TEXAS INSTRUMENTS

MSP430 Family

Architecture User's Guide and Module Library

1995

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

MSP430 Family
Architecture Guide and Module Library

Topics

Figures

Tables

List of Notes

Purpose and Convention

The MSP430 User's Guide is intended to aid development of MSP430 family products by putting together and presenting hardware and software information in an easy to use manner for engineers and programmers.

Short description and naming conventions of signals, processor states:

- ADC — Analog to digital converter
- CPUOff mode — Low power mode with RAM contents and I/O signals unchanged Modules using auxiliary clock (32 768 Hz crystal) are active
- DCO — Digital controlled oscillator
- LCD — Liquid crystal display
- FF — Flip-Flop
- MAB — Memory address bus. It is the address bus between the individual modules. It can be any width from 16-bit to 4bit. Together with MS signal it defines the physical address.
- MDB — Memory data bus. It is the data bus between the individual modules. It can be 8-bit or 16-bit wide.
- MS — Module select. This is the pre-decoded address space. Together with the MAB it defines the physical address.
- MSFR — Module special function register. This is the pre-decoded address space (0h to 0Fh) of the special function registers.
- OSCOff mode — Lowest power mode. RAM contents and I/O signals unchanged. Crystal oscillator stopped
- OTP — One-time programmable
- POR — Power-on reset
- PUC — Power-up clear - "1" set processor's start condition
- SAR — Successive approximation register
- SCI — Serial communication interface to handle synchronous and asynchronous protocols
- SCG — System clock generator
- SFR — Special function register
- TBD — To be defined
- TOS — Top of stack
- UART — Universal asynchronous receive transmit Most common used serial communication protocol
- WD,WDT — Watchdog, Watchdog Timer

Bit Type Convention for Register Bit

- rw: read/write
- r: read only
- r0: read as '0'
- w: write only
- (w): no register bit implemented; writing a '1' will result in a pulse the register bit is always read as '0'
- -0,-1: condition after PUC
- h0: cleared by hardware

Notations

Operations

@	Register indirect addressing
&	Absolute address
-->	Data transfer direction
+	Addition
-	Subtraction
x	Multiplication
/	Division
.AND.	logical AND
.OR.	logical OR
.XOR.	logical Exclusive-OR
.NOT.	logical NOT

Register Symbols

R0 or PC	Register 0 or Program Counter
R1 or SP	Register 1 or Stack Pointer
R2 or SR/CG1	Register 2 or Status Register/Constant Generator 1
R3 or CG2	Register 3 or Constant Generator 2
R4 to R15	Working Register, general-purpose

Contents of Status Register

C	Carry or borrow
Z	Zero
N	Negative
CPUOff	CPU Off Bit
OscOff	System Oscillator Off Bit
GIE	General Interrupt Enable
SCG0	System Clock Generator, Control Bit 0
SCG1	System Clock Generator, Control Bit 1
V	Overflow

Others

=	Equal Sign
‡	Not Equal Sign
>, <,≥,≤	Comparison Signs
" "	ASCII Character inside
h	Hexadecimal Data
b	Binary Data
#	Immediate Data
E	Exponent
&	Absolute Address Mode Indicator

Assembler Directives

.equ	Equate command
.sect	section directive
.word	word data
.byte	byte data
;	comment indicator

1 MSP430 Family

This section discusses the features of the MSP430 family of controllers with special capabilities for analog processing control. All family members are software compatible, allowing easy migration within the MSP430 family by maintaining a software base, design expertise and development tools.

The concept of a CPU designed for various applications with a 16-bit structure is presented. It uses a "von-Neumann Architecture" and hence has RAM, ROM and all peripherals in one address space.

1.1 Features and Capabilities

- Up to 64K byte addressing space for allocation of ROM, RAM, EERAM and peripherals as needed. Future expansion to 1M byte planned.
- No limitation of interrupt and subroutine levels due to stack processing
- Only 3 instruction formats. Strong orthogonality without any exception
- 1word/instruction as far as possible
- Seven address modes in source
- Four address modes in destination
- External interrupt pins: I/O-pins are used for interrupt purposes too
- Prioritized interrupts (simultaneously occurring interrupts are handled prioritized)
- Nested interrupt structure: interrupt routines may be interrupted by higher priority interrupts
- Memory mapped I/O-ports: peripheral registers in RAM
- UART on chip planned: seperate interrupts for transmit and receive part
- Timer with interrupt (1 or 2) usable as Event Counter
- Watchdog
- ADC (10 bits or more) with 8 inputs and current source
- EPROM version (OTP)
- LCD-driver
- Stable processor frequency using a FLL and a clock crystal of 32,768 Hz
- Easy program development because of the orthogonal structure: all instructions with all addressing modes
- C-compiler planned
- Modular design concept: Modules are strictly memory mapped

1.2 System Key Features

- Ultra-low current consumption: CPUOff and OscOff modes
- Full operation down to 2.5 V
- System building blocks: LCD-Drive, A/D-Converter, I/O-Ports, UART, Watchdog Timer, EEPROM ..... on chip
- Only microcomputer mode, no microprocessor mode
- Ease of use
  The powerful and convenient instruction set allows fast software development.
- Software may run in RAM
  Programs loaded into the RAM via the UART or test paths..., can execute jobs under real-time conditions. This reduces test costs, calibration expenses,.....
- Every ROM/RAM mix is possible in the common address range of 64k byte
- High level language (HLL) programming capabilities
  Large register file (12 general purpose registers)
  Stack orientation
  Large ROM and RAM spaces
  Orthogonal instruction set without exception
  Table processing orientation due to addressing modes
- Fast hexadecimal to decimal conversion with special instruction DADD

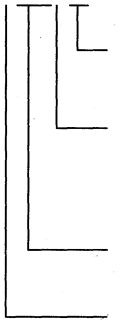
- Instructions are commonly used for ROM references, RAM access, data handling, I/Os and other peripherals: There are no special instructions!
- Potential of CPU far exceeds the requirements of intelligent sensor signal systems. The real-time capability opens fields in other low power systems including usage of other peripherals e.g. body functions in automotive

1.3 MSP430 Family Devices

The MSP430 family devices can be summarized as follows:

- Naming convention:

MSP430CxxxCFN



Package Code, 1 or 2 characters

Temperature range, 1 character
C: 0 to 70°C
Q: Customer defined
A: -40 to +125 °C

Unique number for each family member
or software number, 3 characters

Memory Code: C = CMOS
P = OTP
E = EPROM
S = SRAM

- Development tools include software simulator **SIM430**, assembler and linker **ASM430**, C-compiler (future product) **CCP430** and hardware in-circuit emulator **ICE430**. All development tools are PC-based using integrated desktop features compatible to the windows SAA standard.
The minimum requirements for the PC are:
IBM compatible
DOS 3.3 or later
Windows 3.1 or later
Personal computer with a 386 or higher processor running
4 MB of available memory
One 3.5" hidh-density disk drive
A hard disk with 5 MB available

	MSP430x320	MSP430x310	MSP430Cxxx
Max. internal clock rate Frequency of crystal	1.1 MHz 32.768 kHz	1.1 MHz 32.768 kHz	x MHz 32.768 kHz
Operating Temperature	-40°C to +85°C	-40°C to +85°C	-40°C to +85°C
Program memory MSP430Cxxx: MSP430Pxxx: MSP430Exxx: Memory expansion	 4/8k byte ROM 16K byte OTP 16K byte wind. EPROM NO	 4/8K byte ROM 8K byte OTP 8K byte wind. EPROM NO	 TBD TBD TBD NO
Internal RAM	256/512 Bytes	256 Bytes	TBD
Data EEPROM	No	N0	TBD
Modules Port0, 8-bit, all interrupt Watchdog timer Basic Timer/Real clock Basic Timer1/Real clock 8-bit Timer/Counter Timer/Port ,2x8-bit Timer 3,16-bit SPI SCI LCD ADC/Current source DAC	 Yes Yes No Yes Yes Yes No No (8b Tim./Cnt. + SW) Yes Yes/Yes No	 Yes Yes No Yes Yes Yes No No (8b Tim./Cnt. + SW) Yes see Timer/Port No	 Yes Yes TBD TBD TBD TBD TBD TBD TBD TBD TBD TBD
I/O lines	8	8	TBD
Interrupts/Reset External Vectors total Sources total	 11 16	 11 16	 TBD TBD TBD
Package Type	64 QFP	56 SSOP	

Table 1.1: MSP430 Family Feature Summary

2 Architectural Overview

This section describes the basic functions of a MSP430 based system.

The MSP430 devices contain the following major functions:

- Central Processing Unit (CPU)
- Program Memory (ROM or EPROM)
- Data Memory (RAM or EEPROM)
- Control of operation
- Peripheral Modules
- Oscillator + Frequency Multiplier.

The architecture of the MSP430 family is based on a memory-to-memory architecture, a common address space for all functional blocks and a reduced instruction set applicable for all functional blocks.

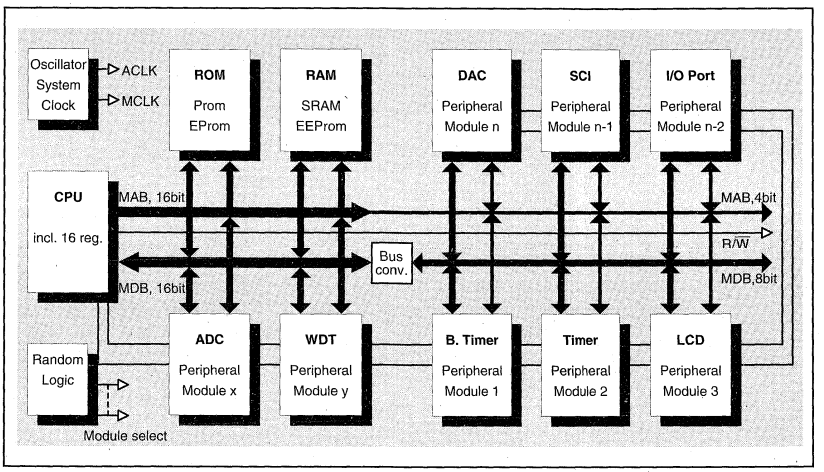


Figure 2.1: MSP430 system configuration principle

2.1 CPU

The central processing unit incorporates following reduced, highly transparent instruction set and highly orthogonal design. It consists of a sixteen bit ALU, sixteen registers and an instruction control logic. Four of these registers are used for special purposes, these are the Program Counter PC, Stack Pointer SP, Status Register SR and Constant Generator CG2. All registers - except R3/CG2 and part of R2/CG1- can be used as general-purpose registers applying the complete instruction set to registers. The constant generator supplies constants for performing the instruction not for storing any data. The addressing mode used on CG1 separates the data of the constants.

The complete control over Program Counter, processor's Status Register and Stack Pointer with the reduced instruction set opens development of applications with complex addressing modes or SW algorithm.

2.2 Code Memory

Access to the Code Memory is always word organized for fetching code, data can be read with word or byte access. Any access uses the 16-bit Memory Data Bus and as many of the least significant address lines as are needed to access the memory locations. Blocks of memory are automatically selected via Module Enable signals as a technique to reduce overall current consumption. Program memory can be integrated as programmable (EPROM) or mask programmable (ROM) memory. Standard members of the MSP430 family support OTP and mask programmed versions. Support of external memory will be the subject of future enhancements.

Sixteen words of memory are reserved for reset and interrupt vectors at the top of the lowest 64K byte address space from 0FFFFh down to 0FFE0h.

Access to Program Memory via software program is fully supported for read operation (MOV &0FFA0h,R5) not for write (ROM).

Future enhancements:

The address space will be enhanced using segmented memory areas. The expanded addressable space is supported mainly using three extensions: branch and call long instructions, code segment pointer CSP and data pointer DPP. The code segment pointer is located within the status register SR. This enhanced address space is used for instruction codes (CSP + PC) and for data memory ([DPPi] + operand address):

MAB = CSP * 10000h + PC	during any access to code memory
MAB = DDPi * 4000h + Rs/d	during any access to stack or data memory

For basic devices using up to 64K byte addressing space the content of CSP and DPP is unused by the Memory Address Bus.

2.3 Data Memory (RAM)

The Data Memory is connected to the CPU via two busses, the Memory Address Bus (MAB) and the Memory Data Bus (MDB). The Data Memory can be integrated into the specific family member either with full (word) data width or with reduced (byte) data width.

The entire instruction set operates fully on byte and word data. All operations on stack and PC are word operations and should use only even aligned addresses.

2.4 Control of operation

The operations of the different MSP430 members are controlled mainly with the information stored in special function registers SFRs. The different bits in the SFRs enable interrupts, support the software on the status of the interrupt flags and define the operating modes of the peripherals. Peripherals that are disabled stop their functional operation to reduce current consumption. All data stored in the module's register are retained. Peripherals that have their operating mode controlled can be identified in the specific sections.

2.5 Peripherals

Peripheral modules are connected to the CPU via Memory Address Bus MAB, Memory Data Bus MDB and interrupt service and request lines. The MAB is usually a 5-bit bus for most of the peripherals. The MDB is an 8-bit or 16-bit bus. Modules with an 8-bit data bus are connected via bus conversion circuitry to the 16-bit CPU. The data exchange with these modules should be handled with byte instructions without excepeption. Instruction execution on word oriented peripherals operate without any restrictions. Most of the peripherals are operating in byte format. The SFRs are handled within 8-bit data range without any exception. The operation to 8-bit peripherals follows the described orders.

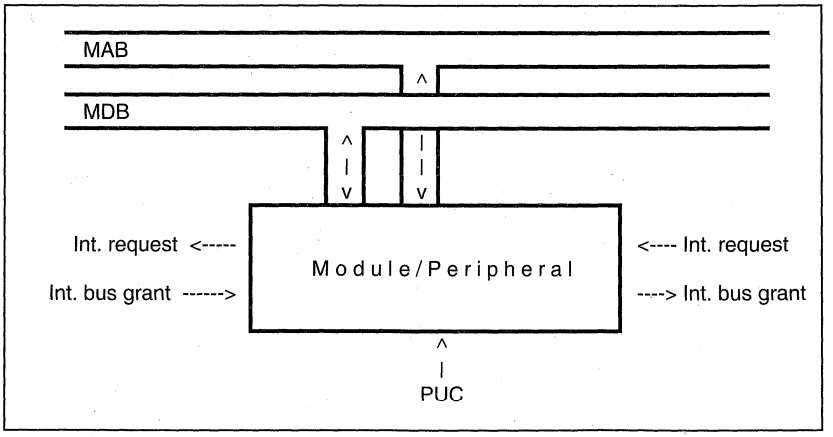


Figure 2.2: Bus connection of modules/peripherals

2.6 Oscillator, Frequency Multiplier and Clock Generator

The oscillator is specially designed for the commonly used clock crystal of 32,768 Hz with low current consumption. All analog components are integrated; only the crystal has to be connected.

This oscillator is the direct source for some modules with low frequency requirements. For the CPU and other modules the crystal's frequency is multiplied by a first order frequency lock loop circuitry FLL. The FLL starts after power-up with its lowest possible frequency and is regulated to the proper frequency by controlling a digital controlled oscillator DCO.

The long-term deviation is limited by the stability of the crystal and oscillator.

The frequency of the clock generator for the processor's operation is a fixed multiple of the crystal and supports the clock MCLK.

3 System Reset, Interrupts and Operating Modes

3.1 System Reset & Initialization

The MSP430 has four possible reset sources: applying supply voltage to V_{CC} pin, a low input to the RST/NMI pin, a programmable watchdog timer time-out and a security key violation during write access to WDTCTL register.

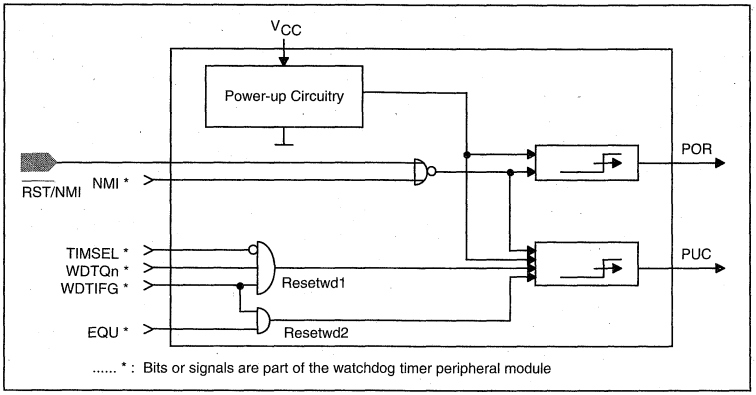


Figure 3.1: System Reset Function

After the occurrence of a reset, the program can interrogate flags according to the reset source. The program can determine the source of reset in order to take appropriate action.

The MSP430 starts hardware initialization after applying V_{CC}:

- All I/O-pins are switched to the input direction
- The I/O-flags are cleared as described in the appropriate peripheral descriptions
- The address contained in the reset vector at word address 0FFFEh is placed into the Program Counter
  The CPU starts at the address contained in the power-up clear (PUC) vector.
- The status register SR is reset.
- All registers have to be initialized by the user's program (e.g., the Stack Pointer, the RAM, ...) - except PC and SR.
- Registers located in the peripherals are handled as described in the appropriate section.
- The frequency controlled system clock starts with the lowest frequency of the digital controlled oscillator. After the start of the crystal clock the frequency is regulated to the target value.
  The RST/NMI pin is configured with the reset function after applying V_{CC}. It remains reset as long as the reset function is selected. When the pin is configured with the reset function, the MSP430 starts operation after the RST/NMI pin is pulled down to Gnd and released with the following:

- The address contained in the reset vector at word address 0FFFEh is placed into the Program Counter
- The CPU starts at the address contained in the reset vector after the release of the RST/NMI pin.
- The status register SR is reset.
- All registers have to be initialized by the user's program (e.g., the Stack Pointer, the RAM, ...) - except PC and SR.
- Registers located in the peripherals are handled as described in the appropriate section.
- The frequency controlled system clock starts with the lowest frequency of the DCO. After the start of the crystal clock the frequency is regulated to the target value.

3.2 Global Interrupt Structure

There are three types of interrupts:
- System reset
- Non-maskable interrupts
- Maskable interrupts

Sources causing a system reset are:
- Applying supply voltage @ POR, PUC
- 'low' on RST/NMI (if reset mode selected) @ POR, PUC
- Watchdog timer overflow (if watchdog mode selected) @ PUC
- Watchdog timer security key violation @ PUC
- (write to WDTCTL with incorrect password)

A non-maskable interrupt can be generated by:
- Edge on RST/NMI-pin (if NMI mode selected)
- Oscillator fault

Note: Oscillator fault

The oscillator fault is maskable by an individual enable bit OFIE. It is not disabled during general interrupt enable (GIE) reset.

Sources for maskable interrupts are:
- Watchdog timer overflow (if timer mode selected)
- other modules having interrupt capability

MSP430 Interrupt Priority Scheme

The interrupt priority of the modules is defined by the arrangement of the modules in the connection chain:
the nearer a module in the chain is towards the CPU/NMIRS the higher is the priority.

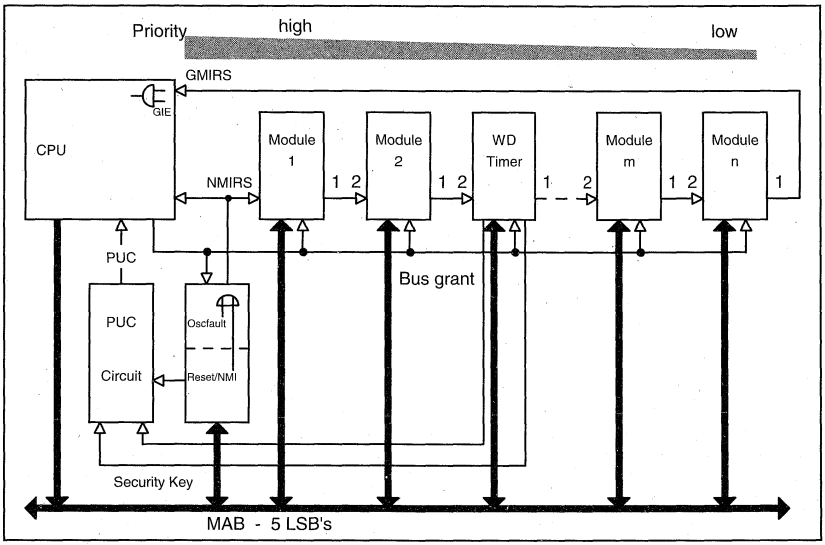


Figure 3.2: Interrupt Priority Scheme

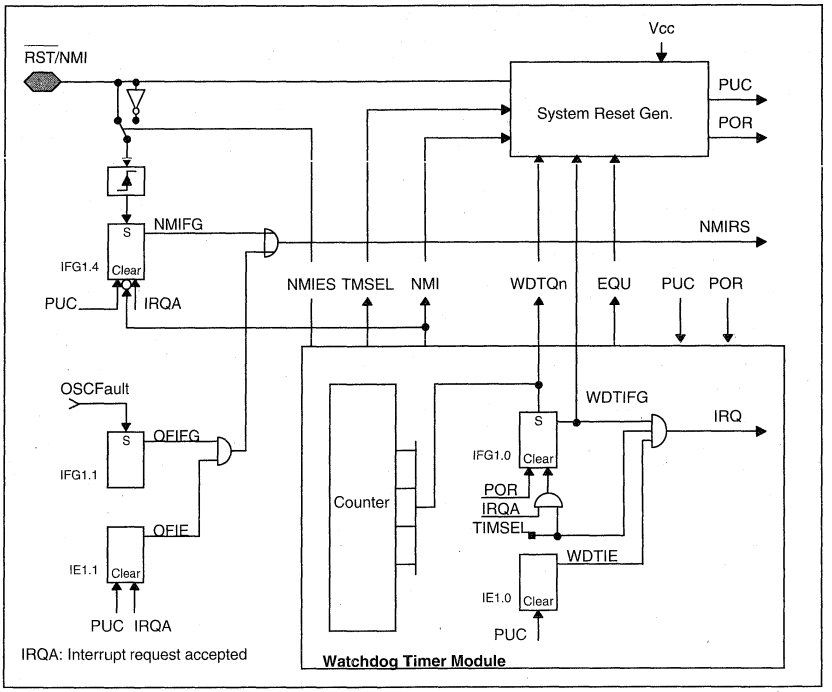


Figure 3.3: Reset/NMI-mode selection

Reset and NMI can be used only alternatively because they use the same input pin. The associated control bits are located in the Watchdog Timer Control register and are also password protected.

