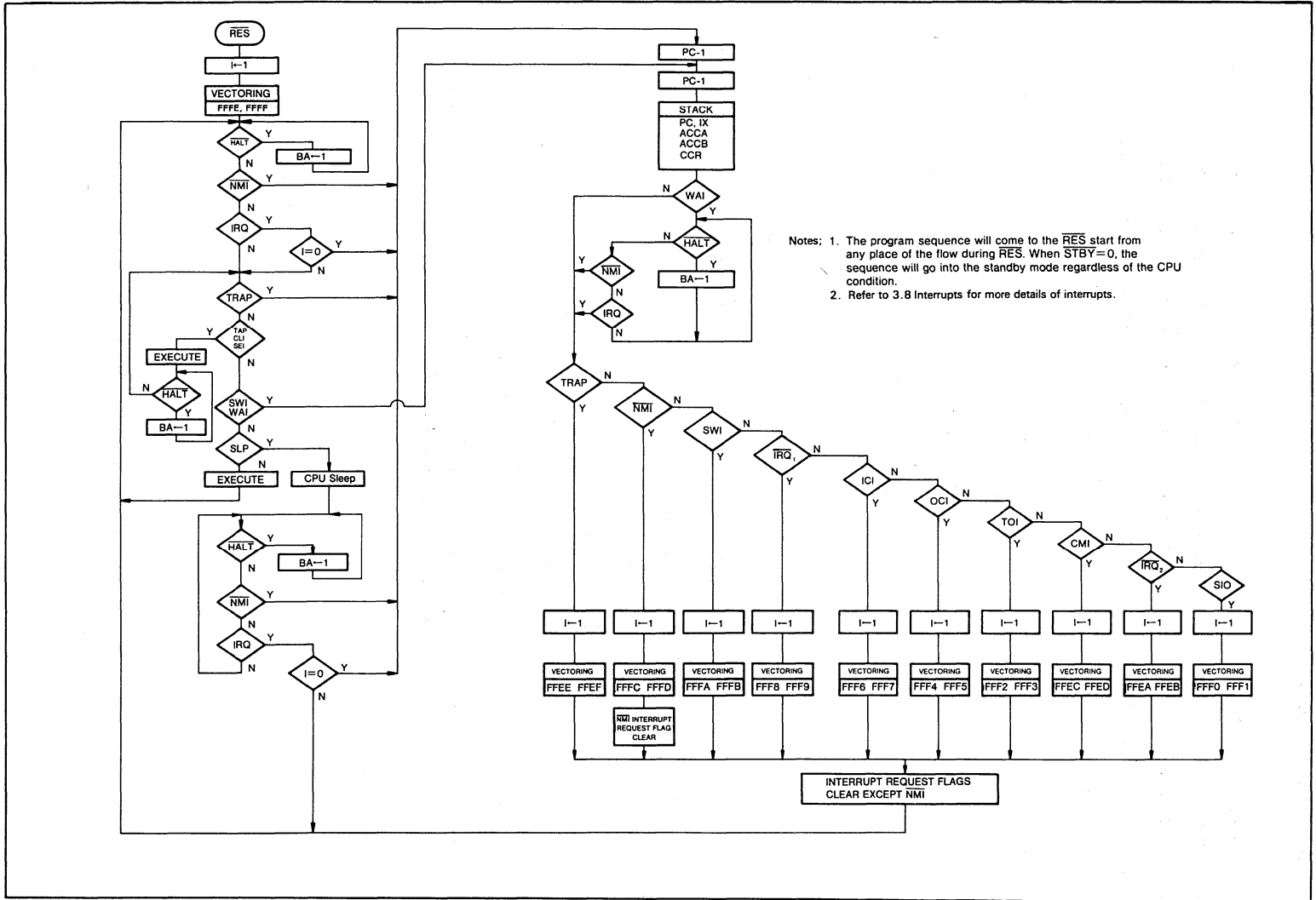


Figure 3-2. CPU Operation Mode Transitions





Notes: 1. The program sequence will come to the RES start from any place of the flow during RES. When STBY=0, the sequence will go into the standby mode regardless of the CPU condition.  
 2. Refer to 3.8 Interrupts for more details of interrupts.

Figure 3-3. System Flowchart

Table 3-6. CPU Operating States and Port States

Port	Mode	Reset	Standby <sup>3</sup>	Halt <sup>4</sup>	Sleep
1 (A <sub>0</sub> -A <sub>7</sub> )	1, 2	High	High impedance	High impedance	High
	3	High impedance	High impedance		Keep
2	1, 2	High impedance	High impedance	Keep	Keep
	3	High impedance	High impedance		Keep
3 (D <sub>0</sub> -D <sub>7</sub> )	1,2	High impedance	High impedance	High impedance	High impedance
	3	High impedance	High impedance		Keep
4 (A <sub>8</sub> -A <sub>15</sub> )	1	High	High impedance	High impedance	High
	2	High impedance	High impedance	High impedance	Note 5
	3	High impedance	High impedance		Keep
5	1, 2	High impedance	High impedance	Keep	Keep
	3	High impedance	High impedance		Keep
6	1, 2	High impedance	High impedance	Keep	Keep
	3	High impedance	High impedance		Keep
7	1, 2	Note 1	High impedance	Note 2	Note 1
	3	High impedance	High impedance		Keep

Notes:

1.  $\overline{RD}$ ,  $\overline{WR}$ ,  $R/\overline{W}$ ,  $\overline{LIR}$  = high; BA = low.
2.  $\overline{RD}$ ,  $\overline{WR}$ ,  $R/\overline{W}$  = high impedance;  $\overline{LIR}$ , BA = high.
3. E is high impedance in standby state.
4.  $\overline{HALT}$  cannot be accepted in mode 3.
5. Address output pin = high; Input port = high impedance.

### 3.5 Low Power Dissipation Modes

The MCU has two low power dissipation modes, sleep and standby. Table 3-7 shows the MCU state in sleep and standby modes.

Table 3-7. Sleep and Standby Modes

	Sleep Mode	Standby Mode
Oscillation circuits	Continue operation	Stop
CPU	Stop	Stop
CPU registers	Hold	Undefined
RAM	Hold	Hold
I/O pins	Hold	High impedance
Timers	Continue operation	Stop
SCI	Continue operation	Stop
Internal Registers	Hold	Reset
How to release	Interrupt STBY = low Reset start	$\overline{STBY}$ = high before reset start (Hold RES low after $\overline{STBY}$ high until oscillator stabilizes, 20 ms min)

### 3.5.1 Sleep Mode

The MCU goes into sleep mode when the SLP instruction is executed. In the sleep mode, the CPU stops operation while maintaining the registers' contents. Peripherals such as the timers and the SCI continue their functions. One-fifth as much power is dissipated in sleep mode as in the operating mode.

The sleep mode is terminated by an interrupt, or a  $\overline{RES}$  or  $\overline{STBY}$  signal.  $\overline{RES}$  causes the MCU to reset,  $\overline{STBY}$  causes it to go into standby mode. When the CPU receives an interrupt request, it returns to operating mode. If the interrupts are enabled, it branches to the interrupt service routine. If they are masked, it executes the next instruction. However, if timer 1 or 2 prohibits a timer interrupt, the CPU won't cancel the sleep mode because there is no interrupt request to the CPU.

The sleep mode reduces power dissipation for a system that doesn't need the CPU's continuous operation. Figure 3-4 is the sleep instruction timing chart.

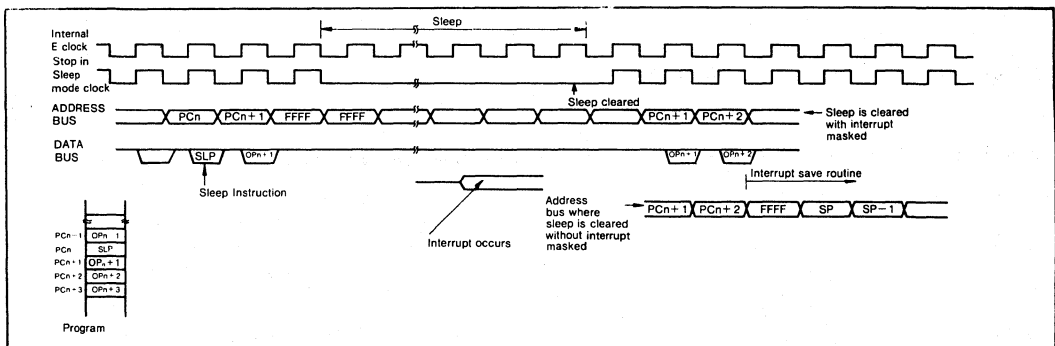


Figure 3-4. Sleep Instruction Timing

### 3.5.2 Standby Mode

When the  $\overline{\text{STBY}}$  input goes low, the MPU stops all clocks and goes to the reset state. In this mode, power dissipation is greatly reduced. All pins except  $V_{\text{CC}}$ ,  $V_{\text{SS}}$ ,  $\overline{\text{STBY}}$ , and XTAL (outputs 0) are detached from the MCU internally, and go to high impedance.

In standby mode, power is supplied to the MCU, so that the contents of RAM are retained. The MCU returns from this mode with a reset.

An example of the use of this mode follows. First, save the CPU state and SP contents in RAM by an NMI routine. Then disable the RAME bit in the RAM control register and set the STBY PWR bit to go to standby mode. If the STBY PWR bit is still set after reset start, power has been supplied to the MCU and the RAM contents have been retained properly. The system can restore itself by returning the pre-standby information to the SP and registers. Figure 3-5 shows the timing at the  $\overline{\text{NMI}}$ ,  $\overline{\text{RES}}$  and  $\overline{\text{STBY}}$  pins.

Note: In standby mode, the mode program pins,  $\text{MP}_0$  and  $\text{MP}_1$ , should be held according to the operation mode. If they are opened, the standby current will increase over the specified value.

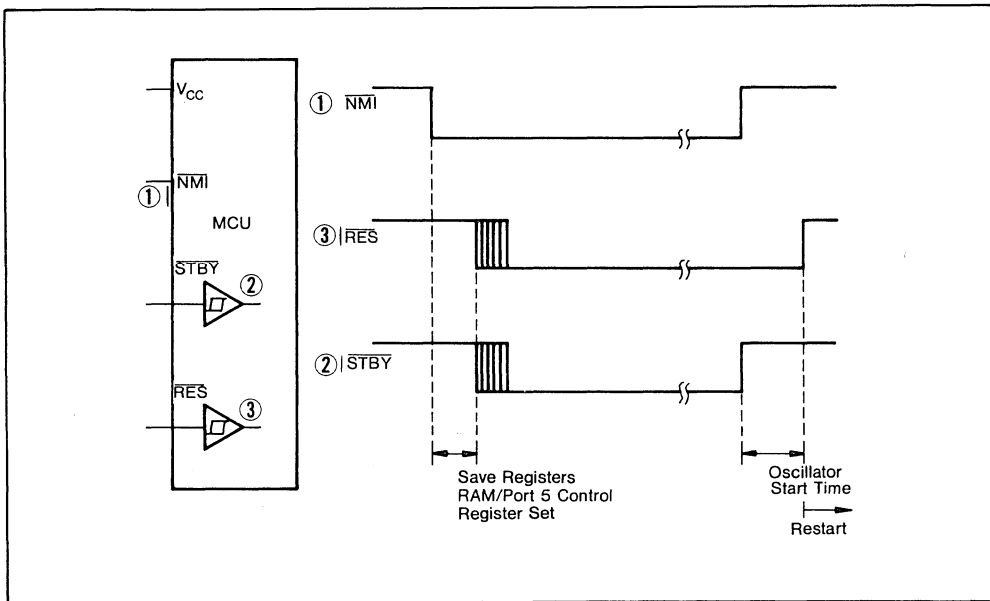


Figure 3-5. Standby Mode Timing

## 3.6 Trap Function

The CPU generates an interrupt with the highest priority (TRAP) when it fetches an undefined instruction or an instruction from outside of memory space. The trap function prevents system malfunctions caused by noise or program error.

### 3.6.1 Opcode Error

When the CPU fetches an undefined opcode, it saves the CPU registers as well as performing the normal interrupt procedure and branches to TRAP (\$FFEE, \$FFEF). This has the highest priority next to reset.

### 3.6.2 Address Error

When an instruction is fetched from outside the internal ROM, RAM, and external memory area, the MCU generates an address interrupt as well as an opcode error. But on a system with no external memory, a trap is not generated if an instruction is fetched from the external memory area. Table 3-8 shows the addresses where an address error occurs in each mode. This function is available only for an instruction fetch, and does not apply to data read/write.

Table 3-8. Address Error Addresses

Mode	Address
1	\$0000-\$001F
2	\$0000-\$001F
3	\$0000-\$003F, \$0100-\$EFFF

### 3.6.3 Caution

The trap function has a retry function other interrupts do not have. The program flow returns to the address where the trap occurred when RTI returns the CPU to the main routine from the TRAP routine. The retry can prevent problems caused by noise, etc. However, if another trap occurs, the program can repeat the retry/TRAP cycle forever. Consideration is necessary in programming.

In figure 3-6, after executing instruction OPn, the MPU fetches and decodes an undefined opcode and generates a trap interrupt. When the RTI is executed in the trap interrupt servicing routine, the MPU will put \$FF03 in the PC, fetch the same opcode, and generate the trap again. The MPU will endlessly repeat loop ABC.

In figure 3-7, after executing the BSR, the branch destination address is output to the address bus to fetch the first instruction of the subroutine. If \$0001 is erroneously output as the address, the MPU will decode it and generate a trap interrupt. When the RTI is executed in the trap interrupt servicing routine, the MPU will put \$0001 in the PC, and start from this address. This will generate another trap, in an endless loop.

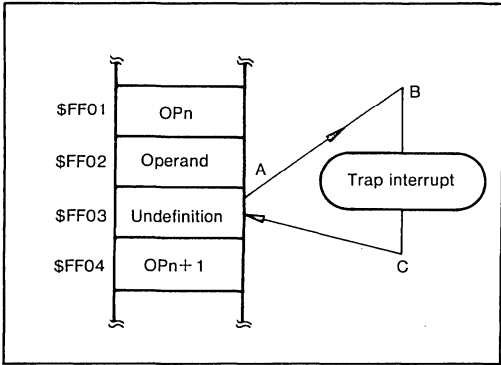


Figure 3-6. Executing an Undefined Opcode

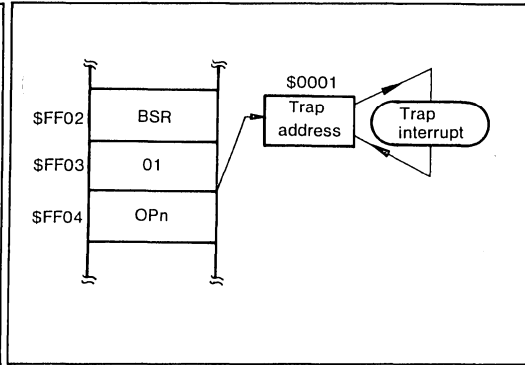


Figure 3-7. Erroneous Fetch

### 3.7 Reset

To reset the MCU during operation, hold  $\overline{\text{RES}}$  low for at least 3 system-clock cycles. At the third cycle, when the clock signal is low, all the address buses become high. While  $\overline{\text{RES}}$  is low, the buses remain high. When  $\overline{\text{RES}}$  goes high, the MCU starts the following operations.

1. Latches the value of the mode program pins,  $\text{MP}_1$  and  $\text{MP}_0$ .
2. Initializes the internal registers (see table 2-3).
3. Sets the interrupt mask bit. For the CPU to recognize the maskable interrupts  $\overline{\text{IRQ}}_1$ ,  $\overline{\text{IRQ}}_2$ , and  $\text{IRQ}_3$ , this bit should be cleared in advance.
4. Puts the contents (= start address) of the last two addresses (\$FFFE, \$FFFF) into the program counter and starts the program from this address. See table 2-4.

The MCU cannot accept a reset input until the clock oscillation is stable after power-on (20 ms maximum). This is because the reset signal is internally synchronized to the clock as shown in figure 3-8. Until oscillation starts, the MCU is undefined. As the I/O ports are controlled directly by the  $\overline{\text{RES}}$  pin, they are reset after power-on reset. At this time, the data registers of these ports don't change. Refer to 2.4 Ports for the state of the ports during reset. Figure 3-9 shows reset timing.

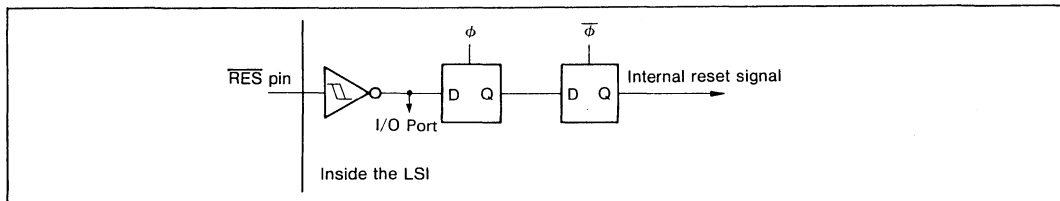


Figure 3-8 Reset Circuit

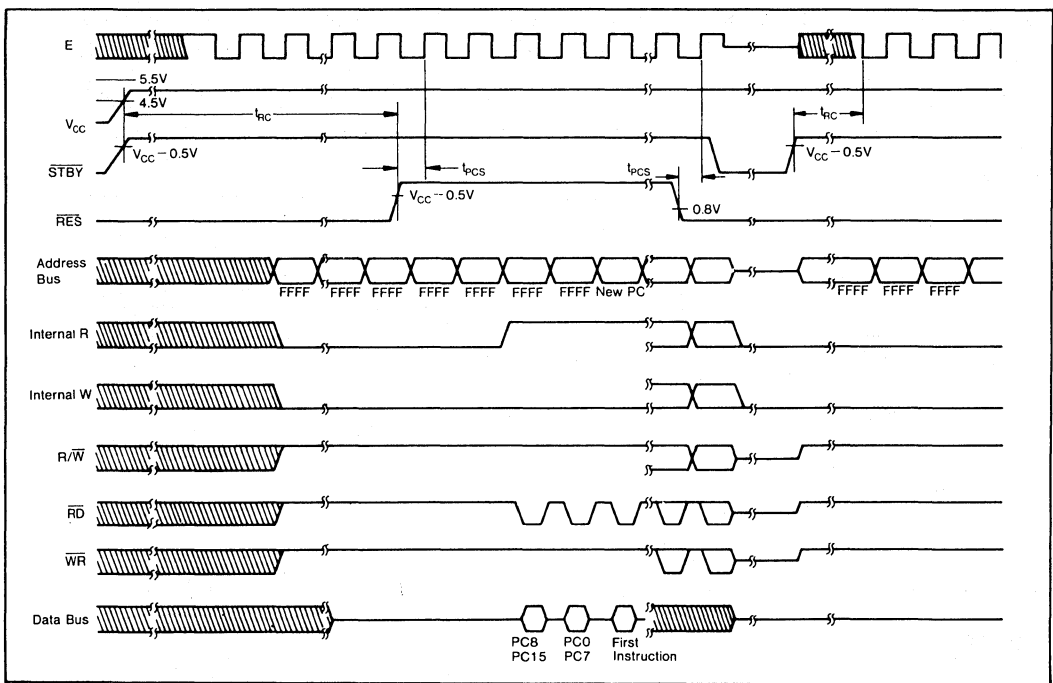


Figure 3-9 Reset Timing

### 3.8 Interrupts

The CPU will complete the current instruction before accepting the request. If the interrupt mask bit in the condition code register is set, the request will be ignored. When the interrupt sequence starts, the contents of the program counter, index register, accumulators, and condition code register will be saved onto the stack. Then the CPU sets the interrupt mask bit and will not respond to further maskable interrupt requests. In the last cycle of the interrupt, the CPU fetches the vectors shown in table 3-9, transfers their contents to the program counter and branches to the interrupt service routine.

The external interrupt pins  $\overline{IRQ}_1$  and  $\overline{IRQ}_2$  are also used as P5<sub>0</sub> and P5<sub>1</sub>. The function is chosen by the enable bits in the RAM/port 5 control register (bits 0 and 1) at \$0014. See 2.5 RAM/Port 5 Control Register for details.

When one of the internal interrupts, ICI, OCI, TOI, CMI, or SIO is generated, the CPU produces the internal interrupt signal, IRQ<sub>3</sub>. IRQ<sub>3</sub> functions just the same as  $\overline{IRQ}_1$  or  $\overline{IRQ}_2$ , except for its vector address. Table 3-9 is an interrupt vector map, figure 3-10 is the interrupt sequence, and figure 3-11 is the interrupt circuit block diagram.

Table 3-9. Interrupt Vector Memory Map

Priority	Vector Location		Interrupt
	MSB	LSB	
Highest ↑ ↓ Lowest	FFFE	FFFF	$\overline{RES}$
	FFEE	FFEF	TRAP
	FFFC	FFFD	$\overline{NMI}$
	FFFA	FFFB	SWI (Software interrupt)
	FFF8	FFF9	$\overline{IRQ}_1$ , ISF (Port 6 input strobe)
	FFF6	FFF7	ICI (Timer 1 input capture)
	FFF4	FFF5	OCI (Timer 1 output compare 1, 2)
	FFF2	FFF3	TOI (Timer 1 overflow)
	FFEC	FFED	CMI (Timer 2 counter match)
	FFEA	FFEB	$\overline{IRQ}_2$
	Lowest	FFF0	FFF1

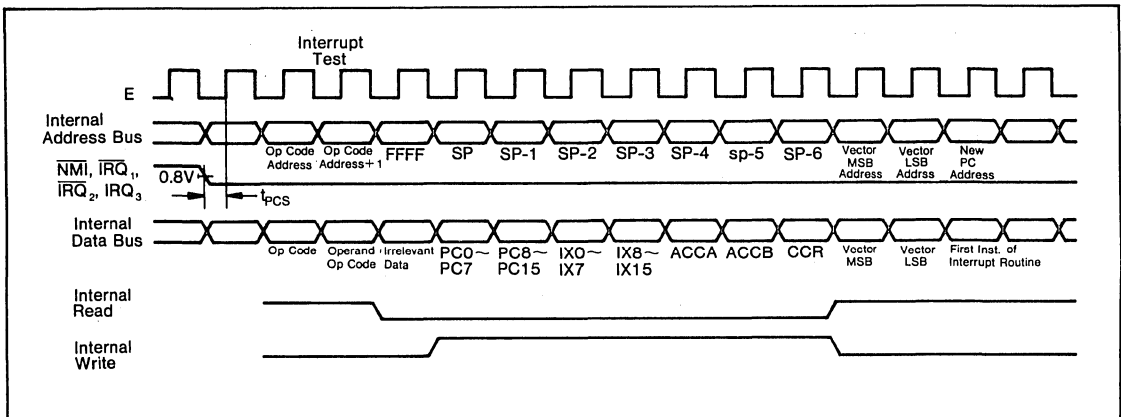


Figure 3-10. Interrupt Sequence





# Section 4. Timer 1

The 16-bit programmable timer, timer 1, can measure an input waveform and independently generate two independent waveforms. The pulse widths of the input and output waveforms can vary from microseconds to seconds.

Timer 1 has the following components (figure 4-1).

- Control/status register 1 (8 bits)
- Control/status register 2 (7 bits)
- Free-running counter (16 bits)
- Output compare register 1 (16 bits)
- Output compare register 2 (16 bits)
- Input capture register (16 bits)

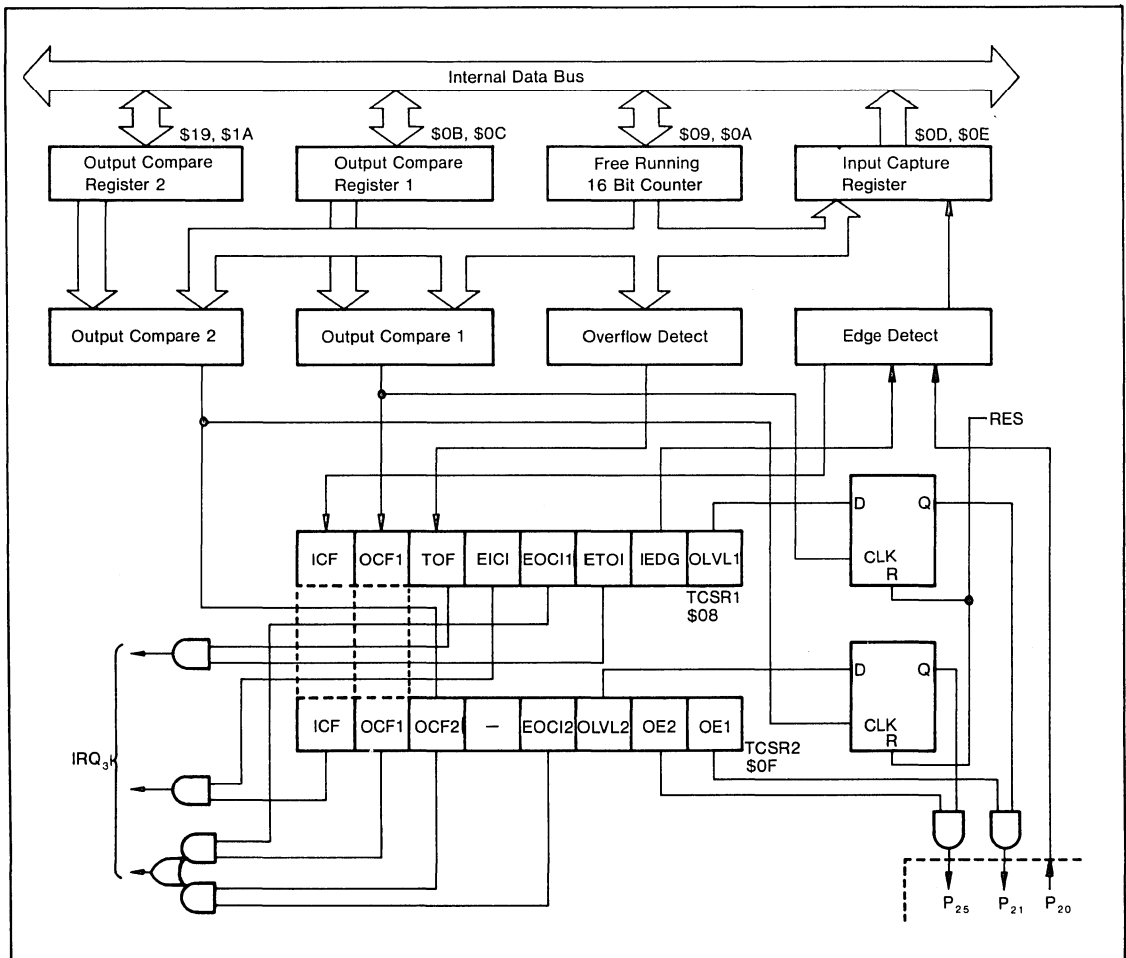


Figure 4-1. Timer 1 Block Diagram



## 4.1 Free-Running Counter (FRC)

The key element of timer 1 is the 16-bit free-running counter. It is incremented by the system clock. The counter value can be read by software without affecting the counter. Reset clears the counter.

The free-running counter is located at addresses \$0009 and \$000A. When the CPU writes to the high byte of the FRC (\$0009), a preset value (\$FFF8) is actually written to both bytes of the counter, regardless of the write data value. When the CPU writes to the low byte (\$000A) after the high byte, both the low and high byte of the write data value are written to the FRC. See figure 4-2. The counter operates this way when written to by double-byte store instructions (STD, STX, etc).

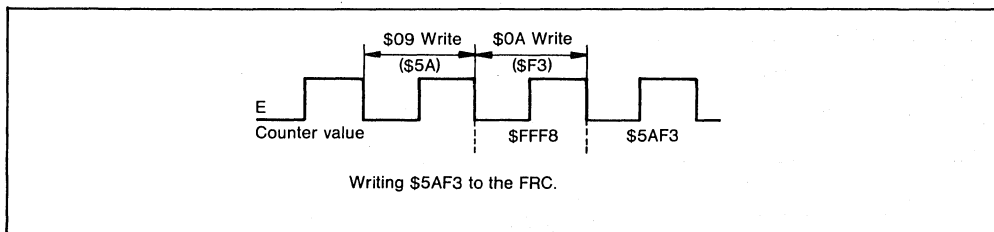


Figure 4-2. Counter Write Timing

## 4.2 Output Compare Registers (OCR)

The output compare registers are 16-bit read/write registers that control the output waveforms. They are located at \$000B, \$000C (OCR1) and \$0019, \$001A (OCR2).

The OCR's are constantly compared to the FRC. When the data matches, the output compare flag (OCF) in the timer control/status register (TCSR) is set. If an output enable bit (OE) in TCSR2 is set to 1, an output level bit (OLVL) will be output to bit 1 (Tout1) and bit 5 (Tout2) of port 2. To determine the output level for the next compare match, change OCR and OLVL.

The OCR is set to \$FFFF after reset. The compare function is inhibited for a cycle just after a write to the OCR or the upper byte of the FRC. This is so that the 16-bit value will be valid in the OCR, and because \$FFF8 is set after the FRC's upper byte is written.

To write to the OCR, use a 2-byte transfer instruction, such as STX.

### 4.3 Input Capture Register (ICR)

The input capture register is a 16-bit read-only register located at \$000D, \$000E. It stores the FRC's value when an external input signal transition at P2<sub>0</sub> generates an input capture pulse. Which transition generates the pulse is defined by the input edge bit (IEDG) in TCSR1.

To input an edge bit to the edge detector, clear bit 0 of port 2's DDR. When an input transition occurs at the next cycle of the CPU's ICR upper-byte read, the input capture pulse will be delayed one cycle. To ensure input capture, the CPU must read the ICR with a 2-byte transfer instruction. The ICR is cleared to all zeros during reset.

### 4.4 Timer Control/Status Register 1 (TCSR1)

The timer control/status register 1 is an 8-bit register located at \$0008 (figure 4-3). All of the bits can be read and the lower 5 can be written to. The 3 upper read-only bits indicate the timer status.

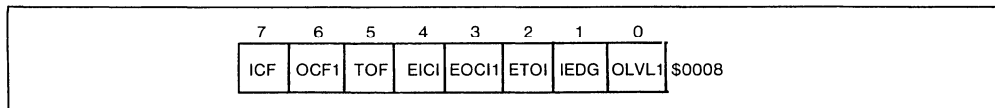


Figure 4-3. Timer Control/Status Register 1

#### 4.4.1 Output Level 1 (OLVL1)

OLVL1 is transferred to port 2, bit 1 when a match occurs between the counter and OCR1. If OE1, bit 0 of TCSR2 is set to 1, OLVL1 will be output at port 2 bit 1. Bit 0.

#### 4.4.2 Input Edge (IEDG)

IEDG determines whether the rising edge or the falling edge of P2<sub>0</sub> will trigger data transfer from the counter to the ICR. IEDG = 0 specifies a falling edge (high to low); IEDG = 1 specifies a rising edge (low to high). Bit 0 of port 2's DDR must be cleared for this function to operate. Bit 1.

#### 4.4.3 Enable Timer Overflow Interrupt (ETOI)

Setting ETOI to 1 enables timer overflow interrupt (TOI) to trigger an internal interrupt (IRQ<sub>3</sub>). When ETOI is cleared, the interrupt is inhibited. Bit 2.

#### 4.4.4 Enable Output Compare Interrupt 1 (EOC1)

Setting EOC1 to 1 enables output compare interrupt 1 (OC1) to trigger an internal interrupt (IRQ<sub>3</sub>). When EOC1 is cleared, the interrupt is inhibited. Bit 3.

#### 4.4.5 Enable Input Capture Interrupt (EICI)

Setting EIC1 to 1 enables input capture interrupt (ICI) to trigger an internal interrupt (IRQ<sub>3</sub>). When EIC1 is cleared, the interrupt is inhibited. Bit 4.

#### 4.4.6 Timer Overflow Flag (TOF)

TOF is set when the counter value increments from \$FFF to \$0000. TOF is cleared when CPU reads the TCSR1, then the counter's upper byte (at \$0009). Bit 5, read only.

#### 4.4.7 Output Compare Flag 1 (OCF1)

OCF1 is set when a match has occurred between the FCR and OCR1. Writing to OCR1 (\$000B or \$000C) after reading the TCSR1 or TCSR2 clears OCF1. Bit 6, read only.

#### 4.4.8 Input Capture Flag (ICF)

ICF is set when the transition of the P2<sub>0</sub> input signal selected by IEDG causes the counter to transfer its data to the ICR. Reading the high byte of the ICR (\$000D) after reading TCSR1 or TCSR2 clears ICF. Bit 7, read only.

### 4.5 Timer Control/Status Register 2 (TCSR2)

The timer control/status register 2 is a 7-bit register located at \$000F (figure 4-4). All of the bits can be read and the lower 4 can be written to. The 3 upper read-only bits indicate the timer status.

Both TCSR1 and TCSR2 are cleared during reset.

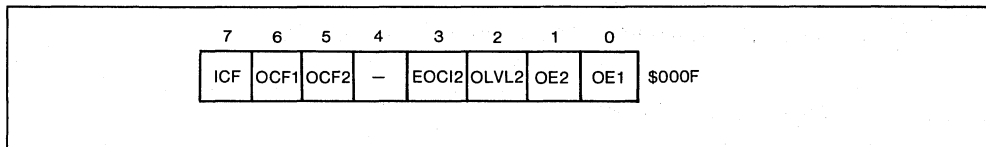


Figure 4-4. Timer Control/Status Register 2

#### 4.5.1 Output Enable 1 (OE1)

Setting OE1 to 1 enables OLVL1 to appear at P2<sub>1</sub> when a match has occurred between the counter and the output compare register 1 (OCR1). Clearing OE1 makes P2<sub>1</sub> an I/O port. Bit 0.

#### 4.5.2 Output Enable 2 (OE2)

Setting OE2 to 1 enables OLVL2 to appear at P2<sub>5</sub> when a match occurs between the counter and the output compare register 2 (OCR2). Clearing OE2 makes P2<sub>5</sub> an I/O port. Bit 1.

Note: If OE1 or OE2 is set to 1 before the first output compare match after reset, P2<sub>1</sub> or P2<sub>5</sub> will output 0.

#### 4.5.3 Output Level 2 (OLVL2)

OLVL2 is transferred to P2<sub>5</sub> when a match occurs between the counter and OCR2. If OE2 (bit 1 of TCSR2) is set to 1, OVLV2 will be output at P2<sub>5</sub>. Bit 2.

#### 4.5.4 Enable Output Compare Interrupt 2 (EOCI2)

Setting EOCI2 to 1 enables output compare interrupt 2 (OCI2) to trigger an internal interrupt (IRQ<sub>3</sub>). When EOCI2 is cleared, the interrupt is inhibited. Bit 3.

#### 4.5.5 Output Compare Flag 2 (OCF2)

OCF2 is set when a match has occurred between the FCR and OCR2. Writing to OCR2 (\$0019 or \$001A) after reading TCSR2 clears OCF1. Bit 6, read only.

#### 4.5.6 Output Compare Flag 1 (OCF1) and Input Capture Flag (ICF)

The OCF1 and ICF addresses are partially decoded. The CPU reading TCSR1/TCSR2 makes it possible to read OCF1 and ICF into bits 6 and 7.

## 4.6 Timer Status Flags

Table 4-1 shows set and clear conditions of each status flag in timer 1.

If flag set and clear conditions occur at the same time, timer 1 flags will be set.

Table 4-1 Timer 1 Status Flags

Flag	Set Condition	Clear Condition
Timer 1 ICF	· FRC → ICR at edge of P2 <sub>0</sub>	· Read TRCSR1 or TRCSR2, then ICR <sub>H</sub> · $\overline{RES} = 0$
OCF1	· OCR1 = FRC	· Read TRCSR1 or TRCSR2, then write OCR1 <sub>H</sub> or OCR1 <sub>L</sub> · $\overline{RES} = 0$
OCF2	· OCR2 = FRC	· Read TRCSR2, then write OCR2 <sub>H</sub> or OCR2 <sub>L</sub> · $\overline{RES} = 0$
TOF	· FRC = \$FFFF + 1 cycle	· Read TRCSR, then FRC <sub>H</sub> · $\overline{RES} = 0$

## 4.7 Precautions on Clearing OCF

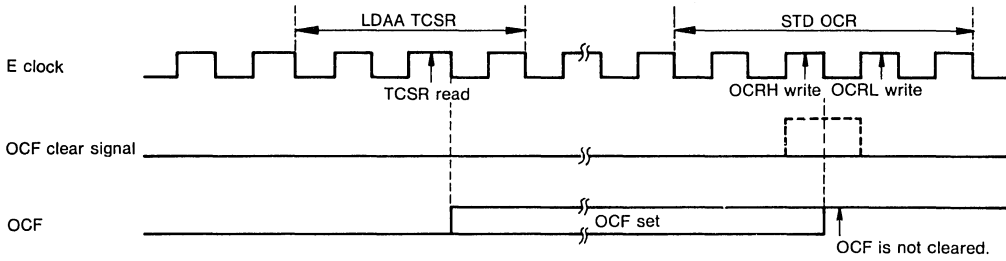
Writing to the OCR after reading the TCSR when the OCF is 1 clears the OCF. However, the OCF is not cleared under the following conditions.

1. A compare match is found before the CPU writes to the OCR after reading the TCSR with OCF = 0.
2. A compare match is found at the same time as the CPU writes to the OCR after reading the TCSR with OCF = 1.

See figure 4-5.

The OCF will always be cleared if you assure that a compare match does not occur between the TCSR read and the OCR write. In example 1, figure 4-6, the OCR is loaded with the contents of the free-running counter (FRC) before the TCSR is read. A compare match will not occur until the FRC is counted up. In example 2, an OCR write cycle is executed immediately before and after TCSR read. A compare match will not occur until a match occurs between the contents of the FRC and the OCR write data.

1. When OCF is not cleared  
 (1) case 1



(2) case 2

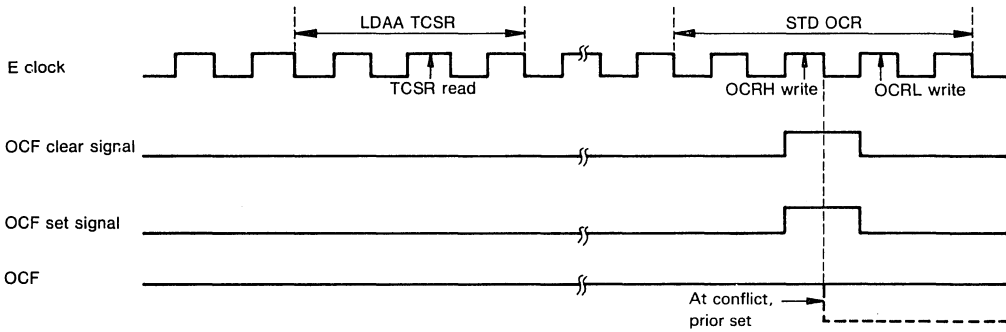


Figure 4-5. OCF Clearing Timing on Condition

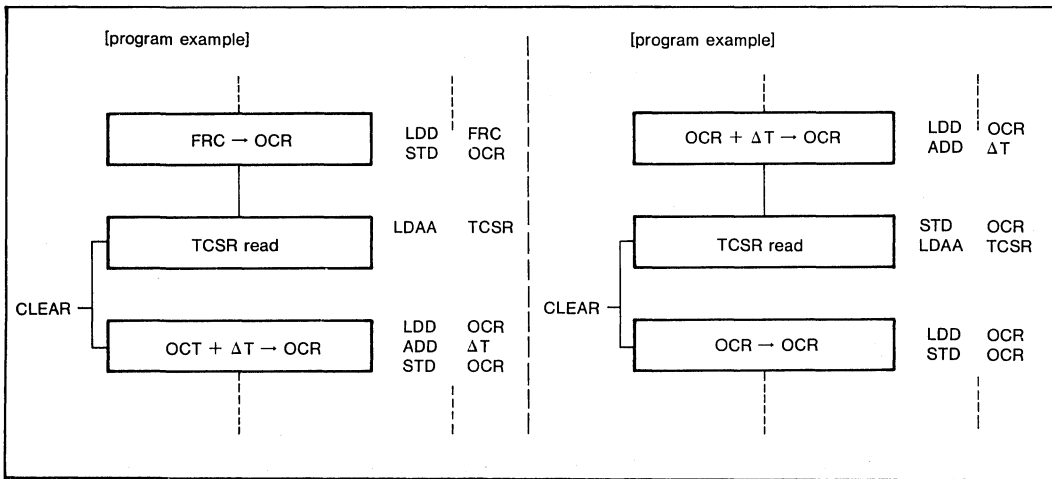


Figure 4-6 Clearing the OCF





If the CPU writes to the counter during a cycle when it is being cleared, it will not be cleared, but will take the value written by the CPU.

## 5.2 Time Constant Register (TCNR)

The 8-bit write-only time constant register is located at \$001C. It is always being compared to the upcounter.

When it matches, the counter match flag (CMF) of the timer control/status register 3 (TCSR3) is set. P2<sub>6</sub> will then output the value selected by TOS0 and TOS1 of the TCSR3. When the CMF is set, the counter will be cleared simultaneously and start counting from \$00. This enables regular interrupts and waveform output without any software attention. TCONR is set to \$FF during reset.

When a write-only register like TCONR is read by the MCU, \$FF always appears on the data bus. Whenever the MCU performs an arithmetic or logic operation between memory, and a write-only register, the result will be \$FF.

## 5.3 Timer Control/Status Register 3 (TCSR3)

The 7-bit timer control/status register is located at \$001B (figure 5-2). All bits can be read and all bits can be written except CMF (bit 7). TCSR3 is cleared at reset.

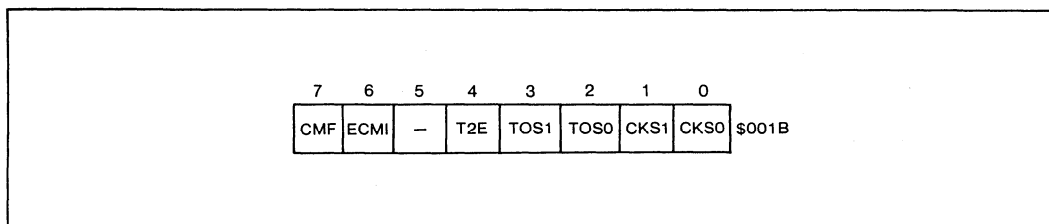


Figure 5-2. Timer Control/Status Register 3

### 5.3.1 Input Clock Select 0 and 1 (CKS0, CKS1)

CKS0 and CKS1 select the counter clock as shown in table 5-1. When the external clock is selected, the rising edge of P2<sub>7</sub> increments the counter. The external clock's frequency can be up to one-half the system clock frequency. If the E clock divided by 8 or 128 is selected, the clock comes from timer 1, so do not write to the FRC. Bits 0 and 1.

Table 5-1. Input Clock Select

CKS1	CKS0	Input Clock
0	0	E clock
0	1	E/8
1	0	E/128
1	1	External clock (P2 <sub>7</sub> )

### 5.3.2 Timer Output Select 0 and 1 (TOS0, TOS1)

When the upcounter matches TCONR, timer 2 will output to P2<sub>6</sub> as selected by TOS0 and TOS1 (table 5-2). When TOS0 and TOS1 are 0, P2<sub>6</sub> will be an I/O port. When toggle output is selected, the P2<sub>6</sub> output level reverses each time the upcounter and TCONR match. This produces a 50% duty cycle square wave at P2<sub>6</sub> without software support. Bits 2 and 3.

Table 5-2. Timer 2 Output Select

TOS1	TOS0	Timer Output
0	0	Timer output inhibited
0	1	Toggle output
1	0	Output 0
1	1	Output 1

### 5.3.3 Timer 2 Enable (T2E)

When T2E is cleared to 0, the clock input to the upcounter is inhibited, and the upcounter stops. When T2E is set, the clock selected by CKS0 and CKS1 is input to the upcounter. Bit 4.

Note: P2<sub>6</sub> outputs 0 when T2E bit is 0 and timer 2 is enabled by TOS0 and TOS1. It also outputs 0 when T2E is 1 and timer 2 is output enabled before the first match occurs.

### 5.3.4 Enable Counter Match Interrupt (ECMI)

Setting ECMI to 1 enables CMI to trigger an internal interrupt (IRQ<sub>3</sub>). When ECMI is cleared, the interrupt is inhibited. Bit 6.

### 5.3.5 Counter Match Flag (CMF)

The read-only CMF bit is set when the upcounter matches the TCONR. It is cleared by writing a zero to it. Bit 7.

## 5.4 Timer Status Flag

Table 5-3 shows set and clear condition of each status flag in timer 2.

If flag set and clear condition occurs at the same time, timer 2 flag will be set.

Table 5-3. Timer 2 Status Flag

Flag	Set Condition	Clear Condition
CMF	· T2CNT = TCONR	· Write 0 to CMF · $\overline{\text{RES}} = 0$

## 5.5 Precaution for toggle pulse function of HD6301Y0/HD6303Y/HD63701Y0 Timer 2

Please pay attention to the following items when using Timer 2 as Toggle pulse output function.

### PHENOMENAN

Just when T2CNT's content equals a TCR's content, after writing "1" to T2E bit of TCSR3 to output toggle pulses from P26, the abnormal rising edge occurs at P26 and the first pulse width will be 1/2 E clock cycle.

Therefore, in the application which needs off-and-on pulse groups, you can't get 50%-duty output pulse at anytime.

Timing chart of Timer 2 and P26 is shown in Fig. 1 when  $TCR = N$  and  $CKS = 0$   $CKS1 = 0$  to select E-clock as input pulse.

Case ① is normal operation because  $T2CNT \neq TCR$ . Case ② is abnormal operation because  $T2CNT = TCR$ , and 1/2 E pulse is generated.

### CAUTION

In the case of outputting the off-and-on toggle pulse from P26, please write "1" to T2E bit of TCSR3, when content of T2CNT isn't equal to one of TCR. To realize above method, please write "1" to T2E bit after writing "00" to T2CNT.

Explanation T2CNT: Timer 2 Up Counter T2E: Timer 2 Enable bit  
TCR: Timer Constant Resister

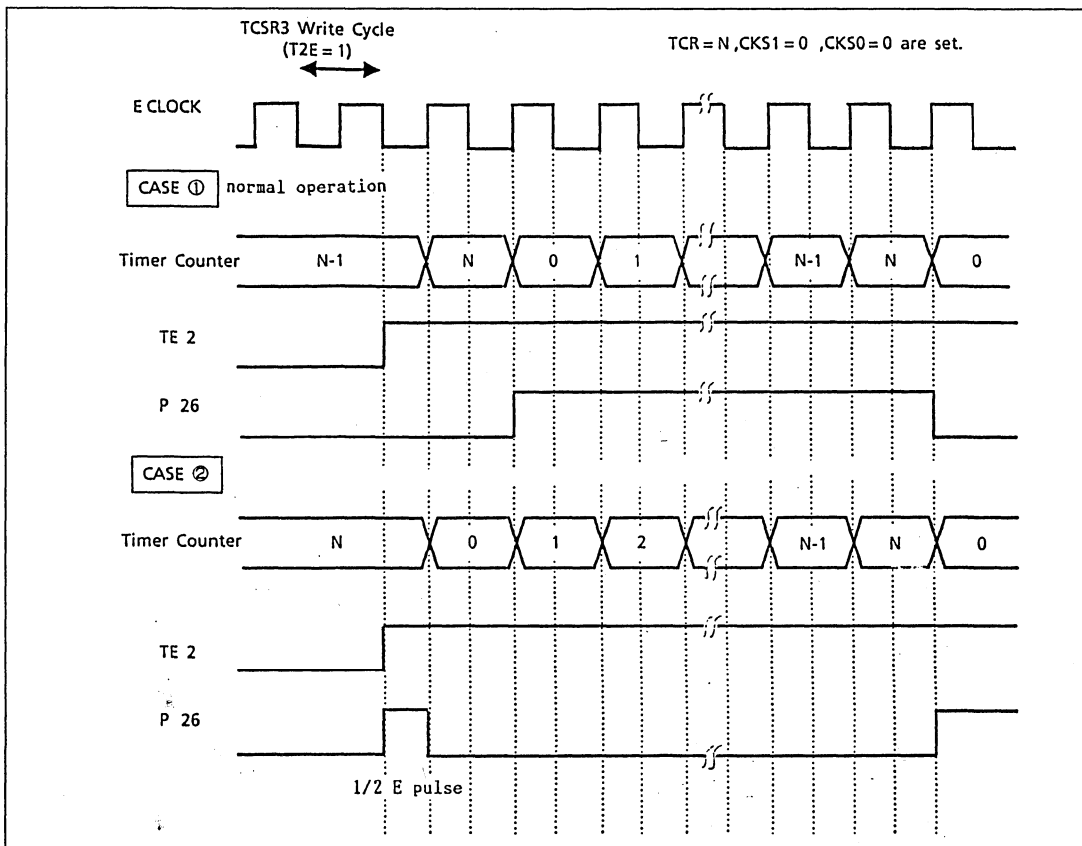


Figure 1. The Timing Chart of Timer 2 Counter and P26 (T OUT3)

## Section 6. Serial Communications Interface

The serial communications interface (SCI) has two operating modes: asynchronous with NRZ format, and clock synchronous. The synchronous mode transfers data synchronized with the serial clock.

The SCI has the following components (figure 6-1).

- Transmit/receive control/status register 1 (TRCSR1)
- Transmit/receive control/status register 2 (TRCSR2)
- Rate/mode control register (RMCR)
- Receive data register (RDR)
- Receive data shift register (RDSR)
- Transmit data register (TDR)
- Transmit data shift register (TDSR)

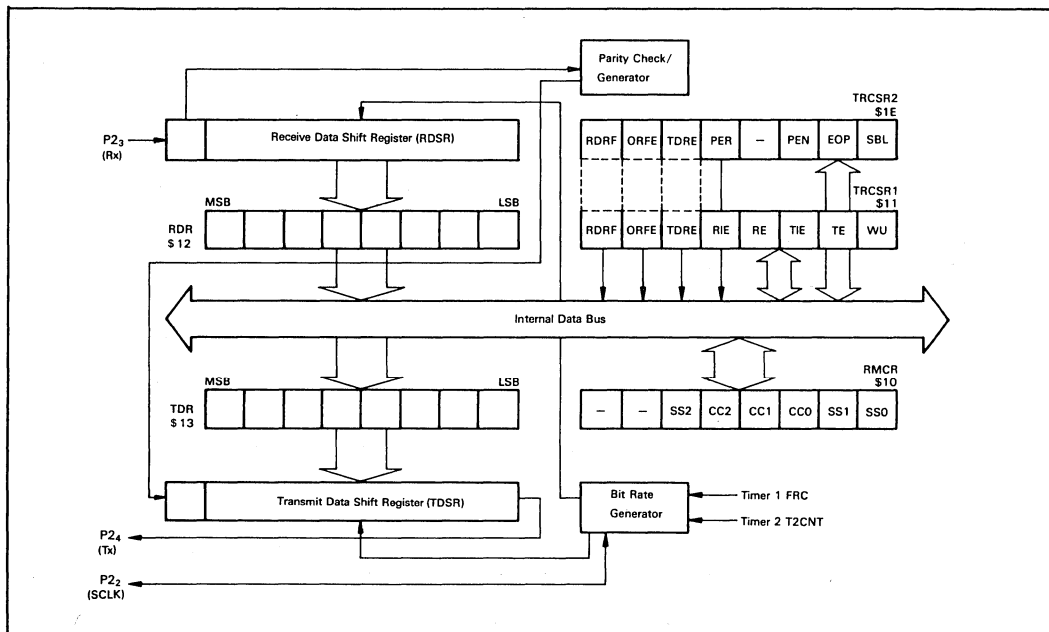


Figure 6-1. SCI Block Diagram

### 6.1 Initialization

The serial I/O hardware must be initialized by the software for operation. The usual procedure follows.

1. Write the desired operating mode to the RMCR.
2. Write the desired operating mode to the TRCSR.

The TE and RE bits may only be set when P2<sub>3</sub> and P2<sub>4</sub> are used for serial I/O only. But TE and RE should be 0 when you set the baud rate and operating mode. Clearing and setting TE and RE again must take more than one cycle at the current baud rate. If they are set within less than 1 cycle, transmit/receive initialization may fail.

## 6.2 Asynchronous Mode

The asynchronous mode has two data formats:

- 1 start bit + 8 data bits + 1 stop bit
- 1 start bit + 9 data bits + 1 stop bit

In addition, the ninth bit can be set to 1 in the 9-bit format to form a third format:

- 1 start bit + 8 data bits + 2 stop bits

### 6.2.1 Asynchronous Transmission

Setting TE in the TRCSR enables transmission. P2<sub>4</sub> becomes the serial output port regardless of the state of bit 4 of the DDR.

Both RMCR and TRCSR should be set to the desired operating conditions. When TE is set, a 10-bit preamble (8-bit format) or 11-bit preamble (9-bit format) will be sent. When it is being sent, internal synchronization will stabilize, and the transmitter will become ready to send.

At this point, if the TDR is empty (TDRE = 1), all 1's will be output, to indicate the idle state. If the TDR contains data (TDRE = 0), the data is sent to the transmit data shift register, and transmission begins.

During transmission, first a 0 start bit is sent. Then 8 or 9 bits of data, starting at bit 0, are transmitted, followed by a stop bit of 1.

When the TDR is empty, hardware sets the TDRE flag bit. If the CPU doesn't respond to the TDRE flag before the next normal transfer should start, the transmitter sends 1's (instead of the 0 start bit) until data is provided to the data register. While the TDRE is set, the transmitter will not send a 0.

### 6.2.2 Asynchronous Reception

Setting the RE bit enables reception. P2<sub>3</sub> becomes the serial input port, regardless of the state of bit 3 of the DDR. The contents of TRCSR and RMCR select the data receive operating mode. The first 0 (space) synchronizes the receive bit flow. Each bit of the following data will be strobed in the middle.

If the stop bit is not 1, the receiver assumes a framing error, and sets ORFE. When a framing error occurs, the receiver transfers the data to the receive data register and the CPU can read the data that caused the error. This makes it possible to detect line breaks.

If the stop bit is 1, the data is transferred to the receive data register and the interrupt flag RDRF is set. If the RDRF is still set when the stop bit of the next data is received, the receiver sets ORFE to indicate an overrun.

When the CPU reads the RDR in response to the RDRF or ORFE in the TRCSR, the RDRF or ORFE bit is cleared to 0.

### 6.2.3 Asynchronous Clock Source

When using an internal clock for asynchronous serial I/O, keep the following in mind:

- Set CC1 and CC0 to 1 and 0, respectively (table 6-3).
- A clock will be generated regardless of the value of TE and RE.
- The maximum clock rate is E/16.
- The output clock rate is the same as the bit rate.

When using an external clock, keep the following in mind:

- Set CC1 and CC0 to 1 and 1, respectively.
- The external clock frequency should be set to 16 times the baud rate.
- Maximum clock frequency is that of the system clock

## 6.3 Clock Synchronous Mode

In the clock synchronous mode, data transmission is synchronized with a clock pulse. The SCI has a fully independent transmitter and receiver, which make full duplex asynchronous operation possible.

Therefore, in synchronous mode, the only clock I/O pin is P2<sub>2</sub>, so simultaneous transmit and receive is not available. In synchronous mode, TE and RE should not be set to 1 at the same time. Figure 6-2 is the clock and data format for synchronous mode.

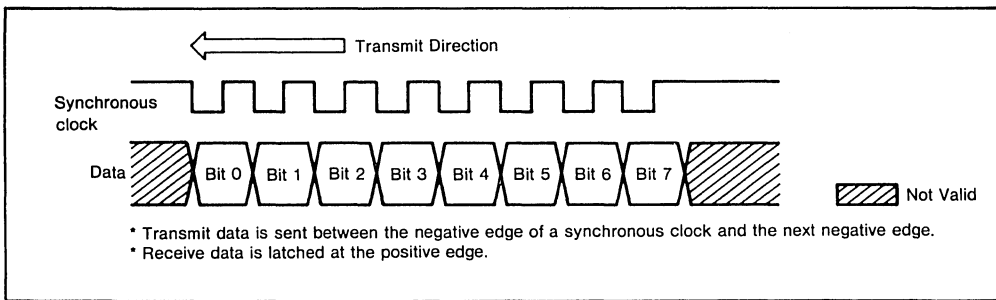


Figure 6-2. Clock Synchronous Mode

### 6.3.1 Synchronous Transmission

Setting the TE bit in the TRCSR enables transmission. P2<sub>4</sub> becomes the serial output port regardless of bit 4 in the DDR. Both the TRCSR and RMCR should be set to the desired operating conditions for transmission.

When external clock input is selected, data is transmitted under the TDRE flag 0 from P2<sub>4</sub>, synchronized with 8 clock pulses input to P2<sub>2</sub>. Data is transmitted from bit 0, and TDRE is set when the transmit data shift register is empty. More than 8 external clock pulses are ignored.

When the transmitter is selected to output the clock, the SCI outputs the clock and synchronous data when the TDRE flag is cleared.

### 6.3.2 Synchronous Reception

Setting the RE bit enables data reception. P2<sub>3</sub> becomes the serial input port regardless of bit 3 in the DDR. TRCSR and RMCR select the data reception operating mode.

If external clock input is selected, the RE bit should be set while the clock signal at P2<sub>2</sub> is high. After the RE bit is set, 8 external clock pulses and synchronized bits of receive data are input at P2<sub>2</sub> and P2<sub>3</sub> respectively. The SCI puts a bit of data into the receive data shift register at every clock pulse, and sets the RDRF flag after 8 bits have been received. More than 8 pulses are ignored. When the CPU reads the received data, RDRF is cleared, and the SCI starts receiving the next data. Clear RDRF when P2<sub>2</sub> is high.

When the receiver is selected to output the clock, 8 clocks are output to P2<sub>2</sub> when the RE bit is set. The receive data should appear at P2<sub>3</sub> synchronously with this clock. When the first byte of data is received, the SCI sets the RDRF flag. To receive the next byte, clear the RDRF flag to start the clock and start receiving.



## 6.4 Transmit/Receive Control Status Register (TRCSR)

The TRCSR is located at \$0011 (figure 6-3). All 8 bits can be read, and bits 0-4 can be written to. TRCSR is initialized to \$20 during reset.

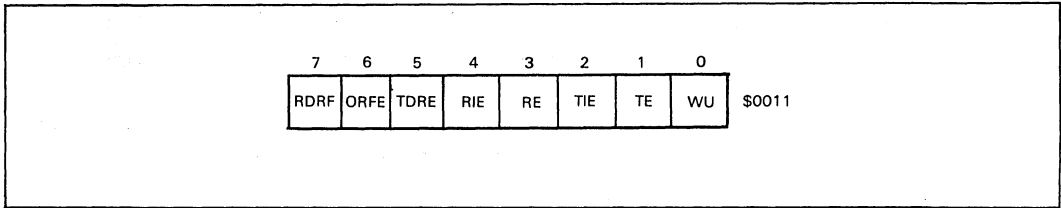


Figure 6-3. Transmit/Receive Control Status Register

### 6.4.1 Wake-Up (WU)

In a typical multiprocessor configuration, the software protocol provides the destination address as the first byte of a message. The wake-up function allows uninterested MCU's to ignore the rest of the message. When the WU bit is set, the SCI stops receiving data until the next message.

The wake-up function is triggered by one frame length of consecutive 1's (10 bits for 8-bit data, 11 bits for 9-bit data). This function is only available in asynchronous mode. Do not set WU in clock synchronous mode. Receiving these consecutive 1's wakes up the SCI and clears WU. The SCI starts receiving data. The RE flag should be set before WU is set. Bit 0.

### 6.4.2 Transmit Enable (TE)

When TE is set, transmit data will appear at P2<sub>4</sub> after a 1-frame preamble in asynchronous transmission, or immediately in clock synchronous transmission. P2<sub>4</sub> will be the serial output regardless of the state of bit 4 of port 2's DDR. If TE is cleared, serial I/O doesn't affect P2<sub>4</sub>. Bit 1.

### 6.4.3 Transmit Interrupt Enable (TIE)

Setting TIE enables TDRE to trigger an internal interrupt (IRQ<sub>3</sub>). Clearing TIE inhibits the interrupt. Bit 2.

### 6.4.4 Receive Enable (RE)

Setting RE inputs the signal at P2<sub>3</sub> regardless of the state of bit 3 of port 2's DDR. When RE is cleared, serial I/O doesn't affect P2<sub>3</sub>. Bit 3.

### 6.4.5 Receive Interrupt Enable (RIE)

Setting RIE enables RDRF or ORFE (TRCSR bit 6 or 7) to trigger an internal interrupt (IRQ<sub>3</sub>). Clearing RIE inhibits the interrupt. Bit 4.

### 6.4.6 Transmit Data Register Empty (TDRE)

In asynchronous mode, the SCI sets TDRE when the TDR is transferred to the TDSR. In the clock synchronous mode, SCI sets TDRE when the TDSR is empty. TDRE is reset by reading the TRCSR and writing new transmit data to the transmit data register. TDRE is set to 1 at reset. Bit 5, read only.

### 6.5.7 Overrun/Framing Error (ORFE)

The SCI sets ORFE when an overrun or framing error is generated during data receive. An overrun error occurs when new receive data is ready to be transferred to the RDR, and RDRF is still set. A framing error occurs when a stop bit is not 0. ORFE is only affected in asynchronous mode. Reading the RDR after reading the TRCSR clears the ORFE. It is cleared at reset. Bit 6, read only.

### 6.4.8 Receive Data Register Full (RDRF)

RDRF is set when the RDSR is transferred to the RDR. Reading the RDR after reading the TRCSR clears the RDRF. It is cleared at reset. Bit 7, read only.

Note: When more than 1 of bits 5, 6, and 7 are set, one TRCSR read will clear them all. It is not necessary to read the TRCSR once for each bit.

## 6.5 Transmit Rate/Mode Control Register (RMCR)

The RMCR (figure 6-4) controls the following for serial I/O:

- Baud rate
- Clock source
- Operation mode
- Data format
- P2<sub>2</sub> function

In addition, if the 9-bit asynchronous format is used, RMCR holds the ninth bit. All bits can be read, and all bits can be written to, except bit 7 (RD8).

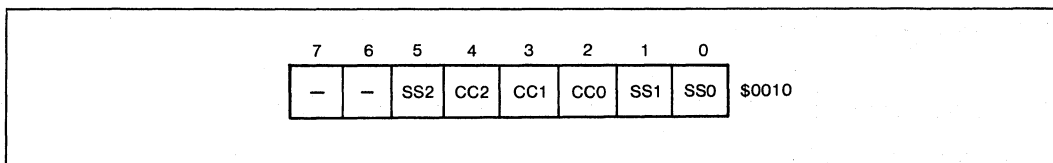


Figure 6-4. Transfer Rate/Mode Control Register

### 6.5.1 Speed Select (SS0, SS1, SS2)

SS0-SS2 control the baud rate used for the SCI. Table 6-1 lists the available baud rates. The timer 1 FRC (SS2 = 0) and the timer 2 upcounter (SS2 = 1) provide the internal clock to the SCI. When SS2 is set, timer 2 functions as the baud rate generator. Timer 2 generates a baud rate dependent on TCONR as shown in table 6-2. Bits 0, 1, and 5.

Table 6-1. SCI Bit Times and Transfer Rates

#### Asynchronous

SS0	SS1	SS2	XTAL	2.4576 MHz	4.0 MHz	4.9152 MHz
			E	614.4 kHz	1.0 MHz	1.2288 MHz
0	0	0	E/16	26 $\mu$ s/38400 baud	16 $\mu$ s/62500 baud	13 $\mu$ s/76800 baud
0	0	1	E/128	208 $\mu$ s/4800 baud	128 $\mu$ s/7812.5 baud	104.2 $\mu$ s/9600 baud
0	1	0	E/1024	1.67 ms/600 baud	1.024 ms/976.6 baud	833.3 ms/1200 baud
0	1	1	E/4096	6.67 ms/150 baud	4.096 ms/244.1 baud	3.333 ms/300 baud
1	X	X		Note 1	Note 1	Note 1

Note:

- When SS2 = 1, timer 2 is the SCI clock. The baud rate is as follows:  
 Baud rate =  $f/[32(TCONR + 1)]$   
 Where:  
 f = timer 2 input clock frequency  
 TCONR = contents of timer constant register, 0-255

Table 6-1. SCI Bit Times and Transfer Rates (cont.)

**Clock Synchronous (Note 1)**

SS2	SS1	SS0	XTAL	4.0 MHz	6.0 MHz	8.0 MHz	12.0 MHz
			E	1.0 MHz	1.5 MHz	2.0 MHz	3.0 MHz
0	0	0	E/2	2 μs/bit	1.33 μs/bit	1 μs/bit	0.667 μs/bit
0	0	1	E/16	16 μs/bit	10.7 μs/bit	8 μs/bit	5.33 μs/bit
0	1	0	E/128	128 μs/bit	85.3 μs/bit	64 μs/bit	42.7 μs/bit
0	1	1	E/512	512 μs/bit	341 μs/bit	256 μs/bit	171 μs/bit
1	X	X		Note 2	Note 2	Note 2	Note 2

Notes:

1. Bit rates for internal clock operation. External clock can operate from DC to 1/2 system clock frequency.
2. When SS2 is 1, timer 2 is the SCI clock. The bit rate is as follows:  
 Bit rate (μs/bit) = 4(TCONR + 1)/f  
 Where:  
 f = timer 2 input clock frequency  
 TCONR = contents of timer constant register, 0-255

Table 6-2. Baud Rate and Time Constant Register Example

Baud Rate	XTAL Frequency					
	2.4576 MHz	3.6864 MHz	4.0 MHz	4.9152 MHz	8.0 MHz	12.0 MHz
110 (note 1)	21	32	35	43	70	106
150	127	191	207	255	51	77
300	63	95	103	127	207	38
600	31	47	51	63	103	155
1200	15	23	25	31	51	77
2400	7	11	12	15	25	38
4800	3	5		7	12	19
9600	1	2		3		9
19200	0			1		
38400				0		

Note:

1. E/8 is used as the clock for 110 baud, E is used for all other baud rates.

## 6.5.2 Clock Control/Format Select (CC0, CC1, CC2)

CC0, CC1, and CC2 control the clock source and data format (table 6-3). They are cleared during reset, so the MCU will be in clock synchronous mode with external clock. Therefore, P2<sub>2</sub> starts out as a clock input. To use P2<sub>2</sub> as an output port, set bit 2 of the port 2 DDR to 1 and set CC1 and CC0 to 0, 1. Bits 2, 3, and 4.

Table 6-3. SCI Format and Clock Source Control

CC2	CC1	CC0	Format	Mode	Clock Source	P2 <sub>2</sub>
0	0	0	8-bit data	Clock synchronous	External	Clock input
0	0	1	8-bit data	Asynchronous	Internal	Not used
0	1	0	8-bit data	Asynchronous	Internal	Clock output (note 1)
0	1	1	8-bit data	Asynchronous	External	Clock input
1	0	0	8-bit data	Clock synchronous	Internal	Clock output
1	0	1	7-bit data	Asynchronous	Internal	Not used
1	1	0	7-bit data	Asynchronous	Internal	Clock output (note 1)
1	1	1	7-bit data	Asynchronous	External	Clock input

Note:

1. Clock output regardless of bits TE and RE of TRCSR.

## 6.6 SCI Receiving Margin

The receiving margin for the SCI is as follows.

Allowable deviation of bit error  $(t - t_0)/t_0 = \pm 43.7\%$

Allowable deviation of character error  $(T - T_0)/T_0 = \pm 4.37\%$

T, T<sub>0</sub>, t, and t<sub>0</sub> are defined in figure 6-5. When a modem is used for communication, waveform distortion may exceed the allowable value, depending on the modem and channel.

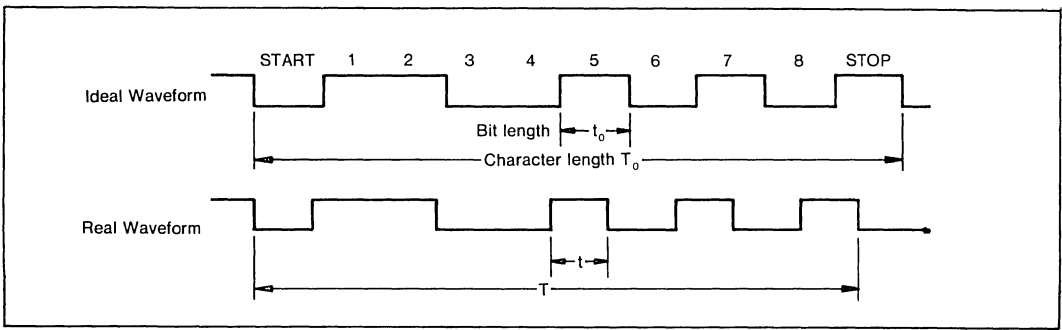


Figure 6-5. Bit and Character Error

## 6.7 SCI Status Flags

Table 6-4 shows set and clear conditions of each status flag in the SCI.

If flag set and clear conditions occur at the same time, the SCI flags will be cleared.

Table 6-4. SCI Status Flags

Flag	Set Condition	Clear Condition
SCI	RDRF · RDSR → RDR	· Read TRCSR1 or TRCSR2, then RDR · $\overline{RES} = 0$
ORFE	· Framing error (async mode). Stop bit = 0 · Overrun error (async mode). RDSR → RDR when RDRF = 1	· Read TRCSR1 or TRCSR2, then RDR · $\overline{RES} = 0$
TDRE	· TDR → TDSR (async mode) · TDSR is empty (clock sync mode)	· Read TRCSR1 or TRCSR2, then write to TDR
PER	· PEN = 1	· Read TRCSR2, then RDR $\overline{RES} = 0$

## 6.8 Precaution for clock-synchronous serial communication interface

When transmitting through clock-synchronous serial communication interface, TE bit should not be cleared with TDRE of TRCSR (\$11) is "0".

The TDRE set and clear conditions of SCI are shown as follows.

	Set Condition	Clear Condition
	1. TDR → transmit shift register (asynchronous)	When writing to TDR after TRCSR read, with TDRE = 1, TDRE is cleared.
TDRE	2. Transmit shift register is empty. (clock-synchronous)	
	3. $\overline{RES} = 0$	

If transmit data is written to TDR, and then TE bit is cleared with TDRE = 0 to stop transmitting, TDRE remains "0".

In this case, even if TE bit is set and transmit data is written again, the TDR data is not transmitted.

Please note that TE bit must be cleared after the last data has been transmitted.

(This caution is not applied to asynchronous serial communication interface.)

# Section 7. HD63701Y0 Programmable ROM (EPROM)

## Programmable ROM Operation

The HD63701Y0's on-chip EPROM is programmed in the PROM mode (figures 37 and 38). PROM mode is set by bringing  $MP_0$ ,  $MP_1$ , and  $STBY$  low. In PROM mode, the MCU doesn't operate. It can be programmed like a

standard 27256 EPROM using a standard PROM programmer and a socket adapter. Table 18 lists recommended PROM programmers and socket adapters.

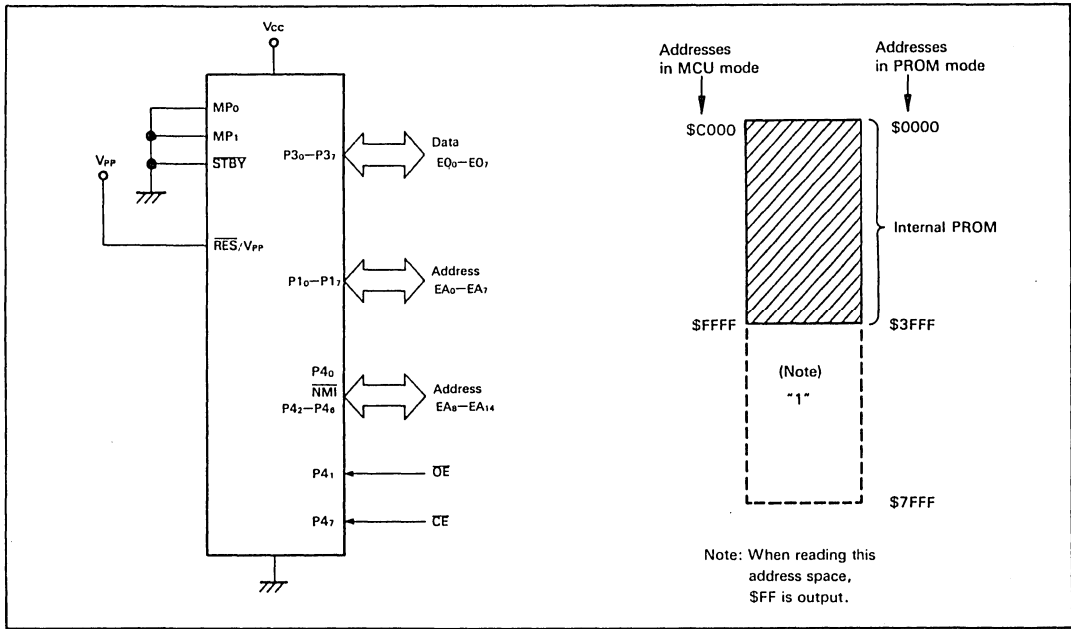


Figure 7.1 PROM Mode Functional Diagram and Memory Map

Table 7.1 PROM Programmers and Socket Adapters

PROM Programmer		Socket Adapter			
Maker	Type Name	Maker	Type Name		
DATA I/O			DP-64S, DC-64S	FP-64	CP-68
	121B	Hitachi	HS31YESS11H	HS31YESF01H	HS31YESC01H
	22B				
	29B				

Table 7.2 PROM Mode Selection

Mode	Pin			
	$\overline{CE}$	$\overline{OE}$	$V_{pp}$	$E_{0}-E_{7}$
Programming	Low	High	$V_{pp}$	Data input
Verify	High	Low	$V_{pp}$	Data output
Programming inhibited	High	High	$V_{pp}$	High impedance



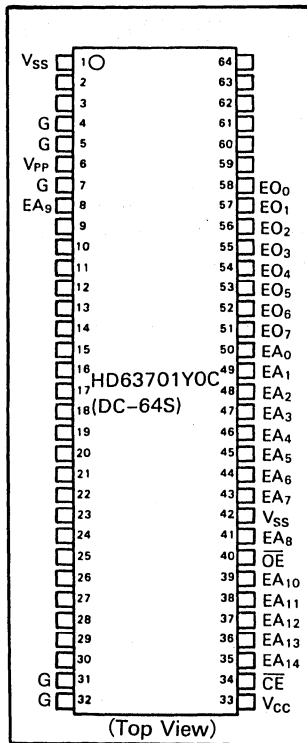


Figure 7-2. PROM Mode Pin Arrangement

## 7.1 Programming and Verification

When the  $\overline{CE}$  pin is held low after the programming voltage ( $V_{PP}$ ) is applied, data can be programmed in PROM one byte at a time through port 3. To verify the data, hold the  $V_{PP}/\overline{OE}$  and  $\overline{CE}$  pins low after programming, and the programmed data will be output from port 3.

When  $\overline{CE}$  is returned high, port 3 will be high impedance, and PROM programming/verification will be inhibited.

**Programming precautions:** The PROM memory cells should be programmed under specific voltage and timing conditions. The higher the program voltage and the longer the program pulse is applied, the more electrons will be injected into the floating gate. However, if an overvoltage is applied to  $V_{PP}$ , the p-n junction may be permanently damaged. Pay particular attention to PROM programmer overshoot. Negative voltage noise will cause a parasitic transistor effect, which may reduce breakdown voltage. The address range must be \$000 through \$3FFF because the on-chip EPROM is 16K bytes. Fill remainder of EPROM area with FFFF for PROM programmer to correctly verify.

The HD63701Y0 is connected electrically to the PROM programmer through a socket adapter. Therefore, pay attention to the following:

1. Confirm that the socket adapter is firmly fixed on the PROM programmer.
2. Do not touch the socket adapter or the LSI during programming. Mis-programming can be caused by poor contacts.

## 7.2 Erasing (Window Package)

The EPROM is erased by exposing the LSI to ultraviolet light. All erased bits are in 1's.

The conditions for erasing are: ultraviolet light with wavelength of 2537 Å, and a minimum irradiation of  $15 \text{ W} \cdot \text{s}/\text{cm}^2$ . These conditions are satisfied by exposing the LSI to an ultraviolet light rated at  $12,000 \mu\text{W}/\text{cm}^2$  for 15-20 minutes, at a distance of 1 inch.

## 7.3 Characteristics and Applications

### 7.3.1 Principles of Programming/Erasing

The HD63701Y0's memory cells are the same as an EPROM's. Therefore they are programmed by applying high voltage to control gates and drains, which injects hot electrons into the floating gate (figure 7-3). The condensed electrons in the floating gate are stable, surrounded by an energy barrier of  $\text{SiO}_2$  film. Such a cell becomes a 0 bit due to the memory threshold voltage change. A cell with no condensed electrons at its floating gate appears as a 1 bit.

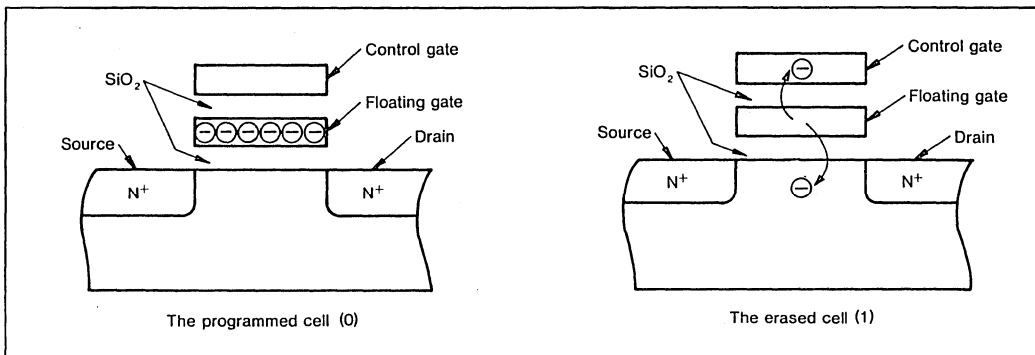


Figure 7-3. Cross-Section of EPROM Memory Cell

The electron charge in memory cells may decrease as time goes by. This can be caused by:

1. Ultraviolet light, discharged by photo-emitting electrons (erasure principle)
2. Heat, discharged by thermal emitting electrons
3. High voltage, discharged by a high electric field at the control gate or drain

If the oxide film covering a floating gate is defective, the erasure rate is great. Normally, electron erasure does not occur, because such defective devices are found and removed during testing.

### 7.3.2 Window-Type Package Precautions

**Glass Erasure Window:** If the glass window comes in contact with plastic or anything with a static charge, the LSI may malfunction due to the electrostatic charge on the surface of the window. If this occurs, exposing the LSI to ultraviolet light for a few minutes neutralizes the charge, and restores the LSI to normal operation. However, charge stored in the floating gate decreases at the same time, so reprogramming is recommended.

Electrostatic charge buildup on the window is a fundamental cause of malfunctions. Measures for its prevention are the same as those for preventing electrostatic breakdown:

1. Operators should be grounded when handling equipment.
2. Do not rub the glass window with plastics.
3. Be careful of coolant sprays, which may contain a few ions.
4. The ultraviolet shading label (which includes conductive material) effectively neutralizes charge.

**Ultraviolet Shading Label:** If the LSI is exposed to fluorescent light or sunlight, its memory contents may be erased by the small quantity of ultraviolet light in these sources. In strong light, the MCU may fail under the influence of photocurrent. To prevent these problems, it is recommended that the device be used with an ultraviolet shading label covering the erasure window after programming.

Special labels are sold for this purpose. They contain metal to absorb ultraviolet light. When choosing a label, note the following:

1. Adhesion (mechanical intensity)—Re-use and dust reduce adhesion. Peeling off a label may cause static electricity. Therefore, erasing and rewriting is recommended after peeling. Sticking a new label over the old one is better than replacing a label.
2. Allowable temperature range—The allowable environmental temperature range of the label should be noted. If it is used under conditions outside this range, the paste may stiffen or adhere to the label, causing paste to remain on the window when the label is removed.
3. Moisture resistance—The allowable moisture range and environmental conditions of the label should be noted. It is difficult to find a shade label applicable to all conditions. The proper label should be selected depending on the intended use of the MCU.

# Section 8. Applications

## 8.1 HD6301Y0 in Expanded Mode

Figure 8-1 shows a microcomputer system using all CMOS peripheral LSI's as an application example of the HD6301Y0 in the expanded mode (modes 1, 2).

Ports 1 and 4 are used for address output, and port 3 is used for data I/O. The system is controlled by directly connecting  $\overline{RD}$  and  $\overline{WR}$  as memory control signals and  $R/\overline{W}$  and E as peripheral controls.

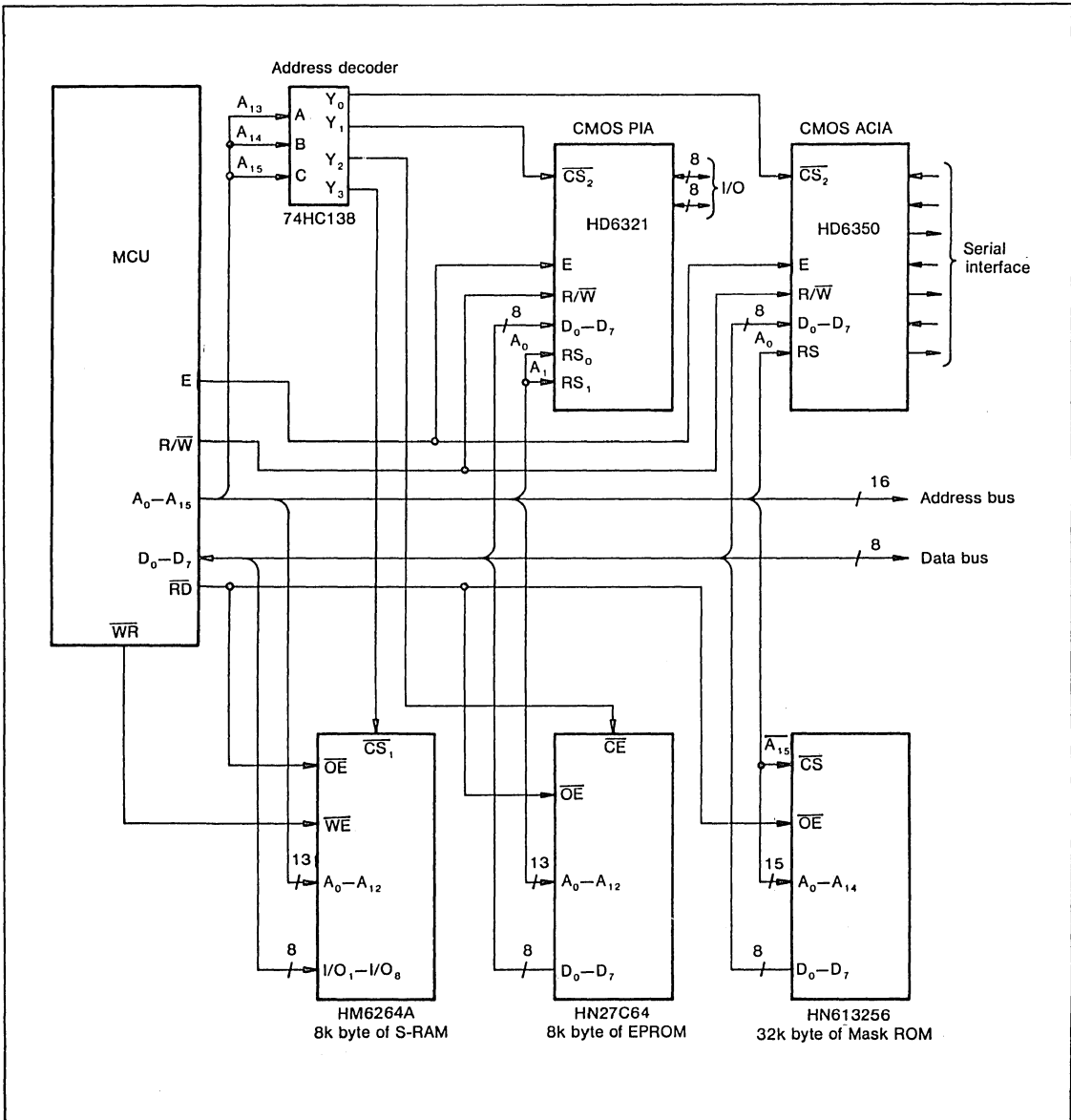


Figure 8-1. All CMOS Microcomputer System



## 8.2 HD6301Y0 in Single-Chip Mode

Figure 8-2 shows a printer controller using the HD6301Y0 in the single-chip mode (mode 3).

The HD6301Y0 controls a 16-dot printer using I/O lines as its ports. Data from the host is transferred to the MCU through the serial interface or through a Centronics interface at port 3.

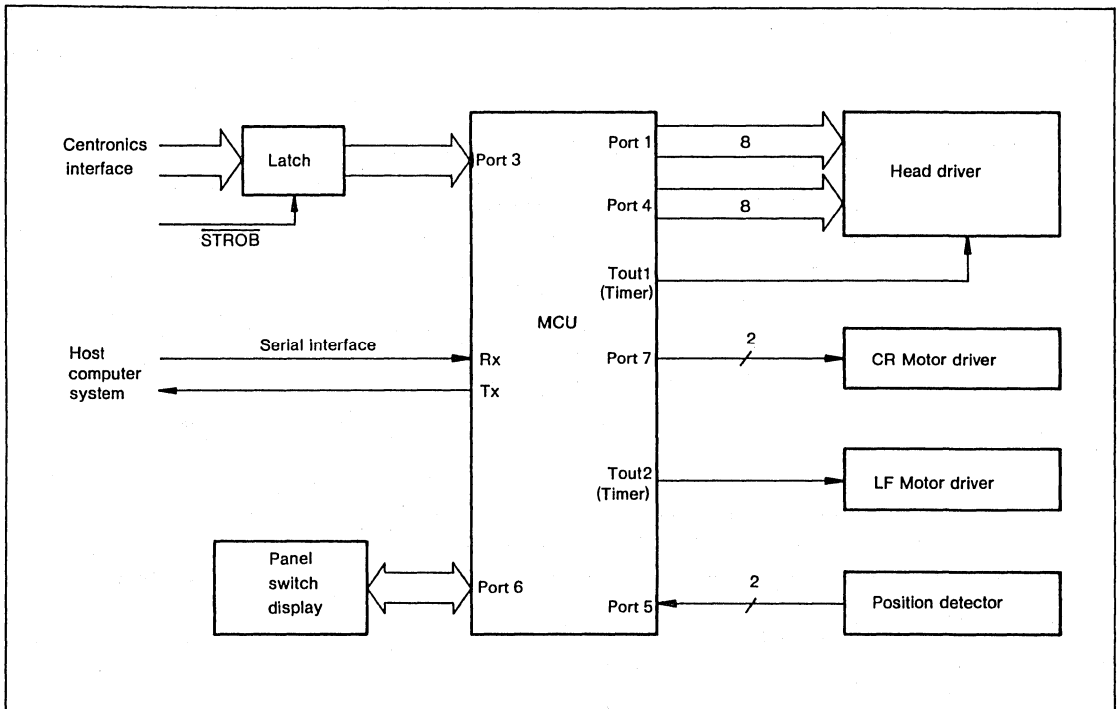


Figure 8-2. Printer Controller

## 8.3 Timer Applications

### 8.3.1 Timer 1

Timer 1 is a 16-bit programmable timer with the same architecture as the timer on the HD6301V1, but with an output compare register added. Timer 1 can perform the following four operations:

1. Waveform generation or interval timing using output compare register 1 (OCR1)
2. Waveform generation or interval timing using output compare register 2 (OCR2)
3. Pulse width or pulse cycle measurement using the input capture register
4. Interval timing with overflow interrupt

**Waveform Generation.** The values of the output compare registers (OCR1, OCR2) are compared with the free-running counter (FRC) at every E cycle. When a match occurs, an output compare flag (OCF1, OCF2) is set. When an output enable bit (OE1E, OE2E) is set, the value of the output level bit (OLVL1, OLVL2) is output at port 2 (Tout1: P2<sub>1</sub>, Tout2: P2<sub>5</sub>). Figure 9-3 is a flowchart for OCR1 waveform generation.

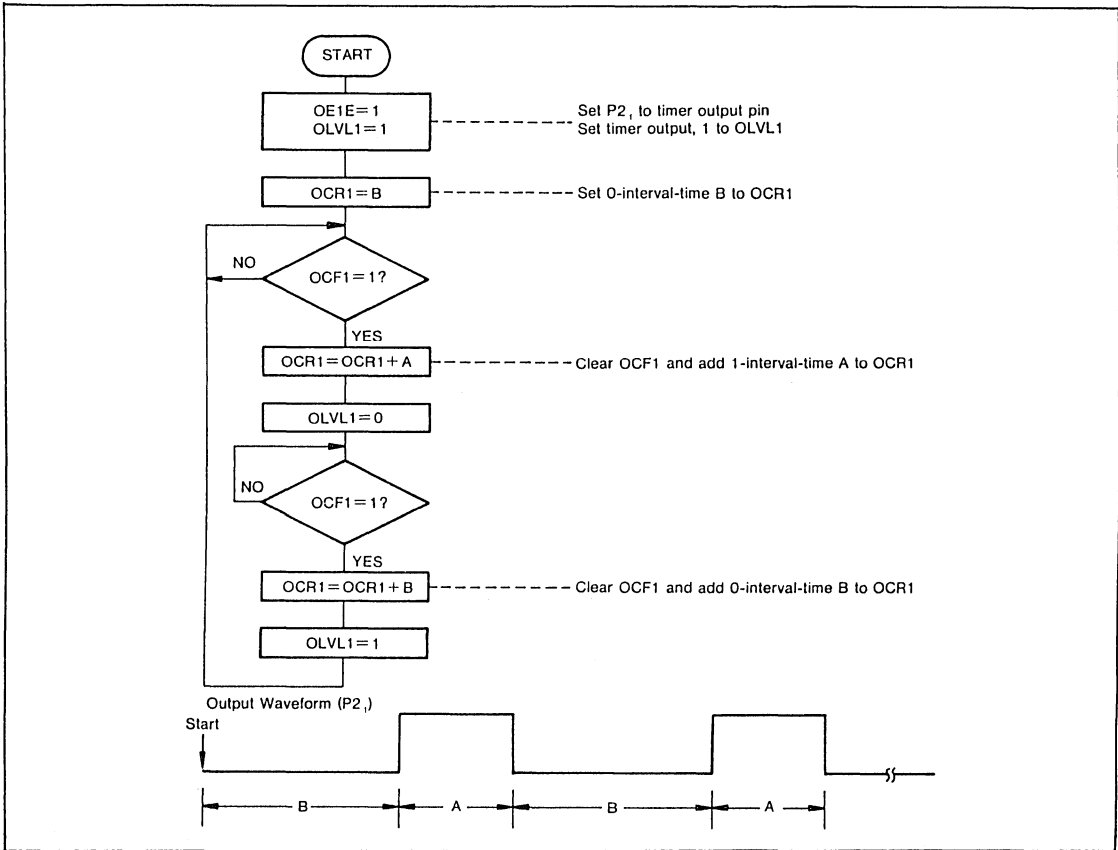


Figure 8-3. OCR1 Waveform Generation

**Pulse Width Measurement.** The input capture register (ICR) latches the free-running counter value at the transition of the external input signal, measuring the pulse width or cycle. Figure 8-4 is a flowchart of pulse width measurement.

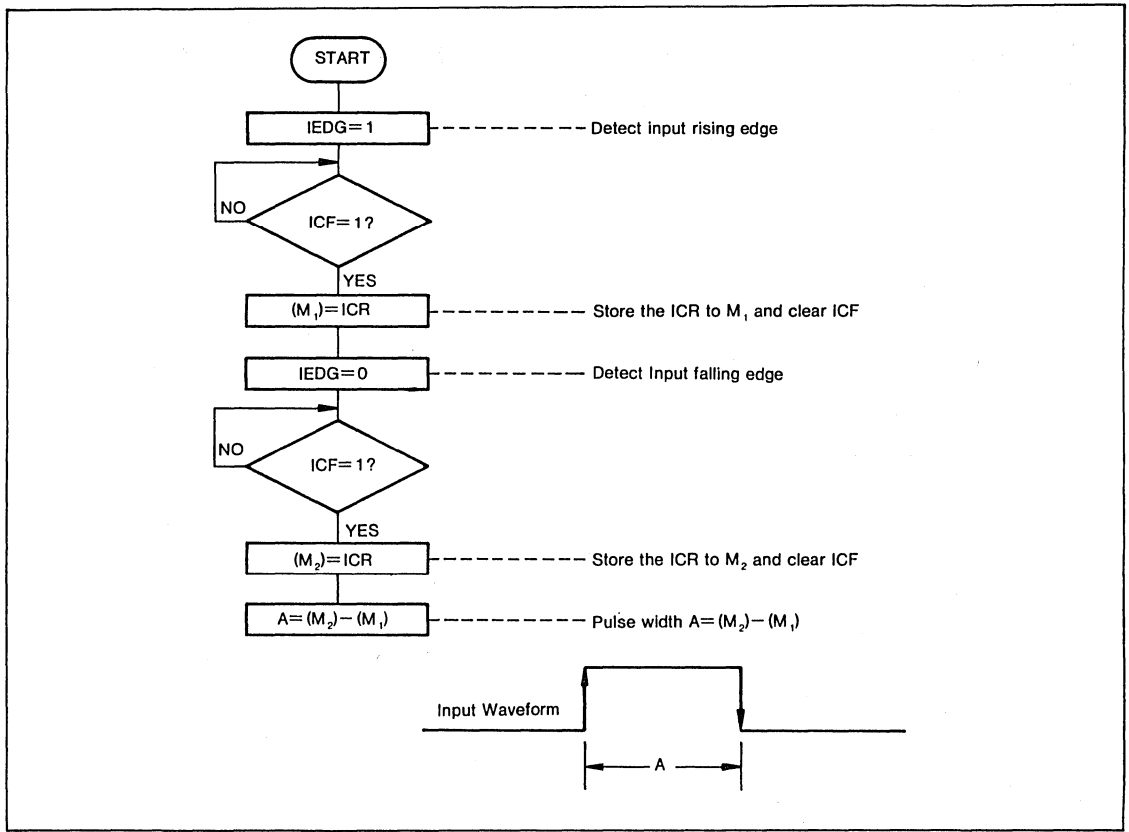


Figure 8-4. ICR Pulse Width Measurement

### 8.3.2 Timer 2

The 8-bit reloadable timer provides such functions as an external event counter, interval timer, waveform generator, and SCI baud rate generator.

**External Event Counter.** Operate timer 2 as an external event counter by setting input clock select, CKS0 and CKS1, to external clock and writing 1 into T2E. The timer 2 upcounter is incremented by the external clock's rising edge. Figure 9-5 shows the routine that generates an interrupt after N external events occur (where N is an integer between 1 and 256).

### 8.3.2 Timer 2

The 8-bit reloadable timer provides such functions as an external event counter, interval timer, waveform generator, and SCI baud rate generator.

**External Event Counter.** Operate timer 2 as an external event counter by setting input clock select, CKS0 and CKS1, to external clock and writing 1 into T2E. The timer 2 upcounter is incremented by the external clock's rising edge. Figure 9-5 shows the routine that generates an interrupt after N external events occur (where N is an integer between 1 and 256).

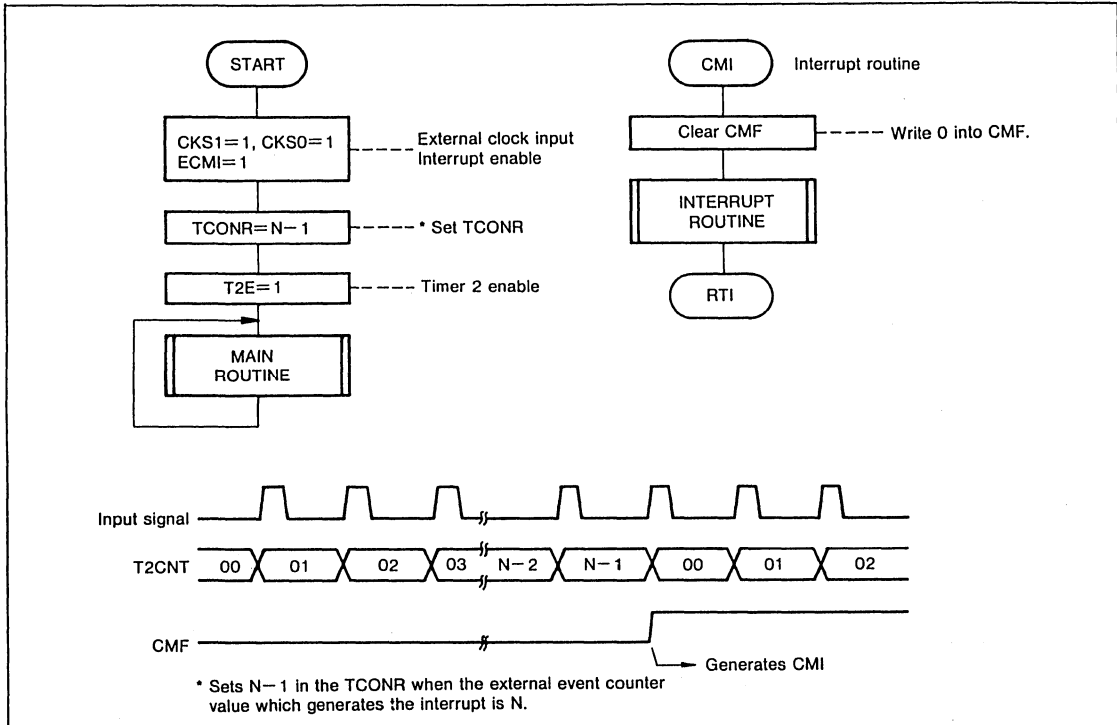


Figure 8-5. External Event Counter

**Square-Wave Generator.** Timer 2 can generate a continuous square wave without software supervision. Figure 8-6 shows this routine.

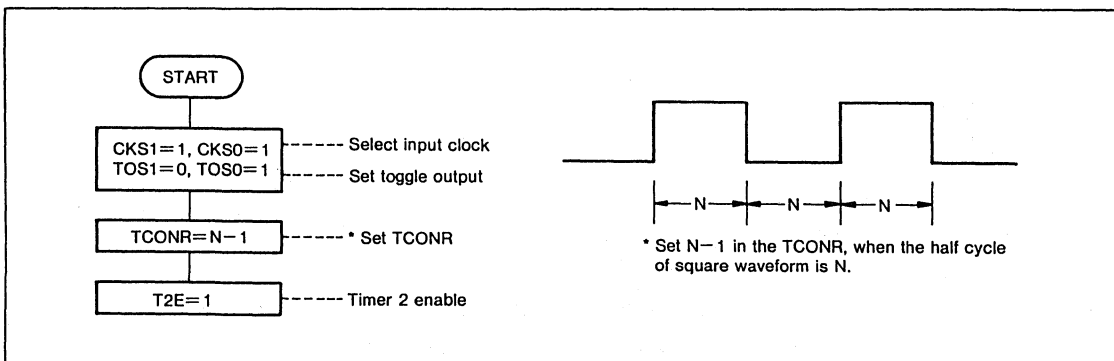


Figure 8-6. Square-Wave Generator



## 8.4 SCI Applications

### 8.4.1 Timer 2 Baud Rate Generator

The SCI can use six kinds of clock source: timer 1's FRC (four kinds), timer 2, and an external clock. The timer 1 baud rate clocks are not adjustable, but timer 2 can provide any baud rate. Figure 8-7 shows how timer 2 can provide the baud rate.

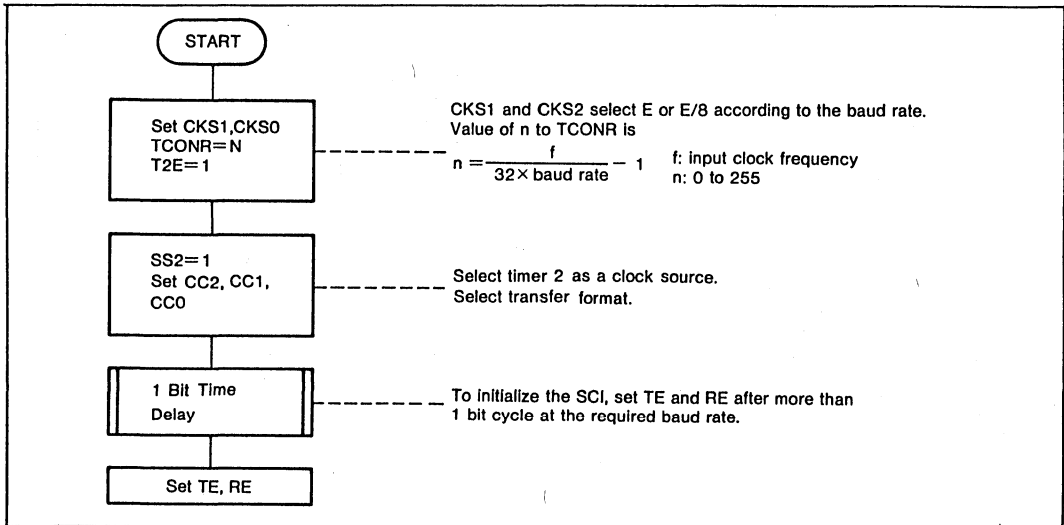


Figure 8-7. Timer 2 as Baud Rate Generator

### 8.4.2 Interface between HD6301Y0 and HD6305X0

An HD6301Y0 can interface to an HD6305X0 in the clock synchronous mode. This gives 99 I/O lines, suitable for systems requiring many I/O lines. Figure 8-8 shows an example of this interface.

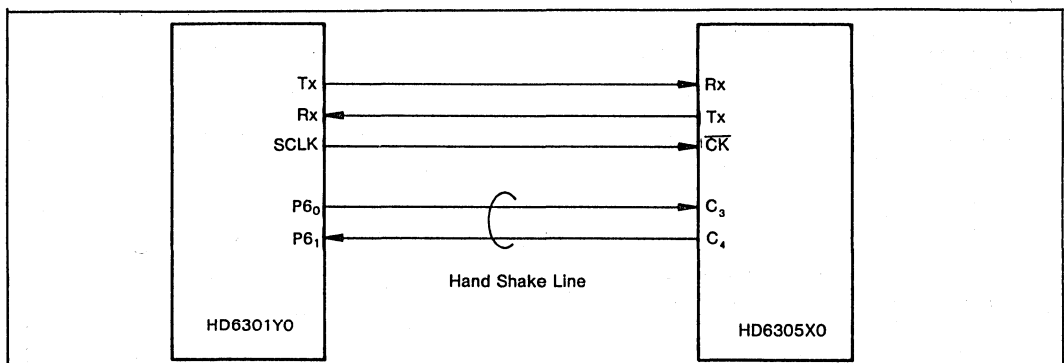


Figure 8-8. HD6301Y0 to HD6305X0 Interface

Employing the clock synchronous mode enables the HD6301Y0 to interface easily to peripheral devices (A/D converter, real-time clock, etc) which use a clock synchronous interface, as well as to the HD6305X0.

### 8.4.3 I/O Expansion

The SCI can be used in the clock synchronous mode to supplement the available parallel I/O ports. Use an external shift register to perform the serial-to-parallel conversion. Figure 8-9 shows this kind of I/O expansion.

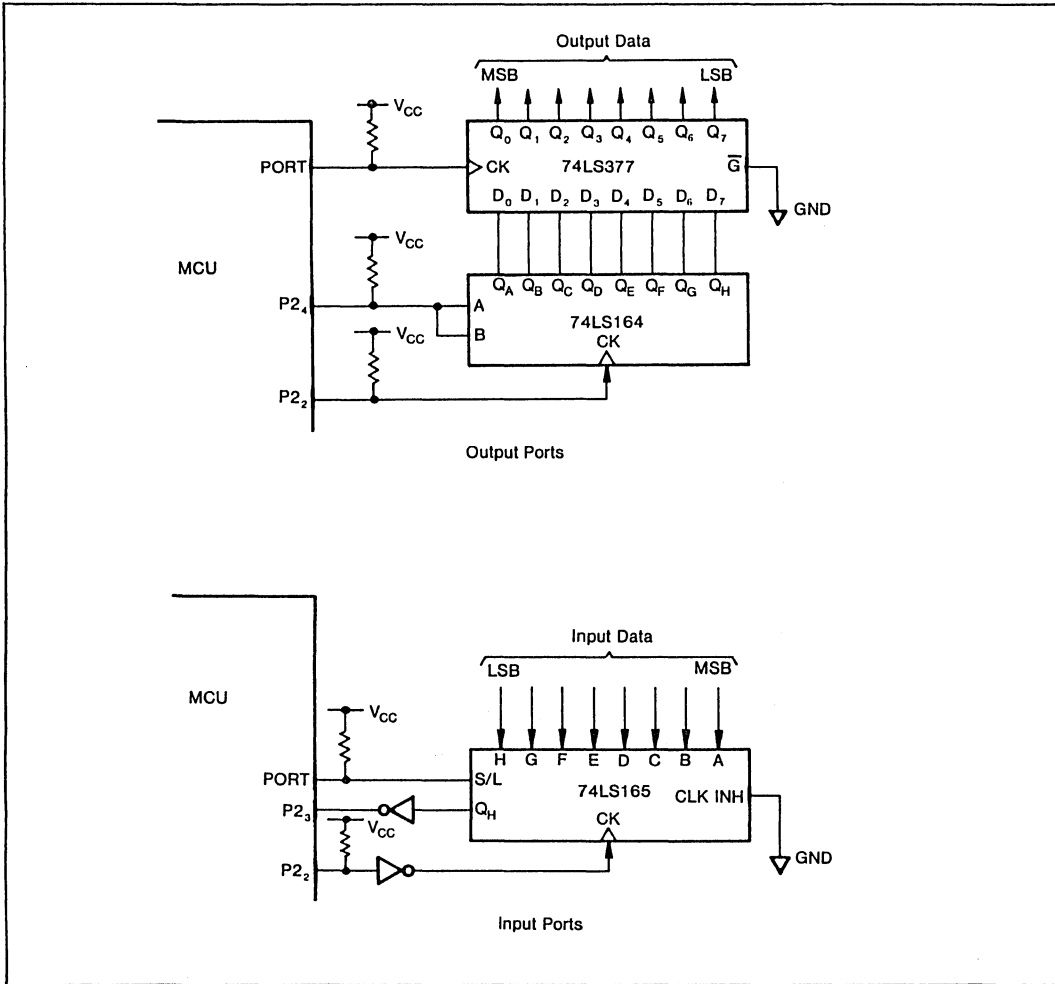


Figure 8-9. I/O Expansion in Clock Synchronous Mode

### 8.4.4 SCI Multiplexer

Use an analog multiplexer as shown in figure 8-10 to use the SCI with both an asynchronous and a clock synchronous device, such as an HD6305X0 and an RS-232C.

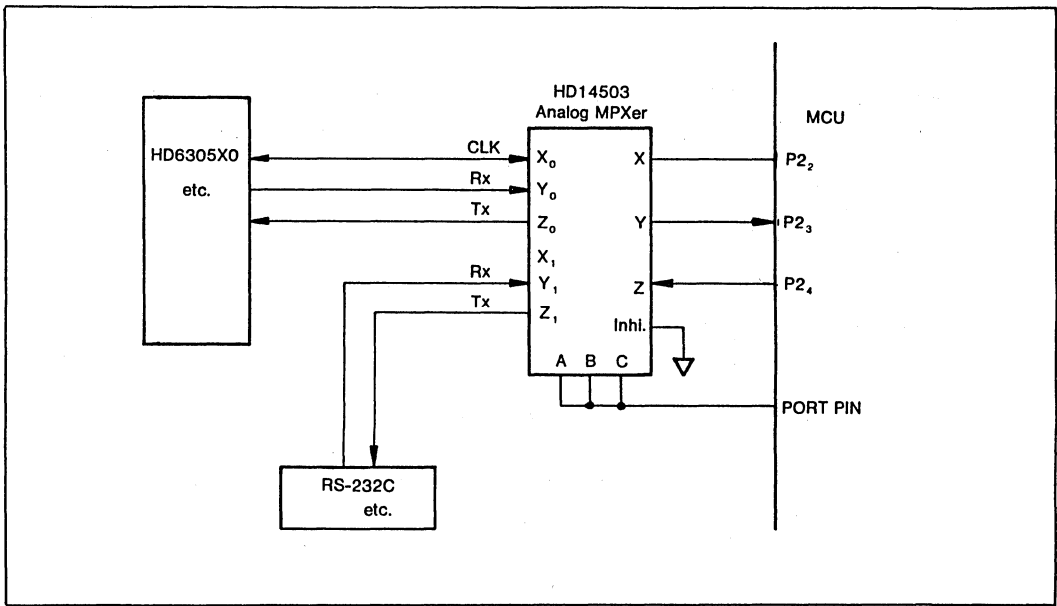


Figure 8-10. Multiplexed SCI

## 8.5 Lowering Operating Current

### 8.5.1 Lowering Operating Frequency

The HD6301Y0/HD6303Y operating current is approximately proportional to the operating frequency (figure 8-11). Therefore, if the system does not require a high-speed MCU, power can be reduced by lowering the operating frequency.

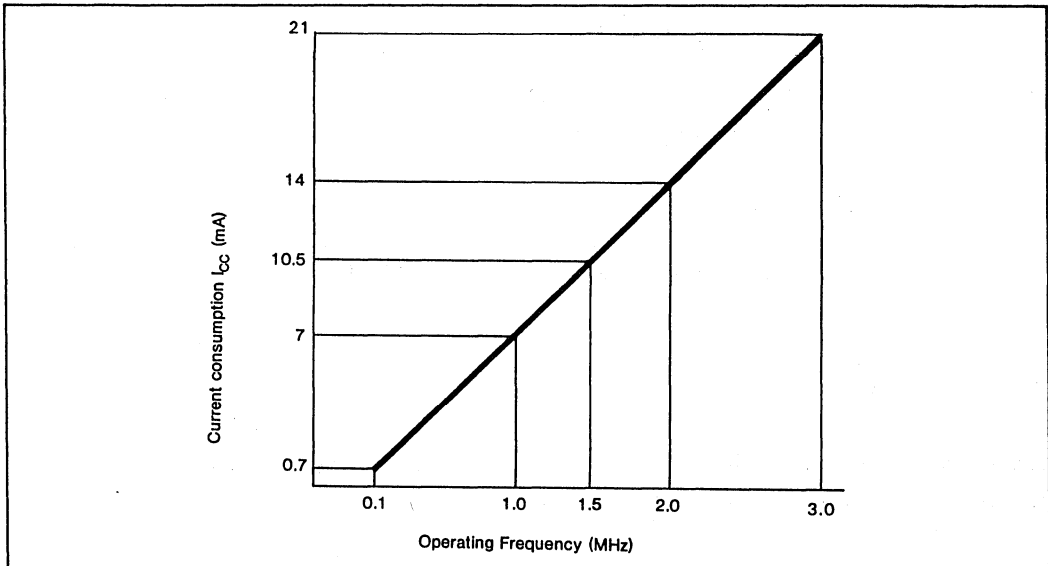


Figure 8-11. Operating Frequency and Current (Typical)

## 8.5.2 Sleep Mode

The SLP instruction puts the MCU into the sleep mode. In the sleep mode, current consumption is reduced to one-fourth to one-fifth of that in the operating state. When the CPU acknowledges an interrupt request, it cancels the sleep mode. The average power consumption can be reduced by putting the CPU in sleep mode whenever it doesn't actually execute any instructions, such as in interrupt wait state or polling. Figure 8-12 shows a routine which wakes the CPU up every 65 ms, using the overflow interrupt of the timer 1 FRC.

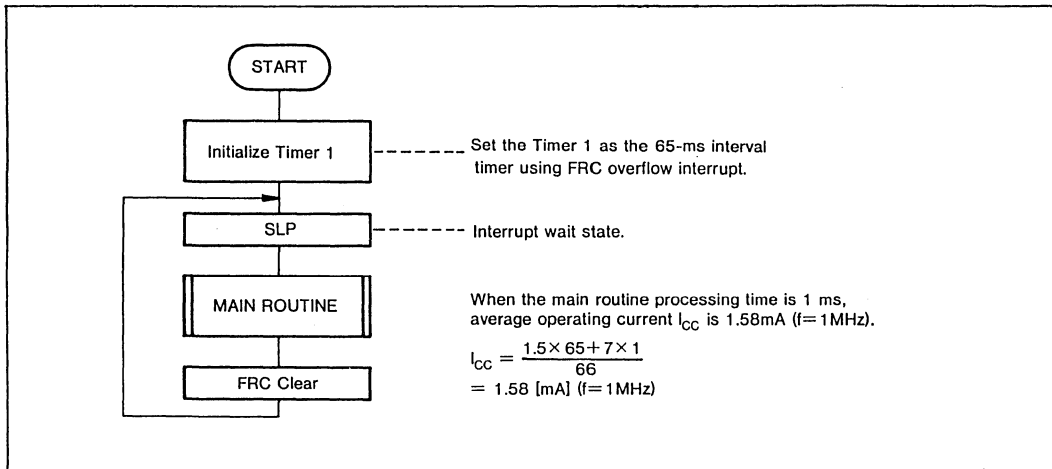


Figure 8-12. Low Power Consumption Using the Sleep Mode

## 8.5.3 Standby Mode

Bringing  $\overline{\text{STBY}}$  (pin 7) low puts the MCU into standby mode. In standby mode, the oscillator stops and the MCU goes into the reset state. The contents of RAM are maintained as long as  $V_{CC}$  is greater than or equal to 2 V. In standby mode, current consumption is reduced to a few  $\mu\text{A}$ . RAM can be maintained by battery.

Bringing  $\overline{\text{STBY}}$  high cancels standby mode. The MCU releases the reset state and starts oscillation.  $\overline{\text{RES}}$  (pin 6) should be held low for at least the oscillation stabilization time ( $t_{RC}$ ) after  $\overline{\text{STBY}}$  high. Figure 8-13 gives an example of a circuit that sets standby from software. Figure 8-14 shows the timing for this circuit, and figure 8-15 is an operating flowchart.

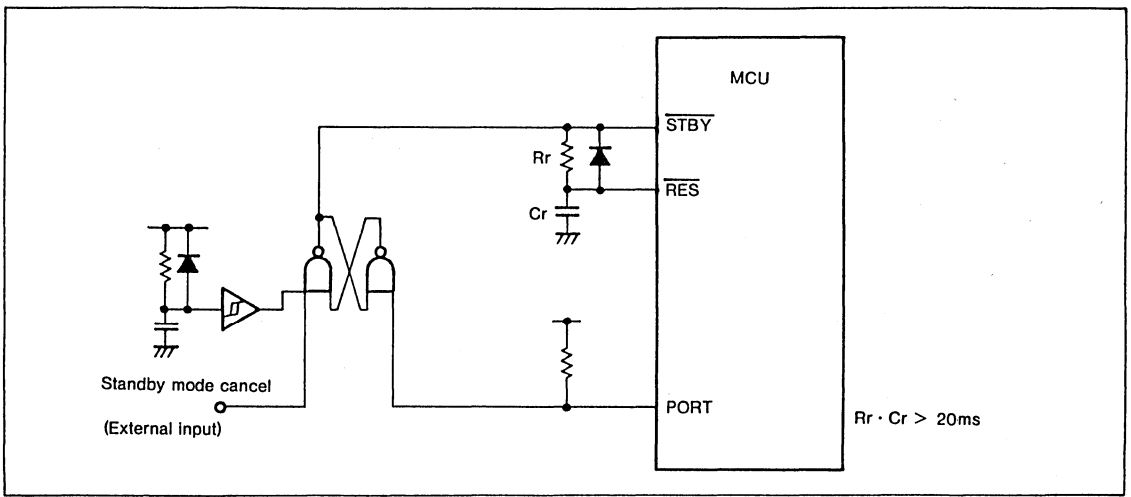


Figure 8-13. Standby Circuit Example

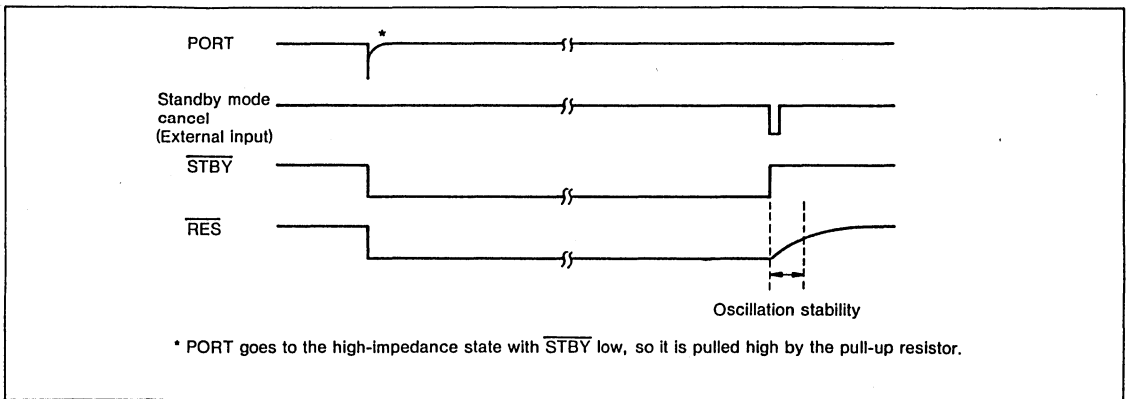


Figure 8-14. Standby Timing

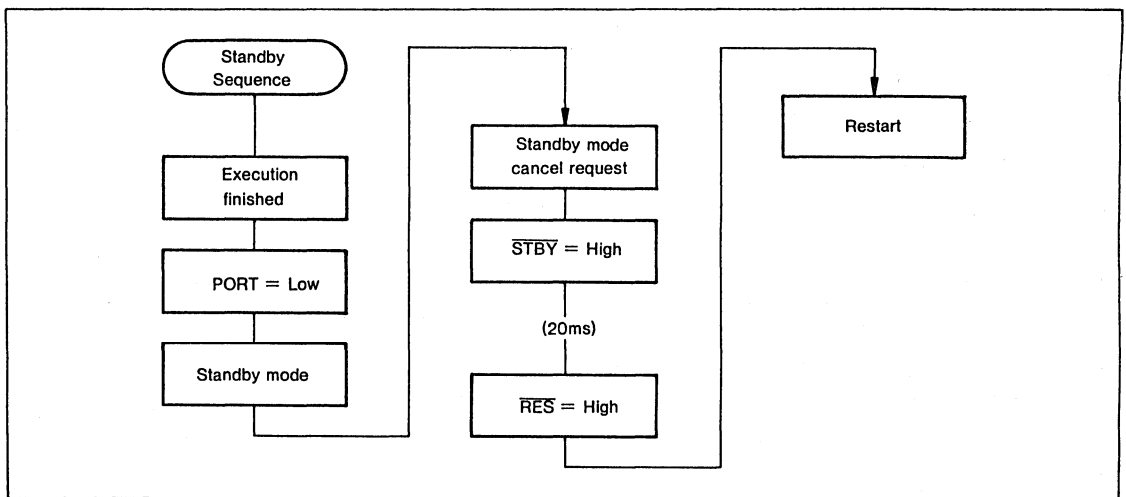


Figure 8-15. Standby Circuit Flowchart

## 8.6 Memory Ready Application

The memory ready function allows the MCU to access low-speed memories or low-speed devices. Figure 8-16 shows a circuit example, and figure 8-17 is its timing chart.

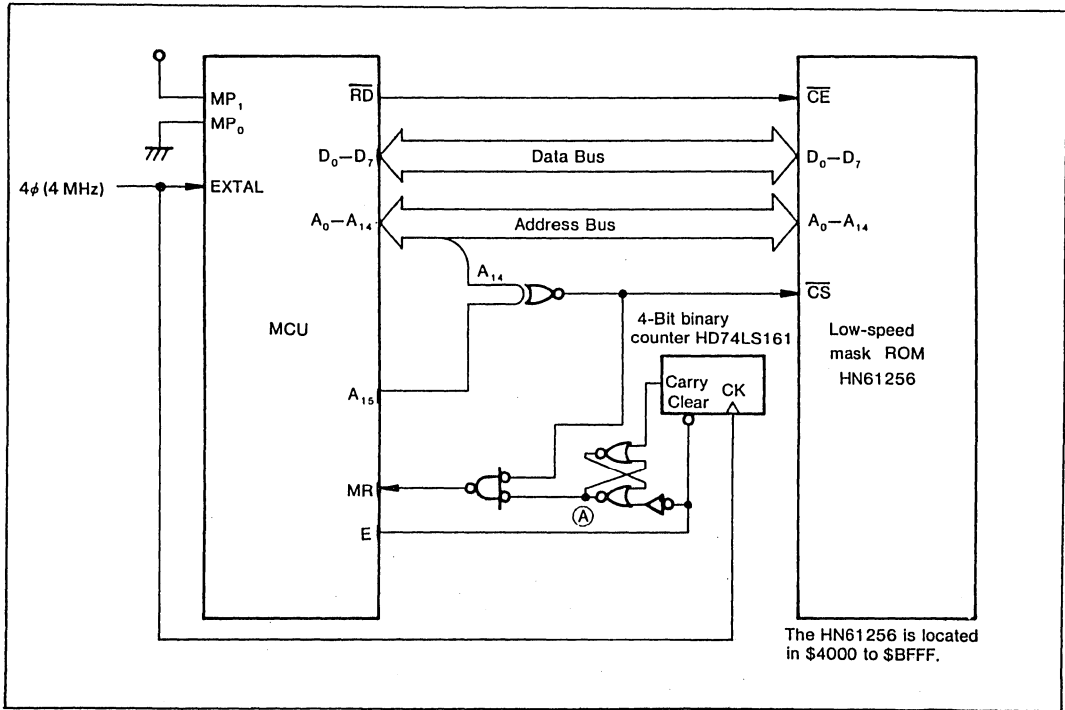


Figure 8-16. Low-Speed Memory Access Circuit

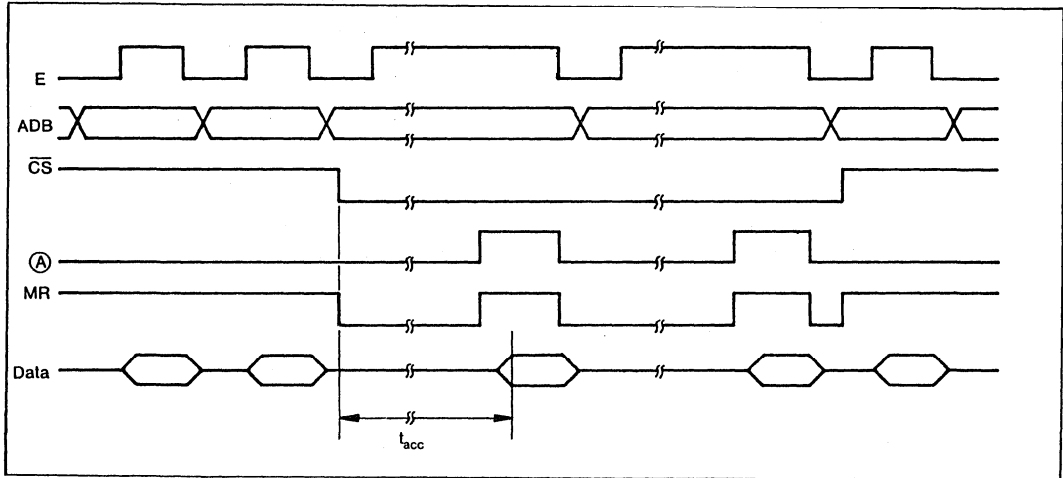


Figure 8-17. Memory Ready Bus Timing

## 8.7 Halt Application

The halt function enables the MCU in the expanded mode to interface with a DMAC (HD6844) and execute DMA (figure 8-18).

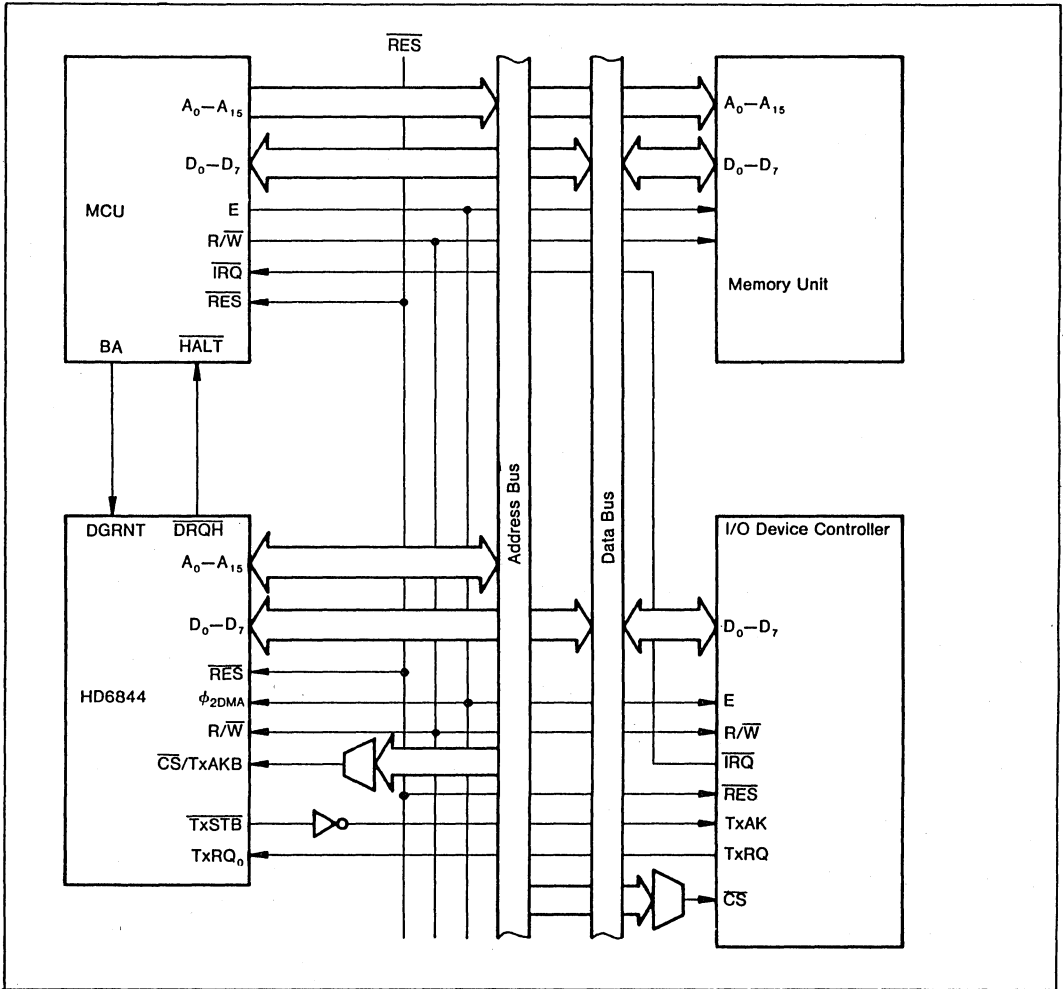


Figure 8-18. One-Channel DMAC Interface Example

## 8.8 $\overline{RD}$ , $\overline{WR}$ Application

$\overline{RD}$  and  $\overline{WR}$ , as well as E and  $R/\overline{W}$ , can act as external interface signals.  $\overline{RD}$  and  $\overline{WR}$  allow the MCU to easily interface with the 80xx family peripherals as well as with the 6800 series. Figure 8-19 shows an example of an interface between an MCU and an 8255.

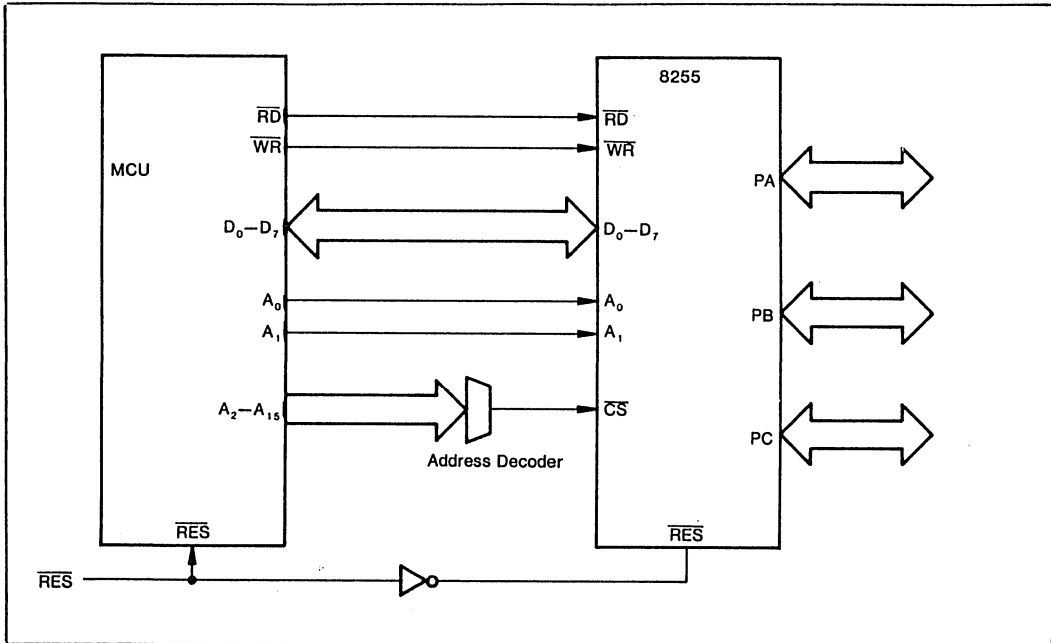


Figure 8-19. HD6301Y0 and 8255 Interface

## 8.9 LCD-II Interface Application

Figure 8-20 and 8-21 show examples of interfaces between an HD6301Y0 and a liquid crystal driver (LCD-II). The interface lines are TTL compatible. The HD6301Y0 in the expanded mode in figure 8-20 interfaces with the LCD-II directly through the external bus lines. Port 3 connects to the LCD-II data bus,  $R/\overline{W}$  connects to  $R/\overline{W}$ ,  $A_0$  connects to RS, and the rest of the address bus is decoded and ANDed with E to connect with E on the LCD-II.

The HD6301Y0 in the single-chip mode in figure 8-21 interfaces with the LCD-II through the I/O port. The read/write operation should be performed with care for the timing of the LCD-II E signal and others.



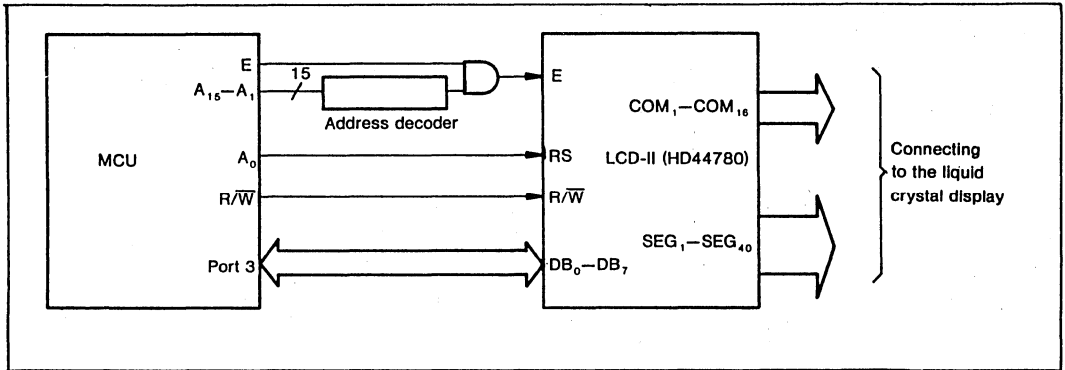


Figure 8-20. LCD-II Interface, Expanded Mode

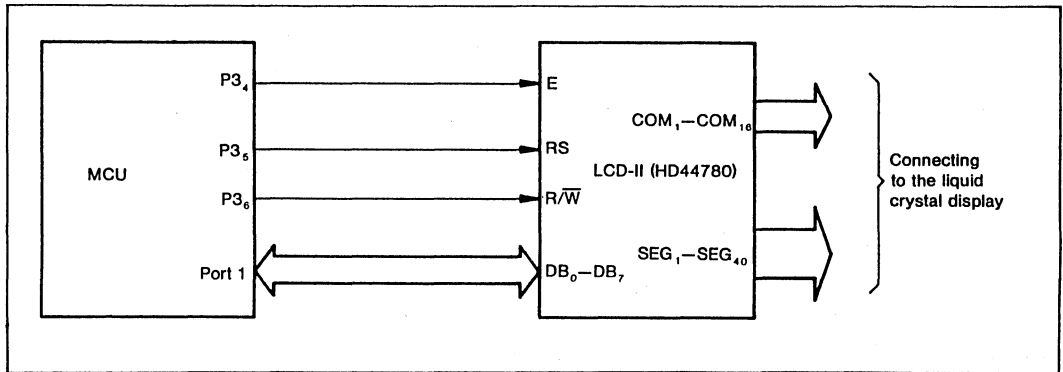


Figure 8-21. LCD-II Interface, Single-Chip Mode

## 8.10 Oscillation Circuit Board Design

Keep the following rules in mind when designing the circuit to connect the crystal resonator with the XTAL and EXTAL pins (figure 8-22, 8-23).

1. The crystal and load capacitors should be as close to the LSI as possible. External noise at the XTAL and EXTAL pins will disturb normal oscillation.
2. Keep the lines from XTAL and E as far apart as possible. Avoid parallel wiring. Interference from E to XTAL will disturb normal oscillation.
3. Do not allow signal or power lines to cross or run closely parallel to the oscillator lines (signals A, B, C in figure 8-22). They will disturb normal oscillation. Keep the resistance between XTAL and EXTAL pins and the next nearest pins greater than 10 MΩ.

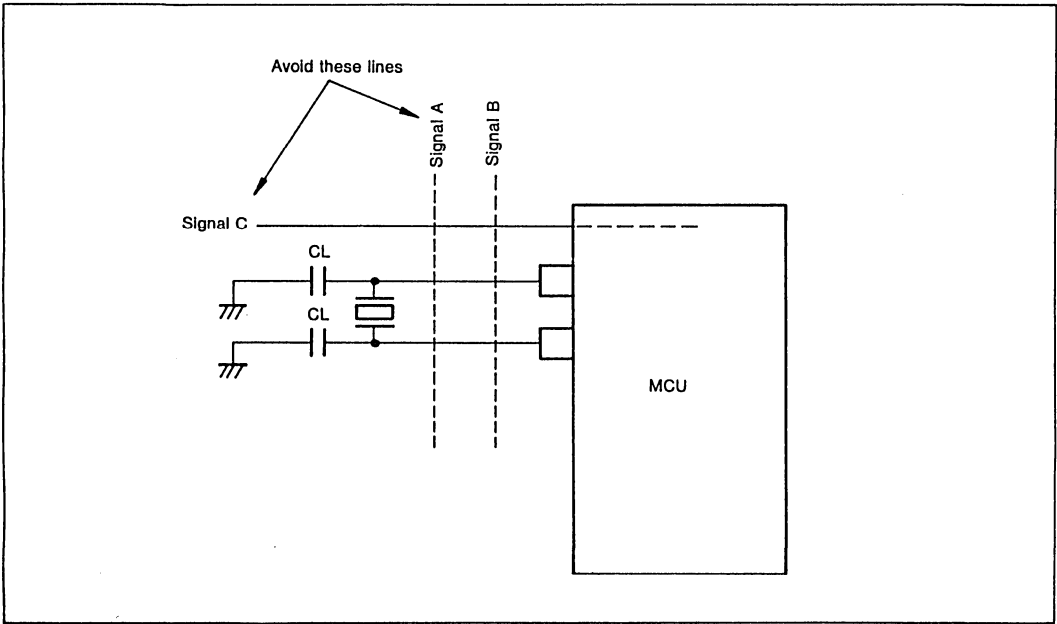


Figure 8-22. Oscillation Circuit Precautions

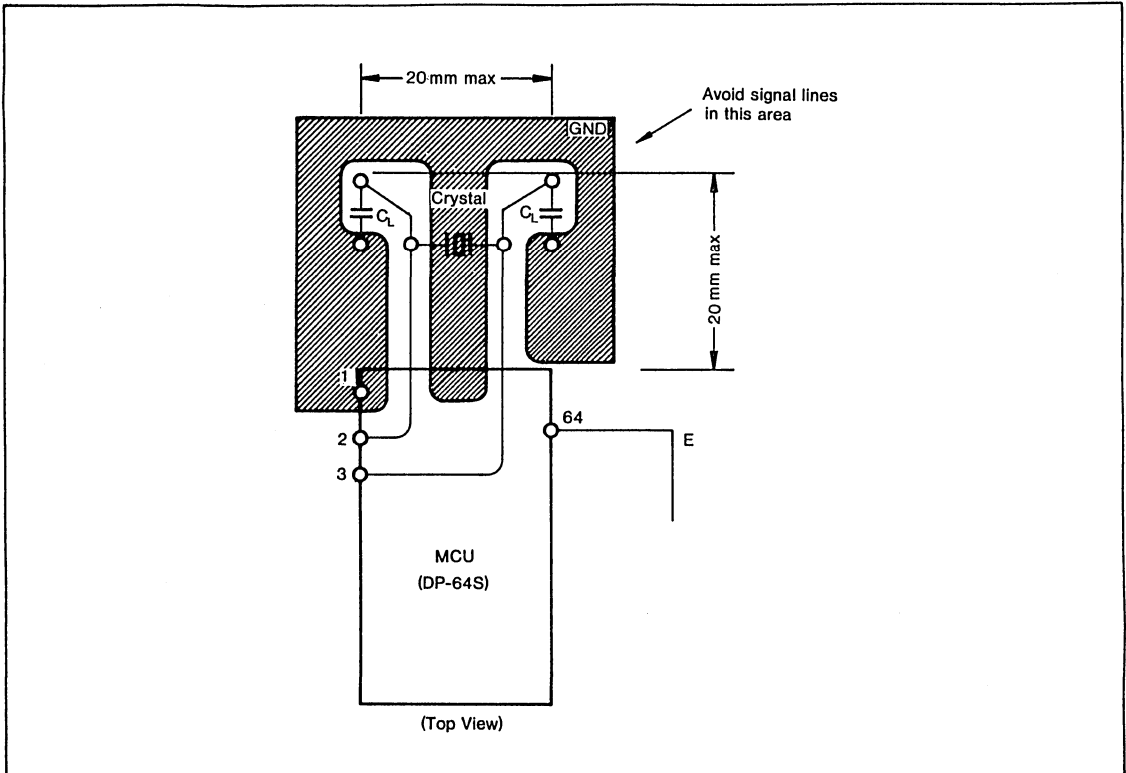


Figure 8-23. Oscillation Circuit Board Design Example

# Appendix I. Electrical Characteristics

## I.1 HD6301Y0, HD63A01Y0, HD63B01Y0, HD63C01Y0 Electrical Characteristics

### Absolute Maximum Ratings

Item	Symbol	Value	Unit
Supply voltage	$V_{CC}$	-0.3 to +7.0	V
Input voltage	$V_{in}$	-0.3 to $V_{CC}+0.3$	V
Operating temperature	$T_{opr}$	-20 to +70	°C
Storage temperature	$T_{stg}$	-55 to +150	°C

Note: This product has protection circuits in input terminal from high static electricity voltage and high electric field. But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend  $V_{in}$ ,  $V_{out}$ :  $V_{SS} \leq (V_{in} \text{ or } V_{out}) \leq V_{CC}$ .

### Electrical Characteristics

#### DC Characteristics

( $V_{CC}=5.0\text{ V} \pm 10\%$ ,  $f=0.1$  to 3.0 MHz,  $V_{SS}=0\text{ V}$ ,  $T_a=-20$  to +70°C)

Item		Symbol	Min	Typ	Max	Unit	Test Condition
Input high voltage	RES, STBY,	$V_{IH}$	$V_{CC}-0.5$		$V_{CC}+0.3$	V	$V_{in}=0.5$ to $V_{CC}-0.5\text{ V}$
	EXTAL		$V_{CC} \times 0.7$		$V_{CC}+0.3$	V	
	Other inputs		2.0		$V_{CC}+0.3$	V	
Input low voltage	All other inputs	$V_{IL}$	-0.3		0.8	V	
Input leakage current	RES, NMI, STBY, MP0, MP1	$ I_{in} $			1.0	$\mu\text{A}$	$V_{in}=0.5$ to $V_{CC}-0.5\text{ V}$
Three state leakage current	A0-A15, D0-D7, RD, WR, R/W, Ports 2, 5, 6	$ I_{TSI} $			1.0	$\mu\text{A}$	$V_{in}=0.5$ to $V_{CC}-0.5\text{ V}$
Output high voltage		$V_{OH}$	2.4			V	$I_{OH}=-200\ \mu\text{A}$
			$V_{CC}-0.7$			V	$I_{OH}=-10\ \mu\text{A}$
Output low voltage		$V_{OL}$			0.4	V	$I_{OL}=1.6\text{ mA}$
Darlington drive current	Ports 2, 6	$-I_{OH}$	1.0		10.0	mA	$V_{out}=1.5\text{ V}$
Input capacitance	All other inputs	$C_{in}$			12.5	pF	$V_{in}=0\text{ V}$ , $f=1\text{ MHz}$ , $T_a=25^\circ\text{C}$
Standby current	Not operating	$I_{STB}$		3.0	15.0	$\mu\text{A}$	
Current dissipation <sup>1</sup>		$I_{SLP}$		1.5	3.0	mA	Sleeping ( $f=1\text{ MHz}^2$ )
				2.3	4.5	mA	Sleeping ( $f=1.5\text{ MHz}^2$ )
				3.0	6.0	mA	Sleeping ( $f=2\text{ MHz}^2$ )
				4.5	9.0	mA	Sleeping ( $f=3\text{ MHz}^2$ )
		$I_{CC}$		7.0	10.0	mA	Operating ( $f=1\text{ MHz}^2$ )
				10.5	15.0	mA	Operating ( $f=1.5\text{ MHz}^2$ )
				14.0	20.0	mA	Operating ( $f=2\text{ MHz}^2$ )
				21.0	30.0	mA	Operating ( $f=3\text{ MHz}^2$ )
RAM standby voltage		$V_{RAM}$	2.0			V	

Notes:

- $V_{IH\ min}=V_{CC}-1.0\text{V}$ ,  $V_{IH\ max}=0.8\text{V}$  (All output terminals are at no load.)
- Current dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ. or max. values about current dissipations at  $x\text{ MHz}$  operation are decided according to the following formula:  
 typ. value ( $f=x\text{ MHz}$ ) = typ. value ( $f=1\text{ MHz}$ )  $\times x$   
 max. value ( $f=x\text{ MHz}$ ) = max. value ( $f=1\text{ MHz}$ )  $\times x$   
 (both the sleeping and operating)

## AC Characteristics

(VCC = 5.0 V ± 10%, f = 0.1 to 3.0 MHz, VSS = 0 V, Ta = -20 to +70°C, unless otherwise noted.)

### Bus Timing

Item	Symbol	HD6301Y0			HD63A01Y0			HD63B01Y0			HD63C01Y0			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Cycle time	t <sub>cyc</sub>	1		10	0.666		10	0.5		10	0.333		10	μs	Fig. I-1
Enable rise time	t <sub>Er</sub>			25			25			25			20	ns	
Enable fall time	t <sub>Ef</sub>			25			25			25			20	ns	
Enable pulse width high level <sup>1</sup>	PW <sub>EH</sub>	450			300			220			140			ns	
Enable pulse width low level <sup>1</sup>	PW <sub>EL</sub>	450			300			220			140			ns	
Address, R/ $\bar{W}$ delay time <sup>1</sup>	t <sub>AD</sub>			250			190			160			120	ns	
Data delay time (Write)	t <sub>DDW</sub>			200			160			120			100	ns	
Data set-up time (Read)	t <sub>DSR</sub>	80			70			60			50			ns	
Address, R/ $\bar{W}$ hold time <sup>1</sup>	t <sub>AH</sub>	80			50			40			20			ns	
Data hold time (Write) <sup>1</sup>	t <sub>HW</sub>	80			50			40			20			ns	
Data hold time (Read)	t <sub>HR</sub>	0			0			0			0			ns	
$\bar{R}$ D, $\bar{W}$ R pulse width <sup>1</sup>	PW <sub>RW</sub>	450			300			220			140			ns	
$\bar{R}$ D, $\bar{W}$ R delay time	t <sub>RWD</sub>			40			40			40			40	ns	
$\bar{R}$ D, $\bar{W}$ R hold time	t <sub>HRW</sub>			20			20			20			20	ns	
$\bar{L}$ R delay time	t <sub>DLR</sub>			200			160			120			80	ns	
$\bar{L}$ R hold time	t <sub>HLR</sub>	10			10			10			5			ns	
Peripheral read access time <sup>1</sup>	t <sub>ACC</sub>										180			ns	
MR set-up time <sup>1</sup>	t <sub>SMR</sub>	400			280			230			170			ns	Fig. I-2
MR hold time <sup>1</sup>	t <sub>HMR</sub>			100			70			50			25	ns	
E clock pulse width at MR	PW <sub>EMR</sub>			9			9			9			9	μs	
Processor control set-up time	t <sub>PCS</sub>	200			200			200			100			ns	Figs. I-3, I-13, I-14
Processor control rise time	t <sub>PCr</sub>			100			100			100			50	ns	Figs. I-2, I-3
Processor control fall time	t <sub>PCf</sub>			100			100			100			50	ns	
BA delay time	t <sub>BA</sub>			250			190			160			120	ns	Fig. I-3
Oscillator stabilization time	t <sub>RC</sub>	20			20			20			20			ms	Fig. I-14
Reset pulse width	PW <sub>RST</sub>	3			3			3			3			t <sub>cyc</sub>	

Note: 1. These timings change in approximate proportion to t<sub>cyc</sub>. The figures in this characteristics represent those when t<sub>cyc</sub> is minimum (=in the highest speed operation).

## Peripheral Port Timing

Item	Symbol	HD6301Y0			HD63A01Y0			HD63B01Y0			HD63C01Y0			Test	
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Unit	Condition
Peripheral data set-up time	(Ports 1, 2, 3, 4, 5, 6) tPDSU	200			200			200			200			ns	Fig. I-5
Peripheral data hold time	(Ports 1, 2, 3, 4, 5, 6) tPDH	200			200			200			200			ns	
Delay time (From enable fall edge peripheral output)	(Ports 1, 2, 3, 4, 5, 6, 7) tPWD			300			300			300			300	ns	Fig. I-6
Input strobe pulse width	tPWIS	200			200			200			200			ns	Fig. I-10
Input data hold time	(Port 6) tIH	150			150			150			150			ns	
Input data set-up time	(Port 6) tIS	100			100			100			100			ns	
Output strobe time	tOSD1			200			200			200			200	ns	Fig. I-11
	tOSD2														

## Timer, SCI Timing

Item	Symbol	HD6301Y0			HD63A01Y0			HD63B01Y0			HD63C01Y0			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Timer 1 input pulse width	tPWT	2.0			2.0			2.0			2.0			t <sub>cyc</sub>	Fig. I-9
Delay time (enable positive transition to timer output)	tTOD			400			400			400			400	ns	Figs. I-7, I-8
SCI input clock cycle	(Async mode) tScyc	1.0			1.0			1.0			1.0			t <sub>cyc</sub>	Fig. I-9
	(Clock sync.)	2.0			2.0			2.0			2.0			t <sub>cyc</sub>	Fig. I-4
SCI transmit data delay time (Clock sync. mode)	tTXD			220			220			220			220	ns	Fig. I-4
SCI receive data set-up time (Clock sync. mode)	tSRX	260			260			260			260			ns	
SCI receive data hold time (Clock sync. mode)	tHRX	100			100			100			100			ns	
SCI input clock pulse width	tPWSCC	0.4	0.6	0.4	0.6	0.4	0.6	0.4	0.6	0.4	0.6	0.4	0.6	t <sub>Scyc</sub>	Fig. I-9
Timer 2 input clock cycle	tCyc	2.0			2.0			2.0			2.0			t <sub>cyc</sub>	
Timer 2 input clock pulse width	tPWTCCK	200			200			200			200			ns	
Timer 1 - 2, SCI input clock rise time	tCKr			100			100			100			50	ns	
Timer 1 - 2, SCI input clock fall time	tCKf			100			100			100			50	ns	

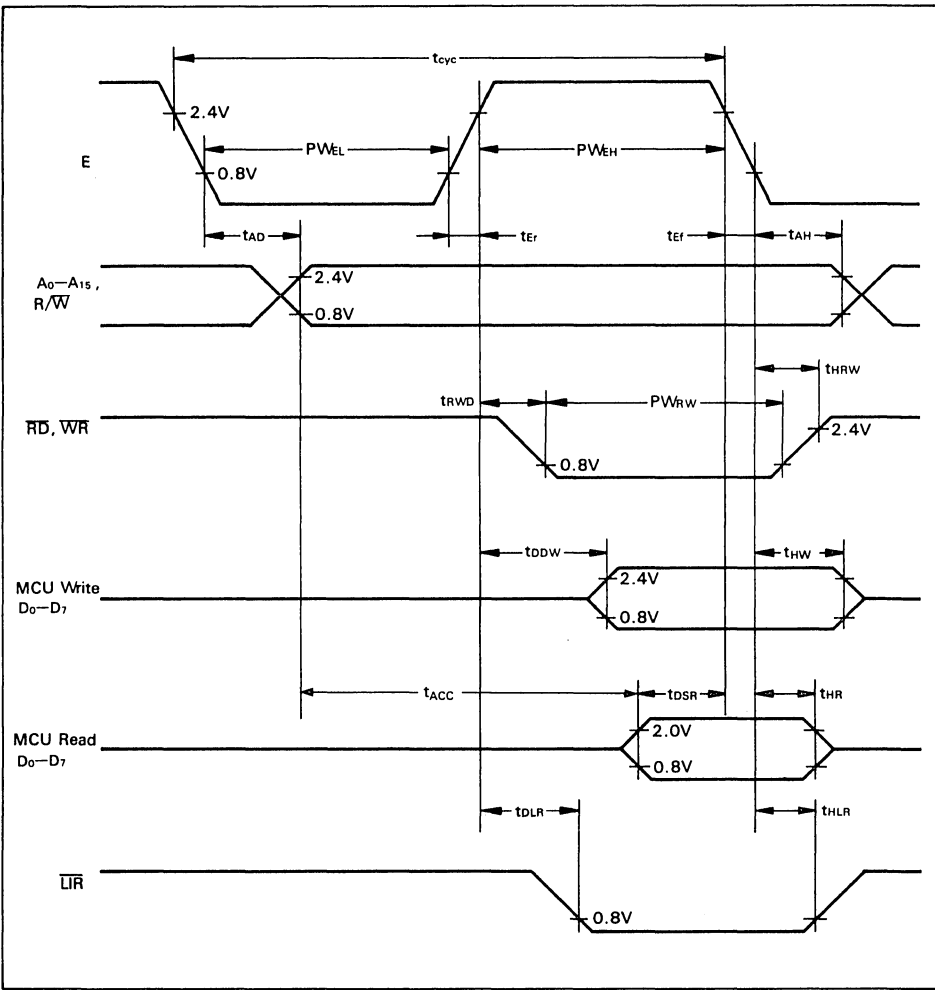


Figure I-1. Mode 1, Mode 2 Bus Timing

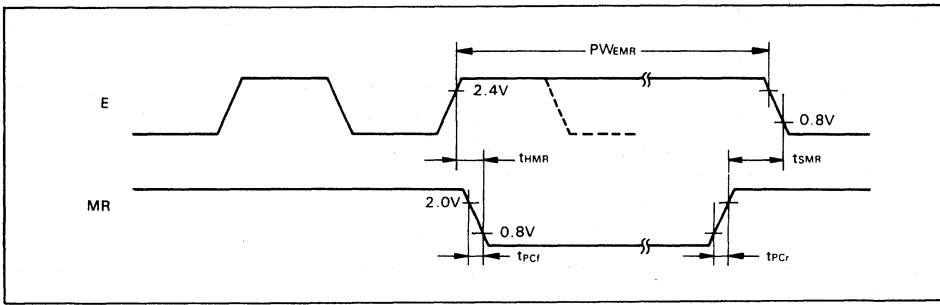


Figure I-2. Memory Ready and E Clock Timing

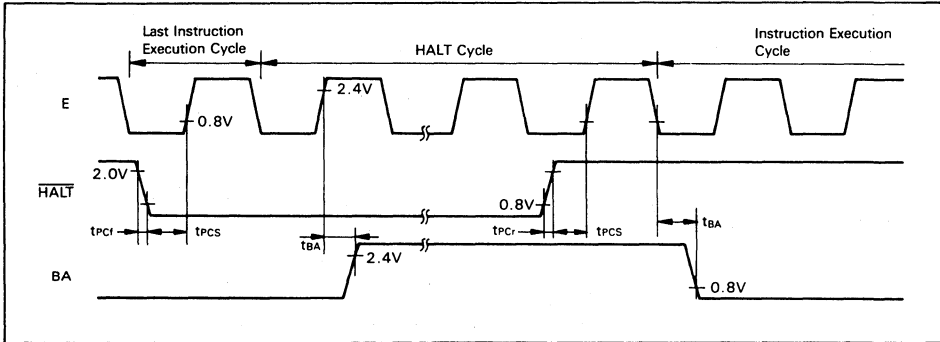


Figure I-3. HALT and BA Timing

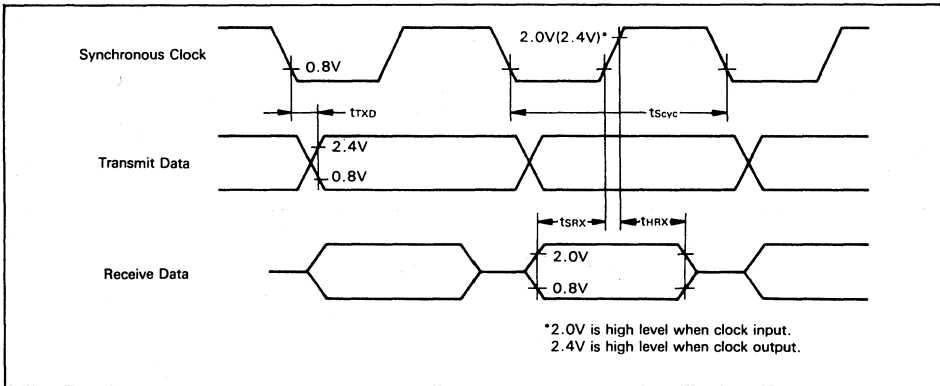


Figure I-4. SCI Clocked Synchronous Timing

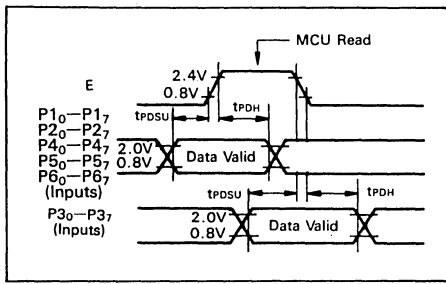


Figure I-5. Port Data Set-up and Hold Times (MCU Read)

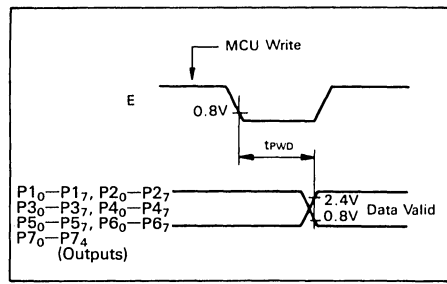


Figure I-6. Port Data Delay Times (MCU Write)

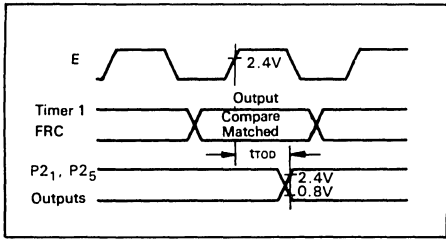


Figure I-7. Timer 1 Output Timing

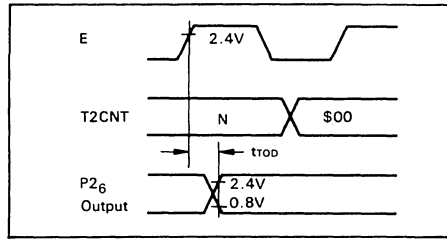


Figure I-8. Timer 2 Output Timing

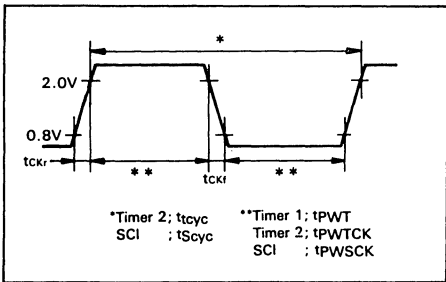


Figure I-9. Timer 1-2, SCI Input Clock Timing

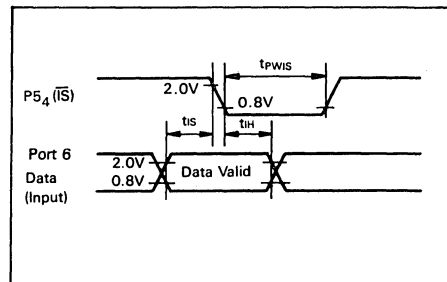


Figure I-10. Port 6 Input Latch Timing

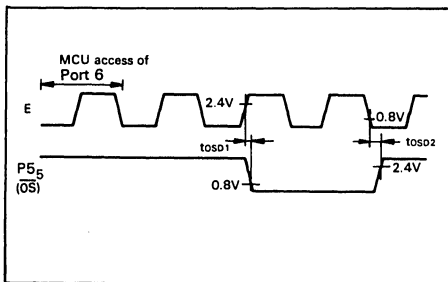


Figure I-11. Output Strobe Timing

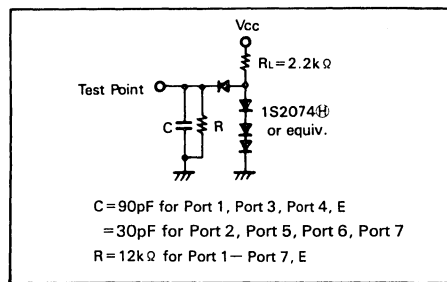


Figure I-12. Bus Timing Test Loads (TTL Load)



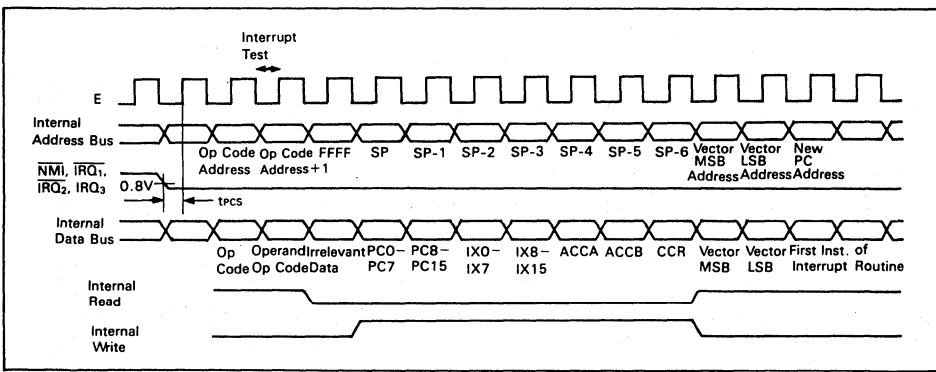


Figure I-13. Interrupt Sequence

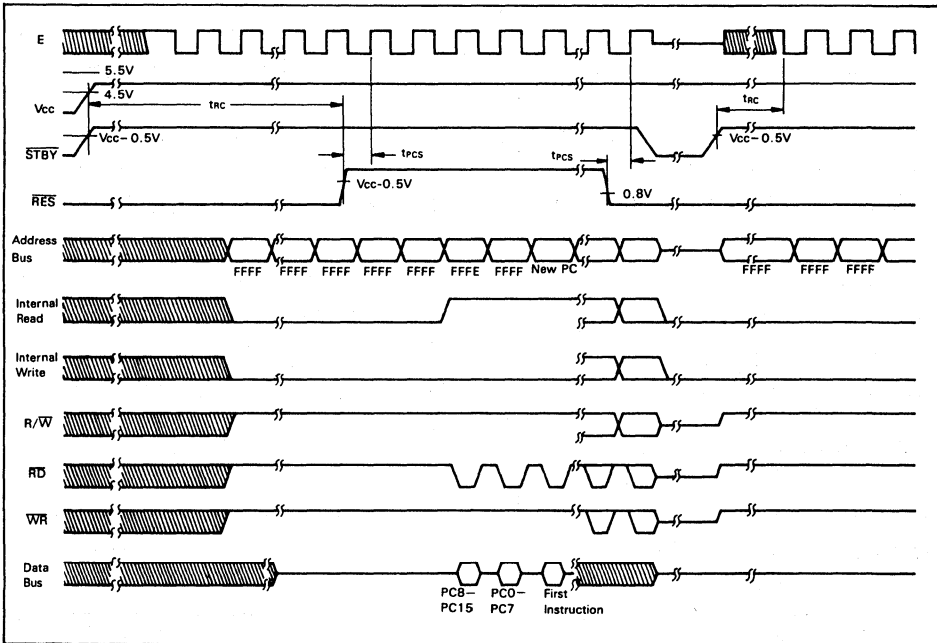


Figure I-14. Reset Timing

## I.2 HD6303Y, HD63A03Y, HD63B03Y, HD63C03Y Electrical Characteristics

### Absolute Maximum Ratings

Item	Symbol	Value	Unit
Supply voltage	V <sub>CC</sub>	-0.3 to +7.0	V
Input voltage	V <sub>in</sub>	-0.3 to V <sub>CC</sub> +0.3	V
Operating temperature	T <sub>opr</sub>	-20 to +70	°C
Storage temperature	T <sub>stg</sub>	-55 to +150	°C

Note: This product has protection circuits in input terminal from high static electricity voltage and high electric field. But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend V<sub>in</sub>, V<sub>out</sub>: V<sub>SS</sub> ≤ (V<sub>in</sub> or V<sub>out</sub>) ≤ V<sub>CC</sub>.

### Electrical Characteristics

#### DC Characteristics

(V<sub>CC</sub> = 5.0 V ± 10%, f = 0.1 to 3.0 MHz, V<sub>SS</sub> = 0 V, Ta = -20 to +70°C)

Item		Symbol	Min	Typ	Max	Unit	Test Condition	
Input high voltage	RES, STBY	V <sub>IH</sub>	V <sub>CC</sub> -0.5		V <sub>CC</sub> +0.3	V		
	EXTAL		V <sub>CC</sub> ×0.7		V <sub>CC</sub> +0.3	V		
	Other inputs		2.0		V <sub>CC</sub> +0.3	V		
Input low voltage	All other inputs	V <sub>IL</sub>	-0.3		0.8***	V		
Input leakage current	RES, NMI, STBY, MP0, MP1	I <sub>in</sub>			1.0	μA	V <sub>in</sub> =0.5 to V <sub>CC</sub> -0.5 V	
Three state leakage current	A <sub>0</sub> -A <sub>15</sub> , D <sub>0</sub> -D <sub>7</sub> , RD WR, R/W, Ports 2, 5, 6	I <sub>TS</sub>			1.0	μA	V <sub>in</sub> =0.5 to V <sub>CC</sub> -0.5 V	
Output high voltage		V <sub>OH</sub>	2.4			V	I <sub>OH</sub> = -200 μA	
			V <sub>CC</sub> -0.7			V	I <sub>OH</sub> = -10 μA	
Output low voltage		V <sub>OL</sub>		0.4		V	I <sub>OL</sub> = 1.6 mA	
Darlington drive current	Ports 2, 6	-I <sub>OH</sub>	1.0		10.0	mA	V <sub>out</sub> = 1.5 V	
Input capacitance	All other inputs	C <sub>in</sub>			12.5	pF	V <sub>in</sub> =0V, f=1 MHz Ta=25°C	
Standby current*	Not operating	I <sub>STB</sub>		3.0	15.0	μA		
Current dissipation*		I <sub>SLP</sub>		1.5	3.0	mA	Sleeping (f=1 MHz)**	
				2.3	4.5	mA	Sleeping (f=1.5 MHz)**	
				3.0	6.0	mA	Sleeping (f=2 MHz)**	
				4.5	9.0	mA	Sleeping (f=3 MHz)**	
			I <sub>CC</sub>		7.0	10.0	mA	Operating (f=1 MHz)**
					10.5	15.0	mA	Operating (f=1.5 MHz)**
					14.0	20.0	mA	Operating (f=2 MHz)**
					21.0	30.0	mA	Operating (f=3 MHz)**
RAM standby voltage		V <sub>RAM</sub>	2.0			V		

#### Notes :

- \* V<sub>IH</sub> min=V<sub>CC</sub>-1.0V, V<sub>IH</sub> max=0.8V (All output terminals are at no load.)
- \*\* Current dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ. or max. values about current dissipations at x MHz operation are decided according to the following formula :  
 typ. value (f=x MHz) = typ. value (f=1 MHz) × x  
 max. value (f=x MHz) = max. value (f=1 MHz) × x  
 (both the sleeping and operating)
- \*\*\* In case of SCLK Input. V<sub>IL</sub> = 0.6V (-20°C ~ 0°C)

## AC Characteristics

(V<sub>CC</sub> = 5.0 V ± 10%, f = 0.1 to 3.0 MHz, V<sub>SS</sub> = 0 V, T<sub>a</sub> = -20 to +70°C)

### Bus Timing

Item	Symbol	HD6303Y			HD63A03Y			HD63B03Y			HD63C03Y			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Cycle time	t <sub>cyc</sub>	1		10	0.666		10	0.5		10	0.333		10	μs	Fig. I-15
Enable rise time	t <sub>Er</sub>			25			25			25			20	ns	
Enable fall time	t <sub>Ef</sub>			25			25			25			20	ns	
Enable pulse width high level <sup>1</sup>	PW <sub>EH</sub>	450			300			220			140			ns	
Enable pulse width low level <sup>1</sup>	PW <sub>EL</sub>	450			300			220			140			ns	
Address, R/W delay time <sup>1</sup>	t <sub>AD</sub>			250			190			160			120	ns	
Data delay time (Write)	t <sub>DDW</sub>			200			160			120			100	ns	
Data set-up time (Read)	t <sub>DSR</sub>	80			70			60			50			ns	
Address, R/W hold time <sup>1</sup>	t <sub>AH</sub>	80			50			40			20			ns	
Data hold time (Write) <sup>1</sup>	t <sub>HW</sub>	70			50			40			20			ns	
Data hold time (Read)	t <sub>HR</sub>	0			0			0			0			ns	
$\overline{RD}$ , $\overline{WR}$ pulse width <sup>1</sup>	PW <sub>RW</sub>	450			300			220			140			ns	
$\overline{RD}$ , $\overline{WR}$ delay time	t <sub>RWD</sub>			40			40			40			40	ns	
$\overline{RD}$ , $\overline{WR}$ hold time	t <sub>HRW</sub>			20			20			20			20	ns	
$\overline{LIR}$ delay time	t <sub>DLR</sub>			200			160			120			80	ns	
$\overline{LIR}$ hold time	t <sub>HLR</sub>	10			10			10			5			ns	
Peripheral read access time <sup>1</sup>	t <sub>ACC</sub>										180			ns	
MR set-up time <sup>1</sup>	t <sub>SMR</sub>	400			280			230			170			ns	Fig. I-16
MR hold time <sup>1</sup>	t <sub>HMR</sub>			100			70			50			25	ns	
E clock pulse width at MR	PW <sub>EMR</sub>			9			9			9			9	μs	
Processor control set-up time	t <sub>PCS</sub>	200			200			200			100			ns	Figs. I-17, I-27, I-28
Processor control rise time	t <sub>PCr</sub>			100			100			100			50	ns	Figs. I-16, I-17
Processor control fall time	t <sub>PCf</sub>			100			100			100			50	ns	
BA delay time	t <sub>BA</sub>			250			190			160			120	ns	Fig. I-17
Oscillator stabilization time	t <sub>RC</sub>	20			20			20			20			ms	Fig. I-28
Reset pulse width	PW <sub>RST</sub>	3			3			3			3			t <sub>cyc</sub>	

Note: 1. These timings change in approximate proportion to t<sub>cyc</sub>. The figures in this characteristics represent those when t<sub>cyc</sub> is minimum (= in the highest speed operation).

## Peripheral Port Timing

Item	Symbol	HD6303Y			HD63A03Y			HD63B03Y			HD63C03Y			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Peripheral data set-up time (Ports 2, 5, 6)	t <sub>pDSU</sub>	200		200			200			200			200	ns	Fig. I-19
Peripheral data hold time (Ports 2, 5, 6)	t <sub>pDH</sub>	200		200			200			200			200	ns	
Delay time (From enable fall edge to peripheral output) (Ports 2, 5, 6, 7)	t <sub>pWD</sub>			300			300			300			300	ns	Fig. I-20
Input strobe pulse width	t <sub>pWIS</sub>	200		200			200			200			200	ns	Fig. I-35
Input data hold time (Port 6)	t <sub>IH</sub>	150		150			150			150			150	ns	
Input data set-up time (Port 6)	t <sub>IS</sub>	100		100			100			100			100	ns	
Output strobe time	t <sub>OSD1</sub>			200			200			200			200	ns	Fig. I-25
	t <sub>OSD2</sub>														

## Timer, SCI Timing

Item	Symbol	HD6303Y			HD63A03Y			HD63B03Y			HD63C03Y			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Timer 1 input pulse width	t <sub>PWT</sub>	2.0		2.0			2.0			2.0			2.0	t <sub>cyc</sub>	Fig. I-23
Delay time (enable positive transition to timer output)	t <sub>TOD</sub>			400			400			400			400	ns	Figs. I-21, I-22
SCI input clock cycle (Async mode)	t <sub>Scyc</sub>	1.0		1.0			1.0			1.0			1.0	t <sub>cyc</sub>	Fig. I-23
SCI input clock cycle (Clock sync.)		2.0		2.0			2.0			2.0			2.0	t <sub>cyc</sub>	Fig. I-18
SCI transmit data delay time (Clock sync. mode)	t <sub>TXD</sub>			220			220			220			220	ns	Fig. I-18
SCI receive data set-up time (Clock sync. mode)	t <sub>SRX</sub>	260		260			260			260			260	ns	
SCI receive data hold time (Clock sync. mode)	t <sub>HRX</sub>	100		100			100			100			100	ns	
SCI input clock pulse width	t <sub>PWSCK</sub>	0.4		0.6	0.4		0.6	0.4		0.6	0.4		0.6	t <sub>Scyc</sub>	Fig. I-23
Timer 2 input clock cycle	t <sub>cyc</sub>	2.0		2.0			2.0			2.0			2.0	t <sub>cyc</sub>	
Timer 2 input clock pulse width	t <sub>PWTCK</sub>	200		200			200			200			200	ns	
Timer 1 + 2, SCI input clock rise time	t <sub>CKr</sub>			100			100			100			100	50	ns
Timer 1 + 2, SCI input clock fall time	t <sub>CKf</sub>			100			100			100			100	50	ns

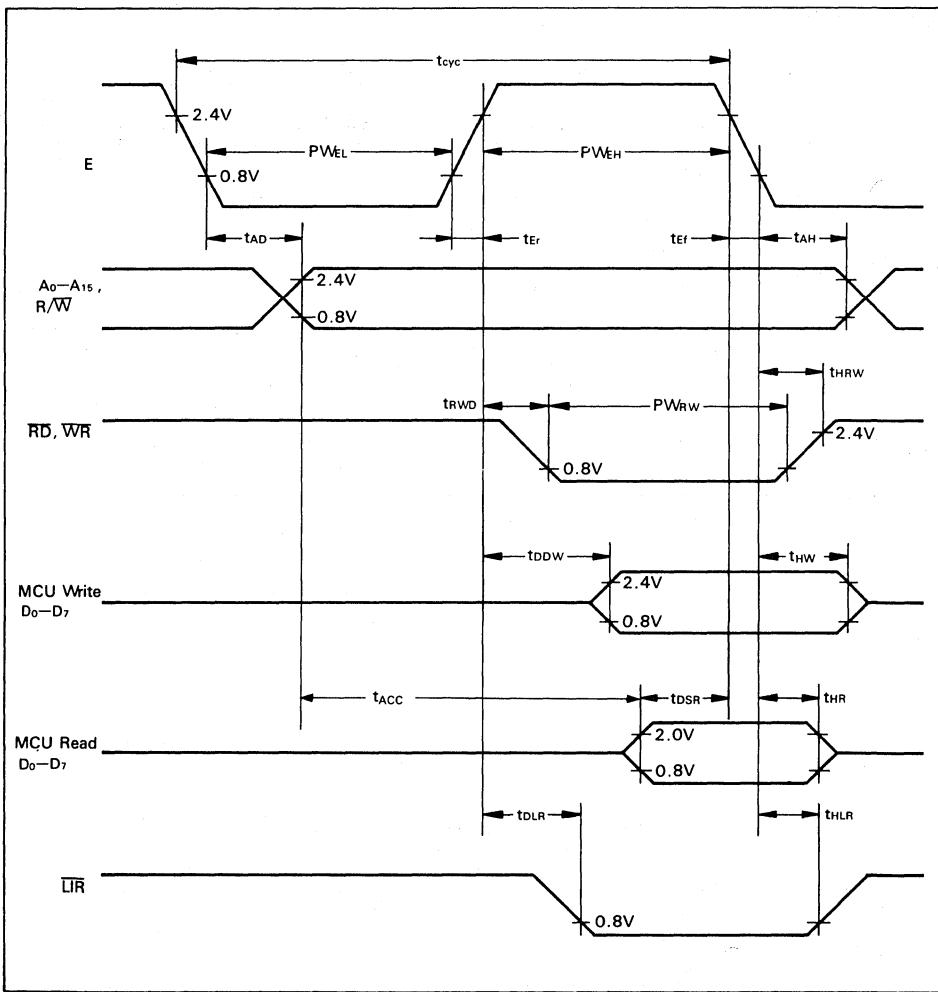


Figure I-15. Bus Timing

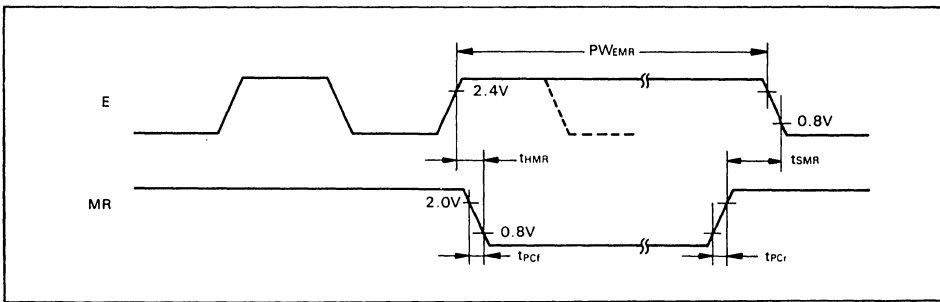


Figure I-16. Memory Ready and E Clock Timing

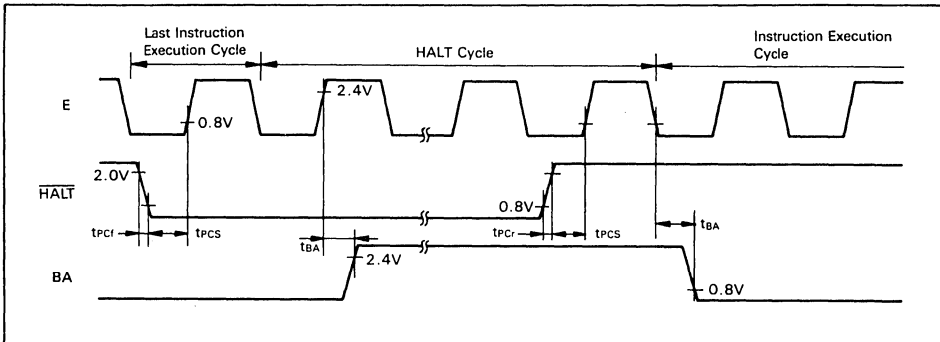


Figure I-17. HALT and BA Timing

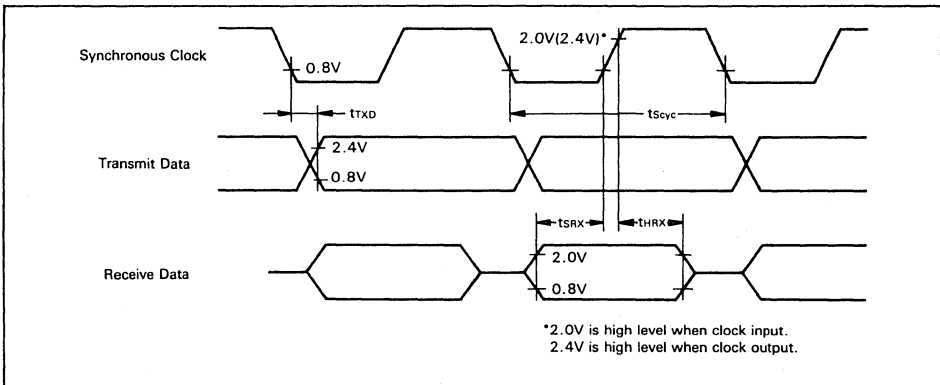


Figure I-18. SCI Clocked Synchronous Timing

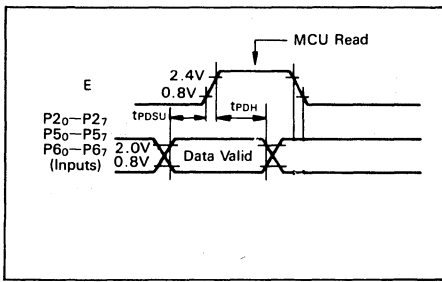


Figure I-19. Port Data Set-up and Hold Times (MCU Read)

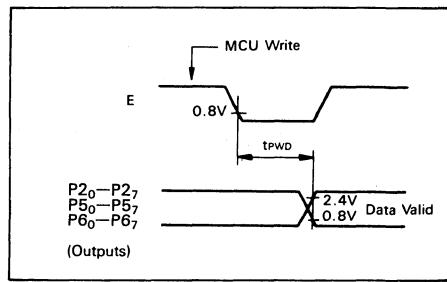


Figure I-20. Port Data Delay Times (MCU Write)

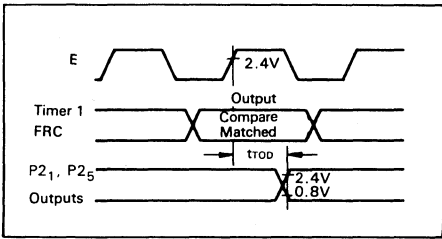


Figure I-21. Timer 1 Output Timing

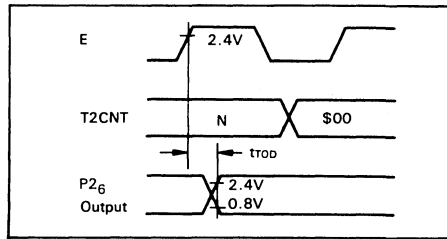


Figure I-22. Timer 2 Output Timing

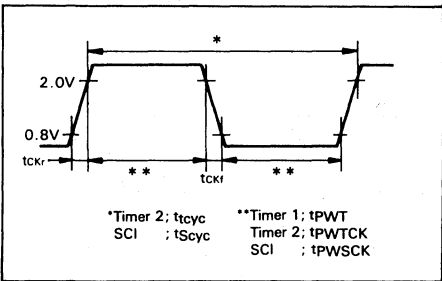


Figure I-23. Timer 1-2, SCI Input Clock Timing

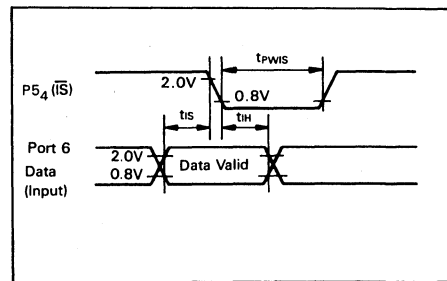


Figure I-24. Port 6 Input Latch Timing

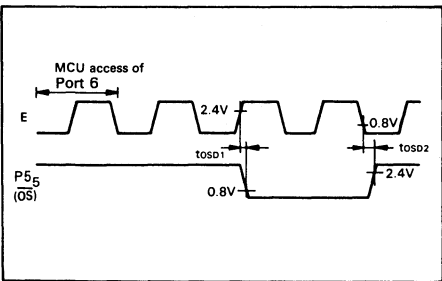


Figure I-25. Output Strobe Timing

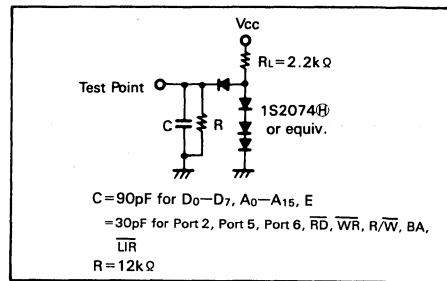


Figure I-26. Bus Timing Test Loads (TTL Load)

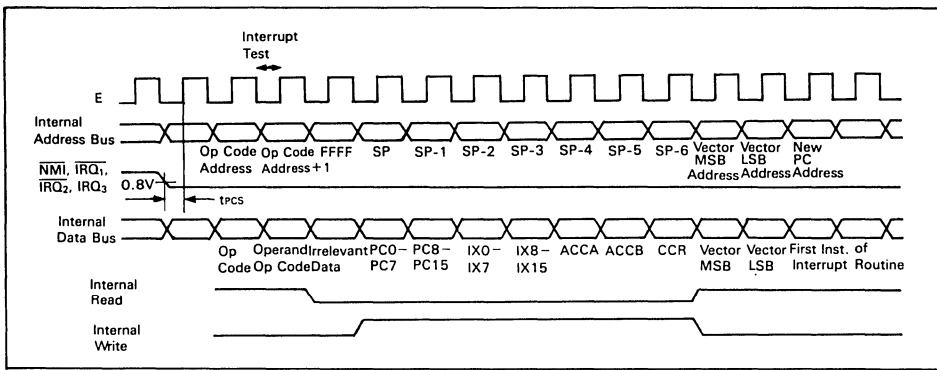


Figure I-27. Interrupt Sequence

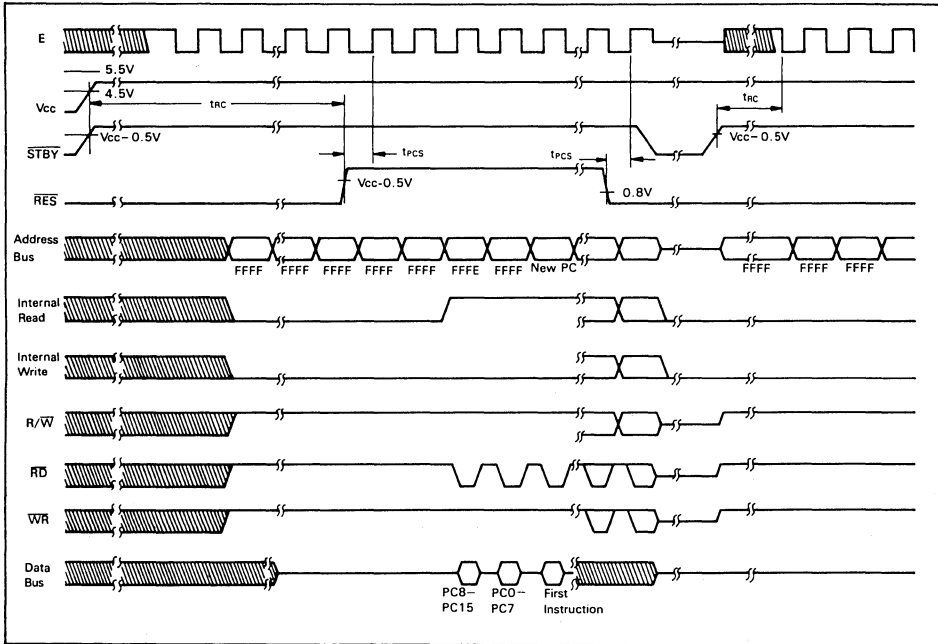


Figure I-28. Reset Timing



# 1.3 HD63701Y0, HD637A01Y0, HD637B01Y0 Electrical Characteristics

## Absolute Maximum Ratings

Item	Symbol	Value	Unit
Supply voltage	$V_{CC}$	-0.3 to +7.0	V
$V_{PP}$ voltage	$V_{PP}$	-0.3 to +13.0	V
Input voltage	$V_{in}$	-0.3 to $V_{CC}+0.3$	V
Operating temperature	$T_{opr}$	0 to +70	°C
Storage temperature	$T_{stg}$	-55 to +125	°C

Note: This product has protection circuits in input terminal from high static electricity voltage and high electric field. But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend  $V_{in}$ ,  $V_{out}$ :  $V_{SS} \leq (V_{in} \text{ or } V_{out}) \leq V_{CC}$ .

## Electrical Characteristics

### DC Characteristics

( $V_{CC}=5.0\text{ V} \pm 10\%$ ,  $f=0.1$  to 2.0 MHz,  $V_{SS}=0\text{ V}$ ,  $T_a=0$  to +70 °C, unless otherwise noted.)

Item		Symbol	Min	Typ	Max	Unit	Test Condition
Input high voltage	RES, STBY, MP <sub>0</sub> , MP <sub>1</sub>	$V_{IH}$	$V_{CC}-0.5$		$V_{CC}+0.3$	V	
	EXTAL		$V_{CC} \times 0.7$		$V_{CC}+0.3$	V	
	Other inputs		2.0		$V_{CC}+0.3$	V	
Input low voltage	RES, MP <sub>0</sub> , MP <sub>1</sub> , SCLK(P2) <sup>3</sup>	$V_{IL}$	-0.3		0.6	V	
	All other inputs		-0.3		0.8	V	
Input leakage current	RES	$ I_{in} $			10.0	μA	$V_{in}=0.5$ to $V_{CC}-0.5\text{ V}$
	NMI, STBY, MP <sub>0</sub> , MP <sub>1</sub>				1.0	μA	
Three state leakage current	Ports 1, 2, 3, 4, 5, 6, 7	$ I_{TSI} $			1.0	μA	$V_{in}=0.5$ to $V_{CC}-0.5\text{ V}$
Output high voltage		$V_{OH}$	2.4			V	$I_{OH}=-200\text{ μA}$
			$V_{CC}-0.7$			V	$I_{OH}=-10\text{ μA}$
Output low voltage		$V_{OL}$			0.4	V	$I_{OL}=1.6\text{ mA}$
Darlington drive current	Ports 2, 6	$-I_{OH}$	1.0		10.0	mA	$V_{out}=1.5\text{ V}$
Input capacitance	RES	$C_{in}$			65	pF	$V_{in}=0\text{ V}$ , $f=1\text{ MHz}$ , $T_a=25^\circ\text{C}$
	All other inputs				12.5	pF	
Standby current	Not operating	$I_{STB}$		3.0	15.0	μA	
Current dissipation <sup>1</sup>		$I_{SLP}$		1.5	3.0	mA	Sleeping ( $f=1\text{ MHz}^2$ )
				2.3	4.5	mA	Sleeping ( $f=1.5\text{ MHz}^2$ )
				3.0	6.0	mA	Sleeping ( $f=2\text{ MHz}^2$ )
		$I_{CC}$		7.0	10.0	mA	Operating ( $f=1\text{ MHz}^2$ )
				10.5	15.0	mA	Operating ( $f=1.5\text{ MHz}^2$ )
				14.0	20.0	mA	Operating ( $f=2\text{ MHz}^2$ )
RAM standby voltage		$V_{RAM}$	2.0			V	

Notes :

- $V_{IH\ min}=V_{CC}-1.0\text{V}$ ,  $V_{IL\ max}=0.8\text{V}$  (All output terminals are at no load.)
- Current dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ. or max. values about current dissipations at x MHz operation are decided according to the following formula:
 

typ. value	( $f=x\text{ MHz}$ )	=	typ. value ( $f=1\text{ MHz}$ )	$\times x$
max. value	( $f=x\text{ MHz}$ )	=	max. value ( $f=1\text{ MHz}$ )	$\times x$

 (both the sleeping and operating)
- Only serial clock use.



## AC Characteristics

( $V_{CC}=5.0\text{ V} \pm 10\%$ ,  $f=0.1$  to  $2.0\text{ MHz}$ ,  $V_{SS}=0\text{ V}$ ,  $T_a=0$  to  $+70\text{ }^\circ\text{C}$ , unless otherwise noted.)

### Bus Timing

Item	Symbol	HD63701Y0			HD637A01Y0			HD637B01Y0			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Cycle time	$t_{cyc}$	1		10	0.666		10	0.5		10	$\mu\text{s}$	Fig. 40
Enable rise time	$t_{Er}$			25			25			25	ns	
Enable fall time	$t_{Ef}$			25			25			25	ns	
Enable pulse width high level <sup>1</sup>	$PW_{EH}$	450			300			220			ns	
Enable pulse width low level <sup>1</sup>	$PW_{EL}$	450			300			220			ns	
Address, $R/\overline{W}$ delay time <sup>1</sup>	$t_{AD}$			250			190			160	ns	
Data delay time (Write)	$t_{DDW}$			200			160			120	ns	
Data set-up time (Read)	$t_{DSR}$	80			70			60			ns	
Address, $R/\overline{W}$ hold time <sup>1</sup>	$t_{AH}$	80			50			40			ns	
Data hold time (Write) <sup>1</sup>	$t_{HW}$	80			50			40			ns	
Data hold time (Read)	$t_{HR}$	0			0			0			ns	
$\overline{RD}$ , $\overline{WR}$ pulse width <sup>1</sup>	$PW_{RW}$	450			300			220			ns	
$\overline{RD}$ , $\overline{WR}$ delay time	$t_{RWD}$			40			40			40	ns	
$\overline{RD}$ , $\overline{WR}$ hold time	$t_{HRW}$			20			20			20	ns	
$\overline{LIR}$ delay time	$t_{DLR}$			200			160			120	ns	
$\overline{LIR}$ hold time	$t_{HLR}$	10			10			10			ns	
MR set-up time <sup>1</sup>	$t_{SMR}$	400			280			230			ns	Fig. 41
MR hold time <sup>1</sup>	$t_{HMR}$			100			70			50	ns	
E clock pulse width at MR	$PW_{EMR}$			9			9			9	$\mu\text{s}$	
Processor control set-up time	$t_{PCS}$	200			200			200			ns	Figs. 42, 52, 53
Processor control rise time	$t_{PCr}$			100			100			100	ns	Figs. 41, 42
Processor control fall time	$t_{PCf}$			100			100			100	ns	
BA delay time	$t_{BA}$			250			190			160	ns	Fig. 42
Oscillator stabilization time	$t_{RC}$	20			20			20			ms	Fig. 53
Reset pulse width	$PW_{RST}$	3			3			3			$t_{cyc}$	

Note: 1. These timings change in approximate proportion to  $t_{cyc}$ . The figures in this characteristics represent those when  $t_{cyc}$  is minimum (=in the highest speed operation).

## Peripheral Port Timing

Item	Symbol	HD63701Y0			HD637A01Y0			HD637B01Y0			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Peripheral data set-up time	(Ports 1, 2, 3, 4, 5, 6) tPDSU	200			200			200			ns	Fig. 44
Peripheral data hold time	(Ports 1, 2, 3, 4, 5, 6) tPDH	200			200			200			ns	
Delay time (From enable fall edge to peripheral output)	(Ports 1, 2, 3, 4, 5, 6, 7) tPWD			300			300			300	ns	Fig. 45
Input strobe pulse width	tPWIS	200			200			200			ns	Fig. 49
Input data hold time (Port 6)	tIH	150			150			150			ns	
Input data set-up time (Port 6)	tIS	100			100			100			ns	
Output strobe delay time	tOSD1			200			200			200	ns	Fig. 50
	tOSD2											

## Timer, SCI Timing

Item	Symbol	HD63701Y0			HD637A01Y0			HD637B01Y0			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Timer 1 input pulse width	tPWT	2.0			2.0			2.0			t <sub>cyc</sub>	Fig. 48
Delay time (enable positive transition to timer output)	tTOD			400			400			400	ns	Figs. 46, 47
SCI input clock cycle	(Async. mode) tScyc	1.0			1.0			1.0			t <sub>cyc</sub>	Fig. 48
	(Clock sync.)	2.0			2.0			2.0			t <sub>cyc</sub>	Fig. 43
SCI transmit data delay time (Clock sync. mode)	tTXD			220			220			220	ns	Fig. 43
SCI receive data set-up time (Clock sync. mode)	tSRX	260			260			260			ns	
SCI receive data hold time (Clock sync. mode)	tHRX	100			100			100			ns	
SCI input clock pulse width	tPWCK	0.4		0.6	0.4		0.6	0.4		0.6	t <sub>Scyc</sub>	Fig. 48
Timer 2 input clock cycle	t <sub>cyc</sub>	2.0			2.0			2.0			t <sub>cyc</sub>	
Timer 2 input clock pulse width	tPWCK	200			200			200			ns	
Timer 1 • 2, SCI input clock rise time	tCKr			100			100			100	ns	
Timer 1 • 2, SCI input clock fall time	tCKf			100			100			100	ns	

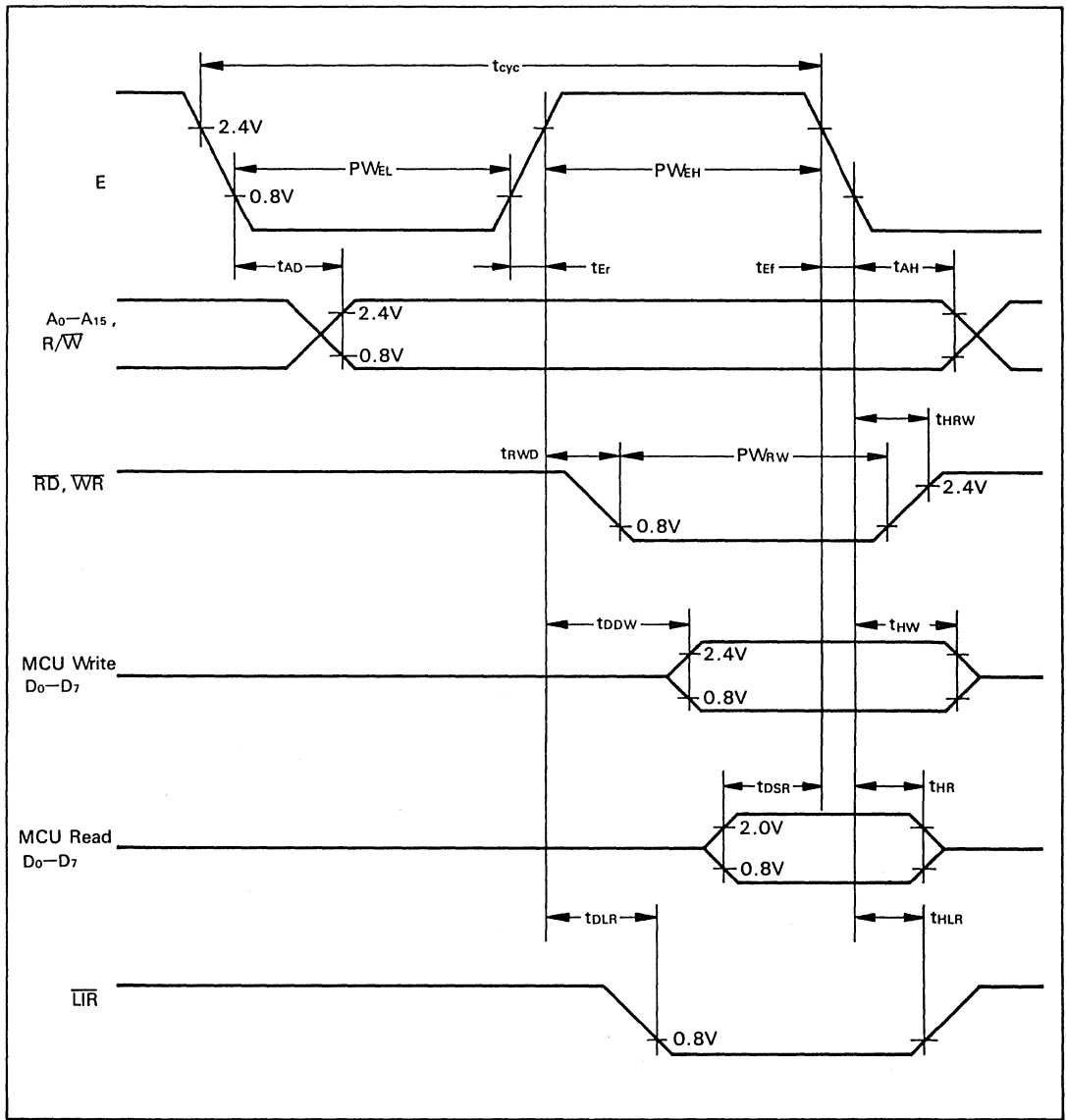


Figure 1-29. Mode 1, Mode 2 Bus Timing

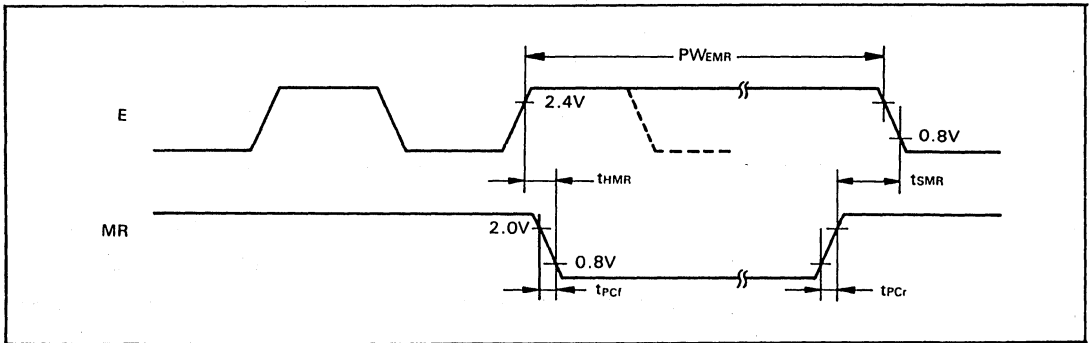


Figure 1-30. Memory Ready and E Clock Timing

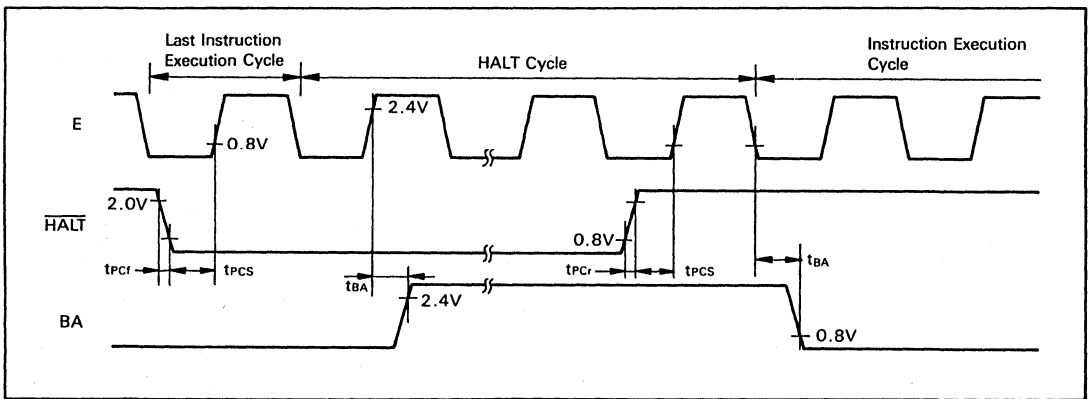


Figure 1-31.  $\overline{HALT}$  and BA Timing

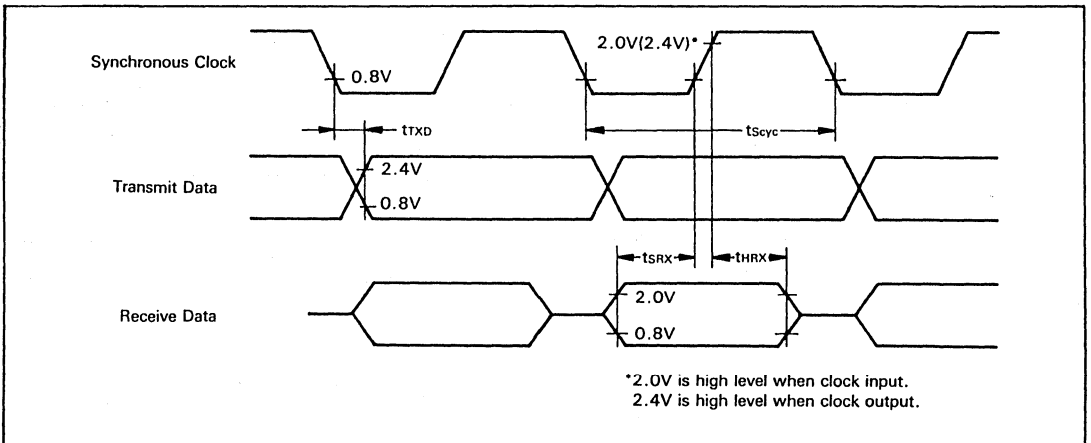


Figure 1-32. SCI Clocked Synchronous Timing

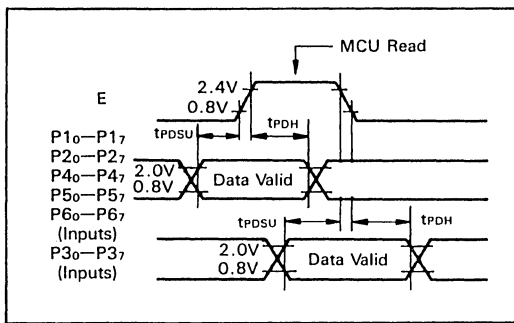


Figure 1-33. Port Data Set-up and Hold Times (MCU Read)

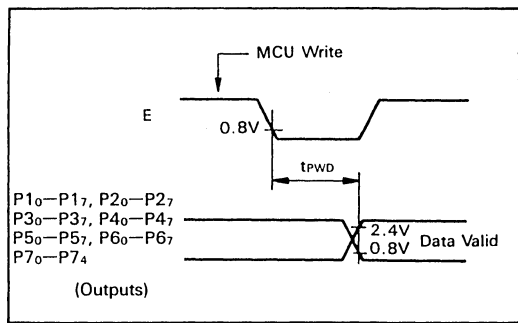


Figure 1-34. Port Data Delay Times (MCU Write)

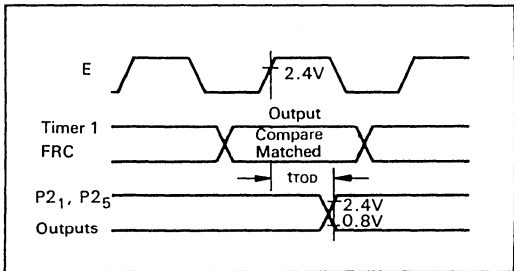


Figure 1-35. Timer 1 Output Timing

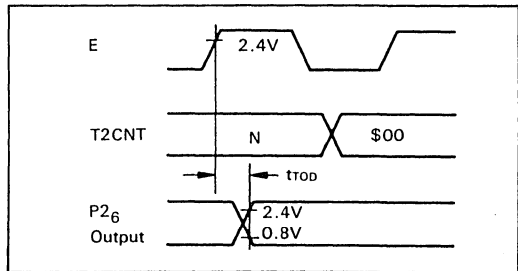


Figure 1-36. Timer 2 Output Timing

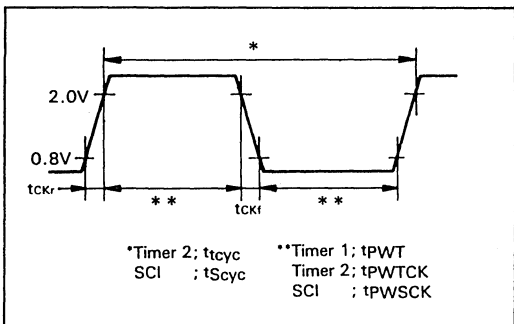


Figure 1-37. Timer 1, 2 SCI Input Clock Timing

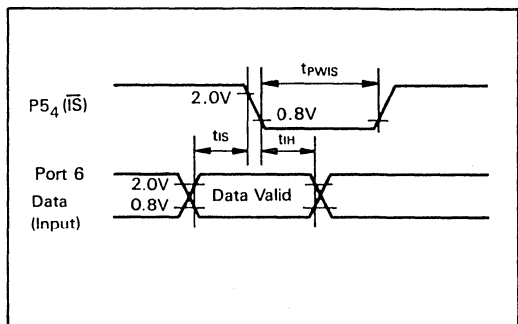


Figure 1-38. Port 6 Input Latch Timing

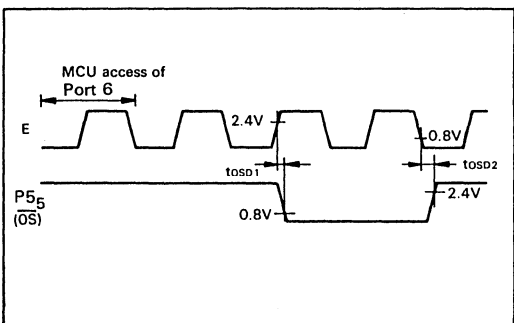


Figure 1-39. Output Strobe Timing

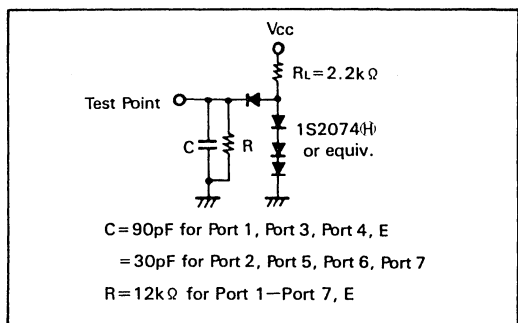


Figure 1-40. Bus Timing Test Loads (TTL Load)

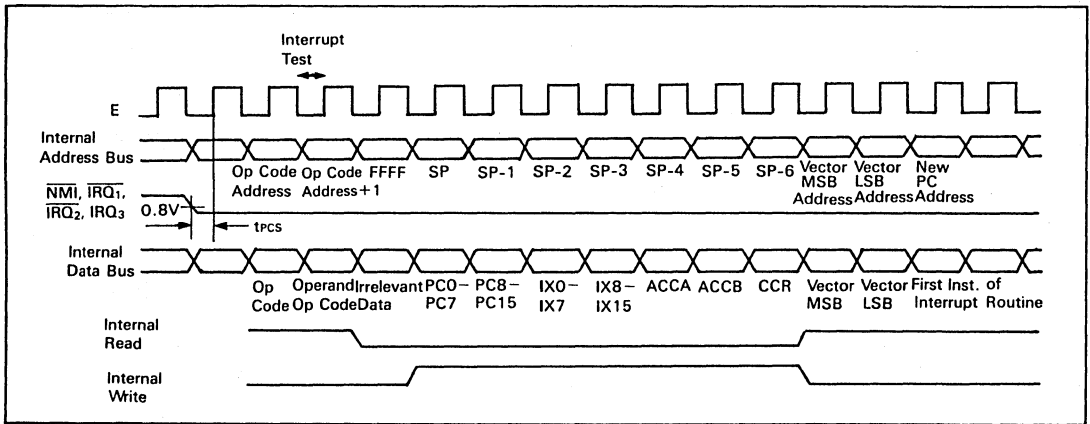


Figure 1-41. Interrupt Sequence

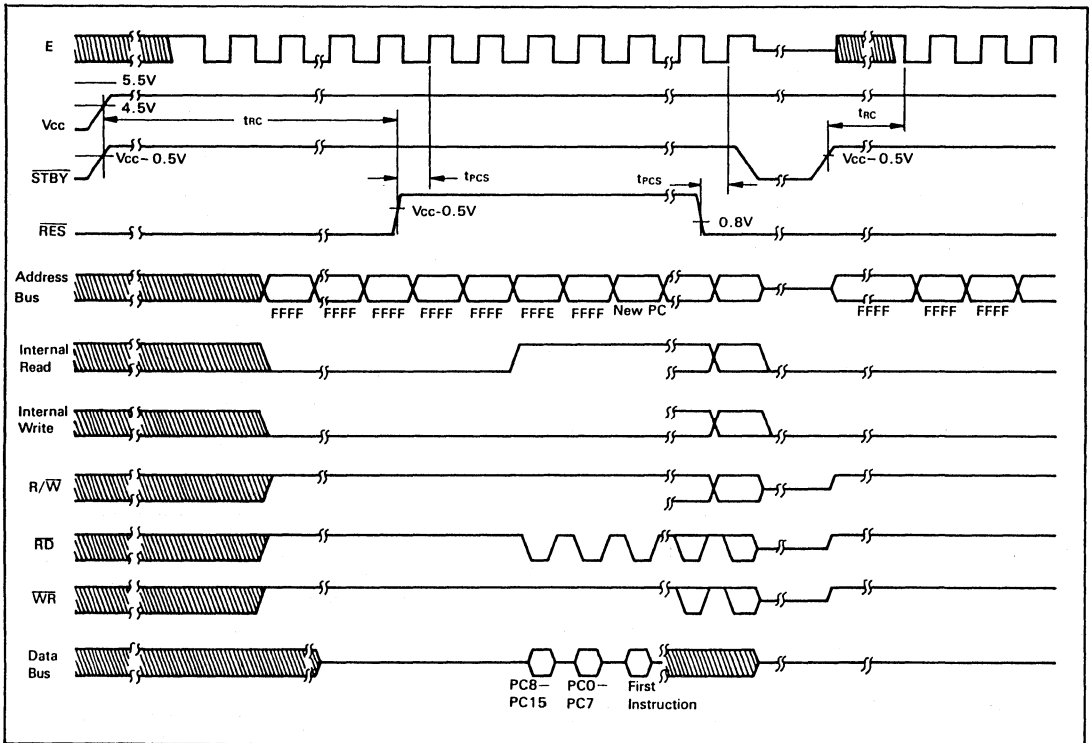


Figure 1-42. Reset Timing

# Programming Electrical Characteristics

## DC Characteristics

( $V_{CC}=6\text{ V} \pm 0.25\text{ V}$ ,  $V_{PP}=12.5\text{ V} \pm 0.3\text{ V}$ ,  $V_{SS}=0\text{ V}$ ,  $T_a=25^\circ\text{C} \pm 5^\circ\text{C}$ , unless otherwise notes.)

Item		Symbol	Min	Typ	Max	Unit	Test Condition
Input high voltage	$O_0-O_7, A_0-A_{14}, \overline{OE}, \overline{CE}$	$V_{IH}$	2.2	—	$V_{CC}+0.3$	V	
Input low voltage	$O_0-O_7, A_0-A_{14}, \overline{OE}, \overline{CE}$	$V_{IL}$	-0.3	—	0.8	V	
Output high voltage	$O_0-O_7$	$V_{OH}$	2.4	—	—	V	$I_{OH} = -200\mu\text{A}$
Output low voltage	$O_0-O_7$	$V_{OL}$	—	—	0.45	V	$I_{OL} = 1.6\text{mA}$
Input leakage current	$O_0-O_7, A_0-A_{14}, \overline{OE}, \overline{CE}$	$ I_{LI} $	—	—	2	$\mu\text{A}$	$V_{in} = 5.25\text{V}/0.5\text{V}$
$V_{CC}$ current		$I_{CC}$	—	—	30	mA	
$V_{PP}$ current		$I_{PP}$	—	—	40	mA	

## AC Characteristics

( $V_{CC}=6\text{ V} \pm 0.25\text{ V}$ ,  $V_{PP}=12.5\text{ V} \pm 0.3\text{ V}$ ,  $T_a=25^\circ\text{C} \pm 5^\circ\text{C}$ , unless otherwise noted.)

Item	Symbol	Min	Typ	Max	Unit	Test Condition
Address set-up time	$t_{AS}$	2	—	—	$\mu\text{s}$	Fig. 54*
$\overline{OE}$ set-up time	$t_{OES}$	2	—	—	$\mu\text{s}$	
Data set-up time	$t_{DS}$	2	—	—	$\mu\text{s}$	
Address hold time	$t_{AH}$	0	—	—	$\mu\text{s}$	
Data hold time	$t_{DH}$	2	—	—	$\mu\text{s}$	
Output disable delay time	$t_{DF}$	—	—	130	ns	
$V_{PP}$ set-up time	$t_{VPS}$	2	—	—	$\mu\text{s}$	
Program pulse width	$t_{PW}$	0.95	1.0	1.05	ms	
$\overline{CE}$ pulse width when overprogramming	$t_{OPW}$	2.85	—	78.75	ms	
$V_{CC}$ set-up time	$t_{VCS}$	2	—	—	$\mu\text{s}$	
Data output delay time	$t_{OE}$	0	—	500	ns	

Note: \* Input Pulse level 0.8~2.2V  
 Input rising/falling time  $\leq 20\text{ns}$   
 Timing reference level { input : 1.0V, 2.0V  
 output : 0.8V, 2.0V

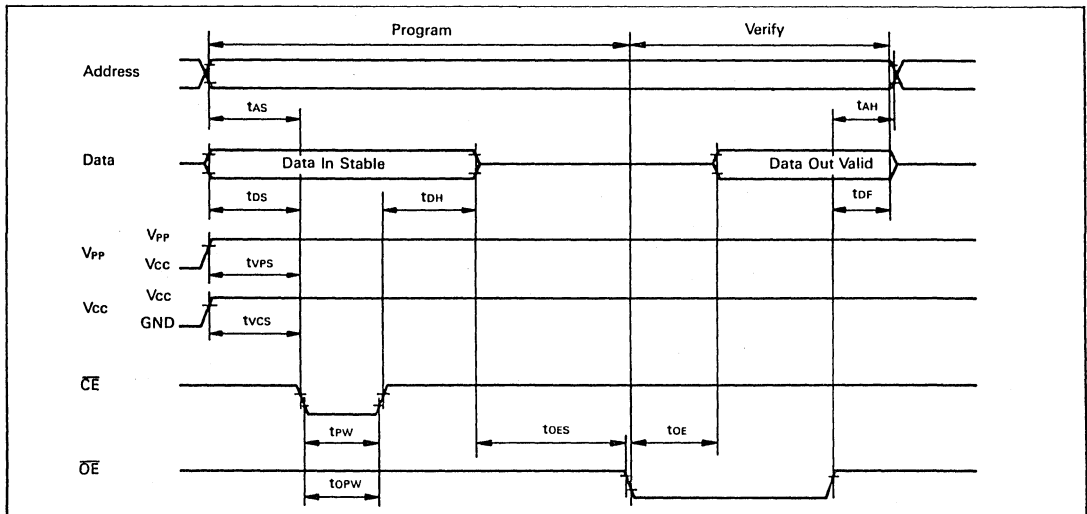


Figure 1-43. PROM Programming/Verify timing



# Appendix II. Instruction Execution Cycles

## II.1 Instruction Execution Cycles

So attention is necessary to the counting of the instruction cycles because it is different from the existent one ..... op-code fetch to the next instruction op-code.

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
-----------------------------	--------	---------	-------------	-----	-----------------	-----------------	------------------	----------

### IMMEDIATE

ADC ADD	2	1	Op Code Address+1	1	0	1	1	Operand Data
AND BIT		2	Op Code Address+2	1	0	1	0	Next Op Code
CMP EOR								
LDA ORA								
SBC SUB								
ADDD CPX	3	1	Op Code Address+1	1	0	1	1	Operand Data (MSB)
LDD LDS		2	Op Code Address+2	1	0	1	1	Operand Data (LSB)
LDX SUBD		3	Op Code Address+3	1	0	1	0	Next Op Code

### DIRECT

ADC ADD	3	1	Op Code Address+1	1	0	1	1	Address of Operand (LSB)
AND BIT		2	Address of Operand	1	0	1	1	Operand Data
CMP EOR		3	Op Code Address+2	1	0	1	0	Next Op Code
LDA ORA								
SBC SUB								
STA	3	1	Op Code Address+1	1	0	1	1	Destination Address
		2	Destination Address	0	1	0	1	Accumulator Data
		3	Op Code Address+2	1	0	1	0	Next Op Code
ADDD CPX	4	1	Op Code Address+1	1	0	1	1	Address of Operand (LSB)
LDD LDS		2	Address of Operand	1	0	1	1	Operand Data (MSB)
LDX SUBD		3	Address of Operand+1	1	0	1	1	Operand Data (LSB)
		4	Op Code Address+2	1	0	1	0	Next Op Code
STD STS	4	1	Op Code Address+1	1	0	1	1	Destination Address (LSB)
STX		2	Destination Address	0	1	0	1	Register Data (MSB)
		3	Destination Address+1	0	1	0	1	Register Data (LSB)
		4	Op Code Address+2	1	0	1	0	Next Op Code
JSR	5	1	Op Code Address+1	1	0	1	1	Jump Address (LSB)
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer-1	0	1	0	1	Return Address (MSB)
		5	Jump Address	1	0	1	0	First Subroutine Op Code
TIM	4	1	Op Code Address+1	1	0	1	1	Immediate Data
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	Op Code Address+3	1	0	1	0	Next Op Code
AIM EIM	6	1	Op Code Address+1	1	0	1	1	Immediate Data
OIM		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	FFFF	1	1	1	1	Restart Address (LSB)
		5	Address of Operand	0	1	0	1	New Operand Data
		6	Op Code Address+3	1	0	1	0	Next Op Code

(continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
-----------------------------	--------	---------	-------------	-----	-----------------	-----------------	------------------	----------

### INDEXED

JMP	3	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Jump Address	1	0	1	0	First Op Code of Jump Routine
ADC ADD AND BIT CMP EOR LDA ORA SBC SUB TST	4	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	1	0	1	1	Operand Data
		4	Op Code Address+2	1	0	1	0	Next Op Code
STA	4	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	0	1	0	1	Accumulator Data
		4	Op Code Address+2	1	0	1	0	Next Op Code
ADDD CPX LDD LDS LDX SUBD ADD	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	1	0	1	1	Operand Data (MSB)
		4	IX+Offset+1	1	0	1	1	Operand Data (LSB)
		5	Op Code Address+2	1	0	1	0	Next Op Code
STD STS STX	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	0	1	0	1	Register Data (MSB)
		4	IX+Offset+1	0	1	0	1	Register Data (LSB)
		5	Op Code Address+2	1	0	1	0	Next Op Code
JSR	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer-1	0	1	0	1	Return Address (MSB)
		5	IX+Offset	1	0	1	0	First Subroutine Op Code
ASL ASR COM DEC INC LSR NEG ROL ROR	6	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	1	0	1	1	Operand Data
		4	FFFF	1	1	1	1	Restart Address (LSB)
		5	IX+Offset	0	1	0	1	New Operand Data
		6	Op Code Address+2	1	0	1	0	Next Op Code
TIM	5	1	Op Code Address+1	1	0	1	1	Immediate Data
		2	Op Code Address+2	1	0	1	1	Offset
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	IX+Offset	1	0	1	1	Operand Data
		5	Op Code Address+3	1	0	1	0	Next Op Code
CLR	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	1	0	1	1	Operand Data
		4	IX+Offset	0	1	0	1	00
		5	Op Code Address+2	1	0	1	0	Next Op Code
AIM EIM OIM	7	1	Op Code Address+1	1	0	1	1	Immediate Data
		2	Op Code Address+2	1	0	1	1	Offset
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	IX+Offset	1	0	1	1	Operand Data
		5	FFFF	1	1	1	1	Restart Address (LSB)
		6	IX+Offset	0	1	0	1	New Operand Data
		7	Op Code Address+3	1	0	1	0	Next Op Code

(continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	RD	WR	LIR	Data Bus
-----------------------------	--------	---------	-------------	-----	----	----	-----	----------

**EXTEND**

JMP	3	1	Op Code Address+1	1	0	1	1	Jump Address (MSB)
		2	Op Code Address+2	1	0	1	1	Jump Address (LSB)
		3	Jump Address	1	0	1	0	Next Op Code
ADC ADD TST AND BIT CMP EOR LDA ORA SBC SUB	4	1	Op Code Address+1	1	0	1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	Op Code Address+3	1	0	1	0	Next Op Code
STA	4	1	Op Code Address+1	1	0	1	1	Destination Address (MSB)
		2	Op Code Address+2	1	0	1	1	Destination Address (LSB)
		3	Destination Address	0	1	0	1	Accumulator Data
		4	Op Code Address+3	1	0	1	0	Next Op Code
ADDD CPX LDD LDS LDX SUBD	5	1	Op Code Address+1	1	0	1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data (MSB)
		4	Address of Operand+1	1	0	1	1	Operand Data (LSB)
		5	Op Code Address+3	1	0	1	0	Next Op Code
STD STS STX	5	1	Op Code Address+1	1	0	1	1	Destination Address (MSB)
		2	Op Code Address+2	1	0	1	1	Destination Address (LSB)
		3	Destination Address	0	1	0	1	Register Data (MSB)
		4	Destination Address+1	0	1	0	1	Register Data (LSB)
		5	Op Code Address+3	1	0	1	0	Next Op Code
JSR	6	1	Op Code Address+1	1	0	1	1	Jump Address (MSB)
		2	Op Code Address+2	1	0	1	1	Jump Address (LSB)
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	Stack Pointer	0	1	0	1	Return Address (LSB)
		5	Stack Pointer-1	0	1	0	1	Return Address (MSB)
		6	Jump Address	1	0	1	0	First Subroutine Op Code
ASL ASR COM DEC INC LSR NEG ROL ROR	6	1	Op Code Address+1	1	0	1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	FFFF	1	1	1	1	Restart Address (LSB)
		5	Address of Operand	0	1	0	1	New Operand Data
		6	Op Code Address+3	1	0	1	0	Next Op Code
CLR	5	1	Op Code Address+1	1	0	1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	Address of Operand	0	1	0	1	00
		5	Op Code Address+3	1	0	1	0	Next Op Code

(continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
-----------------------------	--------	---------	-------------	-----	-----------------	-----------------	------------------	----------

**IMPLIED**

ABA ABX ASL ASLD ASR CBA CLC CLI CLR CLV COM DEC DES DEX INC INS INX LSR LSRD ROL ROR NOP SBA SEC SEI SEV TAB TAP TBA TPA TST TSX TXS	1	1	Op Code Address+1	1	0	1	0	Next Op Code
DAA XGDX	2	1 2	Op Code Address+1 FFFF	1 1	0 1	1 1	0 1	Next Op Code Restart Address (LSB)
PULA PULB	3	1 2 3	Op Code Address+1 FFFF Stack Pointer+1	1 1 1	0 1 0	1 1 1	0 1 1	Next Op Code Restart Address (LSB) Data from Stack
PSHA PSHB	4	1 2 3 4	Op Code Address+1 FFFF Stack Pointer Op Code Address+1	1 1 0 1	0 1 1 0	1 1 0 1	1 1 0 0	Next Op Code Restart Address (LSB) Accumulator Data Next Op Code
PULX	4	1 2 3 4	Op Code Address+1 FFFF Stack Pointer+1 Stack Pointer+2	1 1 1 1	0 1 0 0	1 1 1 1	0 1 1 1	Next Op Code Restart Address (LSB) Data from Stack (MSB) Data from Stack (LSB)
PSHX	5	1 2 3 4 5	Op Code Address+1 FFFF Stack Pointer Stack Pointer-1 Op Code Address+1	1 1 0 0 1	0 1 1 0 0	1 1 0 0 1	1 1 0 1 0	Next Op Code Restart Address (LSB) Index Register (LSB) Index Register (MSB) Next Op Code
RTS	5	1 2 3 4 5	Op Code Address+1 FFFF Stack Pointer+1 Stack Pointer+2 Return Address	1 1 1 1 1	0 1 0 0 0	1 1 1 1 1	1 1 1 1 0	Next Op Code Restart Address (LSB) Return Address (MSB) Return Address (LSB) First Op Code of Return Routine
MUL	7	1 2 3 4 5 6 7	Op Code Address+1 FFFF FFFF FFFF FFFF FFFF FFFF	1 1 1 1 1 1 1	0 1 1 1 1 1 1	1 1 1 1 1 1 1	0 1 1 1 1 1 1	Next Op Code Restart Address (LSB) Restart Address (LSB) Restart Address (LSB) Restart Address (LSB) Restart Address (LSB) Restart Address (LSB) Restart Address (LSB)

(continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
-----------------------------	--------	---------	-------------	-----	-----------------	-----------------	------------------	----------

**IMPLIED**

WAI	9	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer-1	0	1	0	1	Return Address (MSB)
		5	Stack Pointer-2	0	1	0	1	Index Register (LSB)
		6	Stack Pointer-3	0	1	0	1	Index Register (MSB)
		7	Stack Pointer-4	0	1	0	1	Accumulator A
		8	Stack Pointer-5	0	1	0	1	Accumulator B
		9	Stack Pointer-6	0	1	0	1	Conditional Code Register
RTI	10	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer+1	1	0	1	1	Conditional Code Register
		4	Stack Pointer+2	1	0	1	1	Accumulator A
		5	Stack Pointer+3	1	0	1	1	Accumulator B
		6	Stack Pointer+4	1	0	1	1	Index Register (MSB)
		7	Stack Pointer+5	1	0	1	1	Index Register (LSB)
		8	Stack Pointer+6	1	0	1	1	Return Address (MSB)
		9	Stack Pointer+7	1	0	1	1	Return Address (LSB)
		10	Return Address	1	0	1	0	First Op Code of Return Routine
SWI	12	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer-1	0	1	0	1	Return Address (MSB)
		5	Stack Pointer-2	0	1	0	1	Index Register (LSB)
		6	Stack Pointer-3	0	1	0	1	Index Register (MSB)
		7	Stack Pointer-4	0	1	0	1	Accumulator A
		8	Stack Pointer-5	0	1	0	1	Accumulator B
		9	Stack Pointer-6	0	1	0	1	Conditional Code Register
		10	Vector Address FFFA	1	0	1	1	Address of SWI Routine (MSB)
		11	Vector Address FFFB	1	0	1	1	Address of SWI Routine (LSB)
		12	Address of SWI Routine	1	0	1	0	First Op Code of SWI Routine
SLP	4	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		↑						
		Sleep						
		↓						
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	Op Code Address+1	1	0	1	0	Next Op Code

(continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/ $\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
-----------------------------	--------	---------	-------------	-------------------	-----------------	-----------------	------------------	----------

**RELATIVE**

BCC BCS	3	1	Op Code Address+1	1	0	1	1	Branch Offset	
BEQ BGE		2	FFFF	1	1	1	1	Restart Address (LSB)	
BGT BHI		3	3	{ Branch Address...Test="1" Op Code Address+2...Test="0"	1	0	1	0	First Op Code of Branch Routine Next Op Code
BLE BLS									
BLT BMT		5	1	Op Code Address+1	1	0	1	1	Offset
BNE BPL			2	FFFF	1	1	1	1	Restart Address (LSB)
BRA BRN			3	Stack Pointer	0	1	0	1	Return Address (LSB)
BVC BVS			4	Stack Pointer-1	0	1	0	1	Return Address (MSB)
BSR	5		Branch Address	1	0	1	0	First Op Code of Subroutine	

# Appendix III. Questions and Answers

This appendix contains some frequently asked questions about the HD6301Y0 and HD6303Y.

## III.1 Parallel Ports

### III.1.1 DDR and Data Register

**Question:** Which should be set first, the data register or DDR (data direction register), when an I/O port functions as an output port?

**Answer:** Output data should be stored in the data register first, then DDR should be set (DDR = 1). If DDR is set first, unknown data will be output from the port.

**Supplement:** DDR (data direction register)

DDR programs I/O port as an input or output.

DDR = 1: output

DDR = 0: input

DDR is initialized to 0 during reset.

### III.1.2 Port 7 Upper Bits

**Question:** What is the state of the upper 3 bits in port 7 (5-bit output port) when reading port 7 in mode 3 (single chip mode)?

**Answer:** The upper 3 bits in port 7 are all set to 1. The contents of the port 7 data register can be read, therefore the bit manipulation instructions can be used.

**Supplement:** Ports 1 and 4 can also be read with bit manipulation instructions.

### III.1.3 SCLK/P2<sub>2</sub> Pin

**Question:** How do you use the P2<sub>2</sub> (SCLK/P2<sub>2</sub> multiplexed pin) as an I/O port?

**Answer:** To use the P2<sub>2</sub> as an I/O port, set bit 1 in the port 2 DDR (data direction register), and CC0, CC1, and CC2 in the RMCR (rate/mode control register) as in table III-1.

Table III-1. P2<sub>2</sub> I/O Settings

Bit	Setting
Bit 1 of port 2 DDR (Note1)	0 (Input port) 1 (Output port)
CC0 (Note 2)	1
CC1	0
CC2	0 or 1

Notes:

1. The port 2 DDR selects respectively the direction of P2<sub>0</sub>-P2<sub>7</sub>.
2. During reset, CC0, CC1 and CC2 are cleared to 0 and the P2<sub>2</sub> functions as SCLK pin.

**Supplement:** The CC0, CC1, and CC2 (clock control format select) program the SCI data format and the SCI clock direction.

The DDR (data direction register) programs the direction of the I/O port.

DDR = 0: Input

DDR = 1: Output

### III.1.4 P5<sub>3</sub>/ $\overline{\text{HALT}}$ Pin

**Question:** How do you use the P5<sub>3</sub> (P5<sub>3</sub>/ $\overline{\text{HALT}}$  multiplexed pin) as an input-only port in expanded mode (modes 1 and 2)?

**Answer:** In expanded mode, P5<sub>3</sub> functions as  $\overline{\text{HALT}}$  pin with HLTE bit = 1 during reset. To use P5<sub>3</sub> as an input port, hold it high until 0 is written in the HLTE bit after reset, inhibiting  $\overline{\text{HALT}}$  input.

### III.1.5 Port 4 in Mode 2

**Question:** Port 4 can be used as an upper address output in mode 2 (expanded mode). In this case, which bit can be used when not all 8 bits are necessary as an address and the remaining bits can be used as input ports?

**Answer:** Any bit can be used.

In mode 2, any bit can be used as an upper address output or an input port.

When the port 4 data direction register (DDR) is cleared by reset, port 4 becomes an input port; when "1" is set, port 4 becomes an address output.



### III.1.6 Port 4

**Question:** When reading port 4 (8 bit I/O port), used as upper address outputs and as input ports in mode 2 (expanded mode), what data is read out from the bits used as upper address outputs?

**Answer:** The upper address is read out; in this case, "0". When reading bits used as I/O ports, the port states are read.

### III.1.7 P5<sub>5</sub>/ $\overline{OS}$ pin and Port 6

**Question:** Please explain the timing of output strobe ( $\overline{OS}$ ) generation by writing into port 6 (8 bit I/O port) and the timing of data output.

**Answer:** See figure III-1

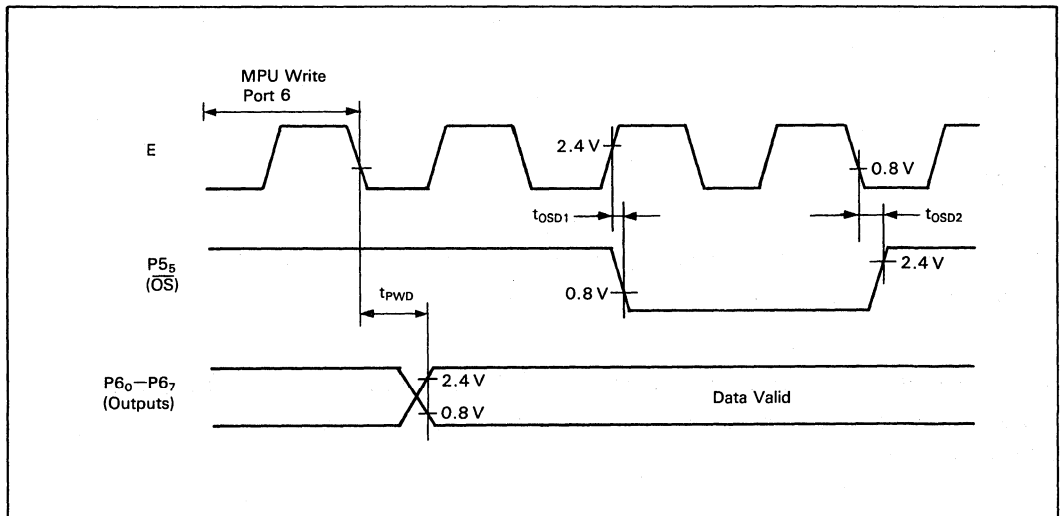


Figure III-1.  $\overline{OS}$  timing

### III.1.8 Port 2

**Question:** When setting port 2 (timer 1, timer 2 and SCI I/O pin/8 bit I/O port) as I/O port after having been used as a timer or SCI I/O, what is the I/O state of each bit?

**Answer:** The I/O state of each bit is the same as that when used as a timer or SCI I/O pin. When set as a timer, SCI I/O pin, the DDR of each bit is also set or cleared at the same time.

## III.2 Serial Port

### III.2.1 RDRF in Wake-Up Mode

**Question:** When using the SCI in the asynchronous mode with the receive enable bit (RE) of the transmit/receive control status register (TRCSR) = 1 and wake-up bit (WU) = 1, what is the state of the receive data register full bit (RDRF)? See figure III-2.

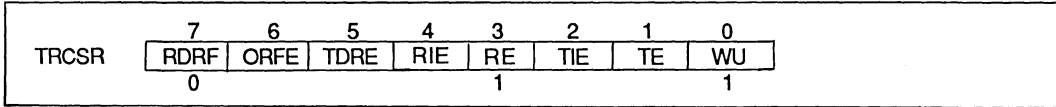


Figure III-2. Transmit/Receive Control Status Register in Wake-Up Mode

**Answer:** When the wake-up flag is set (WU = 1), the RDRF flag is not set (RDRF = 0).

### III.2.2 SCLK Direction and DDR

**Question:** When using the P2<sub>2</sub> (SCLK/bit 2 of I/O port 2) as the SCI clock I/O, is the clock direction determined by CC0, CC1, and CC2 (clock control/format select) in the RMCR (rate/mode control register) regardless of bit 2 of the port 2 DDR?

**Answer:** Yes, it is determined by CC0, CC1, and CC2 independently of the port 2 DDR. When used as an I/O port, its I/O direction is determined by bit 2 of the port 2 DDR. In this case, CC0, CC1, and CC2 should be set to a mode where P2<sub>2</sub> is not used as SCI clock (CC0, CC1, and CC2 set to 101, or 100). CC0, CC1, and CC2 are cleared to 0 at reset (table III-2).

Table III-2. P2<sub>2</sub> Direction

	P2 <sub>2</sub>	SCLK
Port 2 DDR	Input or output	No effect
CC0	1	CC0, CC1, CC2 determine
CC1	0	clock form, direction
CC2	0 or 1	

**Supplement:** The CC0, CC1, and CC2 (clock control format select) program the SCI data format and the SCI clock direction.

The DDR (data direction register) programs the direction of the I/O port.

DDR = 0: Input

DDR = 1: Output

### III.2.3 Receive Sampling Clock

**Question:** What is the relation between the receive data sampling clock at the SCI receive, and the data transfer rate?

**Answer:** The sampling clock is sixteen times as the transfer rate.

### III.2.4 Sampling Error

**Question:** What does "sampling error" mean?

**Answer:** "Sampling error" means receive margin in SCI operation. The HD6301Y0 detects a start bit at the negative edge of the sampling clock, and samples the start bit and data bit at the positive edge of the sampling clock.

The general equation of the receive margin is shown as follows (figure III.3).

$$M = \{(0.5 - 1/2N) - (D - 0.5)/N - (L - 0.5)F\} \times 100 (\%)$$

M: Receive margin

N: Baud rate ratio to sampling clock

D: Duty of the longer sampling clock of high and low (0.5 - 1)

L: Frame Length (7 - 12)

F: Absolute value of deviation of sampling clock frequency

An abbreviated version is:

$$M = (0.5 - 1/2N) \times 100 (\%) \quad (\text{Condition: } D = 0.5, F = 0)$$

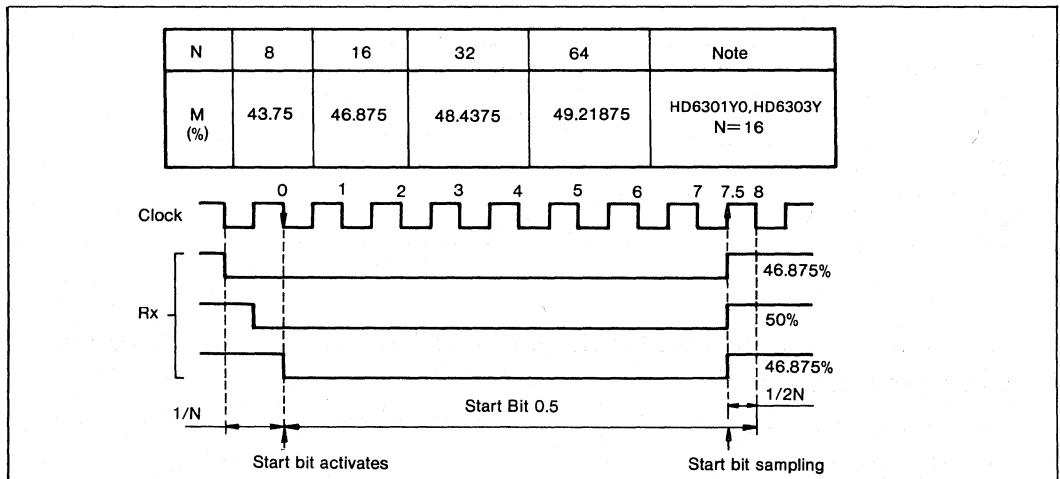


Figure III-3. Sampling Error



### III.2.5 SCI Receiving Operation in Asynchronous Mode

**Question:** When a framing error occurs while the serial communication interface (SCI) is receiving data in the asynchronous mode as shown in figure III-4, can the SCI receive the next data?

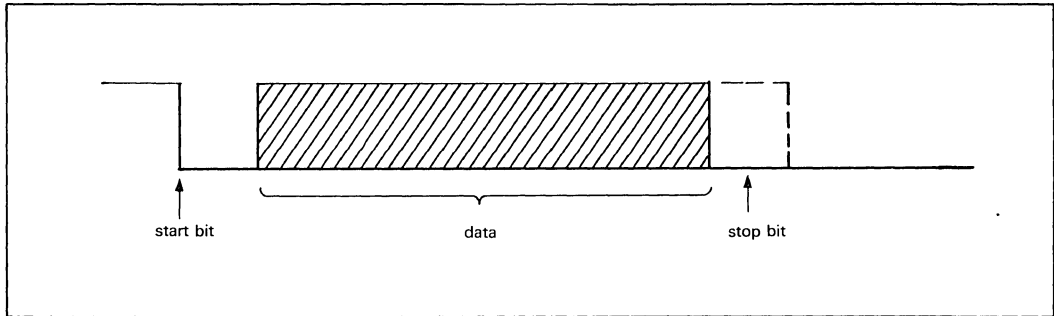


Figure III-4. Framing Error

**Answer:** Yes, it can.

As the start bit is detected by the level (Low), the next data can be received after a framing error without a falling edge. At the falling edge of the start bit, the sampling timing counter is cleared. So, if there is no falling edge, the next data is sampled in the former timing. Using this, a line break can be detected.

### III.2.6 RE, WU Set Timing in Asynchronous Mode

**Question:** In asynchronous mode, can a receive enable (RE) bit and a wake-up (WU) bit of the SCI be set at the same time?

**Answer:** No, they can't. Set RE first and then WU, or WU will not be set.

### III.2.7 Wake-Up

**Question:** Can WU of the SCI TRCSR be cleared by software? (=Can "0" be written into WU?)

**Answer:** Yes, it can.

Receive operation is activated from the cycle after executing an instruction writing "0".

### III.2.8 RDRF and Error Flags in Receive Operation

**Question:** Please explain the states of the SCI RDRF and error flags in receive operation in the asynchronous mode according to the following conditions.

- (1) normal operation
- (2) overrun error
- (3) parity error
- (4) framing error

**Answer:** See table III-3

Table III-3. Receive Operation Flags

	RDRF	ORFE	PER
Normal operation	1	0	0
Overrun error	1	1	0
Parity error	0	0	1
Framing error	0	1	0

### III.2.9 Each Flag State in Error Overlapping

**Question:** In SCI asynchronous mode, explain the states of each error flag and RDRF when errors overlap as follows;

- (1) an overrun error overlaps a parity error
- (2) an overrun error overlaps a framing error
- (3) an overrun error overlaps a parity error and a framing error
- (4) a parity error overlaps a framing error

**Answer:** See table III-4

Table III-4. Error Overlapping Flags

	RDRF	ORFE	PER
(1)	1	1	0
(2)	1	1	0
(3)	1	1	0
(4)	0	1	1

In cases (1) through (3), the error is checked as an overrun error.

In the case of (4), both ORFE and PER are set; RDRF not. In this case, the data causing both parity and framing errors can be read out.

### III.2.10 Checking Stop Bit

**Question:** When setting the stop bit length to 2 in the SCI transfer format in asynchronous mode, is the framing error checked by

- (1) first stop bit or
- (2) second stop bit or
- (3) both first and second bit?

**Answer:** The framing error is checked by both bits.

### III.2.11 Overrun Error

**Question:** When an overrun error occurs during the SCI receiving operation in asynchronous mode, is the data-causing error transferred to the receive data register (RDR) and can the CPU read it in the following cases?

- (1) After an overrun error, when the next data is not sent to the receive data shift register.
- (2) After an overrun error, when the next data is sent to the receive data shift register.

**Answer:** When  $RDRF = 1$  and  $ORFE = 1$ , error causing data is not sent to the RDR in both cases. That is, data received except with  $RDRF = 0$  is not transferred to the RDR. So, the CPU cannot read it.

### III.3 Timer/Counter

#### III.3.1 Reading the FRC

**Question:** When you read the free-running counter (FRC) of the timer 1 by a double-byte load instruction, is the read value correct?

**Answer:** It is correct. In the first cycle, the high byte of the FRC is read, when the low byte is set in a temporary register. At the next cycle, the data stored in the temporary register is read (figure III-5).

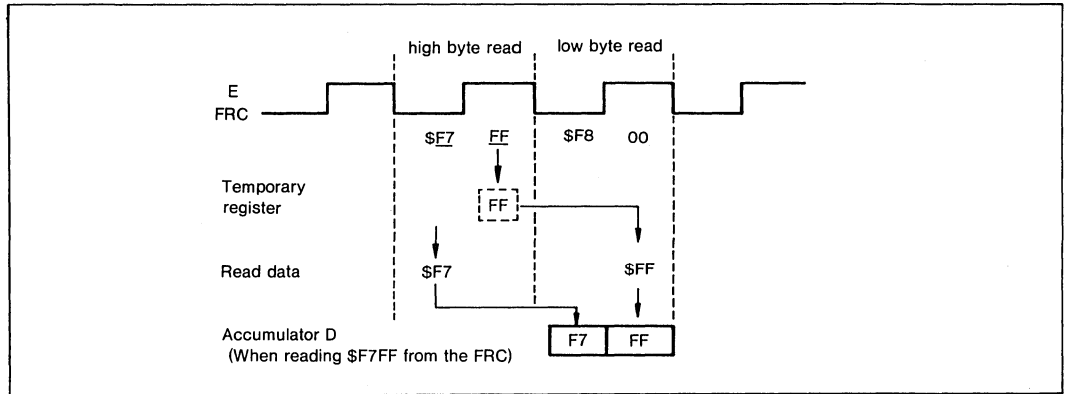


Figure III-5. FRC Double-Byte Read

**Supplement:** To read the timer FRC correctly, use double-byte load instructions (LDD, LDX).

#### III.3.2 Reading the FRC in the HD6801

**Question:** How is FRC writing in the HD6301Y0, HD6303Y, and HD63701Y0, different from the HD6801?

**Answer:** The difference is shown in table III-5.

Table III-5. HD6301Y0/HD6303Y and HD6801 Write Differences

Type	How to Write (Preset)
HD6801	The FRC is always preset to \$FFF8.
HD6301Y0, HD6303Y	Writing to the high byte presets the FRC to \$FFF8. Data is set in the FRC by a double-byte store instruction.

See figure III-6 for an example.

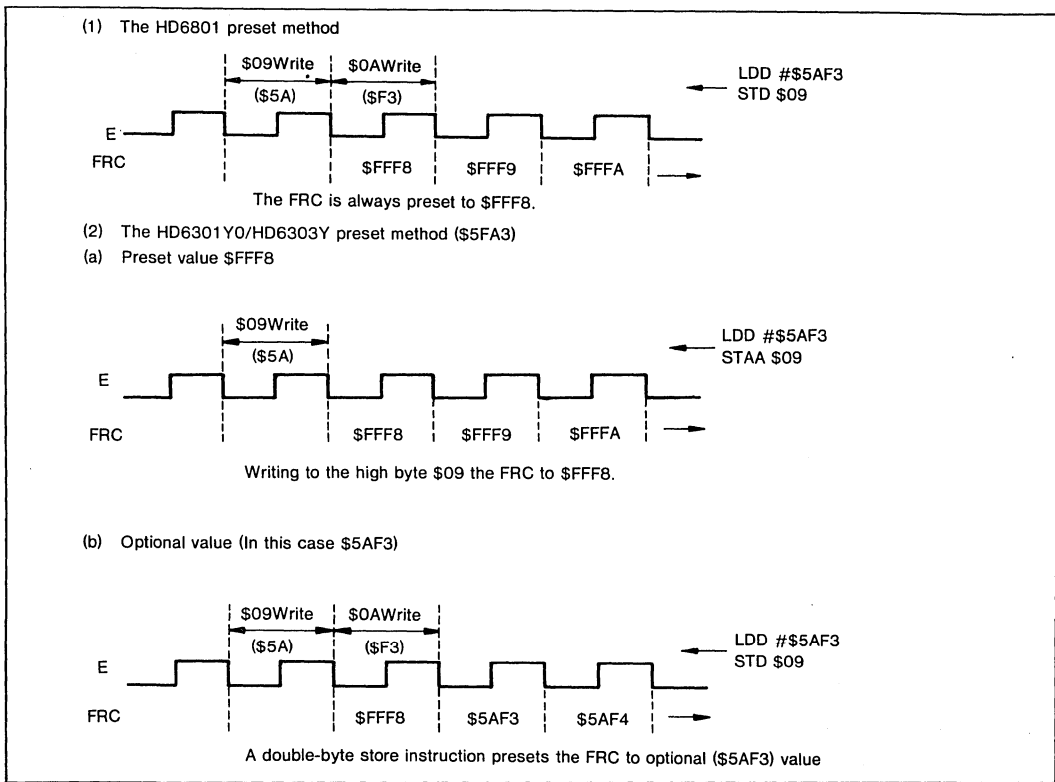


Figure III-6. FRC Writing for HD6301Y0/HD6303Y and HD6801

### III.3.3 ECMI Interrupt

**Question:** Timer 2 is used by writing 0 to enable counter match interrupt (ECMI) of the timer control/status register 3 (TCSR3). When a counter match flag (CMF) of TCSR3 becomes 1, 1 is written to ECMI. Does this generate an interrupt?

**Answer:** Yes. When the time constant register (TCONR) matches the timer 2 counter, the CMF is set to 1 and kept at 1 unless 0 is written in by software. An interrupt will occur if ECMI = 1 after CMF = 1.

**Supplement:** A timer 2 interrupt is generated with CMF = 1 and ECMI = 1.

ECMI defines internal interrupt (IRQ3) enable/disable.

ECMI = 0: disable

ECMI = 1: enable

### III.3.4 SCI and Writing to Timers

**Question:** When the SCI is operating, can data be written into the timer 1 FRC or timer 2 T2CNT?

**Answer:** If the SCI is operating by an external clock, the timer 1 FRC and the timer 2 T2CNT



can be written into. In the case of an internal clock, either the FRC or the T2CNT is used as a clock-source counter (note 1). No clock-source counter can be written to. Note that there are some restrictions, as follows:

1. External clock operation
  - a. Timer 1 FRC can be written to
  - b. Timer 2 T2CNT can be written to
2. Internal clock operation
  - a. Using timer 1 FRC as an internal clock
    - Don't write to the timer 1 FRC during SCI operation.
    - Timer 2 T2CNT can be written to.
  - b. Using timer 2 T2CNT as an internal clock
    - The timer 1 FRC can be written to, except when input clock to T2CNT is E/8 or E/128. E/8, E/128 come from the timer 1 FRC. If these clocks are selected as T2CNT input clocks, writing to the FRC will delay them.
    - Don't write to timer 2 T2CNT during SCI operation.

**Supplement:** When an internal clock is operating the SCI, writing to the clock-source counter will delay the SCI transfer rate.

### III.3.5 Timing for Timer 2 Output and CMF

**Question:** When counting events using timer 2, a counter match occurs. How does timer 2 output (port 2 bit 6) change? And also, when is the counter match flag (CMF) set?

**Answer:** See figure III-7

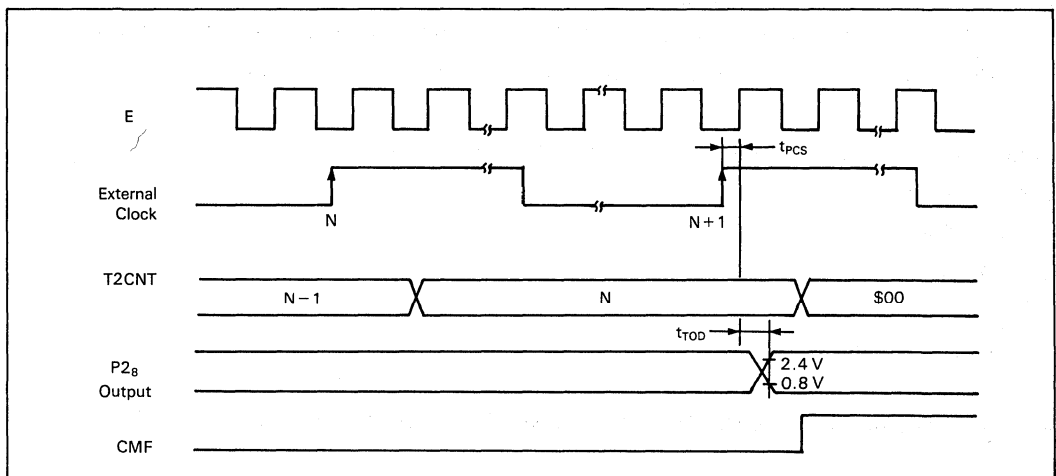


Figure III-7. Timing for Timer 2 Output and CMF



## III.4 Bus Interface

### III.4.1 E and Memory Ready

**Question:** What is the internal E clock state when the CPU uses the memory ready function?

**Answer:** Internal E clock operates at normal frequency (figure III-8). Since the timer count and the SCI transfer rate are set by the internal E clock, they are not also affected by the memory ready function.

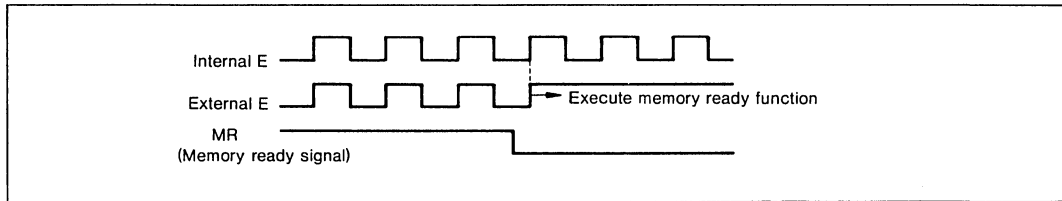


Figure III-8. Internal and External E Clocks

**Supplement:** It is impossible to examine the internal E clock from an external pin when using the memory ready function.

### III.4.2 Memory Ready and Halt After Reset

**Question:** After reset, are memory ready and halt functions enabled or disabled?

**Answer:** Both are enabled. MR and  $\overline{\text{HALT}}$  in three operating modes is shown in table III-6.

Table III-6. Operating Modes

Operating Mode	Memory Ready	Halt
Expanded mode	1 Enabled (note)	Enabled
	2 Enabled (note)	Enabled
Single-chip mode	No memory ready function	No halt function

Note: Invalid when accessing internal address space

**Supplement:** In the expanded mode (modes 1, 2), the memory ready enable bit (MRE) and halt enable bit (HLTE) of the RAM/port 5 control register are set to 1 during reset, enabling memory ready and halt functions.

### III.4.3 Buses at Internal Address Access

**Question:** When you access internal memory space, what states are the address buses, data buses, and control lines in?

**Answer:** Address buses and control lines ( $\overline{RD}$ ,  $\overline{WR}$ ,  $R/\overline{W}$ ) are always output regardless of internal or external address space accessing. During writes to the internal address space, the same data is output from the data bus. During reads, the data buses become high impedance.

### III.4.4 External Access to Register Addresses

**Question:** When using external memory at the addresses shown below in expanded modes (modes 1, 2), some addresses overlap internal registers and RAM addresses (figure III-9). In such a case, are there any problems?

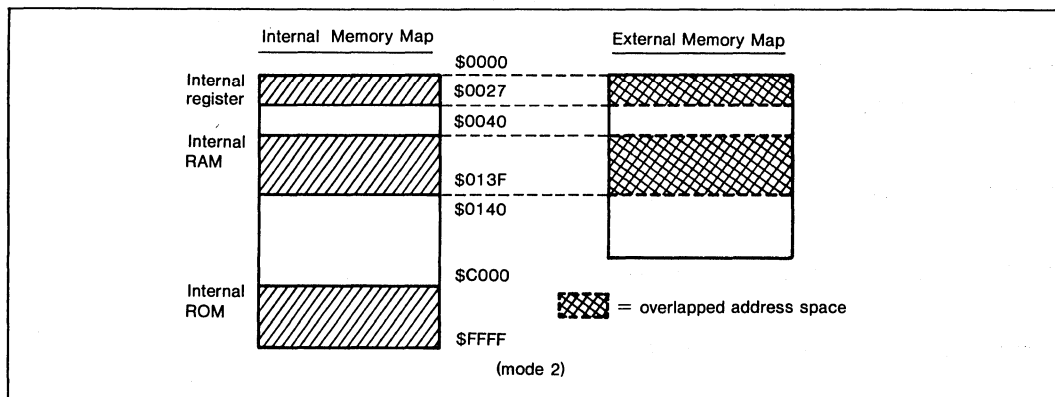


Figure III-9. Overlapping Addresses

**Answer:** There are no problems, but the overlapped addresses in the external memory space should not be used. When writing to the overlapping addresses, the same data is written into the internal and external address space. When reading, data is read from the internal, and the external address data is ignored.

**Supplement:** If the RAM enable bit (RAME) of the RAM/port 5 control register is 0, a read/write from/to the internal RAM space is invalid, and both operations are executed to the overlapped external address space.

### III.4.5 Buses During WAI

**Question:** What states are address buses, data buses, and control lines in after WAI instruction execution?

**Answer:** They are as in table III-7.

Table III-7. WAI State

Line	State
Address bus	FFFF (High)
Data bus	High impedance
$\overline{R\overline{W}}$	High
$\overline{RD}$	High
$\overline{WR}$	High

**III.4.6 Timing for Memory Ready and E Clock**

**Question:** What do  $t_{HMR}$  (memory ready hold time) and  $t_{SMR}$  (memory ready set up time) mean in the timing for “memory ready” and E clock? See figure III-10.

**Answer:**

$t_{HMR}$ : When MR becomes low within  $t_{HMR}$  from the E clock rising edge, the E clock is extended (max setting).

$t_{SMR}$ : When MR becomes high within  $t_{SMR}$  before the E clock falling edge (point A), E clock becomes low in the cycle (minimum setting)

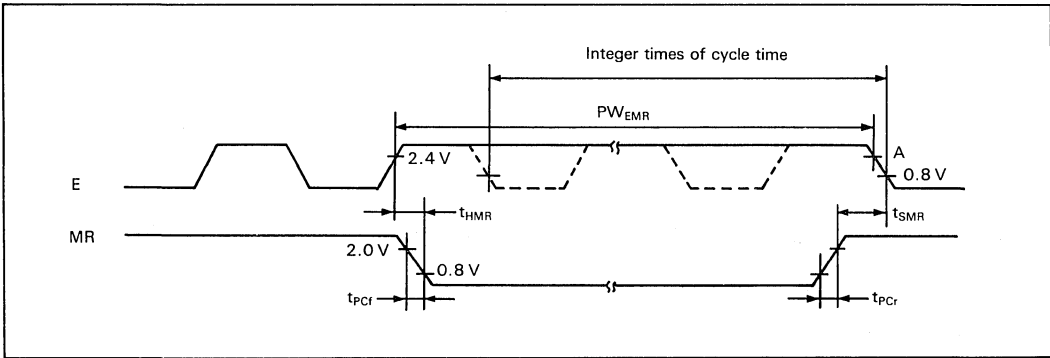


Figure III-10 Timing for Memory Ready and E Clock

### III.4.7 Limit of Halt Time

**Question:** Is the halt time limited?

**Answer:** No. If the halt pin has been low before a restart, the halt functions after a reset vector has been output and after the first instruction has been fetched.

**Supplement:** When the halt signal is set to low, the CPU stops after an instruction being executed finishes, and goes to the halt state. The halted CPU sets the bus available (BA) to high and the  $\overline{RD}$ ,  $\overline{WR}$  and  $R/\overline{W}$  to high impedance.

## III.5 Interrupt Control

### III.5.1 $\overline{IRQ1}$ During Standby

**Question:** When the CPU is returning from standby mode ( $\overline{RES} = \text{low}$ ,  $\overline{STBY} = \text{low}$ ) with  $\overline{IRQ1}$  low, can the interrupt be accepted if  $\overline{IRQ1}$  low continues after return?

**Answer:** It cannot. Interrupts can be accepted when  $IRQ1E = 1$  and  $I = 0$ . After the CPU returns from standby, it has  $IRQ1E = 0$  and  $I = 1$ . To accept the interrupt, the software should make  $IRQ1E = 1$ ,  $I = 0$  after resetting.

**Supplement:**  $IRQ1E$  is the  $\overline{IRQ1}$  interrupt enable bit of the RAM/port 5 control register. When  $IRQ1E = 1$ ,  $P5_0$  can be used as an interrupt pin.  $I$  is the interrupt mask bit. When  $I = 0$ , the CPU accepts interrupts.

### III.5.2 Trap Interrupt

**Question:** How does the trap interrupt differ from other interrupts ( $\overline{NMI}$ ,  $\overline{IRQ1}$ ,  $\overline{IRQ2}$  and  $IRQ3$ )?

**Answer:** The differences are:

- Return address (figure III-11)
- Interrupt sequence (figure III-12)

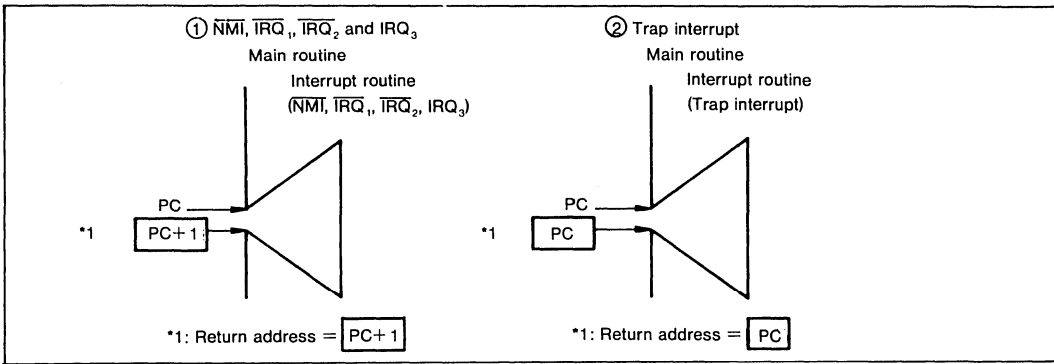


Figure III-11. Return Address

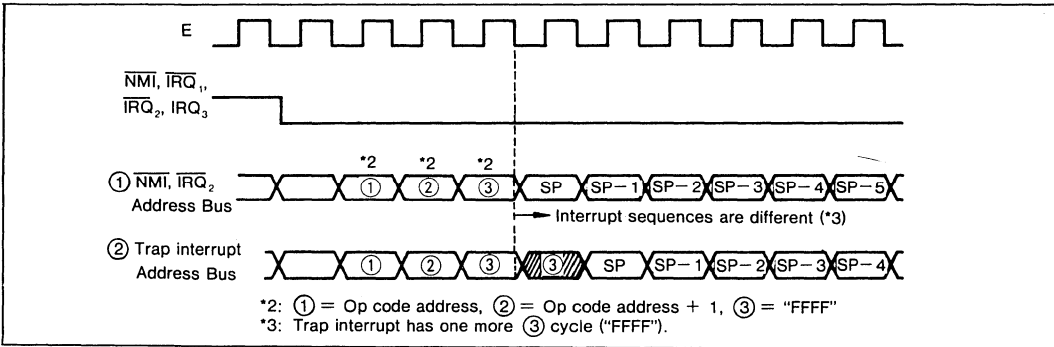


Figure III-12. Interrupt Sequence

### III.5.3 $\overline{\text{LIR}}$ During Interrupt

**Question:** What is the output state of load instruction register ( $\overline{\text{LIR}}$ ) in the interrupt sequence?

**Answer:** The output state of  $\overline{\text{LIR}}$  is low in the following cycles:

1. Prefetch cycle of the last instruction cycle opcode just before interrupt sequence
2. Fetch cycle of the first opcode of the interrupt routine

The output state of  $\overline{\text{LIR}}$  in the interrupt sequence is shown below.

1. Last instruction execution cycle just before the interrupt sequence (figure III-13).

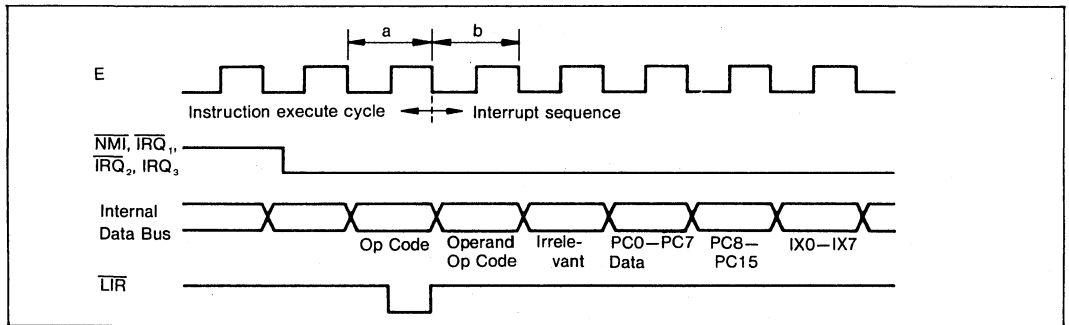


Figure III-13. Last Cycle Before Interrupt

- $\overline{\text{LIR}}$  output is low at the last instruction execution cycle just before interrupt sequence opcode prefetch.
- The first cycle of the interrupt sequence (b in figure III-13) is a dummy fetch cycle. In this cycle, there are two cases; an operand is on the data bus, or an opcode is on the bus. In both cases,  $\overline{\text{LIR}}$  output is not low.

2. First opcode fetch cycle in the interrupt routine (figure III-14).

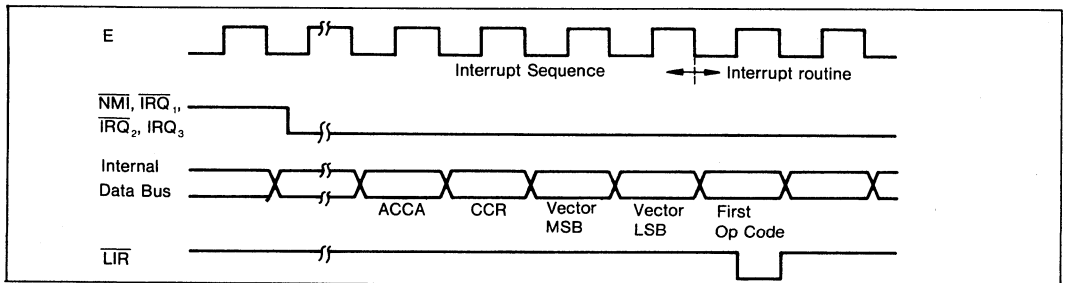


Figure III-14. First Cycle in Interrupt

$\overline{\text{LIR}}$  output is low when the first opcode of the interrupt routine is fetched.



**Supplement:** Load instruction register ( $\overline{\text{LIR}}$ ) low shows that instruction opcode is on the data bus.

### III.5.4 Accepting an IS Interrupt

**Question:** Is an input strobe (IS) interrupt accepted during the execution of the  $\overline{\text{IRQ}}_1$  interrupt routine?

**Answer:** Yes. When an IS interrupt is generated during the execution of the  $\overline{\text{IRQ}}_1$  interrupt routine, with the input strobe enable (IS  $\text{IRQ}_1$  ENABLE) being set, and the IS flag is set;

1. It is accepted if the interrupt mask bit (I) of condition code register (CCR) has been cleared.

However, in this case, the interrupt factor of the  $\overline{\text{IRQ}}_1$  must have been cleared before clearing the I bit, that is, by setting the  $\overline{\text{IRQ}}_1$  pin low and clearing  $\text{IRQ}_1\text{E}$ .

2. If the I bit of the CCR is set, it is accepted after the  $\overline{\text{IRQ}}_1$  interrupt routine finishes.

**Supplement:** Since the  $\overline{\text{IRQ}}_1$  and IS share an interrupt vector, levels of the input strobe flag (IS FLAG) and  $\overline{\text{IRQ}}_1$  pin are checked to determine which interrupt is generated, by reading P50 (bit 0 of port 5).

## III.6 Oscillation Circuit

### III.6.1 E Clock Triggering

**Question:** With which edge of the EXTAL clock does the E clock change, the rising or falling edge?

**Answer:** It changes synchronously with the falling edge (figure III-15).

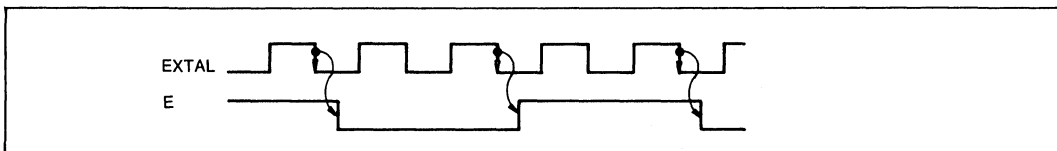


Figure III-15. E Clock Timing

## III.7 Reset

### III.7.1 Ports at Reset

**Question:** What is the state of each port at reset?

**Answer:** It is as shown in table III-8.



Table III-8. Port State at Reset

Port	Mode	Reset
1 (A <sub>0</sub> -A <sub>7</sub> )	1, 2	High
	3	High impedance
2	1, 2	High impedance
	3	High impedance
3 (D <sub>0</sub> -D <sub>7</sub> )	1, 2	High impedance
	3	High impedance
4 (A <sub>8</sub> -A <sub>15</sub> )	1	High
	2, 3	High impedance
5	1, 2	High impedance
	3	High impedance
6	1, 2	High impedance
	3	High impedance
7	1, 2	Note 1
	3	High impedance

Note:

1.  $\overline{RD}$ ,  $\overline{WR}$ ,  $R/\overline{W}$ ,  $\overline{LIR}$  = high; BA = low

**Supplement:** E clock at reset is output at normal frequency after oscillation stabilization time.

### III.7.2 I/O Port Output After Reset

**Question:** What data does an I/O port output when the data direction register (DDR) = 1 after reset?

**Answer:** After reset, undefined data is output from the I/O port, since the data register of an I/O port is undefined. For the output state, put data in the data register before setting the DDR = 1.

### III.7.3 $\overline{\text{RES}}$ Schmitt Trigger

**Question:** Is a Schmitt trigger circuit provided with  $\overline{\text{RES}}$ ?

**Answer:** Yes (figure III-16).

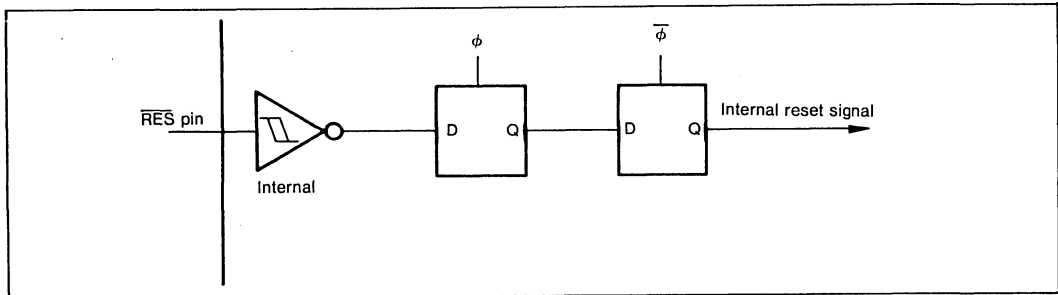


Figure III-16. Reset Circuit

### III.7.4 Reset Circuit Capacitance

**Question:** Does  $C_r$  in the reset circuit shown in figure III-17 ( $R_r \times C_r > 20 \text{ ms}$ ), have an upper limit?

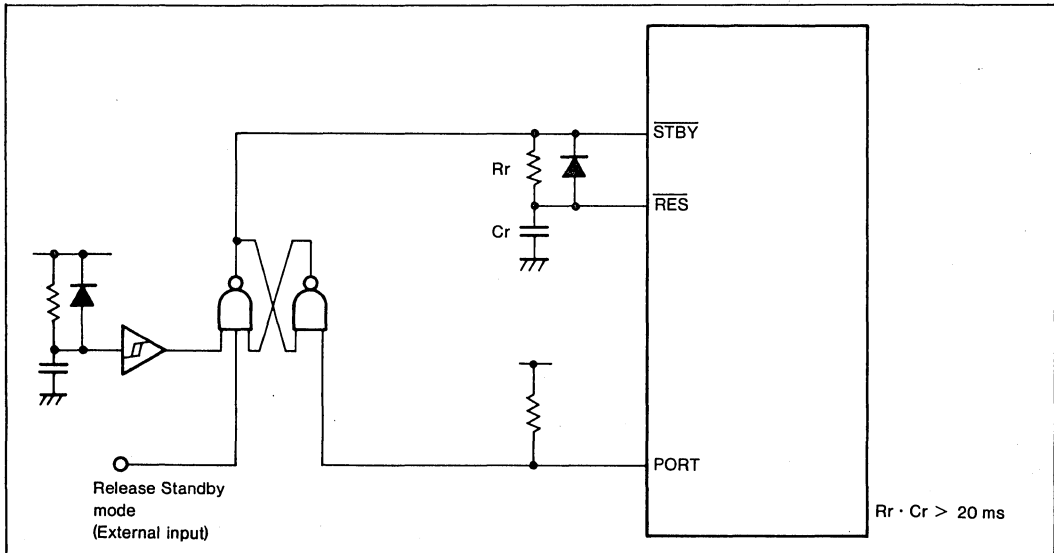


Figure III-17. Reset Input Circuit

**Answer:** No, because  $\overline{\text{RES}}$  is provided with a Schmitt trigger circuit (figure III-16).

### III.7.5 State of I/O Ports during the 20 ms after a Power-on Reset

**Question:** What state are I/O ports in for the 20 ms after a power-on reset during which time the oscillation is unstable?

**Answer:** The I/O ports are in the reset state immediately after a power-on reset because it is directly controlled by the  $\overline{RES}$  pin. However, at this time, the contents of the data register of each port are undefined (figure III-18).

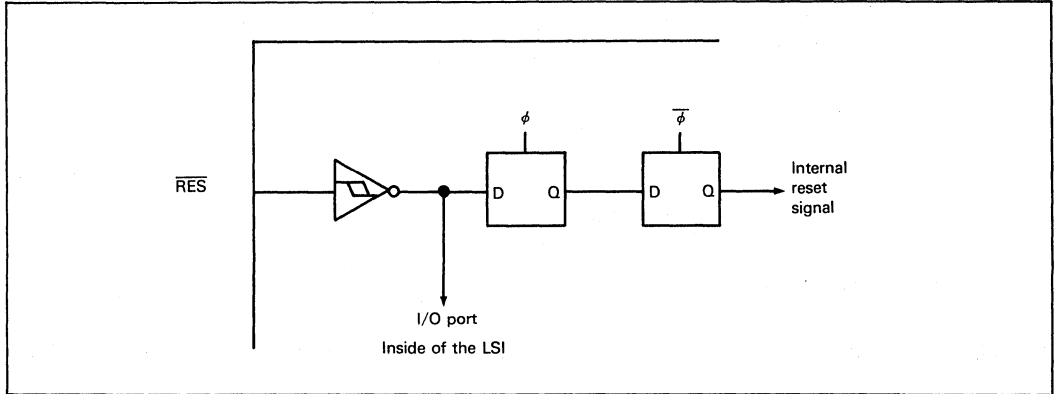


Figure III-18 Reset Circuit

### III.7.6 State of Port 4 after a Reset

**Question:** What is the state of port 4 (8 bit I/O port) of the HD6301Y0 after a reset?

**Answer:** Table III-9 shows the state of port 4 after a reset.

Table III-9. Port 4 After Reset

Mode	State of port 4	
Extended modes	Mode 1	Address bus high-order output (*1)
	Mode 2	Input port
Single chip mode	Mode 3	Input port

\*1: In mode 1, the data direction register (DDR) is forcibly set and port 4 outputs high-order addresses.

### III.7.7 State of Address Bus if Reset during Operation

**Question:** If reset occurs during operation, when does the address bus become \$ FFFF?

**Answer:** Timing of  $\overline{\text{RES}}$  and the address bus are as follows (figure III-19).

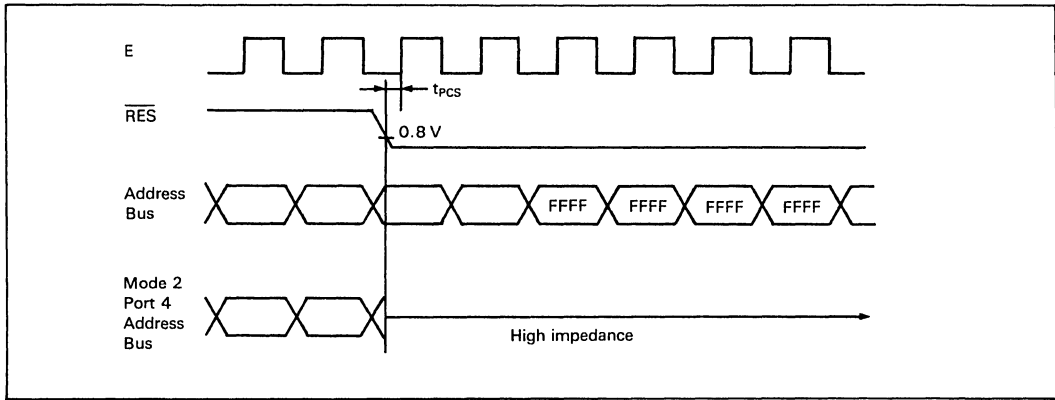


Figure III-19. Timing of  $\overline{\text{RES}}$  and the address bus

## III.8 Low Power Dissipation Mode

### III.8.1 Standby During Instruction Execution

**Question:** Does the CPU wait until the current instruction is executed to enter the standby mode?

**Answer:** No. The CPU enters standby mode regardless of the current instruction; the CPU goes into reset condition and the oscillator stops with  $\overline{\text{STBY}}$  low (figure III-20).

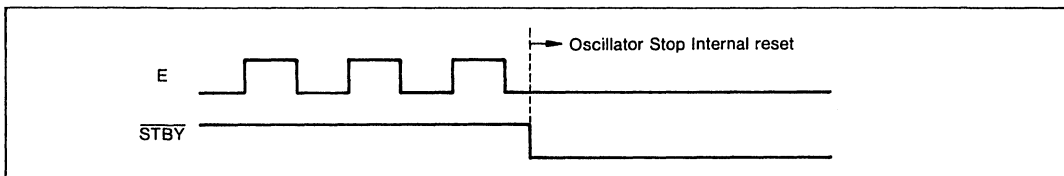


Figure III-20. E During Standby

### III.8.2 Standby Timing

**Question:** The timing for the standby mode is shown in figure III-21 (see also figure 3-5). Is T1 in the figure defined?

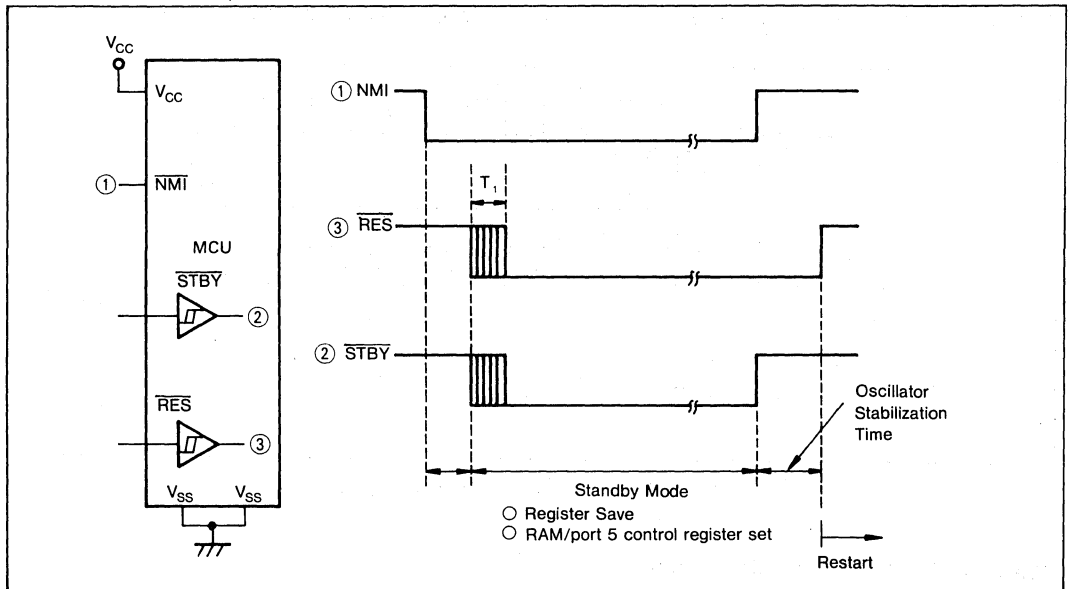


Figure III-21. Standby Mode Timing

**Answer:** It is not, but if the time for nonmaskable interrupt ( $\overline{\text{NMI}}$ ) is guaranteed, either  $\overline{\text{RES}}$  or  $\overline{\text{STBY}}$  can go low with no priority.

**Supplement:** The CPU goes to the standby mode independently of instruction execution sequence. Use the  $\overline{\text{NMI}}$  routine before entering standby mode.

### III.8.3 Ports at Standby

**Question:** What is the state of each I/O port during standby?

**Answer:** Each I/O port and the E pin during standby are high impedance.

### III.8.4 Return from Standby Without Reset

**Question:** What occurs when the CPU returns from the standby mode without using reset start?

**Answer:** The CPU does not operate normally because the contents of each register are not defined. Therefore, always use the reset start when returning from the standby mode.

### III.8.5 Sleep and Standby Internal States

**Question:** What are the internal states in the sleep or standby mode?

**Answer:** They are as shown in table III-10.

Table III-10. Sleep and Standby Mode States

	Sleep Mode	Standby Mode
Oscillation circuit	Continues	Stops
CPU (register)	Stops (retained)	Stops (undefined)
RAM	Retained	Retained
I/O	Retained	High impedance
Timer	Continues	Stops
Serial communications	Continues	Stops
Internal registers	Retained	Reset
Cancel	Interrupt STBY = low Reset start	Reset start after $\overline{\text{STBY}} = \text{high}$ (at hardware standby) Reset start (at software standby)

**Supplement:** Internal states in the standby mode are the same as those in reset. Use the reset start when returning from the standby mode. In this case  $\overline{\text{RES}}$  should be kept low from  $\overline{\text{STBY}} = \text{high}$  during oscillation stabilization time (20 ms minimum).

### III.8.6 Sequence of Going to Standby Mode by Software

**Question:** How can the CPU go to the standby mode using software?

**Answer:** The CPU can go to the standby mode using software by clearing the standby flag (STBY FLAG) of the RAM/port 5 control register. In this case, before going to the standby mode, the standby power bit (STBY PWR) of the RAM/port 5 control register must be set and the RAM enable bit (RAME) should be cleared. Below is shown an example of the method of going to the standby mode by software.

```

•
•
•
OIM #80, $14 - - - (Setting STBY PWR)
AIM #9F, $14 - - - (Clearing RAME, STBY FLAG)
•

```

### III.8.7 Timing of Going to the Standby Mode by Software

**Question:** In the case that the CPU goes to the standby mode by clearing the standby flag (STBY FLAG) of the RAM/port 5 control register, how many cycles after clearing does the oscillator stop and does the CPU go into the standby mode?

**Answer:** The oscillator stops and the CPU goes to the standby mode at E clock's low level of the first cycle after the STBY FLAG is cleared (figure III-22).

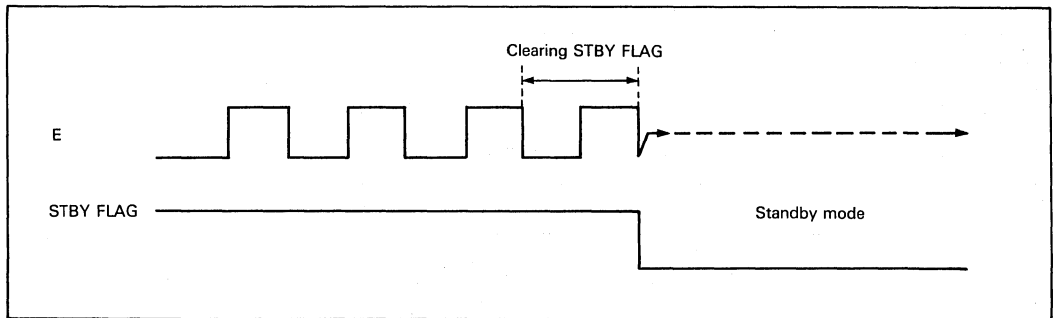


Figure III-22 Timing of Standby Mode by Software

### III.8.8 Writing to STBY PWR

**Question:** Is it possible to write "0" into the standby power bit (STBY PWR) of the RAM/port 5 control register?

**Answer:** Yes. The STBY PWR can be used as a normal read/write flag.

## III.9 Software

### III.9.1 Bit Manipulation Instructions

**Question:** How should the bit manipulation instructions of the HD6301Y0, HD6303Y, and HD63701Y0, be written?

**Answer:** They are written as shown in figure III-23.

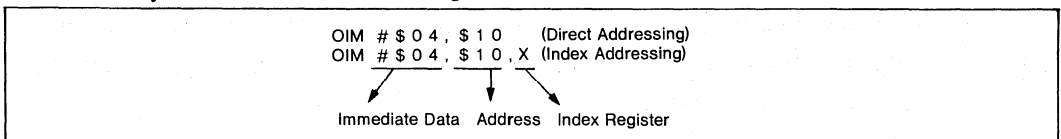


Figure III-23 OIM Example

This is an example of an OR operation between the immediate data and the memory which stores the result in the memory. The AIM, EIM, and TIM instructions are written in the same way.

The bit manipulations in table III-11 have different mnemonics with the same opcode.

Table III-11. Shared Opcodes

Bit Manipulation Instruction	Instructions Having the Same Opcode	
	Mnemonic	Function
AIM	BCLR	0 AND Mi The memory bit i (i = 0 to 7) is cleared and the other bits don't change
OIM	BSET	1 OR Mi The memory bit i (i = 0 to 7) is set and the other bits don't change
EIM	BTGL	Mi EOR Mi The memory bit i (i = 0 to 7) is inverted and the other bits don't change
TIM	BTST	1 AND Mi AND operation test of the memory bit i (i = 0 to 7) and 1 is executed and its corresponding condition code is changed.

The mnemonics mentioned above can be written as in figure III-24.

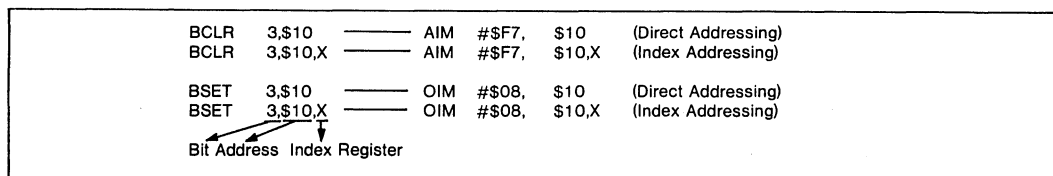


Figure III-24. Shared Opcode Instruction Format

## III.10 Others

### III.10.1 RAME Disabled

**Question:** When executing a program with the RAM enable bit (RAME) of the RAM/port 5 control register disabled (RAME = 0),

1. What occurs if the internal RAM address is accessed?
2. What occurs if interrupt requests are generated?

**Answer:**

1. The internal RAM cannot be accessed. It is neither readable nor writable with RAME = 0, so in mode 1 or 2, the external memory is read/written into.
2. Interrupts are accepted, but the CPU will fail when returning from the interrupt with no



stacking area other than the internal RAM.

**Supplement:**

1. RAME = 0; internal RAM is invalid. In modes 1 or 2, data can be read from the external memory.
2. RAME = 1; internal RAM is enabled.

**III.10.2 RAME at Reset**

**Question:** Is the RAM enable bit (RAME) set on reset at  $\overline{\text{RES}}$  low or the rising edge of  $\overline{\text{RES}}$ ?

**Answer:** It is set at the rising edge of  $\overline{\text{RES}}$  (figure III-25).

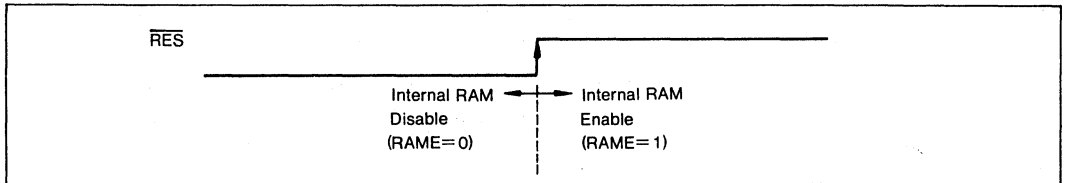


Figure III-25. RAME at Reset

**Supplement:** RAME is set/cleared by the software.

1. RAME = 0; Internal RAM is invalid. In mode 1 or 2, data can be read from the external memory.
2. RAME = 1; Internal RAM is enabled.

# Appendix IV: The Differences Between HD63701Y0 and HD6301Y0

Item	HD63701Y0	HD6301Y0									
Input low voltage of $\overline{\text{RES}}$ , $\text{MP}_0$ , $\text{MP}_1$	$V_{\text{IL}} = 0.6 \text{ V max}$	$V_{\text{IL}} = 0.8 \text{ V max}$									
$I_{\text{in}}$ and $C_{\text{in}}$ of $\overline{\text{RES}}$	$I_{\text{in}} = 10 \mu\text{A max}$ $C_{\text{in}} = 65 \text{ pF max}$ $I_{\text{in}}$ and $C_{\text{in}}$ are larger than HD6301Y0 because $\overline{\text{RES}}$ is also used as $V_{\text{PP}}$ .	$I_{\text{in}} = 1.0 \mu\text{A max}$ $C_{\text{in}} = 12.5 \text{ pF max}$									
Crystal oscillator characteristics	Internal resistance of crystal oscillator $R_{\text{S}}$	Internal resistance of crystal oscillator $R_{\text{S}}$									
	<table border="1"> <tr> <td>Frequency (MHz)</td> <td>2.5</td> <td>4.0</td> <td>6.0</td> <td>8.0</td> </tr> <tr> <td><math>R_{\text{S}} \text{ max } (\Omega)</math></td> <td>500</td> <td>120</td> <td>80</td> <td>60</td> </tr> </table>	Frequency (MHz)	2.5	4.0	6.0	8.0	$R_{\text{S}} \text{ max } (\Omega)$	500	120	80	60
Frequency (MHz)	2.5	4.0	6.0	8.0							
$R_{\text{S}} \text{ max } (\Omega)$	500	120	80	60							
Storage temperature	$T_{\text{stg}} = -55 \text{ to } 125 \text{ }^\circ\text{C}$	$T_{\text{stg}} = -55 \text{ to } 150 \text{ }^\circ\text{C}$									
Caution	The HD63701Y0 differs from HD6301Y0 in chip design and manufacturing process. When applying the HD63701Y0 system to HD6301Y0, and HD6301Y0 system to HD63701Y0, note that characteristic values are not exactly the same even if guaranteed values are the same.										

# Appendix V: Program Development Procedure and Support System

## V.1 Overview

The cross assembler and the hardware emulator using various types of computer are prepared by the company as supporting systems to develop user's programs. User's programs are mask programmed into the ROM and delivered as the LSI by the company.

Figure V-1 shows the typical program design procedure and table V-1 shows the system development support tool for the HD6301Y0 which are used in these processes.

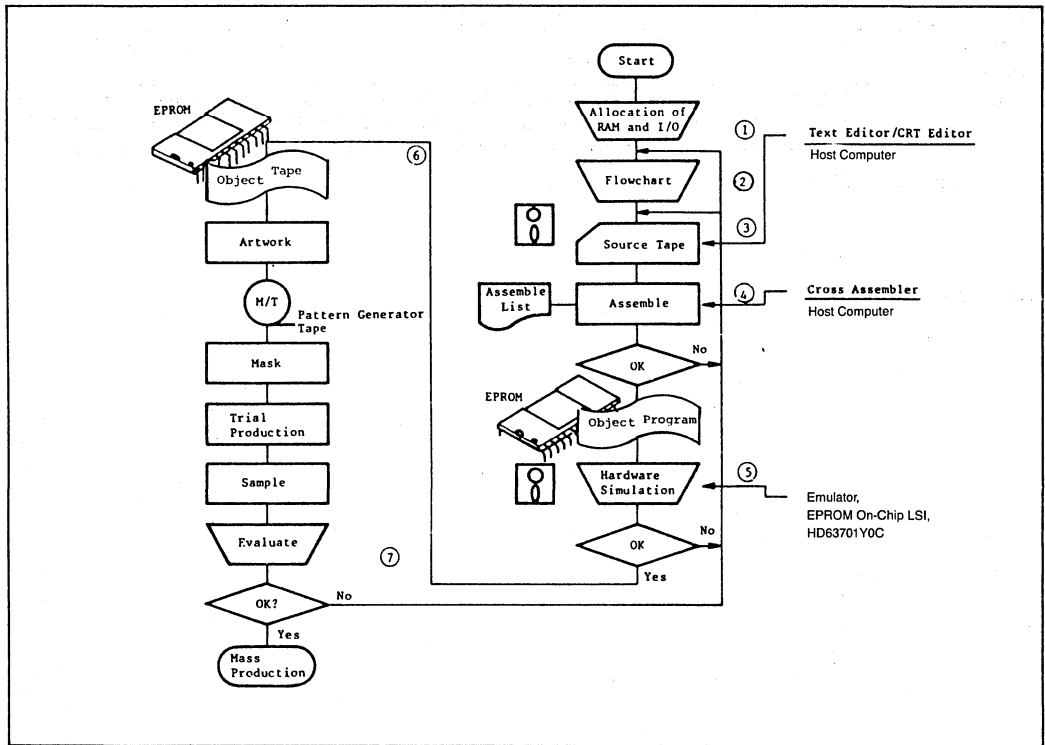


Figure V-1. Program Design Procedure

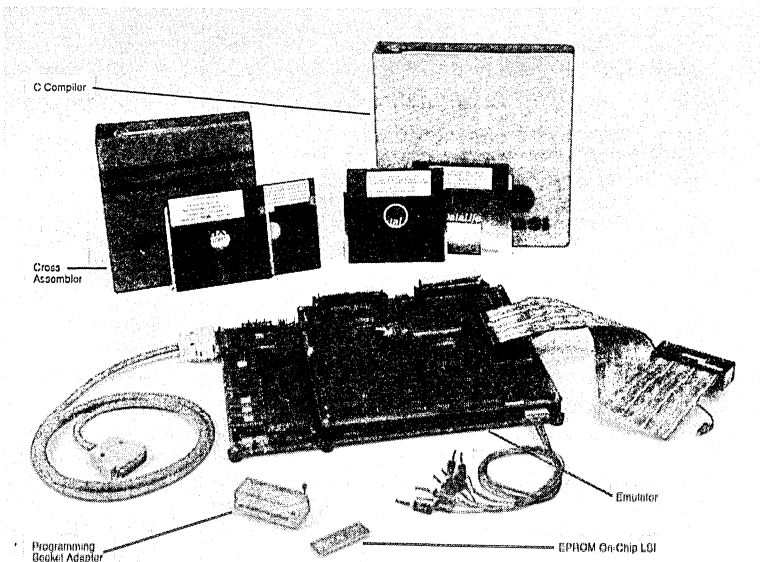
**(Explanation)**

1. When the user programs the system using the HD6301Y0 series, a functional assignment of each I/O pin and an allocation of RAM area should be specified adjusting to designed system before actual programming.
2. A flowchart is designed to implement the functions and it is coded by using the HD6301Y0 mnemonic code.
3. Write the software coded according to the flowchart on a floppy disk to make a source program.
4. Assemble the source program to generate an object program using a computer. Assembly errors are also detected.
5. Verify the program through hardware emulation with an emulator, H68SD5/5A, H680SD200 or EPROM on-chip type microcomputer.
6. Send the completed program to the company in the form of EPROM. Send Single-chip microcomputer order specification and Mask option list at that time.
7. ROM and mask option are masked by the company. LSI is testatively produced and the sample is handed in to the user. If a user doesn't see any problem in programming, mass production can be started.

Table V-1. Support Tools

Part No.	Emulator Set	EPROM On-Chip LSI	EPROM On-Chip LSI Programming Socket Adaptor	IBM PC* Cross Assembler	IBM PC C Compiler
HD6301Y0 HD6303Y	H31MIX3 (HS31YEML03H)	HD63701Y0C	HS31YESS11H	S31IBM PC	US31PCL1SF

Notes: IBM PC is a trademark of International Business Machine Corporation.



**HD6301Y0 and HD6303Y Development Tools**

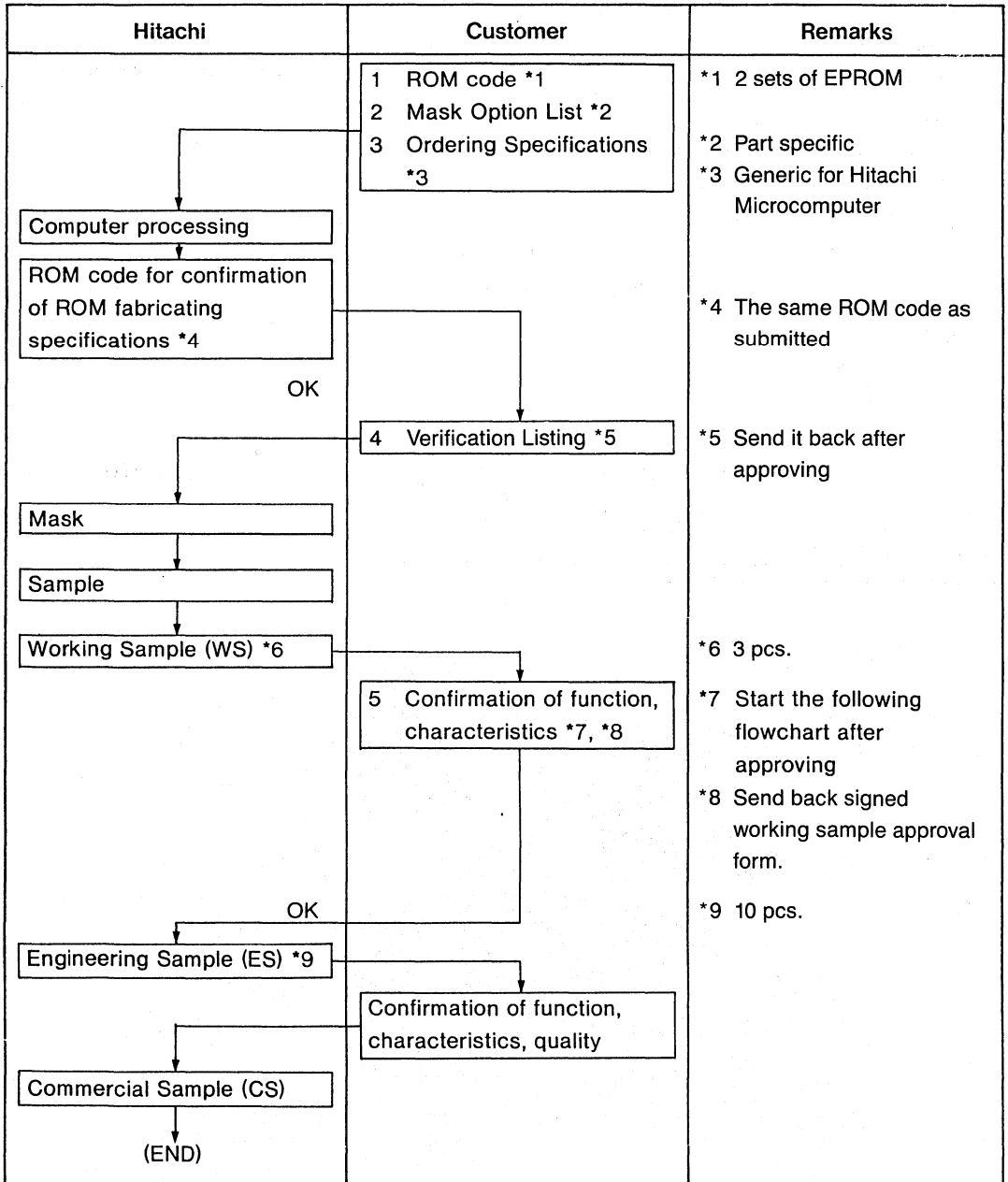


## V.2 Single Chip Microcomputer ROM Ordering Procedure

### V.2.1 Development Flowchart

Single chip microcomputer device is developed according to the following flowchart after program development.

Device Development Flowchart



Note: Please send in 1, 2, and 3 at ROM ordering, and send back 4, 5 after approving.



## V.2.2 Data you send and precautions

(a) Ordering specifications..... Common style for all Hitachi single chip microcomputer devices. Please enter as for the followings. The format is shown in the next page.

- Basic ITEM
- Environment Check List
- Check List of attached data
- Customer

(b) ROM code..... Please send in the ordering ROM code by 2 sets of EPROM the same contents are written. Enter ROM code No. in them. It is desirable to send in program list for easy confirmation of the program contents.

## V.2.3 Change of ROM code

Note that if you change the ROM code once sended in or other specification, the ROM must be developed from the beginning. The cost of mask charge should be provided again in this case.

## V.2.4 Samples and Mass production

(Working Sample) ..... Sample for confirmation of the contents of ROM code and that of mask option. Normally 3 samples are sent, but not guaranteed as for reliability. Please evaluate and approve immediately because the following sample making and mass production are set about after obtaining your evaluation.

(Engineering Sample) ..... Sample for evaluating also reliability. 10 pcs are included in mask charge.

(Commercial Sample) ..... Samples for pre-production which may be purchased separately.

(Mass Product) ..... Products for actual mass production. Please enter the plan of mass production in full.

**HD6301Y0**  
**ORDERING SPECIFICATIONS**

**(1) GENERAL CHARACTERISTICS** (Fill in blank space or check appropriate box .)

Customer		Package Outline (See page 523.)	<input type="checkbox"/> DP-64S	<input type="checkbox"/> CP-68
Device Type			<input type="checkbox"/> FP-64	
Application (be specific)		Options/Remarks:		
Customer ROM Code ID				
ZTAT™ Conversion	<input type="checkbox"/> Yes <input type="checkbox"/> No			
ROM Code Media	<input type="checkbox"/> EPROM™    Must Specify: Customer Programmed Start Address _____ <input type="checkbox"/> ZTAT™    Customer Programmed Stop Address _____			
Operating Temperature	<input type="checkbox"/> Standard <input type="checkbox"/> J (-40° C to +85° C) version if offered			
Remask	<input type="checkbox"/> Yes <input type="checkbox"/> No    Previous Hitachi P/N _____			

**(2) OPERATING CHARACTERISTICS** (Fill in blank space or check appropriate box .)

LSI Ambient Temperature	Typical		°C	Target Level Of Reliability	<input type="checkbox"/> 1000 Fit	<input type="checkbox"/> (_____)
	Range		°C- °C		<input type="checkbox"/> 500 Fit	
LSI Ambient Humidity	Typical		%	Acceptable Quality Level	Electrical	<input type="checkbox"/> 0.25% <input type="checkbox"/> (_____)
	Range		%- %		Major Visual	<input type="checkbox"/> 0.65% <input type="checkbox"/> (_____)
Power On Duration	Typical		Hours/Day	LSI Operating Speed (Specify MHz or KHz)		
Maximum Applied Voltage To LSI	Power Supply	Max.	V	Remarks:		
	I/O	Max.	V			

**(3) ELECTRICAL CHARACTERISTICS** (Fill in blank space or check appropriate box .)

<input type="checkbox"/> Purchasing Specifications	<input type="checkbox"/> Hitachi's Standard Specifications
_____	Refer To Data Sheet: _____

**For Hitachi Use Only**

**(4) CUSTOMER APPROVAL**

Customer Name \_\_\_\_\_

PO# \_\_\_\_\_

Approved By (print) \_\_\_\_\_

Approved By (signature) \_\_\_\_\_

Date \_\_\_\_\_

**(5) ROM CODE VERIFICATION**

LSI Type No.	
Shipping Date of ROM To Customer	
Approved Date of ROM From Customer	



# HD6301/HD6303 SERIES HANDBOOK

Section Seven

## Software Application Notes

7





## FOREWORD

The HD6301/HD6303 is a family of 8-bit single chip CMOS microcomputers controlled by microprogramming. This family aids high speed data process by adding bit operation instruction, logical operation instruction, lower power consumption mode instruction, and accumulator and index register swapping instructions and adopting pipeline control, compared with the NMOS HD6801/HD6803 FAMILY.

APPLICATION NOTES summarize typical programs for the HD6301/6303 FAMILY to help users better understand instruction set and to provide them with references for making more customized programs.

Programs described in APPLICATION NOTES have already been debugged.  
However, please be sure to check the operation in actual use.

# Section 7

## Software Application Notes

### Table of Contents

	Page
1. HOW TO USE APPLICATION NOTES .....	675
1.1 Formats .....	675
1.1.1 Specification Format (Format 1) .....	677
1.1.2 Description Format (Format 2) .....	683
1.1.3 Flowchart Format (Format 3).....	686
1.1.4 Program Listing Format (Format 4).....	688
1.2 How to Execute Programs .....	690
1.3 Symbols .....	692
PROGRAM APPLICATION EXAMPLES	
Program Application Table .....	693
MOVING DATA	
1. Filling Constant Values (Fill) .....	694
2. Moving Memory Blocks (Move) .....	698
3. Moving Strings (Moves).....	703
BRANCHING FROM TABLE	
4. Branching From Table (CCASE) .....	708
HANDLING ASCII	
5. Converting ASCII Lowercase Into Uppercase (TPR) .....	714
6. Converting ASCII Into 1-Byte Hexadecimal (Nibble).....	719
7. Converting 8-Bit Binary Data Into ASCII (COBYTE).....	724
BIT MANIPULATION	
8. Counting Number of Logical "1" Bits In 8-Bit Data (HCNT).....	729
9. Shifting 32-Bit Data (SHR).....	733
COUNTER	
10. 4-Digit BCD Counter (DECNT) .....	738
COMPARISON	
11. Comparing 32-Bit Binary Data (CMP) .....	743

ARITHMETIC OPERATION

12. Adding 32-Bit Binary Data (ADD) .....	749
13. Subtracting 32-Bit Binary Data (SUB) .....	755
14. Multiplying 16-Bit Binary Data (MUL) .....	761
15. Dividing 16-Bit Binary Data (DIV) .....	768
16. Adding 8-Digit BCD (ADDD) .....	774
17. Subtracting 8-Digit BCD (SUBD) .....	780
18. 16-Bit Square Root (SQRT) .....	786

CONVERTING BCD INTO HEXADECIMALS

19. Converting 2-Byte Hexadecimals Into 5-Digit BCD (HEX) .....	791
20. Converting 5-Digit BCD Into 2-Byte Hexadecimals (BCD) .....	796

SORTING

21. Sorting (SORT) .....	803
--------------------------	-----



1. How to Use APPLICATION NOTES

1.1 Formats

APPLICATION NOTES consist of Formats 1 to 4, shown in Fig. 1.1.

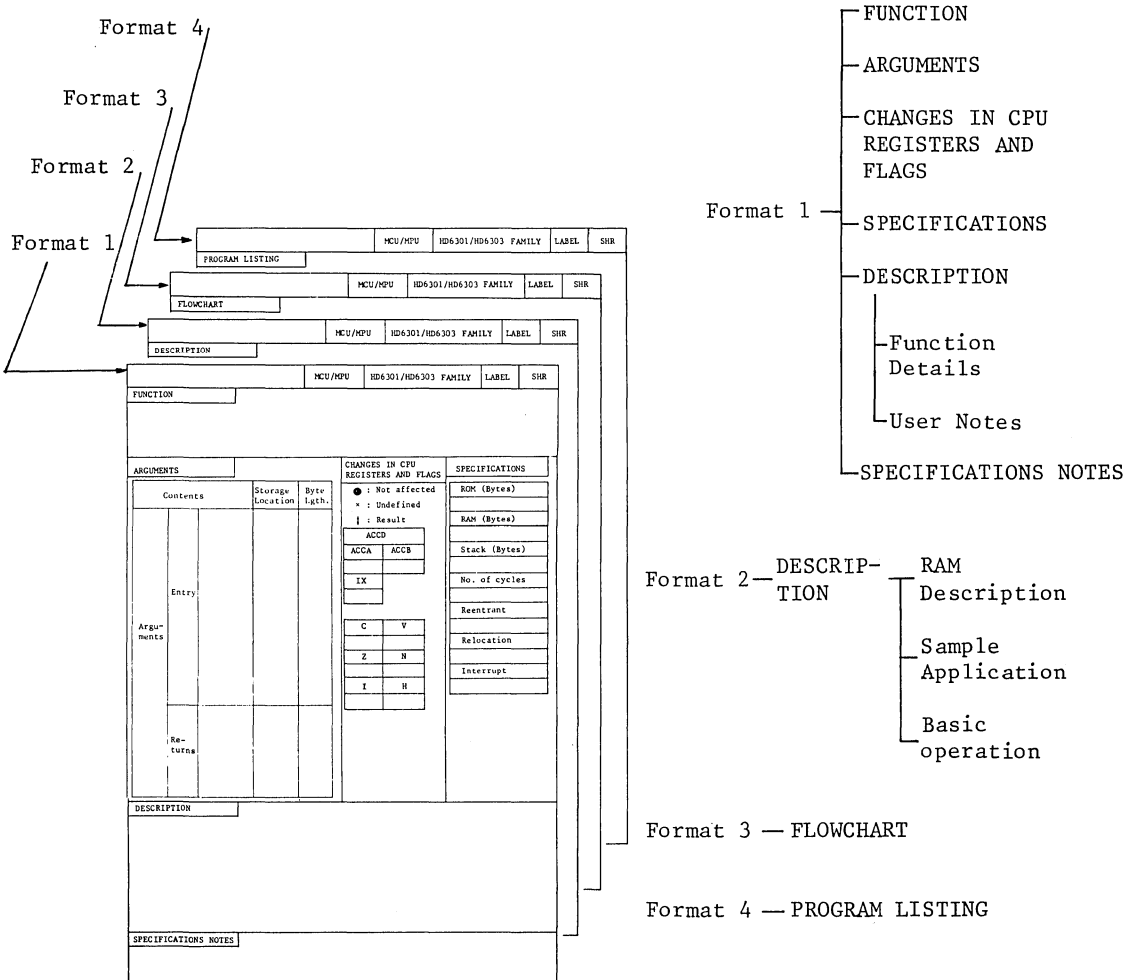


Fig. 1.1 APPLICATION NOTES Formats

Programs in APPLICATION NOTES can be implemented in two ways, i.e.  
(1) without change or (2) partially changed. Read the information that  
applies to the type of implementation to be carried out.

(1) Without change

(a) All of Format 1

(b) RAM Description and Sample Application in Format 2

(c) PROGRAM LISTING in Format 4

(2) Partially changed (user originals)

All of Formats 1 to 4 after reading these formats, change the  
FLOWCHART and PROGRAM LISTING according to user specification.

### 1.1.1 SPECIFICATION Format (Format 1)

The SPECIFICATION Format is represented in Fig. 1.2. It gives program functions and specifications. Each item in the format is described using Fig. 1.2.

(1)	ITEM NUMBER AND PROGRAM NAME	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	
(4)	FUNCTION	(2)			
			(6)		(7)
(5)	ARGUMENTS	CHANGES IN CPU REGISTERS AND FLAGS		SPECIFICATIONS	
	Contents	Storage Location	Byte Lgth.	<ul style="list-style-type: none"> <li>● : Not affected</li> <li>× : Undefined</li> <li>↓ : Result</li> </ul>	
	Arguments	Entry		ACCD	
				ACCA	ACCB
	Re- turns			IX	
				C	V
				Z	N
				I	H
				<ul style="list-style-type: none"> <li>ROM (Bytes)</li> <li>RAM (Bytes)</li> <li>Stack (Bytes)</li> <li>No. of cycles</li> <li>Reentrant</li> <li>Relocation</li> <li>Interrupt</li> </ul>	
(8)	DESCRIPTION				
	(1) Function Details				
	(2) User Notes				
(9)	SPECIFICATIONS NOTES				

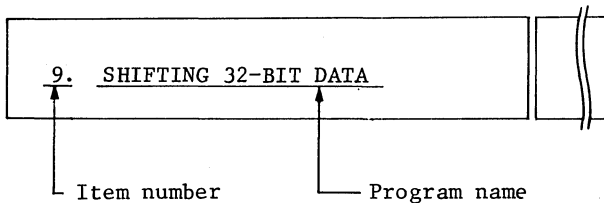
Fig. 1.2 SPECIFICATION Format



(1) ITEM NUMBER AND PROGRAM NAME:

Indicates item number and program name in APPLICATION NOTES.

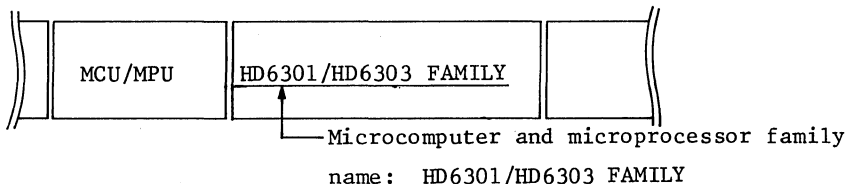
<Example>



(2) MCU/MPU:

Indicates names of microcomputer and microprocessor family applicable to a program.

<Example>

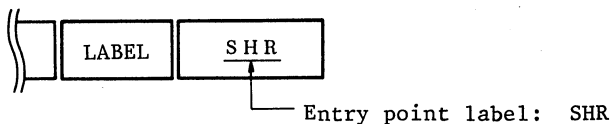


(3) LABEL:

Indicates the name identifying program entry point.

When using a program as it is, call the label "SHR".

<Example>



(4) FUNCTION:

Explains program functions.

<Example>

FUNCTION
(a) Shifts 32-bit binary data in IX and ACCD to right.
(b) Permits number of shifts to be freely determined.
(c) Permits easy multiplication of 32-bit binary data by $2^{-n}$ , (n:number of shifts)

(5) ARGUMENTS:

Explains entry arguments which must be set before execution of a program, and return arguments after execution.

(a) Contents:

Explains meanings of arguments.

(b) Storage Location:

Indicates registers and RAMs in which arguments are to be set. The RAM is presented as a label followed by "(RAM)".

(c) Byte length:

Indicates byte length of the arguments.

<Example>

ARGUMENTS				
Contents			Storage Location	Byte Lgth.
Argu- ments	Entry	Upper 16 bits of 32-bit binary data to be shifted to right	IX	2
		Lower 16 bits of 32-bit binary data to be shifted to right	ACCD	2
		Number of shifts	SFCNTR (RAM)	1
Re- turns		Upper 16 bits of shift result	IX	2
		Lower 16 bits of shift result	ACCD	2

(6) CHANGES IN CPU REGISTERS AND FLAGS:

Explains changes in CPU registers after executing a program and flag changes of condition code register. Meanings of abbreviations and symbols in the table are given as follows:

(a) CPU register

ACCA: Accumulator A

ACCB: Accumulator B

ACCD: Double accumulator (ACCA:ACCB)

IX: Index register

(b) Flags of condition code register

- C: Carry/borrow flag (carry and borrow)
- V: Overflow flag (Indication in case of 2's complement operation).
- Z: Zero flag (Indication in case of 0)
- N: Negative flag (Indication in case of negative)
- I: Interrupt flag (Interrupt mask)
- H: Half carry flag (Carry from bit 3 to bit 4)

(c) State of CPU registers and condition code register flags

- : Not affected: Maintains previous values after executing a program.
- ×: Undefined : Does not maintain previous values after executing a program.
- ↑: Result : Be set with the result of executing a program

<Example>

CHANGES IN CPU REGISTERS AND FLAGS	
● : Not affected	
× : Undefined	
↑ : Result	
ACCD	
ACCA	ACCB
↑	↑
IX	
↑	
C	V
×	×
Z	N
×	×
I	H
●	●

(Notes)

In the example, after executing a program, contents of index register (IX), condition code register (CCR), bit C, bit V, bit N and bit Z will be destroyed. Thus, register contents which will be destroyed should be saved before executing a program.

(7) SPECIFICATIONS:

Explains program specifications.

- (a) ROM (Bytes): Indicates ROM capacity used in a program.
- (b) RAM (Bytes): Indicates RAM capacity used in a program.
- (c) Stack (Bytes): Indicates stack size used in a program. The RAM capacity in this table does not include the stack size. When the program is executed,

it is necessary to reserve the stack size in RAM.

- (d) No. of cycles : Indicates maximum number of execution cycles when MCU executes a program. Calculate the execution time of the program as follows:

$$\text{Execution time (sec)} = \text{Cycle number} \times \text{cycle time}$$

$$\text{Cycle time (sec)} = 4/(\text{External oscillator (Hz)})$$

- (e) Reentrant : Indicates whether a program has a structure which can be called from two or more routines at the same time.
- (f) Relocation : Indicates whether a program can be located in any memory space.
- (g) Interrupt : Indicates whether MCU executes a program normally after serving an interrupt routine during program execution. If impossible, inhibit interrupt before the program is called.

<Example>

SPECIFICATIONS	
ROM (Bytes)	11
RAM (Bytes)	1
Stack (Bytes)	0
No. of cycles	261
Reentrant	No
Relocation	No
Interrupt	Yes



## 1.1.2 DESCRIPTION Format (Format 2)

The DESCRIPTION Format is represented in Fig. 1.3. It gives remaining Function Details, User Notes, RAM Description, Sample Application and Basic Operation.

Each item in the format is described using Fig. 1.3.

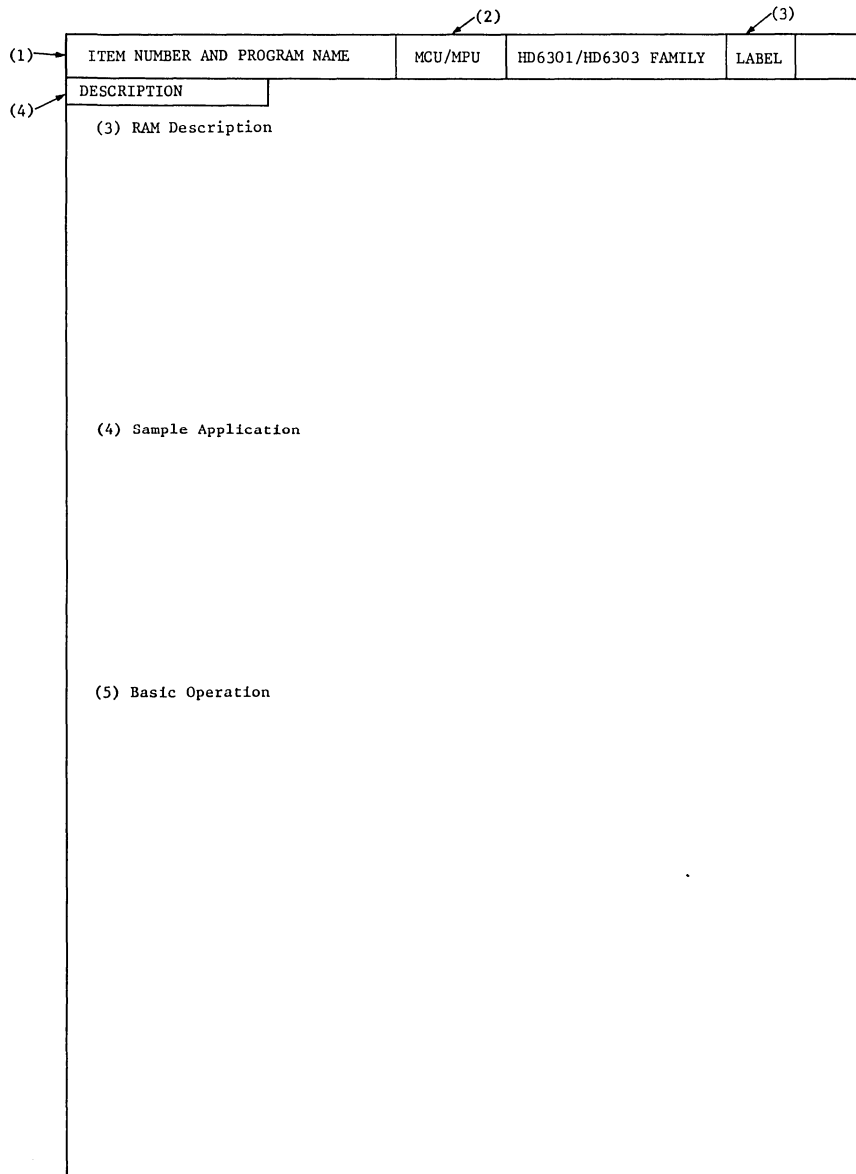


Fig. 1.3 DESCRIPTION Format

- |                                  |                                |
|----------------------------------|--------------------------------|
| (1) ITEM NUMBER AND PROGRAM NAME | } Same as SPECIFICATION Format |
| (2) MCU/MPU                      |                                |
| (3) LABEL                        |                                |



(c) Basic Operation: Indicates operating principles of a program.

<Example>

(5) Basic Operation

- (a) Uses 16-bit shift instruction (LSRD) provided in the HD6301/HD6303 FAMILY.
- (b) Upper 16 bits in 32-bit binary data are shifted to right. Here LSB is rotated to bit C. Lower 16 bits are rotated to right. At this time, LSB in bit C is rotated to MSB of lower 16 bits.
- (c) SFCNTR(RAM) is used to keep track of number of shifts. SFCNTR(RAM) is decremented every time (b) is executed.
- (d) Loops (b) to (c) until SFCNTR (RAM) is "0".



### 1.1.3 FLOWCHART Format (Format 3)

The FLOWCHART Format is represented in Fig. 4. It gives a program flowchart. Each item in the format is described using Fig. 1.4.

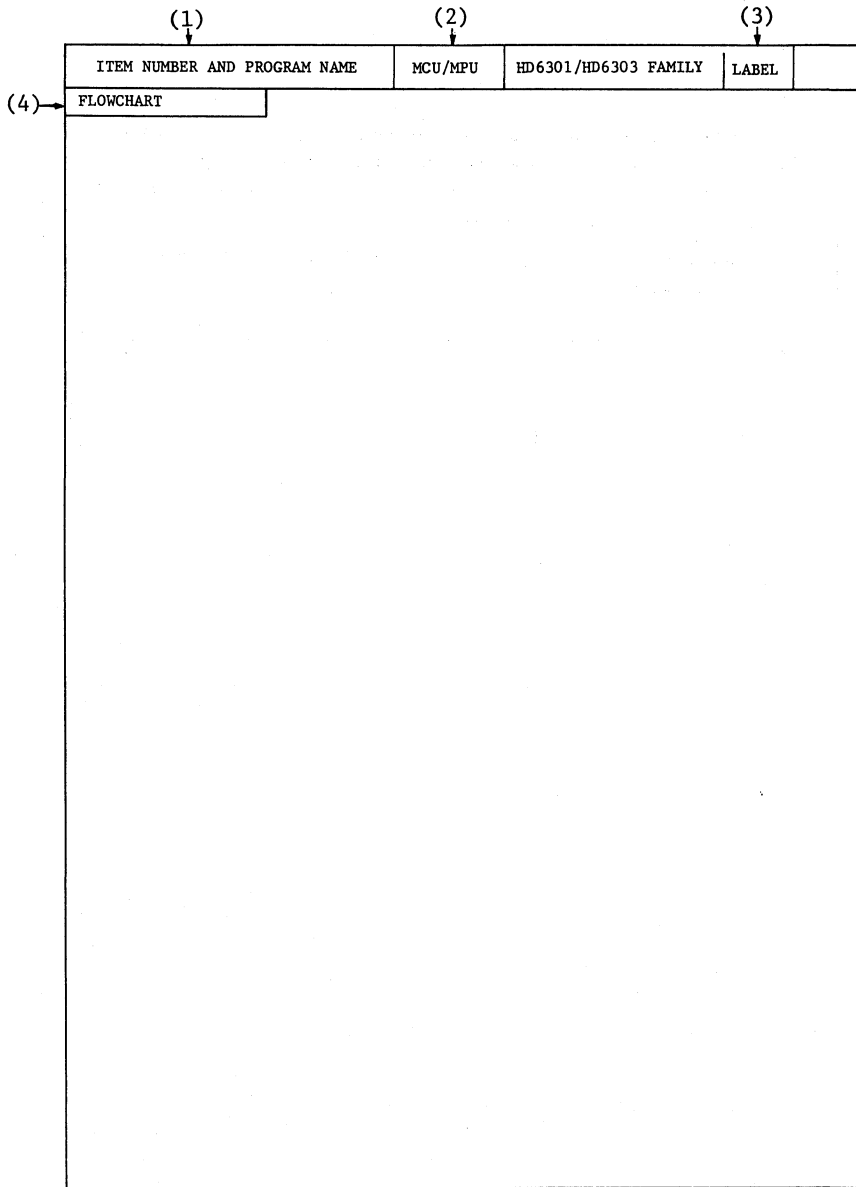


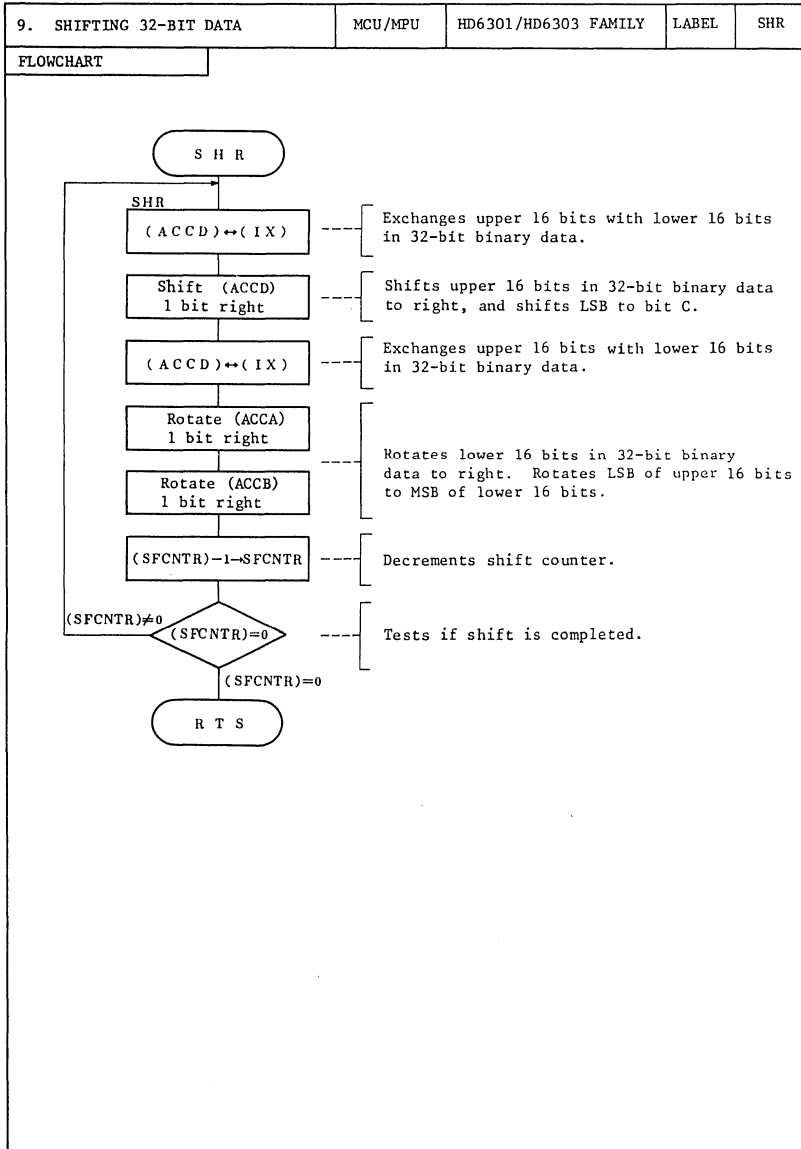
Fig. 1.4 FLOWCHART Format

- (1) ITEM NUMBER AND PROGRAM NAME
  - (2) MCU/MPU
  - (3) LABEL
- } Same as SPECIFICATION Format

(4) FLOWCHART:

Comments on the FLOWCHART are described in the column on the right.

<Example>



7

### 1.1.4 PROGRAM LISTING Format (Format 4)

The PROGRAM LISTING Format is represented in Fig. 1.5. Each item in the format is described using Fig. 1.5.

(1) ITEM NUMBER AND PROGRAM NAME	(2) MCU/MPU	HD6301/HD6303 FAMILY	(3) LABEL	
(4) PROGRAM LISTING				

Fig. 1.5 PROGRAM LISTING Format

- (1) ITEM NUMBER AND PROGRAM NAME
  - (2) MCU/MPU
  - (3) LABEL
- } Same as SPECIFICATION Format



(4) PROGRAM LISTING :

<Example>

9. SHIFTING 32-BIT DATA	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	SHR
PROGRAM LISTING				
00001				
00002				
00003				
00004				
00005				
00006				
00007				
00008				
00009				
00010				
00011				
00012				
00013				
00014				
C0015A	0080	ORG	\$80	
00016				
00017A	0080	0001 A SFCNTR	RMB 1	Shift counter
00018				
00019A	F000		ORG \$F000	
00020				
00021	F000	A SHR	EDU *	Entry point
00022A	F000	18	XGDX	Exchange 16-bit binary data
00023A	F001	04	LSRD	Shift upper 16-bit binary data
00024A	F002	18	XGDX	Exchange 16-bit binary data
00025A	F003	46	RORA	Shift Lower 16-bit binary data
00026A	F004	56	RORB	
00027A	F005	7A 0080 A	DEC	SFCNTR Decrement shift counter
00028A	F008	26 F6 F000	BNE	SHR Loop until shift counter = 0
00029A	F00A	39	RTS	

- (a) NAME : Name of a program. ( ) means entry point label.
- (b) ENTRY : shows storage location and contents of entry arguments.
- (c) RETURNS: shows storage location and contents of returns arguments.
- (d) SHR : shows entry point label.



## 1.2 How to Execute Programs

Relation between the programs in APPLICATION NOTES and user program is shown in Fig. 1.6. All programs in APPLICATION NOTES are formed as a subroutine, they should be proceeded as shown in Fig. 1.6 and (1) to (5) on the next page.

An example of a user program in which a program in APPLICATION NOTES accessed as a subroutine is shown in Fig. 1.7.

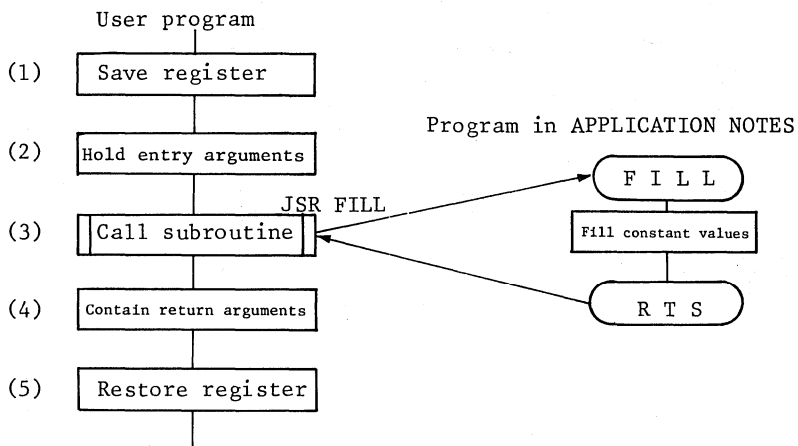
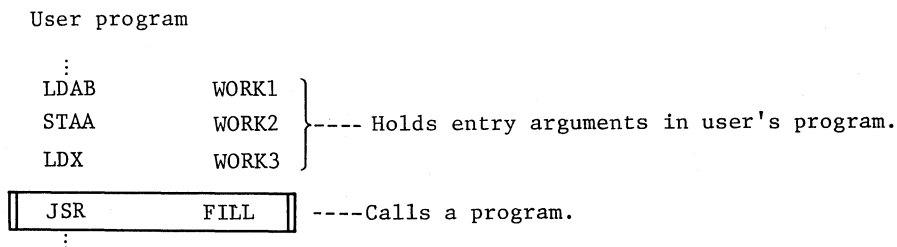


Fig. 1.6 Relation between User Program and Program in APPLICATION NOTES



Note) In the programs in APPLICATION NOTES, registers are saved when they are used as work areas and not as arguments.

Fig. 1.7 Example Showing How to Execute a Program

(1) Save register contents that will be destroyed by program execution:

CPU registers used in the programs may return to the user program while destroying the contents of the registers. Thus, registers should be saved, if necessary. Refer to the "CHANGES IN CPU REGISTERS AND FLAGS" column in SPECIFICATION Format (Format 1) for the register conditions after a program is executed.

(2) Hold entry arguments:

Holds entry arguments to the CPU registers or a particular address in the memory before calling a program in the user program. Refer to "ARGUMENTS" in the SPECIFICATION Format (Format 1) for entry arguments to be held.

(3) Call subroutine:

A program is called.

(4) Contain return arguments:

After a program is executed, the result contained in the return arguments must be handled according to the user's purpose.

Refer to "ARGUMENTS" in the SPECIFICATION Format (Format 1) for results.

(5) Restore register:

Registers saved in (1) are restored here. When (1) is operated, (5) must also be operated.

Moreover, note that when a program is used as a subroutine, the stack area shown in "SPECIFICATIONS" (Refer to (7) in Fig. 1.2) is necessary in addition to that for the subroutine call in the user program. When a subroutine is called, the above stack area must be assured.

### 1.3 Symbols

Symbols and abbreviations used in APPLICATION NOTES are defined as follows.

#### (a) Operation

( ) = Contents  
( ( ) ) = Index register addressing  
→ = Data transfer direction  
+ = Addition  
- = Subtraction  
× = Multiplication  
/ = Division  
∧ = AND  
∨ = OR  
⊕ = Exclusive OR  
¬ = NOT

#### (b) Register symbols in MCU/MPU

ACCA = Accumulator A  
ACCB = Accumulator B  
ACCD = Double accumulator (ACCA : ACCB)  
CCR = Condition code register  
IX = 16-bit index register  
IXH = Upper 8-bit index register  
IXL = Lower 8-bit index register

#### (c) Contents of bits 0 through 4 of condition code register

C = Carry or borrow	bit 0
V = 2's complement operation overflow	bit 1
Z = Zero	bit 2
N = Negative	bit 3
I = Interrupt mask	bit 4
H = Carry from bit 3 to bit 4	bit 5

#### (d) Others

= = Equal sign  
≠ = Not-equal sign  
>  
<  
≠ } = Comparison signs  
≠  
' ' = ASCII inside ' '  
\$ = Hexadecimal data  
: = Labels of sequential addresses

## PROGRAM APPLICATION TABLE

Item	Program	Label	Page
1	FILLING CONSTANT VALUES	FILL	26
2	MOVING MEMORY BLOCKS	MOVE	30
3	MOVING STRINGS	MOVES	35
4	BRANCHING FROM TABLE	CCASE	40
5	CONVERTING ASCII LOWERCASE INTO UPPERCASE	TPR	46
6	CONVERTING ASCII INTO 1-BYTE HEXADECIMAL	NIBBLE	51
7	CONVERTING 8-BIT BINARY DATA INTO ASCII	COBYTE	56
8	COUNTING NUMBER OF LOGICAL "1" BITS IN 8-BIT DATA	HCNT	61
9	SHIFTING 32-BIT DATA	SHR	65
10	4-DIGIT BCD COUNTER	DECNT	70
11	COMPARING 32-BIT BINARY DATA	CMP	75
12	ADDING 32-BIT BINARY DATA	ADD	81
13	SUBTRACTING 32-BIT BINARY DATA	SUB	87
14	MULTIPLYING 16-BIT BINARY DATA	MUL	93
15	DIVIDING 16-BIT BINARY DATA	DIV	100
16	ADDING 8-DIGIT BCD	ADDD	106
17	SUBTRACTING 8-DIGIT BCD	SUBD	112
18	16-BIT SQUARE ROOT	SQRT	118
19	CONVERTING 2-BYTE HEXADECIMALS INTO 5-DIGIT BCD	HEX	123
20	CONVERTING 5-DIGIT BCD INTO 2-BYTE HEXADECIMALS	BCD	128
21	SORTING	SORT	135



1. FILLING CONSTANT VALUES	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	FILL
----------------------------	---------	----------------------	-------	------

**FUNCTION**

(a) Stores one-byte constant in RAM.  
 (b) Permits RAM location and byte length to be freely selected.  
 (c) Permits easy clearing of RAM.

ARGUMENTS				CHANGES IN CPU REGISTERS AND FLAGS	SPECIFICATIONS																																											
Contents		Storage Location	Byte Lgth.	● : Not affected × : Undefined † : Result  <table border="1"> <tr><th colspan="2">ACCD</th></tr> <tr><th>ACCA</th><th>ACCB</th></tr> <tr><td>●</td><td>×</td></tr> <tr><th>IX</th><td></td></tr> <tr><td>×</td><td></td></tr> </table> <table border="1"> <tr><th>C</th><th>V</th></tr> <tr><td>●</td><td>×</td></tr> <tr><th>Z</th><td>N</td></tr> <tr><td>×</td><td>×</td></tr> <tr><th>I</th><td>H</td></tr> <tr><td>●</td><td>●</td></tr> </table>	ACCD		ACCA	ACCB	●	×	IX		×		C	V	●	×	Z	N	×	×	I	H	●	●	<table border="1"> <tr><th colspan="2">ROM (Bytes)</th></tr> <tr><td>7</td></tr> <tr><th colspan="2">RAM (Bytes)</th></tr> <tr><td>0</td></tr> <tr><th colspan="2">Stack (Bytes)</th></tr> <tr><td>0</td></tr> <tr><th colspan="2">No. of cycles</th></tr> <tr><td>149</td></tr> <tr><th colspan="2">Reentrant</th></tr> <tr><td>Yes</td></tr> <tr><th colspan="2">Relocation</th></tr> <tr><td>Yes</td></tr> <tr><th colspan="2">Interrupt</th></tr> <tr><td>Yes</td></tr> </table>	ROM (Bytes)		7	RAM (Bytes)		0	Stack (Bytes)		0	No. of cycles		149	Reentrant		Yes	Relocation		Yes	Interrupt		Yes
ACCD																																																
ACCA	ACCB																																															
●	×																																															
IX																																																
×																																																
C	V																																															
●	×																																															
Z	N																																															
×	×																																															
I	H																																															
●	●																																															
ROM (Bytes)																																																
7																																																
RAM (Bytes)																																																
0																																																
Stack (Bytes)																																																
0																																																
No. of cycles																																																
149																																																
Reentrant																																																
Yes																																																
Relocation																																																
Yes																																																
Interrupt																																																
Yes																																																
Arguments	Entry	Constant	ACCA	1																																												
		Byte Length	ACCB	1																																												
		Starting Address	IX	2																																												
Returns	—	—	—	—																																												

**DESCRIPTION**

(1) Function Details

(a) Argument details  
 ACCA: Holds one-byte constant in RAM.  
 ACCB: Holds byte length of constant.  
 IX : Holds starting address of RAM.

(b) Fig. 1 shows example of FILL execution.  
 If entry arguments are as shown in part ① of Fig. 1, \$57 in ACCA is stored in RAM as shown in part ② of Fig. 1.

① Entry arguments

ACCA (\$57)	b7 ACCA b0
ACCB (\$0A)	ACC B
IX (\$0090)	b15 IX b0

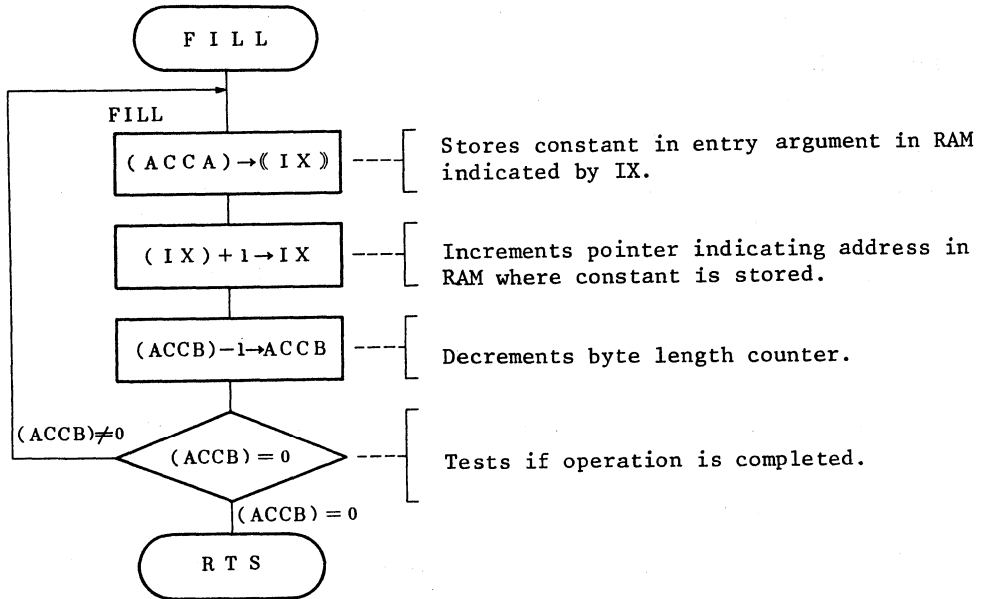
Fig. 1 Example of FILL execution

**SPECIFICATIONS NOTES**

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to write constant in 16-byte RAM.



## FLOWCHART



PROGRAM LISTING
-----------------

```

00001
00002
00003
00004
00005
00006
00007
00008
00009
00010
00011
00012
00013
00014A F000
00015
00016
00017A F000 A7 00 A
00018A F002 08
00019A F003 5A
00020A F004 26 FA F000
00021A F006 39

*****
*
*      NAME : FILLING CONSTANT VALVE (FILL)
*
*
*****
*
*      ENTRY      :      ACCA (CONSTANT)
*                  ACCB (BYTE COUNTER)
*                  IX  (START ADDR)
*
*      RETURNS   :      NOTHING
*
*****
*
*      ORG      $F000
*
*      EQU      *      Entry point
*      STAA    0.X    Store constant
*      INX     Increment ADDR
*      DECB   Decrement byte counter
*      BNE    FILL   Loop until byte counter = 0
*      RTS

```

**FUNCTION**

(a) Moves data block in memory to RAM.  
 (b) Permits byte length and source and destination addresses to be freely selected in memory.

<b>ARGUMENTS</b>		<b>CHANGES IN CPU REGISTERS AND FLAGS</b>	<b>SPECIFICATIONS</b>																																																									
<table border="1" style="width:100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2">Contents</th> <th>Storage Location</th> <th>Byte Lgth.</th> </tr> </thead> <tbody> <tr> <td rowspan="3">Arguments</td> <td>Entry</td> <td></td> <td></td> </tr> <tr> <td>Source starting address</td> <td>IX</td> <td>2</td> </tr> <tr> <td>Destination starting address</td> <td>DEA(RAM)</td> <td>2</td> </tr> <tr> <td>Byte length</td> <td>ACCB</td> <td>1</td> </tr> <tr> <td>Re-returns</td> <td>—</td> <td>—</td> <td>—</td> </tr> </tbody> </table>	Contents		Storage Location	Byte Lgth.	Arguments	Entry			Source starting address	IX	2	Destination starting address	DEA(RAM)	2	Byte length	ACCB	1	Re-returns	—	—	—		<p>● : Not affected          × : Undefined          ↓ : Result</p> <table border="1" style="width:100%; border-collapse: collapse; margin-bottom: 10px;"> <tr><th colspan="2">ACCD</th></tr> <tr> <td>ACCA</td> <td>ACCB</td> </tr> <tr> <td>×</td> <td>×</td> </tr> <tr> <td>IX</td> <td></td> </tr> <tr> <td>×</td> <td></td> </tr> </table> <table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td>C</td> <td>V</td> </tr> <tr> <td>●</td> <td>×</td> </tr> <tr> <td>Z</td> <td>N</td> </tr> <tr> <td>×</td> <td>×</td> </tr> <tr> <td>I</td> <td>H</td> </tr> <tr> <td>●</td> <td>●</td> </tr> </table>	ACCD		ACCA	ACCB	×	×	IX		×		C	V	●	×	Z	N	×	×	I	H	●	●	<table border="1" style="width:100%; border-collapse: collapse;"> <tr><td>ROM (Bytes)</td></tr> <tr><td>16</td></tr> <tr><td>RAM (Bytes)</td></tr> <tr><td>2</td></tr> <tr><td>Stack (Bytes)</td></tr> <tr><td>2</td></tr> <tr><td>No. of cycles</td></tr> <tr><td>501</td></tr> <tr><td>Reentrant</td></tr> <tr><td>No</td></tr> <tr><td>Relocation</td></tr> <tr><td>No</td></tr> <tr><td>Interrupt</td></tr> <tr><td>Yes</td></tr> </table>	ROM (Bytes)	16	RAM (Bytes)	2	Stack (Bytes)	2	No. of cycles	501	Reentrant	No	Relocation	No	Interrupt	Yes
	Contents		Storage Location	Byte Lgth.																																																								
	Arguments	Entry																																																										
		Source starting address	IX	2																																																								
Destination starting address		DEA(RAM)	2																																																									
Byte length	ACCB	1																																																										
Re-returns	—	—	—																																																									
ACCD																																																												
ACCA	ACCB																																																											
×	×																																																											
IX																																																												
×																																																												
C	V																																																											
●	×																																																											
Z	N																																																											
×	×																																																											
I	H																																																											
●	●																																																											
ROM (Bytes)																																																												
16																																																												
RAM (Bytes)																																																												
2																																																												
Stack (Bytes)																																																												
2																																																												
No. of cycles																																																												
501																																																												
Reentrant																																																												
No																																																												
Relocation																																																												
No																																																												
Interrupt																																																												
Yes																																																												

**DESCRIPTION**

(1) Function Details

(a) Argument details

IX: Holds source starting address in 2-byte hexadecimal number.  
 DEA(RAM): Holds destination starting address in 2-byte hexadecimal number.  
 ACCB: Holds byte length of data block to be moved in 1-byte hexadecimal number.

**SPECIFICATIONS NOTES**

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed for 16-byte data move.

DESCRIPTION

(b) Fig. 1 shows example of MOVE execution.  
 If entry arguments are as shown in part ① of Fig. 1, data in source (\$F000 - \$F009) is moved to destination (\$0090 - \$0099) as shown in part ② of Fig. 1.

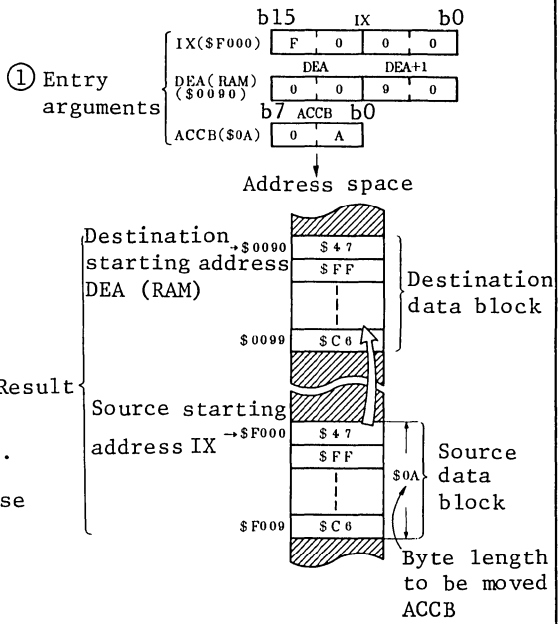


Fig. 1 Example of MOVE execution

(2) User Notes

- (a) As ACCB is only one byte in length, its data must be between \$01 and \$FF.
- (b) ACCB should not held to "0", otherwise data of 256 bytes will be moved.
- (c) Sets entry so that source area (Fig. 2 A) and destination area (Fig. 2 C) do not overlap. If they do, the source data in overlapping area (Fig. 2 B) will be destroyed.

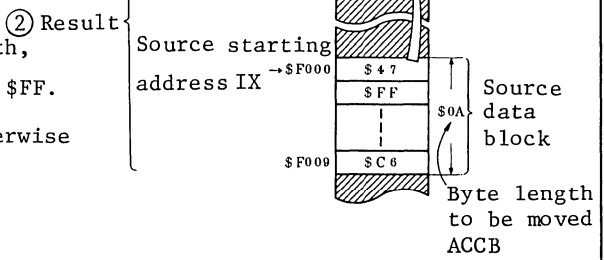
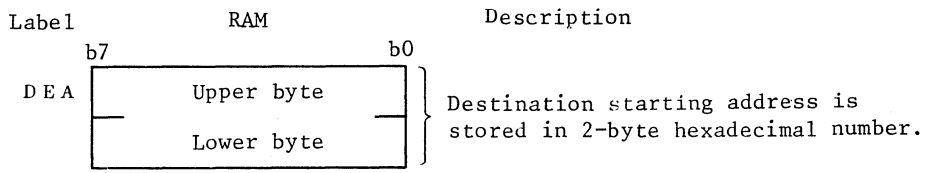


Fig. 2 Example of overlapping the source area with destination area

(3) RAM Description



DESCRIPTION

(4) Sample Application

Subroutine MOVE is called after source starting address, destination starting address and byte length to be moved are held.

```

WORK1   RMB   2  ----- Reserves memory byte for source starting
                               address.
WORK2   RMB   2  ----- Reserves memory byte for destination starting
                               address.
WORK3   RMB   1  ----- Reserves memory byte for byte length to
                               be moved.
        |
        |
        |
        PSHA      ----- Saves register contents that will be
                               destroyed by executing MOVE.
        LDX      WORK1 ----- Loads source starting address into
                               entry argument (IX).
        LDD      WORK2 }----- Stores destination starting address into
        STD      DEA  }----- entry argument (DEA).
        LDAB     WORK3 ----- Loads byte length to be moved into entry
                               argument (ACCB).

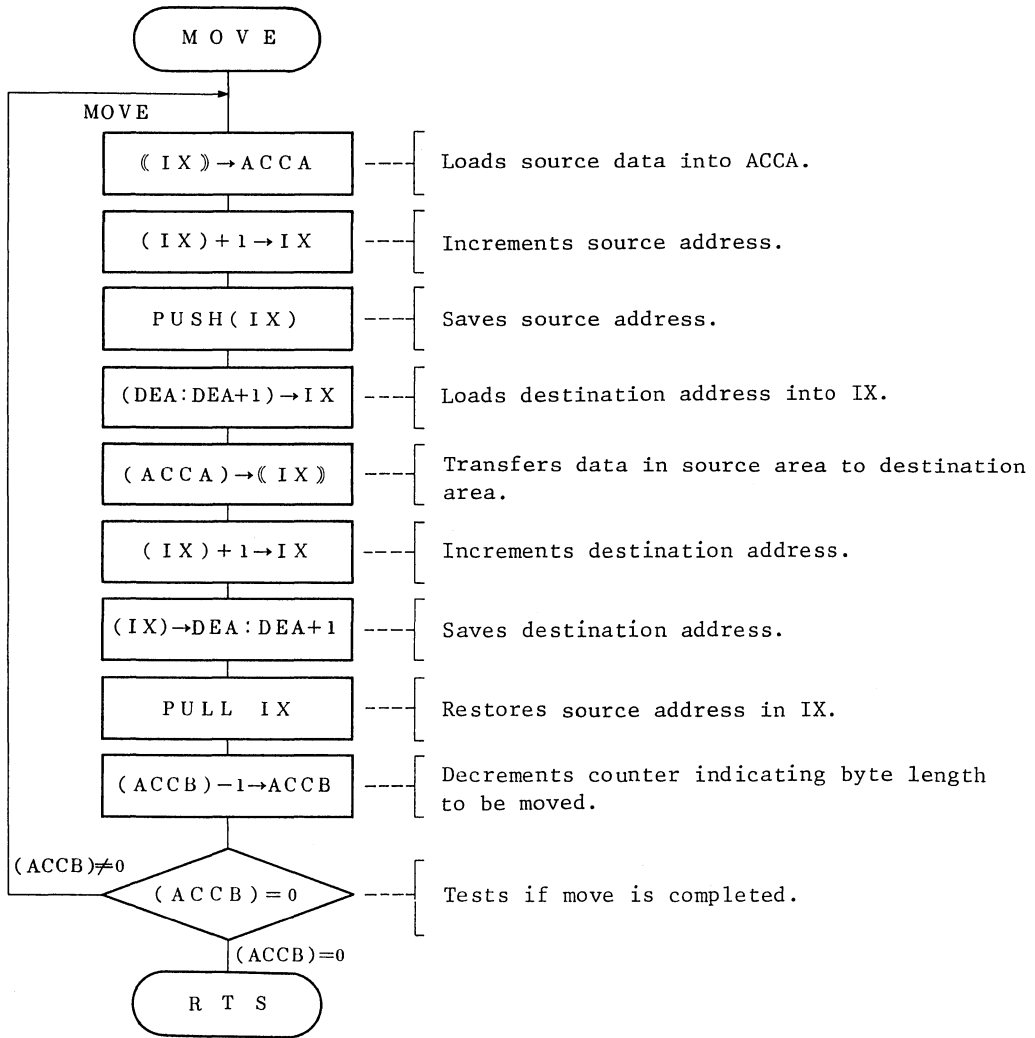
        JSR      MOVE ----- Calls subroutine MOVE.
        PULA      ----- Restores register.
        |
        |

```

(5) Basic Operation

- (a) IX is used to indicate source and destination addresses, which are alternately loaded into IX.
- (b) Data is moved from source area to destination area, one by one in order, using index addressing mode.
- (c) ACCB is used to indicate byte length to be moved. It is decremented each time (b) is executed. (b) is looped until ACCB is "0".

## FLOWCHART





## PROGRAM LISTING

```

00001 *****
00002 *
00003 *      NAME : MOVING MEMORY BLOCKS (MOVE)      *
00004 *
00005 *****
00006 *
00007 *      ENTRY : IX (SOURCE ADDR)                  *
00008 *      DEA (DESTINATION ADDR)                  *
00009 *      ACCB (TRANSFER COUNTER)                  *
00010 *      RETURNS : NOTHING                          *
00011 *
00012 *****
00013 *
00014A 0080          ORG      $80
00015 *
00016A 0080      0002  A  DEA      RMB      2      Destination ADDR
00017 *
00018A F000          ORG      $F000
00019 *
00020          F000  A  MOVE      EQU      *      Entry point
00021A F000 A6 00      A      LDAA     0.X      Load transfer data
00022A F002 08          INX          Increment source ADDR
00023A F003 3C          PSHX         Push source ADDR
00024A F004 DE 80      A      LDX     DEA      Load destination ADDR
00025A F006 A7 00      A      STAA    0.X      Store transfer data
00026A F008 08          INX          Increment destination ADDR
00027A F009 DF 80      A      STX     DEA      Store destination ADDR
00028A F00B 38          PULX         Pull source ADDR
00029A F00C 5A          DECB         Decrement transfer counter
00030A F00D 26      F1 F00D      BNE     MOVE      Loop until transfer counter = 0
00031A F00F 39          RTS

```

3. MOVING STRINGS	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	MOVES
-------------------	---------	----------------------	-------	-------

**FUNCTION**

- (a) Moves data block in memory to RAM.
- (b) Terminates moving process when terminator \$00 is found in data block.
- (c) Permits source and destination addresses to be freely selected in memory.

**ARGUMENTS**

Contents		Storage Location	Byte Lgth.
Arguments	Entry	IX	2
		DEAS(RAM)	2
Re-returns	—	—	—

**CHANGES IN CPU REGISTERS AND FLAGS**

- : Not affected
- × : Undefined
- ↓ : Result

ACCD	
ACCA	ACCB
×	●
IX	
×	

C	V
●	×
Z	N
×	×
I	H
●	●

**SPECIFICATIONS**

ROM (Bytes)	17
RAM (Bytes)	2
Stack (Bytes)	2
No. of cycles	507
Reentrant	No
Relocation	No
Interrupt	Yes

**DESCRIPTION**

(1) Function Details

(a) Argument details

- IX : Holds source starting address in 2-byte hexadecimal number.
- DEAS(RAM): Holds destination starting address in 2-byte hexadecimal number.

(2) Result

(b) Fig. 1 shows example of MOVES execution.

If entry arguments are as shown in part (1) of Fig. 1, data in source (\$F000) is moved to destination (\$0090) as shown

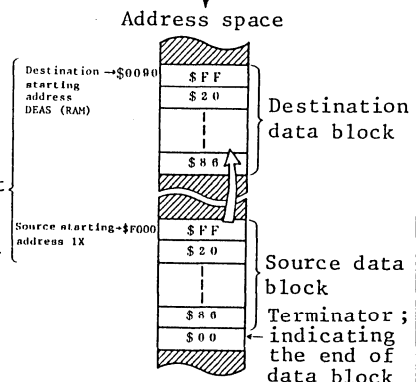
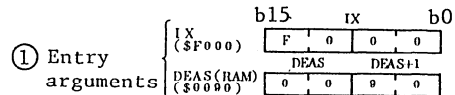


Fig. 1 Example of MOVES execution

**SPECIFICATIONS NOTES**

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed, when terminator is put at the 16th byte.

DESCRIPTION

in part ② of Fig. 1. When it loads terminator \$00, MCU terminates moving process.

(2) User Notes

- (a) Source data block is 64k bytes long or less. Last byte contains \$00 as terminator.
- (b) Source data must not contain any \$00 function other than terminator.
- (c) Holds entry arguments so that source area (Fig. 2 ①) and destination area (Fig. 2 ②) do not overlap. If they do, the source data in overlapping area (Fig. 2 ③) will be destroyed.

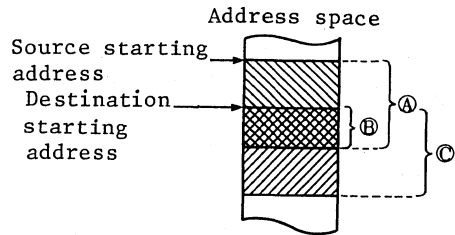


Fig. 2 Example of overlapping the source area with destination area

(3) RAM Description

Label	RAM	Description
DEAS	<div style="display: flex; justify-content: space-between;"> <span>b7</span> <span>b0</span> </div> <div style="border: 1px solid black; padding: 5px; margin: 5px auto; width: 80%;"> <div style="display: flex; justify-content: space-between;"> <div style="width: 45%;">Upper byte</div> <div style="width: 45%;">Lower byte</div> </div> </div>	Destination starting address is stored in 2-byte hexadecimal number.

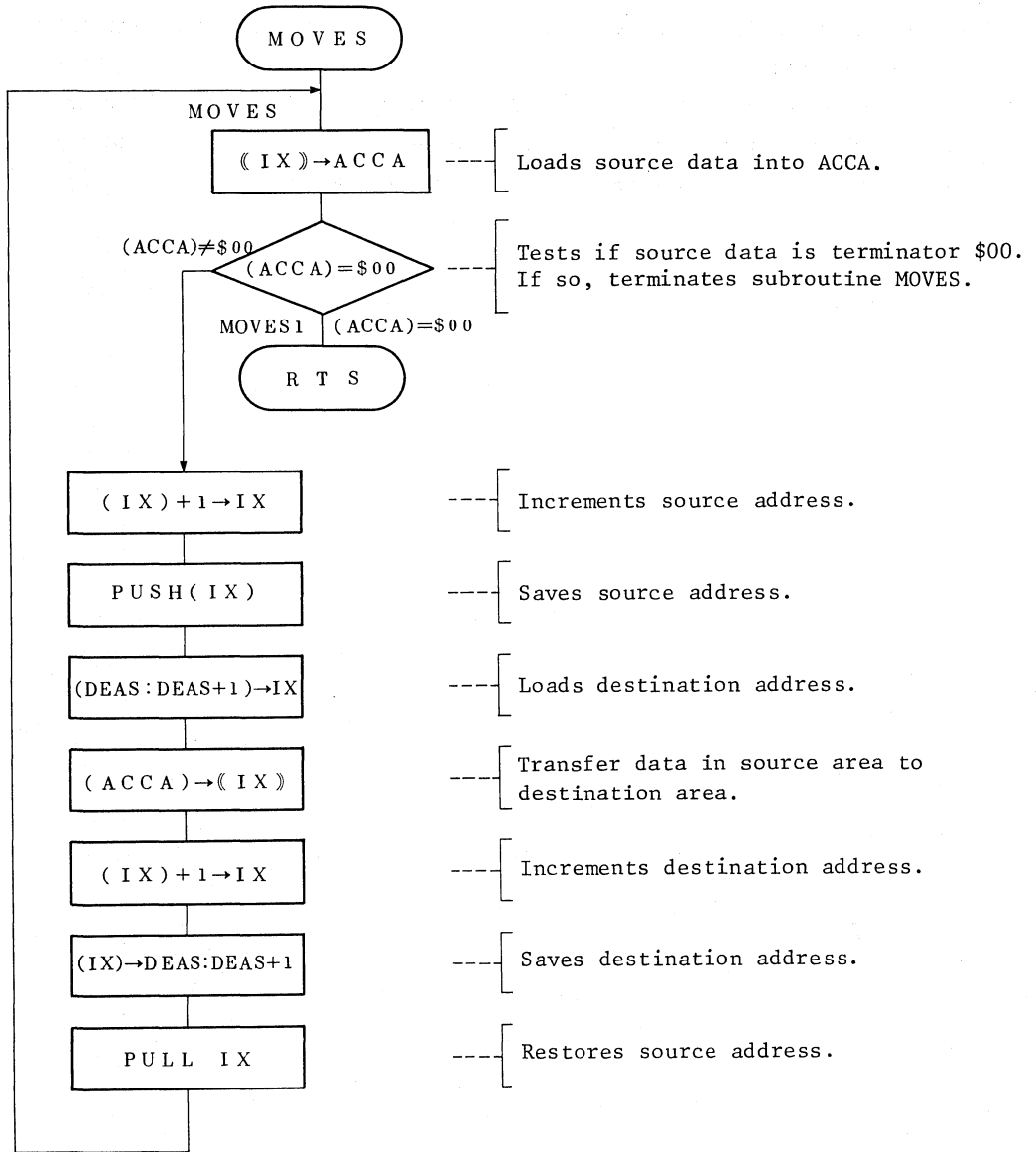
(4) Sample Application

Subroutine MOVES is called after source starting address and destination starting address are held.

WORK1	RMB	2	----	Reserves memory byte for source starting address.
WORK2	RMB	2	----	Reserves memory byte for destination starting address.
	⋮			
	PSHA		----	Saves register contents that will be destroyed by executing MOVES.
	LDX	WORK2	}	---- Stores destination starting address into entry argument (DEAS).
	STX	DEAS		
	LDX	WORK1	----	Loads source starting address into entry argument (IX).
	JSR MOVES		----	Calls subroutine MOVES.
	PULA		----	Restores register.
	⋮			

3. MOVING STRINGS	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	MOVES
DESCRIPTION				
<p>(5) Basic Operation</p> <p>(a) IX is used to indicate source and destination addresses, which are alternately loaded into IX.</p> <p>(b) Source data is loaded into ACCA using index addressing mode. Data in ACCA is tested if it is terminator. If so, subroutine MOVES is terminated. If not, moving process continues until the terminator is found.</p>				

FLOWCHART



PROGRAM LISTING

```

00001          *****
00002          *
00003          *      NAME : MOVING STRINGS (MOVES)      *
00004          *
00005          *****
00006          *
00007          *      ENTRY : IX (SOURCE ADDR)           *
00008          *      DEAS (DESTINATION ADDR)           *
00009          *      RETURNS : NOTHING                 *
00010          *
00011          *****
00012          *
00013A 0080          *      ORG      $80
00014          *
00015A 0080          *      0002 A DEAS  RMB  2      Destination ADDR
00016          *
00017A F000          *      ORG      $F000
00018          *
00019          *      F000 A MOVES EQU *      Entry point
00020A F000 A6 00 A LDAA 0.X      Load transfer data
00021A F002 27 0C F010 BEQ MOVSI  Branch if transfer data = 0
00022A F004 08          *      INX          Increment source ADDR
00023A F005 3C          *      PSHX         Push source ADDR
00024A F006 DE 80 A LDX  DEAS     Load destination ADDR
00025A F008 A7 00 A STAA 0.X     Store transfer data
00026A F00A 08          *      INX          Increment destination ADDR
00027A F00B DF 80 A STX  DEAS     Store destination ADDR
00028A F00D 38          *      PULX         Pull source ADDR
00029A F00E 20 F0 F000 BRA  MOVES  Branch MOVES
00030A F010 39          *      MOVSI    RTS

```

7

**FUNCTION**

(a) Loads service routine starting address into IX corresponding to the 1-byte command in ACCA.

(b) Permits easy decoding and processing of keyboard and other data inputted.

<b>ARGUMENTS</b>			<b>CHANGES IN CPU REGISTERS AND FLAGS</b>		<b>SPECIFICATIONS</b>																																								
			● : Not affected × : Undefined † : Result		<table border="1" style="width:100%; border-collapse: collapse;"> <tr><td>ROM (Bytes)</td></tr> <tr><td>18</td></tr> <tr><td>RAM (Bytes)</td></tr> <tr><td>0</td></tr> <tr><td>Stack (Bytes)</td></tr> <tr><td>0</td></tr> <tr><td>No. of cycles</td></tr> <tr><td>72</td></tr> <tr><td>Reentrant</td></tr> <tr><td>Yes</td></tr> <tr><td>Relocation</td></tr> <tr><td>Yes</td></tr> <tr><td>Interrupt</td></tr> <tr><td>Yes</td></tr> </table>	ROM (Bytes)	18	RAM (Bytes)	0	Stack (Bytes)	0	No. of cycles	72	Reentrant	Yes	Relocation	Yes	Interrupt	Yes																										
ROM (Bytes)																																													
18																																													
RAM (Bytes)																																													
0																																													
Stack (Bytes)																																													
0																																													
No. of cycles																																													
72																																													
Reentrant																																													
Yes																																													
Relocation																																													
Yes																																													
Interrupt																																													
Yes																																													
<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <th style="width:15%;">Contents</th> <th style="width:15%;">Storage Location</th> <th style="width:15%;">Byte Lgth.</th> </tr> <tr> <td rowspan="2" style="text-align:center; vertical-align:middle;">Entry</td> <td>Command</td> <td>ACCA</td> <td>1</td> </tr> <tr> <td>Data table starting address</td> <td>IX</td> <td>2</td> </tr> <tr> <td rowspan="2" style="text-align:center; vertical-align:middle;">Re-returns</td> <td>Service routine starting address</td> <td>IX</td> <td>2</td> </tr> <tr> <td>Command existence</td> <td>bit C (CCR)</td> <td>1</td> </tr> </table>			Contents	Storage Location	Byte Lgth.	Entry	Command	ACCA	1	Data table starting address	IX	2	Re-returns	Service routine starting address	IX	2	Command existence	bit C (CCR)	1	<table border="1" style="width:100%; border-collapse: collapse;"> <tr><td colspan="2" style="text-align:center;">ACCD</td></tr> <tr> <td style="width:50%;">ACCA</td> <td style="width:50%;">ACCB</td> </tr> <tr> <td style="text-align:center;">●</td> <td style="text-align:center;">×</td> </tr> <tr> <td style="text-align:center;">IX</td> <td></td> </tr> <tr> <td style="text-align:center;">†</td> <td></td> </tr> <tr><td colspan="2"> </td></tr> <tr> <td style="text-align:center;">C</td> <td style="text-align:center;">V</td> </tr> <tr> <td style="text-align:center;">†</td> <td style="text-align:center;">×</td> </tr> <tr> <td style="text-align:center;">Z</td> <td style="text-align:center;">N</td> </tr> <tr> <td style="text-align:center;">×</td> <td style="text-align:center;">×</td> </tr> <tr> <td style="text-align:center;">I</td> <td style="text-align:center;">H</td> </tr> <tr> <td style="text-align:center;">●</td> <td style="text-align:center;">●</td> </tr> </table>		ACCD		ACCA	ACCB	●	×	IX		†				C	V	†	×	Z	N	×	×	I	H	●	●
Contents	Storage Location	Byte Lgth.																																											
Entry	Command	ACCA	1																																										
	Data table starting address	IX	2																																										
Re-returns	Service routine starting address	IX	2																																										
	Command existence	bit C (CCR)	1																																										
ACCD																																													
ACCA	ACCB																																												
●	×																																												
IX																																													
†																																													
C	V																																												
†	×																																												
Z	N																																												
×	×																																												
I	H																																												
●	●																																												

**DESCRIPTION**

(1) Function Details

(a) Argument details

ACCA: Holds command such as ASCII.

IX : Holds data table starting address. After CCASE execution, IX contains service routine starting address corresponding to the command in ACCA as 2-byte hexadecimal number.

① Entry arguments

IX(\$FE00)	†	bit C	1	b15	F	E	0	0	IX	b0
ACCA(\$+2)	4	b7	ACCA	b0	4	2				

② Return arguments

IX(\$10+5)	1	bit C	1	b15	1	0	4	5	IX	b0
------------	---	-------	---	-----	---	---	---	---	----	----

Fig. 1 Example of CCASE execution

Starting address of data table → \$FE00

IX

Address	Address space	Description
\$+1		Command 'A'
\$+0		Service routine starting address for command 'A'
\$+2		Command 'B'
\$+1		Service routine starting address for command 'B'
\$+5		
...		
\$00		Terminator : indicating the end of data block

Fig. 2 Example of data table

**SPECIFICATIONS NOTES**

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to find data at the end of three data units.

4. BRANCHING FROM TABLE	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	CCASE
<p>DESCRIPTION</p> <p>bit C : Indicates CCASE termination.</p> <p>bit C=1 : Data in data table is the same as that in ACCA.</p> <p>bit C=0 : Data in data table differs from that in ACCA.</p> <p>(b) Fig. 1 shows example of CCASE execution.</p> <p>If entry arguments are as shown in part ① of Fig. 1, CCASE locates starting address of command service routine in data table (Fig. 2) and contains it in IX as shown in part ② of Fig. 1.</p> <p>(c) Data table shown in Fig. 2 must be set up before CCASE execution. It contains 3-byte data units beginning at \$FE00 and terminator indicating the end of the table. The first byte of the 3-byte data units is command. The second and the third bytes contain upper and lower bytes of command service routine starting address respectively.</p> <p>(2) User Notes</p> <p>Do not use \$00 as argument (IX) or as command in data table. It functions as terminator only.</p> <p>(3) RAM Description</p> <p>RAM is not used during CCASE execution.</p>				



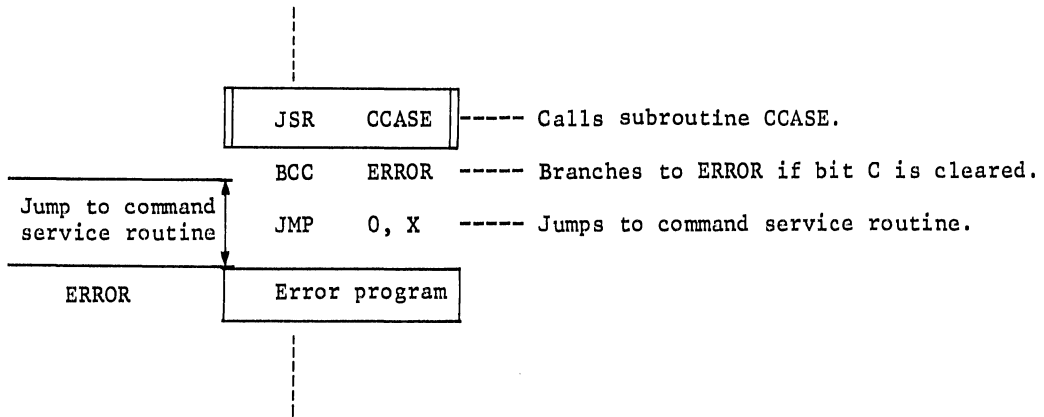
4. BRANCHING FROM TABLE	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	CCASE
DESCRIPTION				
(4) Sample Application				
Subroutine CCASE is called after command and starting address of data table are held.				
WORK1	RMB	1	-----	Reserves memory byte for command.
	PSHB		-----	Saves register contents that will be destroyed by CCASE execution.
	LDX	#DTABLE	-----	Loads data table starting address into entry argument (IX).
	LDAA	WORK1	-----	Loads command into entry argument (ACCA).
	JSR	CCASE	-----	Calls subroutine CCASE.
	PULB		-----	Restores register.
	BCC	ERROR	-----	Tests if there is command corresponding to inputted command in data table.
				*
				Program branching to command service routine
				Error program
ERROR			-----	Executes error program because there is no command corresponding to inputted command in data table.
	ORG	\$FD00	-----	Data table starting address.
DTABLE	FCC	'A'	-----	Command 'A'.
	FDB	\$F020	-----	Service routine starting address for command 'A'.
	FCC	'B'	-----	Command 'B'.
	FDB	\$F045	-----	Service routine starting address for command 'B'.
				-----
	FCB	\$00	-----	Terminator

4. BRANCHING FROM TABLE	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	CCASE
-------------------------	---------	----------------------	-------	-------

DESCRIPTION

(Note)

\* Example of branching to command service routine after CCASE execution;  
 CCASE functions only to store starting address of command service routine in IX. Program as in the example below to branch to the command service routine.

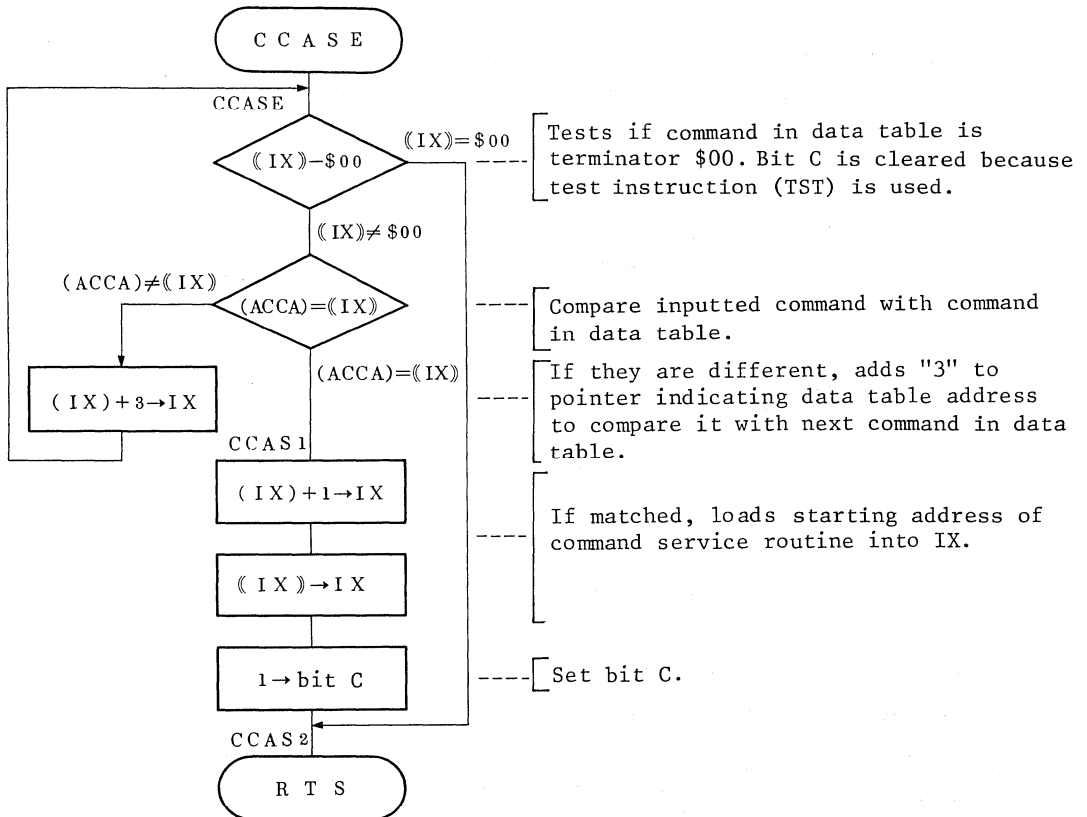


(5) Basic Operation

- (a) IX is used to indicate data table starting address.
- (b) Commands in data table are read in order from starting address using index addressing mode and compared with inputted command.
- (c) If commands in data table match the inputted command (ACCA), stores service routine starting address for the inputted command, sets bit C and subroutine CCASE is terminated.
- (d) If terminator \$00 is found in data table, clears bit C and subroutine CCASE is terminated.

7

## FLOWCHART



## PROGRAM LISTING

```

00001 *****
00002 *
00003 *      NAME : BRANCHING FROM TABLE (CCASE) *
00004 *
00005 *****
00006 *
00007 *      ENTRY      : ACCA (COMMAND ) *
00008 *                : IX (TABLE ADDR) *
00009 *      RETURNS    : IX (MODULE ADDR) *
00010 *                : CARRY(C=1:TRUE.C=0:FALES) *
00011 *
00012 *****
00013 *
00014A F000      ORG      $F000
00015 *
00016          F000 A CCASE EQU *      Entry point
00017A F000 6D 00 A      TST      0.X      Command of table = 0 ? (0 -> carry)
00018A F002 27 0D F011 BEQ      CCAS2    Branch if command of table = 0
00019A F004 A1 00 A      CMPA     0.X      Command = command of table ?
00020A F006 27 05 F00D BEQ      CCAS1    Branch if equal
00021A F008 08          INX          Increment pointer of table ADDR
00022A F009 08          INX
00023A F00A 08          INX
00024A F00B 20 F3 F000 BRA      CCASE    Branch CCASE
00025A F00D 08          CCAS1 INX      Increment pointer of table ADDR
00026A F00E EE 00 A      LDX      0.X      Load module ADDR
00027A F010 0D          SEC          Set carry bit to "1"
00028A F011 39          CCAS2 RTS

```

5. CONVERTING ASCII LOWERCASE INTO UPPERCASE		MCU/MPU	HD6301/HD6303 FAMILY	LABEL	TPR																																																	
FUNCTION																																																						
(a) Converts ASCII lowercase data in ACCA into uppercase and loads result into ACCA.																																																						
(b) Utilizes 7-bit ASCII in arguments.																																																						
ARGUMENTS			CHANGES IN CPU REGISTERS AND FLAGS		SPECIFICATIONS																																																	
<table border="1"> <thead> <tr> <th colspan="2">Contents</th> <th>Storage Location</th> <th>Byte Lgth.</th> </tr> </thead> <tbody> <tr> <td rowspan="2">Arguments</td> <td>Entry</td> <td>Lowercase (ASCII)</td> <td>ACCA</td> <td>1</td> </tr> <tr> <td>Re-returns</td> <td>Uppercase (ASCII)</td> <td>ACCA</td> <td>1</td> </tr> </tbody> </table>			Contents		Storage Location	Byte Lgth.	Arguments	Entry	Lowercase (ASCII)	ACCA	1	Re-returns	Uppercase (ASCII)	ACCA	1	<ul style="list-style-type: none"> <li>● : Not affected</li> <li>× : Undefined</li> <li>↓ : Result</li> </ul> <table border="1" style="margin-left: 20px;"> <thead> <tr> <th colspan="2">ACCD</th> </tr> <tr> <th>ACCA</th> <th>ACCB</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">↓</td> <td style="text-align: center;">●</td> </tr> <tr> <th>IX</th> <td></td> </tr> <tr> <td style="text-align: center;">●</td> <td></td> </tr> </tbody> </table> <table border="1" style="margin-left: 20px;"> <tbody> <tr> <td>C</td> <td>V</td> </tr> <tr> <td>×</td> <td>×</td> </tr> <tr> <td>Z</td> <td>N</td> </tr> <tr> <td>×</td> <td>×</td> </tr> <tr> <td>I</td> <td>H</td> </tr> <tr> <td style="text-align: center;">●</td> <td style="text-align: center;">●</td> </tr> </tbody> </table>		ACCD		ACCA	ACCB	↓	●	IX		●		C	V	×	×	Z	N	×	×	I	H	●	●	<table border="1"> <tbody> <tr> <td>ROM (Bytes)</td> <td>11</td> </tr> <tr> <td>RAM (Bytes)</td> <td>0</td> </tr> <tr> <td>Stack (Bytes)</td> <td>0</td> </tr> <tr> <td>No. of cycles</td> <td>17</td> </tr> <tr> <td>Reentrant</td> <td>Yes</td> </tr> <tr> <td>Relocation</td> <td>Yes</td> </tr> <tr> <td>Interrupt</td> <td>Yes</td> </tr> </tbody> </table>	ROM (Bytes)	11	RAM (Bytes)	0	Stack (Bytes)	0	No. of cycles	17	Reentrant	Yes	Relocation	Yes	Interrupt	Yes
Contents		Storage Location	Byte Lgth.																																																			
Arguments	Entry	Lowercase (ASCII)	ACCA	1																																																		
	Re-returns	Uppercase (ASCII)	ACCA	1																																																		
ACCD																																																						
ACCA	ACCB																																																					
↓	●																																																					
IX																																																						
●																																																						
C	V																																																					
×	×																																																					
Z	N																																																					
×	×																																																					
I	H																																																					
●	●																																																					
ROM (Bytes)	11																																																					
RAM (Bytes)	0																																																					
Stack (Bytes)	0																																																					
No. of cycles	17																																																					
Reentrant	Yes																																																					
Relocation	Yes																																																					
Interrupt	Yes																																																					
DESCRIPTION																																																						
(1) Function Details																																																						
(a) Argument details																																																						
ACCA: Holds ASCII lowercase data.																																																						
After TPR execution, contains the ① Entry argument (Lowercase 'a' \$61)																																																						
corresponding uppercase data.																																																						
(b) Fig. 1 shows example of TPR execution.																																																						
If lowercase 'a' (\$61) is held in ACCA as shown in part ① of Fig. 1, it is converted into uppercase 'A' (\$41), and the result is contained in ACCA as shown in part ② of Fig. 1.																																																						
<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;"> <p>① Entry argument (Lowercase 'a' \$61)</p> </div> <div style="text-align: center;"> <p>② Return argument (Uppercase 'A' \$41)</p> </div> </div> <p style="text-align: center;">Fig. 1 Example of TPR execution</p>																																																						
SPECIFICATIONS NOTES																																																						



5. CONVERTING ASCII LOWERCASE INTO UPPERCASE

MCU/MPU

HD6301/HD6303 FAMILY

LABEL

TPR

DESCRIPTION

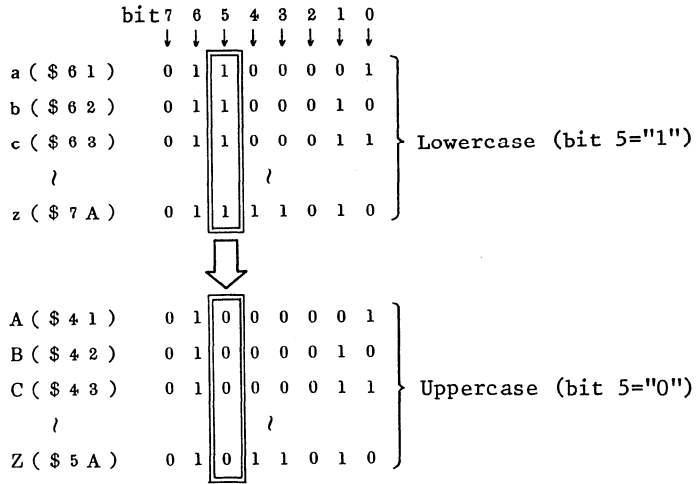
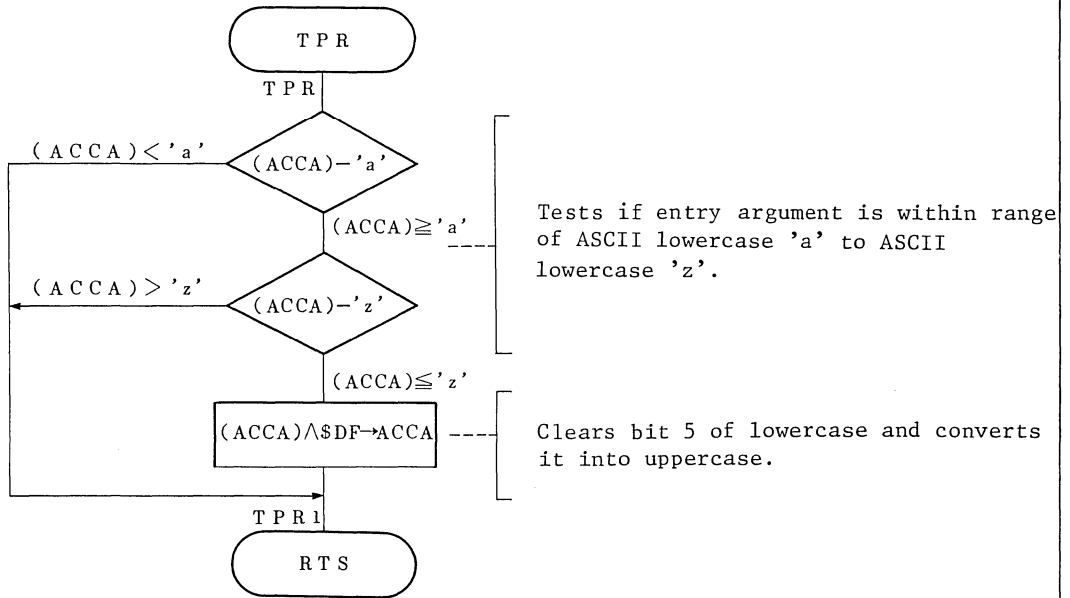


Fig. 2 Lowercase and uppercase of 7-bit ASCII

FLOWCHART





PROGRAM LISTING

```

00001 *****
00002 *
00003 * NAME : CONVERTING ASCII LOWERCASE INTO *
00004 * UPPERCASE (TPR) *
00005 *
00006 *****
00007 *
00008 * ENTRY : ACCA (ASCII LOWERCASE) *
00009 * RETURNS : ACCA (ASCII UPPERCASE) *
00010 *
00011 *****
00012 *
00013A F000 ORG $F000
00014 *
00015 F000 A TPR EQU * Entry point
00016A F000 81 61 A CMPA H'a ACCA-'a' ?
00017A F002 25 06 F00A BCS TPR1 Branch if ACCA<'a'
00018A F004 81 7A A CMPA H'z ACCA-'z' ?
00019A F006 22 02 F00A BHI TPR1 Branch if ACCA>'z'
00020A F008 84 DF A ANDA H$DF Convert Lowercase into Uppercase
00021A F00A 39 TPR1 RTS

```

6. CONVERTING ASCII INTO 1-BYTE HEXADECIMAL			MCU/MPU	HD6301/HD6303 FAMILY	LABEL	NIBBLE																																				
FUNCTION																																										
(a) Converts ASCII '0' to '9' and 'A' to 'F' in ACCA into 1-byte hexadecimal number and loads result into ACCA.																																										
(b) Utilizes 7-bit ASCII in arguments.																																										
ARGUMENTS			CHANGES IN CPU REGISTERS AND FLAGS		SPECIFICATIONS																																					
			<ul style="list-style-type: none"> <li>● : Not affected</li> <li>× : Undefined</li> <li>↑ : Result</li> </ul>		<table border="1"> <tr><td>ROM (Bytes)</td></tr> <tr><td>20</td></tr> <tr><td>RAM (Bytes)</td></tr> <tr><td>0</td></tr> <tr><td>Stack (Bytes)</td></tr> <tr><td>0</td></tr> <tr><td>No. of cycles</td></tr> <tr><td>28</td></tr> <tr><td>Reentrant</td></tr> <tr><td>Yes</td></tr> <tr><td>Relocation</td></tr> <tr><td>Yes</td></tr> <tr><td>Interrupt</td></tr> <tr><td>Yes</td></tr> </table>		ROM (Bytes)	20	RAM (Bytes)	0	Stack (Bytes)	0	No. of cycles	28	Reentrant	Yes	Relocation	Yes	Interrupt	Yes																						
ROM (Bytes)																																										
20																																										
RAM (Bytes)																																										
0																																										
Stack (Bytes)																																										
0																																										
No. of cycles																																										
28																																										
Reentrant																																										
Yes																																										
Relocation																																										
Yes																																										
Interrupt																																										
Yes																																										
<table border="1"> <thead> <tr> <th colspan="2">Contents</th> <th>Storage Location</th> <th>Byte Lgth.</th> </tr> </thead> <tbody> <tr> <td rowspan="2">Arguments</td> <td>Entry</td> <td>ASCII</td> <td>ACCA 1</td> </tr> <tr> <td rowspan="2">Re-returns</td> <td>1-byte hexadecimal number</td> <td>ACCA 1</td> </tr> <tr> <td></td> <td>Conversion/not conversion</td> <td>bit C (CCR) 1</td> </tr> </tbody> </table>			Contents		Storage Location	Byte Lgth.	Arguments	Entry	ASCII	ACCA 1	Re-returns	1-byte hexadecimal number	ACCA 1		Conversion/not conversion	bit C (CCR) 1	<table border="1"> <thead> <tr><th colspan="2">ACCD</th></tr> <tr><th>ACCA</th><th>ACCB</th></tr> </thead> <tbody> <tr><td>↓</td><td>●</td></tr> <tr><th>IX</th><td></td></tr> <tr><td>●</td><td></td></tr> <tr><th colspan="2"> </th></tr> <tr><th>C</th><th>V</th></tr> <tr><td>↓</td><td>×</td></tr> <tr><th>Z</th><th>N</th></tr> <tr><td>×</td><td>×</td></tr> <tr><th>I</th><th>H</th></tr> <tr><td>●</td><td>×</td></tr> </tbody> </table>		ACCD		ACCA	ACCB	↓	●	IX		●				C	V	↓	×	Z	N	×	×	I	H	●	×
Contents		Storage Location	Byte Lgth.																																							
Arguments	Entry	ASCII	ACCA 1																																							
	Re-returns	1-byte hexadecimal number	ACCA 1																																							
		Conversion/not conversion	bit C (CCR) 1																																							
ACCD																																										
ACCA	ACCB																																									
↓	●																																									
IX																																										
●																																										
C	V																																									
↓	×																																									
Z	N																																									
×	×																																									
I	H																																									
●	×																																									
DESCRIPTION																																										
(1) Function Details																																										
(a) Argument details																																										
ACCA: Holds ASCII. After NIBBLE execution contains 1-byte hexadecimal number.																																										
bit C : Shows state when NIBBLE is (CCR) executed.																																										
bit C=1 : Shows entry argument was ASCII other than '0' to '9' or 'A' to 'F'.																																										
bit C=0 : Shows subroutine NIBBLE is executed normally.																																										
(b) Fig. 1 shows example of NIBBLE execution. If entry argument is as shown in part ① of Fig. 1, \$0F, data converted from ASCII into 1-byte hexadecimal number, is contained in ACCA as shown in part ② of Fig. 1.																																										
<p>Fig. 1 Example of NIBBLE execution</p>																																										
SPECIFICATIONS NOTES																																										

6. CONVERTING ASCII INTO 1-BYTE HEXADECIMAL	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	NIBBLE
DESCRIPTION				
(2) User Notes				
Entry argument (ASCII) should be held within range of '0' to '9' or 'A' to 'F', otherwise data in ACCA is destroyed after NIBBLE execution.				
(3) RAM Description				
RAM is not used during NIBBLE execution.				
(4) Sample Application				
Subroutine NIBBLE is called after ASCII is held.				
WORK1	RMB	1	-----	Reserves memory byte for 1-digit ASCII.
WORK2	RMB	1	-----	Reserves memory byte for 1-byte hexadecimal number.
	LDAA	WORK1	-----	Loads ASCII into entry argument (ACCA).
	JSR	NIBBLE	-----	Calls subroutine NIBBLE.
	BCS	SKIP	-----	If ASCII is other than '0' to '9' or 'A' to 'F', branches to service routine.
	STAA	WORK2	-----	Stores 1-byte hexadecimal number (return argument (ACCA)) in RAM.
SKIP	<div style="border: 1px solid black; padding: 5px; display: inline-block;">           Service routine for ASCII other than '0' to '9' or 'A' to 'F'         </div>			

DESCRIPTION

(5) Basic Operation










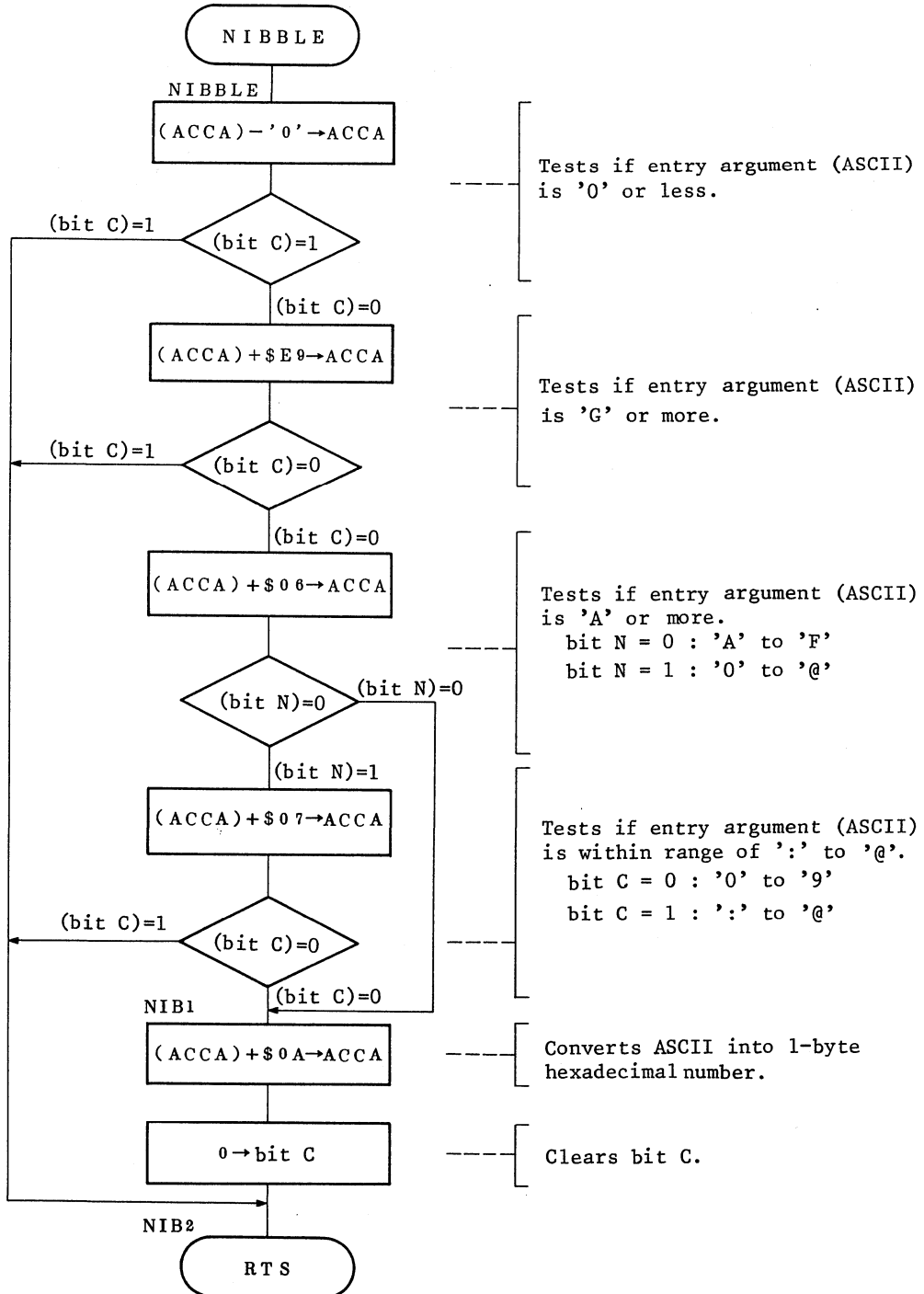
- (a) Bit C, resulting from comparison and subtraction of data in ACCA, is used to test if the data is within range of '0' to 'F' in ASCII table  
(note:  blocked area in table).
- (b) Addition continues between '0' and '@' . Then ':' to '@' (note:  cross hatched area in table) is deleted.
- (c) In cases other than between '0' and '9' or 'A' and 'F', bit C is set during (a) or (b) above.

Table 1. ASCII Table

LSD \ MSD	0	1	2	3	4	5	6	7
	000	001	010	011	100	101	110	111
0 0000	NUL	DLE	SP	0		P	\	p
1 0001	SOH	DC1	!	1	A	Q	a	q
2 0010	STX	DC2	"	2	B	R	b	r
3 0011	ETX	DC3	#	3	C	S	c	s
4 0100	EOT	DC4	\$	4	D	T	d	t
5 0101	ENG	NAK	%	5	E	U	e	u
6 0110	ACK	SYN	&	6	F	V	f	v
7 0111	BEL	ETB	'	7	G	W	g	w
8 1000	BS	CAN	(	8	H	X	h	x
9 1001	HT	EM	)	9	I	Y	i	y
A 1010	LF	SUB	*		J	Z	j	z
B 1011	VT	ESC	+		K	[	k	{
C 1100	FF	FS	,		L	\	l	
D 1101	CR	GS	-		M	]	m	}
E 1110	SO	RS	.		N	↑	n	~
F 1111	SI	VS	/		O	←	o	DEL

7

FLOWCHART



PROGRAM LISTING
-----------------

```

00001 *****
00002 *
00003 *      NAME : CONVERTING ASCII INTO      *
00004 *              1-BYTE HEXADECIMAL (NIBBLE) *
00005 *
00006 *****
00007 *
00008 *      ENTRY :  ACCA (ASCII)              *
00009 *      RETURNS : ACCA (BINARY DATA)     *
00010 *              CARRY(C=1;TRUE,C=0;FALES) *
00011 *****
00012 *
00013A F000          DRG      $F000
00014 *
00015          F000 A NIBBLE EQU      *      Entry point
00016A F000 80 30 A      SUBA      #'0      ACCA(ASCII code) - '0' ?
00017A F002 25 0F F013  BCS      NIB2      Branch if ACCA<'0'
00018A F004 8B E9 A      ADDA      #$E9      ACCA - 'G' -> ACCA
00019A F006 25 0B F013  BCS      NIB2      Branch if ACCA='G'
00020A F008 8B 06 A      ADDA      #6       Test '0'-'@' or 'A'-'F'
00021A F00A 2A 04 F010  BPL      NIB1      Branch if ACCA='A'-'F'
00022A F00C 8B 07 A      ADDA      #7       Test '0'-'9' or ':'-'@'
00023A F00E 25 03 F013  BCS      NIB2      Branch if ACCA=':'-'@'
00024A F010 8B 0A A NIB1  ADDA      #$A      Convert ASCII into binary data
00025A F012 0C          CLC          Clear carry
00026A F013 39          NIB2      RTS

```

7

7. CONVERTING 8-BIT BINARY DATA INTO ASCII	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	COBYTE
--	---------	----------------------	-------	--------

**FUNCTION**

- (a) Converts 8-bit binary data in ACCA into two ASCII characters and stores result in RAM.
- (b) Utilizes 7-bit ASCII in arguments.

**ARGUMENTS**

Contents		Storage Location	Byte Lgth.
Entry	8-bit binary data	ACCB	1
Arguments	Re-returns 2 ASCII characters	ACCD	2

**CHANGES IN CPU REGISTERS AND FLAGS**

- : Not affected
- × : Undefined
- ↓ : Result

ACCD	
ACCA	ACCB
↓	↓
IX	
●	

C	V
×	×
Z	N
×	×
I	H
●	●

**SPECIFICATIONS**

ROM (Bytes)	23
RAM (Bytes)	0
Stack (Bytes)	2
No. of cycles	57
Reentrant	Yes
Relocation	Yes
Interrupt	Yes

**DESCRIPTION**

(1) Function Details

(a) Argument details

ACCB: Holds 8-bit binary data to be converted into ASCII.

ACCD: Contains data converted, from upper and lower 4 bits of 8-bit binary data into 2 ASCII characters.

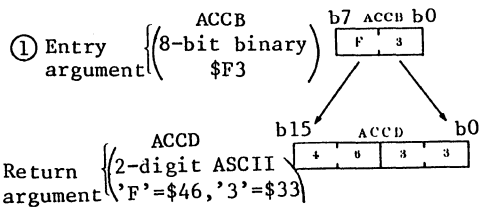


Fig. 1 Example of COBYTE execution

(b) Fig. 1 shows example of COBYTE

execution. If entry argument is as shown in part ① of Fig. 1, data converted from 8-bit binary data into ASCII is contained to ACCD as shown in part ② of Fig. 1.