WDTCTL
0120h

7							0
HOLD	**NMIES**	**NMI**	TMSEL	CNTCL	SSEL	IS1	IS0
rw-0	rw-0	rw-0	rw-0	(w)-0	rw-0	rw-0	rw-0

BIT 5: The NMI-Bit selects the function of the $\overline{\text{RST}}$/NMI-input pin. It is cleared after PUC.

NMI = 0: The $\overline{\text{RST}}$/NMI input works as Reset input.
As long as the $\overline{\text{RST}}$/NMI-pin is held 'low', the internal PUC-signal is active (level sensitive).

NMI = 1: The $\overline{\text{RST}}$/NMI input works as edge sensitive non-maskable interrupt input.

BIT 6: This bit selects the activating edge of the $\overline{\text{RST}}$/NMI input if NMI function is selected. It is cleared after PUC.
NMIES = 0: A rising edge triggers a NMI-interrupt.
NMIES = 1: A falling edge triggers a NMI-interrupt.

Operation of global interrupt - Reset/NMI

If the Reset function is selected, the CPU is held in the reset state as long as the $\overline{\text{RST}}$/NMI-pin is held 'low'. After the input has changed to high, the CPU starts program execution at the word address which is stored in word location 0FFFEh (Reset vector).

If the NMI function is selected, an edge according to the NMIES-bit generates an unconditional interrupt and program execution is resumed at the address which is stored in location 0FFFCh. The $\overline{\text{RST}}$/NMI flag in the SFR (IFG1.4) is also set. It is automatically reset during interrupt request service. The $\overline{\text{RST}}$/NMI pin should never be held permanently 'low'. When a situation happens that activates the PUC, the consecutive reset of the bits in WDTCTL register forces the reset function on $\overline{\text{RST}}$/NMI pin. An enduring 'low' at $\overline{\text{RST}}$/NMI pin results in a permanent reset and system hold.

Note: NMI edge select

When NMI mode is selected and the NMI edge select bit is changed an NMI can be generated pending on the actual level at $\overline{\text{RST}}$/NMI pin.

When the NMI edge select bit is changed before selecting the NMI mode no NMI is generated.

Operation of global interrupt - Oscillator fault control

As described in the oscillator section, the FLL oscillator is still working even if the crystal is defective, but it runs at the lowest possible frequency. The second limit is the highest possible frequency. Both cases are usually error conditions and must be detectable by the CPU. Therefore the oscillator fault signal can be enabled by SFR bit IE1.1 to generate an NMI interrupt. By testing the interrupt flag IFG1.1 in the SFR, the CPU can determine if the interrupt was caused by an oscillator fault.

Operation of global interrupt - Power-up-clear (PUC)

Three sources can initiate system reset:
- Power-up logic
- $\overline{\text{RST}}$/NMI input
- Watchdog overflow.

Resets caused by $\overline{\text{RST}}$/NMI and the watchdog can be evaluated by software through testing the associated interrupt flag in SFR bit IFG1.0.

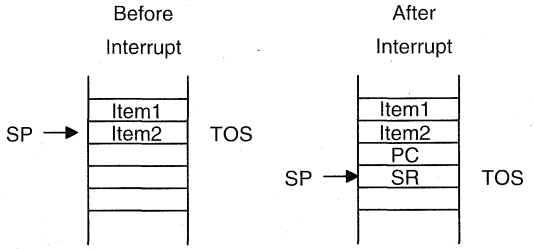
3.3 Interrupt Processing

The MSP430 programmable interrupt structure allows flexible on-chip and external interrupt configurations to meet real-time interrupt-driven system requirements. Interrupt sources may come from processor's operating conditions like watchdog overflow, peripheral modules or external events. Each interrupt source can be disabled individually by an interrupt enable bit or all interrupts are disabled by general interrupt enable bit GIE in status register.

Whenever an interrupt is requested and the appropriate interrupt enable bit and the General Interrupt Enable Bit (GIE) is set, the interrupt service routine becomes active:

- CPU active: The currently executed instruction is completed.
  CPU stopped: The low power modes are terminated.
- The Program Counter pointing to the next instruction is pushed onto the stack.
- The Status Register is pushed onto the stack.
- The interrupt with the highest priority is selected if multiple interrupts occurred during the last instruction and are pending for service.
- The appropriate interrupt requesting flag is reset automatically on single source flags. Multiple source flags remain set for servicing by software.
- The general interrupt enable bit GIE is reset,
  the CPUOff bit, the OscOff bit and the SCG1*) bit are cleared,
  the status bits N, Z and C are reset.
- The content of the appropriate interrupt vector is loaded into the Program Counter: The program continues with the interrupt handling routine at that address.

*) SCG0 is left unchanged and FLL loop control remains in previous operating condition.



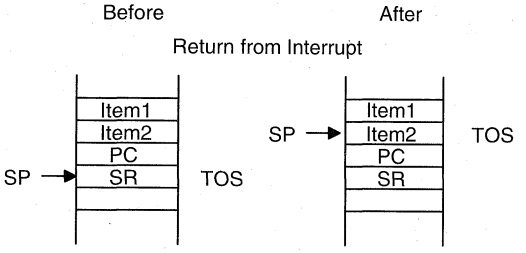
The interrupt latency is six cycles starting with acceptance of an interrupt request until beginning execution of the first instruction of the appropriate interrupt service routine.

The interrupt handling routine terminates with the instruction:

RETI

which does the following actions:

- The Status Register is popped from the stack.
  The interrupted software continues with exactly the same status as before the interrupt
  including OscOff, CPUOff and GIE bits.
  The GIE bit in the Status Register replaces the logical state present during interrupt service with the pushed state from TOS. It is set anyway - because it was set prior to accepting the interrupt.
- The Program Counter is popped from the stack.



The return from an interrupt service routine with the RETI instruction takes five cycles.

Interrupt nesting is empowered if the GIE-bit is set inside the interrupt handling routine.

The general interrupt enable bit GIE is located in the Status Register SR/R2 which is included in the CPU:

15	8	7							0
reserved for future enhancements	V	SCG1	SCG0	Osc Off	CPU Off	GIE	N	Z	C
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Figure 3.4: Status Register SR

Other then the GIE bit, other sources of interrupt requests can be enabled/disabled individually or in groups. The interrupt enable flags are located together within two addresses of the special function register SFR. The program flow conditions on interrupt requests can be comfortably adjusted by extensive use of the interrupt enable masks. The hardware serves the highest priority within the empowered interrupt source.

3.3.1 Interrupt Control Bits in Special Function Registers SFRs

Most of the interrupt control bits, interrupt flags and interrupt enable bits are collected in SFRs under a few addresses. The Special Function Registers are located in the lower

address range and are implemented in byte format. SFRs should be only accessed with byte instructions.

Address	7 0
000Fh	Not defined implemented yet
000Eh	:
000Dh	:
000Ch	:
000Bh	:
000Ah	:
0009h	:
0008h	:
0007h	:
0006h	:
0005h	Module enable 2; ME2.x
0004h	Module enable 1; ME1.x
0003h	Interrupt flag reg. 2; IFG2.x
0002h	Interrupt flag reg. 1; IFG1.x
0001h	Interrupt enable 2; IE2.x
0000h	Interrupt enable 1; IE1.x

The different devices of the MSP430 Family support the SFRs with the correct logic and function within the individual modules. Each module interrupt source, except the non-maskable sources, can be individually enabled to access the interrupt function and the operation. Full software control of these configuration bits enables the application software to react to system requirements on interrupt enable mask.

Interrupt Enable 1 and 2

Bit position	Short form	Initial state*	Comment
IE1.0	WDTIE	reset	Watchdog Timer enable signal. Inactive if watchdog mode is selected.
IE1.1	OFIE	reset	Oscillator fault enable
IE1.2	P0IE.0	reset	Dedicated I/O P0.0
IE1.3	P0IE.1	reset	Dedicated I/O P0.1 or 8-bit Timer/Counter
IE1.4		reset	reserved, not defined yet
IE1.5	SPI	reset	reserved, not defined yet
IE1.6		reset	reserved, not defined yet
IE1.7		reset	reserved, not defined yet
IE2.0		reset	reserved, not defined yet
IE2.1		reset	reserved, not defined yet
IE2.2	ADIE / TPIE	reset	ADC or Timer/Port enable signal ('310 config.)

* Initial state is the logical state after PUC. For the WDTIFG see the appropriate comment.

Bit position	Short form	Initial state	Comment
IE2.3	TPIE	reset	Timer/Port ('320 config.)
IE2.4		reset	reserved, not defined yet
IE2.5		reset	reserved, not defined yet
IE2.6		reset	reserved, not defined yet
IE2.7	BTIE	reset	Basic Timer enable signal

Interrupt Flag Register 1 and 2

Bit position	Short form	Initial state	Comment
IFG1.0	WDTIFG	unchanged	Set on verflow or security key violation;
		or reset	Reset on VCC power-on or reset condition at `RST/NMI-pin
IFG1.1	OFIFG	set	Flag set on oscillator fault
IFG1.2	P0IFG.0	reset	Dedicated I/O P0.0
IFG1.3	P0IFG.1	reset	Dedicated I/O P0.1 or 8-bit Timer/Counter
IFG1.4	NMIIFG	reset	Signal at`RST/NMI-pin
IFG1.5	SPI		reserved, not defined yet
IFG1.6			reserved, not defined yet
IFG1.7			reserved, not defined yet
IFG2.0			reserved, not defined yet
IFG2.1			reserved, not defined yet
IFG2.2	ADIFG	reset	ADC, set on end-of-conversion
IFG2.3			reserved, not defined yet
IFG2.4			reserved, not defined yet
IFG2.5			reserved, not defined yet
IFG2.6			reserved, not defined yet
IFG2.7	BTIFG	unchanged	Basic Timer flag

Module enable 1and 2

Bit position	Short form	Initial state	Comment
ME1.0			reserved, not defined yet
ME1.1			reserved, not defined yet
ME1.2			reserved, not defined yet
ME1.3			reserved, not defined yet
ME1.4			reserved, not defined yet
ME1.5			reserved, not defined yet
ME1.6			reserved, not defined yet
ME1.7			reserved, not defined yet
ME2.0			reserved, not defined yet
ME2.1			reserved, not defined yet
ME2.2			reserved, not defined yet
ME2.3			reserved, not defined yet

ME2.4			reserved, not defined yet
ME2.5			reserved, not defined yet
ME2.6			reserved, not defined yet
ME2.7	BTME	unchanged	Module function halted at present state (Basic Timer)

Interrupt Vector Addresses

The interrupt vectors and the power-up starting address are located in the ROM, using address range 0FFFFh - 0FFE0h. The vector contains the 16-bit address of the appropriate interrupt handler instruction sequence. The interrupt vectors are shown in decreasing priority order:

Interrupt source	Interrupt flag	System Interrupt	Word Address	Priority
Power-up ext. Reset Watchdog	WDTIFG	Reset	0FFFEh	15, highest
NMI OSC. fault	RSTI * OFIFG *	non-maskable (non-)maskable	0FFFCh	14
Dedicated I/O	P0IFG.0	maskable	0FFFAh	13
Dedicated I/O	P0IFG.1	maskable	0FFF8h	12
(SPI)**		maskable	0FFF6h	11
Watchdog timer	WDTIFG	maskable	0FFF4h	10
(TimerA) **		maskable	0FFF2h	9
(TimerA) **		maskable	0FFF0h	8
(SCI)		maskable	0FFEEh	7
Timer/Port[1)]		maskable	0FFECh	6
ADC, Timer/Port[2)]	ADCIFG	maskable	0FFEAh	5
(TimerB) **		maskable	0FFE8h	4
		maskable	0FFE6h	3
		maskable	0FFE4h	2
Basic Timer	BTIFG	maskable	0FFE2h	1
I/O Port 0	P0IFG.27 *	maskable	0FFE0h	0, lowest

*) multiple source flags
**) Preliminary definition
1) Timer/Port vector in '320 configuration
2) Timer/Port vector in '310 configuration

Table 3.1: Interrupt sources, flags and vectors

3.3.2 External Interrupts

An entire port of eight bits is fully implemented for interrupt processing of external events. All individual I/O bits are programmable independently:

Any combinations of inputs, outputs and interrupt conditions are possible. This allows an easy adaptation to different I/O configurations.

> **Note: Minimum pulse width of external interrupt signals**
>
> All external interrupt signals should have a minimum pulse width of 1.5 MCLK to ensure stable interrupt acknowledge, but shorter signals may also request an interrupt service

Three separate vectors are allocated to the port 0 module. The signals on P0.0, P0.1 and the remaining port signals P0.2 to P0.7 are used as the three interrupt sources. The vector contained in ROM is loaded into the Program Counter by an interrupt event: vector in word address 0FFFAh on P0.0 events, vector in word address 0FFF8h on P0.1 events. Any other interrupt event on bit 2 to 7 (P0.2 ..... P0.7) uses the common vector specified in ROM word location 0FFE0h.

The 6 registers used for the control of the I/O-pins are shown below:

Address	Name	7	0
010h	P0IN	Input register	R
011h	P0OUT	Output register	R/W
012h	P0DIR	Direction register	R/W
013h	P0IFG	Interrupt Flags *)	R/W
014h	P0IES	Interrupt Edge Select	R/W
015h	P0IE	Interrupt Enable	R/W

*) The two LSBs of the Interrupt Flags and Interrupt Enable bite are located in SFRs

Figure 3.5: I/O Control Registers

Input Register:

Read only register, to scan the signals at the I/O-pins.

Output Register:

The Output Register shows the information of the output buffer, an 8-bit register that contains the information output by the I/O-pins if used as outputs. The output buffer can be modified by all instructions that write to a destination. When read, the contents of the output buffer are read independently of the direction. A direction change does not modify the output buffer contents.

Direction Register:

This register contains eight independent bits which define the direction of the I/O-pins:
Bit = 0: The I/O-pin is switched to input direction
Bit = 1: The I/O-pin is switched to output direction

Interrupt Flags:

This register contains six flags that contain information if the I/O-pins are used as interrupt inputs:
Bit = 0: No interrupt is pending
Bit = 1: An interrupt is pending due to a transition at the I/O-pin.
Writing a zero to an Interrupt Flag resets it.
Writing a one to an Interrupt Flag sets it. Device operation continues just the same way as if an interrupt event had occured.

Interrupt Edge Select:

This register contains a bit for each I/O-pin that selects which transition triggers the interrupt flag.
Bit = 0: The interrupt flag is set with LO/HI transition
Bit = 1: The interrupt flag is set with HI/LO transition

Interrupt Enable:

This register contains six bits for the I/O-pins P0.2 to P0.7 to enable interrupt request on an interrupt event.
Bit = 0: The interrupt request is disabled
Bit = 1: The interrupt request is enabled

Note: How the interrupts on Port0 are handled

Only transitions, not static levels cause interrupts.

The interrupt routine must reset the multiple used Interrupt Flags P0IFG.2 .... P0IFG.7. The single source flags P0IFG.0 and P0IFG.1 are reset when they are serviced.

If an Interrupt Flag is still set (because the transition occurred during the interrupt routine) when the *RETI* instruction is executed, an interrupt occurs again after the *RETI* is completed. This ensures that each transition is seen by the software.

I/O-PIN interrupt handler for P0.2 to P0.7, Programming Example

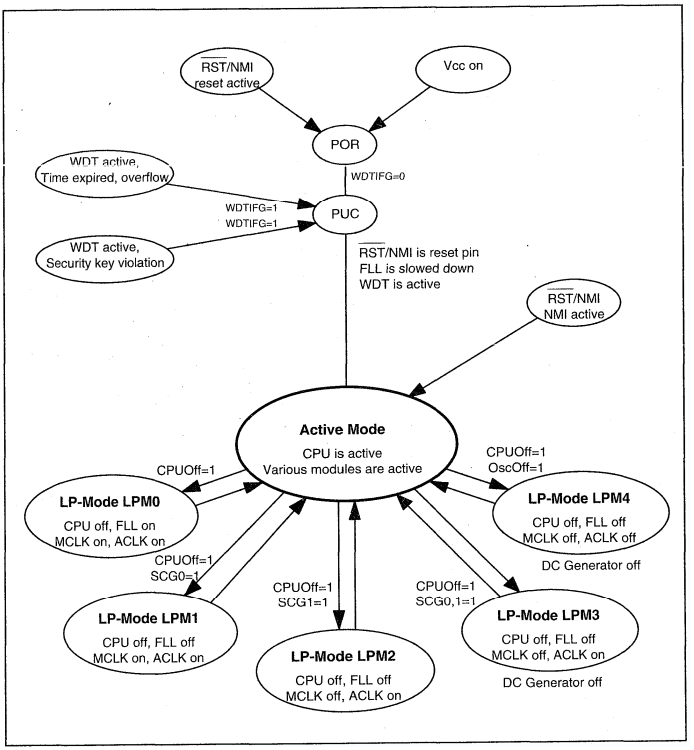
```
; The I/O-PIN interrupt handler for P0.2 to P0.7starts here
;
IOINTR      PUSH        R5              ; Save R5
            MOV.B       &P0IFG,R5       ; Read interrupt flags
            BIC.B       R5,&P0IFG       ; Clear status flags with the read
                                        ; data
                                        ; Additional set bits are not cleared!
            EINT                        ; Allow interrupt nesting
;
; R5 contains information which I/O-pin(s) caused interrupt:
; the processing starts here.
;
            .........
            .........
            POP         R5              ; JOB done: restore R5
            RETI                        ; Return from interrupt
            .........
            .........
; Definition of interrupt vector table
            .sect       "IO27_vec",0FFE0h
            .WORD       IOINTR          ; I/O-Pin (2 To 7) Vector In ROM
            ;
            .sect       "RST_vec",0FFFEh; Interrupt Vectors
            .WORD       RESET
```

3.4 Operating Modes

The MSP430 operating modes support in an advanced manner various requirements for ultra-low power and ultra-low energy consumption. This is combined with an intelligent management of operations during the different module and CPU states. An interrupt event awakes the system from each of the various operating modes and the RETI instruction returns operation to the mode that was selected before the interrupt event.

The MSP430 Family has been developed for ultra-low power applications and uses different levels of operating modes.

Ultra-low power system design in CMOS technology follows three primary intentions:
- the desire for speed and data throughput is contrary to design for ultra-low power
- minimize individual current consumption
- limit activity state to its minimum required.

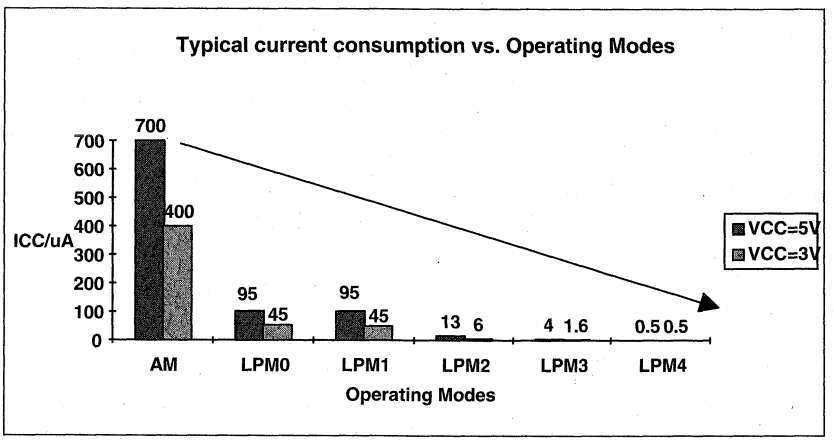


There are five operating modes the software can configure:

- **Active Mode AM,**
with different combinations of active peripheral modules
- **Low Power Mode 0 LPM0,**
with CPUOff bit set the CPU is disabled,
peripheral's operation is not halted by CPUOff,
ACLK and MCLK signal are active. Loop control for MCLK is active.
@ SCG1=0, SCG0=0, OSCOff=0, CPUOff=1
- **Low Power Mode 1 LPM1,**
with CPUOff bit set the CPU is disabled,
peripheral's operation is not halted by CPUOff,
loop control (frequency-lock-loop) for MCLK is inactive,
ACLK and MCLK signal are active.
@ SCG1=0, SCG0=1, OSCOff=0, CPUOff=1
- **Low Power Mode 2 LPM2,**
with CPUOff bit set the CPU is disabled,
peripheral's operation is not halted by CPUOff,

loop control for MCLK signal is inactive,
ACLK signal is active.
@ SCG1=1, SCG0=0, OSCOff=0, CPUOff=1

- **Low Power Mode 3 LPM3,**
  with CPUOff bit set the CPU is disabled,
  peripheral's operation is not halted by CPUOff,
  Loop control for MCLK and MCLK signal are inactive,
  DC generator of the DCO (-> MCLK generator) is switched off.
  ACLK signal is active.
  @ SCG1=1, SCG0=1, OSCOff=0, CPUOff=1
- **Low Power Mode 4 LPM4,**
  with CPUOff bit set the CPU is disabled,
  peripheral's operation is not halted by CPUOff,
  loop control for MCLK signal is inactive,
  DC generator of the DCO (-> MCLK generator) is switched off,
  ACLK signal is inactive; the crystal oscillator is stopped.
  @ SCG1=X, SCG0=X, OSCOff=1, CPUOff=1



The activity state of individual peripheral modules and the CPU can be controlled using the appropriate low power mode and various options to stop operation of parts or of entire peripheral modules. There are different ways to configure, with the software on an application specific basis, the lowest potential current consumption. The special function registers include module enable bits that stop or enable the operational function of the specific peripheral module. All registers of the peripherals may be accessed even during disable mode. Other current saving functions can be implemented into peripherals that are accessed via the state of register bits. An example is the enables/disable of the analog voltage generator in the LCD peripheral that is turned on or off via one register bit. The most general bits that influence the current consumption and support fast turn-on from low power operating modes are located in the status register SR. There are four bits that control the CPU and the system clock generator.