**SPECIFICATIONS NOTES**

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to convert \$AA into ASCII.





## DESCRIPTION

## (5) Basic Operation

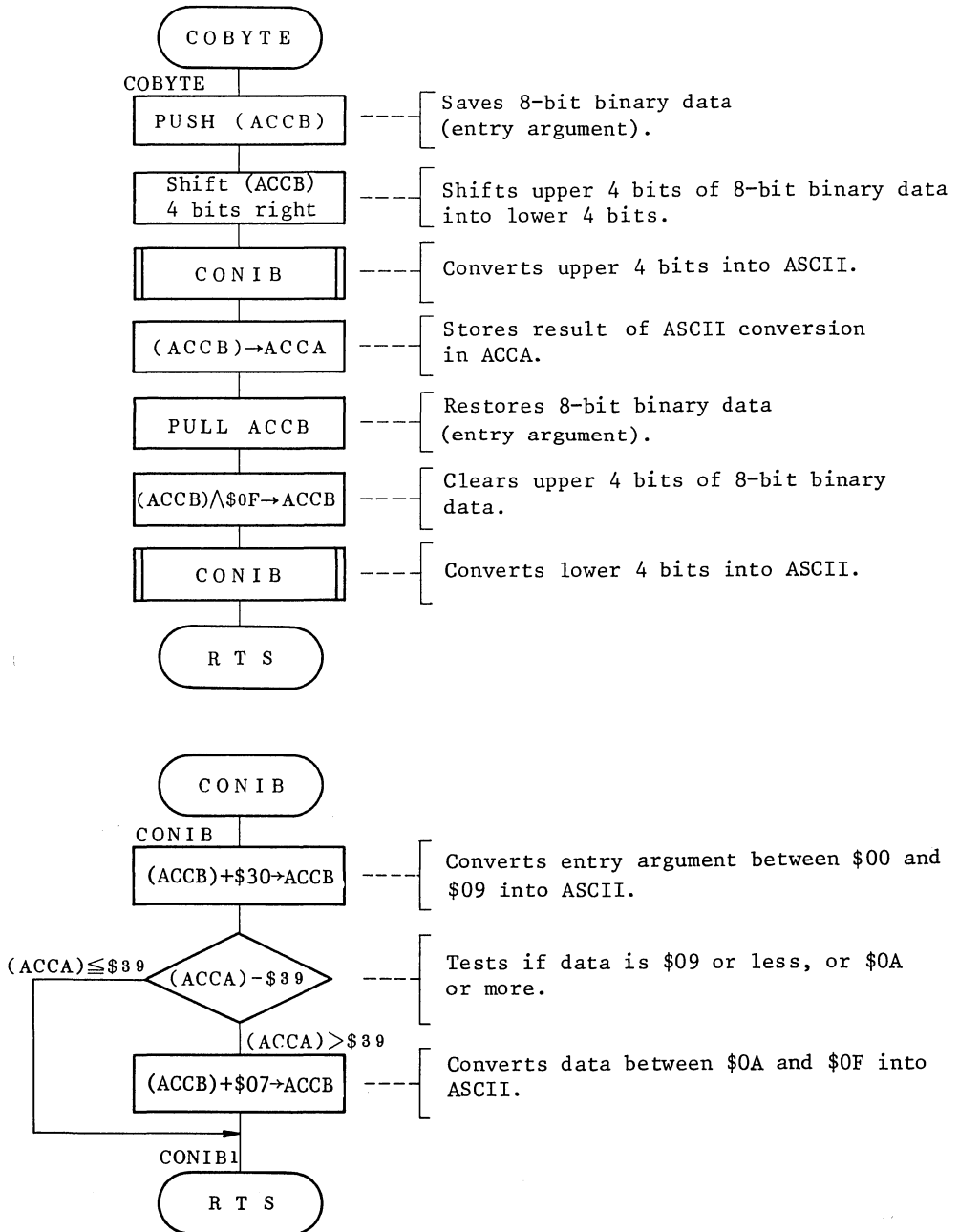
(a) 8-bit binary data in ACCB is divided into 4 upper and 4 lower bits.

(b) Divided data is then checked by a comparison instruction (CMP). If data is between \$00 and \$09 (□: blocked area in ASCII table as shown in Table 1), \$30 is added. If data is between \$0A and \$0F (□ in Table 1), \$37 is added. Result is converted into ASCII.

Table 1 ASCII Table

		MSD							
LSD		0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P	\	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	\$	4	D	T	d	t
5	0101	ENG	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	
D	1101	CR	GS	-	=	M	]	m	}
E	1110	SO	RS	.	>	N	↑	n	~
F	1111	SI	VS	/	?	O	←	o	DEL

FLOWCHART



PROGRAM LISTING

```

00001 *****
00002 *
00003 *      NAME : CONVERTING 8-BIT BINARY DATA      *
00004 *      INTO ASCII      (COBYTE)                  *
00005 *
00006 *****
00007 *
00008 *      ENTRY  : ACCB (8-BIT BINARY DATA)         *
00009 *      RETURNS : ACCD (2-BYTE ASCII)              *
00010 *
00011 *****
00012 *
00013A F000      DRG      $F000
00014 *
00015      F000 A COBYTE EQU      *      Entry point
00016A F000 37      PSHB      Push 8-bit binary data
00017A F001 54      LSRB      Shift upper 4 bits to Lower 4 bits
00018A F002 54      LSRB
00019A F003 54      LSRB
00020A F004 54      LSRB
00021A F005 8D 07 F00E BSR      CONIB      Convert upper 4 bits into ASCII
00022A F007 17      TBA      Transfer ASCII to ACCA
00023A F008 33      PULB      Pull 8-bit binary data
00024A F009 C4 0F      A      ANDB      #$0F      Mask upper 4 bits
00025A F00B 8D 01 F00E BSR      CONIB      Convert lower 4 bits to ASCII
00026A F00D 39      RTS
00027A F00E CB 30      A CONIB  ADDB      #'0      Convert into ASCII ('0'-'9')
00028A F010 C1 39      A      CMPB      #'9      ACCB= '0'-'9' or 'A'-'F' ?
00029A F012 23 02 F016 BLS      CONIB1    Branch if ACCB='0'-'9'
00030A F014 CB 07      A      ADDB      #$07      Convert into ASCII ('A'-'F')
00031A F016 39      CONIB1 RTS
    
```

8. COUNTING NUMBER OF LOGICAL "1" BITS IN 8-BIT DATA	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	HCNT
--	---------	----------------------	-------	------

FUNCTION

- (a) Counts number of logical "1" bits in 8-bit data in ACCA, and loads result in ACCB.
- (b) Permits easy parity checking.

ARGUMENTS

Contents		Storage Location	Byte Lgth.	
Arguments	Entry	8-bit data	ACCA	1
	Re-returns	Number of logical "1" bits	ACCB	1

CHANGES IN CPU REGISTERS AND FLAGS

- : Not affected
- × : Undefined
- ↑ ↓ : Result

ACCD	
ACCA	ACCB
●	↑
IX	
×	

C	V
●	×
Z	N
×	×
I	H
●	●

SPECIFICATIONS

ROM (Bytes)	13
RAM (Bytes)	0
Stack (Bytes)	0
No. of cycles	82
Reentrant	Yes
Relocation	Yes
Interrupt	Yes

DESCRIPTION

(1) Function Details

(a) Argument details

ACCA: Holds 8-bit data whose number of logical "1" bits is counted.

ACCB: Contains number of logical "1" bits in 8-bit data.

- (b) Fig. 1 shows example of HCNT execution. If entry argument is as shown in part ① of Fig. 1, number of logical "1" bits in 8-bit data is contained in ACCB as shown in part ② of Fig. 1.

(c) Contents of ACCA are saved after HCNT execution.

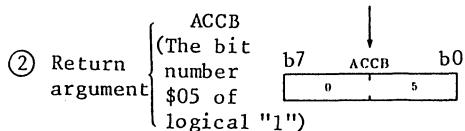
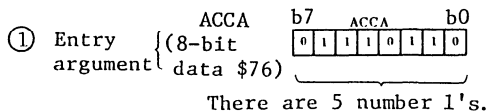


Fig. 1 Example of HCNT execution

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to count number of logical "1" bits in \$FF data.

8. COUNTING NUMBER OF LOGICAL "1" BITS IN 8-BIT DATA	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	HCNT
--	---------	----------------------	-------	------

DESCRIPTION

(2) User Notes

When counting number of logical "0" bits, take I's complement of ACCA before HCNT execution.

(3) RAM Description

RAM is not used during HCNT execution.

(4) Sample Application

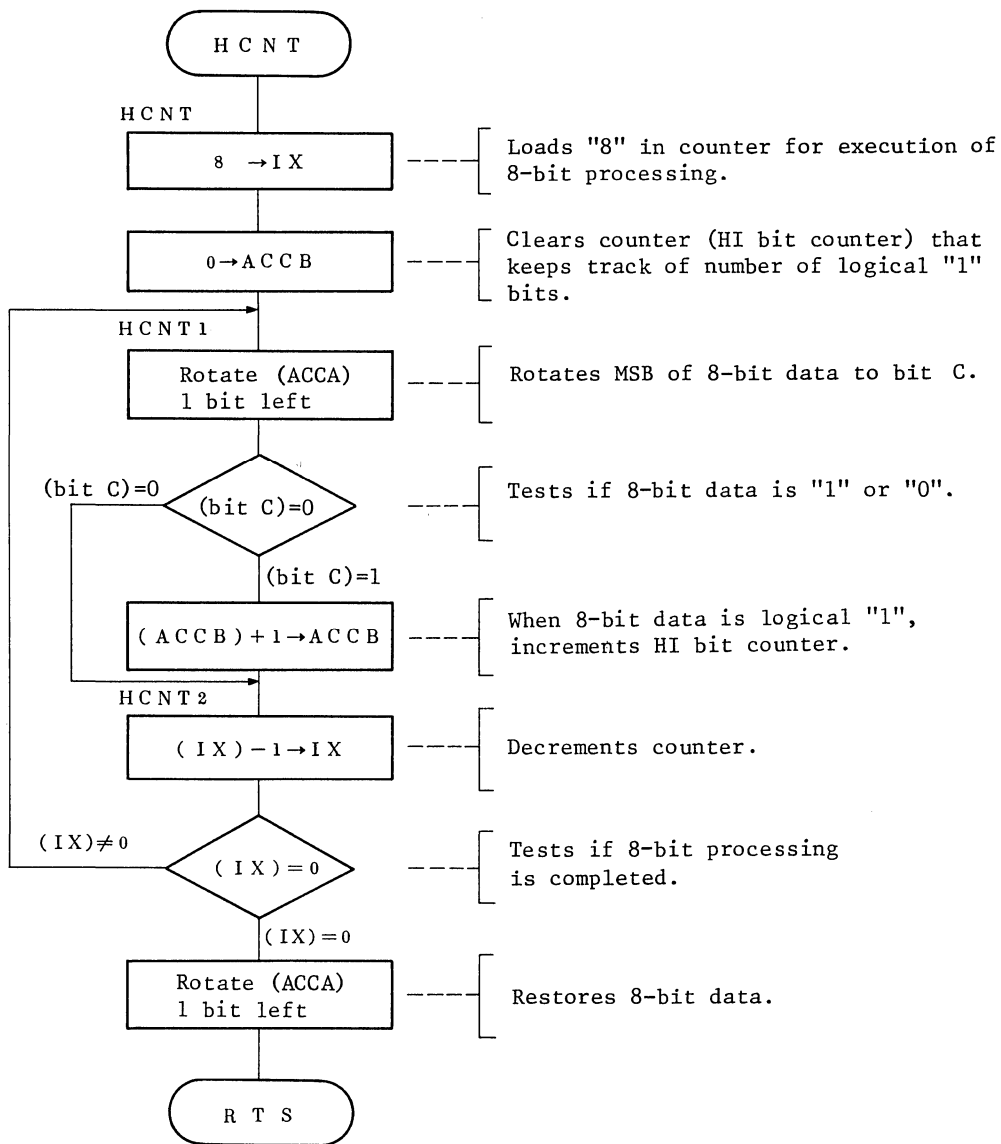
Subroutine HCNT is called after 8-bit data is held.

WORK1	RMB	1	-----	Reserves memory byte for 8-bit data.		
WORK2	RMB	1	-----	Reverse memory byte for number of logical "1" bits.		
	⋮					
	PSHX		-----	Saves register contents that will be destroyed by executing HCNT.		
	LDAA	WORK1	-----	Loads 8-bit data into entry argument (ACCA).		
	<table border="1" data-bbox="306 888 542 951"> <tr> <td>JSR</td> <td>HCNT</td> </tr> </table>	JSR	HCNT		-----	Calls subroutine HCNT.
JSR	HCNT					
				Stores number of logical "1" bits		
	STAB	WORK2	-----	(return argument (ACCB)) in RAM.		
	PULX		-----	Restores register.		
	⋮					

(5) Basic Operation

- (a) IX is used to indicate number of 8-bit data rotations.
- (b) Using rotate instruction (ROL), data in ACCA is loaded into bit C one by one.
- (c) Bit C is checked. If "1", ACCB is incremented. If "0", no operation applied.
- (d) IX is decremented each time (b) and (c) is executed.
- (e) Loops (b) to (d) until IX is "0".

FLOWCHART



8. COUNTING NUMBER OF LOGICAL  
"1" BITS IN 8-BIT DATA

MCU/MPU

HD6301/HD6303 FAMILY

LABEL

HCNT

PROGRAM LISTING

```

00001 *****
00002 *
00003 *          NAME : COUNTING NUMBER OF LOGICAL "1" *
00004 *          BITS IN 8-BIT DATA      (HCNT)      *
00005 *
00006 *****
00007 *
00008 *          ENTRY :  ACCA (8-BIT DATA) *
00009 *          RETURNS : ACCB (HIGH BIT COUNTER) *
00010 *
00011 *****
00012 *
00013A F000          DRG      $F000
00014 *
00015          F000 A HCNT EQU      *      Entry point
00016A F000 CE 0008 A      LDX      #8      Set rotate counter
00017A F003 5F          CLR      CLR      Clear high bit counter
00018A F004 49          HCNT1 ROLA     Rotate 8-bit data left
00019A F005 24 01 F008   BCC      HCNT2   Branch if carry=0
00020A F007 5C          INCB     INCB     Increment high bit counter
00021A F008 09          HCNT2 DEX      DEX      Decrement rotate counter
00022A F009 26 F9 F004   BNE      HCNT1   Loop until rotate counter=0
00023A F00B 49          ROLA     ROLA     Replace 8-bit data
00024A F00C 39          RTS      RTS

```

9. SHIFTING 32-BIT DATA	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	SHR
-------------------------	---------	----------------------	-------	-----

**FUNCTION**

(a) Shifts 32-bit binary data in IX and ACCD to right.  
 (b) Permits number of shifts to be freely determined.  
 (c) Permits easy multiplication of 32-bit binary data by  $2^{-n}$ . (n:number of shifts)

ARGUMENTS				CHANGES IN CPU REGISTERS AND FLAGS		SPECIFICATIONS																																		
Arguments	Entry	Upper 16 bits of 32-bit binary data to be shifted to right	IX	2	● : Not affected × : Undefined ↓ : Result ACCD <table border="1"> <tr><td>ACCA</td><td>ACCB</td></tr> <tr><td>↓</td><td>↓</td></tr> <tr><td>IX</td><td></td></tr> <tr><td>↓</td><td></td></tr> </table> <table border="1"> <tr><td>C</td><td>V</td></tr> <tr><td>×</td><td>×</td></tr> <tr><td>Z</td><td>N</td></tr> <tr><td>×</td><td>×</td></tr> <tr><td>I</td><td>H</td></tr> <tr><td>●</td><td>●</td></tr> </table>	ACCA	ACCB	↓	↓	IX		↓		C	V	×	×	Z	N	×	×	I	H	●	●	<table border="1"> <tr><td>ROM (Bytes)</td></tr> <tr><td>11</td></tr> <tr><td>RAM (Bytes)</td></tr> <tr><td>1</td></tr> <tr><td>Stack (Bytes)</td></tr> <tr><td>0</td></tr> <tr><td>No. of cycles</td></tr> <tr><td>261</td></tr> <tr><td>Reentrant</td></tr> <tr><td>No</td></tr> <tr><td>Relocation</td></tr> <tr><td>No</td></tr> <tr><td>Interrupt</td></tr> <tr><td>Yes</td></tr> </table>	ROM (Bytes)	11	RAM (Bytes)	1	Stack (Bytes)	0	No. of cycles	261	Reentrant	No	Relocation	No	Interrupt	Yes
		ACCA	ACCB																																					
		↓	↓																																					
	IX																																							
	↓																																							
	C	V																																						
×	×																																							
Z	N																																							
×	×																																							
I	H																																							
●	●																																							
ROM (Bytes)																																								
11																																								
RAM (Bytes)																																								
1																																								
Stack (Bytes)																																								
0																																								
No. of cycles																																								
261																																								
Reentrant																																								
No																																								
Relocation																																								
No																																								
Interrupt																																								
Yes																																								
Lower 16 bits of 32-bit binary data to be shifted to right	ACCD	2																																						
Number of shifts	SFCNTR (RAM)	1																																						
Re-returns	Upper 16 bits of shift result	IX	2																																					
	Lower 16 bits of shift result	ACCD	2																																					

**DESCRIPTION**

(1) Function Details  
 (a) Argument details  
 IX : Holds upper 16 bits of 32-bit binary data to be shifted to right.  
 After SHR execution, contains upper 16 bits of shift result.  
 ACCD : Holds lower 16 bits of 32-bit binary data to be shifted to right.  
 After SHR execution, contains lower 16 bits of shift result.

**SPECIFICATIONS NOTES**

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to shift 32-bit binary data 16 bits right.



DESCRIPTION	
-------------	--

SFCNTR: Holds number of shifts.  
(RAM)

(b) Fig. 1 shows example of SHR execution. If entry arguments are held as shown in part ① of Fig. 1, 32-bit binary data is shifted to right as shown in part ② of Fig. 1. In this case, "0" in upper 2 bits.

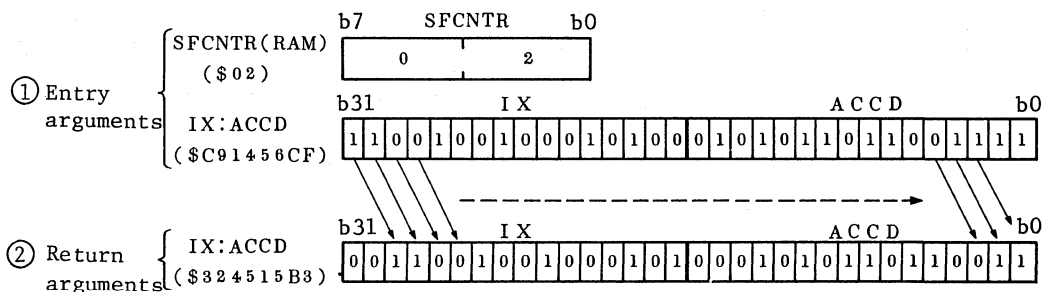


Fig. 1 Example of SHR execution

(2) User Notes

Number of shifts should be held within range of \$01 to \$1F, otherwise ACCD and IX become "0".

(3) RAM description

Label	RAM	Description			
SFCNTR	<table border="1" style="margin: auto;"> <tr> <td style="width: 20px; text-align: center;">b7</td> <td style="width: 100px; height: 30px;"></td> <td style="width: 20px; text-align: center;">b0</td> </tr> </table>	b7		b0	} Number of shifts is stored.
b7		b0			

DESCRIPTION	
-------------	--

(4) Sample Application

Subroutine SHR is called after number of shifts and 32-bit binary data to be shifted to right are held.

WORK1     RMB     4 ----- Reserves memory byte for 32-bit binary data.

WORK2     RMB     1 ----- Reserves memory byte for number of shifts.

WORK3     RMB     4 ----- Reserves memory byte for shift result.

|  
|  
|

LDAA	WORK2	}	Stores number of shifts into entry argument (SFCNTR).
STAA	SFCNTR		
LDX	WORK1	}	Loads 32-bit binary data to be shifted to right into entry argument (IX, ACCD).
LDD	WORK1+2		

JSR	SHR	----- Calls subroutine SHR.
-----	-----	-----------------------------

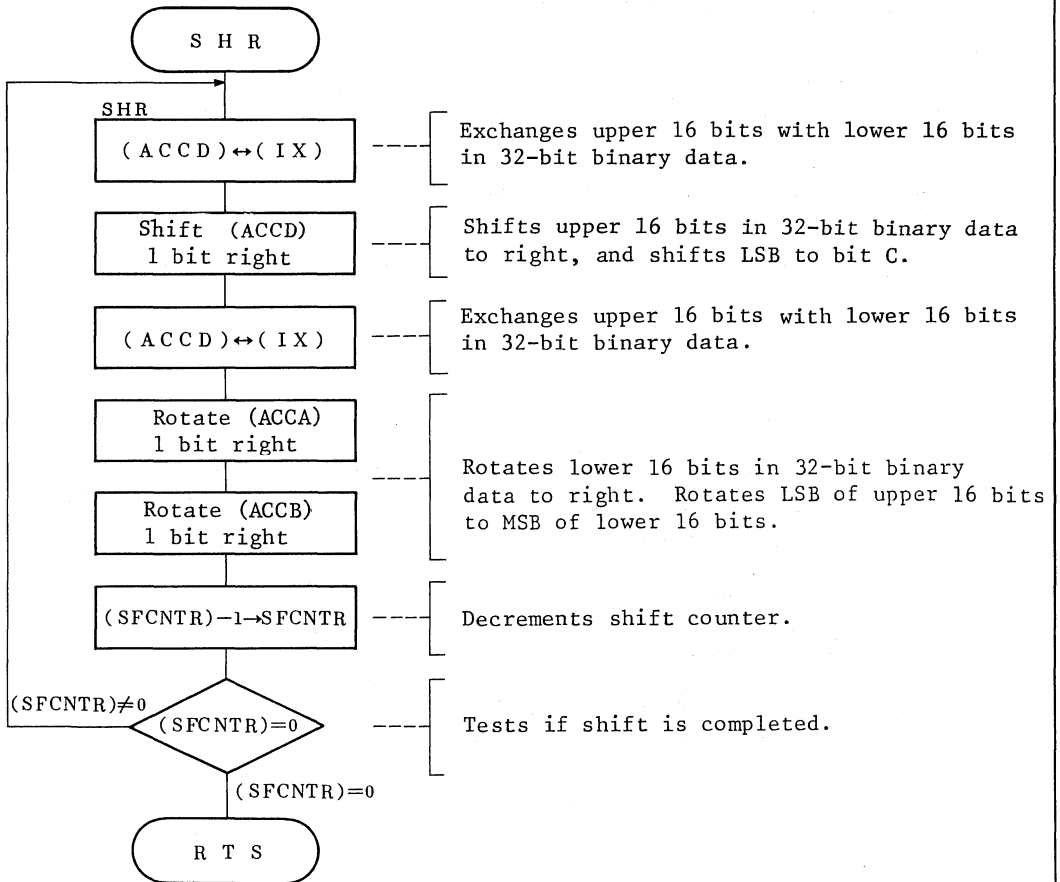
STX	WORK3	}	Stores shift result (return argument (IX, ACCD)) in RAM.
STD	WORK3+2		

|  
|  
|

(5) Basic Operation

- (a) Uses 16-bit shift instruction (LSRD) provided in the HD6301/HD6303 FAMILY.
- (b) Upper 16 bits in 32-bit binary data are shifted to right. Here LSB is rotated to bit C. Lower 16 bits are rotated to right. At this time, LSB in bit C is rotated to MSB of lower 16 bits.
- (c) SFCNTR(RAM) is used to keep track of number of shifts. SFCNTR(RAM) is decremented every time (b) is executed.
- (d) Loops (b) to (c) until SFCNTR (RAM) is "0".

FLOWCHART



PROGRAM LISTING

```

00001          *****
00002          *
00003          *   NAME :   SHIFTING 32-BIT DATA   (SHR)   *
00004          *
00005          *****
00006          *
00007          *   ENTRY : IX      (UPPER 16-BIT BINARY DATA) *
00008          *           ACCD (LOWER 16-BIT BINARY DATA) *
00009          *           SFCNTR (SHIFT COUNTER)           *
00010          *   RETURNS : IX      (UPPER 16-BIT BINARY DATA) *
00011          *           ACCD (LOWER 16-BIT BINARY DATA) *
00012          *
00013          *****
00014          *
00015A 0080          DRG      $80
00016          *
00017A 0080    0001  A  SFCNTR  RMB      1      Shift counter
00018          *
00019A F000          DRG      $F000
00020          *
00021          F000  A  SHR      EQU      *      Entry point
00022A F000 18          XGDX          Exchang 16-bit binary data
00023A F001 04          LSRD          Shift upper 16-bit binary data
00024A F002 18          XGDX          Exchang 16-bit binary data
00025A F003 46          RORA          Shift Lower 16-bit binary data
00026A F004 56          RORB
00027A F005 7A 0080  A  DEC      SFCNTR  Decrement shift counter
00028A F008 26 F6 F000 BNE      SHR      Loop until shift counter = 0
00029A F00A 39          RTS
    
```

10. 4-DIGIT BCD COUNTER	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	DECNT
-------------------------	---------	----------------------	-------	-------

**FUNCTION**

(a) Increments 4-digit BCD counter in RAM.  
 (b) Permits easy counting (external interrupts, timer interrupts and so on).

ARGUMENTS					CHANGES IN CPU REGISTERS AND FLAGS		SPECIFICATIONS												
Contents		Storage Location	Byte Lgth.		● : Not affected × : Undefined † : Result		ROM (Bytes)												
Arguments	Entry	—	—	—	<table border="1"> <tr><th colspan="2">ACCD</th></tr> <tr><th>ACCA</th><th>ACCB</th></tr> <tr><td>×</td><td>×</td></tr> <tr><th>IX</th><td></td></tr> <tr><td>×</td><td></td></tr> </table>		ACCD		ACCA	ACCB	×	×	IX		×		15		
	ACCD																		
ACCA	ACCB																		
×	×																		
IX																			
×																			
	Re-returns	4-digit BCD counter	DCNTR (RAM)	2	<table border="1"> <tr><th>C</th><th>V</th></tr> <tr><td>†</td><td>×</td></tr> <tr><th>Z</th><th>N</th></tr> <tr><td>×</td><td>×</td></tr> <tr><th>I</th><th>H</th></tr> <tr><td>●</td><td>×</td></tr> </table>		C	V	†	×	Z	N	×	×	I	H	●	×	RAM (Bytes)
C	V																		
†	×																		
Z	N																		
×	×																		
I	H																		
●	×																		
		Counter overflow or not	bit C (CCR)	1			2												
							Stack (Bytes)												
							0												
							No. of cycles												
							41												
							Reentrant												
							No												
							Relocation												
							No												
							Interrupt												
							Yes												

**DESCRIPTION**

(1) Function Details

(a) Argument details

DCNTR: Used as 4-digit BCD counter which (RAM) counts up every DECNT execution.

bit C: Indicates counter status after (CCR) DECNT execution.

bit C=1: Shows counter overflow.  
(See Fig. 2).

bit C=0: Shows counter not-overflow.

(b) Fig. 1 shows example of DECNT execution. When DECNT is executed, 4-digit BCD counter completes counting up as shown in part ② of Fig. 1.

Fig. 1 Example of DECNT execution

① Before execution { DCNTR(RAM) (4099) }  $\begin{matrix} \text{b15} & \text{DCNTR} & \text{DCNTR} & \text{DCNTR} & \text{b0} \\ \hline & 4 & 0 & 9 & 9 \end{matrix}$

② Return arguments { DCNTR(RAM) (4100) }  $\begin{matrix} \text{bit C} & \text{b15} & \text{DCNTR} & \text{DCNTR} & \text{b0} \\ \hline 0 & 4 & 1 & 0 & 0 \end{matrix}$

Fig. 2 Example of counter overflow

① Before execution { DCNTR(RAM) (9999) }  $\begin{matrix} & 9 & 9 & 9 & 9 \\ \hline & & & & \end{matrix}$

② Return arguments { DCNTR(RAM) (0000) }  $\begin{matrix} \text{bit C} & & & & \\ \hline 1 & 0 & 0 & 0 & 0 \end{matrix}$

**SPECIFICATIONS NOTES**

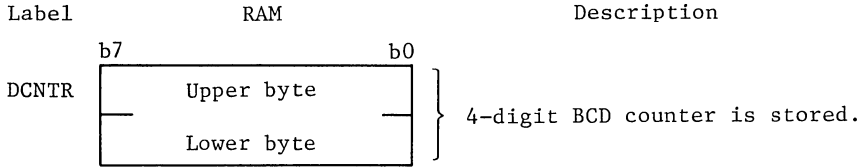


DESCRIPTION

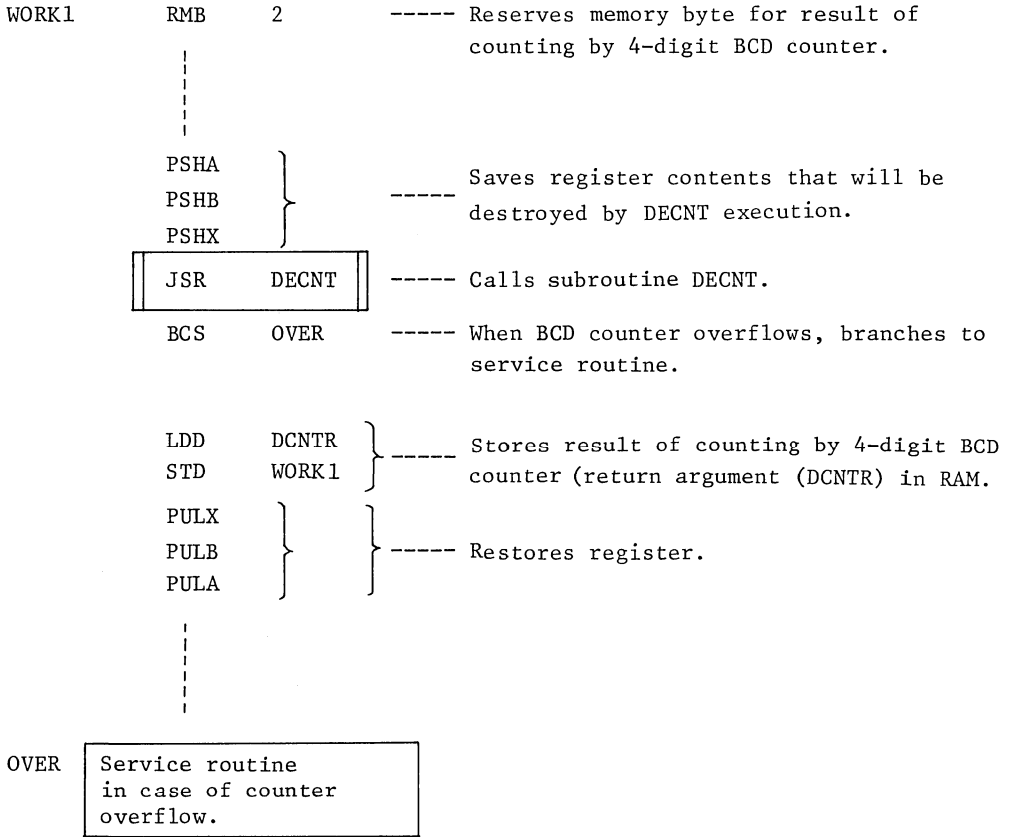
(2) User Notes

If counter overflows as shown in Fig. 2, counter is cleared.

(3) RAM Description



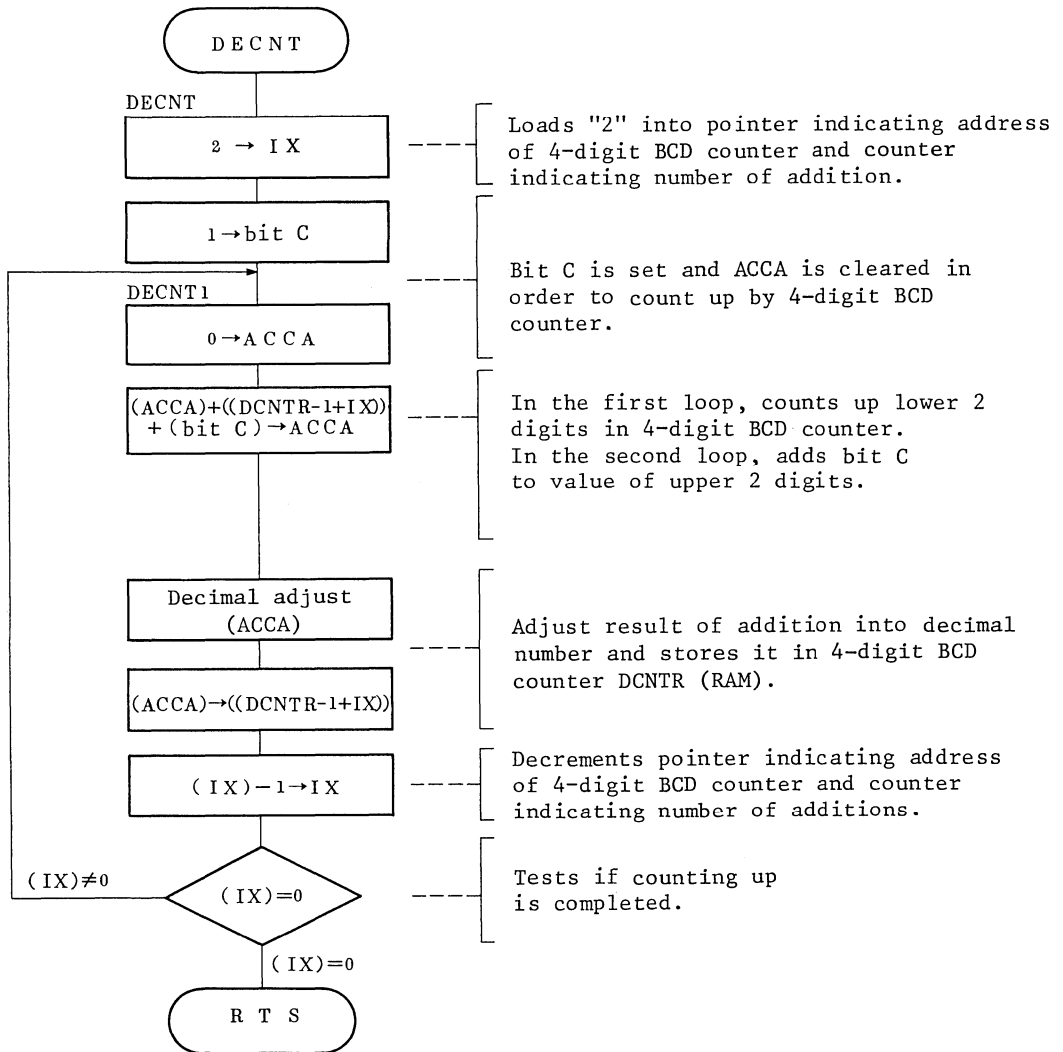
(4) Sample Application



7

10. 4-DIGIT BCD COUNTER	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	DECNT
<p>DESCRIPTION</p> <p>(5) Basic Operation</p> <p>(a) IX is used to indicate address of BCD counter and is also used to keep track of number of addition.</p> <p>(b) Set bit C for counting "1"s.</p> <p>(c) Executes (Formula 1) using index addressing mode.            (Bit C is set at the first a-dition. When a carry is generated after (Formula 1) execution, bit C is also set.)</p> $0 + ((DCNTR - 1 + IX)) + (\text{bit C}) \rightarrow \text{ACCA} \text{ ----- (Formula 1)}$ <p>(d) Decrements IX.</p> <p>(e) Loops addition of upper byte and bit C until IX is "0".</p>				

## FLOWCHART





PROGRAM LISTING
-----------------

```

00001 *****
00002 *
00003 *   NAME : 4-DIGIT BCD COUNTER   (DECNT)   *
00004 *
00005 *****
00006 *
00007 *   ENTRY  :  NOTHING                *
00008 *   RETURNS :  DCNTR (BCD COUNTER)   *
00009 *           CARRY (C=0:TRUE,C=1:OVER FLOW) *
00010 *
00011 *****
00012 *
00013A 0080          ORG   $80
00014 *
00015A 0080    0002  A  DCNTR  RMB   2          BCD counter
00016 *
00017A F000          ORG   $F000
00018 *
00019          F000  A  DECNT  EQU   *          Entry point
00020A F000 CE 0002  A          LDX   #2          Load ADDR pointer (addition counter)
00021A F003 0D          SEC          Set carry bit
00022A F004 86 00    A  DECNT1 LDAA  #0          Clear ACCA
00023A F006 A9 7F    A          ADCA  DCNTR-1,X Increment BCD counter
00024A F00B 19          DAA          Convert into BCD
00025A F009 A7 7F    A          STAA  DCNTR-1,X Store BCD counter
00026A F00B 09          DEX          Decrement ADDR pointer
00027A F00C 26 F6 F004 BNE   DECNT1  Loop until ADDR pointer = 0
00028A F00E 39          RTS

```

11. COMPARING 32-BIT BINARY DATA	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	CMP
----------------------------------	---------	----------------------	-------	-----

**FUNCTION**

(a) Determines larger than / smaller than relationship (>,<=,<) of 32-bit binary data of 2 groups, and loads result into bit C and bit Z of CCR.

(b) Utilizes unsigned integers in arguments.

ARGUMENTS				CHANGES IN CPU REGISTERS AND FLAGS		SPECIFICATIONS	
Contents		Storage Location	Byte Lgth.	● : Not affected × : Undefined ↑ : Result	ROM (Bytes)		9
					RAM (Bytes)		4
Arguments	Entry	Upper 16 bits of First value	IX	2	Stack (Bytes)		0
		Lower 16 bits of First value	ACCD	2	No. of cycles		20
	Second value	CMT (RAM)	4	Reentrant		No	Relocation
Re- turns	Compa- rison result	bit C bit Z (CCR)	1	Interrupt		Yes	

**DESCRIPTION**

(1) Function Details

(a) Argument details

IX : Holds upper 16 bits of the first 32-bit binary value.

ACCD: Holds lower 16 bits of the first 32-bit binary value.

CMT : Holds the second 32-bit binary value.  
(RAM)

bit C, bit Z: Bit C and bit Z of CCR are contains according to comparison result. (CCR)

(b) Table 1 shows example of CMP execution.

If entry arguments are as shown in Table 1, bit C and bit Z of CCR are set accordingly.

**SPECIFICATIONS NOTES**

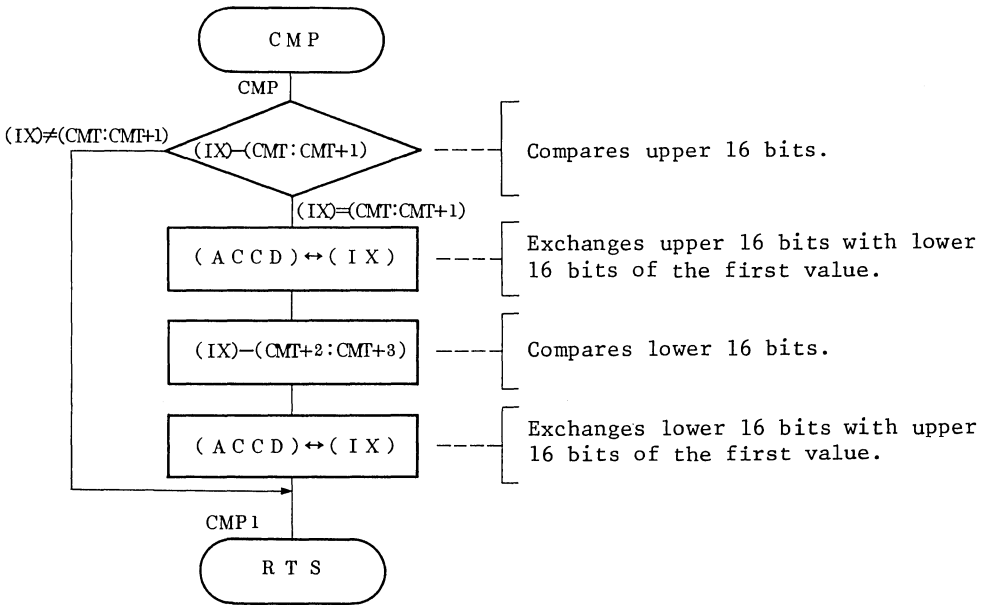
"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed when comparand and comparative number are equal.



11. COMPARING 32-BIT BINARY DATA	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	CMP
DESCRIPTION				
(4) Sample Application				
Subroutine CMP is called after the first value and the second value are held.				
WORK1	RMB	4	-----	Reserves memory byte for the first 32-bit binary value.
WORK2	RMB	4	-----	Reserves memory byte for the second 32-bit binary value.

11. COMPARING 32-BIT BINARY DATA	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	CMP		
<table border="1"> <thead> <tr> <th data-bbox="104 118 387 157">DESCRIPTION</th> </tr> </thead> <tbody> <tr> <td data-bbox="104 157 1224 1593"> <p>(5) Basic Operation</p> <p>(a) Uses 16-bit comparison instruction (CPX) comparing IX with 2-byte memory, provided in the HD6301/HD6303 FAMILY.</p> <p>(b) Bit C and bit Z of CCR are determined as return argument, after 16-bit comparison instruction (CPX) is executed.</p> <p>(c) Upper 16 bits are compared using 16-bit comparison instruction (CPX). When equal, lower 16 bits are compared. When not equal, subroutine CMP is terminated.</p> </td> </tr> </tbody> </table>					DESCRIPTION	<p>(5) Basic Operation</p> <p>(a) Uses 16-bit comparison instruction (CPX) comparing IX with 2-byte memory, provided in the HD6301/HD6303 FAMILY.</p> <p>(b) Bit C and bit Z of CCR are determined as return argument, after 16-bit comparison instruction (CPX) is executed.</p> <p>(c) Upper 16 bits are compared using 16-bit comparison instruction (CPX). When equal, lower 16 bits are compared. When not equal, subroutine CMP is terminated.</p>
DESCRIPTION						
<p>(5) Basic Operation</p> <p>(a) Uses 16-bit comparison instruction (CPX) comparing IX with 2-byte memory, provided in the HD6301/HD6303 FAMILY.</p> <p>(b) Bit C and bit Z of CCR are determined as return argument, after 16-bit comparison instruction (CPX) is executed.</p> <p>(c) Upper 16 bits are compared using 16-bit comparison instruction (CPX). When equal, lower 16 bits are compared. When not equal, subroutine CMP is terminated.</p>						

FLOWCHART



PROGRAM LISTING
-----------------

```

00001 *****
00002 *
00003 *   NAME : COMPARING 32-BIT BINARY DATA (CMP) *
00004 *
00005 *****
00006 *
00007 *   ENTRY : IX  (UPPER 16-BIT COMPARAND) *
00008 *           ACCD (LOWER 16-BIT COMPARAND) *
00009 *           CMT  (32-BIT COMPARATIVE NUMBER) *
00010 *   RETURNS : CARRY & BIT 2 (COMPARISON RESULT) *
00011 *
00012 *****
00013 *
00014A 0080          ORG    $80
00015
00016A 0080    0004  A CMT   RMB    4          Comparative number
00017
00018A F000          ORG    $F000
00019
00020          F000  A CMP   EQU    *          Entry point
00021A F000 9C 80    A      CPX   CMT      Compare IX with CMT+2:CMT+3
00022A F002 26 04 F008    BNE   CMP1     Branch if IX equal CMT+2:CMT+3
00023A F004 18          XGDX          Exchange ACCD & IX
00024A F005 9C 82    A      CPX   CMT+2   Compare IX with CMT:CMT+1
00025A F007 18          XGDX
00026A F008 39          CMP1   RTS

```

12. ADDING 32-BIT BINARY DATA	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	ADD
-------------------------------	---------	----------------------	-------	-----

**FUNCTION**

(a) Performs addition of 32-bit binary number and loads addition result into IX and ACCD.

(b) Utilizes unsigned integers in arguments.

ARGUMENTS				CHANGES IN CPU REGISTERS AND FLAGS		SPECIFICATIONS																									
Contents		Storage Location	Byte Lgth.	● : Not affected																											
				× : Undefined																											
				↑ : Result																											
Arguments	Entry	Upper 16 bits of augend	IX	2	<table border="1"> <tr><td colspan="2">ACCD</td></tr> <tr><td>ACCA</td><td>ACCB</td></tr> <tr><td>↑</td><td>↓</td></tr> <tr><td>IX</td><td></td></tr> <tr><td>↑</td><td></td></tr> <tr><td colspan="2"> </td></tr> <tr><td>C</td><td>V</td></tr> <tr><td>↑</td><td>×</td></tr> <tr><td>Z</td><td>N</td></tr> <tr><td>×</td><td>×</td></tr> <tr><td>I</td><td>H</td></tr> <tr><td>●</td><td>×</td></tr> </table>			ACCD		ACCA	ACCB	↑	↓	IX		↑				C	V	↑	×	Z	N	×	×	I	H	●	×
		ACCD																													
		ACCA	ACCB																												
	↑	↓																													
	IX																														
	↑																														
C	V																														
↑	×																														
Z	N																														
×	×																														
I	H																														
●	×																														
Lower 16 bits of augend	ACCD	2																													
Addend	ADER (RAM)	4																													
Re-returns	Upper 16 bits of addition result	IX	2																												
	Lower 16 bits of addition result	ACCD	2																												
	Carry or no carry	Bit C (CCR)	1																												
				ROM (Bytes)	9																										
				RAM (Bytes)	4																										
				Stack (Bytes)	0																										
				No. of cycles	19																										
				Reentrant	No																										
				Relocation	No																										
				Interrupt	Yes																										

**DESCRIPTION**

(1) Function Details

(a) Argument details

IX : Holds upper 16 bits of augend. After ADD execution, contains upper 16 bits of addition result.

ACCD: Holds lower 16 bits of augend. After ADD execution, contains lower 16 bits of addition result.

ADER: Holds 32-bit binary addend. (RAM)

**SPECIFICATIONS NOTES**

7



DESCRIPTION

bit C (CCR) : Indicates whether carry is generated or not after ADD execution.

bit C = 1 : Carry is generated in addition result.

(see Fig. 2)

bit C = 0 : No carry is generated in addition result.

(b) Fig. 1 shows example of ADD execution. If entry arguments are as shown in part ① of Fig. 1, addition result is contained in IX and ACCD as shown in part ② of Fig. 1.

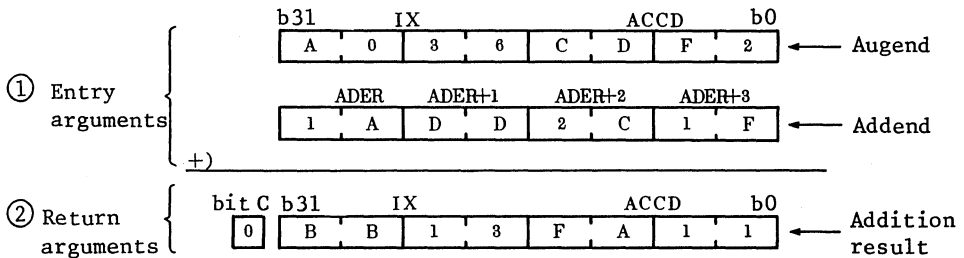


Fig. 1 Example of ADD execution

(2) User Notes

(a) As shown in Fig. 3, when not using upper byte, the upper byte should be held to "0", otherwise addition result will not be correct, because addition is performed with undefined data in the upper byte.

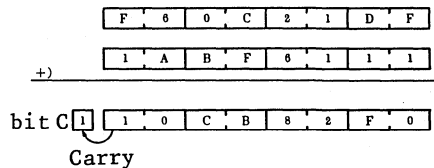


Fig. 2 Example of addition when carry is generated.

(b) After ADD execution, augend is destroyed because addition result is contained in IX and ACCD. If augend needs to be retained after ADD execution, it should be saved in memory before execution.

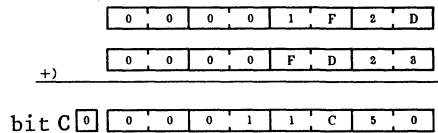
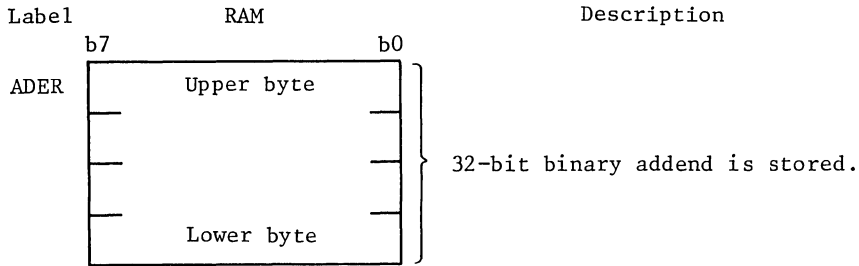


Fig. 3 Example of addition when upper byte is not used.

DESCRIPTION

(3) RAM Description



(4) Sample Application

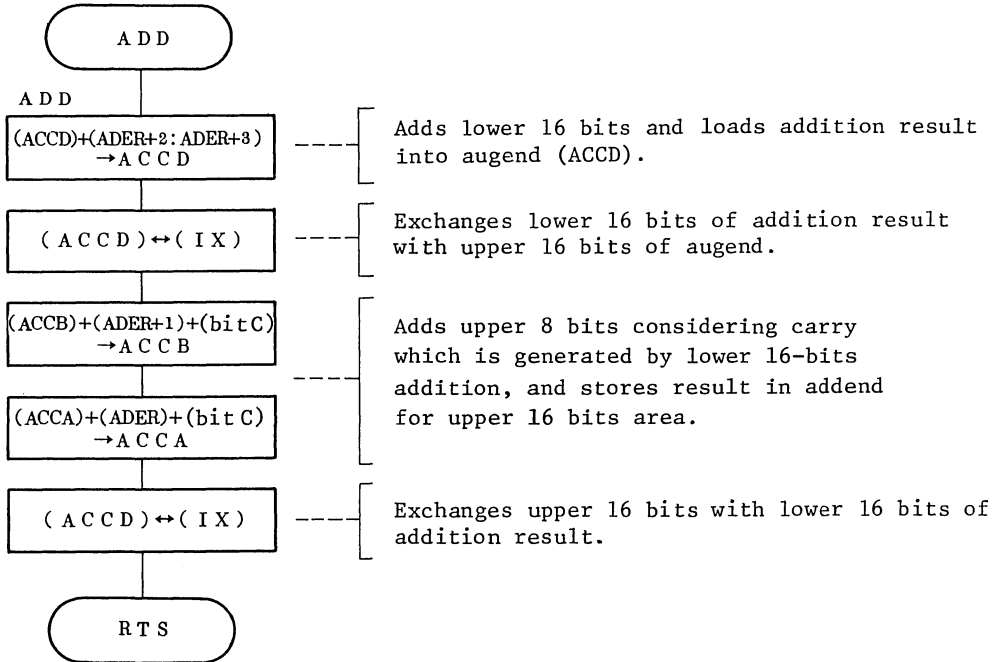
Subroutine ADD is called after augend and addend are held.

WORK1	RMB	4	-----	Reserves memory byte for 32-bit binary augend.
WORK2	RMB	4	-----	Reserves memory byte for 32-bit binary addend.
WORK3	RMB	4	-----	Reserves memory byte for 32-bit binary addition result.
	LDD	WORK2	}	Stores 32-bit binary addend into entry argument (ADDER).
	STD	ADDER		
	LDD	WORK2+2		
	STD	ADDER+2	}	
	LDD	WORK1	}	Loads 32-bit binary augend into entry argument (IX, ACCD).
	LDD	WORK1+2		
	JSR ADD		-----	Calls subroutine ADD.
	BCS	OVER	-----	If carry is generated in addition result, branches to service routine.
	STX	WORK3	}	Stores addition result (return arguments (IX,ACCD)) in RAM.
	STD	WORK3+2		
OVER	Service routine in case of carry			

7

12. ADDING 32-BIT BINARY DATA	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	ADD		
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%; padding: 5px;">DESCRIPTION</td> <td style="padding: 5px;"> <p>(5) Basic Operation</p> <p>(a) Uses 16-bit addition instruction (ADDD) provided in the HD6301/HD6303 FAMILY.</p> <p>(b) (Formula 1) shows addition of lower 16 bits using 16-bit addition instruction (ADDD). When carry is generated after performing (Formula 1), bit C is set.</p> <math display="block">(ACCD) + (ADER+2:ADER+3) \rightarrow ACCD \text{ ----- (Formula 1)}</math> <p>(c) Upper 16 bits are added using 8-bit addition instruction shown in (Formula 2) and (Formula 3) (ADC) considering bit C.</p> <math display="block">(ACCB) + (ADER+1) + (\text{bit C}) \rightarrow ACCB \text{ ----- (Formula 2)}</math> <math display="block">(ACCA) + (ADER) + (\text{bit C}) \rightarrow ACCA \text{ ----- (Formula 3)}</math> <p>Bit C is taken into consideration because there is carry involved with the addition result by executing (b).</p> </td> </tr> </table>					DESCRIPTION	<p>(5) Basic Operation</p> <p>(a) Uses 16-bit addition instruction (ADDD) provided in the HD6301/HD6303 FAMILY.</p> <p>(b) (Formula 1) shows addition of lower 16 bits using 16-bit addition instruction (ADDD). When carry is generated after performing (Formula 1), bit C is set.</p> $(ACCD) + (ADER+2:ADER+3) \rightarrow ACCD \text{ ----- (Formula 1)}$ <p>(c) Upper 16 bits are added using 8-bit addition instruction shown in (Formula 2) and (Formula 3) (ADC) considering bit C.</p> $(ACCB) + (ADER+1) + (\text{bit C}) \rightarrow ACCB \text{ ----- (Formula 2)}$ $(ACCA) + (ADER) + (\text{bit C}) \rightarrow ACCA \text{ ----- (Formula 3)}$ <p>Bit C is taken into consideration because there is carry involved with the addition result by executing (b).</p>
DESCRIPTION	<p>(5) Basic Operation</p> <p>(a) Uses 16-bit addition instruction (ADDD) provided in the HD6301/HD6303 FAMILY.</p> <p>(b) (Formula 1) shows addition of lower 16 bits using 16-bit addition instruction (ADDD). When carry is generated after performing (Formula 1), bit C is set.</p> $(ACCD) + (ADER+2:ADER+3) \rightarrow ACCD \text{ ----- (Formula 1)}$ <p>(c) Upper 16 bits are added using 8-bit addition instruction shown in (Formula 2) and (Formula 3) (ADC) considering bit C.</p> $(ACCB) + (ADER+1) + (\text{bit C}) \rightarrow ACCB \text{ ----- (Formula 2)}$ $(ACCA) + (ADER) + (\text{bit C}) \rightarrow ACCA \text{ ----- (Formula 3)}$ <p>Bit C is taken into consideration because there is carry involved with the addition result by executing (b).</p>					

## FLOWCHART



PROGRAM LISTING

```

00001 *****
00002 *
00003 *   NAME : ADDING 32-BIT BINARY DATA (ADD) *
00004 *
00005 *****
00006 *
00007 *   ENTRY : IX (UPPER 16-BIT AUGEND) *
00008 *   ACCD (LOWER 16-BIT AUGEND) *
00009 *   ADER (32-BIT ADDEND) *
00010 *   RETURNS : IX (UPPER 16-BIT SUM) *
00011 *   ACCD (LOWER 16-BIT SUM) *
00012 *   CARRY(C=0;TRUE.C=1;OVER FLOW) *
00013 *
00014 *****
00015 *
00016A 0080   ORG   $80
00017 *
00018A 0080   0004 A ADER RMB 4   Addend
00019 *
00020A F000   ORG   $F000
00021 *
00022   F000 A ADD EQU *   Entry point
00023A F000 D3 82 A ADDD ADER+2 Add ACCD and ADER+2:ADER+3
00024A F002 18 XGDX Exchange ACCD & IX
00025A F003 D9 81 A ADCB ADER+1 Add ACCB and ADER+1
00026A F005 99 80 A ADCA ADER Add ACCA and ADER
00027A F007 18 XGDX Exchange ACCD & IX
00028A F008 39 RTS

```

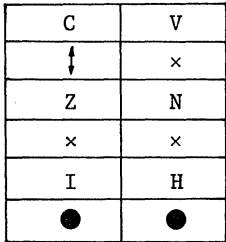
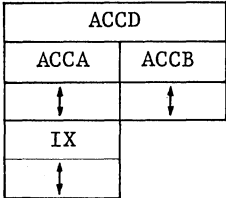
13. SUBTRACTING 32-BIT BINARY DATA	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	SUB
------------------------------------	---------	----------------------	-------	-----

**FUNCTION**

(a) Performs subtraction of 32-bit binary data and loads subtraction result into IX and ACCD.

(b) Utilizes unsigned integers in arguments.

ARGUMENTS				CHANGES IN CPU REGISTERS AND FLAGS		SPECIFICATIONS	
Arguments	Entry	Upper 16 bits of minuend	IX	2	● : Not affected × : Undefined † : Result	ROM (Bytes)	
		Lower 16 bits of minuend	ACCD	2		9	
		Subtrahend	SBER (RAM)	4		RAM (Bytes)	
	Re-returns	Upper 16 bits of subtraction result	IX	2		4	
		Lower 16 bits of subtraction result	ACCD	2		Stack (Bytes)	
		Borrow or no borrow	bit C (CCR)	1		0	
						No. of cycles	
				19			
				Reentrant			
				No			
				Relocation			
				No			
				Interrupt			
				Yes			



**DESCRIPTION**

(1) Function Details

(a) Argument details

IX : Holds upper 16 bits of minuend. Contains upper 16 bits of subtraction result after SUB execution.

ACCD: Holds lower 16 bits of minuend. Contains lower 16 bits of subtraction result after SUB execution.

SBER: Holds 32-bit binary subtrahend. (RAM)

**SPECIFICATIONS NOTES**

**DESCRIPTION**

bit C (CCR): Indicates whether borrow is generated or not after SUB execution.  
 bit C = 1 : Borrow is generated in subtraction result.  
 (see Fig. 2)  
 bit C = 0 : No borrow is generated in subtraction result.

(b) Fig. 1 shows example of SUB execution. If entry arguments are as shown in part ① of Fig. 1, subtraction result is contained in IX and ACCD as shown in part ② of Fig. 1.

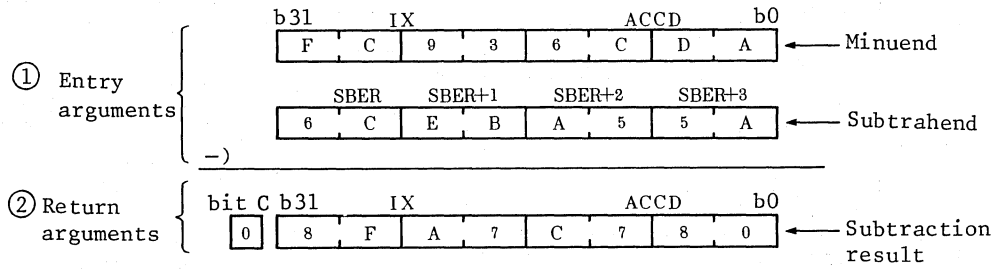


Fig. 1 Example of SUB execution

(2) User Notes

(a) When subtraction result is negative, (Minuend < Subtrahend), the result is 2's complement.

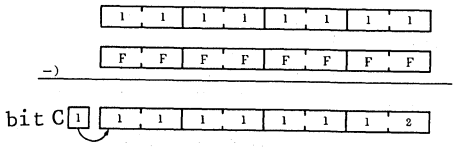


Fig. 2 Example of subtraction when borrow is generated

(b) As shown in Fig. 3, when not using upper byte, the upper byte should be held to "0", otherwise subtraction result will not be correct, because subtraction is performed with undefined data in the upper byte.

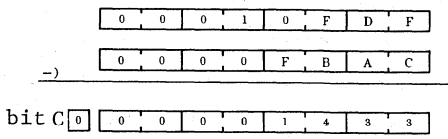
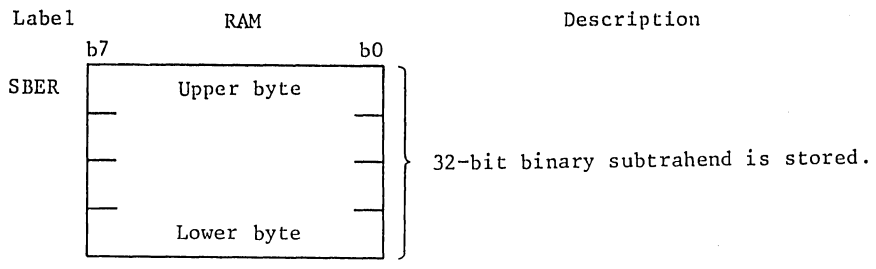


Fig. 3 Example of subtraction when upper byte is not used

(c) After SUB execution, minuend is destroyed because subtraction result is contained into IX and ACCD. If minuend needs to be retained after SUB execution, it should be saved in memory before execution.

DESCRIPTION

(3) RAM Description



(4) Sample Application

Subroutine SUB is called after minuend and subtrahend are loaded.

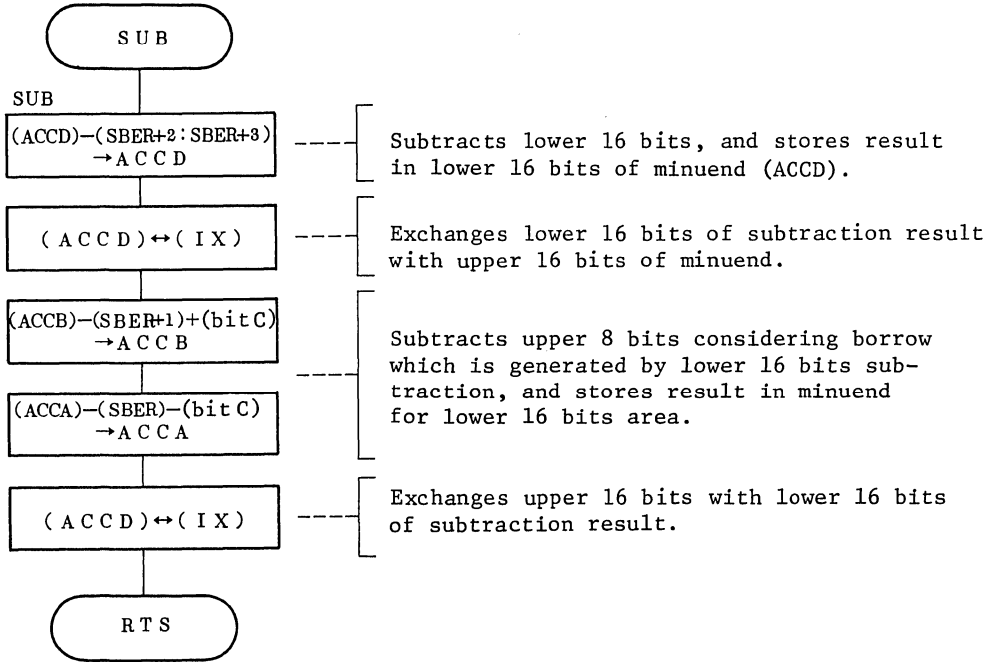
```

WORK1   RMB   4   ----- Reserves memory byte for 32-bit binary
                          minuend.
WORK2   RMB   4   ----- Reserves memory byte for 32-bit binary
                          subtrahend.
WORK3   RMB   4   ----- Reserves memory byte for 32-bit binary
                          subtraction result.
      .
      .
      .
      LDD   WORK2   }
      STD   SBER    } ----- Stores 32-bit binary subtrahend into
      LDD   WORK2+2 } entry argument (SBER).
      STD   SBER+2 }
      LDX   WORK1   } ----- Loads 32-bit binary minuend into
      LDD   WORK1+2 } entry argument (IX, ACCD).
      JSR   SUB     } ----- Calls subroutine SUB.
      BCS   OVER    } ----- If borrow is generated in subtraction
                          result, branches to service routine.
      .
      .
      .
      STX   WORK3   } ----- Stores subtraction result
      STD   WORK3+2 } return arguments (IX, ACCD) in RAM.
      .
      .
      .
OVER    [ Service routine
        in case of borrow ]
      .
      .
      .
  
```



13. SUBTRACTING 32-BIT BINARY DATA	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	SUB
<p>DESCRIPTION</p> <p>(5) Basic Operation</p> <p>(a) Uses 16-bit subtraction instruction (SUBD) provided in the HD6301/HD6303 FAMILY.</p> <p>(b) (Formula 1) shows subtraction of lower 16 bits using 16-bit subtraction instruction (SUBD). When borrow is generated after performing (Formula 1), bit C is set.</p> $(ACCD) - (SBER+2:SBER+3) \rightarrow ACCD \text{ ----- (Formula 1)}$ <p>(c) Upper 16 bits are subtracted using 8-bit subtraction instruction shown in (Formula 2), (Formula 3) (SBC) considering bit C.</p> $(ACCB) - (SBER+1) - (\text{bit C}) \rightarrow ACCB \text{ ----- (Formula 2)}$ $(ACCA) - (SBER) - (\text{bit C}) \rightarrow ACCA \text{ ----- (Formula 3)}$ <p>Bit C is taken into consideration because there is borrow involved with the subtraction result by executing (b).</p>				

FLOWCHART





14. MULTIPLYING 16-BIT BINARY DATA	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	MUL
------------------------------------	---------	----------------------	-------	-----

**FUNCTION**

(a) Performs multiplication of 16-bit binary data in RAM and stores result in 32-bit binary in RAM.

(b) Utilizes 16-bit unsigned integers in arguments.

<b>ARGUMENTS</b>		<b>CHANGES IN CPU REGISTERS AND FLAGS</b>		<b>SPECIFICATIONS</b>																																																						
<table border="1"> <thead> <tr> <th colspan="2">Contents</th> <th>Storage Location</th> <th>Byte Lgth.</th> </tr> </thead> <tbody> <tr> <td rowspan="2">Arguments</td> <td>Entry</td> <td> <table border="1"> <tr> <td>Multi- plicand</td> <td>MCAND (RAM)</td> <td>2</td> </tr> <tr> <td>Multi- plier</td> <td>MER (RAM)</td> <td>2</td> </tr> </table> </td> <td></td> </tr> <tr> <td>Re- turns</td> <td>Product</td> <td>PRDCT (RAM)</td> <td>4</td> </tr> </tbody> </table>		Contents		Storage Location	Byte Lgth.	Arguments	Entry	<table border="1"> <tr> <td>Multi- plicand</td> <td>MCAND (RAM)</td> <td>2</td> </tr> <tr> <td>Multi- plier</td> <td>MER (RAM)</td> <td>2</td> </tr> </table>	Multi- plicand	MCAND (RAM)	2	Multi- plier	MER (RAM)	2		Re- turns	Product	PRDCT (RAM)	4	<p>● : Not affected          × : Undefined          ↑ : Result</p> <table border="1"> <tr> <th colspan="2">ACCD</th> </tr> <tr> <td>ACCA</td> <td>ACCB</td> </tr> <tr> <td>×</td> <td>×</td> </tr> <tr> <td>IX</td> <td></td> </tr> <tr> <td>●</td> <td></td> </tr> </table> <table border="1"> <tr> <td>C</td> <td>V</td> </tr> <tr> <td>×</td> <td>×</td> </tr> <tr> <td>Z</td> <td>N</td> </tr> <tr> <td>×</td> <td>×</td> </tr> <tr> <td>I</td> <td>H</td> </tr> <tr> <td>●</td> <td>●</td> </tr> </table>		ACCD		ACCA	ACCB	×	×	IX		●		C	V	×	×	Z	N	×	×	I	H	●	●	<table border="1"> <tr> <td>ROM (Bytes)</td> <td>42</td> </tr> <tr> <td>RAM (Bytes)</td> <td>8</td> </tr> <tr> <td>Stack (Bytes)</td> <td>0</td> </tr> <tr> <td>No. of cycles</td> <td>97</td> </tr> <tr> <td>Reentrant</td> <td>No</td> </tr> <tr> <td>Relocation</td> <td>No</td> </tr> <tr> <td>Interrupt</td> <td>Yes</td> </tr> </table>	ROM (Bytes)	42	RAM (Bytes)	8	Stack (Bytes)	0	No. of cycles	97	Reentrant	No	Relocation	No	Interrupt	Yes
Contents		Storage Location	Byte Lgth.																																																							
Arguments	Entry	<table border="1"> <tr> <td>Multi- plicand</td> <td>MCAND (RAM)</td> <td>2</td> </tr> <tr> <td>Multi- plier</td> <td>MER (RAM)</td> <td>2</td> </tr> </table>	Multi- plicand	MCAND (RAM)	2	Multi- plier	MER (RAM)	2																																																		
	Multi- plicand	MCAND (RAM)	2																																																							
Multi- plier	MER (RAM)	2																																																								
Re- turns	Product	PRDCT (RAM)	4																																																							
ACCD																																																										
ACCA	ACCB																																																									
×	×																																																									
IX																																																										
●																																																										
C	V																																																									
×	×																																																									
Z	N																																																									
×	×																																																									
I	H																																																									
●	●																																																									
ROM (Bytes)	42																																																									
RAM (Bytes)	8																																																									
Stack (Bytes)	0																																																									
No. of cycles	97																																																									
Reentrant	No																																																									
Relocation	No																																																									
Interrupt	Yes																																																									

**DESCRIPTION**

(1) Function Details

(a) Argument details

MCAND (RAM): Holds 16-bit binary multiplicand.

MER (RAM): Holds 16-bit binary multiplier.

PRDCT (RAM): Contains 32-bit binary product.

**SPECIFICATIONS NOTES**

7

DESCRIPTION

(b) Fig. 1 shows example of MUL execution.  
 If entry arguments are as shown in part ① of Fig. 1, Product is contained in PRDCT (RAM) as shown in part ② of Fig. 1.  
 (c) Table 1 shows result when "0" is held as entry arguments.

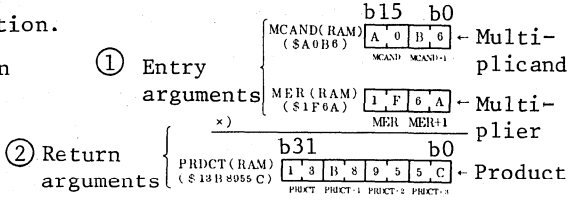


Fig. 1 Example of MUL execution

Table 1. Product when holding "0" as entry arguments

Entry arguments		Return argument
Multiplicand (MCAND:MCAND+1)	Multiplier (MER:MER+1)	Product (PRDCT:PRDCT+1:PRDCT+2:PRDCT+3)
\$ * * * * * 1	\$ 0 0 0 0	\$ 0 0 0 0 0 0 0 0
\$ 0 0 0 0	\$ * * * * * 1	\$ 0 0 0 0 0 0 0 0
\$ 0 0 0 0	\$ 0 0 0 0	\$ 0 0 0 0 0 0 0 0

(NOTE) \*1 \$\*\*\*\* indicates hexdecimals

(2) User Notes

(a) As shown in Fig. 2, when not using upper byte, the upper byte should be held to "0", otherwise product will not be correct, because multiplication is performed with undefined data in the upper byte.

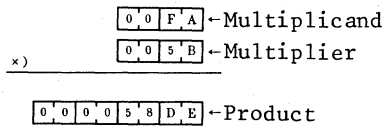


Fig. 2 Multiplication example when upper byte is not used

14. MULTIPLYING 16-BIT BINARY DATA	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	MUL					
DESCRIPTION									
(3) RAM Description									
Label	RAM		Description						
	b7	b0							
MCAND	<table border="1"> <tr><td>Upper byte</td></tr> <tr><td>Lower byte</td></tr> </table>		Upper byte	Lower byte	} 16-bit binary multiplicand is stored.				
Upper byte									
Lower byte									
MER	<table border="1"> <tr><td>Upper byte</td></tr> <tr><td>Lower byte</td></tr> </table>		Upper byte	Lower byte	} 16-bit binary multiplier is stored.				
Upper byte									
Lower byte									
PRDCT	<table border="1"> <tr><td>Upper byte</td></tr> <tr><td> </td></tr> <tr><td> </td></tr> <tr><td> </td></tr> <tr><td>Lower byte</td></tr> </table>		Upper byte				Lower byte	} 32-bit binary product is stored.	
Upper byte									
Lower byte									



14. MULTIPLYING 16-BIT BINARY DATA	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	MUL
------------------------------------	---------	----------------------	-------	-----

DESCRIPTION

(5) Basic Operation

- (a) Uses 8-bit binary multiplication instruction (MUL) provided in the HD6301/HD6303 FAMILY.
- (b) Multiplication of 16-bit binary data is executed by obtaining partial products (as shown in Fig. 3 ①, ②, ③ and ④), and adding them to product (Fig. 3 ⑤).

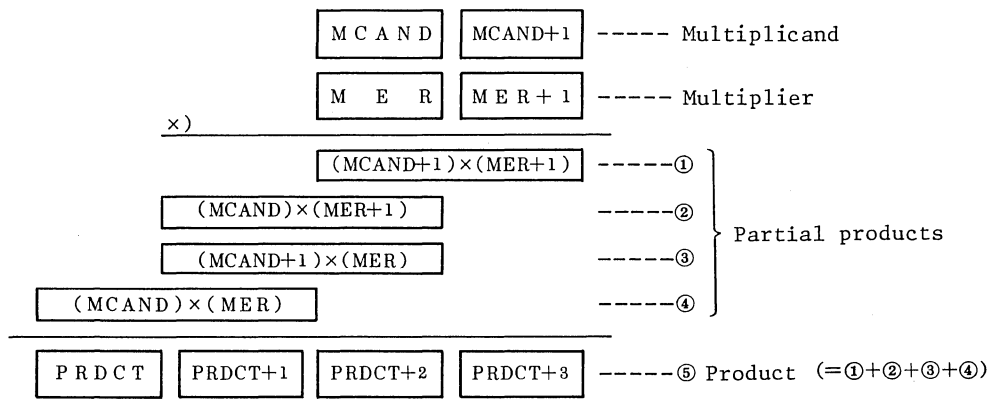
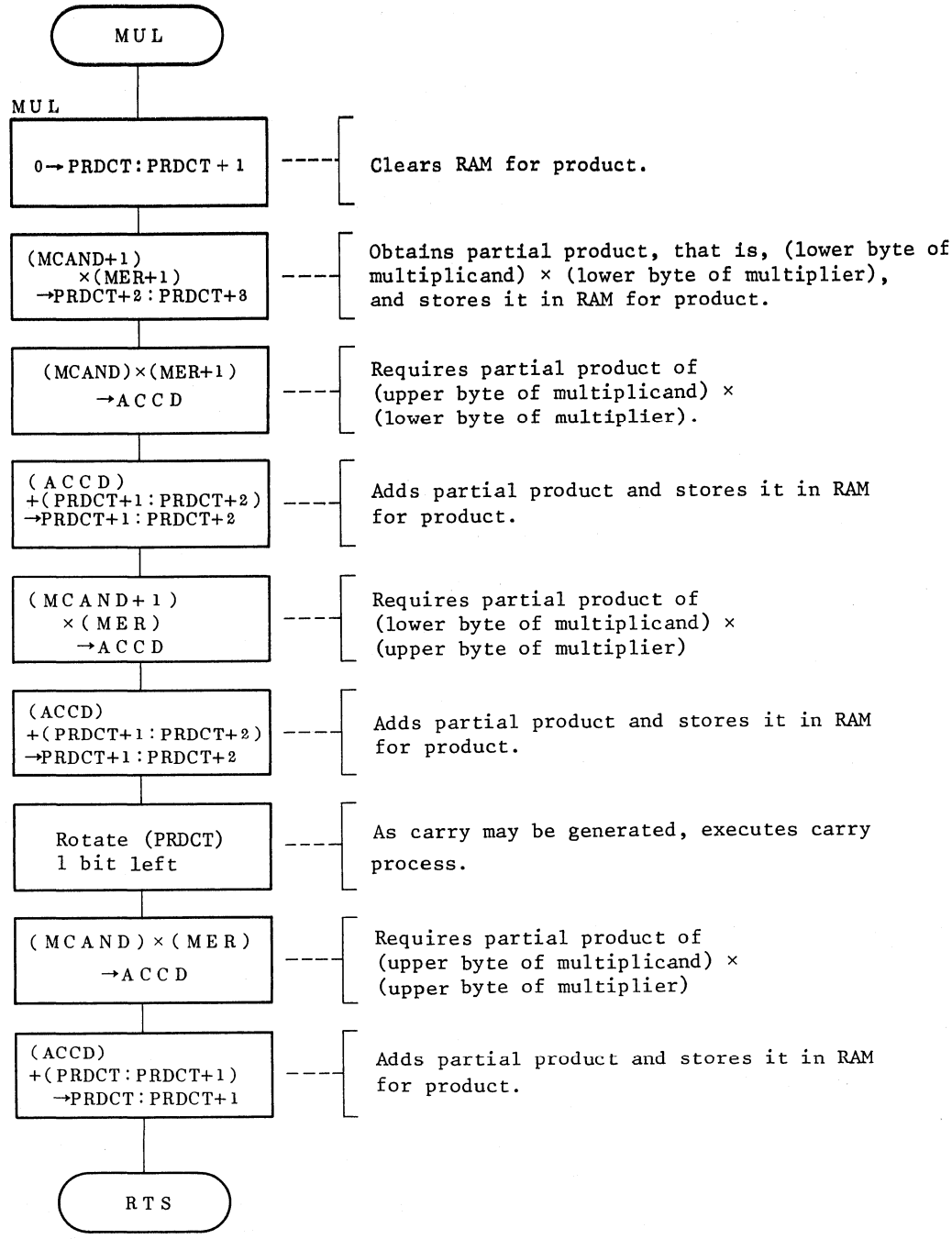


Fig. 3 Multiplication



FLOWCHART



PROGRAM LISTING
-----------------

```

00001 *****
00002 *
00003 * NAME : MULTIPLYING 16-BIT BINARY DATA (MUL) *
00004 *
00005 *****
00006 *
00007 *          ENTRY : MCAND (MULTIPLICAND) *
00008 *                   MER (MULTIPLIER) *
00009 *          RETURNS : PRDCT (PRODUCT) *
00010 *
00011 *****
00012 *
00013A 0080          *          ORG    $80
00014          *
00015A 0080          0002 A MCAND  RMB    2          Multiplicand
00016A 0082          0002 A MER    RMB    2          Multiplier
00017A 0084          0004 A PRDCT  RMB    4          Product
00018          *
00019A F000          *          ORG    $F000
00020          *
00021          F000 A MUL    EQU    *          Entry point
00022A F000 4F          CLRA          Clear product area
00023A F001 5F          CLR    CLR    B
00024A F002 DD 84      A          STD    PRDCT
00025A F004 96 81      A          LDAA  MCAND+1 (MCAND+1) * (MER+1) -> PRDCT
00026A F006 D6 83      A          LDAB  MER+1
00027A F008 3D          MUL
00028A F009 DD 86      A          STD    PRDCT+2
00029A F00B 96 80      A          LDAA  MCAND (MCAND) * (MER+1) -> ACCD
00030A F00D D6 83      A          LDAB  MER+1
00031A F00F 3D          MUL
00032A F010 D3 85      A          ADDD  PRDCT+1 ACCD + (PRDCT) -> PRDCT
00033A F012 DD 85      A          STD    PRDCT+1
00034A F014 96 81      A          LDAA  MCAND+1 (MCAND+1) * (MER) -> ACCD
00035A F016 D6 82      A          LDAB  MER
00036A F018 3D          MUL
00037A F019 D3 85      A          ADDD  PRDCT+1 ACCD + (PRDCT) -> PRDCT
00038A F01B DD 85      A          STD    PRDCT+1
00039A F01D 79 0084    A          ROL    PRDCT
00040A F020 96 80      A          LDAA  MCAND (MCAND) * (MER) -> ACCD
00041A F022 D6 82      A          LDAB  MER
00042A F024 3D          MUL
00043A F025 D3 84      A          ADDD  PRDCT ACCD + PRDCT -> PRDCT
00044A F027 DD 84      A          STD    PRDCT
00045A F029 39          RTS

```

15. DIVIDING 16-BIT BINARY DATA	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	DIV
---------------------------------	---------	----------------------	-------	-----

**FUNCTION**

(a) Performs divisions of 16-bit binary data and stores result (quotient and residual) in 16-bit binary.

(b) Stores dividend and divisor in IX and RAM.

(c) Utilizes unsigned integers in arguments.

<b>ARGUMENTS</b>			<b>CHANGES IN CPU REGISTERS AND FLAGS</b>		<b>SPECIFICATIONS</b>																																																										
<table border="1"> <thead> <tr> <th colspan="2">Contents</th> <th>Storage Location</th> <th>Byte Lgth.</th> </tr> </thead> <tbody> <tr> <td rowspan="2">Arguments</td> <td>Entry</td> <td>Dividend</td> <td>IX</td> <td>2</td> </tr> <tr> <td></td> <td>Divisor</td> <td>DVS (RAM)</td> <td>2</td> </tr> <tr> <td rowspan="2">Returns</td> <td></td> <td>Quotient</td> <td>IX</td> <td>2</td> </tr> <tr> <td></td> <td>Residual</td> <td>ACCD</td> <td>2</td> </tr> </tbody> </table>			Contents		Storage Location	Byte Lgth.	Arguments	Entry	Dividend	IX	2		Divisor	DVS (RAM)	2	Returns		Quotient	IX	2		Residual	ACCD	2	<p>● : Not affected          × : Undefined          ↓ : Result</p> <table border="1"> <thead> <tr> <th colspan="2">ACCD</th> </tr> <tr> <th>ACCA</th> <th>ACCB</th> </tr> </thead> <tbody> <tr> <td>↓</td> <td>↓</td> </tr> <tr> <td>IX</td> <td></td> </tr> <tr> <td>↓</td> <td></td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th>C</th> <th>V</th> </tr> </thead> <tbody> <tr> <td>×</td> <td>×</td> </tr> <tr> <td>Z</td> <td>N</td> </tr> <tr> <td>×</td> <td>×</td> </tr> <tr> <td>I</td> <td>H</td> </tr> <tr> <td>●</td> <td>×</td> </tr> </tbody> </table>		ACCD		ACCA	ACCB	↓	↓	IX		↓		C	V	×	×	Z	N	×	×	I	H	●	×	<table border="1"> <tr><td>ROM (Bytes)</td></tr> <tr><td>25</td></tr> <tr><td>RAM (Bytes)</td></tr> <tr><td>3</td></tr> <tr><td>Stack (Bytes)</td></tr> <tr><td>0</td></tr> <tr><td>No. of cycles</td></tr> <tr><td>476</td></tr> <tr><td>Reentrant</td></tr> <tr><td>No</td></tr> <tr><td>Relocation</td></tr> <tr><td>No</td></tr> <tr><td>Interrupt</td></tr> <tr><td>Yes</td></tr> </table>	ROM (Bytes)	25	RAM (Bytes)	3	Stack (Bytes)	0	No. of cycles	476	Reentrant	No	Relocation	No	Interrupt	Yes
Contents		Storage Location	Byte Lgth.																																																												
Arguments	Entry	Dividend	IX	2																																																											
		Divisor	DVS (RAM)	2																																																											
Returns		Quotient	IX	2																																																											
		Residual	ACCD	2																																																											
ACCD																																																															
ACCA	ACCB																																																														
↓	↓																																																														
IX																																																															
↓																																																															
C	V																																																														
×	×																																																														
Z	N																																																														
×	×																																																														
I	H																																																														
●	×																																																														
ROM (Bytes)																																																															
25																																																															
RAM (Bytes)																																																															
3																																																															
Stack (Bytes)																																																															
0																																																															
No. of cycles																																																															
476																																																															
Reentrant																																																															
No																																																															
Relocation																																																															
No																																																															
Interrupt																																																															
Yes																																																															

**DESCRIPTION**

(1) Function Details

(a) Argument details

IX: Holds 16-bit binary dividend. Contains quotient after DIV execution.

DVS (RAM): Holds 16-bit binary divisor.

ACCD: Contains 16-bit binary residual.

**SPECIFICATIONS NOTES**

DESCRIPTION

- (b) Fig. 1 shows example of DIV execution.  
If entry arguments are as shown in part ① of Fig. 1, division result is contained in IX and ACCD as shown in part ② of Fig. 1.
- (c) Table 1 shows result when "0" is held as entry arguments.

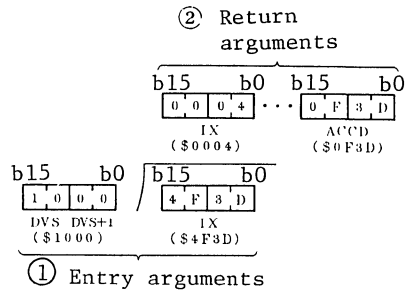


Fig. 1 Example of DIV execution

Table 1 Result when holding "0" as entry arguments

Entry arguments		Return arguments	
Dividend (IX)	Divisor (DVS)	Quotient (IX)	Residual (ACCD)
\$ * * * * *	\$ 0 0 0 0	\$ F F F F	\$ * * * * *
\$ 0 0 0 0	\$ * * * * *	\$ 0 0 0 0	\$ 0 0 0 0
\$ 0 0 0 0	\$ 0 0 0 0	\$ F F F F	\$ 0 0 0 0

(NOTE) \* \$\*\*\*\* indicates hexadecimals.

(2) User Notes

- (a) When not using upper byte as shown in Fig. 2, the upper byte should be held to "0", otherwise division result will not be correct, because division is performed with underfined data in the upper byte.
- (b) After DIV execution, dividend is destroyed because quotient is contained in IX. If dividend needs to be retained after DIV execution, it should be saved in memory before execution.

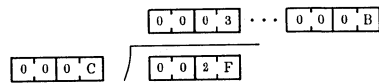
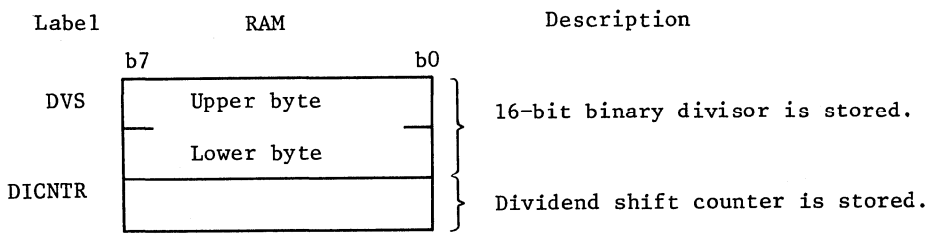


Fig. 2 DIV example when upper byte is not used

DESCRIPTION

(3) RAM Description



(4) Sample Application

Subroutine DIV is called after dividend and divisor are held.

```

WORK1   RMB   2   ----- Reserves memory byte for 16-bit binary
                               dividend.
WORK2   RMB   2   ----- Reserves memory byte for 16-bit binary
                               divisor.
WORK3   RMB   2   ----- Reserves memory byte for 16-bit binary
                               quotient.
WORK4   RMB   2   ----- Reserves memory byte for 16-bit binary
                               residual.
        |
        |
        |
        LDX  WORK1  ----- Loads 16-bit binary dividend into
                               entry argument (IX).
        LDD  WORK2  }
        STD  DVS    } ----- Stores 16-bit binary divisor into
                               entry argument (DVS).
        JSR  DIV    } ----- Calls subroutine DIV.
        STX  WORK3  }
        STD  WORK4  } ----- Stores division result in RAM.
        |
        |
        |
    
```



DESCRIPTION	
-------------	--

(5) Basic Operation

(a) In binary code division, quotient and residual are obtained by repeated subtraction. Fig. 3 shows example of division (\$0D ÷ \$03).

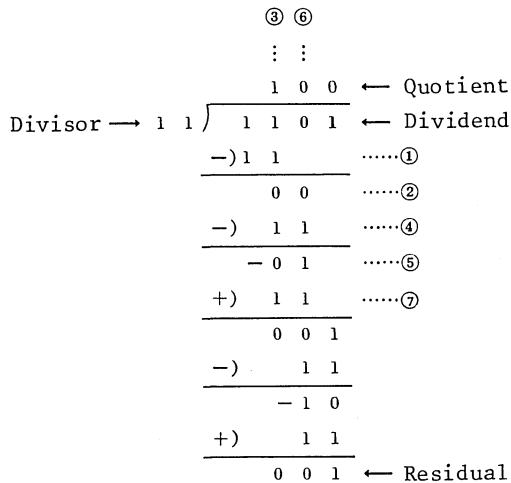


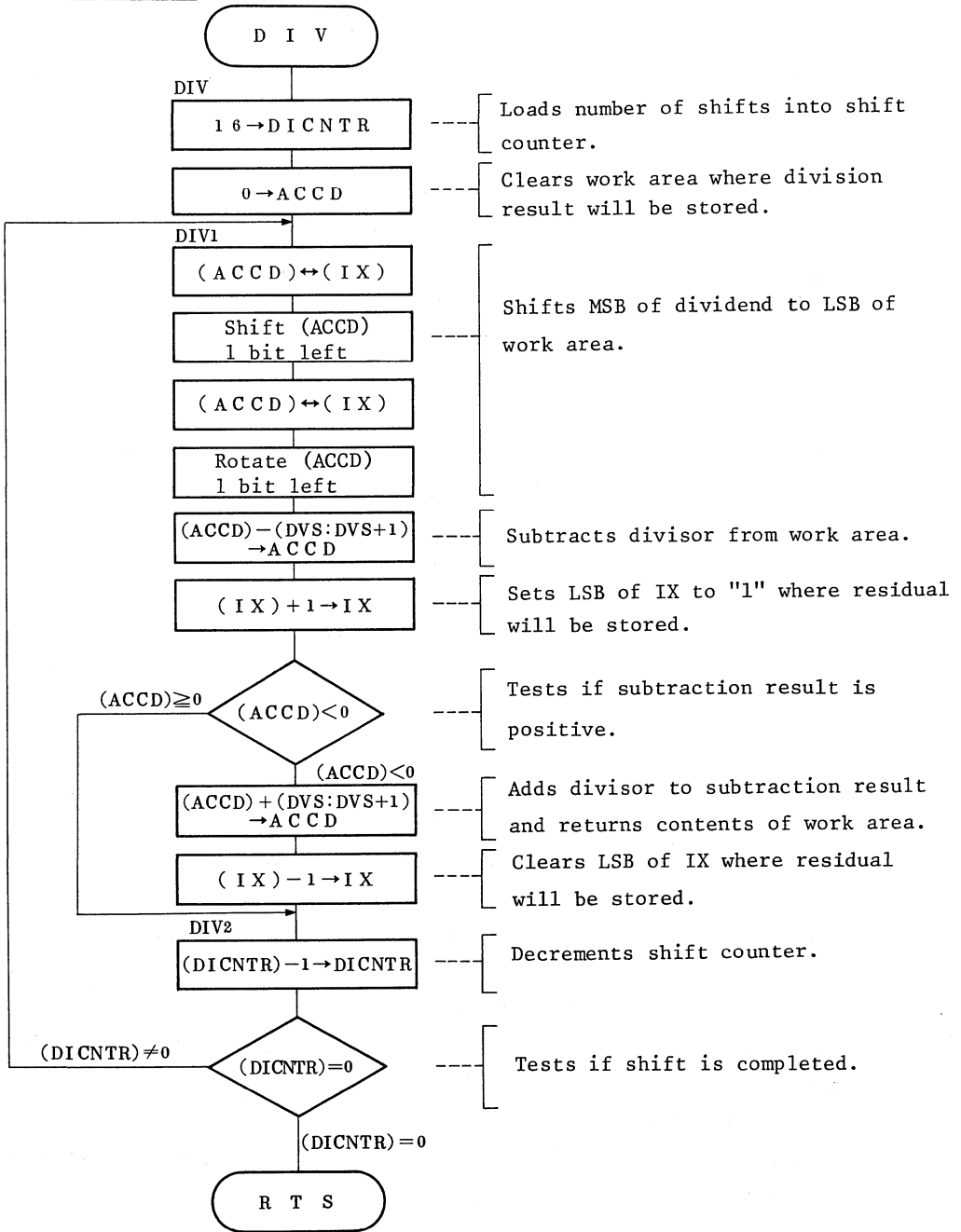
Fig. 3 Division example (\$0D ÷ \$03)

(b) Referring to Fig. 3, the program is explained as follows:

- (i) Loads number of shifts in shift counter (DICNTR) and clears ACCD in which residual will be loaded.
- (ii) Shifts dividend (IX) and ACCD left 1 bit, then shifts MSB of IX to LSB of ACCD. This is because when performing subtraction, the upper bits are fetched one by one from dividend.
- (iii) Subtracts divisor (DVS:DVS+1) from ACCD. If subtraction result is positive, sets LSB of IX to "1".  
(Fig. 3 ①→②→③).  
If subtraction result is negative, clears LSB of IX, adds divisor to subtraction result and restores the previous subtraction result.  
(Fig. 3 ⑤→⑥→⑦).
- (iv) Decrements shift counter.
- (v) Loops (ii) to (iv) until shift counter is "0".

7

FLOWCHART



## PROGRAM LISTING

```

00001 *****
00002 *
00003 * NAME : DIVIDING 16-BIT BINARY DATA (DIV) *
00004 *
00005 *****
00006 *
00007 * ENTRY : IX (DIVIDEND) *
00008 * DVS (DIVISOR) *
00009 * RETURNS : ACCD (QUOTIENT) *
00010 * IX (RESIDUAL) *
00011 *
00012 *****
00013 *
00014A 0080 * ORG $80
00015 *
00016A 0080 0002 A DVS RMB 2 Divisor
00017A 0082 0001 A DICNTR RMB 1 Shift counter
00018 *
00019A F000 * ORG $F000
00020 *
00021 F000 A DIV EQU * Entry point
00022A F000 86 10 A LDAA #16 Set shift counter
00023A F002 97 82 A STAA DICNTR
00024A F004 4F CLRA Clear work (Set quotient afterward)
00025A F005 5F CLRB
00026A F006 18 DIV1 XGDX Shift dividend and set MSB of-
00027A F007 05 ASLD -dividend to LSB of work
00028A F008 18 XGDX
00029A F009 59 ROLB
00030A F00A 49 ROLA
00031A F00B 93 80 A SUBD DVS Work - Divisor -> Work
00032A F00D 08 INX Set high to LSB of residual area
00033A F00E 24 03 F013 BCC DIV2 Branch if work>=divisor
00034A F010 03 80 A ADDD DVS Work + Divisor -> Work
00035A F012 09 DEX Clear LSB of residual area
00036A F013 7A 0082 A DIV2 DEC DICNTR Decrement shift counter
00037A F016 26 EE F006 BNE DIV1 Loop until shift counter = 0
00038A F018 39 RTS

```



**FUNCTION**

(a) Performs addition of 8-digit BCD number in RAM, and stores result in 8-digit BCD number in RAM.

(b) Utilizes unsigned integers in arguments.

ARGUMENTS	CHANGES IN CPU REGISTERS AND FLAGS	SPECIFICATIONS																																																						
<table border="1" style="width:100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2">Contents</th> <th>Storage Location</th> <th>Byte Lgth.</th> </tr> </thead> <tbody> <tr> <td rowspan="2" style="text-align: center; vertical-align: middle;">Arguments</td> <td style="text-align: center;">Entry</td> <td>Augend ABD (RAM)</td> <td style="text-align: center;">4</td> </tr> <tr> <td style="text-align: center;">Returns</td> <td>Addend ACD (RAM)</td> <td style="text-align: center;">4</td> </tr> <tr> <td rowspan="2" style="text-align: center; vertical-align: middle;">Returns</td> <td style="text-align: center;">Addition result</td> <td>ABD (RAM)</td> <td style="text-align: center;">4</td> </tr> <tr> <td style="text-align: center;">Carry or no carry</td> <td>bit C (CCR)</td> <td style="text-align: center;">1</td> </tr> </tbody> </table>	Contents		Storage Location	Byte Lgth.	Arguments	Entry	Augend ABD (RAM)	4	Returns	Addend ACD (RAM)	4	Returns	Addition result	ABD (RAM)	4	Carry or no carry	bit C (CCR)	1	<p>● : Not affected          × : Undefined          † : Result</p> <table border="1" style="width:100%; border-collapse: collapse; margin-bottom: 10px;"> <tr><th colspan="2">ACCD</th></tr> <tr><td style="text-align: center;">ACCA</td><td style="text-align: center;">ACCB</td></tr> <tr><td style="text-align: center;">×</td><td style="text-align: center;">×</td></tr> <tr><td style="text-align: center;">IX</td><td></td></tr> <tr><td style="text-align: center;">×</td><td></td></tr> </table> <table border="1" style="width:100%; border-collapse: collapse;"> <tr><td style="text-align: center;">C</td><td style="text-align: center;">V</td></tr> <tr><td style="text-align: center;">†</td><td style="text-align: center;">×</td></tr> <tr><td style="text-align: center;">Z</td><td style="text-align: center;">N</td></tr> <tr><td style="text-align: center;">×</td><td style="text-align: center;">×</td></tr> <tr><td style="text-align: center;">I</td><td style="text-align: center;">H</td></tr> <tr><td style="text-align: center;">●</td><td style="text-align: center;">×</td></tr> </table>	ACCD		ACCA	ACCB	×	×	IX		×		C	V	†	×	Z	N	×	×	I	H	●	×	<table border="1" style="width:100%; border-collapse: collapse;"> <tr><td>ROM (Bytes)</td></tr> <tr><td style="text-align: center;">15</td></tr> <tr><td>RAM (Bytes)</td></tr> <tr><td style="text-align: center;">8</td></tr> <tr><td>Stack (Bytes)</td></tr> <tr><td style="text-align: center;">0</td></tr> <tr><td>No. of cycles</td></tr> <tr><td style="text-align: center;">81</td></tr> <tr><td>Reentrant</td></tr> <tr><td style="text-align: center;">No</td></tr> <tr><td>Relocation</td></tr> <tr><td style="text-align: center;">No</td></tr> <tr><td>Interrupt</td></tr> <tr><td style="text-align: center;">Yes</td></tr> </table>	ROM (Bytes)	15	RAM (Bytes)	8	Stack (Bytes)	0	No. of cycles	81	Reentrant	No	Relocation	No	Interrupt	Yes
Contents		Storage Location	Byte Lgth.																																																					
Arguments	Entry	Augend ABD (RAM)	4																																																					
	Returns	Addend ACD (RAM)	4																																																					
Returns	Addition result	ABD (RAM)	4																																																					
	Carry or no carry	bit C (CCR)	1																																																					
ACCD																																																								
ACCA	ACCB																																																							
×	×																																																							
IX																																																								
×																																																								
C	V																																																							
†	×																																																							
Z	N																																																							
×	×																																																							
I	H																																																							
●	×																																																							
ROM (Bytes)																																																								
15																																																								
RAM (Bytes)																																																								
8																																																								
Stack (Bytes)																																																								
0																																																								
No. of cycles																																																								
81																																																								
Reentrant																																																								
No																																																								
Relocation																																																								
No																																																								
Interrupt																																																								
Yes																																																								

**DESCRIPTION**

(1) Function Details

(a) Argument details

ABD : Holds 8-digit BCD augend. (RAM) ① Entry arguments

ACD : Holds 8-digit BCD added. (RAM) ② Return arguments

ABD (RAM) (12478082)	+	ACD (RAM) (70008901)	=	Result
1 2 4 7 8 0 3 2		7 6 0 0 8 9 0 1		1 2 4 7 8 0 3 2
ABD		ACD		ABD+1 ABD+2 ABD+8

ABD (RAM) (88480983)	+	ACD (RAM) (70008901)	=	Result
3 8 4 8 6 9 3 3		7 6 0 0 8 9 0 1		1 2 4 7 8 0 3 2
ABD		ACD		ABD+1 ABD+2 ABD+8

Fig. 1 Example of ADD execution

**SPECIFICATIONS NOTES**

**DESCRIPTION**

bit C: Indicates whether a carry is (CCR) generated or not after ADDD execution.

bit C=1: Carry is generated in addition result.  
(See Fig. 2)

bit C=0: No carry is generated in addition result.

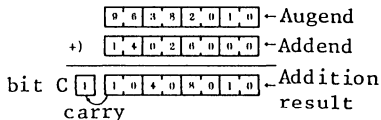


Fig. 2 Addition example when carry is generated

(b) Fig. 1 shows example of ADDD execution. If entry arguments are as shown in part ① of Fig. 1, addition result is contained in ABD(RAM) as shown in part ② of Fig. 1.

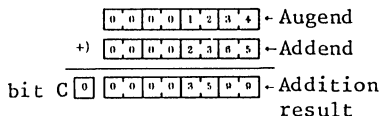


Fig. 3 Addition example when upper byte is not used

**(2) User Notes**

- (a) As shown in Fig. 3, when not using upper byte, the upper byte should be held to "0", otherwise addition result will not be correct, because addition is performed with undefined data in the upper byte.
- (b) After ADDD execution, augend is destroyed because addition result is contained in ABD(RAM). If augend needs to be retained after ADDD execution, it should be saved in memory before execution.
- (c) BCD number should be held in augend and addend, otherwise addition result will not be correct.

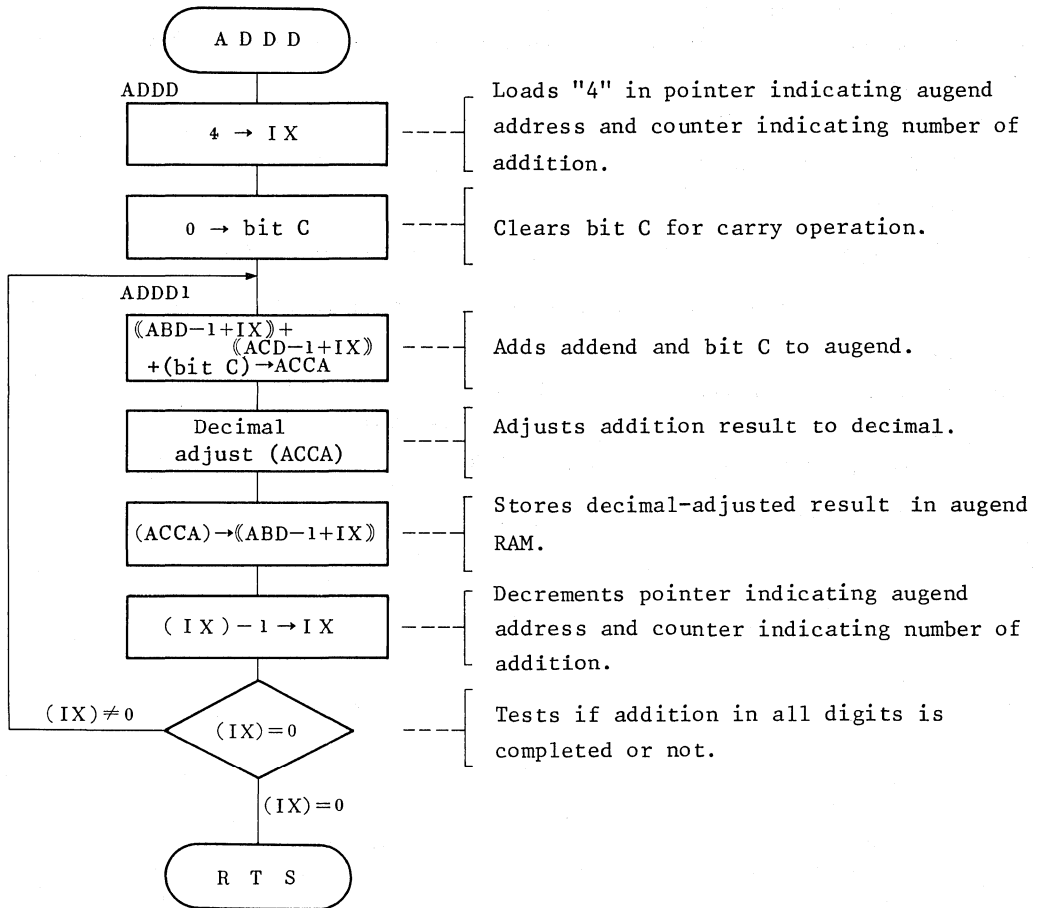
**(3) RAM Description**

Label	RAM	Description
ABD	b7	} 8-digit BCD augend is stored before execution.
	Upper byte	
	Lower byte	
	b0	
ACD	Upper byte	} 8-digit BCD addend is stored.
	Lower byte	



16. ADDING 8-DIGIT BCD	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	ADDD
DESCRIPTION				
<p>(5) Basic Operation</p> <p>(a) When addition of more than 2 bytes are executed, addition result can be obtained by repeating addition for each byte.</p> <p>(b) IX is used to indicate augend and addend addresses and is also used to count number of additions.</p> <p>(c) Clears bit C at first.</p> <p>(d) Performs (Formula 1) on each byte of augend and addend using index addressing mode.</p> <p style="text-align: center;">Augend + Addend + (bit C) → ACCA ----- (Formula 1)</p> <p>Bit C is added in (Formula 1) because addition result of lower bytes generate carry.</p> <p>(e) Adjusts addition result of (d) to decimal value using decimal adjust ACCA instruction (DAA), and holds it in augend ABD (RAM).</p> <p>(f) Decrements IX every time (d) to (e) is executed.</p> <p>(g) Loops (d) to (f) until IX is "0".</p>				

## FLOWCHART



PROGRAM LISTING
-----------------

```

00001          *****
00002          *
00003          *   NAME : ADDING 8-DIGIT BCD       (ADDD)   *
00004          *
00005          *****
00006          *
00007          *   ENTRY  : ABD (AUGEND)                *
00008          *           ACD (ADDEND)                *
00009          *   RETURNS : ABD (SUM)                 *
00010          *           CARRY (C=0;TRUE,C=1;OVER FLOW) *
00011          *
00012          *****
00013          *
00014A 0080          *           ORG   $80
00015          *
00016A 0080          *           ORG   $80
00017A 0084          *           ORG   $84
00018          *
00019A F000          *           ORG  $F000
00020          *
00021          *           ORG  $F000
00022A F000 CE 0004 A ADDD EQU *           Entry point
00023A F003 0C          *           LDX  #4           Set ADDR pointer(addition counter)
00024A F004 A6 7F A ADDD1 CLC                   Clear carry bit
00025A F006 A9 83 A ADDD1 LDAA ABD-1.X         Augend+addend
00026A F008 19          *           ADCA ACD-1.X
00027A F009 A7 7F A ADDD1 DAA                   Convert into BCD
00028A F00B 09          *           STAA ABD-1.X        Store in augend area
00029A F00C 26 F6 F004 BNE ADDD1              Decrement ADDR pointer
00030A F00E 39          *           RTS                   Loop until ADDR pointer = 0

```

7

**FUNCTION**

(a) Performs subtraction of 8-DIGIT BCD number in RAM and stores result in 8-digit BCD number in RAM.

(b) Utilizes unsigned integers in arguments.

ARGUMENTS	CHANGES IN CPU REGISTERS AND FLAGS	SPECIFICATIONS																																																						
<table border="1" style="width:100%; border-collapse: collapse;"> <thead> <tr> <th style="width:10%;"></th> <th style="width:15%;">Contents</th> <th style="width:15%;">Storage Location</th> <th style="width:10%;">Byte Lgth.</th> </tr> </thead> <tbody> <tr> <td rowspan="3" style="text-align: center; vertical-align: middle;">Arguments</td> <td style="text-align: center;">Entry</td> <td>Minuend SUBEDS (RAM)</td> <td style="text-align: center;">4</td> </tr> <tr> <td style="text-align: center;">Subtra-hend</td> <td>SUBERS (RAM)</td> <td style="text-align: center;">4</td> </tr> <tr> <td style="text-align: center;">Re- turns</td> <td>Subtraction result SUBEDS (RAM)</td> <td style="text-align: center;">4</td> </tr> <tr> <td></td> <td style="text-align: center;">Borrow or no borrow</td> <td>bit C (CCR)</td> <td style="text-align: center;">1</td> </tr> </tbody> </table>		Contents	Storage Location	Byte Lgth.	Arguments	Entry	Minuend SUBEDS (RAM)	4	Subtra-hend	SUBERS (RAM)	4	Re- turns	Subtraction result SUBEDS (RAM)	4		Borrow or no borrow	bit C (CCR)	1	<p>● : Not affected          × : Undefined          † : Result</p> <table border="1" style="width:100%; border-collapse: collapse; margin-bottom: 10px;"> <tr><th colspan="2" style="text-align: center;">ACCD</th></tr> <tr><td style="text-align: center;">ACCA</td><td style="text-align: center;">ACCB</td></tr> <tr><td style="text-align: center;">×</td><td style="text-align: center;">×</td></tr> <tr><td style="text-align: center;">IX</td><td></td></tr> <tr><td style="text-align: center;">×</td><td></td></tr> </table> <table border="1" style="width:100%; border-collapse: collapse;"> <tr><td style="text-align: center;">C</td><td style="text-align: center;">V</td></tr> <tr><td style="text-align: center;">†</td><td style="text-align: center;">×</td></tr> <tr><td style="text-align: center;">Z</td><td style="text-align: center;">N</td></tr> <tr><td style="text-align: center;">×</td><td style="text-align: center;">×</td></tr> <tr><td style="text-align: center;">I</td><td style="text-align: center;">H</td></tr> <tr><td style="text-align: center;">●</td><td style="text-align: center;">×</td></tr> </table>	ACCD		ACCA	ACCB	×	×	IX		×		C	V	†	×	Z	N	×	×	I	H	●	×	<table border="1" style="width:100%; border-collapse: collapse;"> <tr><td style="text-align: center;">ROM (Bytes)</td></tr> <tr><td style="text-align: center;">29</td></tr> <tr><td style="text-align: center;">RAM (Bytes)</td></tr> <tr><td style="text-align: center;">8</td></tr> <tr><td style="text-align: center;">Stack (Bytes)</td></tr> <tr><td style="text-align: center;">0</td></tr> <tr><td style="text-align: center;">No. of cycles</td></tr> <tr><td style="text-align: center;">120</td></tr> <tr><td style="text-align: center;">Reentrant</td></tr> <tr><td style="text-align: center;">No</td></tr> <tr><td style="text-align: center;">Relocation</td></tr> <tr><td style="text-align: center;">No</td></tr> <tr><td style="text-align: center;">Interrupt</td></tr> <tr><td style="text-align: center;">Yes</td></tr> </table>	ROM (Bytes)	29	RAM (Bytes)	8	Stack (Bytes)	0	No. of cycles	120	Reentrant	No	Relocation	No	Interrupt	Yes
	Contents	Storage Location	Byte Lgth.																																																					
Arguments	Entry	Minuend SUBEDS (RAM)	4																																																					
	Subtra-hend	SUBERS (RAM)	4																																																					
	Re- turns	Subtraction result SUBEDS (RAM)	4																																																					
	Borrow or no borrow	bit C (CCR)	1																																																					
ACCD																																																								
ACCA	ACCB																																																							
×	×																																																							
IX																																																								
×																																																								
C	V																																																							
†	×																																																							
Z	N																																																							
×	×																																																							
I	H																																																							
●	×																																																							
ROM (Bytes)																																																								
29																																																								
RAM (Bytes)																																																								
8																																																								
Stack (Bytes)																																																								
0																																																								
No. of cycles																																																								
120																																																								
Reentrant																																																								
No																																																								
Relocation																																																								
No																																																								
Interrupt																																																								
Yes																																																								

**DESCRIPTION**

(1) Function Details

(a) Argument details

SUBEDS: Holds 8-digit BCD minuend. (RAM)

After SUBD execution, contains subtraction result.

SUBERS: Holds 8-digit BCD subtra-hend. (RAM)

① Entry arguments

SUBEDS (RAM) (90123456)

SUBERS (RAM) (12345678)

b31 b0

9	0	1	2	3	4	5	6	-	7	8
SUBERS SUBERS SUBERS SUBERS									SUBEDS SUBEDS	

Minuend

② Return arguments

SUBEDS (RAM) (77777778)

bit C b31 b0

1	7	7	7	7	7	7	7	8	-	7	8
SUBERS SUBERS SUBERS SUBERS									SUBEDS SUBEDS		

Subtraction result

Fig. 1 Example of SUBD execution

**SPECIFICATIONS NOTES**



17. SUBTRACTING 8-DIGIT BCD	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	SUBD
-----------------------------	---------	----------------------	-------	------

DESCRIPTION

bit C: Indicates whether borrow is (CCR) generated or not after SUBD execution

bit C=1: Borrow is generated in subtraction result.

bit C=0: No borrow is generated in subtraction result.

(See Fig. 2).

- (b) Fig. 1 shows example of SUB execution. If entry arguments are as shown in part ① of Fig. 1, subtraction result is contained in SUBEDS (RAM) as shown in part ② of Fig. 1.

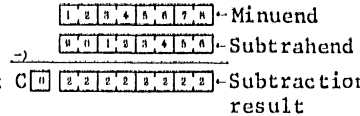


Fig. 2 Subtraction example when borrow is generated

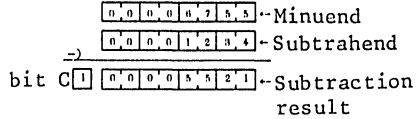


Fig. 3 Subtraction example when not using upper byte

(2) User Notes

- (a) As shown in Fig. 3, when not using upper byte, the upper byte should be held to "0", otherwise subtraction result will not be correct, because subtraction result is performed with undefined data held in the upper byte.
- (b) After SUBD execution, minuend is destroyed because subtraction result is stored in SUBEDS (RAM). If minuend needs to be retained after SUBD execution, it should be saved in memory before execution.
- (c) BCD number should be held in minuend and subtrahend, otherwise subtraction result will not be correct.

(3) RAM Description

Label	RAM	Description
SUBEDS	Upper byte	8-digit BCD minuend is stored before execution.
	Lower byte	
SUBERS	Upper byte	8-digit BCD subtrahend is stored.
	Lower byte	

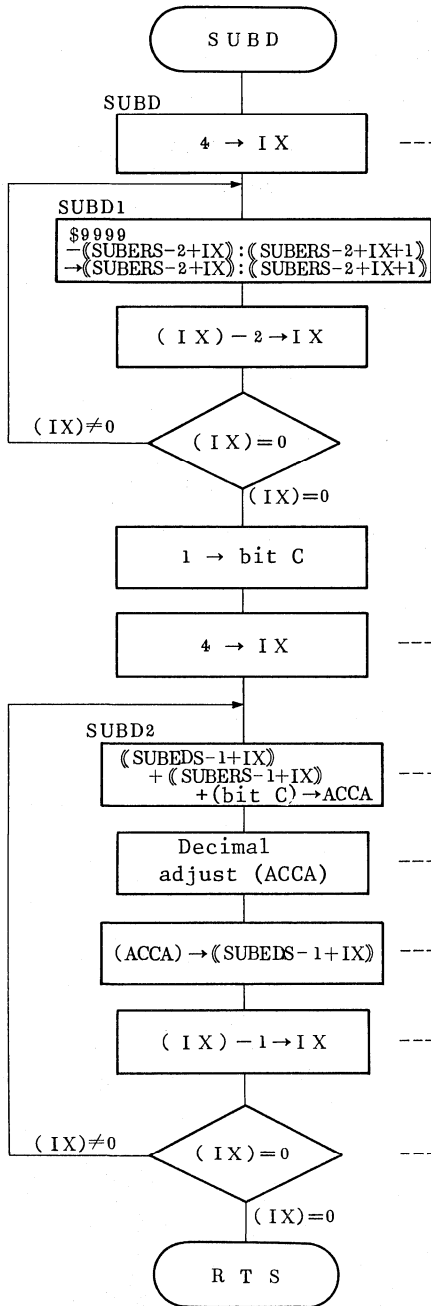




17. SUBTRACTING 8-DIGIT BCD	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	SUBD
DESCRIPTION				
<p>(5) Basic Operation</p> <p>(a) Subtraction of BCD can be performed by (Formula 1) and (Formula 2).  <math display="block">\text{Minuend-Subtrahend} = \text{Minuend} + 10\text{'s complement of subtrahend} \dots \text{(Formula 1)}</math> <math display="block">10\text{'s complement of minuend} = \\$99 - \text{Subtrahend} + 1 \dots \text{(Formula 2)}</math></p> <p>(b) Using (Formula 1) and (Formula 2), the program is described as follows:</p> <p>(i) Takes 10's complements of 8-digit subtrahend by (Formula 2) and stores them in SUBERS (RAM).</p> <p>(ii) IX is used to indicate minuend and subtrahend, and is also used to count number of subtraction.</p> <p>(iii) Performs (Formula 3) on every byte from LSB of 10's complement of minuend and subtrahend using index addressing mode.  <math display="block">\text{Minuend} + 10\text{'s complement of subtrahend} + (\text{bit C}) \rightarrow \text{ACCA} \dots \text{(Formula 3)}</math></p> <p>(iv) Adjusts subtraction result of (iii) to decimal number using decimal adjust instruction (DAA) and stores it in minuend RAM (SUBEDS).</p> <p>(v) Decrements IX.</p> <p>(vi) Loops (iii) to (v) until IX is "0".</p>				

7

FLOWCHART



Takes 10's complement of subtrahend, and stores it in subtrahend RAM. However, only (\$9999 - subtrahend) is calculated in this part. Adds "1" in the next step by setting bit C (See Formula 2 in "(5) Basic Operation").

Loads "4" in pointer indicating minuend address and counter indicating number of subtraction.

Adds 10's complement of subtrahend and bit C to minuend.

Adjusts addition result to decimal.

Stores decimal-adjusted result in minuend RAM.

Decrements pointer indicating minuend address and counter indicating number of subtraction.

Tests if subtraction in all digits is completed or not.

PROGRAM LISTING

```

00001          *****
00002          *
00003          *   NAME : SUBTRACTING 8-DIGIT BCD   (SUBD)   *
00004          *
00005          *****
00006          *
00007          *   ENTRY  : SUBEDS (MINUEND)             *
00008          *           SUBERS (SUBTRAHEND)         *
00009          *   RETURNS : SUBEDS (RESIDUAL)         *
00010          *           CARRY  (C=1:TRUE,C=0:BORROW)  *
00011          *
00012          *****
00013          *
00014A 0080          ORG      $80
00015          *
00016A 0080          0004 A SUBEDS RMB      4      Minuend -> Residual
00017A 0084          0004 A SUBERS RMB      4      Subtrahend
00018          *
00019A F000          ORG      $F000
00020          *
00021          F000 A SUBD   EQU      *      Entry point
00022A F000 CE 0004 A      LD      #4      99999999-Subtrahend -> SUBERS
00023A F003 CC 9999 A SUBD1 LD      #$9999
00024A F006 A3 82  A      SUBD   SUBERS-2,X
00025A F008 ED 82  A      STD    SUBERS-2,X
00026A F00A 09          DEX
00027A F00B 09          DEX
00028A F00C 26 F5 F003 BNE      SUBD1
00029A F00E 0D          SEC
00030A F00F CE 0004 A      LD      #4      Set carry bit
00031A F012 A6 7F  A SUBD2 LDAA   SUBEDS-1,X Minuend+negative of subtrahend
00032A F014 A9 83  A      ADCA   SUBERS-1,X
00033A F016 19          DAA      Convert into BCD
00034A F017 A7 7F  A      STAA   SUBEDS-1,X Store in SUBEDS area
00035A F019 09          DEX      Decrement ADDR pointer
00036A F01A 26 F6 F012 BNE      SUBD2      Loop until ADDR pointer = 0
00037A F01C 39          RTS

```

18. 16-BIT SQUARE ROOT	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	SQRT
------------------------	---------	----------------------	-------	------

**FUNCTION**

(a) Obtains square root of 16-bit binary data in IX, and stores result in RAM.

(b) Utilizes unsigned integers in arguments.

<b>ARGUMENTS</b>		<b>CHANGES IN CPU REGISTERS AND FLAGS</b>		<b>SPECIFICATIONS</b>																																															
<table border="1"> <thead> <tr> <th colspan="2">Contents</th> <th>Storage Location</th> <th>Byte Lgth.</th> </tr> </thead> <tbody> <tr> <td rowspan="2">Arguments</td> <td>Entry</td> <td>IX</td> <td>2</td> </tr> <tr> <td>Returns</td> <td>SANS+1 (RAM)</td> <td>1</td> </tr> </tbody> </table>		Contents		Storage Location	Byte Lgth.	Arguments	Entry	IX	2	Returns	SANS+1 (RAM)	1	<ul style="list-style-type: none"> <li>● : Not affected</li> <li>× : Undefined</li> <li>↑ : Result</li> </ul> <table border="1" style="margin: 5px 0;"> <tr><th colspan="2">ACCD</th></tr> <tr><td>ACCA</td><td>ACCB</td></tr> <tr><td>×</td><td>×</td></tr> <tr><td>IX</td><td></td></tr> <tr><td>×</td><td></td></tr> </table> <table border="1" style="margin: 5px 0;"> <tr><td>C</td><td>V</td></tr> <tr><td>×</td><td>×</td></tr> <tr><td>Z</td><td>N</td></tr> <tr><td>×</td><td>×</td></tr> <tr><td>I</td><td>H</td></tr> <tr><td>●</td><td>●</td></tr> </table>		ACCD		ACCA	ACCB	×	×	IX		×		C	V	×	×	Z	N	×	×	I	H	●	●	<table border="1"> <tr><td>ROM (Bytes)</td></tr> <tr><td>54</td></tr> <tr><td>RAM (Bytes)</td></tr> <tr><td>3</td></tr> <tr><td>Stack (Bytes)</td></tr> <tr><td>0</td></tr> <tr><td>No. of cycles</td></tr> <tr><td>478</td></tr> <tr><td>Reentrant</td></tr> <tr><td>No</td></tr> <tr><td>Relocation</td></tr> <tr><td>No</td></tr> <tr><td>Interrupt</td></tr> <tr><td>Yes</td></tr> </table>	ROM (Bytes)	54	RAM (Bytes)	3	Stack (Bytes)	0	No. of cycles	478	Reentrant	No	Relocation	No	Interrupt	Yes
Contents		Storage Location	Byte Lgth.																																																
Arguments	Entry	IX	2																																																
	Returns	SANS+1 (RAM)	1																																																
ACCD																																																			
ACCA	ACCB																																																		
×	×																																																		
IX																																																			
×																																																			
C	V																																																		
×	×																																																		
Z	N																																																		
×	×																																																		
I	H																																																		
●	●																																																		
ROM (Bytes)																																																			
54																																																			
RAM (Bytes)																																																			
3																																																			
Stack (Bytes)																																																			
0																																																			
No. of cycles																																																			
478																																																			
Reentrant																																																			
No																																																			
Relocation																																																			
No																																																			
Interrupt																																																			
Yes																																																			

**DESCRIPTION.**

(1) Function Details

(a) Argument details

IX : Holds 16-bit binary data to be squared.

SANS+1: Contains 16-bit binary square root.

(b) Fig. 1 shows example of SQRT execution. If entry argument is as shown in part ① of Fig. 1, square root is contained in SANS+1 (RAM) as shown in part 2 of Fig. 1.

Fig. 1 Example of SQRT execution

Fig. 2 Example when upper byte is not used

**SPECIFICATIONS NOTES**

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to get square root of \$FFFF.

DESCRIPTION

(2) User Notes

- (a) When not using upper byte as shown in Fig. 2, the upper byte should be held to "0", otherwise square root will not be correct, because square root is obtained with undefined data in the upper byte.
- (b) Values to the right of the binary point are truncated.

(3) RAM Description

Label	RAM	Description
SANS (SANS+1)	<div style="display: flex; align-items: center; justify-content: center;"> <div style="margin-right: 5px;">b7</div> <div style="border: 1px solid black; padding: 5px; text-align: center; width: 150px;">           Upper byte  Lower byte         </div> <div style="margin-left: 5px;">b0</div> </div>	Work area is reserved to square root before execution. 8-bit binary square root is stored in SANS+1, and "0" is stored in SANS after execution.
SCNTR		Shift counter for counting number of shifts of data to be squared is stored.

(4) Sample Application

Subroutine SQRT is called after data to be squared is held.

```

WORK1   RMB   2   ----- Reserves memory byte for 16-bit
                          binary data to be squared.
WORK2   RMB   1   ----- Reserves memory byte for 16-bit binary
                          square root.
        |
        |
        |
        PSHA   }
        PSHB   } ----- Saves register contents that will be
        PSHX   } destroyed by SQRT execution.
        LDX   WORK1 ----- Loads 16-bit binary data to be squared
                          into entry argument (IX).

        JSR   SQRT ----- Calls subroutine SQRT.
        LDAA  SANS+1 }
        STAA  WORK2 } ----- Stores 16-bit binary square root
                          (return argument (SANS+1)) in RAM.

        PULX   }
        PULB   } ----- Restores register.
        PULA   }
        |
        |
        |
    
```

## DESCRIPTION

## (5) Basic Operation

(a) Fig. 3 shows calculation used to obtain 16-bit binary square root.

(Data to be squared=\$22, square root=\$05).

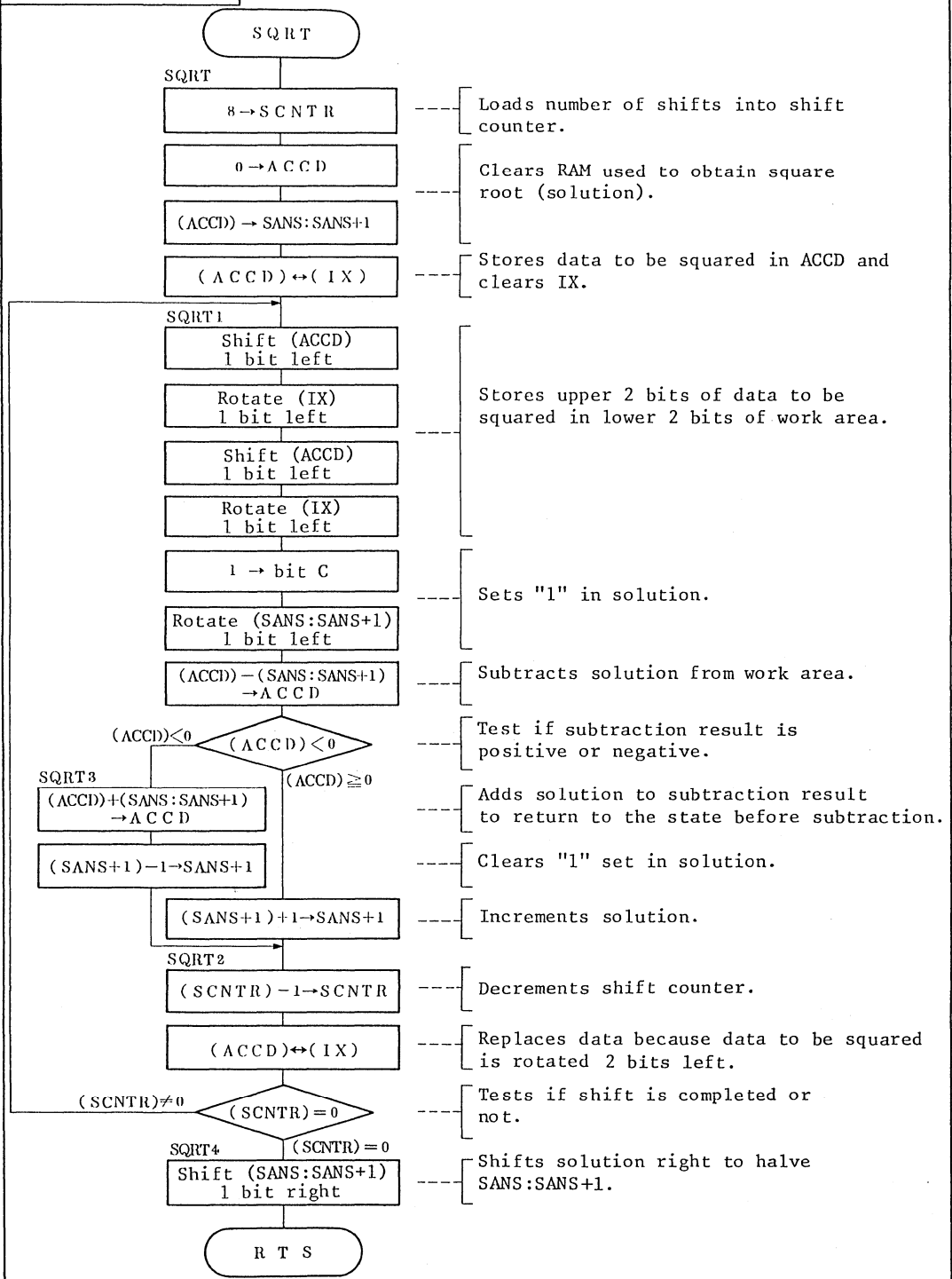
A	B	C	D	E	F	
1			1	0	1	① ..... Square root
1			1	0	0	② ..... Data to be squared
1	0	0	1	0	0	③
1	0	0	1	0	0	④
0			0	0		⑤
1	0	0	1	0	0	⑥
		1	1	0	0	⑦
1	0	1	0	0	1	⑧

Fig. 3 Calculating a square root

(b) The calculation in Fig. 3 is explained as follows:

- (i) Clears square root area, SANS:SANS+1 and work area ACCD.
- (ii) Rotates IX and ACCD two bits left to fetch upper 2 bits of data to be squared and sets upper 2 bits of data to be squared in ACCD.  
(Fig. 3 ① - ②)
- (iii) Sets "1" in 2-byte area, SANS:SANS+1 from RAM address shown in SANS.  
(Fig. 3 ③ - ④).
- (iv) Subtracts SANS:SANS+1 from ACCD, and stores obtained result in ACCD.  
(Fig. 3 ⑤ - ⑥, ⑦, ⑧).
- (v) When result is positive, increments SANS+1. (Fig. 3 ③ - ④)  
When result is negative, decrements SANS+1, and adds SANS:  
SANS+1 to ACCD. (Fig. 3 ⑤, ⑦ - ⑧).
- (vi) In subroutine SQRT, loops (ii) to (v) 8 times and then shifts SANS:SANS+1  
1 bit right to halve SANS:SANS+1. (Fig. 3 ③, ④ - ⑧ is square root).

FLOWCHART





PROGRAM LISTING
-----------------

```

00001          *****
00002          *
00003          *   NAME : 16-BIT SQUARE ROOT   (SQRT)   *
00004          *
00005          *****
00006          *
00007          *           ENTRY : IX   (16-BIT BINARY DATA) *
00008          *           RETURNS : SANS (SQUARE ROOT)      *
00009          *
00010          *****
00011          *
00012A 0080          *           ORG   $B0
00013          *
00014A 0080          0002 A SANS  RMB   2   Square root
00015A 0082          0001 A SCNTR RMB   1   Shift Counter
00016          *
00017A F000          *           ORG   $F000
00018          *
00019          F000 A SQRT  EQU   *   Entry point
00020A F000 86 08 A      LDAA  #8   Set shift counter
00021A F002 97 82 A      STAA  SCNTR
00022A F004 4F          CLRA          Clear ACCD
00023A F005 5F          CLR B          CLR B
00024A F006 DD 80 A      STD   SANS   Clear square root area (SANS)
00025A F008 18          XGDX          0 -> IX , DATA->ACCD
00026A F009 05          ASLD          Rotate upper 2bits of data-
00027A F00A 18          XGDX          -to lower 2bits of ACCD
00028A F00B 59          ROL B
00029A F00C 49          ROLA
00030A F00D 18          XGDX
00031A F00E 05          ASLD
00032A F00F 18          XGDX
00033A F010 59          ROL B
00034A F011 49          ROLA
00035A F012 0D          SEC           Set LSB of SANS
00036A F013 79 0081 A   ROL   SANS+1
00037A F016 79 0080 A   ROL   SANS
00038A F019 93 80 A     SUBD  SANS   ACCD - SANS -> ACCD
00039A F01B 25 10 F02D BCS  SQRT3   Branch if minus
00040A F01D 7C 0081 A   INC   SANS+1  SANS + 1 -> SANS
00041A F020 7A 0082 A   SQRT2 DEC   SCNTR   Decrement shift counter
00042A F023 18          XGDX          ACCD <-> IX
00043A F024 26 E3 F009 BNE   SQRT1   Loop until shift counter = 0
00044A F026 77 0080 A   SQRT4 ASR   SANS   Halve SANS:SANS+1
00045A F029 76 0081 A   ROR   SANS+1
00046A F02C 39          RTS
00047A F02D D3 80 A     SQRT3 ADDD  SANS   Add again
00048A F02F 7A 0081 A   DEC   SANS+1  SANS - 1 -> SANS
00049A F032 20 EC F020 BRA   SQRT2   Branch SQRT2

```

**FUNCTION**

(a) Converts 2-byte hexadecimal number in RAM into 5-digit BCD number and stores result in RAM.

(b) Utilizes unsigned integers in arguments.

ARGUMENTS	CHANGES IN CPU REGISTERS AND FLAGS	SPECIFICATIONS																																															
<table border="1" style="width:100%; border-collapse: collapse;"> <thead> <tr> <th style="width:10%;"></th> <th style="width:15%;">Contents</th> <th style="width:15%;">Storage Location</th> <th style="width:10%;">Byte Lgth.</th> </tr> </thead> <tbody> <tr> <td rowspan="2" style="text-align: center; vertical-align: middle;">Arguments</td> <td style="text-align: center;">Entry</td> <td style="text-align: center;">2-byte hexa- decimal number  HEXD (RAM)</td> <td style="text-align: center;">2</td> </tr> <tr> <td style="text-align: center;">Re- turns</td> <td style="text-align: center;">5-digit BCD number  DECD (RAM)</td> <td style="text-align: center;">3</td> </tr> </tbody> </table>		Contents	Storage Location	Byte Lgth.	Arguments	Entry	2-byte hexa- decimal number  HEXD (RAM)	2	Re- turns	5-digit BCD number  DECD (RAM)	3	<p>● : Not affected × : Undefined ↓ : Result</p> <table border="1" style="width:100%; border-collapse: collapse; margin-bottom: 10px;"> <tr><th colspan="2" style="text-align: center;">ACCD</th></tr> <tr><th style="width:50%;">ACCA</th><th style="width:50%;">ACCB</th></tr> <tr><td style="text-align: center;">×</td><td style="text-align: center;">×</td></tr> <tr><th style="text-align: center;">IX</th><td></td></tr> <tr><td style="text-align: center;">×</td><td></td></tr> </table> <table border="1" style="width:100%; border-collapse: collapse;"> <tr><th style="width:50%;">C</th><th style="width:50%;">V</th></tr> <tr><td style="text-align: center;">×</td><td style="text-align: center;">×</td></tr> <tr><td style="text-align: center;">×</td><td style="text-align: center;">N</td></tr> <tr><td style="text-align: center;">×</td><td style="text-align: center;">×</td></tr> <tr><td style="text-align: center;">I</td><td style="text-align: center;">H</td></tr> <tr><td style="text-align: center;">●</td><td style="text-align: center;">×</td></tr> </table>	ACCD		ACCA	ACCB	×	×	IX		×		C	V	×	×	×	N	×	×	I	H	●	×	<table border="1" style="width:100%; border-collapse: collapse;"> <tr><td style="text-align: center;">ROM (Bytes)</td></tr> <tr><td style="text-align: center;">31</td></tr> <tr><td style="text-align: center;">RAM (Bytes)</td></tr> <tr><td style="text-align: center;">5</td></tr> <tr><td style="text-align: center;">Stack (Bytes)</td></tr> <tr><td style="text-align: center;">0</td></tr> <tr><td style="text-align: center;">No. of cycles</td></tr> <tr><td style="text-align: center;">1184</td></tr> <tr><td style="text-align: center;">Reentrant</td></tr> <tr><td style="text-align: center;">No</td></tr> <tr><td style="text-align: center;">Relocation</td></tr> <tr><td style="text-align: center;">No</td></tr> <tr><td style="text-align: center;">Interrupt</td></tr> <tr><td style="text-align: center;">Yes</td></tr> </table>	ROM (Bytes)	31	RAM (Bytes)	5	Stack (Bytes)	0	No. of cycles	1184	Reentrant	No	Relocation	No	Interrupt	Yes
	Contents	Storage Location	Byte Lgth.																																														
Arguments	Entry	2-byte hexa- decimal number  HEXD (RAM)	2																																														
	Re- turns	5-digit BCD number  DECD (RAM)	3																																														
ACCD																																																	
ACCA	ACCB																																																
×	×																																																
IX																																																	
×																																																	
C	V																																																
×	×																																																
×	N																																																
×	×																																																
I	H																																																
●	×																																																
ROM (Bytes)																																																	
31																																																	
RAM (Bytes)																																																	
5																																																	
Stack (Bytes)																																																	
0																																																	
No. of cycles																																																	
1184																																																	
Reentrant																																																	
No																																																	
Relocation																																																	
No																																																	
Interrupt																																																	
Yes																																																	

**DESCRIPTION**

(1) Function Details

(a) Argument details

HEXD : Holds 2-byte hexadecimal number (RAM) to be converted into BCD number.

DECD : Holds 5-digit BCD number. (RAM)

① Entry argument

HEXD (RAM) (\$CDFE)

DECD (RAM) (52734)

② Return argument

(b) Fig. 1 shows example of HEX execution.

If entry argument is as shown in part ① Fig. 1 Example of HEX execution of Fig. 1, 5-digit BCD number, in this case "52734" is held in DECD (RAM) (see part ② of Fig. 1).

**SPECIFICATIONS NOTES**



19. CONVERTING 2-BYTE HEXADECIMALS INTO 5-DIGIT BCD	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	HEX
---	---------	----------------------	-------	-----

DESCRIPTION

(5) Basic Operation

(a) 4-bit binary (ABCD) construction is shown in Fig. 2 (Formula 1, Formula 2).

$$\begin{aligned}
 ABCD &= A \times 2^3 + B \times 2^2 + C \times 2^1 + D \times 2^0 && \text{----- (Formula 1)} \\
 &= [ \{ (A \times 2) + B \} \times 2 + C ] \times 2 + D && \text{----- (Formula 2)}
 \end{aligned}$$

Fig. 2 4-bit binary (ABCD)

(b) 2-byte hexadecimal number can be converted into 5-digit BCD number by calculating (Formula 2).

First, calculate  $\alpha = (A \times 2) + B$ , and adjust result into decimal. Next, the same calculation is done for  $\beta = (\alpha \times 2) + C$ , and  $\gamma = (\beta \times 2) + D$ , both of which are adjusted into decimal.

(c) HEX uses HEXD (RAM) and DECD (RAM) for  $\alpha = (A \times 2) + B$

(i) Shifts 2-byte hexadecimal number (HEXD) 1 bit left and rotates MSB to bit C.

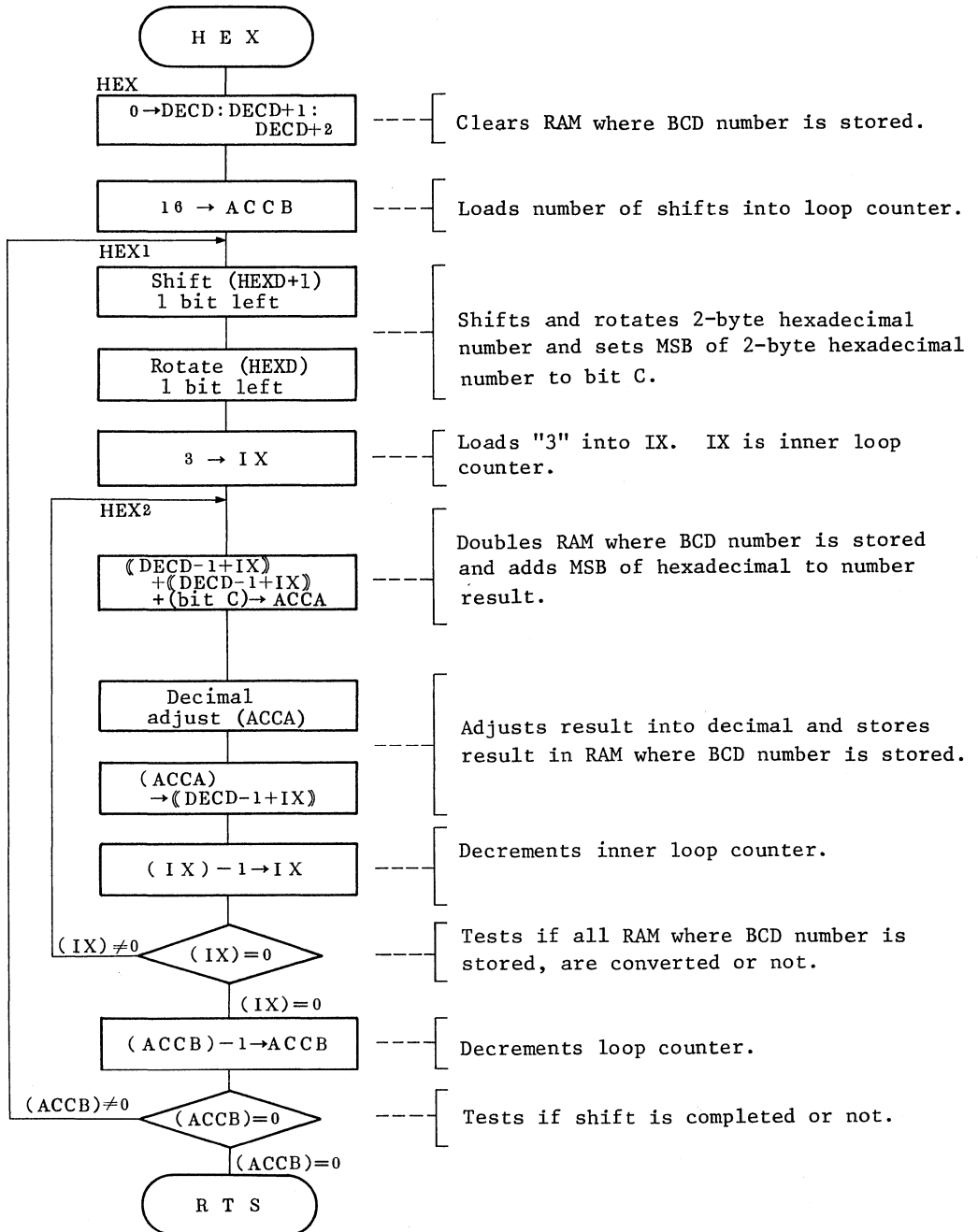
(ii) Loads 5-digit BCD number (DECD) into ACCA and calculates  $(ACCA) + (DECD) + (\text{bit C}) \rightarrow (ACCA)$ , where  $\alpha = (A \times 2) + B$  is executed.

(iii) Adjust result into decimal and stores result in DECD (RAM).

(iv) Loops (i) to (iii) 16 times to convert 2-byte hexadecimal number into 5-digit BCD number.

7

FLOWCHART



PROGRAM LISTING
-----------------

```

00001 *****
00002 *
00003 *      NAME : CONVERTING 2-BYTE HEXADECIMALS INTO *
00004 *              5-DIGIT BCD          (HEX)          *
00005 *
00006 *****
00007 *
00008 *      ENTRY :  HEXD (2-BYTE HEXADECIMAL)          *
00009 *      RETURNS : DECD (5-DIGIT BCD)                *
00010 *
00011 *****
00012
00013 *
00014A 0080          DRG      $80
00015 *
00016A 0080      0002  A  HEXD  RMB  2      2-byte hexadecimal
00017A 0082      0003  A  DECD  RMB  3      5-digit BCD
00018 *
00019A F000          DRG      $F000
00020 *
00021          F000  A  HEX      EQU      *      Entry point
00022A F000 4F          CLRA          Clear ACCA
00023A F001 5F          CLRB          Clear ACCB
00024A F002 0D 82      A      STD      DECD      Clear 5-digit BCD area
00025A F004 97 84      A      STAA     DECD+2
00026A F006 C6 10      A      LDAB     #16      Set shift counter
00027A F008 78 0081  A  HEX2  ASL      HEXD+1  Shift MSB of HEXD to carry
00028A F008 79 0080  A      ROL      HEXD
00029A F00E CE 0003  A      LDX      #3      Set ADDR pointer (addition counter)
00030A F011 A6 81      A  HEX1  LDAA     DECD-1.X  DECD * 2 + C -> ACCA
00031A F013 A9 81      A      ADCA     DECD-1.X
00032A F015 19          DAA          Convert into BCD
00033A F016 A7 81      A      STAA     DECD-1.X  Store 5-digit BCD area
00034A F018 09          DEX          Decrement ADDR pointer
00035A F019 26 F6 F011  BNE     HEX1   Loop until ADDR pointer = 0
00036A F01B 5A          DECB         Decrement shift counter
00037A F01C 26 EA F008  BNE     HEX2   Loop until shift counter = 0
00038A F01E 39          RTS

```

20. CONVERTING 5-DIGIT BCD INTO 2-BYTE HEXADECIMALS	MCU/MPU	HD6301/HD6303 FAMILY	LABEL	BCD
---	---------	----------------------	-------	-----

**FUNCTION**

(a) Converts 5-digit BCD number in RAM into 2-byte hexadecimal number and stores result in RAM.

(b) Utilizes insigned integers in arguments.

<b>ARGUMENTS</b>			<b>CHANGES IN CPU REGISTERS AND FLAGS</b>		<b>SPECIFICATIONS</b>																																																
<table border="1"> <thead> <tr> <th colspan="2">Contents</th> <th>Storage Location</th> <th>Byte Lgth.</th> </tr> </thead> <tbody> <tr> <td rowspan="2">Arguments</td> <td>Entry</td> <td>5-digit number DEC (RAM)</td> <td>3</td> </tr> <tr> <td>Returns</td> <td>2-byte hexadecimal number HDATA (RAM)</td> <td>2</td> </tr> </tbody> </table>			Contents		Storage Location	Byte Lgth.	Arguments	Entry	5-digit number DEC (RAM)	3	Returns	2-byte hexadecimal number HDATA (RAM)	2	<p>● : Not affected × : Undefined ↓ : Result</p> <table border="1"> <tr><th colspan="2">ACCD</th></tr> <tr><th>ACCA</th><th>ACCB</th></tr> <tr><td>×</td><td>×</td></tr> <tr><th>IX</th><td></td></tr> <tr><td>×</td><td></td></tr> </table> <table border="1"> <tr><th>C</th><th>V</th></tr> <tr><td>×</td><td>×</td></tr> <tr><th>Z</th><th>N</th></tr> <tr><td>×</td><td>×</td></tr> <tr><th>I</th><th>H</th></tr> <tr><td>●</td><td>●</td></tr> </table>		ACCD		ACCA	ACCB	×	×	IX		×		C	V	×	×	Z	N	×	×	I	H	●	●	<table border="1"> <tr><th>ROM (Bytes)</th></tr> <tr><td>59</td></tr> <tr><th>RAM (Bytes)</th></tr> <tr><td>6</td></tr> <tr><th>Stack (Bytes)</th></tr> <tr><td>0</td></tr> <tr><th>No. of cycles</th></tr> <tr><td>361</td></tr> <tr><th>Reentrant</th></tr> <tr><td>No</td></tr> <tr><th>Relocation</th></tr> <tr><td>No</td></tr> <tr><th>Interrupt</th></tr> <tr><td>Yes</td></tr> </table>		ROM (Bytes)	59	RAM (Bytes)	6	Stack (Bytes)	0	No. of cycles	361	Reentrant	No	Relocation	No	Interrupt	Yes
Contents		Storage Location	Byte Lgth.																																																		
Arguments	Entry	5-digit number DEC (RAM)	3																																																		
	Returns	2-byte hexadecimal number HDATA (RAM)	2																																																		
ACCD																																																					
ACCA	ACCB																																																				
×	×																																																				
IX																																																					
×																																																					
C	V																																																				
×	×																																																				
Z	N																																																				
×	×																																																				
I	H																																																				
●	●																																																				
ROM (Bytes)																																																					
59																																																					
RAM (Bytes)																																																					
6																																																					
Stack (Bytes)																																																					
0																																																					
No. of cycles																																																					
361																																																					
Reentrant																																																					
No																																																					
Relocation																																																					
No																																																					
Interrupt																																																					
Yes																																																					

**DESCRIPTION**

(1) Function Details

(a) Argument details

DEC : Holds 5-digit BCD number to be converted into hexadecimal.

HDATA: Holds 2-byte hexadecimal number.

(b) Fig. 1 shows example of BCD execution. If entry argument is as shown in part ① of Fig. 1, 2-byte hexadecimal number is stored in HDATA (RAM) as part ② of Fig. 1 shows.

RAM  
b7 b0

DEC	5
DEC+1	2
DEC+2	3

(decimal;52734)

b15 b0

HDATA (RAM)	C	D	F	E
HDATA	HDATA	HDATA+1		

(hexadecimal; \$CDFE)

**SPECIFICATIONS NOTES**

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to convert 59999 into hexadecimal.

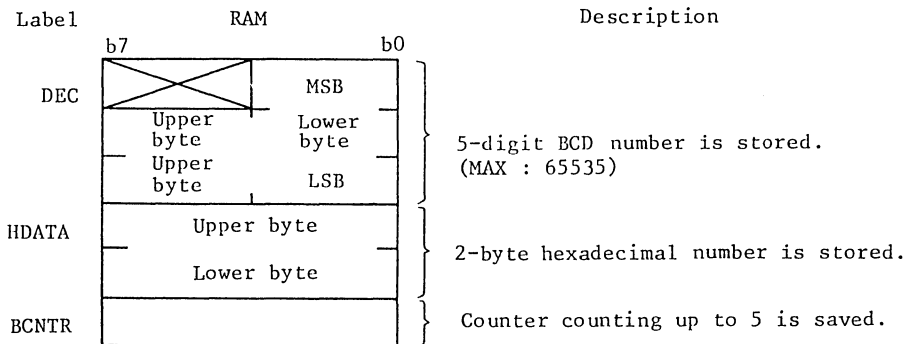


DESCRIPTION

(2) User Notes

- (a) Setting 5-digit BCD number greater than 65536 will yield in correct result.
- (b) Entry argument should be held as BCD number, otherwise result will not be correct.

(3) RAM Description



(4) Sample Application

Subroutine BCD is called after 5-digit BCD number is held.

```

WORK1   RMB   3   ----- Reserves memory byte for 5-digit BCD,
                               number.
WORK2   RMB   2   ----- Reserves memory byte for 2-byte
                               hexadecimal number.
      .
      .
      PSHA
      PSHB
      PSHX
      LDD   WORK1
      STD   DEC
      LDAA  WORK1+2
      STAA  DEC+2
      JSR   BCD
      LDD   HDATA
      STD   WORK2
      PULX
      PULB
      PULA
      .
      .
    
```

} ----- Saves register contents that will be destroyed by BCD execution.

} ----- Stores 5-digit BCD number into entry argument (DEC).

} ----- Calls subroutine BCD.

} ----- Stores 2-byte hexadecimal number (return argument (HDATA)) in RAM.

} ----- Restores register.



DESCRIPTION

(5) Basic Operation

(a) Subroutine BCD consists of 2 operations; one is to fetch 5-digit BCD, digit by digit as shown in Fig. 2, the other is to convert fetched data into hexadecimal by 4 bits units.

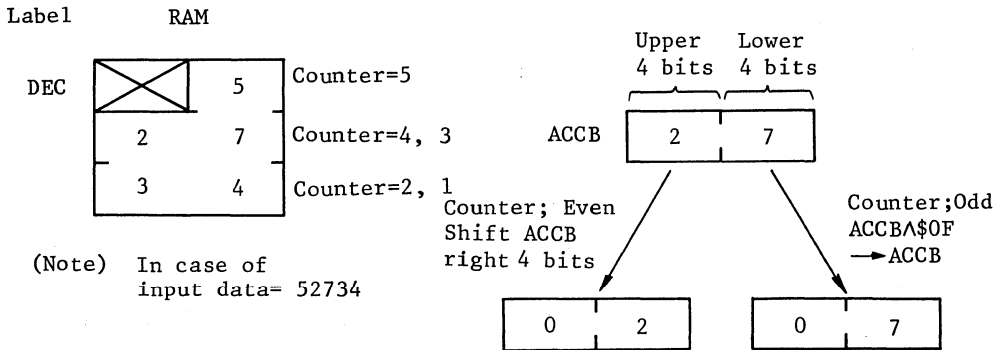


Fig. 2 Dividing 1 byte of RAM data into two parts

(b) Fetching (see Fig. 2)

(i) IX is used to indicate memory address of input 5-digit BCD number. Stores "5" in counter to convert 5 digits.

(ii) Loads input data into ACCB in order from MSB using index addressing mode and select upper or lower 4 bits.

(iii) Decrements counter every time one digit is loaded into ACCA.

(iv) Loops (ii) to (iii) until counter is "0".

(v) During (ii) and (iii), CPU checks whether counter is an even or an odd number.

If odd, AND ACCB to \$0F and fetch lower 4 bits. If even, shift ACCB 4 bits right and fetch upper 4 bits.

DESCRIPTION

(c) Converting BCD number into hexadecimal

(i) 4-digit BCD construction is shown in Fig. 3 (Formula 1), (Formula 2).

$$ABCD = A \times 10^3 + B \times 10^2 + C \times 10^1 + D \times 10^0 \text{ ---- (Formula 1)}$$

$$= \left[ \left\{ (A \times 10) + B \right\} \times 10 + C \right] \times 10 + D \text{ ---- (Formula 2)}$$

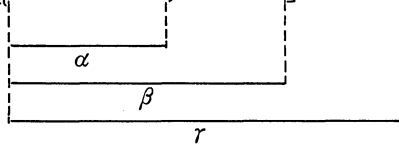


Fig. 3 4-digit BCD (ABCD)

(ii) 5-digit BCD number can be converted into hexadecimal as follows.

First, calculate  $\alpha = (A \times 10) + B$  to determine in Fig. 3 (Formula 2).

Then calculate  $\beta = (\alpha \times 10) + C$  and  $r = (\beta \times 10) + D$  to determine.

(iii) Calculation of  $A \times 10$  is shown in (Formula 3), (formula 4);

$$A \times 10 = A \times (2 + 8) \text{ -----(Formula 3)}$$

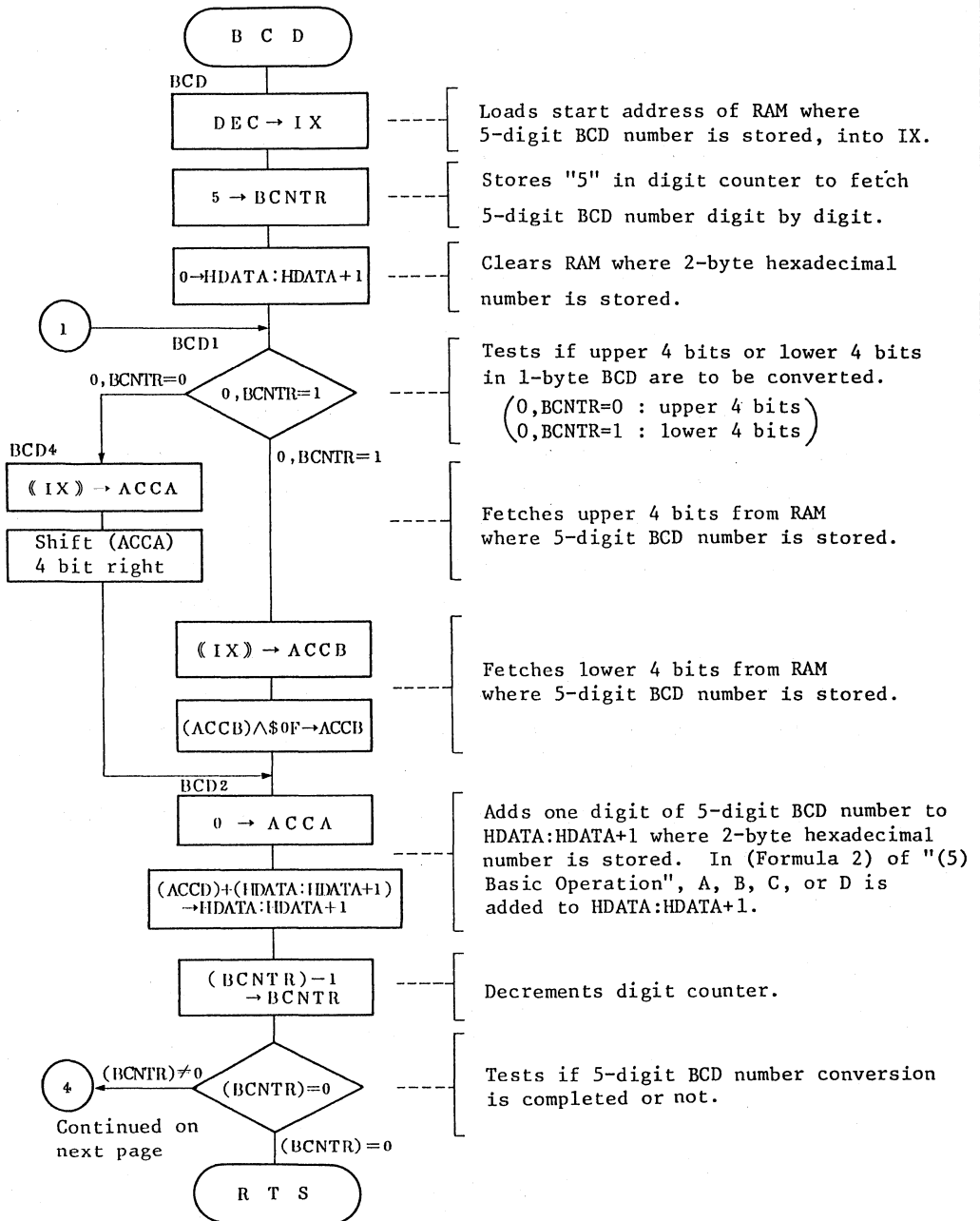
$$= A \times 2(1 + 2^2) \text{ -----(Formula 4)}$$

(iv) When calculating (Formula 4), BCD subroutine uses HDATA:H ATA+1(RA).

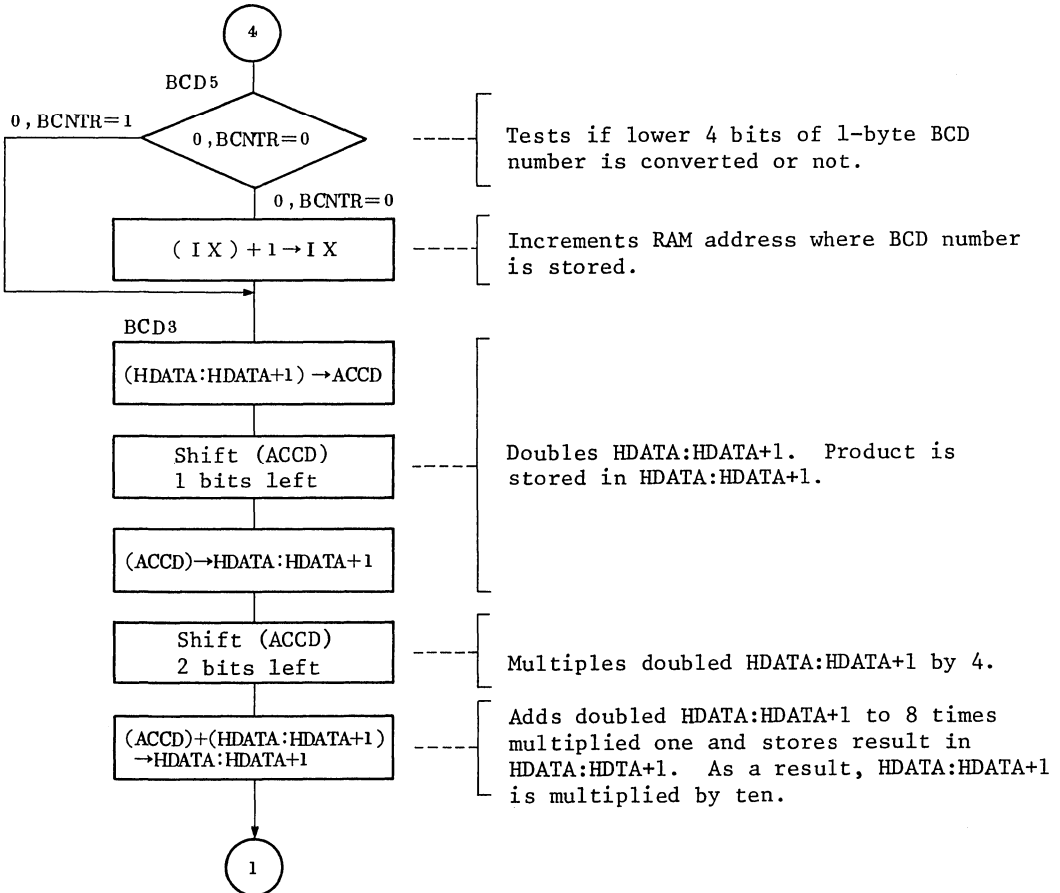
That is, store A in ACCD (Formula 4), shift A left 1 bit and store result in HDATA:HDATA+1. Next, shift ACCD left 2 bits and add ACCD to HDATA:HDATA+1 to determine  $A \times 10$ .

(d) Loops (b) and (c) 5 times to complete conversion of 5-digit BCD number into 16-bit binary number.

FLOWCHART



FLOWCHART



Continued on previous page

PROGRAM LISTING
-----------------

```

00001                *****
00002                *
00003                *   NAME :   CONVERTING 5-DIGIT BCD INTO   *
00004                *                   2-BYTE HEXADECIMALS   (BCD) *
00005                *
00006                *****
00007                *
00008                *   ENTRY :   DEC   (5-DIGIT BCD)         *
00009                *   RETURNS :  HDATA (2-BYTE HEXADECIMALS) *
00010                *
00011                *****
00012                *
00013A 0080          .          DRG   $80
00014                *
00015A 0080          0003 A DEC   RMB   3           5-digit BCD
00016A 0083          0002 A HDATA RMB   2           2-byte Hexadecimals
00017A 0085          0001 A BCNTR RMB   1           Digit counter
00018                *
00019A F000          .          DRG   $F000
00020                *
00021                F000 A BCD   EQU   *           Entry point
00022A F000 CE 0080 A          LDX   #DEC         Load BCD data ADDR
00023A F003 86 05 A          LDAA  #5           Set figure counter
00024A F005 97 85 A          STAA  BCNTR
00025A F007 7F 0083 A        CLR   HDATA         Clear Hex area
00026A F00A 7F 0084 A        CLR   HDATA+1
00027A F00D 7B 01 85 BCD1   BTST  0,BCNTR      Test bit0 of BCNTR
00028A F010 27 0F F021 A        BEQ   BCD4         Branch if bit0 = 0
00029A F012 E6 00 A          LDAB  0,X          Set lower 4 bits of BCD data
00030A F014 C4 0F A          ANDB  #$0F
00031A F016 4F          BCD2   CLRA          Clear ACCA for work use
00032A F017 D3 83 A          ADDD  HDATA        ACCD + HDATA:HDATA+1 -> HDATA:HDA
00033A F019 DD 83 A          STD   HDATA
00034A F01B 7A 0085 A        DEC   BCNTR        Decrement digit counter
00035A F01E 26 09 F029 A        BNE  BCD5         Loop until figure counter = 0
00036A F020 39          RTS
00037A F021 E6 00 A BCD4   LDAB  0,X          Set upper 4 bits of BCD data
00038A F023 54          LSRB
00039A F024 54          LSRB
00040A F025 54          LSRB
00041A F026 54          LSRB
00042A F027 20 ED F016 A        BRA  BCD2         Branch BCD2
00043A F029 7B 01 85 BCD5   BTST  0,BCNTR      Test bit0 of BCNTR
00044A F02C 26 01 F02F A        BNE  BCD3         Branch if bit0 = 1
00045A F02E 08          INX          Increment BCD data ADDR
00046A F02F DC 83 A BCD3   LOD   HDATA        HDATA:HDATA+1 * 2 -> ACCD
00047A F031 05          ASLD
00048A F032 DD 83 A          STD   HDATA        ACCD -> HDATA:HDATA+1
00049A F034 05          ASLD          HDATA:HDATA+1 * 4 -> ACCD
00050A F035 05          ASLD
00051A F036 D3 83 A          ADDD  HDATA        ACCD + HDATA:HDATA+1 -> ACCD
00052A F038 DD 83 A          STD   HDATA        ACCD -> HDATA:HDATA+1
00053A F03A 20 D1 FOOD A        BRA  BCD1         Branch BCD1

```

FUNCTION

- (a) Sorts unsigned byte oriented data in RAM in descending order.
- (b) Permits number of bytes to be sorted to be freely selected.
- (c) Utilizes unsigned integers in arguments.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	No. of bytes to be sorted	ACCA	1
		Starting address of data to be sorted	IX	2
	Returns	—	—	—

CHANGES IN CPU REGISTERS AND FLAGS

- : Not affected
- × : Undefined
- ↑ : Result

ACCD	
ACCA	ACCB
×	×
IX	
×	

C	V
×	×
Z	N
×	×
I	H
●	●

SPECIFICATIONS

ROM (Bytes)	22
RAM (Bytes)	7
Stack (Bytes)	2
No. of cycles	400
Reentrant	No
Relocation	No
Interrupt	Yes

DESCRIPTION

(1) Function Details

(a) Argument details

ACCA: Holds number of bytes to be sorted; (No. of bytes to be sorted - 1) in 1-byte hexadecimal number.

IX : Holds starting address of data to be sorted in 1-byte hexadecimal number.

① Entry arguments

② Result

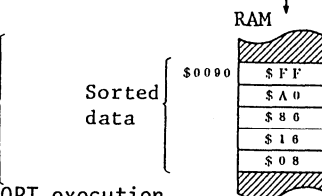
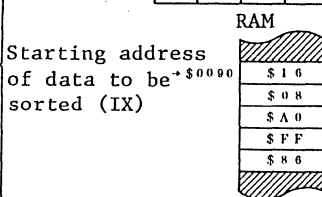
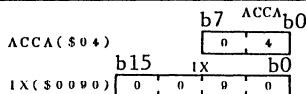


Fig. 1 Example of SORT execution

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to sort 5-byte ascending data to descending.

DESCRIPTION
-------------

(b) Fig. 1 shows example of SORT execution. If entry arguments are as shown in part ① of Fig. 1, sorted data is stored from address \$90 in descending order (see part ② of Fig. 1).  
 As data to be sorted is 5-byte, \$04 (No. of bytes to be sorted (\$05) - 1) is held in ACCA.

(2) User Notes

When loading number of bytes to be sorted, hold "No. of bytes to be sorted - 1" to ACCA for effective loop processing.

(3) RAM Description

Label	RAM	Description
	b7 <span style="margin-left: 100px;">b0</span>	
SCNT 1		} Counter showing how many bytes remain to be sorted is stored.
SCNT 2		
		} Counter showing how many bytes remain to be compared is stored.

(4) Sample Application

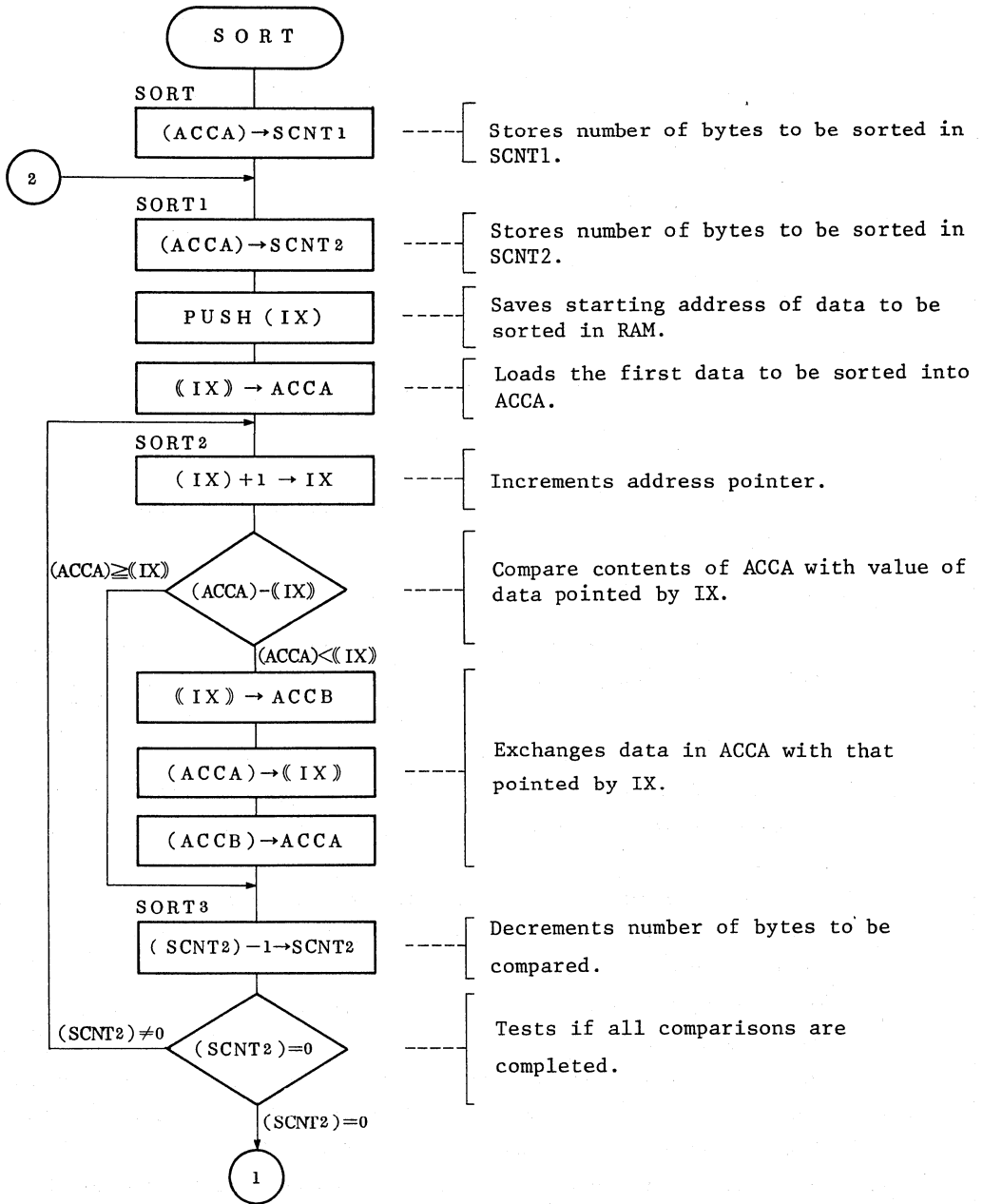
Subroutine SORT is called after starting address and number of bytes to be sorted are held.

WORK1	RMB	1	----	Reserves memory byte for number of bytes to be sorted.
WORK2	RMB	2	----	Reserves memory byte for starting address to be sorted.
	LDAA	WORK1	----	Loads number of bytes to be sorted into entry argument (ACCA).
	LDX	WORK2	----	Loads starting address of data to be sorted into entry argument (IX).
	JSR	SORT	----	Calls subroutine SORT.

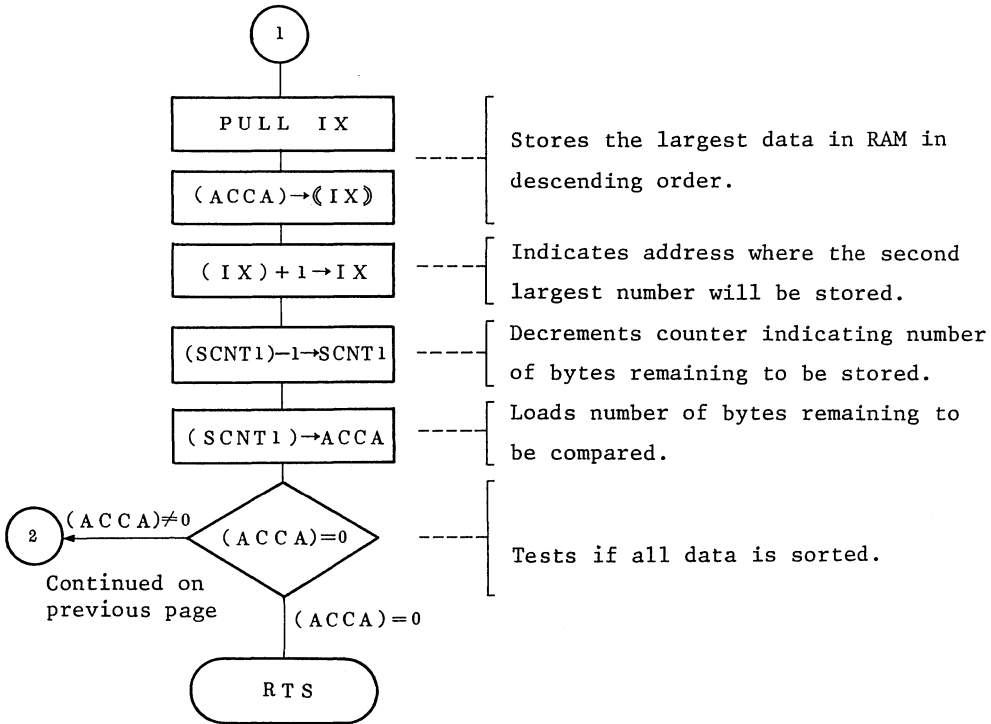




FLOWCHART



## FLOWCHART



PROGRAM LISTING
-----------------

```

00001          *****
00002          *
00003          *           NAME :  SORTING           (SORT)           *
00004          *
00005          *****
00006          *
00007          *           ENTRY :  ACCA (VOLUME OF SORTING DATA)      *
00008          *           IX   (TOP ADDR OF SORTING DATAS)            *
00009          *           RETURNS :  NOTHING                          *
00010          *
00011          *****
00012          *
00013A 0080          ORG   $80
00014          *
00015A 0080   0001  A SCNT1  RMB   1           Counter for sorting data
00016A 0081   0001  A SCNT2  RMB   1           Counter for comparing data
00017          *
00018A F000          ORG   $F000
00019          *
00020          F000  A SORT  EQU   *           Entry point
00021A F000 97 80   A      STAA  SCNT1      Store reste of sorting data
00022A F002 97 81   A SORT1 STAA  SCNT2      Store reste of comparing data
00023A F004 3C          PSHX          Push sorting data ADDR
00024A F005 A6 00   A      LDAA  0.X        Load sorting data
00025A F007 08          SORT2  INX          Set next sorting data ADDR
00026A F008 A1 00   A      CMPA  0.X        Compare comparing data with sorting data
00027A F00A 24 05 F011 BCC  SORT3        Branch if comparing data > sorting data
00028A F00C E6 00   A      LDAB  0.X        Exchange each data
00029A F00E A7 00   A      STAA  0.X
00030A F010 17          TBA          Transfer exchanged data
00031A F011 7A 0081 A SORT3  DEC  SCNT2      Decrement comparing data counter
00032A F014 26 F1 F007 BNE  SORT2      Loop until comparing data counter = 0
00033A F016 38          PULX          Pull sorting data ADDR
00034A F017 A7 00   A      STAA  0.X        Store max data
00035A F019 08          INX          Increment sorting data ADDR
00036A F01A 7A 0080 A      DEC  SCNT1      Decrement soting data counter
00037A F01D 96 80   A      LDAA  SCNT1      Loop until sorting data counter = 0
00038A F01F 26 E1 F002 BNE  SORT1
00039A F021 39          RTS

```

# HD6301/HD6303 SERIES HANDBOOK

## Section Eight

# Hardware Application Notes

## FOREWORD

The HD6301/HD6303 are CMOS 8-bit single chip microcomputers controlled by microprogramming. The HD6301/HD6303 provide 8-bit parallel handshake interfacing, pipeline control, halt and memory-ready functions for various kinds of data processing.

APPLICATION NOTES are written to help users design hardware systems using examples of simple application functions with circuit diagrams, timing charts and program examples.

Application examples in APPLICATION NOTES used in actual systems should be tested for proper operation.

NOTE

The following hardware application notes were prepared for HD6301Y0/HD6303Y devices. The applications, however, are generic in nature and also apply to HD6301V1/HD6303R and HD6301X0/HD6303X devices.

## Section 8

### Hardware Application Notes

### Table of Contents

	<b>Page</b>
APPLICATION NOTES GUIDE . . . . .	815
1. Symbols . . . . .	815
2. Application Example Configuration . . . . .	817
3. 1st Section (Hardware) . . . . .	819
4. 2nd Section (Software). . . . .	823
5. 3rd Section (Program Module) . . . . .	825
5.1 Specification . . . . .	826
5.2 Description. . . . .	832
5.3 Flowchart . . . . .	836
6. 4th Section (Subroutine) . . . . .	837
7. 5th Section (Program Listing) . . . . .	840
8. Program Module Execution . . . . .	845
SYSTEM APPLICATION EXAMPLES . . . . .	848
System Application Examples . . . . .	848
I/O PORT APPLICATIONS . . . . .	849
1. HD61830 (LM200) Graphic Mode . . . . .	849
2. Darlington Transistor Drive (LED Dynamic Display) . . . . .	883
TIMER APPLICATIONS . . . . .	896
3. Duty Control of Pulse Output and DA Conversion . . . . .	896
4. Pulse Width Measurement . . . . .	912
5. Input Pulse Count . . . . .	922
6. 8 × 4 Key Matrix . . . . .	932
INTERRUPT APPLICATION. . . . .	947
7. A/D Converter (HA16613A) Control . . . . .	947
PARALLEL HANDSHAKE APPLICATIONS . . . . .	956
8. Standard Keyboard Interface. . . . .	956
9. Centronics Interface. . . . .	970
SCI APPLICATIONS . . . . .	982
10. Data Transfer With Asynchronous SCI . . . . .	982
11. Liquid Crystal Drived (HD61100A) Control. . . . .	998

EXTERNAL EXPANSION APPLICATIONS ..... 1010

12. External Expansion ..... 1010

13. Slow Device Interface ..... 1043

LOW POWER DISSIPATION/FAIL-SAFE APPLICATIONS ..... 1057

14. Low Power Dissipation Mode. .... 1057

15. HA1835P Control and Error Detection ..... 1074





## 1. Symbols

Symbols and abbreviations used in APPLICATION NOTES are described below.

### 1.1 Operation

( ) = Contents  
(( )) = Index addressing  
a→b = Data transfer from a to b  
+ = Addition  
- = Subtraction  
× = Multiplication  
/ = Division  
^ = AND  
∨ = OR  
⊕ = Exclusive OR  
x̄ = NOT

### 1.2 Register Symbols in MCU/MPU

ACCA = Accumulator A  
ACCB = Accumulator B  
ACCD = Double accumulator (ACCA : ACCB)  
CCR = Condition code register  
I X = 8-bit, 16-bit index register  
IXH = Upper 8 bits of index register  
IXL = Lower 8 bits of index register.

### 1.3 Description of bits 0 through 5 of condition code register

C = Carry or borrow	bit 0
V = Overflow in 2's complement operation	bit 1
Z = Zero	bit 2
N = Negative	bit 3
I = Interrupt mask	bit 4
H = Half carry	bit 5

## 1.4 Others

=	=	Equal sign
≠	=	Not-equal sign
>	=	Greater than
<	=	Less than
≥	=	Greater than or equal
≤	=	Less than or equal
' ,	=	Delineates ASCII characters
\$	=	Hexadecimal data
:	=	Labels of successive addresses
SCI	=	Serial communication interface
DDR	=	Data direction register
FRC	=	Free running counter
OCR1	=	Output compare register 1
OCR2	=	Output compare register 2
ICR	=	Input capture register
TCSR1	=	Timer control/status register 1
TCSR2	=	Timer control/status register 2
TCSR3	=	Timer control/status register 3
RMCR	=	Transfer rate/mode control register
TRCSR	=	Tx/Rx control status register
RDR	=	Receive data register
TDR	=	Transmit data register
RP5CR	=	RAM/port 5 control register
TCONR	=	Time constant register
T2CNT	=	Timer 2 up counter

## 2. Application Example Configuration

This chapter explains the configuration of each system application example following this chapter.

Each application example in APPLICATION NOTES is divided into 5 sections, as shown in figure 1.

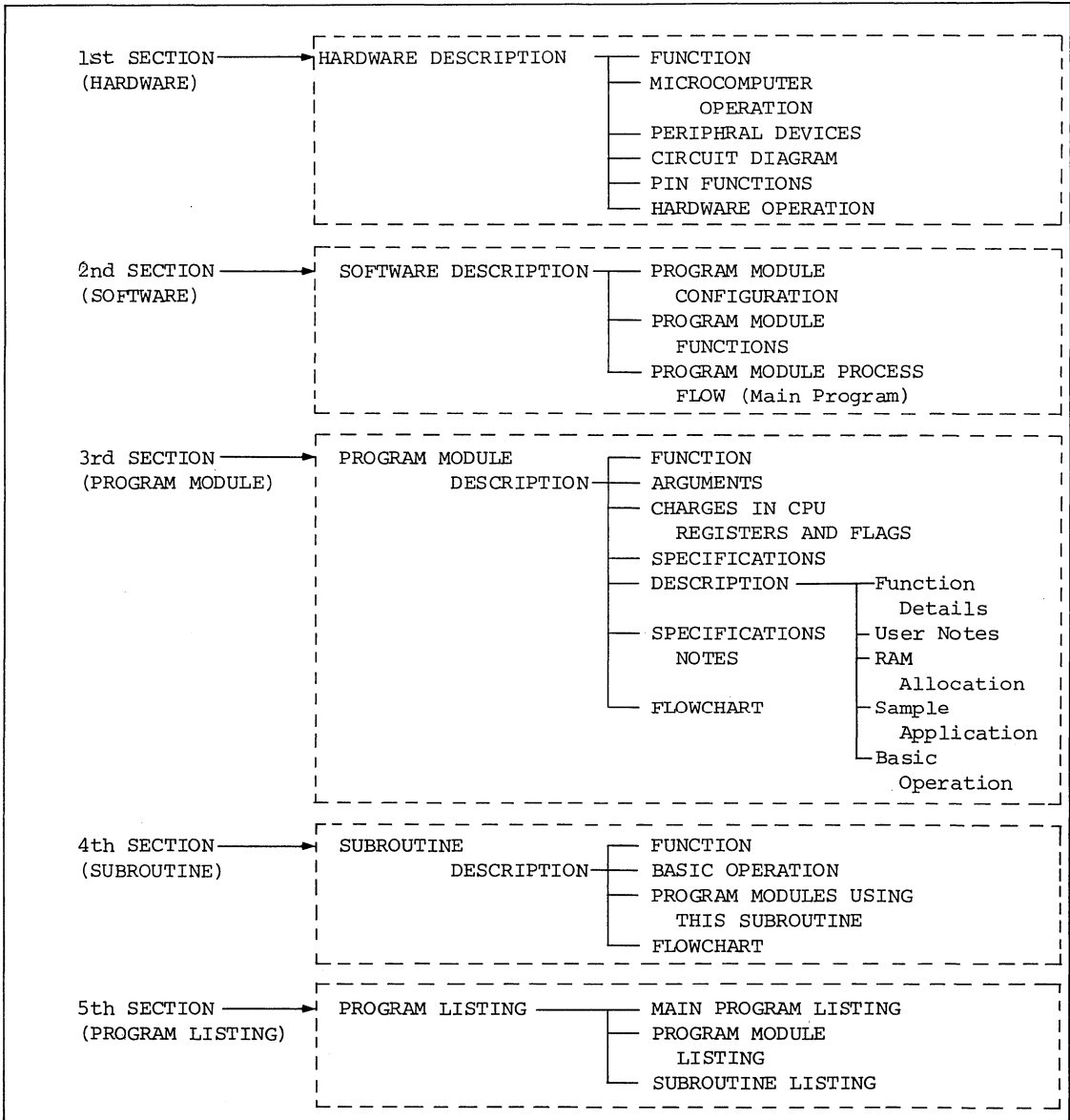


Figure 1. Application Example Configuration

(1) 1st Section (Hardware)

Describes functions, circuit diagram, hardware operation for each hardware application example and making specific use of HD6301Y0/HD6303Y's characteristic functions.

(2) 2nd Section (Software)

Describes program module configuration which controls hardware application example explained in the 1st Section. Also shows main program of sample application.

(3) 3rd Section (Program Module)

Describes program modules except main program, presented in the 2nd Section, in detail. Each program module is described in the same format so that users can use them independently.

(4) 4th Section (Subroutine)

Describes subroutine used by each program module. When using program modules explained in the 3rd Section, refer to these subroutines, if necessary.

(5) 5th Section (Program Listing)

Provides program listings for sample application explained in the 1st section.

A detailed explanation of all five sections follows.

### 3. 1st Section (Hardware)

#### 3.1 Function

Describes system specifications for the hardware used in a particular application.

Example:

##### 1.1.1 Function

Initializes graphic mode and displays dot graphics on the LM200 liquid crystal module.

#### 3.2 Microcomputer Operation

Describes typical functions of the microcomputer used in a particular application.

Example:

##### 1.1.2 Microcomputer Operation

The HD6301Y0 transfers display data to the dot matrix liquid crystal graphic display controller LSI HD61830 (LCTC) from port 3 onto the LCTC data bus ( $DB_0 \sim DB_7$ ), and transmits control signals E, R/W, and RS through port 1. Ports 1 and 3 are controlled by software.

#### 3.3 Peripheral Devices

Describes typical functions of the peripheral devices used in a particular application.

Example:

##### 1.1.3 Peripheral Devices

HD61830 LCTC: Receives control signals and display data from the HD6301Y0 and in turn controls the HM6116 Display RAM and LM200.

LM200 Liquid Crystal Module: Receives graphic display data and control signals from the HD61830 LCTC. A resolution of  $64 \times 240$  pixels is provided in LM200 graphic mode. In this application, the figures "日立", meaning HITACHI, are displayed.



### 3.5 Pin Functions

Describes interface between microcomputer and the external circuit using a table.

Example:

#### 1.1.5 Pin Functions

Pin functions at the interface between the HD6301Y0 and LCTC are shown in table 1-1.

Table 1-1. Pin Functions

Pin Name (HD6301Y0)	Input/Output	Active Level (High or Low)	Function	Pin Name (LCTC)	Program Label
P <sub>10</sub>	Output	High	Enables signal	E	P1DTR
P <sub>11</sub>	Output	High	Reads data	R/W	
		Low	Writes data		
P <sub>12</sub>	Output	High	Selects instruction register	RS	
		Low	Selects data register		
P <sub>30</sub>	Input/Output	—	Data Lines	DB <sub>0</sub>	P3DTR
P <sub>31</sub>	Input/Output	—		DB <sub>1</sub>	
P <sub>32</sub>	Input/Output	—		DB <sub>2</sub>	
P <sub>33</sub>	Input/Output	—		DB <sub>3</sub>	
P <sub>34</sub>	Input/Output	—		DB <sub>4</sub>	
P <sub>35</sub>	Input/Output	—		DB <sub>5</sub>	
P <sub>36</sub>	Input/Output	—		DB <sub>6</sub>	
P <sub>37</sub>	Input/Output	—		DB <sub>7</sub>	

"Active Level" in the table indicates the following:

High : Logical 1

Low : Logical 0

— : Logical 1 or 0



### 3.6 Hardware Operation

Describes hardware operation for controlling an external circuit using a timing chart.

Example:

#### 1.1.6 Hardware Operation

The timing chart for interfacing between the HD6301Y0 and each signal is shown in figure 1-2. ① and ② in figure 1-2 show timing for read and write.

- ① Data from LCTC can be read during ① period.
- ② Data can be written to LCTC at the falling edge of signal E.

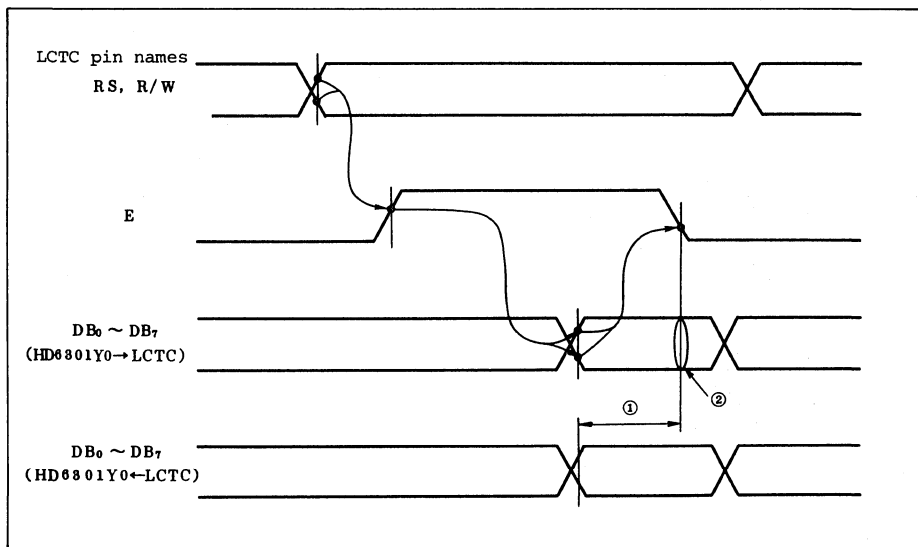


Figure 1-2. HD6301Y0 ↔ LCTC Interface

## 4. 2nd Section (Software)

### 4.1 Program Module Configuration

Describes program module configuration to control the hardware application example. Each program module is numbered. No. of main program is "0", and the other program modules are numbered from 1 to N.

Example:

#### 1.2.1 Program Module Configuration

The program module configuration for graphic display on the liquid crystal module is shown in figure 1-3.

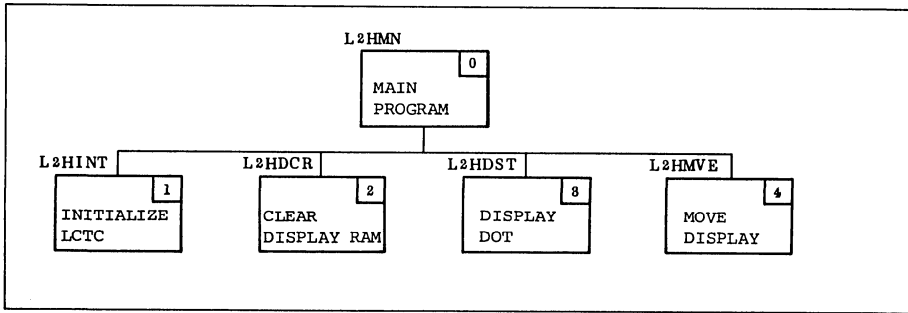


Figure 1-3. Program Module Configuration

### 4.2 Program Module Functions

Describes function of each program module using a table. "No." in the table matches "No." in the Program Module Configuration.

Example:

#### 1.2.2 Program Module Functions

Program module functions are summarized in table 1-2.

Table 1-2. Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	L2HMN	Demonstrates graphic display on LM200.
1	INITIALIZE LCTC	L2HINT	Initializes LCTC for graphic mode.
2	CLEAR DISPLAY RAM	L2HDCR	Clears display RAM to clear display.
3	DISPLAY DOT	L2HDST	Turns on and off 1 dot specified by row or column coordinate.
4	MOVE DISPLAY	L2HMVE	Moves dot display up, down, left, or right.

### 4.3 Program Module Process Flow (Main Program)

Describes sample main program to execute program modules, explained in

(1) Program Module Configuration.

Note) Stack pointer is initialized only in the main program.  
Example:

#### 1.2.3 Program Module Process Flow (Main Program)

The following flowchart (figure 1-4) demonstrates the process for displaying graphics on the LM200 liquid crystal display, using the modules described above. Figure 1-5 shows this applications display.

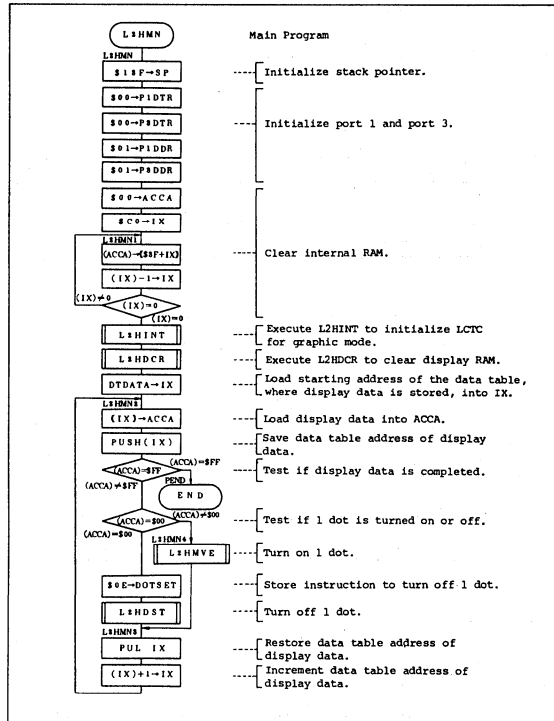


Figure 1-4. Main Program Flowchart

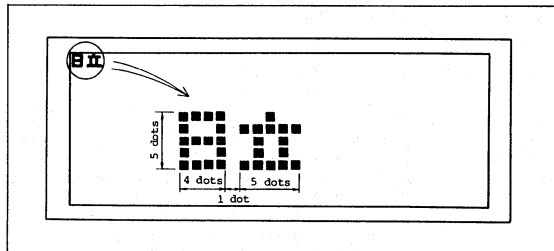
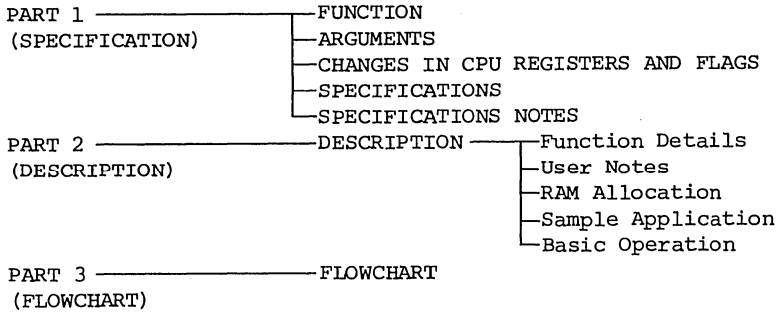


Figure 1-5. Example of L2HMN Execution

5. 3rd Section (Program Module)

The 3rd Section consists of the parts as shown in figure 2.



Program Module Name:	MCU/MPU:	Label:
----------------------	----------	--------

Flowchart:

Program Module Name:	MCU/MPU:	Label:
----------------------	----------	--------

Description:

Program Module Name:	MCU/MPU:	Label:
----------------------	----------	--------

<u>Function:</u>		

<u>Arguments:</u>	<u>Changes in CPU Registers and Flags:</u>	<u>Specifications:</u>																																			
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 30%;">Contents</th> <th style="width: 30%;">Storage Location</th> <th style="width: 40%;">No. of Bytes</th> </tr> </thead> <tbody> <tr> <td>Entry</td> <td></td> <td></td> </tr> <tr> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>Re-returns</td> <td></td> <td></td> </tr> <tr> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> </tr> </tbody> </table>	Contents	Storage Location	No. of Bytes	Entry									Re-returns									<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">ACCD</td> <td style="text-align: center;">ACCB</td> </tr> <tr> <td style="text-align: center;">ACCA</td> <td style="text-align: center;">ACCB</td> </tr> <tr> <td style="text-align: center;">IX</td> <td style="text-align: center;">IX</td> </tr> <tr> <td style="text-align: center;">C</td> <td style="text-align: center;">V</td> </tr> <tr> <td style="text-align: center;">Z</td> <td style="text-align: center;">N</td> </tr> <tr> <td style="text-align: center;">I</td> <td style="text-align: center;">H</td> </tr> <tr> <td style="text-align: center;">I</td> <td style="text-align: center;">H</td> </tr> </table> <p style="font-size: small; margin-top: 5px;">● : Not affected            * : Undefined            † : Result</p>	ACCD	ACCB	ACCA	ACCB	IX	IX	C	V	Z	N	I	H	I	H	ROM (Bytes): RAM (Bytes): Stack (Bytes): No. of cycles: Reentrant: Relocatable: Interrupt OK?:
Contents	Storage Location	No. of Bytes																																			
Entry																																					
Re-returns																																					
ACCD	ACCB																																				
ACCA	ACCB																																				
IX	IX																																				
C	V																																				
Z	N																																				
I	H																																				
I	H																																				

Description:

--

Specifications Notes:

--

Figure 2. Program Module Section

## 5.1 Specification

The Specification Part is shown in figure 3 ([ ]): blocked off area in figure 3). This part explains function, arguments, changes in CPU registers and flags, specifications and specifications notes. Each numbered item in the figure is described below.

(1) Program Module Name:

(2) MCU/MPU:

(3) Label:

(4) Function:

(5) Arguments:

(6) Changes in CPU Registers and Flags:

(7) Specifications:

(8) Specifications Notes:

Contents	Storage Location	No. of Bytes
Entry		
Re- turns		

**Changes in CPU Registers and Flags:**

ACCD  
ACCA ACCB

--	--

IX

--	--

C      V

--	--

Z      N

--	--

I      H

--	--

● : Not affected  
 × : Undefined  
 † : Result

**Specifications:**

ROM (Bytes):

RAM (Bytes):

Stack (Bytes):

No. of cycles:

Reentrant:

Relocatable:

Interrupt OK?:

**Description:**

1. Function Details

2. User Notes

Figure 3. Specification Part

(1) PROGRAM MODULE NAME:

Example:

Program Module Name: DISPLAY DOT

(2) MCU/MPU: Indicates microcomputer and microprocessor applicable to the program.

Example:

MCU/MPU: HD6301Y0

(3) LABEL: Indicates the name identifying program entry point. When using the program without modification, use this label to call the program.

Example:

Label: L2HDST

(4) FUNCTION: Describes program function.

Example:

Function:

Turns on or off 1 dot specified by row and column coordinates in entry arguments.

(5) ARGUMENTS: Describes entry arguments necessary to execute a program, and return arguments resulting from Program execution.

(a) Contents: Describes contents of entry and return arguments.

(b) Storage Location: Describes registers and RAM in which entry and return arguments are set. In case of RAM, the storage location is denoted by a label followed by "(RAM)".

(c) No. of Bytes: Describes number of bytes for entry and return arguments.

Example:

Arguments:		
Contents	Storage Location	No. of Bytes
Entry Turn on/off indicator	DOTSET (RAM)	1
Dot column coordinate	DTX (RAM)	1
Dot row coordinate	DTY (RAM)	1
Re- turns	_____	_____

(6) CHANGES IN CPU REGISTERS AND FLAGS: Describes changes in CPU registers and flags after program execution. Symbols and abbreviations in the table are shown below.

(a) CPU registers.

ACCA: Accumulator A

ACCB: Accumulator B

ACCD: Double accumulator (ACCA : ACCB)

IX : Index register

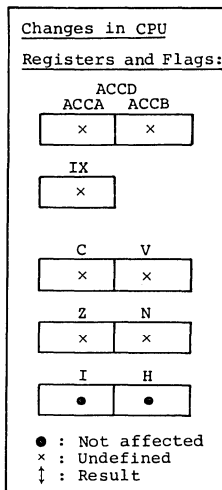
(b) Flags in condition code register

- C : Carry or borrow
- V : Overflow in 2's complement operation
- Z : Zero
- N : Negative
- I : Interrupt mask
- H : Half carry

(c) Status of CPU registers and condition code flags

- : Not affected : Previous values retained after program execution.
- × : Undefined : Previous values destroyed after program execution.
- ↑ : Result : Program result contained.