These four bits are very useful to support the request for discontinuous active mode AM and to limit the time period of the full operating mode. The four bits are CPUOff, OscOff, SCG0 and SCG1. The major advantage of including the operating mode bits into the status register is that the present state of the operating condition are saved onto stack during an interrupt request service. As long as the stored status register information is not altered the processor continues (after *RETI*) with the same operating mode as before the interrupt event. Another program flow may be selected by manipulation of the data stored on the stack or the stack pointer. The easy access of the stack and stack pointer with the instruction set allows individual optimized program structures.

3.5 Low Power Modes

The module enable bits in the SFRs enable the configuration of individual power consuming controller operation states. The users program defines the state of the peripheral modules to be active or inactive. The current consumption of disabled modules is decreased to leakage current for all parts that can be disabled. The only active parts of a module are those which are mandatory to get it to the enable state or to pass interrupt requests to the CPU (e.g. external hardware interrupt).

In addition to the individual enable options there are five more current saving modes possible: The CPU off mode (LPM0) and four operating configurations of the system clock generator. They are entered if one or more of the bits CPUOff, SCG1, SCG0, OscOff - located in the Status Register - are set. The reaction of the system clock generator module on the status of the bits SCG1, SCG0 and OscOff with its four low power modes are described in detail in the system clock generation section.

Enter interrupt routine
The interrupt routine is entered and processed if an enabled interrupt wakes-up the MSP430:
- The SR and PC are stored onto the stack with the content present at interrupt event.
- Subsequently the operation mode control bits OscOff, SCG1 and CPUOff are cleared in Status Register automatically.

Return from interrupt
Two different ways back from interrupt service routine to continue flow of operation are practicable:
- Return with set low power mode bits
  When returning from the interrupt the program counter points to the next instruction. The instruction pointed to is not executed since the restored low power mode stops CPU activity.
- Return with reset low power mode bits
  When returning from the interrupt, the program continues at the address following the instruction which set the OscOff or CPUOff-bit in the Status Register.

3.5.1 Low Power Mode 0, LPM0

Low power mode 0 is selected if the appropriate bit CPUOff in the status register is set. Immediately after the bit is set the CPU stops operation and the normal operation of the system core is stopped. The operation of the CPU is halted until any interrupt request or reset is effective. All internal bus activities are stopped. The system clock generator is not affected and the clock signals MCLK and ACLK are active pending on the state of the other three bits, SCG0, SCG1 and OscOff in the status register.

LPM0, Peripheral Modules Operation with Bit SCG1 reset

Those peripherals are active that are enabled and clocked with the MCLK or ACLK signal. All pins of I/O ports and the RAM/registers are unchanged. Wake-up is possible by all enabled interrupts.

LPM0, Peripheral Modules Operation with Bit SCG1 set

Those peripherals are active that are enabled and clocked with the ACLK signal. Peripherals that are operating with the MCLK signal are inactive because the MCLK signal is inactive. All pins of I/O ports and the RAM/registers are unchanged. Wake-up is possible by those enabled interrupts coming from system clock (MCLK) independent sources.

```
; === Main program flow with switch to CPUOff Mode ========================
;
                BIS   #18h,SR     ; Enter LPM0 + enable general interrupt GIE.
                                  ; The PC is incremented during execution of this in-
                                  ; struction and points to the consecutive program step.
                .....             ; The program continues here if CPUOff bit is reset
                                  ; during the interrupt service routine

; === Interrupt service routine ============================================
                .....
                .....
                RETI              ; RETI restores the same state of CPU before
                interrupt.
                                  ; This is possible because control registers GIE,
                                  ; CPUOff, OscOff, SGC1 and SCG0 are located in the
                                  ; status register which is restored during execution of
                                  ; return-from-interrupt.
```

3.5.2 Low Power Mode 4, LPM4

All activities cease, only the RAM contents, Port and registers are maintained. Wake-up is only possible by enabled external interrupts.

Before activating LPM4, the software flow should consider the conditions that are applied to the system during the period of this low power mode. The two and most important figures that should be looked at are the environment situation with the influence at the DCO and the clocked operation conditions. The environment situation

defines whether the actual value of the frequency integrator should be hold of corrected. A correction can be intended when ambient conditions would increase the system frequency drastically. When clocked operation is applied it should be considered that the loop can lose control over the frequency if there remaining time slot is unsufficient to hold the closed loop in the correct operating range.

The following example shows the entering of the low power mode 4 (OscOff):

```
BIS   #B8h,SR   ; Enter LPM4 + enable general interrupt GIE.
                ; The CPU must be switched of with LPMs.
                ; Additionally the DCO operation is enabled.
                ; When during the interrupt routine the LPM4 is going
                ; to be disrupted, DCO operation is prepared.
.....           ; The program continues here if OscOff bit is reset
.....           ; during the interrupt service routine.
.....           ; Otherwise it retains in OscOff mode
```

3.6 Basic Hints for Low Power Applications

There are some general basics that should looked at when the current consumption is critical part of a system application:

- Tie unused FETI input to V_{SS}
- Switch-off the Analog Generator in the LCD if convenient
- Do not tie the JTAG inputs TMS, TCK and TDI to V_{SS}

4 Memory Organization

The MSP430 family's memory space is configured in a "von-Neumann Architecture" and has code memory (ROM, EPROM, RAM) and data memory (RAM, EEPROM, ROM) in one adress space using a unique address and data bus.

All the physically separated memory areas, the internal areas for ROM, RAM, SFRs and peripheral modules and the external memory, are mapped into the common address space. The total addressable memory space provided is 64KB in the small memory model and 1MB in the large memory model. The small memory model use a linear address space while in the large memory model the address space is arranged in sixteen segments of 64KB at code access and 64 pages of 16KB at data access.

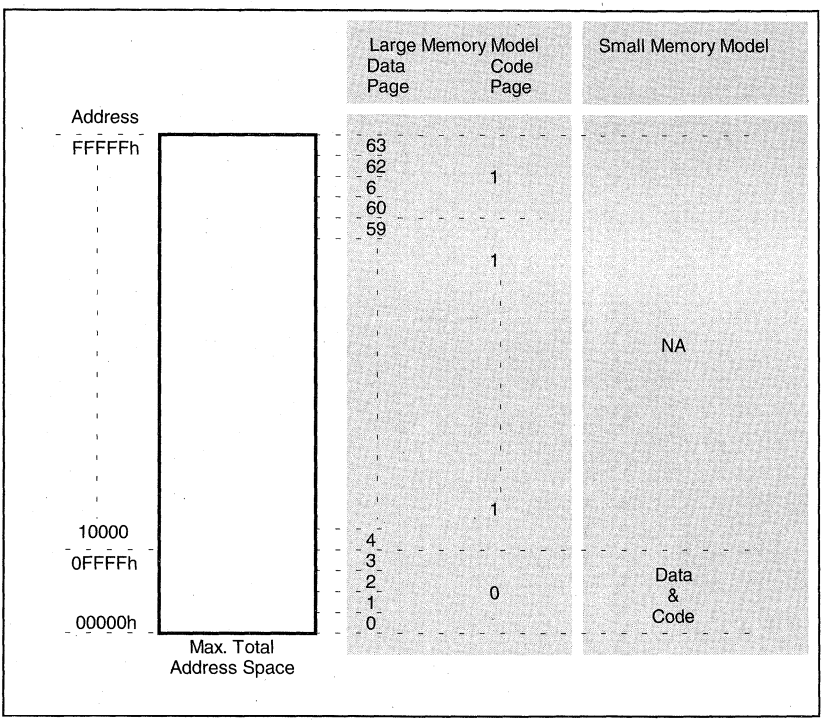


Figure 4.1: Total Memory Address Space

Devices with a memory configuration of 64KB or less use the small memory model with basic address range of the lowest 64KB and do not care about code segments and data pages.

The configuration according to the small memory model and data bus width is shown below:

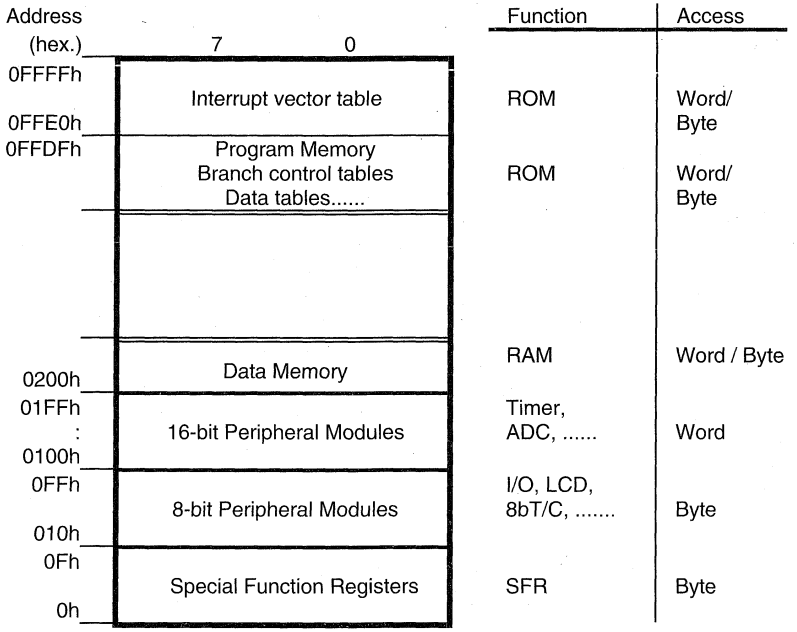
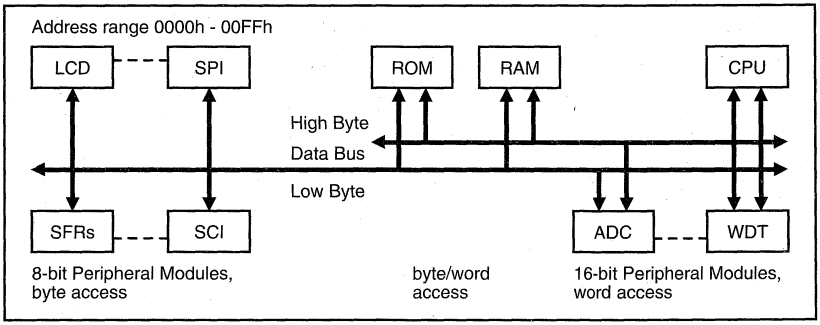


Figure 4.2: Memory Map of Basic Address Space

The Data Bus is 16-bit or 8-bit wide. For those modules that can be accessed with word data it is always 16 bits and for the other modules it is 8 bits and they should only be accessed with byte instructions. The Program Memory (ROM), the Data Memory (RAM) can be accessed with byte or word instructions. Parts of peripheral modules are realized as 16-bit wide or 8-bit wide modules. The access should use the proper instructions, byte or word.

Many peripheral modules are connected to the CPU with an 8-bit Memory Data Bus (MDB), the 5 least significant bits of the Memory Address Bus (MAB) plus two Module Enable signals (ME), two interrupt control/request lines and a power-up signal.

The access to these modules should be always performed using byte instruction formats. Other 16-bit peripheral modules are connected to the 16-bit MDB with full supporting word processing and should use word instruction format for any access.



4.1 Data in the Memory

Bytes are located at even or odd addresses. Words are located in the ascending memory locations aligned to even addresses: the low byte is at the even address followed by the high byte at the next odd address.

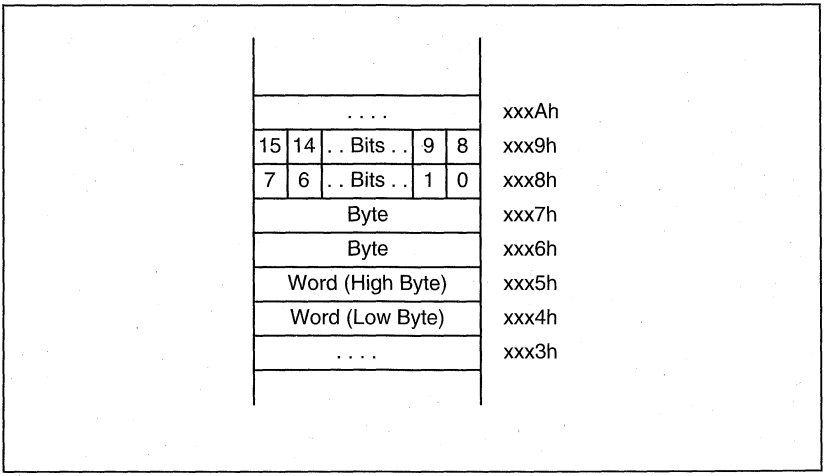


Figure 4.3: Bit, Byte and Word in a byte organized Memory

4.2 Internal ROM Organisation

Various ROM-sizes up to 64K bytes are possible. The common address space is shared with special function registers, peripheral module registers, data and code memory. The special function registers and peripheral modules are mapped into the address range starting with 0 and up to 01FFh. The remaining address space 0200h to 0FFFFh is shared by data and code memory.

The start address for all different sizes of ROM is at the same address 0FFFEh. The interrupt vector table starts with highest priority at this highest ROM word address too. The program counter and hence the flow of instructions is in the opposite direction - from lower addresses toward higher addresses. The program counter is increased by two, four or six according to the address mode used - program flow control instruction Jumps, branches and calls excluded.

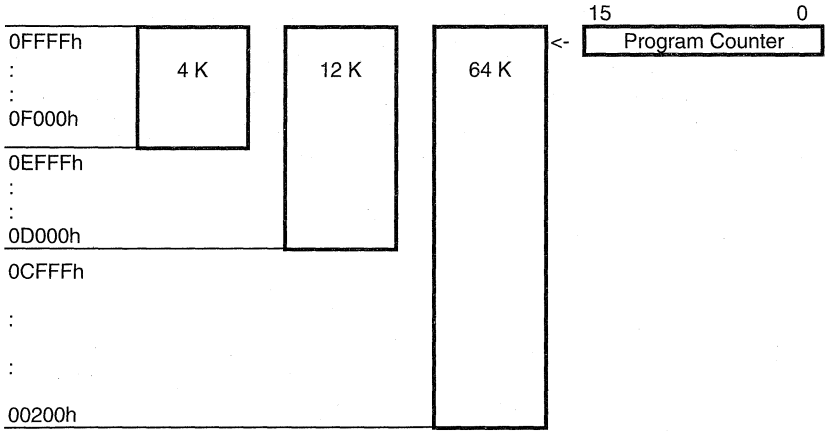


Figure 4.4: ROM Organization

The interrupt vectors and the power-up vector are located in the ROM, starting at address 0FFFEh. The vectors contain the 16-bit addresses of the appropriate interrupt handler instruction sequence.

4.2.1 Processing of ROM Tables

The MSP430 architecture allows the storage of large tables in the ROM. To access these tables, all word and byte instructions can be used. This offers various advantages to flexible and ROM saving programming constructions:

- Storage of an Output-PLA for display character conversion inside the ROM
- As many OPLA-terms as needed (no restriction to n terms)

- OTP version includes automatically OPLA programmability
- Computed table accesses (eg. for a bar graph display)
- Table supported program flows.

The processing of tables is a very important feature, which allows very fast and clear programming. Especially for sensor applications it is advantageous to have the sensor data in tables eg. for linearization, compensation etc.

4.2.2 Computed Branches and Calls

Computed branches and subroutine calls are possible using standard instructions. The CALL and BR instructions use the same addressing modes as the other instructions (see programming examples).

The addressing modes allow indirect-indirect addressing, ideally suited for computed branches and calls. The fully use of this programming technique permits a program structure different to conventional 8- and 16-bit controllers. A lot of routines can be handled easily using software status handling instead of 'Flag' type program flow control .

The computed branches and subroutine calls are valid within a 64KB code segment.

4.3 RAM and Peripheral Organization

The entire RAM can be accessed in byte or word data using the appropriate instruction suffix. The peripheral modules are located in two different address spaces:

- the special function registerss are byte oriented by hardware and mapped into the address space from 0h up to 0Fh
- the peripheral modules that are byte oriented by hardware are mapped into the address space from 010h up to 0FFh
- and peripheral modules that are word oriented by hardware are mapped into the address space from 100h up to 01FFh

4.3.1 RAM

The RAM can be used for both, code and data memory. Code accesses are always made on even byte addresses.

The suffix at the instruction memonic defines the access of the data as word or byte data.

Example:

```
ADD.B    &TCDATA,TCSUM_L                          ;Byte acess
ADDC.B   TCSUM_H                                   Byte acess
ADD      R5,SUM_A     ≡    ADD.W    R5,SUM_A;    ;Word acess
ADDC     SUM_B        ≡    ADDC.W   SUM_A        ;Word acess
```

A Word consists of two bytes, a Highbyte (bit 15 to bit 8) and a Lowbyte (bit 7 to bit 0) and should always be aligned to even addresses.

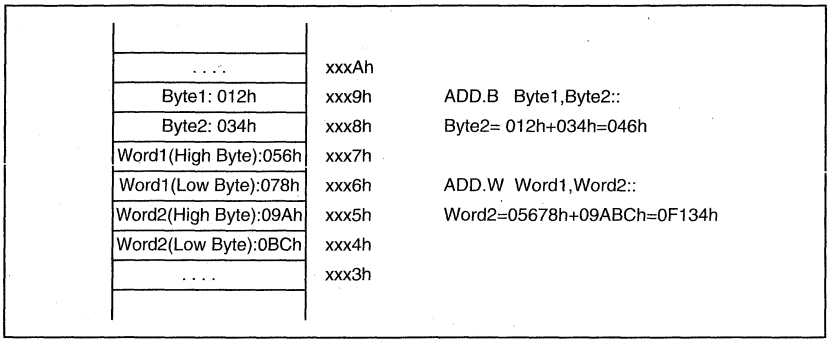


Figure 4.5: Byte and Word Operation

All operations on Stack and PC are word operations and use even aligned memory addresses.

Word-to-word and byte-to-byte operations are performed fully correct; both the results of operation and the status bit information.

Word-word operation:

R5 = 0F28Eh
EDE .EQU 0212h
Mem(0F28Eh) = 0FFFEh
Mem(0212h) = 00112h

ADD @R5,&EDE

Mem(0212h) = 00110h
C = 1, Z = 0, N = 0

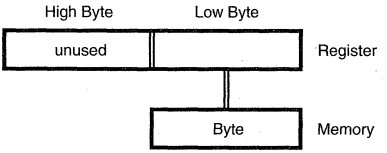
Byte-byte operation

R5 = 0223h
EDE .EQU 0202h
Mem(0223h) = 05Fh
Mem(0202h) = 043h

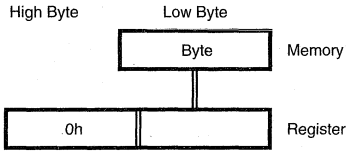
ADD.B @R5,&EDE

Mem(0202h) = 0A2h
C = 0, Z = 0, N = 1

Register-Byte operation:



Byte-Register operation:



Example Register-Byte operation
R5 = 0A28Fh
R6 = 0203h
Mem(0203h) = 012h

ADD.B R5,0(R6)

```
  08Fh
+ 012h
  0A1h
```

Highbyte is 0
Mem(0203h) = 0A1h
C = 0, Z = 0, N = 1

(Lowbyte of register)
+ (addressed byte)
->(addressed byte)

Example Byte-Register operation
R5 = 01202h
R6 = 0223h
Mem(0223h) = 05Fh

ADD.B @R6,R5

```
  05Fh
+ 002h  ;Lowbyte of R5
  061h  ;-> store into R5 -
```

R5 = 061h
C = 0, Z = 0, N = 0

(addressed byte)
+ (Lowbyte of register)
->(Lowbyte of register, zero to Highbyte)

Note: Word-Byte operation

Word-Byte or Byte-Word operations on memory data are n o t supported. Each register-byte and byte-register operation is performed as a byte operation.

4.3.2 Peripheral Modules - Address Allocation

All peripheral modules are accessed and controlled by the software. All instructions are approved for the data interchange operation. Since there are modules using physically the MDB with its word construction and modules that use only the eight least significant bits, the address space from 0100 to 01FFh is reserved for word modules and the address space from 00h to 0FFh is reserved for byte modules.

Peripheral modules mapped into the word address space should be accessed with word instructions (e.g. MOV R5,&WDTCTL). Peripheral modules mapped into the word address space should be accessed with byte instructions (MOV.B #1,&TCCTL).

The addressing of both is made via the absolute addressing mode or via the 16-bit working registers using the indexed, indirect or indirect autoincrement addressing mode.

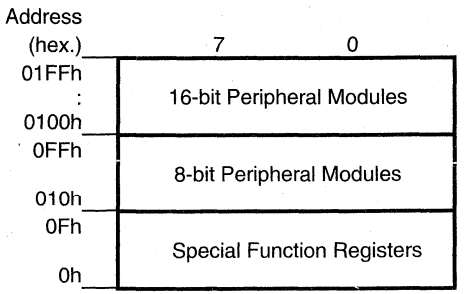


Figure 4.6: RAM/peripheral organization example

Word modules

Word modules are peripherals that are connected to the complete 16-bit MDB.

Access to word modules is always in word format and byte access is not supported since the hardware is constructed for word operation only.

The peripheral file address space is organized in sixteen frames and each frame represents eight words.

Address	Description
1F0h - 1FFh	reserved
1E0h - 1EFh	reserved
130h - 13Fh	reserved
120h - 12Fh	Watchdog Timer
110h - 11Fh	Analog-to-Digital Converter
100h - 10Fh	reserved

Figure 4.7: Peripheral File Address Map - Word Modules

Byte modules

Byte modules are peripherals that are connected to the reduced (eight LSB) MDB. The access to byte modules is always a byte access. The hardware in the peripheral byte modules takes the LowByte - the least significant bits - along with a write operation.

Byte instructions operates on byte modules without any restriction. Read access to the data of a peripheral byte module with word instructions results in unpredictable data on the Highbyte. Word data are written into a byte module by writing the LowByte to the appropriate peripheral register and ignoring the HighByte.

The peripheral file address space is organized in sixteen frames.

Address	Description
00F0h - 00FFh	reserved
00E0h - 00EFh	reserved
00D0h - 00DFh	reserved
00C0h - 00CFh	reserved
00B0h - 00BFh	reserved
00A0h - 00AFh	reserved
0090h - 009Fh	reserved
0080h - 008Fh	reserved
0070h - 007Fh	reserved
0060h - 006Fh	reserved
0050h - 005Fh	System Clock Generator registers
0040h - 004Fh	Basic Timer, 8-bit Timer/Counter registers
0030h - 003Fh	LCD registers
0020h - 002Fh	reserved
0010h - 001Fh	Digital I/O Port 0 control registers
0000h - 000Fh	Special Function Registers

Figure 4.8: Peripheral File Address Map - Byte Modules

4.3.3 Peripheral Modules - Special Function Registers SFRs

The system configuration and the individual reaction of the peripheral modules to processor operation modes are mainly defined in Special Function Registers. The Special Function Registers are located in the lower address range and are realized in **byte** manner. SFRs should be only accessed with byte instructions. Even if specific SFR bits share the same address space they can be implemented physically within the associated module.

Address	Data Bus 7 0
000Fh	Not defined / implemented yet
000Eh	:
000Dh	:
000Ch	:
000Bh	:
000Ah	:
0009h	:
0008h	:
0007h	:
0006h	:
0005h	Module enable 2; ME2.2
0004h	Module enable 1; ME1.1
0003h	Interrupt flag reg. 2; IFG2.x
0002h	Interrupt flag reg. 1; IFG1.x
0001h	Interrupt enable 2; IE2.x
0000h	Interrupt enable 1; IE1.x

Figure 4.9: Special Function Register Address Map

The different devices of the MSP430 Family support SFRs with the correct logic and function within the individual modules. Each module can be enabled individually to access the interrupt function and the operation. Full software control of these configuration bits enables the application software to react to system requirements on interrupt enable mask.

The power consumption of the system is influenced by the amount and function of the enabled modules. Disabling a module from the actual operation mode reduces power consumption while other parts of the controller are fully active. Two parts can not be disabled: ROM and RAM. The processor core can be switched to disabled mode - CPUOff Mode - with all internal functions disabled: CPU and bus activities are stopped.

5 CPU, 16bit

The equal width of the PC and the working registers allows new features, for example seven addressing modes.

The "von-Neumann-Architecture" used in the MSP430 has RAM and ROM in one adress space using a single address and data bus.

5.1 CPU Registers

Fourteen 16-bit registers (R0, R1, R4 to R15) are used for data and addresses. These registers are implemented in the CPU. They are able to address up to 64KBytes (ROM, RAM, EERAM, Peripherals,...) without any segmentation. The complete CPU register set is shown below. The registers which are used for special purposes are marked. The registers R0, R1, R2 and R3 are restricted in their common use due to their special functions described later.

Program Counter PC	R0
Stack Pointer SP	R1
Status Register SR	R2
Constant Generator CG1	
Constant Generator CG2	R3
Working Register R4	R4
Working Register R5	R5
:	:
:	:
Working Register R13	R13
Working Register R14	R14
Working Register R15	R15

Table 5.1: Register by functions

5.1.1 The Program Counter PC

The 16-bit Program Counter PC defines which instruction will be executed next. Each instruction uses an even number of bytes, two, four or six bytes. The instruction accesses are performed on word boundaries and so the program counter is aligned to even addresses. The PC is double-incremented during the fetch cycle of an instruction: it points to the word following the currently executed instruction. This makes two additional addressing modes possible (Immediate Mode, Symbolic Mode), which use the word following the instruction for information.

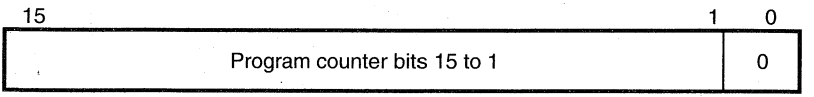


Figure 5.1: Program Counter PC

5.1.2 The System Stack Pointer SP

The system Stack Pointer SP should always be aligned to even addresses since the stack is accessed with word data during interrupt request service. The system Stack Pointer SP is used by the CPU for the storage of the return addresses of subroutine calls and interrupts. It uses a pre-decrement, post-increment scheme. This scheme has the advantage that the item on the top of the stack (TOS) is avalaible. The SP may be used by the user's software (PUSH and POP instructions) but it should be remembered that the CPU uses the Stack Pointer too.

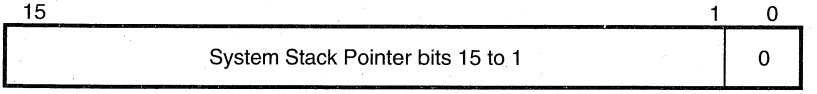


Figure 5.2: System Stack Pointer SP

Note: Software stack pinter using general purpose registers

The general purpose registers R4 to R15 can be used as SW-stackpointers.

Pushing item onto a **word SW-stack** controlled by Rn:

```
DECD    Rn          ; Double-decrement SW-SP Rn
MOV     item,0(Rn)  ; PUSH item on SW-stack
```

Popping item off a SW-stack is made by:

```
MOV     @Rn+,item   ; POP ITEM off SW-stack
```

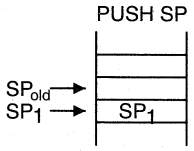
Pushing item onto a **byte SW-stack** controlled by Rm:

```
DEC     Rm          ; Decrement SW-SP Rm
MOV.B   item,0(Rn)  ; PUSH item on SW-stack
```

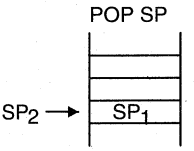
Popping item off a byte SW-stack is made by:

```
MOV.B   @Rn+,item   ; POP ITEM off SW-stack
```

Special condition on **PUSH** and **POP** of the System Stack Pointer.



The Stack Pointer is not changed after PUSH SP instruction



The Stack Pointer SP is loaded with the data of the memory pointed to by SP before executing POP SP instruction

After the sequence

```
PUSH SP          ; SP1 is stack pointer after 1. inst.
  |
  |
POP  SP          ; SP2 is stack pointer after 2. inst.
```

the Stack Pointer is two bytes lower than before this sequence.

Examples for System Stack Pointer addressing (refer to figure Stack Usage):

```
MOV   SP,R4      ; #0xxxh - 4  -> R4
MOV   @SP,R5     ; Item I3 (TOS) -> R5
MOV   2(SP),R6   ; Item I2 -> R6
MOV   R7,0(SP)   ; overwrite TOS with R7
MOV   R8,4(SP)   ; modify item I1
PUSH  R12        ; store R12 in address 0xxxh - 6; SP points to same address
POP   R12        ; restore R12 from address 0xxxh - 6; SP points to 0xxxh - 4
MOV   @SP+,R5    ; item I3 -> R5 (popped from Stack); same as POP instruction
PUSH  #1
POP   R8
```

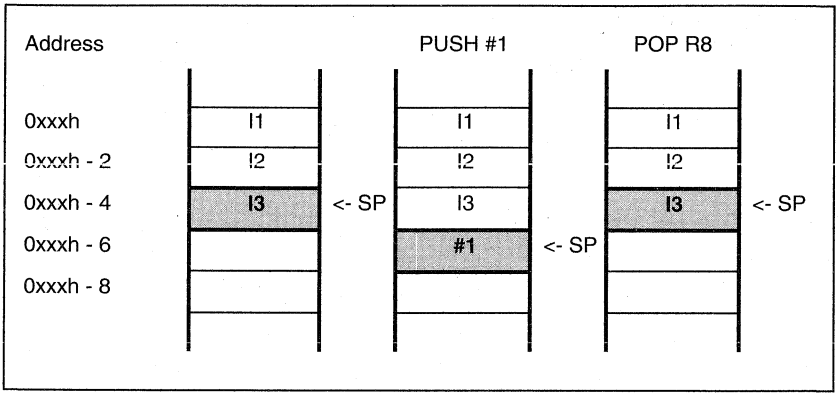


Figure 5.3: Stack Usage

5.1.3 The Status Register SR

The Status Register SR contains the CPU status bits:

- **V** Overflow Bit
- **SCG1** System Clock Generator Control Bit 1
- **SCG0** System Clock Generator Control Bit 0
- **OscOff** Crystal Oscillator Off Bit
- **CPUOff** CPU Off Bit
- **GIE** General Interrupt Enable Bit
- **N** Negative Bit
- **Z** Zero Bit
- **C** Carry Bit

15 ... 9	8	7							0
reserved for future enhancements	V	SCG1	SCG0	OscOff	CPUOff	GIE	N	Z	C
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Figure 5.4: Status Register SR

Description of status bits

- **Overflow Bit (V):** Set if the result of an arithmetic operation overflows the signed variable range. It is valid for both data formats, byte and word:

  ADD(.B), ADDC(.B) — Set when: Positive + Positive = Negative; Negative + Negative = Positive, otherwise reset

SUB(.B), SUBC(.B),CMP(.B): Set when:
Positive - Negative = Negative
Negative - Positive = Positive
otherwise reset

- **SCG1, SCG0:** These bits control four activity states of the system clock generator and therefore influence the operation of the processor system.
- **Oscillator Off:** If set, the Crystal Oscillator enters the Off Mode: All activities cease, the RAM contents, Port and registers are maintained. Wake-up is possible only by enabled external interrupts when GIE is set and from NMI. This bit should **n o t** be set without simultanously setting CPUOff bit.
- **CPU Off:** If set, the CPU enters the Off Mode: All activities ceases, the RAM, Port and registers and specially enabled peripherals e.g. Basic Timer, UART ... stay active. Wake-up is possible by all enabled interrupts.
- **GIE Bit (GIE):** If set, all enabled interrupts are handled. If reset all interrupts are disabled. The GIE Bit is cleared by interrupts and restored by the RETI instruction. It can be also changed by appropriate instructions.
- **Negative Bit (N):** Set if the result of an operation is negative.
  Word operations: Negative bit is set to the value of bit 15 of the result.
  Byte operations: Negative bit at is set to the value of bit 7 of the result.
- **Zero Bit (Z):** Set if the result of an operation is 0, cleared if the result is not 0.
- **Carry Bit(C):** Set if the result of an operation produced a carry, cleared if no carry occured.
  Word operation: The carry is as the result of the word operation.
  Byte operation: The carry is as the result of the byte operation.
  Some instructions have the carry bit modified with the inverted zero bits.

Note: Status bits V, N, Z and C

The status bits V, N, Z and C are modified only with the appropriate instruction. Please see the detailed description of the instruction set, MSP430 Software User's Guide.

5.1.4 The Constant Generator Registers CG1 and CG2

The most often used constants can be generated with the constant registers R2 and R3 without occupying an additional 16-bit-word. The used constant for immediate values is defined by the addressing bits As:

Register	As	constant	remarks
R2	00	- - - - -	Register mode
R2	01	(0)	absolute address mode
R2	10	00004h	+4, bit processing
R2	11	00008h	+8, bit processing
R3	00	00000h	0, word processing
R3	01	00001h	+1
R3	10	00002h	+2, bit processing
R3	11	0FFFFh	-1, word processing

Table 5.2: Values of constant generators CG1, CG2

The major advantages are allied with the use of this type of constant generation:
- No special instructions
- No additional word for the seven most used constants
- Shorter instruction cycles time: direct access without use of MDB

The assembler uses the R2 or R3 modes automatically, if one of the six constants is used in immediate mode as a source operand. The Status Register SR/R2 - used as source or destination register - can be used in register mode only. The remaining combinations of address bits As are used to support absolute address mode and bit processing without adding additional code. Register R2 and R3 used in the 'constant mode' cannot be addressed explicitly; they act just like a source only register.

The Constant Generator Registers allow the emulation of several instructions by other ones. The CPU is much simpler this way. Only 27 instructions are needed for the complete instruction set. For example the Single Operand Instruction:

```
CLR     dst
```

is emulated by the Double Operand Instruction with the same length:

```
MOV     R3,dst
or equivalent
MOV     #0,dst
```

where #0 is replaced by the assembler with R3 used with As = 00:
- one word instruction
- no additional control operation or hardware within CPU
- register addressing mode for source: no extra fetch cycle for constants (#0).

5.2 Addressing modes

All seven addressing modes for the source operand and all four addressing modes for the destination operand can address the complete address space. The bit numbers show the contents of the As and Ad mode bits.

As/Ad	Addressing Mode	Syntax	Description
00/0	Register Mode	Rn	Register contents are operand
01/1	Indexed Mode	X(Rn)	(Rn + X) points to the operand. X is stored in the next word
01/1	Symbolic Mode	ADDR	(PC + X) points to the operand. X is stored in the next word. Indexed Mode X(PC) is used
01/1,	Absolute Mode	&ADDR	The word following the instruction contains the absolute address.
10/-	Indirect Register Mode	@Rn	Rn is used as a pointer to the operand
11/-	Indirect Autoincrement	@Rn+	Rn is used as a pointer to the operand. Rn is incremented afterwards
11/-	Immediate Mode	#N	The word following the instruction contains the immediate constant N. Indirect Autoincrement Mode @PC+ is used

Note: Addressing modes

The addressing modes using the PC as the working register use the normal effects of the addressing modes. The special addressing modes are caused by the pointing of the PC to the ROM word following the currently executed instruction.

The seven addressing modes are explained in detail by examples. Most of the examples show the same addressing modes for source and destination, but any valid combination of source and destination addressing modes is possible with an instruction.

5.2.1 Register mode

Assembler Code	Content of ROM
MOV R10,R11	MOV R10,R11

Length: 1 or 2 word

Operation: Move the content of R10 to R11. R10 is not affected.

Comment: Valid for source and destination

Example: MOV R10,R11

	Before		After
R10	0A023h	R10	0A023h
R11	0FA15h	R11	0A023h
PC	PC_{old}	PC	$PC_{old} + 2$

Note: Data in registers

Since the data in the registers are word data, any operation register-register should be a word operation and word instructions should be used.

5.2.2 Indexed mode

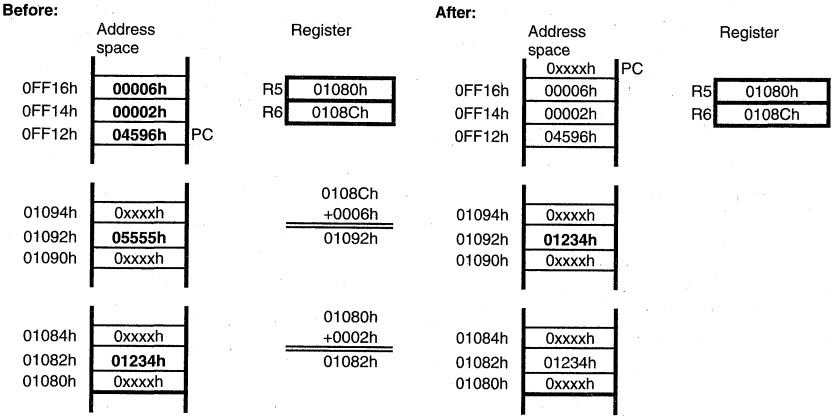
Assembler Code	Content of ROM
MOV 2(R5),6(R6)	MOV X(R5),Y(R6) X = 2 Y = 6

Length: 2 or 3 words

Operation: Move the contents of the source address (contents of R5 + 2) to the destination address (contents of R6 + 6). The source and destination registers (R5 and R6) are not affected. With Indexed mode, the PC is incremented automatically so that program execution continues with the next instruction.

Comment: Valid for source and destination

Example: MOV 2(R5),6(R6):



5.2.3 Symbolic mode

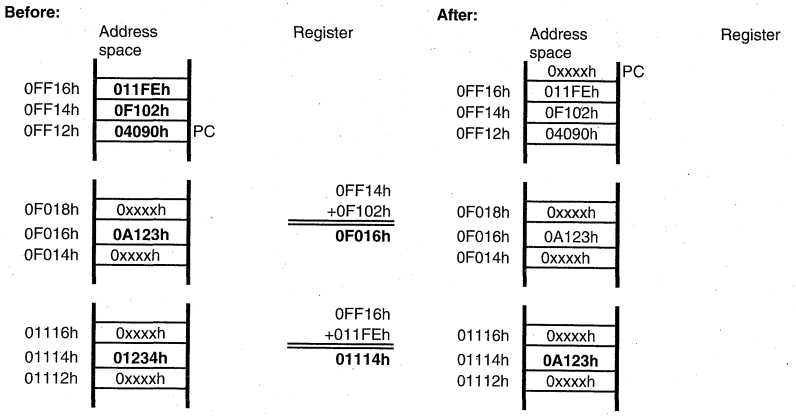
Assembler Code	Content of ROM
MOV EDE,TONI	MOV X(PC),Y(PC) X = EDE - PC Y = TONI - PC

Length: 2 or 3 words

Operation: Move the contents of the source address EDE (contents of PC + X) to the destination address TONI (contents of PC + Y). The words after the instruction contain the differences of the PC and the source or destination addresses. The assembler computes and inserts the offsets X and Y automatically. With Symbolic mode, the PC is incremented automatically so that program execution continues with the next instruction.

Comment: Valid for source and destination

Example:
```
MOV EDE,TONI      ;Source address EDE=0F016h,
;dest. address TONI=01114h
```



5.2.4 Absolute mode

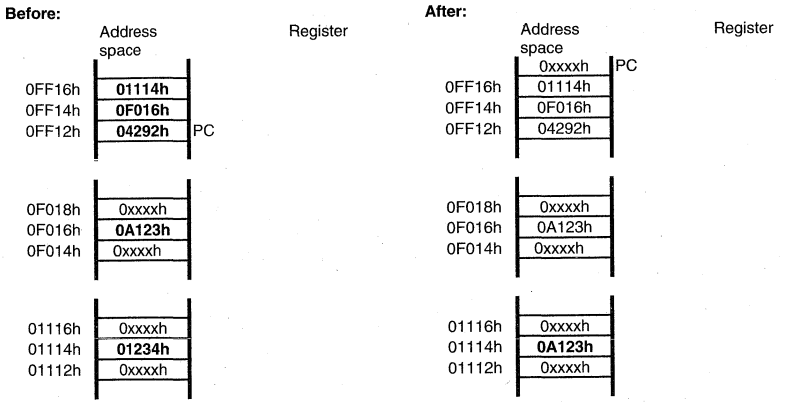
Assembler Code	Content of ROM
MOV &EDE,&TONI	MOV X(0),Y(0)
	X = EDE
	Y = TONI

Length: 2 or 3 words

Operation: Move the contents of the source address EDE to the destination address TONI . The words after the instruction contain the absolute address of the source resp. destination addresses. With absolute mode, the PC is incremented automatically so that program execution continues with the next instruction.

Comment: Valid for source and destination

Example:
```
MOV  &EDE,&TONI    ; Source address EDE=0F016h,
                   ; dest. address TONI=01114h
```



The main use of this address mode is for hardware peripheral modules that are located at an absolute, fixed address. These should be addressed with absolute mode to ensure software transportability e.g. position independent code (PIC) programming techniques. Absolute mode always uses code segment 0.