Example:



Note)

In this example, ACCA, ACCB, IX and bit C, bit V, bit N, bit Z of CCR are destroyed after program execution. Thus, registers and flags, which will be destroyed, should be saved before execution, if necessary.



- (7) SPECIFICATIONS: Describes program module specifications.
- (a) ROM (Bytes): Indicates amount of ROM used in the program module.
  - (b) RAM (Bytes): Indicates amount of RAM used in the program module. RAM used by the stack, however, is not included.
  - (c) Stack (Bytes): Indicates stack size used in the program module. Stack size used by called subroutines however, is not included. When executing a program module, the total stack size must be reserved.
  - (d) No. of cycles: Indicates the maximum number of cycles when executing a program module. Execution time of the program module can be calculated using No. of cycles as follows.  
$$\text{Execution time (sec)} = \text{No. of cycles} \times \text{Cycle time}$$
$$\text{Cycle time (sec)} = 4 / (\text{External oscillator (Hz)})$$
  - (e) Reentrant : Indicates whether or not the program module can be called by two or more programs at the same time.
  - (f) Relocatable : Indicates whether or not the program module can be located in other memory space.
  - (g) Interrupt OK?: Indicates whether or not the program module can be interrupted by other programs. If "No", disable interrupts before program execution and enable them after.

Example:

<u>Specifications:</u>
ROM (Bytes): 134
RAM (Bytes): 9
Stack (Bytes): 6
No. of cycles: 513
Reentrant: No
Relocatable: No
Interrupt OK?: Yes

(8) SPECIFICATIONS NOTES: Describes notes on items listed in "(7) SPECIFICATIONS".

Example:

<u>Specifications Notes:</u>
1. Values in "Specifications" include values for subroutines called by L2HDST.
2. "No. of cycles" in "Specifications" indicates the number of cycles required when L2HBSY is executed in the minimum number of cycles (no waiting for LM200).

## 5.2 Description

The Description Part is shown in figure 4. ( [ ] : blocked off area in figure 4). This part explains function details, user notes, RAM allocation, sample application and basic operation. Each numbered item in the figure is described below.

(1) ↓	(2) ↓	(3) ↓	(1) ↓	(2) ↓	(3) ↓																		
Program Module Name:	MCU/MPU:	Label:	Program Module Name:	MCU/MPU:	Label:																		
Function:			Description: 3. RAM Allocation ← (4)  4. Sample Application  5. Basic Operation																				
Arguments:	Changes in CPU Registers and Flags:	Specifications:																					
<table style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Storage Location</th> <th style="text-align: left;">No. of Bytes</th> </tr> </thead> <tbody> <tr> <td>Entry</td> <td></td> </tr> <tr> <td>Re- turns</td> <td></td> </tr> </tbody> </table>	Storage Location	No. of Bytes				Entry		Re- turns		<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">ACCD</td> <td style="text-align: center;">ACCB</td> </tr> <tr> <td style="text-align: center;">ACCA</td> <td style="text-align: center;">ACCB</td> </tr> <tr> <td style="text-align: center;">IX</td> <td></td> </tr> <tr> <td style="text-align: center;">C</td> <td style="text-align: center;">V</td> </tr> <tr> <td style="text-align: center;">Z</td> <td style="text-align: center;">N</td> </tr> <tr> <td style="text-align: center;">I</td> <td style="text-align: center;">H</td> </tr> </table> <p>● : Not Affected                  * : Undefined                  † : Result</p>	ACCD	ACCB	ACCA	ACCB	IX		C	V	Z	N	I	H	ROM (Bytes): RAM (Bytes): Stack (Bytes): No. of cycles: Reentrant: Relocatable: Interrupt OK?:
Storage Location	No. of Bytes																						
Entry																							
Re- turns																							
ACCD	ACCB																						
ACCA	ACCB																						
IX																							
C	V																						
Z	N																						
I	H																						
Description: 1. Function Details 2. User Notes ← (4)																							
Specifications Notes:																							

Figure 4. Description Part

- |   |   |                             |
|---|---|-----------------------------|
| (1) PROGRAM MODULE NAME<br>(2) MCU/MPU<br>(3) LABEL | } | Same as "5.1 Specification" |
|---|---|-----------------------------|

(4) DESCRIPTION: Describes function details, user notes, RAM allocation, sample application and basic operation of the program module.

(a) Function Details: Describes detailed functions of the program module referring to the execution example.

Example:

1. Function Details

a. Argument details

DOTSET(RAM): Data to indicate turning on or off 1 dot.

DOTSET(RAM)=\$0E : Turn off 1 dot.

DOTSET(RAM)=\$0F : Turn on 1 dot.

DTX(RAM) : Dot column coordinate in hexadecimal number.

DTY(RAM) : Dot row coordinate in hexadecimal number.

b. Example of L2HDST execution is shown in figure 1-6. If entry arguments are as shown in part ① of figure 1-6, 1 dot is displayed as shown in part ② of figure 1-6.

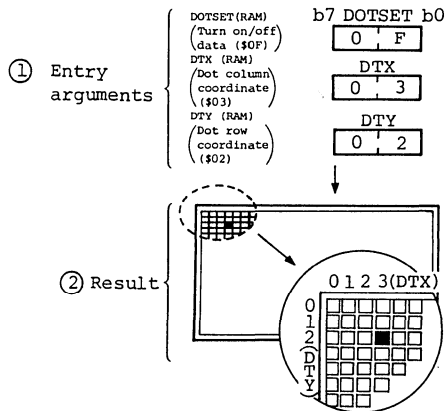


Figure 1-6. Example of L2HDST Execution

c. L2HDST calls subroutines shown in table 1-6.

Table 1-6. Subroutines Called by L2HDST

Subroutine Name	Label	Function
Store Cursor Address	L2HCST	Stores LCTC cursor address.
Continuous Display	L2HIST	Stores data in LCTC instruction register and data register.
Check Busy Flag	L2HBSY	Checks LCTC busy flag.

- (b) User Notes: Describes notes and limitations when executing the program module.

Be sure to read these items when using program modules without modification.

Example:

## 2. User Notes

The following procedure must be executed before L2HDST execution.

- a. Select DDR of port 1 and port 3 as output.
- b. Initialize LCTC display mode.
- c. Store entry arguments.

- (c) RAM Allocation: Describes labels and contents of RAM used in program module.

Example:

## 3. RAM Allocation

Label	RAM		Description
	b7	b0	
INSTR			Data to be written to LCTC instruction register
DATAR			Data to be written to LCTC data register
DTX			Dot column coordinate
DTY			Dot row coordinate
CURH			Upper byte of cursor address
CURL			Lower byte of cursor address
CCNT			Work area for calculating cursor address based on column coordinate
DTWK			Work area for obtaining 1 dot to be turned on/off
DOTSET			Data to indicate turning on/off 1 dot



### 5.3 Flowchart

The Flowchart Part is shown in figure 5. This part gives the program module flowchart.

Example:

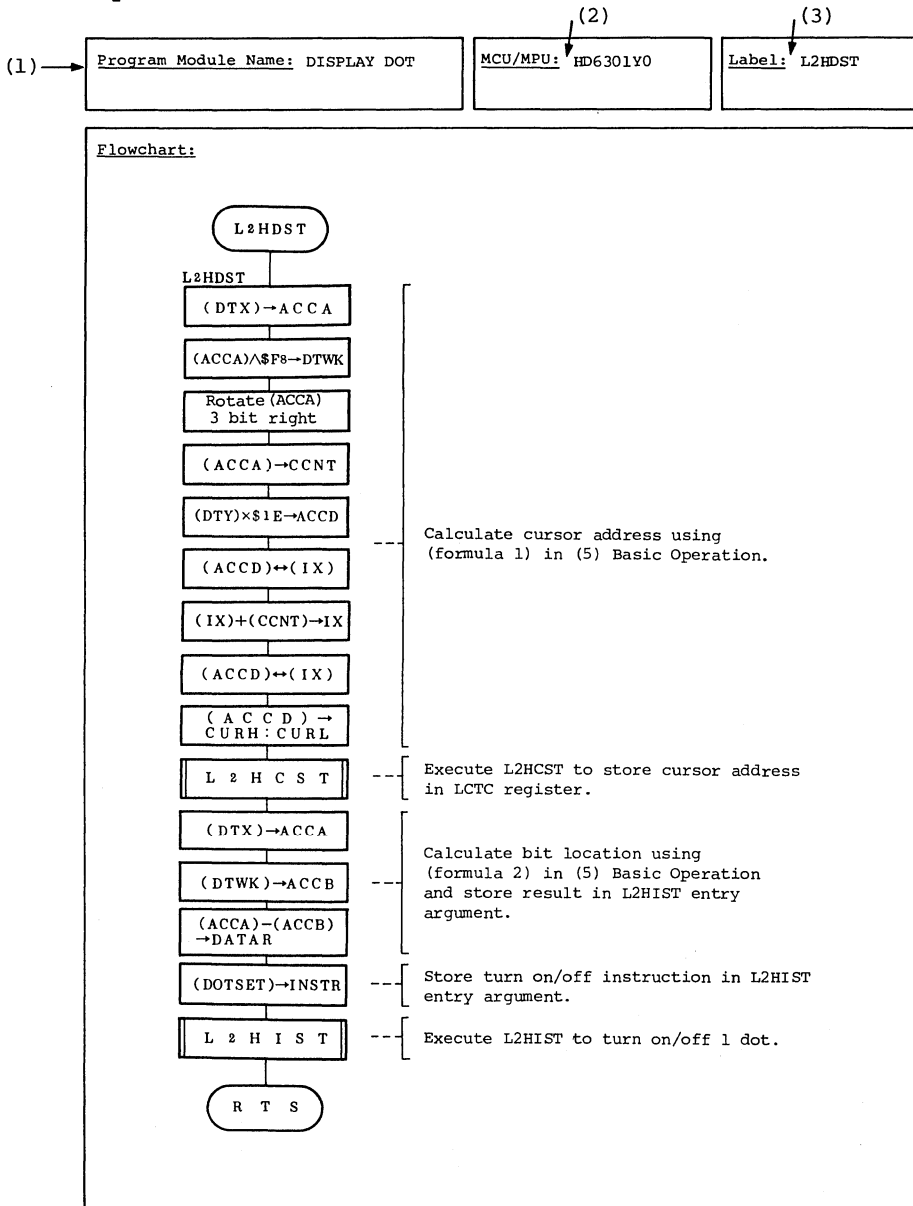


Figure 5. Flowchart Part

- |                         |                               |
|-------------------------|-------------------------------|
| (1) PROGRAM MODULE NAME | } Same as "5.1 Specification" |
| (2) MCU/MPU             |                               |
| (3) LABEL               |                               |

## 6. 4th Section (Subroutine)

The Subroutine Section is shown in figure 6. Each numbered item is described as follows.

(1) →	<u>Subroutine Name:</u>	(2) ↓ <u>MCU/MPU:</u>	(3) ↓ <u>Label:</u>
(4) →	<u>Function:</u>		
(5) →	<u>Basic Operation:</u>		
(6) →	<u>Program Module Using This Subroutine:</u>		
(7) →	<u>Flowchart:</u>		

Figure 6. Subroutine Section



(1) SUBROUTINE NAME:

Example:

Subroutine Name: STORE DISPLAY  
INSTRUCTION

- (2) MCU/MPU: Indicates microcomputer or microprocessor applicable to the subroutine.

Example:

MCU/MPU: HD6301Y0

- (3) LABEL: Indicates the name identifying subroutine entry point.  
When using the subroutine without modification, use this label to call the subroutine.

Example:

Label: L2HIST

- (4) FUNCTION: Describes subroutine function.

Example:

Function:  
Writes instruction and data to LCTC.

- (5) BASIC OPERATION: Explains how a subroutine is executed.

Example:

Basic Operation:  
1. LCTC busy flag is checked.  
2. Data is written to LCTC through port 1 controlling DS, R/W, E signals of LCTC.

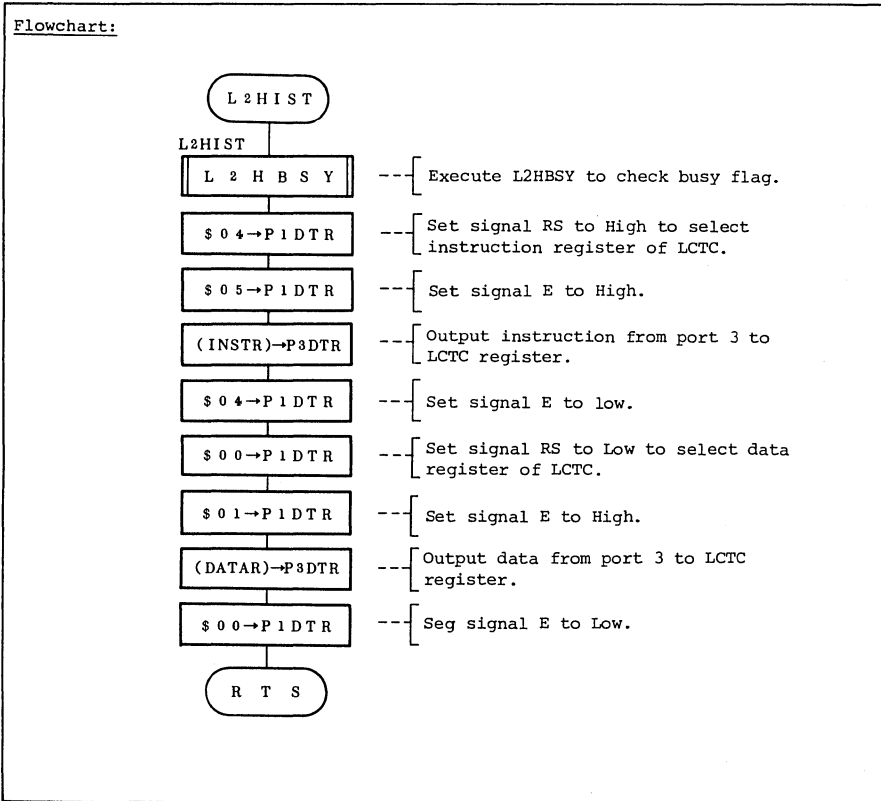
- (6) PROGRAM MODULES USING THIS SUBROUTINE: Lists program modules using the subroutine.

Example:

Program Module Using This Subroutine: L2HINT, L2HDCR, L2HDST, L2HMVE

- (7) FLOWCHART: Gives subroutine flowchart.

Example:



7. 5th Section (Program Listing)

The Program Listing Section explains RAM allocation and CPU register allocation, and gives program module and subroutine listings.

- (1) RAM Allocation: RAM used in program modules or subroutines is allocated as shown below.

Example:

```

00001
00002          (a) { *
00003             *
00004A 0040    (b) { *
00005             *
00006A 0040    0001 A INSTR RMB 1      LCTC instruction register data
00007A 0041    0001 A DATAR RMB 1      LCTC data register data
00008A 0042    0001 A CURL  RMB 1      Lower byte of cursor address
00009A 0043    0001 A CURH  RMB 1      Upper byte of cursor address
00010A 0044    0002 A DCOUNT RMB 2      Counter for continuous display
00011A 0046    0001 A DATA RMB 1      Display data
00012A 0047    0001 A CCNT  RMB 1      Work area for cursor address
00013A 0048    0001 A DTX   RMB 1      Dot column coordinate
00014A 0049    0001 A DTY   RMB 1      Dot row coordinate
00015A 004A    0001 A DTWK  RMB 1      Work area for turning on/off data
00016A 004B    0001 A DOTSET RMB 1     Turning on/off data

```

- (a) The title "RAM ALLOCATION" is followed by the actual RAM allocation used.

(b) RAM label.

- (2) CPU Register Allocation: Symbols used in a program module or subroutine are defined as shown below.

Example:

```

00017
00018          (a) { *
00019             *
00020          (b) { *
00021             *
00022    0000 A P1DDR EQU $00      Port 1 data direction register
00023    0002 A P1DTR EQU $02      Port 1 data register
00024    0004 A P3DDR EQU $04      Port 3 data direction register
00025    0006 A P3DTR EQU $06      Port 3 data register

```

- (a) The title is always "SYMBOL DEFINITIONS".

(b) Symbol definitions.

(3) Main Program: Gives main program listing of a sample application.

Example:

```

00024
00025
00026
00027
00028
00029
00030A C000
00031
00032A C000 8E 013F A L2HMN
00033A C003 4F
00034A C004 97 02 A
00035A C006 97 06 A
00036A C008 86 01 A
00037A C00A 97 00 A
00038A C00C 97 04 A
00039A C00E 4F
00040A C00F CE 00C0 A
00041A C012 A7 3F A L2HMN1
00042A C014 09
00043A C015 26 FB C012
00044A C017 8D C052 A
00045A C01A BD C03D A
00046A C01D CE C14E A
00047A C020 A6 00 A L2HMN2
00048A C022 3C
00049A C023 81 FF A
00050A C025 27 14 C03B
00051A C027 81 00 A
00052A C029 26 0B C036
00053A C02B 86 0E A
00054A C02D 97 4B A
00055A C02F BD C068 A
00056A C032 38 L2HMN3
00057A C033 0B
00058A C034 20 EA C020
00059A C036 BD C093 A L2HMN4
00060A C039 20 F7 C032
00061A C03B 20 FE C03B PEND

```

\*\*\*\*\*  
\*  
\* MAIN PROGRAM : L2HMN  
\*  
\*\*\*\*\*

(a) {

(b) ↘

ORG \$C000

\* L2HMN LDS #13F Initialize stack pointer

CLRA

STAA P1DTR Initialize port 1

STAA P3DTR Initialize port 3

LDAA #101 Select port 3 as output

STAA P1DDR

STAA P3DDR

CLRA Clear RAM

LDX #C0

STAA \$3F.X

DEX

BNE L2HMN1

JSR L2HINT Clear display RAM

JSR L2HDCR Initialize LCTC

LDX #0TDATA Load data table starting address

LDAA 0.X Load display data into ACCA

PSHX Save data table address

CMPA #FFF

PEND Loop until display is completed

#00 Test if 1 dot is turned on/off

BNE L2HMN4 Branch if turned on

LDAA #00E Store instruction to turn off

STAA DOTSET

JSR L2HDST Turn off 1 dot

L2HMN3 PULX Restore data table address

INX Increment data table address

L2HMN2

L2HMN4 JSR L2HMVE Turn on 1 dot

L2HMN3

PEND End of program

(a) The title is always "MAIN PROGRAM". Label after colon shows entry point label.

(b) Entry point label.

(4) Program Module: Gives program module listing of a sample application.

Example:

```
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112
00113A C068 96 48 A L2HDST LDAA DTX Load column coordinate
00114A C06A 84 FB A ANDA #FB DTX AND $FB->DTWK
00115A C06C 97 4A A STAA DTWK
00116A C06E 44 LSRA (DTX AND $FB)/8->CCNT
00117A C06F 44 LSRA
00118A C070 44 LSRA
00119A C071 97 47 A STAA CCNT
00120A C073 96 49 A LDAA DTY DTY*30->IX
00121A C075 C6 1E A LDAB #$1E
00122A C077 3D MUL
00123A C078 18 XGDX
00124A C079 06 47 A LDAB CCNT IX+CCNT->CURH:CURL
00125A C07B 3A ABX
00126A C07C 18 XGDX
00127A C07D 97 43 A STAA CURH
00128A C07F 07 42 A STAB CURL
00129A C081 8D C125 A JSR L2HCST Store cursor address
00130A C084 96 48 A LDAA DTX
00131A C086 06 4A A LDAB DTWK
00132A C088 10 SBA DTX-DTWK->DATAR
00133A C089 97 41 A STAA DATAR
00134A C08B 96 48 A LDAA DOTSET Store turning on/off data
00135A C08D 97 40 A STAA INSTR
00136A C08F 8D C103 A JSR L2HIST Turn on/off 1 dot
00137A C092 39 RTS
```

(a) {

```
*****
*
* NAME : L2HDST (DISPLAY DOT)
*
*****
*
* ENTRY : DTX(DOT COLUMN COORDINATE)
*          DTY(DOT ROW COORDINATE)
*          DOTSET(TURN ON/OFF INDICATOR)
* RETURNS : NOTHING
*
*****
```

(b) {

(a) Program module title is always followed by the entry point label in parenthesis and description of entry and return arguments.

(b) Entry point label.

(5) Subroutine: Gives subroutine listing.

Example:

```

00217
00218
00219
00220
00221
00222A C103 BD COEA A L2HIST JSR L2HBSY Check LCTC busy flag
00223A C106 86 04 A LDAA H$04 Set RS=1,R/W=0,E=0
00224A C108 97 02 A STAA P1DTR
00225A C10A 86 05 A LDAA H$05 Set E=1
00226A C10C 97 02 A STAA P1DTR
00227A C10E 96 40 A LDAA INSTR Output instruction through port3
00228A C110 97 06 A STAA P3DTR
00229A C112 86 04 A LDAA H$04 Set E=0
00230A C114 97 02 A STAA P1DTR
00231A C116 7F 0002 A CLR P1DTR Set RS=0
00232A C119 86 01 A LDAA H$01 Set E=1
00233A C11B 97 02 A STAA P1DTR
00234A C11D 96 41 A LDAA DATAR Output data through port3
00235A C11F 97 06 A STAA P3DTR
00236A C121 7F 0002 A CLR P1DTR Set E=0
00237A C124 39 RTS

```

(a) {

(b) L2HIST

(a) Subroutine title is followed by the entry point label in parenthesis.

(b) Entry Point label.

(6) Data Table: Describes data table used in the main program, program modules and subroutines.

Example:

```

00254
00255
00256
00257
00258
00259A C13C 00 A LCTC1 FCB $0,$32 *Instruction and data-
00260A C13E 01 A FCB $1,$07 to initialize LCTC
00261A C140 02 A FCB $2,$1D
00262A C142 03 A FCB $3,$1F
00263A C144 04 A FCB $4,$00
00264A C146 08 A FCB $8,$00
00265A C148 09 A FCB $9,$00
00266A C14A 0A A FCB $A,$00
00267A C14C 0B A FCB $B,$00
00268A C14E 01 A DTDATA FCB $01 *Display data
00269A C14F 01 A FCB $01
00270A C150 01 A FCB $01

```

(a) {

(b) LCTC1

(a) The title is always "DATA TABLE".

(b) Data table label.

(7) Vector Address: Describes vector address allocation.

Example:

```
00311
00312
00313
00314
00315
00316
00317A FFEA
00318
00319A FFEA C000 A
00320A FFEC C000 A
00321A FFEE C000 A
00322A FFF0 C000 A
00323A FFF2 C000 A
00324A FFF4 C000 A
00325A FFF6 C000 A
00326A FFF8 C000 A
00327A FFFA C000 A
00328A FFFC C000 A
00329A FFFE C000 A
00330
00331
```

```
*****
*
*          VECTOR ADDRESSES          *
*
*****
*
*          ORG      $FFEA
*
*          FDB      L2HMN      IRQ2
*          FDB      L2HMN      CMI
*          FDB      L2HMN      TRAP
*          FDB      L2HMN      SID
*          FDB      L2HMN      TOI
*          FDB      L2HMN      OCI
*          FDB      L2HMN      ICI
*          FDB      L2HMN      IRQ1/ISF
*          FDB      L2HMN      SWI
*          FDB      L2HMN      NMI
*          FDB      L2HMN      RES
*
*          (b)
*          END
```

(a) The title is always "VECTOR ADDRESSES".

(b) Indicates the end of a program. This can be moved, if necessary.

## 8. Program Module Execution

The programs in APPLICATION NOTES have been written considering efficiency and portability. The following shows how to execute these programs and how to modify them according to user requirements.

The procedure for calling programs in APPLICATION NOTES from user programs is shown in figure 7. All programs in APPLICATION NOTES are written as subroutines and should be called as shown. An example of a user program in which a program in APPLICATION NOTES is called as a subroutine is shown in figure 8.

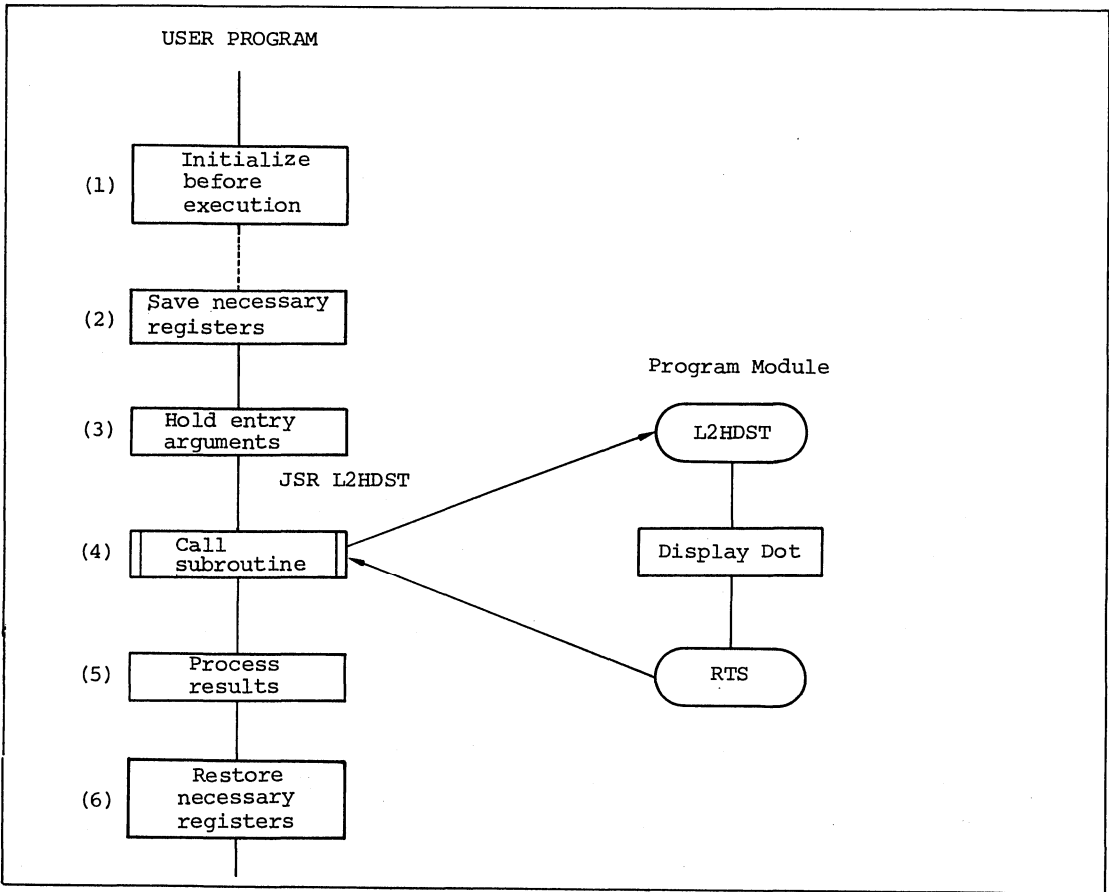


Figure 7. Procedure for Calling Program Module in APPLICATION NOTES



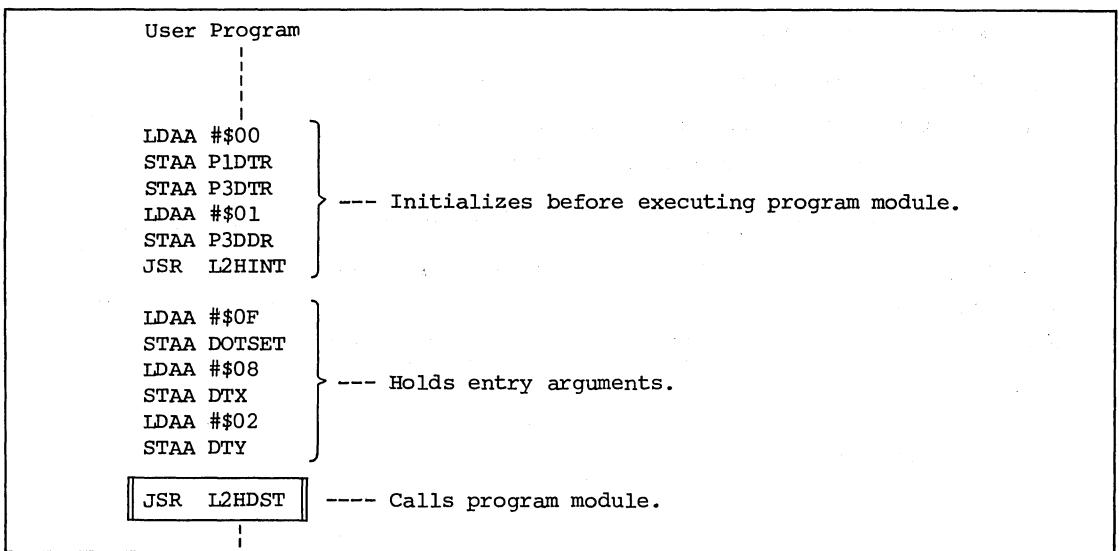


Figure 8. Example of How to Execute a Program Module

Explanation of figure 7.

(1) Initialize

Examples of items requiring initialization are input/output ports, control registers and counters used by the program module. Refer to Program Module Sample Application for data details.

(2) Save registers

The program modules use CPU registers for arithmetic operations destroying the original contents. Thus register contents should be saved if needed. Refer to the "CHANGES OF CPU REGISTERS AND FLAGS" column in "SPECIFICATIONS" (Part 1 in the 3rd SECTION-PROGRAM MODULE DESCRIPTION) for register status after a program module is executed.

(3) Hold entry arguments

Holds entry arguments in CPU registers or memory before calling a program module in the user program. Refer to "ARGUMENTS" in SPECIFICATIONS (Format 1 in 3RD SECTION - Program Module) for details.

(4) Call subroutine

Program module is called.

(5) Process result

After a program module is executed, the results returned in the return arguments must be processed as required. Refer to "ARGUMENTS" in

SPECIFICATIONS (Format 1 in 3RD SECTION - Program Module) for details.

(6) Restore necessary registers

Registers saved in (2) should be restored here. Note that when a program module is used as a subroutine, the stack area shown in SPECIFICATIONS (Format 1 in 3RD SECTION - Program Module) is necessary in addition to the stack area required by the subroutine calls in the user program. When any subroutine is called, this stack area must be reserved.

## System Application Examples

No.	Item	Microcomputer	Function	Device	Page
1	HD61830 (LM200) Graphic Mode	HD6301Y0	I/O Port (Port1) (Port3)	HD6301Y0 HD61830 MM6116 LM200	39
2	Darlington Transistor Drive (LED Dynamic Display)	HD6301Y0	I/O Port (Port1) (Port6)	HD6301Y0 8-digit×8-segment LED	73
3	Duty Control of Pulse Output and DA Conversion	HD6301Y0 (HD6303Y)	Timer2 Tout 3 pin	HD6301Y0	86
4	Pulse Width Measurement	HD6301Y0 (HD6303Y)	Timer 1 Tin Pin	HD6301Y0	102
5	Input Pulse Count	HD6301Y0 (HD6303Y)	Timer 2 TCLK pin	HD6301Y0	112
6	8 × 4 Key Matrix	HD6301Y0	I/O Port (Port3) (Port4)  Timer 1	HD6301Y0 8×4 key matrix	122
7	A/D Converter (HA16613A) Control	HD6301Y0	I/O port (port3) (port5)  IRQ <sub>1</sub> pin	HD6301Y0 HA16613A	137
8	Standard Keyboard Interface	HD6301Y0 (HD6303Y)	I/O port (port6) IS pin	HD6301Y0 ASCII keyboard	146
9	Centronics Interface	HD6301Y0 (HD6303Y)	I/O port (port6) IS pin, OS pin	HD6301Y0 Centronics interface printer	160
10	Data Transfer with Asynchronous SCI	HD6301Y0 (HD6303Y)	I/O port (port5) Asynchronous SCI	HD6301Y0 Console typewriter	172
11	Liquid Crystal Driven (HD61100A) Control	HD6301Y0 (HD6303Y)	I/O port (port2) Clock synchronous SCI	HD6301Y0 HD61100A 10-digit×8-segment LCD	188
12	External Expansion	HD6301Y0 (HD6303Y)	External expansion function	HD6301Y0 HD6321, HN27C64 HD6350, HM6264 H2571	200
13	Slow Device Interface	HD63B01Y0	MR pin  External expansion function	HD6301Y0 HN482764G-3 HM6264LP	233
14	Low Power Dissipation Mode	HD6301Y0	Low power dis- sipation mode (standby) (sleep)  I/O port (port1) (port3) (port6)	HD6301Y0	247
15	HA1835P Control and Error Detection	HD6301Y0	Trap function I/O port (port5) (port7)	HD6301Y0 HA1835P	264

1.1 HARDWARE DESCRIPTION

1.1.1 Function

Initializes graphic mode and displays dot graphics on the LM200 liquid crystal module.

1.1.2 Microcomputer Operation

The HD6301Y0 transfers display data to the dot matrix liquid crystal graphic display controller LSI HD61830 (LCTC) from port 3 onto the LCTC data bus (DB<sub>0</sub> ~ DB<sub>7</sub>), and transmits control signals E, R/W, and RS through port 1. Ports 1 and 3 are controlled by software.

1.1.3 Peripheral Devices

HD61830 LCTC: Receives control signals and display data from the HD6301Y0 and in turn controls the HM6116 Display RAM and LM200.

LM200 Liquid Crystal Module: Receives graphic display data and control signals from the HD61830 LCTC. A resolution of 64 × 240 pixels is provided in LM200 graphic mode. In this application, the figures "日立", meaning HITACHI, are displayed.

### 1.1.4 Circuit Diagram

LCTC control circuit is shown in figure 1-1.

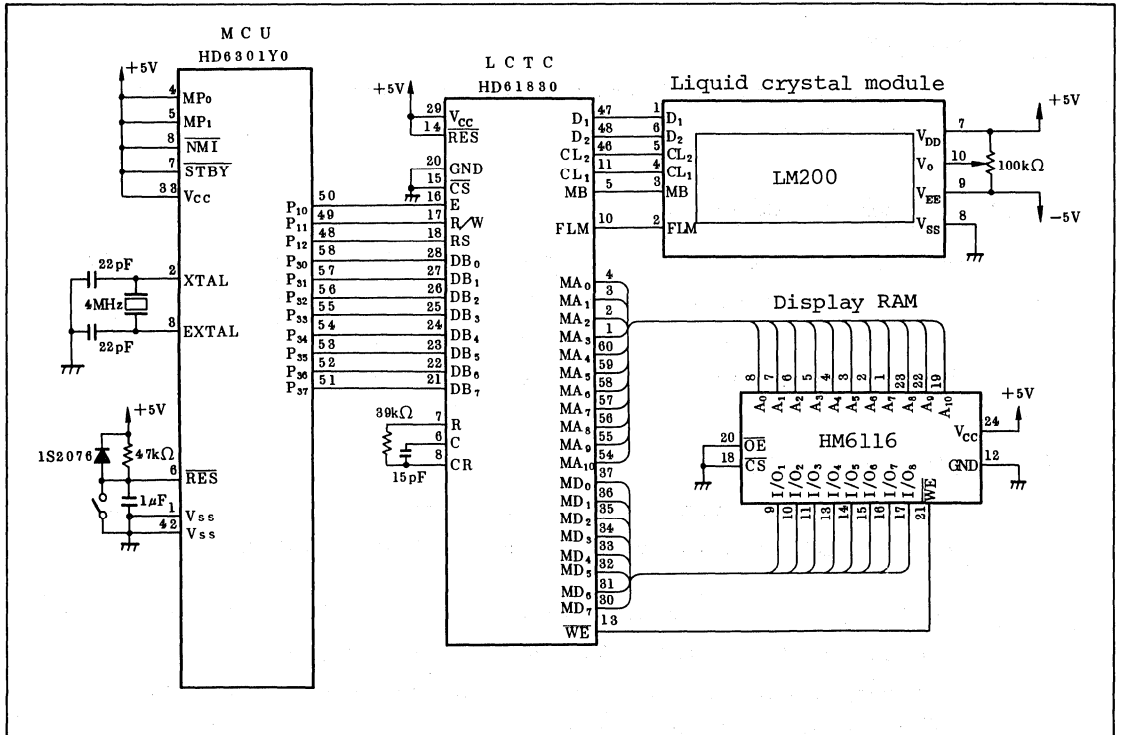


Figure 1-1. LCTC Control Circuit

### 1.1.5 Pin Functions

Pin functions at the interface between the HD6301Y0 and LCTC are shown in table 1-1.

Table 1-1. Pin Functions

Pin Name (HD6301Y0)	Input/Output	Active Level (High or Low)	Function	Pin Name (LCTC)	Program Label
P <sub>10</sub>	Output	High	Enables signal	E	P1DTR
P <sub>11</sub>	Output	High	Reads data	R/W	
		Low	Writes data		
P <sub>12</sub>	Output	High	Selects instruction register	RS	
		Low	Selects data register		
P <sub>30</sub>	Input/Output	—	Data Lines	DB <sub>0</sub>	P3DTR
P <sub>31</sub>	Input/Output	—		DB <sub>1</sub>	
P <sub>32</sub>	Input/Output	—		DB <sub>2</sub>	
P <sub>33</sub>	Input/Output	—		DB <sub>3</sub>	
P <sub>34</sub>	Input/Output	—		DB <sub>4</sub>	
P <sub>35</sub>	Input/Output	—		DB <sub>5</sub>	
P <sub>36</sub>	Input/Output	—		DB <sub>6</sub>	
P <sub>37</sub>	Input/Output	—		DB <sub>7</sub>	

### 1.1.6 Hardware Operation

The timing chart for interfacing between the HD6301Y0 and each signal is shown in figure 1-2. ① and ② in figure 1-2 show timing for read and write.

- ① Data from LCTC can be read during ① period.
- ② Data can be written to LCTC at the falling edge of signal E.

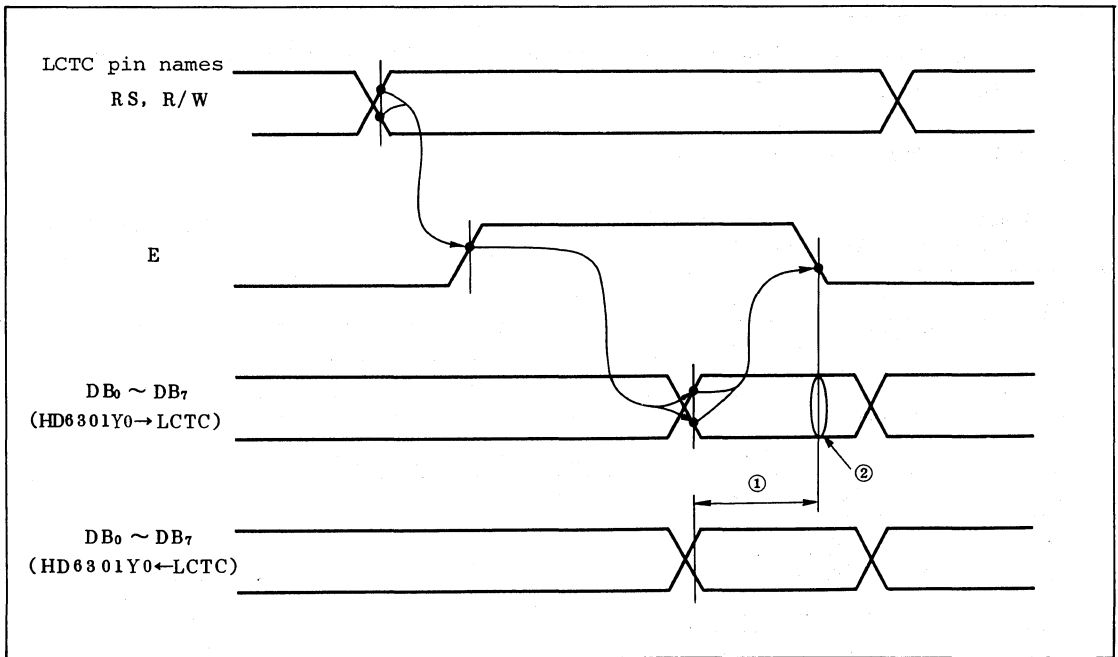


Figure 1-2. HD6301Y0  $\leftrightarrow$  LCTC Interface

## 1.2 SOFTWARE DESCRIPTION

### 1.2.1 Program Module Configuration

The program module configuration for graphic display on the liquid crystal module is shown in figure 1-3.

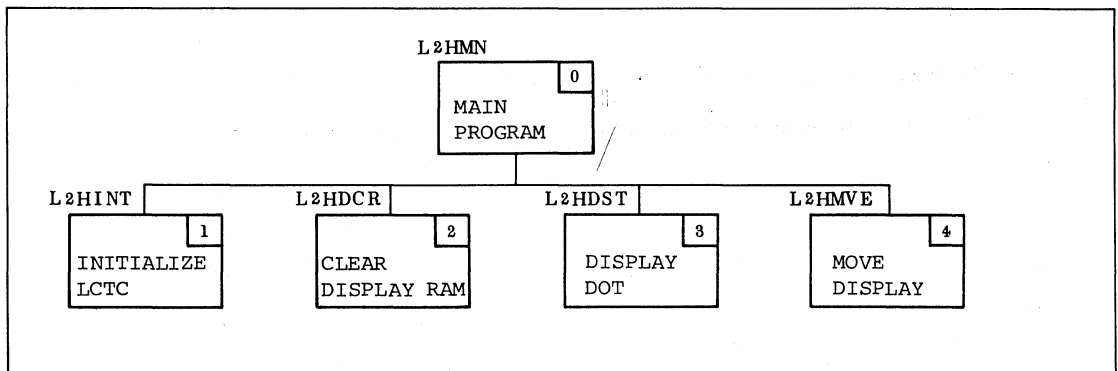


Figure 1-3. Program Module Configuration

### 1.2.2 Program Module Functions

Program module functions are summarized in table 1-2.

Table 1-2. Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	L2HMN	Demonstrates graphic display on LM200.
1	INITIALIZE LCTC	L2HINT	Initializes LCTC for graphic mode.
2	CLEAR DISPLAY RAM	L2HDCL	Clears display RAM to clear display.
3	DISPLAY DOT	L2HDST	Turns on and off 1 dot specified by row or column coordinate.
4	MOVE DISPLAY	L2HMVE	Moves dot display up, down, left, or right.

### 1.2.3 Program Module Process Flow (Main Program)

The following flowchart (figure 1-4) demonstrates the process for displaying graphics on the LM200 liquid crystal display, using the modules described above. Figure 1-5 shows this applications display.



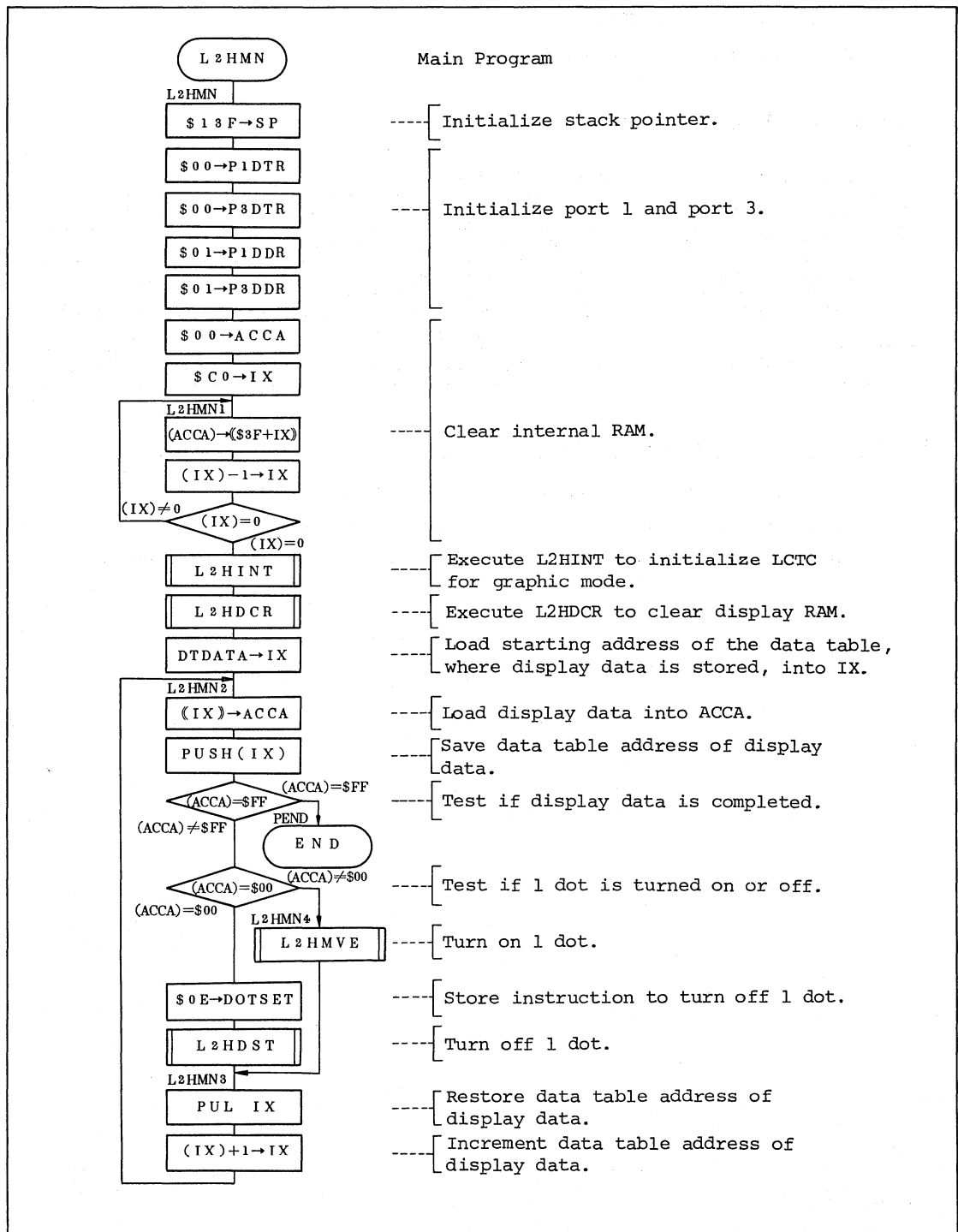


Figure 1-4. Main Program Flowchart

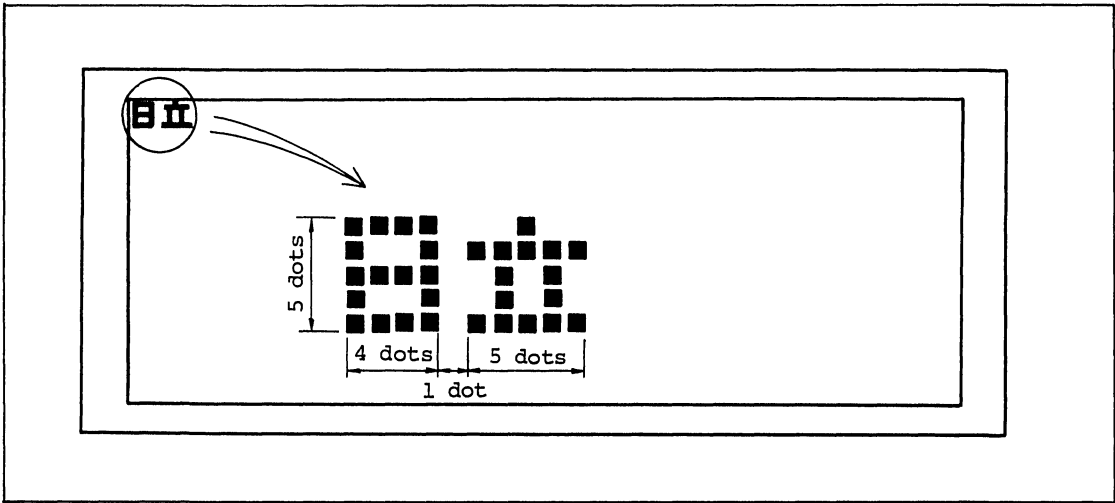


Figure 1-5. Example of L2HMN Execution

### 1.3 PROGRAM MODULE DESCRIPTION

Program Module Name: INITIALIZE LCTC

MCU/MPU: HD6301Y0

Label: L2HINT

Function:

Initializes LCTC for graphic mode.

Arguments:

None

Changes in CPU

Registers and Flags:

ACCD	
ACCA	ACCB
x	●

IX
x

C	V
x	x

Z	N
x	x

I	H
●	●

● : Not affected  
x : Undefined  
↑ : Result

Specifications:

ROM (Bytes): 90  
RAM (Bytes): 2  
Stack (Bytes): 4  
No. of cycles: 123  
Reentrant: No  
Relocatable: No  
Interrupt OK?: Yes

Description:

1. Function Details

- a. L2HINT has no arguments.
- b. Instruction and data in table 1-3 are written to LCTC to initialize LCTC for graphic mode.

Specifications Notes:

1. Values in "Specifications" include values for subroutines called by L2HINT.
2. "No. of cycles" in "Specifications" indicates the number of cycles required when L2HBSY is executed in the minimum number of cycles (no waiting for LM200).

Program Module Name: INITIALIZE LCTC

MCU/MPU: HD6301Y0

Label: L2HINT

Description:

Table 1-3. Instruction and Data to initialize LCTC.

Instruction	Data	Function
\$00	\$32	Selects display on, master mode, graphic mode.
\$01	\$07	Displays 8-bit data sent from RAM.
\$02	\$1D	Defines number of horizontal bytes.
\$03	\$1F	Defines duty rate as 1/32.
\$04	\$00	Selects cursor position. (Note)
\$08	\$00	Selects display starting address to \$0000.
\$09	\$00	
\$0A	\$00	Selects cursor address to \$0000.
\$0B	\$00	

Note: Display initialized for graphic mode, cursor is not displayed.

c. L2HINT calls subroutines shown in table 1-4.

Table 1-4. Subroutines Called by L2HINT

Subroutine Name	Label	Function
Stores Display Instruction	L2HIST	Writes data LCTC instruction register and data register.
Check Busy Flag	L2HBSY	Checks LCTC busy flag.

2. User Notes

The following procedure must be executed before L2HDCR execution.

- a. Reserve instructions and data in a data table.
- b. Select DDR of port 1 and port 3 as output.

3. RAM Allocation

Label	b7	RAM	b0	Description
INSTR				} Data to be written to LCTC instruction register
DATAR				
				} Data to be written to LCTC data register

Description:

## 4. Sample Application

```

      |
      |
      | CLRA
      | STAA   P1DTR
      | STAA   P3DTR
      | LDAA   #$01
      | STAA   P1DDR
      | STAA   P3DDR
      | } ..... Initialize port 1 and port 3.

```

```

      |
      | J S R   L2HINT } ..... Call L2HINT

```

```

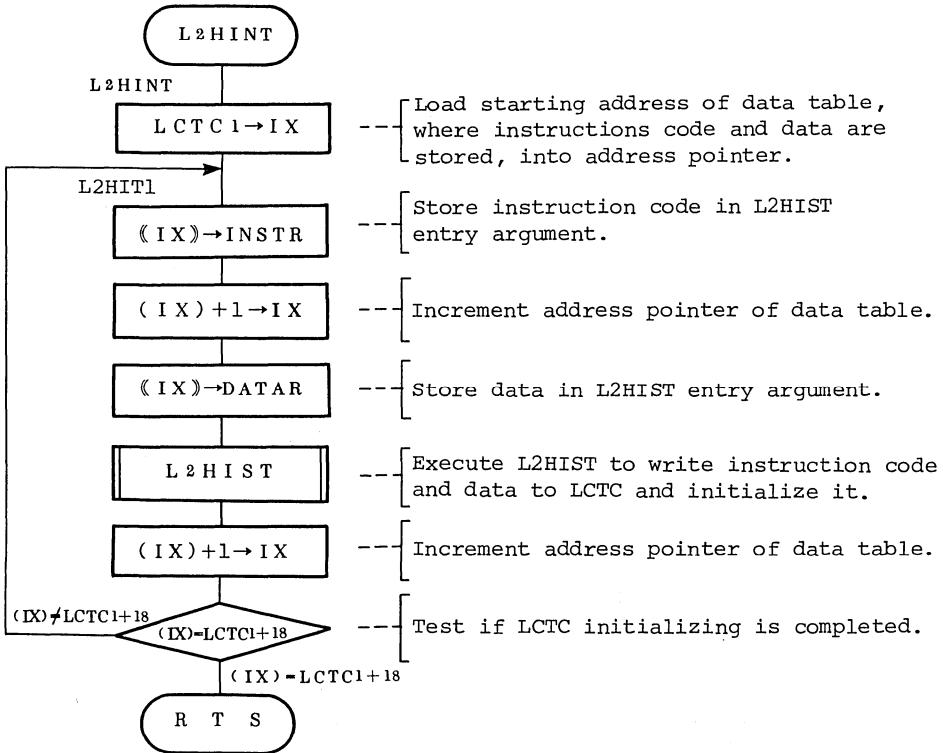
      |
      | LCTC1 FCB   $0, $32 } ..... Reserve data table

```

## 5. Basic Operation

- a. To initialize graphic mode, instructions and data in table 1-3 are written to LCTC instruction register and data register, respectively.
- b. Instruction register and data register are used in pairs, and are selected by RS signal.
- c. Instruction and data in table 1-3 are held in INSTR(RAM) and DATAR(RAM) using index addressing mode. If L2HIST is called, data is written to LCTC instruction register and data register.
- d. In L2HINT, port 1 controls R/W, E, and RS signals.

Flowchart:



Program Module Name: CLEAR DISPLAY  
RAM

MCU/MPU: HD6301Y0

Label: L2HDCR

Function:

Stores \$00 in display RAM to clear display on LM200.

Arguments:

None

Changes in CPU

Registers and Flags:

ACCD	
ACCA	ACCB
x	•

IX
•

C	V
x	x

Z	N
x	x

I	H
•	•

• : Not affected  
x : Undefined  
↑ : Result

Specifications:

ROM (Bytes): 133

RAM (Bytes): 6

Stack (Bytes): 6

No. of cycles: 211474

Reentrant: No

Relocatable: No

Interrupt OK?: Yes

Description:

1. Function Details

- a. L2HDCR has no arguments.
- b. After L2HDCR execution, display or LM200 is cleared.
- c. L2HDCR calls subroutines shown in table 1-5.

Specifications Notes:

1. Values in "Specifications" include values for subroutines called by L2HDCR.
2. "No. of cycles" in "Specifications" indicates the number of cycles required when L2HBSY is executed in the minimum number of cycles (no waiting for LM200).

Program Module Name: CLEAR DISPLAY  
RAM

MCU/MPU: HD6301Y0

Label: L2HDCR

Description:

Table 1-5. Subroutines called by L2HDCR

Subroutine Name	Label	Function
Store Cursor Address	L2HCST	Stores LCTC cursor address.
Continuous Display	L2HDSP	Continuously displays on LM200.
Store Display Instruction	L2HIST	Stores data in LCTC instruction register and data register.
Check Busy Flag	L2HBSY	Checks LCTC busy flag.

2. User Notes

The following procedure must be executed before L2HDCR execution.

- a. Select DDR of port 1 and port 3 as output.
- b. Initialize LCTC display mode.

3. RAM Allocation

Label	RAM	Description
CURL	b7	} Lower byte of cursor address
CURH	b0	
DCOUNT	Upper	} Counter for repeated
	Lower	
DATA		} Display data

4. Sample Application

```
-----  
CLRA  
STAA P1DTR  
STAA P3DTR  
LDAA #$01  
STAA P1DDR  
STAA P3DDR  
JSR L2HINT  
-----  
JSR L2HDCR  
-----
```

..... Initialize port 1 and port 3.  
..... Call L2HINT to initialize LCTC.  
..... Call L2HDCR



Program Module Name: CLEAR DISPLAY  
RAM

MCU/MPU: HD6301Y0

Label: L2HDCR

Description:

5. Basic Operation

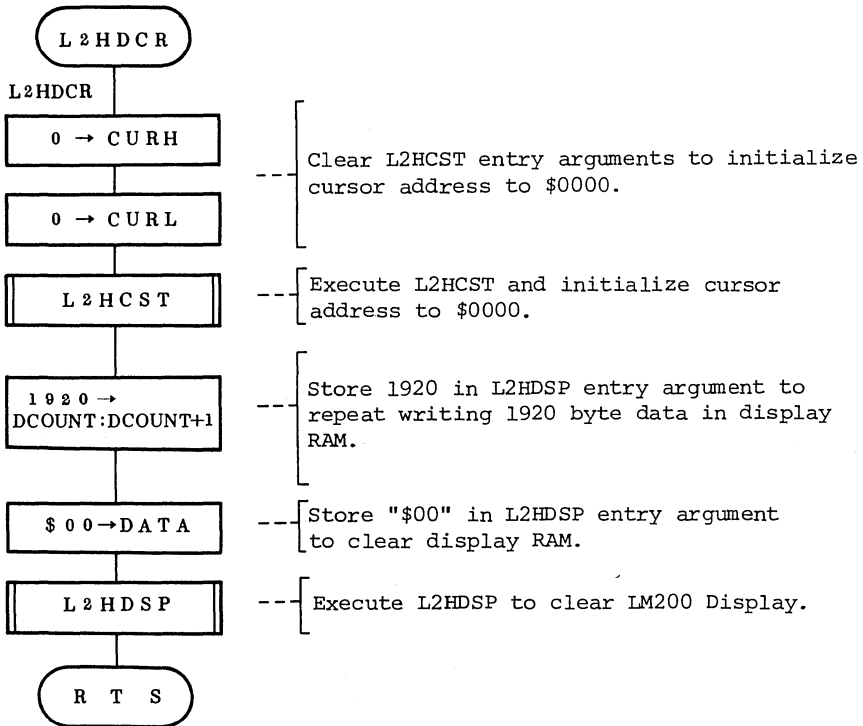
- a. When displaying graphics on LM200, cursor address and display data are written to LCTC.
- b. L2HCTS is called to store \$0000 in cursor address.
- c. LM200 uses 1920 bytes in 1 display screen.
- d. L2HDSP is called to store \$00 throughout RAM so that display on LM200 can be cleared. L2HDSP uses auto-increment function for cursor address.

Program Module Name: CLEAR DISPLAY  
RAM

MCU/MPU: HD6301Y0

Label: L2HDCR

Flowchart:



Program Module Name: DISPLAY DOT

MCU/MPU: HD6301Y0

Label: L2HDST

Function:

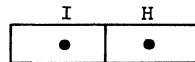
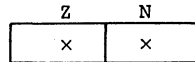
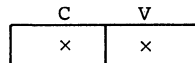
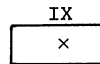
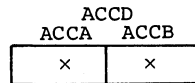
Turns on or off 1 dot specified by row and column coordinates in entry arguments.

Arguments:

Contents	Storage Location	No. of Bytes
Entry Turn on/off indicator	DOTSET (RAM)	1
Dot column coordinate	DTX (RAM)	1
Dot row coordinate	DTY (RAM)	1
Re- turns	_____	_____

Changes in CPU

Registers and Flags:



● : Not affected  
x : Undefined  
↑ : Result

Specifications:

ROM (Bytes): 134

RAM (Bytes): 9

Stack (Bytes): 6

No. of cycles: 513

Reentrant: No

Relocatable: No

Interrupt OK?: Yes

Description:

1. Function Details

a. Argument details

DOTSET(RAM): Data to indicate turning on or off 1 dot.

DOTSET(RAM)=\$0E : Turn off 1 dot.

DOTSET(RAM)=\$0F : Turn on 1 dot.

DTX(RAM) : Dot column coordinate in hexadecimal number.

DTY(RAM) : Dot row coordinate in hexadecimal number.

Specifications Notes:

1. Values in "Specifications" include values for subroutines called by L2HDST.
2. "No. of cycles" in "Specifications" indicates the number of cycles required when L2HBSY is executed in the minimum number of cycles (no waiting for LM200).

Description:

- b. Example of L2HDST execution is shown in figure 1-6. If entry arguments are as shown in part ① of figure 1-6, 1 dot is displayed as shown in part ② of figure 1-6.

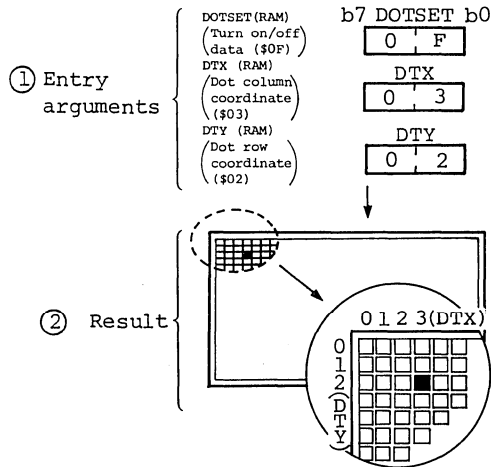


Figure 1-6. Example of L2HDST Execution

- c. L2HDST calls subroutines shown in table 1-6.

Table 1-6. Subroutines Called by L2HDST

Subroutine Name	Label	Function
Store Cursor Address	L2HCST	Stores LCTC cursor address.
Continuous Display	L2HIST	Stores data in LCTC instruction register and data register.
Check Busy Flag	L2HBSY	Checks LCTC busy flag.

2. User Notes

The following procedure must be executed before L2HDST execution.

- Select DDR of port 1 and port 3 as output.
- Initialize LCTC display mode.
- Store entry arguments.



Description:

## 5. Basic Operation

- a. The formula below calculates cursor address and dot to be turned on/off, based on column and row coordinates.

$$\text{Row coordinate} \times 30 + \{(\text{Column coordinate} \wedge \$F8)/8\}$$

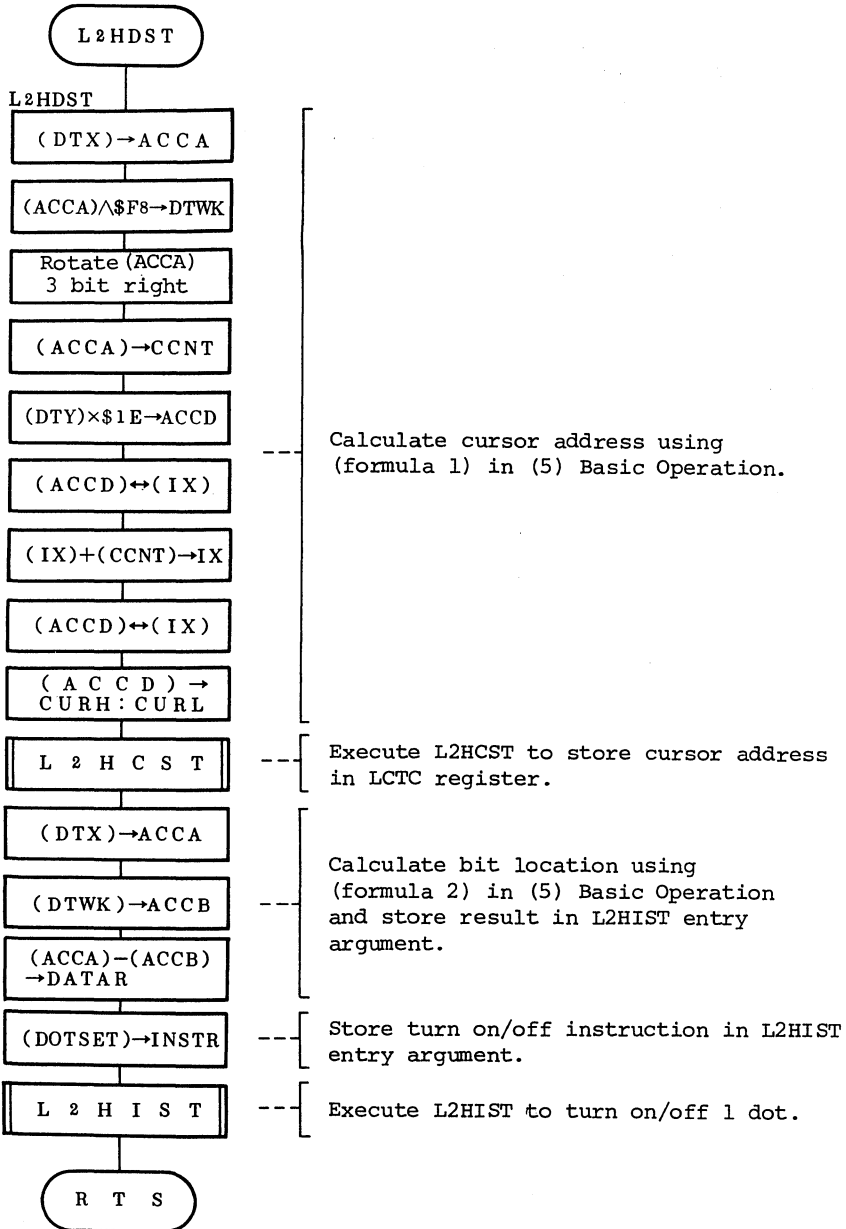
= Cursor address ..... (Formula 1)

$$\text{Column coordinate} - (\text{Column coordinate} \wedge \$F8)$$

= Number of bits ..... (Formula 2)

- b. After cursor address is calculated by Formula 1, upper byte and lower byte of cursor address are held in CURH(RAM) and CURL(RAM), respectively. If L2HCST is executed, cursor address is written to LCTC.
- c. If number of bits obtained by Formula 2 is held in DATAR(RAM), L2HIST execution turns on or off 1 dot.

Flowchart:



Program Module Name: MOVE DISPLAY

MCU/MPU: HD6301Y0

Label: L2HMVE

Function:

Moves current displayed dot up, down, left, or right 1 dot.

Arguments:

Contents	Storage Location	No. of Bytes
Entry Moving direction	ACCA	1

Re-  
turns

Changes in CPU

Registers and Flags:

ACCD	
ACCA	ACCB
x	x

IX
x

C	V
x	x

Z	N
x	x

I	H
●	●

● : Not affected  
x : Undefined  
↑ : Result

Specifications:

ROM (Bytes): 200  
RAM (Bytes): 9  
Stack (Bytes): 8  
No. of cycles: 560  
Reentrant: No  
Relocatable: No  
Interrupt OK?: Yes

Description:

1. Function Details

a. Argument details

ACCA : Data indicating which direction 1 dot will be moved.

ACCA=\$01 : Move 1 dot right.

ACCA=\$02 : Move 1 dot left.

ACCA=\$03 : Move 1 dot down.

ACCA=\$04 : Move 1 dot up.

Specifications Notes: 1. Values in "Specifications" include values for other program modules and subroutines called by L2HMVE.

2. "No. of cycles" in "Specifications" indicates the number of cycles required when L2HBSY is executed in the minimum number of cycles (no waiting for LM200).



Description:

- b. Example of L2HMVE execution is shown in figure 1-7. If entry argument is as shown in part ① of figure 1-7, dots are displayed as shown in part ② of figure 1-7.
- c. L2HDST calls other program modules and subroutines shown in table 1-7.

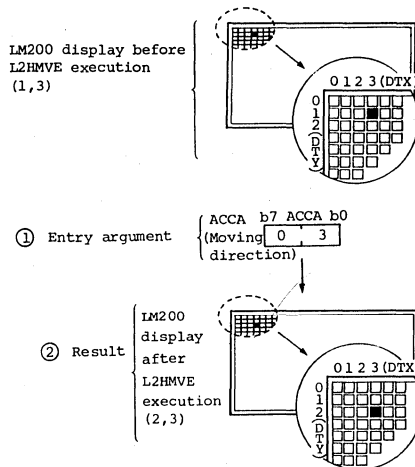


Figure 1-7. Example of L2HMVE Execution

Table 1-7. Program Modules and Subroutines Called by L2HMVE

Program Module/ Subroutine Name	Label	Function
Display Dot	L2HDST	Turns on/off 1 dot.
Store Cursor Address	L2HCST	Stores LCTC cursor address.
Continuous Display	L2HIST	Stores data in LCTC instruction register and data register.
Check Busy Flag	L2HBSY	Checks LCTC busy flag.

2. User Notes

The following procedure must be executed before L2HMVE execution.

- a. Select DDR of port 1 and port 3 as output.
- b. Initialize LCTC display mode.
- c. Load entry argument.

Description:

3. RAM Allocation

Label	RAM	Description
	b7                      b0	
INSTR		Data to be written to LCTC instruction register
DATAR		Data to be written to LCTC data register
DTX		Dot column coordinate
DTY		Dot row coordinate
CURH		Upper byte of cursor address
CURL		Lower byte of cursor address
CCNT		Work area for calculating cursor address based on column coordinate
DTWK		Work area for obtaining 1 dot to be turned on/off
DOTSET		Data to indicate turning on/off 1 dot

4. Sample Application

```

:
:
CLRA
STAA P1DTR
STAA P3DTR
LDAA #$01
STAA P1DDR
STAA P3DDR
JSR L2HINT
LDAA #$01
..... Initialize port 1 and port 3.
..... Call L2HINT to initialize LCTC.
..... Load entry argument

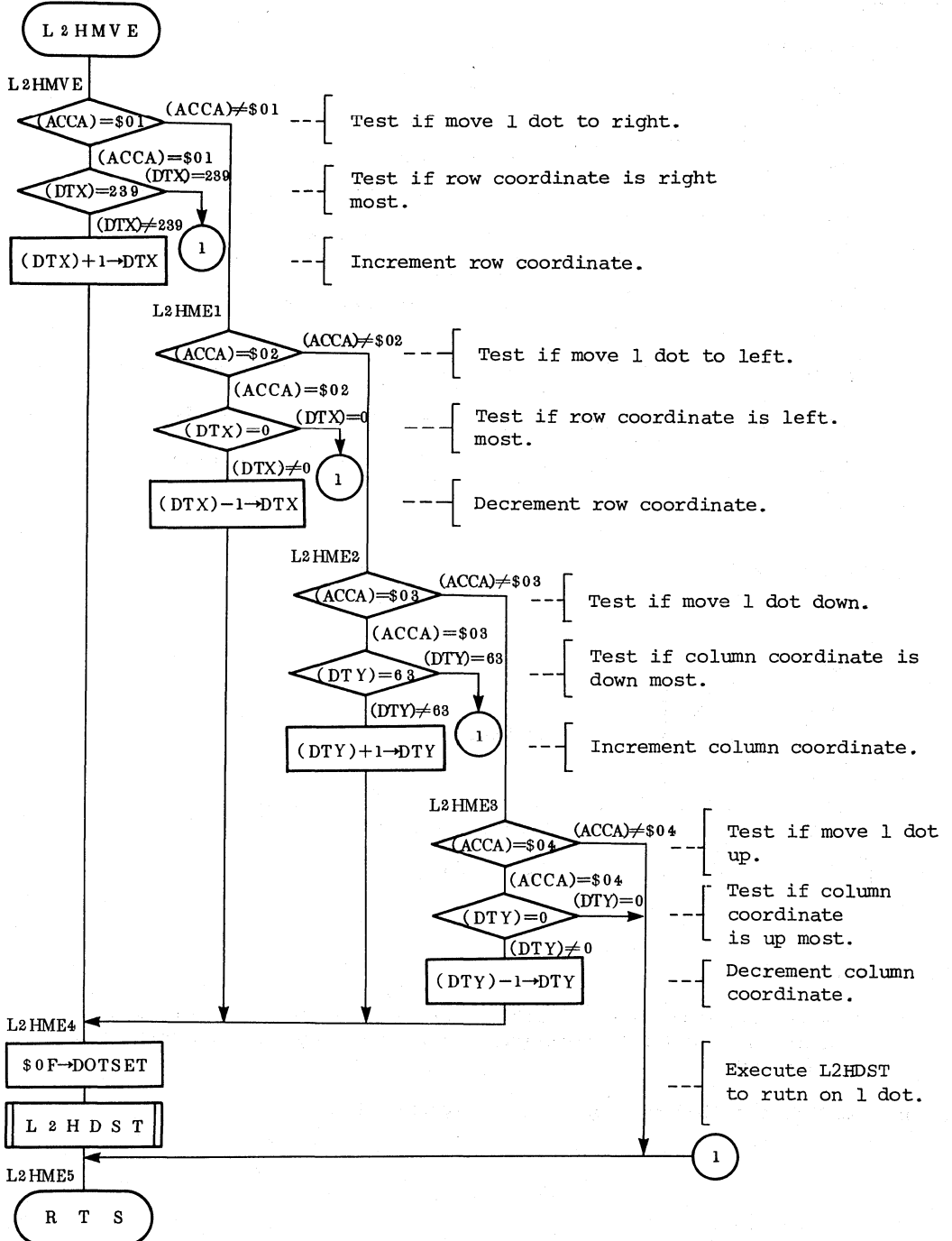
JSR L2HMVE
..... Call L2HMVE
:
:

```

5. Basic Operation

- a. After moving determined direction, DTX(RAM) pointing to dot column coordinate or DTY(RAM) pointing to row coordinate are incremented or decremented.
- b. L2HDST is called to display 1 dot specified in (a).

Flowchart:



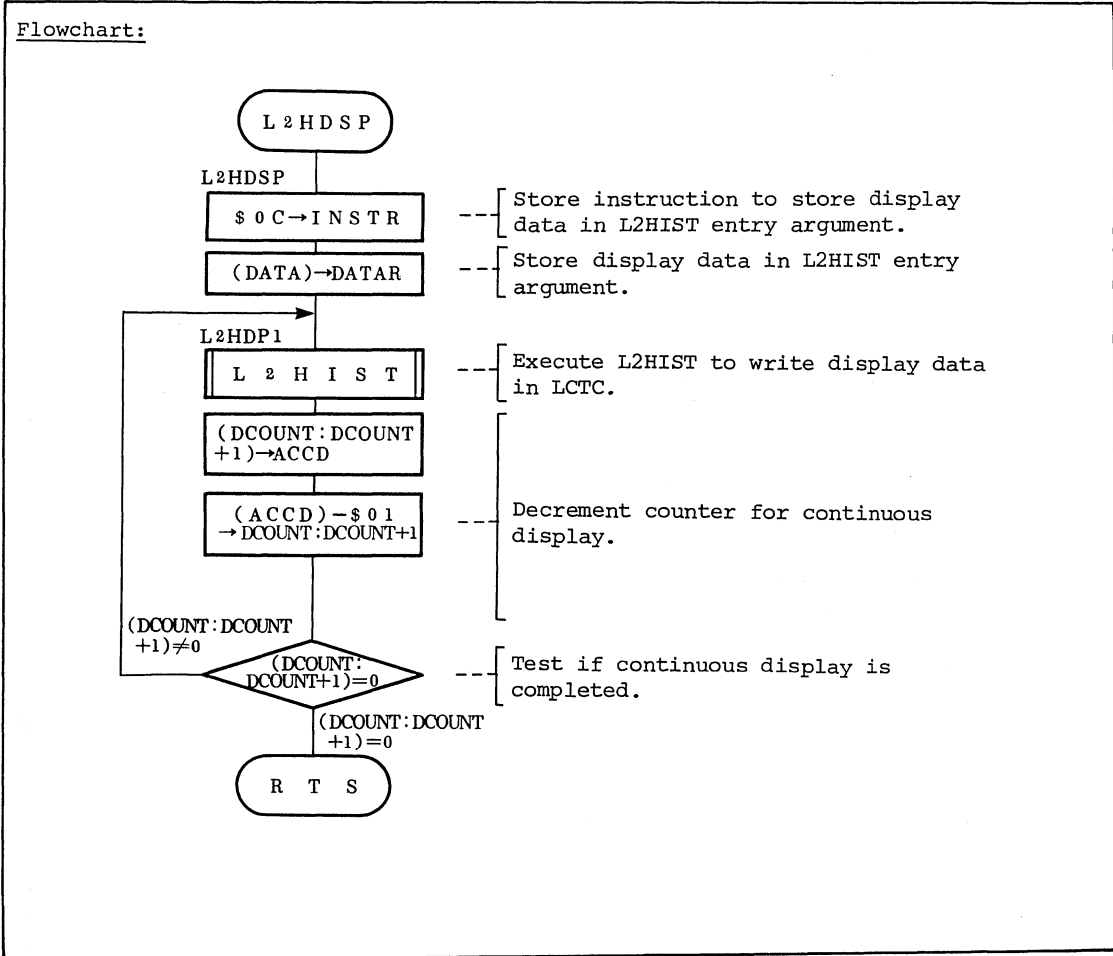
1.4 SUBROUTINE DESCRIPTION

<u>Subroutine Name:</u> CONTINUOUS DISPLAY	<u>MCU/MPU:</u> HD6301Y0	<u>Label:</u> L2HDSP
--	--------------------------	----------------------

Function:  
 Displays specified bytes continuously from the present cursor address.

- Basic Operation:
1. DCOUNT (RAM) is used as a counter to execute L2HIST, writing display data to LCTC until counter is "0".
  2. L2HDSP uses auto-increment function of cursor address.

Program Module Using This Subroutine: L2HDCR



Subroutine Name: STORE CURSOR  
ADDRESS

MCU/MPU: HD6301Y0

Label: L2HCST

Function:

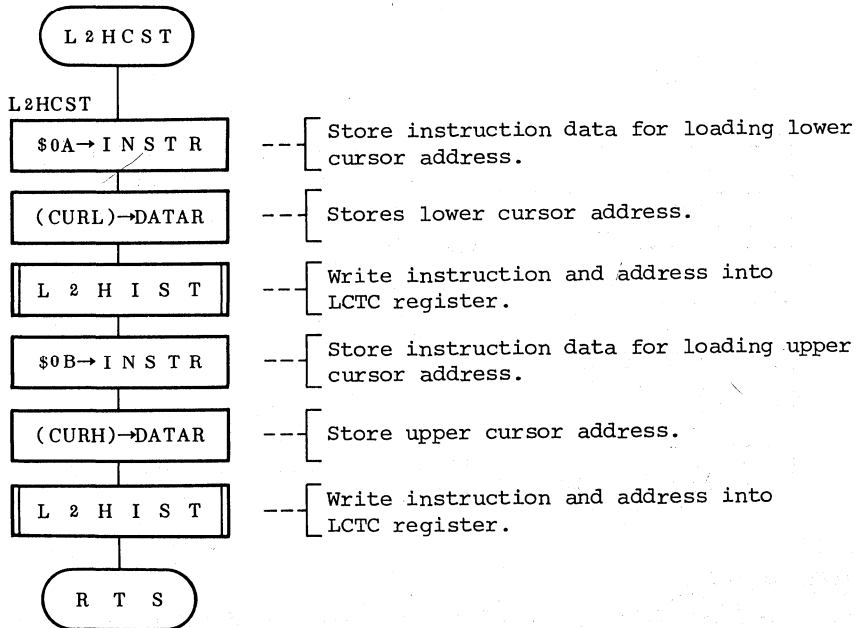
Writes upper and lower bytes of cursor address to LCTC.

Basic Operation:

L2HIST is called twice since lower byte of cursor address is written to LCTC first, and then upper byte to LCTC.

Program Module Using This Subroutine: L2HDCR, L2HDST, L2HMVE

Flowchart:



Subroutine Name: STORE DISPLAY  
INSTRUCTION

MCU/MPU: HD6301Y0

Label: L2HIST

Function:

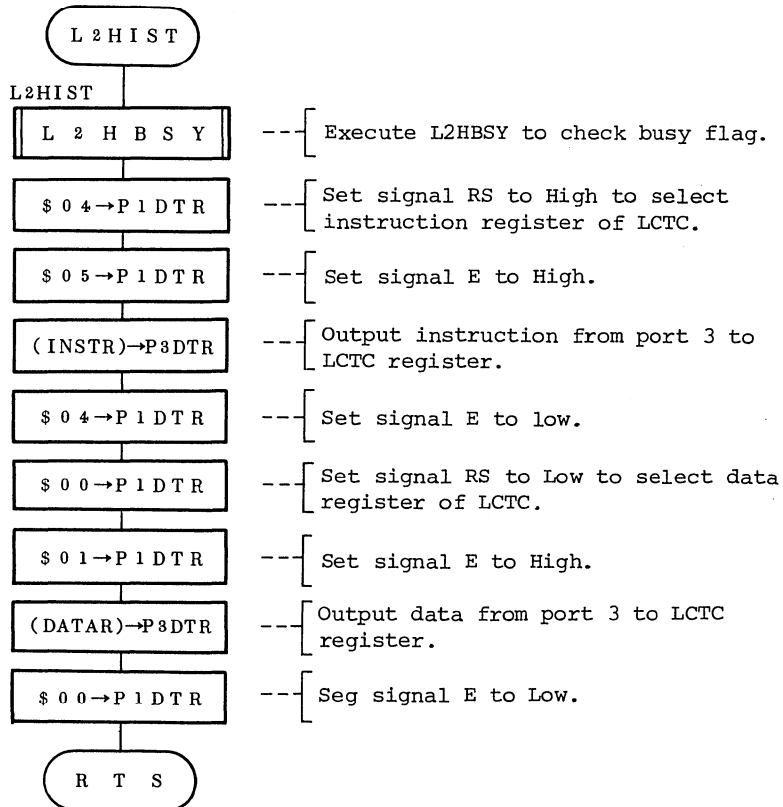
Writes instruction and data to LCTC.

Basic Operation:

1. LCTC busy flag is checked.
2. Data is written to LCTC through port 1 controlling DS, R/W, E signals of LCTC.

Program Module Using This Subroutine: L2HINT, L2HDCR, L2HDST, L2HMVE

Flowchart:



Subroutine Name: CHECK BUSY FLAG

MCU/MPU: HD6301Y0

Label: L2HBSY

Function:

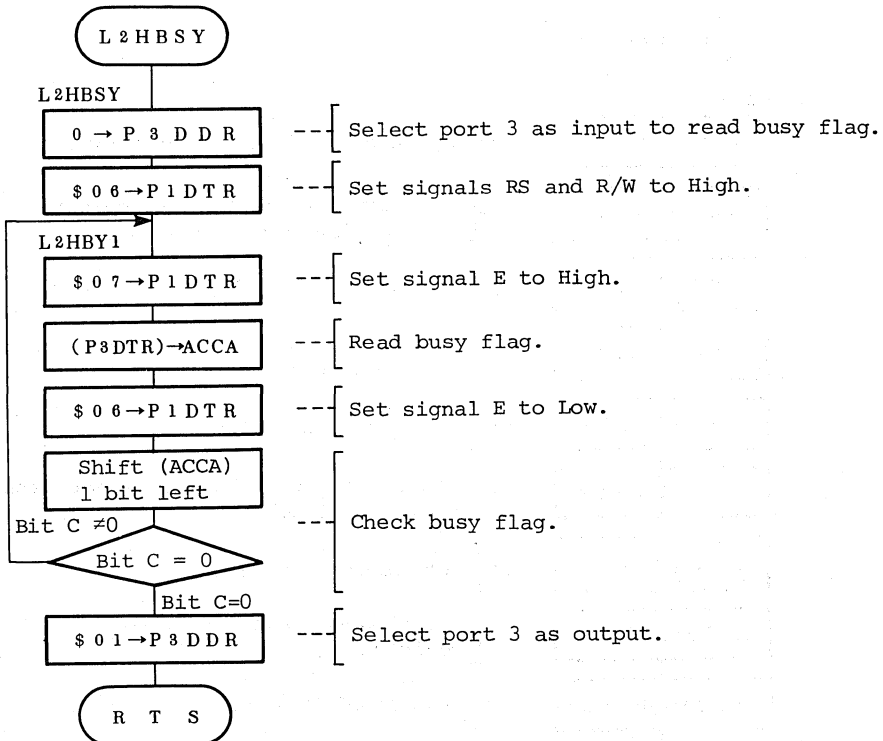
Tests if LCTC is in operation, and waits until LCTC is ready.

Basic Operation:

1. Since the microcomputer cannot access LCTC when LCTC is in operation, microcomputer determines LCTC condition by checking busy flag.
2. Busy flag is read through port 1 controlling RS, R/W, E signals.

Program Module Using This Subroutine: L2HINT, L2HDCR, L2HDST, L2HMVE

Flowchart:



1.5 PROGRAM LISTING

```

00001          *
00002          ***      RAM ALLOCATION      *****
00003          *
00004A 0040          ORG      $40
00005          *
00006A 0040 0001  A INSTR  RMB    1      LCTC instruction register data
00007A 0041 0001  A DATAR  RMB    1      LCTC data register data
00008A 0042 0001  A CURL   RMB    1      Lower byte of cursor address
00009A 0043 0001  A CURH   RMB    1      Upper byte of cursor address
00010A 0044 0002  A DCOUNT RMB    2      Counter for continuous display
00011A 0046 0001  A DATA  RMB    1      Display data
00012A 0047 0001  A CCNT   RMB    1      Work area for cursor address
00013A 0048 0001  A DTX    RMB    1      Dot column coordinate
00014A 0049 0001  A DTY    RMB    1      Dot row coordinate
00015A 004A 0001  A DTWK   RMB    1      Work area for turning on/off data
00016A 004B 0001  A DOTSET  RMB    1      Turning on/off data
00017          *
00018          ***      SYMBOL DEFINITIONS      *****
00019          *
00020          0000  A P1DDR  EQU    $00      Port 1 data direction register
00021          0002  A P1DTR  EQU    $02      Port 1 data register
00022          0004  A P3DDR  EQU    $04      Port 3 data direction register
00023          0006  A P3DTR  EQU    $06      Port 3 data register
00024          *****
00025          *
00026          *
00027          *
00028          *****
00029          *
00030A C000          ORG      $C000
00031          *
00032A C000 8E 013F A L2HMN  LDS    #$13F      Initialize stack pointer
00033A C003 4F          CLRA
00034A C004 97 02  A          STAA  P1DTR      Initialize port 1
00035A C006 97 06  A          STAA  P3DTR      Initialize port 3
00036A C008 86 01  A          LDAA  #$01      Select port 3 as output
00037A C00A 97 00  A          STAA  P1DDR
00038A C00C 97 04  A          STAA  P3DDR
00039A C00E 4F          CLRA      Clear RAM
00040A C00F CE 00C0 A          LDX   #$C0
00041A C012 A7 3F  A L2HMN1 STAA  $3F.X
00042A C014 09          DEX
00043A C015 26 FB C012 BNE   L2HMN1
00044A C017 BD C052 A          JSR   L2HINT      Clear display RAM
00045A C01A BD C03D A          JSR   L2HDSCR      Initialize LCTC
00046A C01D CE C14E A          LDX   #DTDATA      Load data table starting address
00047A C020 A6 00  A L2HMN2 LDAA  0.X      Load display data into ACCA
00048A C022 3C          PSHX      Save data table address
00049A C023 81 FF  A          CMPA  #$FF
00050A C025 27 14 C03B BEQ   PEND      Loop until display is completed
00051A C027 81 00  A          CMPA  #00      Test if 1 dot is turned on/off
00052A C029 26 0B C036 BNE   L2HMN4      Branch if turned on
00053A C02B 86 0E  A          LDAA  #$0E      Store instruction to turn off
00054A C02D 97 4B  A          STAA  DOTSET
00055A C02F BD C068 A          JSR   L2HDST      Turn off 1 dot
00056A C032 38          L2HMN3 PULX      Restore data table address
00057A C033 0B          INX      Increment data table address

```



```

00058A C034 20 EA C020      BRA      L2HMN2
00059A C036 BD C093 A      L2HMN4 JSR      L2HMVE      Turn on 1 dot
00060A C039 20 F7 C032      BRA      L2HMN3
00061A C03B 20 FE C03B      PEND     BRA      PEND      End of program
00062      *****
00063      *
00064      *
00065      *
00066      *****
00067      *
00068      *
00069      *
00070      *
00071      *
00072A C03D 7F 0043 A      L2HDCR CLR      CURH      Load $0000 into cursor address
00073A C040 7F 0042 A      CLR      CURL
00074A C043 BD C125 A      JSR      L2HCST      Write $0000 to cursor address
00075A C046 CC 0780 A      LOD      #1920      Load data to repeat writing 1920 bytes
00076A C049 DD 44 A      STD      DCCUNT
00077A C04B 7F 0046 A      CLR      DATA      Load $00
00078A C04E BD C0D5 A      JSR      L2HDSP      Clear display
00079A C051 39      RTS
00080      *****
00081      *
00082      *
00083      *
00084      *****
00085      *
00086      *
00087      *
00088      *
00089      *
00090A C052 CE C13C A      L2HINT LDX      HLCTC1      Load data table starting address
00091A C055 A6 00 A      L2HIT1 LDAA     0,X          Store LCTC instruction,
00092A C057 97 40 A      STAA     INSTR
00093A C059 08      INX
00094A C05A A6 00 A      LDAA     0,X          Store LCTC data
00095A C05C 97 41 A      STAA     DATAR
00096A C05E BD C103 A      JSR      L2HIST      Write instructin and data to LCTC
00097A C061 08      INX          Increment data table address
00098A C062 8C C14E A      CPX      #LCTC1+1B Test if LCTC is initialized
00099A C065 26 EE C055      BNE     L2HIT1
00100A C067 39      RTS
00101      *****
00102      *
00103      *
00104      *
00105      *****
00106      *
00107      *
00108      *
00109      *
00110      *
00111      *
00112      *
00113A C068 96 48 A      L2HDST LDAA     DTX          Load column coordinate
00114A C06A 84 FB A      ANDA    #$FB          DTX AND $FB->DTWK

```

```

00115A C06C 97 4A   A   STAA   DTWK
00116A C06E 44           LSRA   (DTX AND $FB)/8->CCNT
00117A C06F 44           LSRA
00118A C070 44           LSRA
00119A C071 97 47   A   STAA   CCNT
00120A C073 96 49   A   LDAA   DTY   DTY*30->IX
00121A C075 C6 1E   A   LDAB   #$1E
00122A C077 3D           MUL
00123A C078 18           XGDX
00124A C079 D6 47   A   LDAB   CCNT   IX+CCNT->CURH:CURL
00125A C07B 3A           ABX
00126A C07C 18           XGDX
00127A C07D 97 43   A   STAA   CURH
00128A C07F D7 42   A   STAB   CURL
00129A C081 BD C125 A   JSR    L2HCST Store cursor address
00130A C084 96 48   A   LDAA   DTX
00131A C086 D6 4A   A   LDAB   DTWK
00132A C088 10           SBA    DTX-DTWK->DATAR
00133A C089 97 41   A   STAA   DATAR
00134A C08B 96 48   A   LDAA   DOTSET Store turning on/off data
00135A C08D 97 40   A   STAA   INSTR
00136A C08F BD C103 A   JSR    L2HIST Turn on/off 1 dot
00137A C092 39           RTS
00138
00139 *****
00140 *
00141 *   NAME : L2HMVE (MOVE DISPLAY)
00142 *
00143 *****
00144 *
00145 *   ENTRY : ACCA (MOVING DIRECTION)
00146 *   RETURNS : NOTHING
00147 *
00148A C093 81 01   A   L2HMVE CMPA   #$01   Test if move 1 dot right
00149A C095 26 0B COA2 BNE   L2HME1
00150A C097 D6 48   A   LDAB   DTX
00151A C099 C1 EF   A   CMPB   #239   Test if DTX is right most
00152A C09B 27 37 COD4 BEQ   L2HMES
00153A C09D 5C           INCB   Increment DTX
00154A C09E D7 48   A   STAB   DTX
00155A COA0 20 28 COCD BRA   L2HME4
00156A COA2 81 02   A   L2HME1 CMPA   #$02   Test if move 1 dot left
00157A COA4 26 0B COB1 BNE   L2HME2
00158A COA6 D6 48   A   LDAB   DTX   Test if DTX is left most
00159A COA8 C1 00   A   CMPB   #00
00160A COAA 27 28 COD4 BEQ   L2HMES
00161A COAC 5A           DECB   Decrement DTX
00162A COAD D7 48   A   STAB   DTX
00163A COAF 20 1C COCD BRA   L2HME4
00164A COB1 81 03   A   L2HME2 CMPA   #$03   Test if move 1 dot down
00165A COB3 26 0B COCO BNE   L2HME3
00166A COB5 D6 49   A   LDAB   DTY
00167A COB7 C1 3F   A   CMPB   #63   Test if DTY is bottom
00168A COB9 27 19 COD4 BEQ   L2HMES
00169A COBB 5C           INCB   Increment DTY
00170A COBC D7 49   A   STAB   DTY
00171A COBE 20 0D COCD BRA   L2HME4

```

```

00172A COC0 81 04 A L2HME3 CMPA #H04 Test if move 1 dot up
00173A COC2 26 10 COD4 BNE L2HMES
00174A COC4 06 49 A LDAB DTY
00175A COC6 C1 00 A CMPB #H00 Test if DTY is top
00176A COC8 27 0A COD4 BEQ L2HMES
00177A COCA 5A DECB Decrement DTY
00178A COCB D7 49 A STAB DTY
00179A COCD 86 0F A L2HME4 LDAA #H0F Store turning on instruction
00180A COCF 97 4B A STAA DOTSET
00181A COD1 BD C068 A JSR L2HDST Turn on 1 dot
00182A COD4 39 L2HMES RTS
00183 *****
00184 *
00185 * NAME : L2HDSP (CONTINUOUS DISPLAY) *
00186 *
00187 *****
00188A COD5 86 0C A L2HDSP LDAA #H0C Store instruction
00189A COD7 97 40 A STAA INSTR
00190A COD9 96 46 A LDAA DATA Store display data
00191A CODB 97 41 A STAA DATAR
00192A CODD BD C103 A L2HDP1 JSR L2HIST Write display data to LCTC
00193A COE0 DC 44 A LDD DCOUNT Decrement counter
00194A COE2 83 0001 A SUBD #H01
00195A COE5 DD 44 A STD DCOUNT
00196A COE7 24 F4 CODD BCC L2HDP1 Test if display is completed
00197A COE9 39 RTS
00198 *****
00199 *
00200 * NAME : L2HBSY (CHECK BUSY FLAG) *
00201 *
00202 *****
00203A COEA 4F L2HBSY CLRA
00204A COEB 97 04 A STAA P3DDR Select port 3 as input
00205A COED 86 06 A LDAA #H06 Set RS=1,R/W=1,E=0
00206A COEF 97 02 A STAA P1DTR
00207A COF1 86 07 A L2HBY1 LDAA #H07 Set E=1
00208A COF3 97 02 A STAA P1DTR
00209A COF5 96 06 A LDAA P3DTR Read LCTC busy flag
00210A COF7 C6 06 A LDAB #H06 Set E=0
00211A COF9 D7 02 A STAB P1DTR
00212A COFB 48 ASLA Set busy flag to bit C
00213A COFC 25 F3 COF1 BCS L2HBY1 Test if busy flag=0 ?
00214A COFE 86 01 A LDAA #H01 Select port 3 as output
00215A C100 97 04 A STAA P3DDR
00216A C102 39 RTS
00217 *****
00218 *
00219 * NAME : L2HIST (STORE DISPLAY INSTRUCTION) *
00220 *
00221 *****
00222A C103 BD COEA A L2HIST JSR L2HBSY Check LCTC busy flag
00223A C106 86 04 A LDAA #H04 Set RS=1,R/W=0,E=0
00224A C108 97 02 A STAA P1DTR
00225A C10A 86 05 A LDAA #H05 Set E=1
00226A C10C 97 02 A STAA P1DTR
00227A C10E 96 40 A LDAA INSTR Output instruction through port3
00228A C110 97 06 A STAA P3DTR

```

```

00229A C112 86 04 A LDAA #04 Set E=0
00230A C114 97 02 A STAA P10TR
00231A C116 7F 0002 A CLR P10TR Set RS=0
00232A C119 86 01 A LDAA #01 Set E=1
00233A C118 97 02 A STAA P10TR
00234A C11D 96 41 A LDAA DATAR Output data through port3
00235A C11F 97 06 A STAA P3DTR
00236A C121 7F 0002 A CLR P10TR Set E=0
00237A C124 39 RTS
00238 *****
00239 * *
00240 * NAME : L2HCST (STORE CURSOR ADDRESS) *
00241 * *
00242 *****
00243A C125 86 0A A L2HCST LDAA #0A Store instruction
00244A C127 97 40 A STAA INSTR
00245A C129 96 42 A LDAA CURL Store data
00246A C12B 97 41 A STAA DATAR
00247A C12D BD C103 A JSR L2HIST Write lower cursor ADDR to LCTC
00248A C130 86 0B A LDAA #0B Store instruction
00249A C132 97 40 A STAA INSTR
00250A C134 96 43 A LDAA CURH Store data
00251A C136 97 41 A STAA DATAR
00252A C138 BD C103 A JSR L2HIST Write upper cursor ADDR to LCTC
00253A C13B 39 RTS
00254 *****
00255 * *
00256 * DATA TABLE *
00257 * *
00258 *****
00259A C13C 00 A LCTC1 FCB $0,$32 *Instruction and data-
00260A C13E 01 A FCB $1,$07 to initialize LCTC
00261A C140 02 A FCB $2,$1D
00262A C142 03 A FCB $3,$1F
00263A C144 04 A FCB $4,$00
00264A C146 08 A FCB $8,$00
00265A C148 09 A FCB $9,$00
00266A C14A 0A A FCB $A,$00
00267A C14C 0B A FCB $B,$00
00268A C14E 01 A DTDATA FCB $01 *Display data
00269A C14F 01 A FCB $01
00270A C150 01 A FCB $01
00271A C151 03 A FCB $03
00272A C152 03 A FCB $03
00273A C153 03 A FCB $03
00274A C154 03 A FCB $03
00275A C155 02 A FCB $02
00276A C156 02 A FCB $02
00277A C157 02 A FCB $02
00278A C158 04 A FCB $04
00279A C159 04 A FCB $04
00280A C15A 04 A FCB $04
00281A C15B 04 A FCB $04
00282A C15C 03 A FCB $03
00283A C15D 03 A FCB $03
00284A C15E 01 A FCB $01
00285A C15F 01 A FCB $01

```

```

00286A C160 01 A FCB $01
00287A C161 01 A FCB $01
00288A C162 00 A FCB $00
00289A C163 01 A FCB $01
00290A C164 00 A FCB $00
00291A C165 04 A FCB $04
00292A C166 01 A FCB $01
00293A C167 01 A FCB $01
00294A C168 04 A FCB $04
00295A C169 03 A FCB $03
00296A C16A 01 A FCB $01
00297A C16B 01 A FCB $01
00298A C16C 02 A FCB $02
00299A C16D 03 A FCB $03
00300A C16E 03 A FCB $03
00301A C16F 03 A FCB $03
00302A C170 01 A FCB $01
00303A C171 02 A FCB $02
00304A C172 02 A FCB $02
00305A C173 02 A FCB $02
00306A C174 02 A FCB $02
00307A C175 01 A FCB $01
00308A C176 04 A FCB $04
00309A C177 04 A FCB $04
00310A C178 FF A FCB $FF
00311 *****
00312 *
00313 * VECTOR ADDRESSES *
00314 *
00315 *****
00316 *
00317A FFEA ORG $FFEA
00318 *
00319A FFEA C000 A FDB L2HMN IRQ2
00320A FFEC C000 A FDB L2HMN CMI
00321A FFEE C000 A FDB L2HMN TRAP
00322A FFF0 C000 A FDB L2HMN SIO
00323A FFF2 C000 A FDB L2HMN TOI
00324A FFF4 C000 A FDB L2HMN OCI
00325A FFF6 C000 A FDB L2HMN ICI
00326A FFF8 C000 A FDB L2HMN IRQ1/ISF
00327A FFFA C000 A FDB L2HMN SWI
00328A FFFC C000 A FDB L2HMN NMI
00329A FFFE C000 A FDB L2HMN RES
00330 *
00331 END

```

2.1 HARDWARE DESCRIPTION

2.1.1 Function

Drives LEDs by amplifying signals from the HD6301Y0, displaying "76543210" on the LED display.

2.1.2 Microcomputer Operation

The HD6301Y0 executes output compare interrupt 1 every 1.25 ms using timer 1 to drive LEDs by outputting segment data through port 1 and digit data through port 6. Darlington transistor are driven directly through port 6.

2.1.3 Peripheral Devices

LEDs: Driven dynamically at a frame frequency of 100 Hz and duty rate of 1/8.

2.1.4 Circuit Diagram

8-digit × 8-segment LED control circuit is shown in figure 2-1.

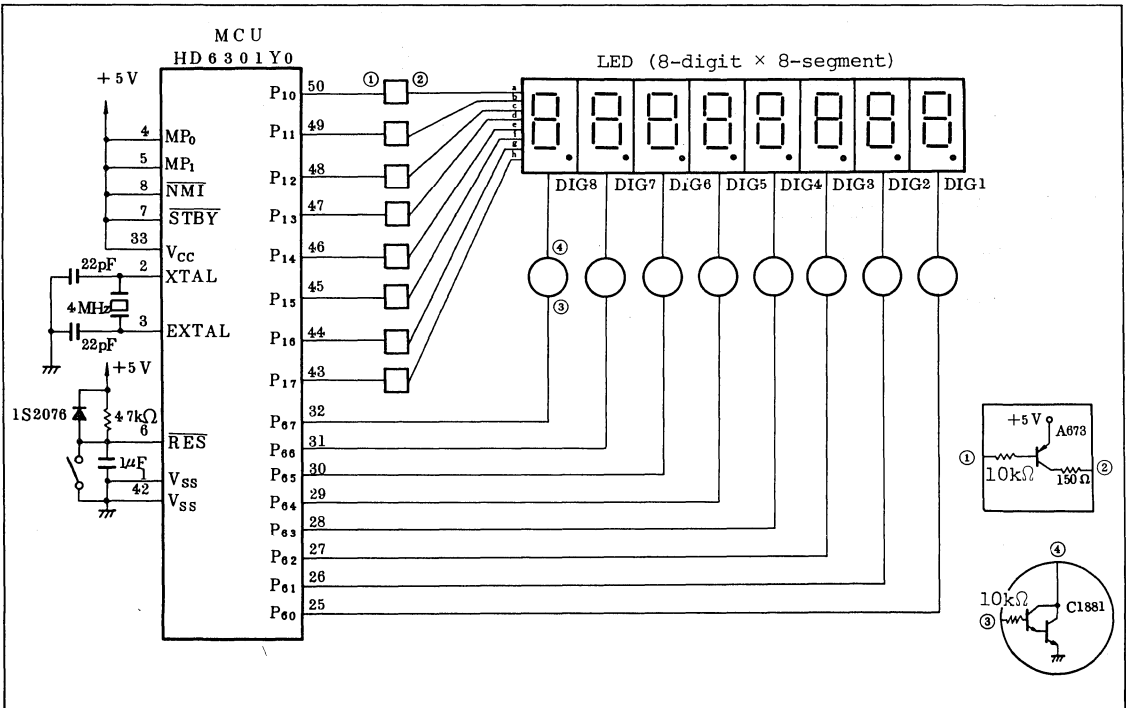


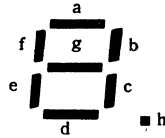
Figure 2-1. 8-digit × 8-segment LED Control Circuit

## 2.1.5 Pin Functions

Pin functions at the interface between the HD6301Y0 and LED are shown in table 2-1.

Table 2-1. Pin Functions

Pin Name (HD6301Y0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (LED)	Program Label
P <sub>60</sub>	Output	High	Outputs digit data to LED.	DIG1	P1DTR
P <sub>61</sub>	Output	High		DIG2	
P <sub>62</sub>	Output	High		DIG3	
P <sub>63</sub>	Output	High		DIG4	
P <sub>64</sub>	Output	High		DIG5	
P <sub>65</sub>	Output	High		DIG6	
P <sub>66</sub>	Output	High		DIG7	
P <sub>67</sub>	Output	High		DIG8	
P <sub>10</sub>	Output	Low	Outputs segment data to LED. "a~h" in Pin Name (LED) corresponds to segment pattern below.	a	P1DTR
P <sub>11</sub>	Output	Low		b	
P <sub>12</sub>	Output	Low		c	
P <sub>13</sub>	Output	Low		d	
P <sub>14</sub>	Output	Low		e	
P <sub>15</sub>	Output	Low		f	
P <sub>16</sub>	Output	Low		g	
P <sub>17</sub>	Output	Low		Segment Pattern	



## 2.1.6 Hardware Operation

The timing chart for segment signals and digit signals is shown in figure 2-2.

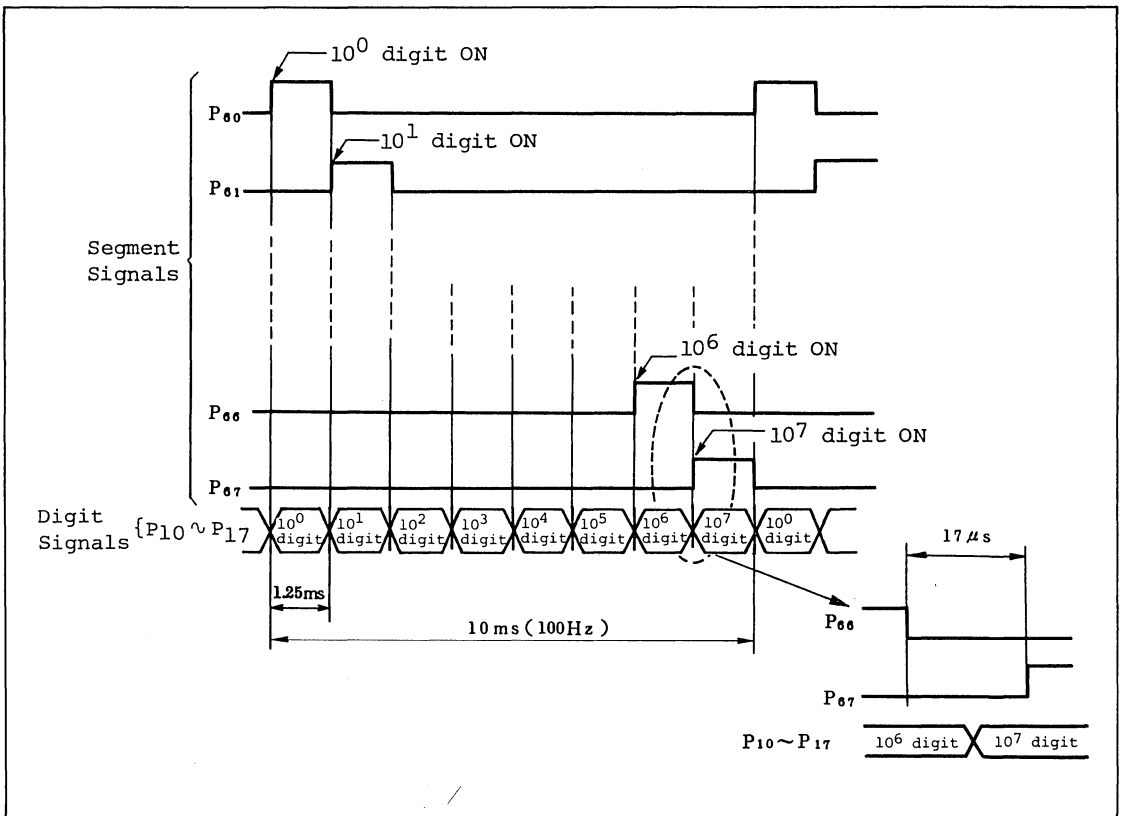


Figure 2-2. Timing Chart of Segment Signals and Digit Signals

## 2.2 SOFTWARE DESCRIPTION

### 2.2.1 Program Module Configuration

The program module configuration for displaying digits on LED is shown in figure 2-3.

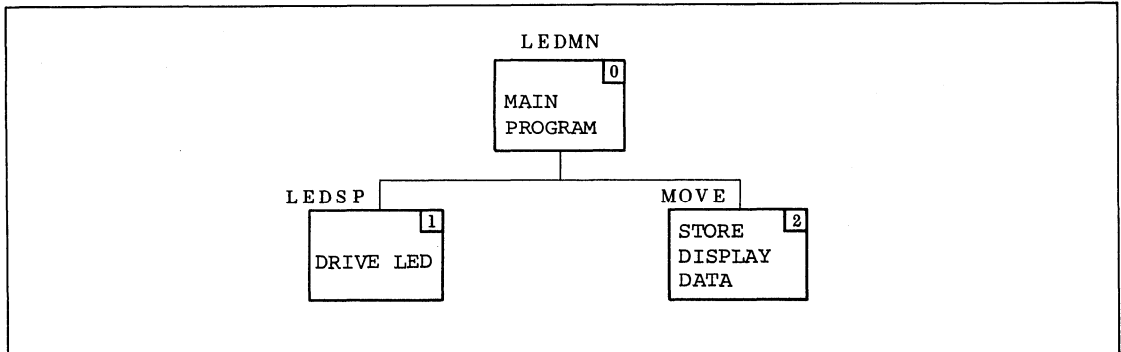


Figure 2-3. Program Module Configuration



### 2.2.2 Program Module Functions

Program module functions are summarized in table 2-2.

Table 2-2. Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	LEDMN	Demonstrates display data on LED.
1	DRIVE LED	LEDSP	Displays digits on LED using dynamic drive.
2	STORE DISPLAY DATA	MOVE	Stores display data in display RAM. Refer to MOVE in HD6301/HD6303 FAMILY APPLICATION NOTES (SOFTWARE) for details.

### 2.2.3 Program Module Process Flow (Main Program)

The flowchart in figure 2-4 is an example of the 8-digits × 8-segment LED display performed by the program module in figure 2-3. The main program in figure 2-4 demonstrates the display on LED shown in figure 2-5.

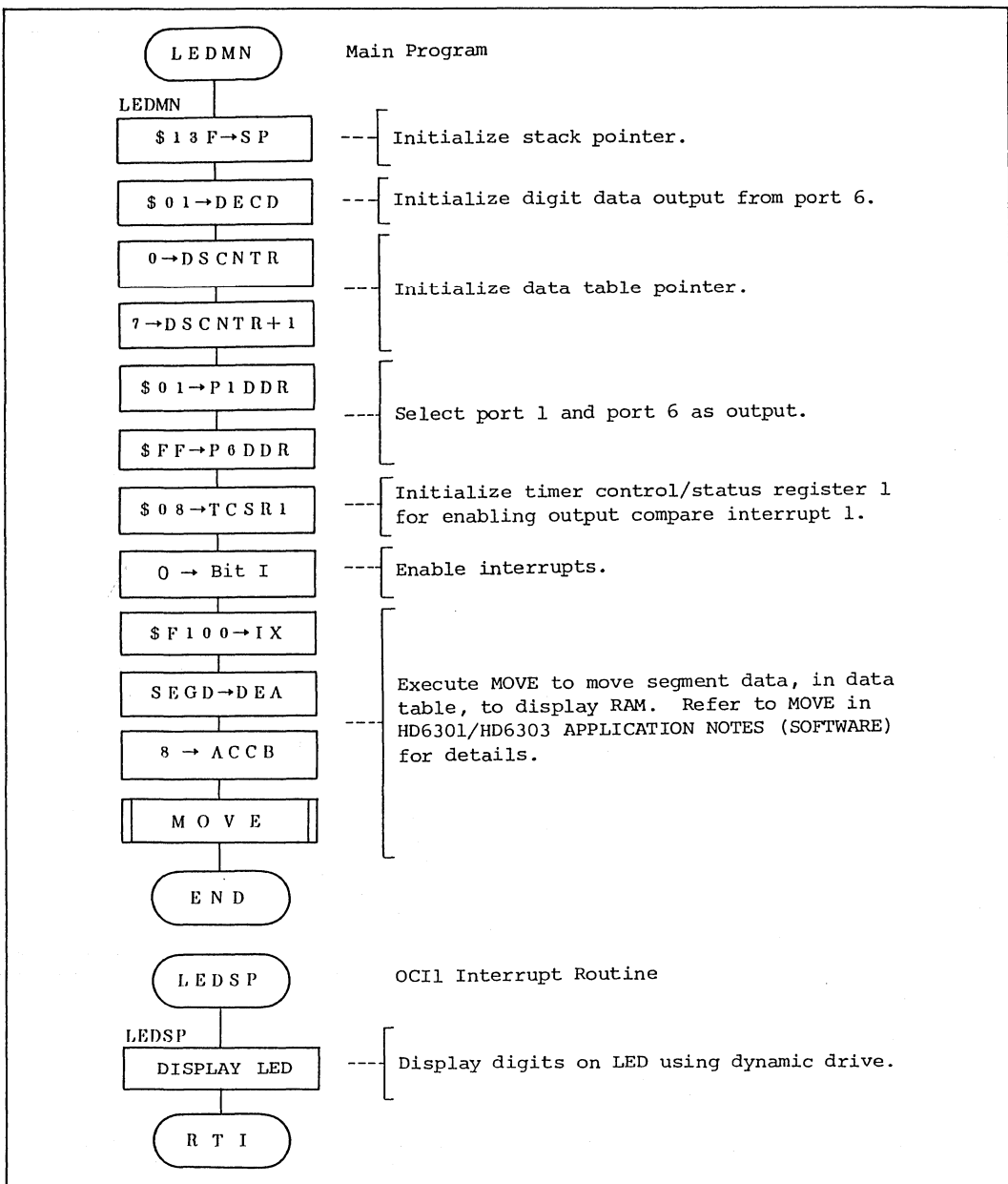


Figure 2-4. Program Module Flowchart

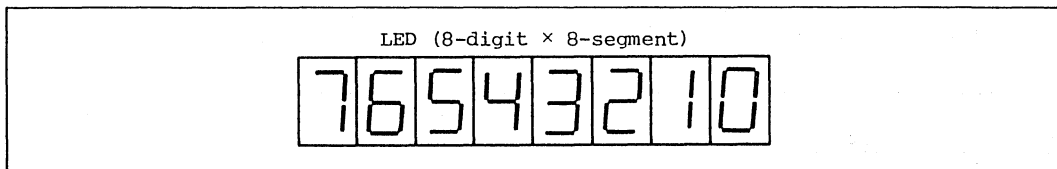


Figure 2-5. Example of 8-digit × 8-segment LED Display

## 2.3 PROGRAM MODULE DESCRIPTION

Program Module Name: DRIVE LED

MCU/MPU: HD6301Y0

Label: LEDSP

Function:

Displays digits on 8-digit × 8-segment LED using dynamic drive.

Arguments:

Contents	Storage Location	No. of Bytes
Entry Display data	SEGD (RAM)	8

Re-  
turns

Changes in CPU

Registers and Flags:

ACCD	
ACCA	ACCB
×	×

IX
×

C	V
×	×

Z	N
×	×

I	H
●	●

● : Not affected  
 × : Undefined  
 † : Result

Specifications:

ROM (Bytes): 48  
 RAM (Bytes): 11  
 Stack (Bytes): 0  
 No. of cycles: 69  
 Reentrant: No  
 Relocatable: No  
 Interrupt OK?: No

Description:

1. Function Details
  - a. Argument details  
 SEGD (RAM) : Holds display data.
  - b. Example of LEDSP execution is shown in figure 2-6. If entry arguments are as shown in part 1 of figure 2-6, data is displayed on LED as shown in part 2 of figure 2-6. Table 2-3 shows relation between segment data and display.
  - c. LEDSP calls no subroutines.

Specifications Notes:

"No. of cycles" in "Specifications" indicates the number of cycles required to display the 10<sup>0</sup> digit (rightmost) on LED.

Description:

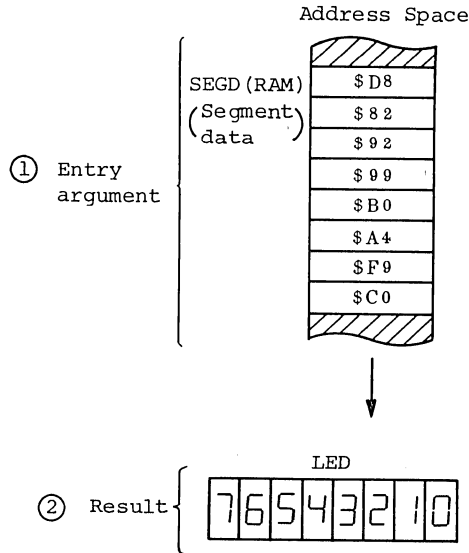


Figure 2-6. Example of LEDSP Execution

Table 2-3. Relation between Segment Data and Display

Segment Data	Display	Segment Data	Display
\$C0	0	\$92	5
\$F9	1	\$82	6
\$A4	2	\$D8	7
\$B0	3	\$80	8
\$99	4	\$90	9

2. User Notes

The following procedure must be performed before LEDSP execution.

- a. Initialize digit data.
- b. Initialize display RAM pointer.
- c. Select port 1 and port 6 as output.



Program Module Name: DRIVE LED

MCU/MPU: HD6301Y0

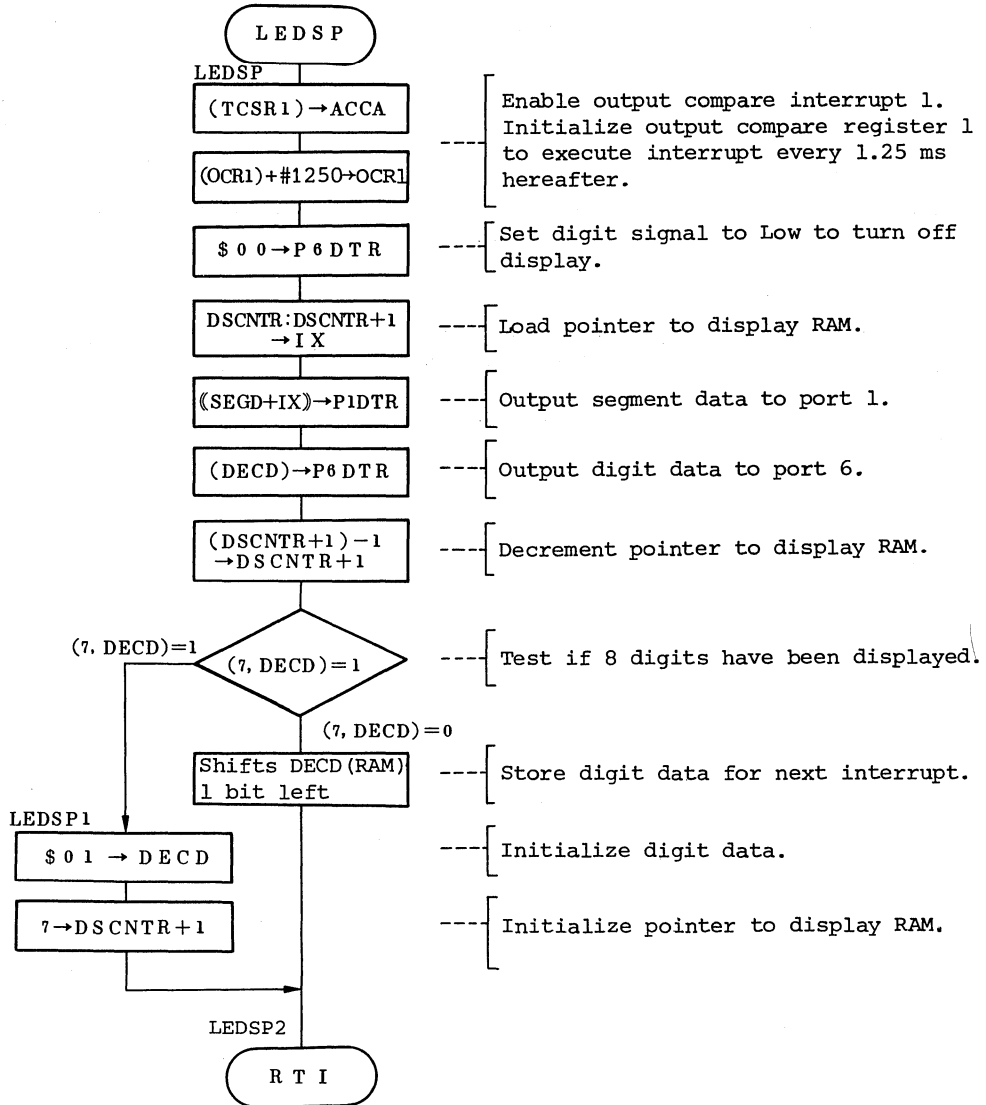
Label: LEDSP

Description:

5. Basic Operation

- a. Each digit is dynamically displayed one by one from the  $10^0$  digit every interrupt.
- b. In the interrupt routine, the procedure below is performed.
  - i. A specific digit signal is set to Low to turn off its display.
  - ii. Segment data for next digit is read from display RAM and output to ports.
  - iii. Digit signal is set to High to turn on display.
- c. DSCNR(RAM) is used as a pointer to display RAM, and incremented every interrupt. After  $10^7$  digit displayed, DSCNR(RAM) is cleared.
- d. DECD(RAM) contains digit data, and is shifted 1 bit left to indicate next display digit.

Flowchart:



## 2.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

## 2.5 PROGRAM LISTING

```

00001          *
00002          **** RAM ALLOCATION *****
00003          *
00004A 0040          ORG    $40
00005          *
00006A 0040    0008 A SEGD  RMB    8      8-digit segment data
00007A 0048    0002 A DSCNTR RMB    2      Display RAM pointer
00008A 004A    0001 A DECD   RMB    1      Digit data
00009A 0048    0002 A DADR   RMB    2      Destination address
00010          *
00011          **** SYMBOL DEFINITIONS *****
00012          *
00013          0000 A P1DDR  EQU    $00      Port 1 data direction register
00014          0002 A P1DTR  EQU    $02      Port 1 data register
00015          0008 A TCSR1  EQU    $08      Timer control/status register 1
00016          0009 A FRC    EQU    $09      Free running counter
00017          000B A OCR1   EQU    $0B      Output compare register 1
00018          0016 A P6DDR  EQU    $16      Port 6 data direction register
00019          0017 A P6DTR  EQU    $17      Port 6 data register
00020          ****
00021          *
00022          *      MAIN PROGRAM : LEDMN      *
00023          *
00024          ****
00025          *
00026A C000          ORG    $C000
00027          *
00028A C000 8E 013F A LEDMN  LDS    #$13F   Initialize stack pointer
00029A C003 86 01  A        LDAA   #$01   Initialize digit data
00030A C005 97 4A  A        STAA   DECD
00031A C007 7F 0048 A        CLR    DSCNTR  Initialize display RAM pointer
00032A C00A 86 07  A        LDAA   #7
00033A C00C 97 49  A        STAA   DSCNTR+1
00034A C00E 86 01  A        LDAA   #$01   Select port 1 and port 6 as output
00035A C010 97 00  A        STAA   P1DDR
00036A C012 86 FF  A        LDAA   #$FF
00037A C014 97 16  A        STAA   P6DDR
00038A C016 86 08  A        LDAA   #$08   Enable OCI interrupt
00039A C018 97 08  A        STAA   TCSR1
00040A C01A 0E      A        CLI
00041A C01B CE F100 A        LDX    #$F100  Load source address
00042A C01E CC 0040 A        LDD    #SEGD   Load destination address
00043A C021 DD 4B  A        STD    DEA
00044A C023 C6 08  A        LDAB   #8      Load no. of bytes to be moved
00045A C025 8D 02 C029 BSR    MOVE   Move segment data to display RAM
00046A C027 20 FE C027 PEND  BRA    PEND  End of program
00047          ****
00048          *
00049          *      NAME : MOVE (MOVE MEMORY BLOCKS)      *
00050          *
00051          ****
00052          *
00053          *      ENTRY : IX (SOURCE STARTING ADDRESS)      *
00054          *      DEB (DESTINATION STARTING ADDRESS) *
00055          *      ACCB (NO. OF BYTES)                *
00056          *      RETURNS : NOTHING                    *
00057          *

```



```

00058 *****
00059A C029 A6 00 A MOVE LDAA 0,X Load transfer data
00060A C02B 08 INX Increment source ADDR
00061A C02C 3C PSHX Push source ADDR
00062A C02D DE 4B A LDX DEA Load destination ADDR
00063A C02F A7 00 A STAA 0,X Store transfer data
00064A C031 08 INX Increment destination ADDR
00065A C032 DF 4B A STX DEA Store destination ADDR
00066A C034 38 PULX Pull source ADDR
00067A C035 5A DECB Decrement transfer counter
00068A C036 26 F1 C029 BNE MOVE Loop until transfer counter = 0
00069A C038 39 RTS
00070 *****
00071 * *
00072 * NAME : LEDSP (DRIVE LED) *
00073 * *
00074 *****
00075 * *
00076 * ENTRY : SEG D (DISPLAY DATA) *
00077 * RETURNS : NOTHING *
00078 * *
00079 *****
00080A C039 96 08 A LEDSP LDAA TCSR1
00081A C03B DC 0B A LDD OCR1 Set interrupt timing
00082A C03D C3 04E2 A ADDD #1250
00083A C040 DD 0B A STD OCR1
00084A C042 7F 0017 A CLR P6DTR Turn off display
00085A C045 DE 48 A LDX OSCNTR Load display RAM pointer
00086A C047 A6 40 A LDAA SEG D.X Output segment data
00087A C049 97 02 A STAA P1DTR
00088A C04B 96 4A A LDAA DECD Output digit data
00089A C04D 97 17 A STAA P6DTR
00090A C04F 7A 0049 A DEC OSCNTR+1 Decrement display RAM pointer
00091A C052 7B 80 4A BTST 7,DECD 8-digit displayed?
00092A C055 26 05 C05C BNE LEDSP1
00093A C057 78 004A A ASL DECD Store next digit data
00094A C05A 20 08 C064 BRA LEDSP2
00095A C05C 86 01 A LEDSP1 LDAA # $01 Initialize digit data
00096A C05E 97 4A A STAA DECD
00097A C060 86 07 A LDAA #7 Initialize display RAM pointer
00098A C062 97 49 A STAA OSCNTR+1
00099A C064 3B LEDSP2 RTI
00100 *****
00101 * *
00102 * DATA TABLE *
00103 * *
00104 *****
00105 *
00106A F100 * ORG $F100
00107 *
00108A F100 DB A FCB $D8,$82,$92,$99,$B0,$A4,$F9,$C0
00109 *****
00110 * *
00111 * VECTOR ADDRESSES *
00112 * *
00113 *****
00114 *

```

00115A	FFEA			ORG	\$FFEA	
00116			*			
00117A	FFEA	C000	A	FDB	LEDNM	IRQ2
00118A	FFEC	C000	A	FDB	LEDNM	CMI
00119A	FFEE	C000	A	FDB	LEDNM	TRAP
00120A	FFF0	C000	A	FDB	LEDNM	SIO
00121A	FFF2	C000	A	FDB	LEDNM	TOI
00122A	FFF4	C039	A	FDB	LEDSP	OCI
00123A	FFF6	C000	A	FDB	LEDNM	ICI
00124A	FFF8	C000	A	FDB	LEDNM	IRQ1/ISF
00125A	FFFF	C000	A	FDB	LEDNM	SWI
00126A	FFFF	C000	A	FDB	LEDNM	NMI
00127A	FFFF	C000	A	FDB	LEDNM	RES
00128			*			
00129				END		

3.1 HARDWARE DESCRIPTION3.1.1 Function

Outputs pulse with 0-100% duty rate and performs digital-to-analog conversion of output pulse with an external integration circuit.

3.1.2 Microcomputer Operation

The HD6301Y0 increases duty rate by 4% every 0.3s. This increase in duty rate changes the output voltage from 0 to 5 V in 0.2 V increments. High or Low is output from the Tout3 pin by executing the counter match interrupt routine with timer 2. In addition, the High and Low Period of the pulse is controlled by changing values in the time constant register.

3.1.3 Peripheral Devices

Integration circuit: Performs level conversion of the output pulse through the HD14050B, and converts the result from digital to analog. The operational amplifier here prevents the fluctuation of analog output voltage caused by the load in the user system.

3.1.4 Circuit Diagram

Pulse output circuit is shown in figure 3-1.

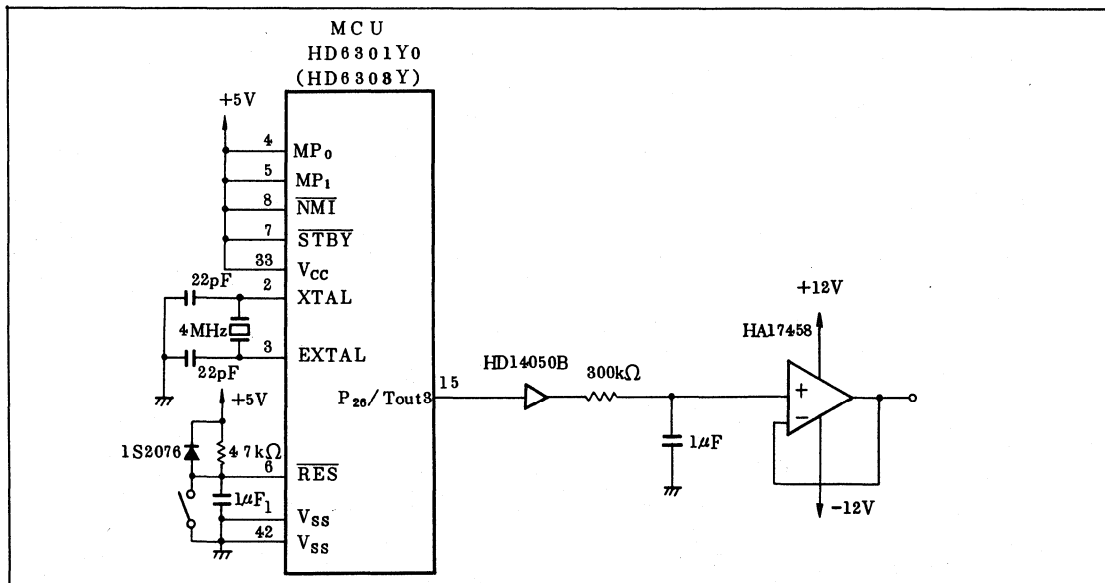


Figure 3-1. Pulse Output Circuit

### 3.1.5 Pin Function

Pin function for output pulse is shown in Table 3-1.

Table 3-1. Pin Function

Pin Name (HD6301Y0)	Input/ Output	Active Level (High or Low)	Function	Program Label
P26/Tout3	Output	—	Outputs pulse	—

### 3.1.6 Hardware Operation

Pulse output and DA conversion is shown in figure 3-2.

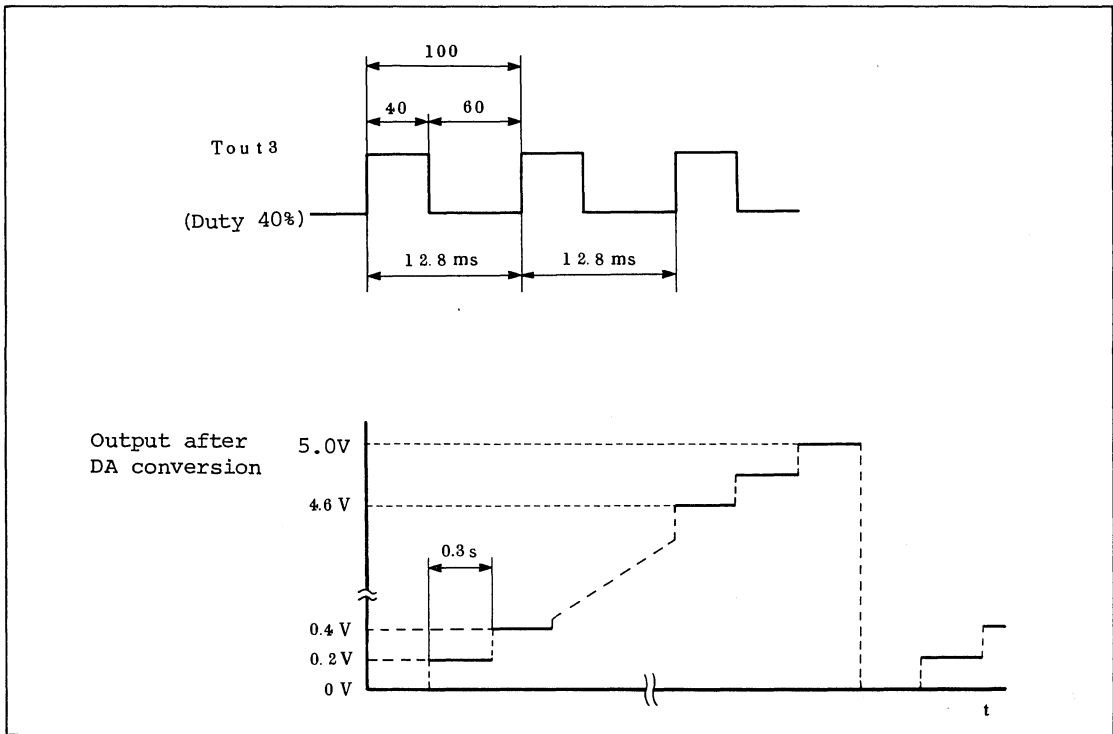


Figure 3-2. Pulse Output and Output State after DA Conversion

## 3.2 SOFTWARE DESCRIPTION

### 3.2.1 Program Module Configuration

The program module configuration for pulse output is shown in figure 3-3.

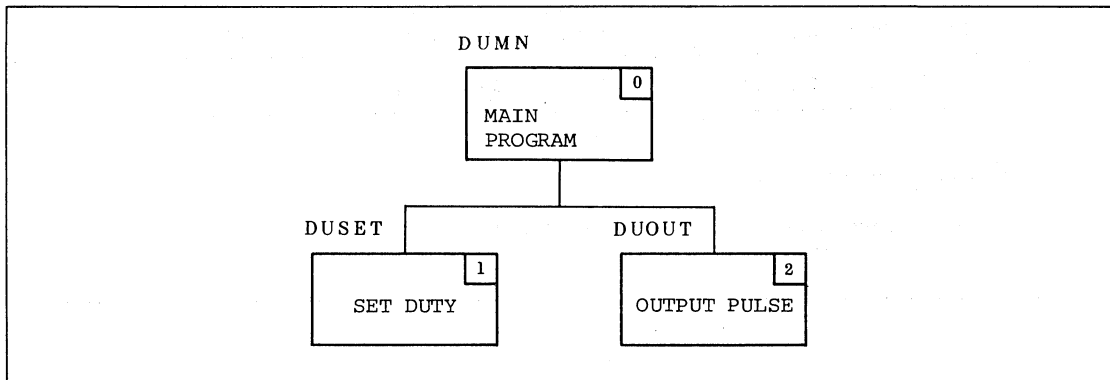


Figure 3-3. Program Module Configuration

### 3.2.2 Program Module Functions

Program module functions are summarized in table 3-2.

Table 3-2. Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	DUMN	Outputs pulse.
1	SET DUTY	DUSET	Sets duty rate of pulse.
2	OUTPUT PULSE	DUOUT	Outputs pulse.

### 3.2.3 Program Module Process Flow (Main Program)

The flowchart in figure 3-4 is an example of controlling pulse output and digital-to-analog conversion, using the program module in figure 3-3.



### 3.3 PROGRAM MODULE DESCRIPTION

Program Module Name: SET DUTY

MCU/MPU: HD6301Y0/  
HD6303Y

Label: DUSET

Function:

Defines pulse status, which is output in DUOUT, based on duty rate.

Arguments:

Contents		Storage Location	No. of Bytes
Entry Duty rate		VCOUNT (RAM)	1
Re-returns	High period of pulse	HTIME (RAM)	1
	Low period of pulse	LTIME (RAM)	1
	Pulse status flag	HLOUT (RAM)	1

Changes in CPU

Registers and Flags:

ACCD	
ACCA	ACCB
×	×

IX
×

C	V
×	×

Z	N
×	×

I	H
●	×

● : Not affected  
 × : Undefined  
 ↓ : Result

Specifications:

ROM (Bytes): 55  
 RAM (Bytes): 4  
 Stack (Bytes): 0  
 No. of cycles: 49  
 Reentrant: No  
 Relocatable: No  
 Interrupt OK?: Yes

Description:

1. Function Details

a. Argument details

VCOUNT (RAM): Duty rate as a hexadecimal number.

(Duty rate=1/4 actual duty rate. See "2. User Notes".)

HTIME (RAM) : High period of pulse.

LTIME (RAM) : Low period of pulse.

HLOUT (RAM) : Flag indicating which output is performed: Low consecutive output, High consecutive output, or pulse output.

Table 3-3 shows flag functions.

Specifications Notes:

"No. of cycles" in "Specifications" represents the number of cycles required when duty rate is other than 0% or 100%.

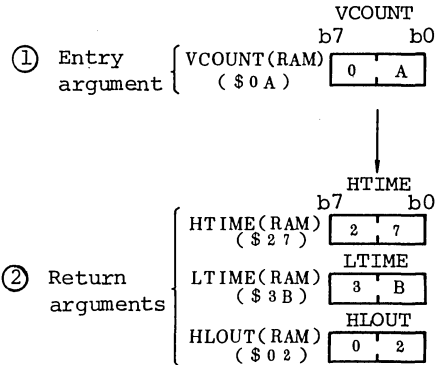
Description:

Table 3-3. Flag Functions

Label	bit 1	bit 0	Function
HLOUT	0	0	Outputs Low consecutively from Tout3 pin.
	1	0	Outputs Pulse from Tout3 pin.
	1	1	Outputs High consecutively from Tout3 pin.

b. Example of DUSET execution is shown in figure 3-5. If entry argument is as shown in part ① of figure 3-5, High and Low period of duty 40% pulse are stored as shown in part ② of figure 3-5.

c. DUSET calls neither the program modules nor subroutines.



2. User Notes

The following procedure must be executed before DUSET execution.

- a. Reserve the High and Low period of pulse in a data table.
- b. Load entry argument.
- c. When specifying duty rate, load the actual duty rate divided by 4, since duty rate is defined every 4%.
- d. Data in VCOUNT(RAM) must be within the range  $0 \leq \text{VCOUNT} \leq 16$ , otherwise neither the High nor Low period of pulse can be obtained.

Figure 3-5. Example of DUSET Execution

3. RAM Allocation

Label	RAM	Description
HTIME	b7 b0	} High period of pulse
LTIME		
HLOUT		} Pulse status flag
VCOUNT		} Duty rate



Description:

## 4. Sample Application

```

LDAA    #$10    }
STAA    VCOUNT } ..... Load entry argument.
JSR     DUSET    ..... Call DUSET.

```

## 5. Basic Operation

- a. DUSET execution stores High and Low period of pulse in HTIME : LTIME(RAM) from a data table based on duty rate  
Data in data table contains actual High and Low periods less 1 as shown in Formula 1.

$$\text{Cycle of pulse} = \{(\text{High period of pulse} + 1) + (\text{Low period of pulse} + 1)\} \times 128\mu\text{s} = 12.8 \text{ ms} \dots \dots \dots (\text{Formula 1})$$

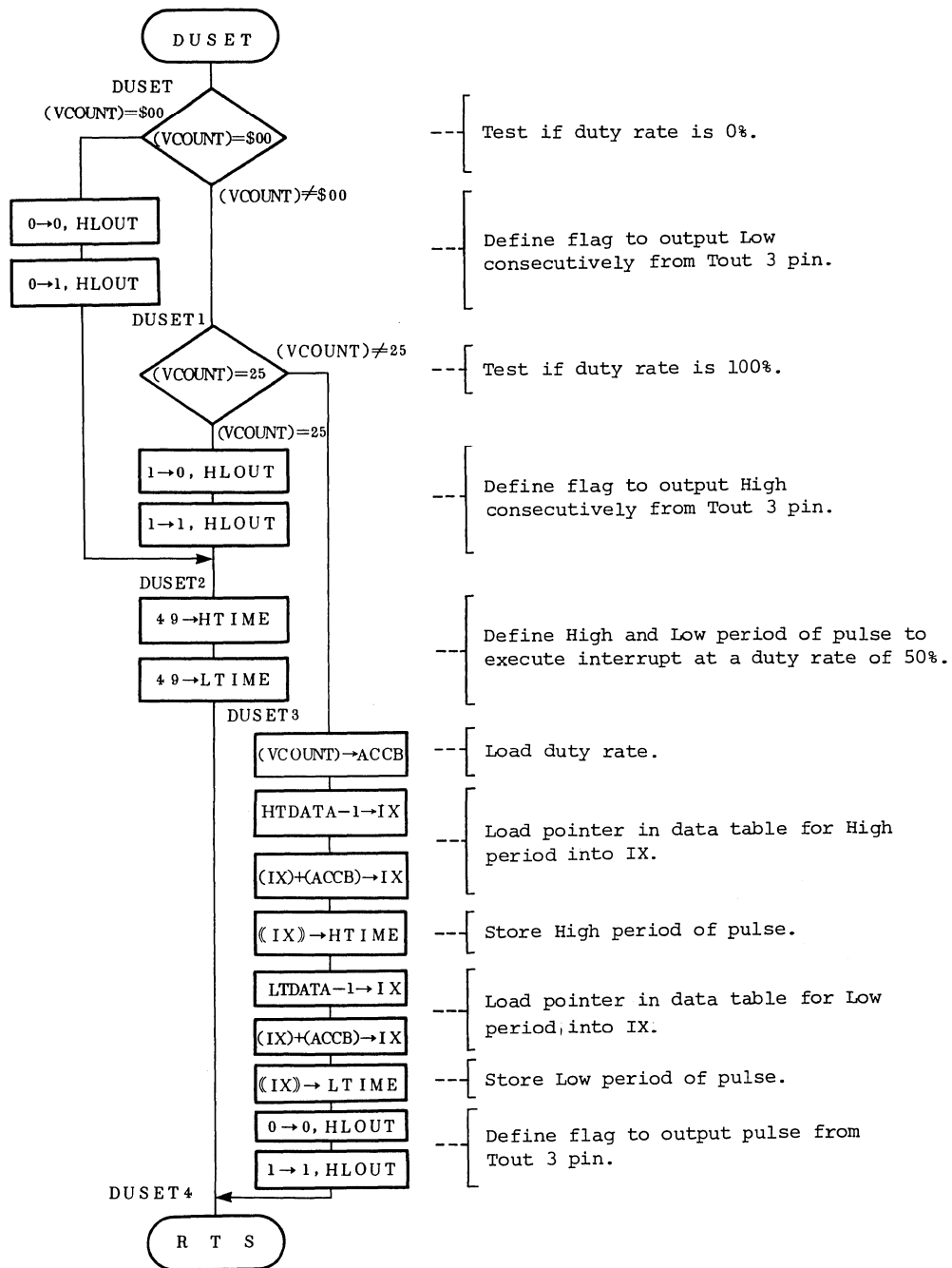
- b. i. VCOUNT(RAM) is used as a pointer to data table.

- ii. Duty rate is checked and:

If duty rate is 0% or 100%, define pulse status flag to output Low or High consecutively. Next, hold High and Low period of pulse so that counter match interrupt is executed at a duty rate of 50%.

If duty rate is 4 ~ 96%, load High and Low period of pulse using index addressing mode, and then define flag to output pulse.

Flowchart:



Program Module Name: OUTPUT PULSE

MCU/MPU: HD6301Y0/  
HD6303Y

Label: DUOUT

Function:

Output pulse from Tout3 pin.

Arguments:

Contents	Storage Location	No. of Bytes
Entry High period of pulse	HTIME (RAM)	1
Low period of pulse	LTIME (RAM)	1
Pulse status flag	HLOUT (RAM)	1
Re- turns	—	—

Changes in CPU

Registers and Flags:

ACCD	
ACCA	ACCB
x	•

IX
•

C	V
•	x

Z	N
x	x

I	H
•	•

• : Not affected  
x : Undefined  
↓ : Result

Specifications:

ROM (Bytes): 35  
RAM (Bytes): 3  
Stack (Bytes): 0  
No. of cycles: 41  
Reentrant: No  
Relocatable: No  
Interrupt OK?: No

Description:

1. Function Details

a. Argument details

HTIME (RAM): High period of pulse.

LTIME (RAM): Low period of pulse.

HLOUT (RAM): Flag indicating which output is performed:

Low consecutive output, High consecutive output, or pulse output.

Table 3-4 shows flag functions.

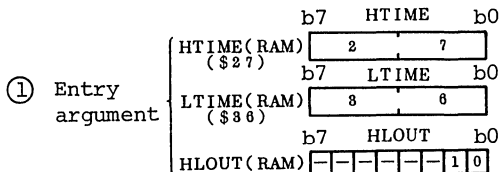
Specifications Notes:

Description:

Table 3-4. Flag Functions

Label	bit 1	bit 0	Function
HLOUT	0	0	Output Low consecutively from Tout3 pin.
	1	0	Output Pulse from Tout3 pin.
	1	1	Output High consecutively from Tout3 pin.

b. Example of DUOUT execution is shown in figure 3-6. If entry arguments are as shown in part ① of figure 3-6, duty rate 40% pulse is output as shown in part ② of figure 3-6.



c. DUOUT calls neither the program modules nor subroutines.

2. User Notes

a. Initialize timer control register to enable counter match interrupt, enable clock input to timer 2 up counter, select E clock 1/128.

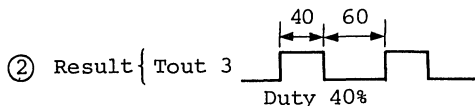


Figure 3-6. Example of DUOUT Execution

b. Enable interrupts.

c. Store entry arguments.

3. RAM Allocation

Label	RAM	Description
HTIME	b7 } [ ] b0 }	High period of pulse
LTIME	[ ]	Low period of pulse
HLOUT	[ ]	Pulse status flag

Description:

## 4. Sample Application

```

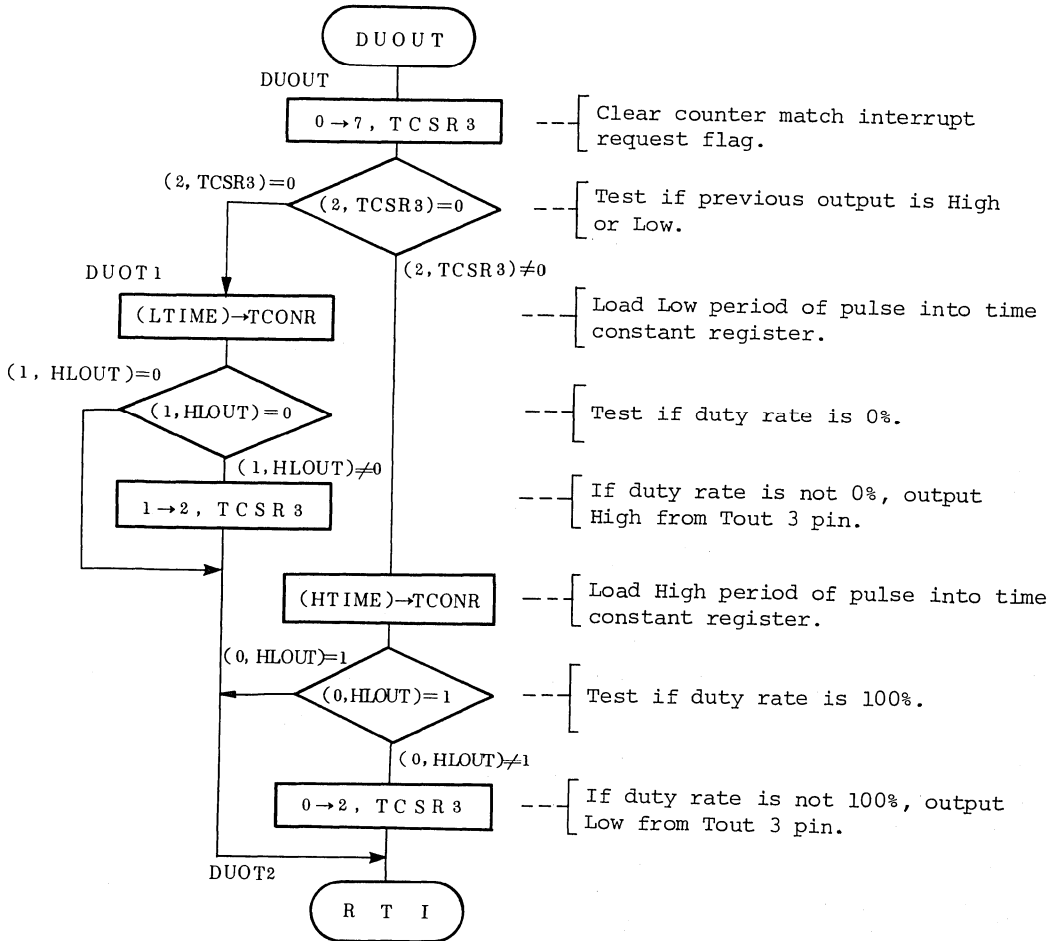
WORK1   RMB      1      ..... Reserve memory for duty rate.
        |
        | LDAA    #5A    } ..... Initialize timer control register.
        | STAA    TCSR3 }
        |
        | CLI     ..... Enable interrupt.
        |
        | LDAA    WORK1  } ..... Load entry argument of DUSET.
        | STAA    VCOUNT }
        | JSR     DUSET  }
        |
        |

```

## 5. Basic Operation

- a. Data in HTIME(RAM), LTIME(RAM) is loaded in time constant register, and pulse output.
- b.
  - i. Output status is checked and converted (High→Low or Low→High) at every timer interrupt. At this time, High and Low period of pulse is loaded into time constant register.
  - ii. If duty rate is 0% or 100%, define time constant register at a duty rate of 50%, and maintain output status (High→High or Low→Low).

Flowchart:



### 3.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

### 3.5 PROGRAM LISTING

```
00001          *
00002          ****  RAM ALLOCATION  ****
00003          *
00004A 0040          ORG    $40
00005          *
00006A 0040    0001  A DATA  RMB    1      Duty data
00007A 0041    0001  A VCOUNT RMB    1      Duty rate
00008A 0042    0001  A HTIME   RMB    1      High period of pulse
00009A 0043    0001  A LTIME   RMB    1      Low period of pulse
00010A 0044    0001  A HLOUT   RMB    1      Pulse status flag
00011A 0045    0001  A WORK    RMB    1      Work area for entry argument
00012          *
00013          ****  SYMBOL DEFINITIONS ****
00014          *
00015          001B  A TCSR3  EQU    $1B      Timer control/status register3
00016          001C  A TCONR  EQU    $1C      Time constant register
00017          ****
00018          *
00019          *      MAIN PROGRAM : DUMN      *
00020          *
00021          ****
00022          *
00023A C000          ORG    $C000
00024          *
00025A C000 8E 013F  A DUMN   LDS    #$13F    Initialize stack pointer
00026A C003 4F          CLRA          Clear RAM for work area
00027A C004 97 45      A          STAA   WORK
00028A C006 86 5A      A          LDAA   #$5A    Initialize TCSR3
00029A C008 97 1B      A          STAA   TCSR3
00030A C00A 0E          CLI          Enable interrupt
00031A C00B 96 45      A DUMN1  LDAA   WORK    Load entry argument of DUSET
00032A C00D 97 41      A          STAA   VCOUNT
00033A C00F 8D 3E C04F BSR    DUSET    Execute DUSET
00034A C011 96 41      A          LDAA   VCOUNT    Increment duty for next pulse
00035A C013 4C          INCA          DUTY+4%→entry argument
00036A C014 97 41      A          STAA   VCOUNT
00037A C016 81 1A      A          CMPA   #26    DUTY=104% ?
00038A C018 26 03 C01D BNE    DUMN2
00039A C01A 4F          CLRA          DUMN2
00040A C01B 97 41      A          STAA   VCOUNT    Store 0% duty
00041A C01D 97 45      A DUMN2  STAA   WORK    Store duty in entry argument
00042A C01F CE 00FF  A          LDX    #$FF    Execute 0.3s software timer
00043A C022 86 FF      A DUMN3  LDAA   #$FF
00044A C024 4A          DUMN4  DECA
00045A C025 26 FD C024 BNE    DUMN4
00046A C027 09          DEX
00047A C028 26 F8 C022 BNE    DUMN3
00048A C02A 20 DF C00B BRA    DUMN1
00049          ****
00050          *
00051          *      NAME : DUOUT (OUTPUT PULSE)      *
00052          *
00053          ****
00054          *
00055          *      ENTRY : HTIME (HIGH PERIOD OF PULSE)      *
00056          *      LTIME (LOW PERIOD OF PULSE)      *
00057          *      HLOUT (PULSE STATUS FLAG)      *
```

```

00058          *   RETURNS : NOTHING          *
00059          *                               *
00060          *****
00061A C02C 71 7F 1B DUOUT BCLR 7.TCSR3 Clear interrupt request flag
00062A C02F 7B 04 1B      BTST 2.TCSR3 Output is High or Low ?
00063A C032 27 0E C042    BEQ  DUOT1 Branch if Low output
00064A C034 96 42      A    LDAA HTIME Store High period in TCONR
00065A C036 97 1C      A    STAA TCONR
00066A C038 7B 01 44      BTST 0.HLOUT DUTY=100% ?
00067A C03B 26 11 C04E    BNE  DUOT2
00068A C03D 71 FB 1B      BCLR 2.TCSR3 Output Low
00069A C040 20 0C C04E    BRA  DUOT2
00070A C042 96 43      A DUOT1 LDAA LTIME Store Low period in TCONR
00071A C044 97 1C      A    STAA TCONR
00072A C046 7B 02 44      BTST 1.HLOUT DUTY=0% ?
00073A C049 27 03 C04E    BEQ  DUOT2
00074A C04B 72 04 1B      BSET 2.TCSR3 Output High
00075A C04E 3B          DUOT2 RTI
00076          *****
00077          *                               *
00078          *   NAME : DUSET (SET DUTY)       *
00079          *                               *
00080          *****
00081          *                               *
00082          *   ENTRY : VCOUNT (DUTY RATE)   *
00083          *   RETURNS : HTIME (HIGH PERIOD OF PULSE) *
00084          *           LTIME (LOW PERIOD OF PULSE)  *
00085          *           HLOUT (PULSE STATUS FLAG)   *
00086          *                               *
00087          *****
00088A C04F 96 41      A DUSET LDAA VCOUNT
00089A C051 26 0B C05B    BNE  DUSET1 Test if duty=0%
00090A C053 71 FE 44      BCLR 0.HLOUT Define flag to output Low
00091A C056 71 FD 44      BCLR 1.HLOUT
00092A C059 20 0A C065    BRA  DUSET2
00093A C05B 81 19      A DUSET1 CMPA #25 Test if duty=100%?
00094A C05D 26 0E C06D    BNE  DUSET3
00095A C05F 72 01 44      BSET 0.HLOUT Define flag to output High
00096A C062 72 02 44      BSET 1.HLOUT
00097A C065 86 31      A DUSET2 LDAA #49 Set 50% duty rate
00098A C067 97 42      A    STAA HTIME
00099A C069 97 43      A    STAA LTIME
00100A C06B 20 18 C085    BRA  DUSET4
00101A C06D 06 41      A DUSET3 LDAB VCOUNT
00102A C06F CE C085      A    LDX #HTDATA-1 Load duty rate
00103A C072 3A          ABX
00104A C073 A6 00      A    LDAA 0.X Load High period pointer into IX
00105A C075 97 42      A    STAA HTIME Store High period of pulse
00106A C077 CE C09D      A    LDX #LTDATA-1 Load Low period pointer into IX
00107A C07A 3A          ABX
00108A C07B A6 00      A    LDAA 0.X
00109A C07D 97 43      A    STAA LTIME Store Low period of pulse
00110A C07F 71 FE 44      BCLR 0.HLOUT Define flag to output pulse
00111A C082 72 02 44      BSET 1.HLOUT
00112A C085 39          DUSET4 RTS
00113          *****
00114          *                               *

```



		*	DATA TABLE	*
00115		*		*
00116		*		*
00117		*****		
00118A	C086	03	A HTDATA FCB 3	*High period of pulse
00119A	C087	07	A FCB 7	
00120A	C088	0B	A FCB 11	
00121A	C089	0F	A FCB 15	
00122A	C08A	13	A FCB 19	
00123A	C08B	17	A FCB 23	
00124A	C08C	1B	A FCB 27	
00125A	C08D	1F	A FCB 31	
00126A	C08E	23	A FCB 35	
00127A	C08F	27	A FCB 39	
00128A	C090	2B	A FCB 43	
00129A	C091	2F	A FCB 47	
00130A	C092	33	A FCB 51	
00131A	C093	37	A FCB 55	
00132A	C094	3B	A FCB 59	
00133A	C095	3F	A FCB 63	
00134A	C096	43	A FCB 67	
00135A	C097	47	A FCB 71	
00136A	C098	4B	A FCB 75	
00137A	C099	4F	A FCB 79	
00138A	C09A	53	A FCB 83	
00139A	C09B	57	A FCB 87	
00140A	C09C	5B	A FCB 91	
00141A	C09D	5F	A FCB 95	
00142A	C09E	5F	A LTDATA FCB 95	*Low period of pulse
00143A	C09F	5B	A FCB 91	
00144A	COA0	57	A FCB 87	
00145A	COA1	53	A FCB 83	
00146A	COA2	4F	A FCB 79	
00147A	COA3	4B	A FCB 75	
00148A	COA4	47	A FCB 71	
00149A	COA5	43	A FCB 67	
00150A	COA6	3F	A FCB 63	
00151A	COA7	3B	A FCB 59	
00152A	COA8	37	A FCB 55	
00153A	COA9	33	A FCB 51	
00154A	COAA	2F	A FCB 47	
00155A	COAB	2B	A FCB 43	
00156A	COAC	27	A FCB 39	
00157A	COAD	23	A FCB 35	
00158A	COAE	1F	A FCB 31	
00159A	COAF	1B	A FCB 27	
00160A	COB0	17	A FCB 23	
00161A	COB1	13	A FCB 19	
00162A	COB2	0F	A FCB 15	
00163A	COB3	0B	A FCB 11	
00164A	COB4	07	A FCB 7	
00165A	COB5	03	A FCB 3	
00166			*****	
00167		*		*
00168		*	VECTOR ADDRESSES	*
00169		*		*
00170		*****		
00171		*		*

00172A	FFEA			DRG	\$FFEA	
00173			*			
00174A	FFEA	C000	A	FDB	DUMN	IRQ2
00175A	FFEC	C02C	A	FDB	DUOUT	CMI
00176A	FFEE	C000	A	FDB	DUMN	TRAP
00177A	FFF0	C000	A	FDB	DUMN	SIO
00178A	FFF2	C000	A	FDB	DUMN	TOI
00179A	FFF4	C000	A	FDB	DUMN	OCI
00180A	FFF6	C000	A	FDB	DUMN	ICI
00181A	FFF8	C000	A	FDB	DUMN	IRQ1/ISF
00182A	FFFA	C000	A	FDB	DUMN	SWI
00183A	FFFC	C000	A	FDB	DUMN	NMI
00184A	FFFE	C000	A	FDB	DUMN	RES
00185			*			
00186				END		

## SECTION 4. PULSE WIDTH MEASUREMENT

### 4.1 HARDWARE DESCRIPTION

#### 4.1.1 Function

Measures the High period of a pulse to determine pulse width in the range from  $100\mu\text{s}$  to  $65535\mu\text{s}$  \* with an accuracy of plus or minus  $1\mu\text{s}$ ; stores result as a binary coded decimal (BCD) number.

#### 4.1.2 Microcomputer Operation

The HD6301Y0 uses the input capture interrupt function of timer 1 to fetch values in the free running counter on the rising and falling edges of the Tin pin, using the difference between these values to measure the pulse width.

#### 4.1.3 Circuit Diagram

Pulse width measurement circuit is shown in figure 4-1.

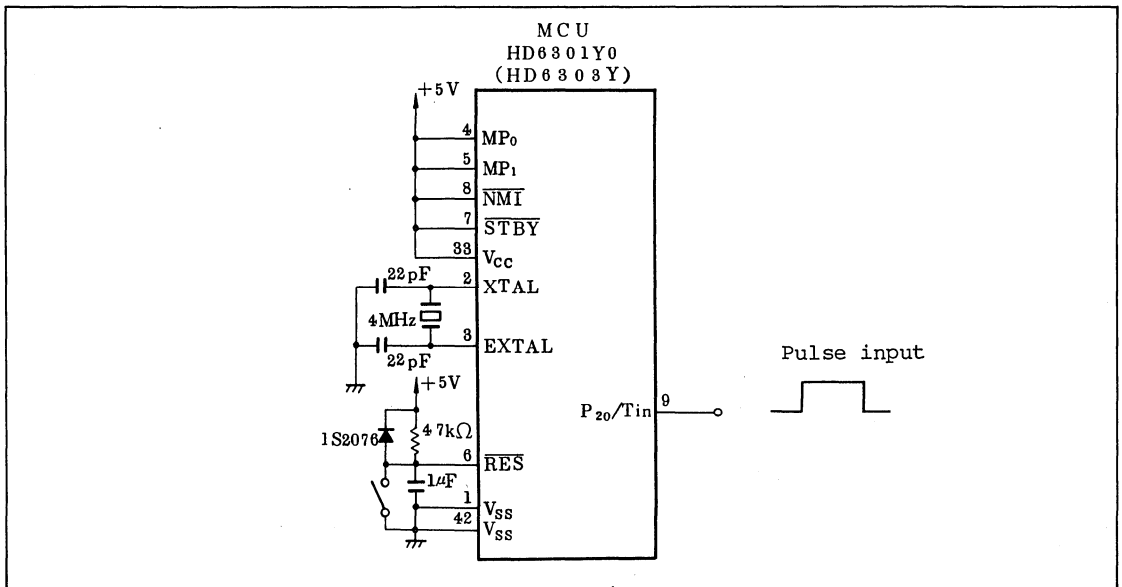


Figure 4-1. Pulse Width Measurement Circuit

#### 4.1.4 Pin Functions

Pin functions for pulse width measurement is shown in table 4-1.

Table 4-1. Pin Functions

Pin Name (HD6301Y0)	Input/ Output	Active level (High or Low)	Function	Program Label
P20/Tin	Input	—	Detects rising and falling edges.	—

#### 4.1.5 Hardware Operation

Figure 4-2 shows pulse width measurement. Since oscillator frequency is 4 MHz, E clock cycle is 1 $\mu$ s. In figure 4-2, pulse width is 4 $\mu$ s.

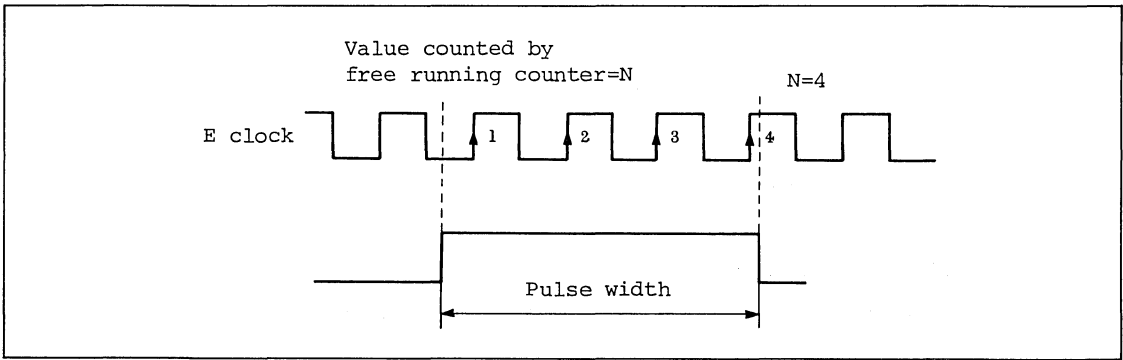


Figure 4-2. Measure Pulse Width

4.2.1 Program Module Configuration

The program module configuration for pulse width measurement and BCD conversion is shown in figure 4-3.

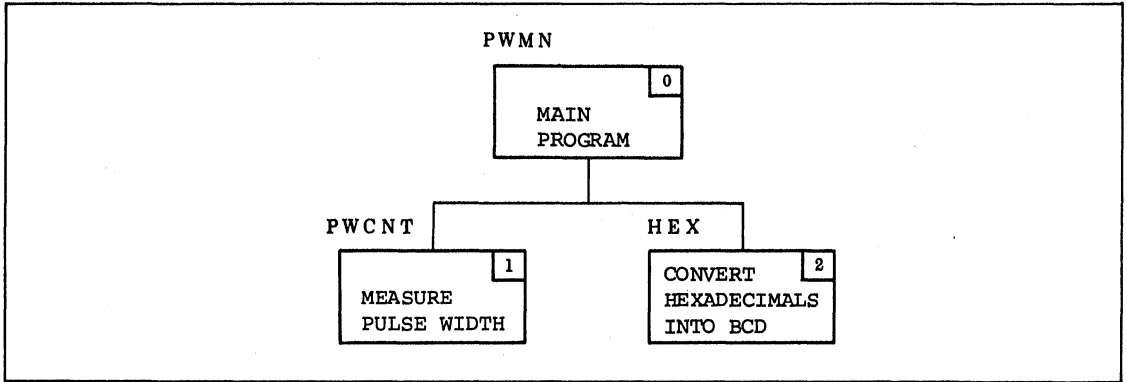


Figure 4-3. Program Module Configuration

4.2.2 Program Module Functions

Program module functions are summarized in table 4-2.

Table 4-2. Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	PWMN	Measures pulse width as a BCD number.
1	MEASURE PULSE WIDTH	PWCNT	Obtains pulse width as a 2-byte hexadecimal number.
2	CONVERT HEXADECIMALS INTO BCD	HEX	Converts 2-byte hexadecimal number into BCD number. (Refer to HEX in HD6301/HD6303 FAMILY APPLICATION NOTES (SOFTWARE) for details.

### 4.2.3 Program Module Process Flow (Main Program)

The flowchart in figure 4-4 is an example of the pulse width measurement performed by the program module in figure 4-3.

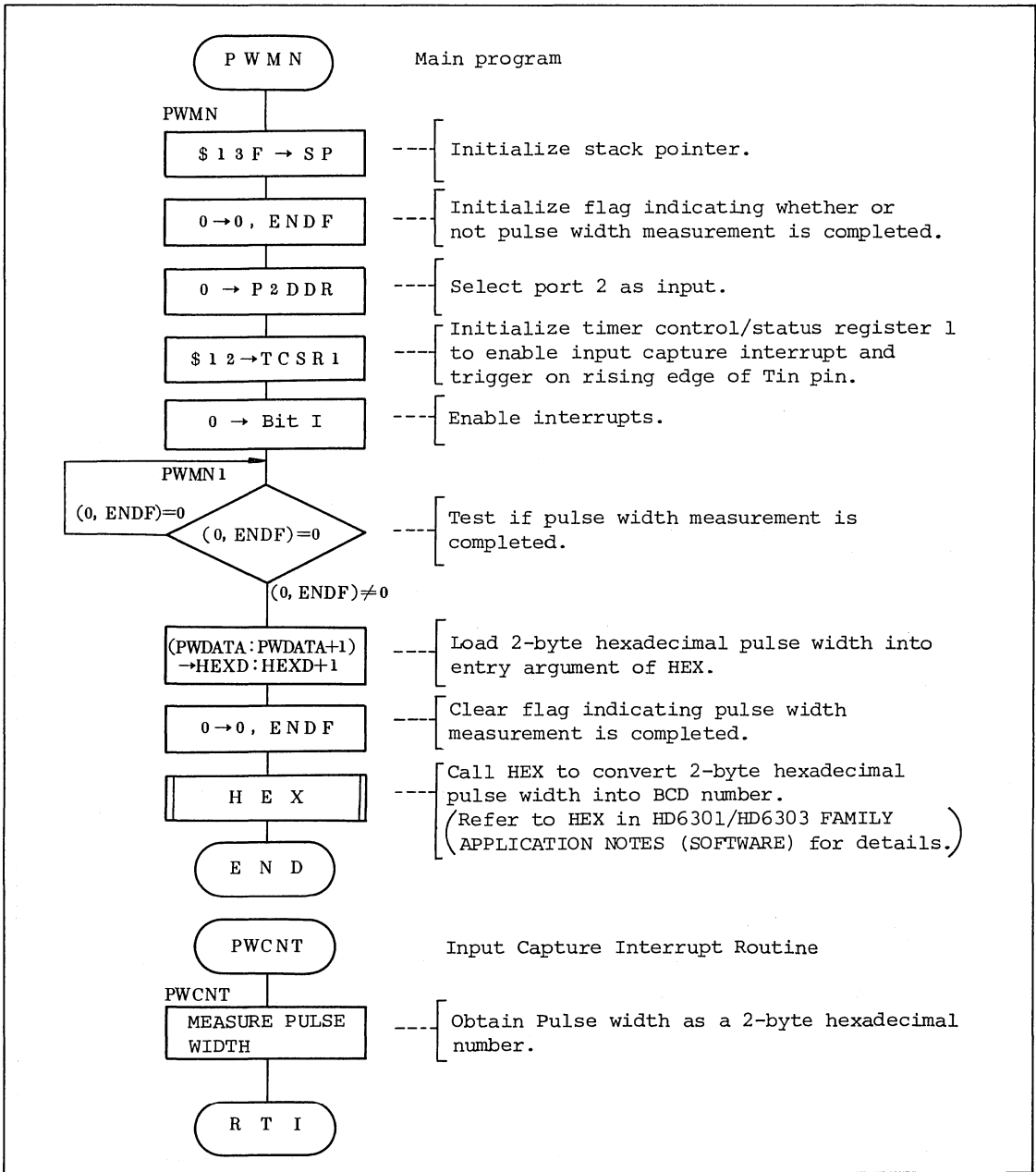


Figure 4-4. Program Module Flowchart

#### 4.3 PROGRAM MODULE DESCRIPTION

**Program Module Name:** MEASURE PULSE WIDTH

**MCU/MPU:** HD6301Y0/  
HD6303Y

**Label:** PWCNT

**Function:**

Obtains pulse width as a 2-byte hexadecimal number, and stores result in PWDAT (RAM).

**Arguments:**

Contents	Storage Location	No. of Bytes
Entry	_____	_____

Re- turns	Pulse width	PWDATA (RAM)	2
	Flag in- dicating completion of measure- ment	ENDF (RAM)	1

**Changes in CPU**

**Registers and Flags:**

ACCD	
ACCA	ACCB
x	x

IX
●

C	V
x	x

Z	N
x	x

I	H
●	●

● : Not affected  
x : Undefined  
↑ : Result

**Specifications:**

ROM (Bytes): 29  
RAM (Bytes): 5  
Stack (Bytes): 0  
No. of cycles: 400  
Reentrant: No  
Relocatable: No  
Interrupt OK?: No

**Description:**

1. Function Details

a. Argument details

PWDATA (RAM): Contains pulse width as a 2-byte hexadecimal number.

ENDF (RAM) : Contains flag indicating whether or not pulse width measurement is completed.

Table 4-3 shows flag function.

**Specifications Notes:**

Program Module Name: MEASURE PULSE  
WIDTH

MCU/MPU: HD6301Y0/  
HD6303Y

Label: PWCNT

Description:

Table 4-3. Flag Functions

Label	bit 0	Function
ENDF	0	Indicates pulse width measurement is not completed.
	1	Indicates pulse width measurement is completed.

- b. Example of PWCNT execution is shown in figure 4-5. If pulse, whose High period of pulse is 150  $\mu$ s, is input as shown in part ① of figure 4-5, measurement result is stored in PWCNT (RAM) as a hexadecimal number and "1" is stored in ENDF (RAM).

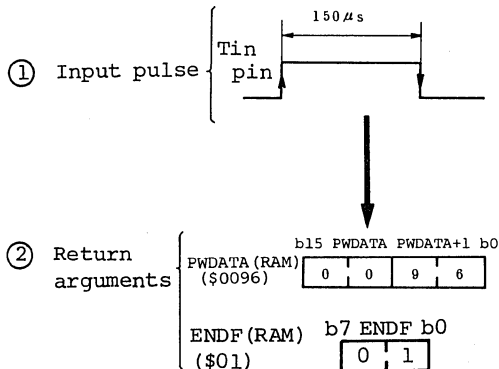


Figure 4-5. Example of PWCNT Execution

- c. PWCNT calls neither the program modules nor subroutines.

2. User Notes

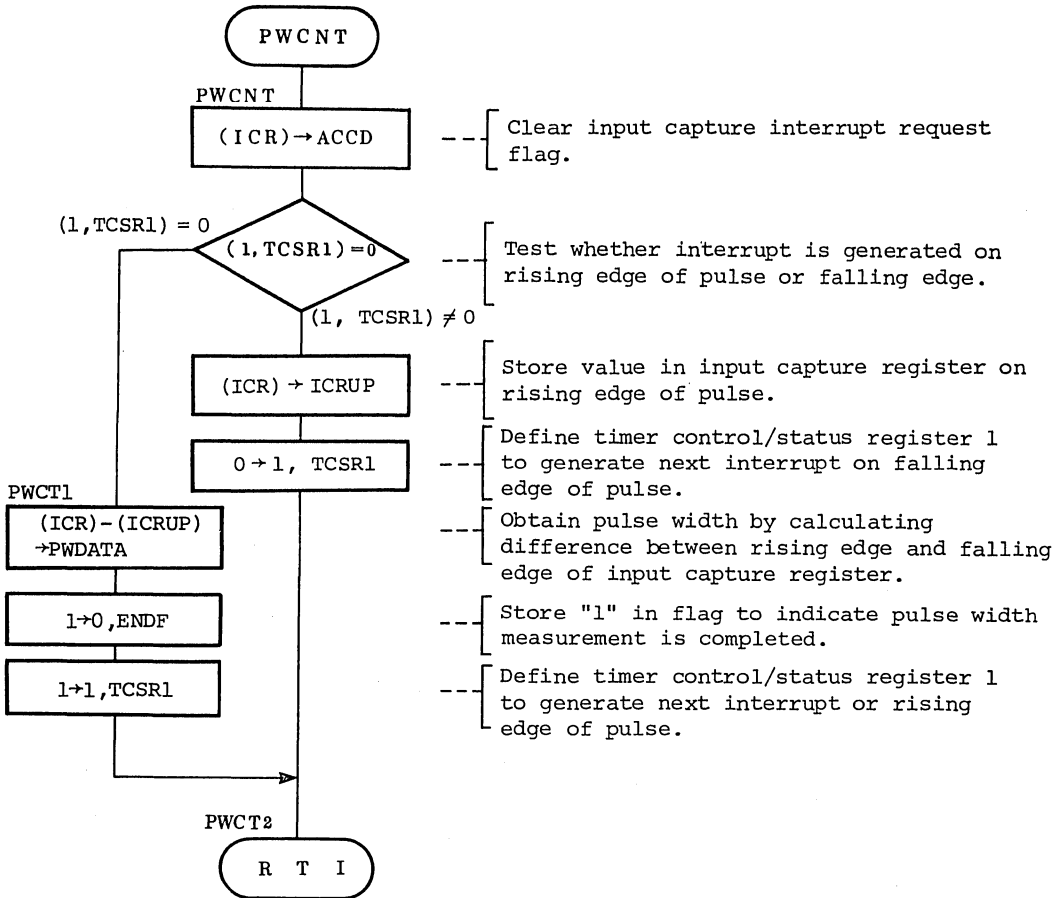
The following procedure must be performed before PWCNT execution.

- Initialize flag indicating whether or not pulse width measurement is completed.
- Select bit 0 of port 2 as input.
- Initialize timer control/status register 1 to enable input capture interrupt and trigger on rising edge of Tin pin.
- Enable interrupts.





Flowchart:



#### 4.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

#### 4.5 PROGRAM LISTING

```

00001          *
00002          ****  RAM ALLOCATION  *****
00003          *
00004A 0040          ORG      $40
00005          *
00006A 0040      0002  A  ICRUP  RMB      2      ICR data on rising edge
00007A 0042      0002  A  PWDATA RMB      2      Pulse width data
00008A 0044      0001  A  ENDF   RMB      1      Measurement completion flag
00009A 0045      0002  A  HEXD   RMB      2      Pulse width in HEX data
00010A 0047      0003  A  DECD   RMB      3      Pulse width in BCD data
00011          *
00012          ****  SYMBOL DEFINITIONS *****
00013          *
00014          0001  A  P2DDR  EQU     $01      Port 2 data direction register
00015          0008  A  TCSR1  EQU     $08      Timer control/status register1
00016          000D  A  ICR    EQU     $0D      input capture register
00017          *****
00018          *
00019          *          MAIN PROGRAM : PWMN          *
00020          *
00021          *****
00022          *
00023A C000          ORG      $C000
00024          *
00025A C000 8E 013F A  PWMN   LDS     #$13F   Initialize stack pointer
00026A C003 71 FE 44          BCLR   0.ENDF   Clear flag
00027A C006 4F          CLRA
00028A C007 97 01      A      STAA   P2DDR
00029A C009 86 12      A      LDAA   #$12      Initialize TCSR1
00030A C00B 97 08      A      STAA   TCSR1
00031A C00D 0E          CLI
00032A C00E 7B 01 44  PWMN1  BTST   0.ENDF   Test if measure pulse width end?
00033A C011 27 FB C00E  BEQ     PWMN1   Branch if not
00034A C013 DC 42      A      LDD   PWDATA   Load pulse width in HEX data
00035A C015 DD 45      A      STD   HEXD
00036A C017 71 FE 44          BCLR   0.ENDF   Clear flag
00037A C01A BD C038 A      JSR    HEX      Convert HEX data into BCD data
00038A C01D 20 FE C01D  PEND   BRA     PEND   End of program
00039          *****
00040          *
00041          *          NAME : PWCNT (MEASURE PULSE WIDTH)          *
00042          *
00043          *****
00044          *
00045          *          ENTRY : NOTHING          *
00046          *          RETURNS : PWDATA (PULSE WIDTH)          *
00047          *          ENDF   (MEASUREMENT COMPLETION FLAG)*
00048          *
00049          *****
00050A C01F DC 0D      A  PWCNT  LDD     ICR      Clear ICF
00051A C021 7B 02 08          BTST   1.TCSR1   Branch if IEDG=0
00052A C024 27 07 C02D  BEQ     PWCT1
00053A C026 DD 40      A      STD   ICRUP
00054A C028 71 FD 08          BCLR   1.TCSR1   Define TCSR1
00055A C02B 20 0A C037  BRA     PWCT2
00056A C02D 93 40      A  PWCT1  SUBD   ICRUP   Calculate pulse width
00057A C02F DD 42      A      STD   PWDATA   store pulse width in PWDATA

```

```

00058A C031 72 01 44      BSET 0.ENDF Set flag
00059A C034 72 02 08      BSET 1.TCSR1 Define TCSR1
00060A C037 3B      PWCT2 RTI
00061      *****
00062      *
00063      *      NAME : HEX (CONVERT 2-BYTE HEXADECIMAL      *
00064      *      NUMBER INTO 5-DIGIT BCD NUMBER)*
00065      *
00066      *****
00067      *
00068      *      ENTRY : ACCD (2-BYTE HEXADECIMAL NUMBER)      *
00069      *      RETURNS : DECD (5-DIGIT BCD NUMBER)      *
00070      *
00071      *****
00072A C038 4F      HEX      CLRA      Clear ACCA
00073A C039 5F      CLR B      Clear ACCB
00074A C03A DD 47 A      STD      DECD      Clear 5-digit BCD
00075A C03C 97 49 A      STAA     DECD+2
00076A C03E C6 10 A      LDAB     #16      Store shift counter
00077A C040 7B 0046 A HEX2 ASL      HEXD+1 Shift MSB of HEXD to carry
00078A C043 79 0045 A      ROL      HEXD
00079A C046 CE 0003 A      LDX      #3      Set addition counter ADDR
00080A C049 A6 46 A HEX1 LDAA     DECD-1.X DECD * 2 + C -> ACCA
00081A C04B A9 46 A      ADCA     DECD-1.X
00082A C04D 19      DAA      Convert into BCD data
00083A C04E A7 46 A      STAA     DECD-1.X Store 5-digit BCD area
00084A C050 09      DEX      Decrement ADDR pointer
00085A C051 26 F6 C049 BNE     HEX1      Loop until ADDR pointer=0
00086A C053 5A      DECB     Decrement shift counter
00087A C054 26 EA C040 BNE     HEX2      Loop until shift counter=0
00088A C056 39      RTS
00089      *****
00090      *
00091      *      VECTOR ADDRESSES      *
00092      *
00093      *****
00094      *
00095A FFEA      ORG      $FFEA
00096      *
00097A FFEA C000 A      FDB     PWMN      IRQ2
00098A FFEC C000 A      FDB     PWMN      CMI
00099A FFEE C000 A      FDB     PWMN      TRAP
00100A FFF0 C000 A      FDB     PWMN      SIO
00101A FFF2 C000 A      FDB     PWMN      TOI
00102A FFF4 C000 A      FDB     PWMN      OCI
00103A FFF6 C01F A      FDB     PWCNT     ICI
00104A FFF8 C000 A      FDB     PWMN      IRQ1/ISF
00105A FFFA C000 A      FDB     PWMN      SWI
00106A FFFC C000 A      FDB     PWMN      NMI
00107A FFFE C000 A      FDB     PWMN      RES
00108      *
00109      END

```

## 5.1 HARDWARE DESCRIPTION

## 5.1.1 Function

Counts input pulses up to 255 pulses; the count value is returned as a binary coded decimal (BCD) number.

## 5.1.2 Microcomputer Operation

The HD6301Y0 uses TCLK pin to input pulses and timer 2 up counter to count the input pulses. Beginning and ending of pulse counting is performed by setting and clearing timer 2 enable bit in timer control/status register 3.

## 5.1.3 Circuit Diagram

Input pulse measurement circuit is shown in figure 5-1.

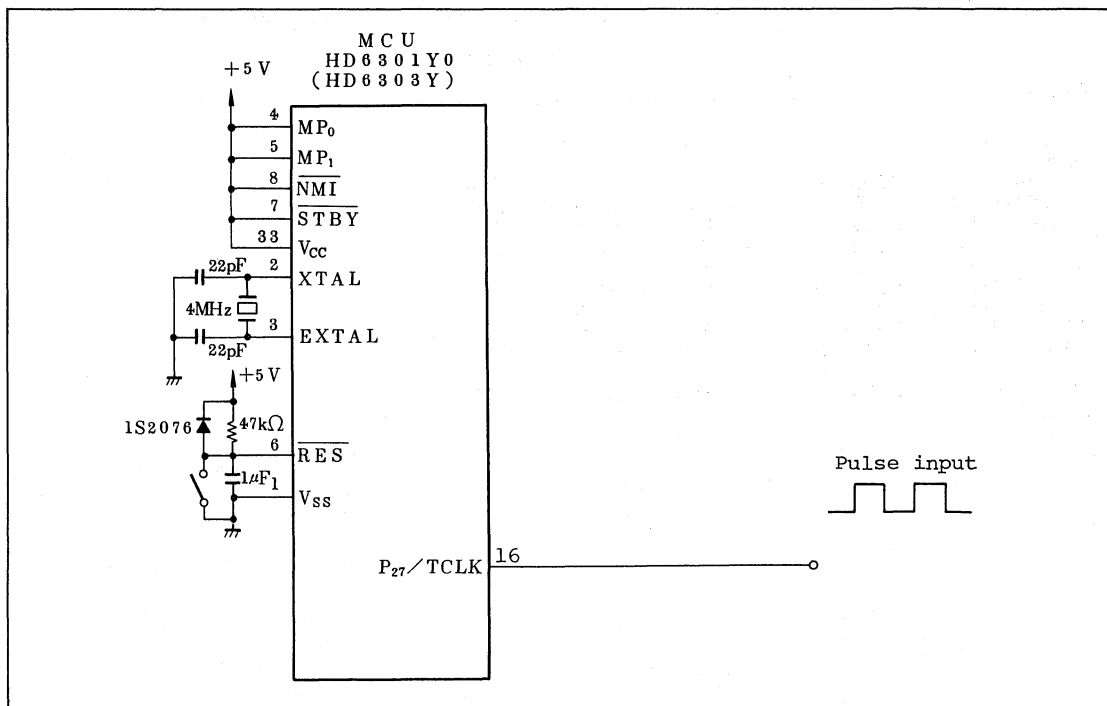


Figure 5-1. Input Pulse Measurement Circuit

#### 5.1.4 Pin Functions

Pin functions of HD6301Y0 for counting pulses is shown in Table 5-1.

Table 5-1. Pin Functions

Pin Name (HD6301Y0)	Input/ Output	Active Level (High or Low)	Function	Program Label
P27/TCLK	Input	——	Inputs pulse event	——

#### 5.1.5 Hardware Operation

Figure 5-2 shows input pulse count using TCLK pin of the HD6301Y0. To set start/end timing for counting input pulses, the procedure below must be performed in the main program.

- i. Set flag in STRTF(RAM).
- ii. Execute 200 ms software timer.
- iii. Clear flag is STRTF(RAM).

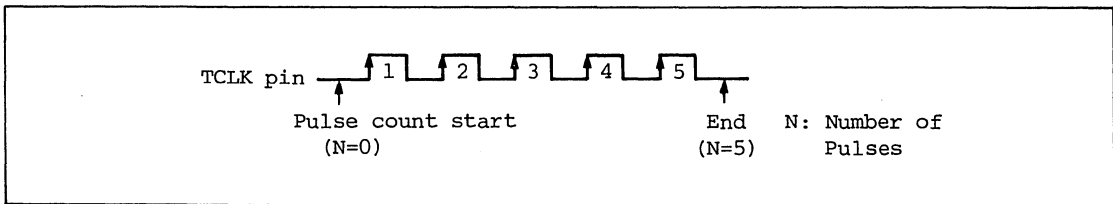


Figure 5-2. Input Pulses Count

## 5.2 SOFTWARE DESCRIPTION

### 5.2.1 Program Module Configuration

The program module configuration for input pulse count is shown in figure 5-3.

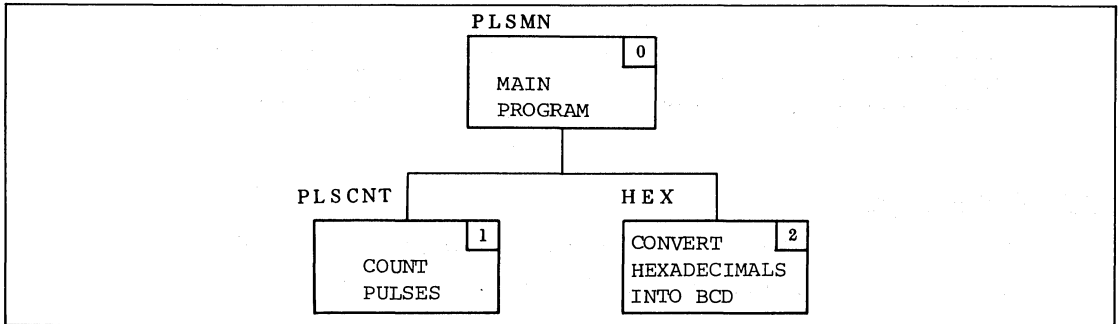


Figure 5-3. Program Module Configuration

### 5.2.2 Program Module Functions

Program module functions are summarized in table 5-2.

Table 5-2. Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	PLSMN	Counts input pulses as a BCD numbers.
1	COUNT PULSES	PLSCNT	Counts input pulses as a hexadecimal number.
2	CONVERT HEXADECIMALS INTO BCD.	HEX	Converts 2-byte hexadecimal number into BCD number. Refer to HEX in HD6301/HD6303 FAMILY APPLICATION NOTES (SOFTWARE) for details.

### 5.2.3 Program Module Process Flow (Main Program)

The flowchart in figure 5-4 is an example of counting input pulses, performed by the program module in figure 5-3.

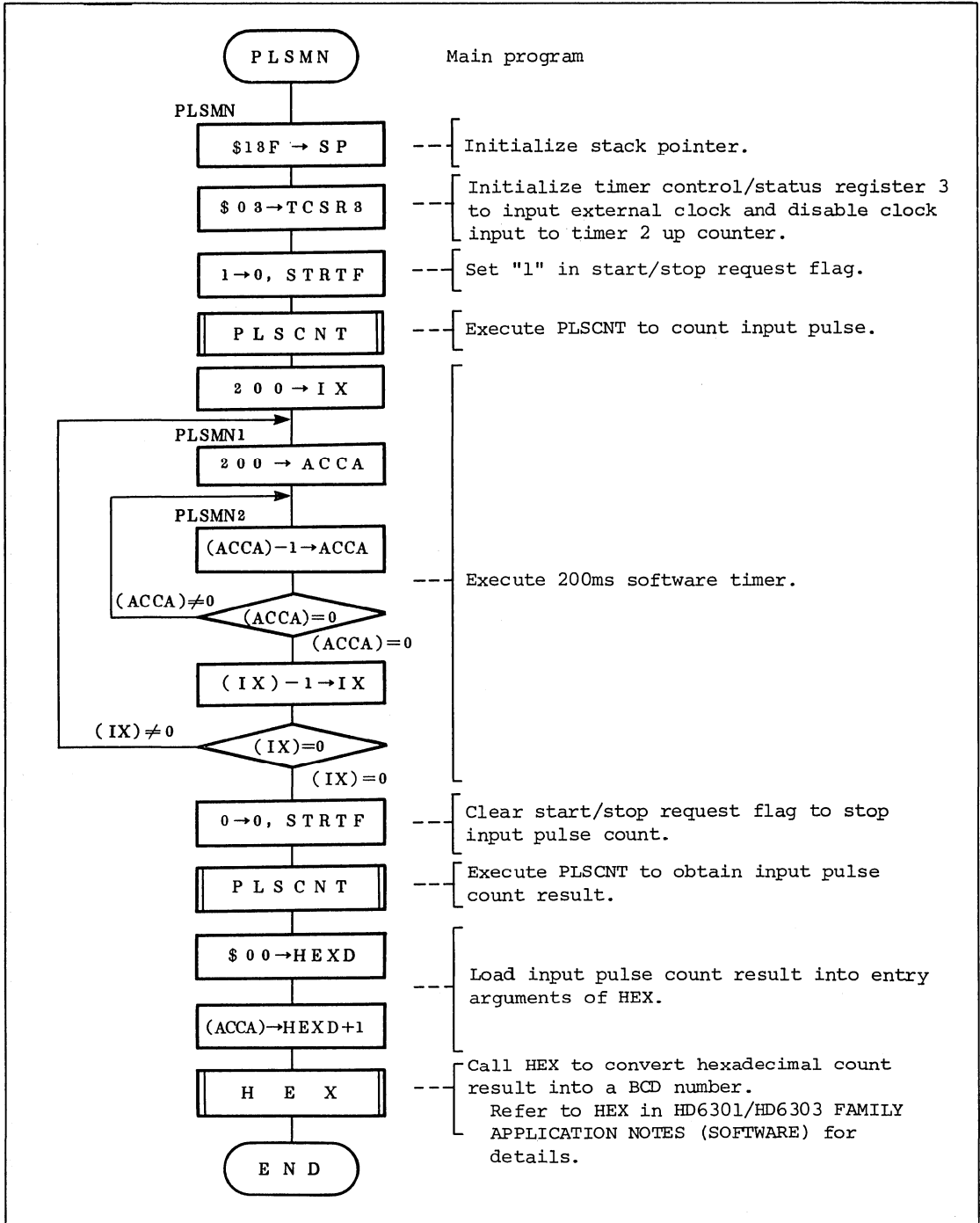


Figure 5-4. Program Module Flowchart



### 5.3 PROGRAM MODULE DESCRIPTION

Program Module Name: COUNT PULSES

MCU/MPU: HD6301Y0/  
HD6303Y

Label: PLSCNT

Function:

Counts pulses input from TCLK pin, and loads count result into ACCA.

Arguments:

Contents	Storage Location	No. of Bytes
Entry Start/stop request flag	STRTF (RAM)	1
Re- turns Pulse count result	ACCA	1

Changes in CPU

Registers and Flags:

ACCD	
ACCA	ACCB
×	●

IX
●

C	V
×	×

Z	N
×	×

I	H
●	●

● : Not affected  
 × : Undefined  
 ↓ : Result

Specifications:

ROM (Bytes): 20  
 RAM (Bytes): 1  
 Stack (Bytes): 0  
 No. of cycles: 30  
 Reentrant: No  
 Relocatable: No  
 Interrupt OK?: Yes

Description:

1. Function Details

a. Argument details

STRTF (RAM): Holds flag indicating whether input pulse count will start or stop. Table 5-3 shows flag functions.

ACCA: Contains input pulse count result as a 1-byte hexadecimal number.

Specifications Notes:

"No. of cycles" in "Specifications" indicates the number of cycles required to start input pulse count.

Description:

- b. Example of PLSCNT execution is shown in figure 5-5.
  - i. If bit 0 of entry argument STRTF(RAM) is set to "1", input pulse count starts as shown in port ① of figure 5-5.
  - ii. If bit 0 of entry argument STRTF(RAM) is set to "0" as shown in part ① of figure 5-5, input pulse count result is stored as shown in part ② of figure 5-5.

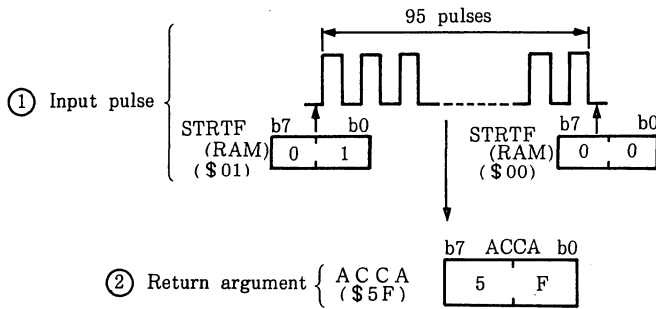


Figure 5-5. Example of PLSCNT Execution

- c. PLSCNT calls neither the program modules nor subroutines.

2. User Notes

The following procedure must be performed before PLSCNT execution.

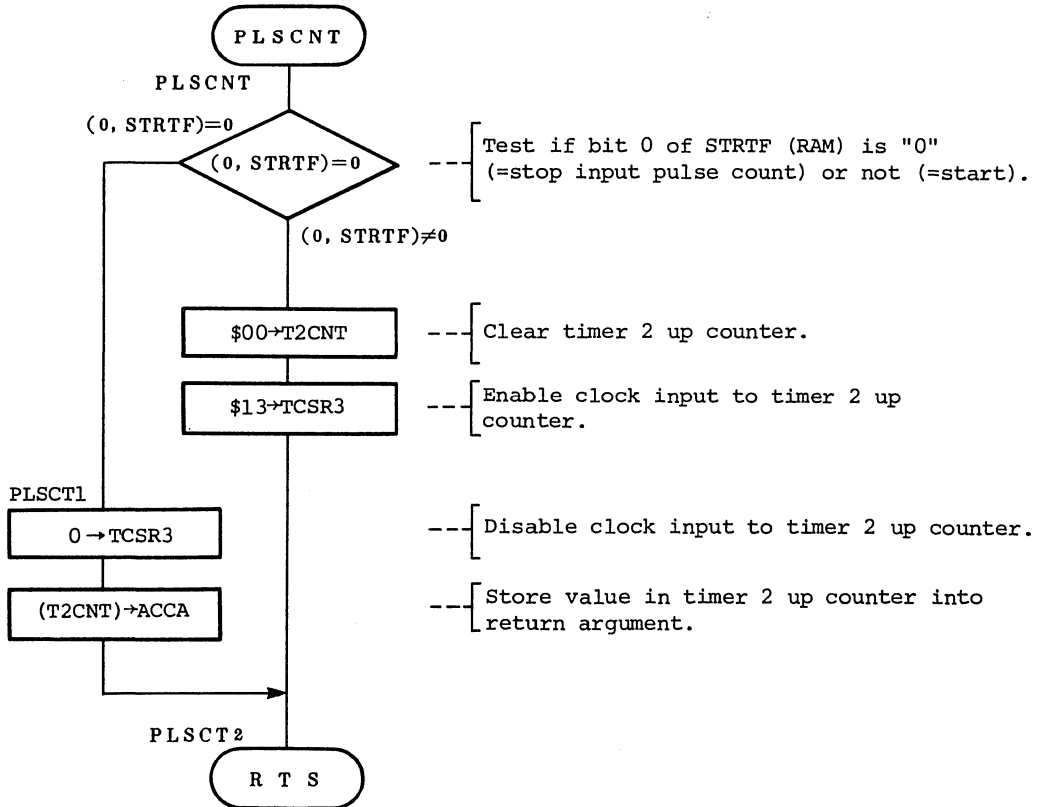
- a. Initialize timer control/status register 3 to input external clock and disable input to timer 2 up counter.
- b. Initialize entry argument indicating whether input pulse count will start or stop.

3. RAM Allocation

Label	RAM	Description
STRTF	b7 <input type="text"/> b0 <input type="text"/>	Flag indicating whether input pulse will start or stop.



Flowchart:



## 5.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

## 5.5 PROGRAM LISTING

```

00001          *
00002          **** RAM ALLOCATION *****
00003          *
00004A 0040          ORG      $40
00005          *
00006A 0040      0001  A  STRTF  RMB   1      Start/stop request flag
00007A 0041      0002  A  HEXD  RMB   2      Count result in HEX data
00008A 0043      0003  A  DECD  RMB   3      Count result in BCD data
00009          *
00010          **** SYMBOL DEFINITIONS *****
00011          *
00012          001B  A  TCSR3  EQU   $1B      Timer control/status register 3
00013          001D  A  T2CNT  EQU   $1D      Timer 2 up counter
00014          *****
00015          *
00016          *                MAIN PROGRAM : PLSMN                *
00017          *
00018          *****
00019          *
00020A C000          ORG      $C000
00021          *
00022A C000 8E 013F  A  PLSMN  LDS   #$13F   Initialize stack pointer
00023A C003 86 03   A          LDAA  #$03   Initialize TCSR3
00024A C005 97 1B   A          STAA  TCSR3
00025A C007 72 01 40 BSET  0,STRTF Set flag to request starting
00026A C00A 8D 19 C025 BSR   PLSCNT Start pulse count
00027A C00C CE 00C8  A          LDX   #200  Execute 200ms software timer
00028A C00F 86 CB   A  PLSMN1 LDAA  #200
00029A C011 4A          PLSMN2 DECA
00030A C012 26 FD C011 BNE   PLSMN2
00031A C014 09          DEX
00032A C015 26 F8 C00F BNE   PLSMN1
00033A C017 71 FE 40 BCLR  0,STRTF Clear flag to request stopping
00034A C01A 8D 09 C025 BSR   PLSCNT Stop pulse count
00035A C01C 7F 0041  A          CLR   HEXD  Clear upper byte of HEXD
00036A C01F 97 42   A          STAA  HEXD+1 Load count result into lower byte
00037A C021 8D 17 C03A BSR   HEX  Convert count result into BCD data
00038A C023 20 FE C023 PEND  BRA   PEND  End of program
00039          *****
00040          *
00041          *                NAME : PLSCNT (COUNT PULSE)                *
00042          *
00043          *****
00044          *
00045          *                ENTRY : STRTF (START/STOP REQUEST FLAG)        *
00046          *                RETURNS : ACCA (PULSE COUNT RESULT)            *
00047          *
00048          *****
00049A C025 7B 01 40  PLSCNT BTST  0,STRTF Test if count start or stop?
00050A C028 27 09 C033 BEQ   PLSC11 Branch if stop
00051A C02A 7F 001D  A          CLR   T2CNT Clear T2CNT
00052A C02D 86 13   A          LDAA  #$13 Start pulse count
00053A C02F 97 1B   A          STAA  TCSR3
00054A C031 20 06 C039 BRA   PLSC12
00055A C033 86 03   A  PLSC11 LDAA  #$03 Stop pulse count
00056A C035 97 1B   A          STAA  TCSR3
00057A C037 96 1D   A          LDAA  T2CNT Load pulse count result into ACCA

```

```

00058A C039 39      PLSC2 RTS
00059              *****
00060              *
00061              *   NAME : HEX (CONVERTING 2-BYTE HEXADECIMALS   *
00062              *           INTO 5-DIGIT BCD)                       *
00063              *
00064              *****
00065              *
00066              *           ENTRY : HEXD (2-BYTE HEXADECIMAL NUMBER)*
00067              *           RETURNS : DECD (5-DIGIT BCD NUMBER)   *
00068              *
00069              *****
00070A C03A 4F      HEX      CLRA      Clear ACCA
00071A C03B 5F      CLRAB     Clear ACCB
00072A C03C DD 43   A      STD      DECD      Clear 5-digit BCD area
00073A C03E 97 45   A      STAA     DECD+2
00074A C040 C6 10   A      LDAB     #16      Set shift counter
00075A C042 78 0042 A HEX2   ASL      HEXD+1   Shift MSB of HEXD to carry
00076A C045 79 0041 A      ROL      HEXD
00077A C048 CE 0003 A      LDX      #3       Set ADDR pointer(addition counter)
00078A C04B A6 42   A HEX1   LDAA     DECD-1.X  DECD * 2 + C -> ACCA
00079A C04D A9 42   A      ADCA     DECD-1.X
00080A C04F 19      DAA      Convert into BCD
00081A C050 A7 42   A      STAA     DECD-1.X  Store 5-digit BCD area
00082A C052 09      DEX      Decrement ADDR pointer
00083A C053 26 F6 C04B BNE     HEX1     Loop until ADDR pointer=0
00084A C055 5A      DECB     Decrement shift counter
00085A C056 26 EA C042 BNE     HEX2     Loop until shift counter=0
00086A C058 39      RTS
00087              *****
00088              *
00089              *           VECTOR ADDRESSES                       *
00090              *
00091              *****
00092              *
00093A FFEA              ORG      $FFEA
00094              *
00095A FFEA C000 A      FDB     PLSMN   IRQ2
00096A FFEC C000 A      FDB     PLSMN   CMI
00097A FFEE C000 A      FDB     PLSMN   TRAP
00098A FFF0 C000 A      FDB     PLSMN   SIO
00099A FFF2 C000 A      FDB     PLSMN   TOI
00100A FFF4 C000 A      FDB     PLSMN   OCI
00101A FFF6 C000 A      FDB     PLSMN   ICI
00102A FFF8 C000 A      FDB     PLSMN   IRQ1/ISF
00103A FFFA C000 A      FDB     PLSMN   SWI
00104A FFFC C000 A      FDB     PLSMN   NMI
00105A FFFE C000 A      FDB     PLSMN   RES
00106              *
00107              *           END

```

6.1 HARDWARE DESCRIPTION

6.1.1 Function

Performs key scan of 8 × 4 key matrix, invalidating simultaneous depression of more than 2 keys by software, and converting valid key data into ASCII characters (A~Z or 1~6).