5.2.5 Indirect mode

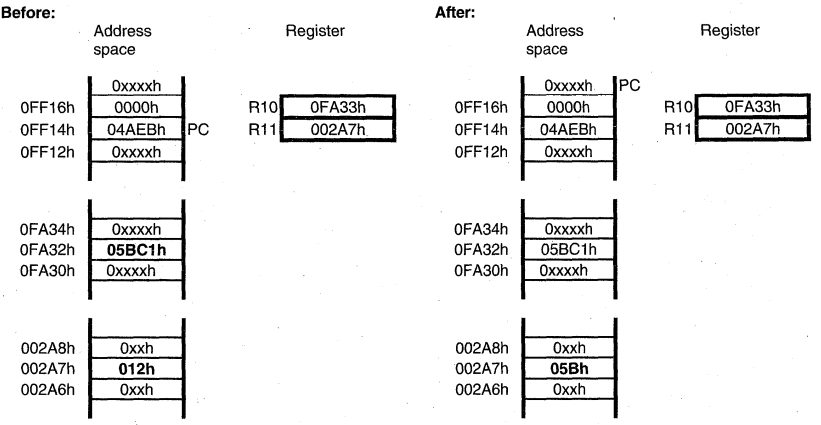
Assembler Code	Content of ROM
MOV @R10,0(R11)	MOV @R10,0(R11)

Length: 1 or 2 word(s)

Operation: Move the contents of the source address (contents of R10) to the destination address (contents of R11). The registers are not modified.

Comment: Valid only for source operand. Substitute for destination operand is 0(Rd).

Example: MOV.B @R10,0(R11)



5.2.6 Indirect autoincrement mode

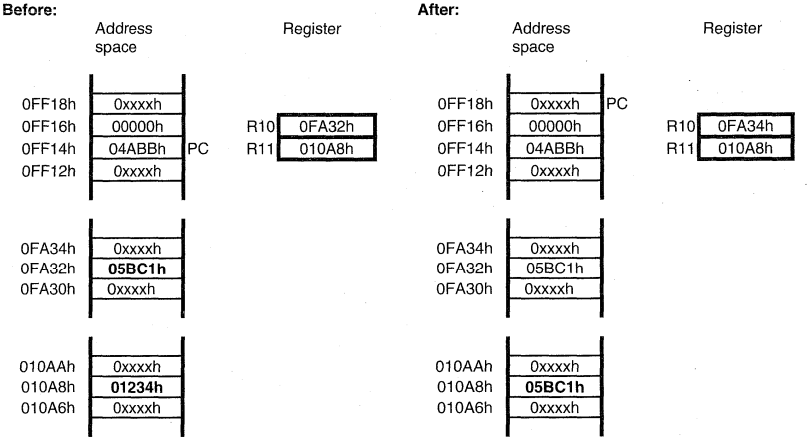
Assembler Code	Content of ROM
MOV @R10+,0(R11)	MOV @R10+,0(R11)

Length: 1 or 2 word(s)

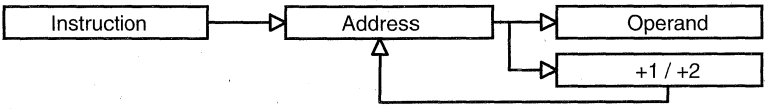
Operation: Move the contents of the source address (contents of R10) to the destination address (contents of R11). The register R10 is incremented by one (byte operation) or two (word operation) after the fetch: It points to the next address now without any overhead. This is very useful for table processing.

Comment: Valid only for source operand. Substitute for destination operand is 0(Rd) plus second instruction INCD Rd.

Example: MOV @R10+,0(R11)



The autoincrement of the registers' content is done after the operand is fetched for performing the operation.



5.2.7 Immediate mode

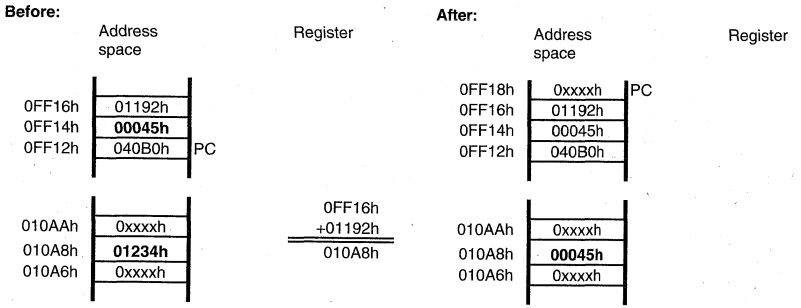
Assembler Code	Content of ROM
MOV #45,TONI	MOV @PC+,X(PC) 45 X = TONI - PC

Length: 2 or 3 words
It is 1 word less if a constant of CG1 or CG2 can be used.

Operation: Move the immediate constant 45 which is contained in the word following the instruction to the destination address TONI. When fetching the source, the PC points to the word after the instruction and moves the contents to the destination.

Comment: Valid only for source operand.

Example: MOV #45,TONI



5.2.8 Clock cycles, Length of Instruction

The operating speed of the CPU is independent from individual instructions. It depends on the instruction format and the addressing modes. The number of clock cycles refer to the internal oscillator frequency.

Format I Instructions

AddressMode		#of cycles	Length of instruction	Example
As	Ad			
00, Rn	0, Rm	1	1	MOV R5,R8
	0,PC	2	1	BR R9
00, Rn	1, x(Rm)	4	2	ADD R5,3(R6)
	1, EDE		2	XOR R8,EDE
	1, &EDE		2	MOV R5,&EDE
01, x(Rn)	0, Rm	3	2	MOV 2(R5),R7
01, EDE			2	AND EDE,R6
01, &EDE				MOV &EDE,R8
01, x(Rn)	1, x(Rm)	6	3	ADD 3(R4),6(R9)
01, EDE	1, TONI		3	CMP EDE,TONI
01, &EDE	1, &TONI		3	MOV 2(R5),&TONI
				ADD EDE,&TONI
10, @Rn	0, Rm	2	1	AND @R4,R5
10, @Rn	1, x(Rm)	5	2	XOR @R5,8(R6)
	1, EDE		2	MOV @R5,EDE
	1, &EDE		2	XOR @R5,&EDE
11, @Rn+	0, Rm	2	1	ADD @R5+,R6
	0, PC	3	1	BR @R9+
11, #N	0, Rm	2	2	MOV #20,R9
	0, PC	3	2	BR #2AEh
11, @Rn+	1, x(Rm)	5	2	MOV @R9+,2(R4)
11, #N	1, EDE		3	ADD #33,EDE
11, @Rn+	1, &EDE		2	MOV @R9+,&EDE
11, #N			3	ADD #33,&EDE

Note: DADD instruction execution cycles

The DADD instruction needs 1 extra cycle in '201 configuration.

Format II Instructions

Address Mode $A_{(s/d)}$	#of cycles		Length of instruction [words]	Example
	RRA RRC SWPB SXT	PUSH/ CALL		
00, Rn	1	3/4	1	SWPB R5
01, x(Rn) 01, EDE 01,&EDE	4 4	5 5	2 2	CALL 2(R7) PUSH EDE SXT &EDE
10, @Rn	3	4	1	RRC @R9
11, @Rn+ see Note 11, #N	3	4/5	1 2	SWPB @R10+ CALL #81h

Note: Instruction Format II immediate mode

Instructions RRA, RRC, SWPB and SXT should not be used with the immediate mode in the destination field. This would result in unpredictable program operation.

Format III Instructions

Jxx - instructions need all the same #-of-cycles independent of a successfull Jump or not.

Clock Cycle:	2 Cycle
Length of Instruction:	1 word

Miscellanous Instructions or Operations

RETI	Clock Cycle:	5 Cycle
	Length of instruction:	1 word
Interrupt	Clock Cycle:	6 Cycle
WDTreset	Clock Cycle:	4 Cycle
Reset (RST/NMI)	Clock Cycle:	4 Cycle

5.3 Instruction set overview

The following gives a short overview to the instruction set.

The effects of an instruction to the Status Register Bits are shown below:

*	The Status Bit is affected
-	The Status Bit is not affected
0	The Status Bit is cleared
1	The Status Bit is set

The source and destination parts of an instruction are defined by two fields each (the addressing modes are described above):

src	The source operand defined by As and S-reg
dst	The destination operand defined by Ad and D-reg
As	The addressing bits responsible for the addressing mode used for the source src
S-reg	The used Working Register for the source src
Ad	The addressing bits responsible for the addressing mode used for the destination dst
D-reg	The used Working Register for the destination dst
B/W	Byte or word operation: 0: word operation 1: byte operation

Note: Destination

The destination can be anywhere in the 64kByte address range. Operations that write data back should use address ranges into those data can be written, otherwise the data is lost.

5.3.1 Double operand instructions

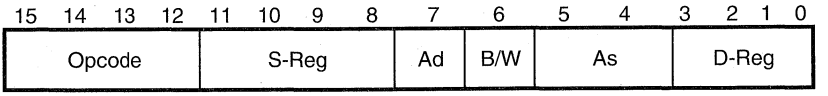


Figure 5.5: Double Operand Instruction Format

			Status Bits V	N	Z	C
MOV	src,dst	src -> dst	-	-	-	-
ADD	src,dst	src + dst -> dst	*	*	*	*
ADDC	src,dst	src + dst + C -> dst	*	*	*	*
SUB	src,dst	dst + .not.src + 1 -> dst	*	*	*	*
SUBC	src,dst	dst + .not.src + C -> dst	*	*	*	*
CMP	src,dst	dst - src	*	*	*	*
DADD	src,dst	src + dst + C -> dst (dec)	*	*	*	*
AND	src,dst	src .and. dst -> dst	0	*	*	*
BIT	src,dst	src .and. dst	0	*	*	*
BIC	src,dst	.not.src .and. dst -> dst	-	-	-	-
BIS	src,dst	src .or. dst -> dst	-	-	-	-
XOR	src,dst	src .xor. dst -> dst	*	*	*	*

Note: Instructions CMP and SUB

The instructions CMP and SUB are identical except the storage of the result. The same is true for the BIT and the AND instruction.

5.3.2 Single operand instructions

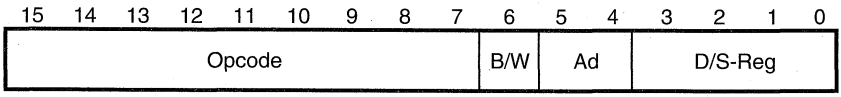


Figure 5.6: Single Operand Instruction Format

			Status Bits V	N	Z	C
RRC	dst	C -> MSB -> .....LSB -> C	*	*	*	*
RRA	dst	MSB -> MSB ->...LSB -> C	0	*	*	*
PUSH	src	SP - 2 -> SP, src -> @SP	-	-	-	-
SWPB	dst	swap bytes	-	-	-	-
CALL	dst	SP - 2 -> SP PC+2 -> stack, dst -> PC	-	-	-	-
RETI		TOS -> SR, SP <- SP + 2 TOS -> PC, SP <- SP + 2	x	x	x	x
SXT	dst	Bit7 -> Bit8 ..... Bit15	0	*	*	*

All addressing modes are possible for the CALL instruction. If the Symbolic Mode (ADDRESS), the Immediate Mode (#N), the Absolute Mode (&EDE) or the Indexed Mode X(Rn)) is used, the instructions have the address information contained in the following word.

5.3.3 Conditional Jumps

The conditional jumps allow program branches relative to the Program Counter. The possible range is from -511 to +512 words relative to the PC state of the Jump instruction. The 10-bit PC offset is treated as a signed 10-bit value which is doubled and added to the Program Counter. The conditional jumps do not affect the Status Bits.

The instruction code fetch and PC increment technique used ends with the formula:

$$PC_{new} = PC_{old} + 2 + PC_{offset} * 2$$

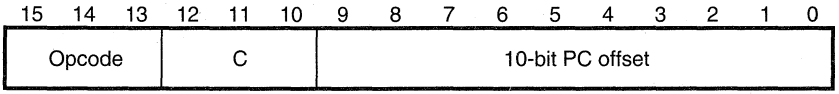


Figure 5.7: Conditional Jump Instruction Format

JEQ/JZ	Label	Jump to Label if Zero-bit is set
JNE/JNZ	Label	Jump to Label if Zero-bit is reset
JC	Label	Jump to Label if Carry-bit is set
JNC	Label	Jump to Label if Carry-bit is reset
JN	Label	Jump to Label if Negative-bit is set
JGE	Label	Jump to Label if (N .XOR. V) = 0
JL	Label	Jump to Label if (N .XOR. V) = 1
JMP	Label	Jump to Label unconditionally

The instruction code fetch and PC increment technique used ends with the formula:

$$PC_{new} = PC_{old} + 2 + PC_{offset} * 2$$

5.3.4 Short form of emulated instructions

The basic instructions together with the constant generator form the emulated instruction which supplies popular instructions. The status bits are set according to the result of the basic instructions.

Mnemonic		Description	Statusbits V	N	Z	C	Emulation	
Arithmetical instructions								
ADC[.W]	dst	Add carry to destination	*	*	*	*	ADDC	#0,dst
ADC.B	dst	Add carry to destination	*	*	*	*	ADDC.B	#0,dst
DADC[.W]	dst	Add carry decimal to destination	*	*	*	*	DADD	#0,dst
DADC.B	dst	Add carry decimal to destination	*	*	*	*	DADD.B	#0,dst
DEC[.W]	dst	Decrement destination	*	*	*	*	SUB	#1,dst
DEC.B	dst	Decrement destination	*	*	*	*	SUB.B	#1,dst
DECD[.W]	dst	Double-Decrement destination	*	*	*	*	SUB	#2,dst
DECD.B	dst	Double-Decrement destination	*	*	*	*	SUB.B	#2,dst
INC[.W]	dst	Increment destination	*	*	*	*	ADD	#1,dst
INC.B	dst	Increment destination	*	*	*	*	ADD.B	#1,dst
INCD[.W]	dst	Increment destination	*	*	*	*	ADD	#2,dst
INCD.B	dst	Increment destination	*	*	*	*	ADD.B	#2,dst
SBC[.W]	dst	Subtract carry from destination	*	*	*	*	SUBC	#0,dst
SBC.B	dst	Subtract carry from destination	*	*	*	*	SUBC.B	#0,dst
Logical instructions								
INV[.W]	dst	Invert destination	*	*	*	*	XOR	#0FFFFh,dst
INV.B	dst	Invert destination	*	*	*	*	XOR.B	#0FFFFh,dst
RLA[.W]	dst	Rotate left arithmetically	*	*	*	*	ADD	dst,dst
RLA.B	dst	Rotate left arithmetically	*	*	*	*	ADD.B	dst,dst
RLC[.W]	dst	Rotate left through carry	*	*	*	*	ADDC	dst,dst
RLC.B	dst	Rotate left through carry	*	*	*	*	ADDC.B	dst,dst
Data instructions (common use)								
CLR[.W]		lear destination	-	-	-	-	MOV	#0,dst
CLR.B		lear destination	-	-	-	-	MOV.B	#0,dst
CLRC		lear carry bit	-	-	-	0	BIC	#1,SR
CLRN		lear negative bit	-	0	-	-	BIC	#4,SR
CLRZ		lear zero bit	-	-	0	-	BIC	#2,SR
POP	dst	Item from stack	-	-	-	-	MOV	@SP+,dst
SETC		Set carry bit	-	-	-	1	BIS	#1,SR
SETN		Set negative bit	-	1	-	-	BIS	#4,SR
SETZ		Set zero bit	-	-	1	-	BIS	#2,SR
TST[.W]	dst	Test destination	0	*	*	*	CMP	#0,dst
TST.B	dst	Test destination	0	*	*	*	CMP.B	#0,dst
Program flow instructions								
BR	dst	Branch to	-	-	-	-	MOV	dst,PC
DINT		Disable interrupt	-	-	-	-	BIC	#8,SR
EINT		Enable interrupt	-	-	-	-	BIS	#8,SR
NOP		No operation	-	-	-	-	MOV	#0h,#0h
RET		Return from subroutine	-	-	-	-	MOV	@SP+,PC

5.3.5 Miscellaneous

No instructions without operands like CPUOff etc. are provided. These functions are switched on or off by setting or clearing of the function bits in the Status Register or the appropriate I/O-register. Others are emulated by Dual Operand Instructions.

Some examples are given below:

```
BIC     #1,SR           ; Clear Carry
MOV     #0,#0           ; No Operation
BIC     #8,SR           ; Disable Interrupts
BIS     #28h,SR         ; Enter OscOff Mode
                        ; + enable gen. interrupt GIE
BIS     #18h,SR         ; Enter CPUOff Mode
                        ; + enable gen. interrupt GIE
BIC     #SVCC,ACTL      ; SWITCH SVCC OFF
```

5.4 Instruction map

The following instruction map is a proposal how to encode the instructions. Room is free for more instructions if needed.

	000	040	080	0C0	100	140	180	1C0	200	240	280	2C0	300	340	380	3C0
0x																
04x																
08x																
0Cx																
10x	RRC	RRC.B	SWPB		RRA	RRA.B	SXT		PUSH	PUSH.B	CALL		RETI			
14x																
18x																
1Cx																
20x	JNE/JNZ															
24x	JEQ/JZ															
28x	JNC															
2Cx	JC															
30x	JN															
34x	JGE															
38x	JL															
3Cx	JMP															
40x..4Cx	MOV, MOV.B															
50x..5Cx	ADD, ADD.B															
60x..6Cx	ADDC, ADDC.B															
70x..7Cx	SUBC, SUBC.B															
80x..8Cx	SUB, SUB.B															
90x..9Cx	CMP, CMP.B															
A0x..ACx	DADD, DADD.B															
B0x..BCx	BIT, BIT.B															
C0x..CCx	BIC, BIC.B															
D0x..DCx	BIS, BIS.B															
E0x..ECx	XOR, XOR.B															
F0x..FCx	AND, AND.B															

Figure 5.8: Core instruction map

6 Oscillator and System Clock Generator

The oscillator and the system clock generator follow the major targets of low system cost and low power consumption.

External component count is reduced down to a commonly used crystal to achieve the target of low system cost. The use of a low frequency crystal and oscillator combined with a multiplier meets system cycle speed and the second target of low power consumption.

Features for current limited applications

Special other features are obviously mandatory in very low power consuming devices that use the different operating modes extended. These features include startup timing, long term frequency stability on voltage, temperature and time, high-stable time base for real time clocks.

Current limited real-time applications demand two conflicting requirements: Low system clock frequency for energy conservation and high system clock frequency for fast reactions onto requesting events. Especially battery based applications are very critical with respect to current consumption. Response to external events or time requests requires typically but occasionally high speed in real-time applications.

A processor clock generator with fast start-up empowers exhaustive use of different power dissipation modes that could theoretically solve this dilemma. On the other hand fast start-up is closely combined with unacceptable low frequency stability. Design with multiple clock sources or different clock operations could take into account the clock requirements of certain peripheral components for real-time applications like low frequency communication, display (e.g.LCD), timers and counters.

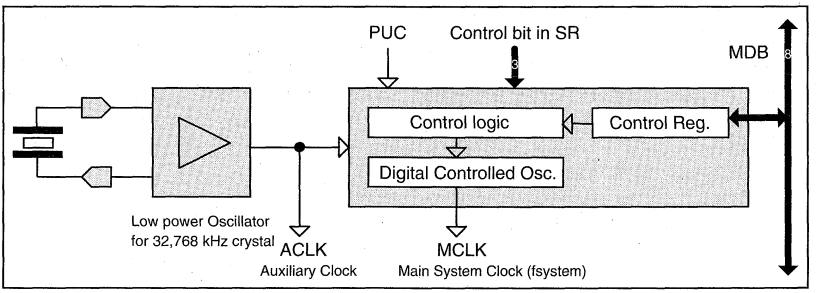


Figure 6.1: Principle of Clock Generation

The output of the low frequency crystal oscillator provides the clock signals for the CPU operation and the peripheral modules. The oscillator of the MSP430 operates with the widely used crystal without any external components.

The different requirements of CPU and modules driven by current consumption objectives oblige use of two clock signals:
- Auxiliary Clock ACLK with crystal's frequency
- System Clock MCLK with higher frequency: N x $f_{crystal}$.

6.1 Crystal Oscillator

The special design of the oscillator supports feature of low current consumption and the use of a 32 768Hz crystal. The crystal is connected to two pins without any other external components. All components for stabilizing the operation state or phase shifter capacitors are integrated.

Two factors dominate the choice of the well-known and widely used watch crystal:
- oscillator and time base for low current consumption
- optimize system costs.

The oscillator starts operating after applying VCC due to reset of the control bit OscOff in the Status Register SR. It can be stopped by setting the OscOff bit.

15	8	7							0
reserved for future enhancements	V	SCG 1	SCG 0	Osc Off	CPU Off	GIE	N	Z	C
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Figure 6.2: Status Register SR

6.2 Processor Clock Generator

The System Clock of controllers has to meet different requirements according to the application and system conditions:
- High frequency to react fast onto system hardware requests or events
- Low frequency to minimize current consumption, EMI, ...
- Stable frequency for timer applications e.g. real time clock RTC
- Low Q-oscillators to enable start-stop operation with 'zero' delay to operation.

All the contrary but essential requests can not be handled whether with high-Q, fast frequency crystals nor with low-Q RC-type oscillators. Proper current consumption and the frequency stability mention the application of a low frequency crystal. The compromise used in the MSP430 is to use a low frequency crystal and to multiply its frequency up to the nominal operating range:

$$f_{System} = N \times f_{crystal}$$

Different ways for the multiplication of the crystal's frequency to the system frequency are known and several practiced. The most known methods are Phase-Locked-Loop PLL technique and Frequency-Locked-Loop FLL.