6.1.2 Microcomputer Operation

The HD6301Y0 uses timer 1 to execute output compare interrupt 1 every 8ms. Key scan is performed by an output strobe signal through port 4, controlling DDR (data direction register) of port 4. Since all parts except port 4 are input ports (high impedance state), diodes for preventing output signal collision are not necessary. Key scan data is fetched through port 3 during the interrupt routine.

6.1.3 Peripheral Devices

8 × 4 Key matrix : Keys to be depressed.

6.1.4 Circuit Diagram

Key scan control circuit is shown in figure 6-1.





## 6.1.5 Pin Functions

Pin functions at the interface between the HD6301Y0 and the key matrix are shown in table 6-1.

Table 6-1. Pin Functions

Pin Name (HD6301Y0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (Key matrix)	Program Label
P <sub>40</sub>	Input/ Output	Low	Outputs strobe signal	KR <sub>0</sub>	P4DTR
P <sub>41</sub>	Input/ Output	Low		KR <sub>1</sub>	
P <sub>42</sub>	Input/ Output	Low		KR <sub>2</sub>	
P <sub>43</sub>	Input/ Output	Low		KR <sub>3</sub>	
P <sub>30</sub>	Input	—	Inputs key data	KC <sub>0</sub>	P3DTR
P <sub>31</sub>	Input	—		KC <sub>1</sub>	
P <sub>32</sub>	Input	—		KC <sub>2</sub>	
P <sub>33</sub>	Input	—		KC <sub>3</sub>	
P <sub>34</sub>	Input	—		KC <sub>4</sub>	
P <sub>35</sub>	Input	—		KC <sub>5</sub>	
P <sub>36</sub>	Input	—		KC <sub>6</sub>	
P <sub>37</sub>	Input	—		KC <sub>7</sub>	

## 6.1.6 Hardware Operation

The timing chart for key scan is shown in figure 6-2.

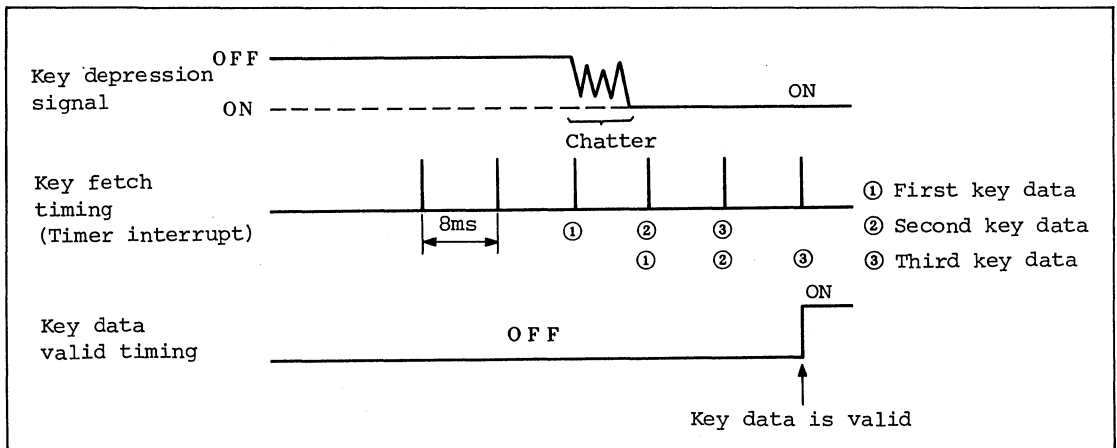


Figure 6-2. Chatter Prevention Timing

Key depression signal is checked every 8 ms. If key data is the same 3 consecutive times, it will then be valid, and invalid otherwise.

## 6.2 SOFTWARE DESCRIPTION

### 6.2.1 Program Module Configuration

The program module configuration for key scan of 8 × 4 key matrix is shown in figure 6-3.

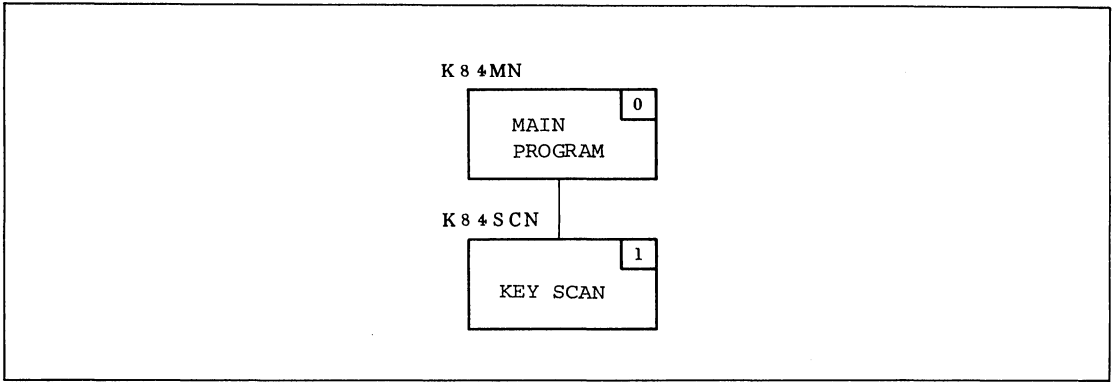


Figure 6-3. Program Module Configuration

### 6.2.2 Program Module Functions

Program module functions are summarized in table 6-2.

Table 6-2. Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	K84MN	Performs key scan of 8×4 key matrix and converts key data into ASCII.
1	KEY SCAN	K84SCN	Performs key scan of 8×4 key matrix.



### 6.3 PROGRAM MODULE DESCRIPTION

Program Module Name: KEY SCAN

MCU/MPU: HD6301Y0

Label: K84SCN

Function:

Performs key scan of 8×4 key matrix to store key data in KEYDAT(RAM).

Arguments:

Contents	Storage Location	No. of Bytes
Entry	---	---

Re- turns	Key data	KEYDAT (RAM)	1
	Key data valid/ invalid flag	KEYONF (RAM)	1

Changes in CPU

Registers and Flags:

ACCD	
ACCA	ACCB
×	×

IX
×

C	V
×	×

Z	N
×	×

I	H
●	×

● : Not affected  
 × : Undefined  
 † : Result

Specifications:

ROM (Bytes): 120  
 RAM (Bytes): 8  
 Stack (Bytes): 0  
 No. of cycles: 332  
 Reentrant: No  
 Relocatable: No  
 Interrupt OK?: Yes

Description:

1. Function Details

a. Argument details

KEYDAT(RAM): Contains key data.

KEYONF(RAM): Contains flag indicating whether or not key data is valid.  
 Table 6-3 shows flag functions.

Specifications Notes:

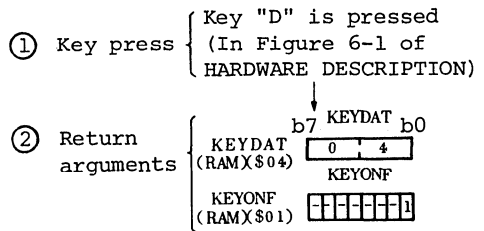
Description:

Table 6-3. Flag Functions

Label	bit 0	Function
KEYONF	0	Indicates key data is invalid.
	1	Indicates key data is valid.

b. Example of K84SCN execution is shown in figure 6-5. If a key is pressed as shown in part ① of figure 6-5, key data is stored in KEYDAT (RAM) and bit 0 of KEYONF (RAM) is set as shown in part ② of figure 6-5.

c. K84SCN calls neither the program modules nor subroutines.



2. User Notes

The following procedure must be performed before K84SCN execution.

- a. Clear RAM used by K84SCN.
- b. Initialize timer control/status register 1 to enable output compare interrupt.
- c. Enable interrupts.

Figure 6-5. Example of K84SCN Execution

3. RAM Allocation

Label	RAM	Description
OLDKEY	<div style="display: flex; justify-content: space-between;"> <span>b7</span> <span>b0</span> </div>	Previous key data
NEWKEY		Current key data
STBDAT		Data for output strobe signal
KEYNUM		Key number
TOTLKY		Total number of depressed keys
KEYONF		Flag indicating whether or not key data is valid
KEYDAT		Key data
CHATFL		bits 0~3: Counter indicating number of key scan bit 7: Flag indicating key data is valid



Program Module Name: KEY SCAN

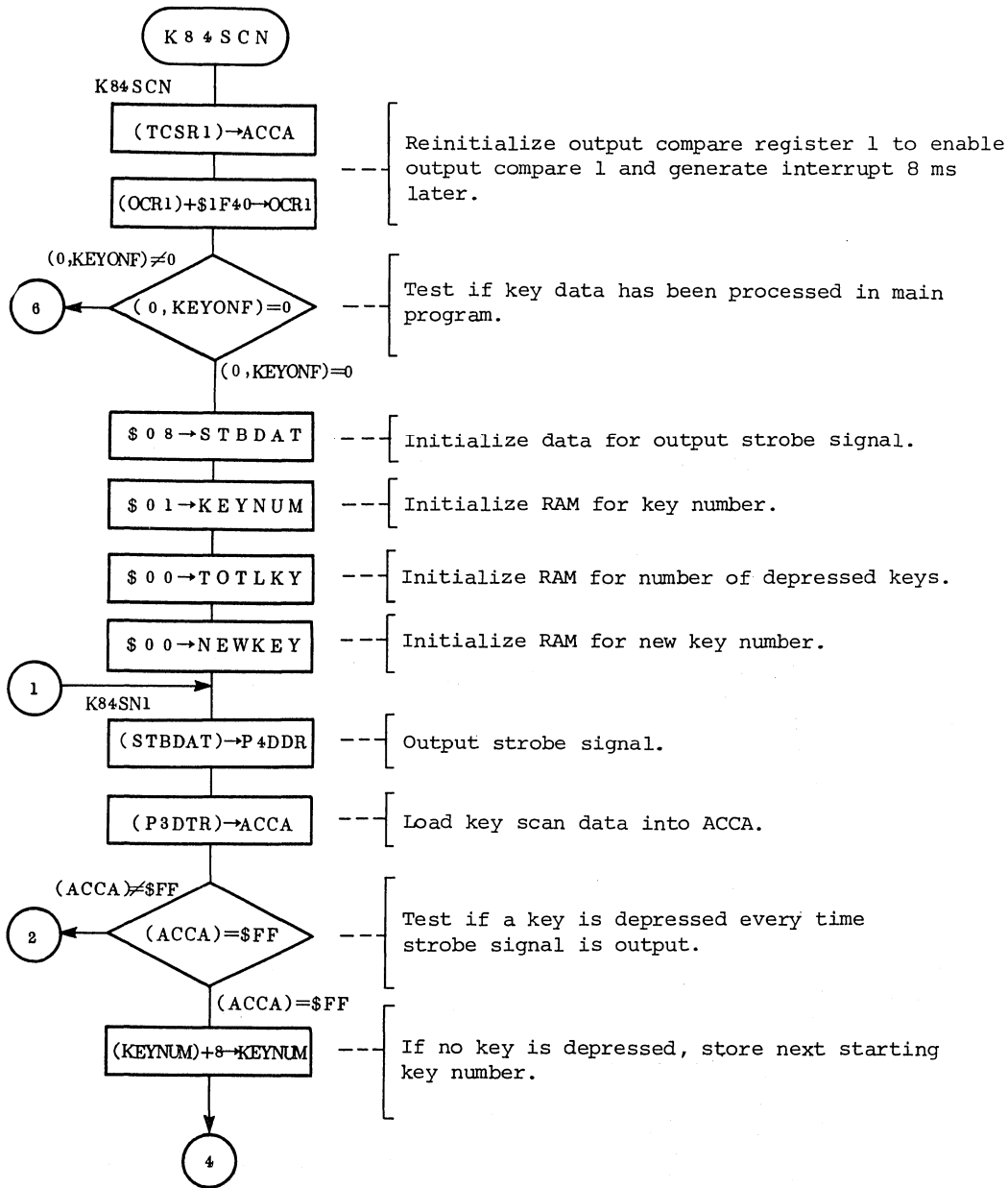
MCU/MPU: HD6301Y0

Label: K84SCN

Description:

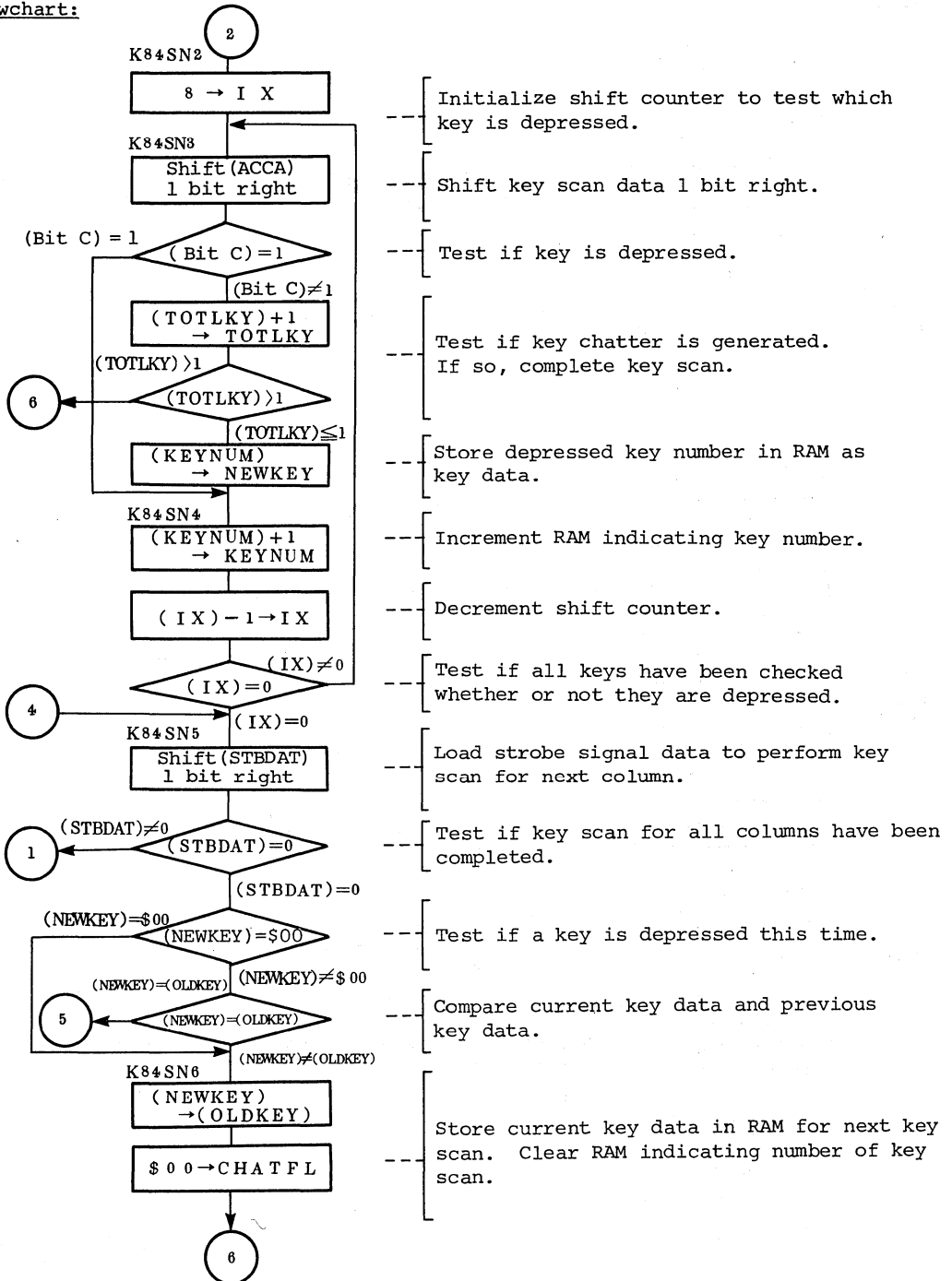
- d. Key data (NEWKEY(RAM)) obtained in (C) is compared with previous key data (OLDKEY(RAM)). If they are the same, chatter counter (CHATEL(RAM)) is counted up. When chatter counter becomes "3", key data is valid. If key data is valid, MSB of CHATFL(RAM) is set to "1" to indicate key data is valid. CHATFL(RAM) includes both a counter and a flag. CHATFL(RAM) is cleared, when NEWKEY(RAM) data differs from OLDKEY(RAM) data or no key is depressed.

Flowchart:

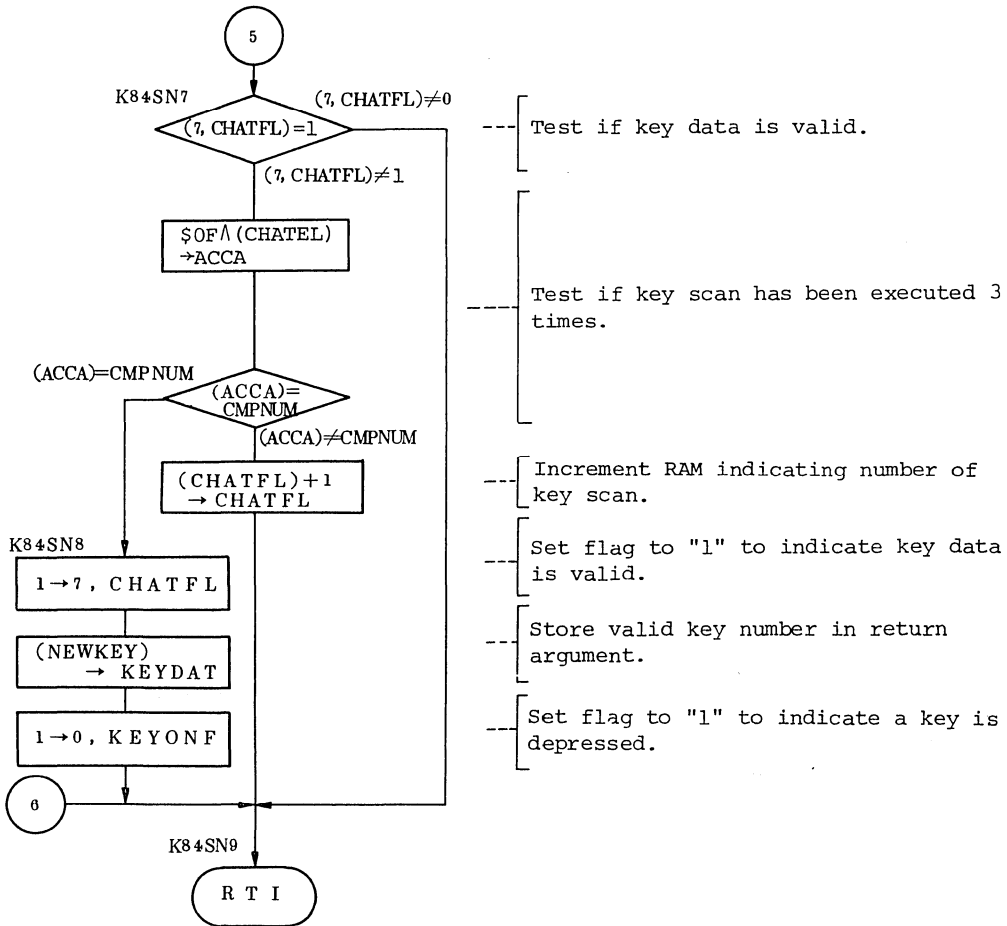




Flowchart:



Flowchart:



## 6.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

## 6.5 PROGRAM LISTING

```

00001
00002      ****  RAM ALLOCATION  ****
00003      *
00004A 0040      ORG    $40
00005      *
00006A 0040      0001  A  KEYSET RMB    1      ASCII
00007A 0041      0001  A  OLDKEY RMB    1      Previous key data
00008A 0042      0001  A  NEWKEY RMB    1      Current key data
00009A 0043      0001  A  CHATFL RMB    1      Chatter counter and flag
00010A 0044      0001  A  KEYONF RMB    1      Key data valid flag
00011A 0045      0001  A  KEYDAT RMB    1      Key data
00012A 0046      0001  A  TOTLKY RMB    1      Total no. of depressed keys
00013A 0047      0001  A  KEYNUM RMB    1      Key number
00014A 0048      0001  A  STBDAT RMB    1      Data for strobe signal
00015      *
00016      ****  SYMBOL DEFINITIONS  ****
00017      *
00018      0004  A  P3DDR  EQU    $04      Port 3 data direction register
00019      0006  A  P3DTR  EQU    $06      Port 3 data register
00020      0008  A  TCSR1  EQU    $08      Timer control/status register1
00021      000B  A  OCR1   EQU    $0B      Output compare register 1
00022      0005  A  P4DDR  EQU    $05      Port 4 data direction register
00023      0007  A  P4DTR  EQU    $07      Port 4 data register
00024      0003  A  CMPNUM EQU    $03      Chatter number
00025      ****
00026      *
00027      *          MAIN PROGRAM : K84MN          *
00028      *
00029      ****
00030      *
00031A C000      ORG    $C000
00032      *
00033A C000 8E 013F A  K84MN  LDS    #13F    Initialize stack pointer
00034A C003 7F 0041 A          CLR    OLDKEY  Initialize RAM
00035A C006 71 FE 44      BCLR   0,KEYONF  Clear key data valid flag
00036A C009 4F          CLRA
00037A C00A 97 07      A          STAA   P4DTR    Initialize port 4
00038A C00C 86 08      A          LDAA   #08      Initialize TCSR1
00039A C00E 97 08      A          STAA   TCSR1
00040A C010 0E          CLI          Enable interrupts
00041A C011 7B 01 44      K84MN1 BTST   0,KEYONF  Test if key is pressed?
00042A C014 27 FB C011      BEQ    K84MN1
00043A C016 D6 45      A          LDAB   KEYDAT    Load key data
00044A C018 71 FE 44      BCLR   0,KEYONF  Clear key data valid flag
00045A C01B CE C09C A          LOX    #KEYCD-1  Load data table starting address
00046A C01E 3A          ABX          Add data table address to key data
00047A C01F A6 00      A          LDAA   0,X      Store ASCII in RAM
00048A C021 97 40      A          STAA   KEYSET
00049A C023 20 FE C023 PEND   BRA    PEND    End of program
00050      ****
00051      *
00052      *          NAME : K84SCN (KEY SCAN)          *
00053      *
00054      ****
00055      *
00056      *          ENTRY : NOTHING          *
00057      *          RETURNS : KEYDAT (KEY DATA)          *

```

```

0005B          *          KEYONF (KEY DATA VALID/INVALID *
00059          *          FLAG)          *
00060          *          *****
00061A C025 96 08 A KB4SCN LDAA TCSR1 Clear interrupt request flag
00062A C027 DC 0B A LDD OCR1 Initialize OCR1
00063A C029 C3 1F40 A ADDD #$1F40
00064A C02C DD 0B A STD OCR1
00065A C02E 7B 01 44 BTST 0,KEYONF Test if key data processed in MAIN
00066A C031 26 69 C09C BNE KB4SN9
00067A C033 86 0B A LDAA #$08 Initialize strobe signal data
00068A C035 97 48 A STAA STBDAT
00069A C037 86 01 A LDAA #$01 Initialize key number
00070A C039 97 47 A STAA KEYNUM
00071A C03B 7F 0046 A CLR TOTLKY Initialize total key number
00072A C03E 7F 0042 A CLR NEWKEY Initialize current key number
00073A C041 96 48 A KB4SN1 LDAA STBDAT Output strobe signal
00074A C043 97 05 A STAA P4DDR
00075A C045 96 06 A LDAA P3DTR Load key data
00076A C047 81 FF A CMPA #$FF Test if key is pressed
00077A C049 26 08 C053 BNE KB4SN2 Branch if pressed
00078A C04B C6 08 A LDAB #B Store next key number
00079A C04D DB 47 A ADDB KEYNUM
00080A C04F D7 47 A STAB KEYNUM
00081A C051 20 19 C06C BRA KB4SN5
00082A C053 CE 0008 A KB4SN2 LDX #B Initialize shift counter
00083A C056 44 KB4SN3 LSRASR Test if key is pressed
00084A C057 25 0D C066 BCS KB4SN4 Branch if not pressed
00085A C059 7C 0046 A INC TOTLKY Increment total key number
00086A C05C D6 46 A LDAB TOTLKY Check key chatter generation
00087A C05E C1 01 A CMPB #1
00088A C060 25 3A C09C BCS KB4SN9 Branch if key chatter generated
00089A C062 D6 47 A LDAB KEYNUM Store key data in current key
00090A C064 D7 42 A STAB NEWKEY
00091A C066 7C 0047 A KB4SN4 INC KEYNUM Increment key number
00092A C069 09 DEX Decrement shift counter
00093A C06A 26 EA C056 BNE KB4SN3 Test if 8 bits shifted
00094A C06C 74 0048 A KB4SN5 LSR STBDAT Output next strobe signal
00095A C06F 26 D0 C041 BNE KB4SN1 Test if all scan is completed
00096A C071 96 42 A LDAA NEWKEY Test if new key is pressed
00097A C073 27 04 C079 BEQ KB4SN6
00098A C075 91 41 A CMPA OLDKEY Current key = previous key?
00099A C077 27 07 C080 BEQ KB4SN7 Branch if equal
00100A C079 97 41 A KB4SN6 STAA OLDKEY Store current key in previous key
00101A C07B 7F 0043 A CLR CHATFL Clear chatter counter
00102A C07E 20 1C C09C BRA KB4SN9
00103A C080 7B 80 43 KB4SN7 BTST 7,CHATFL Test if key data is valid
00104A C083 26 17 C09C BNE KB4SN9
00105A C085 86 0F A LDAA #$0F Test if key scan executed 3 times
00106A C087 94 43 A ANDA CHATFL
00107A C089 81 03 A CMPA #CMPNUM
00108A C08B 27 05 C092 BEQ KB4SN8 Branch if chatter number=3
00109A C08D 7C 0043 A INC CHATFL Increment chatter counter
00110A C090 20 0A C09C BRA KB4SN9
00111A C092 72 80 43 KB4SN8 BSET 7,CHATFL Set chatter flag
00112A C095 96 42 A LDAA NEWKEY Store valid key data in RAM
00113A C097 97 45 A STAA KEYDAT
00114A C099 72 01 44 BSET 0,KEYONF Set key data valid flag

```

```

00115A C09C 3B      K84SN9 RTI
00116                *****
00117                *
00118                *
00119                *
00120                *****
00121A C09D   41    A KEYCD FCC  "ABCDEFGH"
00122A C0A5   49    A      FCC  "IJKLMNOP"
00123A C0AD   51    A      FCC  "QRSTUVWXYZ"
00124A C0B5   59    A      FCC  "YZ123456"
00125                *****
00126                *
00127                *
00128                *
00129                *
00130                *****
00131A FFEA                ORG   $FFEA
00132                *
00133A FFEA   C000  A      FDB   K84MN   IRQ2
00134A FFEC   C000  A      FDB   K84MN   CMI
00135A FFEE   C000  A      FDB   K84MN   TRAP
00136A FFF0   C000  A      FDB   K84MN   SCI
00137A FFF2   C000  A      FDB   K84MN   TOI
00138A FFF4   C025  A      FDB   K84SCN  OCI
00139A FFF6   C000  A      FDB   K84MN   ICI
00140A FFF8   C000  A      FDB   K84MN   IRQ1/ISF
00141A FFFA   C000  A      FDB   K84MN   SWI
00142A FFFC   C000  A      FDB   K84MN   NMI
00143A FFFE   C000  A      FDB   K84MN   RES
00144                *
00145                END

```



### 7.1.5 Pin Functions

Pin functions at the interface between the HD6301Y0 and the HA16613A are shown in table 7-1.

Table 7-1. Pin Functions

Pin Name (HD6301Y0)	Input/Output	Active Level (High or Low)	Function	Pin Name (HA16613A)	Program Label
P <sub>50</sub> / $\overline{\text{IRQ}}_1$	Input	Low	A/D conversion end signal	ADE	P5DTR
P <sub>51</sub>	Output	Low	A/D conversion start signal	ADS	
P <sub>52</sub>	Output	Low	Selects analog input IN <sub>1</sub>	CHS	
		High	Selects analog input IN <sub>2</sub>		
P <sub>30</sub>	Input	—	A/D conversion data	2 <sup>0</sup>	P3DTR
P <sub>31</sub>	Input	—		2 <sup>1</sup>	
P <sub>32</sub>	Input	—		2 <sup>2</sup>	
P <sub>33</sub>	Input	—		2 <sup>3</sup>	
P <sub>34</sub>	Input	—		2 <sup>4</sup>	
P <sub>35</sub>	Input	—		2 <sup>5</sup>	
P <sub>36</sub>	Input	—		2 <sup>6</sup>	
P <sub>37</sub>	Input	—		2 <sup>7</sup>	

### 7.1.6 Hardware Operation

The timing chart between the HD6301Y0 and the HA16613A is shown in figure 7-2.

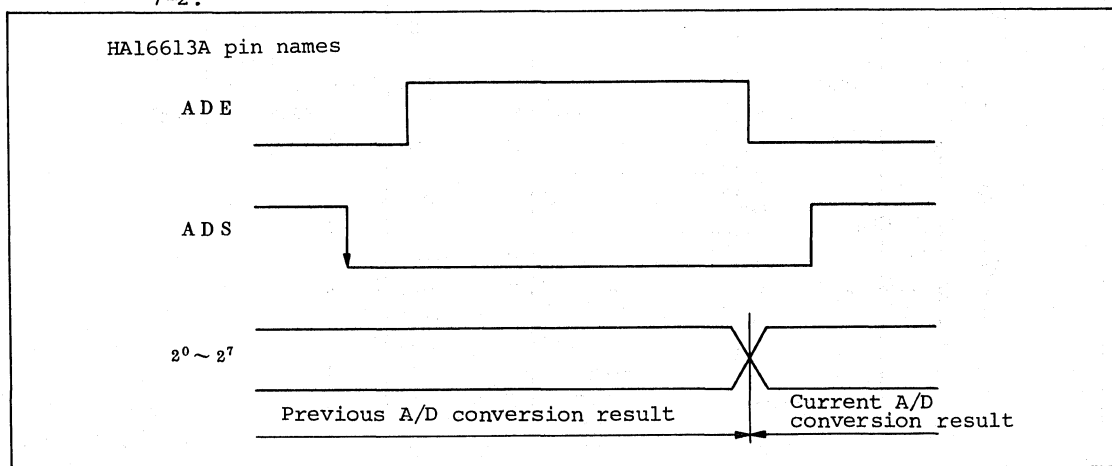


Figure 7-2. HD6301Y0 ↔ HA16613A Timing Chart



7.2.1 Program Module Configuration

The program module configuration for A/D conversion using the HA16613A is shown in figure 7-3.

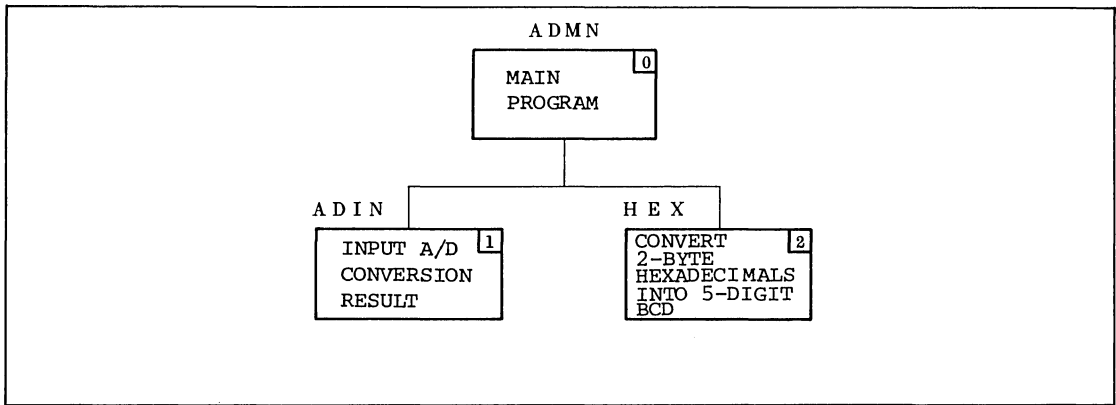


Figure 7-3. Program Module Configuration

7.2.2 Program Module Functions

Program module functions are summarized in table 7-2.

Table 7-2. Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	ADMN	Stores A/D conversion result as a BCD number.
1	INPUT A/D CONVERSION RESULT	ADIN	Inputs A/D conversion result.
2	CONVERT 2-BYTE HEXADECIMALS INTO 5-DIGIT BCD	HEX	Converts 2-byte hexadecimal number into 5-digit BCD. Refer to HEX in HD6301/HD6303 FAMILY APPLICATION NOTES (SOFTWARE) for details.



7.2.3 Program Module Process Flow (Main Program)

The flowchart in figure 7-4 shows the procedure for performing A/D conversion, using the program module in figure 7-3.

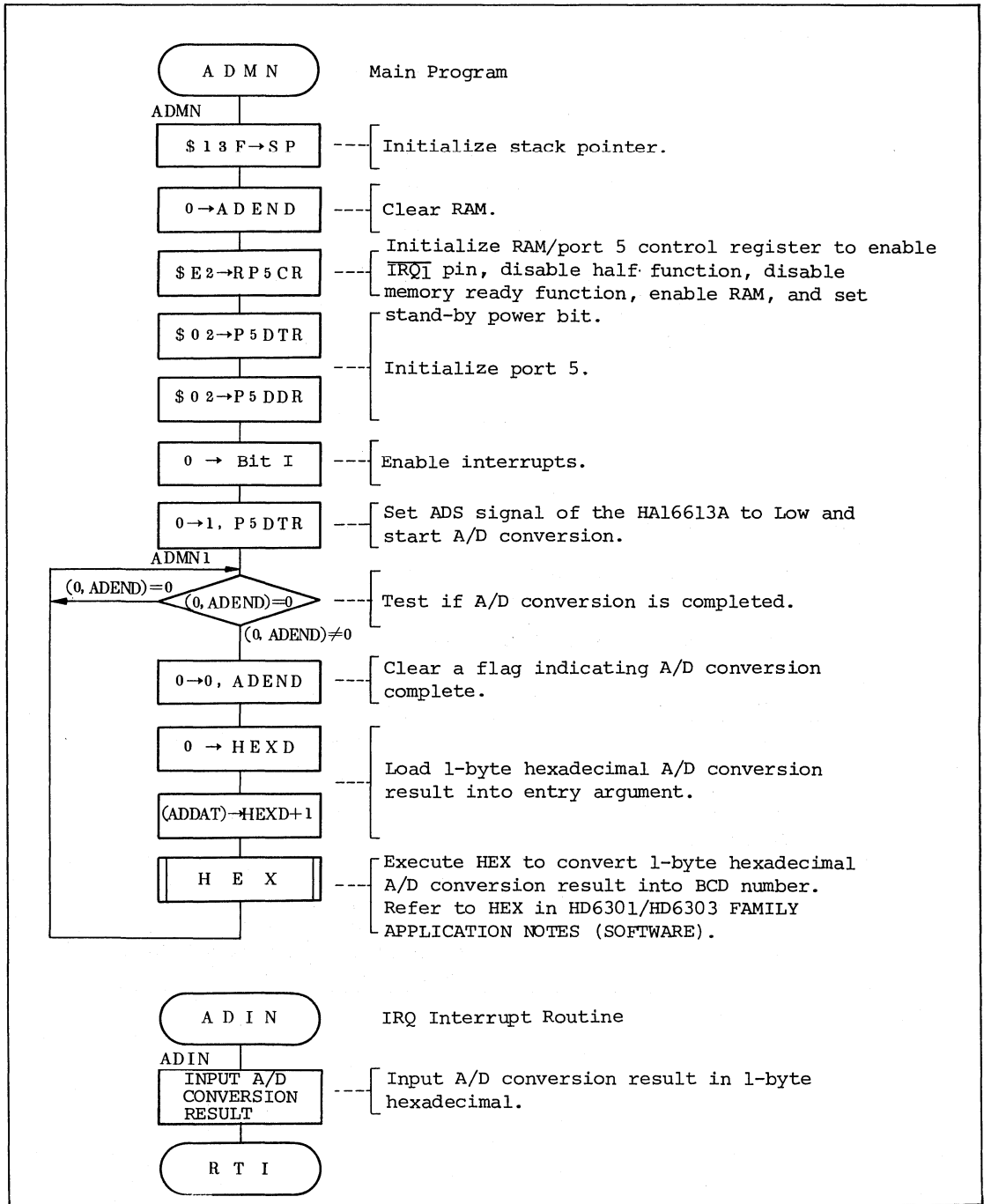


Figure 7-4. Program Module Flowchart



### 7.3 PROGRAM MODULE DESCRIPTION

Program Module Name: INPUT A/D  
CONVERSION  
RESULT

MCU/MPU: HD6301Y0

Label: ADIN

Function:

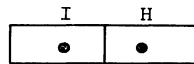
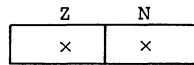
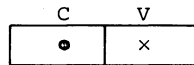
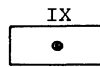
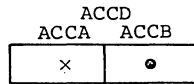
Stores A/D conversion result in ADDAT (RAM).

Arguments:

Contents	Storage Location	No. of Bytes
Entry	---	---
Re- turns	A/D con- version result	ADDAT (RAM) 1
	A/D con- version complete flag	ADEND (RAM) 1

Changes in CPU

Registers and Flags:



● : Not affected  
× : Undefined  
↓ : Result

Specifications:

ROM (Bytes): 14  
RAM (Bytes): 2  
Stack (Bytes): 0  
No. of cycles: 34  
Reentrant: No  
Relocatable: No  
Interrupt OK?: Yes

Description:

1. Function Details

a. Argument details

ADDAT (RAM): Contains A/D conversion result in 1 byte hexadecimal.  
ADEND (RAM): Contains a flag indicating whether or not A/D conversion is completed.

Table 7-3. Flag Functions

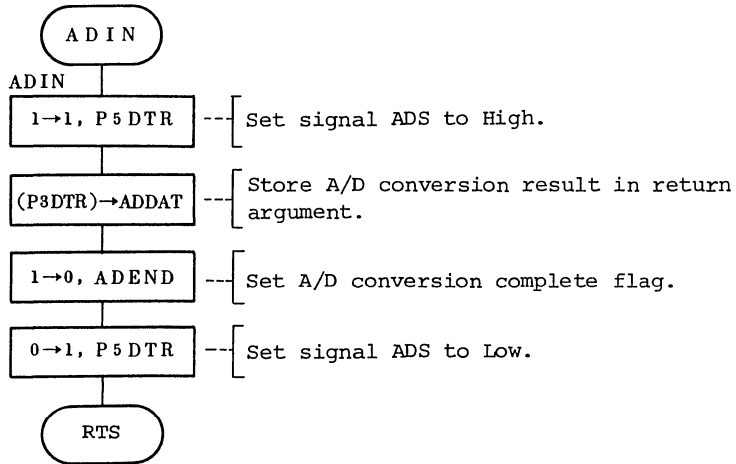
Label	bit 0	Functions
ADEND	0	Indicates A/D conversion is not completed.
	1	Indicates A/D conversion is completed.

Specifications Notes:

N/A



Flowchart:



## 7.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

## 7.5 PROGRAM LISTING

```

00001          *
00002          ***** RAM ALLOCATION *****
00003          *
00004A 0040          ORG      $40
00005          *
00006A 0040 0001 A ADEND RMB 1      A/D conversion complete flag
00007A 0041 0001 A ADDAT RMB 1      A/D conversion result
00008A 0042 0002 A HEXD  RMB 2      A/D conversion result in HEX data
00009A 0044 0003 A DECD  RMB 3      A/D conversion result in BCD data
00010          *
00011          ***** SYMBOL DEFINITIONS *****
00012          *
00013          0006 A P3DTR EQU  $06    Port 3 data register
00014          0014 A RPSCR EQU  $14    RAM/Port 5 control register
00015          0015 A P5DTR EQU  $15    Port 5 data register
00016          0020 A P5DDR EQU  $20    Port 5 data direction register
00017          *****
00018          *
00019          *                MAIN PROGRAM : ADMN                *
00020          *
00021          *****
00022          *
00023A C000          ORG      $C000
00024          *
00025A C000 8E 013F A ADMN  LDS  #$13F  Initialize stack pointer
00026A C003 4F          CLRA          Clear RAM
00027A C004 97 40      A          STAA  ADEND
00028A C006 86 E2      A          LDAA  #$E2  Initialize RAM/Port 5 control register
00029A C008 97 14      A          STAA  RPSCR
00030A C00A 86 02      A          LDAA  #$02  Initialize port 5
00031A C00C 97 15      A          STAA  P5DTR
00032A C00E 97 20      A          STAA  P5DDR
00033A C010 0E          CLI          Enable interrupts
00034A C011 71 FD 15    ADMN1 BCLR  1,P5DIR Set ADS to Low
00035A C014 7B 01 40    ADMN1 BTST  0,ADEND Test if A/D conversion complete
00036A C017 27 FB C014 ADMN1 BEQ  ADMN1
00037A C019 71 FE 40    ADMN1 BCLR  0,ADEND Clear complete flag
00038A C01C 4F          CLRA          Load A/D conversion result in HEX data
00039A C01D 97 42      A          STAA  HEXD
00040A C01F 96 41      A          LDAA  ADDAT
00041A C021 97 43      A          STAA  HEXD+1
00042A C023 BD C036 A    ADMN1 JSR  HEX Convert HEX data into BCD data
00043A C026 20 EC C014 ADMN1 BRA  ADMN1
00044          *****
00045          *
00046          *                NAME : ADIN (INPUT A/D CONVERSION *
00047          *                RESULT)                *
00048          *
00049          *****
00050          *
00051          *                ENTRY : NOTHING                *
00052          *                RETURNS : ADDAT (A/D CONVERSION RESULT) *
00053          *                ADEND (A/D CONVERSION COMPLETE *
00054          *                FLAG)                *
00055          *
00056          *****
00057A C028 72 02 15    ADMN  BSET  1,P5DIR Set ADS to High

```

```

00058A C02B 96 06 A LDA P3DTR Load A/D conversion result
00059A C02D 97 41 A STAA ADDAT
00060A C02F 72 01 40 BSET 0,ADEND Set A/D conversion complete flag
00061A C032 71 FE 15 BCLR 0,P5DTR Set ADS to Low
00062A C035 3B RTI
00063 *****
00064 *
00065 * NAME : HEX (CONVERT 2-BYTE HEXADECIMALS
00066 * INTO 5-DIGIT BCD) *
00067 *
00068 *****
00069 *
00070 * ENTRY : HEXD (2-BYTE HEXADECIMAL NUMBER) *
00071 * RETURNS : DECD (5-DIGIT BCD NUMBER) *
00072 *
00073 *****
00074A C036 4F HEX CLRA Clear ACCA
00075A C037 5F CLR RB Clear ACCB
00076A C038 DD 44 A STD DECD Clear 5-digit BCD
00077A C03A 97 46 A STAA DECD+2
00078A C03C C6 10 A LDAB #16 Store shift counter
00079A C03E 78 0043 A HEX2 ASL HEXD+1 Shift MSB of HEXD to carry
00080A C041 79 0042 A ROL HEXD
00081A C044 CE 0003 A LDX #3 Set ADDR pointer (addition counter)
00082A C047 A6 43 A HEX1 LDAA DECD-1,X DECD * 2 + C -> ACCA
00083A C049 A9 43 A ADCA DECD-1,X
00084A C04B 19 DAA Convert into BCD data
00085A C04C A7 43 A STAA DECD-1,X Store 5-digit BCD area
00086A C04E 09 DEX Decrement ADDR pointer
00087A C04F 26 F6 C047 BNE HEX1 Loop until ADDR pointer=0
00088A C051 5A DECB Decrement shift counter
00089A C052 26 EA C03E BNE HEX2 Loop until shift counter=0
00090A C054 39 RTS
00091 *****
00092 *
00093 * VECTOR ADDRESSES *
00094 *
00095 *****
00096 *
00097A FFEA ORG $FFEA
00098 *
00099A FFEA C000 A FDB ADMN IRQ2
00100A FFEC C000 A FDB ADMN CMI
00101A FFEE C000 A FDB ADMN TRAP
00102A FFF0 C000 A FDB ADMN SID
00103A FFF2 C000 A FDB ADMN TDI
00104A FFF4 C000 A FDB ADMN OCI
00105A FFF6 C000 A FDB ADMN ICI
00106A FFF8 C02B A FDB ADIN IRQ1/ISF
00107A FFFA C000 A FDB ADMN SWI
00108A FFFC C000 A FDB ADMN NMI
00109A FFFE C000 A FDB ADMN RES
00110 *
00111 END

```

## 8.1 HARDWARE DESCRIPTION

## 8.1.1 Function

Receives key data from a standard ASCII keyboard.

## 8.1.2 Microcomputer Operation

The HD6301Y0 accesses data from an ASCII keyboard using a First In-First Out roll buffer. Port 6 control/status register is selected to perform parallel handshaking between the  $\overline{IS}$  pin and port 6. Input data is read at the falling edge of the  $\overline{STROBE}$  signal and data is written to the roll buffer by input strobe interrupt.

## 8.1.3 Peripheral Devices

ASCII keyboard: Outputs ASCII codes and  $\overline{STROBE}$  signal.

## 8.1.4 Circuit Diagram

The interface circuit for reading data from an ASCII keyboard is shown in figure 8-1.

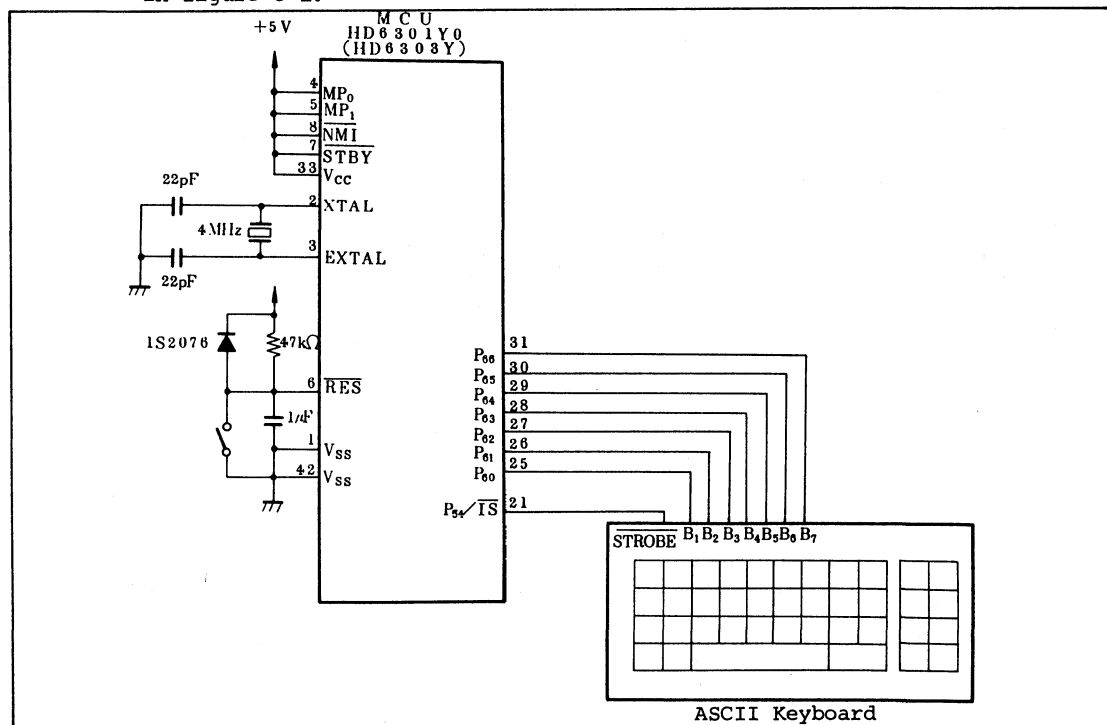


Figure 8-1. Reading Data from ASCII Keyboard

### 8.1.5 Pin Functions

Pin functions at the interface between the HD6301Y0 and ASCII keyboard are shown in table 8-1.

Table 8-1. Pin Functions

Pin Name (HD6301Y0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (Key- board)	Program Label
P <sub>54</sub> / $\overline{IS}$	Input	Low	$\overline{STROBE}$ signal	$\overline{STROBE}$	—
P <sub>60</sub>	Input	—	Key data input signal	B <sub>1</sub>	P6DTR
P <sub>61</sub>	Input	—		B <sub>2</sub>	
P <sub>62</sub>	Input	—		B <sub>3</sub>	
P <sub>63</sub>	Input	—		B <sub>4</sub>	
P <sub>64</sub>	Input	—		B <sub>5</sub>	
P <sub>65</sub>	Input	—		B <sub>6</sub>	
P <sub>66</sub>	Input	—		B <sub>7</sub>	

### 8.1.6 Hardware Operation

The timing chart for the ASCII keyboard is shown in figure 8-2. If a key in ASCII keyboard is depressed, data and  $\overline{STROBE}$  signal are output as shown in figure 8-2.

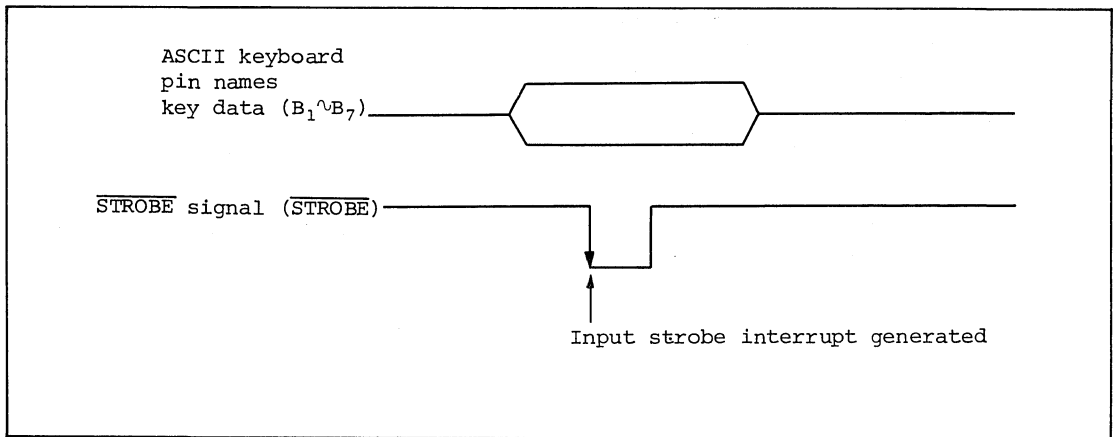


Figure 8-2. ASCII Keyboard Timing Chart



8.2.1 Program Module Configuration

The program module configuration for reading key data from ASCII keyboard is shown in figure 8-3.

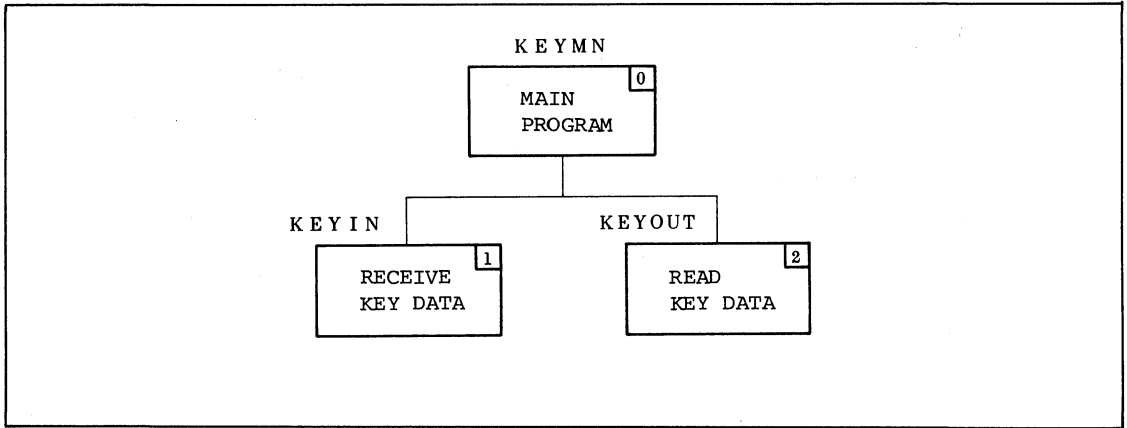


Figure 8-3. Program Module Configuration

8.2.2 Program Module Functions

Program module functions are summarized in table 8-2.

Table 8-2. Program Module Functions

No.	Program Module Name	Label	Functions
0	MAIN PROGRAM	KEYMN	Receives key data from ASCII keyboard and accesses roll buffer.
1	RECEIVE KEY DATA	KEYIN	Receives key data and writes then to roll buffer.
2	READ KEY DATA	KEYOUT	Reads data in roll buffer.

### 8.2.3 Program Module Process Flow (Main Program)

The flowchart in figure 8-4 is an example of key data input from ASCII keyboard performed by the program module in figure 8-3.

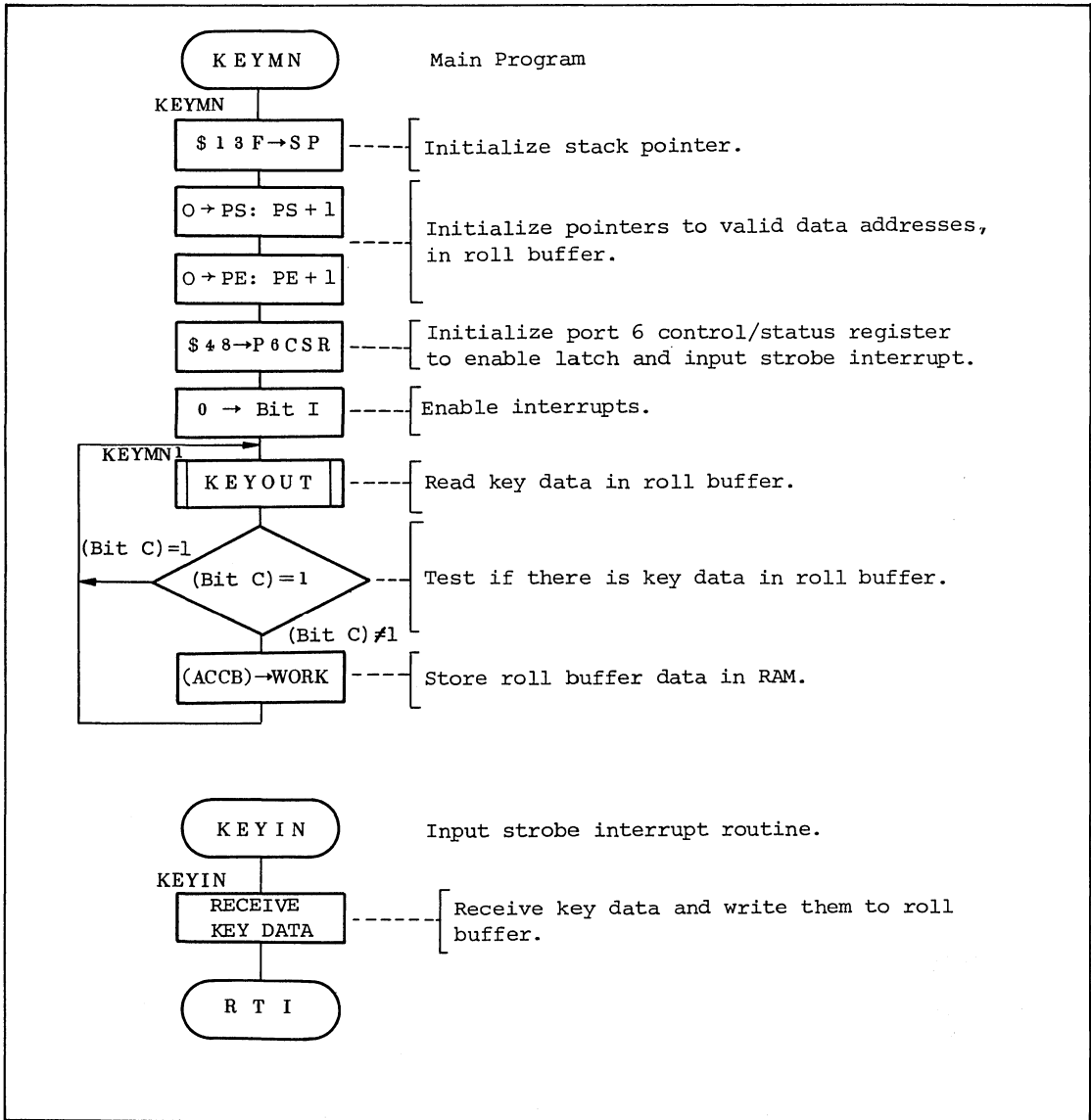


Figure 8-4. Program Module Flowchart

### 8.3 PROGRAM MODULE DESCRIPTION

Program Module Name: RECEIVE KEY  
DATA

MCU/MPU: HD6301Y0/  
HD6303Y

Label: KEYIN

Function:

Receives key data from ASCII key board and writes them to roll buffer.

Arguments: None

Changes in CPU

Registers and Flags:

ACCD	
ACCA	ACCB
x	x

IX
x

C	V
x	x

Z	N
x	x

I	H
●	●

● : Not affected  
x : Undefined  
↓ : Result

Specifications:

ROM (Bytes): 27

RAM (Bytes): 20

Stack (Bytes): 0

No. of cycles: 48

Reentrant: No

Relocatable: No

Interrupt OK?: No

Description:

1. Function Details

- a. KEYIN has no arguments.
- b. Example of KEYIN execution is shown in figure 8-5. If "A" in ASCII keyboard is pressed as shown in part ① of figure 8-5, key data is written to roll buffer as shown in part ② of figure 8-5.
- c. KEYIN calls neither the program modules nor subroutines.

Specifications Notes:

N/A

Description:

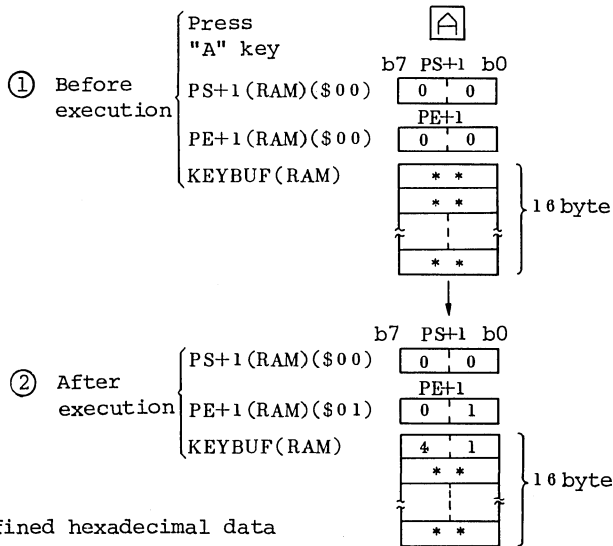


Figure 8-5. Example of KEYIN Execution

2. User Notes

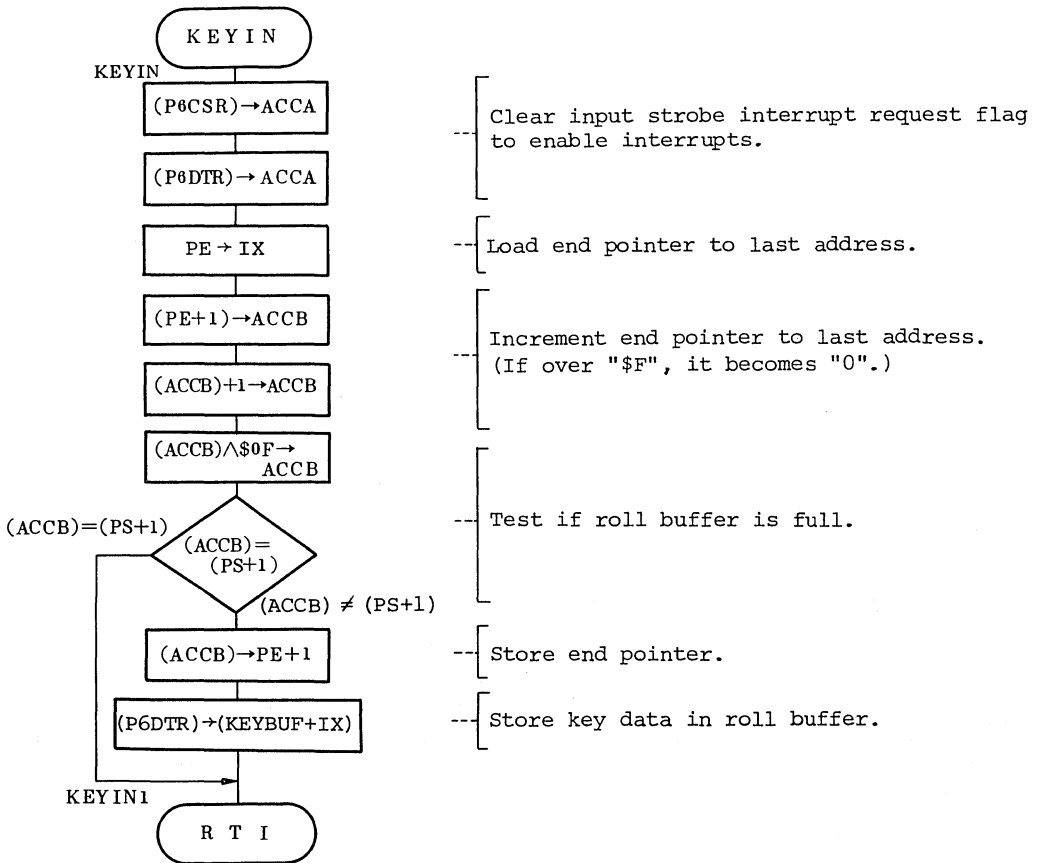
- a. Both KEYIN and KEYOUT must use the same roll buffer.
- b. The following procedure must be performed before KEYIN execution.
  - i. Initialize pointers to valid data addresses.
  - ii. Initialize port 6 control/status register to enable latch and input strobe interrupt.
  - iii. Enable interrupts.
  - iv. Press a key in ASCII keyboard.

3. RAM Allocation

Label	RAM	Description
PS	$\begin{matrix} \text{b7} & & \text{b0} \\ \hline \text{---} & & \text{---} \\ \hline \end{matrix}$	Starting pointer indicating start address of valid data in roll buffer.
PE	$\begin{matrix} \text{---} & & \text{---} \\ \hline \end{matrix}$	
KEYBUF	$\begin{matrix} \text{---} & & \text{---} \\ \hline \end{matrix}$	16 bytes roll buffer to which key data will be written.



Flowchart:



Program Module Name: READ KEY DATA

MCU/MPU: HD6301Y0/  
HD6303Y

Label: KEYOUT

Function:

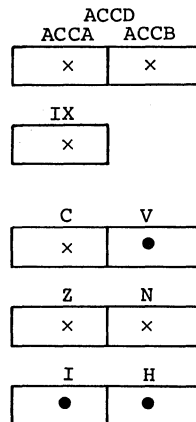
Reads key data from roll buffer.

Arguments:

Contents	Storage Location	No. of Bytes
Entry	_____	_____
Re- turns	Data in roll buffer	1
	Valid data indicator (CCR)	1

Changes in CPU

Registers and Flags:



● : Not Affected  
x : Undefined  
↑ : Result

Specifications:

ROM (Bytes): 20  
RAM (Bytes): 18  
Stack (Bytes): 0  
No. of cycles: 34  
Reentrant: No  
Relocatable: No  
Interrupt OK?: Yes

Description:

1. Function Details

a. Argument details

ACCB: Contains data read from roll buffer.

bit C (CCR): Contains valid data indicator which shows whether or not there are valid data in roll buffer.

bit C=0: Indicates data is read from roll buffer.

Specifications Notes:

N/A

Description:

bit C=1: Indicates there is no data in roll buffer.

- b. Example of KEYOUT execution is shown in figure 8-6. If KEYOUT is executed with the condition shown in part ① of figure 8-6, data is stored in ACCB as shown in part ② of figure 8-6.

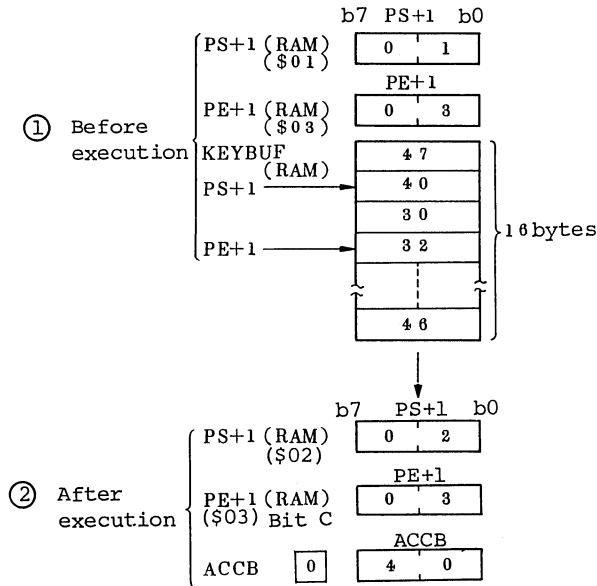


Figure 8-6. Example of KEYOUT Execution

- c. KEYOUT calls neither the program modules nor subroutines.

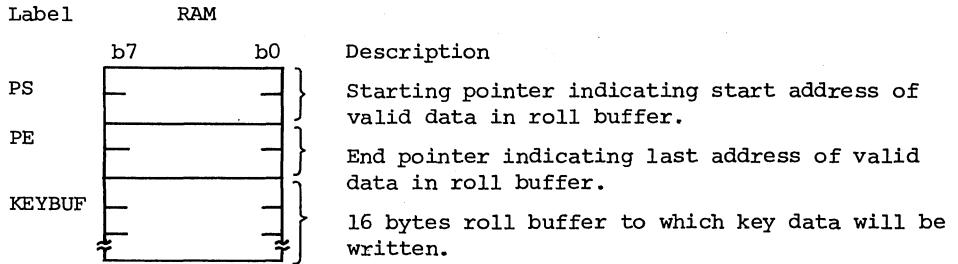
2. User Notes

Both KEYIN and KEYOUT must use the same roll buffer and must be executed in pairs.

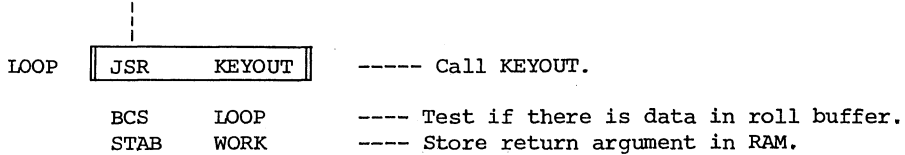


Description:

3. RAM Allocation



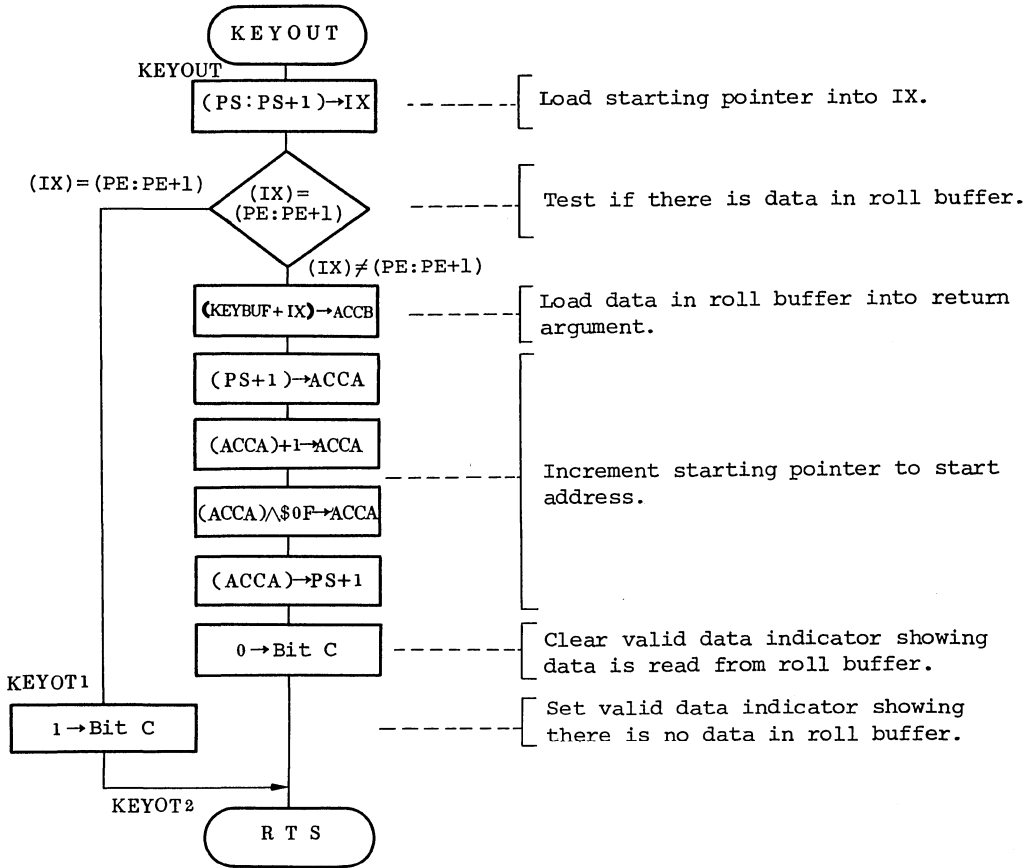
4. Sample Application



5. Basic Operation

- a. Contents of starting pointer PS (RAM), indicating starting data address in roll buffer, is compared with contents of end pointer PE (RAM), indicating last data address in roll buffer.
- b. If equal, bit C is set to "1" since there is no data in roll buffer.
- c. If not equal, data is read from the address indicated by PS+1 (RAM) and PS+1 (RAM) is incremented. Bit C is cleared to indicate data is read from roll buffer.

Flowchart:



## 8.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

## 8.5 PROGRAM LISTING

```
00001      *
00002      ****** RAM ALLOCATION *****
00003      *
00004A 0040      ORG      $40
00005      *
00006A 0040      0002  A PS      RMB      2      Starting pointer
00007A 0042      0002  A PE      RMB      2      End pointer
00008A 0044      0010  A KEYBUF  RMB     16      Key buffer
00009A 0054      0001  A WORK    RMB      1      Work area for data
00010      *
00011      ****** SYMBOL DEFINITIONS *****
00012      *
00013      0016  A P6DDR  EQU     $16      PORT 6 data direction register
00014      0017  A P6DTR  EQU     $17      PORT 6 data register
00015      0021  A P6CSR  EQU     $21      PORT 6 control/status register
00016      *
00017      *
00018      *          MAIN PROGRAM : KEYMN      *
00019      *
00020      ******
00021      *
00022A C000      ORG     $C000
00023      *
00024A C000 8E 013F A KEYMN  LDS     #$13F  Initialize stack pointer
00025A C003 4F          CLRA          Clear pointers
00026A C004 97 40      A          STAA    PS
00027A C006 97 41      A          STAA    PS+1
00028A C008 97 42      A          STAA    PE
00029A C00A 97 43      A          STAA    PE+1
00030A C00C 86 48      A          LDAA   #$48
00031A C00E 97 21      A          STAA   P6CSR  Initialize port6 CSR
00032A C010 0E          CLI           Enable interrupts
00033A C011 BD C01A A KEYMN1 JSR     KEYOUT  Read key data from roll buffer
00034A C014 25 FB C011 BCS     KEYMN1  Check data in roll buffer
00035A C016 D7 54      A          STAB   WORK    Store key data in RAM
00036A C018 20 F7 C011 BRA     KEYMN1
00037      *
00038      *
00039      *          NAME : KEYOUT (READ KEY DATA)      *
00040      *
00041      *
00042      *
00043      *          ENTRY : NOTHING      *
00044      *          RETURNS : ACCB(DATA IN ROLL BUFFER)      *
00045      *          CARRY(C=0:TRUE,C=1:FALES)      *
00046      *
00047      ******
00048A C01A DE 40      A KEYOUT LDX     PS      Load starting pointer
00049A C01C 9C 42      A          CPX     PE      Check data in roll buffer
00050A C01E 27 0C C02C BEQ     KEYOT1  Branch if no data
00051A C020 E6 44      A          LDAB   KEYBUF,X Load key data
00052A C022 96 41      A          LDAA   PS+1
00053A C024 4C          INCA          Increment starting pointer
00054A C025 84 0F      A          ANDA   #$0F
00055A C027 97 41      A          STAA   PS+1
00056A C029 0C          CLC           Clear carry
00057A C02A 20 01 C02D BRA     KEYOT2
```

```

00058A C02C 0D      KEYOT1 SEC          Set carry
00059A C02D 39      KEYOT2 RTS
00060                *****
00061                *
00062                *          NAME: KEYIN (RECEIVE KEY DATA)
00063                *
00064                *****
00065                *
00066                *          ENTRY : NOTHING
00067                *          RETURNS : NOTHING
00068                *
00069                *****
00070A C02E 96 21    A KEYIN LDAA P6CSR   Clear interrupt request flag
00071A C030 96 17    A          LDAA P6DTR
00072A C032 DE 42    A          LDX PE      Load end pointer
00073A C034 D6 43    A          LDAB PE+1
00074A C036 5C          INCB          Increment end pointer
00075A C037 C4 0F    A          ANDB #50F
00076A C039 D1 41    A          CMPB PS+1
00077A C03B 27 04    C041 BEQ KEYIN1   Test if roll buffer is full?
00078A C03D D7 43    A          STAB PE+1   Store end pointer
00079A C03F A7 44    A          STAA KEYBUF,X Store data in roll buffer
00080A C041 3B          KEYIN1 RTI
00081                *****
00082                *
00083                *          VECTOR ADDRESSES
00084                *
00085                *****
00086                *
00087A FFEA          ORG      $FFEA
00088                *
00089A FFEA C000 A      FDB KEYMN   IRQ2
00090A FFEC C000 A      FDB KEYMN   NMI
00091A FFEE C000 A      FDB KEYMN   TRAP
00092A FFFF C000 A      FDB KEYMN   SIQ
00093A FFF2 C000 A      FDB KEYMN   TOI
00094A FFF4 C000 A      FDB KEYMN   OCI
00095A FFF6 C000 A      FDB KEYMN   ICI
00096A FFF8 C02E A      FDB KEYIN   IRQ1/ISF
00097A FFFA C000 A      FDB KEYMN   SWI
00098A FFFC C000 A      FDB KEYMN   CMI
00099A FFFE C000 A      FDB KEYMN   RES
00100
00101                END

```

## 9.1 HARDWARE DESCRIPTION

## 9.1.1 Function

Interfaces the HD6301Y0 with a printer and sends ASCII data, stored in internal RAM, be printed.

## 9.1.2 Microcomputer Operation

The HD6301Y0 defines port 6 control/status register and performs parallel handshaking between the  $\overline{IS}$  pin,  $\overline{OS}$  pin, port 6 and the printer. Port 6 input strobe interrupt routine, is executed at the falling edge of the  $\overline{IS}$  pin to store data in port 6 and output the  $\overline{STROBE}$  signal.

## 9.1.3 Peripheral Devices

Printer: Uses a centronics interface format.

## 9.1.4 Circuit Diagram

The centronics interface circuit is shown in figure 9-1.

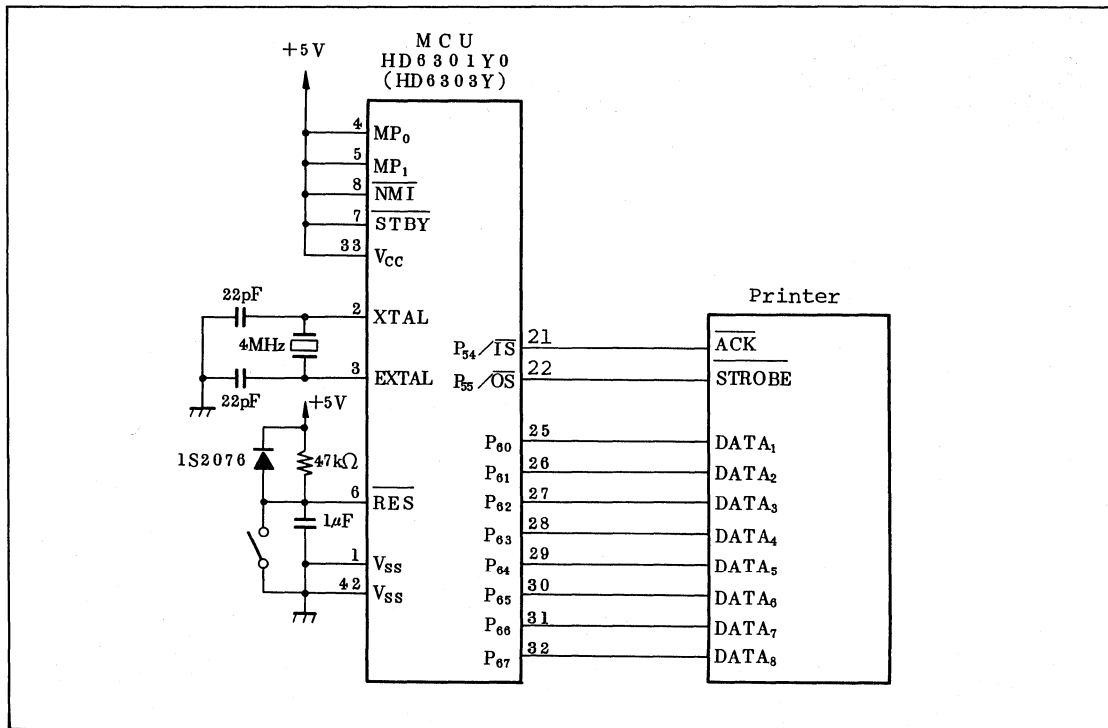


Figure 9-1. Centronics Interface Circuit

### 9.1.5 Pin Functions

Pin functions at the interface between the HD6301Y0 and the printer are shown in table 9-1.

Table 9-1. Pin Functions

Pin Name (HD6301Y0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (Printer)	Program Label
P <sub>54</sub> / $\overline{IS}$	Input	Low	Acknowledge signal	$\overline{ACK}$	—
P <sub>55</sub> / $\overline{OS}$	Output	Low	Strobe signal	$\overline{STROBE}$	—
P <sub>60</sub>	Output	—	Display data	DATA <sub>1</sub>	P6DTR
P <sub>61</sub>	Output	—		DATA <sub>2</sub>	
P <sub>62</sub>	Output	—		DATA <sub>3</sub>	
P <sub>63</sub>	Output	—		DATA <sub>4</sub>	
P <sub>64</sub>	Output	—		DATA <sub>5</sub>	
P <sub>65</sub>	Output	—		DATA <sub>6</sub>	
P <sub>66</sub>	Output	—		DATA <sub>7</sub>	
P <sub>67</sub>	Output	—		DATA <sub>8</sub>	

### 9.1.6 Hardware Operation

$\overline{ACK}$  signal,  $\overline{STROBE}$  signal, and data lines of the printer are controlled by the parallel handshake interface of the HD6301Y0. The timing chart of each signal is shown in figure 9-2.

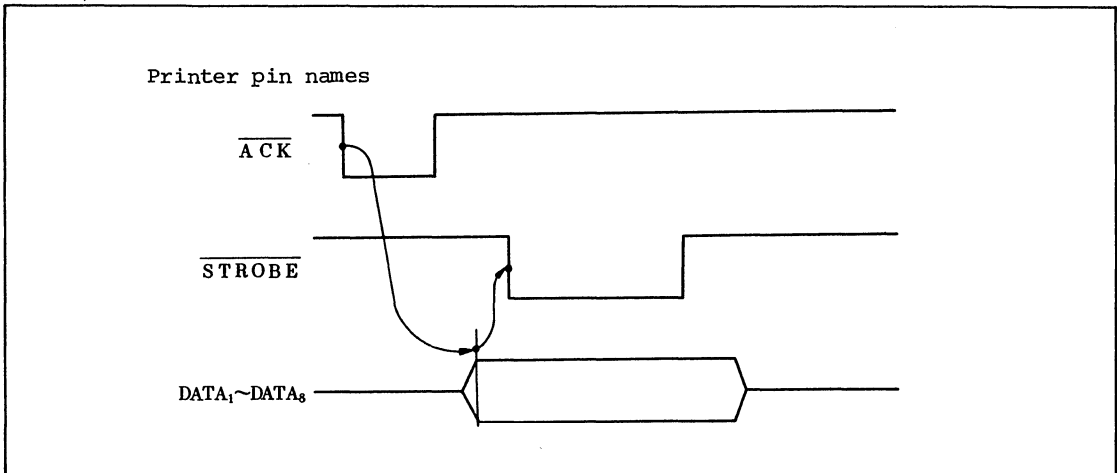


Figure 9-2. Centronics Interface Timing Chart

9.2.1 Program Module Configuration

The program module configuration for printing characters is shown in figure 9-3.

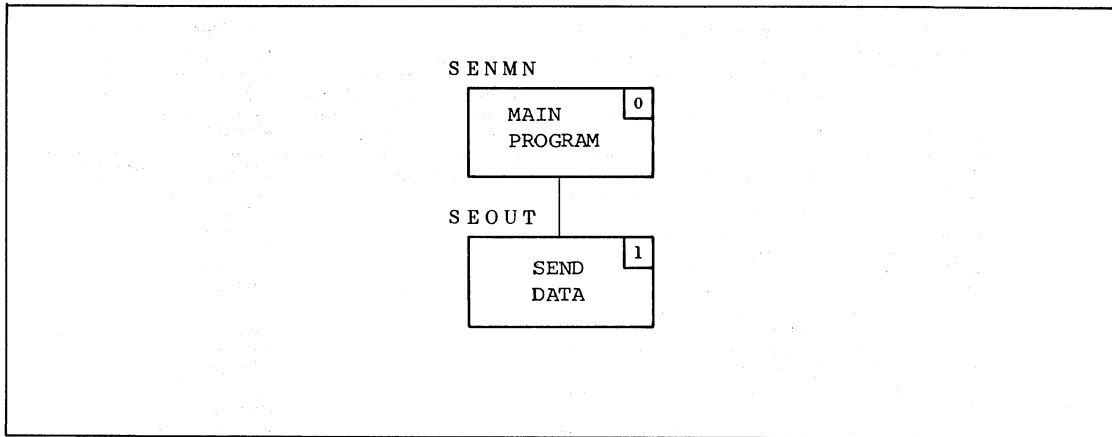


Figure 9-3. Program Module Configuration

9.2.2 Program Module Functions

Program module functions are summarized in table 9-2.

Table 9-2. Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	SENMN	Initializes the interface between the printer and the HD6301Y0.
1	SEND DATA	SEOUT	Sends ASCII character codes to the printer.

### 9.2.3 Program Module Process Flow (Main Program)

The flowchart in figure 9-4 shows the procedure for printing characters as performed by the program module in figure 9-3. An example of printed output is shown in figure 9-5.

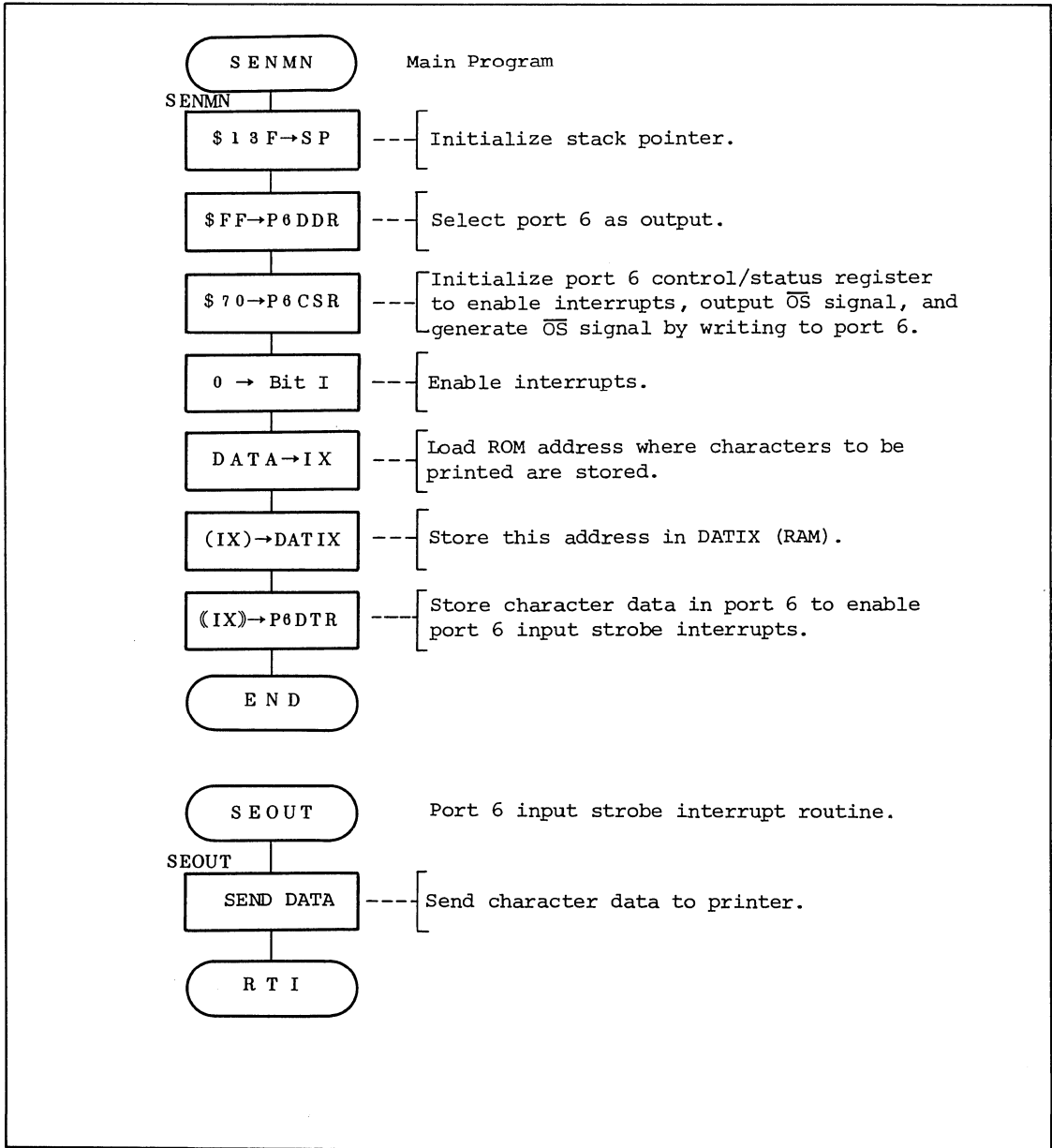


Figure 9-4. Program Module Flowchart



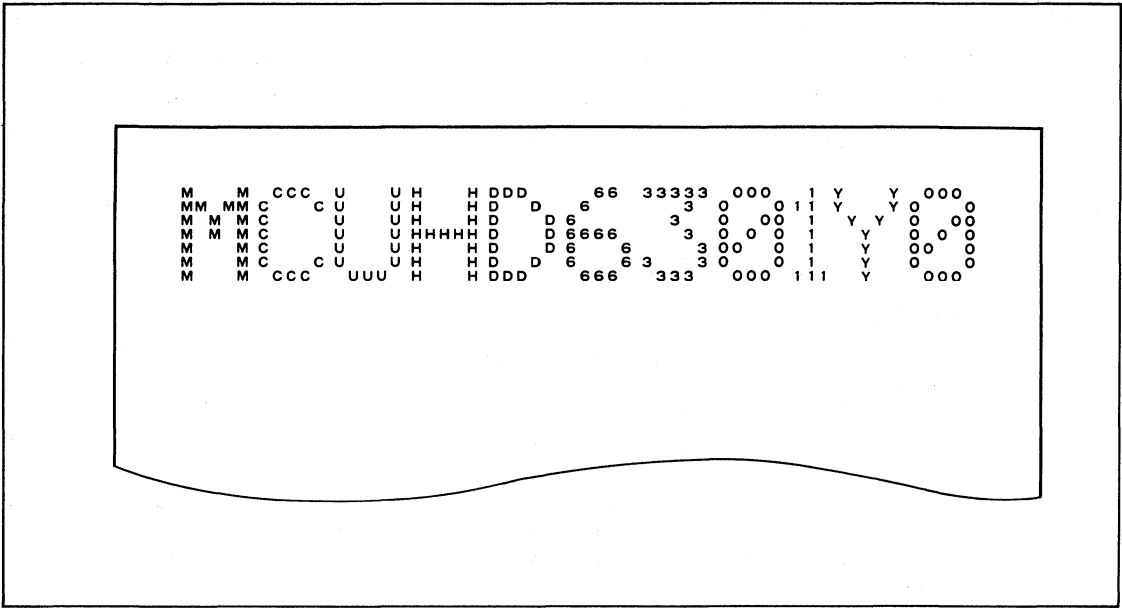


Figure 9-5. Example of Printed Characters

### 9.3 PROGRAM MODULE DESCRIPTION

Program Module Name: SEND DATA

MCU/MPU: HD6301Y0/  
HD6303Y

Label: SEOUT

Function:

1. Sends ASCII codes, stored in data table of memory, to the printer.
2. Finishes sending if \$FF is found in data table.

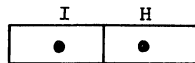
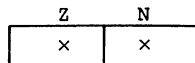
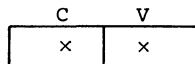
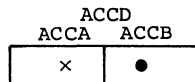
Arguments:

Contents	Storage Location	No. of Bytes
Entry Starting address of data table	DATIX (RAM)	2

Re-returns \_\_\_\_\_

Changes in CPU

Registers and Flags:



- : Not affected
- × : Undefined
- ↑ : Result

Specifications:

ROM (Bytes): 20  
RAM (Bytes): 2  
Stack (Bytes): 0  
No. of cycles: 37  
Reentrant: No  
Relocatable: Yes  
Interrupt OK?: Yes

Description:

1. Function Details

a. Argument details

DATIX(RAM) : Holds the address of the data table that stores characters to be printed in ASCII.

b. Example of SEOUT execution is shown in figure 9-6. If entry arguments are as shown in part ① of figure 9-6, data is printed as shown in part ② of figure 9-6.

c. SEOUT calls neither the program modules nor subroutines.

Specifications Notes:

"No. of cycles" in "Specifications" indicates the number of cycles required to send data.

Description:

2. User Notes

- a. The data table can have a maximum size of 65535 bytes, since index addressing mode is used.
- b. The following procedure must be performed before SEOUT execution.
  - i. Initialize part 6 control/status register for input strobe interrupts.
  - ii. Clear bit I and enable interrupts.
  - iii. Set data to port 6 for input strobe interrupt generation.

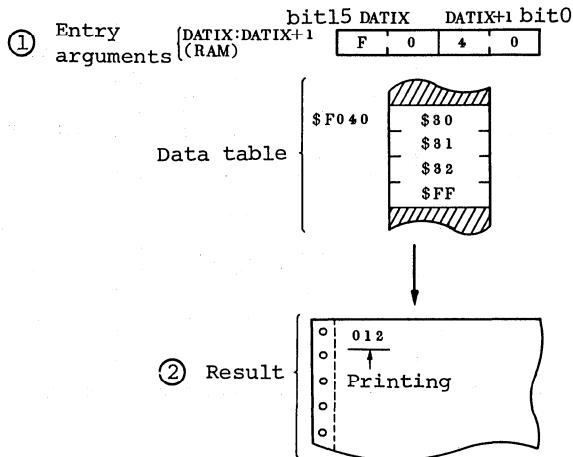


Figure 9-6. Example of SEOUT Execution

Program Module Name: SEND DATA

MCU/MPU: HD6301Y0/  
HD6303Y

Label: SEOUT

Description:

3. RAM Allocation

Label	RAM	Description						
DATIX	<table border="1"><tr><td>b7</td><td>Upper</td><td>b0</td></tr><tr><td></td><td>Lower</td><td></td></tr></table>	b7	Upper	b0		Lower		Used as a pointer to the data table.
b7	Upper	b0						
	Lower							

4. Sample Application

```

:
LDAA  #$FF } ----- Select port 6 as output.
STAA  P6DDR }
LDAA  #$70 } ----- Define parallel handshake interface.
STAA  P6CSR }
CLI   } ----- Enable interrupts.
LDX   #DATA } ----- Load starting address of data table into entry
STX   DATIX } ----- argument
LDAA  0,X } ----- Store data in port 6 for input strobe interrupt
STAA  P6DTR } ----- generation.
:

```

5. Basic Operation

- Dummy-reading of port6 control/status register is performed and interrupt request flag is cleared.
- DATIX(RAM) is loaded into IX and IX is incremented.
- Contents of IX are then restored in DATIX(RAM).
- Character data is loaded into ACCA using index addressing mode. Data loaded is tested for \$FF.
- Contents of ACCA are stored in port6.



## 9.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

## 9.5 PROGRAM LISTING

```

00001          *
00002          ****  RAM ALLOCATION  *****
00003          *
00004A 0040          ORG      $40
00005          *
00006A 0040      0002  A  DATIX  RMB      2      Start address of data table
00007          *
00008          ****  SYMBOL DEFINITIONS  *****
00009          *
00010          0016  A  P6DDR  EQU      $16      Port 6 data direction register
00011          0017  A  P6DTR  EQU      $17      Port 6 data register
00012          0021  A  P6CSR  EQU      $21      Port 6 control/status register
00013          *****
00014          *
00015          *      MAIN PROGRAM : SENMN      *
00016          *
00017          *****
00018          *
00019A C000          ORG      $C000
00020          *
00021A C000 8E 013F  A  SENMN  LDS      #$13F      Initialize stack pointer
00022A C003 86 FF    A          LDAA     H$FF      Select port 6 as output
00023A C005 97 16    A          STAA     P6DDR
00024A C007 86 70    A          LDAA     H$70      Initialize P6CSR
00025A C009 97 21    A          STAA     P6CSR
00026A C00B 0E       A          CLI       Enable interrupts
00027A C00C CE C02B  A          LDX      HDATA     Load data table pointer
00028A C00F DF 40    A          STX      DATIX     Store data table pointer
00029A C011 A6 00    A          LDAA     0.X      Load data
00030A C013 97 17    A          STAA     P6DTR
00031A C015 20 FE C015  PEND    BRA      PEND
00032          *****
00033          *
00034          *      NAME : SEOUT (SEND DATA)      *
00035          *
00036          *****
00037          *
00038          *      ENTRY : NOTHIG      *
00039          *      RETURNS : NOTING      *
00040          *
00041          *****
00042A C017 96 21    A  SEOUT  LDAA     P6CSR      Clear interrupt request flag
00043A C019 DE 40    A          LDX      DATIX     Increment data table pointer
00044A C01B 08       A          INX
00045A C01C DF 40    A          STX      DATIX
00046A C01E A6 00    A          LDAA     0.X      Load data
00047A C020 81 FF    A          CMPA     H$FF      Test if data=$FF
00048A C022 27 04  C02B  BEQ      SEOUT1     Branch if data=$FF
00049A C024 97 17    A          STAA     P6DTR      Store data
00050A C026 20 02  C02A  BRA      SEOUT2     Branch always to SEOUT2
00051A C028 96 17    A  SEOUT1 LDAA     P6DTR      Dummy read Port 6
00052A C02A 3B       SEOUT2 RTI
00053          *****
00054          *
00055          *      DATA TABLE      *
00056          *
00057          *****

```

00058A	C02B	4D	A	DATA	FCC	/M M /	*1 column data
00059A	C031	20	A		FCC	/ CCC /	
00060A	C037	55	A		FCC	/U U /	
00061A	C03D	20	A		FCC	/ / /	
00062A	C040	48	A		FCC	/H H /	
00063A	C046	44	A		FCC	/DDD /	
00064A	C04C	20	A		FCC	/ 66 /	
00065A	C052	33	A		FCC	/33333 /	
00066A	C058	20	A		FCC	/ 000 /	
00067A	C05E	20	A		FCC	/ 1 /	
00068A	C062	59	A		FCC	/Y Y /	
00069A	C068	20	A		FCC	/ 000 /	
00070A	C06E	0D	A		FCB	\$0D.\$0A	
00071A	C070	4D	A		FCC	/MM MM /	*2 column data
00072A	C076	43	A		FCC	/C C /	
00073A	C07C	55	A		FCC	/U U /	
00074A	C082	20	A		FCC	/ / /	
00075A	C085	48	A		FCC	/H H /	
00076A	C08B	44	A		FCC	/D D /	
00077A	C091	20	A		FCC	/ 6 /	
00078A	C097	20	A		FCC	/ 3 /	
00079A	C09D	30	A		FCC	/0 0 /	
00080A	C0A3	31	A		FCC	/11 /	
00081A	C0A7	59	A		FCC	/Y Y /	
00082A	C0AD	30	A		FCC	/0 0 /	
00083A	C0B3	0D	A		FCB	\$0D.\$0A	
00084A	C0B5	4D	A		FCC	/M M M /	*3 column data
00085A	C0BB	43	A		FCC	/C C /	
00086A	C0C1	55	A		FCC	/U U /	
00087A	C0C7	20	A		FCC	/ / /	
00088A	C0CA	48	A		FCC	/H H /	
00089A	C0D0	44	A		FCC	/D D /	
00090A	C0D6	36	A		FCC	/6 /	
00091A	C0DC	20	A		FCC	/ 3 /	
00092A	C0E2	30	A		FCC	/0 00 /	
00093A	C0E8	20	A		FCC	/ 1 /	
00094A	C0EC	20	A		FCC	/ Y Y /	
00095A	C0F2	30	A		FCC	/0 00 /	
00096A	C0F8	0D	A		FCB	\$0D.\$0A	
00097A	C0FA	4D	A		FCC	/M M M /	*4 column data
00098A	C100	43	A		FCC	/C C /	
00099A	C106	55	A		FCC	/U U /	
00100A	C10C	20	A		FCC	/ / /	
00101A	C10F	48	A		FCC	/HHHHH /	
00102A	C115	44	A		FCC	/D D /	
00103A	C11B	36	A		FCC	/6666 /	
00104A	C121	20	A		FCC	/ 3 /	
00105A	C127	30	A		FCC	/0 0 0 /	
00106A	C12D	20	A		FCC	/ 1 /	
00107A	C131	20	A		FCC	/ Y Y /	
00108A	C137	30	A		FCC	/0 0 0 /	
00109A	C13D	0D	A		FCB	\$0D.\$0A	
00110A	C13F	4D	A		FCC	/M M M /	*5 column data
00111A	C145	43	A		FCC	/C C /	
00112A	C148	55	A		FCC	/U U /	
00113A	C151	20	A		FCC	/ / /	
00114A	C154	48	A		FCC	/H H /	

```

00115A C15A      44  A      FCC      /D  D  /
00116A C160      36  A      FCC      /6  6  /
00117A C166      20  A      FCC      /   3  /
00118A C16C      30  A      FCC      /00 0  /
00119A C172      20  A      FCC      /  1  /
00120A C176      20  A      FCC      /  Y  /
00121A C17C      30  A      FCC      /00 0  /
00122A C182      0D  A      FCB      $0D,$0A
00123A C184      4D  A      FCC      /M  M  / *6 column data
00124A C18A      43  A      FCC      /C  C  /
00125A C190      55  A      FCC      /U  U  /
00126A C196      20  A      FCC      /   /
00127A C199      48  A      FCC      /H  H  /
00128A C19F      44  A      FCC      /D  D  /
00129A C1A5      36  A      FCC      /6  6  /
00130A C1AB      33  A      FCC      /3  3  /
00131A C1B1      30  A      FCC      /0  0  /
00132A C1B7      20  A      FCC      /  1  /
00133A C1BB      20  A      FCC      /  Y  /
00134A C1C1      30  A      FCC      /0  0  /
00135A C1C7      0D  A      FCB      $0D,$0A
00136A C1C9      4D  A      FCC      /M  M  / *7 column data
00137A C1CF      20  A      FCC      / CCC /
00138A C1D5      20  A      FCC      / UUU /
00139A C1DB      20  A      FCC      /   /
00140A C1DE      48  A      FCC      /H  H  /
00141A C1E4      44  A      FCC      /DDD /
00142A C1EA      20  A      FCC      / 666 /
00143A C1F0      20  A      FCC      / 333 /
00144A C1F6      20  A      FCC      / 000 /
00145A C1FC      31  A      FCC      /111 /
00146A C200      20  A      FCC      /  Y  /
00147A C206      20  A      FCC      / 000 /
00148A C20C      0D  A      FCB      $0D,$0A,$FF
00149
00150 *****
00151 *
00152 *
00153 *
00154 *
00155A FFEA      *      ORG      $FFEA
00156
00157A FFEA      C000 A      FDB      SENMN      IRQ2
00158A FFEC      C000 A      FDB      SENMN      CMI
00159A FFEE      C000 A      FDB      SENMN      TRAP
00160A FFF0      C000 A      FDB      SENMN      SID
00161A FFF2      C000 A      FDB      SENMN      TOI
00162A FFF4      C000 A      FDB      SENMN      OCI
00163A FFF6      C000 A      FDB      SENMN      ICI
00164A FFFB      C017 A      FDB      SEOUT      IRQ1/ISF
00165A FFFA      C000 A      FDB      SENMN      SWI
00166A FFFC      C000 A      FDB      SENMN      NMI
00167A FFFE      C000 A      FDB      SENMN      RES
00168
00169 *
END

```



10.1 HARDWARE DESCRIPTION10.1.1 Function

Receives ASCII from the console typewriter as asynchronous serial data, and sends ASCII to the console typewriter converting lower case letters into uppercase letters.

10.1.2 Microcomputer Operation

The HD6301Y0 sends/receives data to/from the console typewriter by asynchronous SCI (serial communication interface), defining the band rate as 4800 BPS. RS232C level for data transfer should be selected. Transfer format is defined as 1 start bit + 8 bits of data + 1 stop bit by the rate/mode control register. Signals  $\overline{\text{CTS}}$  and  $\overline{\text{RTS}}$  of the console typewriter are controlled through bits 0 and 1 of port 5.

10.1.3 Peripheral Devices

Console Typewriter: Sends/Receives data to/from the microcomputer.

10.1.4 Circuit Diagram

Asynchronous SCI circuit is shown in figure 10-1.

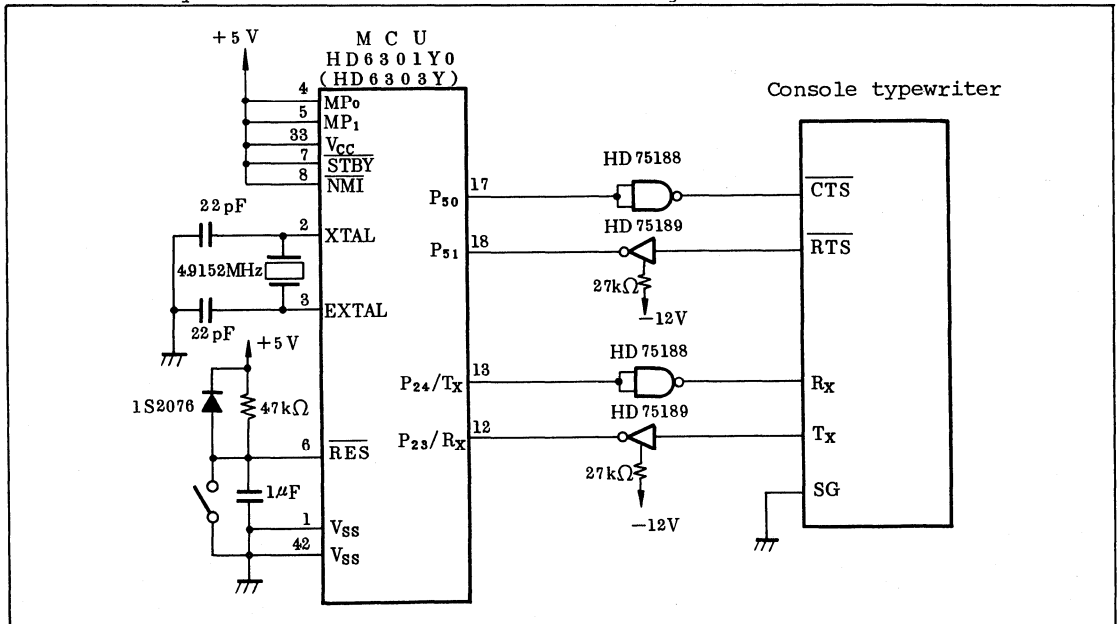


Figure 10-1. Asynchronous SCI Circuit

### 10.1.5 Pin Functions

Pin functions at the interface between the HD6301Y0 and the console typewriter are shown in Table 10-1.

Table 10-1. Pin Functions

Pin Name (HD6301Y0)	Input/Output	Active Level (High or Low)	Function	Pin Name (Console Typewriter)	Program Label
P <sub>50</sub>	Output	Low	Outputs sending data request signal to the console typewriter.	$\overline{\text{CTS}}$	P5DTR
P <sub>51</sub>	Input	Low	Inputs sending data request signal from the console typewriter.	$\overline{\text{RTS}}$	
P <sub>23</sub> /Rx	Input	—	Receives serial data from the console typewriter.	Tx	—
P <sub>24</sub> /Tx	Output	—	Sends serial data back to the console typewriter.	Rx	—

### 10.1.6 Hardware Operation

The timing chart for sending/receiving data and control signals are shown in figure 10-2. This application example generates the timing by software.

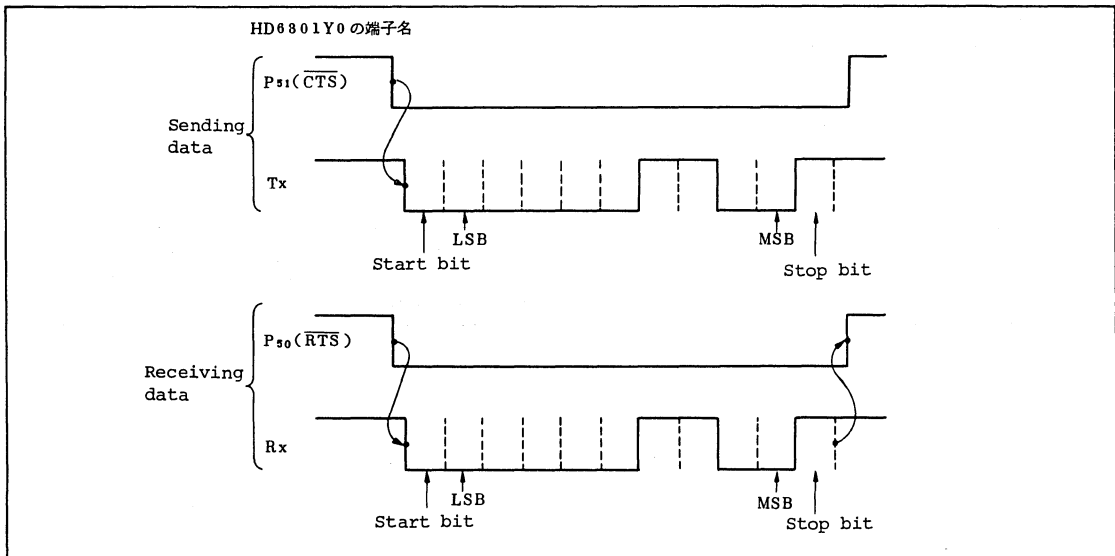


Figure 10-2. Timing Chart for Sending/Receiving Data

10.2.1 Program Module Configuration

The program module configuration for sending/receiving data to/from the console typewriter is shown in figure 10-3.

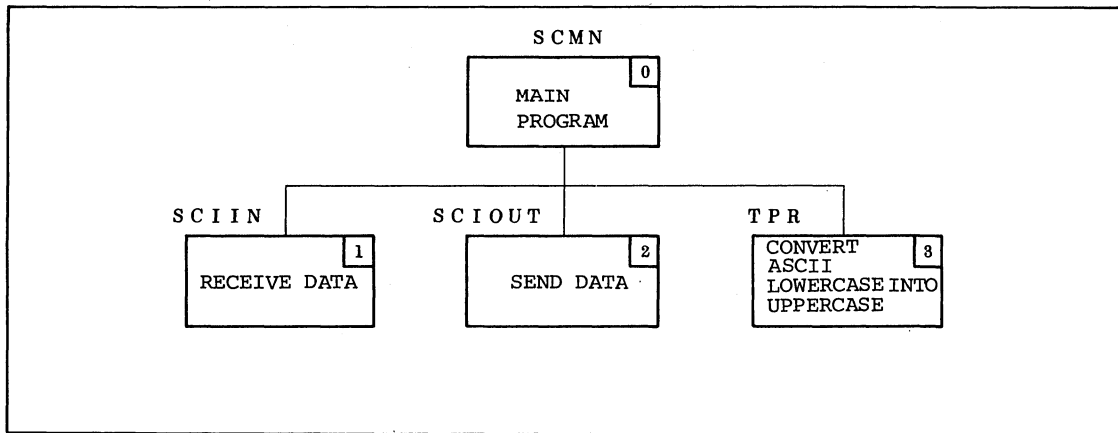


Figure 10-3. Program Module Configuration

10.2.2 Program Module Functions

Program module functions are summarized in table 10-2.

Table 10-2. Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	SCMN	Send/Receive data to/from the console typewriter using asynchronous SCI.
1	RECEIVE DATA	SCIIN	Receive data from the console typewriter.
2	SEND DATA	SCIOUT	Send data back to the console typewriter.
3	CONVERT ASCII LOWERCASE INTO UPPERCASE	TPR	Converts ASCII lowercase into uppercase. Refer to TPR in HD6301/HD6303 FAMILY APPLICATION NOTES (SOFTWARE) for details.

10.2.3 Program Module Process Flow (Main Program)

The flowchart in figure 10-4 shows the procedure for sending/receiving data to/from the console typewriter, using the program module in figure 10-3. If ASCII lowercase characters are received, the main program converts them into uppercase.

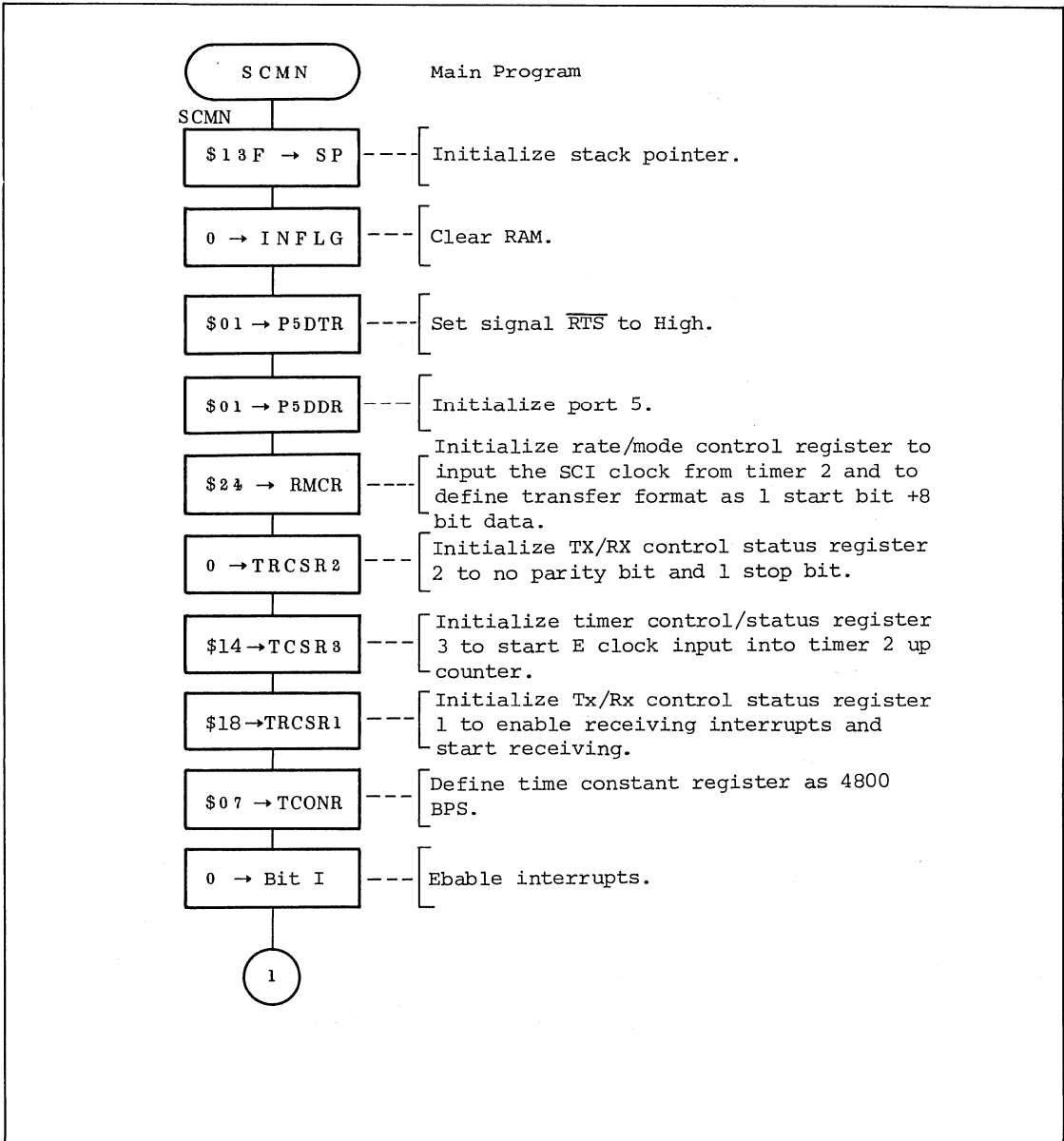


Figure 10-4. Program Module Flowchart

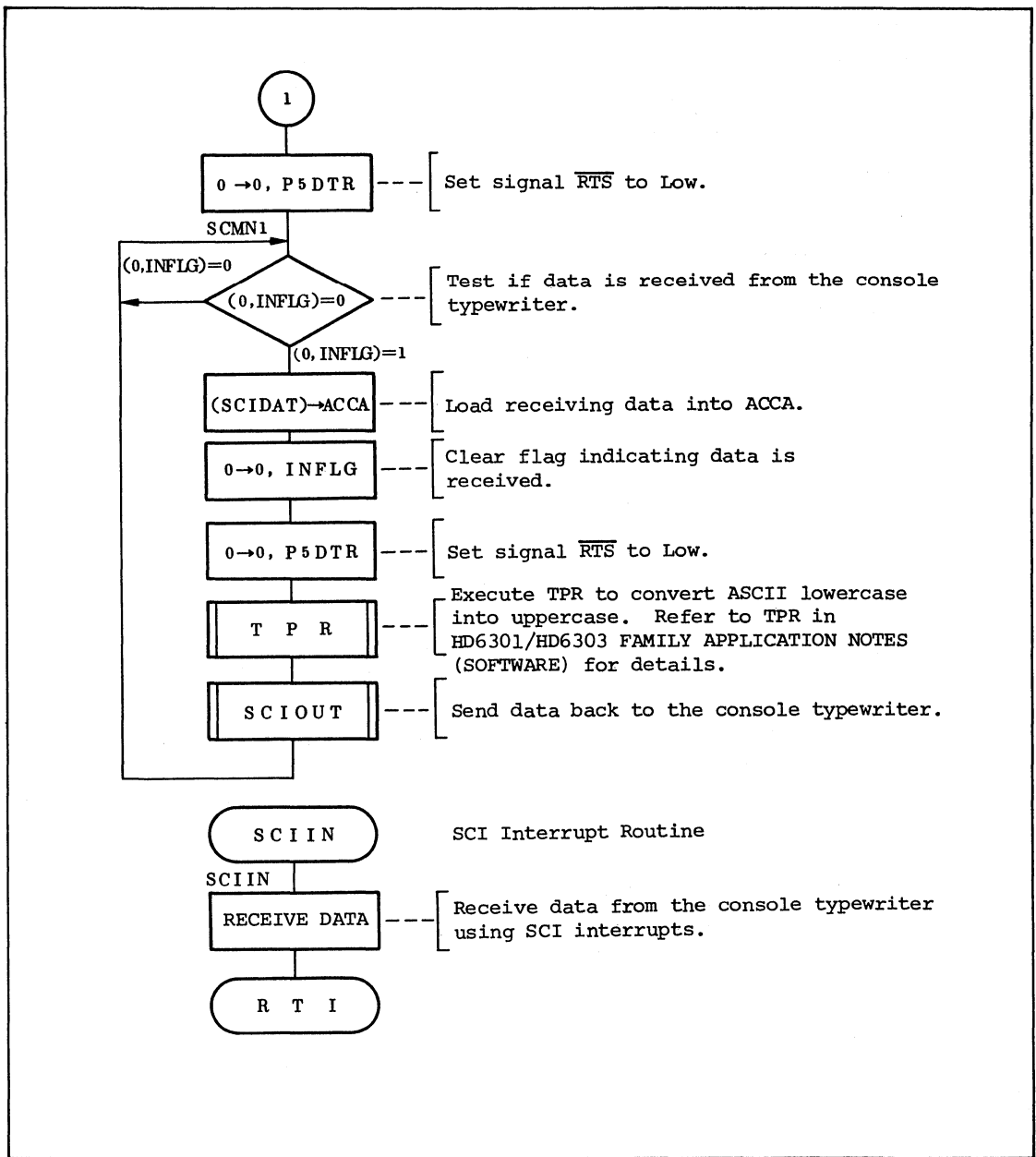


Figure 10-4. Program Module Flowchart (Cont.)

### 10.3 PROGRAM MODULE DESCRIPTION

Program Module Name: RECEIVE DATA

MCU/MPU: HD6301Y0/  
HD6303Y

Label: SCIIN

Function:

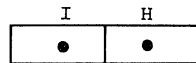
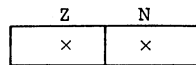
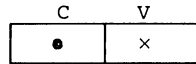
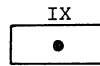
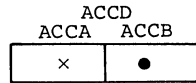
Receives serial data from the console typewriter.

Arguments:

Contents	Storage Location	No. of Bytes	
Entry	---	---	
Re-returns	Data received	SCIDAT (RAM)	1
	Receiving complete flag	INFLG (RAM)	1

Changes in CPU

Registers and Flags:



● : Not affected  
x : Undefined  
↑ : Result

Specifications:

ROM (Bytes): 22  
RAM (Bytes): 2  
Stack (Bytes): 0  
No. of cycles: 41  
Reentrant: No  
Relocatable: No  
Interrupt OK?: Yes

Description:

1. Function Details

a. Argument details

SCIDAT (RAM): Holds data received.

INFLG (RAM) : Used as flag indicating whether or not receiving is completed.

Table 10-3 shows flag functions.

b. Example of SCIIN execution is shown in figure 10-5. If bit 0 of port5 (signal  $\overline{RTS}$ ) is set to Low, data sent from the console typewriter is stored in SCIDAT (RAM).

Specifications Notes:

"No. of cycles" in "Specifications" indicates the number of cycles required to receive data.

Description:

Table 10-3. Flag Functions

Label	bit 0	Function
INFLG	0	Indicates data is not received.
	1	Indicates data is received.

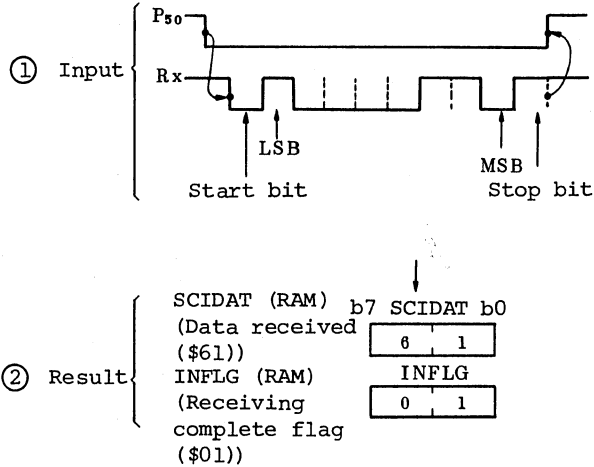


Figure 10-5. Example of SCIIN Execution

c. SCIIN calls neither the program modules nor subroutines.

2. User Notes

The following procedure must be performed before SCIIN execution.

- a. Select DDR of bit 0 of port 5 as output.
- b. Initialize SCI since asynchronous SCI is used.
- c. Enable interrupts for SCI interrupts.

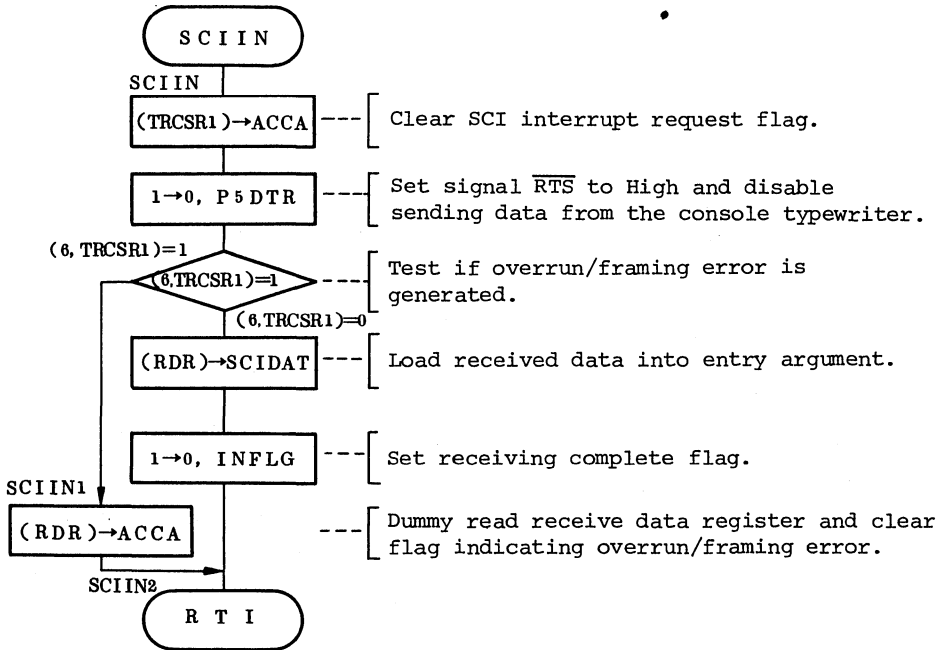
3. RAM Allocation

Label	RAM	Description
SCIDAT	b7                  b0	} Data sent from the console typewriter. } Used as flag indicating whether or not data is received.
INFLG		





Flowchart:



Program Module Name: SEND DATA

MCU/MPU: HD6301Y0/  
HD6303Y

Label: SCIOUT

Function:

Sends data, loaded into ACCA, back to the console typewriter.

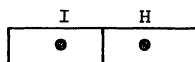
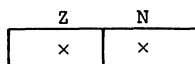
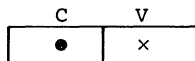
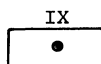
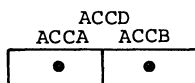
Arguments:

Contents	Storage Location	No. of Bytes
Entry Data to be sent	ACCA	1

Re- turns \_\_\_\_\_

Changes in CPU

Registers and Flags:



● : Not affected  
x : Undefined  
↓ : Result

Specifications:

ROM (Bytes): 16  
RAM (Bytes): 0  
Stack (Bytes): 0  
No. of cycles: 28  
Reentrant: No  
Relocatable: No  
Interrupt OK?: Yes

Description:

1. Function Details

a. Argument details

ACCA: Holds data to be sent back to the console typewriter.

b. Example of SCIOUT execution is shown in figure 10-6.

If entry argument is as shown in part ① of figure 10-6, data is sent to the console typewriter as shown in part ② of figure 10-6.

c. SCIOUT calls neither the program modules nor subroutines.

Specifications Notes:

N/A

Description:

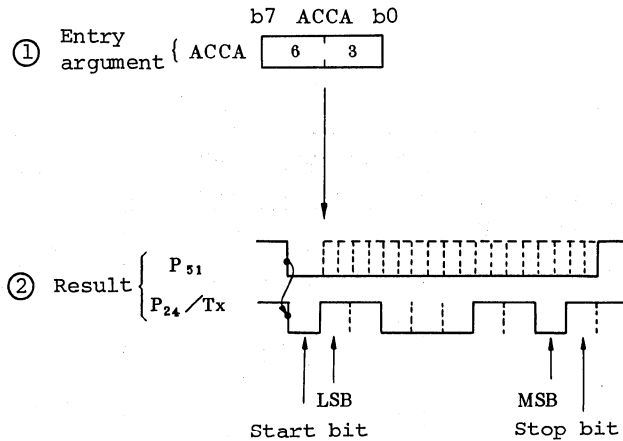


Figure 10-6. Example of SCIOUT Execution

2. User Notes

Initialize SCI for asynchronous SCI.

3. RAM Allocation

RAM is not used during SCIOUT execution.

4. Sample Application

```

LDAA  #$24  } ----- Initialize rate/mode control register.
STAA  RMCR
CLRA
STAA  TRCSR2 } ----- Initialize Tx/Rx control status register 2.
LDAA  #$14
STAA  TCSR3  } ----- Initialize timer control/status register 3.
LDAA  #$18
STAA  TRCSR1 } ----- Initialize Tx/Rx control status register 2.
LDAA  #$07
STAA  TCONR  } ----- Initialize time constant register and define
LDAA  #$30  } ----- Load data to be sent.
JSR   SCIOUT } ----- Execute SCIOUT.
    
```

Program Module Name: SEND DATA

MCU/MPU: HD6301Y0/  
HD6303Y

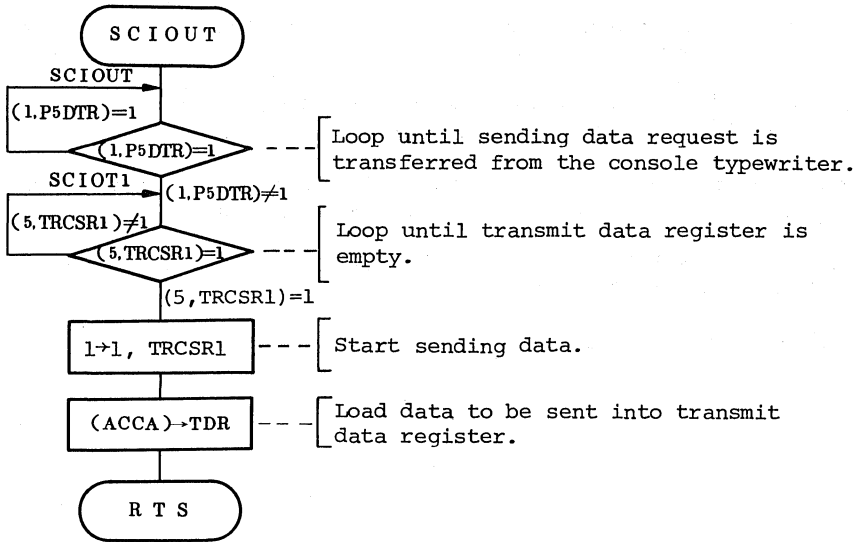
Label: SCIOUT

Description:

5. Basic Operation

- a. Whether or not there is a sending data request from the console typewriter is determined by signal  $\overline{\text{CTS}}$ .
- b. If signal  $\overline{\text{CTS}}$  is Low, data can be sent to the console typewriter. If signal  $\overline{\text{CTS}}$  is High, wait until signal  $\overline{\text{CTS}}$  is Low.
- c. Whether or not transmit data register is empty is tested by transmit data register empty bit (bit 5) of Tx/Rx control status register 1.
- d. Transmit enable bit (bit 1) of Tx/Rx control status register 1 is set to "1" and data sending is started.
- e. Data to be sent is stored in transmit data register.

Flowchart:



## 10.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

## 10.5 PROGRAM LISTING

```

00001          *
00002          ***** RAM ALLOCATION *****
00003          *
00004A 0040          ORG      $40
00005          *
00006A 0040    0001  A INFLG  RMB    1      Receiving complete flag
00007A 0041    0001  A SCIDAT RMB    1      Data received
00008          *
00009          ***** SYMBOL DEFINITIONS *****
00010          *
00011          0010  A RMCR   EQU    $10      RATE/MODE CRTL REG
00012          0011  A TRCSR1 EQU    $11      Tx/Rx CRTL REG 1
00013          0012  A RDR   EQU    $12      RECEIVE DATA REG
00014          0013  A TDR   EQU    $13      TRANSMIT DATA REG
00015          0015  A PSDTR  EQU    $15      PORTS DATA REG
00016          0018  A TCSR3  EQU    $18      TIMER CRTL REG 3
00017          001C  A TCONR  EQU    $1C      TIME CONSTANT REG
00018          001E  A TRCSR2 EQU    $1E      Tx/Rx CRTL REG 2
00019          0020  A PSDDR  EQU    $20      PORTS DDR
00020          *****
00021          *
00022          *          MAIN PROGAM : SCMN          *
00023          *
00024          *****
00025          *
00026A C000          ORG      $C000
00027          *
00028A C000 8E 013F  A SCMN   LDS     #$13F    Initialize stack pointer
00029A C003 4F          CLRA          Clear RAM
00030A C004 97 40      A          STAA   INFLG
00031A C006 86 01      A          LDAA   #$01    Initialize port 5
00032A C008 97 15      A          STAA   PSDTR
00033A C00A 97 20      A          STAA   PSDDR
00034A C00C 86 24      A          LDAA   #$24    Initialize RMCR
00035A C00E 97 10      A          STAA   RMCR
00036A C010 4F          CLRA          Initialize TRCSR2
00037A C011 97 1E      A          STAA   TRCSR2
00038A C013 86 14      A          LDAA   #$14    Initialize TCSR3
00039A C015 97 1B      A          STAA   TCSR3
00040A C017 86 18      A          LDAA   #$18    Initialize TRCSR1
00041A C019 97 11      A          STAA   TRCSR1
00042A C01B 86 07      A          LDAA   #07      Set 4800BPS
00043A C01D 97 1C      A          STAA   TCONR
00044A C01F 0E          CLI           Enable interrupts
00045A C020 71 FE 15    BCLR    0,PSDTR  Set RTS=0
00046A C023 7B 01 40    SCMN1   BTST    0,INFLG  Test if data is received
00047A C026 27 FB C023  BEQ     SCMN1
00048A C028 96 41      A          LDAA   SCIDAT    Load SCI data
00049A C02A 71 FE 40    BCLR    0,INFLG  Clear SCI flag
00050A C02D 71 FE 15    BCLR    0,PSDTR  Set RTS=0
00051A C030 8D C038  A   JSR     TPR      Convert lowercase into uppercase
00052A C033 8D C059  A   JSR     SCIOUT   Send data
00053A C036 20 EB C023  BRA     SCMN1
00054          *****
00055          *
00056          * NAME : TPR (CONVERT 8 BIT BINARY INTO *
00057          *          ASCII)          *

```

```

00058      *
00059      *****
00060      *
00061      *   ENTRY : ACCA (8 BIT BINARY) *
00062      *   RETURNS : ACCA (8 BIT BINARY) *
00063      *
00064      *****
00065      *
00066A C038 81 61   A TPR   CMPA   H'a     ACCA-'a'
00067A C03A 25 06 C042    BCS   TPR1   Branch if ACCA<'a'
00068A C03C 81 7A   A      CMPA   H'z     ACCA-'z'
00069A C03E 22 02 C042    BHI   TPR1   Branch if ACCA>'z'
00070A C040 84 DF   A      ANDA   H$DF   Convert lowercase into uppercase
00071A C042 39      TPR1   RTS
00072      *****
00073      *
00074      *   NAME : SCIIN (RECEIVE DATA) *
00075      *
00076      *****
00077      *
00078      *   ENTRY : NOTHING *
00079      *   RETURNS : SCIDAT (DATA RECEIVED) *
00080      *           INFLG (RECEIVING COMPLETE *
00081      *           FLAG) *
00082      *
00083      *****
00084A C043 96 11   A SCIIN LDAA   TRCSR1  Clear interrupt request flag
00085A C045 72 01 15 BSET   0.PSDTR Set RTS='1'
00086A C048 78 40 11 BTST   6.TRCSR1 ORFE bit='0' or '1'
00087A C04B 26 09 C056 BNE    SCIIN1  Branch if ORFE='1'
00088A C04D 96 12   A      LDAA   RDR     Set receiving complete flag
00089A C04F 97 41   A      STAA  SCIDAT
00090A C051 72 01 40 BSET   0.INFLG Set SCI flag
00091A C054 20 02 C058 BRA    SCIIN2
00092A C056 96 12   A SCIIN1 LDAA   RDR     Clear error flag
00093A C058 38      SCIIN2 RTI
00094      *****
00095      *
00096      *   NAME : SCIOUT (SEND DATA) *
00097      *
00098      *****
00099      *
00100      *   ENTRY : ACCA (DATA TO BE SENT) *
00101      *   RETURNS : NOTHING *
00102      *
00103      *****
00104A C059 78 02 15 SCIOUT BTST   1.PSDTR  Loop until CTS=0
00105A C05C 26 FB C059 BNE    SCIOUT
00106A C05E 78 20 11 SCIOT1 BTST   5.TRCSR1 Loop until TDRE=0
00107A C061 27 FB C05E BEQ    SCIOT1
00108A C063 72 02 11 BSET   1.TRCSR1 Start sending data
00109A C066 97 13   A      STAA  TDR     Load data to be sent
00110A C068 39      RTS
00111      *****
00112      *
00113      *   VECTOR ADDRESSES *
00114      *

```

```

00115 *****
00116 *
00117A FFEA          ORG    $FFEA
00118 *
00119A FFEA    C000  A    FDB    SCMN    IRQ2
00120A FFEC    C000  A    FDB    SCMN    CMI
00121A FFEE    C000  A    FDB    SCMN    TRAP
00122A FFF0    C043  A    FDB    SCIIN   SIO
00123A FFF2    C000  A    FDB    SCMN    TOI
00124A FFF4    C000  A    FDB    SCMN    OCI
00125A FFF6    C000  A    FDB    SCMN    ICI
00126A FFF8    C00C  A    FDB    SCMN    IRQ1/ISF
00127A FFFA    C000  A    FDB    SCMN    SWI
00128A FFFC    C000  A    FDB    SCMN    NMI
00129A FFFE    C000  A    FDB    SCMN    RES
00130 *
00131          END

```





### 11.1.5 Pin Functions

Pin functions at the interface between the HD6301Y0 and the HD61100A are shown in table 11-1.

Table 11-1. Pin Functions

Pin Name (HD6301Y0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (HD61100A LCD)	Program Label
P <sub>21</sub>	Output	—	Outputs alternate signal for LCD driving output.	M	P2DTR
P <sub>20</sub>	Output	—	Resets counter, outputs synchronous signal of latch clock for display data.	CL <sub>1</sub>	
P <sub>23</sub>	Output	—	Outputs common signal to LCD.	COMMON	
P <sub>22</sub> /SCLK	Output	—	Outputs shift clock for display data.	CL <sub>2</sub>	—
P <sub>24</sub> /Tx	Output	—	Inputs display data.	DL	—

### 11.1.6 Hardware Operation

Timing chart of the HD6301Y0, LCD, and the HD61100A is shown in figure 11-2.

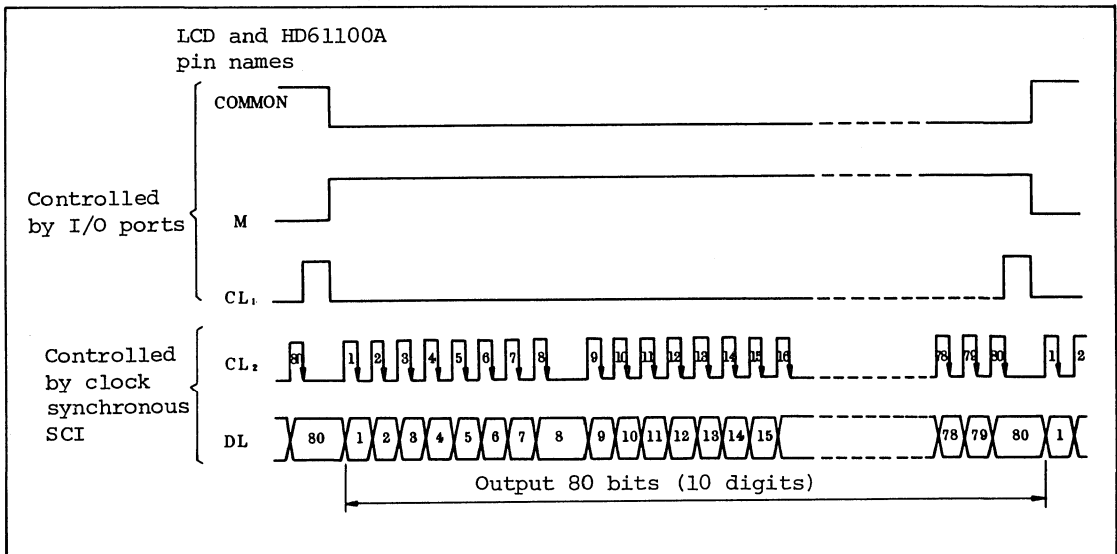


Figure 11-2. Timing Chart of HD6301Y0, LCD, and HD61100A

## 11.2 SOFTWARE DESCRIPTION

### 11.2.1 Program Module Configuration

The program module configuration for character display on LCD is shown in figure 11-3.

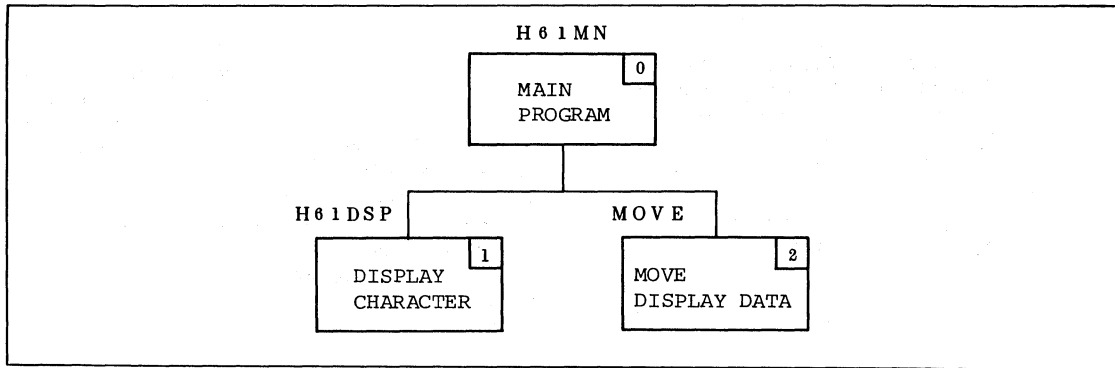


Figure 11-3. Program Module Configuration

### 11.2.2 Program Module Functions

Program module functions are summarized in table 11-2.

Table 11-2. Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	H61MN	Initializes control registers and data addresses used for the interface between the HD6301Y0 and the HD61100A.
1	DISPLAY CHARACTER	H61DSP	Performs static drive of LCD using the HD61100A and displays numerals.
2	MOVE DISPLAY DATA	MOVE	Moves display data in data table to display RAM. Refer to MOVE in HD6301/HD6303 FAMILY APPLICATION NOTES (SOFTWARE) for details.

### 11.2.3 Program Module Process Flow (Main Program)

The flowchart in figure 11-4 shows the procedure for displaying numerals on an LCD as performed by the program module in figure 11-3.

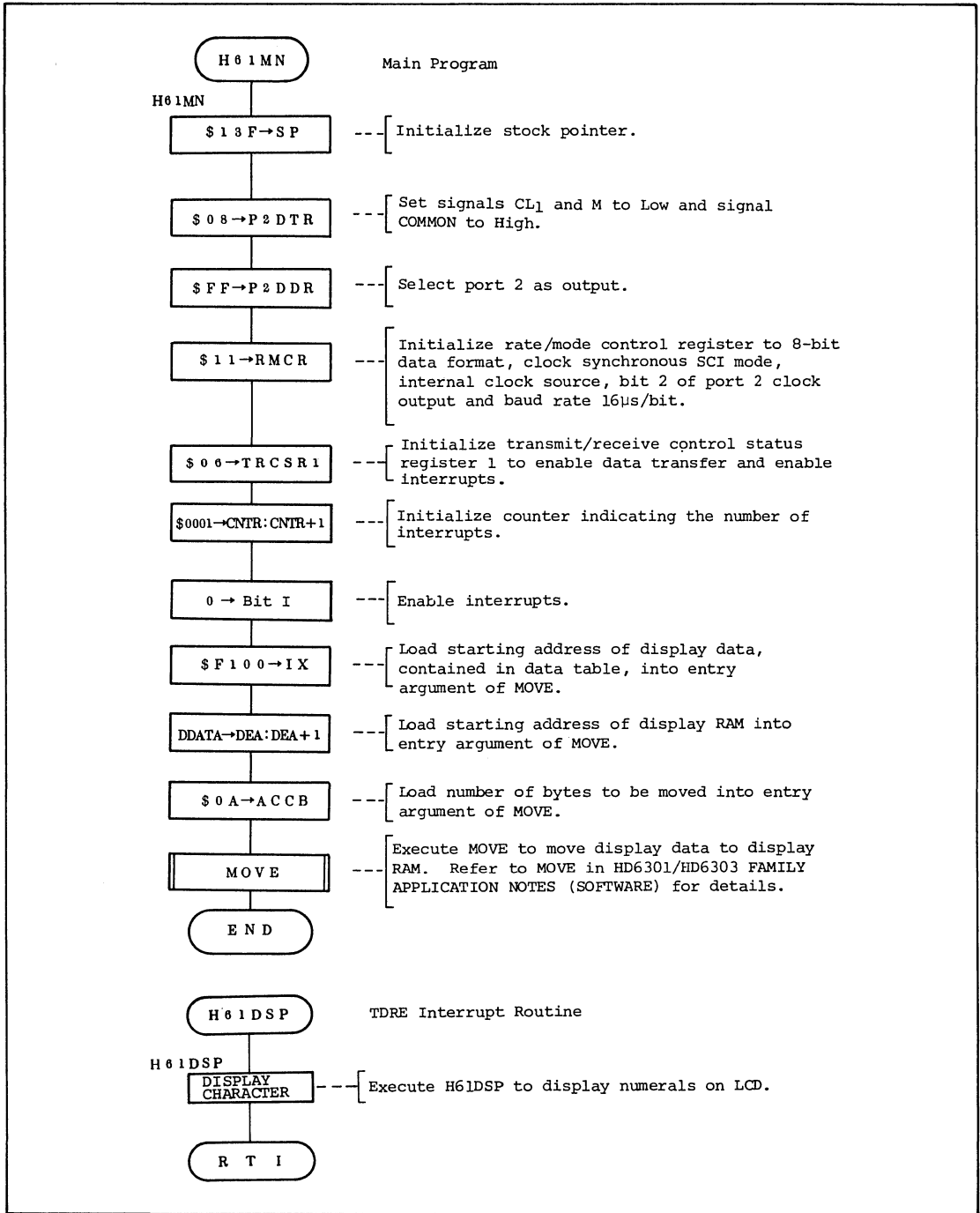


Figure 11-4. Program Module Flowchart

9 8 7 6 5 4 3 2 1 0

Figure 11-5. Example of H61MN Execution

### 11.3 PROGRAM MODULE DESCRIPTION

Program Module Name: DISPLAY  
CHARACTER

MCU/MPU: HD6301Y0/  
HD6303Y

Label: H61DSP

Function:

Sends display data to the HD61100A and displays characters on an LCD.

Arguments:

Contents	Storage Location	No. of Bytes
Entry Display data	DDATA (RAM)	10

Re-  
turns

Changes in CPU

Registers and Flags:

ACCD	
ACCA	ACCB
×	●

IX
×

C	V
●	×

Z	N
×	×

I	H
×	●

● : Not affected  
× : Undefined  
↑ : Result

Specifications:

ROM (Bytes): 41  
RAM (Bytes): 12  
Stack (Bytes): 0  
No. of cycles: 61  
Reentrant: No  
Relocatable: No  
Interrupt OK?: No

Description:

1. Function Details

a. Argument details

DDATA (RAM): Holds 10 bytes of display data.

b. Example of H61DSP execution is shown in figure 11-6. If entry argument is as shown in part 1 of figure 11-6, characters are displayed as shown in part 2 of figure 11-6.

c. H61DSP calls neither the program modules nor subroutines.

Specifications Notes:

N/A

**Program Module Name:** DISPLAY CHARACTER

**MCU/MPU:** HD6301Y0/  
HD6303Y

**Label:** H61DSP

Description:

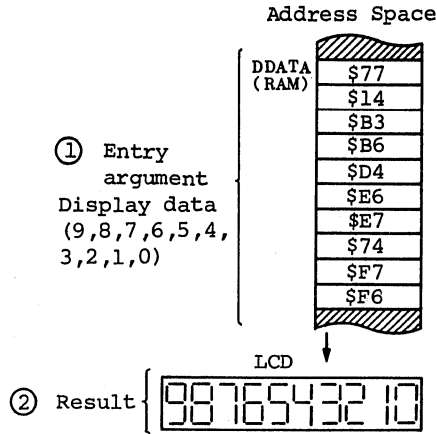


Figure 11-6. Example of H61DSP Execution

2. User Notes

The following procedure must be performed before H61DSP execution.

- a. Select data direction register (DDR) of port 2 as output.
- b. Initialize clock synchronous SCI to send display data.
- c. Clear bit I to enable TDRE interrupts.
- d. Store display data in TDR to generate TDRE interrupts.

3. RAM Allocation

Label	RAM	Description
	b7                      b0	
CNTR	[                      ]	} Used as a pointer to display RAM and as a counter indicating number of interrupts.  } Display data
DDATA	10 <sup>0</sup> digit	
	10 <sup>1</sup> digit	
	10 <sup>2</sup> digit	
	10 <sup>3</sup> digit	
	10 <sup>4</sup> digit	
	10 <sup>5</sup> digit	
	10 <sup>6</sup> digit	
	10 <sup>7</sup> digit	
	10 <sup>8</sup> digit	
	10 <sup>9</sup> digit	

Description:

4. Sample Application

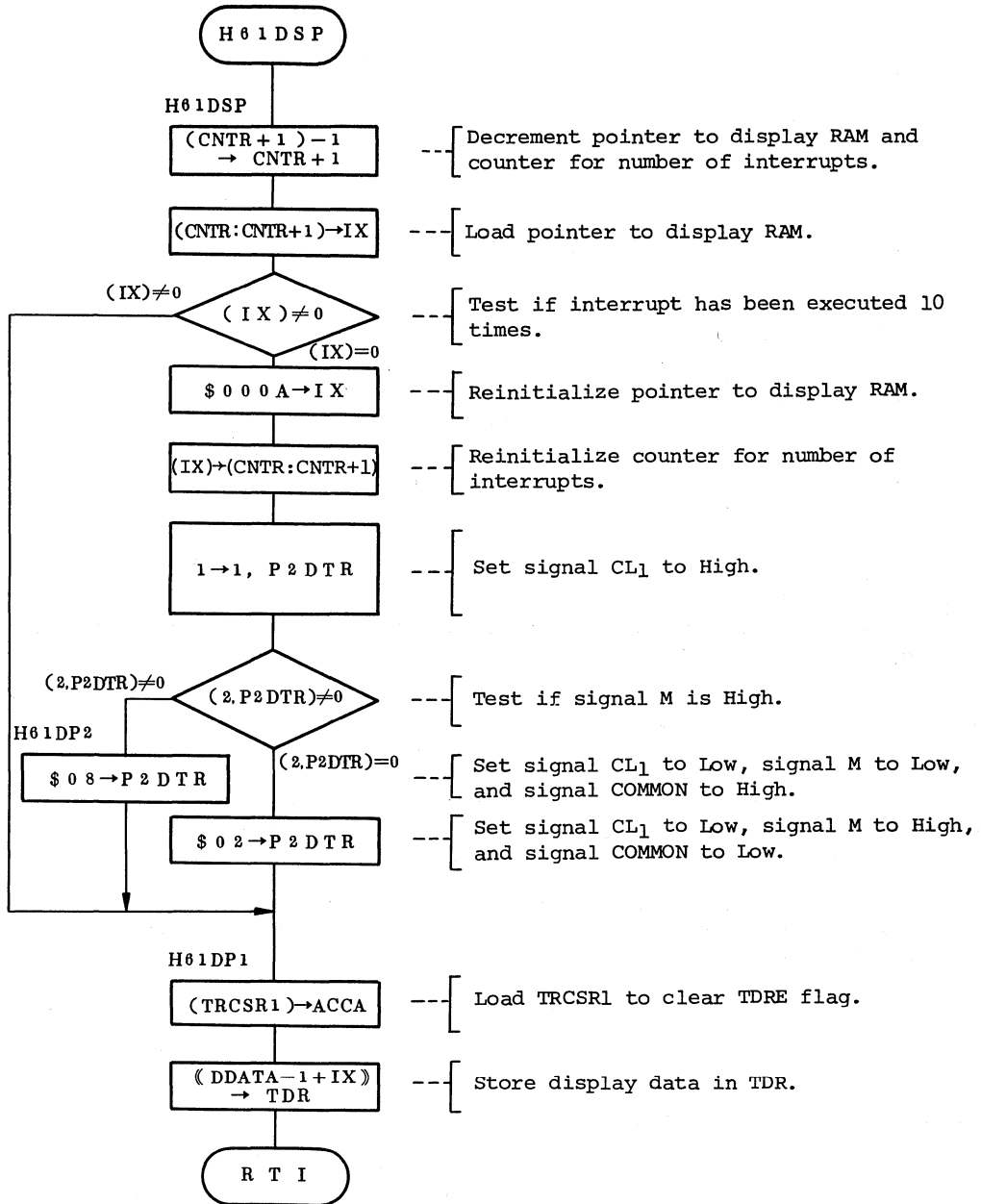
```
      :  
      :  
LDAA  #$08  }  
STAA  P2DTR }----- Initialize port 2.  
LDAA  #$FF  }  
STAA  P2DDR  }  
LDAA  #$11  }  
STAA  RMCR   }----- Initialize synchronous SCI.  
LDAA  #$06  }  
STAA  TRCSR1 }  
LDX   #$0001 }----- Initialize pointer to display RAM and counter  
STX   CNTR   }----- for number of interrupts.  
CLI   }----- Enable interrupts.  
LDX   #$F100 }  
LDD   #DDATA }----- Execute MOVE to load display data into  
STD   DEA    }----- entry argument of H61DSP.  
LDAB  #$0A   }----- (Refer to MOVE in HD6301/HD6303 FAMILY  
JSR   MOVE   }----- APPLICATION NOTES (SOFTWARE) for details.)  
      :  
      :
```

5. Basic Operation

- a. 10 bytes of display data are sent to the HD61100A to display numerals on an 8 segments × 10 digits LCD. Shift clock and data signal are controlled by the clock synchronous SCI of the HD6301Y0.
- b. Display data is stored in display RAM before execution. Each TDRE interrupt executes display of 1 byte of data.
- c. Pointer to display RAM and counter for number of interrupts are decremented every interrupt. CNTR(RAM) is reinitialized each time 10 interrupts are executed.
- d. The first enabling interrupts are performed by the main program. From then on, TDRE interrupts are generated automatically each time display data are outputted.



Flowchart:



## 11.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

## 11.5 PROGRAM LISTING

```
00001          *
00002          ***** RAM ALLOCATION *****
00003          *
00004          *
00005A 0040          ORG      $40
00006          *
00007A 0040    000A  A DDATA  RMB    10      Display data
00008A 004A    0002  A CNTR   RMB     2      Counter for display data
00009A 004C    0002  A DEA    RMB     2      Destination address
00010          *
00011          ***** SYMBOL DEFINITIONS *****
00012          *
00013          0003  A P2DTR  EQU    $03      Port 2 data register
00014          0001  A P2DDR  EQU    $01      Port 2 data direction register
00015          0010  A RMCR   EQU    $10      Rate/mode control register
00016          0011  A TRCSR1 EQU    $11      Tx/Rx control status register1
00017          0013  A TDR    EQU    $13      Transmit data register
00018          *****
00019          *
00020          *          MAIN PROGRAM : H61MN          *
00021          *
00022          *****
00023          *
00024A C000          ORG      $C000
00025          *
00026A C000 8E 013F  A H61MN  LDS    #$13F    Initialize stack pointer
00027A C003 86 08   A        LDAA   #$08      Set CL1=0,M=0,COMMON=1
00028A C005 97 03   A        STAA   P2DTR
00029A C007 86 FF   A        LDAA   #$FF      Select port 2 as output
00030A C009 97 01   A        STAA   P2DDR
00031A C00B 86 11   A        LDAA   #$11      Initialize RMCR
00032A C00D 97 10   A        STAA   RMCR
00033A C00F 86 06   A        LDAA   #$06      Initialize TRCSR1
00034A C011 97 11   A        STAA   TRCSR1
00035A C013 CE 0001  A        LDX    #$0001    Initialize counter
00036A C016 DF 4A   A        STX    CNTR
00037A C018 0E     A        CLI     Enable interrupts
00038A C019 CE F100  A        LDX    #$F100    Load source address
00039A C01C CC 0040  A        LDD    HDDATA    Load destination address
00040A C01F DD 4C   A        STD    DEA
00041A C021 C6 0A   A        LDAB   #$0A      Initialize transfer counter
00042A C023 BD C04D  A        JSR    MOVE     Move data table to display RAM
00043A C026 20 FE C026 PEND  BRA    PEND    End of program
00044          *****
00045          *
00046          *          NAME : H61DSP (DISPLAY CHARACTER)          *
00047          *
00048          *****
00049          *
00050          *          ENTRY : DDATA (DISPLAY DATA)          *
00051          *          RETURNS : NOTHING          *
00052          *
00053          *****
00054A C028 7A 004B  A H61DSP  DEC    CNTR+1    Decrement counter
00055A C02B DE 4A   A        LDX    CNTR      Test if CNTR=0 ?
00056A C02D 26 11 C040 A        BNE    H61DP1    Branch if not CNTR=0
00057A C02F CE 000A  A        LDX    #$000A    Reinitialize counter
```

```

00058A C032 DF 4A A STX CNTR
00059A C034 72 02 03 BSET 1.P2DTR Set CL1=1
00060A C037 7B 04 03 BTST 2.P2DTR Branch if M=1
00061A C03A 26 0B C047 BNE H61DP2 Branch to H61DP2 if M=1
00062A C03C 86 02 A LDAA H$02 Set CL1=0,M=1.COMMON=0
00063A C03E 97 03 A STAA P2DTR
00064A C040 96 11 A H61DP1 LDAA TRCSR1 Clear TDRE
00065A C042 A6 3F A LDAA DDATA-1,X Store display data in TDR
00066A C044 97 13 A STAA TDR
00067A C046 3B RTI
00068A C047 86 08 A H61DP2 LDAA H$08 Set CL1=0,M=0.COMMON=1
00069A C049 97 03 A STAA P2DTR
00070A C04B 20 F3 C040 BRA H61DP1
00071 *****
00072 *
00073 * NAME : MOVE (MOVING MEMORY BLOCKS) *
00074 *
00075 *****
00076 *
00077 * ENTRY : IX (SOURCE ADDRESS) *
00078 * DEB (DESTINATION ADDRESS) *
00079 * ACCB (TRANSFER COUNTER) *
00080 * RETURNS : NOTHING *
00081 *
00082 *****
00083A C04D A6 00 A MOVE LDAA 0,X Load transfer data
00084A C04F 08 INX Increment source address
00085A C050 3C PSHX Push source address
00086A C051 DE 4C A LDX DEB Load destination address
00087A C053 A7 00 A STAA 0,X Store transfer data
00088A C055 08 INX Increment destination address
00089A C056 DF 4C A STX DEB Store destination address
00090A C058 3B PULXB Pull source address
00091A C059 5A DECB Decrement transfer counter
00092A C05A 26 F1 C04D BNE MOVE Branch until transfer counter = 0
00093A C05C 39 RTS
00094 *****
00095 *
00096 * DATA TABLE *
00097 *
00098 *****
00099 *
00100A F100 ORG $F100
00101 *
00102A F100 E7 A FCB $E7,$F7,$43,$F6,$E6 *Segment data
00103A F105 C5 A FCB $C5,$E3,$B3,$41,$77
00104 *****
00105 *
00106 * VECTOR ADDRESSES *
00107 *
00108 *****
00109 *
00110A FFEA ORG $FFEA
00111 *
00112A FFEA C000 A FDB H61MN IRQ2
00113A FFEC C000 A FDB H61MN CMI
00114A FFEF C000 A FDB H61MN TRAP

```

00115A	FFF0	C028	A	FDB	H61DSP	SIO
00116A	FFF2	C000	A	FDB	H61MN	TOI
00117A	FFF4	C000	A	FDB	H61MN	OCI
00118A	FFF6	C000	A	FDB	H61MN	ICI
00119A	FFF8	C000	A	FDB	H61MN	IRQ1/ISF
00120A	FFFA	C000	A	FDB	H61MN	SWI
00121A	FFFC	C000	A	FDB	H61MN	NMI
00122A	FFFE	C000	A	FDB	H61MN	RES
00123			*			
00124				END		

12.1 HARDWARE DESCRIPTION

12.1.1 Function

Receives data from the console typewriter, displays it on the H2571 liquid crystal module, and echoes the same data back to the console typewriter.

12.1.2 Microcomputer Operation

The HD6301Y0 controls the HM6264 RAM, HN27C64 EPROM, HD6350 ACIA and HD6321 PIA using external expansion mode (mode 1 of this MCU). In this mode, P<sub>50</sub> is employed as the  $\overline{IRQ}_1$  pin to receive interrupts from the ACIA. The MCU converts ASCII lowercase, sent from the console typewriter, into uppercase through software.

12.1.3 Peripheral Devices

HD6350 ACIA: Performs asynchronous SCI with the console typewriter, controlling signals  $\overline{RTS}$  and  $\overline{CTS}$  at a baud rate of 4800.

HD6321 PIA : Drives the liquid crystal module through ports A and B after receiving control information from the HD6301Y0.

HD74HC183 Address decoder: Decodes address signals from the MCU for control of the RAM, EPROM, PIA, and ACIA.

12.1.4 Circuit Diagram

External expansion circuit is shown in figure 12-1.

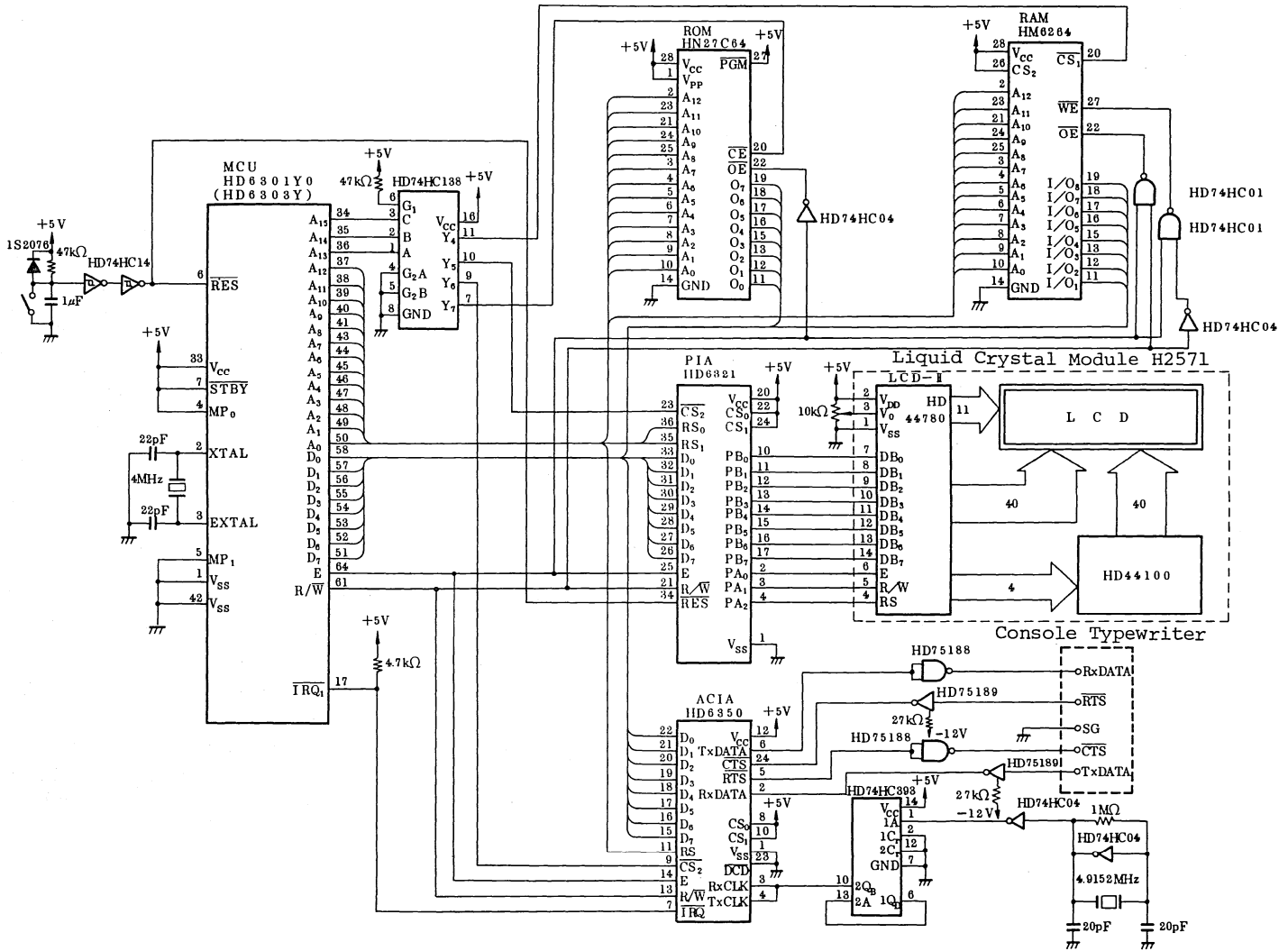


Figure 12-1. External Expansion Circuit

### 12.1.5 Memory Map

Memories and peripheral LSIs are allocated to external address space using an address decoder (HD74HC138).

Address buses A<sub>13</sub> and A<sub>14</sub> are connected to pins A and B of the HD74HC138. Address space \$8000~\$FFFF is divided into 8-byte units. System Address decoding is shown in Table 12-1.

Table 12-1. System Address Decode

HD74HC138						Address							Allocation	
Input			Output			space								
G <sub>1</sub>	G <sub>2A</sub>	G <sub>2B</sub>	C	B	A	Y <sub>0</sub>	Y <sub>1</sub>	Y <sub>2</sub>	Y <sub>3</sub>	Y <sub>4</sub>	Y <sub>5</sub>	Y <sub>6</sub>	Y <sub>7</sub>	
			A <sub>15</sub>	A <sub>14</sub>	A <sub>13</sub>									
H	L	L	H	L	L	H	H	H	H	L	H	H	H	\$8000~\$9FFF RAM
H	L	L	H	L	H	H	H	H	H	H	L	H	H	\$A000~\$BFFF PIA
H	L	L	H	H	L	H	H	H	H	H	L	H	H	\$C000~\$DFFF ACIA
H	L	L	H	H	H	H	H	H	H	H	H	L	H	\$E000~\$FFFF ROM

System memory map is shown in figure 12-2.

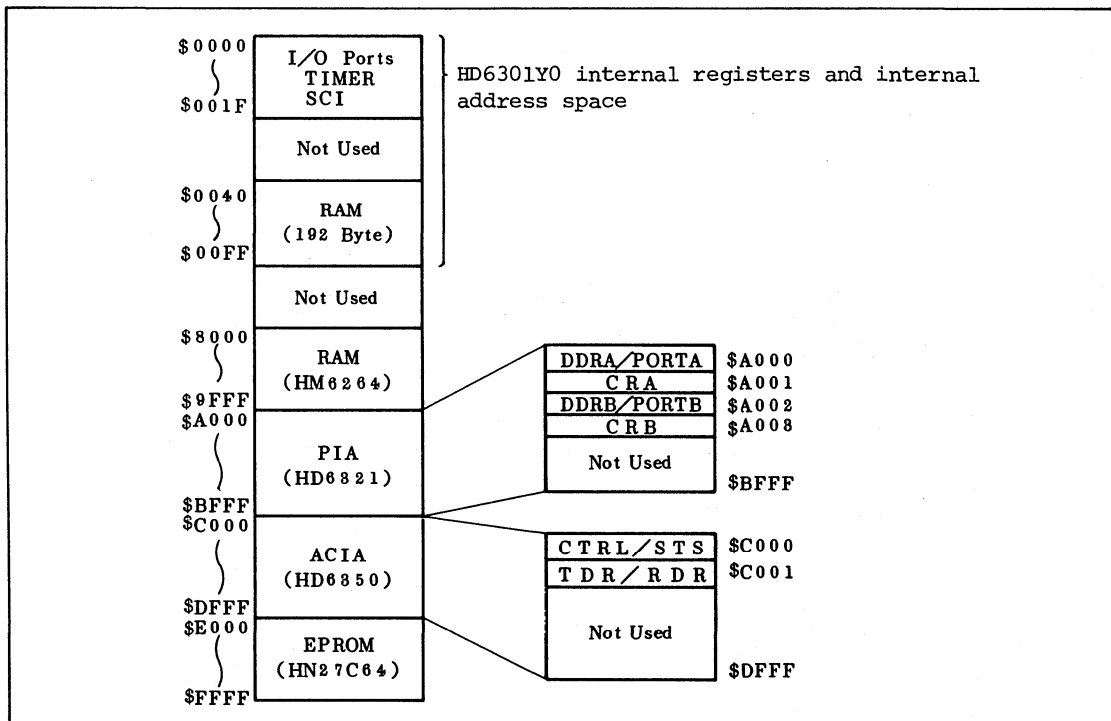


Figure 12-2. System Memory Map



The relation between HD6301Y0 specified addresses and selected PIA registers is shown in table 12-2.

Table 12-2. HD6301Y0 addresses and Selected PIA Registers

Address (HD6301Y0)	Program Label	Selected PIA Register	Pin Name (LCD-II)	Note
\$A000	DDRA	Data direction register A	—	When bit 2 of control register A=0.
	PIRA	Peripheral interface register A	Bit 0:signal E Bit 1:signal R/W Bit 2:signal RS	In case of bit 2 of control register A=1
\$A001	CRA	Control register A	—	—
\$A002	DDRB	Data direction register B	—	In case of bit 2 of control register B=0
	PIRB	Peripheral interface register B	—	In case of bit 2 of control register B=1
\$A003	CRB	Control register B	—	—

Note:

The timing chart between PIA and LCD-II is shown in figure 12-3.

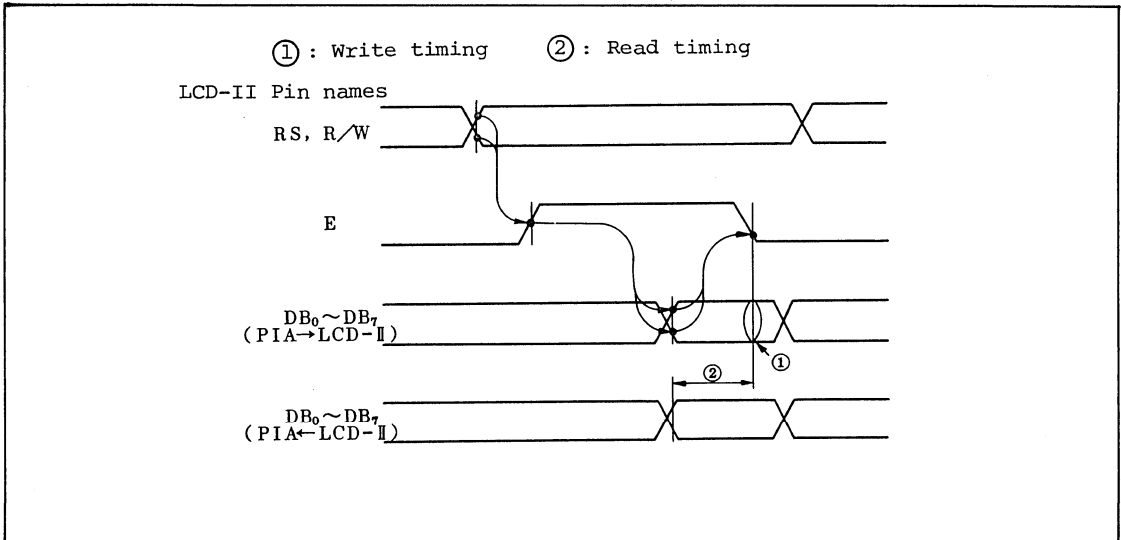


Figure 12-3. PIA↔LCD-II Timing Chart



The relation between the address of the HD6301Y0 and the selected ACIA register is shown in table 12-3.

Table 12-3. Relation between the Address of the HD6301Y0 and the Selected ACIA Register

Address (HD6301Y0)	Program Label	Selected ACIA Register	Note
\$C000	CR	Control register	In case of signal $R/\bar{W}=0$
	SR	Status register	In case of signal $R/\bar{W}=1$
\$C001	TDR	Transmit data register	In case of signal $R/\bar{W}=0$
	RDR	Receive data register	In case of signal $R/\bar{W}=1$

## 12.1.6 Hardware Operation

The interface timing chart between the HD6301Y0 and memories (HN27C64, HM6264) is shown in figure 12-4.

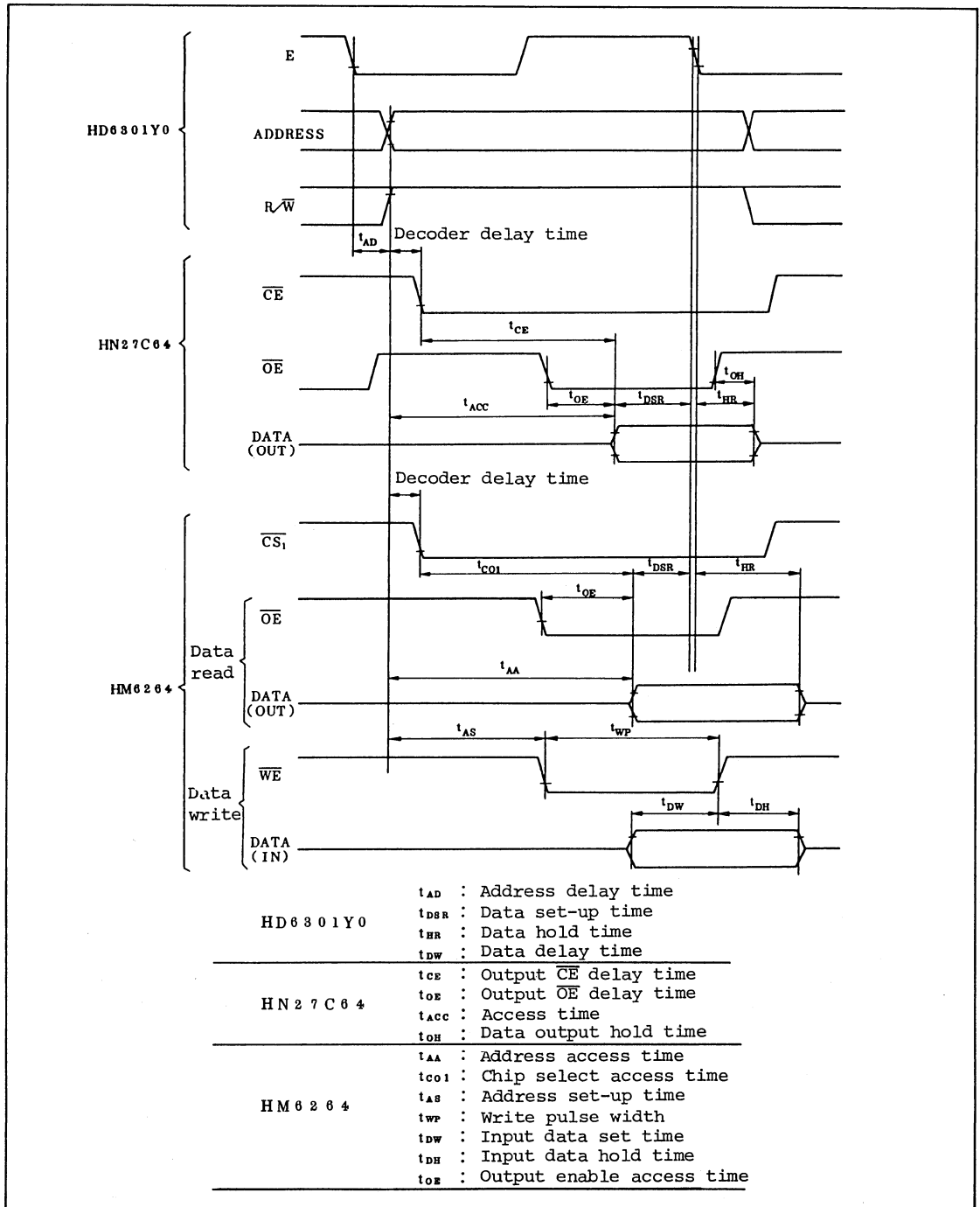


Figure 12-4. Interface Timing Chart between HD6301Y0 and Memories

## 12.2 SOFTWARE DESCRIPTION

### 12.2.1 Program Module Configuration

The program module configuration for receiving key data from the console typewriter and displaying data on both the console typewriter and H2571 is shown in figure 12-5.

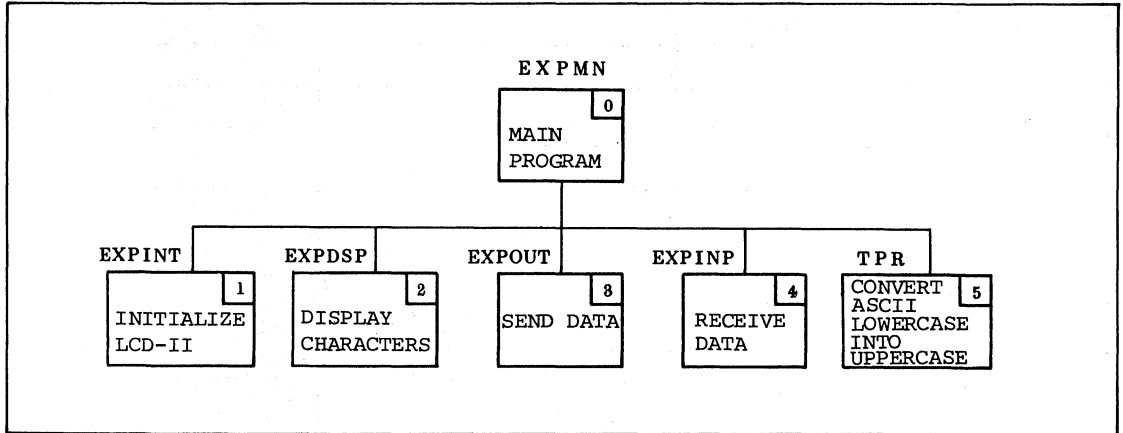


Figure 12-5. Program Module Configuration

### 12.2.2 Program Module Functions

Program module functions are summarized in table 12-4.

Table 12-4. Program Module Functions

No.	Program Module Name	Label	Functions
0	MAIN PROGRAM	EXP MN	Displays key data, received from the console typewriter, on both the H2571 and console typewriter.
1	INITIALIZE LCD-II	EXPINT	Initializes LCD-II contained in the H2571.
2	DISPLAY CHARACTERS	EXPDSP	Displays characters on the H2571.
3	SEND DATA	EXPOUT	Sends display data to the console typewriter.
4	RECEIVE DATA	EXPINP	Receives display data from the console typewriter.
5	CONVERT ASCII LOWERCASE INTO UPPERCASE	TPR	Converts ASCII lowercase into uppercase. Refer to TPR in HD6301/HD6303 FAMILY APPLICATION NOTES (SOFTWARE).

### 12.2.3 Program Module Process Flow (Main Program)

The flowchart in figure 12-6 demonstrates the procedure for displaying key data received from the console typewriter on both the H2571 and console typewriter, as performed by the program module in figure 12-5.

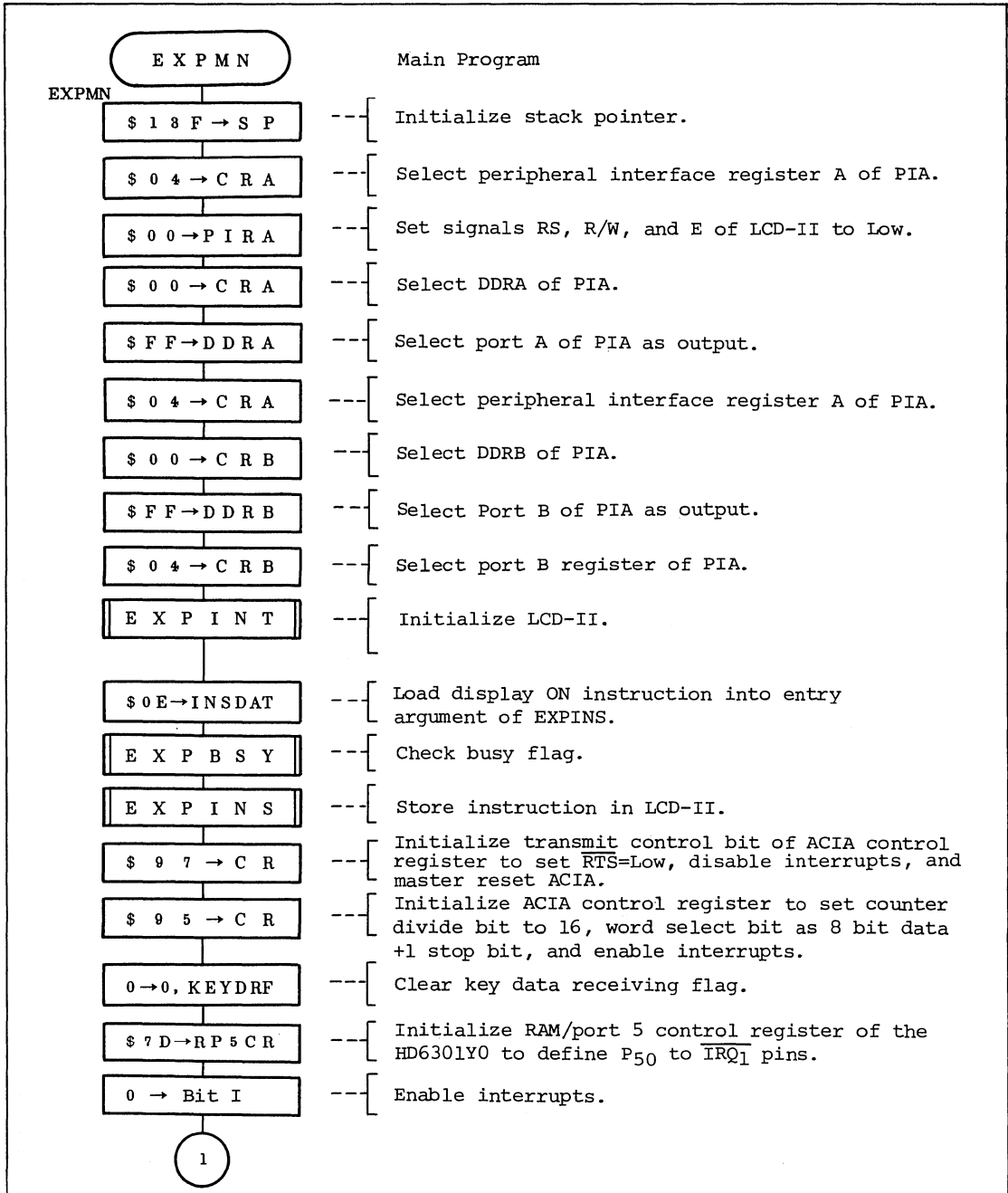


Figure 12-6. Program Module Flowchart

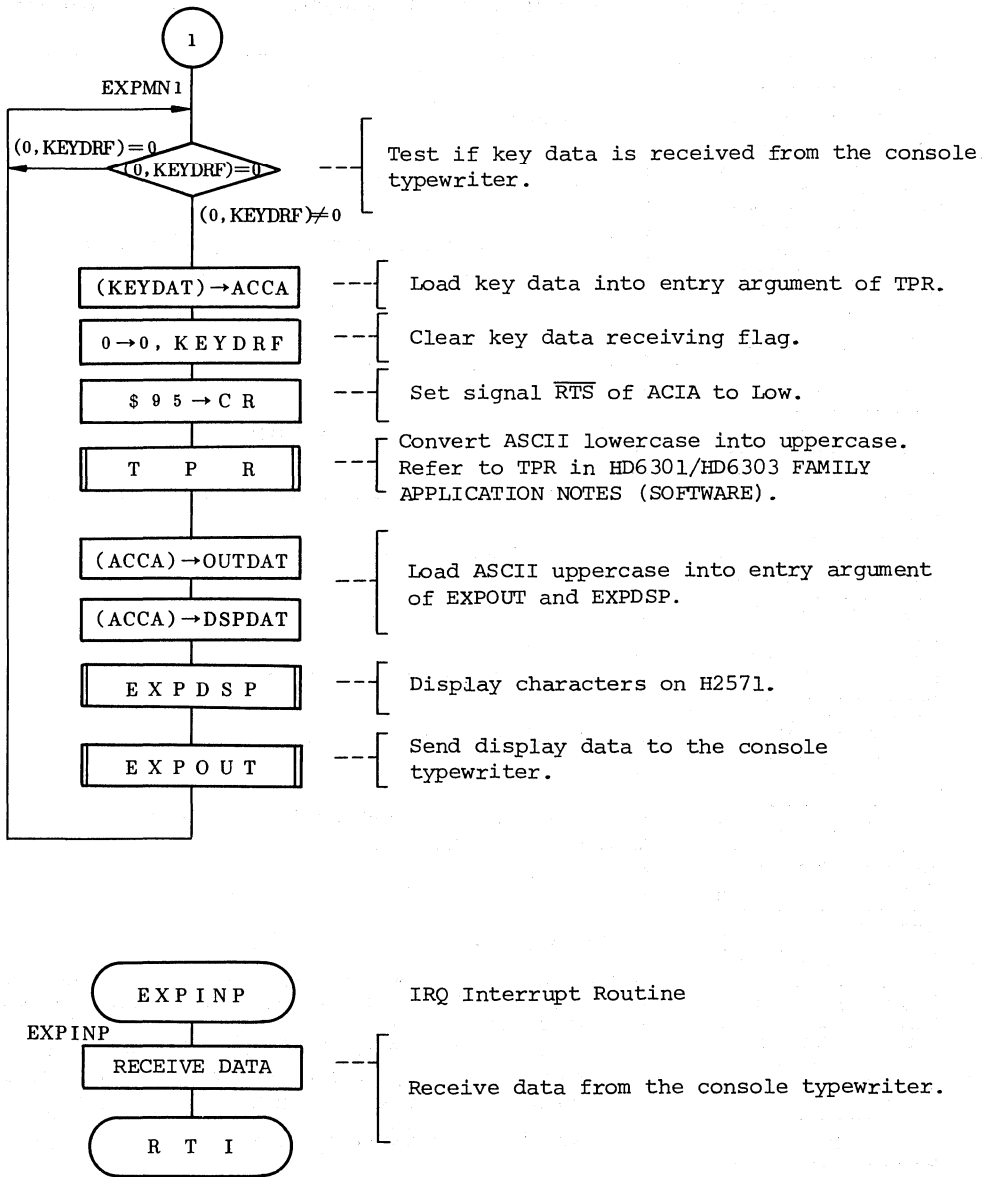


Figure 12-6. Program Module Flowchart (Cont.)

### 12.3 PROGRAM MODULE DESCRIPTION

Program Module Name: INITIALIZE  
LCD-II

MCU/MPU: HD6301Y0/  
HD6303Y

Label: EXPINT

Function:

Initializes LCD-II contained in the H2571 liquid crystal module.

Arguments:

None

Changes in CPU

Registers and Flags:

ACCD	
ACCA	ACCB
×	×

IX
●

C	V
×	×

Z	N
×	×

I	H
●	●

● : Not affected  
 × : Undefined  
 † : Result

Specifications:

ROM (Bytes): 37  
 RAM (Bytes): 1  
 Stack (Bytes): 2  
 No. of cycles: 45637  
 Reentrant: No  
 Relocatable: No  
 Interrupt OK?: Yes

Description:

1. Function Details

- a. EXPINT has no arguments.
- b. LCD-II is reset by instruction. Data in table 12-5 is sent to initialize display mode.

Specifications Notes:

1. "Specifications" includes those used by called subroutines.
2. "No. of cycles" in "Specifications" indicates the number of cycles required when EXPBSY is executed a minimum number of cycles.

Program Module Name: INITIALIZE  
LCD-II

MCU/MPU: HD6301Y0/  
HD6303Y

Label: EXPINT

Description:

Table 12-5. Initialize Data for Display Mode

Data	Function
\$30	Interface length: 8 bits      Columns: 1 Display Font: 5 × 7 dots
\$08	Display OFF, Cursor OFF, Blink OFF
\$01	Clear display, set DDRAM address to \$00
\$06	Increment DDRAM address, No display shift

- c. EXPINT calls other subroutines as shown in table 12-6.

Table 12-6. Subroutines Called by EXPINT

Subroutine Name	Label	Function
STORE INSTRUCTION	EXPINS	Store instruction in LCD-II.
CHECK BUSY FLAG	EXPBSY	Check LCD-II busy flag.

2. User Notes

- a. The following procedure is required before EXPINT execution.
- i. Initialize control signals (signals RS, R/W, E) of LCD-II.
  - ii. Select ports A and B as output.
  - iii. Initialize control register of PIA to select peripheral interface registers A and B.
- b. Instruction data shown in Table 12-5 must be reserved as data table.

3. RAM Allocation

Label	RAM	Description
INSDAT	b7                      b0 [ ]	} Instruction data





Program Module Name: INITIALIZE  
LCD-II

MCU/MPU: HD6301Y0/  
HD6303Y

Label: EXPINT

Description:

The procedure for initializing port A and write operation is as follows:

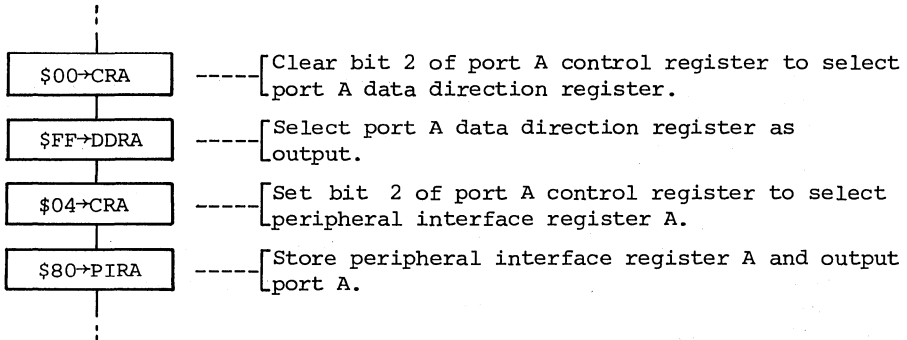


Figure 12-8. Write Operation

b. LCD-II is software reset by the following procedure:

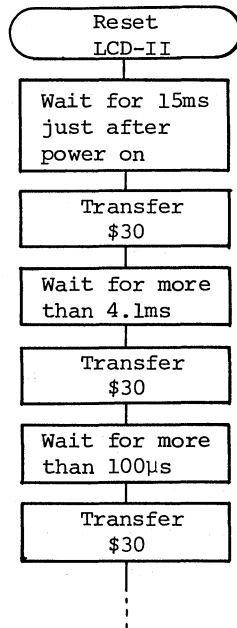


Figure 12-9. Reset LCD-II

Program Module Name: INITIALIZE  
LCD-II

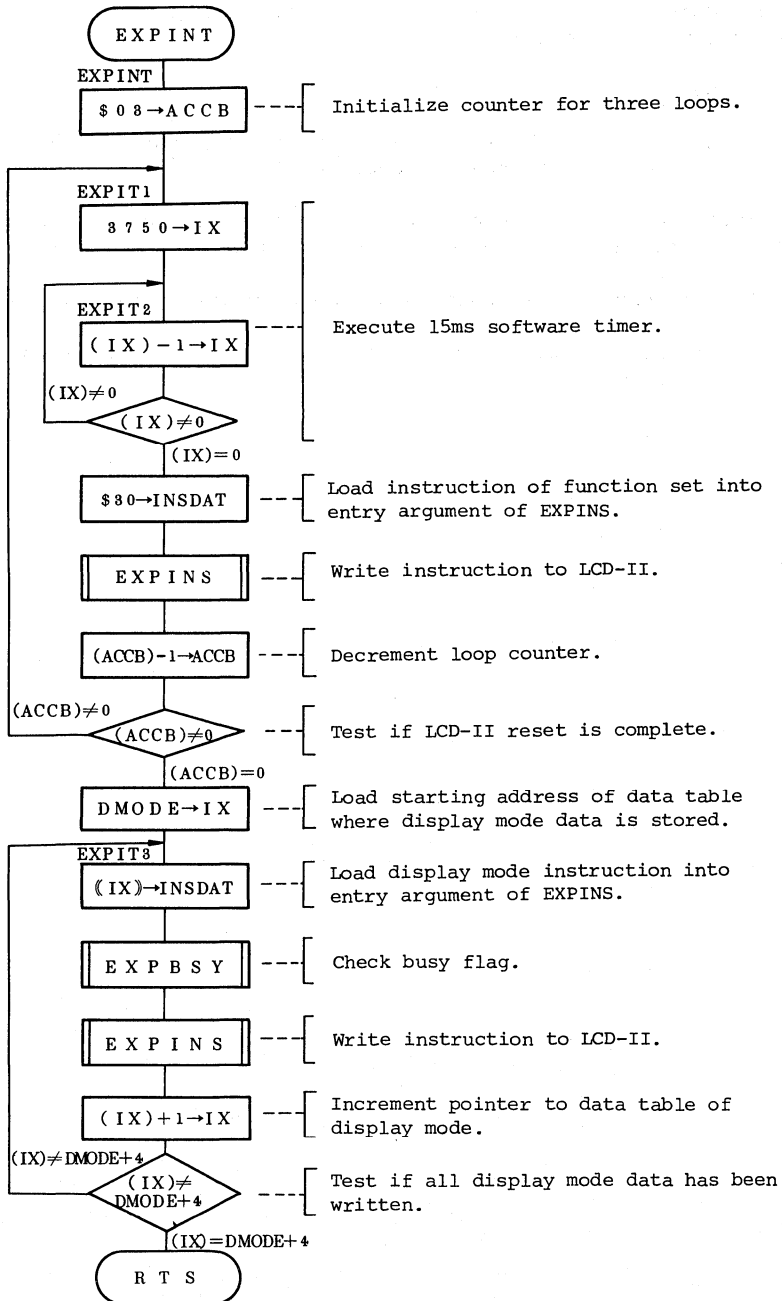
MCU/MPU: HD6301Y0/  
HD6303Y

Label: EXPINT

Description:

- c. Software controls the LCD-II control signal using port A of PIA, and the data bus using port B.
- d. Programming notes
  - i. Both a 15ms wait by software timer and transfer of \$30 to the data bus three times is needed reset LCD-II.
  - ii. Index register is used both as a pointer to the instruction data table and as a counter of the number of data transfers.
  - iii. LCD-II busy flag is checked before outputting instruction data.
  - iv. After process (iii) is executed four times, display mode is initialized.

Flowchart:



Program Module Name: RECEIVE DATA

MCU/MPU: HD6301Y0/  
HD6303Y

Label: EXPINP

Function:

Receives key data from the console typewriter and stores it in RAM.

Arguments:

Contents	Storage Location	No. of Bytes
Entry	_____	_____

Re- turns	Received data (ASCII)	KEYDAT (RAM)	1
	Received flag	KEYDRF (RAM)	1

Changes in CPU

Registers and Flags:

ACCD	
ACCA	ACCB
x	•

IX
•

C	V
x	x

Z	N
x	x

I	H
•	•

• : Not affected  
x : Undefined  
↓ : Result

Specifications:

ROM (Bytes): 26  
RAM (Bytes): 2  
Stack (Bytes): 0  
No. of cycles: 43  
Reentrant: No  
Relocatable: No  
Interrupt OK?: Yes

Description:

1. Function Details

a. Argument details

KEYDAT (RAM): Holds key data from the console typewriter in ASCII.

KEYDRF (RAM): Key data receive Flag. Table 12-7 shows flag functions.

① Entry { Press "a"  
( 'a'=\$61)

a

② Result

KEYDAT	
b7	b0
KEYDAT (RAM) ( 'a'=\$61)	6 1
KEYDRF	
b7	b0
KEYDRF (RAM) \$01	0 1

Figure 12-10. Example of EXPINP Execution

Specifications Notes:

Program Module Name: RECEIVE DATA

MCU/MPU: HD6301Y0/  
HD6303Y

Label: EXPINP

Description:

- b. Example of EXPINP execution is shown in figure 12-10. If "a" is depressed in console typewriter as shown in part ① of figure 12-10, key data is stored in KEYDAT(RAM) and \$01 is set in KEYDRF(RAM).

Table 12-7. Flag Functions

Label	bit 0	Function
KEYDRF	0	Indicates key data is not received.
	1	Indicates key data is received.

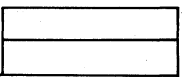
- c. EXPINP calls neither the program modules nor subroutines.

2. User Notes

The following procedure must be executed before EXPINP execution.

- a. Initialize ACIA since ACIA must interface with peripheral device (receives data from the console typewriter).
- b. Initialize RAM/port 5 control register since  $\overline{IRQ_1}$  pin is used.
- c. Clear bit I to enable interrupts since  $\overline{IRQ_1}$  interrupt is used.

3. RAM Allocation

Label	RAM	Description
	b7                      b0	
KEYDAT		} Key data
KEYDRF		
		} Used as a flag indicating whether or not key data is received.

4. Sample Application

```

:
LDAA  #$97      }
STAA  CR        } ----- Master-reset ACIA.
LDAA  #$95      }
STAA  CR        } ----- Initialize control register of ACIA.
BCLR  0,KEYDRF  } ----- Initialize received flag.
LDAA  #$7D      }
STAA  RP5CR     } ----- Initialize RAM/port 5 control register.
CLI                               } ----- Enable interrupts
:

```

Description:

## 5. Basic Operation

- a. Example of initializing ACIA is shown in figure 12-11.

## Note:

For master-reset, "11" is stored in bits 0 and 1 of control register. Bits 5 and 6 must be defined to obtain specified RTS output.

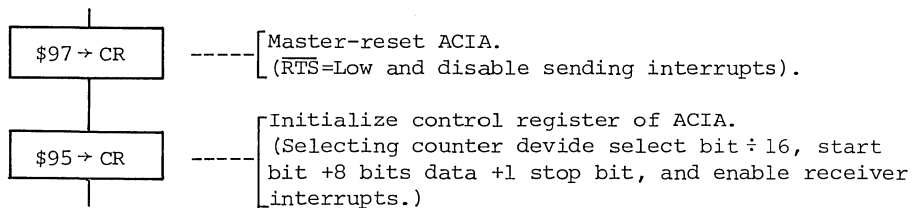
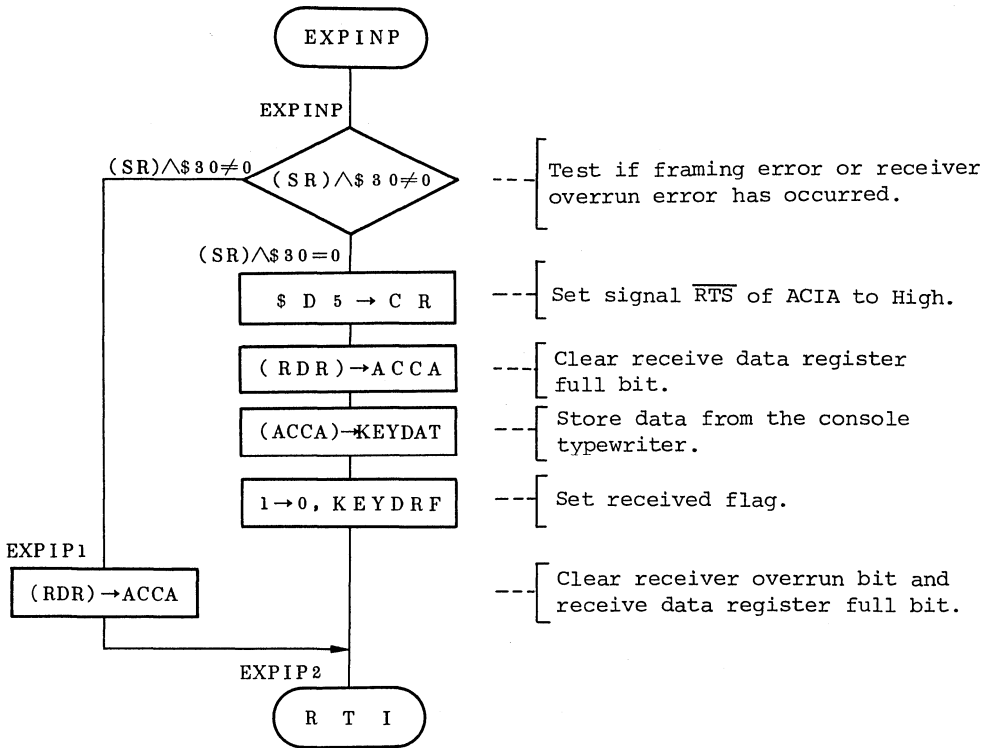


Figure 12-11. Example of Initializing ACIA

## b. Programming notes

- i. Received data is checked for any errors.
- ii. If an error has occurred, received data is ignored.
- iii. If an error has not occurred, received data is stored in KEYDAT (RAM) and received flag is set.
- iv. Receive data register of ACIA is read and interrupts are enabled.