The PLL technique holds two major disadvantages in systems with frequently and time-undefined intermitted operating modes. PLL's are systems following second order response. All the on-off operating modes result in out-of-phase conditions and therefore in continuous 'limp-mode' handling. The wide ranges of off-time conditions are contrary to the use of analog filter-integrator in the closed loop. Changes in the capacitor's charge automatically result in phase and/or frequency deviation and an improper frequency until system is in phase.

The FLL technique in combination with a digital controlled oscillator (DCO) avoids both serious problems.

The major features of the DCO are:
- fast start-up
- digital control signals - not analogs.

Beside these advantages one item needs careful consideration, the variation of the frequency with the supply voltage and temperature.

The DCO is absolutely monotone.

The FLL operates as a continuous frequency integrator. An up/down counter that follows the loop control corrects permanently the multiplication factor N. The follow-up or up-date rate is identically to the crystal's frequency rate. Using a 32,768 kHz crystal the rate is 30.5us.

The accumulated frequency error is the same as the crystal's. The time deviation from one machine cycle to another is typically less than 10%.

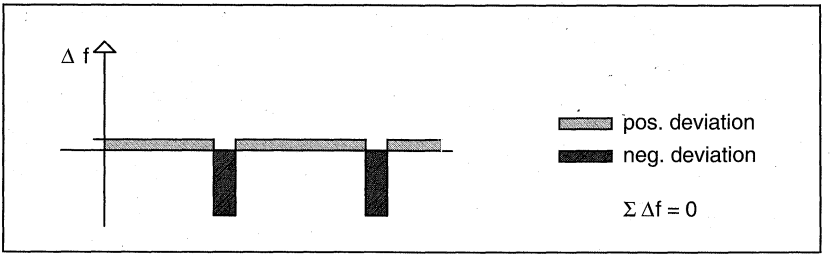


Figure 6.3: System frequency vs. time

The start-up operation of the system clock depends on the previous machine state. During a PUC the DCO is reset to its lowest possible frequency. The control logic starts operation immediately after removing PUC condition. Proper working condition for the control logic needs the presence of stable crystal oscillation.

The frequency integrator of 10bit length controls the frequency the DCO is running with. The integrator - starting at zero digital value after PUC - counts up to run the frequency f_{System} at the selected value N. It takes slightly more periods of the crystal input than the suggested number of 10bit or 1024 if the maximum length of the frequency integrator is needed. The control logic system operates aperiodically.

Applications that run the controller with intermitted operation need some attention to the conditions of handling the system frequency control conditions. The correction of the frequency integrator is possible each period of the crystal (30.5 µs @ 32 768 Hz) plus the period of f_{System}/N. Longer integration periods are mandatory to avoid accumulating deviations in time.

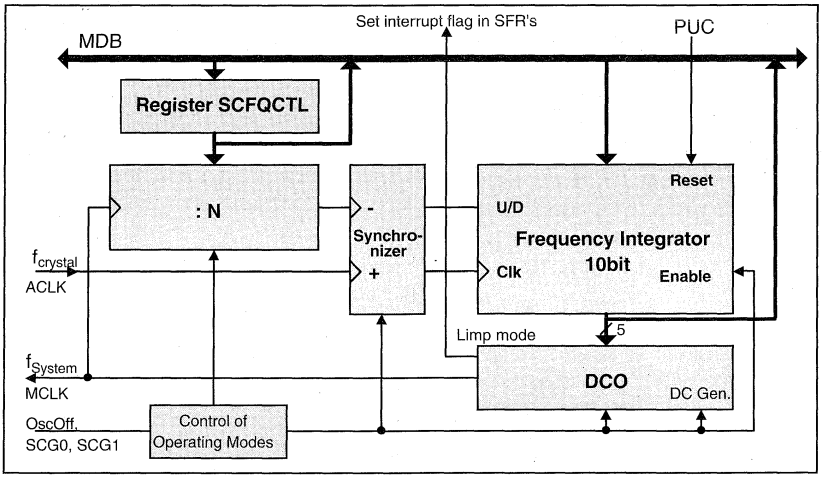


Figure 6.4: Schematic of system frequency generator

In the special function register are two flags incorporated that allows the application program to get back control over the system if the digital controlled oscillator is at its upper or lower frequency limit.

The operation at its upper or lower limit can be easily detected by controlling the frequency integrator via access to SCFI0 and/or SCFI1.

6.3 System Clock Operating Modes

The system clock generator and crystal oscillator are controlled by three signals. These signals are located in the status register SR and are reset during the four different power-up conditions.

These three control signals provide the system application with different operating conditions and maximum flexibility to optimize overall system power consumption. During some combinations of the three control signals the system clock MCLK stops operation; the present value of frequency integrator remains.

SCG1	SCG0	OscOff	Crystal oscillator	DC Generator	DCO	Loop control	Comments
0	0	0	ON	ON	ON	ON	Condition after PUC Crystal and DC oscillator are active Loop control is operating
0	1	0	ON	ON	ON	OFF	Low Power Mode **LPM1** Crystal and DC oscillator are active Loop control is off
1	0	0	ON	ON	OFF	OFF	Low Power Mode **LPM2** Crystal oscillator and DC Generator are active DCO and Loop control are off
1	1	0	ON	OFF	OFF	OFF	Low Power Mode **LPM3** Crystal oscillator is active All other functions are off
X	X	1	OFF	OFF	OFF	OFF	Low Power Mode **LPM4** All functions are disabled $f_{MCLK} = f_{ACIk} = 0Hz$

The three control signals provide five different power down modes supporting ultra-low power applications by intensive use of them. All these different modes provide the system application with the potential for operation with the smallest time slot possible and the optimized current consumption in each time slot.

The SCG0 bit controls the FLL loop if it is operating (SCG0 is reset) or off (SCG0 is set).

Starting from PUC

The system clock control register SCFQCTL is set to 01Fh with PUC and the frequency integrator is reset. The reset of the frequency integrator set the system frequency to its lowest value and counts up continously until it locks at a system frequency that is equal to N times the crystal frequency.

Low Power Mode LPM4, Oscillator off
During the oscillator off mode all parts of the processor are inactive and the current consumption is at its lowest limit. Starting with operation is only possible after power-up circuitry detected a low supply voltage condition or any external interrupt event that will request an interrupt asynchronously. The appropriate enable for interrupt sources should be applied during the program flow.

The start-up sequence of the system clock generator out of oscillator off mode:
- the present system frequency defined by the output value of the frequency integrator and the DCO characteristic will continue running
- the frequency integrator is continously counted down with the frequency of f_{System}/N till the DCO is running at its lowest frequency as long as the crystal oscillator has not started operation
- after the crystal oscillator starts operation the loop control will settle the frequency integrator to the value following $f_{System} = N * f_{crystal}$.

Low Power Mode LPM3, DC Generator off
During the DC generator off mode only the crystal oscillator is active. The DC current of the DC generator that sets the basic timing conditions is switched off. The power consumption constraints force high impedance design. The start of the DCO from power-down mode with DC generator off can take some time (t_{DCGon}) to run with the selected frequency. The time is in the range beween ns up to µs.

Low Power Mode LPM2, DCO off
The crystal oscillator and the DC generator are still active during LPM2 and an immediate start is possible. The start-up delay is limited to some gate delays.

Low Power Mode LPM1, Frequency-lock-loop off
The crystal oscillator, the DC generator and the DCO are still active during LPM1. The processor with all its peripheral modules is fully functional without any limitation. The frequency is determined from the output value of the frequency integrator. This value with the characteristic of the DCO determine the frequency of MCLK signal that is identical to the system frequency f_{System}.
There is no start-up delay, the oscillator is already running. The loop control is activated asynchronously and with a slight frequency variation but it settles fast and aperiodically.

6.4 System Clock Control Register

The system clock generator interacts with other processor parts via three general module registers and the special function registers. The general module registers are mapped into the lower peripheral file address range where all byte modules are located. Three control lines for the operating states, SCG1, SCG0 and OscOff, are supplied from the status register SR of the CPU.

6.4.1 General Module Registers

Two eight bit registers control the system clock generator. The user's software loads one of the registers with the multiplication factor N. The other register holds control bits or signals used for various operating modes. It should be accessed using byte instructions.

System Clock Frequency Control

SCFQCTL
052h

7							0
M	2^6	2^5	2^4	2^3	2^2	2^1	2^0
rw-0	rw-0	rw-0	rw-1	rw-1	rw-1	rw-1	rw-1

The content of register SCFQCTL controls the multiplication of crystal's frequency. The seven bits indicates a range of 3+1 to 127+1.

$$f_{System} = (x*2^6 + x*2^5 + x*2^4 + x*2^3 + x*2^2 + x*2^1 + x*2^0 + 1) * f_{crystal}$$

The default value in SCFQCTL after PUC was active is 31 that results in a factor of 32. The range of the f_{System} is theoretically and depends on the adjustable frequency range of the DCO (see more information in electrical characteristics).

Note: Multiplication factor in SCG

The content of register SCFQCTL (2^6 to 2^0) controls the multiplication of crystal's frequency. The seven bits must be in the a range of 3 to 127. Any value below 3 results in unpredictable operation but also any higher value that will force the MCLK frequency above the device specification.

System Clock Frequency Integrator

SCFI0
050h

7							0
0	0	0	FN_4	FN_3	FN_2	2^1	2^0
r	r	r	r	rw-0	rw-0	rw-0	rw-0	rw-0

SCFI1
051h

7							0
2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

The output of the frequency integrator controls the DCO. This value can be read using the appropriate address of SCFI1 and SCFI0. The digital representation is:

$$N_{DCO} = (x^* 2^9 + x^*2^8 + x^*2^7 + x^*2^6 + x^*2^5) + (1\text{-}M)^*(x^*2^4 + x^*2^3 + x^*2^2 + x^*2^1 + x^*2^0)$$

SCFI0, Bit 1...3: The three bits in the SC control register 0 define the nominal frequency of the DCO.

FN_4	FN_3	FN_2	Frequency
0	0	0	f_{NOM}
0	0	1	$2 \times f_{NOM}$
0	1	X	$3 \times f_{NOM}$
1	X	X	$4 \times f_{NOM}$

6.4.2 Special function register bits, System Clock Generator related

Two bits in the SFR address range handle the system control interaction according to the function implemented in the SCG. These three bits are:

- OscFault Interrupt Flag OFIFG (located in IFG1.1, initial state is unchanged)
- OscFault Interrupt Enable OFIE (located in IE1.1, initial state is reset).

The interrupt flag is part of a multiple source interrupt request. The same interrupt vector is also used for the event at the $\overline{\text{RST}}$/NMI-pin when NMI function is selected. The interrupt is defined to be non-maskable. Non-maskable implies that the general interrupt enable bit GIE can not disable the interrupt request. Since the interrupt shares the same interrupt vector and an oscillator fault is active after PUC the interrupt flag is not automatically reset.

Three different situations should be handled by the software:

- After PUC a proper sequence should be programmed to identify or to set an oscillator condition that prevents active level at OscFault signal and therefore a permanently set of OFIFG. The OFIFG should be reset by software.
  PUC resets the OFIE bit and no interrupt is requested.

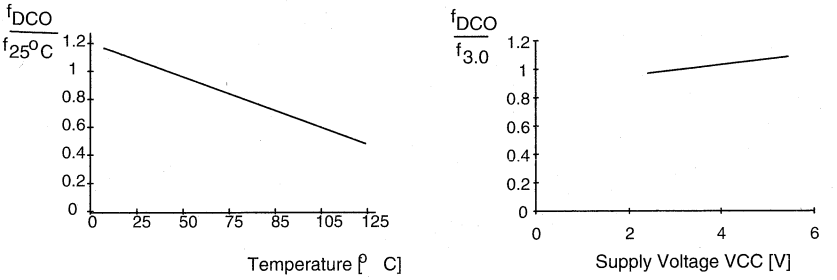
- When an interrupt from the OscFault signal was requested and serviced, the interrupt enable bit OFIE is reset automatically to disable further continuous interrupt requests until proper response from the software conducts to a inactive OscFault signal. After reaching the inactive state the OFIE bit can be set again following the general rules of module interrupts. An oscillator fault event is not affected by the general interrupt enable bit GIE.
- The interrupt flag OFIFG can be used to identify the interrupt source at beginning of the interrupt service routine. The OFIFG is set independently of an additional NMI event and is dominant.

Note: Interrupt flag OFIFG

The interrupt flag OFIFG remains set when an interrupt request was accepted and serviced. This is mandatory because it is a multiple source interrupt together with NMI interrupt and it indicates to the software interrupt handle the event of an oscillator fault. Servicing first the OFIFG condition gives this event priority over the NMI event.

6.5 DCO Characteristic - typical

The digital controlled oscillator varies with temperature and supply voltage. Running the frequency loop this is unimportant for application because the period of control is identical with the period of the ACLK signal. With a 32 768Hz crystal it is 30.5µs.



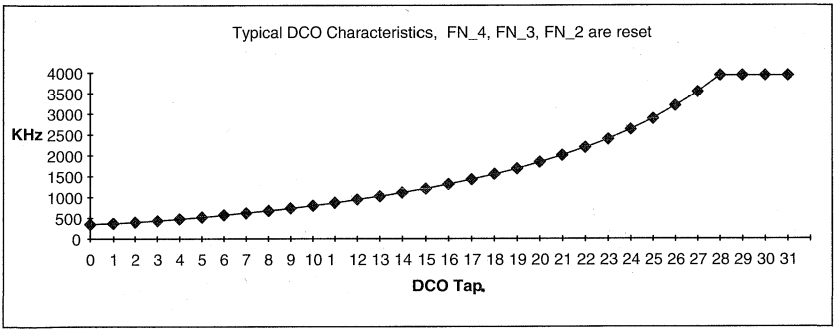


Figure 6.5: DCO Characteristics

Note: DCO Taps

The five most significant bits in the System Clock Frequency Integrator register SCFI1 are feed into the DCO. If the modulation bit M from the register SCFQCTL is set, only the DCO taps are determing the system frequency.

7 Universal Timer/Port Module

The universal Timer/Port Module supports several major system functions:

- Up to six independent outputs
- Two 8-bit counters, cascadeable for 16-bit mode
- Precision comparator for A/D conversion of slope converter type

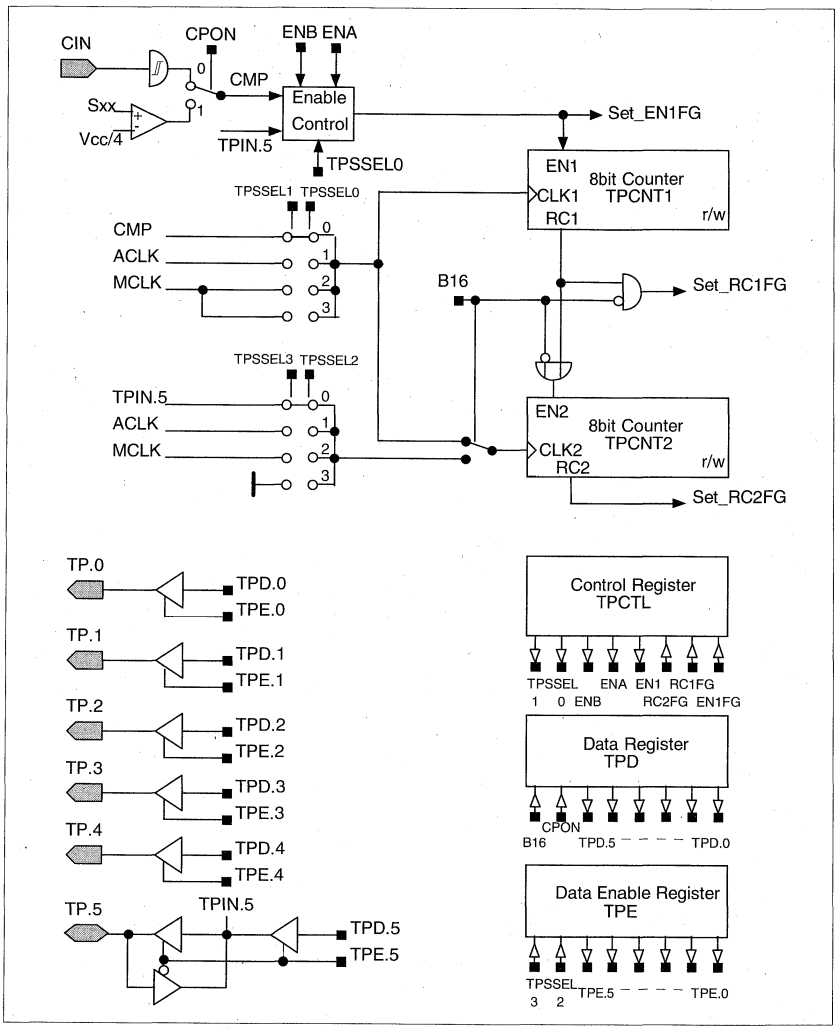


Figure 7.1: Timer/Port configuration

7.1 Timer/Port Module Operation

The Timer/Port Module can be configured through the bits in the control register TPCNTL to operate in different ways.

7.1.1 Timer/Port Counter TPCNT1, 8-bit Operation

The counter TPCNT1 can be read and written with appropriate instructions. The read access to the timer's data can be asynchronous to the clock source when CIN or ACLK is selected. A majority vote from 2 of 3 samples taken by software will ensure that the data read are correct.

When the clock source is MCLK the data read are correct. Since MCLK is the same frequency as the instructions are clocked, the data read is only a sample of the status of the counter and the counter is incremented continously while EN1 is set.

The counter can be written at any time. After the write cycle is performed a re-read of data from the counter can be different if clocks are applied between the write and read access.

Three different clock sources can count-up the counter, MCLK, ACLK or CIN.

The counter is incremented with each positive edge at the clock input when the enable input EN1 of the counter is set. The counter is enabled when one or both signals ENA and ENB are set. With system reset both enable bits are reset and the counter function is disabled. Resetting both enable bits will freeze the present counter data.

The ripple carry signal RC1 is high as long as the counter data is 0FFh. The negative edge of the ripple carry signal RC1 sets the RC1FG bit in the register TPCTL.

The flag RC1FG is set when the counter TPCNT1 rolls from 0FFh to 0. The flag EN1FG is set when EN1 is switched to disable, but not when ENA or ENB is the source of disable. An interrupt service is requested when the enable bit TPIE is set and one of the flags RC1FG, RC2FG or EN1FG is set. The flags RC1FG, RC2FG and EN1FG should be reset by software.

7.1.2 Timer/Port Counter TPCNT2, 8-bit operation

The counter's TPCNT2 operation is different from the counter's TPCNT1 operation by the source of the enable signal and clock signals. The counter is always enabled with 8-bit operation selected. Three different clock sources can count-up the counter, MCLK, ACLK or TPIN.5 .

The ripple carry signal RC2 is high as long as the counter data is 0FFh. The negative edge of the ripple carry signal RC2 sets the RC2FG bit in the register TPCTL.

The interrupt flag RC2FG is set when the counter TPCNT2 rolls from 0FFh to 0.

7.1.3 Timer/Port Counter , 16-bit operation

The 8-bit counter TPCNT1 and the 8-bit counter TPCNT2 can be cascaded to form a 16-bit counter. The bit B16 in the control register is set for this operation.

Any read or write access to the counters remains a byte access. The data of the counter TPCNT1 and TPCNT2 are read or written sequentially. This needs special considerations if the access is done during counter operation.

The enable signal of the counter TPCNT1 is the enable of the 16-bit counter and the clock source of TPCNT2 is the same as for the counter TPCNT1; it is selected only by TPSSEL0 and TPSSEL1. The source select signals TPSSEL2 and TPSEL3 are don't care.

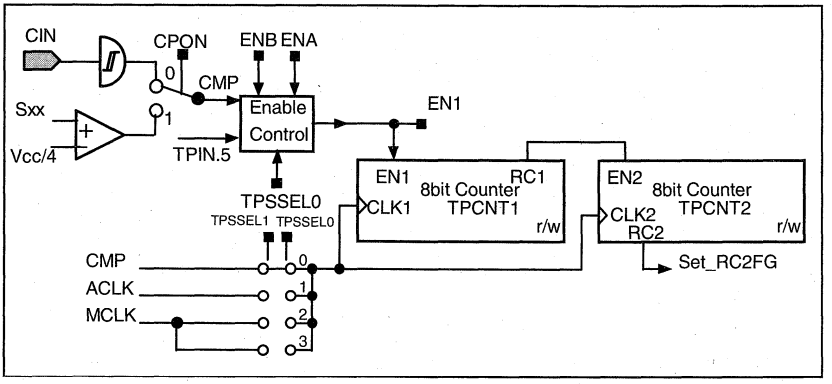


Figure 7.2: Timer/Port counter, 16-bit operation

The four signals ENA, ENB, TPSSEL0 and TPSSEL1 control the operation of the cascaded counters, the count enable and the counter's clock source:

ENB	ENA	TPSSel1	TPSSel0	EN1	CLK1
0	0	0	0	0	CMP
0	0	0	1	0	ACLK
0	0	1	X	0	MCLK
0	1	0	0	1	CMP
0	1	0	1	1	ACLK
0	1	1	X	1	MCLK
1	0	0	0	TPIN.5	CMP
1	0	0	1	TPIN.5	ACLK
1	0	1	0	TPIN.5	MCLK
1	0	1	1	TPIN.5	MCLK
1	1	0	0	$\overline{\text{CMP}}$	CMP
1	1	0	1	$\overline{\text{CMP}}$	ACLK
1	1	1	0	$\overline{\text{CMP}}$	MCLK
1	1	1	1	CMP	MCLK

The 16-bit counter can therefore be halted or counted-up unconditionally with the signal applied to pin CIN or with one of the three clocks: ACLK, MCLK or CIN.