Flowchart:



Program Module Name: DISPLAY  
CHARACTERS

MCU/MPU: HD6301Y0/  
HD6303Y

Label: EXPDSP

Function:

Stores ASCII in display RAM(DDRAM) and displays characters on the LCD.

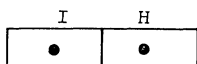
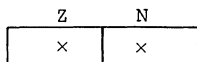
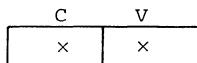
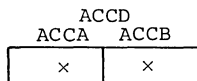
Arguments:

Contents	Storage Location	No. of Bytes
Entry Display data	DSPDAT (RAM)	1

Re-  
turns

Changes in CPU

Registers and Flags:



● : Not affected  
x : Undefined  
↑ : Result

Specifications:

ROM (Bytes): 78  
RAM (Bytes): 1  
Stack (Bytes): 2  
No. of cycles: 106  
Reentrant: No  
Relocatable: No  
Interrupt OK?: Yes

Description:

1. Function Details

a. Argument details

DSPDAT (RAM): Holds display data in 1 byte ASCII.

- b. Example of EXPDSP execution is shown in figure 12-12. If entry argument is as shown in part ① of figure 12-12, a character is displayed on the LCD as shown in part ② of figure 12-12.

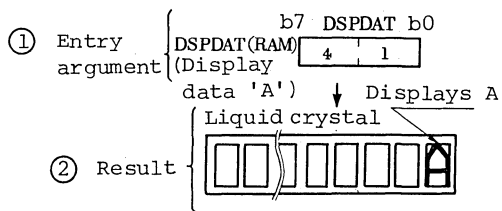


Figure 12-12. Example of EXPDSP Execution

Specifications Notes:

- Values in "Specifications" include those used by subroutines called by EXPDSP.
- "No. of cycles" in "Specifications" indicates the number of cycles required when EXPBSY is executed the minimum number of cycles.



Program Module Name: DISPLAY  
CHARACTERS

MCU/MPU: HD6301Y0/  
HD6303Y

Label: EXPDSP

Description:

c. EXPDSP calls an other subroutine as shown in table 12-8.

Table 12-8. Subroutine Called by EXPDSP

Subroutine Name	Label	Function
CHECK BUSY FLAG	EXPBSY	Checks LCD-II busy flag.

2. User Notes

The following procedure must be executed before EXPDSP execution.

- a. Initialize PIA since PIA must interface with peripheral devices (LCD-II is controlled by ports of PIA).
- b. Initialize LCD-II by executing EXPINT.

3. RAM Allocation

Label	RAM	Description
	b7                      b0	
DSPDAT	<span style="border: 1px solid black; display: inline-block; width: 100px; height: 15px;"></span>	} Display data in ASCII

4. Sample Application

```

|
|
LDAA    #$04    } ----- Select peripheral interface register A of PIA.
STAA    CRA
LDAA    #$02    } ----- Initialize LCD-II control singal.
STAA    PIRA
CLR     A       } ----- Select data direction register A of PIA.
STAA    CRA
LDAA    #$FF    } ----- Select port A of PIA as output.
STAA    DDRA
LDAA    #$04    } ----- Select peripheral interface register A of PIA.
STAA    CRA
CLR     A       } ----- Select data direction register B of PIA.
STAA    CRB
LDAA    #$FF    } ----- Select port B of PIA as output.
STAA    DDRB
LDAA    #$04    } ----- Select peripheral interface register B of PIA.
STAA    CRB
BSR     EXPINT   } ----- Execute EXPINT to initialize LCD-II.
LDA     #$41    } ----- Load entry argument of EXPDSP.
STA     DSPDAT
|
|
JSR     EXPDSP } ----- Execute EXPDSP.
|
|
```

Program Module Name: DISPLAY  
CHARACTERS

MCU/MPU: HD6301Y0/  
HD6303Y

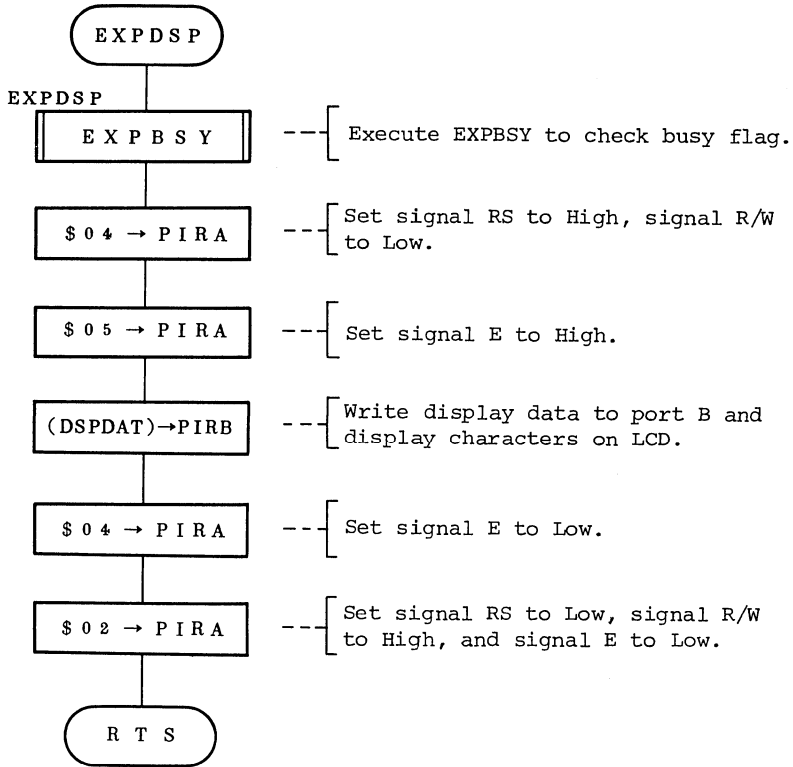
Label: EXPDSP

Description:

5. Basic Operation

- a. LCD-II busy flag is checked by EXPBSY execution.
- b. Control signal of LCD-II is controlled by port A of PIA and display data is output from port B.

Flowchart:



Program Module Name: SEND DATA

MCU/MPU: HD6301Y0/  
HD6303Y

Label: EXPOUT

Function:

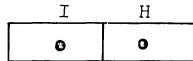
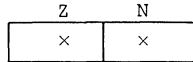
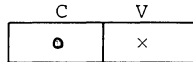
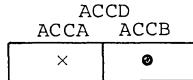
Sends data to the console typewriter.

Arguments:

Contents		Storage Location	No. of Bytes
Entry	Sending data	OUTDAT (RAM)	1
Re-	_____	_____	_____
turns			

Changes in CPU

Registers and Flags:



● : Not affected  
 × : Undefined  
 ↑ : Result

Specifications:

ROM (Bytes): 13  
 RAM (Bytes): 1  
 Stack (Bytes): 0  
 No. of cycles: 21  
 Reentrant: No  
 Relocatable: No  
 Interrupt OK?: Yes

Description:

1. Function Details

a. Argument details

OUTDAT (RAM): Holds data to be sent to the console typewriter in ASCII.

b. Example of EXPOUT execution is shown in figure 12-13. If entry argument is as shown in part ① of figure 12-13, a character is printed as shown in part ② of figure 12-13.

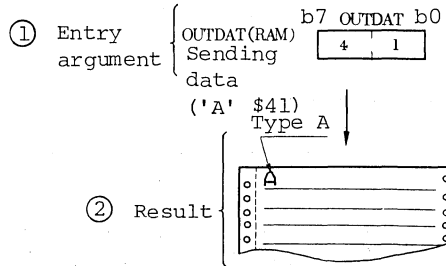


Figure 12-13. Example of EXPOUT Execution

Specifications Notes:

"No. of cycles" in "Specifications" indicates the number of cycles required when the transmit data register is empty.

Description:

c. EXPOUT calls neither the program modules nor subroutines.

2. User Notes

a. The following procedure must be executed before EXPOUT execution.

i. Initialize ACIA since ACIA must interface with peripheral devices (data is sent to the console typewriter).

b. If data has been previously stored in the transmit data register, EXPOUT waits until it is empty so as not to destroy this data.

3. RAM Allocation

Label	RAM	Description
	b7                  b0	
OUTDAT	<input type="text"/>	Character data to be sent to the console typewriter.

4. Sample Application

```

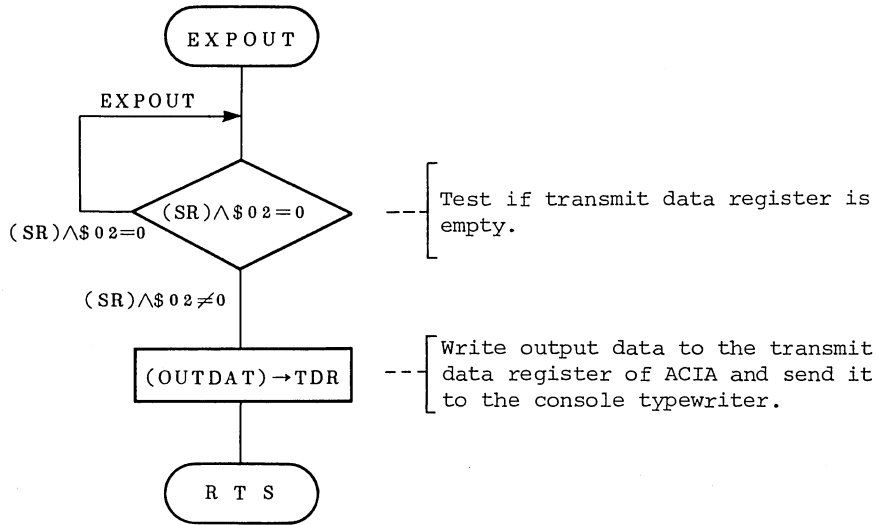
    |
    | LDAA    #$97    } ----- Master-reset ACIA.
    | STAA    CR      }
    | LDAA    #$95    } ----- Initialize control register of ACIA.
    | STAA    CR      }
    | LDAA    #$41    } ----- Load output data into entry argument.
    | STAA    OUTDAT  }
    |
    |  JSR    EXPOUT  } ----- Execute EXPOUT.
    |
    |
  
```

5. Basic Operation

a. Transmit data register empty flag of ACIA is tested for 1 or 0. If it is "1", load output data into the transmit data register of ACIA.

b. Instruction BTST of the HD6301Y0 is replaced by instruction ANDA since instruction BTST cannot be used in extended addressing.

Flowchart:



## 12.4 SUBROUTINE DESCRIPTION

Subroutine Name: STORE INSTRUCTION

MCU/MPU: HD6301Y0/  
HD6303Y

Label: EXPINS

### Function:

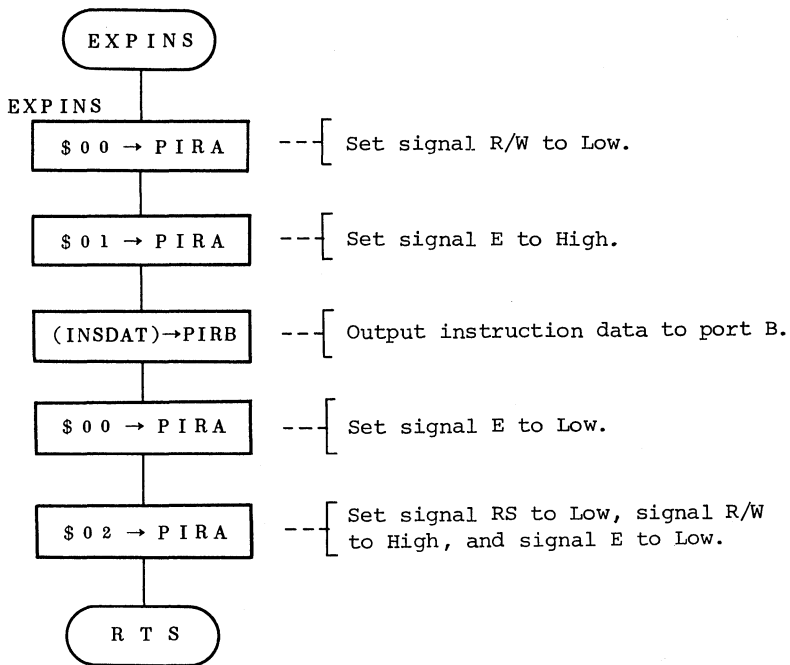
Stores instruction data in LCD-II instruction register.

### Basic Operation:

Signals RS, R/W, and E are controlled by port A of PIA, and instruction data is output from port B.

Program Module Using This Subroutine: EXPINT

### Flowchart:



Subroutine Name: CHECK BUSY FLAG

MCU/MPU: HD6301Y0/  
HD6303Y

Label: EXPBSY

Function:

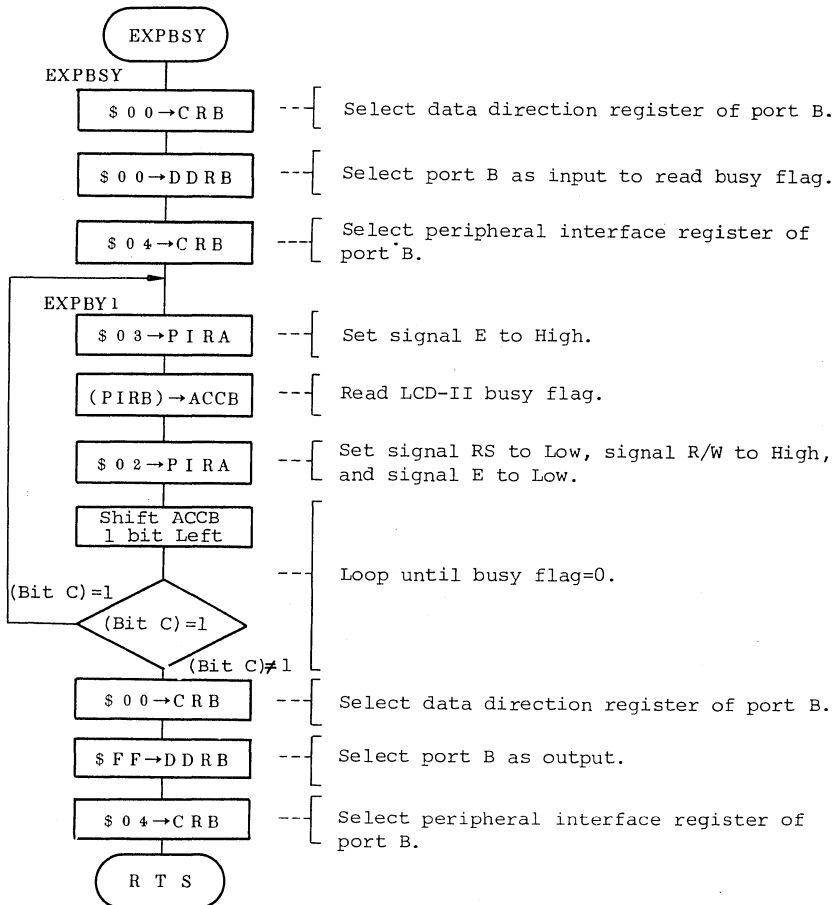
Checks whether or not LCD-II is in operation and waits until it is ready.

Basic Operation:

1. If LCD-II is in operation, the HD6301Y0 cannot access if the LCD-II busy flag indicates whether or not it is in operation.
2. Signals RS, R/W, and E are controlled by port A of PIA, and the busy flag is read by port B.

Program Module Using This Subroutine: EXPINT, EXPDSP

Flowchart:





```

00001          *
00002          **** RAM ALLOCATION *****
00003          *
00004A 0040          ORG      $40
00005          *
00006A 0040 0001 A KEYDRF RMB    1      Existence of receive data
00007          *
00008A 8000          ORG      $8000
00009A 8000 0001 A INSDAT RMB    1      Instruction data
00010A 8001 0001 A OUTDAT RMB    1      Data to be sent
00011A 8002 0001 A DSPDAT RMB    1      Display data
00012A 8003 0001 A KEYDAT RMB    1      Receive data
00013          *
00014          **** SYMBOL DEFINITIONS *****
00015          *
00016          0014 A RPSCR EQU    $14      Port 5 control register
00017          A000 A DDRA  EQU   $A000     Data direction register A(PIA)
00018          A001 A CRA   EQU   $A001     Control register A(PIA)
00019          A002 A DDRB  EQU   $A002     Data direction register B(PIA)
00020          A003 A CRB   EQU   $A003     Control register B(PIA)
00021          A000 A PIRA  EQU   $A000     Peripheral register A(PIA)
00022          A002 A PIRB  EQU   $A002     Peripheral register B(PIA)
00023          C000 A CR    EQU   $C000     Control register(ACIA)
00024          C000 A SR    EQU   $C000     Status register(ACIA)
00025          C001 A RDR   EQU   $C001     Receive data register(ACIA)
00026          C001 A TDR   EQU   $C001     Transmit data register(ACIA)
00027          ****
00028          *
00029          *      MAIN PROGRAM : EXPMN      *
00030          *
00031          ****
00032          *
00033A C000          ORG      $C000
00034          *
00035A C000 8E 013F A EXPMN  LDS    #$13F     Initialize stack pointer
00036A C003 86 04  A        LDAA   #$04     Select peripheral register A
00037A C005 B7 A001 A        STAA   CRA      Set RS=0,R/W=0,E=0
00038A C008 4F          CLRA
00039A C009 B7 A000 A        STAA   PIRA
00040A C00C B7 A001 A        STAA   CRA
00041A C00F 86 FF  A        LDAA   #$FF     Select port A as output
00042A C011 B7 A000 A        STAA   DDRA
00043A C014 86 04  A        LDAA   #$04     Select peripheral register A
00044A C016 B7 A001 A        STAA   CRA
00045A C019 4F          CLRA      Select data direction register B
00046A C01A B7 A003 A        STAA   CRB
00047A C01D 86 FF  A        LDAA   #$FF     Select port B as output
00048A C01F B7 A002 A        STAA   DDRB
00049A C022 86 04  A        LDAA   #$04     Select peripheral register B
00050A C024 B7 A003 A        STAA   CRB
00051A C027 BD COAE A        JSR    EXPINT  Initialize LCD-II
00052A C02A 86 0E  A        LDAA   #$0E
00053A C02C B7 8000 A        STAA   INSDAT  Store LCD display data
00054A C02F 8D 52 C083 BSR    EXPBSY  Check busy flag
00055A C031 8D C0D3 A        JSR    EXPINS  Store instruction
00056A C034 86 97  A        LDAA   #$97     Master-reset ACIA
00057A C036 B7 C000 A        STAA   CR

```

```

00058A C039 86 95 A LDAA H$95 Initialize ACIA
00059A C03B 87 C000 A STAA CR
00060A C03E 71 FE 40 BCLR 0.KEYDRF Initialize key receive flag
00061A C041 86 7D A LDAA H$7D Initialize port 5
00062A C043 97 14 A STAA RPSCR
00063A C045 0E CLI Enable interrupt
00064A C046 7B 01 40 EXPMN1 BTST 0.KEYDRF Test if data is received
00065A C049 27 FB C046 BEQ EXPMN1 Branch if not received
00066A C04B 86 8003 A LDAA KEYDAT Store key data
00067A C04E 71 FE 40 BCLR 0.KEYDRF Clear receive data flag
00068A C051 C6 95 A LDAB H$95 Set RTS=LOW
00069A C053 F7 C000 A STAB CR
00070A C056 BD COFA A JSR TPR Convert ASCII lowercase into uppercase
00071A C059 B7 8001 A STAA OUTDAT Store data to be sent
00072A C05C B7 8002 A STAA DSPDAT Store display data
00073A C05F 8D 05 C066 BSR EXPDSP Display characters
00074A C061 BD COEC A JSR EXPDUT Send data
00075A C064 20 EO C046 BRA EXPMN1
00076 *****
00077 * *
00078 * NAME : EXPDSP (DISPLAY CHARACTERS) *
00079 * *
00080 *****
00081 * *
00082 * ENTRY : DSPDAT (DISPLAY DATA) *
00083 * RETURNS : NOTHING *
00084 * *
00085 *****
00086A C066 8D 1B C083 EXPDSP BSR EXPBSY Check busy flag
00087A C068 86 04 A LDAA H$04 Set RS=1,R/W=0
00088A C06A B7 A000 A STAA PIRA
00089A C06D 86 05 A LDAA H$05 Set E=1
00090A C06F B7 A000 A STAA PIRA
00091A C072 86 8002 A LDAA DSPDAT Output LCD-II data
00092A C075 B7 A002 A STAA PIRB
00093A C078 86 04 A LDAA H$04 Set E=0
00094A C07A B7 A000 A STAA PIRA
00095A C07D 86 02 A LDAA H$02 Set RS=0,R/W=1,E=0
00096A C07F B7 A000 A STAA PIRA
00097A C082 39 RTS
00098 *****
00099 * *
00100 * NAME : EXPBSY (CHECK BUSY FLAG) *
00101 * *
00102 *****
00103A C083 4F EXPBSY CLRA Select data direction register B
00104A C084 B7 A003 A STAA CRB
00105A C087 B7 A002 A STAA DDRB Select port B as input
00106A C08A 86 04 A LDAA H$04
00107A C08C B7 A003 A STAA CRB Select peripheral register B
00108A C08F 86 03 A EXPBY1 LDAA H$03 Set E=1
00109A C091 B7 A000 A STAA PIRA
00110A C094 F6 A002 A LDAB PIRB Read busy flag
00111A C097 86 02 A LDAA H$02 Set E=0
00112A C099 B7 A000 A STAA PIRA
00113A C09C 58 ASLB Set busy flag to bit C
00114A C09D 25 F0 C08F BCS EXPBY1 Loop until busy flag=0

```

```

00115A C09F 4F          CLRA          Select data direction register B
00116A COA0 B7 A003 A   STAA CRB
00117A COA3 86 FF A     LDAA #FFF    Select port B as output
00118A COA5 B7 A002 A   STAA DDRB
00119A COA8 86 04 A     LDAA #04     Select peripheral register B
00120A COAA B7 A003 A   STAA CRB
00121A COAD 39          RTS
00122                *****
00123                *
00124                *      NAME : EXPINT (INITIALIZE LCD-II)      *
00125                *
00126                *****
00127                *
00128                *      ENTRY : FUNC (FUNCTION DATA)          *
00129                *      ENTRY (ENTRY MODE DATA)          *
00130                *      RETURNS : NOTHING                    *
00131                *
00132                *****
00133A COAE C6 03 A     EXPINT LDAB #03    Initialize counter
00134A COB0 CE 0EA6 A   EXPIT1 LDX #3750   Execute softwaer timer
00135A COB3 09          EXPIT2 OEX
00136A COB4 26 FD COB3 BNE EXPIT2
00137A COB6 86 30 A     LDAA #30     Store function data
00138A COB8 B7 8000 A   STAA INSDAT
00139A COBB 8D 16 COD3 BSR EXPINS      Write instruction to LCD-II
00140A COBD 5A          DECB
00141A COBE 26 FO COB0 BNE EXPIT1      Loop until TNCNT=0
00142A COC0 CE C120 A   LDX HDMODE   Load starting ADDR of display mode data tabl
00143A COC3 A6 00 A     EXPIT3 LDAA 0.X    Store instruction data
00144A COC5 B7 8000 A   STAA INSDAT
00145A COC8 8D B9 COB3 BSR EXPBSY      Check busy flag
00146A COCA 8D 07 COD3 BSR EXPINS      Store instruction
00147A COCC 09          DEX
00148A COCD 8C C124 A   CPX HDMODE+4 Loop until IX=HDMODE+4
00149A COD0 26 F1 COC3 BNE EXPIT3
00150A COD2 39          RTS
00151                *****
00152                *
00153                *      NAME : EXPINS (STORE INSTRUCTION)      *
00154                *
00155                *****
00156A COD3 4F          EXPINS CLRA      Set R/W=0
00157A COD4 B7 A000 A   STAA PIRA
00158A COD7 86 01 A     LDAA #01     Set E=1
00159A COD9 B7 A000 A   STAA PIRA
00160A CODC B6 8000 A   LDAA INSDAT  Store instruction
00161A CODF B7 A002 A   STAA PIRB
00162A COE2 4F          CLRA      Set E=0
00163A COE3 B7 A000 A   STAA PIRA
00164A COE6 86 02 A     LDAA #02     Set RS=0.R/W=1.E=0
00165A COE8 B7 A000 A   STAA PIRA
00166A COEB 39          RTS
00167                *****
00168                *
00169                *      NAME : EXPOUT (SEND DATA)          *
00170                *
00171                *****

```

```

00172 *
00173 * ENTRY : OUTDAT (DATA TO BE SENT) *
00174 * RETURNS : NOTHING *
00175 *
00176 *****
00177A COEC 86 02 A EXPDUT LDAA H$02 Test if TDRE=1
00178A COEE B4 C000 A ANDA SR
00179A COF1 27 F9 COEC BEQ EXPDUT Loop until TDRE=0
00180A COF3 B6 8001 A LDAA OUTDAT Store send data
00181A COF6 B7 C001 A STAA TOR
00182A COF9 39 RTS
00183 *****
00184 *
00185 * NAME : TPR (CONVERT ASCII LOWERCASE *
00186 * INTO UPPERCASE) *
00187 *
00188 *****
00189 *
00190 * ENTRY : ACCA (ASCII LOWERCASE) *
00191 * RETURNS : ACCA (ASCII UPPERCASE) *
00192 *
00193 *****
00194A COFA 81 61 A TPR CMPA H'a ACCA - 'a' ?
00195A COFC 25 06 C104 BCS TPR1 Branch if ACCA < 'a'
00196A COFE 81 7A A CMPA H'z ACCA - 'z' ?
00197A C100 22 02 C104 BHI TPR1 Branch if ACCA > 'z'
00198A C102 84 DF A ANDA H$DF Convert Lowercase into uppercase
00199A C104 39 TPR1 RTS
00200 *****
00201 *
00202 * NAME : EXPINP (RECEIVE DATA) *
00203 *
00204 *****
00205 *
00206 * ENTRY : NOTHING *
00207 * RETURNS : KEYDAT (RECEIVED DATA) *
00208 * KEYDRF (RECEIVED FLAG) *
00209 *
00210 *****
00211A C105 86 30 A EXPINP LDAA H$30 Test if RDRF=1?
00212A C107 B4 C000 A ANDA SR
00213A C10A 26 0F C11B BNE EXPIP1 Branch if RDRF=0
00214A C10C 86 05 A LDAA H$05 Set RTS=High
00215A C10E B7 C000 A STAA CR
00216A C111 B6 C001 A LDAA RDR Read received data
00217A C114 B7 8003 A STAA KEYDAT Store receive data
00218A C117 72 01 40 BSET 0.KEYDRF
00219A C11A 3B EXPIP2 RTI
00220A C11B B6 C001 A EXPIP1 LDAA RDR Enable interrupts
00221A C11E 20 FA C11A BRA EXPIP2
00222 *****
00223 *
00224 * DATA TABLE *
00225 *
00226 *****
00227A C120 30 A DMODE FCB $30,$08,$01,$06
00228 *****

```

```

00229          *
00230          *          VECTOR ADDRESSES          *
00231          *
00232          *
00233          *
00234A FFEA          *          ORG          $FFEA          *
00235          *
00236A FFEA          C000 A          FDB          EXPMN          IRQ2
00237A FFEC          C000 A          FDB          EXPMN          CMI
00238A FFEE          C000 A          FDB          EXPMN          TRAP
00239A FFF0          C000 A          FDB          EXPMN          SIO
00240A FFF2          C000 A          FDB          EXPMN          TOI
00241A FFF4          C000 A          FDB          EXPMN          OCI
00242A FFF6          C000 A          FDB          EXPMN          ICI
00243A FFF8          C105 A          FDB          EXPINP          IRQ1/ISF
00244A FFFA          C000 A          FDB          EXPMN          SWI
00245A FFFC          C000 A          FDB          EXPMN          NMI
00246A FFFE          C000 A          FDB          EXPMN          RES
00247          *
00248          *          END

```

13.1 HARDWARE DESCRIPTION

13.1.1 Function

Compares a check sum, obtained from a data block in external memory, with a check sum stored beforehand in the data block and indicates whether or not they are the same using an LED.

13.1.2 Microcomputer Operation

The HD63B01Y0 accesses slow devices using the auto-memory-ready function. Signal  $\overline{CS}_1$  of the slow devices is input directly to the memory-ready pin by setting both the enable bit of memory-ready and the enable bit of auto-memory-ready to "1".

13.1.3 Peripheral Devices

HN482764G-3 EPROM: Used as an external memory. Its maximum access time is 300ms.

13.1.4 Circuit Diagram

Memory-ready circuit is shown in figure 13-1.

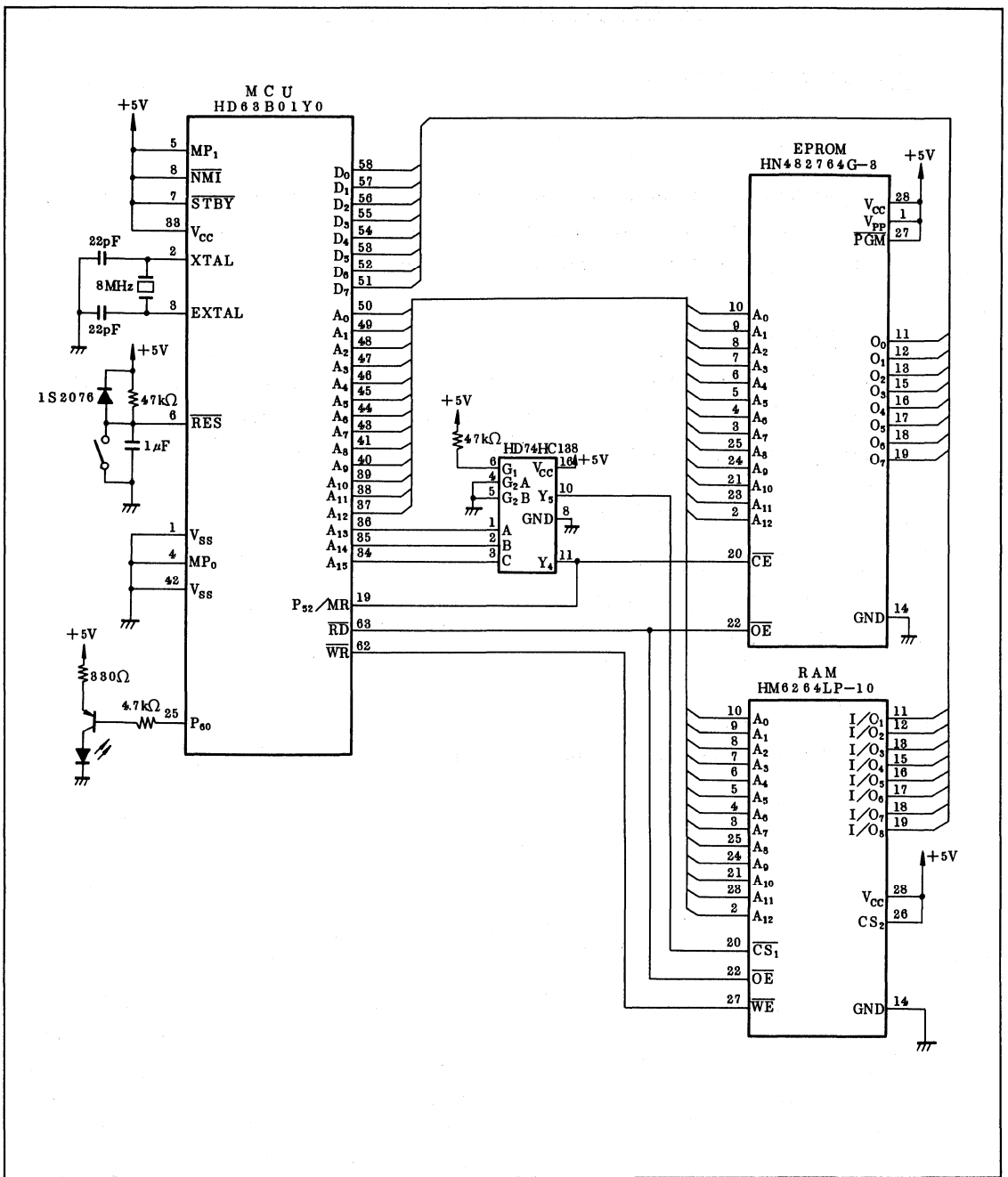


Figure 13-1. Memory-ready Circuit

### 13.1.5 Pin Functions

Pin functions for control of LED connection pins and slow device pins are shown in table 13-1.

Table 13-1. Pin Functions

Pin Name (HD63B01Y0)	Input/ Output	Active Level (High or Low)	Function	Program Label
P60	Output	Low	Turns on LED.	P6DTR
P52/MR	Input	Low	Memory-ready pin.	—

### 13.1.6 Memory Map

Address decoding for this application example is shown in table 13-2. A slow memory device, the HN482764G-3, is allocated as external memory by the HD74HC138 address decoder. Address buses A<sub>13</sub>, A<sub>14</sub> and A<sub>15</sub> are connected to A, B and C pins, respectively, of the HD74HC138. Address space \$8000 ~ \$FFFF is divided into 8k-byte sections.

Table 13-2. Address Decoding

HD74HC138						Address								Allocation		
Input			Output													
G <sub>1</sub>	G <sub>2A</sub>	G <sub>2B</sub>	C	B	A	Y <sub>0</sub>	Y <sub>1</sub>	Y <sub>2</sub>	Y <sub>3</sub>	Y <sub>4</sub>	Y <sub>5</sub>	Y <sub>6</sub>	Y <sub>7</sub>			
			A <sub>15</sub>	A <sub>14</sub>	A <sub>13</sub>											
H	L	L	H	L	L	H	H	H	H	L	H	H	H	8000 ~ 9FFF	EPROM	
H	L	L	H	L	H	H	H	H	H	H	L	H	H	A000 ~ BFFF	Not Used	
H	L	L	H	H	L	H	H	H	H	H	L	H	H	C000 ~ DFFF	Internal ROM	
H	L	L	H	H	H	H	H	H	H	H	H	L	H	E000 ~ FFFF	Internal ROM	



Memory map for this application example is shown in figure 13-2.

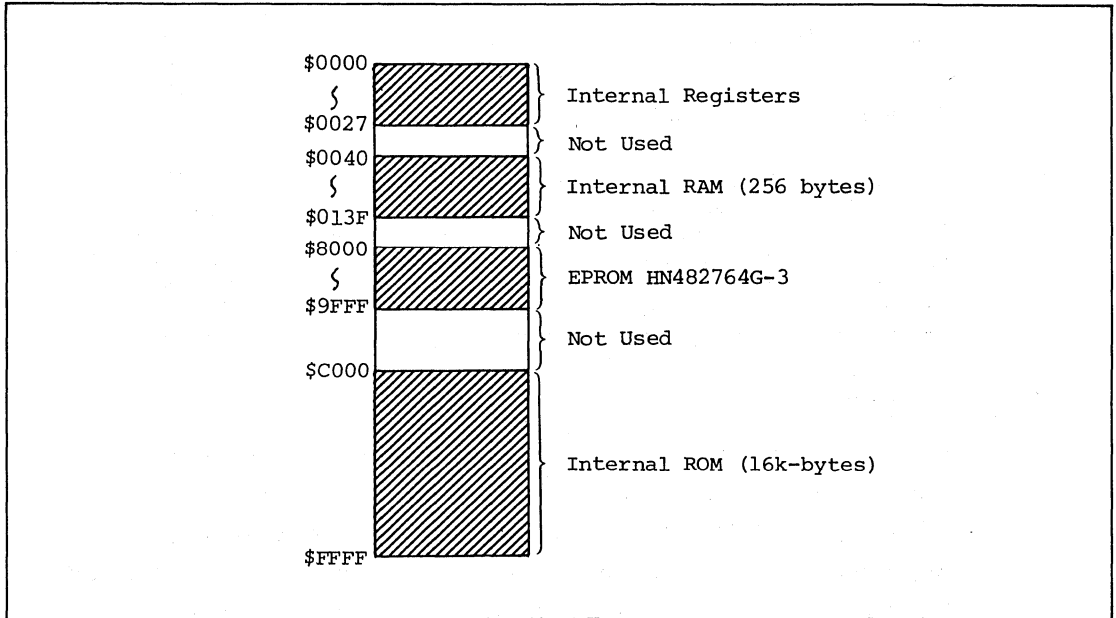


Figure 13-2. Memory Map

### 13.1.7 Hardware Operation

Timing chart for the HD63B01Y0 and slow device HN482764G-3 is shown in figure 13-3.

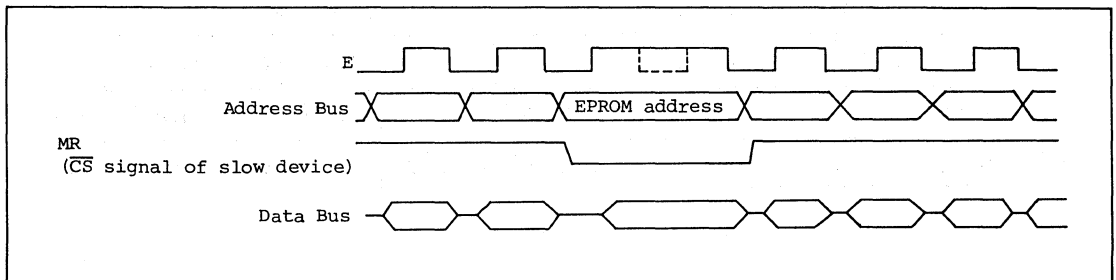


Figure 13-3. Memory-Ready Bus Timing

## 13.2 SOFTWARE DESCRIPTION

### 13.2.1 Program Module Configuration

The program module configuration for determining check sum is shown in figure 13-4.

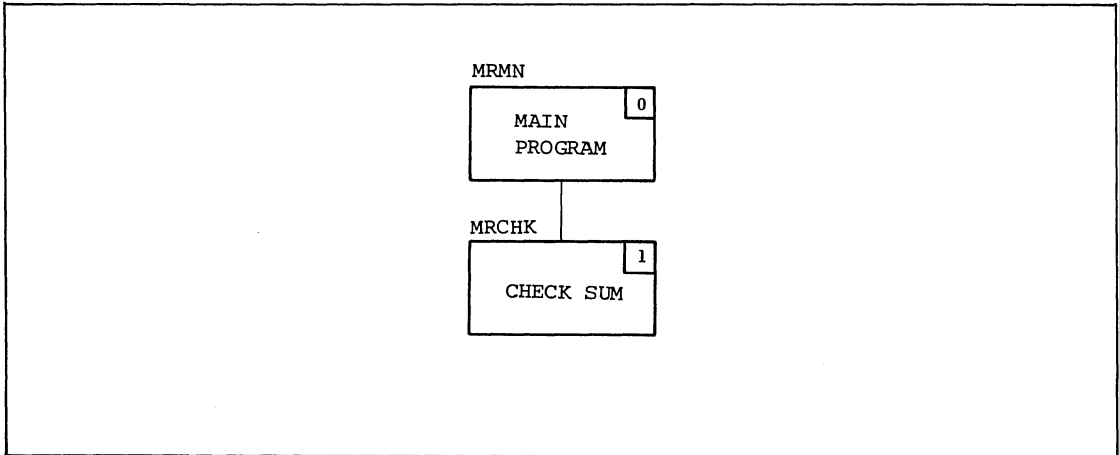


Figure 13-4. Program Module Configuration

### 13.2.2 Program Module Functions

Program module functions are summarized in table 13-3.

Table 13-3. Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	MRMN	Displays result of comparing check sums.
1	CHECK SUM	MRCHK	Obtains check sum of data block in external memory.

### 13.2.3 Program Module Process Flow (Main Program)

The flowchart in figure 13-5 shows the procedure for testing check sums as performed by the program module in figure 13-4.

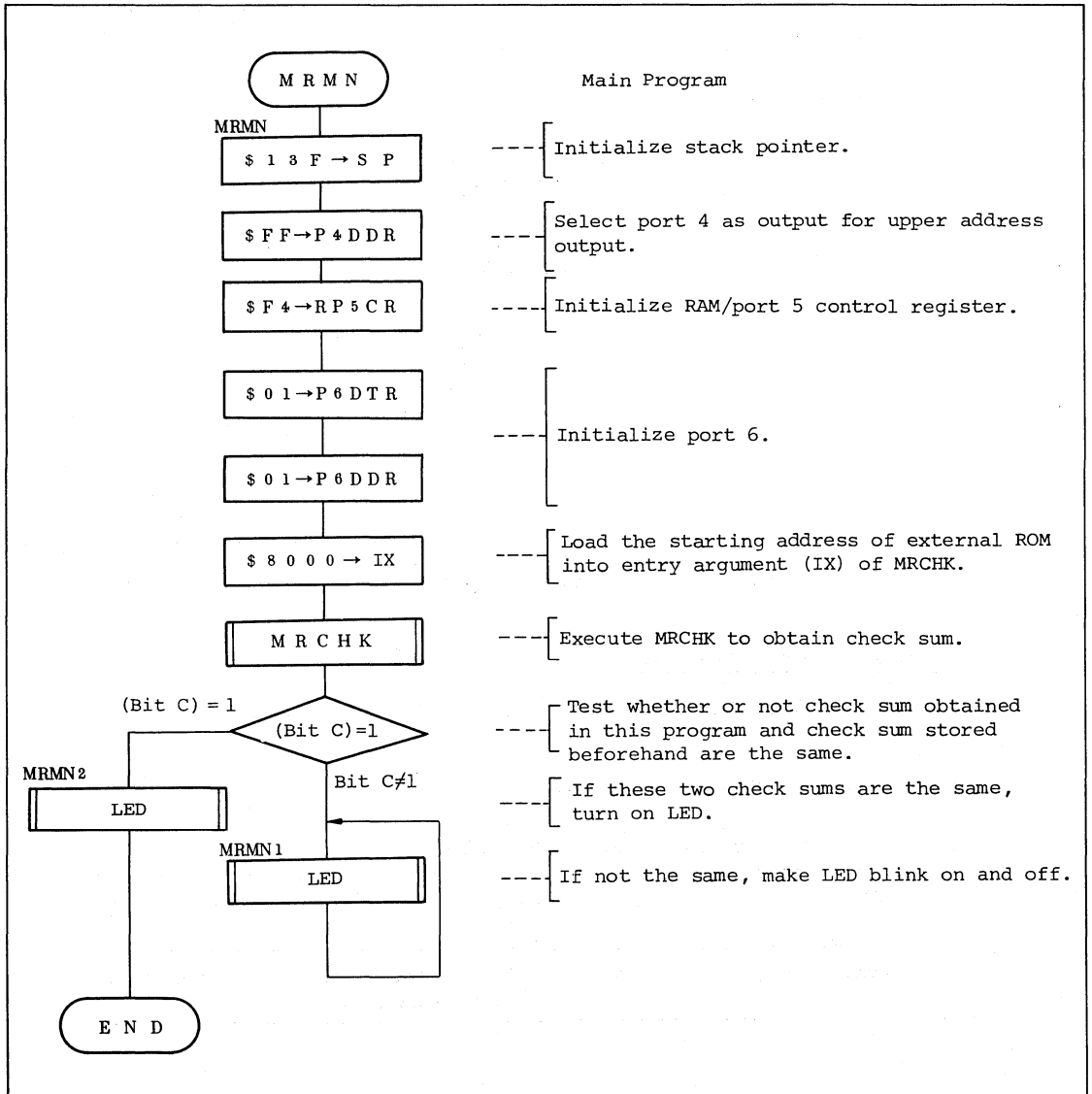


Figure 13-5. Program Module Flowchart

### 13.3 PROGRAM MODULE DESCRIPTION

Program Module Name: CHECK SUM

MCU/MPU: HD6301Y0/  
HD6303Y

Label: MRCHK

Function:

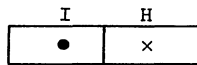
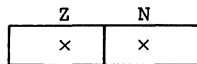
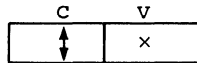
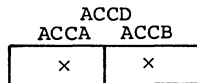
Obtains check sum of data block and compares check sum obtained with check sum stored beforehand.

Arguments:

Contents	Storage Location	No. of Bytes
Entry Data table starting address	IX	2
Re-turns Check sum result indicator	bit C (CCR)	1 bit

Changes in CPU

Registers and Flags:



● : Not Affected  
× : Undefined  
↑ : Result

Specifications:

ROM (Bytes): 23  
RAM (Bytes): 2  
Stack (Bytes): 0  
No. of cycles: 137  
Reentrant: No  
Relocatable: No  
Interrupt OK?: Yes

Description:

1. Function Details

a. Argument details

IX: Holds the starting address of data table:

bit C (CCR): Used as a flag indicating whether or not contents of external memory are correct, determined by check sum result. Table 13-4 shows flag functions.

b. Example of MRCHK execution is shown in figure 13-6. If entry argument is as shown in part ① of figure 13-6, bit C is set to "1" as shown in part ② of figure 13-6 when the two check sums are the same.

Specifications Notes:

"No. of cycles" in "Specifications" indicates the number of cycles required to obtain a check sum for 9 bytes.

Description:

Table 13-4 Flag Functions

Register	bit C	Function
CCR	0	Indicates contents of external memory is not correct.
	1	Indicates contents of external memory is correct.

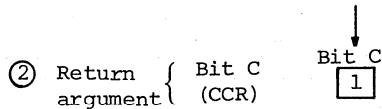
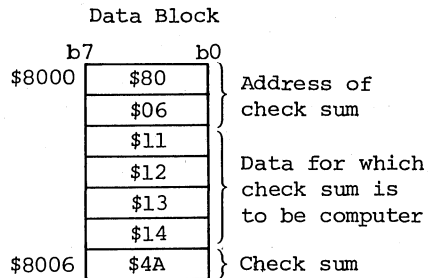
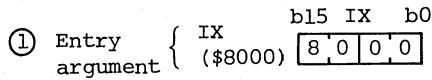


Figure 13-6. Example of MRCHK Execution

c. MRCHK calls neither the program modules nor subroutines.

Program Module Name: CHECK SUM

MCU/MPU: HD6301Y0/  
HD6303Y

Label: MRCHK

Description:

2. User Notes

- a. Initialize RAM/port 5 control register since the auto-memory-ready function is used.
- b. To check the contents of external memory, data, shown in figure 13-7, must be stored in memory.

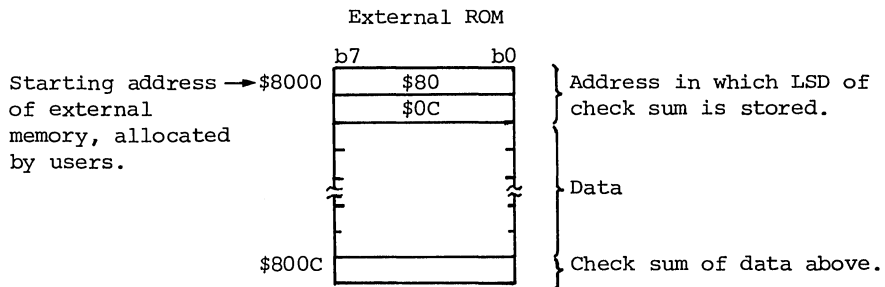
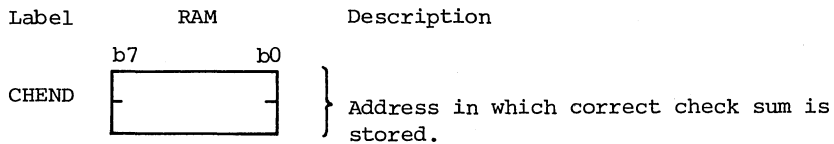


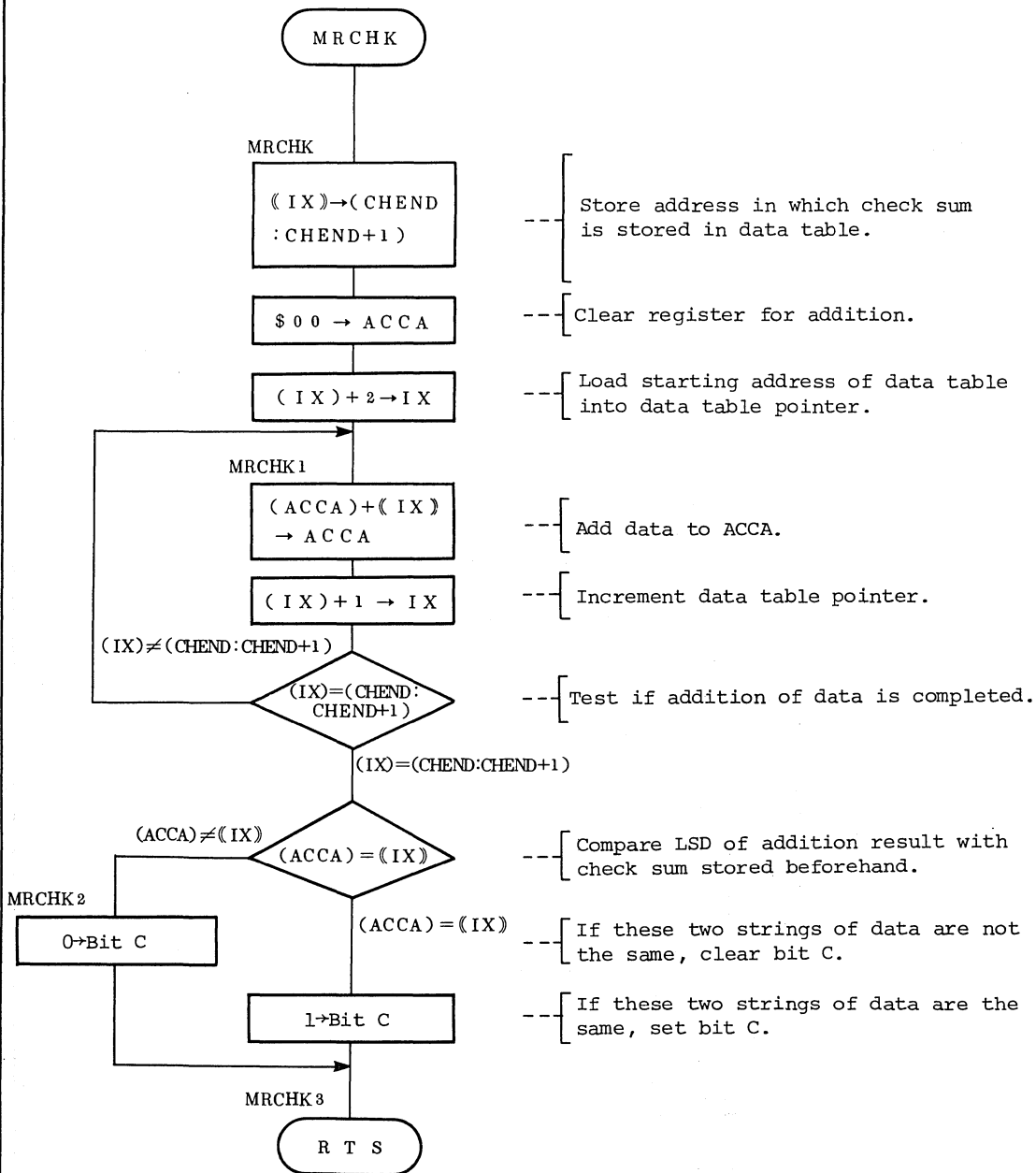
Figure 13-7. External ROM

3. RAM Allocation





Flowchart:





### 13.4 SUBROUTINE DESCRIPTION

Subroutine Name: DISPLAY LED

MCU/MPU: HD6301Y0/  
HD6303Y

Label: MRLED

Function:

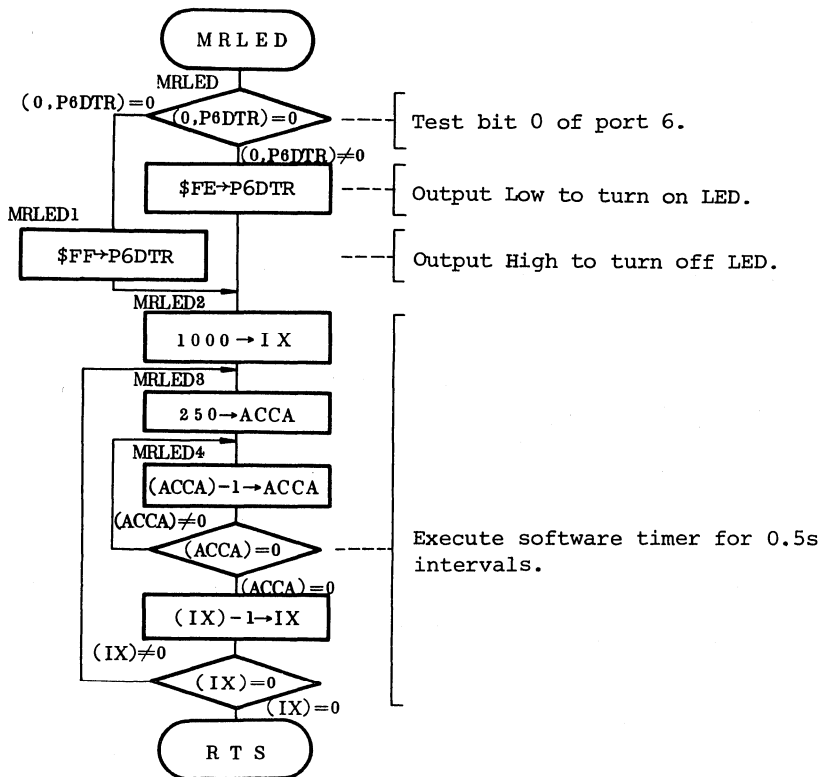
Turns LED on and off, and executes software timer for 0.5s intervals (blinking LED)

Basic Operation:

1. Output condition of P60 is tested.  
If Low, output High to turn off LED.  
If High, output Low to turn on LED.
2. Software timer for 0.5s intervals is executed.

Program Module Using This Subroutine: —

Flowchart:



13.5 PROGRAM LISTING

```

00001      *
00002      ***** RAM ALLOCATION *****
00003      *
00004A 0040      ORG      $40
00005      *
00006A 0040      0002 A CHEND RMB      2      Address of check sum
00007      *
00008      ***** SMYBOL DEFINITIONS *****
00009      *
00010      0006 A P3DTR EQU      $06      Port 3 data register
00011      0005 A P4DDR EQU      $05      Port 4 data direction register
00012      0014 A RP5CR EQU      $14      RAM/port5 control register
00013      0017 A P6DTR EQU      $17      Port 6 data register
00014      0016 A P6DDR EQU      $16      Port 6 data direction register
00015      *
00016      *****
00017      *
00018      *      MAIN PROGRAM : MRMN      *
00019      *
00020      *****
00021      *
00022A C000      ORG      $C000
00023      *
00024A C000 8E 013F A MRMN  LDS      #$13F      Initialize stack pointer
00025A C003 86 FF  A      LDAA     #$FF      Initialize port4 DDR
00026A C005 97 05  A      STAA     P4DDR
00027A C007 86 F4  A      LDAA     #$F4      Initialize RAM/port5 control REG
00028A C009 97 14  A      STAA     RP5CR
00029A C00B 86 01  A      LDAA     #01      Initialize port6
00030A C00D 97 17  A      STAA     P6DTR
00031A C00F 97 16  A      STAA     P6DDR
00032A C011 CE 8000 A      LDX     #$8000      Load strating ADDR of data table
00033A C014 8D 0A C020 BSR     MRCHK      Execute MRCHK
00034A C016 25 04 C01C BCS     MRMN2      Branch if bit C = 1
00035A C018 8D 1D C037 MRMN1 BSR     MRLED      Execute LED
00036A C01A 20 FC C01B BRA     MRMN1      Branch MRMN1
00037A C01C 8D 19 C037 MRMN2 BSR     MRLED      Execute LED
00038A C01E 20 FE C01E BRA     PEND      End of program
00039      *
00040      *****
00041      *
00042      *      NAME : CHECK SUM (MRCHK)      *
00043      *
00044      *****
00045      *
00046      *      ENTRY   : IX (DATA TABLE STARTING ADDR) *
00047      *      RETURNS : BIT C (C=1:TRUE,C=0:FALSE) *
00048      *
00049      *****
00050A C020 EC 00  A MRCHK  LODD     0,X      Store correct check sum data ADDR
00051A C022 DD 40  A      STD     CHEND
00052A C024 4F      CLRA     CLRA      Clear register for addition
00053A C025 08      INX     INX      Load data table starting address
00054A C026 08      INX     INX
00055A C027 AB 00  A MRCHK1 ADDA     0,X      Add data to ACCA
00056A C029 08      INX     INX      Increment data table pointer
00057A C02A 9C 40  A      CPX     CHEND      Test if addition is completed

```

```

00058A C02C 26 F9 C027      BNE      MRCHK1      Branch if not equal
00059A C02E A1 00      A      CMPA      0.X      Compare data with check sum
00060A C030 26 02 C034      BNE      MRCHK2      Branch if not equal
00061A C032 0D              SEC              Set carry
00062A C033 39              MRCHK3 RTS
00063A C034 0C              MRCHK2 CLC              Clear carry
00064A C035 20 FC C033      BRA      MRCHK3
00065      *
00066      *****
00067      *
00068      *          NAME : MRLED (DISPLAY LED)          *
00069      *
00070      *****
00071A C037 7B 01 17      MRLED BTST      0.P6DTR      Test if LED on or off
00072A C03A 27 05 C041      BEQ      MRLED1      Branch if on
00073A C03C 4F              CLRA              Load data to turn on LED
00074A C03D 97 17      A      STAA      P6DTR
00075A C03F 20 04 C045      BRA      MRLED2      Branch to LED2
00076A C041 86 01      A      MRLED1 LDAA      #$01      Load data to turn off LED
00077A C043 97 17      A      STAA      P6DTR
00078A C045 CE 03EB      A      MRLED2 LDX      #1000      Execute software timer for 0.5s
00079A C048 86 FA      A      MRLED3 LDAA      #250
00080A C04A 4A              MRLED4 DECA
00081A C04B 26 FD C04A      BNE      MRLED4
00082A C04D 09              DEX
00083A C04E 26 F8 C048      BNE      MRLED3
00084A C050 39              RTS
00085      *
00086      *****
00087      *
00088      *          VECTOR ADDRESSES          *
00089      *
00090      *****
00091      *
00092A FFEA              ORG      $FFEA
00093      *
00094A FFEA      C000      A      FDB      MRMN      IRQ2
00095A FFEC      C000      A      FDB      MRMN      CMI
00096A FFEE      C000      A      FDB      MRMN      TRAP
00097A FFF0      C000      A      FDB      MRMN      SID
00098A FFF2      C000      A      FDB      MRMN      TOI
00099A FFF4      C000      A      FDB      MRMN      OCI
00100A FFF6      C000      A      FDB      MRMN      ICI
00101A FFF8      C000      A      FDB      MRMN      IRQ1/ISF
00102A FFFA      C000      A      FDB      MRMN      SWI
00103A FFFC      C000      A      FDB      MRMN      NMI
00104A FFFE      C000      A      FDB      MRMN      RES
00105      *
00106      END

```

14.1 HARDWARE DESCRIPTION14.1.1 Function

Executes a test program for two low power dissipation mode: sleep mode and standby mode, and displays current mode and an 8-bit counter on LEDs.

14.1.2 Microcomputer Operation

The HD6301Y0 executes standby mode by clearing standby flag of RAM/port 5 control register during  $\overline{\text{NMI}}$  interrupt routine; and executes sleep mode by instruction SLP execution corresponding to switch input. The MCU returns from standby mode by reset; returns from sleep mode by timer 2 interrupt.

14.1.3 Peripheral Devices

Switches and LEDs: SW1 and SW2 are used to execute sleep mode and standby mode, respectively, and LED1-LED3 indicate the current operating state of the HD6301Y0. Switch and LED setting for each mode are shown in table 14-1. In addition, LED4-LED11 are used to display an 8-bit counter during sleep mode.

Table 14-1. Switch Setting and Display for Each Mode

Test Mode	Switch Setting		Display		
	SW1	SW2	LED1	LED2	LED3
Active Mode	OFF	OFF	OFF	OFF	ON
Sleep Mode	ON	OFF	OFF	ON	OFF
Standby Mode	OFF	ON	ON	OFF	ON (Note)

## Note:

In standby mode, LED1 is displayed after returned from standby mode.

### 14.1.4 Circuit Diagram

Low power dissipation mode circuit is shown in figure 14-1.

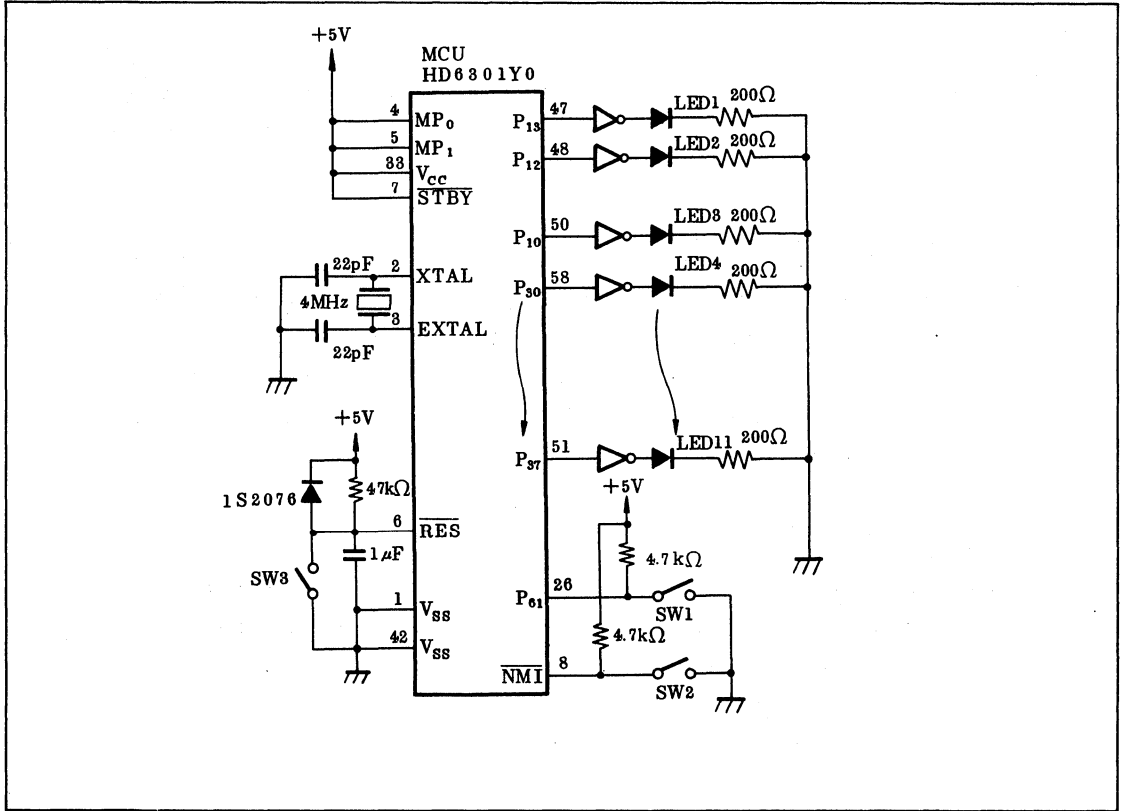


Figure 14-1. Low Power Dissipation Mode Circuit

### 14.1.5 Pin Functions

Pin functions at the interface between the HD6301Y0 and the switches and LEDs are shown in table 14-1.

Table 14-1. Pin Functions

Pin Name (HD6301Y0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (SW, LED)	Program Label
P <sub>10</sub>	Output	High	Drives LED indicating active mode operation.	LED3	P1DTR
P <sub>12</sub>	Output	High	Drives LED after exiting standby mode.	LED2	
P <sub>13</sub>	Output	High	Drives LED indicating sleep mode.	LED1	
P <sub>30</sub>	Output	High	Drives LEDs used as 8-bit binary counter.	LED4	P3DTR
P <sub>31</sub>	Output	High		LED5	
P <sub>32</sub>	Output	High		LED6	
P <sub>33</sub>	Output	High		LED7	
P <sub>34</sub>	Output	High		LED8	
P <sub>35</sub>	Output	High		LED9	
P <sub>36</sub>	Output	High		LED10	
P <sub>37</sub>	Output	High		LED11	
P <sub>61</sub>	Input	Low	Sleep mode switch input	SW1	P6DTR
$\overline{\text{NMI}}$	Input	Low	Standby mode switch input	SW2	—
$\overline{\text{RES}}$	Input	Low	Standby mode reset input	SW3	—

## 14.1.6 Hardware Operation

### a. Standby mode

The timing chart for entering and exiting standby mode by the STBY flag is shown in figure 14-2.

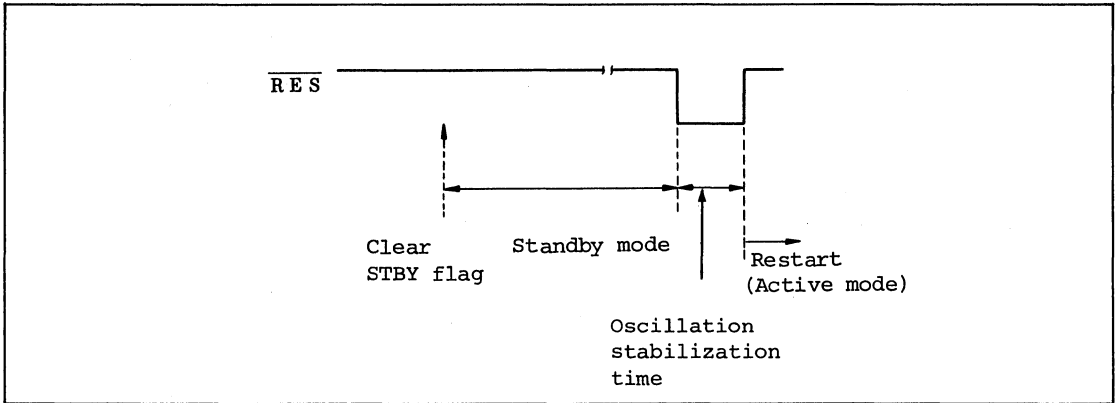


Figure 14-2. Timing Chart for Standby Mode

### b. Sleep mode

The timing chart for sleep mode is shown in figure 14-3.

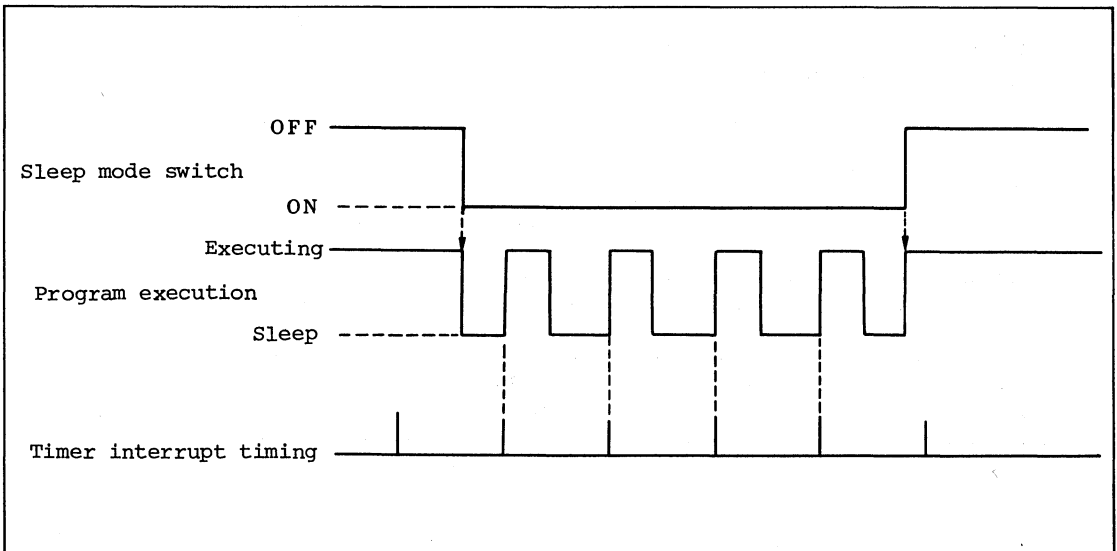


Figure 14-3. Timing Chart for Sleep Mode

14.2.1 Program Module Configuration

The program module configuration for low power dissipation mode is shown in figure 14-4.

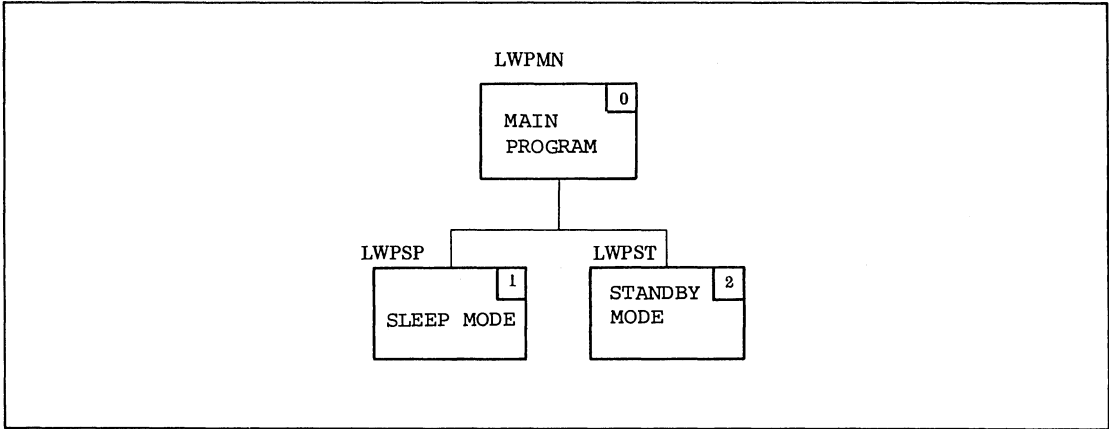


Figure 14-4. Program Module Configuration

14.2.2 Program Module Functions

Program module functions are summarized in table 14-2.

Table 14-2. Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	LWPMN	Executes low power dissipation mode.
1	SLEEP MODE	LWSP	Tests sleep mode operation.
2	STANDBY MODE	LWPST	Tests standby mode operation.



### 14.2.3 Program Module Process Flow (Main Program)

The flowchart in figure 14-5 shows the procedure for performing low power dissipation mode, using the program module in figure 14-4.

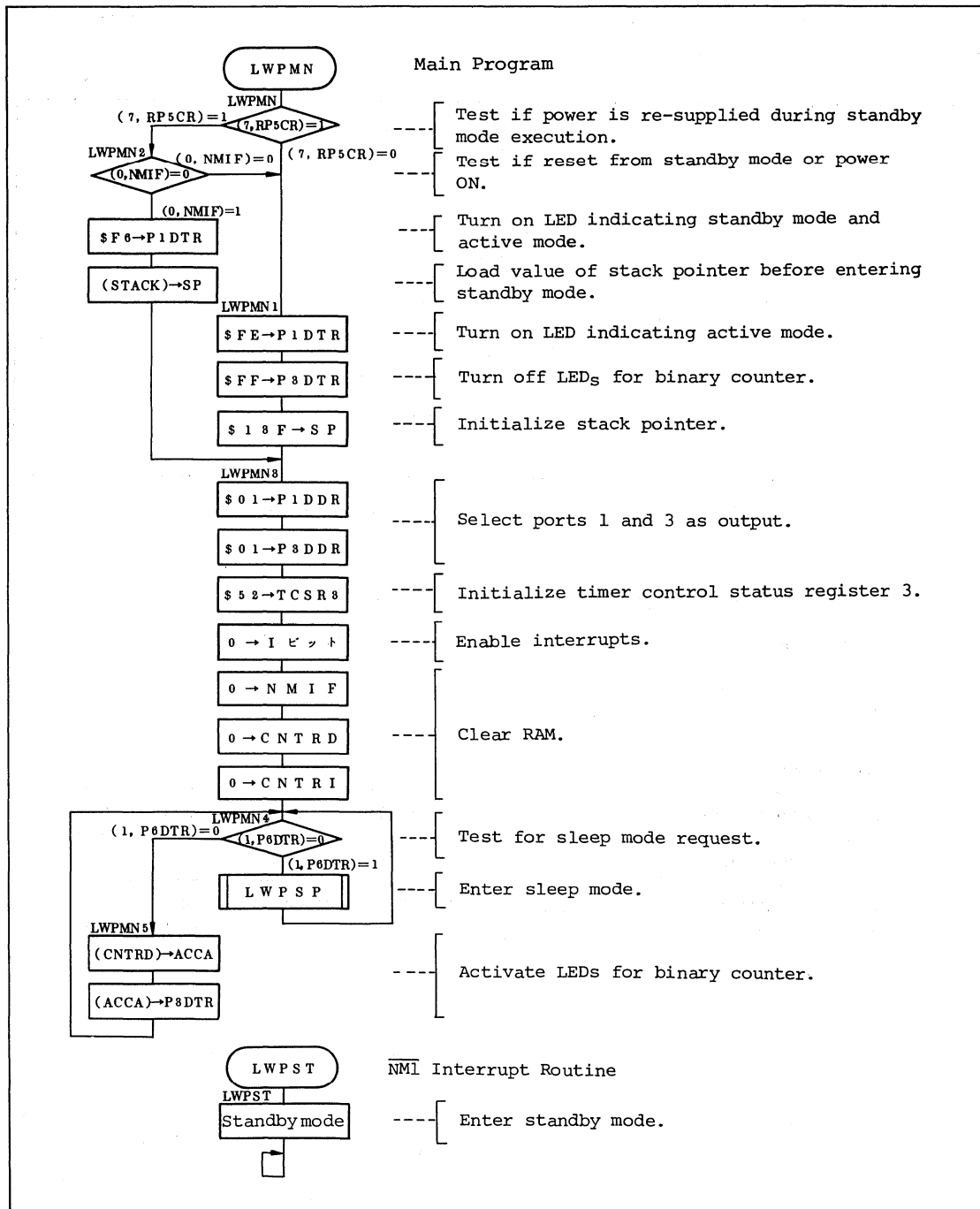


Figure 14-5. Program Module Flowchart

14.3 PROGRAM MODULE DESCRIPTION

Program Module Name: SLEEP MODE

MCU/MPU: HD6301Y0

Label: LWSPSP

Function:

Tests sleep mode operation.

Arguments:

None

Changes in CPU

Registers and Flags:

ACCD	
ACCA	ACCB
×	●

IX	
●	

C	V
●	×

Z	N
×	×

I	H
●	●

● : Not affected  
 × : Undefined  
 ↓ : Result

Specifications:

ROM (Bytes): 10

RAM (Bytes): 0

Stack (Bytes): 0

No. of cycles: 19

Reentrant: No

Relocatable: No

Interrupt OK?: Yes

Description:

1. Function Details

- a. LWSPSP has no arguments.
- b. Sleep mode is entered by switch 1 input.
- c. LWSPSP calls neither the program modules nor subroutines.

2. User Note

The following procedure must be executed before LWSPSP execution.

- a. Select DDR of port 1 as output.

Specifications Notes:

N/A

Description:

## 3. RAM Allocation

RAM is not used during LWSPS execution.

## 4. Sample Application

```

      |
      |
      |
LDAA  # $FF }
STAA  P1DTR } ----- Initialize port 1.
LDAA  # $01 }
STAA  P1DDR }

```

```

┌─── JSR    LWSPS ───┐ ----- Call LWSPS.

```

```

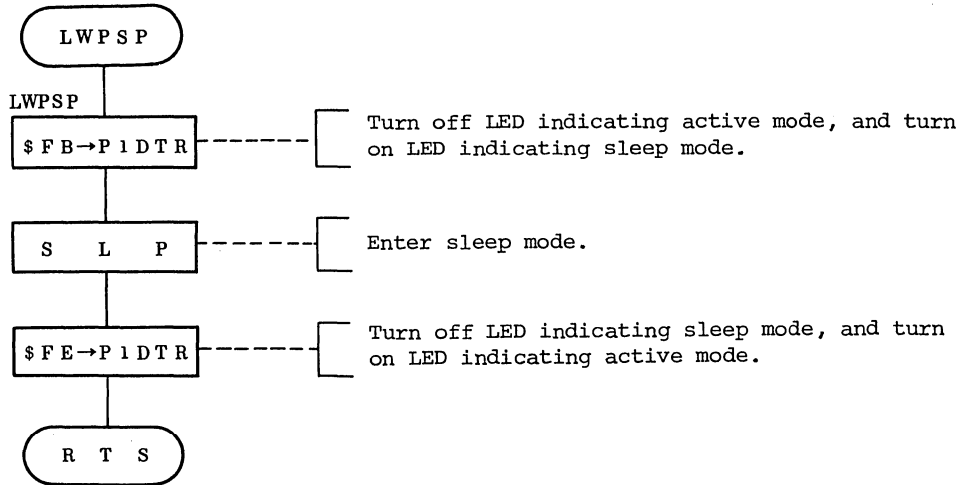
      |
      |
      |

```

## 5. Basic Operation

- a. The LED indicating sleep mode is turned on and sleep mode is entered by the sleep instruction(SLP).
- b. Timer 2 interrupt executes return from sleep mode, and the LED indicating sleep mode is turned off.

Flowchart:



Program Module Name: STANDBY MODE

MCU/MPU: HD6301Y0

Label: LWPST

Function:

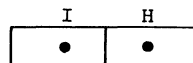
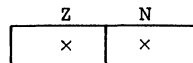
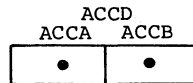
Tests standby mode operation.

Arguments:

Contents	Storage Location	No. of Bytes
Entry	---	---
Re-turns	Value of stack pointer	STACK (RAM) 2
	LWPST execution flag	NMIF (RAM) 1

Changes in CPU

Registers and Flags:



● : Not affected  
x : Undefined  
↑ : Result

Specifications:

ROM (Bytes): 16  
RAM (Bytes): 0  
Stack (Bytes): 0  
No. of cycles: 31  
Reentrant: No  
Relocatable: No  
Interrupt OK?: Yes

Description:

1. Function Details

a. Argument details

STACK (RAM): Value of stack pointer when  $\overline{\text{NMI}}$  interrupt is executed.

NMIF (RAM): Used as flag indicating LWPST execution, i.e., standby mode.

- b. Example of LWPST execution is shown in figure 14-6. Contents of CPU registers, before NMI interrupt by switch 2 input, are saved and LWPST is executed. Value of stack pointer is saved in STACK (RAM) and \$01 is stored in NMIF (RAM).

Specifications Notes:

N/A



Description:

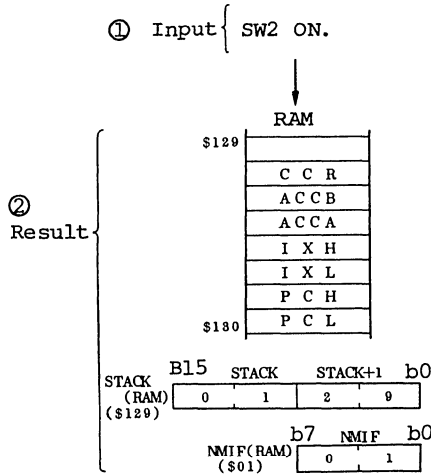


Figure 14-6. Example of LWPST Execution

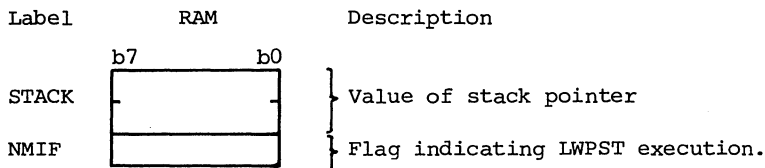
c. LWPST calls neither the program modules nor subroutines.

2. User Notes

The following procedure must be executed before LWPST execution.

a. Initialize stack pointer since  $\overline{\text{NMI}}$  interrupt is executed.

3. RAM Allocation



4. Sample Application

```

    .
    .
    .
    LDS    #$130    ----- Initialize stack pointer.
    .
    .
    .
    
```

Program Module Name: STANDBY MODE

MCU/MPU: HD6301Y0

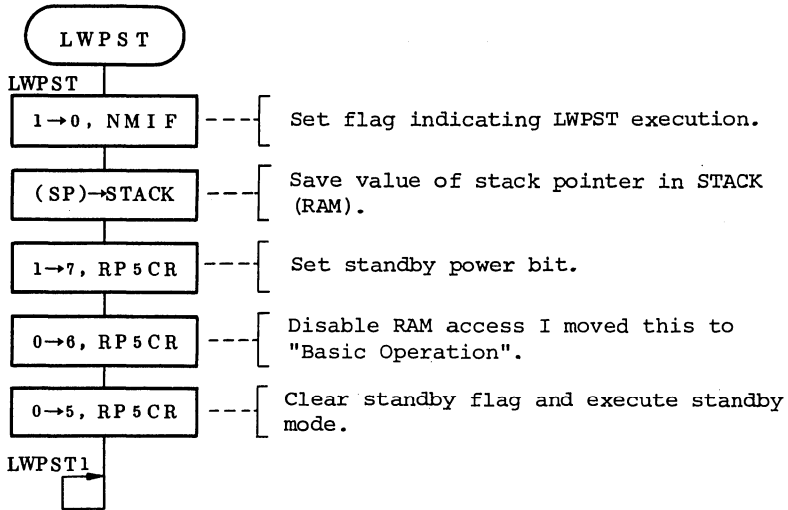
Label: LWPST

Description:

5. Basic Operation

- a. Set flag NMIF(RAM) indicating LWPST execution.
- b. Save stack pointer in STACK(RAM).
- c. Set standby power bit of RAM/port 5 control register.
- d. Clear RAM enable bit of RAM/port 5 control register and disable RAM access to protect RAM data.
- e. Clear standby flag and enter standby mode.

Flowchart:





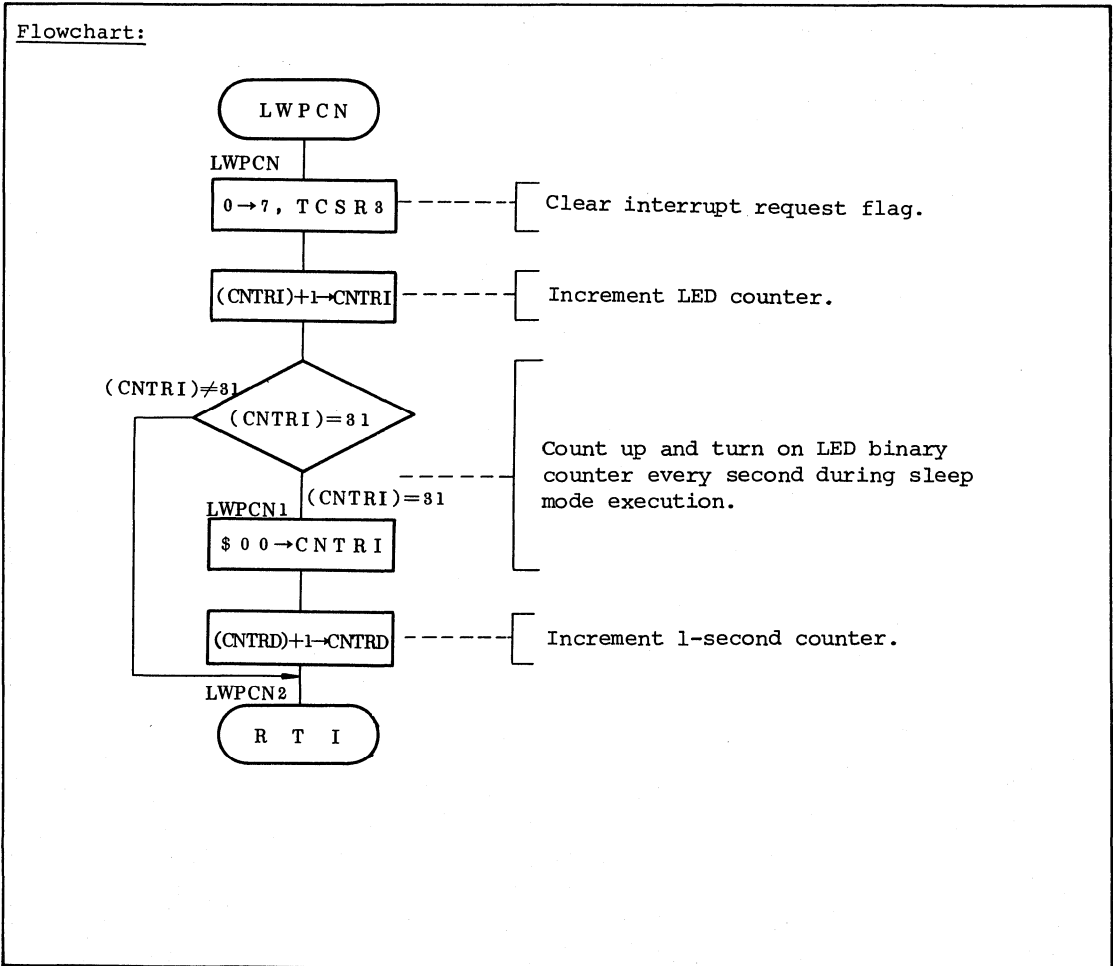
#### 14.4 SUBROUTINE DESCRIPTION

Subroutine Name: INCREMENT COUNTER	MCU/MPU: HD6301Y0	Label: LWPCN
------------------------------------	-------------------	--------------

**Function:**  
Increment counter for LED binary display.

- Basic Operation:**
1. This subroutine is executed at every 32ms interrupt.
  2. Two counters are used to count one second.
  3. LED counter is incremented at every interrupt. When this counter is "31", 1-second counter is counted up.

**Program Module Using This Subroutine:** —



14.5 PROGRAM LISTING

```

00001      *
00002      *
00003      *
00004A 0040      ORG      $40
00005      *
00006A 0040      0001  A  CNTRD  RMB      1      1-second counter
00007A 0041      0001  A  CNTRI  RMB      1      LED counter
00008A 0042      0001  A  NMIF   RMB      1      LWPST execution flag
00009A 0043      0002  A  STACK  RMB      2      Value of stack pointer
00010      *
00011      *
00012      *
00013      0000  A  P1DDR  EQU      $00      Port1 data direction register
00014      0002  A  P1DTR  EQU      $02      Port1 data register
00015      0004  A  P3DDR  EQU      $04      Port3 data direction register
00016      0006  A  P3DTR  EQU      $06      Port3 data register
00017      0016  A  P6DDR  EQU      $16      Port6 data direction register
00018      0017  A  P6DTR  EQU      $17      Port6 data register
00019      0018  A  TCSR3  EQU      $1B      Timer control register3
00020      001C  A  TCONR  EQU      $1C      Time constant register
00021      001D  A  T2CNT  EQU      $1D      Timer2 up counter
00022      0014  A  RP5CR  EQU      $14      RAM/port5 control register
00023      *
00024      *
00025      *
00026      *
00027      *
00028      *
00029A C000      ORG      $C000
00030      *
00031A C000 7B 80 14      LWPMN  BTST  7.RP5CR  Test standby power bit
00032A C003 26 0D C012      BNE    LWPMN2
00033A C005 86 FE      A  LWPMN1  LDAA   #$FE    Turn on active mode LED
00034A C007 97 02      A  STAA   P1DTR
00035A C009 86 FF      A  LDAA   #$FF
00036A C00B 97 06      A  STAA   P3DTR
00037A C00D 8E 013F      A  LDS    #$13F  Initialize stack pointer
00038A C010 20 0B C01D      BRA    LWPMN3
00039A C012 7B 01 42      LWPMN2  BTST  0.NMIF  Test if standby mode execution
00040A C015 27 EE C005      BEQ    LWPMN1
00041A C017 86 F6      A  LDAA   #$F6    Turn on standby mode LED
00042A C019 97 02      A  STAA   P1DTR
00043A C01B 9E 43      A  LDS    STACK   Load stack pointer
00044A C01D 86 01      A  LWPMN3  LDAA   #$01    Select ports 1 and 3 as output
00045A C01F 97 00      A  STAA   P1DDR
00046A C021 97 04      A  STAA   P3DDR
00047A C023 86 52      A  LDAA   #$52
00048A C025 97 1B      A  STAA   TCSR3   Initialize TCSR3
00049A C027 0E      CLI    Enable interrupts
00050A C028 4F      CLRA   Clear RAM
00051A C029 97 42      A  STAA   NMIF
00052A C02B 97 40      A  STAA   CNTRD
00053A C02D 97 41      A  STAA   CNTRI
00054A C02F 7B 02 17      LWPMN4  BTST  1.P6DTR  Test if sleep mode execution
00055A C032 27 05 C039      BEQ    LWPMN5
00056A C034 8D C040 A      JSR    LWSPSP   Execute sleep mode
00057A C037 20 F6 C02F      BRA    LWPMN4

```

```

00058A C039 96 40   A LWPMNS LDAA   CNTRD
00059A C03B 43           COMA
00060A C03C 97 06   A      STAA   P3DTR   Turn on LED binary counter
00061A C03E 20 EF C02F  BRA   LWPMN4
00062 *****
00063 *
00064 *           NAME : LWSPS (SLEEP MODE)
00065 *
00066 *****
00067 *
00068 *           ENTER : NOTHING
00069 *           RETURNS : NOTHING
00070 *
00071 *****
00072A C040 86 FB   A LWSPS  LDAA   #$FB
00073A C042 97 02   A      STAA   P1DTR   Turn on sleep mode LED
00074A C044 1A           SLP           Execute sleep mode
00075A C045 86 FE   A      LDAA   #$FE
00076A C047 97 02   A      STAA   P1DTR   Turn on active mode LED
00077A C049 39           RTS
00078 *****
00079 *
00080 *           NAME : LWPST (STANDBY MODE)
00081 *
00082 *****
00083 *
00084 *           ENTER : NOTHING
00085 *           RETURNS : STACK (STACK POINTER)
00086 *           NMIF (LWPST EXECUTION FLAG)*
00087 *
00088 *****
00089A C04A 72 01 42 LWPST  BSET   0.NMIF  Set LWPST execution flag
00090A C04D 9F 43   A      STS    STACK   Store stack pointer
00091A C04F 72 80 14 BSET   7.RP5CR  Set standby power bit
00092A C052 71 FD 14 BCLR   1.RP5CR  Clear RAM enable bit
00093A C055 71 DF 14 BCLR   5.RP5CR  Clear standby flag
00094A C058 20 FE C058 LWPST1 BRA    LWST1
00095 *****
00096 *
00097 *           NAME : LWPCN (INCREMENT COUNTER)
00098 *
00099 *****
00100A C05A 71 7F 1B LWPCN  BCLR   7.TCSR3  Clear interrupt request flag
00101A C05D 7C 0041 A      INC    CNTRI   Increment LED counter
00102A C060 96 41   A      LDAA   CNTRI
00103A C062 81 1F   A      CMPA   #31
00104A C064 27 01 C067 BEQ    LWPCN1
00105A C066 3B           LWPCN2 RTI
00106A C067 4F           LWPCN1 CLRA
00107A C068 97 41   A      STAA   CNTRI   Turn on LED binary counter
00108A C06A 7C 0040 A      INC    CNTRD   Increment 1-second counter
00109A C06D 20 F7 C066 BRA    LWPCN2
00110 *
00111 *****
00112 *
00113 *           VECTOR ADDRESSES
00114 *

```

```

00115
00116
00117A FFEA          *          ORG      $FFEA
00118A FFEA      C000  A          FDB      LWPMN   IRQ2
00119A FFEC      C05A  A          FDB      LWPCN   CMI
00120A FFEE      C000  A          FDB      LWPMN   TRAP
00121A FFF0      C000  A          FDB      LWPMN   SIO
00122A FFF2      C000  A          FDB      LWPMN   TOI
00123A FFF4      C000  A          FDB      LWPMN   OCI
00124A FFF6      C000  A          FDB      LWPMN   ICI
00125A FFF8      C000  A          FDB      LWPMN   IRQ1/ISF
00126A FFFA      C000  A          FDB      LWPMN   SWI
00127A FFFC      C04A  A          FDB      LWPST   NMI
00128A FFFE      C000  A          FDB      LWPMN   RES
00129          *
00130          END

```

15.1 HARDWARE DESCRIPTION15.1.1 Function

Executes test program of MCU runaway error and trap error detection (operation code error and address error), and displays result on LED.

15.1.2 Microcomputer Operation

The HD6301Y0 sends pulse to the HA1835P voltage regulator controlling bit 0 of port 7 and detects watchdog timer error. In addition, detects operation code error and address error using the trap function.

15.1.3 Peripheral Devices

Switches and LEDs: Switches SW1-SW3 are used to indicate the above three errors for testing. The generation of those errors and subsequent error handling is indicated by LED1-LED3. The relationship between switch settings and LED display is shown in table 15-1.

Table 15-1. Switch Setting and Display for Each Mode

Test Mode	Switch Setting			Display		
	SW1	SW2	SW3	LED1	LED2	LED3
Normal Operation	OFF	OFF	OFF	OFF	OFF	OFF
Watchdog Timer Error	ON	OFF	OFF	ON	OFF	OFF
Operation Code Error	OFF	ON	OFF	OFF	ON	OFF
Address Trap Error	OFF	OFF	ON	OFF	OFF	ON

15.1.4 Circuit Diagram

HA1835P and Error Detection Circuit is shown in figure 15-1.

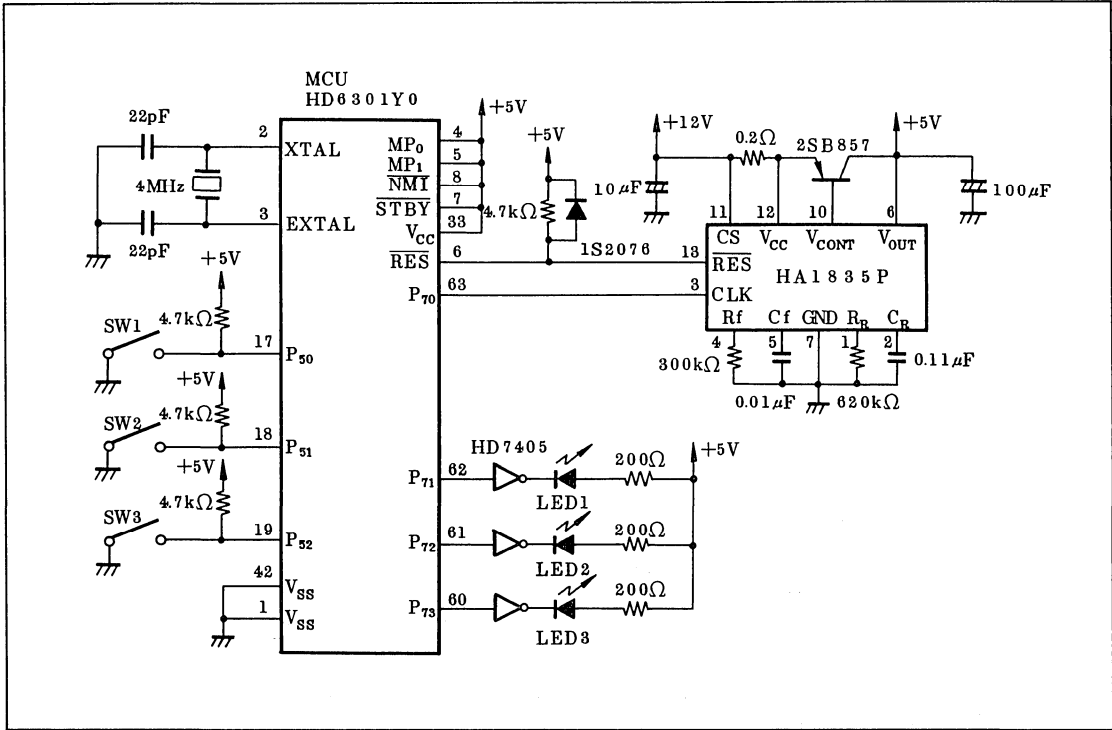


Figure 15-1. HA1835P and Error Detection Circuit

### 15.1.5 Pin Functions

Pin functions at the interface between the HD6301Y0 and switches, LEDs, and the HA1835P are shown in table 15-2.

Table 15-2. Pin Functions

Pin Name (HD6301Y0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (SW, LED, HA1835P)	Program Label
P <sub>50</sub>	Input	Low	Watchdog timer error generation switch	SW1	P5DTR
P <sub>51</sub>	Input	Low	Operation code trap error generation switch	SW2	
P <sub>52</sub>	Input	Low	Address trap error generation switch	SW3	
P <sub>70</sub>	Output	—	Outputs pulse to CLK pin of HA1835P	CLK	P7DTR
P <sub>71</sub>	Output	High	Drives LED indicating watchdog timer error generation	LED1	
P <sub>72</sub>	Output	High	Drives LED indicating operation code trap error generation	LED2	
P <sub>73</sub>	Output	High	Drives LED indicating address trap error generation	LED3	
$\overline{\text{RES}}$	Input	Low	Inputs reset.	$\overline{\text{RES}}$	—

### 15.1.6 Hardware Operation

The timing chart for the watchdog timer function using the HA1835P is shown in figure 15-2.

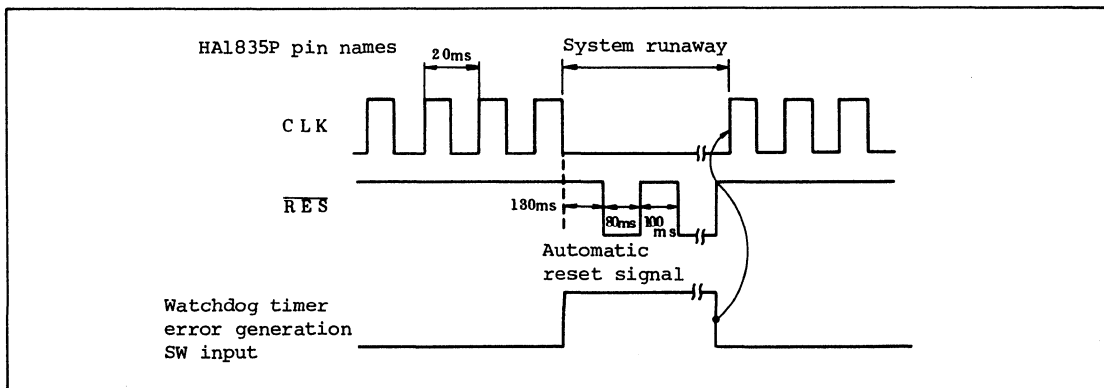


Figure 15-2. Timing Chart for Watchdog Timer



## 15.2 SOFTWARE DESCRIPTION

### 15.2.1 Program Module Configuration

The program module configuration for the HA1835P control and error detection function is shown in figure 15-3.

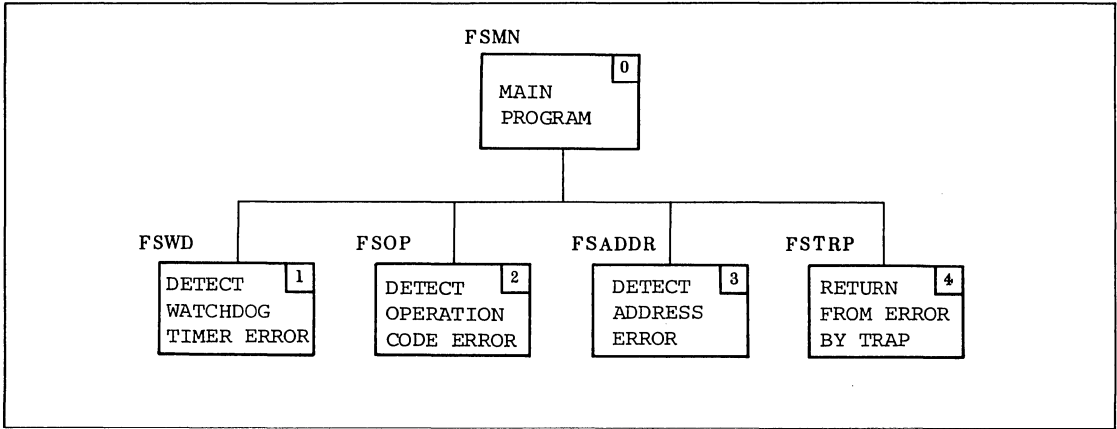


Figure 15-3. Program Module Configuration

### 15.2.2 Program Module Functions

Program module functions are summarized in table 15-3.

Table 15-3. Program Module Functions

No.	Program Module Name	Label	Functions
0	MAIN PROGRAM	FSMN	Perform HA1835P control and error detection.
1	DETECT WATCHDOG TIMER ERROR	FSWD	Stop pulse output to HA1835P and check RES input.
2	DETECT OPERATION CODE ERROR	FSOP	Execute undefined operation code and check operation code error generation.
3	DETECT ADDRESS ERROR	FSADDR	Fetch instruction from other than ROM, RAM and check address error.
4	RETURN FROM ERROR BY TRAP	FSTRP	Return from operation code error and address error.



### 15.2.3 Program Module Process Flow (Main Program)

The flowchart in figure 15-4 demonstrates the procedure for detecting watchdog timer error, operation code error, and address error by SW1-3, using the program module in figure 15-3.

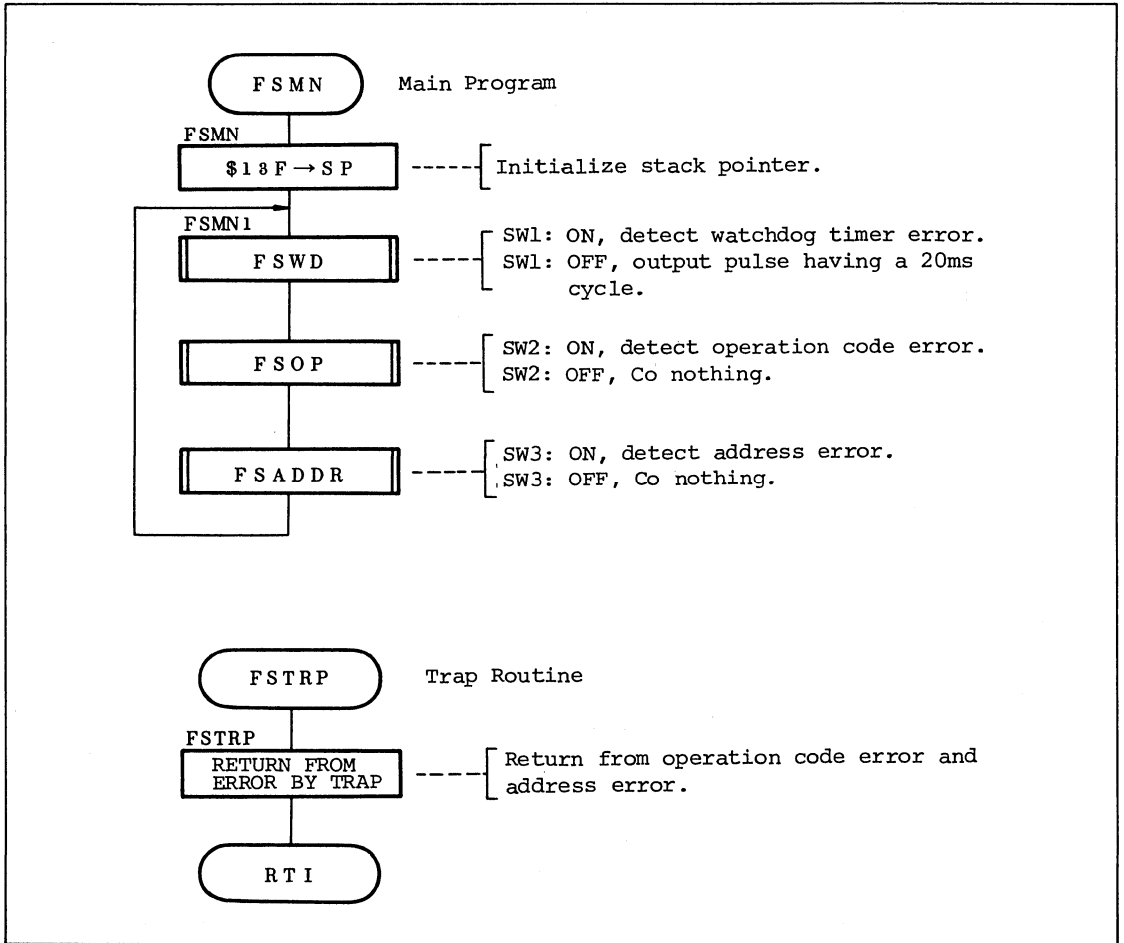


Figure 15-4. Program Module Flowchart

15.3 PROGRAM MODULE DESCRIPTION

Program Module Name: DETECT WATCHDOG  
TIMER ERROR

MCU/MPU: HD6301Y0/  
HD6303Y

Label: FSWD

Function:  
When SW1 is ON, detect watchdog timer error.  
When SW1 is OFF, output pulse to bit 0 of port 7 every 20ms.

Arguments:  
None

Changes in CPU  
Registers and Flags:

ACCD	
ACCA	ACCB
x	•
IX	
•	
C	V
x	x
Z	N
x	x
I	H
•	•

• : Not affected  
x : Undefined  
↓ : Result

Specifications:

ROM (Bytes): 64  
RAM (Bytes): 2  
Stack (Bytes): 0  
No. of cycles: 10041  
Reentrant: No  
Relocatable: No  
Interrupt OK?: Yes

Description:

1. Function Details

a. FSWD has no arguments.

b. Example of FSWD execution is shown in figure 15-5. When SW1 is OFF, output pulse to bit 0 of port 7 every 20ms. When SW1 is ON, stop pulse output and turn on LED1 after reset. When SW1 is OFF again, output pulse to bit 0 of port 7 and turn off LED1.

The timing diagram shows four signals over time. SW1 starts OFF, then transitions to ON. P70 shows a series of pulses with a 20ms interval. RES shows a single pulse. LED1 starts OFF, then transitions to ON when SW1 is ON and RES is active, and returns to OFF when SW1 is OFF.

Figure 15-5. Example of FSWD Execution

Specifications Notes:

Program Module Name: DETECT WATCHDOG  
TIMER ERROR

MCU/MPU: HD6301Y0/  
HD6303Y

Label: FSWD

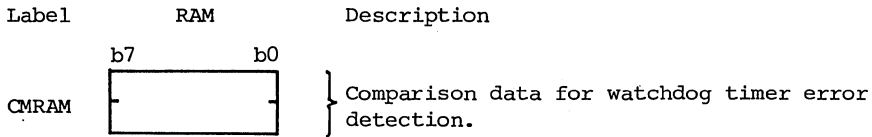
Description:

c. FSWD calls neither the program modules nor subroutines.

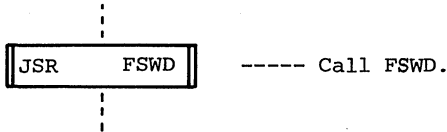
2. User Notes

Use SW1 independently of other switches.

3. RAM Allocation



4. Sample Application



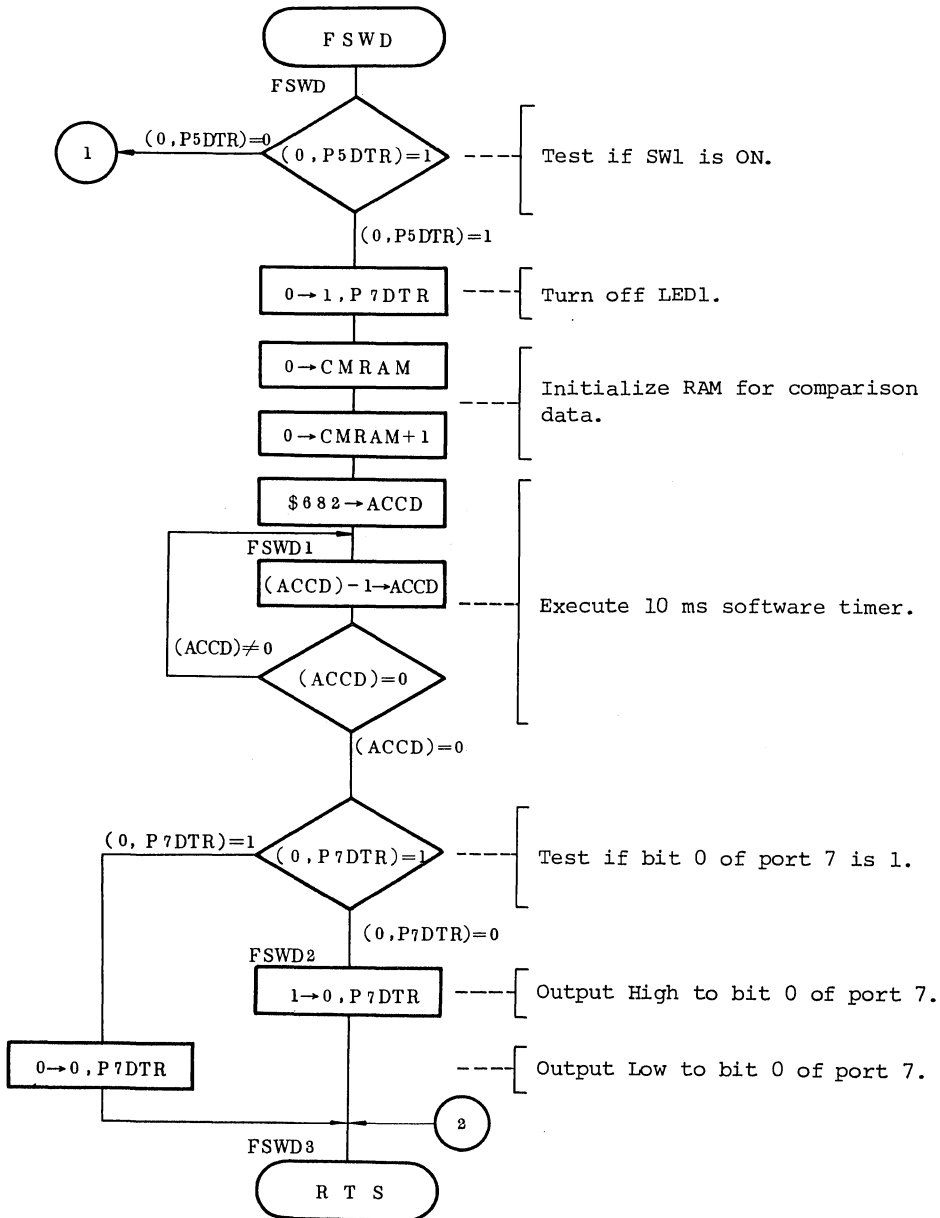
5. Basic Operation

a. When SW1 is ON, the following operations are performed.

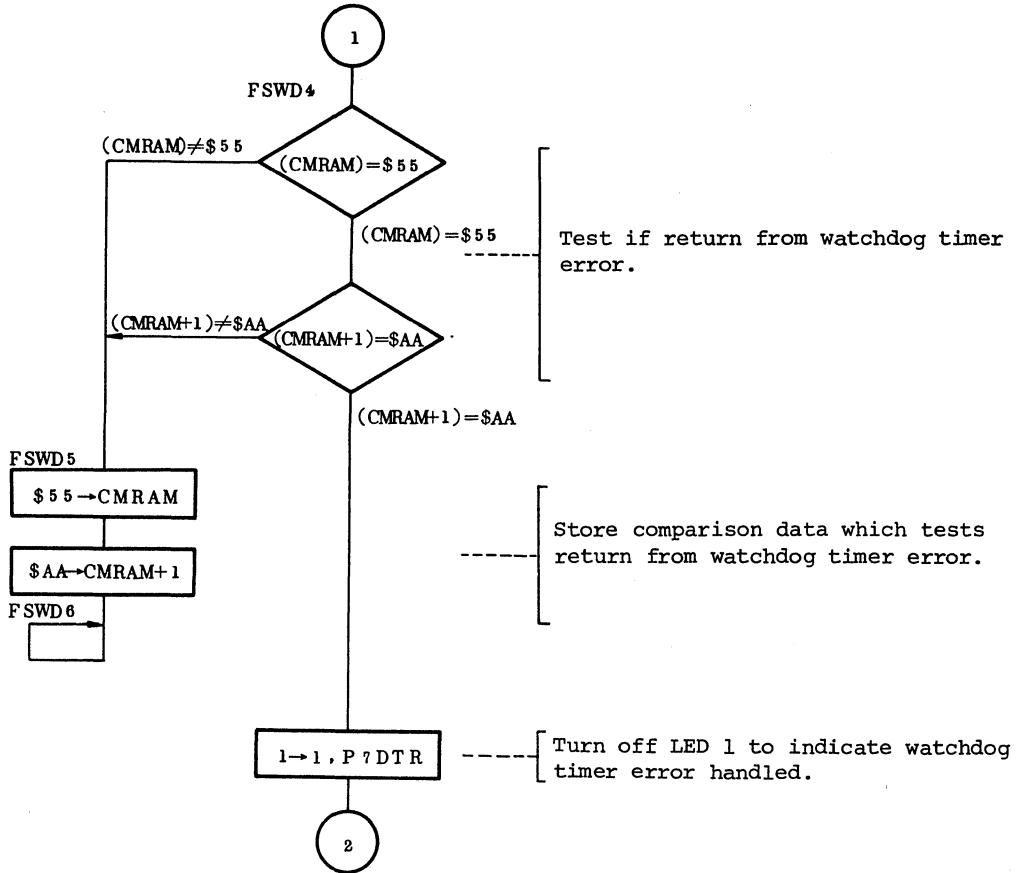
- i. After power ON, data is stored in CDRAM (RAM), an infinite loop is executed, and pulse output to the HA1835P is stop.
- ii. The HA1835P determines this status as system runaway and sets  $\overline{RES}$  pin to LOW.
- iii. After reset, data in CDRAM (RAM) is compared with data previously stored. If these are the same, LED1 is turned on.

b. When SW1 is OFF, 10 ms software timer is executed and the output to bit 0 of port 7 is inverted.

Flowchart:



Flowchart:



Program Module Name: DETECT  
OPERATION CODE  
ERROR

MCU/MPU: HD6301Y0/  
HD6303Y

Label: FSOP

Function:  
When SW2 is ON, execute an undefined operation code and generate an operation code error. If operation code error is detected, turn on LED2. When SW2 is turned OFF, turn off LED2.

Arguments:

Contents	Storage Location	No. of Bytes
Entry	---	---
Re-turns	Error mode	TRMD (RAM) 1

Changes in CPU

Registers and Flags:

ACCD	
ACCA	ACCB
x	x
IX	
x	
C	V
●	x
Z	N
x	x
I	H
●	●

● : Not affected  
x : Undefined  
↓ : Result

Specifications:

ROM (Bytes): 27  
RAM (Bytes): 1  
Stack (Bytes): 0  
No. of cycles: 85  
Reentrant: No  
Relocatable: No  
Interrupt OK?: Yes

Description:

1. Function Details
  - a. Argument details  
TRMD(RAM): Contains data indicating operation code error.
  - b. Example of FSOP execution is shown in figure 15-6. When operation error is generated, turn on LED2.
  - c. FSOP calls an other program module as shown in Table 15-4.

SW2 OFF  
ON  
LED2 OFF

Operation code error generation

Figure 15-6. Example of FSOP Execution

Specifications Notes:  
"No. of cycles" in "Specifications" indicates the number of cycles required to handle an operation code error.



Program Module Name: DETECT  
 OPERATION CODE  
 ERROR

MCU/MPU: HD6301Y0/  
 HD6303Y

Label: FSOP

Description:

Table 15-4. Program Module Called by FSOP

Program Module Name	Label	Function
RETURN FROM ERROR BY TRAP	FSTRP	Return from operation code error or address error.

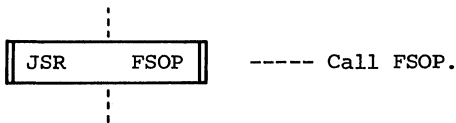
2. User Notes

Use SW2 independently.

3. RAM Allocation

Label	RAM	Description
TRMD	b7      b0 <div style="border: 1px solid black; width: 100px; height: 15px; margin: 5px 0;"></div>	} Data indicating operation code error.

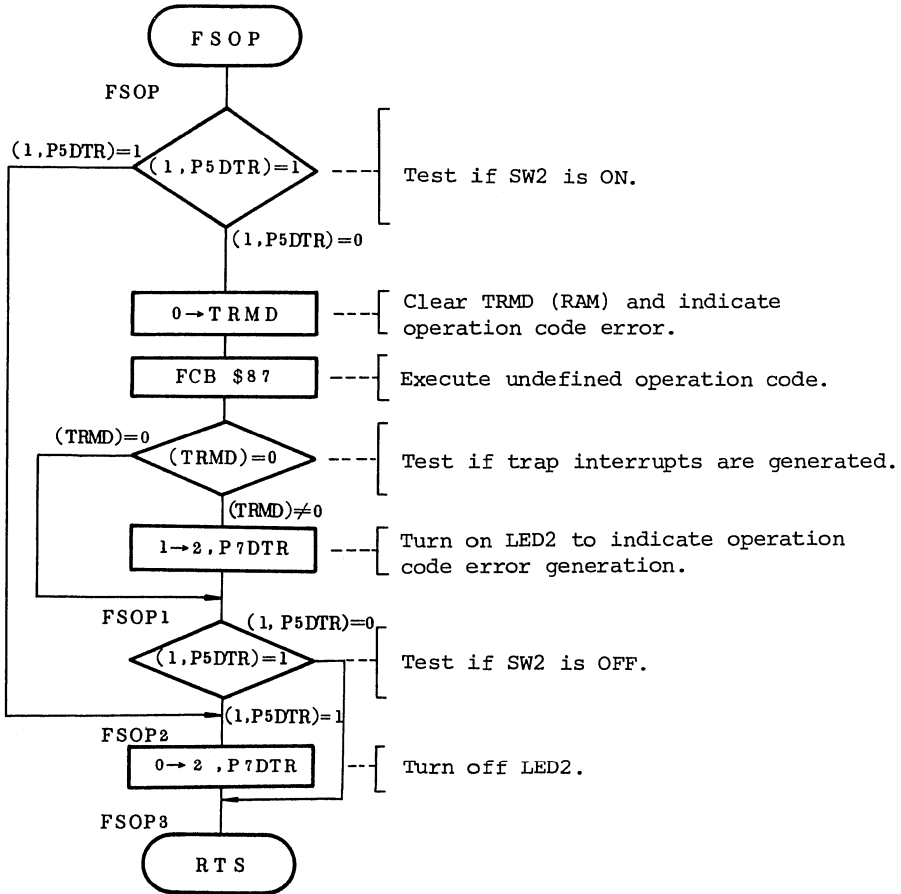
4. Sample Application



5. Basic Operation

- a. When SW2 is ON, execute operation as follows;
  - i. TRMD(RAM) is cleared to indicate operation code error.
  - ii. Undefined operation code "\$87" is executed.
  - iii. LED2 is turned on after returning from trap interrupts.
- b. When SW2 is turned OFF, turn off LED2.

Flowchart:





Program Module Name: DETECT ADDRESS ERROR

MCU/MPU: HD6301Y0/  
HD6303Y

Label: FSADDR

Function:

When SW3 is ON, jump to address for I/O ports and generate address error. If address error is detected, turn on LED3. When SW3 is turned OFF, turn off LED3.

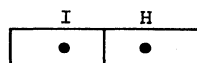
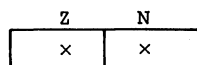
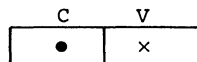
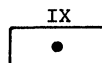
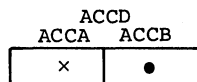
Arguments:

Contents	Storage Location	No. of Bytes
Entry	---	---

Re-	Error	TRMD	1
turns	mode	(RAM)	

Changes in CPU

Registers and Flags:



• : Not affected  
x : Undefined  
↓ : Result

Specifications:

ROM (Bytes): 30  
RAM (Bytes): 1  
Stack (Bytes): 0  
No. of cycles: 78  
Reentrant: No  
Relocatable: No  
Interrupt OK?: Yes

Description:

1. Function Details

a. Argument details

TRMD (RAM): Contains data indicating address error.

- b. Example of FSADDR execution is shown in figure 15-7. When address error is generated, turn on LED3.

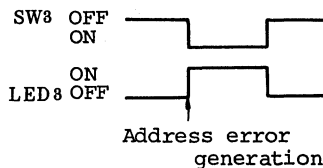


Figure 15-7. Example of FSADDR Execution

Specifications Notes:

"No. of cycles" in "Specifications" indicates the number of cycles required when address error is generated.

Program Module Name: DETECT ADDRESS  
ERROR

MCU/MPU: HD6301Y0/  
HD6303Y

Label: FSADDR

Description:

c. FSADDR calls an other program module as shown in Table 15-5.

Table 15-5. Program Module Called in FSADDR

Program Module Name	Label	Functions
RETURN FROM ERROR BY TRAP	FSADDR	Return from operation code error or address error.

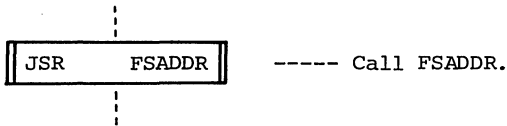
2. User Notes

Use SW3 independently of other switches.

3. RAM Allocation

Label	RAM	Description			
TRMD	<table border="1"><tr><td>b7</td><td> </td><td>b0</td></tr></table>	b7		b0	} Data indicating address error.
b7		b0			

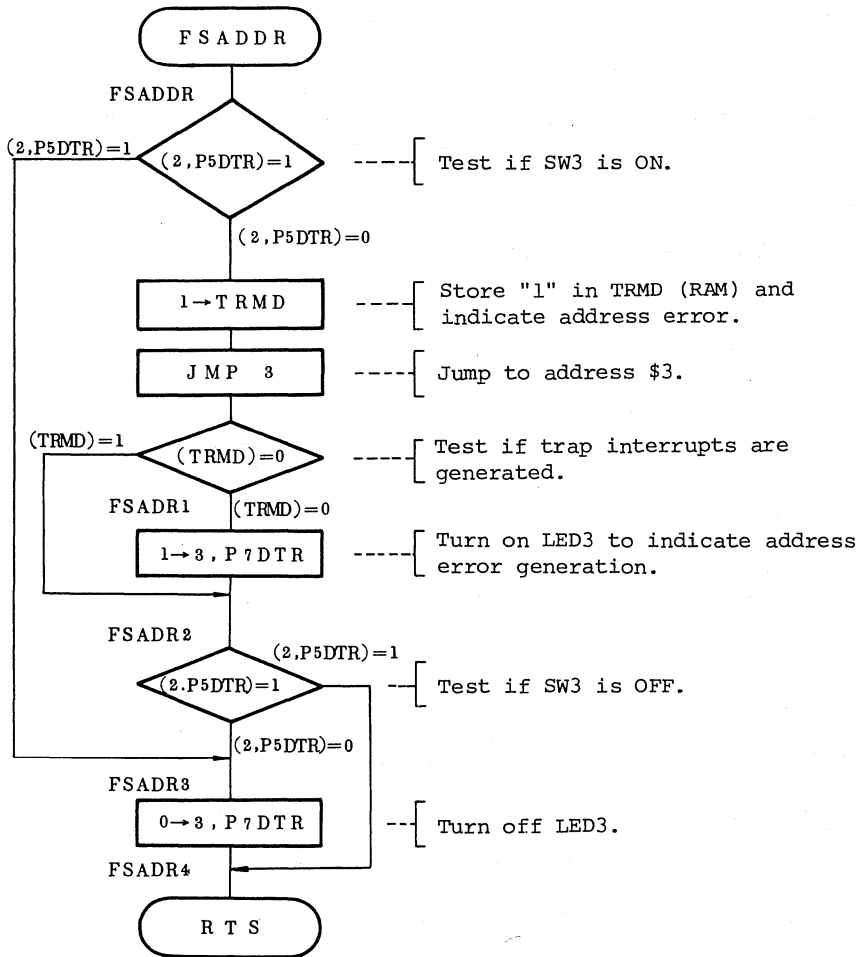
4. Sample Application



5. Basic Operation

- a. When SW3 is ON, execute operations as follows;
  - i. Store "1" in TRMD (RAM) to indicate address error.
  - ii. Execute "JMP 3" to jump to port 3 data register.
  - iii. Turn on LED3 after returning from trap interrupts.
- b. When SW3 is turned OFF, turn off LED3.

Flowchart:



Program Module Name: RETURN FROM  
ERROR BY TRAP

MCU/MPU: HD6301Y0/  
HD6303Y

Label: FSTRP

Function:

When operation code error or address error is generated, return to the program where interrupts are generated by controlling the program counter.

Arguments:

Contents	Storage Location	No. of Bytes
Entry Error Mode	TRMD (RAM)	1

Re-  
turns

Changes in CPU

Registers and Flags:

ACCD	
ACCA	ACCB
x	x

IX
x

C	V
x	x

Z	N
x	x

I	H
●	●

● : Not affected  
x : Undefined  
↑ : Result

Specifications:

ROM (Bytes): 30  
RAM (Bytes): 1  
Stack (Bytes): 0  
No. of cycles: 45  
Reentrant: No  
Relocatable: No  
Interrupt OK?: No

Description:

1. Function Details

a. Argument details

TRMD (RAM): Holds data indicating what error is generated.  
Table 15-6 shows flag functions.

- b. Example of FSTRP execution is shown in figure 15-8. If entry argument is as shown in part ① of figure 15-8, data for program counter in stack area is changed as shown in part ② of figure 15-8.

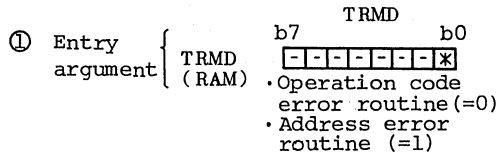
Specifications Notes:

"No of cycles" in "Specifications" indicates the number of cycles required when operation code error is generated.

Description:

Table 15-6. Flag Functions

Label	bit 0	Function
TRMD	0	Execute routine for operation code error generation.
	1	Execute routine for address error generation.



Address	TRMD=0		TRMD=1	
	b7	b0	b7	b0
\$FF	5	4	0	3
\$FE	F	0	0	0
\$FD	0	0	0	0
\$FC	0	0	0	0

② Result

Address	TRMD=0		TRMD=1	
	b7	b0	b7	b0
\$FF	5	5	7	3
\$FE	F	0	F	0
\$FD	0	0	0	0
\$FC	0	0	0	0

Figure 15-8. Example of FSTRP Execution

2. User Notes

Execute FSTRP with routines beginning of labels FSOP or FSADDR.

3. RAM Allocation

Label	RAM	Description				
TRMD	<table border="1" style="display: inline-table;"> <tr> <td style="width: 10px;">b7</td> <td style="width: 10px;">b0</td> </tr> <tr> <td style="width: 20px;"> </td> <td style="width: 20px;"> </td> </tr> </table>	b7	b0			Data indicating operation code error or address error.
b7	b0					

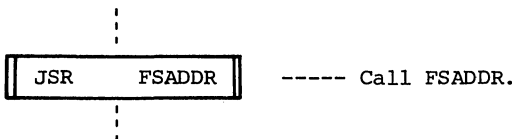
Program Module Name: RETURN FROM  
ERROR BY TRAP

MCU/MPU: HD6301Y0/  
HD6303Y

Label: FSTRP

Description:

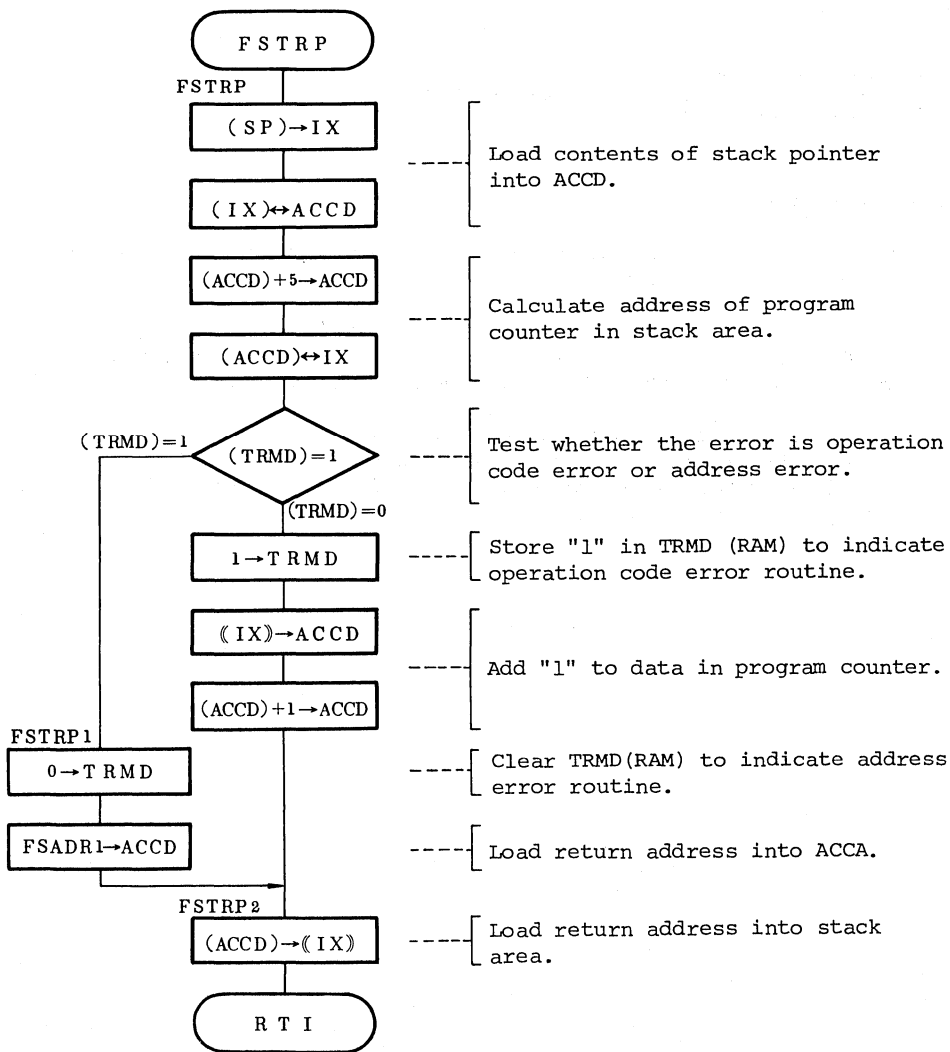
4. Sample Application



5. Basic Operation

- a. Depending on data in TRMD (RAM), either an operation code error or address error has occurred.
- b. In the case of an operation code error, add "1" to the program counter saved on the stack, and execute the program from the instruction address following that where the operation code error was generated.
- c. In the case of an address error, change data in the stack area to execute the program from the label "FSADR1" in routine FSADDR. An address error is generated when "JMP 3" is executed and the program attempts to execute address \$3.

Flowchart:



## 15.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

## 15.5 PROGRAM LISTING

```

00001          *
00002          ****  RAM ALLOCATION  *****
00003          *
00004A 0040          ORG    $40
00005          *
00006A 0040    0002  A CMRAM  RMB    2      Data for comparing
00007A 0042    0001  A TRMD   RMB    1      Error mode
00008          *
00009          ****  SYMBOL DEFINITION  *****
00010          *
00011          0018  A P7DTR  EQU    $18      Port 7 data register
00012          0014  A RPSCR  EQU    $14      RAM/PORTS control register
00013          0015  A P5DTR  EQU    $15      Port 5 data register
00014          0020  A P5DDR  EQU    $20      Port 5 data direction register
00015          *****
00016          *
00017          *      MAIN PROGRAM : FSMN      *
00018          *
00019          *****
00020          *
00021A C000          ORG    $C000
00022          *
00023A C000 8E 013F  A FSMN   LDS    H$13F    Initialize stack pointer
00024A C003 8D 06 C00B FSMN1  BSR    FSWD    Check watchdog timer error
00025A C005 8D 43 C04A        BSR    FSOP    Check operation error
00026A C007 8D 5A C063        BSR    FSADDR  Check address error
00027A C009 20 F8 C003        BRA    FSMN1
00028          *****
00029          *
00030          *      NAME : FSWD (DETECT WATCHDOG TIMER ERROR)  *
00031          *
00032          *****
00033          *
00034          *      ENTRY   : NOTHING      *
00035          *      RETURNS : NOTHING      *
00036          *
00037          *****
00038A C00B 7B 01 15  FSWD   BTST   0.P5DTR  Test if SW1=ON
00039A C00E 27 1F C02F BEQ    FSWD4   Branch if SW1=ON
00040A C010 71 FD 18    BCLR   1.P7DTR  Turn off LED1
00041A C013 7F 0040  A      CLR    CMRAM   Clear CMRAM
00042A C016 7F 0041  A      CLR    CMRAM+1
00043A C019 CC 0682  A      LDD    H$682   Execute 10 ms software timer
00044A C01C 83 0001  A FSWD1  SUBD   H1
00045A C01F 26 FB C01C BNE    FSWD1
00046A C021 7B 01 18    BTST   0.P7DTR  Test if P70 = 1
00047A C024 27 05 C02B BEQ    FSWD2   Branch if P70 = 0
00048A C026 71 FE 18    BCLR   0.P7DTR  Output Low to P70
00049A C029 20 03 C02E BRA    FSWD3
00050A C02B 72 01 18  FSWD2  BSET   0.P7DTR  Output High to P70
00051A C02E 39        FSWD3  RTS
00052A C02F 86 55  A FSWD4  LDAA   H$55    Test if CMRAM=$55
00053A C031 91 40  A        CMPA  CMRAM
00054A C033 26 08 C040        BNE   FSWD5   Branch if not equal
00055A C035 86 AA  A        LDAA  H$AA    Test if CMRAM+1=$AA
00056A C037 91 41  A        CMPA  CMRAM+1
00057A C039 26 05 C040        BNE   FSWD5   Branch if not equal

```



```

00058A C03B 72 02 18      BSET  1.P7DTR  Turn on LED1
00059A C03E 20 EE C02E    BRA   FSWD3
00060A C040 86 55      A FSWD5 LDAA  H$55      Initialize CMRAM:CMRAM+1
00061A C042 97 40      A      STAA  CMRAM
00062A C044 86 AA      A      LDAA  H$AA
00063A C046 97 41      A      STAA  CMRAM+1
00064A C048 20 FE C048 FSWD6  BRA   FSWD6
00065
00066 *****
*
00067 *      NAME : FSOP (DETECT OPERATION CODE ERROR) *
00068 *
00069 *****
00070 *
00071 *      ENTRY   : NOTHING
00072 *      RETURNS : TRMD (ERROR MODE)
00073 *
00074 *****
00075A C04A 7B 02 15    FSOP  BTST  1.P5DTR  Test if SW2=ON
00076A C04D 26 10 C05F BNE  FSWP2  Branch if SW2=ON
00077A C04F 7F 0042 A    CLR  TRMD   CLear TRMD
00078A C052      87      A    FCB  $87    Execut undefined op-code
00079A C053 96 42      A    LDAA  TRMD   Test if TRMD=1?
00080A C055 27 03 C05A BEQ  FSWP1  Branch if TRMD=0
00081A C057 72 04 18      BSET  2.P7DTR  Turn on LED2
00082A C05A 7B 02 15    FSOP1 BTST  1.P5DTR  Test if SW2=OFF
00083A C05D 27 03 C062 BEQ  FSWP3  Branch if SW2=OFF
00084A C05F 71 FB 18    FSOP2 BCLR  2.P7DTR  Turn off LED2
00085A C062 39    FSOP3 RTS
00086 *****
00087 *
00088 *      NAME : FSADDR (DETECT ADDRESS ERROR) *
00089 *
00090 *****
00091 *
00092 *      ENTRY   : NOTHING
00093 *      RETURNS : TRMD (ERROR MODE)
00094 *
00095 *****
00096A C063 7B 04 15    FSADDR BTST  2.P5DTR  Test if SW3=ON
00097A C066 26 13 C07B BNE  FSADR3  Branch if SW3=ON
00098A C068 86 01      A    LDAA  H1     Store 1 in TRMD
00099A C06A 97 42      A    STAA  TRMD
00100A C06C 7E 0003 A    JMP  3       Execute address error mode
00101A C06F 96 42      A    LDAA  TRMD   Test if TRMD=0?
00102A C071 26 03 C076 BNE  FSADR2  Branch if TRMD=L
00103A C073 72 08 18    FSADR1 BSET  3.P7DTR  Turn on LED3
00104A C076 7B 04 15    FSADR2 BTST  2.P5DTR  Test if SW3=OFF
00105A C079 26 03 C07E BNE  FSADR4  Branch if SW3=ON
00106A C07B 71 F7 18    FSADR3 BCLR  3.P7DTR  Turn off LED3
00107A C07E 39    FSADR4 RTS
00108 *****
00109 *
00110 *      NAME : FSTRP (RETURN FROM ERROR BY TRAP) *
00111 *
00112 *****
00113 *
00114 *      ENTRY   : TRMD (ERROR MODE)

```

```

00115          *          RETURNS : NOTHING          *
00116          *
00117          *****
00118A C07F 30          FSTRP  TSX          Load stack pointer into ACCD
00119A C080 18          XGDX
00120A C081 C3 0005  A          ADDD  H$S          Calcurate program counter
00121A C084 18          XGDX
00122A C085 96 42  A          LDAA  TRMD          Test if op-code or address error?
00123A C087 26 0B C094 BNE  FSTRP1        Branch if address error
00124A C089 86 01  A          LDAA  #1          Store 1 in TRMD
00125A C08B 97 42  A          STAA  TRMD
00126A C08D EC 00  A          LDD   0.X          Increment program counter
00127A C08F C3 0001  A          ADDD  #1
00128A C092 20 06 C09A BRA  FSTRP2
00129A C094 7F 0042  A FSTRP1 CLR  TRMD          Clear TRMD
00130A C097 FC C073  A          LDD   FSADR1
00131A C09A ED 00  A FSTRP2 STD  0.X          Store program counter
00132A C09C 3B          RTI
00133          *****
00134          *
00135          *          VECTOR ADDRESSES          *
00136          *
00137          *****
00138          *
00139A FFEA          ORG  $FFEA
00140          *
00141A FFEA  C000  A          FDB  FSMN  IRQ2
00142A FFEC  C000  A          FDB  FSMN  CMI
00143A FFEE  C07F  A          FDB  FSTRP TRAP
00144A FFF0  C000  A          FDB  FSMN  SIO
00145A FFF2  C000  A          FDB  FSMN  TOI
00146A FFF4  C000  A          FDB  FSMN  OCI
00147A FFF6  C000  A          FDB  FSMN  ICI
00148A FFF8  C000  A          FDB  FSMN  IRQ1/ISF
00149A FFFA  C000  A          FDB  FSMN  SWI
00150A FFFC  C000  A          FDB  FSMN  NMI
00151A FFFE  C000  A          FDB  FSMN  RES
00152          *
00153          END

```



# HD6301 /HD6303 SERIES HANDBOOK

## Section Nine

# C Language Programming Techniques



## Section 9

# C Language Programming Techniques

## Table of Contents

	Page
1. HOW TO USE APPLICATION NOTES .....	1105
1.1 Hardware Section .....	1107
1.1.1 Function .....	1107
1.1.2 Microcontroller Applications .....	1107
1.1.3 Circuit Diagram .....	1108
1.1.4 Memory Map .....	1109
1.1.5 Pin Functions .....	1110
1.1.6 Hardware Operation .....	1111
1.2 Software Section .....	1112
1.2.1 Program Module Configuration .....	1112
1.2.2 Program Module Functions .....	1113
1.2.3 Program Module Sample Application (Main Program) .....	1114
1.3 Program Module Section .....	1116
1.3.1 Page Heading .....	1117
1.3.2 Function .....	1117
1.3.3 Arguments .....	1117
1.3.4 Libraries Required for Program Execution .....	1118
1.3.5 Specifications .....	1119
1.3.6 Description .....	1120
1.3.7 PAD .....	1123
1.4 Subroutine Section .....	1124
1.4.1 Page Heading .....	1125
1.4.2 Function .....	1125
1.4.3 Basic Operation .....	1125
1.4.4 Program Modules That Use This Function .....	1126
1.4.5 PAD .....	1126
1.5 Program Listing Section .....	1127
1.5.1 Main Program Listing .....	1127
1.5.2 C Source Listing .....	1128
1.5.3 Output Object Listing of C Compiler .....	1131
1.5.4 Linkage Listing .....	1132

1.6	Program Module Use.....	1135
1.7	Pad Symbols Description.....	1137
1.8	Symbols.....	1138
2.	DARLINGTON TRANSISTOR DRIVE (LED DYNAMIC DISPLAY).....	1139
2.1	Hardware Description.....	1139
2.1.1	Function.....	1139
2.1.2	Microcontroller Applications.....	1139
2.1.3	Circuit Diagram.....	1139
2.1.4	Pin Functions.....	1140
2.1.5	Hardware Operation.....	1141
2.2	Software Description.....	1142
2.2.1	Program Module Configuration.....	1142
2.2.2	Program Module Functions.....	1142
2.2.3	Program Module Sample Application (Main Program).....	1143
2.3	Program Module Description.....	1144
2.4	Program Listing.....	1152
2.4.1	Main Program Listing.....	1152
2.4.2	C Source Listing.....	1153
2.4.3	Output Object Listing of C Compiler.....	1154
2.4.4	Linkage Listing.....	1157
3.	8 x 4 KEY MATRIX.....	1159
3.1	Hardware Description.....	1159
3.1.1	Function.....	1159
3.1.2	Microcontroller Applications.....	1159
3.1.3	Circuit Diagram.....	1160
3.1.4	Pin Functions.....	1161
3.1.5	Hardware Operation.....	1161
3.2	Software Description.....	1162
3.2.1	Program Module Configuration.....	1162
3.2.2	Program Module Functions.....	1162
3.2.3	Program Module Sample Application (Main Program).....	1163

3.3	Program Module Description	1165
3.4	Program Listing	1177
3.4.1	Main Program Listing	1177
3.4.2	C Source Listing	1178
3.4.3	Output Object Listing of C Compiler	1180
3.4.4	Linkage Listing	1185
4.	EXTERNAL EXPANSION	1187
4.1	Hardware Description	1187
4.1.1	Function	1187
4.1.2	Microcontroller Applications	1187
4.1.3	Circuit Diagram	1187
4.1.4	Memory Map	1188
4.1.5	Hardware Operation	1189
4.2	Software Description	1190
4.2.1	Program Module Configuration	1190
4.2.2	Program Module Functions	1190
4.2.3	Program Module Sample Application (Main Program)	1192
4.3	Program Module Description	1194
4.4	Program Listing	1220
4.4.1	Main Program Listing	1220
4.4.2	C Source Listing	1222
4.4.3	Output Object Listing of Compiler	1226
4.4.4	Linkage Listing	1232
APPENDIX A.		
	C Program and Assembly Program Comparison	1235





## FOREWORD

The HD6301 series is composed of 8-bit single chip CMOS microprogrammed microcontrollers. The HD6301 series provides pipeline Control, halt, and memory-ready functions for processing data. This application note contains C language programs which are described using application functions routines as examples. In general, it is difficult to write a program in a high-level language like C which carries out low level functions, such as controlling ports, timer interrupts, etc. However, this application note's program have been written in C, using mainly the hardware control functions listed above, and have been written to help users design hardware systems, employing specific circuit diagrams, timing charts and program modules. This application note also contains assembly language program descriptions, with functions equivalent to the C language programs. Please use these descriptions to compare the two languages.

**Caution:**

**Test the application examples, in this application note for proper results before incorporating them into production operations systems.**

## REFERENCES

- HD6301 Series Application Notes (C Language) (ADJ-502-003)
- HD6301X0, HD6303X Application Notes (Hardware) (68-3-11)
- VAX/VMS6301C Compiler User's Manual (HS31VCLV1S)
- C Language Manual (S999CLL1M)

# SECTION 1. HOW TO USE APPLICATION NOTES

This chapter describes the configuration for all system application examples in this application note.

Each application example in this application note is divided into 5 sections, as shown in Figure 1-1.

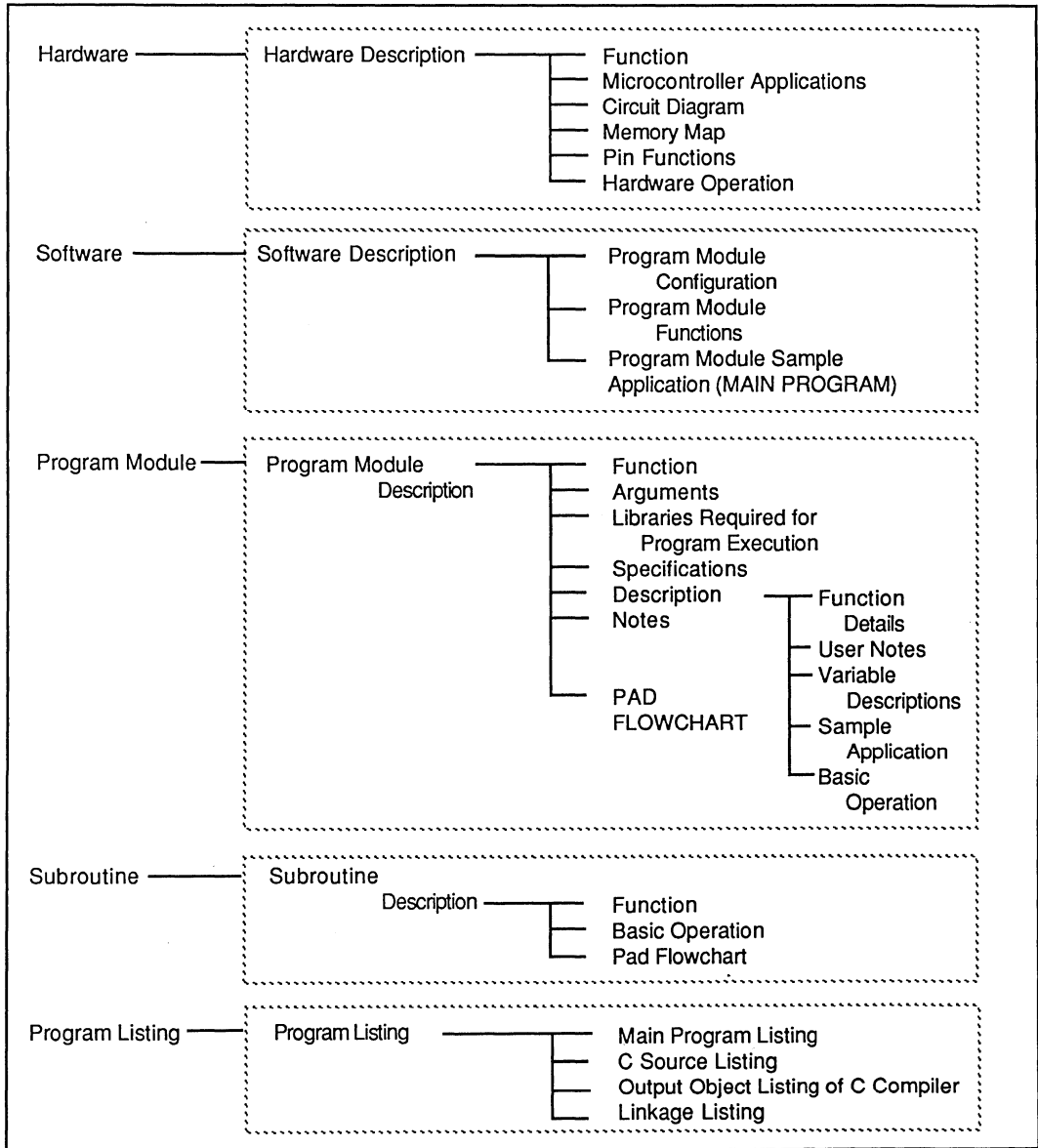


Figure 1-1. Application Example Configuration

**1. Hardware**

Describes the function, circuit diagram, and hardware operation of the HD6301 hardware example.

**2. Software**

Describes the program module which controls the hardware in the hardware section example and shows the main program using all program modules.

**3. Program Module**

Describes the program modules presented in the software section in detail program written in modular format allow more efficient system use.

**4. Subroutine**

Describes the subroutines used in the above program modules.

Refer to this section when necessary while using the program modules.

**5. Program Listing**

Presents the sample application program listings for the above modules.

A detailed explanation of all five sections follows.

## 1.1 Hardware Section

### 1.1.1 Function

"Function" describes system specifications for the hardware used in the particular application (figure 1-2).

#### 4.1.1 Function

The external expansion application controls external memory and peripheral LSIs using the HD6301Y0. It uses the HD6350 (ACIA) as an asynchronous serial interface with a console typewriter. It also controls a liquid crystal module H2571 and displays console typewriter input characters using the HD6321 (PIA).

Figure 1-2. Function Section

### 1.1.2 Microcontroller Applications

"Microcontroller Applications" describe the functions of the microcontroller, in the particular application (figure 1-3).

#### 4.1.2 Microcontroller Applications

This application interfaces with external LSIs through an address bus, data bus, and control signals ( $R/\bar{W}$  and E) using the HD6301Y0 external expansion function.

Figure 1-3. Microcontroller Applications



## 1.1.4 Memory Map

Describes address decoding and system memory map for the application system (figure 1-5). (used in Section 4, "External Expansion" only.)

### 4.1.4 Memory Map

Memories and peripheral LSIs are allocated in external address space using an address decoder (HD74HC138).

Address lines A13, A14 and A15 are connected to pins A, B and C of the HD74HC138.

Address space \$8000-\$FFFF is divided into 8k-byte units. Table 4-1 shows the system address decoding.

Table 4-1. System Address Decoding

HD74HC138										Address	Allocation
Input					Output						
G1	G2A	G2B	C	B	A	Y4	Y5	Y6	Y7		
				A14	A13						
H	L	L	L	L	L	L	H	H	H	\$8000-\$9FFF	RAM
H	L	L	L	L	H	H	L	H	H	\$A000-\$BFFF	PIA
H	L	L	L	H	L	H	H	L	H	\$C000-\$DFFF	ACIA
H	L	L	L	H	H	H	H	L	L	\$E000-\$FFFF	ROM

Figure 4-2 shows system memory map.

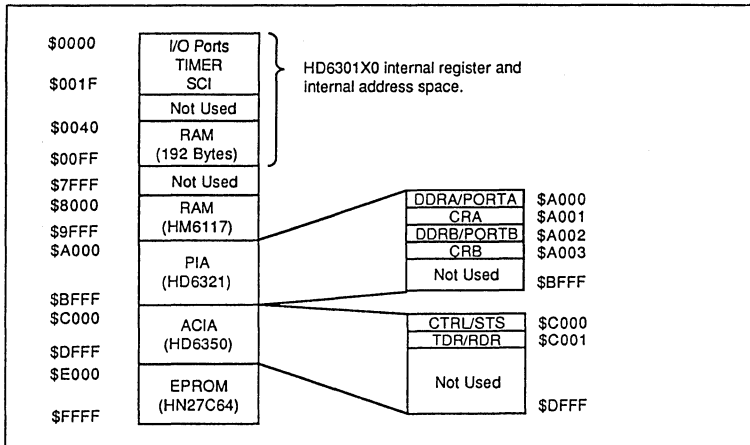


Figure 4-2. System Memory Map

## Figure 1-5. Memory Map Section



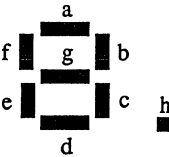
## 1.1.5 Pin Functions

A table describes the pin functions for interfacing with external circuits (figure 1-6).

### 2.1.4 Pin Functions

Table 2-1 shows the pin functions at the interface between the HD6301X0 and an 8-digit x 8-segment LED.

Table 2-1. Pin Functions

Pin Name (HD6301X0)	Input/Output	Active Level (High or Low)	Function	Pin name (LED)	Program Label
P60	Output	High	Outputs digit data to 8-digit x 8-segment LED.	DIG1	P6DTR
P61	Output	High		DIG2	
P62	Output	High		DIG3	
P63	Output	High		DIG4	
P64	Output	High		DIG5	
P65	Output	High		DIG6	
P66	Output	High		DIG7	
P67	Output	High		DIG8	
P10	Output	Low	Outputs segment data to 8-digit x 8-segment LED. 	a	PIDTR
P11	Output	Low		b	
P12	Output	Low		c	
P13	Output	Low		d	
P14	Output	Low		e	
P15	Output	Low		f	
P16	Output	Low		g	
P17	Output	Low		h	

• "Active level" in table 1 indicates the following:

High : logical 1  
 Low : logical 0  
 - : logical 1 or logical 0

Figure 1-6. Pin Functions Section

## 1.1.6 Hardware Operation

Timing charts describe hardware operations required to control external circuits (figure 1-7).

### 4.1.5 Hardware Operation

Figure 4-3 shows the interface timing chart for the HD6301Y0 and external memory (HN27C64, HM6117).

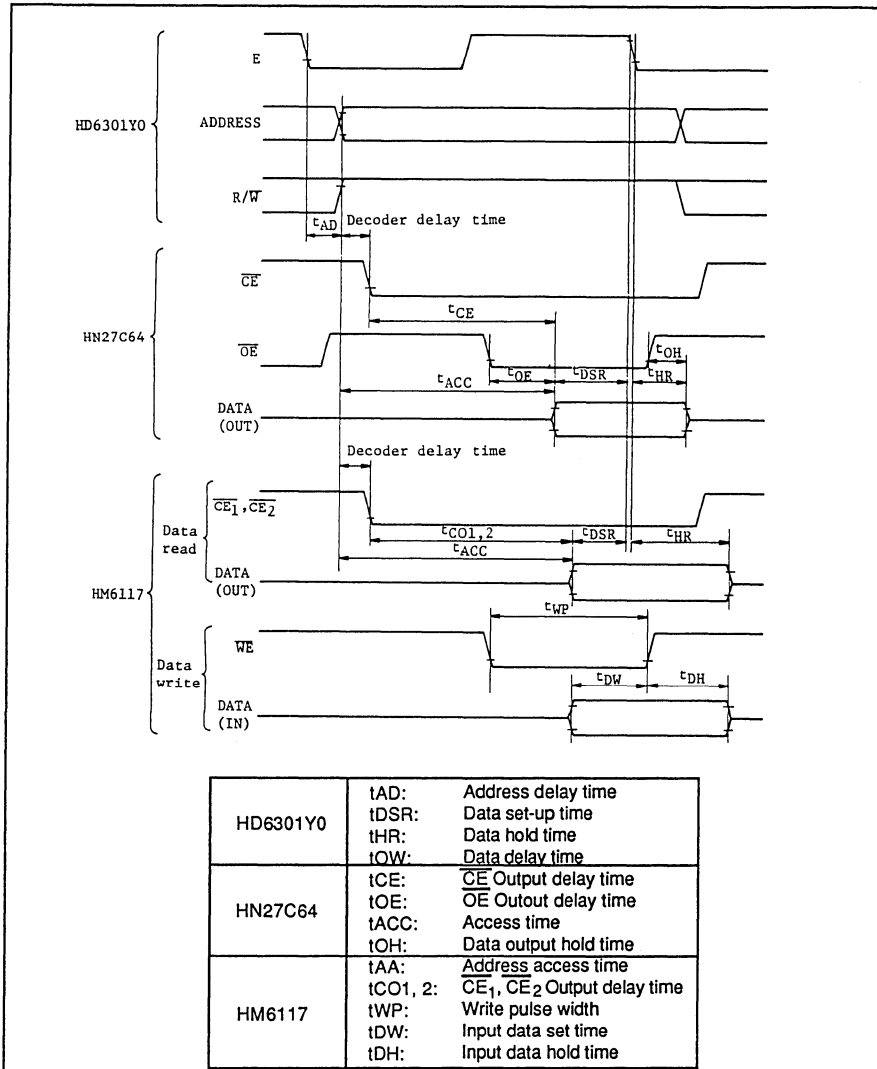


Figure 4-3. Interface Timing Chart for HD6301Y0 and External Memory

### Figure 1-7. Hardware Operation

## 1.2 Software Section

### 1.2.1 Program Module Configuration

"Program Module Configuration" describes the program modules for controlling the hardware specified in the hardware section (figure 1-8). Each module in the program module configuration figure has module number (1-N) in the upper right handcorner. The module number of the main assembly language program is 0.

#### 4.2.1 Program Module Configuration

Figure 4-4 shows the program module configuration which displays data input from a console typewriter, using the circuit in Figure 4-1.

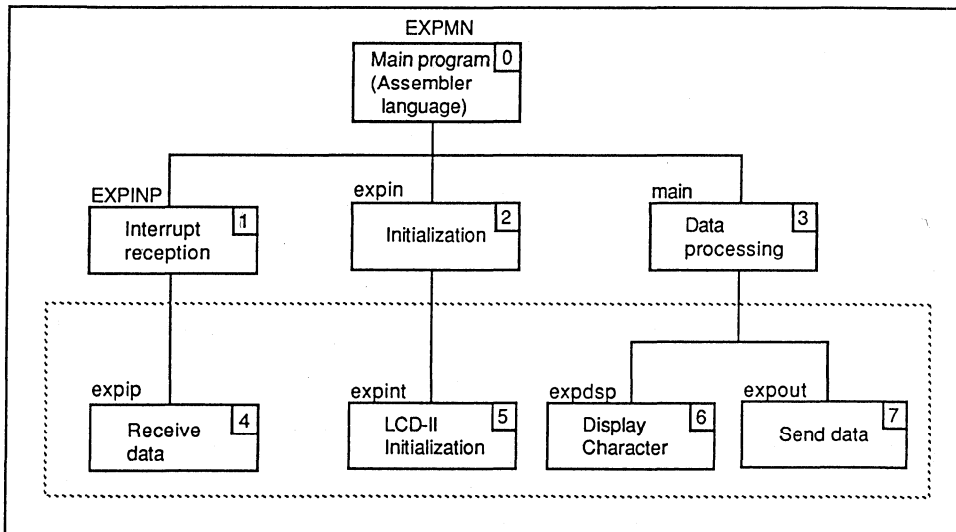


Figure 4-4. Program Module Configuration

Refer to Section 4.3 "Program Module Description" discusses these modules for details.

Figure 1-8. Program Module Configuration Section

## 1.2.2 Program Module Functions

"Program Module Functions" explains the functions of each program module presented in the program module configuration. "No." in the table matches the module number in the program module configuration (figure 1-9).

### 4.2.2 Program Module Functions

Table 4-2 summaries the program module functions.

Table 4-2. Program Module Functions

No.	Program Module Name	Library Function	Function	Language
0	Main program	EXPMN	Initializes instructions, such as ORG, LDS, and CLI, which do not exist in C. Calls expin function and main function	ASM
1	Interrupt reception	EXPINP	Receives and processes IRQ interrupt	ASM
2	Initialization	expin	Initializes global variables, FIA, ACIA, and LCD-II	C
3	Data processing	main	Displays key data, input from console typewriter, on liquid crystal display (H2571) and prints the data on the console typewriter	C
4	Receive data	expip	Receives key data from the console typewriter through an IRQ interrupt	C
5	LCD-II initialization	expint	Initializes LCD-II	C
6	Display Character	expdsp	Displays characters on LCD	C
7	Send data	expout	Sends data to console typewriter	C

Note: C: C Language Program  
ASM: Assembly Language Program

Figure 1-9. Program Module Functions Section

### 1.2.3 Program Module Sample Application (Main Program)

"Program Module Sample Application (Main Program)" explains a sample program in flowchart format using the program module described in the program module configuration (figure 1-10).

#### 4.2.3 Program Module Sample Application (Main Program)

The flowchart in Figure 4-5 is an example of the execution sequence of the program module in Figure 4-4 when it displays key data input from a console typewriter on a liquid crystal display and prints the data on the console typewriter.

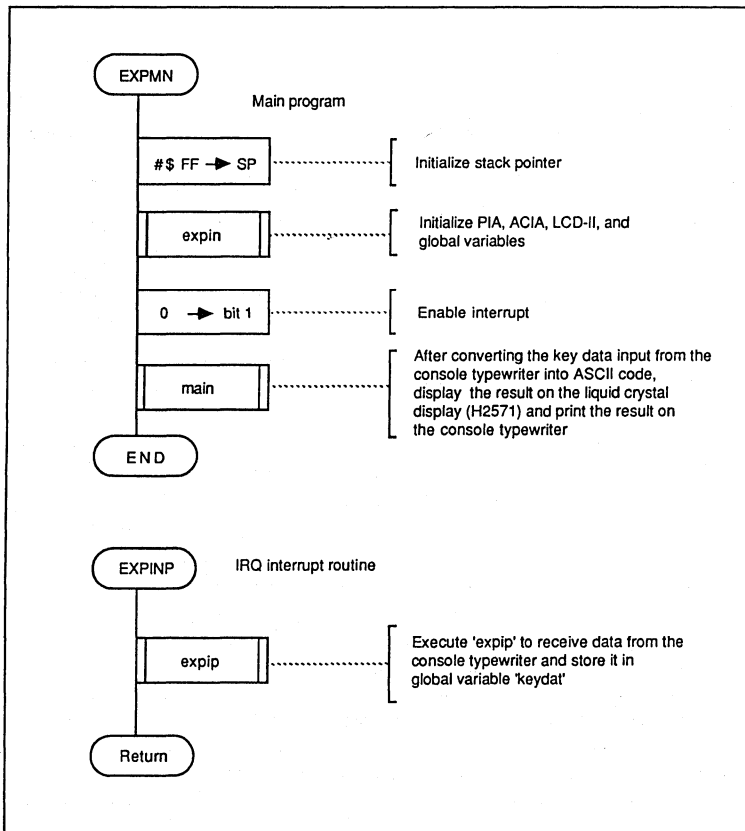


Figure 4-5. Program Module Flowchart

Figure 1-10. Program Module Sample Application (Main Program)

Further figures described the flow of program modules shown in general flowchart of main program (figure 1-11).

Figure 4-6 shows the execution sequence of C language program 'expin'. In 'expin', key data input from console typewriter is displayed on the liquid crystal display (H2571) and printed on the console typewriter.

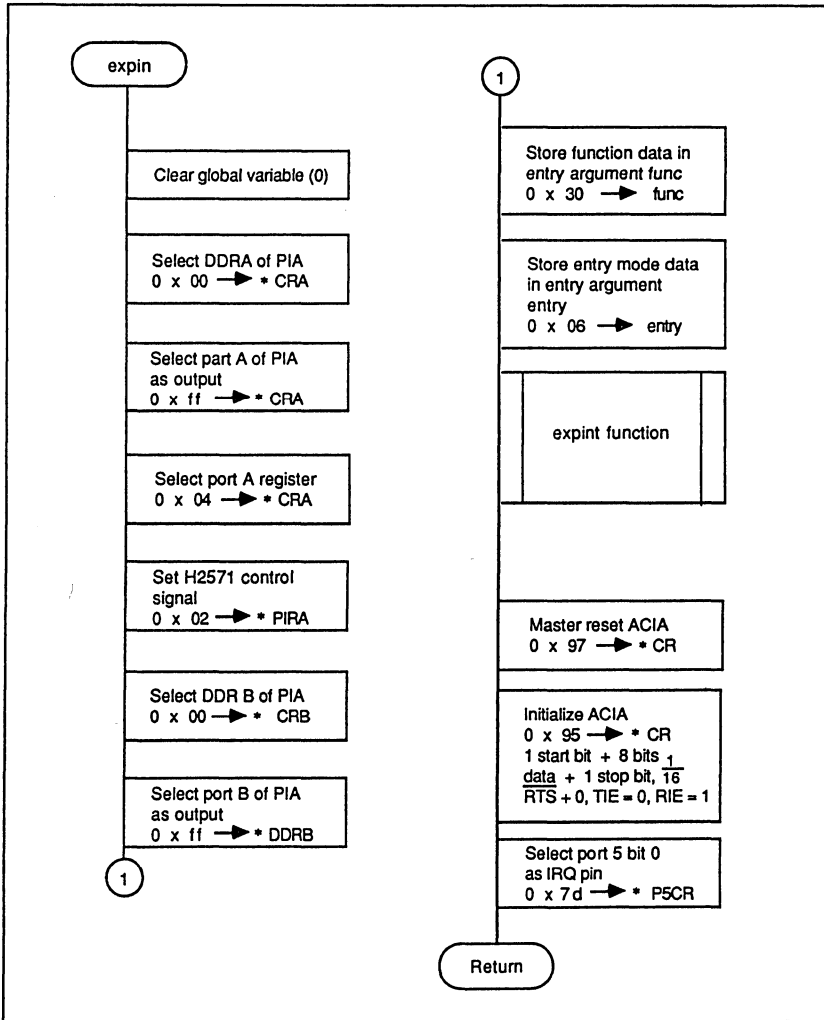


Figure 4-6. Program Module Flowchart

Figure 1-11. Sample Application (Other Routines)

# 1.3 Program Module Section

The program module detailed description format is shown in figure 1-12.

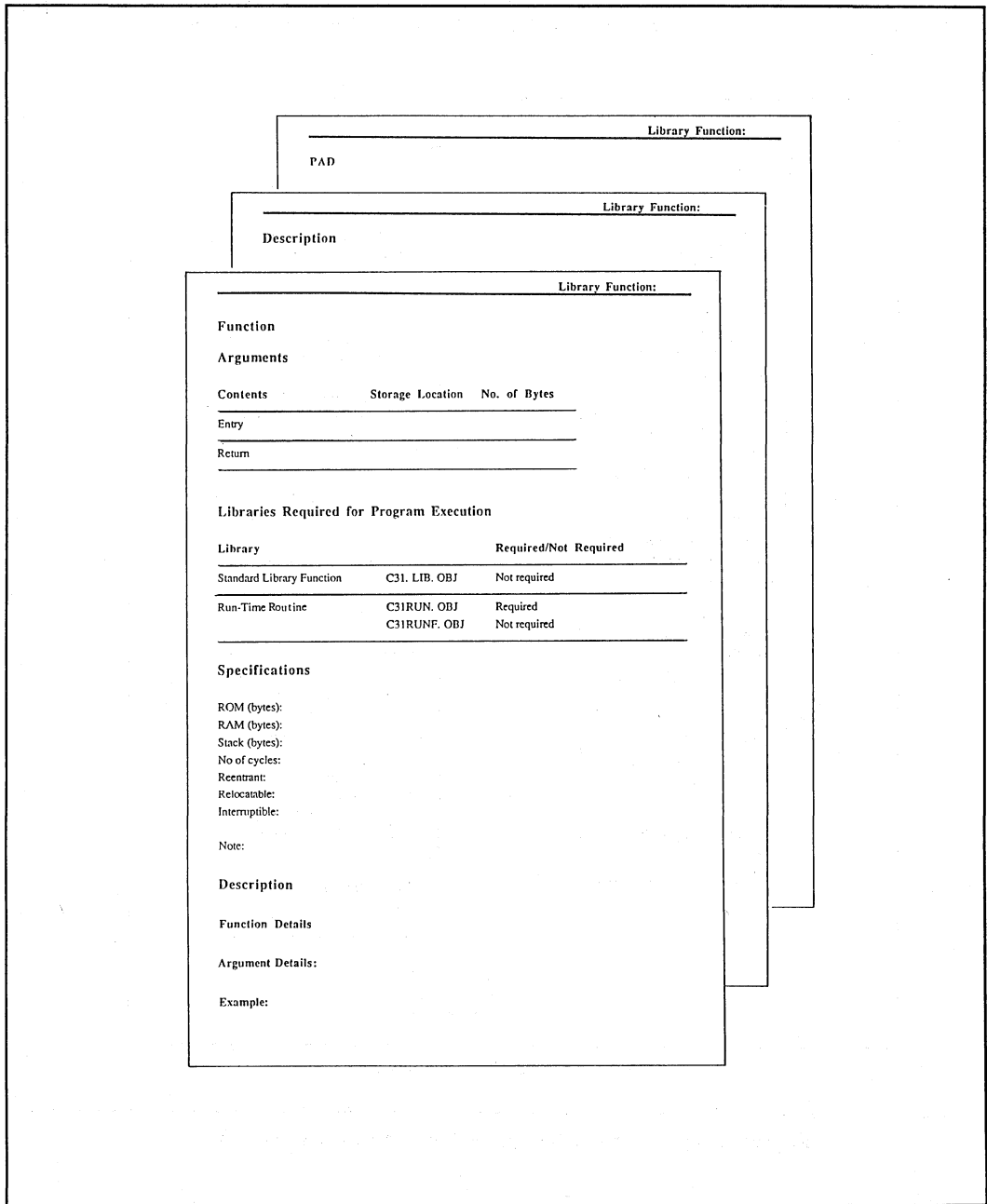


Figure 1-12. Program Module Format

### 1.3.1 Page Heading

Each page in this section is headed by the program modules name and the library function by which it is called (figure 1-12).

### 1.3.2 Function

"Function" describes the program module functions (figure 1-13).

Function
The receive data module receives data from console typewriter and stores key data in global variable 'keydat'.

Figure 1-13. Function Section

### 1.3.3 Arguments

"Arguments" describe both entry and return arguments for the program module (figure 1-14).

- Contents: The contents of the arguments.
- Storage location: Location of arguments (global variables).
- No. of bytes: The argument length.

Contents	Storage Location	No. of Bytes
Entry	—	—
Returns		
Received data (ASCII code)	keydat (global variable)	2
Received data flag	keydrf (global variable)	2

Figure 1-14. Arguments Section



### 1.3.4 Libraries Required for Program Execution

"Libraries Required for Program Execution" describes the libraries which must be linked for the program to execute (figure 1-15).

Standard Library Functions (C31LIB. OBJ): Prior to using a program, the library functions and the subroutines used by the library functions must be linked. The library functions are stored in "C31LIB, OBJ".

Run-Time Routines (C31RUN. OBJ, C31RUNF. OBJ): Run-time routines are called from the object programs, generated by the compiler, during execution.

The following two files are supplied:

- C31RUN. OBJ
- C31RUNF. OBJ

Link "C31RUN. OBJ" when only integers are used in the module or "C31RUNF. OBJ" when integers and floating point numbers are used. These files should not both be linked.

<p>The module in this example does not use the standard library functions. C31LIB. OBJ should not be linked, but run-time routine, C31RUN. OBJ should be linked, since it uses only integers.</p>		
<b>Library</b>		<b>Required/Not Required</b>
Standard Library Function	C31LIB. OBJ	Not required
Run-Time Routine	C31RUN. OBJ	Required
	C31RUNF. OBJ	Not required

**Figure 1-15. Libraries Required for Program Execution Section**

### 1.3.5 Specifications

"Specifications" describes the program module specifications as follows (figure 1-16):

- ROM (bytes): Amount of ROM used by the program module.
- RAM (bytes): Amount of RAM used by the program module. RAM used for stack is not included.
- Stack (bytes): Size of the stack used by the program module. The stack area used by a subroutine called from a user program is not included. When a program module is executed, memory for the stack must be reserved in RAM.
- No. of cycles: Maximum number of execution cycles required by the program module, calculated as follows:  
Execution time (s) = Number of cycles x Cycle time  
Cycle time (s) = 4/(External oscillator (Hz))
- Reentrant: Indicates whether a program module has a structure which can be called from two or more routines at the same time.
- Relocatable: Indicates whether a program module can be located in any memory space.
- Interruptible: Indicates whether the CPU will continue with normal execution after servicing an interrupt routine. If not, inhibit interrupts before and after the program module is called.

Specifications	
ROM (bytes):	48
RAM (bytes):	4
Stack (bytes):	0
No of cycles:	63 (Note)
Reentrant:	No
Relocatable:	No
Interruptible:	No

Note: Ox indicates a hexadecimal number in C.

Figure 1-16. Specifications Section

### 1.3.6 Description

"Description" described the functions of the program module in detail and the precautions to follow (figure 1-17).

Function Details: "Function Details" gives an execution example and detailed functions of the program module.

User Notes: "User Notes" explains notes and limitations on executing the program module.

Be sure to read these notes when using the program modules.

**Argument details:** Global variable 'keydat' contains 1-byte of key data (ASCII) from the console typewriter. Global variable 'keydrf' is a flag indicating that data has been received. Table 4-3 shows flag functions.

**Example:** Figure 4-8 shows an example of program module 'expip' execution. If key "a" on the console typewriter is pressed as shown in ①, the received data is put in the key data buffer and oxff is stored in 'keydrf' as shown in ②.

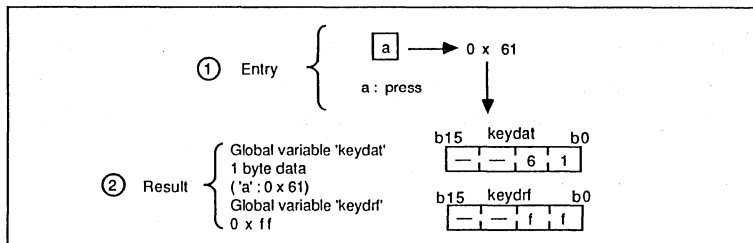


Figure 4-8. Program Module expip Execution Example

Table 4-3. Flag Functions

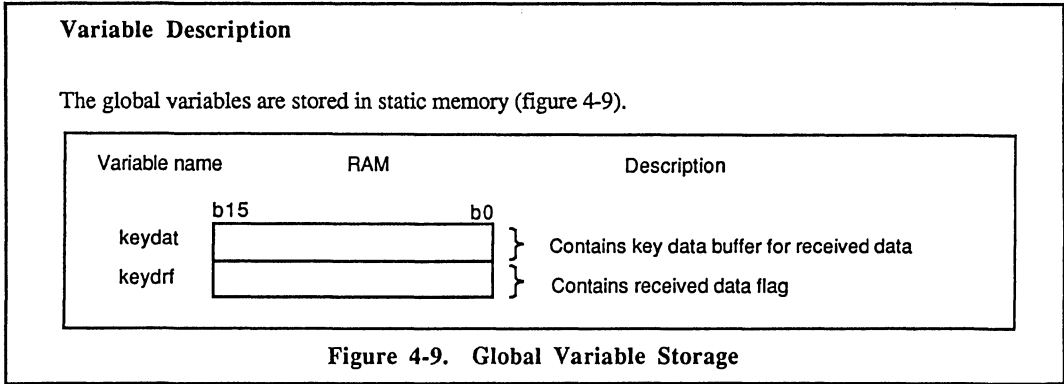
Variable Name	Flag	Indicates
Keydrf	0x00	No data has been received
	0xff	Data has been received and stored in buffer

#### User Notes

1. Initialize ACIA because ACIA is controlled by the microcontroller external extension. After initialization ACIA can receive data from the console typewriter.
2. Clear bit I and enables interrupt for  $\overline{IRQ}$  interrupt.

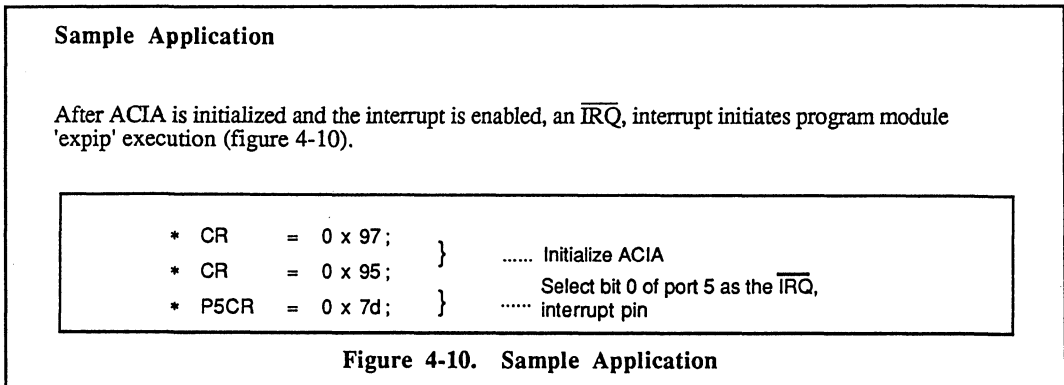
Figure 1-17. Description Section

**Variable Description:** "Variable Description" explains the names and functions of the global variables used by the program module (figure 1-18).



**Figure 1-18. Variable Description Section**

**Sample Application:** "Sample Application" gives an example of the program module execution (figure 1-19).



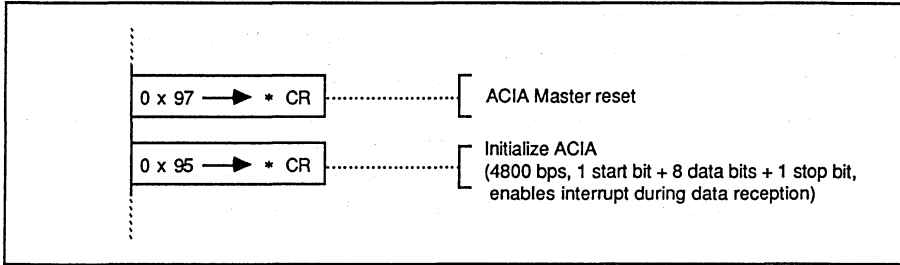
**Figure 1-19. Sample Application Example**

**Basic Operation:** "Basic Operation" explains the basic operation of the program module (figure 1-20).

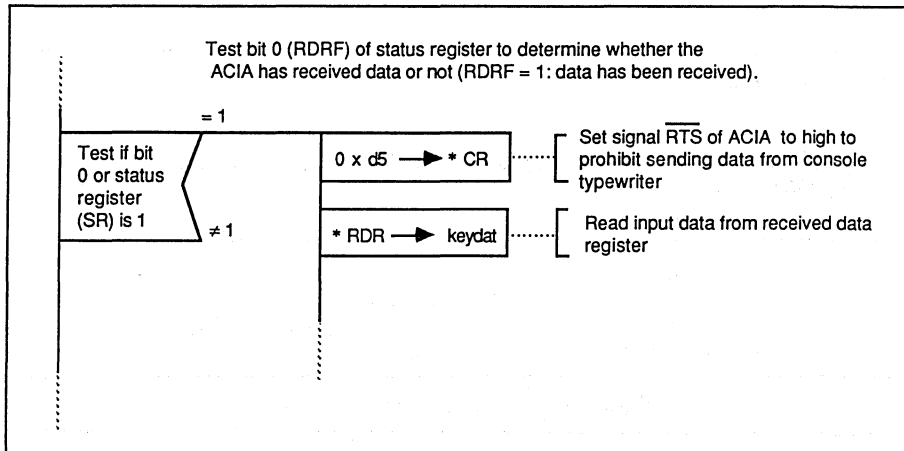
### Basic Operation

Figures 4-11 and 4-12 show ACIA control. Figure 4-11 shows ACIA initialization. Figure 4-12 shows how received data is read after an interrupt.

Note that this control method applies to the system in Figure 4-1 and memory map in Figure 4-2.



**Figure 4-11. ACIA Control (Initialization)**



**Figure 4-12. ACIA Control (Receiving Serial Data)**

When data reception has been completed, set signal  $\overline{\text{RTS}}$  to high to prohibit next data transfer. Finally, store received data from RDR of ACIA in key data buffer.

### Figure 1-20. Basic Operation Section

### 1.3.7 PAD

"PAD" described program flow using a PAD diagram (figure 1-21).

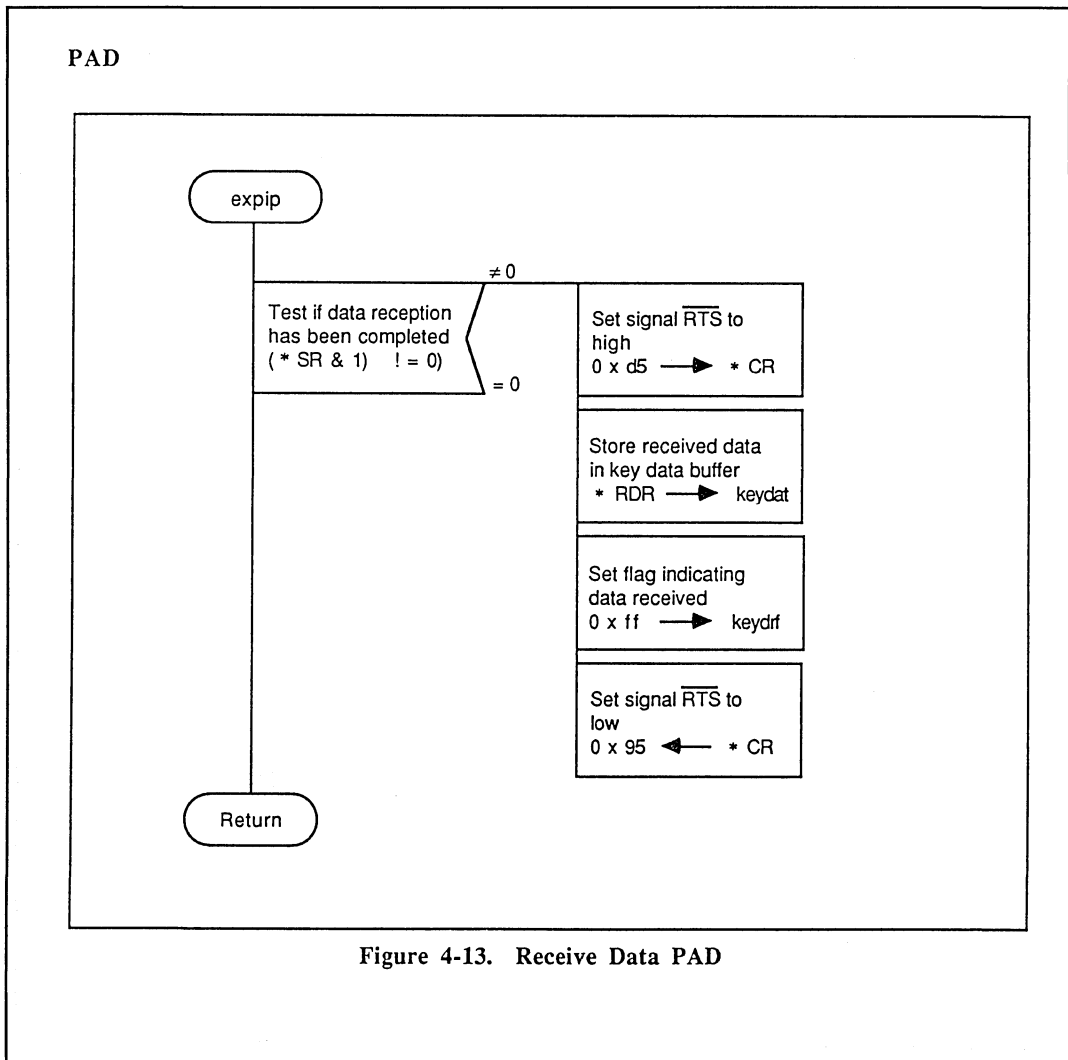


Figure 1-21. PAD Section

## 1.4 Subroutine Section

Figure 1-22 shows the subroutine format. This section describes the subroutines used by the program modules.

Library Function:
Function
Basic Operation
PAD
Program Module That Uses This Function
<div style="border: 1px solid black; height: 150px; width: 100%;"></div>

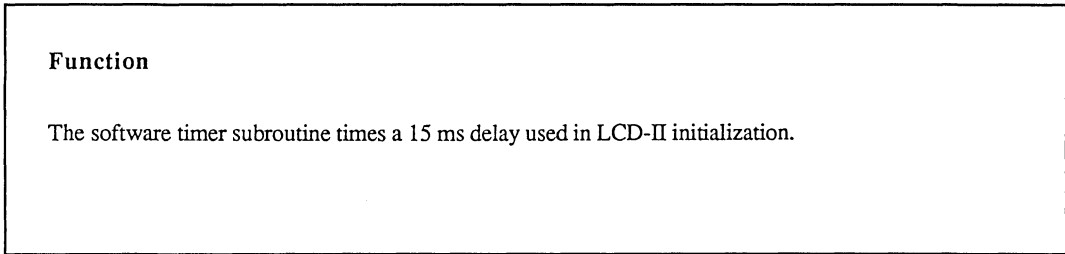
Figure 1-22. Subroutine Format

### 1.4.1 Page Heading

Each page in this section is headed by the subroutines same and the library function by which it is called (figure 1-22).

### 1.4.2 Function

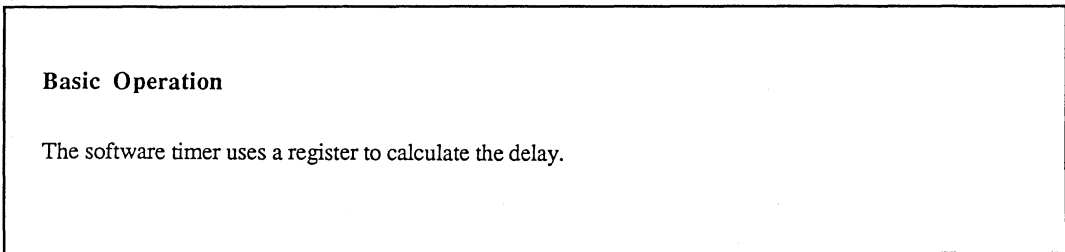
"Function" describes the subroutine functions (figure 1-23).



**Figure 1-23. Function Section**

### 1.4.3 Basic Operation

"Basic Operation" describes the basic subroutine operations (figure 1-24).

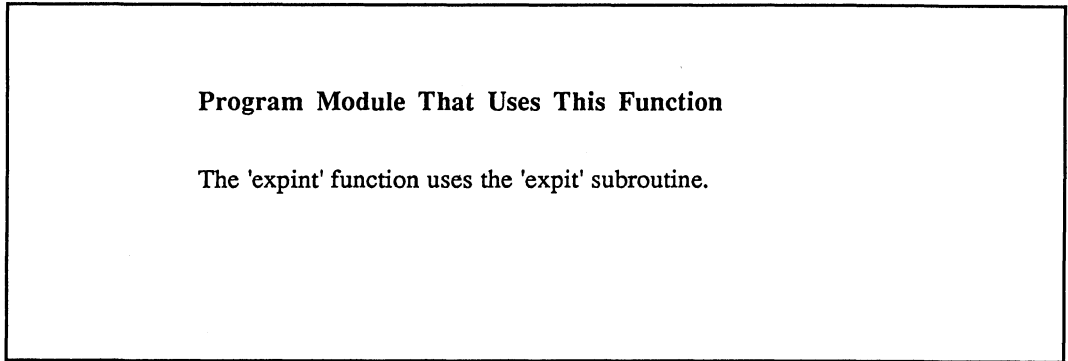


**Figure 1-24. Basic Operation Section**



### 1.4.4 Program Modules That Use This Function

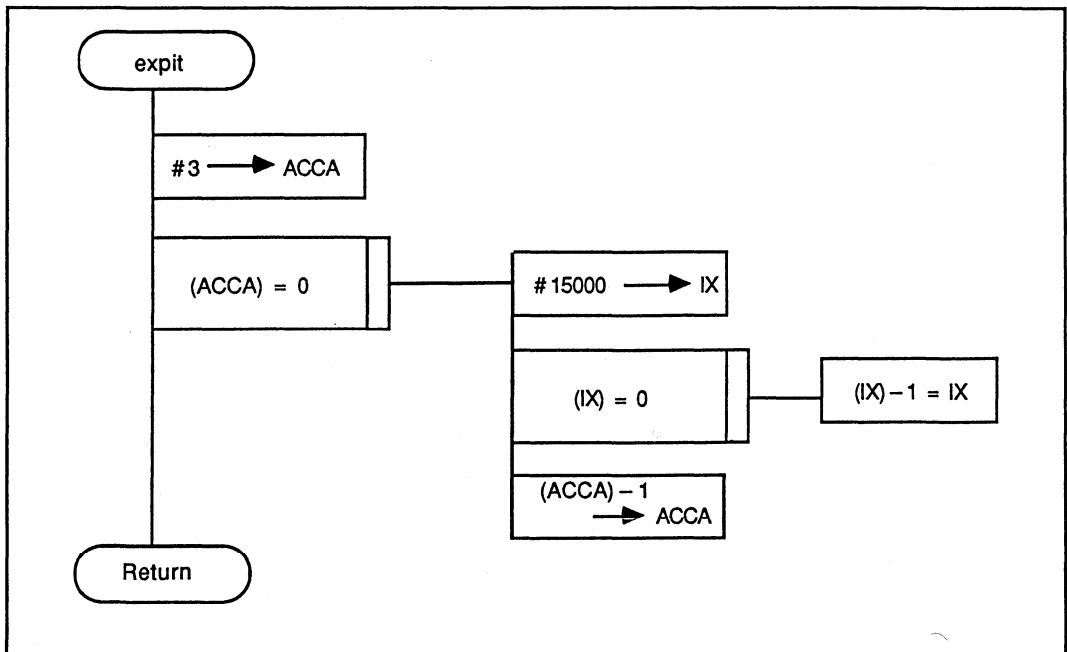
"Program Module That Use This Function" specifies which modules call this subroutine (figure 1-25).



**Figure 1-25. Program Modules That Use This Function Section**

### 1.4.5 PAD

"PAD" describes the flow of the subroutine using a PAD (figure 1-26).



**Figure 1-26. PAD Section**



Figure 1-27 shows the following parts:

- (a) Main assembly program title  
The title "MAIN PROGRAM" is always used followed by the entry point label in parentheses
- (b) Entry point label
- (c) Common subroutine title
- (d) Entry point label (library function)
- (e) The title is always "VECTOR ADDRESSES".
- (f) End of entire program can be moved if necessary.
- (g) Program module call.

### 1.5.2 C Source Listing

Program Symbol Definitions: The symbols used in a program module or common subroutine are defined as follows (figure 1-28):

- (a) The title is always "DECLARATION OF DEFINE".
- (b) Symbol definitions.

```
(a) { *.....*DECLARATION OF DEFINE*.....*/*
    /*
(b) #define P5CR      ((char*)0x14)      /* Port5 control register */
    #define DDRA     ((char*)0xA000)    /* Data direction register A(PIA) */
    #define CRA      ((char*)0xA001)    /* Control register A(PIA) */
    #define DDRB     ((char*)0xA002)    /* Data direction register B(PIA) */
    #define CRB      ((char*)0xA003)    /* Control register B(PIA) */
    #define PIRA     DDRA               /* Peripheral register A(PIA) */
    #define PIRB     DDRB               /* Peripheral register B(PIA) */
    #define CR       ((char*)0xC000)    /* Control register (ACIA) */
    #define SR       CR                 /* Status register (ACIA) */
    #define RDR      ((char*)0xC001)    /* Receive data register (ACIA) */
    #define TDR      RDR                /* Transmit data register (ACIA) */
```

Figure 1-28. Program Symbol Definitions

**Declaration of Global Variables:** Global variables used in program modules and common subroutines are defined as follows (figure 1-29):

- (a) The title is always "DECLARATION OF GLOBAL VARIABLES".
- (b) Declaration of global variables

```

(a) { /* .....DECLARATION OF GLOBAL VARIABLES..... */
      /* ..... */
(b) static direct int outdat; /* Transmit data */
      static direct int dspdat; /* Display data */
      static direct int keydrf; /* Flag of receive data */
      static direct int keydat; /* Receive data */
      static direct int tncnt; /* Counter for initializing LCD-II */
      static direct int func; /* Function data */
      static direct int entry; /* Entry mode data */

```

**Figure 1-29. Declaration of Global Variables**

**C Language Module:** The C language module listing is shown next. Figure 1-30 is an example of C the language main function routine:

- (a) Main Function title
- (b) Library Function

```

(a) { /* .....
      /* .....
      MAIN ROUTINE : MAIN (DISPLAY INPUT DATA FROM CONSOLE ON BOTH LCD-2
      AND CONSOLE)
      /* .....
      /* .....
(b) main() /* Display input data from console on both LCD-II and console */
      {
          while (1) { /* Continuous loop */
              if (keydrf!=0) { /* Test if data is received */
                  if (keydat>='a' && keydat<='z')
                      keydat-=0x20; /* Change lower case to upper */
                  keydrf=0; /* Clear flag of receive data */
                  *CR=0x95; /* Set RTS=low */
                  outdat=dspdat=keydat; /* Set output data in area */
                  expout(); /* Transmit data to console */
                  expdsp(); /* Display characters on LCD-II */
              }
          }
      }

```

**Figure 1-30. C Language Module**

**Program Module:** The program module listing is divided into separate functions (figure 1-31):

- (a) Program module title
- (b) Library Function

```

.....
/*
/*      NAME : EXPIP (RECEIVE DATA)
/*
.....
(a) /*
/*      ENTRY   : NOTHING
/*      RETURNS : KEYDAT (RECEIVED DATA)
/*             : KEYDRF (RECEIVED FLAG)
/*
.....
(b) expip()
    if ((*SR&1)!=0) {
        *CR=0xd5;          /* Test if data is received */
        keydat = *RDR;     /* Set RTS-high */
        keydrf=0xff;      /* Set receive data */
        *CR=0x95;         /* 0xff if receive data is set */
                          /* Set RTS-low */
    }
.....

```

**Figure 1-31. Program Module**

**Common Subroutine:** Next, common subroutines used in the program module are listed (figure 1-32):

- (a) Subroutine title
- (b) Library Function

```

.....
/*
/*      NAME : EXPIN (INITIALIZE PIA,ACIA AND LCD-2)
/*
.....
(a) /*
/*
.....
(b) expin()
    outdat=dspdat=keydrf=keydat=tnent=func=entry=0; /* Initialize */
    *CRA =0x00;          /* Select data direction register A */
    *DDRA=0xff;         /* Select port A as output */
    *CRA =0x04;         /* Select peripheral register A */
    *PIRA=0x02;         /* Set RS=0, R/W=1, E=0 */
    *CRB =0x00;         /* Select data direction register B */
    *DDRB=0xff;         /* Select port B as output */
    *CRB =0x04;         /* Select peripheral register B */
    func=0x30;          /* Set function data */
    entry=0x06;         /* Set entry mode data */
    expint();           /* Initialize LCD-II */
    setins(0x0e);       /* Set instruction to LCD-II */
    *CR=0x97;           /* Master reset of ACIA */
    *CR=0x95;           /* Initialize ACIA */
    *P5CR=0x7d;        /* Initialize port 5 */
.....

```

**Figure 1-32. Common Subroutines**

### 1.5.3 Output Object Listing of C Compiler

6301 C compiler outputs an object code listing in 6301 assembler language (figure 1-33):

- (a) Macro definition generated by the compiler
- (b) Global variable definition
- (c) Compilation result (assembly language output listing) of a C language source program

```

*** CP/M-68K 6301/6801/6800 CROSS MACROASSEMBLER V1.2 ***
ERR  SEQ  LOC  OBJECT      PROGRAM EXPC
      00001          NAM      EXPC
      00002          OPT      REL
      00003          MSEX     MACR
      00004          CLRA
      00005          TSTB
      00006          BPL  \.0
      00007          COMA
      00008          \.0 EQU *
      00009          ENDM
      00010          MLBRA   MACR
      00011          JMP  \0
      00012          ENDM
      00013          MLBSR   MACR
      00014          JSR  \0
      00015          ENDM
      00016          MLBEQ   MACR
      00017          BNE  \.0
      00018          JMP  \0
      00019          \.0 EQU *
      00020          ENDM
      00061          MLBCS   MACR
      00062          BCC  \.0
      00063          JMP  \0
      00064          \.0 EQU *
      00065          ENDM
      00066B 0000          BSCT
      00067B 0000          0002 A OUTDAT BSZ 2
      00068B 0002          BSCT
      00069B 0002          0002 A DSPDAT BSZ 2
      00070B 0004          BSCT
      00071B 0004          0002 A KEYDRF BSZ 2
      00072B 0006          BSCT
      00073B 0006          0002 A KEYDAT BSZ 2
      00074B 0008          BSCT
      00075B 0008          0002 A TNCNT BSZ 2
      00076B 000A          BSCT
      00077B 000A          0002 A FUNC BSZ 2
      00078B 000C          BSCT
      00079B 000C          0002 A ENTRY BSZ 2
      00080F 0000          PSCT
      00081F 0000 20 31 0033  BRA  .SA002
      00082F 0002 DC 04 B  .SA003 LDD  KEYDRF
      00083F 0004 27 2D 0033  BEQ  .SA004
      00084F 0006 DE 06 B      LDX  KEYDAT
      00085F 0008 8C 0061 A     CPX  #97
      00086F 000B 2D 0E 001B    BLT  .SA005
      00087F 000D DE 06 B      LDX  KEYDAT
      00088F 000F 8C 007A A     CPX  #122
      00089F 0012 2E 07 001B    BGT  .SA005
      00090F 0014 DC 08 B      LDD  KEYDAT
      00091B 0016 8C 0000 A     CDB  *??

```

Figure 1-33. Output Object Listing of C Compiler

## 1.5.4 Linkage Listing

**Linkage Command Listing:** The linkage command listing is a sample command main assembly program and a C language sequence for linking and executing a program (figure 1-34):

- ① Load relocatable object module for linking modules
- ② Divide into sections, taking the system memory map into consideration
  - STRP Program Section
  - STRB Base Section
  - STRD Data Section
- ③ Output map listing and symbol listing using OPT command
- ④ Input EXEC command to execute linkage editor

```
*** HMCS6800 CROSS LINKAGE EDITOR VER 1.2 ***
LOAD=B:EXPMN.OBJ,B:EXPC.OBJ,C31RUN.OBJ .....①
STRP=$F000 } .....②
STRB=$60   } .....②
STRD=$40   } .....②
OPT=MAP.SYM .....③
EXEC .....④
```

**Figure 1-34. Linkage Command Listing**

**Undefined Global Symbol Listing:** The compiler outputs an undefined global symbol listing (figure 1-35): (If linkage command errors occur, an ERROR message is issued.)

- ① Undefined Symbol's name
- ② Section containing undefined symbols
- ③ Undefined symbol relocatable object module name
- ④ Total number of undefined symbols

```
*** HMCS6800 CROSS LINKAGE EDITOR VER 1.2 ***
*** UNDEFINED SYMBOLS ***
NAME ① SECTION ② MODULE NAME ③
.ERROR ( )
UNDEFINED SYMBOL = 1 ④ (Note)
```

**Figure 1-35. Undefined Global Symbol Listing**

**Map Listing:** The map listing includes the following (figure 1-36):

- ① Section Load map  
Prints the size of each section (number of bytes), starting address, ending address, and size of common section with name (number of bytes)
- ② Module load map  
Prints module name, load address of base section, data section, and program section for each module
- ③ Common load map  
Prints name, section, size (number of bytes), starting address, and total number of common sections with names

```
*** HMCS6800 CROSS LINKAGE EDITOR  VER 1.2 ***
*** MAP LIST ***
** SECTION LOAD MAP ①
   SECTION  SIZE  START  END  COMMON-SIZE
     A      0016  FFEA  FFFF
     B      000E  0060  006D      0000
     C      0000
     D      0004  0040  0043      0000
     P      07DA  F000  F7D9      0000
** MODULE LOAD MAP ②
   NAME     BSCT  DSCT  PSCT
           EXPC  0060          F000
           F01A
           0040  F208
** COMMON LOAD MAP ③
   NAME     SECTION  SIZE  START
COMMON =    0
```

**Figure 1-36. Map Listing**



Global Symbol Table Listing: Finally, the compiler prints global symbol names, global variable section address, symbol module, and total number of global symbols (figure 1-37):

```

*** HMC56800 CROSS LINKAGE EDITOR  VER 1.2 ***
*** DEFINED SYMBOLS ***
NAME SECTION START MODULE NAME
.DDADD P F7D9 ( )
.DDCMP P F7D9 ( )
.DDEEC P F7D9 ( )
.DDDIV P F7D9 ( )
.DDINC P F7D9 ( )
.DDMOV P F7D9 ( )
.DDMUL P F7D9 ( )
.DDNEG P F7D9 ( )
.DDSTK P F7D9 ( )
.DDSUB P F7D9 ( )
.DDSTOF P F7D9 ( )
.DDSTOI P F7D9 ( )
.DDSTOL P F7D9 ( )
.DDSTST P F7D9 ( )
.DDFDEC P F7D9 ( )
.DDFINC P F7D9 ( )
.DDFMOV P F7D9 ( )
.DDFREG D 0040 ( )
.DDFTOD P F7D9 ( )
.DDFSTST P F7D9 ( )
.DDIASL P F27D ( )
.DDIASR P F292 ( )
.DSIDIV P F23F ( )
.DSIMOD P F2BC ( )
.DSIMUL P F208 ( )
.DDITOD P F7D9 ( )
.DDITOL P F51D ( )
.DDLADD P F32F ( )
.DDLAND P F432 ( )
.DDLBIT P F600 ( )
.DDLCMP P F4BF ( )
.DDLCPL P F501 ( )
.DDLDEC P F54B ( )
.DDLDIV P F3E3 ( )
.DDLINC P F53B ( )
.DDLMOD P F40A ( )
.DDLMOV P F30B ( )
.DDLMUL P F361 ( )
.DDLNEG P F4EC ( )
.DDLOR P F44D ( )
.DDLSHL P F483 ( )
.DDLSHR P F4A1 ( )
.DDLSTK P F55B ( )
.DDLSUB P F348 ( )
.DDLTOD P F7D9 ( )
.DDLTST P F576 ( )
.DDLXOR P F468 ( )
.DDSBIT P F723 ( )
.DDSW1 P F76B ( )
.DDSW2 P F79A ( )
.DDSUDIV P F25B ( )
.DDSULSR P F2A7 ( )
.DDSUMOD P F2EA ( )
.DDUTOD P F7D9 ( )
.DDUTOL P F52E ( )
EXPBSY P F16F ( EXPC )
EXPDSP P F12B ( EXPC )
EXPIN P F050 ( EXPC )
EXPINS P F1D7 ( EXPC )
EXPINT P F0F1 ( EXPC )
EXPIP P F0C0 ( EXPC )
EXFIT P F00A ( )
EXPOUT P F15F ( EXPC )
MAIN P F04D ( EXPC )
SETINS P F1FB ( EXPC )
DEFINED SYMBOL = 65

```

Figure 1-37. Global Symbol Table Listing



## 1.6 Program Module Use

This section explains the order in which the program modules described in the application note are executed.

The program shown in figure 1-38, is an example of a C language program module being called from a main assembly language program and/or vice versa.

If a user program uses an application note module, the user program should be called as shown in figure 1-38 or 1-39.

Figure 1-38 shows an example of a user program (assembly language program) in which a C language module is used as a subroutine. In the C language module, an assembly language module is used as a subroutine.

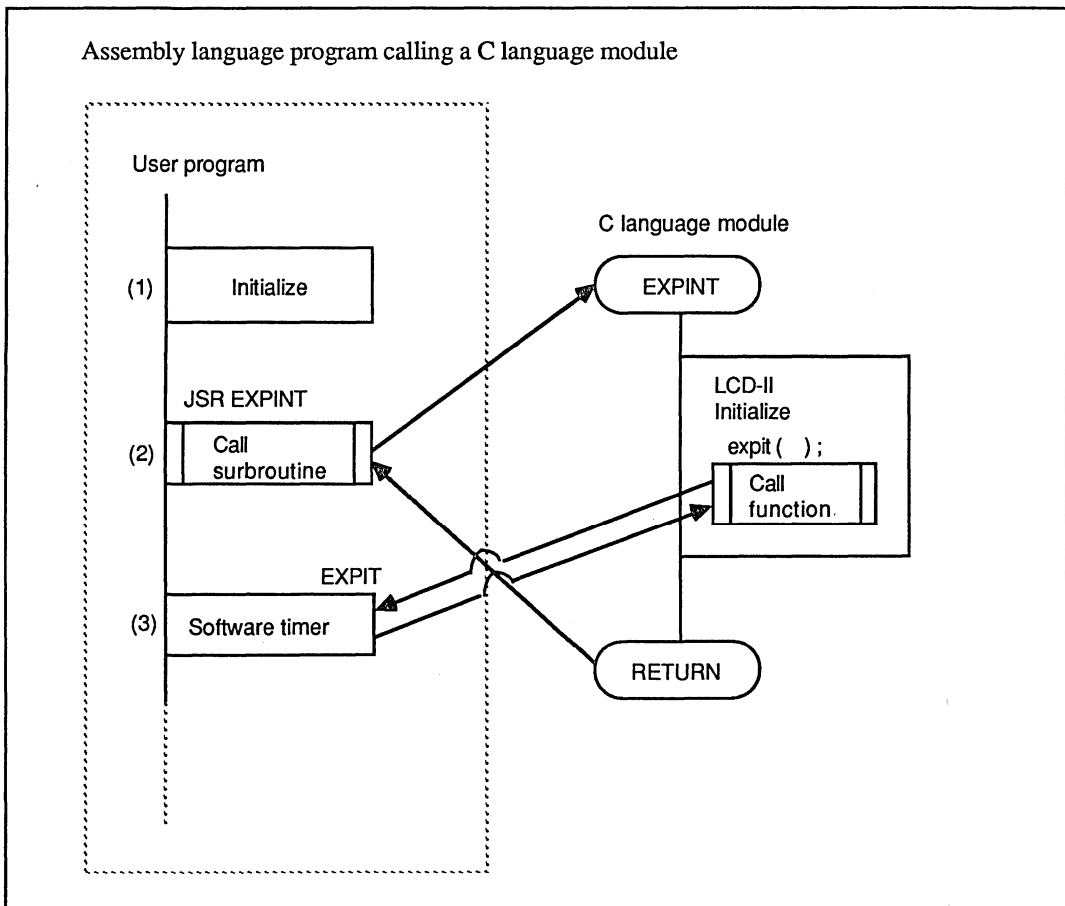
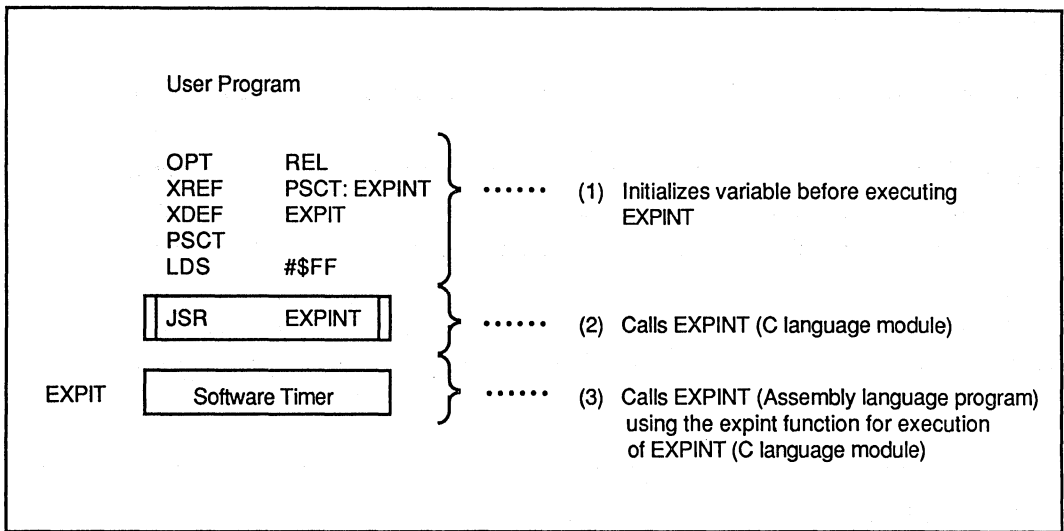


Figure 1-38. Relation between User Program and C Language Module



**Figure 1-39. Program Module Execution Example**

Figure 1-39 is an example of relocatable assembly language item initialization, for linking an assembly language program and a C language module, externally referenced name (XREF), externally define name (XDEF), input/output ports, and global variable.

The example then, calls a C language module from assembly language program.

The C language module itself calls an assembly language program.

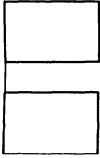
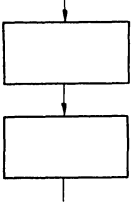
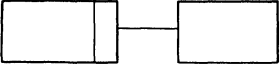
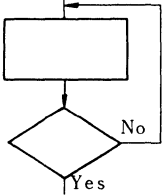
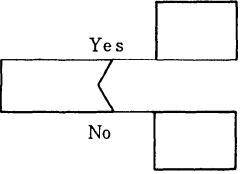
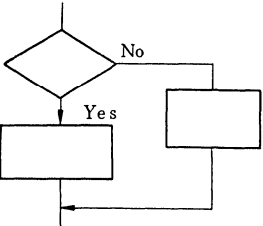
## 1.7 PAD Symbols Description

These application notes uses PAD to describe the flow of the high-level language program. PAD shows the flow of a program by using three basic forms, as shown in table 1-1.

PAD accurately expresses the flow of a program and encourages efficient programming.

It is difficult to determine whether a rhombus in a flowchart means repetition or selection, however this is easy to see in PAD.

**Table 1-1. Basic PAD Forms**

	PAD	C language example	Flowchart
Succession		<pre> ... a = 1 ; b = 2 ; ...</pre>	
Repetition		<pre> while ( a == 0 ) {     or for ( a = 0 ; a = 80 ; a++ ) { </pre>	
Selection		<pre> if ( a == 0 ) { else { </pre>	

## 1.8 Symbols

This application note uses the following Symbols and abbreviations:

DDR	=	Data Direction Register
OCR1	=	Output Compare Register 1
TCSR1	=	Timer Control/Status Register 1
RMCR	=	Transfer Rate/Mode Control Register
TRCSR	=	Tx/Rx Control Status Register
RDR	=	Receive Data Register
TDR	=	Transmit Data Register

# SECTION 2. DARLINGTON TRANSISTOR DRIVE (LED DYNAMIC DISPLAY)

## 2.1 Hardware Description

### 2.1.1 Function

The darlington transistor drive application displays data on an 8-digit x 8-segment LED using the HD6301X0 it controls the LED using a dynamic drive.

### 2.1.2 Microcontroller Applications

1. Executes an interrupt routine every 1.25 ms using the output compare function with interrupt 1 (OCT1) in a 16-bit programmable built-in timer (timer 1).
2. Supplies segment data from port 1 and digit data from port 6 when it executes the interrupt routine.
3. Port 6, directly drives the darlington transistor, controlling the digit signal.

### 2.1.3 Circuit Diagram

Figure 2-1 is the application circuit diagram.

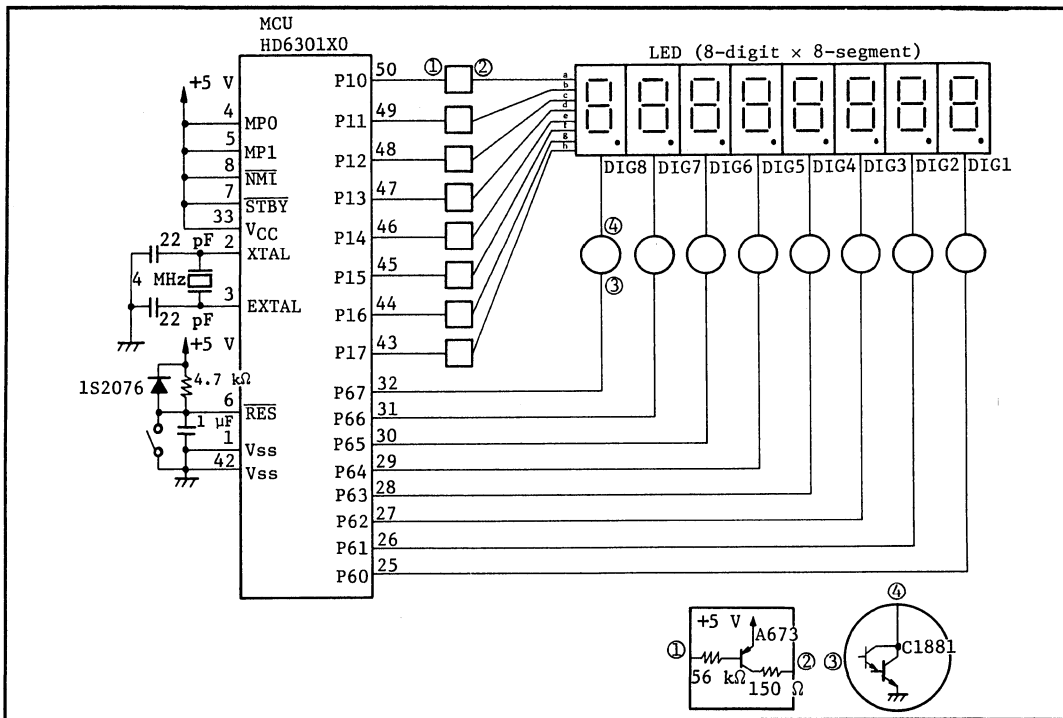
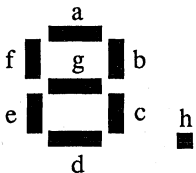


Figure 2-1. 8-Digit x 8-Segment LED Control Circuit

### 2.1.4 Pin Functions

Table 2-1 shows the pin functions at the interface between the HD6301X0 and an 8-digit x 8-segment LED.

**Table 2-1. Pin Functions**

Pin Name (HD6301X0)	Input/Output	Active Level (High or Low)	Function	Pin name (LED)	Program Label
P60	Output	High	Outputs digit data to 8-digit x 8-segment LED.	DIG1	P6DTR
P61	Output	High		DIG2	
P62	Output	High		DIG3	
P63	Output	High		DIG4	
P64	Output	High		DIG5	
P65	Output	High		DIG6	
P66	Output	High		DIG7	
P67	Output	High		DIG8	
P10	Output	Low	Outputs segment data to 8-digit x 8-segment LED.   Segment Pattern	a	P1DTR
P11	Output	Low		b	
P12	Output	Low		c	
P13	Output	Low		d	
P14	Output	Low		e	
P15	Output	Low		f	
P16	Output	Low		g	
P17	Output	Low		h	

## 2.1.5 Hardware Operation

Figure 2-2 shows the 8-digit x 8-segment dynamic LED display timing, with a frame frequency of 100 Hz and a duty rate of 1/8.

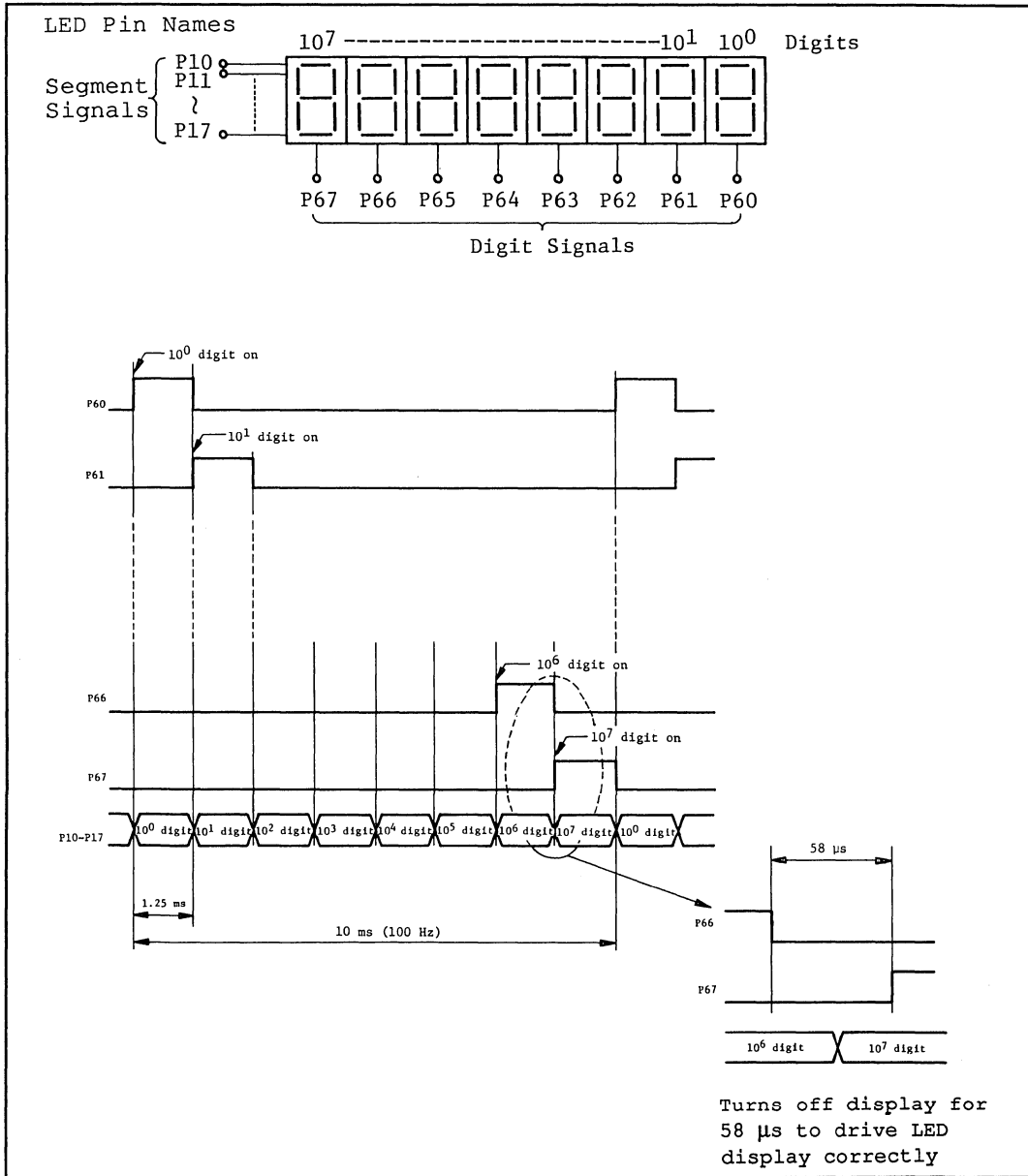


Figure 2-2. Dynamic Drive System



## 2.2 Software Description

### 2.2.1 Program Module Configuration

Figure 2-3 shows the program module configuration for an 8-digit x 8-segment LED display driver.

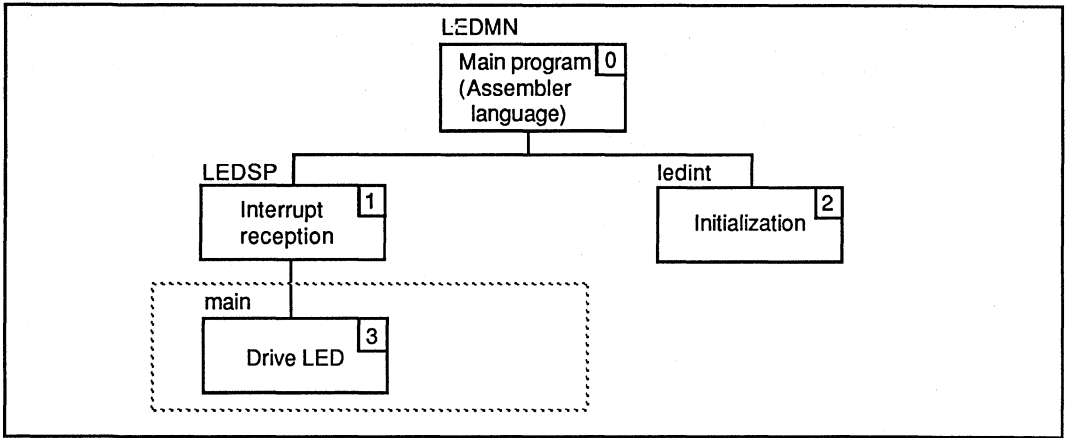


Figure 2-3. Program Module Configuration

Refer to Section 2.3, "Program Module Description" discusses these modules for details.

### 2.2.2 Program Module Functions

Table 2-2 summarizes the program module functions.

Table 2-2. Program Module Functions

No.	Program Module Name	Library Function	Function	Language
0	Main program	LEDMN	Initializes instructions, such as ORG, LDS, and CLI, which do not exist in C. Calls ledint function and main function	ASM
1	Interrupt reception	LEDSP	Receives and process OCI interrupt	ASM
2	Initialization	ledint	Initializes global variables, port, and timer	C
3	Drive LED	main	Drives 8-digit x 8-segment LED and displays data	C

Note: C: C Language Program  
 ASM: Assembly Language Program

### 2.2.3 Program Module Sample Application (Main Program)

The flowchart in Figure 2-4 is an example of an 8-digit x 8-segment LED display composed of the program modules in figure 2-3. The main program in Figure 2-4 enables the interrupt and then displays the LED display shown in Figure 2-6, using the timer interrupt.

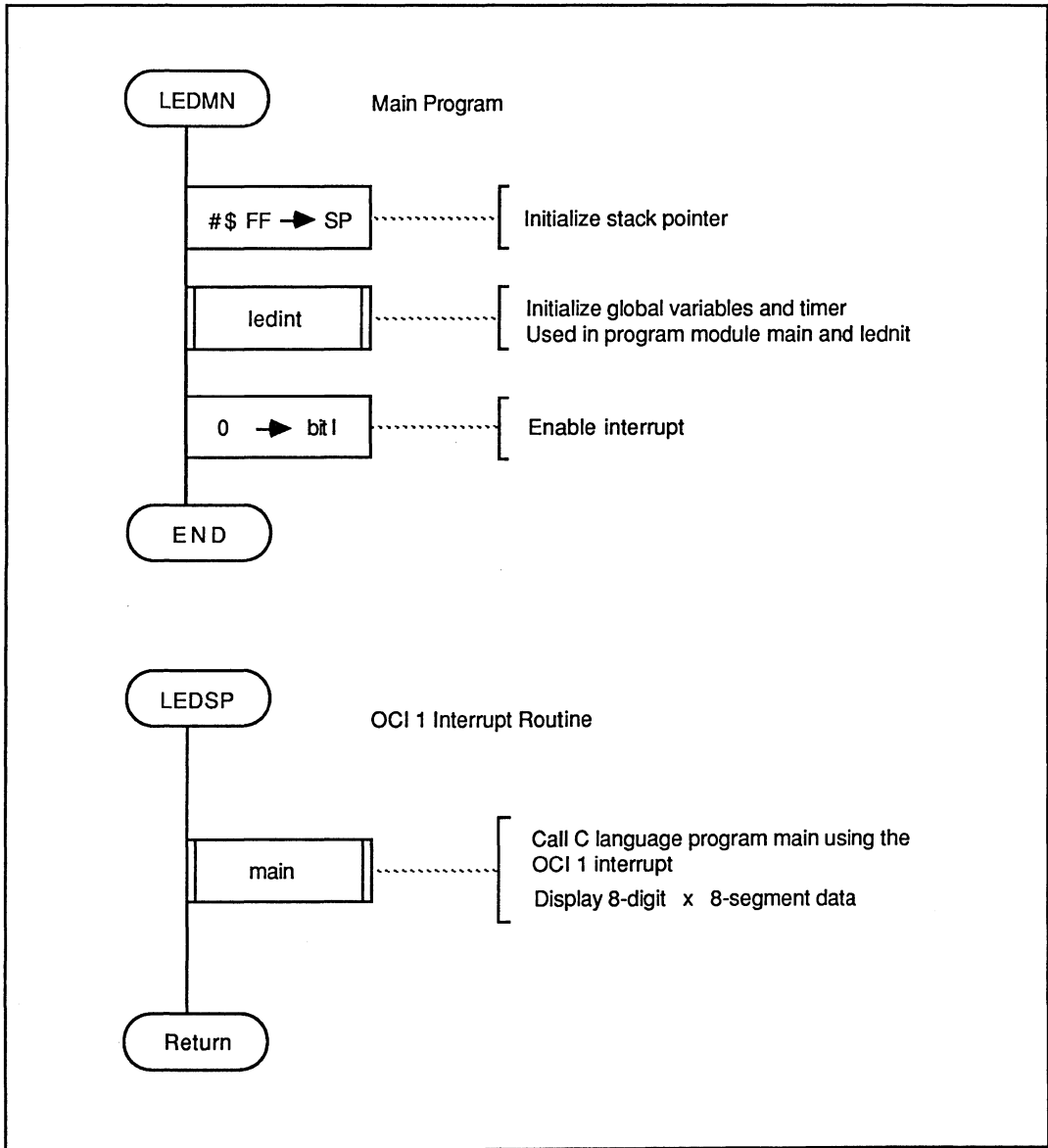


Figure 2-4. Program Module Flowchart

The C Language Program 'ledint' (figure 2-5) initializes the global variable, the timer, and the port.

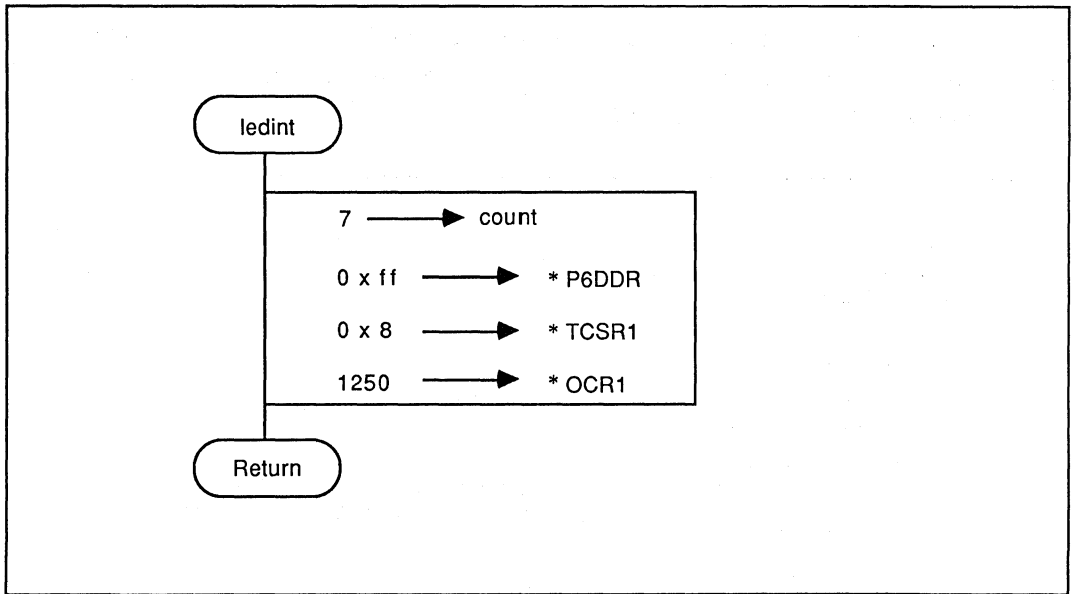


Figure 2-5. Program Module Flowchart (ledint)

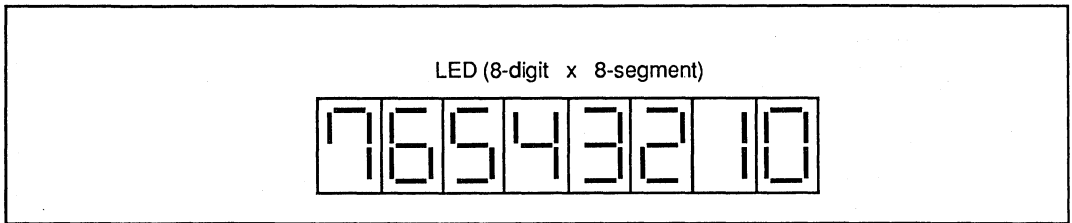


Figure 2-6. 8-Digit x 8-Segment LED Display Example

### 2.3 Program Module Description

The following pages describe the drive LED subroutine.

**Function**

The drive LED module drives an 8-digit x 8-segment LED and displays data.

**Arguments**

Contents	Storage Location	No. of Bytes	
Entry	Segment data	segd (global variable)	8
Returns	—	—	—

**Libraries Required for Program Execution**

Library		Required/Not Required
Standard Library Function	C31LIB. OBJ	Not required
Run-Time Routine	C31RUN. OBJ	Required
	C31RUNF. OBJ	Not required

**Specifications**

ROM (bytes):	89
RAM (bytes):	2
Stack (bytes):	2
No of cycles:	158 (Note)
Reentrant:	No
Relocatable:	No
Interruptible:	No

Note: "No. of cycles" in "Specifications" indicates the number of cycles required for a 1-digit display.

## Description

### Function Details

**Argument Details:** Global Variable 'segd' holds segment data for each digit.

**Example:** Figure 2-7 shows an example of program module 'main' execution.

If the entry argument is set as shown in ①, the LED display will display the data shown in ②.

Table 2-3 shows relationship between the segment data and the display.

Table 2-3. Segment Data and Display Relationship

Segment Data	Display	Segment Data	Display
0 x C0	0	0 x 92	5
0 x F9	1	0 x 82	6
0 x A4	2	0 x F8	7
0 x B0	3	0 x 80	8
0 x 99	4	0 x 90	9

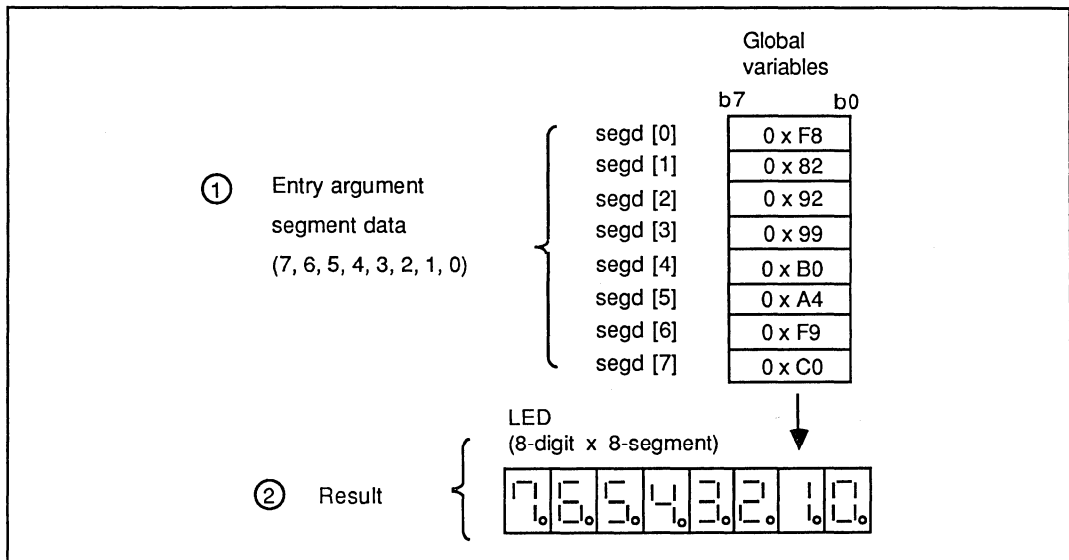


Figure 2-7. Program Module Execution Example

User Notes

1. Store data in display RAM in the form shown in table 2-3.
2. Initialize program module 'main' before execution to output segment data and digit data, using global variable 'count' on the LED display.
3. Select port 6 as the output port.
4. Initialize timer 1.
5. Clear bit I to enable interrupt, because OCI 1 interrupt is used.

Variable Descriptions

The global variables are stored in static memory (figure 2-8).

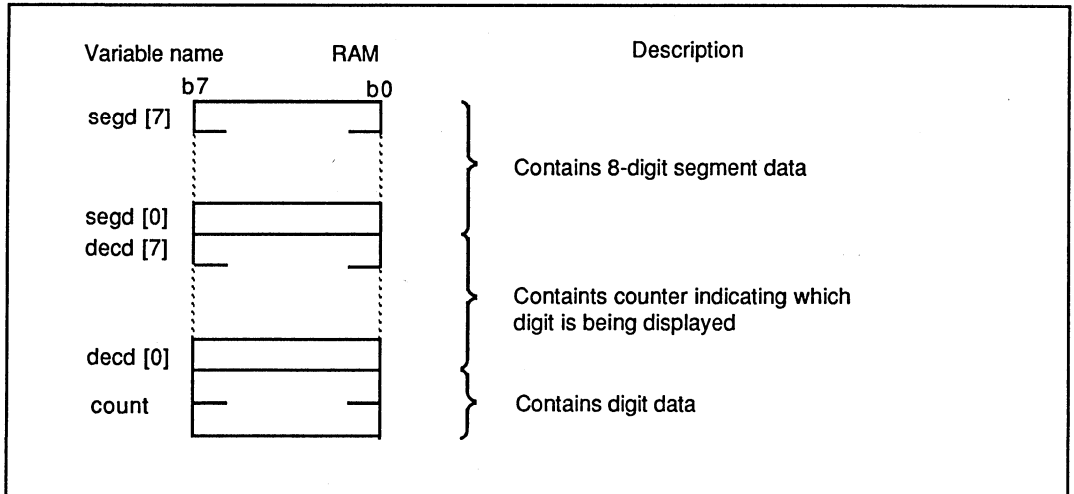


Figure 2-8. Global Variable Storage

Local variables are stored in the stack as 'auto #'.

Figure 2-9 shows an example of a local variable being stored on the stack.

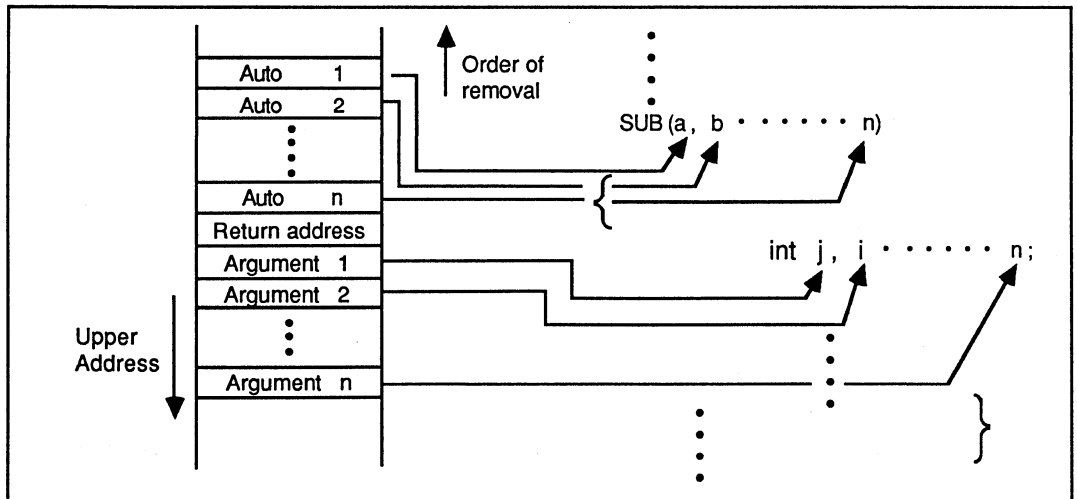


Figure 2-9. Local Variable Storage

The variable 'work' in RAM is a working area for dummy reads of TCSR 1 (figure 2-10).

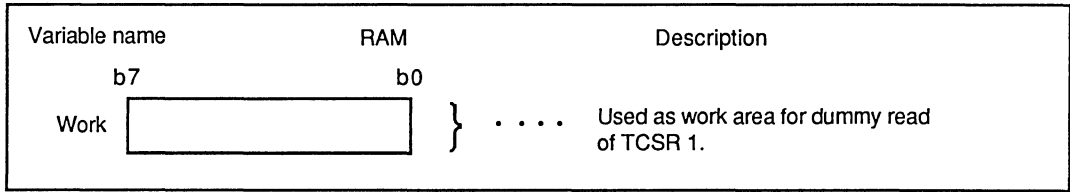


Figure 2-10. Working Area

### Sample Application

After the digit data counter, display counter, port 6, and timer 1 are initialized, 'main' is executed every 1.25 ms, while bit I is cleared, and the LED displays the data (figure 2-11).

```

OPT      REL
XREF     PSCT: MAIN, PSCT: LEDINT

***
LEDMN    LDS     #$FF      ..... Initialize stack pointer
         JSR     LEDINT    ..... Initialize timer, port, and global variable
         CLI     ..... Enable interrupt
PEND     BRA     PEND
LEDSP    JSR     MAIN      ..... Drive 8-digit x 8-segment LED
         RTI

***      Set vector address ***
ORG      $FFEA
FDB     LEDMN    IRQ2
FDB     LEDMN    CMI
FDB     LEDMN    TRAP
FDB     LEDMN    SIO
FDB     LEDMN    TOI
FDB     LEDSP    OCI1
FDB     LEDMN    ICI
FDB     LEDMN    IRQ1
FDB     LEDMN    SWI
FDB     LEDMN    NMI
FDB     LEDMN    RES
END
    
```

Figure 2-11. Sample Application



**Basic Operation**

1. Uses global variable 'count' to find which segment data (segd) and digit data (decd) pair in the array the routine is currently working on.
2. Turns off display for 58  $\mu$ s.
3. Outputs segment data and digit data to port.
4. Repeats steps 1-3, decrementing the counter each time until it becomes 0, indicating that the routine has gone through all the segment data and digit data pairs.

PAD

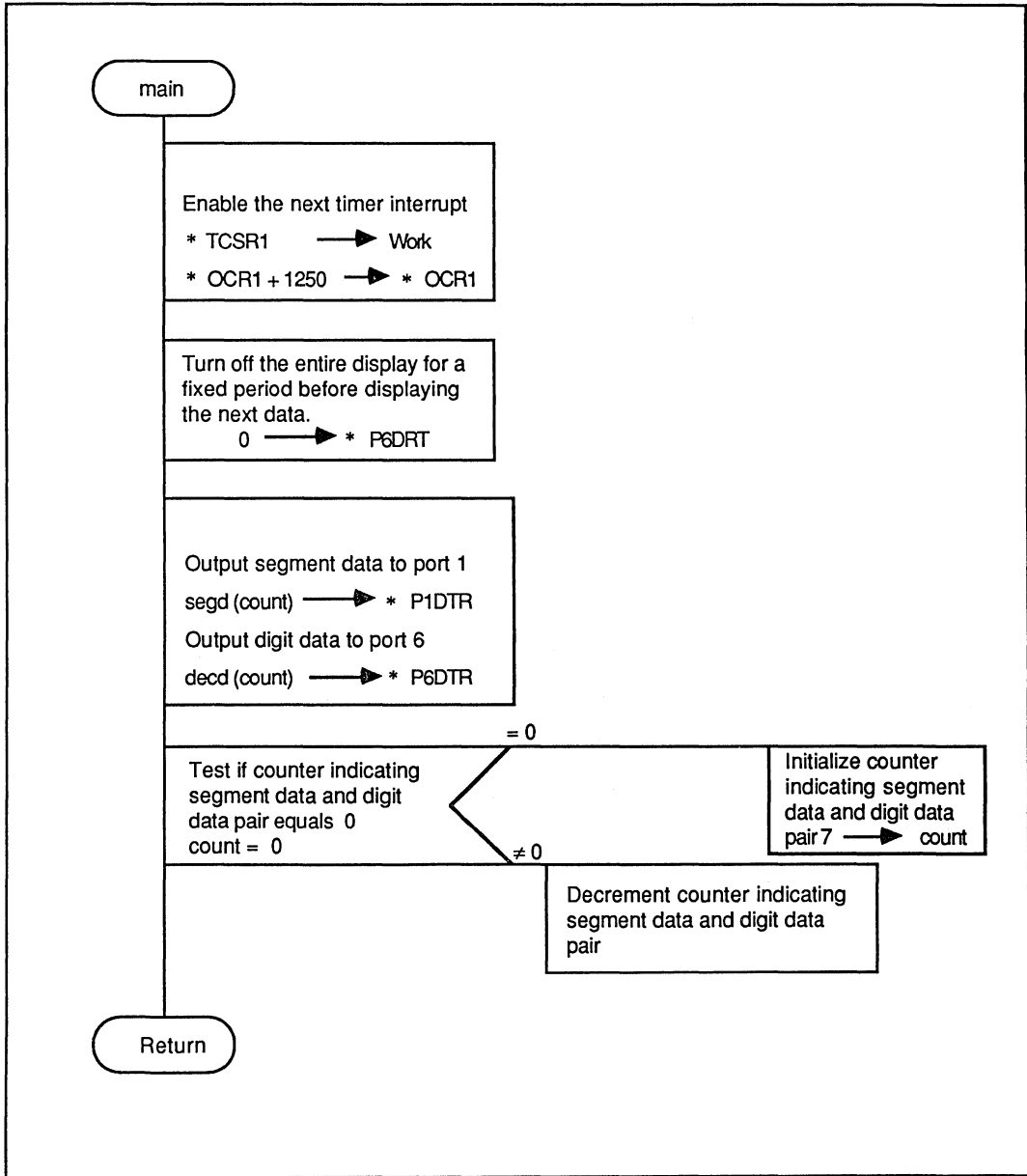


Figure 2-12. Drive LED Module PAD

## 2.4 Program Listing

### 2.4.1 Main Program Listing

\*\*\* CP/M-68K 6301/6801/6800 CROSS MACROASSEMBLER V1.2 \*\*\*

```

ERR  SEQ  LOC  OBJECT  PROGRAM
00001                                     *****
00002                                     *
00003                                     *          MAIN PRUGURAM : LEDMN          *
00004                                     *
00005                                     *****
00006                                     OPT      REL
00007                                     XREF     PSCT:MAIN,PSCT:LEDINT
00008P 0000 8E 00FF  A LEDMN  LDS      #$FF      Set stack pointer
00009P 0003 BD 0000  A      JSR      LEDINT    Initialize variables
00010P 0006 0E      CLI      Enable interrupt
00011P 0007 20 FE 0007 PEND    BRA      PEND      End of program
00012                                     *****
00013                                     *
00014                                     *          NAME : MAIN  (DRIVE LED)        *
00015                                     *
00016                                     *****
00017                                     *
00018                                     *          ENTRY  : SEGD (DISPLAY DATA)  *
00019                                     *          RETURNS : NOTHING              *
00020                                     *
00021                                     *****
00022P 0009 BD 0000  A LEDSP  JSR      MAIN      Drive LED
00023P 000C 3B      RTI
00024                                     *****
00025                                     *
00026                                     *          VECTOR ADDRESS                *
00027                                     *
00028                                     *****
00029                                     *
00030A FFEA      ORG      $FFEA
00031                                     *
00032A FFEA      0000  P      FDB      LEDMN    IRQ2
00033A FFEC      0000  P      FDB      LEDMN    CMI
00034A FFEE      0000  P      FDB      LEDMN    TRAP
00035A FFF0      0000  P      FDB      LEDMN    SIO
00036A FFF2      0000  P      FDB      LEDMN    TOI
00037A FFF4      0009  P      FDB      LEDSP    OC11
00038A FFF6      0000  P      FDB      LEDMN    ICI
00039A FFF8      0000  P      FDB      LEDMN    IRQ1
00040A FFFA      0000  P      FDB      LEDMN    SWI
00041A FFFC      0000  P      FDB      LEDMN    NMI
00042A FFFE      0000  P      FDB      LEDMN    RES
00043                                     *
00044                                     END
**** TOTAL ERRORS 0000--00000

```

## 2.4.2 C Source Listing

```

*
/*****DECLARATION OF DEFINE*****/
/*
#define P1DTR      ((char*)0x2)      /*Port 1 data register*/
#define TCSR1     ((char*)0x8)      /*Timer control status register*/
#define OCR1      ((int*)0xb)      /*Output compare register*/
#define P6DDR     ((char*)0x16)     /*Port 6 data direction register*/
#define P6DTR     ((char*)0x17)     /*Port 6 data register*/
*/
/*****DECLARATION OF GLOBAL VARIABLES*****/
/*
static direct char  decd[8]={128,64,32,16,8,4,2,1}; /*Digit data*/
/*
static direct char  segd[8]={0xf8,0x82,0x92,0x99,0xb0,0xa4,0xf9,0xc0}; /*Segment data*/
static direct int   count; /*Segment, Digit counter*/
/*****
/*
/*      MAIN ROUTINE : MAIN (DRIVE LED)
/*
/*
/*****
/*
/*      ENTRY      : SEGD (DISPLAY DATA)
/*
/*      RETURNS    : NOTHING
/*
/*
/*****
main()
{
    char  work;
        work= *TCSR1; /*Timer controller access only */
        *OCR1 +=1250; /*Set interrupt every 1.25 ms*/
        *P6DTR = 0x0; /*Turn off display*/
        *P1DTR =segd[count]; /*Output segment data*/
        *P6DTR=decd[count]; /*Output digit data*/
        if(count==0) /*Display 8-digit data?*/
            count=7; /*Initialaize counter*/
        .else
            count--; /*Decrement segment,digit counter*/
}
/*****
/*
/*      NAME : LEDINT (INITIALIZE)
/*
/*
/*****
ledint ()
{
    count=7; /*Initialaize digit,segment counter*/
    *P6DDR=0xff; /*Select port6 as output*/
    *TCSR1=0x8; /*Set timer*/
    *OCR1=1250; /*Set 0 in port6*/
}

```

## 2.4.3 Output Object Listing of C Compiler

\*\*\* CP/M-68K 6301/6801/6800 CROSS MACROASSEMBLER V1.2 \*\*\*

ERR	SEQ	LOC	OBJECT	PROGRAM	LEDSP
	00001			NAM	LEDSP
	00002			OPT	REL
	00003		MSEX	MACR	
	00004		CLRA		
	00005		TSTB		
	00006		BPL \.0		
	00007		COMA		
	00008		\.0 EQU *		
	00009		ENDM		
	00010		MLBRA MACR		
	00011		JMP \0		
	00012		ENDM		
	00013		MLBSR MACR		
	00014		JSR \0		
	00015		ENDM		
	00016		MLBEQ MACR		
	00017		BNE \.0		
	00018		JMP \0		
	00019		\.0 EQU *		
	00020		ENDM		
	00021		MLBNE MACR		
	00022		BEQ \.0		
	00023		JMP \0		
	00024		\.0 EQU *		
	00025		ENDM		
	00026		MLBGT MACR		
	00027		BLE \.0		
	00028		JMP \0		
	00029		\.0 EQU *		
	00030		ENDM		
	00031		MLBGE MACR		
	00032		BLT \.0		
	00033		JMP \0		
	00034		\.0 EQU *		
	00035		ENDM		
	00036		MLBLT MACR		
	00037		BGE \.0		
	00038		JMP \0		
	00039		\.0 EQU *		
	00040		ENDM		
	00041		MLBLE MACR		
	00042		BGT \.0		
	00043		JMP \0		
	00044		\.0 EQU *		
	00045		ENDM		
	00046		MLBHI MACR		
	00047		BLS \.0		
	00048		JMP \0		
	00049		\.0 EQU *		
	00050		ENDM		
	00051		MLBLS MACR		
	00052		BHI \.0		
	00053		JMP \0		
	00054		\.0 EQU *		
	00055		ENDM		
	00056		MLBCC MACR		

ERR	SEQ	LOC	OBJECT	PROGRAM	LEDSP
	00057			BCS \.0	
	00058			JMP \.0	
	00059			\.0 EQU *	
	00060			ENDM	
	00061			MLBCS MACR	
	00062			BCC \.0	
	00063			JMP \.0	
	00064			\.0 EQU *	
	00065			ENDM	
	00066P	0000		PSCT	
	00067P	0000	80 A	DECD FCB	-128
	00068P	0001	40 A	FCB	64
	00069P	0002	20 A	FCB	32
	00070P	0003	10 A	FCB	16
	00071P	0004	08 A	FCB	8
	00072P	0005	04 A	FCB	4
	00073P	0006	02 A	FCB	2
	00074P	0007	01 A	FCB	1
	00075P	0008		PSCT	
	00076P	0008	F8 A	SEGD FCB	-8
	00077P	0009	82 A	FCB	-126
	00078P	000A	92 A	FCB	-110
	00079P	000B	99 A	FCB	-103
	00080P	000C	B0 A	FCB	-80
	00081P	000D	A4 A	FCB	-92
	00082P	000E	F9 A	FCB	-7
	00083P	000F	C0 A	FCB	-64
	00084B	0000		BSCT	
	00085B	0000	0002 A	COUNT BSZ	2
	00086P	0010		PSCT	
	00087P	0010	34	MAIN DES	
	00088P	0011	CE 0008 A	LDX	#8
	00089P	0014	E6 00 A	LDAB	0,X
	00090P	0016	30	TSX	
	00091P	0017	E7 00 A	STAB	0,X
	00092P	0019	CE 000B A	LDX	#11
	00093P	001C	EC 00 A	LDD	0,X
	00094P	001E	C3 04E2 A	ADDD	#1250
	00095P	0021	ED 00 A	STD	0,X
	00096P	0023	CE 0017 A	LDX	#23
	00097P	0026	4F	CLRA	
	00098P	0027	5F	CLRB	
	00099P	0028	E7 00 A	STAB	0,X
	00100P	002A	CE 0002 A	LDX	#2
	00101P	002D	3C	PSHX	
	00102P	002E	CC 0008 P	LDD	#SEGD
	00103P	0031	D3 00 B	ADDD	COUNT
	00104P	0033	18	XGDX	
	00105P	0034	E6 00 A	LDAB	0,X
	00106P	0036	38	PULX	
	00107P	0037	E7 00 A	STAB	0,X
	00108P	0039	CE 0017 A	LDX	#23
	00109P	003C	3C	PSHX	
	00110P	003D	CC 0000 P	LDD	#DECD
	00111P	0040	D3 00 B	ADDD	COUNT
	00112P	0042	18	XGDX	

ERR	SEQ	LOC	OBJECT	PROGRAM	LEDSP
	00113P	0043	E6 00	A	LDAB 0,X
	00114P	0045	38		PULX
	00115P	0046	E7 00	A	STAB 0,X
	00116P	0048	DC 00	B	LDD COUNT
	00117P	004A	26 05 0051		BNE \$.A002
	00118P	004C	CC 0007	A	LDD #7
	00119P	004F	20 05 0056		BRA ..1
	00120P	0051	DC 00	B \$.A002	LDD COUNT
	00121P	0053	C3 FFFF	A	ADDD #-1
	00122P	0056	DD 00	B ..1	STD COUNT
	00123P	0058	31	\$.A003	INS
	00124P	0059	39		RTS
	00125P	005A			PSCT
	00126P	005A	CC 0007	A LEDINT	LDD #7
	00127P	005D	DD 00	B	STD COUNT
	00128P	005F	CE 0016	A	LDX #22
	00129P	0062	CC 00FF	A	LDD #255
	00130P	0065	E7 00	A	STAB 0,X
	00131P	0067	CE 0008	A	LDX #8
	00132P	006A	CC 0008	A	LDD #8
	00133P	006D	E7 00	A	STAB 0,X
	00134P	006F	CE 000B	A	LDX #11
	00135P	0072	CC 04E2	A	LDD #1250
	00136P	0075	ED 00	A	STD 0,X
	00137P	0077	39		RTS
	00138				XDEF LEDINT
	00139				XDEF MAIN
	00140				END

\*\*\*\* TOTAL ERRORS 00000--00000

## 2.4.4 Linkage Listing

```
*** HMCS6800 CROSS LINKAGE EDITOR   VER 1.2 ***
LOAD=B:LEDMN.OBJ,B:LEDSP.OBJ,C31RUN.OBJ
STRP=$F000
STRB=$60
STRD=$40
OPT=MAP,SYM
EXEC
```

```
*** HMCS6800 CROSS LINKAGE EDITOR   VER 1.2 ***
*** UNDEFINED SYMBOLS ***
      NAME SECTION  MODULE NAME
      .ERROR (      )
UNDEFINED SYMBOL = 1 (Note)
```

Note: There is an UNDEFINED SYMBOL=1 (library function, ERROR) in the link information but it does not influence the execution of this program. The library function or run-time routines call the ERROR service routine when 0 is used as a divisor in division or modulo operations. Strictly speaking, the user should create an ERROR function. However it is never used in this program, so it is just displayed as an UNDEFINED SYMBOL.

```
*** HMCS6800 CROSS LINKAGE EDITOR   VER 1.2 ***
*** MAP LIST ***
** SECTION LOAD MAP
      SECTION  SIZE  START  END  COMMON-SIZE
      A      0016  FFEA  FFFF
      B      0002  0060  0061      0000
      C      0000
      D      0004  0040  0043      0000
      P      0657  F000  F656      0000
** MODULE LOAD MAP
      NAME     BSCT  DSCT  PSCT
      LEDSP    0060          F000
              0040  F085
** COMMON LOAD MAP
      NAME SECTION  SIZE  START
COMMON = 0
```



\*\*\* DEFINED SYMBOLS \*\*\*

NAME	SECTION	START	MODULE NAME
.\$DADD	P	F656	( )
.\$DCMP	P	F656	( )
.\$DDEC	P	F656	( )
.\$DDIV	P	F656	( )
.\$DINC	P	F656	( )
.\$DMOV	P	F656	( )
.\$DMUL	P	F656	( )
.\$DNEG	P	F656	( )
.\$DSTK	P	F656	( )
.\$DSUB	P	F656	( )
.\$DTON	P	F656	( )
.\$DTOL	P	F656	( )
.\$DTST	P	F656	( )
.\$FDEC	P	F656	( )
.\$FINC	P	F656	( )
.\$FMOV	P	F656	( )
.\$FREG	D	0040	( )
.\$FTOD	P	F656	( )
.\$FTST	P	F656	( )
.\$IASL	P	F0FA	( )
.\$IASR	P	F10F	( )
.\$IDIV	P	F0BC	( )
.\$IMOD	P	F139	( )
.\$IMUL	P	F085	( )
.\$ITOD	P	F656	( )
.\$ITOL	P	F39A	( )
.\$LADD	P	F1AC	( )
.\$LAND	P	F2AF	( )
.\$LBIT	P	F47D	( )
.\$LCMP	P	F33C	( )
.\$LCPL	P	F37E	( )
.\$LDEC	P	F3C8	( )
.\$LDIV	P	F260	( )
.\$LINC	P	F3B8	( )
.\$LMOD	P	F287	( )
.\$LMOV	P	F188	( )
.\$LMUL	P	F1DE	( )
.\$LNEG	P	F369	( )
.\$LOR	P	F2CA	( )
.\$LSHL	P	F300	( )
.\$LSHR	P	F31E	( )
.\$LSTK	P	F3D8	( )
.\$LSUB	P	F1C5	( )
.\$LTOD	P	F656	( )
.\$LTST	P	F3F3	( )
.\$LXOR	P	F2E5	( )
.\$SBIT	P	F5A0	( )
.\$SW1	P	F5E8	( )
.\$SW2	P	F617	( )
.\$UDIV	P	F0D8	( )
.\$ULSR	P	F124	( )
.\$UMOD	P	F167	( )

\*\*\* HMCS6800 CROSS LINKAGE EDITOR VER 1.2 \*\*\*

NAME	SECTION	START	MODULE NAME
.\$UTOD	P	F656	( )
.\$UTOL	P	F3AB	( )
LEDINT	P	F067	( LEDSP )
MAIN	P	F01D	( LEDSP )

DEFINED SYMBOL = 57



# SECTION 3. 8 X 4 KEY MATRIX

## 3.1 Hardware Description

### 3.1.1 Function

The key matrix routine scans an 8 x 4 key matrix using the HD6301X0. It converts the key data into ASCII (A-Z, 1-6).

If two keys are pressed simultaneously, the data is invalid.

### 3.1.2 Microcontroller Applications

1. The interrupt routine is executed every 8 ms by the built-in 16-bit programmable timer (timer 1) and output compare interrupt 1 (OC11).
2. The interrupt routine executes a key scan outputting a strobe signal from port 6.
3. The interrupt routine prevents key chatter errors .
4. The key scan strobe signal is controlled by changing the I/O direction of the port 6 data direction register (DDR). A diode is not necessary to prevent output signal collision since all ports that do not output a strobe signal are input ports (high-impedance state).

### 3.1.3 Circuit Diagram

Figure 3-1 is the application circuit diagram.

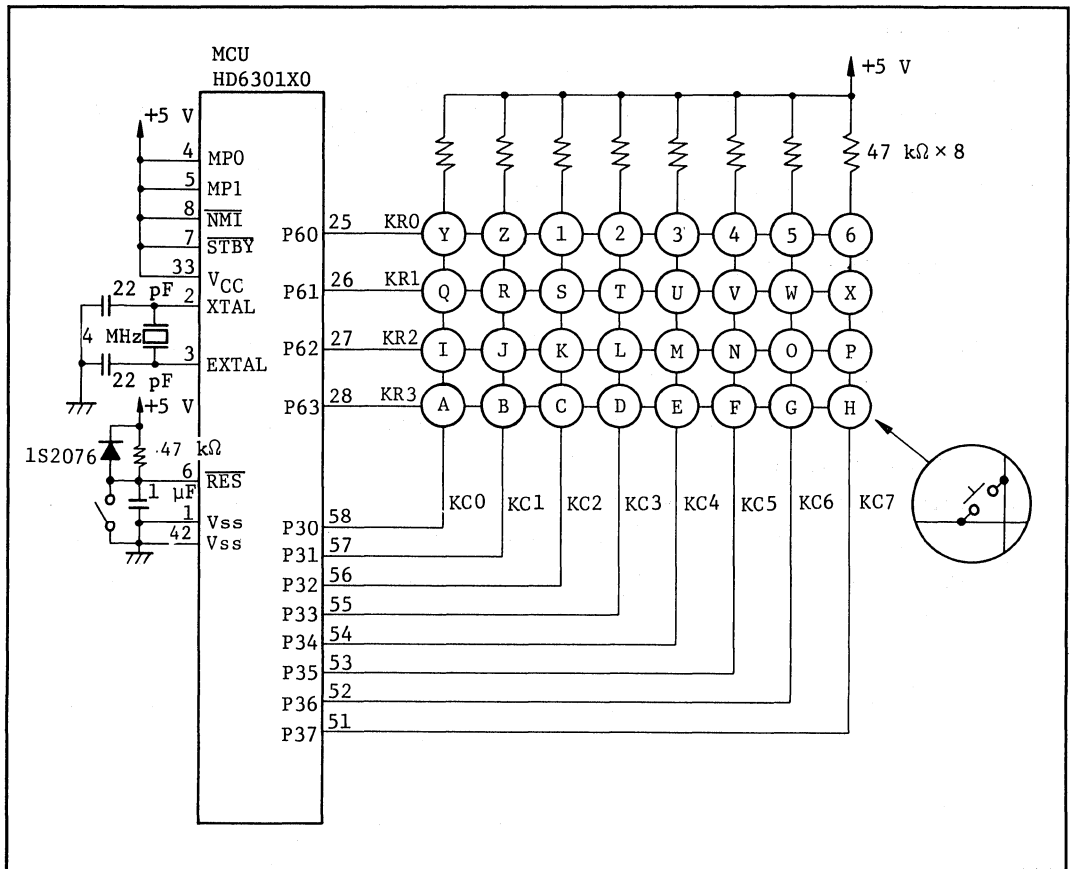


Figure 3-1. Key Scan Control Circuit

### 3.1.4 Pin Functions

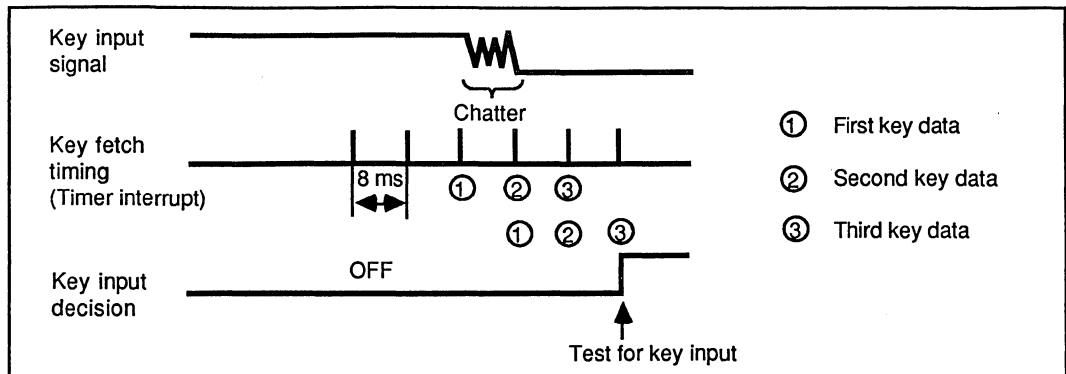
Table 3-1 shows the pin functions at the interface between the HD6301X0 and the key matrix.

**Table 3-1. Pin Functions**

Pin Name (HD6301X0)	Input/Output	Active Level (High or Low)	Function	Pin Name (Key matrix)	Program Label
P63	Input/Output	Low	Outputs strobe for 8 x 4 key matrix retrieval.	KR3	P6DTR
P62	Input/Output	Low		KR2	
P61	Input/Output	Low		KR1	
P60	Input/Output	Low		KR0	
P30	Input	—	Inputs 8 x 4 key matrix key data.	KC0	P3DTR
P31	Input	—		KC1	
P32	Input	—		KC2	
P33	Input	—		KC3	
P34	Input	—		KC4	
P35	Input	—		KC5	
P36	Input	—		KC6	
P37	Input	—		KC7	

### 3.1.5 Hardware Operation

The program prevents errors caused by key chatter (figure 3-2).



**Figure 3-2. Chatter Prevention Timing**

The key input signal is sampled every 8 ms.

If three consecutive key input signals are the same, the key input data is defined. If two or fewer signals are the same, the key input data is not defined, assuming that chatter has occurred.

## 3.2 Software Description

### 3.2.1 Program Module Configuration

Figure 3-3 shows the program module configuration for executing a key scan of an 8 x 4 key matrix.

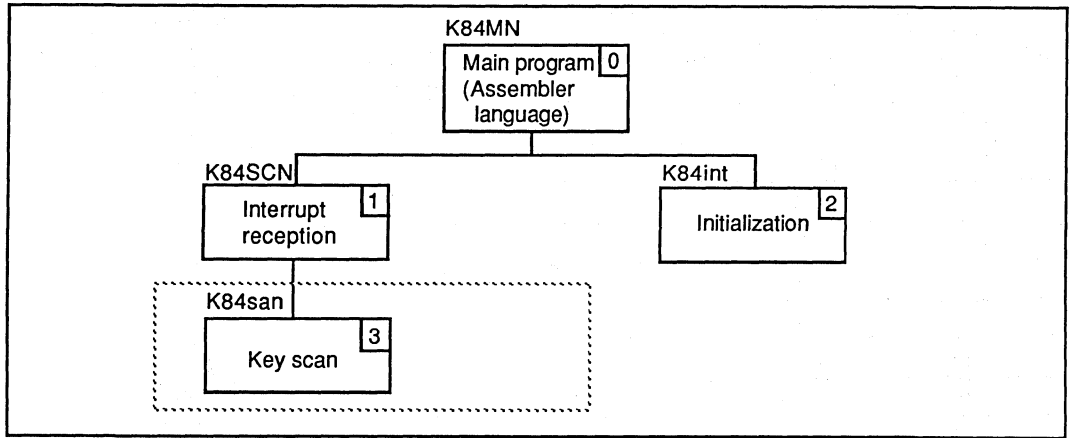


Figure 3-3. Program Module Configuration

Refer to Section 3.3, "Program Module Description" discusses these modules for details.

### 3.2.2 Program Module Functions

Table 3-2 summarizes the program module functions.

Table 3-2. Program Module Functions

No.	Program Module Name	Library Function	Function	Language
0	Main program	K84MN	Initializes instructions, such as ORG, LDS, and CLI, which do not exist in C. Calls K84int function	ASM
1	Interrupt reception	K84SCN	Receives and process OCI 1 interrupt	ASM
2	Initialization	K84int	Initializes global variables, port, and timer	C
3	Key scan	K84san	Converts key data from 8 x 4 key matrix into ASCII	C

Note: C: C Language Program  
 ASM: Assembly Language Program

### 3.2.3 Program Module Sample Application (Main Program)

The flowchart in figure 3-4 is an example of an 8 x 4 key matrix key scan performed by the program module in figure 3-3. The main program in Figure 3-4 calls the C language module and demonstrates storing ASCII in global variable 'keyset'

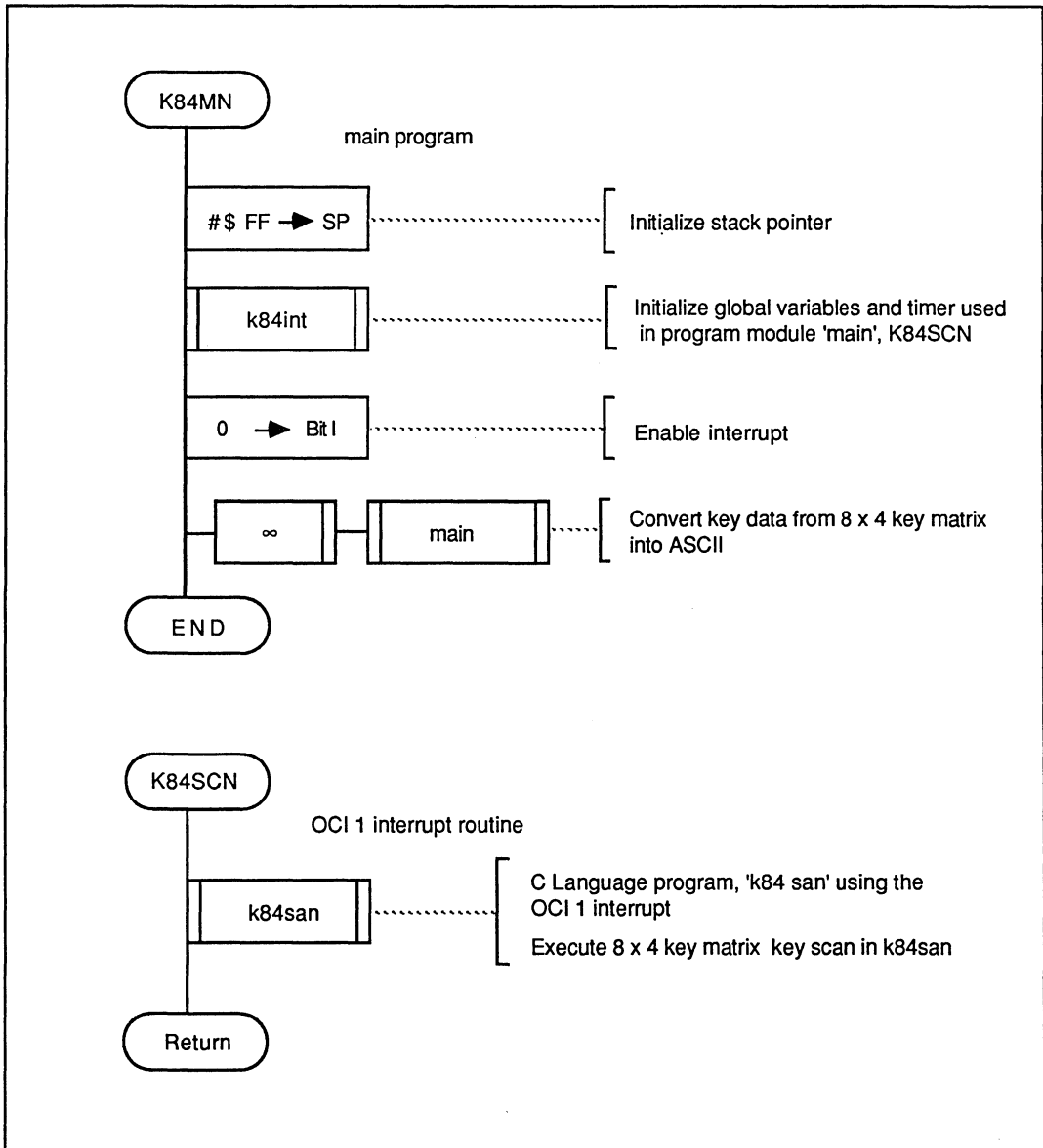


Figure 3-4. Program Module Flowchart

The C Language Program 'K84int ' (figure 3-5) initializes the timer, port, and global variables.

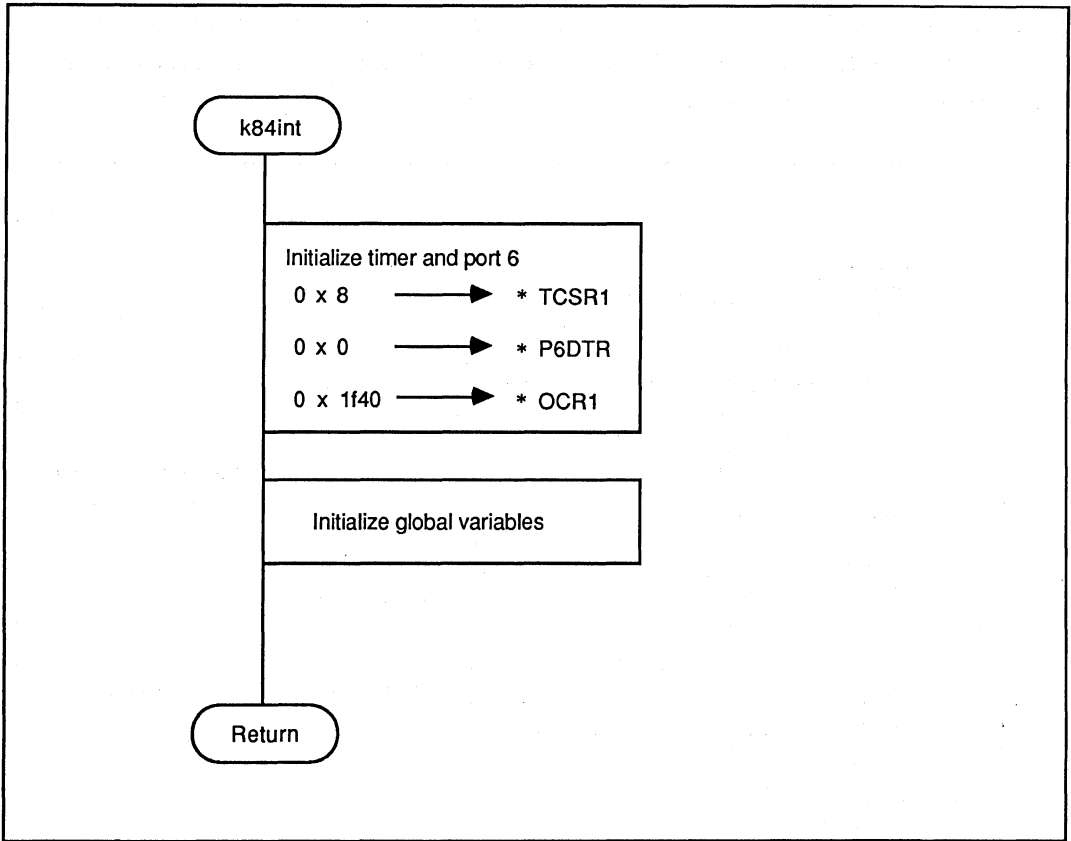


Figure 3-5. Program Module Flowchart

The execution sequence of the C Language Program 'main' decides whether a key has been pressed (figure 3-6). If a key has been pressed, K84san converts the scanned data, into ASCII and store the result in grobal variable 'keyset'.

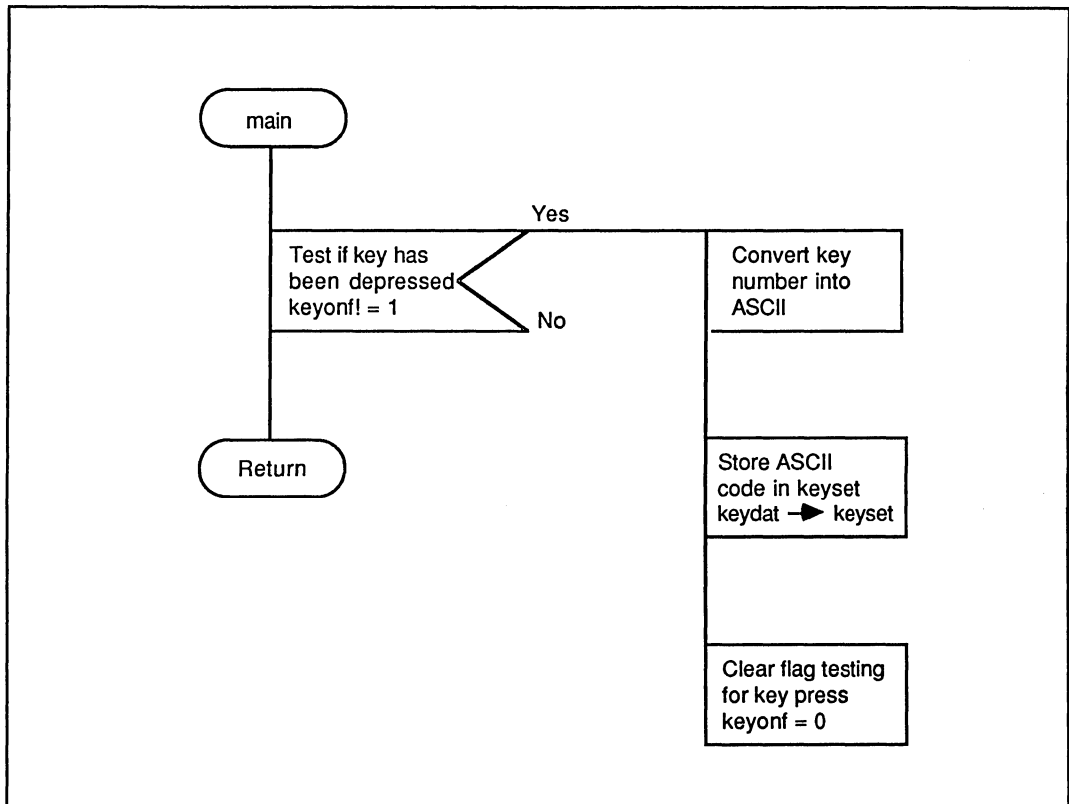


Figure 3-6. Program Module Flowchart

### 3.3 Program Module Description

The following pages describe the key Scan Subroutine.



**Function**

The key scan module scans an 8 x 4 key matrix and stores key scan data in global variable keydat.

**Arguments**

Contents	Storage Locaton	No. of Bytes
Entry	—	—
Returns	key data (global variable)	1
	key data Indicator (global variable)	1

**Libraries Required for Program Execution**

Library	Required/Not Required
Standard Library Function	C31LIB. OBJ Not required
Run-Time Routine	C31RUN. OBJ C31RUNF. OBJ Required Not required

**Specifications**

ROM (bytes):	247
RAM (bytes):	16
Stack (bytes):	8
No of cycles:	1115
Reentrant:	No
Relocatable:	No
Interruptible:	Yes

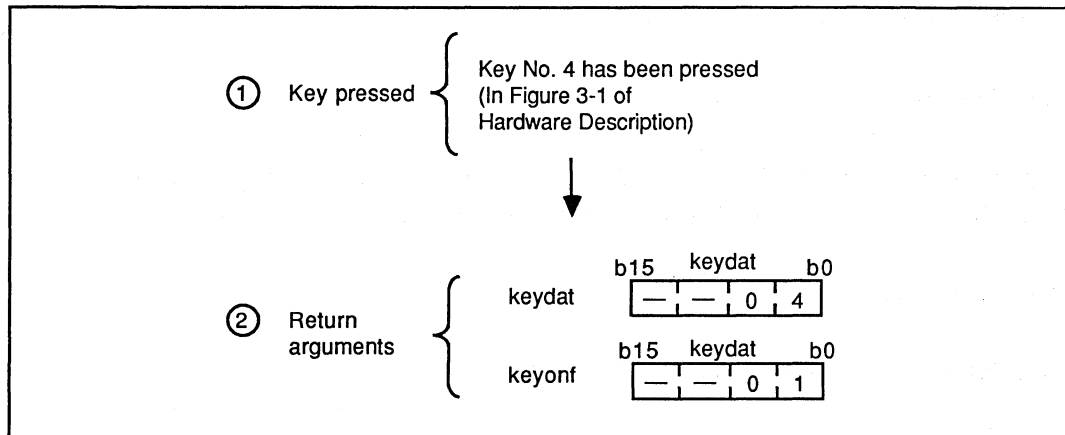
## Description

### Function Details

**Argument Details:** Global variable 'keydat' contains key scan data. Global variable 'keyonf' indicates that program module K84san is done. Flag functions are shown in table 3-3.

**Table 3-3. Keyonf Flag**

Variable Name	Condition	Indicates
Keyonf	0	Key scan data is not stored in global variable 'keydat'
	1	Key scan has executed correctly and the key scan data has been stored in global variable 'keydat'



**Figure3-7. Program Module 'K84san' Execution Example**

Example: Figure 3-7 shows an example of program module 'K84san' stoves execution. If a key in ① is pressed, the key scan data is stored in global variable 'keydat', as shown in ②. Program module K84san does not call any other program modules or subroutines.

### User Notes

1. Clear global variables 'oldkey' and 'keyonf' before executing program module K84san.
2. Initialize timer 1.
3. Clear bit I and enables OCI 1 interrupt.

Variable Descriptions

The global variables are stored in static memory (figure 3-8).

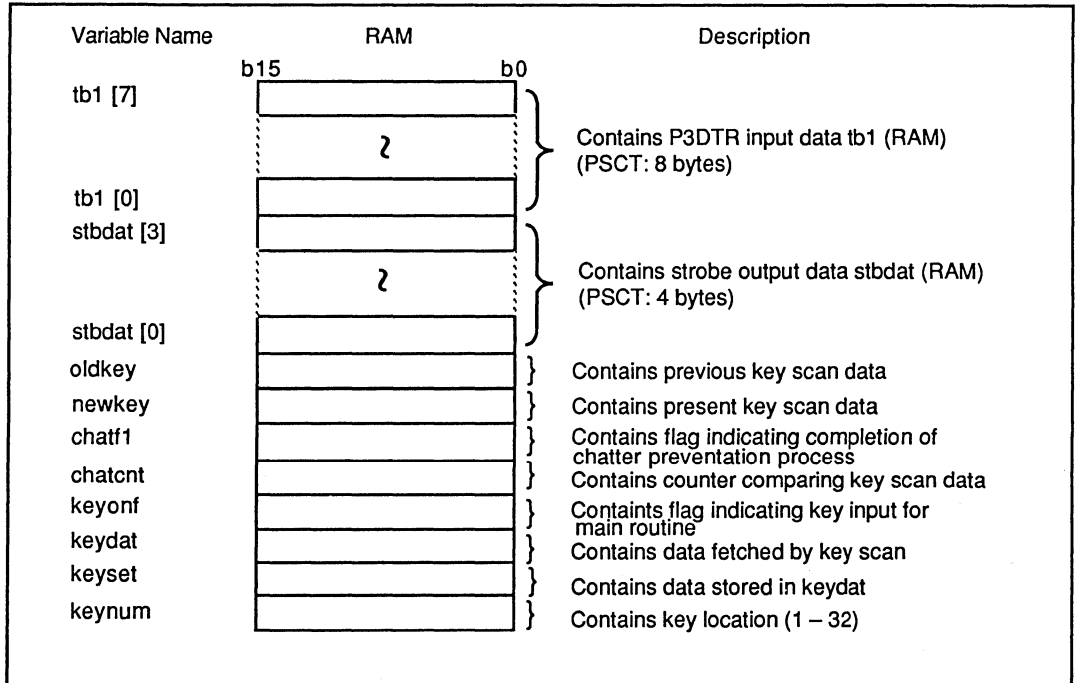


Figure 3-8. Global Variables Storage

Local variables are stored in stack as 'auto #'.

Figure 3-9 shows an example of a local variable being stored on the stack.

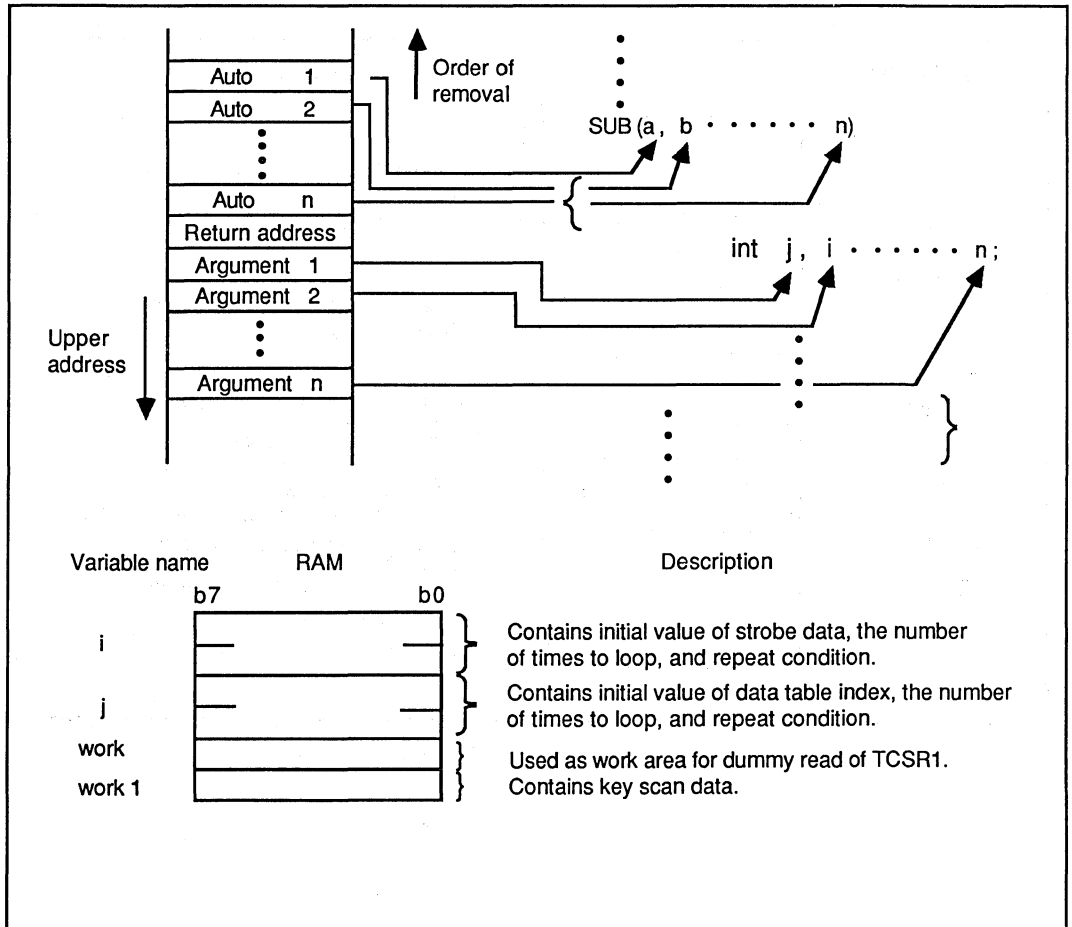


Figure 3-9. Local Variable Storage

## Sample Application

Program module K84SCN is executed every 8 ms after the global variables are initialized and the interrupt is enabled. Figure 3-10 indicates that the timer routine K84san in K84SCN and the timer interrupt is restored with the RTI instruction.

	OPT	REL	
	XREF	PSCT: MAIN, PSCT: K84INT, PSCT: K84SCN	
***			
K84MN	LDS	#\$FF	..... Initialize stack pointer
	JSR	K84INT	..... Initialize port, timer, and global variables
	CLI		..... Enable interrupt
PEND	JSR	MAIN	} ..... Test if key has been pressed
	BRA	PEND	
K84SCN	JSR	K84SAN	
	RTI		
*			
***	Set vector address		***
*			
	ORG	\$FFEA	
	FDB	K84MN	IRQ2
	FDB	K84MN	CMI
	FDB	K84MN	TRAP
	FDB	K84MN	SIO
	FDB	K84MN	TOI
	FDB	K84SCN	OCI 1
	FDB	K84MN	ICI
	FDB	K84MN	IRQ1
	FDB	K84MN	SWI
	FDB	K84MN	NMI
	FDB	K84MN	RES
	END		

Figure 3-10.

**Basic Operation**

1. Executes key scan at timer interrupt (generated every 8 ms).  
Checks key scan execution flag (global variable 'keyonf') to decide whether to execute key scan.
2. Outputs strobe signal from lower 4 bits of port 6.  
Fetches key scan data from port 3.
3. Tests if key scan data, which was fetched in (2), is Oxff.
  - a. If it is Oxff, a key has not been pressed in the current column and the strobe signal for the next column is output.
  - b. If it is not Oxff, the routine tests which row's key has been pressed.  
The routine repeats the following process 8 times. Tests if the data in local variable 'work 1', which holds the key scan data, equals data in 'tbl' (RAM). If the data are equal, a key has been pressed.  
There are eight data table patterns in 'tbl'. Each pattern indicates that one specific key has been pressed. So if two keys are pressed, the patterns do not match. When that occurs the module is existed.
4. Compares key data fetched in step 3 with previous key data. If the data matches three times consecutively, key data is valid and is fetched. At that time, the routine gets chatter prevention flag global variable 'chatcnt' to '1' to indicate that the key data is valid. If the data in global variable 'newkey' differs from that in global variable 'oldkey', or no key is pressed, the chatter prevention flag is cleared.

PAD

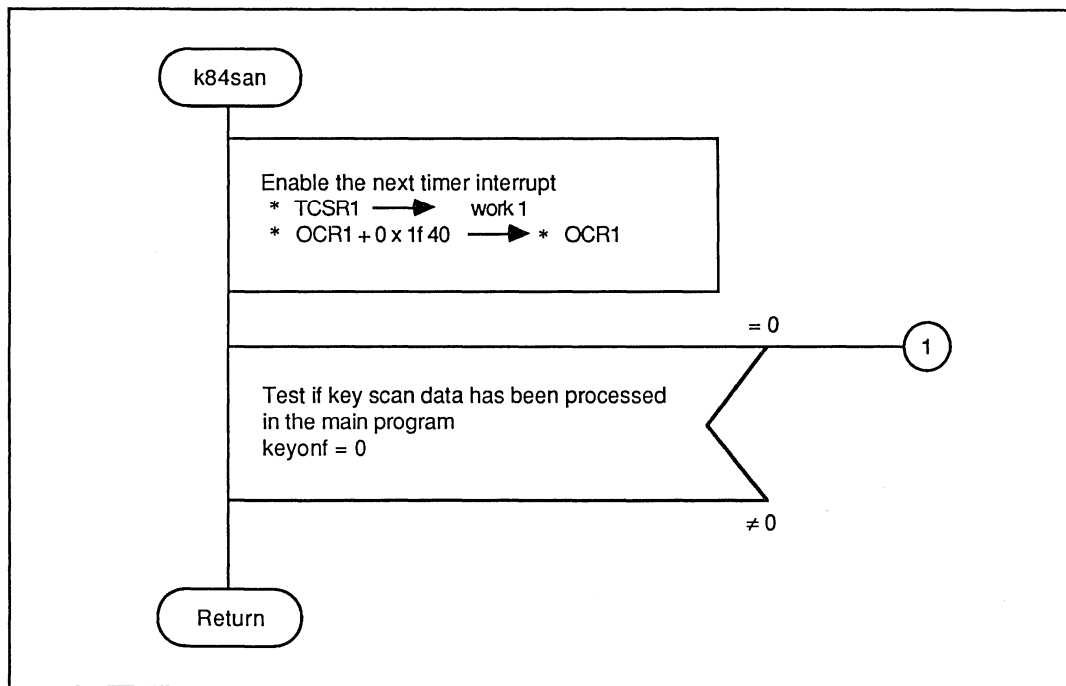


Figure 3-11. Key Scan Module PAD



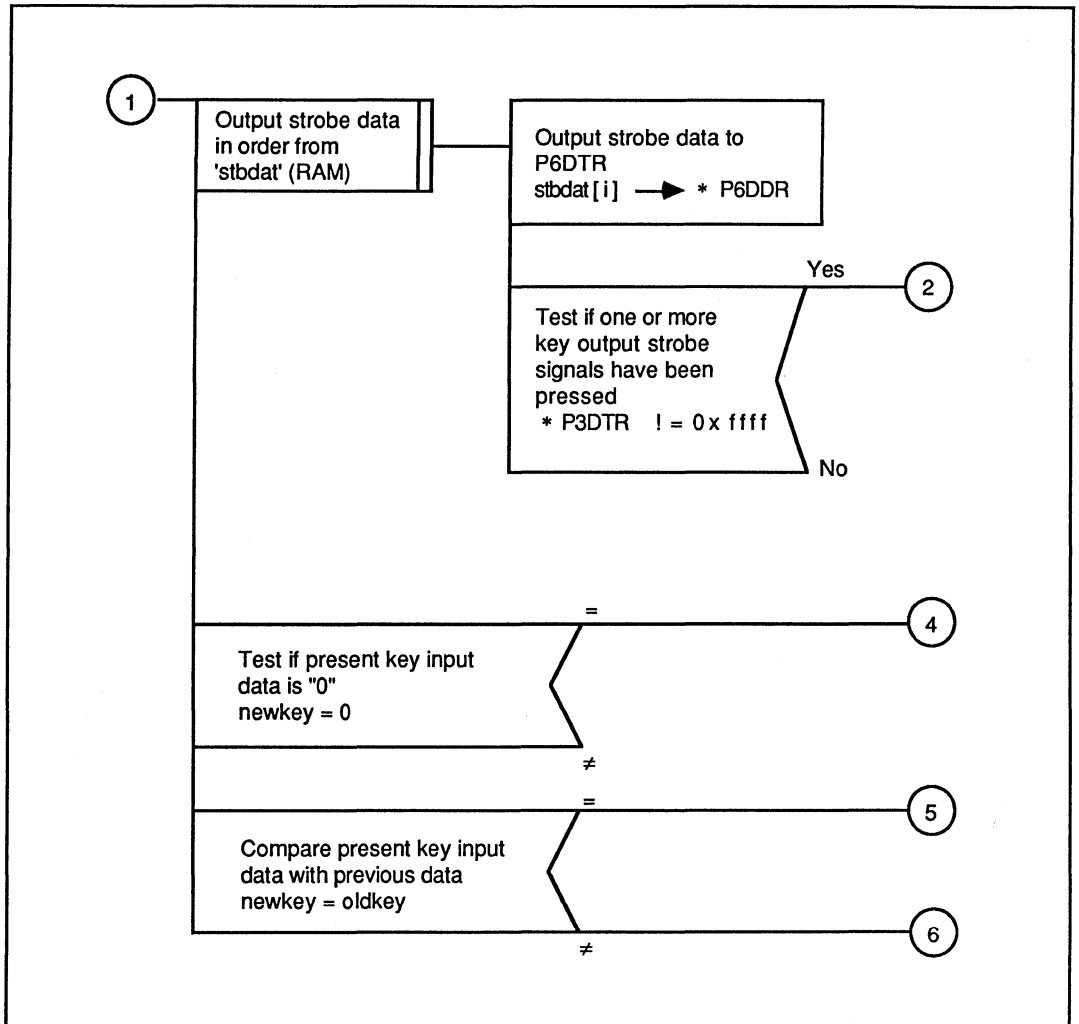


Figure 3-11. Key Scan Module PAD

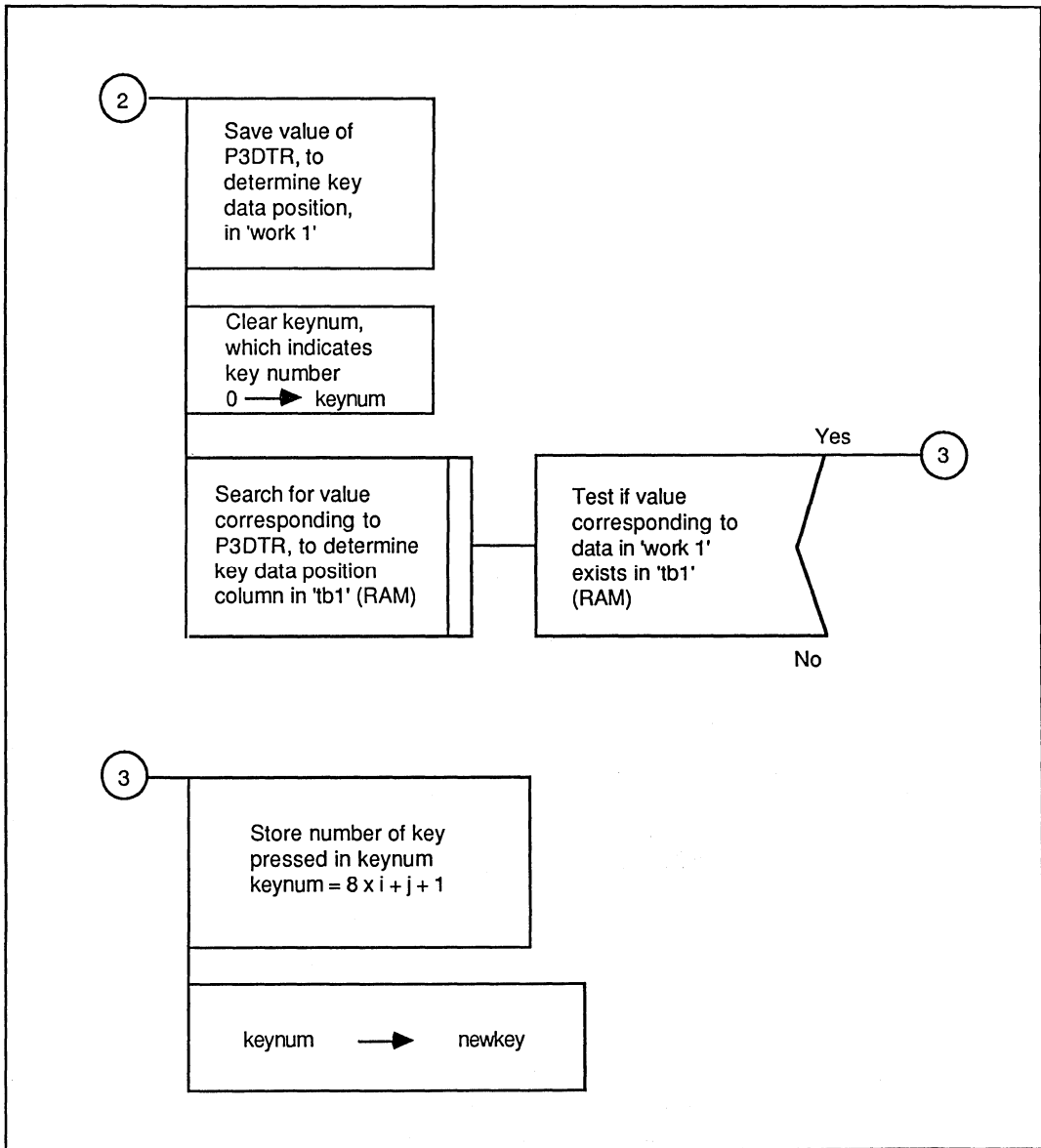


Figure 3-11. Key Scan Module PAD (cont)

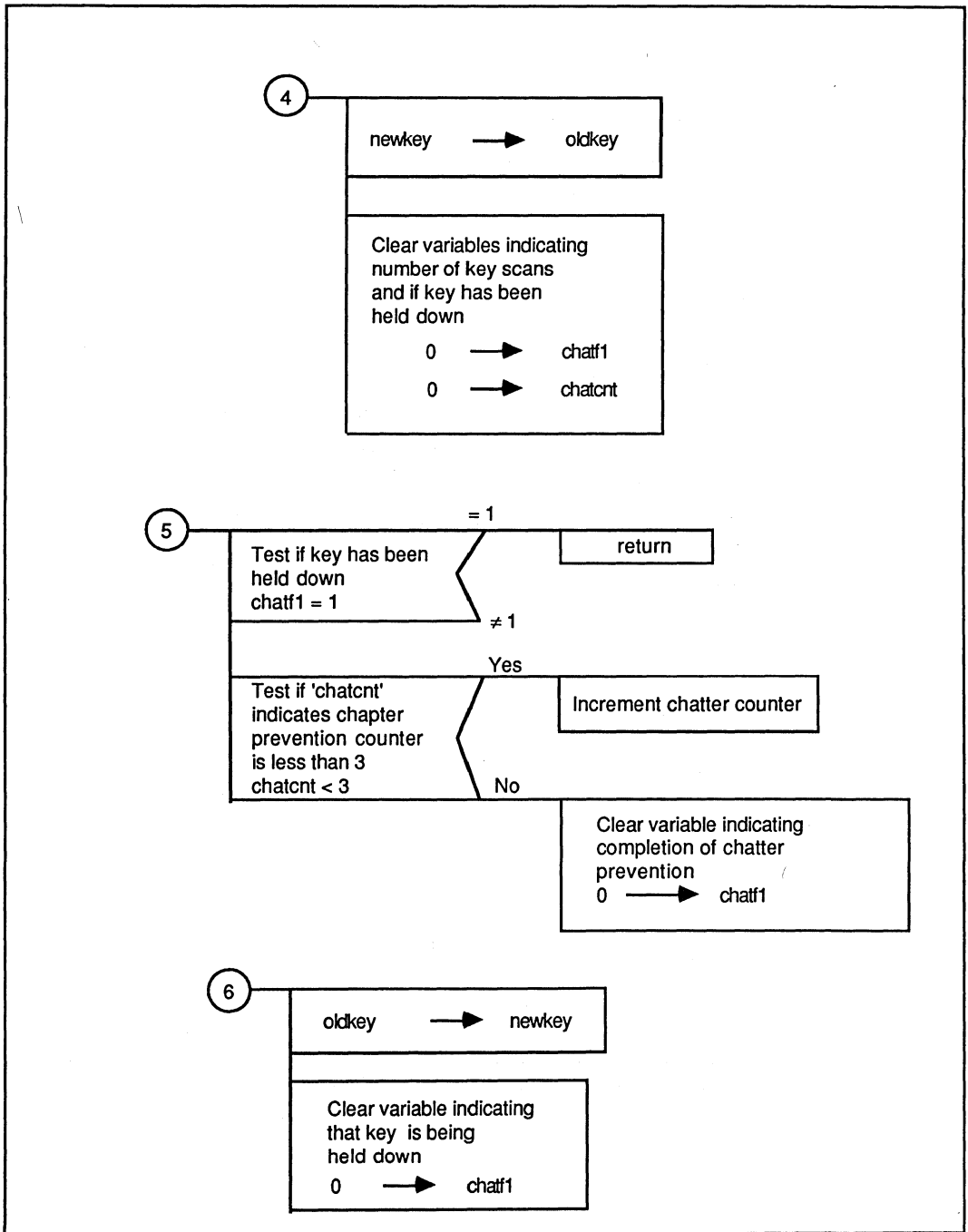


Figure 3-11. Key Scan Module PAD (cont)

## 3.4 Program Listing

### 3.4.1 Main Program Listing

\*\*\* CP/M-68K 6301/6801/6800 CROSS MACROASSEMBLER V1.2 \*\*\*

```
ERR  SEQ  LOC  OBJECT  PROGRAM
00001          *****
00002          *
00003          *
00004          *          MAIN PROGRAM : K84MN
00005          *
00006          *****
00007          OPT      REL
00008          XREF    PSCT:MAIN,PSCT:K84INT,PSCT:K84SAN
00009P 0000 8E 00FF A K84MN LDS  #$FF      Set stack pointer
00010P 0003 BD 0000 A          JSR  K84INT   Initialize variables
00011P 0006 0E          CLI          Enable interrupt
00012P 0007 BD 0000 A PEND  JSR  MAIN     Branch main routine
00013P 000A 20 FB 0007          BRA  PEND     Continuous loop
00014          *****
00015          *
00016          *          NAME : K84SAN (KEY SCAN)
00017          *
00018          *****
00019          *
00020          *          ENTRY      : NOTHING
00021          *          RETURNS   : KEYDAT (KEYDAT DATA)
00022          *          KEYONF    (KEY ON FLAG)
00023          *
00024          *****
00025P 000C BD 0000 A K84SCN JSR  K84SAN   Branch to key scan routine
00026P 000F 3B          RTI          Return from interrupt
00027          *****
00028          *
00029          *          VECTOR ADDRESS
00030          *
00031          *****
00032          *
00033A FFEA          ORG      $FFEA
00034          *
00035A FFEA 0000 P          FDB  K84MN   IRQ2
00036A FFEC 0000 P          FDB  K84MN   CMI
00037A FFEE 0000 P          FDB  K84MN   TRAP
00038A FFF0 0000 P          FDB  K84MN   SIO
00039A FFF2 0000 P          FDB  K84MN   TOI
00040A FFF4 000C P          FDB  K84SCN  OC11
00041A FFF6 0000 P          FDB  K84MN   ICI
00042A FFF8 0000 P          FDB  K84MN   IRQ1
00043A FFFA 0000 P          FDB  K84MN   SWI
00044A FFFC 0000 P          FDB  K84MN   NMI
00045A FFFE 0000 P          FDB  K84MN   RES
00046          *
00047          END
**** TOTAL ERRORS 0000--00000
```

### 3.4.2 C Source Listing

```

*
/*****DECLARATION OF DEFINE*****/
/*
#define TCSR1      ((char*)0x08)    /* Timer control status reg. */
#define OCR1       ((int*)0x0b)    /* Interrupt set */
#define P6DTR      ((char*)0x17)    /* Port 6 data reg. */
#define P6DDR      ((char*)0x16)    /* Port 6 input */
#define P3DTR      ((char*)0x06)    /* Port 3 output */
/*
/*****DECLARATION OF GLOBAL VARIABLES*****/
/*
static direct int      oldkey;          /* Comparison area */
static direct int      newkey;         /* New key data */
static direct int      chatfl;        /* Chatter flag */
static direct int      chatcnt;       /* Chatter counter */
static direct int      keyonf;       /* Key-on flag */
static direct int      keydat;       /* Set key data */
static direct int      keyset;       /* Set ASCII data */
static direct int      keynum;       /* Set key number */
static char            tbl[8]={0xfffe,0xfffd,0xfffb,0xfff7,
                                0xffef,0xffdf,0xffbf,0xff7f};
/* Compare data with P3DTR */
static char            stbdat[4]={0x08,0x04,0x02,0x01};
/* Set strobe signal */
/*****
/*
/*      MAIN ROUTINE : MAIN (JUDGE IF KEY IS HIT AND ASCII CODE OF KEY)
/*
/*
/*****
main()
{
    if (keyonf!=1)
        return;
    else {
        keydat += (keydat<27)?'A'-1:'1'-27;
        /* change keydat to ascii */
        keyset=keydat;    /* set ASCII of keydat */
        keyonf=0;
    }
}
/*****
/*
/*      NAME : K84INT (INITIALIZE)
/*
/*
/*****
k84int()
{
    *TCSR1=0x8;          /* Set timer */
    *P6DTR=0x0;         /* Set 0 in Port6 */
    *OCR1=0x1f40;       /* Set direct interrupt every 8ms */

    oldkey=newkey=chatfl=chatcnt=keyonf=keydat=keyset=0;

    /* oldkey:          Comparing area */
    /* newkey:          New key data */
    /* chatfl:          Chattering flag */
    /* chatcnt:         Chattering counter */
    /* keyonf:          Flag of key-on */
    /* keydat:          Set key data */
    /* keyset:          Set ASCII data */

```

```

}
/*****
/*
/*      NAME      : K84SAN  (KEY SCAN)
/*
/*****
/*
/*      ENTRY      : NOTHING
/*      RETURNS    : KEYDAT  (KEYDAT DATA)
/*                  KEYONF  (KEY ON FLAG)
/*
/*****
k84san()      /* routine of key scan */
{
    char      work,work1;
    int       i,j;

    work= *TCSR1;          /* Just accessed timer controller */
    *OCR1+=0x1f40;        /* Set interrupt every 8ms */
    if (keyonf==0) {     /* Judge if key is hit */
        for (i=0;i<4;i++) {
            *P6DDR=stbdat[1]; /* Give strobe data to P6DDR */
            if (*P3DTR!=0xffff) {
                /* For macro expansion,
                /* upper two bytes of P3DTR
                /* are filled with "$FF"
                work1= *P3DTR;
                keynum=0;
                /* Check the bit number of 0 */
                for (j=0;j<8;j++) {
                    if (work1==tbl[j])
                        keynum=8*i+j+1;
                }
                newkey=keynum; /* Set key number
                               /* in newkey
            }
        }
        if (newkey==0){ /* no key-on */
            oldkey=newkey;
            chatfl=chatcnt=0;
            return;
        }
        if (newkey==oldkey) {
            if (chatfl==1)
                return;
            if (chatcnt<3)
                chatcnt++;
            else{
                chatfl=1;
                keydat=newkey;
                keyonf=1;
            }
        }
        else {
            oldkey=newkey;
            chatfl=0;
        }
    }
}
}

```

### 3.4.3 Output Object Listing of C Compiler

\*\*\* CP/M-68K 6301/6801/6800 CROSS MACROASSEMBLER V1.2 \*\*\*

ERR	SEQ	LOC	OBJECT	PROGRAM	K84SCN
	00001			NAM	K84SCN
	00002			OPT	REL
	00003		MSEX	MACR	
	00004		CLRA		
	00005		TSTB		
	00006		BPL \.0		
	00007		COMA		
	00008		\.0 EQU *		
	00009		ENDM		
	00010		MLBRA	MACR	
	00011		JMP \0		
	00012		ENDM		
	00013		MLBSR	MACR	
	00014		JSR \0		
	00015		ENDM		
	00016		MLBEQ	MACR	
	00017		BNE \.0		
	00018		JMP \0		
	00019		\.0 EQU *		
	00020		ENDM		
	00021		MLBNE	MACR	
	00022		BEQ \.0		
	00023		JMP \0		
	00024		\.0 EQU *		
	00025		ENDM		
	00026		MLBGT	MACR	
	00027		BLE \.0		
	00028		JMP \0		
	00029		\.0 EQU *		
	00030		ENDM		
	00031		MLBGE	MACR	
	00032		BLT \.0		
	00033		JMP \0		
	00034		\.0 EQU *		
	00035		ENDM		
	00036		MLBLT	MACR	
	00037		BGE \.0		
	00038		JMP \0		
	00039		\.0 EQU *		
	00040		ENDM		
	00041		MLBLE	MACR	
	00042		BGT \.0		
	00043		JMP \0		
	00044		\.0 EQU *		
	00045		ENDM		
	00046		MLBHI	MACR	
	00047		BLS \.0		
	00048		JMP \0		
	00049		\.0 EQU *		
	00050		ENDM		
	00051		MLBLS	MACR	
	00052		BHI \.0		
	00053		JMP \0		
	00054		\.0 EQU *		
	00055		ENDM		
	00056		MLBCC	MACR	

```

ERR  SEQ  LOC  OBJECT          PROGRAM K84SCN
00057          BCS \.0
00058          JMP \0
00059          \.0 EQU *
00060          ENDM
00061          MLBCS MACR
00062          BCC \.0
00063          JMP \0
00064          \.0 EQU *
00065          ENDM
00066B 0000          BSCT
00067B 0000    0002  A  OLDKEY BSZ    2
00068B 0002          BSCT
00069B 0002    0002  A  NEWKEY BSZ    2
00070B 0004          BSCT
00071B 0004    0002  A  CHATFL BSZ    2
00072B 0006          BSCT
00073B 0006    0002  A  CHATCN BSZ    2
00074B 0008          BSCT
00075B 0008    0002  A  KEYONF BSZ    2
00076B 000A          BSCT
00077B 000A    0002  A  KEYDAT BSZ    2
00078B 000C          BSCT
00079B 000C    0002  A  KEYSET BSZ    2
00080B 000E          BSCT
00081B 000E    0002  A  KEYNUM BSZ    2
00082P 0000          PSCT
00083P 0000    FE    A  TBL    FCB   -2
00084P 0001    FD    A          FCB   -3
00085P 0002    FB    A          FCB   -5
00086P 0003    F7    A          FCB   -9
00087P 0004    EF    A          FCB  -17
00088P 0005    DF    A          FCB  -33
00089P 0006    BF    A          FCB  -65
00090P 0007    7F    A          FCB  127
00091P 0008          PSCT
00092P 0008    08    A  STBDAT FCB    8
00093P 0009    04    A          FCB    4
00094P 000A    02    A          FCB    2
00095P 000B    01    A          FCB    1
00096P 000C          PSCT
00097P 000C  DE 08    B  MAIN   LDX   KEYONF
00098P 000E  8C 0001  A          CPX   #1
00099P 0011  27 01 0014  BEQ   $.A002
00100P 0013  39          RTS
00101P 0014  DE 0A    B  $.A002 LDX   KEYDAT
00102P 0016  8C 001B  A          CPX   #27
00103P 0019  2C 05 0020  BGE   $.A004
00104P 001B  CC 0040  A          LDD   #64
00105P 001E  20 03 0023  BRA   $.A005
00106P 0020  CC 0016  A  $.A004 LDD   #22
00107P 0023  D3 0A    B  $.A005 ADDD  KEYDAT
00108P 0025  DD 0A    B          STD   KEYDAT
00109P 0027  DD 0C    B          STD   KEYSET
00110P 0029  4F          CLRA
00111P 002A  5F          CLRB
00112P 002B  DD 08    B          STD   KEYONF

```



ERR	SEQ	LOC	OBJECT	PROGRAM	K84SCN
	00113P	002D	39	.\$A003	RTS
	00114P	002E			PSCT
	00115P	002E	CE 0008	A K84INT	LDX #8
	00116P	0031	CC 0008	A	LDD #8
	00117P	0034	E7 00	A	STAB 0,X
	00118P	0036	CE 0017	A	LDX #23
	00119P	0039	4F		CLRA
	00120P	003A	5F		CLRB
	00121P	003B	E7 00	A	STAB 0,X
	00122P	003D	CE 000B	A	LDX #11
	00123P	0040	CC 1F40	A	LDD #8000
	00124P	0043	ED 00	A	STD 0,X
	00125P	0045	4F		CLRA
	00126P	0046	5F		CLRB
	00127P	0047	DD 0C	B	STD KEYSET
	00128P	0049	DD 0A	B	STD KEYDAT
	00129P	004B	DD 08	B	STD KEYONF
	00130P	004D	DD 06	B	STD CHATCN
	00131P	004F	DD 04	B	STD CHATFL
	00132P	0051	DD 02	B	STD NEWKEY
	00133P	0053	DD 00	B	STD OLDKEY
	00134P	0055	39		RTS
	00135P	0056			PSCT
	00136P	0056	3C	K84SAN	PSHX
	00137P	0057	3C		PSHX
	00138P	0058	3C		PSHX
	00139P	0059	CE 0008	A	LDX #8
	00140P	005C	E6 00	A	LDAB 0,X
	00141P	005E	30		TSX
	00142P	005F	E7 05	A	STAB 5,X
	00143P	0061	CE 000B	A	LDX #11
	00144P	0064	EC 00	A	LDD 0,X
	00145P	0066	C3 1F40	A	ADDD #8000
	00146P	0069	ED 00	A	STD 0,X
	00147P	006B	DC 08	B	LDD KEYONF
	00148P	006D			MLBNE .\$A008
	00149P	0072	4F		CLRA
	00150P	0073	5F		CLRB
	00151P	0074	30		TSX
	00152P	0075			MLBRA ..2
	00153P	0078	CE 0016	A .\$A009	LDX #22
	00154P	007B	3C		PSHX
	00155P	007C	CC 0008	P	LDD #STBDAT
	00156P	007F	30		TSX
	00157P	0080	E3 04	A	ADDD 4,X
	00158P	0082	18		XGDX
	00159P	0083	E6 00	A	LDAB 0,X
	00160P	0085	38		PULX
	00161P	0086	E7 00	A	STAB 0,X
	00162P	0088	CE 0006	A	LDX #6
	00163P	008B	E6 00	A	LDAB 0,X
	00164P	008D			MSEX
	00165P	0092	18		XGDX
	00166P	0093	8C FFFF	A	CPX #-1
	00167P	0096			MLBEQ .\$A011
	00168P	009B	CE 0006	A	LDX #6

ERR	SEQ	LOC	OBJECT	PROGRAM	K84SCN
	00169P	009E	E6 00	A	LDAB 0,X
	00170P	00A0	30		TSX
	00171P	00A1	E7 04	A	STAB 4,X
	00172P	00A3	4F		CLRA
	00173P	00A4	5F		CLRB
	00174P	00A5	DD 0E	B	STD KEYNUM
	00175P	00A7	20 31 00DA		BRA ..1
	00176P	00A9	30	.\$A012	TSX
	00177P	00AA	E6 04	A	LDAB 4,X
	00178P	00AC			MSEX
	00179P	00B1	18		XGDX
	00180P	00B2	3C		PSHX
	00181P	00B3	CC 0000	P	LDD #TBL
	00182P	00B6	30		TSX
	00183P	00B7	E3 02	A	ADDD 2,X
	00184P	00B9	18		XGDX
	00185P	00BA	E6 00	A	LDAB 0,X
	00186P	00BC			MSEX
	00187P	00C1	30		TSX
	00188P	00C2	A3 00	A	SUBD 0,X
	00189P	00C4	38		PULX
	00190P	00C5	26 0D 00D4		BNE .\$A014
	00191P	00C7	30		TSX
	00192P	00C8	EC 02	A	LDD 2,X
	00193P	00CA	05		ASLD
	00194P	00CB	05		ASLD
	00195P	00CC	05		ASLD
	00196P	00CD	E3 00	A	ADDD 0,X
	00197P	00CF	C3 0001	A	ADDD #1
	00198P	00D2	DD 0E	B	STD KEYNUM
	00199P	00D4	30	.\$A014	TSX
	00200P	00D5	EC 00	A	LDD 0,X
	00201P	00D7	C3 0001	A	ADDD #1
	00202P	00DA	ED 00	A	..1 STD 0,X
	00203P	00DC	30	.\$A013	TSX
	00204P	00DD	EE 00	A	LDX 0,X
	00205P	00DF	8C 0008	A	CPX #8
	00206P	00E2	2D C5 00A9		BLT .\$A012
	00207P	00E4	DC 0E	B	LDD KEYNUM
	00208P	00E6	DD 02	B	STD NEWKEY
	00209P	00E8	30	.\$A011	TSX
	00210P	00E9	EC 02	A	LDD 2,X
	00211P	00EB	C3 0001	A	ADDD #1
	00212P	00EE	ED 02	A	..2 STD 2,X
	00213P	00F0	30	.\$A010	TSX
	00214P	00F1	EE 02	A	LDX 2,X
	00215P	00F3	8C 0004	A	CPX #4
	00216P	00F6			MLBLT .\$A009
	00217P	00FB	DC 02	B	LDD NEWKEY
	00218P	00FD	26 0A 0109		BNE .\$A015
	00219P	00FF	DD 00	B	STD OLDKEY
	00220P	0101	4F		CLRA
	00221P	0102	5F		CLRB
	00222P	0103	DD 06	B	STD CHATCN
	00223P	0105	DD 04	B	STD CHATFL
	00224P	0107	20 35 013E		BRA ..3

ERR	SEQ	LOC	OBJECT	PROGRAM	K84SCN
	00225P	0109	DE 02	B .SA015	LDX NEWKEY
	00226P	010B	9C 00	B	CPX OLDKEY
	00227P	010D	26 27 0136		BNE .SA016
	00228P	010F	DE 04	B	LDX CHATFL
	00229P	0111	8C 0001	A	CPX #1
	00230P	0114	27 28 013E		BEQ . .3
	00231P	0116	DE 06	B .SA017	LDX CHATCN
	00232P	0118	8C 0003	A	CPX #3
	00233P	011B	2C 09 0126		BGE .SA018
	00234P	011D	DC 06	B	LDD CHATCN
	00235P	011F	C3 0001	A	ADDD #1
	00236P	0122	DD 06	B	STD CHATCN
	00237P	0124	20 18 013E		BRA .SA019
	00238P	0126	CC 0001	A .SA018	LDD #1
	00239P	0129	DD 04	B	STD CHATFL
	00240P	012B	DC 02	B	LDD NEWKEY
	00241P	012D	DD 0A	B	STD KEYDAT
	00242P	012F	CC 0001	A	LDD #1
	00243P	0132	DD 08	B	STD KEYONF
	00244P	0134	20 08 013E		BRA .SA020
	00245P	0136	DC 02	B .SA016	LDD NEWKEY
	00246P	0138	DD 00	B	STD OLDKEY
	00247P	013A	4F		CLRA
	00248P	013B	5F		CLRB
	00249P	013C	DD 04	B	STD CHATFL
	00250		013E	P .SA019	EQU *
	00251		013E	P .SA020	EQU *
	00252		013E	P . .3	EQU *
	00253P	013E	38	.SA008	PULX
	00254P	013F	38		PULX
	00255P	0140	38		PULX
	00256P	0141	39		RTS
	00257				XDEF K84SAN
	00258				XDEF K84INT
	00259				XDEF MAIN
	00260				END

\*\*\*\* TOTAL ERRORS 00000--00000

### 3.4.4 Linkage Listing

```
*** HMCS6800 CROSS LINKAGE EDITOR  VER 1.2 ***
LOAD=B:K84MN.OBJ,B:K84SCN.obj,C31RUN.OBJ
STRP=$F000
STRD=$40
STRB=$60
OPT=MAP,SYM
EXEC
```

```
*** HMCS6800 CROSS LINKAGE EDITOR  VER 1.2 ***
*** UNDEFINED SYMBOLS ***
      NAME      SECTION  MODULE NAME
      .ERROR    (          )
UNDEFINED SYMBOL = 1      (Note)
```

Note: There is an UNDEFINE SYMBOL=1 (library function ERROR) in the link information but it does not influence the execution of this program. The library function or run-time routines call the ERROR service routine when 0 is used as a divisor in division or module operation. Strictly speaking, the user should create an ERROR function. However it is never used in this program, so it is just displayed as an UNDEFINED SYMBOL.

```
*** HMCS6800 CROSS LINKAGE EDITOR  VER 1.2 ***
*** MAP LIST ***
** SECTION LOAD MAP
      SECTION  SIZE  START  END  COMMON-SIZE
      A        0016  FFEA  FFFF
      B        0010  0060  006F      0000
      C        0000
      D        0004  0040  0043      0000
      P        0724  F000  F723      0000
** MODULE LOAD MAP
      NAME      BSCT  DSCT  PSCT
              K84SCN  0060              F000
              0040  F152
** COMMON LOAD MAP
      NAME      SECTION  SIZE  START
COMMON = 0
```

\*\*\* HMCS6800 CROSS LINKAGE EDITOR VER 1.2 \*\*\*  
 \*\*\* DEFINED SYMBOLS \*\*\*

NAME	SECTION	START	MODULE NAME
.\$DADD	P	F723	( )
.\$DCMP	P	F723	( )
.\$DDEC	P	F723	( )
.\$DDIV	P	F723	( )
.\$DINC	P	F723	( )
.\$DMOV	P	F723	( )
.\$DMUL	P	F723	( )
.\$DNEG	P	F723	( )
.\$DSTK	P	F723	( )
.\$DSUB	P	F723	( )
.\$DTOF	P	F723	( )
.\$DTOI	P	F723	( )
.\$DTOL	P	F723	( )
.\$DTST	P	F723	( )
.\$FDEC	P	F723	( )
.\$FINC	P	F723	( )
.\$FMOV	P	F723	( )
.\$FREG	D	0040	( )
.\$FTOD	P	F723	( )
.\$FTST	P	F723	( )
.\$IASL	P	F1C7	( )
.\$IASR	P	F1DC	( )
.\$IDIV	P	F189	( )
.\$IMOD	P	F206	( )
.\$IMUL	P	F152	( )
.\$ITOD	P	F723	( )
.\$ITOL	P	F467	( )
.\$LADD	P	F279	( )
.\$LAND	P	F37C	( )
.\$LBIT	P	F54A	( )
.\$LCMP	P	F409	( )
.\$LCPL	P	F44B	( )
.\$LDEC	P	F495	( )
.\$LDIV	P	F32D	( )
.\$LINC	P	F485	( )
.\$LMOD	P	F354	( )
.\$LMOV	P	F255	( )
.\$LMUL	P	F2AB	( )
.\$LNEG	P	F436	( )
.\$LOR	P	F397	( )
.\$LSHL	P	F3CD	( )
.\$LSHR	P	F3EB	( )
.\$LSTK	P	F4A5	( )
.\$LSUB	P	F292	( )
.\$LTOD	P	F723	( )
.\$LTST	P	F4C0	( )
.\$LXOR	P	F3B2	( )
.\$SBIT	P	F66D	( )
.\$SW1	P	F6B5	( )
.\$SW2	P	F6E4	( )
.\$UDIV	P	F1A5	( )
.\$ULSR	P	F1F1	( )
.\$UMOD	P	F234	( )

\*\*\* HMCS6800 CROSS LINKAGE EDITOR VER 1.2 \*\*\*

NAME	SECTION	START	MODULE NAME
.\$UTOD	P	F723	( )
.\$UTOL	P	F478	( )
K84INT	P	F03E	( K84SCN )
K84SAN	P	F066	( K84SCN )
MAIN	P	F01C	( K84SCN )

DEFINED SYMBOL = 58

# SECTION 4. EXTERNAL EXPANSION

## 4.1 Hardware Description

### 4.1.1 Function

The external expansion application controls external memory and peripheral LSIs using the HD6301Y0. It uses the HD6350 (ACIA) as an asynchronous serial interface with a console typewriter. It also controls a liquid crystal module H2571 and displays console typewriter input characters using the HD6321 (PIA).

### 4.1.2 Microcontroller Applications

This application interfaces with external LSIs through an address bus, data bus, and control signals ( $R/\bar{W}$  and E) using the HD6301Y0 external expansion function.

### 4.1.3 Circuit Diagram

Figure 4-1 is the application circuit diagram.

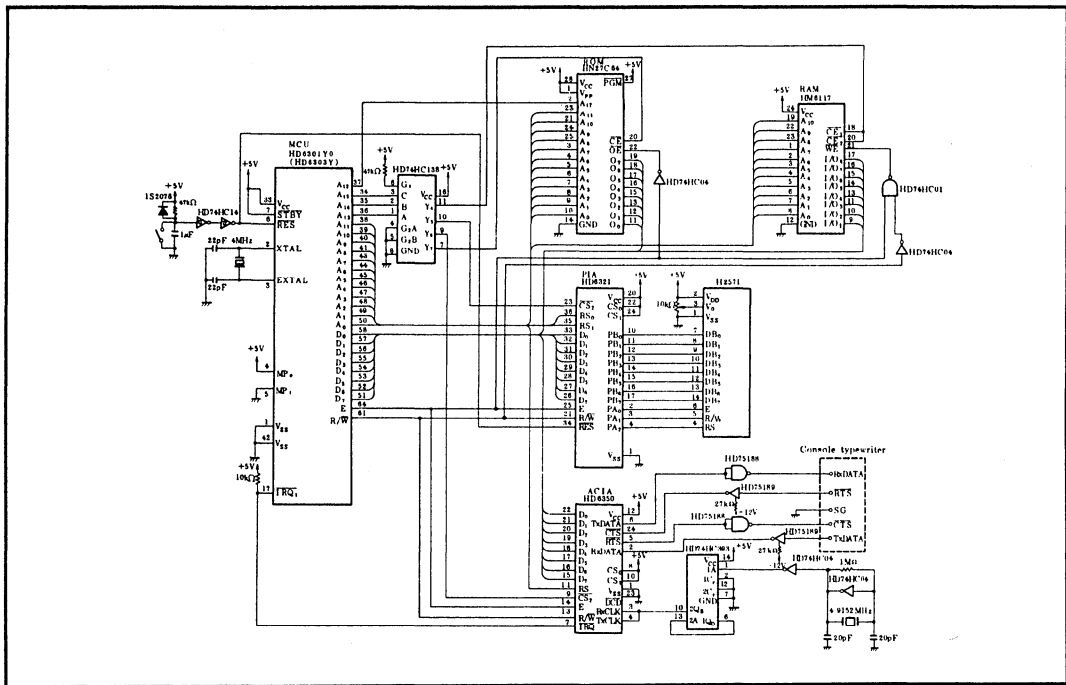


Figure 4-1. External Expansion Circuit Diagram

### 4.1.4 Memory Map

Memories and peripheral LSIs are allocated in external address space using an address decoder (HD74HC138).

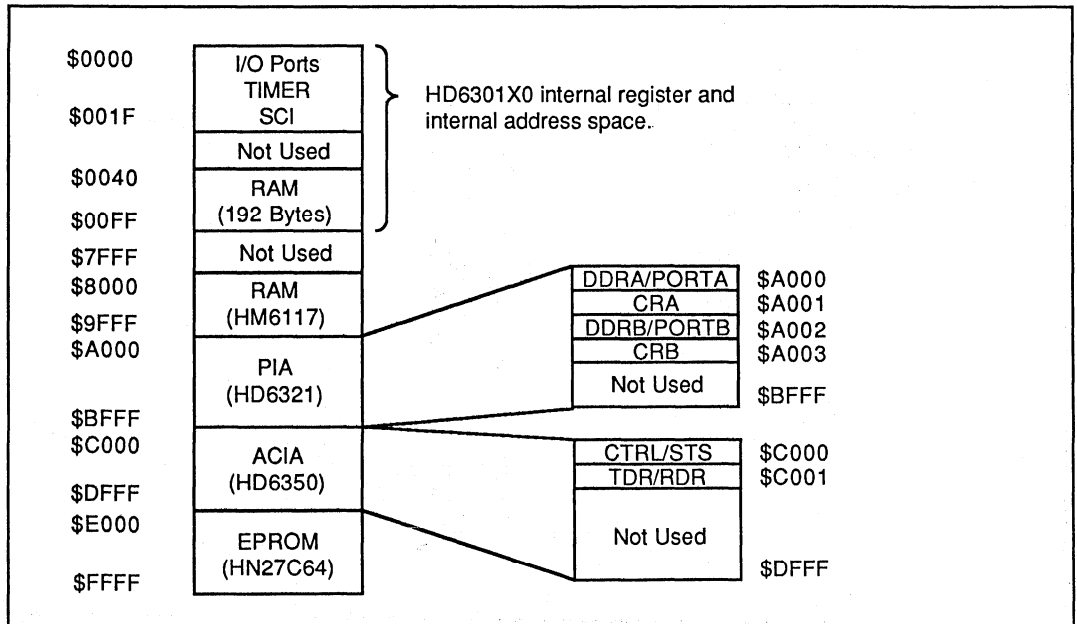
Address lines A13, A14 and A15 are connected to pins A, B and C of the HD74HC138.

Address space \$8000-\$FFFF is divided into 8k-byte units. Table 4-1 shows the system address decoding.

**Table 4-1. System Address Decoding**

HD74HC138											
Input						Output					
G1	G2A	G2BC		B	A	Y4	Y5	Y6	Y7	Address	Allocation
				A14	A13						
H	L	L	L	L	L	L	H	H	H	\$8000-\$9FFF	RAM
H	L	L	L	L	H	H	L	H	H	\$A000-\$BFFF	PIA
H	L	L	L	H	L	H	H	L	H	\$C000-\$DFFF	ACIA
H	L	L	L	H	H	H	H	H	L	\$E000-\$FFFF	ROM

Figure 4-2 shows system memory map.



**Figure 4-2. System Memory Map**

### 4.1.5 Hardware Operation

Figure 4-3 shows the interface timing chart for the HD6301Y0 and external memory (HN27C64, HM6117).

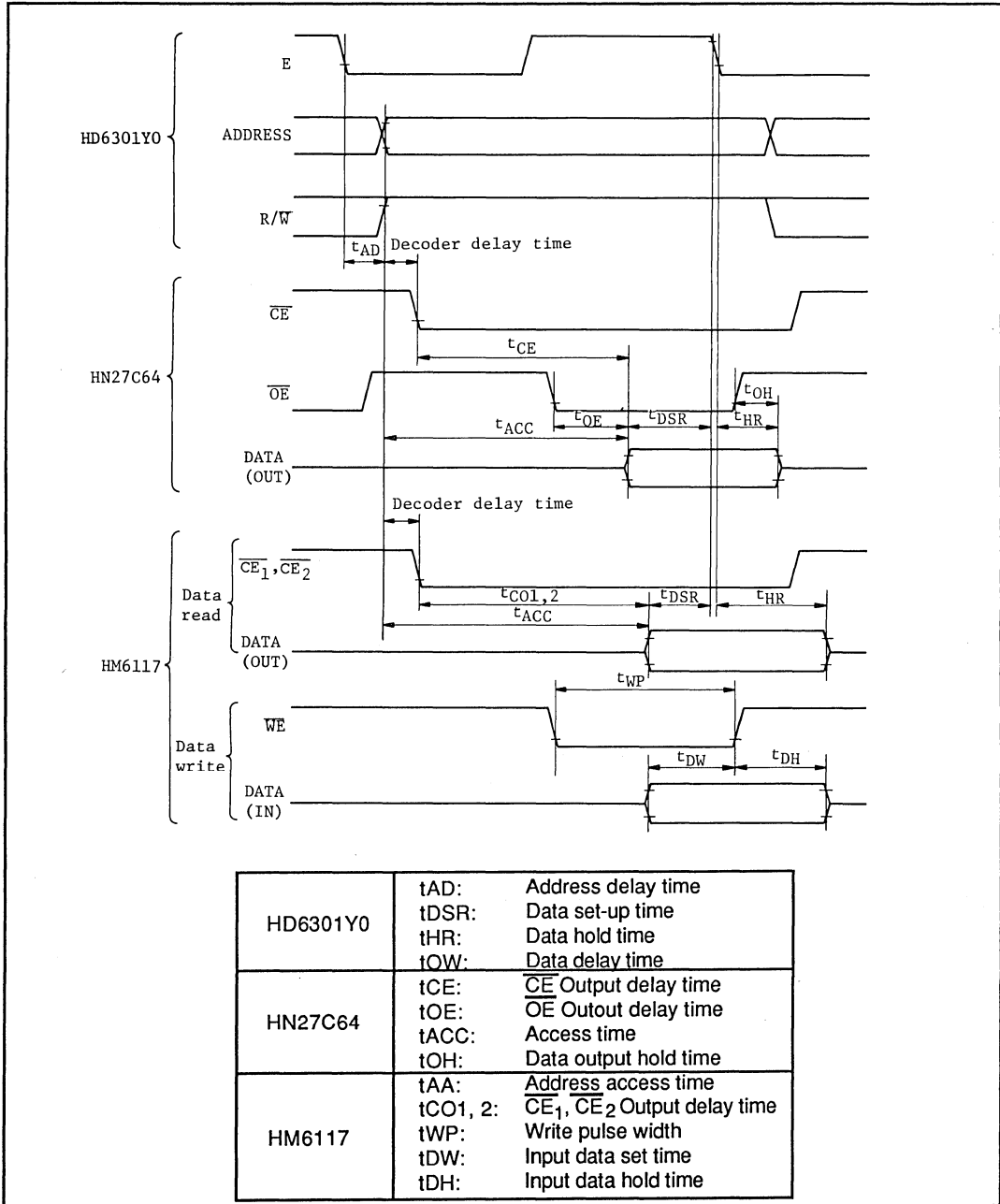


Figure 4-3. Interface Timing Chart for HD6301Y0 and External Memory



## 4.2 Software Description

### 4.2.1 Program Module Configuration

Figure 4-4 shows the program module configuration which displays data input from a console typewriter, using the circuit in Figure 4-1.

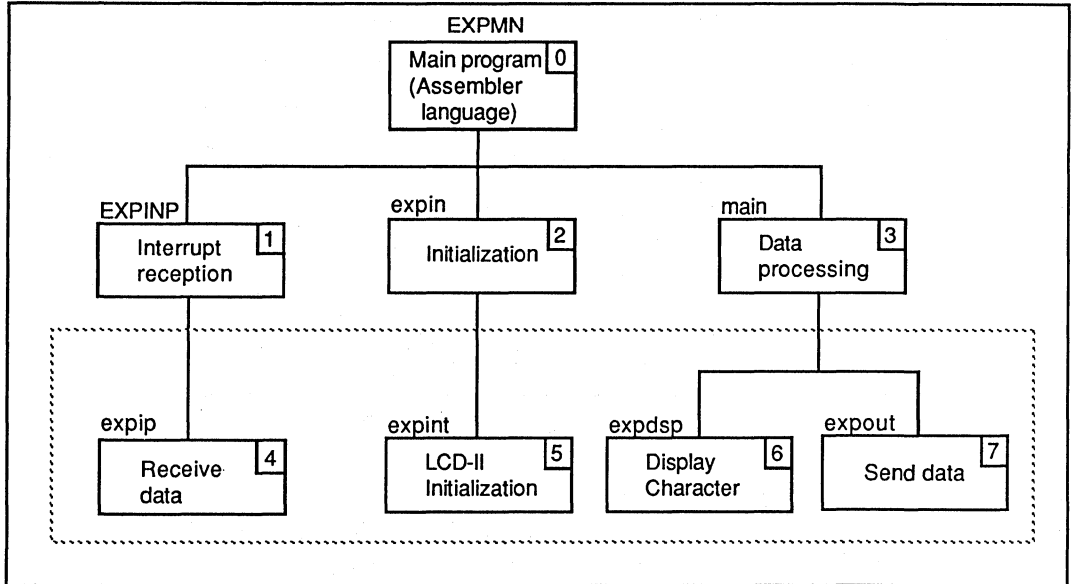


Figure 4-4. Program Module Configuration

Refer to Section 4.3 "Program Module Description" discusses these modules for details.

### 4.2.2 Program Module Functions

Table 4-2 summarizes the program module functions.

**Table 4-2. Program Module Functions**

<b>No.</b>	<b>Program Module Name</b>	<b>Library Function</b>	<b>Function</b>	<b>Language</b>
0	Main program	EXPMN	Initializes instructions, such as ORG, LDS, and CLI, which do not exist in C. Calls expin function and main function	ASM
1	Interrupt reception	EXPINP	Receives and processes IRQ interrupt	ASM
2	Initialization	expin	Initializes global variables, PIA, ACIA, and LCD-II	C
3	Data processing	main	Displays key data, input from console typewriter, on liquid crystal display (H2571) and prints the data on the console typewriter	C
4	Receive data	expip	Receives key data from the console typewriter through an IRQ interrupt	C
5	LCD-II initialization	expint	Initializes LCD-II	C
6	Display Character	expdsp	Displays characters on LCD	C
7	Send data	expout	Sends data to console typewriter	C

Note: C: C Language Program

ASM: Assembly Language Program

### 4.2.3 Program Module Sample Application (Main Program)

The flowchart in Figure 4-5 is an example of the execution sequence of the program module in Figure 4-4 when it displays key data input from a console typewriter on a liquid crystal display and prints the data on the console typewriter.

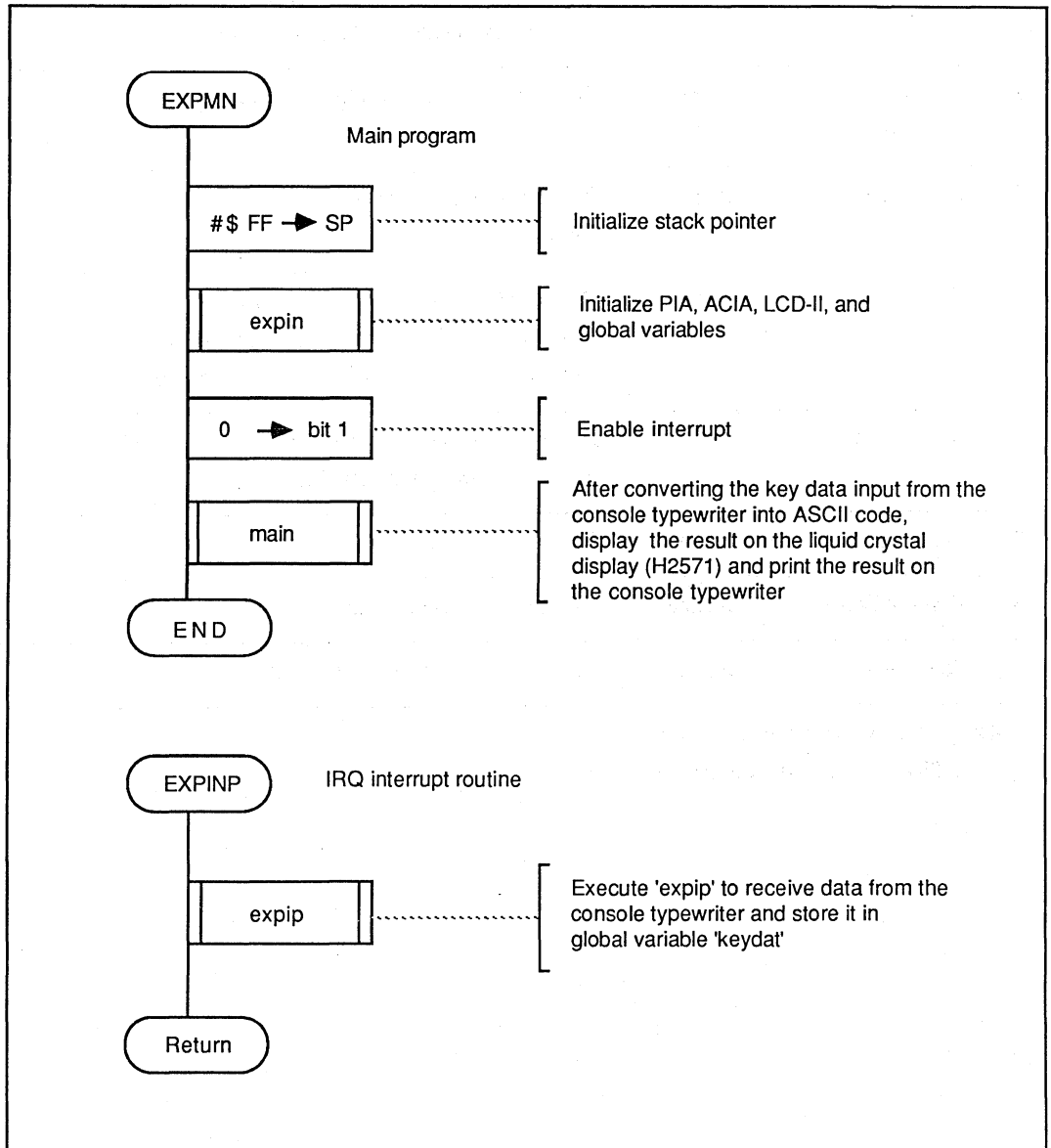


Figure 4-5. Program Module Flowchart

Figure 4-6 shows the execution sequence of C language program 'expin'. In 'expin', key data input from console typewriter is displayed on the liquid crystal display (H2571) and printed on the console typewriter.

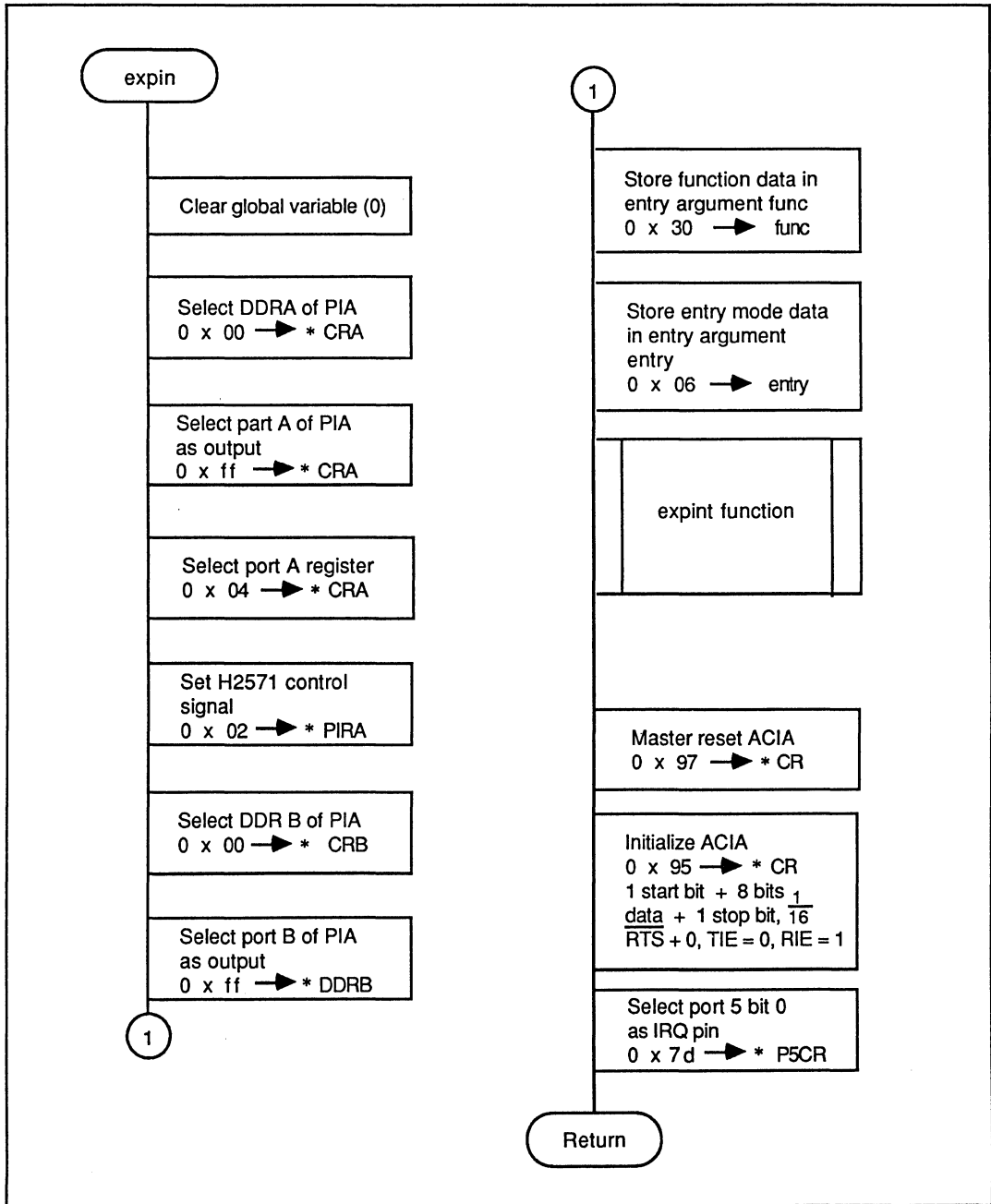


Figure 4-6. Program Module Flowchart

Figure 4-7 shows the execution sequence of C language program 'main'. In 'main', key data input from console typewriter is displayed on the liquid crystal display (H2571) and printed on the console typewriter.

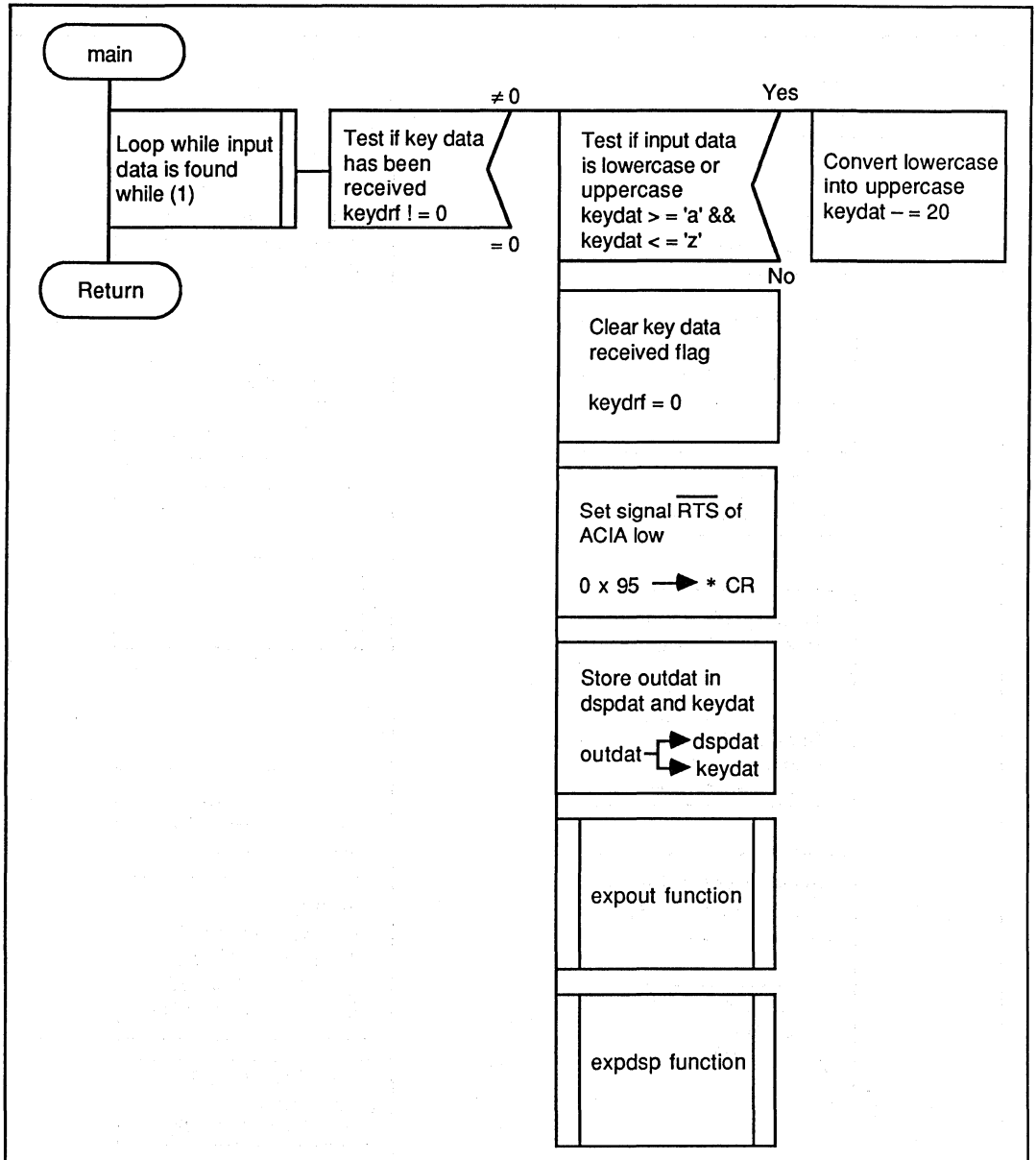


Figure 4-7. Program Module Flowchart

### 4.3 Program Module Description

The following pages describe the external expansion modules.

**Function**

The receive data module receives data from console typewriter and stores key data in global variable 'keydat'.

**Arguments**

Contents		Storage Location	No. of Bytes
Entry	—	—	—
Returns	Received data (ASCII code)	keydat (global variable)	2
	Received data flag	keydrf (global variable)	2

**Libraries Required for Program Execution**

Library		Required/Not Required
Standard Library Function	C31LIB. OBJ	Not required
Run-Time Function	C31RUN. OBJ	Required
	C31RUNF. OBJ	Not required

**Specifications**

ROM (bytes):	48
RAM (bytes):	4
Stack (bytes):	0
No of cycles:	63 (Note)
Reentrant:	No
Relocatable:	No
Interruptible:	No

Note: Ox indicates a hexadecimal number in C.

---

**Description****Function Details**

**Argument details:** Global variable 'keydat' contains 1-byte of key data (ASCII) from the console typewriter. Global variable 'keydrf' is a flag indicating that data has been received. Table 4-3 shows flag functions.

**Example:** Figure 4-8 shows an example of program module 'expip' execution. If key "a" on the console typewriter is pressed as shown in ①, the received data is put in the key data buffer and oxff is stored in 'keydrf' as shown in ②.

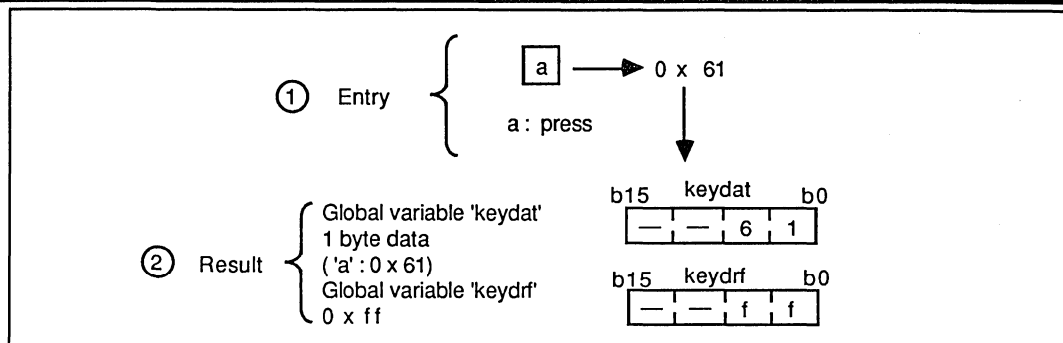


Figure 4-8. Program Module expip Execution Example

Table 4-3. Flag Functions

Variable Name	Flag	Indicates
Keydrf	0x00	No data has been received
	0xff	Data has been received and stored in buffer

User Notes

1. Initialize ACIA because ACIA is controlled by the microcontroller external extension. After initialization ACIA can receive data from the console typewriter.
2. Clear bit I and enables interrupt for  $\overline{IRQ}$  interrupt.

Variable Description

The global variables are stored in static memory (figure 4-9).

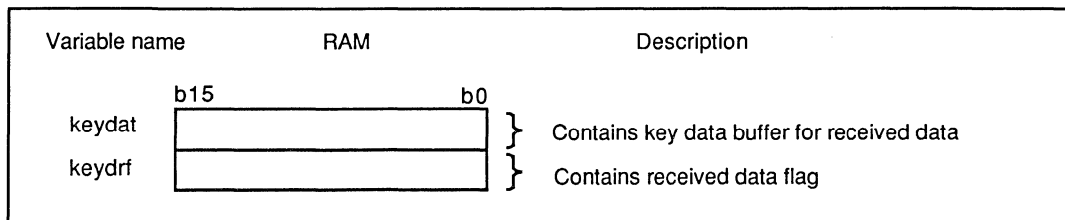


Figure 4-9. Global Variable Storage



Sample Application

After ACIA is initialized and the interrupt is enabled, an  $\overline{\text{IRQ}}$  interrupt initiates program module 'expip' execution (figure 4-10).

```

* CR    = 0 x 97;
* CR    = 0 x 95;
* P5CR  = 0 x 7d;
    }      ..... Initialize ACIA
    }      Select bit 0 of port 5 as the  $\overline{\text{IRQ}}$ ,
    }      ..... interrupt pin
    
```

Figure 4-10. Sample Application

Basic Operation

Figures 4-11 and 4-12 show ACIA control. Figure 4-11 shows ACIA initialization. Figure 4-12 shows now received data is read after an interrupt.

Note that this control method applies to the system in Figure 4-1 and memory map in Figure 4-2.

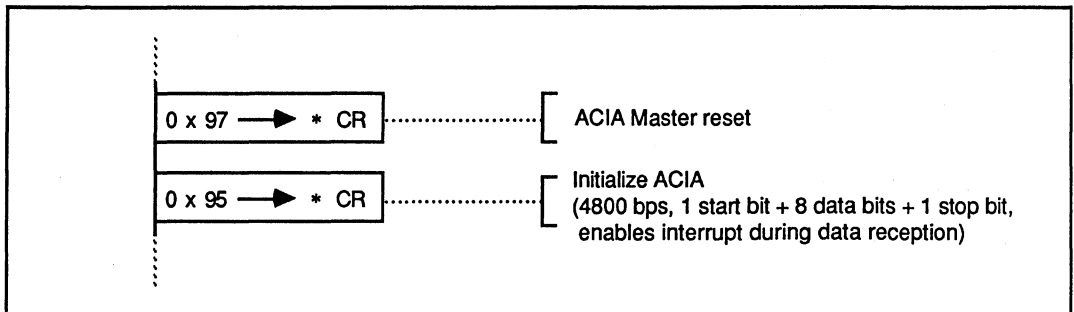


Figure 4-11. ACIA Control (Initialization)

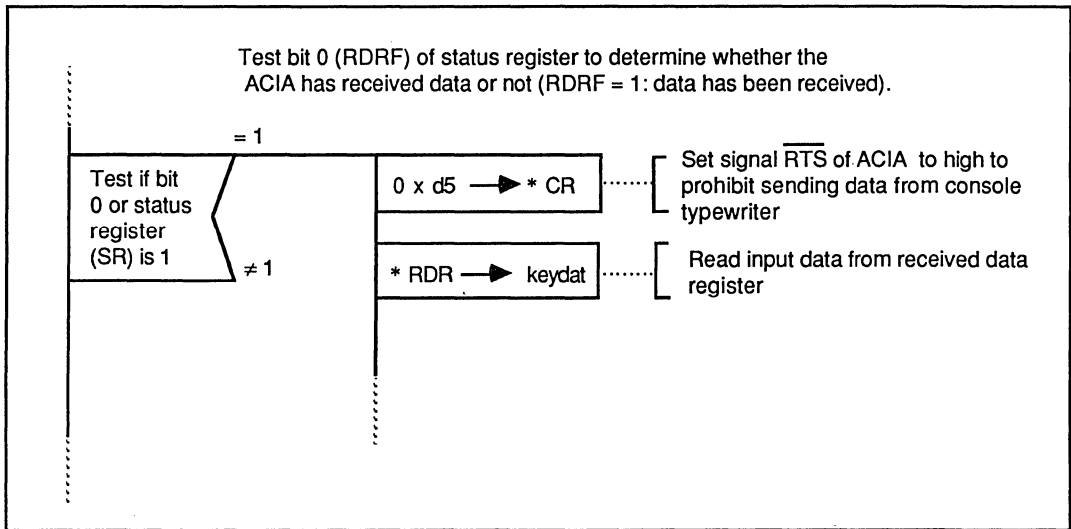


Figure 4-12. ACIA Control (Receiving Serial Data)

When data reception has been completed, set signal  $\overline{\text{RTS}}$  to high to prohibit next data transfer. Finally, store received data from RDR of ACIA in key data buffer.

PAD

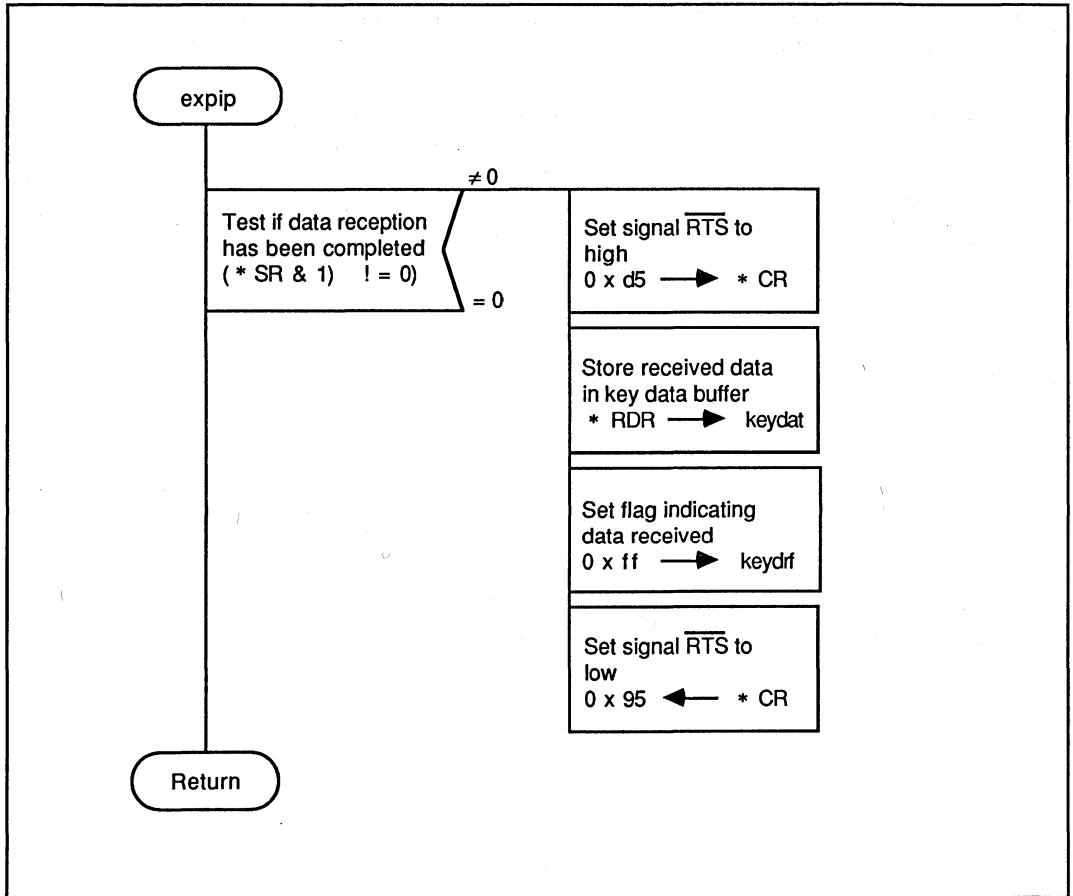


Figure 4-13. Receive Data PAD

**Function**

The send data module sends data to console the typewriter.

**Arguments**

Contents		Storage Location	No. of Bytes
Entry	Data to be sent	outdat (global variable)	2
Returns	—	—	—

**Libraries Required for Program Execution**

Library		Required/Not Required
Standard Library Function	C31LIB. OBJ	Not required
Run-Time Routine	C31RUN. OBJ	Required
	C31RUNF. OBJ	Not required

**Specifications**

ROM (bytes):	15
RAM (bytes):	2
Stack (bytes):	0
No of cycles:	30 (Note)
Reentrant:	No
Relocatable:	No
Interruptible:	Yes

Note: "No. of cycles" indicates the number of cycles required when TDR is empty.

---

**Description****Function Details**

**Argument Details:** Global variable 'outdat' holds data to be sent in ASCII to a console typewriter.

**Example:** Figure 4-14 shows an example of program module 'expout' execution. If entry argument is as shown in ①, the console typewriter prints it as shown in ②.

External Routine: Program module 'expout' does not call any other program modules or subroutines.

User Notes

1. Initialize ACIA because ACIA is controlled by the microcontroller. After initialization the ACIA can transfer data to the console typewriter.
2. If previous data remains in TDR, program module 'expout' will not be executed until TDR is cleared, so as not to destroy the remaining data.

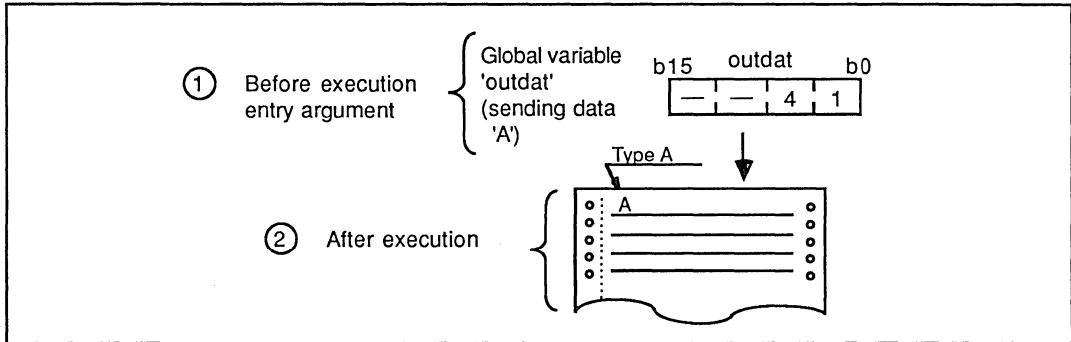


Figure 4-14. Program Module 'expout' Execution Example

Variable Description

The global variable is stored in static memory (figure 4-15).

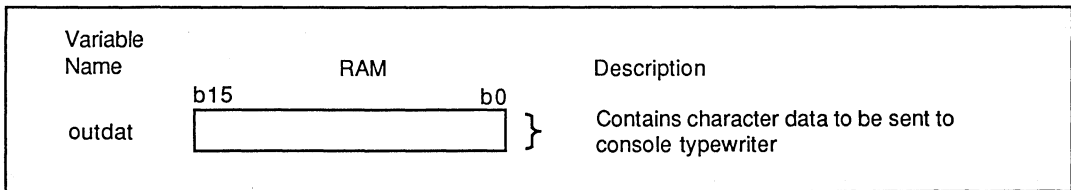


Figure 4-15. Global Variable Storage

## Sample Application

Program module 'expout' is called after ACIA is initialized and data to be sent is stored (figure 4-16).

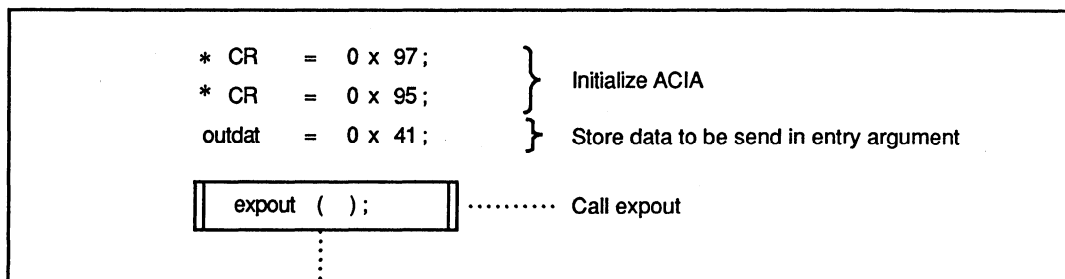


Figure 4-16. Sample Application

## Basic Operation

1. Figure 4-17 shows how to control ACIA to send data.

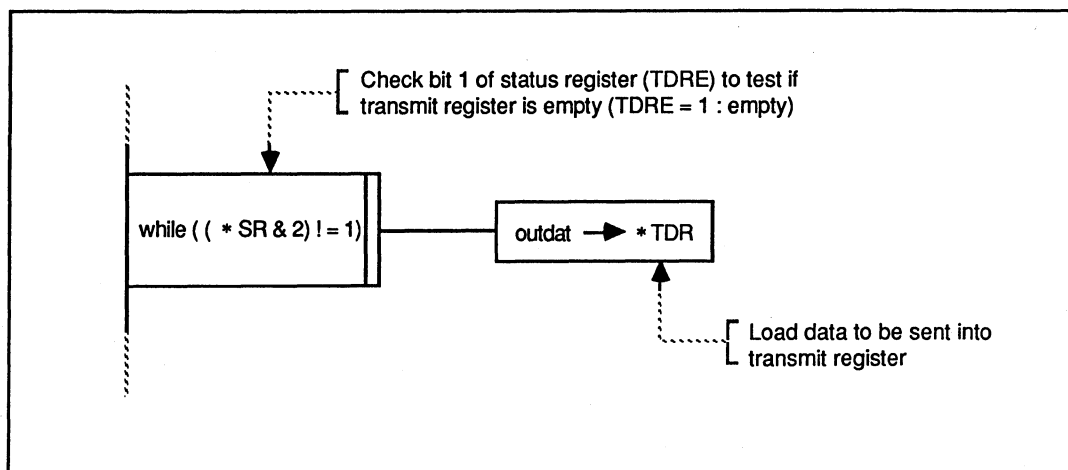


figure 4-17. ACIA Control (Sending Serial Data)

2. Test if bit 1 of status register (TDRE) is "0" or "1". When TDRE is "1", store data to be sent in TDR. When TDRE is "0", wait until TDRE becomes "1", because TDRE = 0 means data remains in TDR.

## PAD

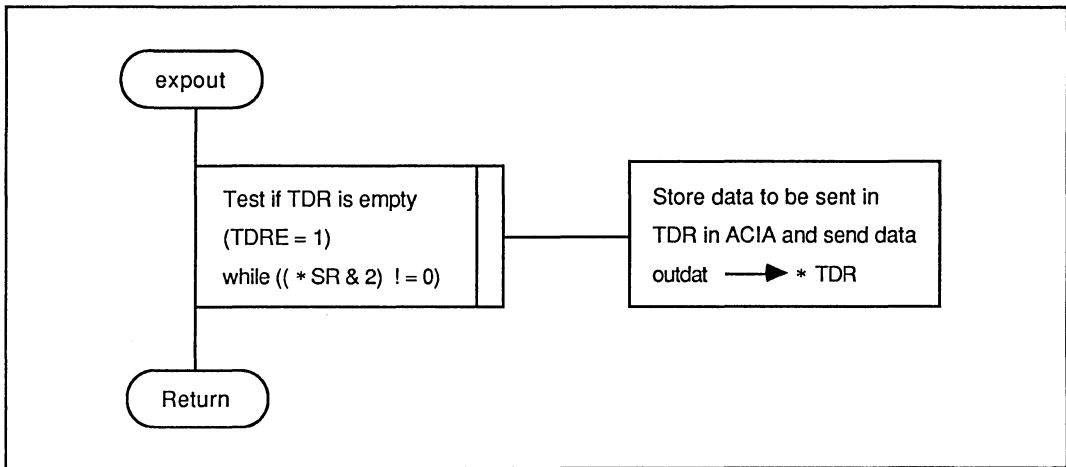


Figure 4-18. Send Data Module PAD



**Function**

The display characters module stores ASCII in (DDRAM) LCD-II display RAM, and displays characters on liquid crystal display.

**Arguments**

Contents	Storage Location	No. of Bytes
Entry     Display data (ASCII)	dspdat (global variable)	2
Returns		

**Libraries Required for Program Execution**

Library		Required/Not Required
Standard Library Function	C31LIB. OBJ	Not required
Run-Time Routine	C31RUN. OBJ C31RUNF. OBJ	Required Not required

**Specifications**

ROM (bytes):	144
RAM (bytes):	2
Stack (bytes):	0
No of cycles:	189 (Note)
Reentrant:	No
Relocatable:	No
Interruptible:	Yes

Note: "No. of cycles" in "Specifications" indicates the number of cycles when subroutine expdsp executes in the minimum cycles.

Description

Function Details

Argument Details: Global variable 'dspdat' holds display data as 1 ASCII byte.

Example: Figure 4-19 shows an example of program module 'expdsp' execution. If entry argument is as shown in ①, 'expdsp' displays characters on the liquid crystal as shown in ②.

External Routine: Program module 'expdsp' calls other program modules and subroutines, as shown in table 4-4.

Table 4-4. Program Modules and Subroutines Called in 'expdsp'

Program Module/ Subroutine Name	Function Name	Function
CHECK BUSY FLAG	expbsy	Checks LCD-II busy flag

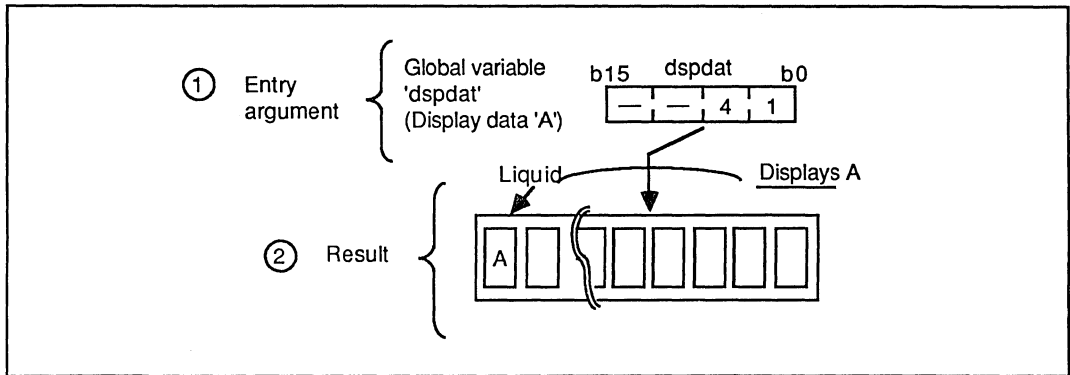


Figure 4-19. Program Module expdsp Execution Example

## User Notes

1. Initialize PIA because PIA is controlled by external extension, and LCD-II is controlled using PIA port.
2. Initialize LCD-II by executing program module expint.

## Variable Description

The global variables are stored in static memory (figure 4-20).

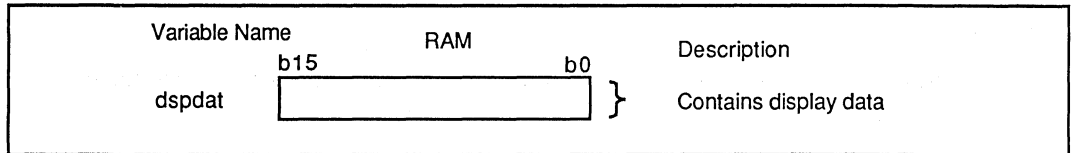


Figure 4-20. Global Variable Storage

## Sample Application

Figure 4-21 shows a sample application using 'expdsp'.

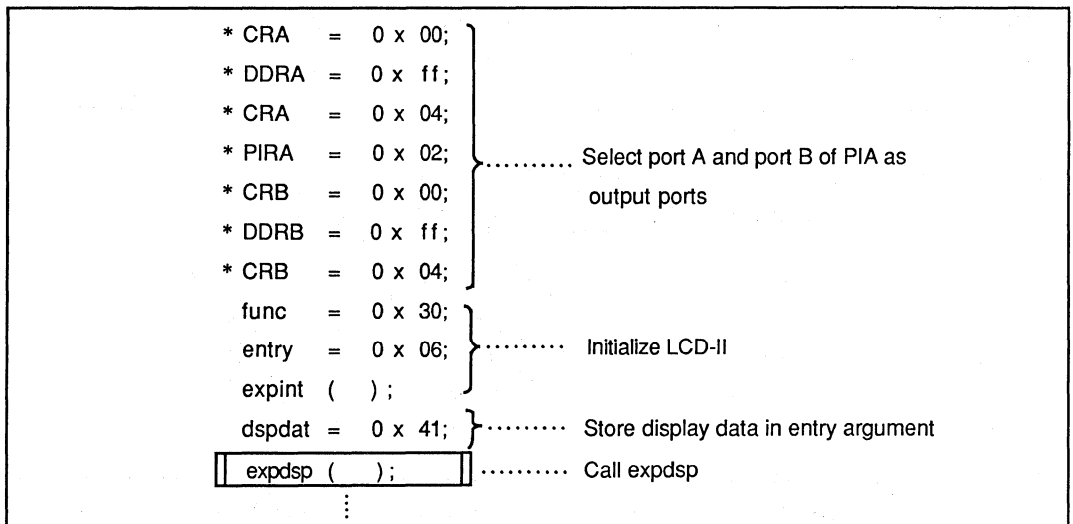


Figure 4-21. Sample Application

## Basic Operation

1. Calls subroutine expbsy and waits until LCD-II can receive instructions.
2. When LCD-II can receive instructions, the routine controls signals RS, R $\bar{W}$ , and E in LCD-II using PIA port A, and the display data, stored in port B of the PIA, in LCD-II to display characters on a liquid crystal display.

## PAD

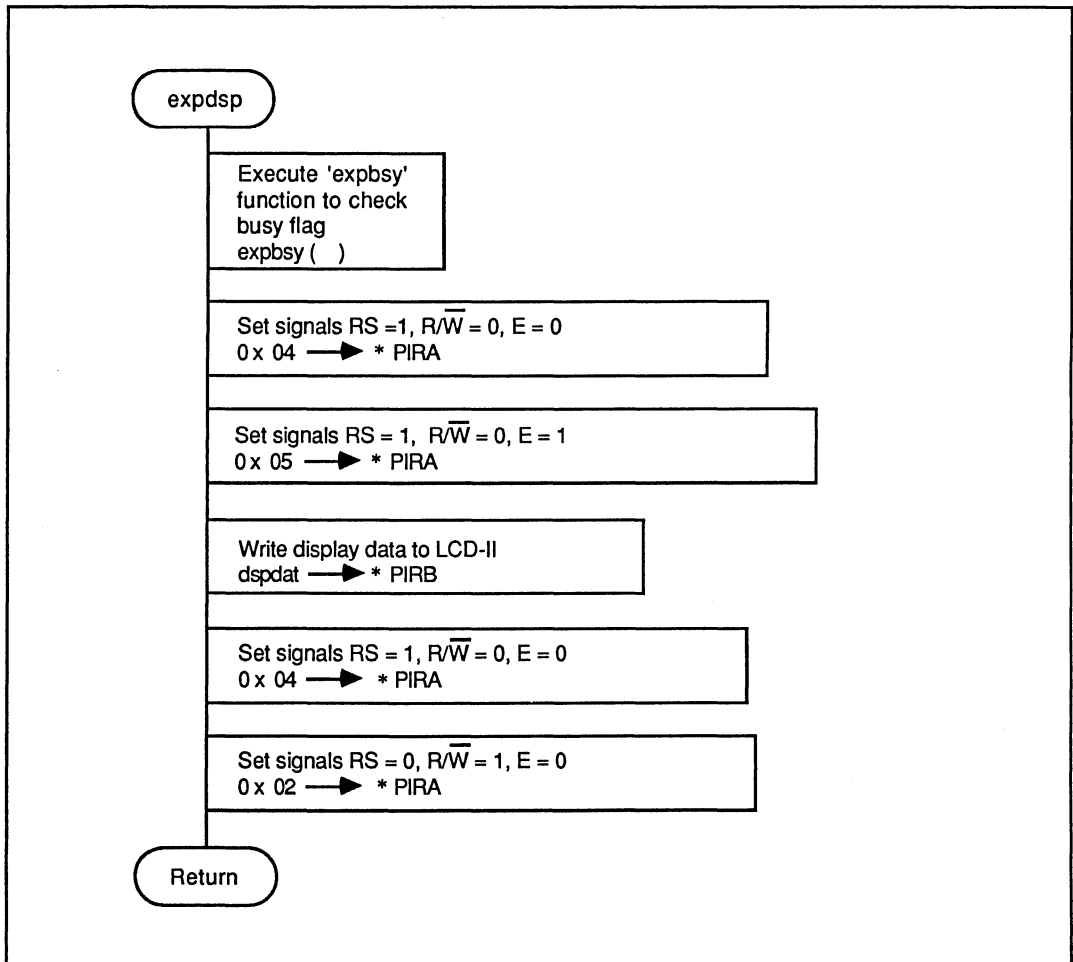


Figure 4-22. Display Characters Module PAD

**Function**

The initialize LCD-II module initializes the LCD-II.

**Arguments**

Contents		Storage Location	No. of Bytes
Entry	Function data	func (global variable)	2
	Entry mode data	entry (global variable)	2

Returns

**Libraries Required for Program Execution**

Library		Required/Not Required
Standard Library Function	C31LIB. OBJ	Not required
Run-Time Routine	C31RUN. OBJ	Required
	C31RUNF. OBJ	Not required

**Specifications**

ROM (bytes):	151
RAM (bytes):	8
Stack (bytes):	4
No of cycles:	541,162 (Note)
Reentrant:	No
Relocatable:	No
Interruptible:	Yes

Note: "No of cycles" in "Specifications" indicates the number of cycles needed when subroutine expbsy executes in the minimum number of cycles.

**Description****Function Details**

**Argument Details:** Global variable 'func' holds function data (0 x 30) for LCD-II instructions

Global variable 'entry' holds entry mode data (0 x 60) for LCD-II instruction.

**Functions:** Program module expint initializes LCD-II with instructions, clears display and selects the following functions.

- Interface data length: 8 bits.
- Display line: 1.
- Character font: 5 x 7 dots.
- Duty rate: 1/8.
- DDRAM address increment.
- Display shift: no.

**External Routines:** Program module expin calls other program modules and subroutines as shown in table 4-5.

**Table 4-5. Program Modules and Subroutines Called in 'expint'**

Program Module/ Subroutine Name	Function Name	Function
STORE INSTRUCTIONS	setins	Stores instructions in LCD-II
CHECK BUSY FLAG	expbsy	Checks LCD-II busy flag

User Notes

Initialize PIA to control LCD-II through PIA external expansion I/O ports.

Variable Description

The global variables are stored in static memory (figure 4-23).

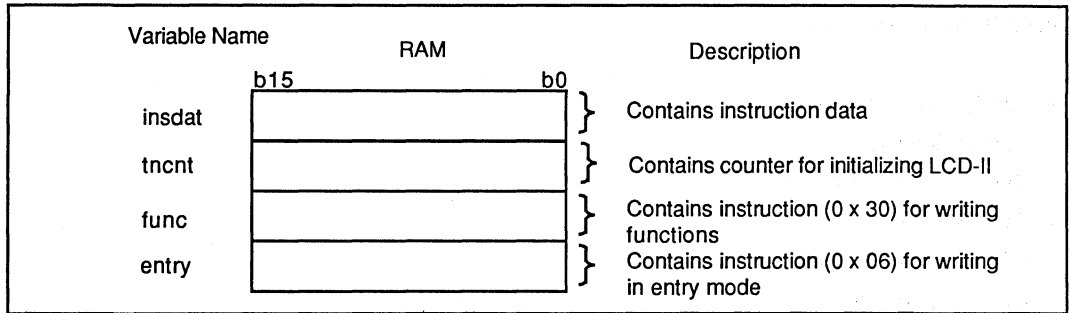


Figure 4-23. Global Variable Storage

Sample Application

Call program module 'expint' after initializing PIA and storing entry argument (figure 4-24).

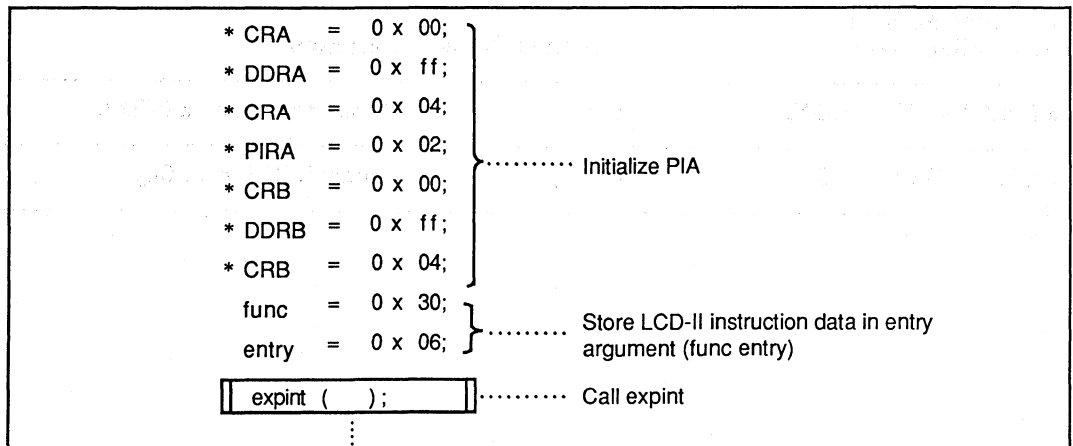


Figure 4-24. Sample Application

Basic Operation

Figure 4-25 and 4-26 show PIA control.

Control of PIA is shown in Figs. 12 and 13.

In figure 4-25, port A outputs data (0 x 80). In figure 4-26, port A inputs data.

Note that this control method applies to the circuit diagram in figure 4-1, and the Memory Map in Figure 4-2.

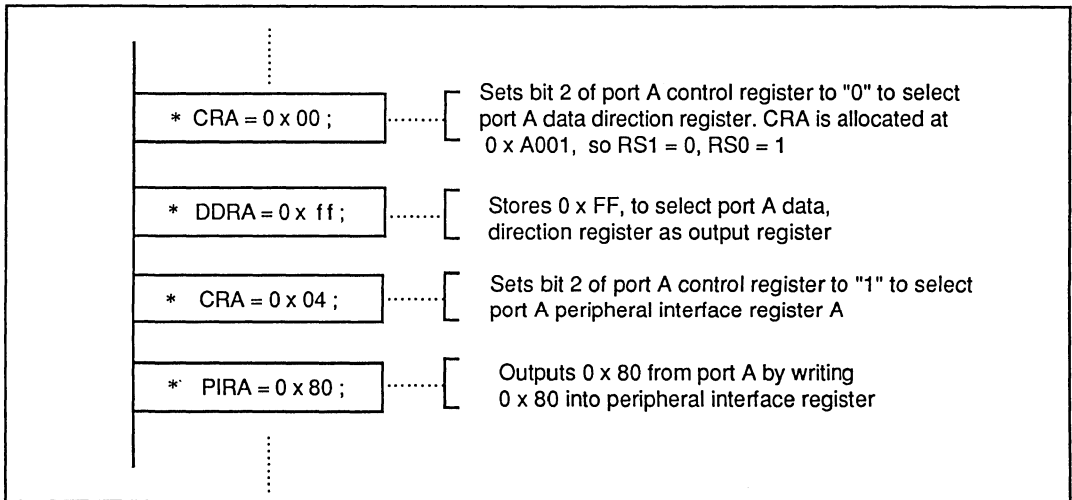


Figure 4-25. PIA Control (Port A: Output port)

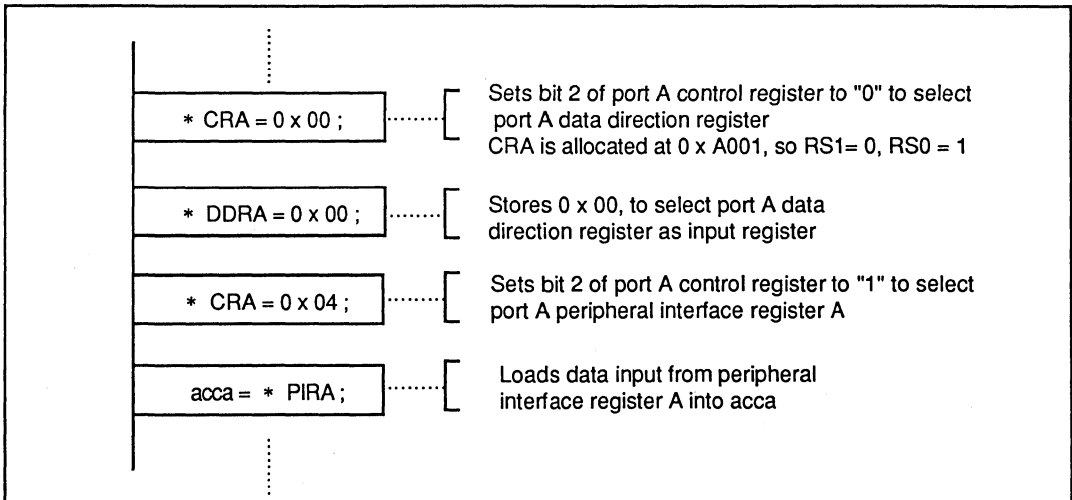


Figure 4-26. PIA Control (Port A: Input port)



Function data (0 x 30) must be written 3 times into LCD-II as shown below to ensure LCD-II internal reset. Afterwards, the LCD-II busy flag can be checked to select function (figure 4-27).

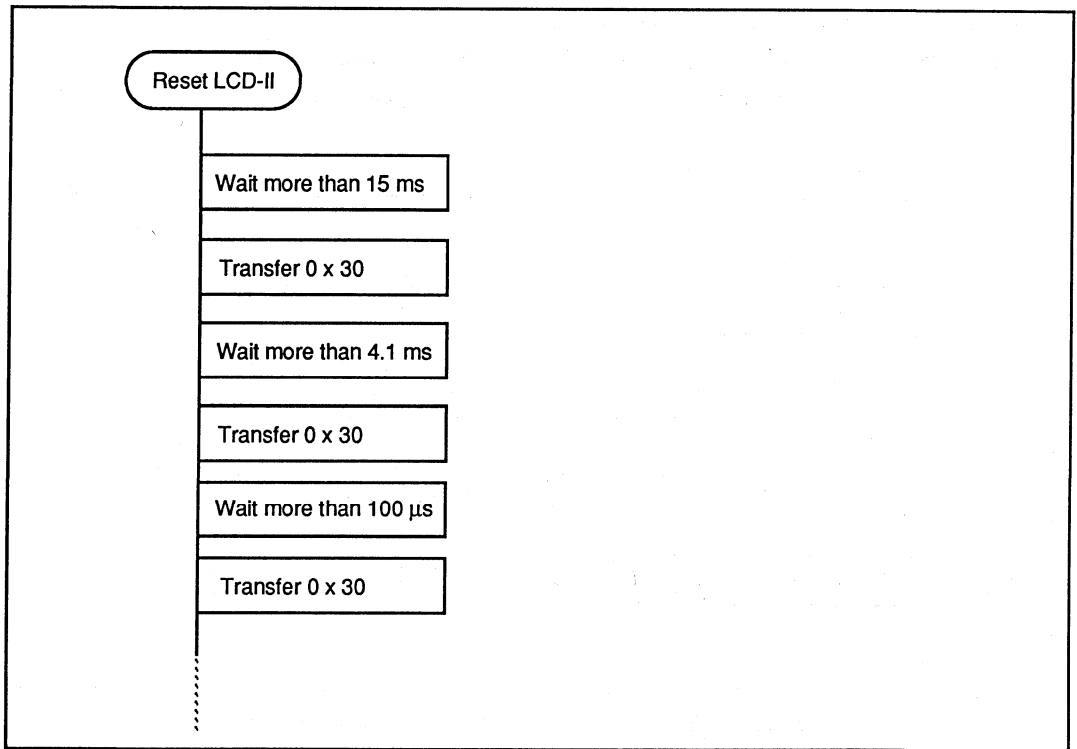


Figure 4-27. LCD-II Reset

Data shown in table 4-6 is transferred to LCD-II using subroutine 'expins'.

Table 4-6. LCD-II Initialization

Data	Function
0 x 30	Interface length: 8 bits
0 x 01	Clears display, sets DDRAM address to 0 x 00
0 x 08	Turns off display
0 x 06	Specifies cursor direction right Does not shift display

PAD

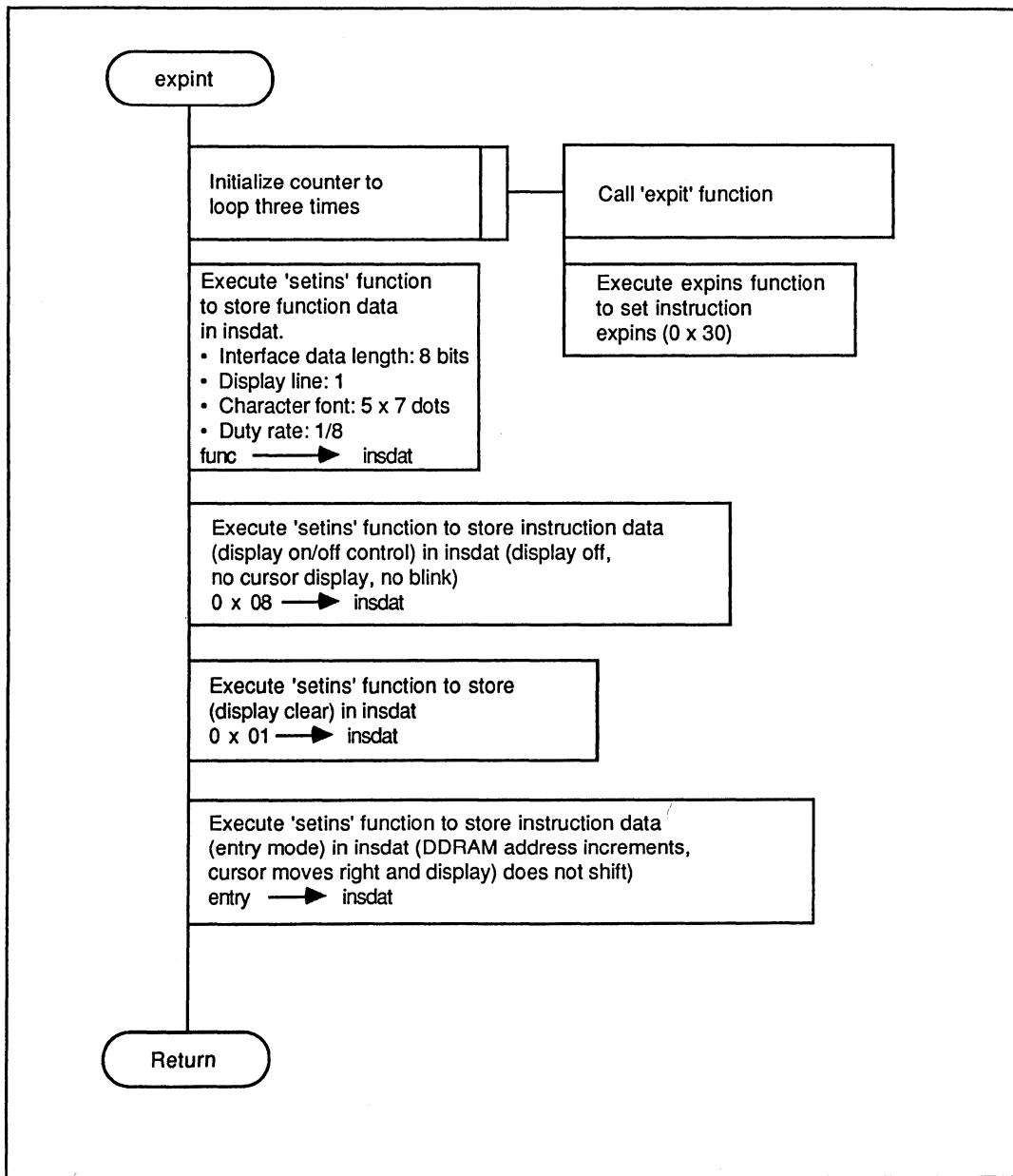


Figure 4-28. Instruction LCD-II Module PAD

**Function**

The software timer subroutine times a 15 ms delay used in LCD-II initialization.

**Basic Operation**

The software timer uses a register to calculate the delay.

**PAD****Program Module That Uses This Function**

The 'expint' function uses the 'expit' subroutine.

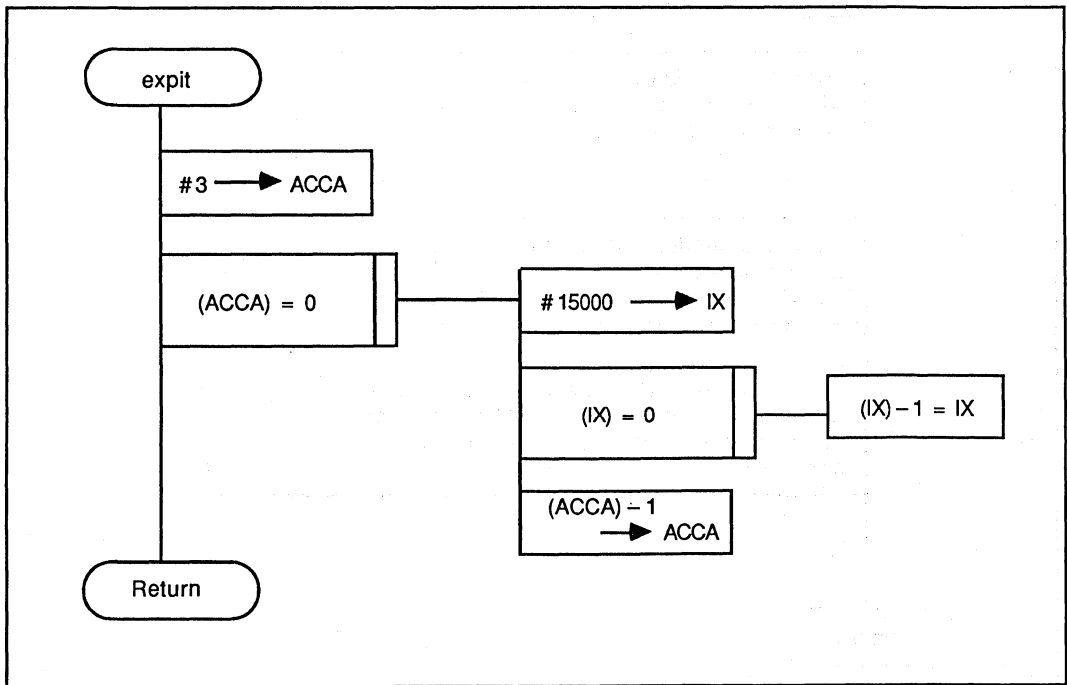


Figure 4-29. Software Timer Subroutine PAD

**Function**

The check busy flag subroutine checks the LCD-II to get if it is in operation. When the LCD-II is in operation, subroutine expbsy loops.

**Basic Operation**

When LCD-II is in operation, it cannot receive data from the MCU. The check busy flag subroutine checks the busy flag to determine the LCD-II operating condition.

When signals RS, R/W, and E are set to low, high, and high respectively, the most significant bit of the LCD-II data bus (DB<sub>0</sub>-DB<sub>7</sub>) becomes the busy flag.

Executes subroutine expbsy while busy flag is "1".

In case of "0", it goes to the next process.

**PAD****Program Module That Use This Function**

The 'setins' and 'expdsp' functions use the 'expbsy' subroutine.

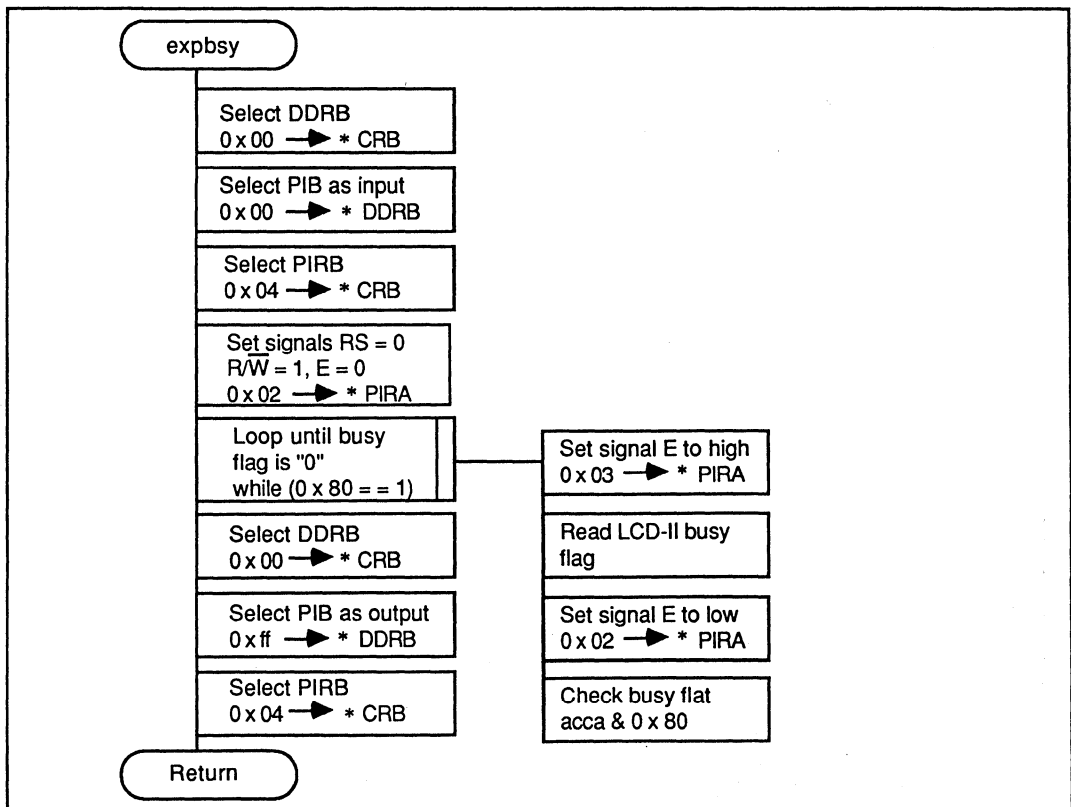


Figure 4-30. Check Busy Subroutine PAD

**Function**

The LCD-II instruction set subroutine stores instructions in the LCD-II by setting control signals (RS,  $\overline{R/W}$ , F) in LCD-II and the data bus using the PIA I/O port.

**Basic Operation**

The LCD-II instruction set subroutine outputs data from PIA port A to set signals RS,  $\overline{R/W}$ , and E to low.

When signal E is set from high to low, data that is stored in port B of PIA is stored in the LCD-II.

**PAD****Program Module That Uses This Function**

The 'expint' function uses the 'expins' subroutine.

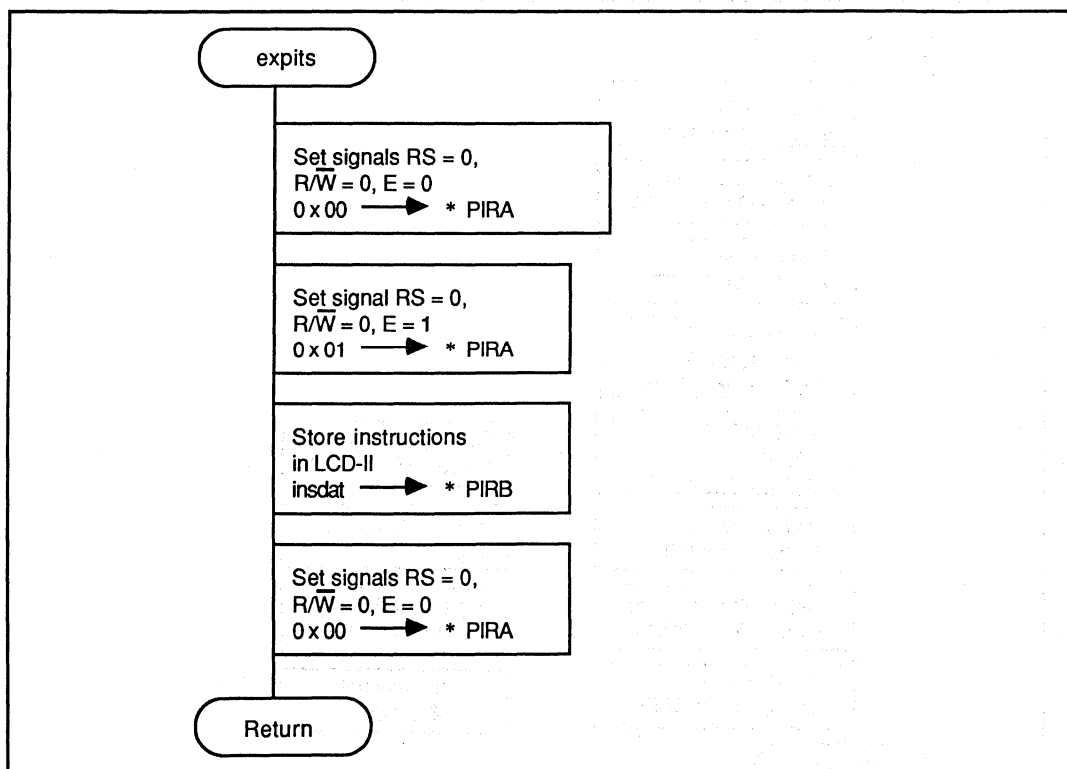


Figure 4-31. LCD-II Instruction Set Subroutine PAD

**Function**

The LCD-II data transfer/receive subroutine checks the LCD-II busy flag and stores instruction data in insdat.

**Basic Operation**

The LCD-II data transfer/receive subroutine calls the 'expbsy' function to check the busy flag. It then executes the 'expins' function to store instruction data in 'expins'.