The application of TPIN.5, TPIN.5 inverted, CIN or CIN inverted signal to the counter enable input EN1 the 16-bit counter will be incremented with each ACLK or MCLK. This feature is used to measure the time period of signals applied to pin CIN or TPIN.5.

The ripple carry signal RC2 is set as long as the counter data is 0FFFFh.

The flag RC2FG is set when the counter rolls from 0FFFFh to '0'. The flag EN1FG is set when the EN1 is switched to disable. The source of enable EN1 is TPIN.5 or CMP. It is not set when EN1 is switched to disable via software using ENA and ENB.

7.2 Timer/Port Registers

The Timer/Port module hardware is byte structured and should be accessed by byte processing instructions (suffix 'B').

Register	short form	Register type	Address	Initial state
• TP control register:	TPCTL	Type of read/write	04Bh	Reset
• TP Counter 1: unchanged	TPCNT1	Type of read/write	04Ch	
• TP Counter 2: unchanged	TPCNT2	Type of read/write	04Dh	
• TP O/P Data register:	TPD	Type of read/write	04Eh	Reset
• TP Data enable register:	TPE	Type of read/write	04Fh	Reset

Timer/Port Control register
The information stored in the control register determines the operation of the Timer/Port module.

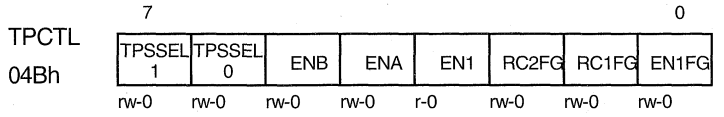


Figure 7.3: Timer/Port Control Register

Bit 0: The enable flag EN1FG is set with the negative edge of enable signal EN1 of counter TPCNT1 if the enable signal came from CIN or TPIN.5 pin. This event sets the EN1FG bit and the software should reset it. Otherwise it will remain set.
The EN1FG bit can be used during the Timer/Port interrupt service routine to decide if the interrupt event was from enable EN1 or from a ripple/carry RC that is set when a counter rolls from 0FFh to 0h.

Bit 1: The bit RC1FG indicates that the counter TPCNT1 has rolled from 0FFh to 0h (overflow condition). This event sets the RC1FG bit and the software should reset it. Otherwise it will remain set.
It is used in the Timer/Port Interrupt Service routine to identify the source of an interrupt event.

Bit 2: The bit RC2FG indicates that the counter TPCNT2 has rolled from 0ffh to 0h (overflow condition). This event sets the RC2FG bit and the software should reset it. Otherwise it will remain set.
It is used in the Timer/Port Interrupt service routine to identfiy the source of an interrupt event.

Bit 3, 4, 5: The enable signal EN1 of the counter TPCNT1 can be read. The level or the signal of the bit EN1 is defined with control signals ENA, ENB, TPSSEL0.

ENB	ENA	TPSSel0	EN1
0	0	X	0
0	1	X	1
1	0	0	$\overline{\text{TPIN.5}}$
1	0	1	TPIN.5
1	1	0	$\overline{\text{CMP}}$
1	1	1	CMP

Bit 7, 6: The Timer/Port clock source select bits TPSSEL0 and TPSSEL1 control the multiplexer to supply 1 of 3 clock sources to the counter TPCNT1.

TPSSel1	TPSSel0	CLK1
0	0	CMP
0	1	ACLK
1	X	MCLK

Timer/Port counter TPCNT1 and TPCNT2

Both counters are 8-bit and any read or write access should be done with byte instructions.

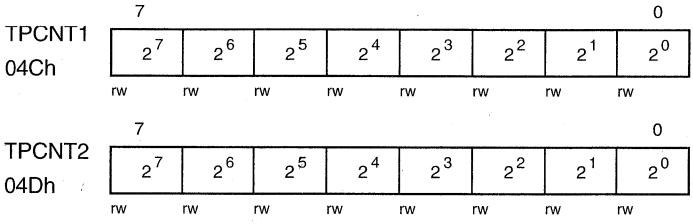


Figure 7.4: Timer/Port Counter Registers

Both counters can be read and written independently. Any reset of a counter is done using the CLEAR instruction.

Timer/Port Data Register

The Data Register holds the value of six outputs and two control bits of the comparator.

Figure 7.5: Timer/Port Data Register

Bit 0 ... 5: The bits TPD.0 to TPD.5 hold the data for the output pins TP.0 to TP.5. The digital signals will be applied to these pins when the 3-state output is enabled by TPE.0 to TPE.5 . They are reset whenever a system reset PUC happens .
The signal at TP.5 is used internally in the module and can be read via the enable bit EN1 located in the control register TPCTL.

Bit 6: The comparator CPON bit switches on the supply of the comparator. It is used to save current during its reset state. Whenever system reset PUC becomes active, the comparator on bit CPON is reset and the comparator is inactive.

Bit 7: The control bit B16 selects the operation of the two counters TPCNT1 and TPCNT2. They can operate as two independent 8-bit counters or as one 16-bit counter. The access is always in byte mode. In the 16-bit mode, any read or write access is done seperately to counter TPCNT1 and counter TPCNT2.

B16 = 0: Two 8-bit counter mode selected.
B16 = 1: One 16-bit counter with TPCNT1 for low byte and TPCNT2 for high byte. The counter TPCNT2 increments its data when the counter TPCNT1 rolls from 0FFh to 0h.

Timer/Port Enable Register

The Timer/Port Enable Register holds the control value of six outputs and two bits indicating counter overflow.

TPE 04Fh

7							0
TPSSEL 3	TPSSEL 2	TPE.5	TPE.4	TPE.3	TPE.2	TPE.1	TPE.0
rw-1	rw-1	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Figure 7.6: Timer/Port Enable Register

Bits 0 ... 5: The bits TPE.0 to TPE.5 hold the enable control (3-state) signals of the outputs TP.0 to TP.5. They are reset whenever a system reset PUC happens and the outputs are high-impedance.

Bit 6, 7: The Timer/Port clock source select bits TPSSEL2 and TPSSEL3 control the multiplexer to supply 1 of 4 clock sources to the counter TPCNT2. The control bit B16 should be reset. When the control bit B16 is set TPSSEL2/3 are don't care and the clock source of counter TPCNT2 is the same as of the counter TPCNT1.

B16	TPSSel3	TPSSel2	CLK2
0	0	0	TPIN.5
0	0	1	ACLK
0	1	0	MCLK
0	1	1	'0' or VSS
1	X	X	≡ CLK1

7.3 Timer/Port Special Function bits

The Timer/Port module covers one interrupt vector, multiple source interrupt flags (not located in the SFR) and one interrupt enable bit.

The Timer/Port interrupt enable TPIE, located in IE.2 register. Initial state is reset.

The multiple source interrupt flags RC1FG, RC2FG and EN1FG are located in the register TPCTL. Their initial state is reset.

The interrupt flags RC1FG, RC2FG and EN1FG are not reset automatically by hardware when the interrupt request is served. The flag EN1FG is set with the negative edge of the counter enable signal EN1 and indicates that the counter was halted. It is not set if software toggles from enable to disable of TPCNT1 by using ENA and ENB control signals.

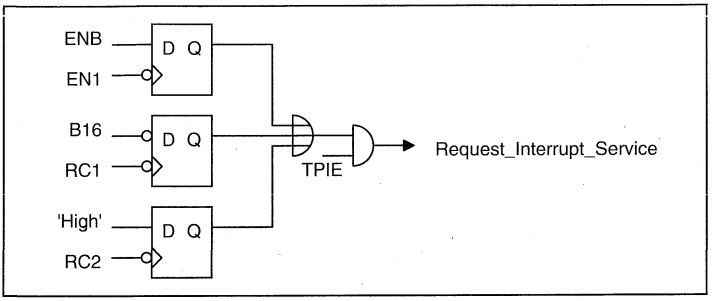


Figure 7.7: Timer/Port Interrupt Scheme

When a Timer/Port interrupt is asserted, the interrupt service routine should consider the different interrupt sources and decide how to proceed. When the 8-bit counter mode is selected, three interrupt sources can request an interrupt service, the negative edge of EN1 signal, an overflow from counter TPCNT1 (RC1 signal) or an overflow from counter TPCNT2 (RC2 signal). In the 16-bit counter mode (B16 is set) two sources can request an interrupt service, the negative edge of EN1 signal and an overflow from counter TPCNT2.

B16	RC2		RC1		EN1
0	↓	OR	↓	OR	↓
1	↓	OR	X	OR	↓

Figure 7.8: Conditions for Timer/Port Interrupt request

The interrupt request bits should be reset during the interrupt service routine. If this is not done, another interrupt is requested immediately after the GIE is set or the RETI instruction is executed. The Timer/Port interrupt enable bit TPIE is reset with PUC. When the TPIE bit is reset, no interrupt request is done. When the TPIE bit and the general interrupt enable bit GIE are set the System/CPU will serve the interrupt if no PUC or NMI is active.

7.4 Timer/Port in ADC Application

The Timer/Port peripheral incorporates all functions to support an A/D converter function for resistive or capacitive sensors.

For temperature measurement the most popular sensor elements are resistors that have positive or negative temperature coefficients. Silicon sensors, NTC Resistsors and Platinium sensors are such sensor types.

7.4.1 Principle of conversion, R/D

The conversion of a resistor value to a digital representation is done by measuring the time that is needed to discharge a capacitor. This capacitor is charged-up before the discharge phase.

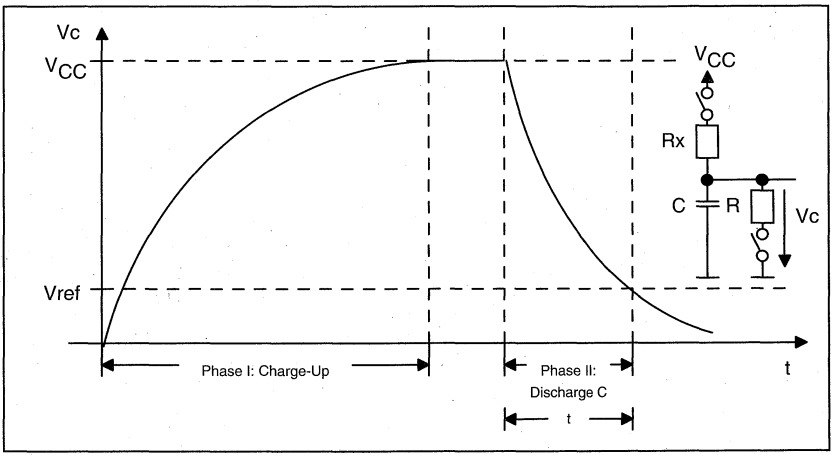


Figure 7.9: Charge-Discharge timing of RC

During the discharge of the capacitor down to the reference voltage Vref, the time of this period is measured using a voltage comparator and a counter. The counter is incremented continuously as long as the voltage level at the capactor C is above the reference level Vref. The increment of the counter is possible because the comparator output is used to enable the counter's operation.

The formular for this time measurement principle is

$$t = -R * C * \ln\frac{Vref}{Vcc}$$

$$t = N * t_{Clock}$$

$$N * t_{Clock} = -R * C * \ln\frac{Vref}{Vcc}$$

$$N = -R\,C\,f_{Clock} * \ln\frac{Vref}{Vcc}$$

The value of C, f_{Clock} and $Vref/V_{CC}$ should be known to determine the value of resistor R. Using a second conversion with a well defined and stable reference resistor, the sensor to be measured can be determined by:

$$\frac{N_{meas}}{N_{ref}} = \frac{-R_{meas} * C * \ln\frac{Vref}{Vcc}}{-R_{ref} * C * \ln\frac{Vref}{Vcc}}$$

$$\frac{N_{meas}}{N_{ref}} = \frac{R_{meas}}{R_{ref}}$$

$$R_{meas} = R_{ref} * \frac{N_{meas}}{N_{ref}}$$

This assumes that the circuit uses the same capacitor and both voltages and the clock period are constant during both conversions.

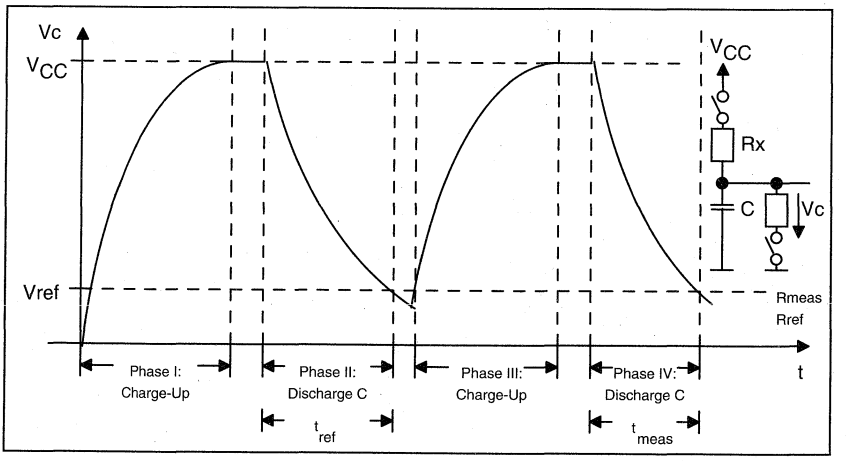


Figure 7.10: Charge-Discharge timing during R/D conversions using Rref and Rmeas

The capacitior C is charged through any resistor, Rx, up to V_{CC} during Phase I and Phase III. It is discharged via Rref or Rmeas. The current is then limited by the resistor(s). If only the resistor Rref used, the time for charging the capacitor is well defined.

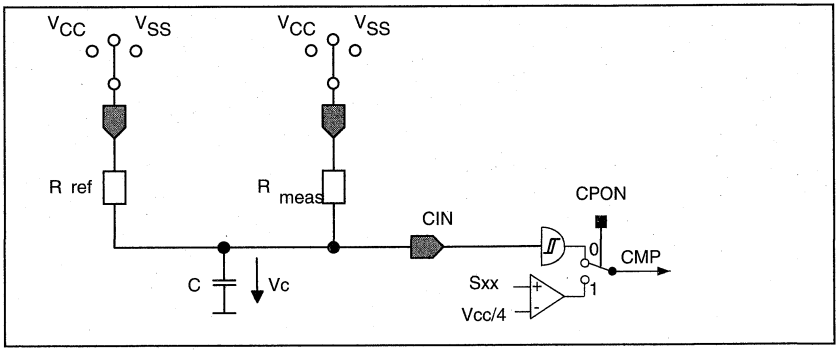


Figure 7.11: Principle Conversion Scheme

The capacitor C is discharged while one of the resistors is connected to Vss. All parasitics of the discharge current path will influence the time to discharge the capacitor C down to Vref.

7.4.2 Conversion with Resolution of >8 bit

The conversion is demonstrated using the comparator, the counters in 16-bit mode and the digital outputs.

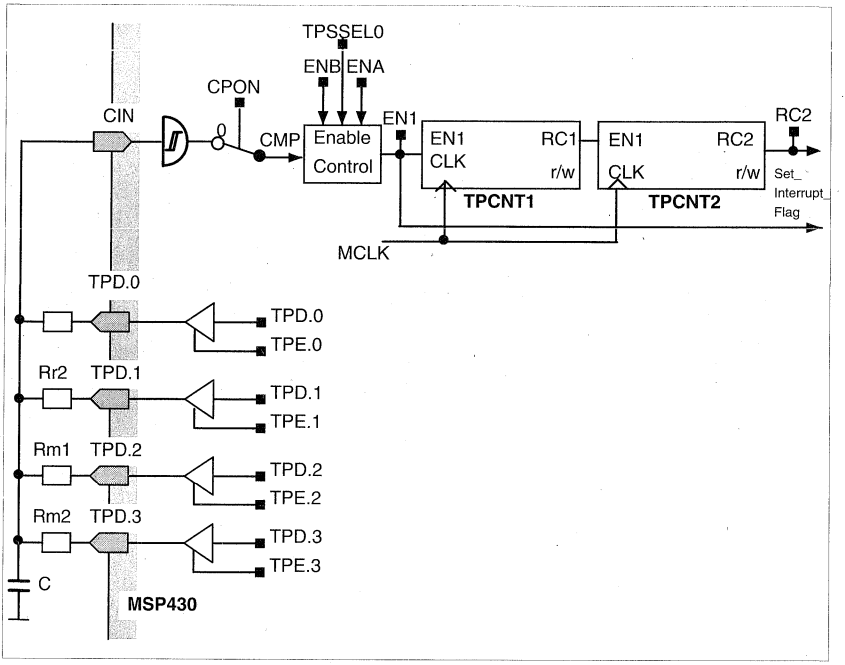


Figure 7.12: ADC Application example

The external components in this application are two reference resistors, Rr1 and Rr2, and two sensor resistors to be measured, Rm1 and Rm2.

The internal configuration is selected via the control register TPCTL to meet the circuit function of the application example. The schematic is reduced to the active connections and block. The control register TPCTL is loaded with 9Eh which means that bits B16, TPSSEL1, TPSSEL0, ENB and ENA are set.

The capacitor C is charged through Rr1 and/or Rr2 up to V_{CC}.

The time of the four discharge phases is measured:

$$t_{r1} = N_{r1} * t_{MCLK} = -R_{r1} * C * \ln \frac{V_{ref}}{V_{CC}}$$

$$t_{r2} = N_{r2} * t_{MCLK} = -R_{r2} * C * \ln \frac{V_{ref}}{V_{CC}}$$

$$t_{m1} = N_{m1} * t_{MCLK} = -R_{m1} * C * \ln \frac{V_{ref}}{V_{CC}}$$

$$t_{m2} = N_{m2} * t_{MCLK} = -R_{m2} * C * \ln \frac{V_{ref}}{V_{CC}}$$

and the following formula is used to determine the resistors Rm1 and Rm2:

$$\frac{R_{r1} - R_{r2}}{R_{mx} - R_{r2}} = \frac{N_{r1} - N_{r2}}{N_{mx} - N_{r2}}$$

$$R_{mx} = R_{r2} + (R_{r1} - R_{r2}) * \frac{N_{mx} - N_{r2}}{N_{r1} - N_{r2}}$$

$$R_{m1} = R_{r2} + (R_{r1} - R_{r2}) * \frac{N_{m1} - N_{r2}}{N_{r1} - N_{r2}}$$

$$R_{m2} = R_{r2} + (R_{r1} - R_{r2}) * \frac{N_{m2} - N_{r2}}{N_{r1} - N_{r2}}$$

8 Digital I/O Configuration

8.1 General Port P0

The general port P0 incorporates all the functions to individually select the function of each pin and to use each signal as interrupt source.
The six registers are used for the control of Port's I/O pins.
The general module registers are mapped into the lower peripheral file address range where all byte modules are located.

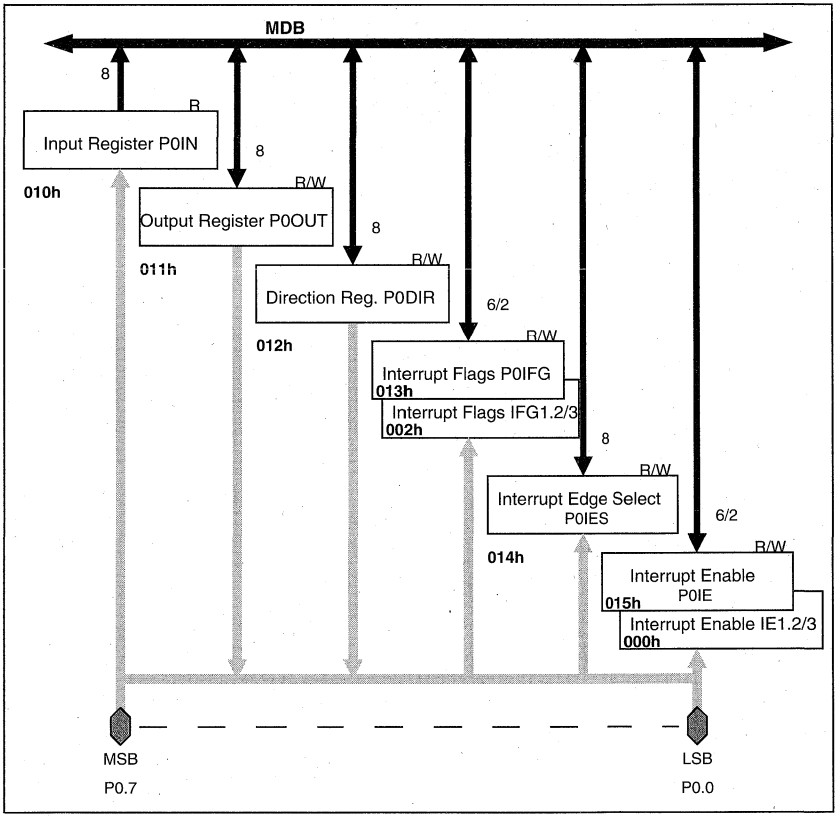


Figure 8.1: Port 0 Configuration

8.1.1 Port 0 Control Registers

The Port 0 is connected to the processor core via the 8-bit MDB structure and MAB. It should be accessed via byte instructions.

The six control registers give maximum flexibility of digital input/output to the application:
- All individual I/O bits are programmable independently:
- Any combination of input, output and interrupt condition is possible.
- Interrupt processing of external events is fully implemented for all eight bits of the port P0.