**PAD****Program Module That Use This Function**

The 'expin' and 'expint' functions use the 'setins' subroutine.

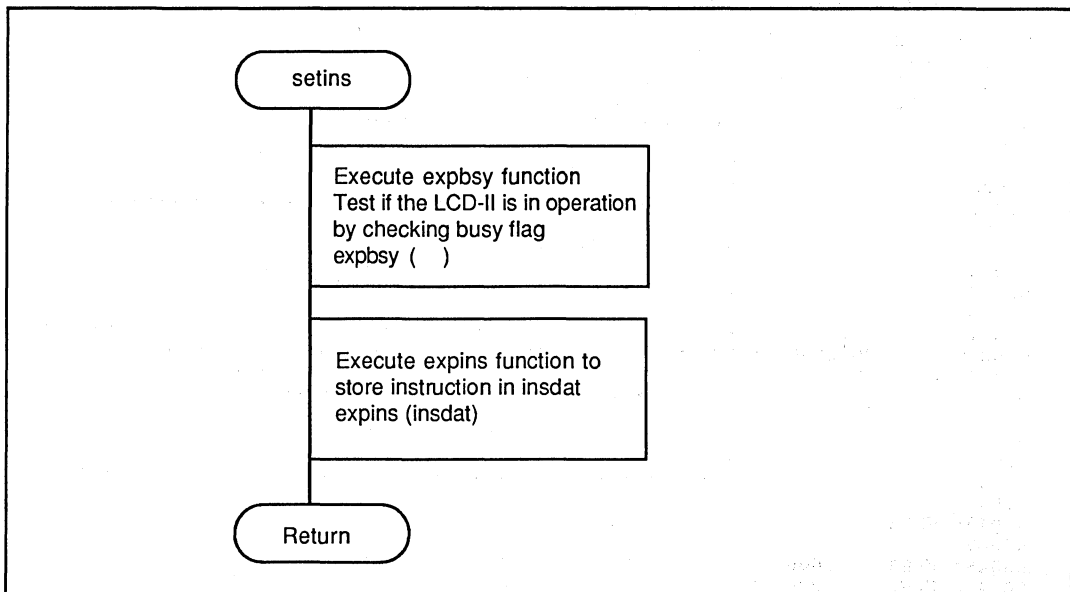


Figure 4-32. LCD-II Data Transfer/Receive Subroutine PAD

## 4.4 Program Listing

### 4.4.1 Main Program Listing

\*\*\* CP/M-68K 6301/6801/6800 CROSS MACROASSEMBLER V1.2 \*\*\*

```

ERR  SEQ  LOC  OBJECT  PROGRAM
00001 *****
00002 *
00003 *
00004 *          MAIN PROGRAM : EXPMN
00005 *
00006 *****
00007          OPT    REL
00008          XREF   PSCT:MAIN,PSCT:EXPIP,PSCT:EXPIN
00009          XDEF   EXPIT
00010          PSCT
00010P 0000
00011P 0000 8E 00FF A EXPMN LDS    #$FF    Set stack pointer
00012P 0003 BD 0000 A      JSR    EXPIN   Initialize PIA, ACIA, and LCD-II
00013P 0006 0E          CLI          Enable interrupt
00014P 0007 BD 0000 A      JSR    MAIN    Branch to main routine
00015 *****
00016 *
00017 *          NAME : EXPIT (INITIALIZE LCD-2)
00018 *
00019 *****
00020P 000A 86 03 A EXPIT LDAA   #3      Execute 15ms software timer
00021P 000C CE 3A98 A EXPIT1 LDX   #15000
00022P 000F 09 EXPIT2 DEX
00023P 0010 26 FD 000F BNE    EXPIT2
00024P 0012 4A DECA
00025P 0013 26 F7 000C BNE    EXPIT1
00026P 0015 39 RTS
00027 *****
00028 *
00029 *          NAME : EXPIP (RECEIVE DATA)
00030 *
00031 *****
00032 *
00033 *          ENTRY : NOTHING
00034 *          RETURNS : KEYDAT (RECEIVED DATA)
00035 *                   : KEYDRF (RECEIVED FLAG)
00036 *
00037 *****
00038P 0016 BD 0000 A EXPINP JSR    EXPIP   Receive data from console
00039P 0019 3B      RTI          Return from interrupt
00040 *****
00041 *
00042 *          VECTOR ADDRESS
00043 *
00044 *****
00045 *
00046A FFEA          ORG    $FFEA
00047 *
00048A FFEA 0000 P      FDB    EXPMN   IRQ2
00049A FFEC 0000 P      FDB    EXPMN   CMI
00050A FFEE 0000 P      FDB    EXPMN   TRAP
00051A FFF0 0000 P      FDB    EXPMN   SIO
00052A FFF2 0000 P      FDB    EXPMN   TOI
00053A FFF4 0000 P      FDB    EXPMN   OCI
00054A FFF6 0000 P      FDB    EXPMN   ICI
00055A FFF8 0016 P      FDB    EXPINP  IRQ1
00056A FFFA 0000 P      FDB    EXPMN   SWI

```

\*\*\* CP/M-68K 6301/6801/6800 CROSS MACROASSEMBLER V1.2 \*\*\*

ERR	SEQ	LOC	OBJECT	PROGRAM			
	00057A	FFFC	0000	P	FDB	EXPMN	NMI
	00058A	FFFE	0000	P	FDB	EXPMN	RES
	00059			*			
	00060				END		
****	TOTAL ERRORS 00000--00000						



## 4.4.2 C Source Listing

```

*
/*****DECLARATION OF DEFINE*****/
/*
#define P5CR      ((char*)0x14)      /* Port5 control register */
#define DDRA      ((char*)0xA000)    /* Data direction register A(PIA) */
#define CRA       ((char*)0xA001)    /* Control register A(PIA) */
#define DDRB      ((char*)0xA002)    /* Data direction register B(PIA) */
#define CRB       ((char*)0xA003)    /* Control register B(PIA) */
#define PIRA      DDRA               /* Peripheral register A(PIA) */
#define PIRB      DDRB               /* Peripheral register B(PIA) */
#define CR        ((char*)0xC000)    /* Control register (ACIA) */
#define SR        CR                 /* Status register (ACIA) */
#define RDR       ((char*)0xC001)    /* Receive data register (ACIA) */
#define TDR       RDR                /* Transmit data register (ACIA) */
/*
/*****DECLARATION OF GLOBAL VARIABLES*****/
/*
static direct int   outdat; /* Transmit data */
static direct int   dspdat; /* Display data */
static direct int   keydrf; /* Flag of receive data */
static direct int   keydat; /* Receive data */
static direct int   tncnt;  /* Counter for initializing LCD-II */
static direct int   func;   /* Function data */
static direct int   entry;  /* Entry mode data */
/*****
/*
/*      MAIN ROUTINE : MAIN (DISPLAY INPUT DATA FROM CONSOLE ON BOTH LCD-2 */
/*                                AND CONSOLE) */
/*
/*
/*****
main() /* Display input data from console on both LCD-II and console */
{
    while (1) { /* Continuous loop */
        if (keydrf!=0) { /* Test if data is received */
            if (keydat>='a' && keydat<='z')
                keydat-=0x20; /* Change lower case to upper */
            keydrf=0; /* Clear flag of receive data */
            *CRA=0x95; /* Set RTS=low */
            outdat=dspdat=keydat; /* Set output data in area */
            expout(); /* Transmit data to console */
            expdsp(); /* Display characters on LCD-II */
        }
    }
}
/*****
/*
/*      NAME : EXPIN (INITIALIZE PIA,ACIA AND LCD-2) */
/*
/*
/*****
expin()
{
    outdat=dspdat=keydrf=keydat=tncnt=func=entry=0; /* Initialize */
    *CRA =0x00; /* Select data direction register A */
    *DDRA=0xff; /* Select port A as output */
    *CRA =0x04; /* Select peripheral register A */
    *PIRA=0x02; /* Set RS=0, R/W=1, E=0 */
    *CRB =0x00; /* Select data direction register B */
    *DDRB=0xff; /* Select port B as output */
    *CRB =0x04; /* Select peripheral register B */
    func=0x30; /* Set function data */
    entry=0x06; /* Set entry mode data */
    expint(); /* Initialize LCD-II */
    setins(0x0e); /* Set instruction to LCD-II */
    *CR=0x97; /* Master reset of ACIA */
    *CR=0x95; /* Initialize ACIA */
    *P5CR=0x7d; /* Initialize port 5 */
}

```

```

}
/*****
/*
/*      NAME : EXPIP   (RECEIVE DATA)
/*
/*****
/*
/*      ENTRY   : NOTHING
/*      RETURNS  : KEYDAT (RECEIVED DATA)
/*                : KEYDRF (RECEIVED FLAG)
/*
/*****
expip()
{
    if ((*SR&1)!=0) {          /* Test if data is received */
        *CR=0xd5;             /* Set RTS=high */
        keydat = *RDR;        /* Set receive data */
        keydrf=0xff;         /* 0xff if receive data is set */
        *CR=0x95;            /* Set RTS=low */
    }
}
/*****
/*
/*      NAME : EXPINT  (INITIALIZE LCD-2)
/*
/*****
/*
/*      ENTRY   : FUNC   (FUNCTION DATA)
/*                : ENTRY (ENTRY MODE DATA)
/*      RETURNS  : NOTHING
/*
/*****
expint()
{
    for(tncnt=0;tncnt<3;tncnt++) { /* Reset LCD-II three times */
        expit();                  /* Execute 15ms software timer */
        expins(0x30);             /* Write function data to LCD-II */
    }
    *PIRA=0x02;                  /* Set R/W=1 */
    setins(func);                /* Set function data to LCD-II */
    setins(0x08);                /* Set instruction (display off) */
    setins(0x01);                /* Set instruction (display clear) */
    setins(entry);               /* Set entry mode data */
}
/*****
/*
/*      NAME : EXPDSP  (DISPLAY CHARACTERS)
/*
/*****
/*
/*      ENTRY   : DSPDAT (DISPLAY DATA)
/*      RETURNS  : NOTHING
/*
/*****

```

```

expdsp()
{
    expbsy();
    *PIRA=0x04;          /* Check busy flag */
    *PIRA=0x05;          /* Set RS=1, R/W=0, E=0 */
    *PIRB=dspdats;      /* Set E=1 */
    *PIRA=0x04;          /* Output data to LCD-II */
    *PIRA=0x02;          /* Set E=0 */
    *PIRA=0x02;          /* Set R/W=1 */
}
/*****
/*
/*      NAME : EXPDOUT  (SEND DATA)
/*
/*****
/*
/*      ENTRY  : OUTDAT  (DATA TO BE SENT)
/*
/*      RETURNS : NOTHING
/*
/*****
expout()
{
    while((*SR&2)!=0)    /* Transmission has been completed */
        *TDR=outdat;    /* Set transmit data in TDR */
}
/*****
/*
/*      NAME : EXPBSY  (CHECK BUSY FLAG)
/*
/*****
expbsy()
{
    int      acca=0x80;
    *CRB=0x00;          /* Set data direction register B */
    *DDR=0x00;          /* Select port B as input */
    *CRB=0x04;          /* Select peripheral register B */
    *PIRA=0x02;          /* Set RS=0, R/W=1, E=0 */

    do {
        *PIRA=0x03;      /* Set E=1 */
        acca=*PIRB;      /* Set PIRB in working area */
        *PIRA=0x02;      /* Set E=0 */
        acca &= 0x80;    /* Read busy flag */
    } while (acca==0x80);
    *CRB=0x00;          /* Select data direction register B */
    *DDR=0xff;          /* Select port B as output */
    *CRB=0x04;          /* Select peripheral register B */
}
/*****
/*
/*      NAME : EXPINS  (STORE INSTRUCTION)
/*
/*****
expins(insdat)
int      insdat;
{
    *PIRA=0x00;          /* Set RS=0, R/W=0, E=0 */
    *PIRA=0x01;          /* Set E=1 */
    *PIRB=insdat;        /* Set instruction in peripheral B */
    *PIRA=0x00;          /* Set E=0 */
}

```

```
}
/*****
/*
/*      NAME : SETINS (SET INSTRUCTION TO LCD-2)
/*
/*
/*****
setins(insdat)
int      insdat;
{
      expbsy();
      expins(insdat);
}
```

### 4.4.3 Output Object Listing of C Compiler

\*\*\* CP/M-68K 6301/6801/6800 CROSS MACROASSEMBLER V1.2 \*\*\*

ERR	SEQ	LOC	OBJECT	PROGRAM	EXPC
	00001			NAM	EXPC
	00002			OPT	REL
	00003		MSEX	MACR	
	00004		CLRA		
	00005		TSTB		
	00006		BPL \.0		
	00007		COMA		
	00008		\.0 EQU *		
	00009		ENDM		
	00010		MLBRA	MACR	
	00011		JMP \0		
	00012		ENDM		
	00013		MLBSR	MACR	
	00014		JSR \0		
	00015		ENDM		
	00016		MLBEQ	MACR	
	00017		BNE \.0		
	00018		JMP \0		
	00019		\.0 EQU *		
	00020		ENDM		
	00021		MLBNE	MACR	
	00022		BEQ \.0		
	00023		JMP \0		
	00024		\.0 EQU *		
	00025		ENDM		
	00026		MLBGT	MACR	
	00027		BLE \.0		
	00028		JMP \0		
	00029		\.0 EQU *		
	00030		ENDM		
	00031		MLBGE	MACR	
	00032		BLT \.0		
	00033		JMP \0		
	00034		\.0 EQU *		
	00035		ENDM		
	00036		MLBLT	MACR	
	00037		BGE \.0		
	00038		JMP \0		
	00039		\.0 EQU *		
	00040		ENDM		
	00041		MLBLE	MACR	
	00042		BGT \.0		
	00043		JMP \0		
	00044		\.0 EQU *		
	00045		ENDM		
	00046		MLBHI	MACR	
	00047		BLS \.0		
	00048		JMP \0		
	00049		\.0 EQU *		
	00050		ENDM		
	00051		MLBLS	MACR	
	00052		BHI \.0		
	00053		JMP \0		
	00054		\.0 EQU *		
	00055		ENDM		
	00056		MLBCC	MACR	

ERR	SEQ	LOC	OBJECT	PROGRAM	EXPC
	00057			BCS \.0	
	00058			JMP \0	
	00059			\.0 EQU *	
	00060			ENDM	
	00061			MLBCS MACR	
	00062			BCC \.0	
	00063			JMP \0	
	00064			\.0 EQU *	
	00065			ENDM	
	00066B	0000		BSCT	
	00067B	0000	0002 A	OUTDAT BSZ	2
	00068B	0002		BSCT	
	00069B	0002	0002 A	DSPDAT BSZ	2
	00070B	0004		BSCT	
	00071B	0004	0002 A	KEYDRF BSZ	2
	00072B	0006		BSCT	
	00073B	0006	0002 A	KEYDAT BSZ	2
	00074B	0008		BSCT	
	00075B	0008	0002 A	TNCNT BSZ	2
	00076B	000A		BSCT	
	00077B	000A	0002 A	FUNC BSZ	2
	00078B	000C		BSCT	
	00079B	000C	0002 A	ENTRY BSZ	2
	00080P	0000		PSCT	
	00081P	0000	20 31 0033	BRA	.\$A002
	00082P	0002	DC 04 B	.\$A003 LDD	KEYDRF
	00083P	0004	27 2D 0033	BEQ	.\$A004
	00084P	0006	DE 06 B	LDX	KEYDAT
	00085P	0008	8C 0061 A	CPX	#97
	00086P	000B	2D 0E 001B	BLT	.\$A005
	00087P	000D	DE 06 B	LDX	KEYDAT
	00088P	000F	8C 007A A	CPX	#122
	00089P	0012	2E 07 001B	BGT	.\$A005
	00090P	0014	DC 06 B	LDD	KEYDAT
	00091P	0016	83 0020 A	SUBD	#32
	00092P	0019	DD 06 B	STD	KEYDAT
	00093P	001B	4F	.\$A005 CLRA	
	00094P	001C	5F	CLRB	
	00095P	001D	DD 04 B	STD	KEYDRF
	00096P	001F	CE C000 A	LDX	#-16384
	00097P	0022	CC 0095 A	LDD	#149
	00098P	0025	E7 00 A	STAB	0,X
	00099P	0027	DC 06 B	LDD	KEYDAT
	00100P	0029	DD 02 B	STD	DSPDAT
	00101P	002B	DD 00 B	STD	OUTDAT
	00102P	002D		MLBSR	EXPDP
	00103P	0030		MLBSR	EXPDP
	00104		0033 P	.\$A004 EQU	*
	00105		0033 P	MAIN EQU	*
	00106P	0033	20 CD 0002	.\$A002 BRA	.\$A003
	00107P	0035	39	RTS	
	00108P	0036		PSCT	
	00109P	0036	4F	EXPIN CLRA	
	00110P	0037	5F	CLRB	
	00111P	0038	DD 0C B	STD	ENTRY
	00112P	003A	DD 0A B	STD	FUNC

ERR	SEQ	LOC	OBJECT	PROGRAM	EXPC
	00113P	003C	DD 08	B	STD TNCNT
	00114P	003E	DD 06	B	STD KEYDAT
	00115P	0040	DD 04	B	STD KEYDRF
	00116P	0042	DD 02	B	STD DSPDAT
	00117P	0044	DD 00	B	STD OUTDAT
	00118P	0046	CE A001	A	LDX #-24575
	00119P	0049	E7 00	A	STAB 0,X
	00120P	004B	CE A000	A	LDX #-24576
	00121P	004E	CC 00FF	A	LDD #255
	00122P	0051	E7 00	A	STAB 0,X
	00123P	0053	CE A001	A	LDX #-24575
	00124P	0056	CC 0004	A	LDD #4
	00125P	0059	E7 00	A	STAB 0,X
	00126P	005B	CE A000	A	LDX #-24576
	00127P	005E	CC 0002	A	LDD #2
	00128P	0061	E7 00	A	STAB 0,X
	00129P	0063	CE A003	A	LDX #-24573
	00130P	0066	4F		CLRA
	00131P	0067	5F		CLRB
	00132P	0068	E7 00	A	STAB 0,X
	00133P	006A	CE A002	A	LDX #-24574
	00134P	006D	CC 00FF	A	LDD #255
	00135P	0070	E7 00	A	STAB 0,X
	00136P	0072	CE A003	A	LDX #-24573
	00137P	0075	CC 0004	A	LDD #4
	00138P	0078	E7 00	A	STAB 0,X
	00139P	007A	CC 0030	A	LDD #48
	00140P	007D	DD 0A	B	STD FUNC
	00141P	007F	CC 0006	A	LDD #6
	00142P	0082	DD 0C	B	STD ENTRY
	00143P	0084			MLBSR EXPINT
	00144P	0087	CC 000E	A	LDD #14
	00145P	008A			MLBSR SETINS
	00146P	008D	CE C000	A	LDX #-16384
	00147P	0090	CC 0097	A	LDD #151
	00148P	0093	E7 00	A	STAB 0,X
	00149P	0095	CE C000	A	LDX #-16384
	00150P	0098	CC 0095	A	LDD #149
	00151P	009B	E7 00	A	STAB 0,X
	00152P	009D	CE 0014	A	LDX #20
	00153P	00A0	CC 007D	A	LDD #125
	00154P	00A3	E7 00	A	STAB 0,X
	00155P	00A5	39		RTS
	00156P	00A6			PSCT
	00157P	00A6	CE C000	A EXPIP	LDX #-16384
	00158P	00A9	E6 00	A	LDAB 0,X
	00159P	00AB			MSEX
	00160P	00B0	4F		CLRA
	00161P	00B1	C4 01	A	ANDB #1
	00162P	00B3	27 21 00D6		BEQ \$.A008
	00163P	00B5	CE C000	A	LDX #-16384
	00164P	00B8	CC 00D5	A	LDD #213
	00165P	00BB	E7 00	A	STAB 0,X
	00166P	00BD	CE C001	A	LDX #-16383
	00167P	00C0	E6 00	A	LDAB 0,X
	00168P	00C2			MSEX

ERR	SEQ	LOC	OBJECT	PROGRAM	EXPC
	00169P	00C7	DD 06	B	STD KEYDAT
	00170P	00C9	CC 00FF	A	LDD #255
	00171P	00CC	DD 04	B	STD KEYDRF
	00172P	00CE	CE C000	A	LDX #-16384
	00173P	00D1	CC 0095	A	LDD #149
	00174P	00D4	E7 00	A	STAB 0,X
	00175P	00D6	39	.\$A008	RTS
	00176P	00D7			PSCT
	00177P	00D7	4F	EXPINT	CLRA
	00178P	00D8	5F		CLRB
	00179P	00D9	20 0E 00E9		BRA ..1
	00180P	00DB		.\$A010	MLBSR EXPIT
	00181P	00DE	CC 0030	A	LDD #48
	00182P	00E1			MLBSR EXPINS
	00183P	00E4	DC 08	B	LDD TNCNT
	00184P	00E6	C3 0001	A	ADDD #1
	00185P	00E9	DD 08	B ..1	STD TNCNT
	00186P	00EB	DE 08	B .\$A011	LDX TNCNT
	00187P	00ED	8C 0003	A	CPX #3
	00188P	00F0	2D E9 00DB		BLT .\$A010
	00189P	00F2	CE A000	A	LDX #-24576
	00190P	00F5	CC 0002	A	LDD #2
	00191P	00F8	E7 00	A	STAB 0,X
	00192P	00FA	DC 0A	B	LDD FUNC
	00193P	00FC			MLBSR SETINS
	00194P	00FF	CC 0008	A	LDD #8
	00195P	0102			MLBSR SETINS
	00196P	0105	CC 0001	A	LDD #1
	00197P	0108			MLBSR SETINS
	00198P	010B	DC 0C	B	LDD ENTRY
	00199P	010D			MLBSR SETINS
	00200P	0110	39		RTS
	00201P	0111			PSCT
	00202P	0111		EXPDSP	MLBSR EXPBSY
	00203P	0114	CE A000	A	LDX #-24576
	00204P	0117	CC 0004	A	LDD #4
	00205P	011A	E7 00	A	STAB 0,X
	00206P	011C	CE A000	A	LDX #-24576
	00207P	011F	CC 0005	A	LDD #5
	00208P	0122	E7 00	A	STAB 0,X
	00209P	0124	CE A002	A	LDX #-24574
	00210P	0127	DC 02	B	LDD DSPDAT
	00211P	0129	E7 00	A	STAB 0,X
	00212P	012B	CE A000	A	LDX #-24576
	00213P	012E	CC 0004	A	LDD #4
	00214P	0131	E7 00	A	STAB 0,X
	00215P	0133	CE A000	A	LDX #-24576
	00216P	0136	CC 0002	A	LDD #2
	00217P	0139	E7 00	A	STAB 0,X
	00218P	013B	39		RTS
	00219P	013C			PSCT
	00220P	013C	20 07 0145		BRA .\$A014
	00221P	013E	CE C001	A .\$A015	LDX #-16383
	00222P	0141	DC 00	B	LDD OUTDAT
	00223P	0143	E7 00	A	STAB 0,X
	00224		0145	P	EXPOUT EQU *



ERR	SEQ	LOC	OBJECT	PROGRAM	EXPC
	00225P	0145	CE C000	A . \$A014	LDX #-16384
	00226P	0148	E6 00	A	LDAB 0,X
	00227P	014A			MSEX
	00228P	014F	4F		CLRA
	00229P	0150	C4 02	A	ANDB #2
	00230P	0152	26 EA 013E		BNE . \$A015
	00231P	0154	39		RTS
	00232P	0155			PSCT
	00233P	0155	3C	EXPBSY	PSHX
	00234P	0156	CC 0080	A	LDD #128
	00235P	0159	30		TSX
	00236P	015A	ED 00	A	STD 0,X
	00237P	015C	CE A003	A	LDX #-24573
	00238P	015F	4F		CLRA
	00239P	0160	5F		CLRB
	00240P	0161	E7 00	A	STAB 0,X
	00241P	0163	CE A002	A	LDX #-24574
	00242P	0166	E7 00	A	STAB 0,X
	00243P	0168	CE A003	A	LDX #-24573
	00244P	016B	CC 0004	A	LDD #4
	00245P	016E	E7 00	A	STAB 0,X
	00246P	0170	CE A000	A	LDX #-24576
	00247P	0173	CC 0002	A	LDD #2
	00248P	0176	E7 00	A	STAB 0,X
	00249P	0178	CE A000	A . \$A017	LDX #-24576
	00250P	017B	CC 0003	A	LDD #3
	00251P	017E	E7 00	A	STAB 0,X
	00252P	0180	CE A002	A	LDX #-24574
	00253P	0183	E6 00	A	LDAB 0,X
	00254P	0185			MSEX
	00255P	018A	30		TSX
	00256P	018B	ED 00	A	STD 0,X
	00257P	018D	CE A000	A	LDX #-24576
	00258P	0190	CC 0002	A	LDD #2
	00259P	0193	E7 00	A	STAB 0,X
	00260P	0195	30		TSX
	00261P	0196	EC 00	A	LDD 0,X
	00262P	0198	4F		CLRA
	00263P	0199	C4 80	A	ANDB #128
	00264P	019B	ED 00	A	STD 0,X
	00265P	019D	EE 00	A	LDX 0,X
	00266P	019F	8C 0080	A	CPX #128
	00267P	01A2	27 D4 0178		BEQ . \$A017
	00268P	01A4	CE A003	A	LDX #-24573
	00269P	01A7	4F		CLRA
	00270P	01A8	5F		CLRB
	00271P	01A9	E7 00	A	STAB 0,X
	00272P	01AB	CE A002	A	LDX #-24574
	00273P	01AE	CC 00FF	A	LDD #255
	00274P	01B1	E7 00	A	STAB 0,X
	00275P	01B3	CE A003	A	LDX #-24573
	00276P	01B6	CC 0004	A	LDD #4
	00277P	01B9	E7 00	A	STAB 0,X
	00278P	01BB	38		PULX
	00279P	01BC	39		RTS
	00280P	01BD			PSCT

ERR	SEQ	LOC	OBJECT	PROGRAM	EXPC
	00281P	01BD	37	EXPINS	PSHB
	00282P	01BE	36		PSHA
	00283P	01BF	CE A000	A	LDX #-24576
	00284P	01C2	4F		CLRA
	00285P	01C3	5F		CLRB
	00286P	01C4	E7 00	A	STAB 0,X
	00287P	01C6	CE A000	A	LDX #-24576
	00288P	01C9	CC 0001	A	LDD #1
	00289P	01CC	E7 00	A	STAB 0,X
	00290P	01CE	CE A002	A	LDX #-24574
	00291P	01D1	3C		PSHX
	00292P	01D2	30		TSX
	00293P	01D3	EC 02	A	LDD 2,X
	00294P	01D5	38		PULX
	00295P	01D6	E7 00	A	STAB 0,X
	00296P	01D8	CE A000	A	LDX #-24576
	00297P	01DB	4F		CLRA
	00298P	01DC	5F		CLRB
	00299P	01DD	E7 00	A	STAB 0,X
	00300P	01DF	38		PULX
	00301P	01E0	39		RTS
	00302P	01E1			PSCT
	00303P	01E1	37	SETINS	PSHB
	00304P	01E2	36		PSHA
	00305P	01E3			MLBSR EXPBSY
	00306P	01E6	30		TSX
	00307P	01E7	EC 00	A	LDD 0,X
	00308P	01E9			MLBSR EXPINS
	00309P	01EC	38		PULX
	00310P	01ED	39		RTS
	00311				XDEF EXPDSP
	00312				XDEF SETINS
	00313				XDEF EXPINS
	00314				XDEF EXPINT
	00315				XDEF EXPBSY
	00316				XDEF EXPIN
	00317				XDEF EXPOUT
	00318				XDEF MAIN
	00319				XDEF EXPIP
	00320				XREF EXPIT
	00321				END

\*\*\*\* TOTAL ERRORS 00000--00000

## 4.4.4 Linkage Listing

```
*** HMCS6800 CROSS LINKAGE EDITOR   VER 1.2 ***
LOAD=B:EXPMN.OBJ,B:EXPC.OBJ,C31RUN.OBJ
STRP=$F000
STRB=$60
STRD=$40
OPT=MAP,SYM
EXEC
```

```
*** HMCS6800 CROSS LINKAGE EDITOR   VER 1.2 ***
*** UNDEFINED SYMBOLS ***
      NAME      SECTION  MODULE NAME
      .ERROR    (        )
UNDEFINED SYMBOL = 1 (Note)
```

Note: There is an UNDEFINED SYMBOL = 1 (library function, ERROR) in the link information but it does not influence the execution of this program. The library function or run-time routine call the ERROR service routine when 0 is used as a divisor in division or modulo operation. Strictly speaking, the user should create an ERROR function. However it is never used in this program, so it is just displayed as an UNDEFINED SYMBOL.  
(When the library function and run-time routine are not linked, the UNDEFINED SYMBOL is not displayed.)

```
*** HMCS6800 CROSS LINKAGE EDITOR   VER 1.2 ***
*** MAP LIST ***
** SECTION LOAD MAP
      SECTION  SIZE  START  END  COMMON-SIZE
      A        0016  FFEA  FFFF
      B        000E  0060  006D  0000
      C        0000
      D        0004  0040  0043  0000
      P        07DA  F000  F7D9  0000
** MODULE LOAD MAP
      NAME      BSCT  DSCT  PSCT
      EXPC      0060          F000
              0040  F01A
              F208
** COMMON LOAD MAP
      NAME      SECTION  SIZE  START
COMMON =      0
```

NAME	SECTION	START	MODULE	NAME
.\$DADD	P	F7D9	(	)
.\$DCMP	P	F7D9	(	)
.\$DDEC	P	F7D9	(	)
.\$DDIV	P	F7D9	(	)
.\$DINC	P	F7D9	(	)
.\$DMOV	P	F7D9	(	)
.\$DMUL	P	F7D9	(	)
.\$DNEG	P	F7D9	(	)
.\$DSTK	P	F7D9	(	)
.\$DSUB	P	F7D9	(	)
.\$DTOF	P	F7D9	(	)
.\$DTOI	P	F7D9	(	)
.\$DTOL	P	F7D9	(	)
.\$DTST	P	F7D9	(	)
.\$FDEC	P	F7D9	(	)
.\$FINC	P	F7D9	(	)
.\$FMOV	P	F7D9	(	)
.\$FREG	D	0040	(	)
.\$FTOD	P	F7D9	(	)
.\$FTST	P	F7D9	(	)
.\$IASL	P	F27D	(	)
.\$IASR	P	F292	(	)
.\$IDIV	P	F23F	(	)
.\$IMOD	P	F2BC	(	)
.\$IMUL	P	F208	(	)
.\$ITOD	P	F7D9	(	)
.\$ITOL	P	F51D	(	)
.\$LADD	P	F32F	(	)
.\$LAND	P	F432	(	)
.\$LBIT	P	F600	(	)
.\$LCMP	P	F4BF	(	)
.\$LCPL	P	F501	(	)
.\$LDEC	P	F54B	(	)
.\$LDIV	P	F3E3	(	)
.\$LINC	P	F53B	(	)
.\$LMOD	P	F40A	(	)
.\$LMOV	P	F30B	(	)
.\$LMUL	P	F361	(	)
.\$LNEG	P	F4EC	(	)
.\$LOR	P	F44D	(	)
.\$LSHL	P	F483	(	)
.\$LSHR	P	F4A1	(	)
.\$LSTK	P	F55B	(	)
.\$LSUB	P	F348	(	)
.\$LTOD	P	F7D9	(	)
.\$LTST	P	F576	(	)
.\$LXOR	P	F468	(	)
.\$SBIT	P	F723	(	)
.\$SW1	P	F76B	(	)
.\$SW2	P	F79A	(	)
.\$UDIV	P	F25B	(	)
.\$ULSR	P	F2A7	(	)
.\$UMOD	P	F2EA	(	)

\*\*\* HMCS6800 CROSS LINKAGE EDITOR VER 1.2 \*\*\*

NAME	SECTION	START	MODULE NAME
.\$UTOD	P	F7D9	( )
.\$UTOL	P	F52E	( )
EXPBSY	P	F16F	( EXPC )
EXPDSP	P	F12B	( EXPC )
EXPIN	P	F050	( EXPC )
EXPINS	P	F1D7	( EXPC )
EXPINT	P	F0F1	( EXPC )
EXPIP	P	F0C0	( EXPC )
EXPIT	P	F00A	( )
EXPOUT	P	F15F	( EXPC )
MAIN	P	F04D	( EXPC )
SETINS	P	F1FB	( EXPC )

DEFINED SYMBOL = 65

# APPENDIX A. C Program and Assembly Program Comparison

This appendix compares application programs previously introduced assembly language system application examples to the C language examples in this application note.

(Assembly language programs are described in the 6301 APPLICATION NOTES (Hardware)).

In general, the size of the C language program is greater than that of the assembly language program.

These examples are hardware control programs that are difficult to write in C and show how to use the 6301 C language compiler. The run-time routines are not included in the C language program size descriptions.

## A.1 Darlington Transistor Drive (LED Dynamic Display)

Table A-1 compare Darlington Transistor Drive Routines written in C and assembler.

Table A-1. Program Comparison

Item	Memory Size (Bytes)	No. of Cycles (Machine cycle)
C Program	131	200
Assembly program	82	120
C program to assembly program ratio	1.6	1.67

## A.2 8 x 4 Key Metrix

Table A-2 compares 8 x 4 key Matrix Routines written in C and assembler.

**Table A-2. Program Comparison**

<b>Item</b>	<b>Memory Size (Bytes)</b>	<b>No. of Cycles (Machine cycle)</b>
C Program	336	1240
Assembly program	181	373
C program to assembly program ratio	1.86	3.32

## A.3 External Expansion

Table A-3 compares External Expansion Routines written in C and assembler.

**Table A-3. Program Comparison**

<b>Item</b>	<b>Memory Size (Bytes)</b>	<b>No. of Cycles (Machine cycle)</b>
C Program	518	1347
Assembly program	318	572
C program to assembly program ratio	1.63	2.35

# HD6301/HD6303 SERIES HANDBOOK

## Section Ten

# APPENDIX

1. HD6301V1/HD6303R Q and A
2. HD6301X0/HD6303X Oscillator Circuit
3. Wide Temperature Range Specifications  
-40°C to 85°C (J Version)





## Section 10—Appendix

### 1. HD6301V1/HD6303R Q and A

#### Table of Contents

	Page
1. HD6301V1/HD6303R Q & A .....	1239
(a) Parallel Port	
(1) Process to Use a Port as an Output .....	1241
(b) Serial Port	
(1) Relation between Writing into the FRC and SCI Operation .....	1242
(2) Writing into the FRC during Serial Receive/Transmit .....	1243
(3) RDRF State When SCI Receiving .....	1244
(4) Serial I/O Operation .....	1245
(5) Serial I/O Register Read .....	1246
(6) Detection of the HD6301V1 Serial Start Bit .....	1247
(c) Timer/Counter	
(1) Free Running Counter Read .....	1249
(2) Preset Method of the Free Running Counter .....	1250
(d) BUS Interface	
(1) Output of Address Strobe (AS) in the Multiplexed Mode .....	1252
(e) Interrupt	
(1) $\overline{\text{TRQ1}}$ Acceptance .....	1253
(2) Timer Interrupt and External Interrupt .....	1254
(3) $\overline{\text{TRQ1}}$ Interrupt and Other Interrupts .....	1255
(4) CLI Instruction and Interrupt Operation .....	1257
(f) Oscillator	
(1) Relation between the External Clock (EXTAL Clock) and Enable Clock (E Clock)	1258
(g) Reset	
(1) Constants of the Reset Circuit .....	1259
(2) Schmitt Trigger Circuit of $\overline{\text{RES}}$ .....	1260
(3) I/O Port State on Resetting .....	1261
(4) SCI (Pin 39) State on Resetting .....	1262
(5) Port Output after Resetting .....	1263
(h) Low Power Consumption	
(1) Schmitt Trigger Circuit of $\overline{\text{STBY}}$ .....	1264
(2) I/O Port State During Standby .....	1265

(3) Return from Standby Mode .....	1266
(4) Going into the Standby Mode .....	1267
(5) Timing for the Standby Mode .....	1268
(i) EPROM-on-Package	
(1) Usage of EPROM Socket Pins for the HD63P01M (No. 1) .....	1269
(2) Usage of EPROM Socket Pins for the HD63P01M (No. 2) .....	1270
(3) Usage of EPROM Socket Pins for the HD63P01M (No. 3) .....	1271
(j) Software	
(1) Usage of Bit Manipulation Instructions .....	1272
(2) Usage of Bit Manipulation Instructions to the Port .....	1274
(k) Others	
(1) RAM Access Disable during Program Execution .....	1275
2. HD6301X0/HD6303X Oscillator Circuit .....	1276
3. Wide Temperature Range Specifications -40°C to +85°C (J Version) .....	1283
HD6301V1, HD63A01V1, HD63B01V1 .....	1284
HD6301X0, HD63A01X0, HD63B01X0 .....	1289
HD6301Y0, HD63A01Y0, HD63B01Y0, HD63C01Y0 .....	1294
HD6303R, HD63A03R, HD63B03R .....	1300
HD6303X, HD63A03X, HD63B03X .....	1305
HD6303Y, HD63A03Y, HD63B03Y, HD63C03Y .....	1310

Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M Software Evaluation kit, Emulator SD SBC	
Theme	Process to Use a Port as an Outputs		Date Nov. 24, 1983	
Question	<p>1, When using an I/O port as an output, is the data stored to the Data Register or is the Data Direction Register (DDR) set at first?</p>		Classification	
			* Parallel Port	
			Serial Port	
			Timer/Counter	
			BUS Interface	
			Interrupt	
			A/D Converter	
			Oscillator	
			Reset	
			Low Power Consm.	
			EPROM-on-package	
			Software	
			Evaluation Kit	
	Emulator			
	SD			
	Data Buffer			
	Others			
Answer	<p>1, Store the data to the Data Register at first and then set the DDR (DDR=1); if not, unknown data is output from the port.</p>		Applicable Manual	
			Title	
			Semiconductor Data Book - 8-Bit Single Chip Microcomputer -	
			Other Data	
			Title	
	Reference Q & A Sheet			
	No.			
Supplement	<p>The DDR defines an I/O port as an input or output.          DDR=1 : output          DDR=0 : input</p>			

Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M Evaluation kit, Emulator	Software SD SBC
Theme	Relation between Writing into the FRC and SCI Operation		Date	Nov. 24, 1983
Question	<p>1, How are writing into the timer Free Running Counter(FRC) and the Serial Communication Interface(SCI) related?</p>			Classification
				Parallel Port
				* Serial Port
				Timer/Counter
				BUS Interface
				Interrupt
				A/D Converter
				Oscillator
				Reset
				Low Power Consm.
EPROM-on-package				
Software				
Evaluation Kit				
Emulator				
SD				
Data Buffer				
Others				
Answer	<p>1, The source of the clock input to the SCI Shift Registers is the timer FRC. Therefore, if new data is written into the FRC, SCI operations are disturbed. See the following diagram.</p> <p style="text-align: center;">* A write into the FRC is prohibited during SCI operations.</p>			Applicable Manual
				Title
				Other Data
				Title
Reference Q & A Sheet	No.	QA631-002A QA631-008A		
Supplement				

Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M Software Evaluation kit, Emulator SD SBC
Theme	Writing into the FRC during Serial Receive/Transmit	Date	Nov. 24, 1983
Question	1, Is it prohibited to write data into the Free Running Counter(FRC) during serial receive/transmit?	Classification	Parallel Port * Serial Port Timer/Counter BUS Interface Interrupt A/D Converter Oscillator Reset Low Power Consm. EPROM-on-package Software Evaluation Kit Emulator SD Data Buffer Others
Answer	1, Yes. If data is written into the FRC during serial receive/transmit, the FRC stops counting up and the baud rate changes. In condition other than serial receive/transmit, it's possible to write. <p>The counter stops and the baud rate changes.</p>	Applicable Manual	Title  Other Data Title  Reference Q & A Sheet No.  QA631-001A QA631-008A
Supplement			



Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M Software Evaluation kit, Emulator SD SBC
Theme	Serial I/O Operation		Date Nov. 24, 1983
Question	<p>1, The serial I/O does not operate satisfactorily. Initialization does not seem to be wrong, but the data is not transmitted. What is wrong?</p> <p>Initialize by User Program</p> <ol style="list-style-type: none"> <li>1 Set the Rate/Mode Control Register (RMCR) to the desired operation.</li> <li>2 Set the Transmit/Receive Control Status Register (TRCSR) to the desired operation.</li> </ol>		<p>Classification</p> <p>Parallel Port</p> <p>* Serial Port</p> <p>Timer/Counter</p> <p>BUS Interface</p> <p>Interrupt</p> <p>A/D Converter</p> <p>Oscillator</p> <p>Reset</p> <p>Low Power Consm.</p> <p>EPROM-on-package</p> <p>Software</p> <p>Evaluation Kit</p> <p>Emulator</p> <p>SD</p> <p>Data Buffer</p> <p>Others</p>
Answer	<p>1, Just after the initialization of serial I/O, the data transmit is not operative during 10 cycles of Baud Rate after setting the TE. The reason is as follows.</p> <p>Setting the transmit enable bit (TE bit) causes ten consecutive "1" of preamble and makes the transmitter section operative. In other words, the transmitter section gets ready after one frame (10 bits) transmitting time according to the Baud rate.</p> <p>(ex.) When the Baud rate is set to 9600 Baud (104.2s at 1 bit),</p> <p>Set the Baud rate    Set TE    Transmit OK</p> <p style="text-align: center;">→ 104.2µs × 10 = 1.042ms ←</p> <p style="text-align: center;">⊠ : Transmit Inoperative Period Preamble Causing Period</p> <p>1.042ms after setting the TE, the transmitter section is operative.</p>		<p>Applicable Manual</p> <p>Title</p> <hr/> <p>Other Data</p> <p>Title</p> <hr/> <p>Reference Q &amp; A Sheet</p> <p>No.</p>
Supplement			



Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M Software Evaluation kit, Emulator SD SBC
Theme	Serial I/O Register Read		Date Nov. 24, 1983
Question	<p>1, When transmitting the data, is reading the Transmit/Receive Control Register(TRCSR) required? When the transfer interval is long enough compared with the Baud rate, Transmit Data Register Empty (TDRE) will be set. In that case, are there any problems when transmitting data without checking the TDRE flag in the TRCSR?</p>		Classification
			Parallel Port
			* Serial Port
			Timer/Counter
			BUS Interface
			Interrupt
			A/D Converter
			Oscillator
			Reset
			Low Power Consm.
			EPROM-on-package
			Software
			Evaluation Kit
	Emulator		
	SD		
	Data Buffer		
	Others		
Answer	<p>1, The TDRE flag shows if the TDRE register is empty or not. When writing a data to the TDR with TDRE=1, it's not necessary to check the TDRE. But reading the TDRE flag tells us the contents of TDR. For example, when new data is written to the TDR with TDRE "0"(TDR already has a data), the old data will be erased. When the transfer interval is long enough compared with the Baud rate, there's no problem. However, check TRCSR if possible.</p>		Applicable Manual
			Title
			Other Data
			Title
			Reference Q & A Sheet
	No.		
Supplement			

Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M Evaluation kit, Emulator	Software SD SBC	
Theme	Detection of the HD6301V1 Serial Start Bit (No.1)		Date	Nov. 24.1983	
Question	<p>1.</p> <p>(1) What is the relation between the HD6301V1 serial sampling clock frequency and the baud rate ?</p> <p>(2) What does "Sampling error" mean ?</p>		Classification		
					Parallel Port
			*		Serial Port
					Timer/Counter
					BUS Interface
					Interrupt
					A/D Converter
					Oscillator
					Reset
					Low Power Consm.
					EPR0M-on-package
					Software
					Evaluation Kit
					Emulator
			SD		
			Data Buffer		
			Others		
Answer	<p>1.</p> <p>(1) The serial sampling clock frequency is eight times the baud rate.</p> <p>(2) "Sampling error" means receive margin at the serial operation time. Refer to the next page for details.</p>		Applicable Manual		
				Title	
			Other Data		
			Title		
			Reference Q & A Sheet		
			No.		
Supplement					

Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M	Software	SD SBC
Theme	Detection of the HD6301V1 Serial Start Bit (No. 2)				

Answer

Receive margin:

The HD6301V1 detects the start bit and samples the data bit using the falling edge of the sampling clock.  
The general equation is shown as follows.

1. General equation

$$M = [ ( 0.5 - 1/N ) - ( D - 0.5 ) / N - ( L - 0.5 ) F ] \times 100 (\%)$$

M: Receive margin

N: Ratio of baud rate to sampling clock ( 0 to 0.5 )

D: Duty of the longer sampling clock of "H", and "L"

L: Frame length ( 7 to 12 bits )

F: Absolute value of deviation of sampling clock frequency

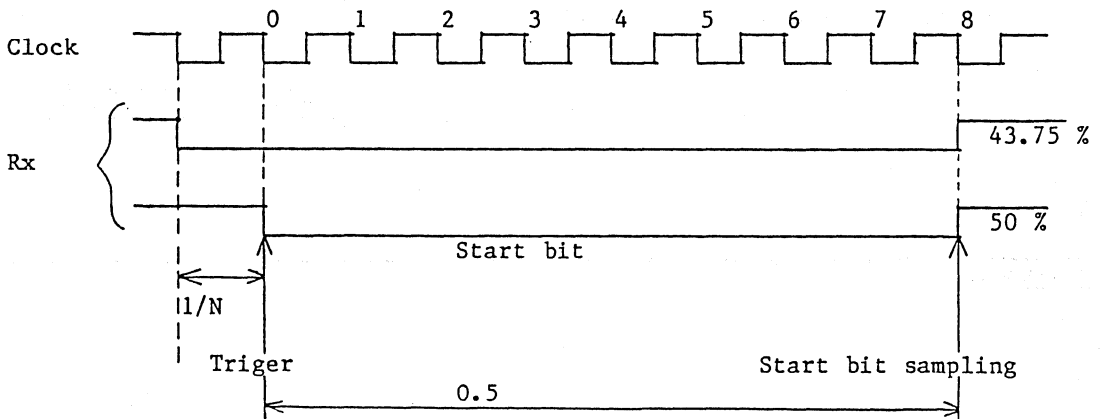
2. Abbreviated equation

$$M = ( 0.5 - 1/N ) \times 100 (\%)$$

Conditions: D = 0.5, F = 0

N	8	16	32	64	Note
M (%)	37.5	43.75 (Fig.1)	46.875	48.4375	In the HD6301V1, N = 8.

Figure 1



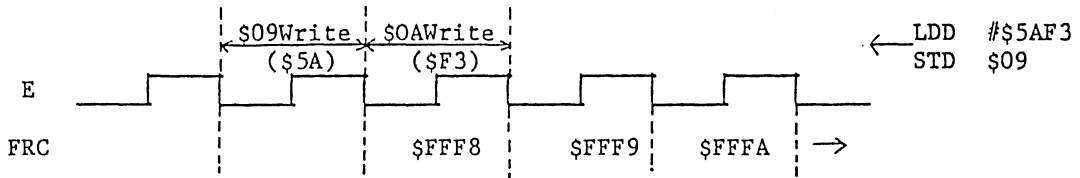
Type	HD6301V1 HD6303R	Device	4S *8S Evaluation kit, Emulator	8M 16M SD SBC	Software
Theme	Free Running Counter Read			Date	Nov. 24, 1983
Question	<p>1. When the FRC of the HD6301V1/HD6303R is read with the double byte load instructions(2 cycle execution for FRC reading), is it read correctly? Double byte load instructions require two cycles to be executed and the cycle to read the low byte of FRC becomes the next cycle of the high byte. Is it OK ?</p> <p>(EX)</p> <p>AccD</p> <p>(Whe reading \$F7FF from the counter)</p>				Classification Parallel Port Serial Port * Timer/Counter BUS Interface Interrupt A/D Converter Oscillator Reset Low Power Consm. EPROM-on-package Software Evaluation Kit Emulator SD Data Buffer Others
Answer	<p>1. The FRC of the HD6301V1/HD6303R contains a parallel temporary register. When the high byte of the FRC is read, the low byte is set in the temporary register. The Low byte data in the temporary register is set to the AccD at the next cycle. Therefore, it is possible to read the FRC correctly.</p> <p>AccD</p> <p>(When reading \$F7FF from the counter)</p>				Applicable Manual Title  Other Data Title  Reference Q & A Sheet No.
Supplement	FRC: Free Running Counter The base counter of the timer which counts up the E clock.				

Type	HD6801V1 HD6301V1	Device	4S *8S 8M 16M Software Evaluation kit, Emulator SD SBC						
Theme	Preset Method of the Free Running Counter (No.1)		Date Nov.24.1983						
Question	<p>1.</p> <p>What is the difference between the HD6801V and HD6301V1 in writing data into the free running counter ?</p>		Classification Parallel Port Serial Port * Timer/Counter BUS Interface Interrupt A/D Converter Oscillator Reset Low Power Consm. EPROM-on-package Software Evaluation Kit Emulator SD Data Buffer Others						
Answer	<p>1. The FRC preset method of the HD6801V is different from the HD6301V1.</p> <table border="1" data-bbox="91 892 810 1121"> <thead> <tr> <th>Type</th> <th>Preset Method</th> </tr> </thead> <tbody> <tr> <td>HD6801V</td> <td>The FRC is always preset to "\$FFF8".</td> </tr> <tr> <td>HD6301V1</td> <td>1. Writing to the high byte presets the FRC to \$FFF8. 2. The FRC is set to desirable data by a double byte store instruction.</td> </tr> </tbody> </table>		Type	Preset Method	HD6801V	The FRC is always preset to "\$FFF8".	HD6301V1	1. Writing to the high byte presets the FRC to \$FFF8. 2. The FRC is set to desirable data by a double byte store instruction.	Applicable Manual Title Semiconductor Data Book - 8-Bit Single Chip Microcomputer - Other Data Title Reference Q & A Sheet No. QA631-001A QA631-002A
Type	Preset Method								
HD6801V	The FRC is always preset to "\$FFF8".								
HD6301V1	1. Writing to the high byte presets the FRC to \$FFF8. 2. The FRC is set to desirable data by a double byte store instruction.								
Supplement	<p>See the next page for the example of this method.</p>								

Type	HD6801V1 HD6301V1	Device	4S *8S 8M 16M	Software	SD SBC
Theme	Preset Method of the Free Running Counter (No.2)				

Answer

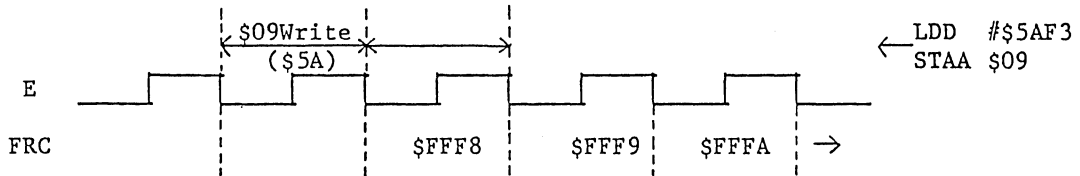
(1) The HD6801V Preset Method



The FRC is always preset to \$FFF8.

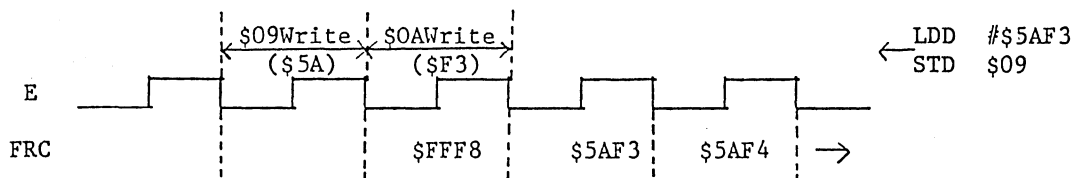
(2) The HD6301V1 Preset Method

1. \$FFF8



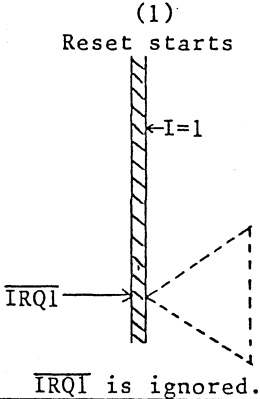
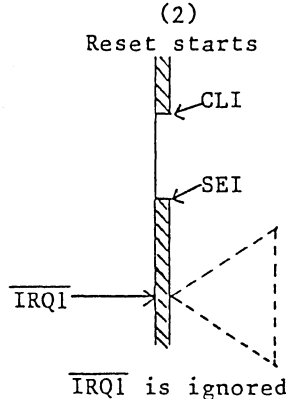
Writing to the high byte presets the FRC to \$FFF8.

2. Optional valve (In this case \$5AF3)



The FRC is set to desirable data (\$5AF3) by a double byte store instruction.

Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M Evaluation kit, Emulator	Software	SD SBC
Theme	Output of Address Strobes(AS) in the Multi-plexed Mode		Date	Nov. 24, 1983	
Question	<p>1, Is AS always output when using the HD6301V1 in the expanded multiplexed mode (mode 2, 4, 6)?</p>			Classification	
				Parallel Port	
				Serial Port	
				Timer/Counter	
				* BUS Interface	
				Interrupt	
				A/D Converter	
				Oscillator	
				Reset	
				Low Power Consm.	
				EPROM-on-package	
				Software	
				Evaluation Kit	
				Emulator	
	SD				
	Data Buffer				
	Others				
Answer	<p>2, Yes. AS is always output in the expanded multiplexed mode, even when the MPU accesses the internal RAM, ROM, etc.</p>			Applicable Manual	
				Title	
				Other Data	
				Title	
	Reference Q & A Sheet				
	No.				
Supplement	<p>In the expanded multiplexed mode, the data buses and lower address buses are multiplexed and output from port 3. AS is the signal needed to demultiplex the data buses and address busses. Mode 2, 4 and 6 of the HD6301V1 are the expanded modes.</p>				

Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M Software Evaluation kit, Emulator SD SBC
Theme	IRQI Acceptance		Date Nov. 24, 1983
Question	<p>1, (1) Is IRQI ignored when the Condition Code Register I mask is set?</p> <p>(2) After the I mask is reset, will the interrupt sequence start by the interrupt request flag having been latched?</p>		<p>Classification</p> <p>Parallel Port</p> <p>Serial Port</p> <p>Timer/Counter</p> <p>BUS Interface</p> <p>* Interrupt</p> <p>A/D Converter</p> <p>Oscillator</p> <p>Reset</p> <p>Low Power Consm.</p> <p>EPROM-on-package</p> <p>Software</p> <p>Evaluation Kit</p> <p>Emulator</p> <p>SD</p> <p>Data Buffer</p> <p>Others</p>
Answer	<p>1, (1) If the Condition Code Register I mask is set, IRQI is completely ignored.</p> <p>(2) With the I mask set, the interrupt request flag will not be latched.</p> <p>(1)</p>  <p>Reset starts</p> <p>← I=1</p> <p>IRQI →</p> <p>IRQI is ignored.</p> <p>(2)</p>  <p>Reset starts</p> <p>← CLI</p> <p>← SEI</p> <p>IRQI →</p> <p>IRQI is ignored.</p>		<p>Applicable Manual</p> <p>Title</p> <p>Other Data</p> <p>Title</p> <p>Reference Q &amp; A Sheet</p> <p>No.</p>
Supplement	<p>CLI : Clears the Condition Code Register I mask</p> <p>SEI : Sets the Condition Code Register I mask</p> <p>* <math>\overline{\text{NMI}}</math> is acceptable regardless of the I mask.</p>		

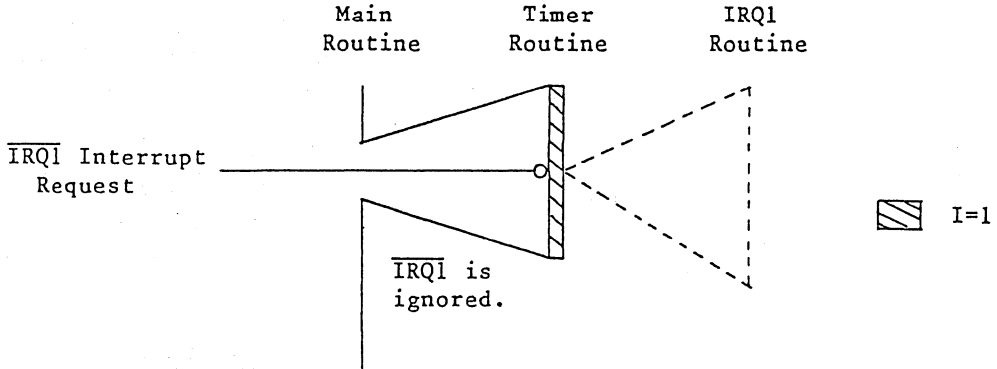


Type	HD6301V1 HD6303R	Device	4S *8S Evaluation kit, Emulator	8M 16M SD SBC	Software
Theme	Timer Interrupt and External Interrupt			Date	Nov. 24, 1983
Question	<p>1, In the routine below, when is the next timer interrupt accepted?</p> <p>Main Timer(OCI) Routine      External Interrupt (IRQ) Routine</p> <p>(Execution time =1.5ms)      (Execution time=3ms)</p> <p>Read the TCSR Store 2.6ms as timer period to the OCR</p> <p>EOCI=0 CLI</p> <p>Next Timer Interrupt Request</p> <p>Execution time is longer than timer interrupt period. * I=1</p> <p>EOCI=1 RTI</p> <p>RTI</p>				Classification
Answer	<p>1, The next timer interrupt is accepted in the main routine just after RTI instruction execution.</p> <p>Main Timer(OCI) Routine      External Interrupt (IRQ) Routine</p> <p>Next Timer Interrupt Request</p> <p>RTI</p> <p>Next Timer (OCI) Routine</p>				* Interrupt A/D Converter Oscillator Reset Low Power Consom. EPROM-on-package Software Evaluation Kit Emulator SD Data Buffer Others
Supplement					Applicable Manual Title Other Data Title Reference Q & A Sheet No. QA631-012A

Type	HD6301V1 HD6303R	Device	4S *8S Evaluation kit, Emulator	8M 16M SD SBC	Software
Theme	IRQ1 Interrupt and Other Interrupts (NO. 1)			Date	Nov. 24, 1983
Question				Classification	
<p>1, <math>\overline{\text{IRQ1}}</math> pin (pin 5) is held at low for <math>10\mu\text{s}</math> but an interrupt does not occur. What should be done to generate an interrupt sequence?</p>				Parallel Port	
				Serial Port	
				Timer/Counter	
				BUS Interface	
				* Interrupt	
				A/D Converter	
				Oscillator	
				Reset	
				Low Power Consm.	
				EPROM-on-package	
				Software	
				Evaluation Kit	
				Emulator	
				SD	
Data Buffer					
Others					
Answer				Applicable Manual	
<p>1, (1) <math>\overline{\text{IRQ1}}</math> is a level sensitive interrupt pin which needs a minimum of 2 machine cycles (<math>2\mu\text{s}</math> at 1MHz) to accept an interrupt. However, if another interrupt has been already generated, no interrupt request is accepted with <math>\overline{\text{IRQ1}}</math> at low for <math>10\mu\text{s}</math>. In such a case, <math>\overline{\text{IRQ1}}</math> should be held at low until the request is accepted.</p>				Title	
				Other Data	
<p>(2) In this case, as a timer interrupt is executed the interrupt mask is automatically set. So <math>\overline{\text{IRQ1}}</math> is ignored.</p> <p>See the next page for the illustration of <math>\overline{\text{IRQ1}}</math> and other interrupts and a countermeasure.</p>				Title	
				Reference Q & A Sheet	
Supplement				No.	
				QA631-011A	

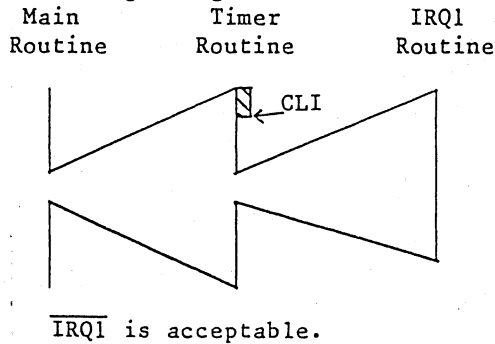
Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M	Software	SD SBC
Theme	IRQI Interrupt and Other Interrupts (No. 2)				

IRQI and Other Interrupts



Countermeasure

Clear the I mask at the beginning of the timer interrupt routine.



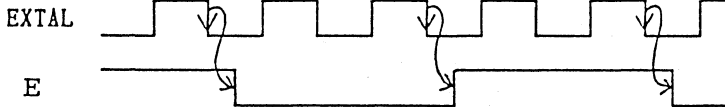
\* CLI : Clears the interrupt mask (I=0).

With this method, note the following ;

- (1) IRQI may be ignored when the request occurs during timer interrupt vectoring.
- (2) Interrupts form NMI or SWI are excluded.

Type	HD6301V1 HD6303R	Device	4S *8S Evaluation kit, Emulator	8M SD	16M SBC	Software				
Theme	CLI Instruction and Interrupt Operation			Date	Nov. 24, 1983					
Question	<p>1, In the HD6301V, a timer interrupt is not accepted in the following program. Is there any problem?</p> <pre style="border: 1px dashed black; padding: 5px;"> Main Routine L01 CLI       NOP       SEI       :       :       BRA L01                     </pre>			Classification Parallel Port Serial Port Timer/Counter BUS Interface * Interrupt A/D Converter Oscillator Reset Low Power Consm. EPROM-on-package Software Evaluation Kit Emulator SD Data Buffer Others						
Answer	<p>1, To accept an interrupt, two machine cycles are necessary between CLI and SEI. That is, in this program, two NOP instructions are necessary. The same thing can be said when using TAP for CLI and SEI.</p> <table style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Using CLI</th> <th style="width: 50%; text-align: center;">Using TAP</th> </tr> </thead> <tbody> <tr> <td style="border: 1px dashed black; padding: 5px;"> <pre> L01 CLI       NOP       NOP       SEI       :       :       BRA L01                     </pre> </td> <td style="border: 1px dashed black; padding: 5px;"> <pre> TAP (Clears the I mask) NOP NOP TAP (Sets the I mask) : :                     </pre> </td> </tr> </tbody> </table>			Using CLI	Using TAP	<pre> L01 CLI       NOP       NOP       SEI       :       :       BRA L01                     </pre>	<pre> TAP (Clears the I mask) NOP NOP TAP (Sets the I mask) : :                     </pre>	Applicable Manual Title HD6301X Data Sheet Semiconductor Data Book - 8-Bit Single Chip Microcomputer - Other Data Title Reference Q & A Sheet No.		
Using CLI	Using TAP									
<pre> L01 CLI       NOP       NOP       SEI       :       :       BRA L01                     </pre>	<pre> TAP (Clears the I mask) NOP NOP TAP (Sets the I mask) : :                     </pre>									
<p>* This is mentioned in the HD6301X data sheet but not in the HD6301V.</p>										
Supplement										

Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M Evaluation kit, Emulator SD SBC
Theme	Relation between the External Clock (EXTAL Clock) and Enable Clock (E Clock)		Date Nov. 24, 1983
Question	1, With which edges of the EXTAL clock does the E clock change synchronously, rising edge (↑) or falling edge (↓)?		Classification
			Parallel Port
			Serial Port
			Timer/Counter
			BUS Interface
			Interrupt
			A/D Converter
			* Oscillator
			Reset
			Low Power Consm.
			EPROM-on-package
			Software
			Evaluation Kit
			Emulator
			SD
			Data Buffer
			Others
Answer	1, It changes synchronously with the falling edge (↓) of the EXTAL clock.		Applicable Manual
			Title
			HD6301V User's Manual
			Other Data
			Title
			Reference Q & A Sheet
			No.
Supplement			





Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M Software Evaluation kit, Emulator SD SBC	
Theme	Schmitt Trigger Circuit of $\overline{\text{RES}}$		Date	Nov. 24, 1983
Question	1, Is Schmitt trigger circuit provided with the HD6303R/HD6301V1 RES?		Classification	
			Parallel Port	
			Serial Port	
			Timer/Counter	
			BUS Interface	
			Interrupt	
			A/D Converter	
			Oscillator	
			* Reset	
			Low Power Consm.	
			EPROM-on-package	
			Software	
			Evaluation Kit	
Emulator				
SD				
Data Buffer				
Others				
Answer	1, Yes. (Mentioned in the HD6301V User's Manual)		Applicable Manual	
<p>To the system power supply</p> <p>To peripherals</p> <p><math>R_1, R_2, R_r, C_r \gg 10\text{ms}</math></p> <p>HD6301V</p>			Title	
			HD6301V User's Manual	
			Other Data	
			Title	
			Reference Q & A Sheet	
			No.	
			QA631-015A	
			QA631-020A	
Supplement	<p>Effects of the Schmitt trigger circuit:</p> <p>Even on the slow rising edge of input pulse, stable and clear waveform can be output.</p>			

Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M	Software	SD SBC													
Theme	I/O Port State on Resetting			Date	Nov. 24, 1983													
Question	1, What is the state of each port on resetting (RES='0')?			Classification														
				Parallel Port														
				Serial Port														
				Timer/Counter														
				BUS Interface														
				Interrupt														
				A/D Converter														
				Oscillator														
				* Reset														
				Low Power Consm.														
				EPROM-on-package														
				Software														
				Evaluation Kit														
				Emulator														
	SD																	
	Data Buffer																	
	Others																	
Answer	1, It is as follows:  <table border="1" data-bbox="113 803 754 1060"> <tr> <td>Port 1</td> <td>High impedance state</td> </tr> <tr> <td>Port 2</td> <td>↑</td> </tr> <tr> <td>* Modes 1, 5</td> <td>↑</td> </tr> <tr> <td rowspan="2">Port 3</td> <td>E: ↑</td> </tr> <tr> <td>E: "1" is output.</td> </tr> <tr> <td>Mode 7</td> <td>High impedance state</td> </tr> <tr> <td>Port 4</td> <td>↑</td> </tr> </table> <p>* The state of Port 3 differs depending on the mode.</p>			Port 1	High impedance state	Port 2	↑	* Modes 1, 5	↑	Port 3	E: ↑	E: "1" is output.	Mode 7	High impedance state	Port 4	↑	Applicable Manual	
Port 1				High impedance state														
Port 2				↑														
* Modes 1, 5				↑														
Port 3				E: ↑														
				E: "1" is output.														
Mode 7				High impedance state														
Port 4				↑														
				Title														
				Other Data														
	Title																	
	Microcomputer Technical Information (D1-23)																	
	Reference Q & A Sheet																	
	No.																	
	QA631-018A																	
Supplement	<p>E: The E clock is "H".</p> <p><math>\bar{E}</math>: The E clock is "L".</p>																	



Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M Evaluation kit, Emulator	Software	SD SBC																
Theme	SCI (Pin 39) State on Resetting		Date	Nov. 24, 1983																	
Question	1, What is the state of SCI (Pin 39) on resetting (RES='0')?			Classification																	
				Parallel Port																	
				Serial Port																	
				Timer/Counter																	
				BUS Interface																	
				Interrupt																	
				A/D Converter																	
				Oscillator																	
				* Reset																	
				Low Power Consm.																	
				EPROM-on-package																	
				Software																	
				Evaluation Kit																	
	Emulator																				
	SD																				
	Data Buffer																				
	Others																				
Answer	1, It is as follows: <table border="1" style="margin: 10px auto; width: 80%;"> <thead> <tr> <th>Mode</th> <th>SCI State</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The address strobe is output.</td> </tr> <tr> <td>1</td> <td style="text-align: center;">↑</td> </tr> <tr> <td>2</td> <td style="text-align: center;">↑</td> </tr> <tr> <td>4</td> <td style="text-align: center;">↑</td> </tr> <tr> <td>5</td> <td>"1" is output.</td> </tr> <tr> <td>6</td> <td>The address strobe is output.</td> </tr> <tr> <td>7</td> <td>High impedance state</td> </tr> </tbody> </table>			Mode	SCI State	0	The address strobe is output.	1	↑	2	↑	4	↑	5	"1" is output.	6	The address strobe is output.	7	High impedance state	Applicable Manual	
Mode				SCI State																	
0				The address strobe is output.																	
1				↑																	
2				↑																	
4				↑																	
5				"1" is output.																	
6				The address strobe is output.																	
7				High impedance state																	
				Title																	
	Other Data																				
	Title																				
	Reference Q & A Sheet																				
	No.																				
	QA631-017A																				
Supplement	SCI: Control signal of the HD6301V. The usage differs depending on the mode.																				

Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M Evaluation kit, Emulator	Software SD SBC
Theme	Port Output After Resetting		Date	Nov. 24, 1983
Question	<p>1, What data does a port output when the Data Direction Register(DDR)=1 after resetting?</p>			Classification
				Parallel Port
				Serial Port
				Timer/Counter
				BUS Interface
				Interrupt
				A/D Converter
				Oscillator
				* Reset
				Low Power Consm.
				EPROM-on-package
				Software
				Evaluation Kit
				Emulator
	SD			
	Data Buffer			
	Others			
Answer	<p>1, After resetting, since the Data Register of a port is undefined, undefined data is output when the DDR=1. Input definite data by programming in the Data Register before setting the DDR=1.</p>			Applicable Manual
				Title
				Other Data
				Title
	Reference Q & A Sheet			
	No.			
Supplement				

Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M Evaluation kit, Emulator	Software	SD SBC	
Theme	Schmitt Trigger Circuit of STBY			Date	Nov. 24, 1983	
Question	1, Is the Schmitt trigger circuit provided with the HD6303R STBY?			Classification		
				Parallel Port		
				Serial Port		
				Timer/Counter		
				BUS Interface		
				Interrupt		
				A/D Converter		
				Oscillator		
				Reset		
				* Low Power Consum.		
	EPROM-on-package					
	Software					
	Evaluation Kit					
	Emulator					
	SD					
	Data Buffer					
	Others					
Answer	1, Yes. (Mentioned in the HD6303R User's Manual.)  To the system power supply			Applicable Manual		
				Title	HD6301V User's Manual	
				Other Data		Title
	<p style="text-align: center;">HD6301V</p> <p><math>R_r \ll R_2, R_r - C_r \gg 20\text{ms}</math></p>			Reference Q & A Sheet		
				No.	QA631-015A QA631-016A	
Supplement				Effects of the Schmitt trigger circuit: Even on the slow rising edge of the input pulse, stable and clear waveform can be output.		

Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M Evaluation kit, Emulator	Software SD SBC								
Theme	I/O Port State During Standby		Date	Nov. 24, 1983								
Question	1, What is the state of each port during standby (STBY='0')?		Classification									
			Parallel Port									
			Serial Port									
			Timer/Counter									
			BUS Interface									
			Interrupt									
			A/D Converter									
			Oscillator									
			Reset									
			* Low Power Consm.									
			EPROM-on-package									
			Software									
			Evaluation Kit									
			Emulator									
			SD									
			Data Buffer									
			Others									
Answer	1, As follows:		Applicable Manual									
	<table border="1"> <tr> <td>Port 1</td> <td>High impedance state</td> </tr> <tr> <td>Port 2</td> <td>↑</td> </tr> <tr> <td>Port 3</td> <td>↑</td> </tr> <tr> <td>Port 4</td> <td>↑</td> </tr> </table>		Port 1	High impedance state	Port 2	↑	Port 3	↑	Port 4	↑	Title	
Port 1	High impedance state											
Port 2	↑											
Port 3	↑											
Port 4	↑											
			Other Data									
			Title									
			Microcomputer technical information (D1-23)									
			Reference Q & A Sheet									
			No.									
			QA631-017A									
			QA631-018A									
Supplement												

Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M Evaluation kit, Emulator	Software	SD SBC
Theme	Return from Standby Mode			Date	Nov. 24, 1983
Question	1, What occurs when returning from the standby mode without using RES?			Classification	
				Parallel Port	
				Serial Port	
				Timer/Counter	
				BUS Interface	
				Interrupt	
				A/D Converter	
				Oscillator	
				Reset	
				* Low Power Consm.	
				EPROM-on-package	
				Software	
				Evaluation Kit	
				Emulator	
	SD				
	Data Buffer				
	Others				
Answer	1, The MPU does not operate normally because the contents of each register are not definite. Therefore, always use the RES when returning from the standby mode.			Applicable Manual	
				Title	
				HD6301V1 data sheet	
				HD6301V user's manual	
	Other Data				
	Title				
	Reference Q & A Sheet				
	No.				
Supplement					

Type	HD6301V1 HD6303R	Device	4S *8S Evaluation kit,	8M Emulator	16M SD	Software SBC
Theme	Going into the Standby Mode			Date	Nov. 24, 1983	
Question	<p>1, Does the MCU go into the standby mode after current instruction execution is completed?</p>			Classification		
				Parallel Port		
				Serial Port		
				Timer/Counter		
				BUS Interface		
				Interrupt		
				A/D Converter		
				Oscillator		
				Reset		
				* Low Power Consm.		
				EPROM-on-package		
				Software		
				Evaluation Kit		
	Emulator					
	SD					
	Data Buffer					
	Others					
Answer	<p>1, No. Because there is no connection between the instruction execution <u>sequence</u> and the standby mode. That is, when the <u>STBY</u> pin goes into "Low", the state is latched at the next rising edge of E clock. Then the internal registers are reset at the next falling edge.</p> <p style="text-align: center;">Internal registers are reset.</p>			Applicable Manual		
				Title		
				Other Data		
				Title		
				Reference Q & A Sheet		
	No.			QA631-024A		
Supplement	<p>As standby mode detection has no connection with the instruction execution sequence, the MCU goes into the standby mode after preparing for standby mode with NMI routine.</p>					

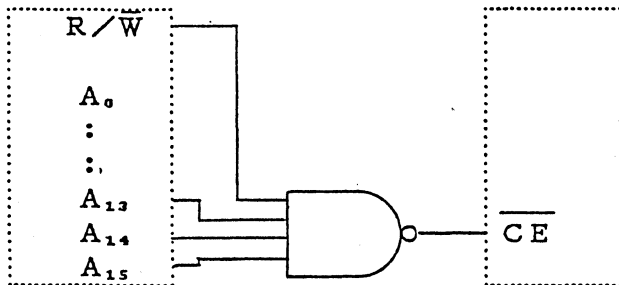
Type	HD6301V1 HD6303R	Device	4S *8S Evaluation kit,	8M Emulator	16M SD	Software SBC
Theme	Timing for the Standby Mode				Date	Nov. 24, 1983
Question	<p>1, The timing for the standby mode is shown in the HD6301V user's manual. <math>T_1</math> is not defined. How long is <math>T_1</math>?</p> <p>① NMI</p> <p>② RES</p> <p>③ STBY</p> <p>RAM Control Register Set</p> <p>Reset Start</p> <p>*<math>T_2</math> : Oscillation Stabilization Time</p>				Classification Parallel Port Serial Port Timer/Counter BUS Interface Interrupt A/D Converter Oscillator Reset * Low Power Consm. EPROM-on-package Software Evaluation Kit Emulator SD Data Buffer Others	
Answer					<p>1, After the RAM Control Register is set in the NMI routine, either STBY or RES can be in the low state with no priority.</p>	
Supplement	<p>RAM Control Register Set → RAM E bit = 0                  STBY PWR bit = 1                  RAM E bit = 0: Internal RAM is not accessible.                  STBY PWR bit = 1: Indicates that the data in standby RAM is valid.</p>					



Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M Software Evaluation kit, Emulator SD SBC
Theme	Usage of EPROM Socket Pins for the HD63P01M (No.1)		Date Nov. 24, 1983
Question	1, Are the data buses of the EPROM socket pins for the HD63P01M bi-directional in order to access not only the EPROM but the RAM?		Classification
			Parallel Port
			Serial Port
			Timer/Counter
			BUS Interface
			Interrupt
			A/D Converter
			Oscillator
			Reset
			Low Power Consm.
			* EPROM-on-package
			Software
			Evaluation Kit
			Emulator
	SD		
	Data Buffer		
	Others		
Answer	1, The data bus output from EPROM socket pins for the HD63P01M is Read only.		Applicable Manual
			Title
			HD63P01M Data Sheet
			Other Data
			Title
	Reference Q & A Sheet		
	No.		
	QA631-026A		
	QA631-027A		
Supplement			



Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M Evaluation kit, Emulator SD SBC
Theme	Usage of EPROM Socket Pins for the HD63P01M (No.2)		Date Nov.24, 1983
Question	1, In EPROM socket pins for the HD63P01M, what is CE composed of?		Classification
			Parallel Port
			Serial Port
			Timer/Counter
			BUS Interface
			Interrupt
			A/D Converter
			Oscillator
			Reset
			Low Power Consum.
			* EPROM-on-package
			Software
			Evaluation Kit
			Emulator
			SD
			Data Buffer
			Others
Answer	1, CE is a NAND circuit of the address bus (A <sub>13</sub> to A <sub>15</sub> ) and the MCU internal R/W signal. (Refer below.) Therefore, CE does not output in the dummy cycle. (When not accessing EPROM of HD63P01M)		Applicable Manual
			Title
			Other Data
			Title
			Reference Q & A Sheet
			No.
			QA631-025A
			QA631-027A
Supplement			



Type	HD6301V1 HD6303R	Device	4S *8S Evaluation kit,	8M Emulator	16M SD	Software SBC
Theme	Usage of EPROM Socket Pins for the HD63P01M (No.3)			Date	Nov. 24, 1983	
Question	<p>1, With EPROM socket pins for the HD63P01M,                  (1) Can pins drive one TTL load or more?                  (2) If not, what can pins drive?</p>			Classification		
				Parallel Port		
				Serial Port		
				Timer/Counter		
				BUS Interface		
				Interrupt		
				A/D Converter		
				Oscillator		
				Reset		
				Low Power Consm.		
				* EPROM-on-package		
				Software		
				Evaluation Kit		
	Emulator					
	SD					
	Data Buffer					
	Others					
Answer	<p>1, (1) The current of each pin is too little to drive one TTL load.                  (2) Each pin can drive one NMOS load.</p>			Applicable Manual		
				Title		
				Other Data		
				Title		
	Reference Q & A Sheet					
	No.					
	QA631-025A					
	QA631-026A					
Supplement						

Type	HD6301V1 HD6303R	Device	4S * 8S 8M 16M Software Evaluation kit, Emulator SD SBC
Theme	Usage of Bit Manipulator Instructions (No.1)		Date Nov. 24, 1983
Q	<p>1. How the bit manipulation instructions of the HD6301V should be written?</p>		Classification
			Parallel Port
			Serial Port
			Timer/Counter
			BUS Interface
			Interrupt
			A/D Converter
			Oscillator
			Reset
			Low Power Consm.
			EPROM-on-package
			* Software
	Evaluation Kit		
	Emulator		
	SD		
	Data Buffer		
	Others		
A	<p>1. They are written as follows; written as follows ;</p> <p>OIM # \$ 0 4 , \$ 1 0 (Direct Addressing) OIM # \$ 0 4 , \$ 1 0, X (Index Addressing)</p> <p style="margin-left: 40px;">↓            ↓            ↘</p> <p>Immediate Data    Address            Index Register</p> <p>This is an example of OR operation of the immediate data and the memory and storing the result in the memory.</p> <p>The HD6301V has the following bit manipulation instructions.</p> <p>OIM ..... (IMM) · (M) → (M) AIM ..... (IMM) + (M) → (M) EIM ..... (IMM) ⊕ (M) → (M) TIM ..... (IMM) · (M)</p> <p>These instructions are written in the same way. * Continued on the next page.</p>		Applicable Manual
			Title
			HD6301V Data Sheet HD6301V User's Manual
			Other Data
			Title
	Reference Q & A Sheet		
	No.		
	QA631-029A		
Supplement			

Type	HD6301V1 HD6303R	Device	4S * 8S 8M 16M Software Evaluation kit, Emulator SD SBC
Theme	Usage of Bit Manipulator Instructions (No.2)		

The following bit manipulations have different mnemonics in the same OP code.

OP code		Bit Manipulation Instruction		
		Mnemonics	Function	
71	61	A I M	B C L R	0 - Mi The memory bit i(i=0 to 7) is cleared and the other bits don't change.
72	62	O I M	B S E T	1 - Mi The memory bit i(i=0 to 7) is set and the other bits don't change.
75	65	E I M	B T G L	Mi - Mi The memory bit i(i=0 to 7) is inverted and the other bits don't change.
7B	6B	T I M	B T S T	1 · Mi AND operation test of the memory bit i(i=0 to 7) and "1" is executed and its corresponding condition code is changed.

↑                      ↙                      ↘  
Direct                  Index  
Addressing              Addressing

The mnemonics mentioned above can be written as follows.

```

B C L R  3, $ 1 0  ↔ A I M  # $ F 7 ,  $ 1 0      (Direct Addressing)
B C L R  3, $ 1 0, X ↔ A I M  # $ F 7 ,  $ 1 0 , X  (Index Addressing)

B S E T  3, $ 1 0  ↔ O I M  # $ 0 8 ,  $ 1 0      (Direct Addressing)
B S E T  3, $ 1 0, X ↔ O I M  # $ 0 8 ,  $ 1 0 , X  (Index Addressing)

```

↓                      ↓                      ↘  
 Bit    Address    Index Register

\*For details, see HD6301V Users Manual.

Type	HD6301V1 HD6303R	Device	4S * 8S Evaluation kit,	8M Emulator	16M SD	Software SBC
Theme	Usage of Bit Manipulation Instructions to the Port		Date	Nov. 24, 1983		
Q	1. Are the bit manipulation instructions (AIM, OIM, EIM, TIM) executable when a port is in the output state (DDR=1)?		Classification			
			Parallel Port			
			Serial Port			
			Timer/Counter			
			BUS Interface			
			Interrupt			
			A/D Converter			
			Oscillator			
			Reset			
			Low Power Consm.			
			EPROM-on-package.			
			* Software			
			Evaluation Kit			
			Emulator			
			SD			
			Data Buffer			
			Others			
A	1. It can be used if the port is in the output state (DDR=1). However, the bit manipulation instruction is executed as follows ;		Applicable Manual			
	1 Reads specified address.		Title			
	2 Executes logical operation					
	3 Writes the result into the specified address.		Other Data			
	Since the specified address(1) reads the pin state of the port, the data is influenced by the pins even if any data is output from the port.		Title			
			Reference Q & A Sheet			
			No.			
			QA631-028A			
Supplement	<p>DDR : Data Direction Register                  This register selects whether in the port is the input or the output state.</p> <p>DDR = 0 : Data input                  DDR = 1 : Data output</p>					

Type	HD6301V1 HD6303R	Device	4S *8S 8M 16M	Software	SD SBC
Theme	RAM Access Disable during Program Execution			Date	Nov. 24, 1983
Question	<p>1, When executing a program with the RAME bit of the RAM Control Register disabled,</p> <p>(1) What occurs if the internal RAM address is accessed?</p> <p>(2) What occurs if the interrupt requests are generated?</p>			Classification	
				Parallel Port	
				Serial Port	
				Timer/Counter	
				BUS Interface	
				Interrupt	
				A/D Converter	
				Oscillator	
				Reset	
				Low Power Consm.	
	EPROM-on-package				
	Software				
	Evaluation Kit				
	Emulator				
	SD				
	Data Buffer				
	* Others				
Answer	<p>1, (1) The external RAM can be accessed; the internal RAM is neither readable nor writable when the RAME bit is disabled.</p> <p>(2) If there is no stacking area other than the internal RAM, the MPU will burst when returning from the interrupt sequence.</p>			Applicable Manual	
				Title	
				HD6301V Data Sheet	
				HD6301V User's Manual	
				Other Data	
				Title	
	Reference Q & A Sheet				
	No.				
Supplement	<p>RAM Control Register</p> <p>\$0014 <span style="margin-left: 20px;">7</span> <span style="margin-left: 10px;">6</span> <span style="margin-left: 20px;">_____</span> <span style="margin-left: 20px;">_____</span> <span style="margin-left: 20px;">_____</span> <span style="margin-left: 20px;">_____</span> <span style="margin-left: 20px;">_____</span> <span style="margin-left: 20px;">_____</span></p> <p style="margin-left: 100px;"> </p> <p style="margin-left: 100px;">RAME bit</p>				
<p>* RAME='0' : Disable the Internal RAM Address</p>					

## 2. HD6301X0/HD6303X OSCILLATOR CIRCUIT

### Quartz Oscillation Circuit

#### 1. Quartz oscillation circuit and oscillation conditions

A typical quartz oscillation circuit and its equivalent circuit are shown in Fig. 1.

Oscillation conditions can be represented as follows:

$$|-R_z| > R_e \dots\dots\dots(1)$$

$$R_z = gm/w^2C_1C_2 \dots\dots\dots(2)$$

-R<sub>z</sub>: Quartz circuit resistance (based on quartz)

gm: Inverter transfer conductance

gm: dlout/dVin

Therefore, normal oscillation can be performed if negative resistance is sufficiently high.

However, oscillation stability is affected not only by external capacitance C<sub>1</sub> and C<sub>2</sub>, but also by external factors such as floating capacitance or resistance dependent on substrate circuit patterns, power stability time, and interference from other signal lines. Accordingly, sufficient care should be taken to pattern designing of the oscillation terminal periphery.

Regarding LSI, oscillation stability is affected by the inverter's gm. gm changes depending on inverter input voltage of the inverter, i.e., bias voltage.

#### 2. Oscillation halt and countermeasure

The oscillation circuit works under condition (1) above. However, in some cases, oscillation conditions are not satisfied because of the mutual interference described in 1 above.

To assure oscillation start, add resistance R<sub>L</sub> to the input (EXTAL) terminal of the oscillation circuit to fix bias voltage. 2 to 5 Mohm resistance is best.

#### 3. Explanation of oscillation halt and its countermeasure

This section explains oscillation halt based on LSI internal circuits.

A quartz oscillation circuit built in a microcomputer consists of inverter A used for oscillation and inverter B providing clocks on the LSI internal circuit.

Parasitic capacitance C<sub>M</sub> between these inverters' output and input generates negative feedback with a feedback ratio of C<sub>M</sub>/C<sub>I</sub>. Since inverter B appears in the same phase as inverter A, this negative feedback prevents oscillation.

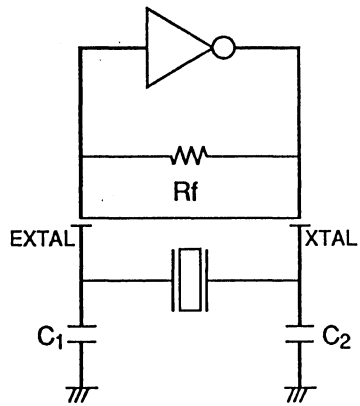
$$gm = \frac{C_i}{G.CM + C_i} gm \text{ ---- (3)}$$

G: Inverter B gain  
(voltage amplification ratio)

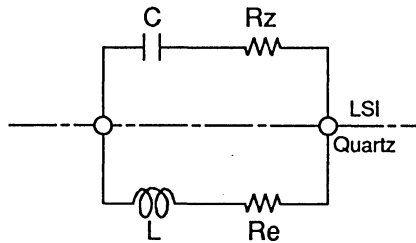
Inverter A's gm relatively reduces to (3), and load resistance R<sub>z</sub> of equation (2) also decreases, which prevents or stops oscillation.

Reducing inverter B's gain G increases gm according to (3). When resistance R<sub>L</sub> is added, inverter B's gain G can be reduced since the bias voltage deviates from the maximum gain point.

However, applying resistance R<sub>L</sub> reduces the gain of oscillation inverter A itself. Too small R<sub>L</sub> results in adverse effects. A stimulation result of R<sub>L</sub>'s optimized value is shown in Fig. 5. This is a transfer curve showing the change of oscillation circuit loop gain due to presence or absence of R<sub>L</sub>. It indicates that R<sub>L</sub> from 2-100 Mohm gives sufficient gain. However, optimum R<sub>L</sub> is 10 or less due to substrate leak current. 2-5 Mohm is best.

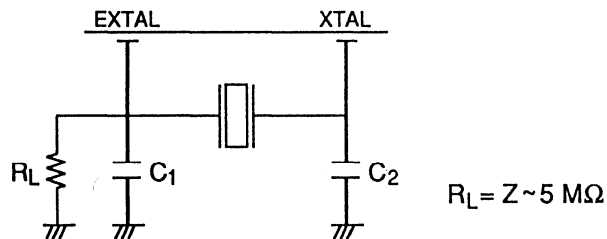


(a) Quartz oscillation circuit



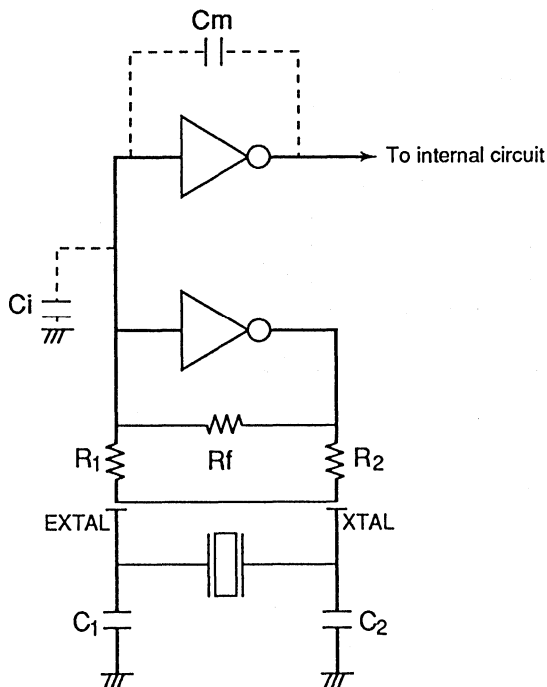
(b) Equivalent circuit

**Fig. 1 Quartz oscillation circuit and equivalent circuit**



**Fig. 2 Oscillation stop countermeasure**

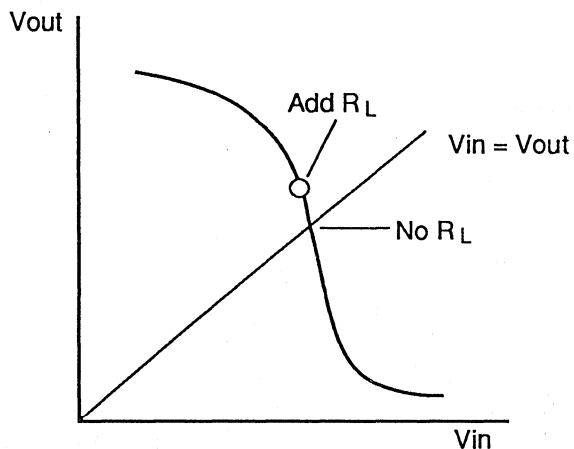




**Fig. 3 Practical oscillation circuit**

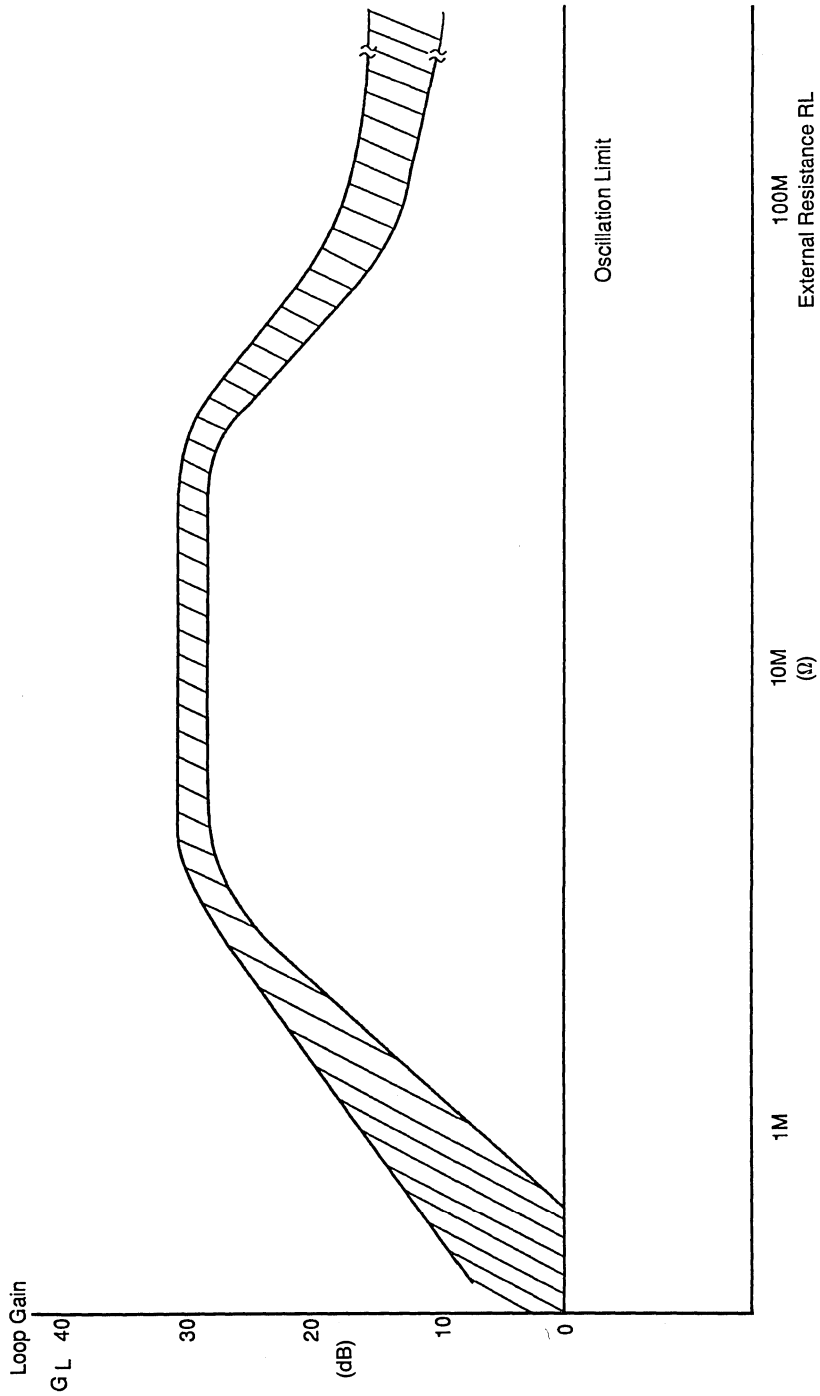
$R_f = 1 \text{ Mohm}$

$R_1, R_2 = 500 \text{ ohm (ESD protective resistance)}$



**Fig. 4 Inverter transfer curve  
(Input/output voltage characteristics)**

Bias voltage moves to the left on the transfer curve by adding  $R_L$



## Supplementary Description

### (1) Inverter parasitic capacitance

The inverter consists of a MOS transistor as shown in Fig. A. The MOS transistor has parasitic capacitance between its gate and drain, and called "mirror capacitance" of the inverter. It is generated since a diffusion layer spreads under the gate during drain formation (Fig. B).

### (2) Inverter gain and bias current

The maximum inverter gain is achieved when an inverter is biased by the voltage level where input voltage is equal to output voltage. The maximum inclination point of the transfer curve of Fig. C corresponds to the maximum gain point. This voltage is called logic threshold voltage VLT.

### (3) Oscillation circuit loop gain

Gain of an oscillation circuit with open loop modification (Fig. D) is represented as follows:

$$GL = \frac{|V_2|}{|V_1|}$$

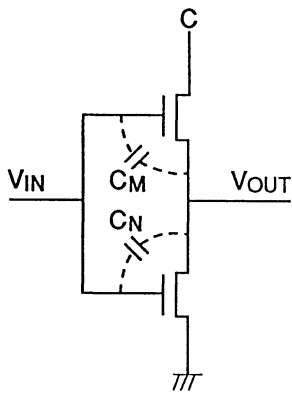


Fig. A Inverter circuit

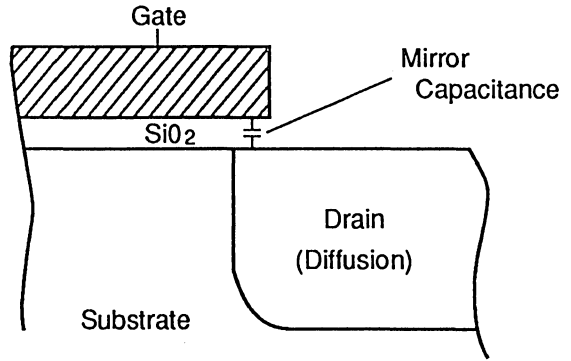


Fig. B MOS transistor cross section

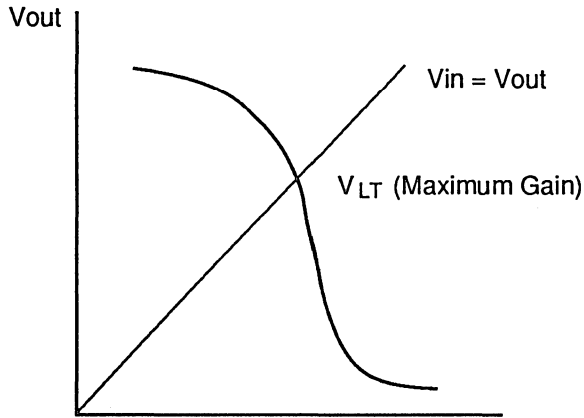


Fig. C Inverter transfer curve

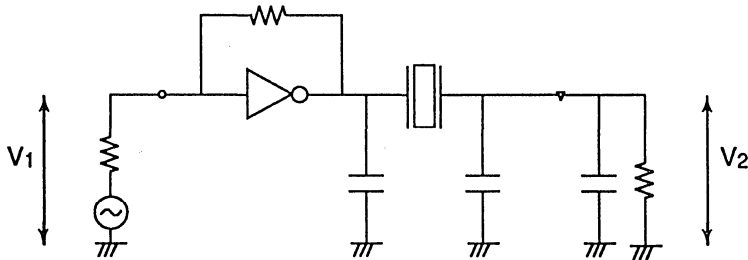


Fig. D Open-loop modification of oscillation circuit



### 3. Wide Temperature Range Specifications -40°C to +85°C (J Version)

The wide temperature range specifications for HD6301 and HD6303 devices are the same as the standard temperature range specifications, unless otherwise noted in Table I. The J version generic ordering number ends with letter J after the package designation, e.g., HD6301V1PJ.

**Table I—Summary of Differences Between Standard and J Version Specifications.**

DEVICE TYPE	SYMBOL	STANDARD VERSION	J VERSION	
HD6301V1 HD6303R	T <sub>a</sub>	0 ~ +70°C	-40°C ~ +85°C	
	T <sub>AHL</sub>	20 (min.)	30 (min.)	
	Package	DP-40, CP-44, CP-52, FP-54, CG-40	DP-40, CP-44, CP-52, FP-54*	
HD6301X0 HD6303X	T <sub>a</sub>	0 ~ +70°C	-40°C ~ +85°C	
	Package	DP-64S, CP-68, FP-80	DP-64S, CP-68, FP-80*	
HD6301Y0 HD6303Y	T <sub>a</sub>	-20°C ~ +70°C	-40°C ~ +85°C	
	T <sub>HLR</sub>	(f = 1, 1.5, 2 MHz)	10	5
		(f = 3 MHz)	5	5
	T <sub>TXD</sub>	220	240	
	V <sub>IH</sub>	RES, STBY	V <sub>CC</sub> - 0.5	Same as standard
		EXTAL	V <sub>CC</sub> × 0.7	Same as standard
		Other Inputs	2.0	2.1
Package	DP-64S, CP-68, FP-64	DP-64S, CP-68, FP-64*		

\*Please contact Hitachi Sales Office.

**WIDE TEMPERATURE  
SPECIFICATIONS**  
-40°C TO +85°C  
(J VERSION)

# HD6301V1, HD63A01V1, HD63B01V1

The HD6301V1 is an 8-bit CMOS single-chip microcomputer unit, Object Code compatible with the HD6801. 4kB ROM, 128 bytes RAM, Serial Communication Interface (SCI), parallel I/O ports and multi function timer are incorporated in the HD6301V1. It is bus compatible with HMCS6800. Execution time of key instructions are improved and several new instructions are added to increase system throughput. The HD6301V1 can be expanded up to 65k words. Like the HMCS6800 family, I/O level is TTL compatible with +5.0V single power supply. As HD6301V1 is fabricated by the advanced CMOS process technology, power dissipation is extremely reduced. In addition to that, HD6301V1 has Sleep Mode and Standby Mode at lower power dissipation mode. Therefore flexible low power consumption application is possible.

## ■ FEATURES

- Object Code Upward Compatible with HD6801 Family
- Abundant On-Chip Functions Compatible with HD6801V0; 4kB ROM, 128 Bytes RAM, 29 Parallel I/O Lines, 2 Lines of Data Strobe, 16-bit Timer, Serial Communication Interface
- Low Power Consumption Mode: Sleep Mode, Standby Mode
- Minimum Instruction Execution Time  
1 $\mu$ s (f=1MHz), 0.67 $\mu$ s (f=1.5MHz), 0.5 $\mu$ s (f=2MHz)
- Bit Manipulation, Bit Test Instruction
- Protection from System Upset: Address Trap, On-Code Trap
- Up to 65k Words Address Space
- Wide Operation Range  
f = 0.1 to 2.0MHz (V<sub>CC</sub> = 5V  $\pm$  10%)

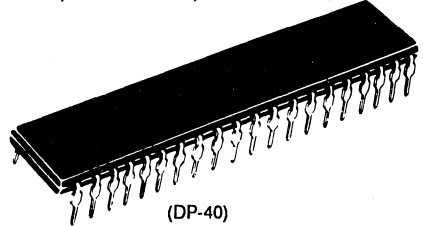
## ■ TYPE OF PRODUCTS

Type No.	Bus Timing
HD6301V1	1 MHz
HD63A01V1	1.5 MHz
HD63B01V1	2 MHz

## ■ GENERIC PART NUMBER

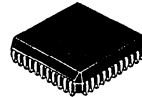
HD6301V1PJ, HD63A01V1PJ, HD63B01V1PJ
HD6301V1LJ, HD63A01V1LJ, HD63B01V1LJ**
HD6301V1CPJ, HD63A01V1CPJ, HD63B01V1CPJ

HD6301V1PJ, HD63A01V1PJ, HD63B01V1PJ



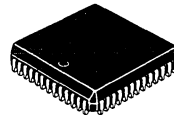
(DP-40)

HD6301V1LJ, HD63A01V1LJ, HD63B01V1LJ\*\*



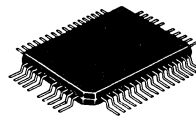
(CP-44)

HD6301V1CPJ, HD63A01V1CPJ, HD63B01V1CPJ



(CP-52)

HD6301V1F\*, HD63A01V1F\*, HD63B01V1F\*



(FP-54)

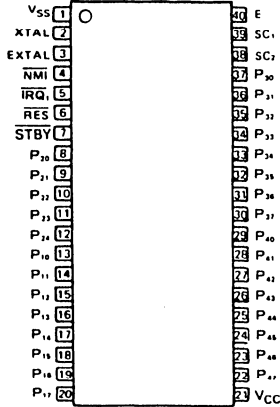
\* Contact Hitachi Sales Office

\*\* HD63B01V1LJ Operates Only in Single Chip Mode

**WIDE TEMPERATURE SPECIFICATIONS**  
**-40°C TO +85°C**  
**(J VERSION)**

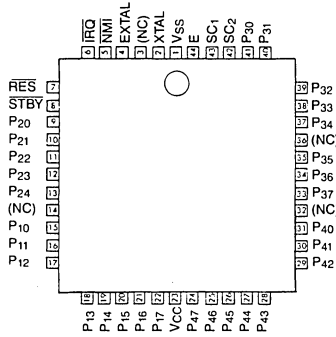
■ PIN ARRANGEMENT

- HD6301V1PJ, HD63A01V1PJ, HD63B01V1PJ (DP-40)



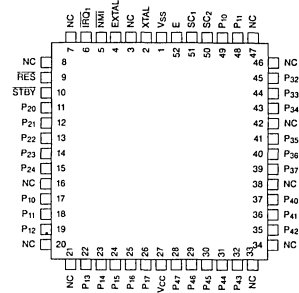
(Top View)

- HD6301V1LJ, HD63A01V1LJ, HD63B01V1LJ (CP-44)

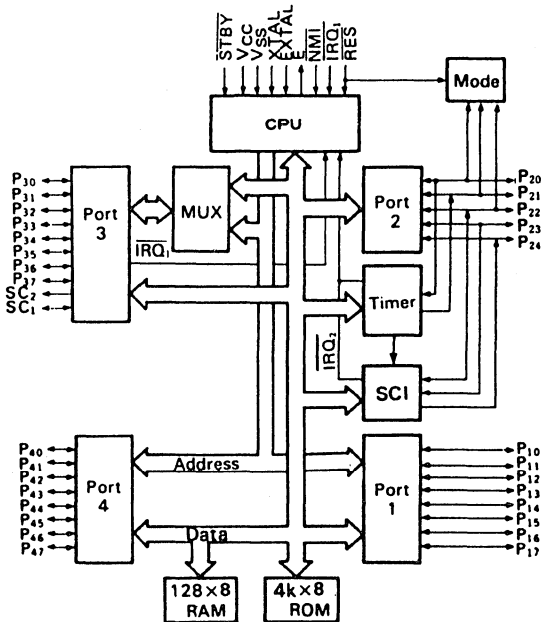


(Top View)

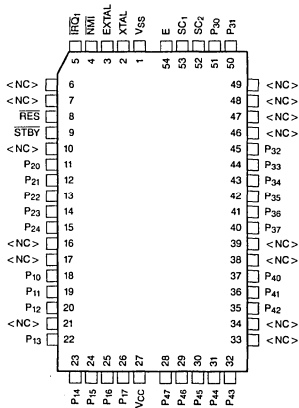
- HD6301V1CPJ, HD63A01V1CPJ, HD63B01V1CPJ (CP-52)



■ BLOCK DIAGRAM



- HD6301V1F\*, HD63A01V1F\*, HD63B01V1F\* (FP-54)



\*Contact Hitachi Sales Office



■ **ABSOLUTE MAXIMUM RATINGS**

Item	Symbol	Value	Unit
Supply Voltage	$V_{CC}$	-0.3 ~ +7.0	V
Input Voltage	$V_{in}$	-0.3 ~ $V_{CC} + 0.3$	V
Operating Temperature*	$T_{opr}$	-40 to +85	°C
Storage Temperature	$T_{stg}$	-55 to +150	°C

(NOTE) This product has protection circuits in input terminal from high static electricity voltage and high electric field.  
But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend  $V_{in}$ ,  $V_{out}$ :  $V_{SS} \leq (V_{in} \text{ or } V_{out}) \leq V_{CC}$ .

\*K version (-40 to +125°C) available. Contact sales office.

■ **ELECTRICAL CHARACTERISTICS**

- **DC CHARACTERISTICS** ( $V_{CC} = 5.0V \pm 10\%$ ,  $V_{SS} = 0V$ ,  $T_a = -40$  to  $+85^\circ C$ , unless otherwise noted.)

Item	Symbol	Test Condition	min	typ	max	Unit	
Input "High" Voltage	$V_{IH}$		$V_{CC} - 0.5$	-	$V_{CC} + 0.3$	V	
			$V_{CC} \times 0.7$	-			
			2.0	-			
Input "Low" Voltage	$V_{IL}$		-0.3	-	0.8	V	
Input Leakage Current	$I_{in}$	$V_{in} = 0.5 \sim V_{CC} - 0.5V$	-	-	1.0	$\mu A$	
Three State (off-state) Leakage Current	$I_{TSI}$	$V_{in} = 0.5 \sim V_{CC} - 0.5V$	-	-	1.0	$\mu A$	
Output "High" Voltage	$V_{OH}$		$I_{OH} = -200\mu A$	2.4	-	V	
			$I_{OH} = -10\mu A$	$V_{CC} - 0.7$	-	V	
Output "Low" Voltage	$V_{OL}$	$I_{OL} = 1.6mA$	-	-	0.55	V	
Input Capacitance	$C_{in}$	$V_{in} = 0V$ , $f = 1.0MHz$ , $T_a = 25^\circ C$	-	-	12.5	pF	
Standby Current	$I_{CC}$	$V_{IL}(\overline{STBY}) = 0 \sim 0.6V$ $V_{IH}(\overline{RES}) = V_{CC} - 0.5 \sim V_{CC} V$ $V_{IL}(\overline{RES}) = 0 \sim 0.6V$	-	2.0	15.0	$\mu A$	
Current Dissipation*	$I_{CC}$		Operating (f=1MHz**)	-	6.0	10.0	mA
			Sleeping (f=1MHz**)	-	1.0	2.0	
RAM Stand-By Voltage	$V_{RAM}$		2.0	-	-	V	

\*  $V_{IH} \text{ min} = V_{CC} - 1.0V$ ,  $V_{IL} \text{ max} = 0.8V$

\*\* Current Dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ. or max. values about Current Dissipations at  $f = x$  MHz operation are decided according to the following formula;

typ. value (f = x MHz) = typ. value (f = 1MHz) x x

max. value (f = x MHz) = max. value (f = 1MHz) x x

(both the sleeping and operating)

• AC CHARACTERISTICS (V<sub>CC</sub> = 5.0V±10%, V<sub>SS</sub> = 0V, T<sub>a</sub> = -40 to +85°C, unless otherwise noted.)

BUS TIMING

Item	Symbol	Test Condition	HD6301V1			HD63A01V1			HD63B01V1			Unit
			min	typ	max	min	typ	max	min	typ	max	
Cycle Time	t <sub>cyc</sub>	Fig. 5-1  Fig. 5-2	1	-	10	0.666	-	10	0.5	-	10	μs
Address Strobe Pulse Width "High"	PW <sub>ASH</sub>		220	-	-	150	-	-	110	-	-	ns
Address Strobe Rise Time	t <sub>ASr</sub>		-	-	20	-	-	20	-	-	20	ns
Address Strobe Fall Time	t <sub>ASf</sub>		-	-	20	-	-	20	-	-	20	ns
Address Strobe Delay Time	t <sub>ASD</sub>		60	-	-	40	-	-	20	-	-	ns
Enable Rise Time	t <sub>Er</sub>		-	-	20	-	-	20	-	-	20	ns
Enable Fall Time	t <sub>Ef</sub>		-	-	20	-	-	20	-	-	20	ns
Enable Pulse Width "High" Level	PW <sub>EH</sub>		450	-	-	300	-	-	220	-	-	ns
Enable Pulse Width "Low" Level	PW <sub>EL</sub>		450	-	-	300	-	-	220	-	-	ns
Address Strobe to Enable Delay Time	t <sub>ASED</sub>		60	-	-	40	-	-	20	-	-	ns
Address Delay Time	t <sub>AD1</sub>		-	-	250	-	-	190	-	-	160	ns
	t <sub>AD2</sub>		-	-	250	-	-	190	-	-	160	ns
Address Delay Time for Latch	t <sub>ADL</sub>		-	-	250	-	-	190	-	-	160	ns
Data Set-up Time	Write t <sub>DSW</sub>		230	-	-	150	-	-	100	-	-	ns
	Read t <sub>DSR</sub>		80	-	-	60	-	-	50	-	-	ns
Data Hold Time	Read t <sub>HR</sub>		0	-	-	0	-	-	0	-	-	ns
	Write t <sub>HW</sub>		20	-	-	20	-	-	20	-	-	ns
Address Set-up Time for Latch	t <sub>ASL</sub>		60	-	-	40	-	-	20	-	-	ns
Address Hold Time for Latch	t <sub>AHL</sub>		30	-	-	20	-	-	20	-	-	ns
Address Hold Time	t <sub>AH</sub>		20	-	-	20	-	-	20	-	-	ns
A <sub>0</sub> ~ A <sub>7</sub> Set-up Time Before E	t <sub>ASM</sub>		200	-	-	110	-	-	60	-	-	ns
Peripheral Read Access Time	Non-Multiplexed Bus (t <sub>ACCN</sub> )		-	-	650	-	-	395	-	-	270	ns
	Multiplexed Bus (t <sub>ACCM</sub> )		-	-	650	-	-	395	-	-	270	ns
Oscillator stabilization Time	t <sub>RC</sub>		Fig.2-7-1	20	-	-	20	-	-	20	-	ms
Processor Control Set-up Time	t <sub>PCS</sub>		Fig.2-8-1	200	-	-	200	-	-	200	-	ns

PERIPHERAL PORT TIMING

Item	Symbol	Test Condition	HD6301V1			HD63A01V1			HD63B01V1			Unit
			min	typ	max	min	typ	max	min	typ	max	
Peripheral Data Set-up Time	Port 1, 2, 3, 4 t <sub>PDSU</sub>	Fig. 5-3 ..	200	-	-	200	-	-	200	-	-	ns
Peripheral Data Hold Time	Port 1, 2, 3, 4 t <sub>PDH</sub>	Fig. 5-3 ..	200	-	-	200	-	-	200	-	-	ns
Delay Time, Enable Positive Transition to OS3 Negative Transition	t <sub>OSD1</sub>	Fig. 5-5 ..	-	-	300	-	-	300	-	-	300	ns
Delay Time, Enable Positive Transition to OS3 Positive Transition	t <sub>OSD2</sub>	Fig. 5-5 ..	-	-	300	-	-	300	-	-	300	ns
Delay Time, Enable Negative Transition to Peripheral Data Valid	Port 1, 2, 3, 4 t <sub>PWD</sub>	Fig. 5-4 ..	-	-	300	-	-	300	-	-	300	ns
Input Strobe Pulse Width	t <sub>PWIS</sub>	Fig. 5-6 ..	200	-	-	200	-	-	200	-	-	ns
Input Data Hold Time	Port 3 t <sub>IH</sub>	Fig. 5-6 ..	150	-	-	150	-	-	150	-	-	ns
Input Data Setup Time	Port 3 t <sub>IS</sub>	Fig. 5-6 ..	0	-	-	0	-	-	0	-	-	ns

\* Except P<sub>21</sub>

\*\* Refer to Pages 189-190

\*\*\* Refer to Pages 159-160



**TIMER, SCI TIMING**

Item	Symbol	Test Condition	HD6301V1			HD63A01V1			HD63B01V1			Unit
			min	typ	max	min	typ	max	min	typ	max	
Timer Input Pulse Width	t <sub>PWT</sub>		2.0	-	-	2.0	-	-	2.0	-	-	t <sub>cyc</sub>
Delay Time, Enable Positive Transition to Timer Out	t <sub>TOD</sub>	Fig. 5-7	-	-	400	-	-	400	-	-	400	ns
SCI Input Clock Cycle	t <sub>Scyc</sub>		2.0	-	-	2.0	-	-	2.0	-	-	t <sub>cyc</sub>
SCI Input Clock Pulse Width	t <sub>PWCK</sub>		0.4	-	0.6	0.4	-	0.6	0.4	-	0.6	t <sub>Scyc</sub>

**MODE PROGRAMMING**

Item	Symbol	Test Condition	HD6301V1			HD63A01V1			HD63B01V1			Unit
			min	typ	max	min	typ	max	min	typ	max	
RES "Low" Pulse Width	PW <sub>RSTL</sub>		3	-	-	3	-	-	3	-	-	t <sub>cyc</sub>
Mode Programming Set-up Time	t <sub>MPS</sub>	Fig. 5-8	2	-	-	2	-	-	2	-	-	t <sub>cyc</sub>
Mode Programming Hold Time	t <sub>MPH</sub>		150	-	-	150	-	-	150	-	-	ns

\*\* Refer to Pages 189-190

# HD6301X0, HD63A01X0, HD63B01X0

**WIDE TEMPERATURE  
SPECIFICATIONS**  
-40°C TO +85°C  
(J VERSION)

The HD6301X0 is a CMOS single-chip microcomputer unit (MCU) which includes a CPU compatible with the HD6301V1, 4k bytes of ROM, 192 bytes of RAM, 53 parallel I/O pins, a Serial Communication Interface (SCI) and two timers on chip.

## ■ FEATURES

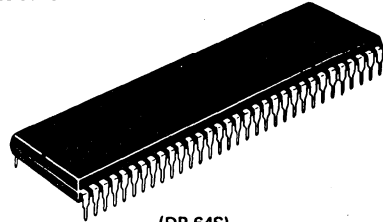
- Instruction Set Compatible with the HD6301V1
- Abundant On-chip Functions
  - 4k Bytes of ROM, 192 Bytes of RAM
  - 53 Parallel I/O Ports
  - 16-Bit Programmable Timer
  - 8-Bit Reloadable Timer
  - Serial Communication Interface
  - Memory Ready
  - Halt
  - Error-Detection (Address Trap, Op Code Trap)
- Interrupts . . . 3 External, 7 Internal
- Operation Mode
  - Mode 1 . . . Expanded (Internal ROM Inhibited)
  - Mode 2 . . . Expanded (Internal ROM Valid)
  - Mode 3 . . . Single-chip Mode
- Low Power Dissipation Mode
  - Sleep
  - Standby
- Wide Range of Operation
  - $V_{CC} = 5V \pm 10\%$ 
    - $f = 0.1$  to 1.0MHz: HD6303Y
    - $f = 0.1$  to 1.5MHz: HD63A03Y
    - $f = 0.1$  to 2.0MHz: HD63B03Y
    - $f = 0.1$  to 3.0MHz: HD63C03Y

## ■ GENERIC PART NUMBER

HD6301X0PJ, HD63A01X0PJ, HD63B01X0PJ

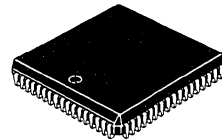
HD6301X0CPJ, HD63A01X0CPJ, HD63B01X0CPJ

HD6301X0PJ, HD63A01X0PJ,  
HD63B01X0PJ



(DP-64S)

HD6301X0CPJ, HD63A01X0CPJ,  
HD63B01X0CPJ



(CP-68)

HD6301X0F\*, HD63A01X0F\*,  
HD63B01X0F\*



(FP-80)

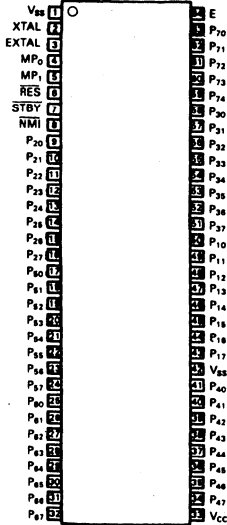
\*Contact Hitachi Sales Office

**WIDE TEMPERATURE  
SPECIFICATIONS  
-40°C TO +85°C  
(J VERSION)**

**HD6301X0, HD63A01X0, HD63B01X0**

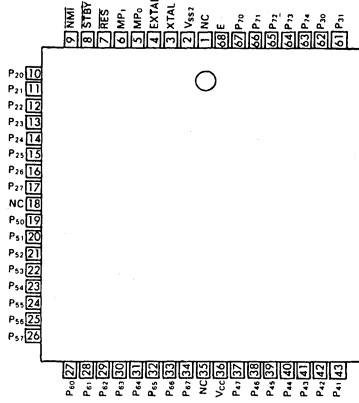
■ **PIN ARRANGEMENT**

- HD6301X0PJ, HD63A01X0PJ, HD63B01X0PJ



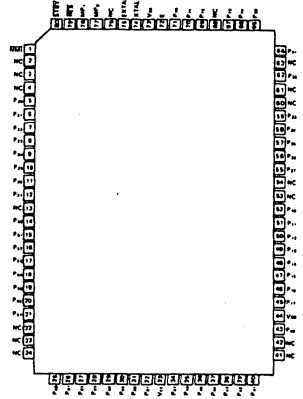
(Top View)  
(DP-64S)

- HD6301X0CPJ, HD63A01X0CPJ, HD63B01X0CPJ



(Top View)  
(CP-68)

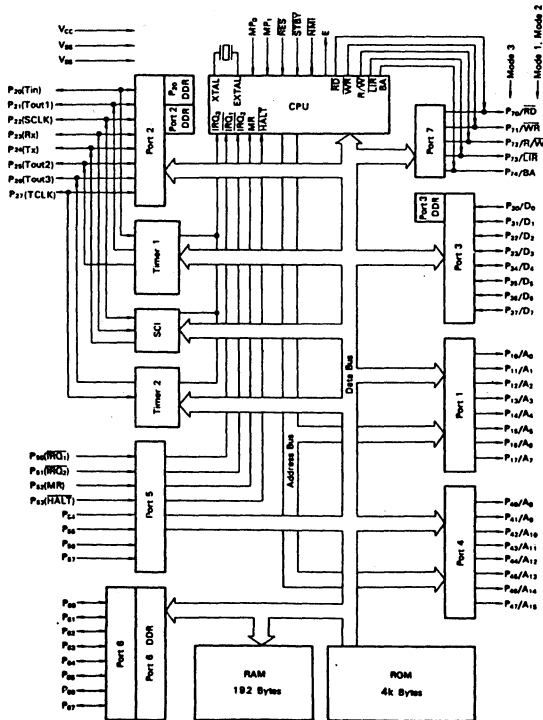
- HD6301X0F\*, HD63A01X0F\*, HD63B01X0F\*



(Top View)  
(FP-80)

\*Contact Hitachi Sales Office

■ **BLOCK DIAGRAM**



■ **ABSOLUTE MAXIMUM RATINGS**

Item	Symbol	Value	Unit
Supply Voltage	$V_{CC}$	-0.3 ~ +7.0	V
Input Voltage	$V_{in}$	-0.3 ~ $V_{CC} + 0.3$	V
Operating Temperature	$T_{opr}$	-40 to +85	°C
Storage Temperature	$T_{stg}$	-55 to +150	°C

(NOTE) This product has protection circuits in input terminal from high static electricity voltage and high electric field.  
But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend  $V_{in}, V_{out}, V_{SS} \leq (V_{in} \text{ or } V_{out}) \leq V_{CC}$ .

■ **ELECTRICAL CHARACTERISTICS**

● **DC CHARACTERISTICS ( $V_{CC} = 5.0V \pm 10\%$ ,  $V_{SS} = 0V$ ,  $T_a = -40$  to  $+85^\circ C$ , unless otherwise noted.)**

Item	Symbol	Test Condition	min	typ	max	Unit	
Input "High" Voltage	RES, STBY	$V_{IH}$	$V_{in} = 0.5 \sim V_{CC} - 0.5V$	$V_{CC} - 0.5$	-	$V_{CC} + 0.3$	V
	EXTAL			$V_{CC} \times 0.7$	-		
	Other Inputs			2.0	-		
Port 22	$V_{IH}$		2.2	-		V	
Input "Low" Voltage	All Inputs	$V_{IL}$	-0.3	-	0.8	V	
Input Leakage Current	NMI, RES, STBY, MP <sub>0</sub> , MP <sub>1</sub> , Port 5	$ I_{in} $			1.0	$\mu A$	
Three State (off-state) Leakage Current	Ports 1, 2, 3, 4, 6, 7	$ I_{TSI} $			1.0	$\mu A$	
Output "High" Voltage	All Outputs	$V_{OH}$	$I_{OH} = -200\mu A$	2.4	-	-	V
			$I_{OH} = -10\mu A$	$V_{CC} - 0.7$	-	-	V
Output "Low" Voltage	All Outputs	$V_{OL}$			0.4	V	
Darlington Drive Current	Ports 2, 6	$-I_{OH}$	$V_{out} = 1.5V$	1.0	-	10.0	mA
Input Capacitance	All Inputs	$C_{in}$	$V_{in} = 0V, f = 1MHz, T_a = 25^\circ C$			12.5	pF
Standby Current	Non Operation	$I_{STB}$			3.0	15.0	$\mu A$
Current Dissipation*	$I_{SLP}$		Sleeping (f = 1MHz)**		1.5	3.0	mA
			Sleeping (f = 1.5MHz)**		2.3	4.5	mA
			Sleeping (f = 2MHz)**		3.0	6.0	mA
	$I_{CC}$		Operating (f = 1MHz)**		7.0	10.0	mA
			Operating (f = 1.5MHz)**		10.5	15.0	mA
			Operating (f = 2MHz)**		14.0	20.0	mA
RAM Standby Voltage	$V_{RAM}$		2.0	-	-	V	

\*  $V_{IH} \text{ min} = V_{CC} - 1.0V, V_{IL} \text{ max} = 0.8V$  (All output terminals are at no load.)

\*\* Current Dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ. or max. values about Current Dissipations at x MHz operation are decided according to the following formula;

typ. value (f = x MHz) = typ. value (f = 1MHz) x x  
 max. value (f = x MHz) = max. value (f = 1MHz) x x  
 (both the sleeping and operating)

● **AC CHARACTERISTICS** ( $V_{CC} = 5.0V \pm 10\%$ ,  $V_{SS} = 0V$ ,  $T_a = -40$  to  $+85^\circ C$ , unless otherwise noted.)

**BUS TIMING**

Item	Symbol	Test Condition	HD6301X0			HD63A01X0			HD63B01X0			Unit
			min	typ	max	min	typ	max	min	typ	max	
Cycle Time	$t_{cyc}$	Fig. I-1 **	1	—	10	0.666	—	10	0.5	—	10	$\mu s$
Enable Rise Time	$t_{Er}$		—	—	25	—	—	25	—	—	25	ns
Enable Fall Time	$t_{Ef}$		—	—	25	—	—	25	—	—	25	ns
Enable Pulse Width "High" Level*	$PW_{EH}$		450	—	—	300	—	—	220	—	—	ns
Enable Pulse Width "Low" Level*	$PW_{EL}$		450	—	—	300	—	—	220	—	—	ns
Address, R/W Delay Time*	$t_{AD}$		—	—	250	—	—	190	—	—	160	ns
Data Delay Time	Write $t_{DDW}$		—	—	200	—	—	160	—	—	120	ns
Data Set-up Time	Read $t_{DSR}$		80	—	—	70	—	—	70	—	—	ns
Address, R/W Hold Time*	$t_{AH}$		80	—	—	50	—	—	35	—	—	ns
Data Hold Time	Write* $t_{HW}$		80	—	—	50	—	—	40	—	—	ns
	Read $t_{HR}$		0	—	—	0	—	—	0	—	—	ns
$\overline{RD}$ , $\overline{WR}$ Pulse Width*	$PW_{RW}$		450	—	—	300	—	—	220	—	—	ns
$\overline{RD}$ , $\overline{WR}$ Delay Time	$t_{RWD}$		—	—	40	—	—	40	—	—	40	ns
$\overline{RD}$ , $\overline{WR}$ Hold Time	$t_{HRW}$		—	—	30	—	—	30	—	—	25	ns
LTR Delay Time	$t_{DLR}$		—	—	200	—	—	160	—	—	120	ns
LTR Hold Time	$t_{HLR}$		10	—	—	10	—	—	10	—	—	ns
MR Set-up Time*	$t_{SMR}$		400	—	—	280	—	—	230	—	—	ns
MR Hold Time*	$t_{HMR}$		—	—	90	—	—	40	—	—	0	ns
E Clock Pulse Width at MR	$PW_{EMR}$		—	—	9	—	—	9	—	—	9	$\mu s$
Processor Control Set-up Time	$t_{PCS}$	Fig. I-3 I-11, I-12	200	—	—	200	—	—	200	—	—	ns
Processor Control Rise Time	$t_{PCR}$	Fig. I-2, I-3**	—	—	100	—	—	100	—	—	100	ns
Processor Control Fall Time	$t_{PCF}$	Fig. I-2, I-3**	—	—	100	—	—	100	—	—	100	ns
BA Delay Time	$t_{BA}$	Fig. I-3 **	—	—	250	—	—	190	—	—	160	ns
Oscillator Stabilization Time	$t_{RC}$	Fig. I-12**	20	—	—	20	—	—	20	—	—	ms
Reset Pulse Width	$PW_{RST}$		3	—	—	3	—	—	3	—	—	$t_{cyc}$

\* These timings change in approximate proportion to  $t_{cyc}$ . The figures in this characteristics represent those when  $t_{cyc}$  is minimum (= in the highest speed operation).

\*\* Refer to Pages 466-469

**PERIPHERAL PORT TIMING**

Item	Symbol	Test Condition	HD6301X0			HD63A01X0			HD63B01X0			Unit
			min	typ	max	min	typ	max	min	typ	max	
Peripheral Data Set-up Time	Ports 2, 3, 5, 6 $t_{PDSU}$	Fig. I-5 **	200	—	—	200	—	—	200	—	—	ns
Peripheral Data Hold Time	Ports 2, 3, 5, 6 $t_{PDH}$	Fig. I-5 **	200	—	—	200	—	—	200	—	—	ns
Delay Time (Enable Negative Transition to Peripheral Data Valid)	Ports 1, 2, 3, 4, 6, 7 $t_{PWD}$	Fig. I-6 **	—	—	300	—	—	300	—	—	300	ns

\*\* Refer to Pages 466-469

**TIMER, SCI TIMING**

Item	Symbol	Test Condition	HD6301X0			HD63A01X0			HD63B01X0			Unit
			min	typ	max	min	typ	max	min	typ	max	
Timer 1 Input Pulse Width	t <sub>PWT</sub>	Fig. I-9**	2.0	–	–	2.0	–	–	2.0	–	–	t <sub>cyc</sub>
Delay Time (Enable Positive Transition to Timer Output)	t <sub>TOD</sub>	Fig. I-7, I-8**	–	–	400	–	–	400	–	–	400	ns
SCI Input Clock Cycle	Async. Mode	Fig. I-9**	1.0	–	–	1.0	–	–	1.0	–	–	t <sub>cyc</sub>
	Clock Sync.		Fig. I-4, I-9**	2.0	–	–	2.0	–	–	2.0	–	–
SCI Transmit Data Delay Time (Clock Sync. Mode)	t <sub>TXD</sub>	Fig. I-4**	–	–	200	–	–	200	–	–	200	ns
SCI Receive Data Set-up Time (Clock Sync. Mode)	t <sub>SRX</sub>		290	–	–	290	–	–	290	–	–	ns
SCI Receive Data Hold Time (Clock Sync. Mode)	t <sub>HRX</sub>		100	–	–	100	–	–	100	–	–	ns
SCI Input Clock Pulse Width	t <sub>PWSCk</sub>	Fig. I-9**	0.4	–	0.6	0.4	–	0.6	0.4	–	0.6	t <sub>Scyc</sub>
Timer 2 Input Clock Cycle	t <sub>tcyc</sub>		2.0	–	–	2.0	–	–	2.0	–	–	t <sub>cyc</sub>
Timer 2 Input Clock Pulse Width	t <sub>PWTCK</sub>		200	–	–	200	–	–	200	–	–	ns
Timer 1-2, SCI Input Clock Rise Time	t <sub>CKr</sub>		–	–	100	–	–	100	–	–	100	ns
Timer 1-2, SCI Input Clock Fall Time	t <sub>CKf</sub>		–	–	100	–	–	100	–	–	100	ns

\*\*Refer to Pages 466–469



**WIDE TEMPERATURE  
SPECIFICATIONS**  
-40°C TO +85°C  
(J VERSION)

# HD6301Y0, HD63A01Y0, HD63B01Y0, HD63C01Y0

The HD6301Y0 is a CMOS 8-bit single-chip microcomputer unit which contains a CPU compatible with the CMOS 8-bit microcomputer HD6301V, 16k bytes of ROM, 256 bytes of RAM, 53 parallel I/O pins, Serial Communication Interface (SCI) and two timers.

## ■ FEATURES

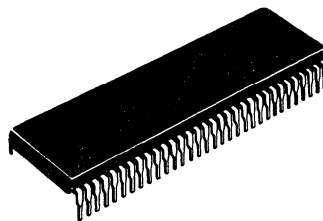
- Instruction Set Compatible with the HD6301V1
- 16k Bytes of ROM, 256 Bytes of RAM
- 53 Parallel I/O Pins  
(48 I/O Pins, 5 Output Pins)
- Parallel Handshake Interface (Port 6)
- Darlington Transistor Drive (Port 2, 6)
- 16-Bit Programmable Timer
  - Input Capture Register × 1
  - Free Running Counter × 1
  - Output Compare Register × 2
- 8-Bit Reloadable Timer
  - External Event Counter
  - Square Wave Generation
- Serial Communication Interface (SCI)
  - Asynchronous Mode (8 Transmit Formats, Hardware Parity)
  - Clocked Synchronous Mode
- Memory Ready
  - 3 Kinds of Memory Ready
- Halt
- Error Detection  
(Address Error, Op-code Error)
- Interrupt — External 3, Internal 7
- Operation Mode
  - Mode 1; Expanded Mode  
(Internal ROM Inhibited)
  - Mode 2; Expanded Mode  
(Internal ROM Valid)
  - Mode 3; Single Chip Mode
- Maximum 65K Bytes Address Space
- Low Power Dissipation Mode
  - Sleep Mode
  - Standby Mode (Hardware Standby, Software Standby)
- Minimum Instruction Execution Time — 0.5μs (f = 2MHz)
- Wide Range of Operation

$$V_{cc} = 5V \pm 10\% \left\{ \begin{array}{l} f = 0.1 \text{ to } 1.0\text{MHz} : \text{HD6301Y0} \\ f = 0.1 \text{ to } 1.5\text{MHz} : \text{HD63A01Y0} \\ f = 0.1 \text{ to } 2.0\text{MHz} : \text{HD63B01Y0} \\ f = 0.1 \text{ to } 3.0\text{MHz} : \text{HD63C01Y0} \end{array} \right.$$

## ■ GENERIC PART NUMBER

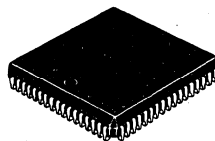
HD6301Y0PJ, HD63A01Y0PJ, HD63B01Y0PJ, HD63C01Y0PJ  
 HD6301Y0CPJ, HD63AA01Y0CPJ, HD63B01Y0CPJ, HD63C01Y0CPJ

HD6301Y0PJ, HD63A01Y0PJ,  
HD63B01Y0PJ, HD63C01Y0PJ



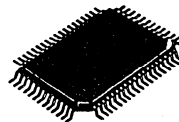
(DP-64S)

HD6301Y0CPJ, HD63A01Y0CPJ,  
HD63B01Y0CPJ, HD63C01Y0CPJ



(CP-68)

HD6301Y0F\*, HD63A01Y0F\*,  
HD63B01Y0F\*, HD63C01Y0F\*

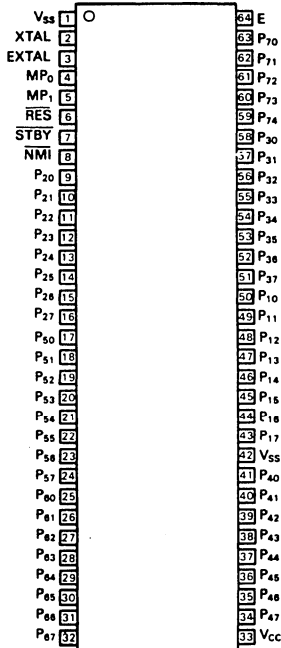


(FP-64)

\*Contact Hitachi Sales Office

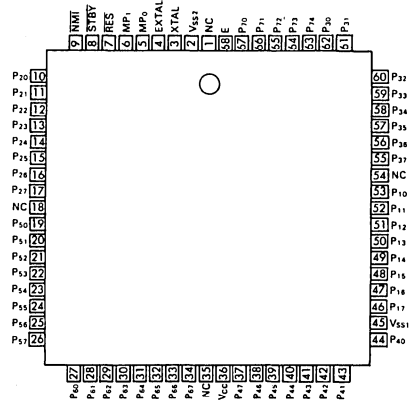
■ PIN ARRANGEMENT

- HD6301Y0PJ, HD63A01Y0PJ, HD63B01Y0PJ, HD63C01Y0PJ



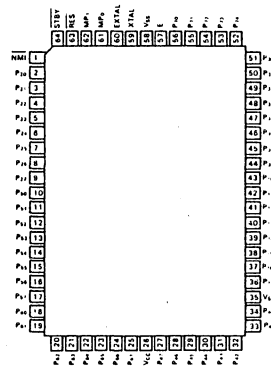
(Top View)  
(DP-64S)

- HD6301Y0CPJ, HD63A01Y0CPJ, HD63B01Y0CPJ, HD6301Y0CPJ



(Top View)  
(CP-68)

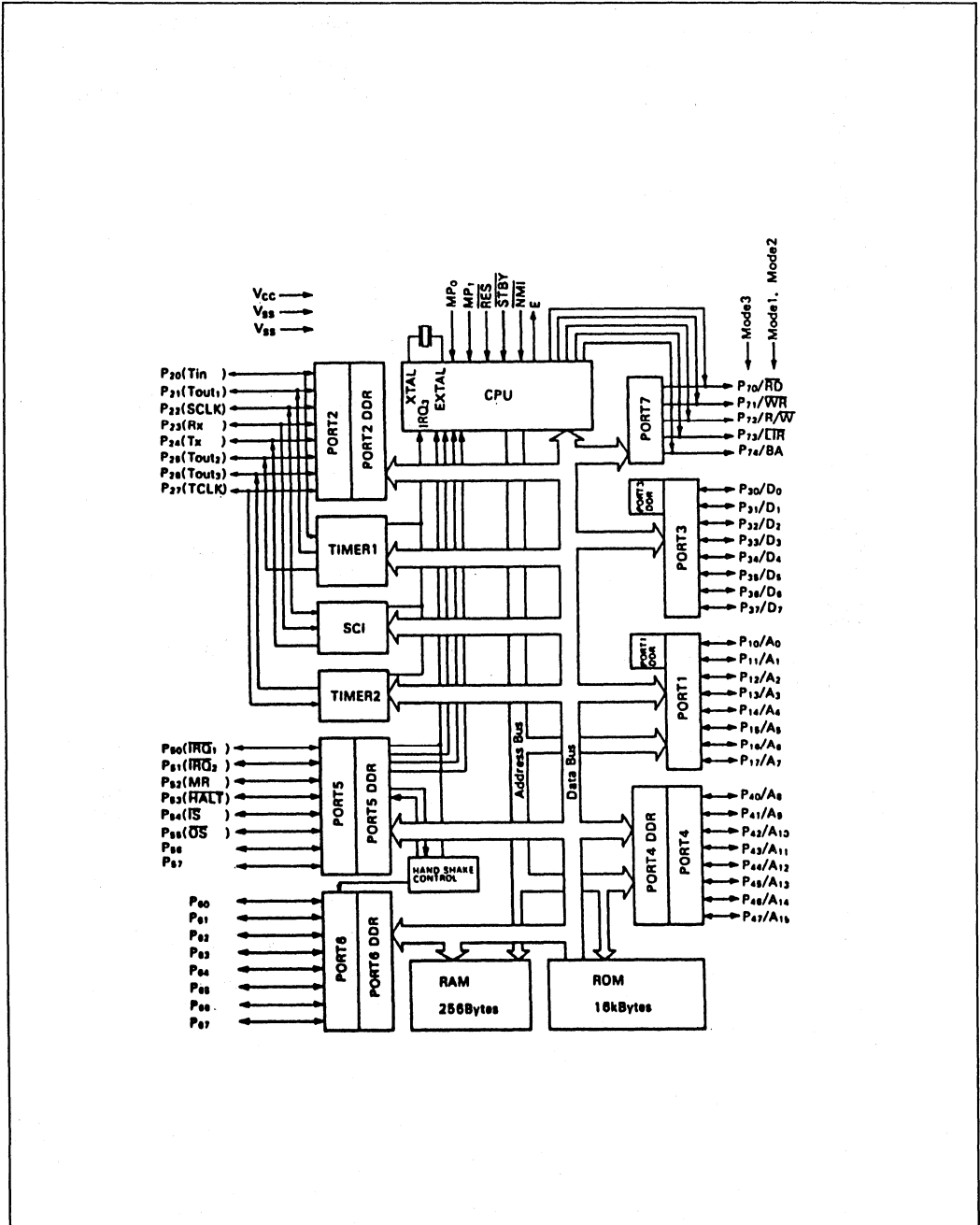
- HD6301Y0F\*, HD63A01Y0F\*, HD63B01Y0F\*, HD63C01Y0F\*



(Top View)  
(FP-64)

\*Contact Hitachi Sales Office

■ **BLOCK DIAGRAM**



**Electrical Characteristics HD6301Y0, HD63A01Y0, HD63B01Y0, and HD63C01Y0**  
**Absolute Maximum Ratings**

Item	Symbol	Value	Unit
Supply voltage	V <sub>CC</sub>	-0.3 to +7.0	V
Input voltage	V <sub>in</sub>	-0.3 to V <sub>CC</sub> +0.3	V
Operating temperature	T <sub>opr</sub>	-40 to +85	°C
Storage temperature	T <sub>stg</sub>	-55 to +150	°C

Note: This product has protection circuits in input terminal from high static electricity voltage and high electric field. But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend V<sub>in</sub>, V<sub>out</sub>: V<sub>SS</sub> ≦ (V<sub>in</sub> or V<sub>out</sub>) ≦ V<sub>CC</sub>.

**Electrical Characteristics**

**DC Characteristics**

(V<sub>CC</sub> = 5.0 V ± 10%, f = 0.1 to 3.0 MHz, V<sub>SS</sub> = 0V, Ta = -40 to +85°C, unless otherwise noted.)

Item		Symbol	Min	Typ	Max	Unit	Test Condition
Input high voltage	RES, STBY,	V <sub>IH</sub>	V <sub>CC</sub> -0.5		V <sub>CC</sub> +0.3	V	
	EXTAL		V <sub>CC</sub> ×0.7		V <sub>CC</sub> +0.3	V	
	Other inputs		2.1		V <sub>CC</sub> +0.3	V	
Input low voltage	All other inputs	V <sub>IL</sub>	-0.3		0.8/0.6 <sup>3</sup>	V	
Input leakage current	RES, NMI, STBY, MP0, MP1	I <sub>in</sub>			1.0	μA	V <sub>in</sub> =0.5 to V <sub>CC</sub> -0.5 V
Three state leakage current	A0-A15, D0-D7, RD, WR, R/W, Ports 2,5,6	I <sub>TSI</sub>			1.0	μA	V <sub>in</sub> =0.5 to V <sub>CC</sub> -0.5 V
Output high voltage	All Outputs	V <sub>OH</sub>	2.4			V	I <sub>OH</sub> =-200 μA
			V <sub>CC</sub> -0.7			V	I <sub>OH</sub> =-10 μA
Output low voltage	All Outputs	V <sub>OL</sub>			0.4	V	I <sub>OL</sub> =1.6 mA
Darlington drive current	Ports 2, 6	-I <sub>OH</sub>	1.0		10.0	mA	V <sub>out</sub> =1.5 V
Input capacitance	All other inputs	C <sub>in</sub>			12.5	pF	V <sub>in</sub> =0 V, f=1 MHz, Ta=25°C
Standby current	Not operating	I <sub>STB</sub>		3.0	15.0	μA	
Current dissipation <sup>1</sup>		I <sub>SLP</sub>		1.5	3.0	mA	Sleeping (f=1 MHz <sup>2</sup> )
				2.3	4.5	mA	Sleeping (f=1.5 MHz <sup>2</sup> )
				3.0	6.0	mA	Sleeping (f=2 MHz <sup>2</sup> )
				4.5	9.0	mA	Sleeping (f=3 MHz <sup>2</sup> )
		I <sub>CC</sub>		7.0	10.0	mA	Operating (f=1 MHz <sup>2</sup> )
				10.5	15.0	mA	Operating (f=1.5 MHz <sup>2</sup> )
				14.0	20.0	mA	Operating (f=2 MHz <sup>2</sup> )
				21.0	30.0	mA	Operating (f=3 MHz <sup>2</sup> )
RAM standby voltage		V <sub>RAM</sub>	2.0			V	

Notes:

- V<sub>IH</sub> min=V<sub>c</sub>-1.0V, V<sub>IH</sub> max=0.8V (All output terminals are at no load.)
- Current dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ. or max. values about current dissipations at x MHz operation are decided according to the following formula:  
 typ. value (f=x MHz) =typ. value (f=1 MHz) × x  
 max. value (f=x MHz) =max. value (f=1 MHz) × x  
 (both the sleeping and operating)
- In case of SCLK input, V<sub>IL</sub> = 0.6V (-20°C ~ 0°C)

## AC Characteristics

( $V_{CC} = 5.0V \pm 10\%$ ,  $f = 0.1$  to  $3.0$  MHz,  $V_{SS} = 0$  V,  $T_a = -40$  to  $+85^\circ\text{C}$ , unless otherwise noted)

### Bus Timing

Item	Symbol	HD6301Y0			HD63A01Y0			HD63B01Y0			HD63C01Y0			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Cycle time	$t_{CYC}$	1		10	0.666		10	0.5		10	0.333		10	$\mu\text{s}$	Fig. I-1*
Enable rise time	$t_{ER}$			25			25			25			20	ns	
Enable fall time	$t_{EF}$			25			25			25			20	ns	
Enable pulse width high level <sup>1</sup>	$PW_{EH}$	450			300			220			140			ns	
Enable pulse width low level <sup>1</sup>	$PW_{EL}$	450			300			220			140			ns	
Address, R/ $\bar{W}$ delay time <sup>1</sup>	$t_{AD}$			250			190			160			120	ns	
Data delay time (Write)	$t_{DDW}$			200			160			120			100	ns	
Data set-up time (Read)	$t_{DSR}$	80			70			60			50			ns	
Address, R/ $\bar{W}$ hold time <sup>1</sup>	$t_{AH}$	80			50			40			20			ns	
Data hold time (Write) <sup>1</sup>	$t_{HW}$	80			50			40			20			ns	
Data hold time (Read)	$t_{HR}$	0			0			0			0			ns	
$\overline{RD}$ , $\overline{WR}$ pulse width <sup>1</sup>	$PW_{RW}$	450			300			220			140			ns	
$\overline{RD}$ , $\overline{WR}$ delay time	$t_{RWD}$			40			40			40			40	ns	
$\overline{RD}$ , $\overline{WR}$ hold time	$t_{HRW}$			20			20			20			20	ns	
$\overline{LIR}$ delay time	$t_{DLR}$			200			160			120			80	ns	
$\overline{LIR}$ hold time	$t_{HLR}$	5			5			5			5			ns	
Peripheral read access time <sup>1</sup>	$t_{ACC}$										180			ns	
MR set-up time <sup>1</sup>	$t_{SMR}$	400			280			230			170			ns	Fig. I-2*
MR hold time <sup>1</sup>	$t_{HMR}$			100			70			50			25	ns	
E clock pulse width at MR	$PW_{EMR}$			9			9			9			9	$\mu\text{s}$	
Processor control <sup>1</sup> set-up time	$t_{PCS}$	200			200			200			100			ns	Figs. I-3, I-13, I-14*
Processor control rise time	$t_{PCR}$			100			100			100			50	ns	Figs. I-2, I-13*
Processor control fall time	$t_{PCF}$			100			100			100			50	ns	
BA delay time	$t_{BA}$			250			190			160			120	ns	Fig. I-3*
Oscillator stabilization time	$t_{RC}$	20			20			20			20			ms	Fig. I-14*
Reset pulse width	$PW_{RST}$	3			3			3			3			$t_{CYC}$	

Note 1: These timings change in approximate proportion to  $t_{CYC}$ . The figures in this characteristics represent those when  $t_{CYC}$  is minimum (= in the highest speed operation).

\*Refer to Pages 611-614

**Peripheral Port Timing**

Item	Symbol	HD6301Y0			HD63A01Y0			HD63B01Y0			HD63C01Y0			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Peripheral data set-up time (Ports 1, 2, 3, 4, 5, 6)	tPDSU	200			200			200			200			ns	Fig. 1-5*
Peripheral data hold time (Ports 1, 2, 3, 4, 5, 6)	tPDH	200			200			200			200			ns	
Delay time (From enable fall edge peripheral output) (Ports 1, 2, 3, 4, 5, 6, 7)	tpWD			300			300			300			300	ns	Fig. 1-6*
Input strobe pulse width	tpWIS	200			200			200			200			ns	Fig. 1-10*
Input data hold time (Port 6)	tIH	150			150			150			150			ns	
Input data set-up time (Port 6)	tIS	100			100			100			100			ns	
Output strobe time	tOSD1			200			200			200			200	ns	Fig. 1-11*
	tOSD2														

\*Refer to Pages 611-614

**Timer, SCI Timing**

Item	Symbol	HD6301Y0			HD63A01Y0			HD63B01Y0			HD63C01Y0			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Timer 1 input pulse width	tpWT	2.0		2.0			2.0			2.0			t <sub>cyc</sub>	Fig. 1-9*	
Delay time (enable positive transition to timer output)	tTOD			400			400			400			400	ns	Fig. 1-7, 1-8*
SCI input clock cycle	(Async. mode) tScyc	1.0		1.0			1.0			1.0			t <sub>cyc</sub>	Fig. 1-9*	
	(Clock sync.)	2.0		2.0			2.0			2.0			t <sub>cyc</sub>	Fig. 1-4*	
SCI transmit data delay time (Clock sync. mode)	tTXD			240			240			240			240	ns	Fig. 1-4*
SCI receive data set-up time (Clock sync. mode)	tSRX	260			260			260			260			ns	
SCI receive data hold time (Clock sync. mode)	tHRX	100			100			100			100			ns	
SCI input clock pulse width	tpWSCK	0.4		0.6	0.4		0.6	0.4		0.6	0.4		0.6	t <sub>Scyc</sub>	Fig. 1-9*
Timer 2 input clock cycle	t <sub>cyc</sub>	2.0			2.0			2.0			2.0			t <sub>cyc</sub>	
Timer 2 input clock pulse width	tpWTCK	200			200			200			200			ns	
Timer 1*2, SCI input clock rise time	tCKr			100			100			100			50	ns	
Timer 1*2, SCI input clock fall time	tCKf			100			100			100			50	ns	

\*Refer to Pages 611-614

**WIDE TEMPERATURE  
SPECIFICATIONS**  
-40°C TO +85°C  
(J VERSION)

# HD6303R, HD63A03R, HD63B03R

The HD6303R is an 8-bit CMOS micro processing unit which has the completely compatible instruction set with the HD6301V1. 128 bytes RAM, Serial Communication Interface (SCI), parallel I/O ports and multi function timer are incorporated in the HD6303R. It is bus compatible with HMCS6800 and can be expanded up to 65k words. Like the HMCS6800 family, I/O levels is TTL compatible with +5.0V single power supply. As the HD6303R is CMOS MPU, power dissipation is extremely low. And also HD6303R has Sleep Mode and Stand-by Mode as lower power dissipation mode. Therefore, flexible low power consumption application is possible.

## ■ FEATURES

- Object Code Upward Compatible with the HD6800, HD6801, HD6802
- Multiplexed Bus ( $D_0 \sim D_7/A_0 \sim A_7$ ), Non Multiplexed Bus
- Abundant On-Chip Functions Compatible with the HD6301V1; 128 Bytes RAM, 13 Parallel I/O Lines, 16-bit Timer, Serial Communication Interface (SCI)
- Low Power Consumption Mode; Sleep Mode, Stand-By Mode
- Minimum Instruction Execution Time  
1 $\mu$ s (f=1MHz), 0.67 $\mu$ s (f=1.5MHz), 0.5 $\mu$ s (f=2.0MHz)
- Bit Manipulation, Bit Test Instruction
- Error Detecting Function; Address Trap, Op Code Trap
- Up to 65k Words Address Space

## ■ TYPE OF PRODUCTS

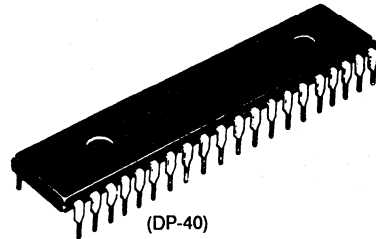
Type No.	Bus Timing
HD6303R	1.0 MHz
HD63A03R	1.5 MHz
HD63B03R	2.0 MHz

## ■ GENERIC PART NUMBER

HD6303RPJ, HD63A03RPJ, HD63B03RPJ

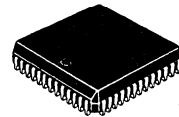
HD6303RCPJ, HD63A03RCPJ, HD63B03RCPJ

HD6303RPJ, HD63A03RPJ,  
HD63B03RPJ



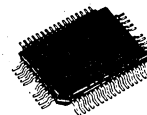
(DP-40)

HD6303RCPJ, HD63A03RCPJ,  
HD63B03RCPJ



(CP-52)

HD6303RF\*, HD63A03RF\*,  
HD63B03RF\*



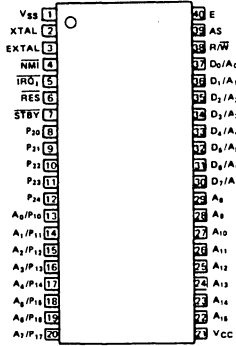
(FP-54)

\*Contact Hitachi Sales Office

**WIDE TEMPERATURE SPECIFICATIONS**  
**-40°C TO +85°C**  
**(J VERSION)**

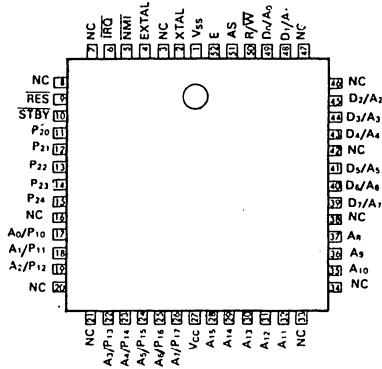
■ PIN ARRANGEMENT

- HD6303RPJ, HD63A03RPJ, HD63B03RPJ



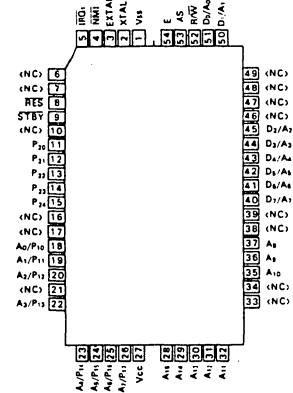
(Top View)  
(DP-40)

- HD6303RCPJ, HD63A03RCPJ, HD63B03RCPJ



(Top View)  
(CP-52)

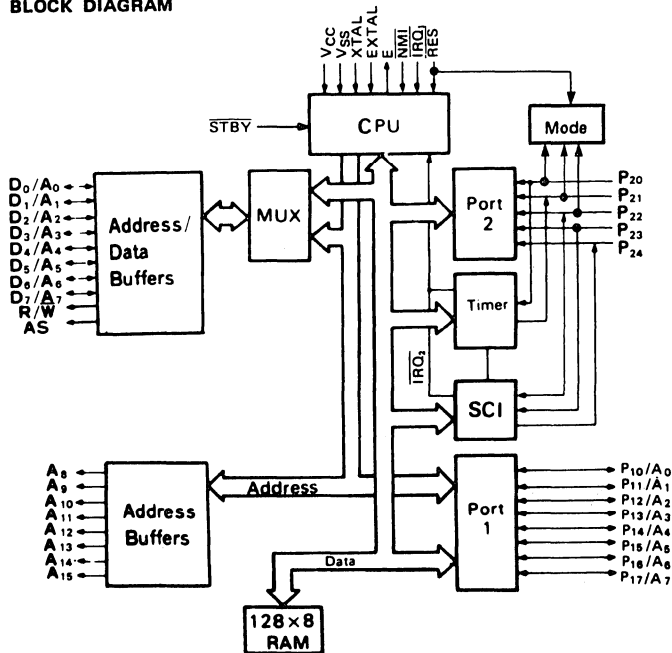
- HD6303RF\*, HD63A03RF\*, HD63B03RF\*



(Top View)  
(FP-54)

\*Contact Hitachi Sales Office

■ BLOCK DIAGRAM





▪ **ABSOLUTE MAXIMUM RATINGS**

Item	Symbol	Value	Unit
Supply Voltage	$V_{CC}$	-0.3 ~ +7.0	V
Input Voltage	$V_{in}$	-0.3 ~ $V_{CC} + 0.3$	V
Operating Temperature*	$T_{opr}$	-40 to +85	°C
Storage Temperature	$T_{stg}$	-55 to +150	°C

(NOTE) This product has protection circuits in input terminal from high static electricity voltage and high electric field. But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend  $V_{in}$ ,  $V_{out}$ ,  $V_{SS} \leq (V_{in} \text{ or } V_{out}) \leq V_{CC}$ .

\*K version (-40 to +125°C) available. Contact sales office.

• **DC CHARACTERISTICS ( $V_{CC} = 5.0V \pm 10\%$ ,  $V_{SS} = 0V$ ,  $T_a = -40$  to  $+85^\circ C$ , unless otherwise noted.)**

Item	Symbol	Test Condition	min	typ	max	Unit		
Input "High" Voltage	RES, STBY	$V_{IH}$		$V_{CC}-0.5$	-	$V_{CC} + 0.3$	V	
	EXTAL			$V_{CC} \times 0.7$	-			
	Other Inputs			2.0	-			
Input "Low" Voltage	All Inputs	$V_{IL}$		-0.3	-	0.8	V	
Input Leakage Current	$\overline{NMI}$ , $\overline{IRQ_1}$ , $\overline{RES}$ , $\overline{STBY}$	$I_{in}$	$V_{in} = 0.5 \sim V_{CC} - 0.5V$	-	-	1.0	$\mu A$	
Three State (off-state) Leakage Current	$P_{10} \sim P_{17}$ , $P_{20} \sim P_{24}$ , $D_0 \sim D_7$ , $A_8 \sim A_{15}$	$ I_{TSI} $	$V_{in} = 0.5 \sim V_{CC} - 0.5V$	-	-	1.0	$\mu A$	
Output "High" Voltage	All Outputs	$V_{OH}$		$I_{OH} = -200\mu A$	2.4	-	V	
				$I_{OH} = -10\mu A$	$V_{CC}-0.7$	-	V	
Output "Low" Voltage	All Outputs	$V_{OL}$	$I_{OL} = 1.6mA$	-	-	0.55	V	
Input Capacitance	All Inputs	$C_{in}$	$V_{in} = 0V$ , $f = 1.0MHz$ , $T_a = 25^\circ C$	-	-	12.5	pF	
Standby Current	Non Operation	$I_{CC}$	$V_{IL}(STBY) = 0 \sim 0.6V$ $V_{IH}(RES) = V_{CC} - 0.5 \sim V_{CC} V$ $V_{IL}(RES) = 0 \sim 0.6V$	-	2.0	15.0	$\mu A$	
Current Dissipation*		$I_{CC}$		Operating ( $f=1MHz^{**}$ )	-	6.0	10.0	mA
				Sleeping ( $f=1MHz^{**}$ )	-	1.0	2.0	
RAM Stand-By Voltage		$V_{RAM}$		2.0	-	-	V	

\*  $V_{IH} \text{ min} = V_{CC} - 1.0V$ ,  $V_{IL} \text{ max} = 0.8V$

\*\* Current Dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ. or max. values about Current Dissipations at  $f = x \text{ MHz}$  operation are decided according to the following formula;

typ. value ( $f = x \text{ MHz}$ ) = typ. value ( $f = 1 \text{ MHz}$ )  $x x$   
 max. value ( $f = x \text{ MHz}$ ) = max. value ( $f = 1 \text{ MHz}$ )  $x x$   
 (both the sleeping and operating)

● AC CHARACTERISTICS (V<sub>CC</sub> = 5.0V±10%, V<sub>SS</sub> = 0V, T<sub>a</sub> = -40 to +85°C, unless otherwise noted.)

BUS TIMING

Item	Symbol	Test Condition	HD6303R			HD63A03R			HD63B03R			Unit
			min	typ	max	min	typ	max	min	typ	max	
Cycle Time	t <sub>cyc</sub>		1	-	10	0.666	-	10	0.5	-	10	μs
Address Strobe Pulse Width "High"	PW <sub>ASH</sub>		220	-	-	150	-	-	110	-	-	ns
Address Strobe Rise Time	t <sub>ASr</sub>		-	-	20	-	-	20	-	-	20	ns
Address Strobe Fall Time	t <sub>ASf</sub>		-	-	20	-	-	20	-	-	20	ns
Address Strobe Delay Time	t <sub>ASD</sub>		60	-	-	40	-	-	20	-	-	ns
Enable Rise Time	t <sub>Er</sub>		-	-	20	-	-	20	-	-	20	ns
Enable Fall Time	t <sub>Ef</sub>		-	-	20	-	-	20	-	-	20	ns
Enable Pulse Width "High" Level	PW <sub>EH</sub>		450	-	-	300	-	-	220	-	-	ns
Enable Pulse Width "Low" Level	PW <sub>EL</sub>		450	-	-	300	-	-	220	-	-	ns
Address Strobe to Enable Delay Time	t <sub>ASED</sub>		60	-	-	40	-	-	20	-	-	ns
Address Delay Time	t <sub>AD1</sub>	Fig. 5-1**	-	-	250	-	-	190	-	-	160	ns
	t <sub>AD2</sub>		-	-	250	-	-	190	-	-	160	ns
Address Delay Time for Latch	t <sub>ADL</sub>	Fig. 5-2**	-	-	250	-	-	190	-	-	160	ns
Data Set-up Time	Write	t <sub>DSW</sub>	230	-	-	150	-	-	100	-	-	ns
	Read	t <sub>DSR</sub>	80	-	-	60	-	-	50	-	-	ns
Data Hold Time	Read	t <sub>HR</sub>	0	-	-	0	-	-	0	-	-	ns
	Write	t <sub>HW</sub>	20	-	-	20	-	-	20	-	-	ns
Address Set-up Time for Latch	t <sub>ASL</sub>		60	-	-	40	-	-	20	-	-	ns
Address Hold Time for Latch	t <sub>AHL</sub>		30	-	-	20	-	-	20	-	-	ns
Address Hold Time	t <sub>AH</sub>		20	-	-	20	-	-	20	-	-	ns
A <sub>0</sub> ~ A <sub>7</sub> Set-up Time Before E	t <sub>ASM</sub>		200	-	-	110	-	-	60	-	-	ns
Peripheral Read Access Time	Non-Multiplexed Bus	(t <sub>ACCN</sub> )	-	-	650	-	-	395	-	-	270	ns
	Multiplexed Bus	(t <sub>ACCM</sub> )	-	-	650	-	-	395	-	-	270	ns
Oscillator stabilization Time	t <sub>RC</sub>	Fig. 2-7-1	20	-	-	20	-	-	20	-	-	ms
Processor Control Set-up Time	t <sub>PCS</sub>	Fig. 2-8-1	200	-	-	200	-	-	200	-	-	ns

PERIPHERAL PORT TIMING

Item	Symbol	Test Condition	HD6303R			HD63A03R			HD63B03R			Unit	
			min	typ	max	min	typ	max	min	typ	max		
Peripheral Data Set-up Time	Port 1, 2	t <sub>PDSU</sub>	Fig. 5-3**	200	-	-	200	-	-	200	-	-	ns
Peripheral Data Hold Time	Port 1, 2	t <sub>PDH</sub>	Fig. 5-3**	200	-	-	200	-	-	200	-	-	ns
Delay Time, Enable Negative Transition to Peripheral Data Valid	Port 1, 2*	t <sub>PWD</sub>	Fig. 5-5**	-	-	300	-	-	300	-	-	300	ns

\* Except P<sub>21</sub>

\*\* Refer to Pages 189-190

\*\*\* Refer to Pages 159-160

**TIMER, SCI TIMING**

Item	Symbol	Test Condition	HD6303R			HD63A03R			HD63B03R			Unit
			min	typ	max	min	typ	max	min	typ	max	
Timer Input Pulse Width	t <sub>PWT</sub>		2.0	-	-	2.0	-	-	2.0	-	-	t <sub>cytc</sub>
Delay Time, Enable Positive Transition to Timer Out	t <sub>TOD</sub>	Fig. 5-7	-	-	400	-	-	400	-	-	400	ns
SCI Input Clock Cycle	t <sub>Scyc</sub>		2.0	-	-	2.0	-	-	2.0	-	-	t <sub>cytc</sub>
SCI Input Clock Pulse Width	t <sub>PWSCK</sub>		0.4	-	0.6	0.4	-	0.6	0.4	-	0.6	t <sub>Scyc</sub>

**MODE PROGRAMMING**

Item	Symbol	Test Condition	HD6303R			HD63A03R			HD63B03R			Unit
			min	typ	max	min	typ	max	min	typ	max	
RES "Low" Pulse Width	PW <sub>RSTL</sub>		3	-	-	3	-	-	3	-	-	t <sub>cytc</sub>
Mode Programming Set-up Time	t <sub>MPS</sub>	Fig. 5-8	2	-	-	2	-	-	2	-	-	t <sub>cytc</sub>
Mode Programming Hold Time	t <sub>MPH</sub>		150	-	-	150	-	-	150	-	-	ns

\*\*Refer to Pages 189-190

# HD6303X, HD63A03X, HD63B03X

**WIDE TEMPERATURE  
SPECIFICATIONS**  
-40°C TO +85°C  
(J VERSION)

The HD6303X is a CMOS 8-bit micro processing unit (MPU) which includes a CPU compatible with the HD6301V1, 192 bytes of RAM, 24 parallel I/O pins, a Serial Communication Interface (SCI) and two timers on chip.

## ■ FEATURES

- Instruction Set Compatible with the HD6301V1
- Abundant On-chip Functions
  - 192 Bytes of RAM
  - 24 Parallel I/O Ports
  - 16-Bit Programmable Timer
  - 8-Bit Reloadable Timer
  - Serial Communication Interface
  - Memory Ready
  - Halt
  - Error-Detection (Address Trap, Op Code Trap)
- Interrupts . . . 3 External, 7 Internal
- Up to 65k Bytes Address Space
- Low Power Dissipation Mode
  - Sleep
  - Standby
- Minimum Instruction Execution Time
  - 1  $\mu$ s (f = 1 MHz), 0.67  $\mu$ s (f = 1.5 MHz), 0.5  $\mu$ s (f = 2.0 MHz)
- Wide Operating Range
  - V<sub>CC</sub> = 3 ~ 6V (f = 0.1 ~ 0.5MHz).
  - f = 0.1 to 2.0 MHz (V<sub>CC</sub> = 5V  $\pm$  10%)

## ■ TYPE OF PRODUCTS

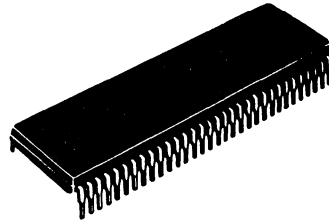
Type No.	Bus Timing
HD6303X	1 MHz
HD63A03X	1.5 MHz
HD63B03X	2 MHz

## ■ GENERIC PART NUMBER

HD6303XPJ, HD63A03XPJ, HD63B03XPJ

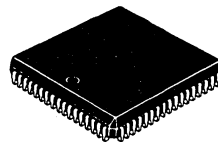
HD6303XCPJ, HD63A03XCPJ, HD63B03XCPJ

HD6303XPJ, HD63A03XPJ,  
HD63B03XPJ



(DP-64S)

HD6303XCPJ, HD63A03XCPJ,  
HD63B03XCPJ



(CP-68)

HD6303XF\*, HD63A03XF\*,  
HD63B03XF\*



(FP-80)

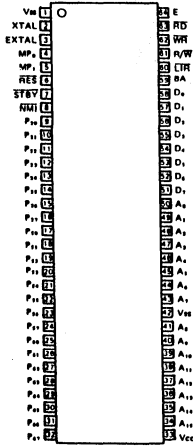
\*Contact Hitachi Sales Office

**WIDE TEMPERATURE  
SPECIFICATIONS  
-40°C TO +85°C  
(J VERSION)**

**HD6303X, HD63A03X, HD63B03X**

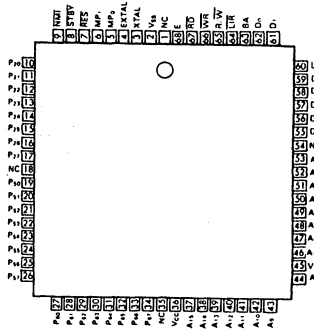
■ PIN ARRANGEMENT

- HD6303XPJ, HD63A03XPJ, HD63B03XPJ



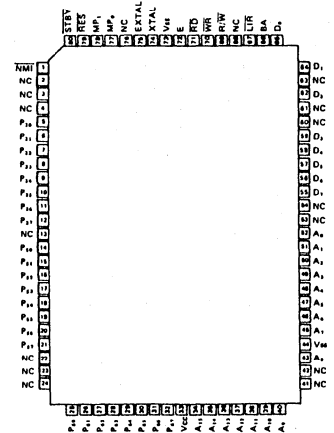
(Top View)  
(DP-64S)

- HD6303XCPJ, HD63A03XCPJ, HD63B03XCPJ



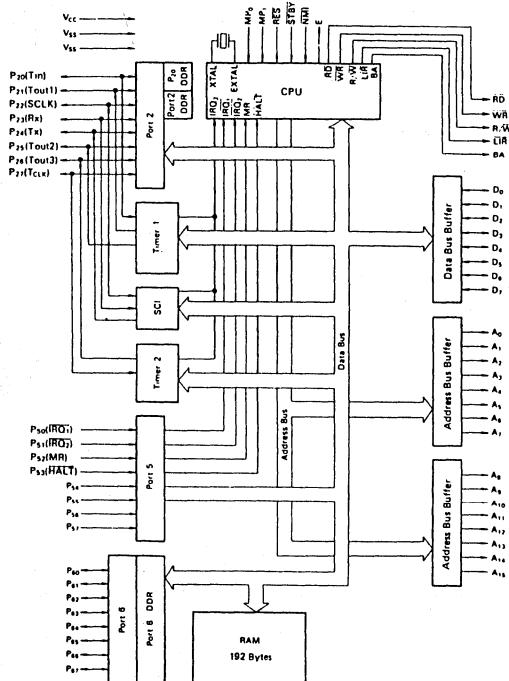
(Top View)  
(CP-68)

- HD6303XF\*, HD63A03XF\*, HD63B03XF\*



(Top View)  
(FP-80)

■ BLOCK DIAGRAM



■ **ABSOLUTE MAXIMUM RATINGS**

Item	Symbol	Value	Unit
Supply Voltage	$V_{CC}$	-0.3 ~ +7.0	V
Input Voltage	$V_{in}$	-0.3 ~ $V_{CC} + 0.3$	V
Operating Temperature	$T_{opr}$	-40 to +85	°C
Storage Temperature	$T_{stg}$	-55 to +150	°C

(NOTE) This product has protection circuits in input terminal from high static electricity voltage and high electric field. But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend  $V_{in}, V_{out}, V_{SS} \leq (V_{in} \text{ or } V_{out}) \leq V_{CC}$ .

■ **ELECTRICAL CHARACTERISTICS**

● **DC CHARACTERISTICS** ( $V_{CC} = 5.0V \pm 10\%$ ,  $V_{SS} = 0V$ ,  $T_a = -40 \text{ to } +85^\circ\text{C}$ , unless otherwise noted.)

Item	Symbol	Test Condition	min	typ	max	Unit	
Input "High" Voltage	RES, STBY	$V_{IH}$	$V_{CC}-0.5$	-	$V_{CC} + 0.3$	V	
	EXTAL		$V_{CC} \times 0.7$	-			
	Other Inputs		2.0	-			
Port 22	Pin 22	$V_{IH}$	2.2	-	-	V	
Input "Low" Voltage	All Inputs	$V_{IL}$	-0.3	-	0.8	V	
Input "Low" Voltage	All Inputs	$V_{IL}$	-0.3	-	0.8	V	
Input Leakage Current	NMI, RES, STBY, MP <sub>0</sub> , MP <sub>1</sub> , Port 5	$I_{IH}$	$V_{in} = 0.5 \sim V_{CC}-0.5V$	-	-	1.0	μA
Three State (off-state) Leakage Current	A <sub>0</sub> ~A <sub>15</sub> , D <sub>0</sub> ~D <sub>7</sub> , RD, WR, R/W, Port 2, Port 6	$I_{TSIL}$	$V_{in} = 0.5 \sim V_{CC}-0.5V$	-	-	1.0	μA
Output "High" Voltage	All Outputs	$V_{OH}$	$I_{OH} = -200\mu A$	2.4	-	-	V
			$I_{OH} = -10\mu A$	$V_{CC}-0.7$	-	-	V
Output "Low" Voltage	All Outputs	$V_{OL}$	$I_{OL} = 1.6mA$	-	-	0.4	V
Darlington Drive Current	Ports 2, 6	$-I_{OH}$	$V_{out} = 1.5V$	1.0	-	10.0	mA
Input Capacitance	All Inputs	$C_{in}$	$V_{in} = 0V, f = 1MHz, T_a = 25^\circ C$	-	-	12.5	pF
Standby Current	Non Operation	$I_{STB}$		-	3.0	15.0	μA
Current Dissipation*	$I_{SLP}$		Sleeping (f = 1MHz**)	-	1.5	3.0	mA
			Sleeping (f = 1.5MHz**)	-	2.3	4.5	mA
			Sleeping (f = 2MHz**)	-	3.0	6.0	mA
	$I_{CC}$		Operating (f = 1MHz**)	-	7.0	10.0	mA
			Operating (f = 1.5MHz**)	-	10.5	15.0	mA
			Operating (f = 2MHz**)	-	14.0	20.0	mA
RAM Standby Voltage		$V_{RAM}$		2.0	-	-	V

\*  $V_{IH} \text{ min} = V_{CC}-1.0V, V_{IL} \text{ max} = 0.8V$ , All output terminals are at no load.

\*\* Current Dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ. or max. values about Current Dissipations at x MHz operation are decided according to the following formula;

typ. value (f = x MHz) = typ. value (f = 1MHz) x x

max. value (f = x MHz) = max. value (f = 1MHz) x x

(both the sleeping and operating)

- AC CHARACTERISTICS ( $V_{CC} = 5.0V \pm 10\%$ ,  $V_{SS} = 0V$ ,  $T_a = -40$  to  $+85^\circ C$ , unless otherwise noted.)

**BUS TIMING**

Item	Symbol	Test Condition	HD6303X			HD63A03X			HD63B03X			Unit
			min	typ	max	min	typ	max	min	typ	max	
Cycle Time	$t_{cyc}$	Fig. I-13**	1	—	10	0.666	—	10	0.5	—	10	$\mu s$
Enable Rise Time	$t_{ER}$		—	—	25	—	—	25	—	—	25	ns
Enable Fall Time	$t_{EF}$		—	—	25	—	—	25	—	—	25	ns
Enable Pulse Width "High" Level*	$PW_{EH}$		450	—	—	300	—	—	220	—	—	ns
Enable Pulse Width "Low" Level*	$PW_{EL}$		450	—	—	300	—	—	220	—	—	ns
Address, R/W Delay Time*	$t_{AD}$		—	—	250	—	—	190	—	—	160	ns
Data Delay Time	Write $t_{DDW}$		—	—	200	—	—	160	—	—	120	ns
Data Set-up Time	Read $t_{DSR}$		80	—	—	70	—	—	70	—	—	ns
Address, R/W Hold Time*	$t_{AH}$		80	—	—	50	—	—	35	—	—	ns
Data Hold Time	Write* $t_{HW}$		80	—	—	50	—	—	40	—	—	ns
	Read $t_{HR}$		0	—	—	0	—	—	0	—	—	ns
RD, WR Pulse Width*	$PW_{RW}$		450	—	—	300	—	—	220	—	—	ns
RD, WR Delay Time	$t_{RWD}$		—	—	40	—	—	40	—	—	40	ns
RD, WR Hold Time	$t_{HRW}$		—	—	30	—	—	30	—	—	25	ns
LIR Delay Time	$t_{DLR}$		—	—	200	—	—	160	—	—	120	ns
LIR Hold Time	$t_{HLR}$		10	—	—	10	—	—	10	—	—	ns
MR Set-up Time*	$t_{SMR}$		Fig. I-14**	400	—	—	280	—	—	230	—	—
MR Hold Time*	$t_{HMR}$	—		—	90	—	—	40	—	—	0	ns
E Clock Pulse Width at MR	$PW_{EMR}$	—		—	9	—	—	9	—	—	9	$\mu s$
Processor Control Set-up Time	$t_{PCS}$	Fig. I-15** I-20, I-24	200	—	—	200	—	—	200	—	—	ns
Processor Control Rise Time	$t_{PCr}$	Fig. I-14, I-15**	—	—	100	—	—	100	—	—	100	ns
Processor Control Fall Time	$t_{PCf}$		—	—	100	—	—	100	—	—	100	ns
BA Delay Time	$t_{BA}$	Fig. I-15**	—	—	250	—	—	190	—	—	160	ns
Oscillator Stabilization Time	$t_{RC}$	Fig. I-24**	20	—	—	20	—	—	20	—	—	ms
Reset Pulse Width	$PW_{RST}$		3	—	—	3	—	—	3	—	—	$t_{cyc}$

\* These timings change in approximate proportion to  $t_{cyc}$ . The figures in this characteristics represent those when  $t_{cyc}$  is minimum (= in the highest speed operation).

\*\* Refer to Pages 472-476

**PERIPHERAL PORT TIMING**

Item	Symbol	Test Condition	HD6303X			HD63A03X			HD63B03X			Unit	
			min	typ	max	min	typ	max	min	typ	max		
Peripheral Data Set-up Time	Ports 2, 5, 6	$t_{PDSU}$	Fig. I-17**	200	—	—	200	—	—	200	—	—	ns
Peripheral Data Hold Time	Ports 2, 5, 6	$t_{PDH}$	Fig. I-17**	200	—	—	200	—	—	200	—	—	ns
Delay Time (Enable Negative Transition to Peripheral Data Valid)	Ports 2, 6	$t_{PWD}$	Fig. I-18**	—	—	300	—	—	300	—	—	300	ns

\*\* Refer to Pages 472-476

TIMER, SCI TIMING

Item	Symbol	Test Condition	HD6303X			HD63A03X			HD63B03X			Unit
			min	typ	max	min	typ	max	min	typ	max	
Timer 1 Input Pulse Width	t <sub>PWT</sub>	Fig. I-21**	2.0	-	-	2.0	-	-	2.0	-	-	t <sub>cyc</sub>
Delay Time (Enable Positive Transition to Timer Output)	t <sub>TOD</sub>	Fig. I-19** I-20	-	-	400	-	-	400	-	-	400	ns
SCI Input Clock Cycle	Async. Mode	Fig. I-21**	1.0	-	-	1.0	-	-	1.0	-	-	t <sub>cyc</sub>
	Clock Sync.		Fig. I-16, I-21**	2.0	-	-	2.0	-	-	2.0	-	-
SCI Transmit Data Delay Time (Clock Sync. Mode)	t <sub>TXD</sub>	Fig. I-16**	-	-	200	-	-	200	-	-	200	ns
SCI Receive Data Set-up Time (Clock Sync. Mode)	t <sub>SRX</sub>		290	-	-	290	-	-	290	-	-	ns
SCI Receive Data Hold Time (Clock Sync. Mode)	t <sub>HRX</sub>		100	-	-	100	-	-	100	-	-	ns
SCI Input Clock Pulse Width	t <sub>PWSCK</sub>	Fig. I-21**	0.4	-	0.6	0.4	-	0.6	0.4	-	0.6	t <sub>Scyc</sub>
Timer 2 Input Clock Cycle	t <sub>tcyc</sub>		2.0	-	-	2.0	-	-	2.0	-	-	t <sub>cyc</sub>
Timer 2 Input Clock Pulse Width	t <sub>PWTCK</sub>		200	-	-	200	-	-	200	-	-	ns
Timer 1-2, SCI Input Clock Rise Time	t <sub>ckr</sub>		-	-	100	-	-	100	-	-	100	ns
Timer 1-2, SCI Input Clock Fall Time	t <sub>ckf</sub>		-	-	100	-	-	100	-	-	100	ns

\*\* Refer to Pages 472-476



**WIDE TEMPERATURE  
SPECIFICATIONS**  
-40°C TO +85°C  
(J VERSION)

# HD6303Y, HD63A03Y, HD63B03Y, HD63C03Y

The HD6303Y is a CMOS 8-bit single-chip microprocessing unit which contains a CPU compatible with the CMOS 8-bit microcomputer HD6301V, 256 bytes of RAM, 24 parallel I/O pins, Serial Communication Interface (SCI) and two timers.

## ■ FEATURES

- Instruction Set Compatible with the HD6301V1
- 256 Bytes of RAM
- 24 Parallel I/O Pins
- Parallel Handshake Interface (Port 6)
- Darlington Transistor Drive (Port 2, 6)
- 16-Bit Programmable Timer
  - Input Capture Register × 1
  - Free Running Counter × 1
  - Output Compare Register × 2
- 8-Bit Reloadable Timer
  - External Event Counter
  - Square Wave Generation
- Serial Communication Interface (SCI)
  - Asynchronous Mode (8 Transmit Formats, Hardware Parity)
  - Clocked Synchronous Mode
- Memory Ready
  - 3 Kinds of Memory Ready
- Halt
- Error Detection
  - (Address Error, Op-code Error)
- Interrupt – External 3, Internal 7
- Maximum 65k Bytes Address Space
- Low Power Dissipation Mode
  - Sleep Mode
  - Standby Mode (Hardware Standby, Software Standby)
- Minimum Instruction Execution Time – 0.5μs (f = 2MHz)
- Wide Range of Operation

$$V_{CC} = 3 \text{ to } 5.5\text{V} \quad (f = 0.1 \text{ to } 0.5\text{MHz})$$

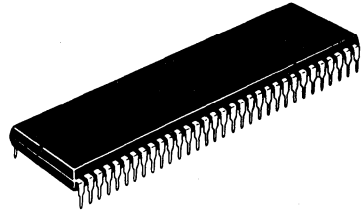
$$V_{CC} = 5\text{V} \pm 10\% \begin{cases} f = 0.1 \text{ to } 1.0\text{MHz} : \text{HD6301Y0} \\ f = 0.1 \text{ to } 1.5\text{MHz} : \text{HD63A01Y0} \\ f = 0.1 \text{ to } 2.0\text{MHz} : \text{HD63B01Y0} \\ f = 0.1 \text{ to } 3.0\text{MHz} : \text{HD63C01Y0} \end{cases}$$

## ■ GENERIC PART NUMBER

HD6303YPJ, HD63A03YPJ, HD63B03YPJ, HD63C03YPJ

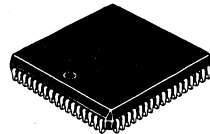
HD6303YCPJ, HD63A03YCPJ, HD63B03YCPJ, HD63C03YCPJ

HD6303YPJ, HD63A03YPJ,  
HD63B03YPJ, HD63C03YPJ



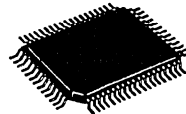
(DP-64S)

HD6303YCPJ, HD63A03YCPJ,  
HD63B03YCPJ, HD63C03YCPJ



(CP-68)

HD6303YF\*, HD63A03YF\*,  
HD63B03YF\*, HD63C03YF\*

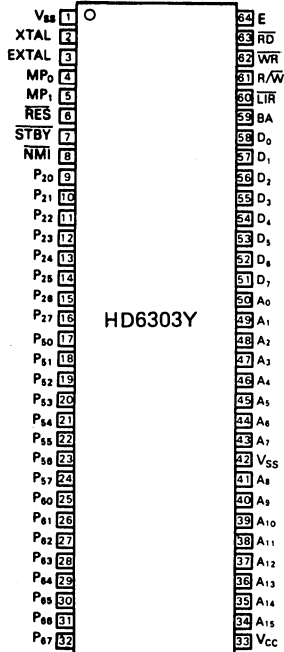


(FP-64)

\*Contact Hitachi Sales Office

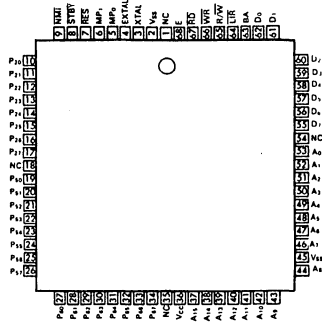
**WIDE TEMPERATURE  
SPECIFICATIONS  
-40°C TO +85°C  
(J VERSION)**

- HD6303YPJ, HD63A03YPJ,  
HD63B03YPJ, HD63C03YPJ



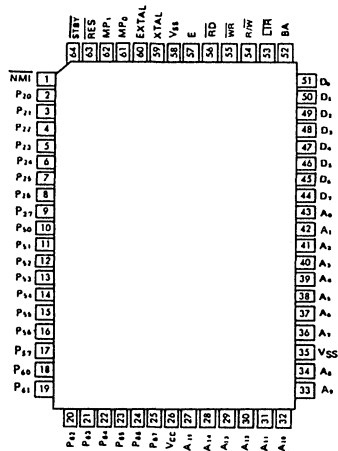
(Top View)  
(DP-64S)

- HD6303YCPJ, HD63A03YCPJ,  
HD63B03YCPJ, HD63C03YCPJ



(Top View)  
(CP-68)

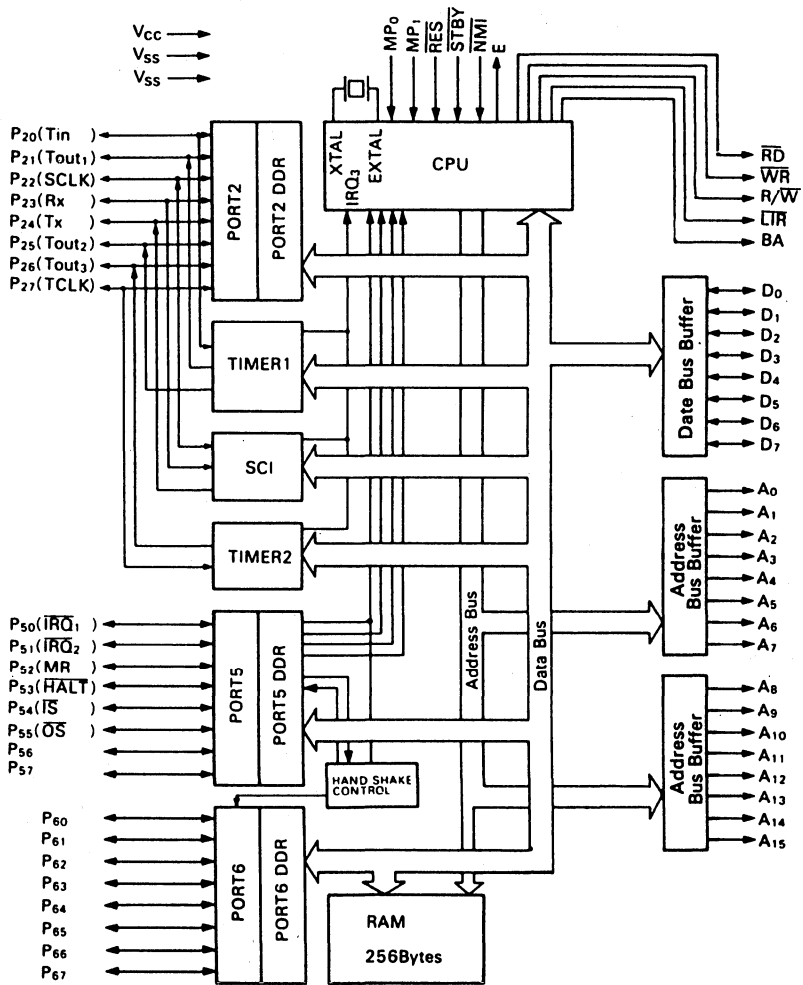
- HD6303YF\*, HD63A03YF\*,  
HD63B03YF\*, HD63C03YF\*



(Top View)  
(FP-64)

\*Contact Hitachi Sales Office

■ **BLOCK DIAGRAM**



## I.2 HD6303Y, HD63A03Y, HD63B03Y, Electrical Characteristics

### Absolute Maximum Ratings

Item	Symbol	Value	Unit
Supply voltage	V <sub>CC</sub>	-0.3 to +7.0	V
Input voltage	V <sub>in</sub>	-0.3 to V <sub>CC</sub> +0.3	V
Operating temperature	T <sub>opr</sub>	-40 to +85	°C
Storage temperature	T <sub>stg</sub>	-55 to +150	°C

Note: This product has protection circuits in input terminal from high static electricity voltage and high electric field. But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend V<sub>in</sub>, V<sub>out</sub>: V<sub>is</sub> ≦ (V<sub>in</sub> or V<sub>out</sub>) ≦ V<sub>CC</sub>.

### Electrical Characteristics

#### DC Characteristics

(V<sub>CC</sub> = 5.0V ± 10%, f = 0.1 to 3.0 MHz, V<sub>SS</sub> = 0 V, Ta = -40 to +85°C, unless otherwise noted)

Item	Symbol	Min	Typ	Max	Unit	Test Condition
Input high voltage	RES, STBY	V <sub>IH</sub>	V <sub>CC</sub> -0.5	V <sub>CC</sub> +0.3	V	
	EXTAL		V <sub>CC</sub> ×0.7	V <sub>CC</sub> +0.3	V	
	Other inputs		2.1	V <sub>CC</sub> +0.3	V	
Input low voltage	All other inputs	V <sub>IL</sub>	-0.3	0.8/0.6 <sup>3</sup>	V	
Input leakage current	RES, NMI, STBY, MP0, MP1	I <sub>in</sub>		1.0	μA	V <sub>in</sub> =0.5 to V <sub>CC</sub> -0.5 V
Three state leakage current	A <sub>0</sub> -A <sub>15</sub> , D <sub>0</sub> -D <sub>7</sub> , RD WR, R/W, Ports 2, 5, 6	I <sub>TSI</sub>		1.0	μA	V <sub>in</sub> =0.5 to V <sub>CC</sub> -0.5 V
Output high voltage	All outputs	V <sub>OH</sub>	2.4		V	I <sub>OH</sub> = -200 μA
			V <sub>CC</sub> -0.7		V	I <sub>OH</sub> = -10 μA
Output low voltage	All outputs	V <sub>OL</sub>		0.4	V	I <sub>OL</sub> = 1.6 mA
Darlington drive current	Ports 2, 6	-I <sub>OH</sub>	1.0	10.0	mA	V <sub>out</sub> = 1.5 V
Input capacitance	All other inputs	C <sub>in</sub>		12.5	pF	V <sub>in</sub> = 0V, f = 1 MHz Ta = 25°C
Standby current	Not operating	I <sub>STB</sub>	3.0	15.0	μA	
Current dissipation <sup>1</sup>	I <sub>SLP</sub>		1.5	3.0	mA	Sleeping (f = 1 MHz <sup>2</sup> )
			2.3	4.5	mA	Sleeping (f = 1.5 MHz <sup>2</sup> )
			3.0	6.0	mA	Sleeping (f = 2 MHz <sup>2</sup> )
			4.5	9.0	mA	Sleeping (f = 3 MHz <sup>2</sup> )
	I <sub>CC</sub>		7.0	10.0	mA	Operating (f = 1 MHz <sup>2</sup> )
			10.5	15.0	mA	Operating (f = 1.5 MHz <sup>2</sup> )
			14.0	20.0	mA	Operating (f = 2 MHz <sup>2</sup> )
			21.0	30.0	mA	Operating (f = 3 MHz <sup>2</sup> )
RAM standby voltage	V <sub>RAM</sub>	2.0			V	

Notes :

- V<sub>in</sub> min = V<sub>CC</sub> - 1.0V, V<sub>in</sub> max = 0.8V (All output terminals are at no load.)
- Current dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ. or max. values about current dissipations at x MHz operation are decided according to the following formula:  
 typ. value (f = x MHz) = typ. value (f = 1 MHz) × x  
 max. value (f = x MHz) = max. value (f = 1 MHz) × x  
 (both the sleeping and operating)
- In case of SCLK input, V<sub>IL</sub> = 0.6V (-20°C ~ 0°C)

**AC Characteristics**

( $V_{CC} = 5.0V \pm 10\%$ ,  $f = 0.1$  to  $3.0$  MHz,  $V_{SS} = 0$  V,  $T_a = -40$  to  $+85^\circ\text{C}$ , unless otherwise noted)

**Bus Timing**

Item	Symbol	HD6303Y			HD63A03Y			HD63B03Y			HD63C03Y			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Cycle time	$t_{cyc}$	1		10	0.666		10	0.5		10	0.333		10	$\mu\text{s}$	Fig. I-15*
Enable rise time	$t_{Er}$			25			25			25			20	ns	
Enable fall time	$t_{Ef}$			25			25			25			20	ns	
Enable pulse width high level <sup>1</sup>	$PW_{EH}$	450			300			220			140			ns	
Enable pulse width low level <sup>1</sup>	$PW_{EL}$	450			300			220			140			ns	
Address, R/W delay time <sup>1</sup>	$t_{AD}$			250			190			160			120	ns	
Data delay time (Write)	$t_{DDW}$			200			160			120			100	ns	
Data set-up time (Read)	$t_{DSR}$	80			70			60			50			ns	
Address, R/W hold time <sup>1</sup>	$t_{AH}$	80			50			40			20			ns	
Data hold time (Write) <sup>1</sup>	$t_{HW}$	70			50			40			20			ns	
	(Read)	$t_{HR}$	0		0			0			0			ns	
RD, WR pulse width <sup>1</sup>	$PW_{RW}$	450			300			220			140			ns	
RD, WR delay time	$t_{RWD}$			40			40			40			40	ns	
RD, WR hold time	$t_{HRW}$			20			20			20			20	ns	
LIR delay time	$t_{DLR}$			200			160			120			80	ns	
LIR hold time	$t_{HLR}$	5			5			5			5			ns	
Peripheral read access time <sup>1</sup>	$t_{ACC}$										180			ns	
MR set-up time <sup>1</sup>	$t_{SMR}$	400			280			230			170			ns	Fig. I-16*
MR hold time <sup>1</sup>	$t_{HMR}$			100			70			50			25	ns	
E clock pulse width at MR	$PW_{EMR}$			9			9			9			9	$\mu\text{s}$	
Processor control set-up time	$t_{PCS}$	200			200			200			100			ns	Figs. I-17, I-27, I-28*
Processor control rise time	$t_{PCr}$			100			100			100			50	ns	Figs. I-16, I-17*
Processor control fall time	$t_{PCf}$			100			100			100			50	ns	
BA delay time	$t_{BA}$			250			190			160			120	ns	Fig. I-17*
Oscillator stabilization time	$t_{RC}$	20			20			20			20			ms	Fig. I-28*
Reset pulse width	$PW_{RST}$	3			3			3			3			$t_{cyc}$	

Note: 1. These timings change in approximate proportion to  $t_{cyc}$ . The figures in this characteristics represent those when  $t_{cyc}$  is minimum (=in the highest speed operation).

\*Refer to Pages 618-621

**Peripheral Port Timing**

Item	Symbol	HD6303Y			HD63A03Y			HD63B03Y			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Peripheral data set-up time (Ports 2, 5, 6)	t <sub>pDSU</sub>	200		200		200				ns	Fig. I-19*	
Peripheral data hold time (Ports 2, 5, 6)	t <sub>pDH</sub>	200		200		200				ns		
Delay time (From enable fall edge to peripheral output) (Ports 2, 5, 6, 7)	t <sub>pWD</sub>			300		300			300	ns	Fig. I-20*	
Input strobe pulse width	t <sub>pWIS</sub>	200		200		200				ns	Fig. I-35*	
Input data hold time (Port 6)	t <sub>pH</sub>	150		150		150				ns		
Input data set-up time (Port 6)	t <sub>pS</sub>	100		100		100				ns		
Output strobe time	t <sub>OSD1</sub> t <sub>OSD2</sub>			200		200			200	ns	Fig. I-25*	

\*Refer to Pages 618–621

**Timer, SCI Timing**

Item	Symbol	HD6303Y			HD63A03Y			HD63B03Y			Unit	Test Condition
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Timer 1 input pulse width	t <sub>pWT</sub>	2.0		2.0		2.0				t <sub>cyc</sub>	Fig. I-23*	
Delay time (enable positive transition to timer output)	t <sub>TOD</sub>			400		400			400	ns	Figs. I-21, I-22*	
SCI input clock cycle	t <sub>Scyc</sub>	(Async mode)	1.0		1.0		1.0			t <sub>cyc</sub>	Fig. I-23*	
		(Clock sync.)	2.0		2.0		2.0			t <sub>cyc</sub>	Fig. I-18*	
SCI transmit data delay time (Clock sync. mode)	t <sub>TXD</sub>			240		240			240	ns	Fig. I-18*	
SCI receive data set-up time (Clock sync. mode)	t <sub>SRX</sub>	260		260		260				ns		
SCI receive data hold time (Clock sync. mode)	t <sub>HRX</sub>	100		100		100				ns		
SCI input clock pulse width	t <sub>pWSCX</sub>	0.4		0.6	0.4	0.6	0.4		0.6	t <sub>Scyc</sub>	Fig. I-23*	
Timer 2 input clock cycle	t <sub>cyc</sub>	2.0		2.0		2.0				t <sub>cyc</sub>		
Timer 2 input clock pulse width	t <sub>pWTCK</sub>	200		200		200				ns		
Timer 1 • 2, SCI input clock rise time	t <sub>CKr</sub>			100		100			100	ns		
Timer 1 • 2, SCI input clock fall time	t <sub>CKf</sub>			100		100			100	ns		

\*Refer to Pages 618–621

**NOTES:**



**NOTES:**



---

# Hitachi America, Ltd.

SEMICONDUCTOR and IC DIVISION

Hitachi America, Ltd.  
Semiconductor & IC Division  
Hitachi Plaza  
2000 Sierra Point Parkway  
Brisbane, CA 94005-1819  
Telephone: 415-589-8300  
Telex: 17-1581  
Twx: 910-338-2103  
FAX: 415-583-4207

---

## REGIONAL OFFICES

### MID-ATLANTIC REGION

Hitachi America, Ltd.  
1700 Galloping Hill Rd.  
Kenilworth, NJ 07033  
201/245-6400

### NORTHEAST REGION

Hitachi America, Ltd.  
5 Burlington Woods Drive  
Burlington, MA 01803  
617/229-2150

### NORTH CENTRAL REGION

Hitachi America, Ltd.  
500 Park Blvd., Suite 415  
Itasca, IL 60143  
312/773-4864

### NORTHWEST REGION

Hitachi America, Ltd.  
2000 Sierra Point Parkway  
Brisbane, CA 94005-1819  
415/589-8300

### SOUTH CENTRAL REGION

Hitachi America, Ltd.  
Two Lincoln Centre, Suite 865  
5420 LBJ Freeway  
Dallas, TX 75240  
214/991-4510

### SOUTHWEST REGION

Hitachi America, Ltd.  
18300 Von Karman Avenue,  
Suite 730  
Irvine, CA 92715  
714/553-8500

### SOUTHEAST REGION

Hitachi America, Ltd.  
4901 N.W. 17th Way, Suite 302  
Fort Lauderdale, FL 33309  
305/491-6154

### AUTOMOTIVE

Hitachi America, Ltd.  
6 Parklane Blvd., #558  
Dearborn, MI 48126  
313/271-4410

---

## DISTRICT OFFICES

Hitachi America, Ltd.  
3800 W. 80th Street  
Suite 1050  
Bloomington, MN 55431  
612/896-3444

Hitachi America, Ltd.  
21 Old Main Street, Suite 104  
Fishkill, NY 12524  
914/897-3000

Hitachi America, Ltd.  
6161 Savoy Dr., Suite 850  
Houston, TX 77036  
713/974-0534

Hitachi (Canadian) Ltd.  
2625 Queensview Dr.  
Ottawa, Ontario, Canada K2A 3Y4  
613/596-2777

Hitachi America, Ltd.  
401 Harrison Oaks Blvd.  
Suite #317  
Cary, NC 27513  
919/481-3908





Our Standards Set Standards

Hitachi America, Ltd.  
Semiconductor and IC Division  
Hitachi Plaza  
2000 Sierra Point Parkway, Brisbane, CA 94005-1819  
1-415-589-8300

---