The six registers are:

Register	short form	Register type	Address	Initial State
• Input register:	P0IN	read only	010h	-----
• Output register:	P0OUT	read/write	011h	unchanged
• Direction register:	P0DIR	read/write	012h	reset
• Interrupt Flags:	P0IFG	read/write	013h	reset
• Interrupt Edge Select:	P0IES	read/write	014h	unchanged
• Interrupt Enable:	P0IE	read/write	015h	reset

All these registers contain eight bits except the two LSBs in the interrupt flag register and the interrupt enable register. These two bits are included into the special function register SFR. The registers should be accessed with byte instructions.

Input Register P0IN
The input register is a read only register to scan the signals at the I/O pins. The direction of the pin should be selected for input.

> **Note: Writing to read only register P0IN**
>
> Writing to this read only register results in an increased current consumption as long as the write is active.

Output Register P0OUT
The Output Register shows the information of the output buffer, an eight bit register that contains the information output by the I/O pins if used as outputs. The output buffer can be modified by all instructions that write to a destination. If read, the contents of the output buffer are read independently of the direction. A direction change does not modify the output buffer contents.

Direction Register P0DIR
This register contains eight independent bits that define the direction of the I/O pin. All bits are reset by PUC:

Bit = 0: The I/O pin is switched to input direction
Bit = 1: The I/O pin is switched to output direction

Interrupt Flags P0IFG

This register contains six flags that contain information if the I/O pins are used as interrupt inputs:

P0IFG
013h

7							0
P0IFG.7	P0IFG.6	P0IFG.5	P0IFG.4	P0IFG.3	P0IFG.2		
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r0	r0

Bit = 0: No interrupt is pending
Bit = 1: An interrupt is pending due to a transition at the I/O pin. Manipulation on P0OUT and P0DIR can also set bits of P0IFG.

Writing a zero to an Interrupt Flag resets it.
The six flags are located in bit 7 to 2 according to the pin P0.7 to P0.2. The remaining interrupt flags of pin P0.1 and P0.0 are located in the SFRs.

Note: Interrupt Flags P0FLG.2...7

The Interrupt Flags P0FLG.2 to P0FLG.7 use only one interrupt vector: it is a multiple source interrupt vector. The interrupt flags P0IFG.2 to P0IFG.7 are not reset automatically when any interrupt from these events is served. The software decides which event will be served and should reset the appropriate flag.

Any external interrupt event should be as long as 1.5 times MCLK or longer to ensure that it is accepted and the according interrupt flag is set.

Interrupt Edge Select P0IES

This register contains a bit for each I/O pin that selects which transition triggers the interrupt flag. All eight bits according to pin P0.7 to P0.0 are located in this register. The bits have the following meaning:

Bit = 0: The interrupt flag is set with LO/HI transition
Bit = 1: The interrupt flag is set with HI/LO transition

Note: Change of P0IES bit(s)

Any change of P0IES bit(s) may result in setting the associated interrupt flags.

Interrupt Enable P0IE

This register contains a bit for six I/O pins to enable interrupt request on an interrupt event. Two interrupt enable bits for P0.0 and P0.1 are located in special function register IE1.2 and IE1.3. Six bits according to pin P0.7 to P0.2 are located in the P0IE register.

P0IE
015h

7							0
P0IE.7	P0IE.6	P0IE.5	P0IE.4	P0IE.3	P0IE.2		
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r0	r0

The bits have the following meaning:

Bit = 0: The interrupt request is disabled
Bit = 1: The interrupt request is enabled

Note: Port0 interrupt sensitivity

Only transitions, not static levels cause interrupts.

The interrupt routine must reset the multiple used Interrupt Flags P0IFG.2... P0IFG.7. The single source flags P0IFG.0 and P0IFG.1 are reset when they are serviced.

If an Interrupt Flag is still set (because the transition occurred during the interrupt routine) when the *RETI* instruction is executed, an interrupt occurs again after *RETI* is completed. This ensures, that each transition is seen by the software.

8.1.2 Port 0 Schematic

The pin logic of each individual signal of port P0 is built from five identically register bits and one 3-state bus driver. The register bits include the bit storing element plus the associated control logic:

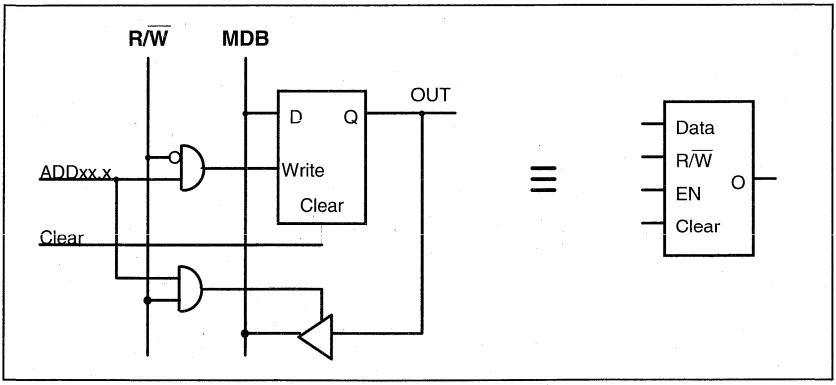


Figure 8.2: Schematic of bit register

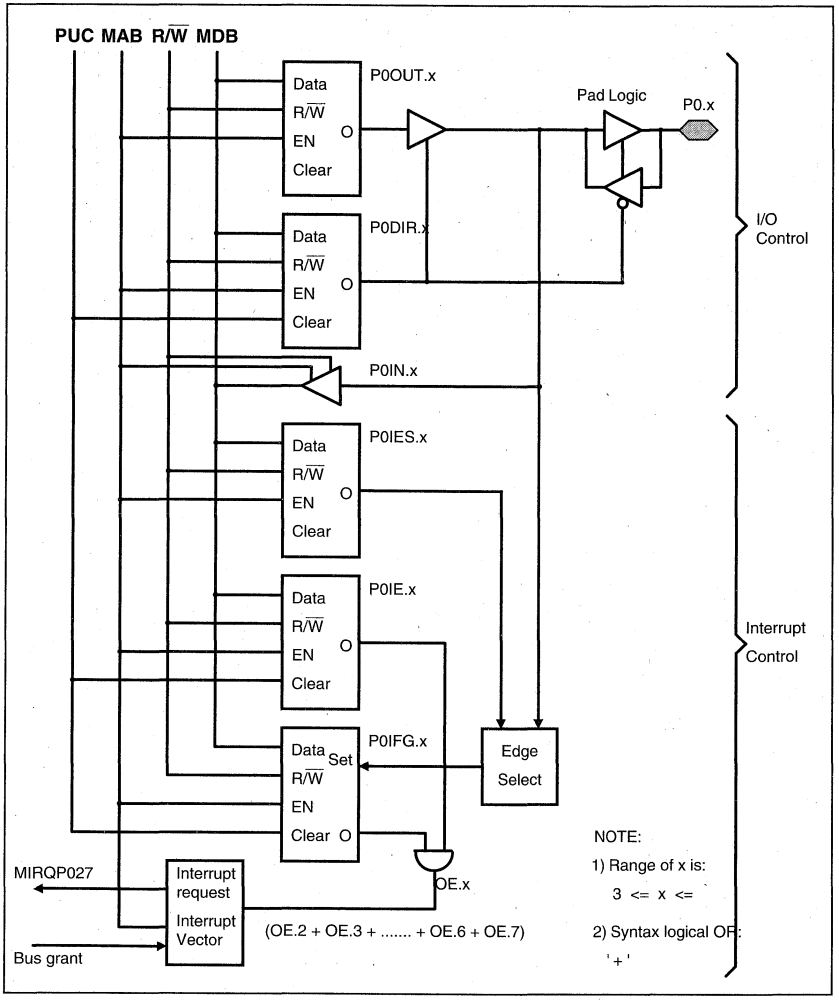


Figure 8.3: Schematic of bits P0.7 to P0.3

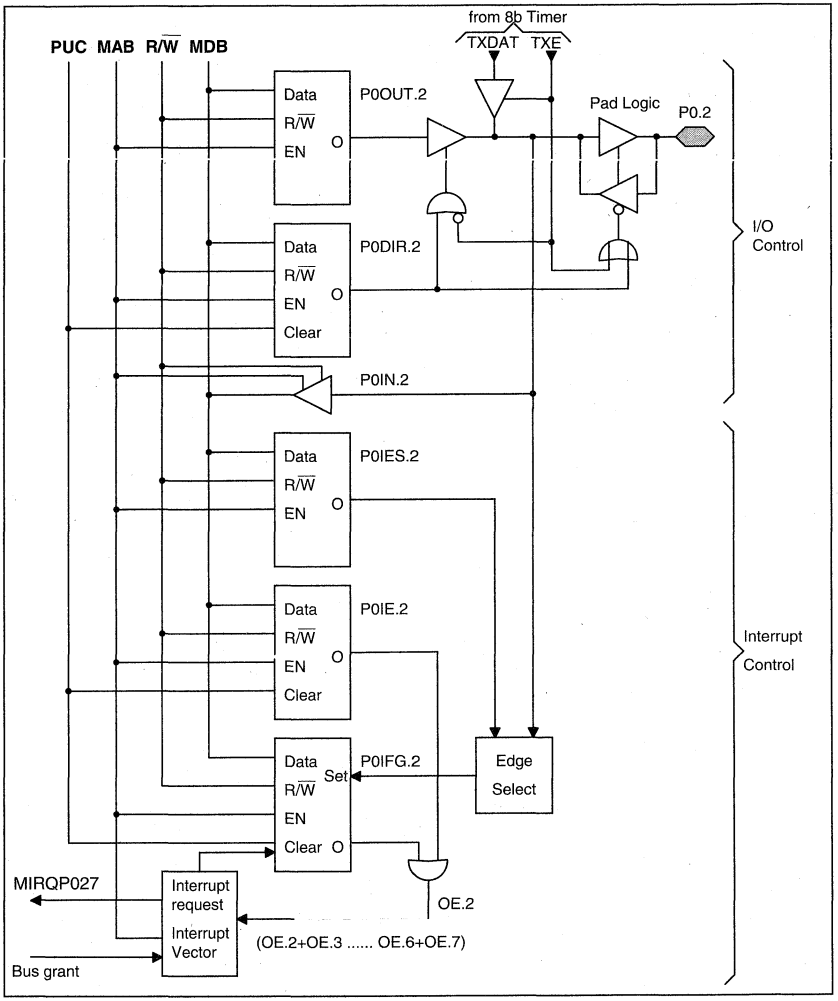


Figure 8.4: Schematic of bits P0.2

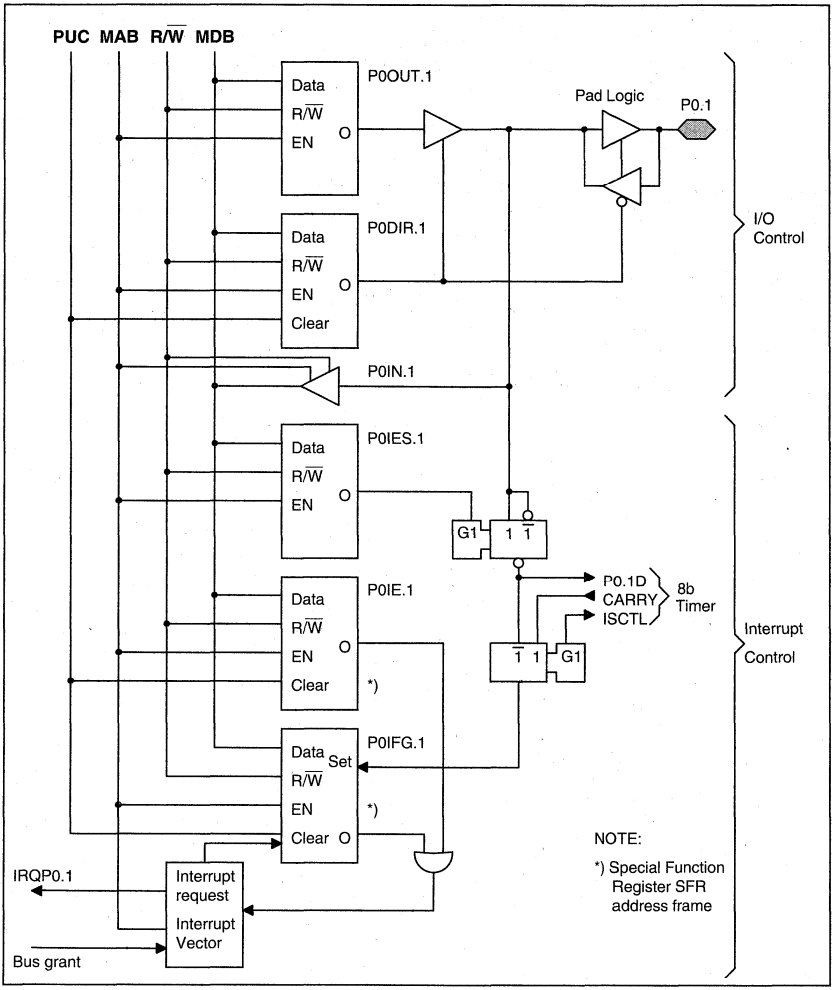


Figure 8.5: Schematic of P0.1

The I/O function of P0.7 to P0.2 differs from the function of P0.1 and P0.0 in the interrupt control section. Each interrupt event on pin P0.1 and P0.0 triggers one interrupt flag and results in a different vector address. Event on any of the pins P0.7 to P0.2 are collected into a 'group' or multiple source interrupt. An event on one or more of the pins P0.7 to P0.2 triggers the appropriate and enabled IO flag P0IFG.x. If any of these IO flags

P0IFG.x is set and the appropriate interrupt enable bit is set too it will result in an interrupt on the same vector address. The software should identify the trigger source testing the individual flags (P0IFG.x). Multiple interrupt source flags are not reset during interrupt request service. They should be handled in the interrupt service routine.

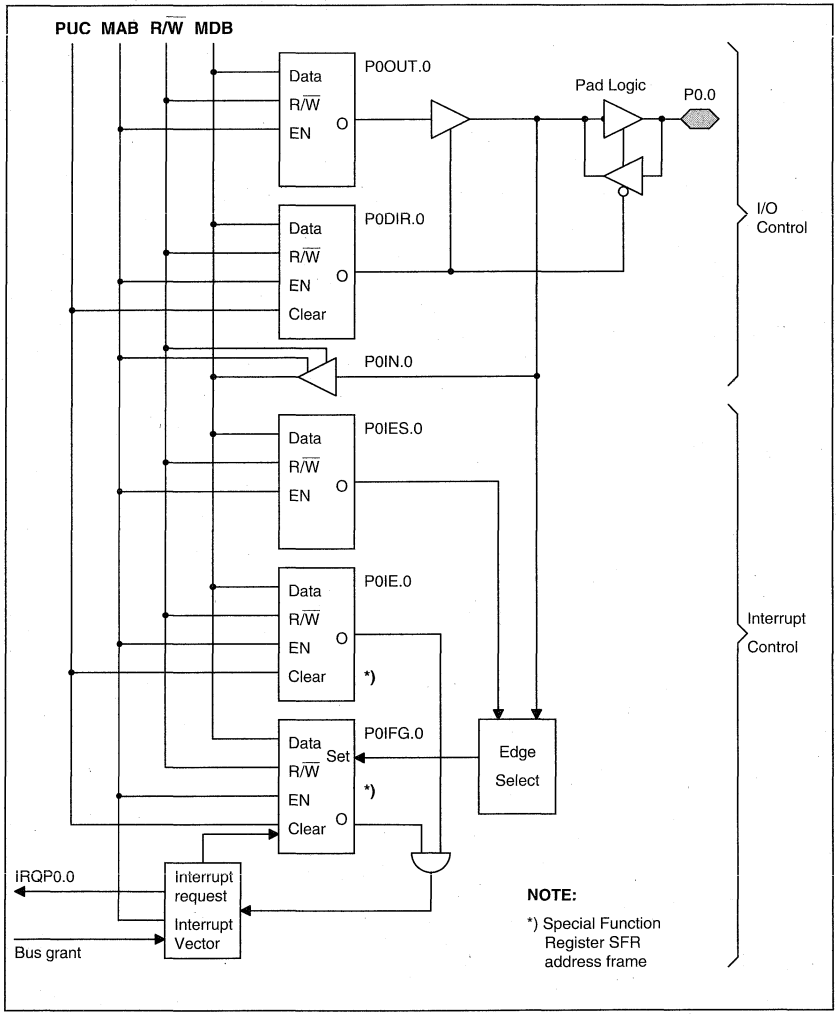


Figure 8.6: Schematic of P0.0

8.1.3 Port0 interrupt control functions

Port0 uses eight bits for interrupt flags, eight bits for interrupt enable, eight bits to select the effective edge of an interrupt event and three different interrupt vector addresses. The three interrupt vector addresses are assigned to:

- P0.0
- P0.1/RXD
- P0.2 to P0.7

Two port0 signals P0.0 and P0.1/RXD are used for dedicated signal processing. Four bits in the SFR address range and two bits in the port0 address frame handle the interrupt events on P0.0 and P0.1/RXD :

- P0.0 Interrupt Flag P0IFG.0 (located in IFG1.2, initial state is reset)
- P0.1/RXD Interrupt Flag P0IFG.1 (located in IFG1.3, initial state is reset)
- P0.0 Interrupt Enable P0IE.0 (located in IE1.2, initial state is reset)
- P0.1/RXD Interrupt Enable P0IFG.1 (located in IE1.3, initial state is reset)
- P0.0 Interrupt Edge Select (located in P0IES.0, initial state is reset)
- P0.1/RXD Interrupt Edge Select (located in P0IES.1, initial state is reset)

Both interrupt flags are single source interrupt flags and are automatically reset when the processor system serves them. The enable bits and edge select bits remain unchanged.

The interrupt control bits of the remaining six I/O signals P0.2 to P0.7 are located in the I/O address frame. Each signal uses three bits that define reaction on interrupt events:

- interrupt flag, P0IFG.2 to P0IFG.7
- interrupt enable bit, P0IE.2 to P0IE.7
- interrupt edge select bit, P0IES.2 to P0IES.7

The interrupt flags P0IFG.2 to P0IFG.7 are part of a multiple source interrupt request. Any interrupt event on one or more pin P0.2 to P0.7 will request an interrupt when two conditions are met, the appropriate individual bit P0IE.x ($2 \leq x \leq 7$) is set and the general interrupt enable bit GIE is set. The six interrupt sources use the same interrupt vector. Since the interrupt shares the same interrupt vector interrupt flags P0.2 to P0.7 are not automatically reset.

The software of the interrupt service routine handles the detection of the source and also resets the appropriate flag when it is serviced.

Note: Multiple Source interrupt flags P0IFG.2 to P0IFG.7

The interrupt flags P0IFG.2 to P0IFG.7 remain set when an interrupt request has been accepted and serviced. This is mandatory because it is a multiple source interrupt.

8.2 LCD Ports

The LCD ports can be selected either to drive a liquid crystal display or to act as digital outputs driving static output signals. The control of a liquid crystal display requires common and segment output stages that can drive analog signals if multiplex rates beyond 2Mux are supported.

LCD outputs

The LCD outputs use transmission gates to transfer the analog voltage to the output pin when they are used to drive liquid crystal displays. Groups of LCD outputs can be configured by software to operate as digital outputs.

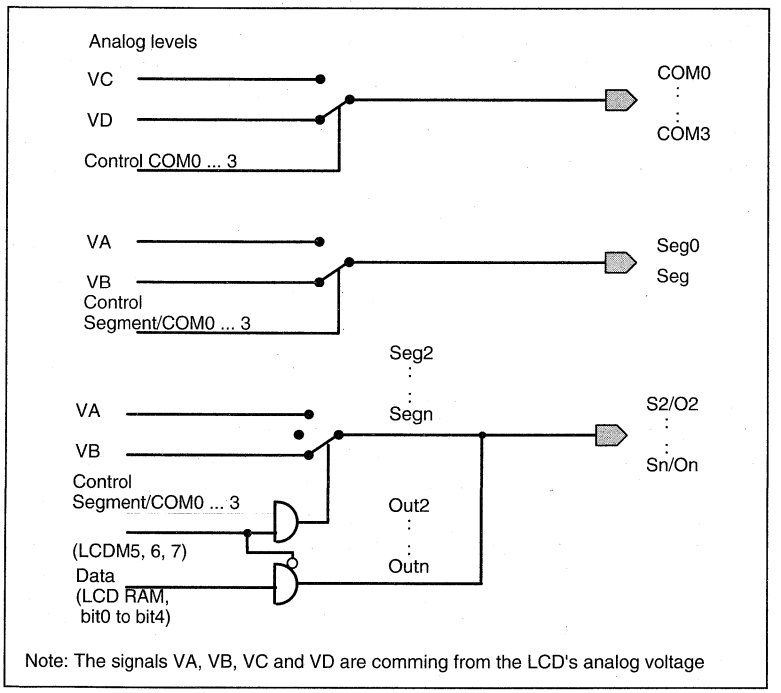


Figure 8.7: Schematic of LCD pin configuration

Three bits in the LCD control register LCDM5, LCDM6 and LCDM7 control the function of these groups of signals. For more information on control of these outputs see LCD description.

8.3 LCD Port - Timer/Port Comparator

The comparator associated with the Timer/Port module is shared typically with one segment line. The segment line function is selected for this pin after PUC signal was active. The comparator input is selected when the CPON bit - located in the Timer/Port module - is set the first time. It remains set as long as it is reset by PUC.

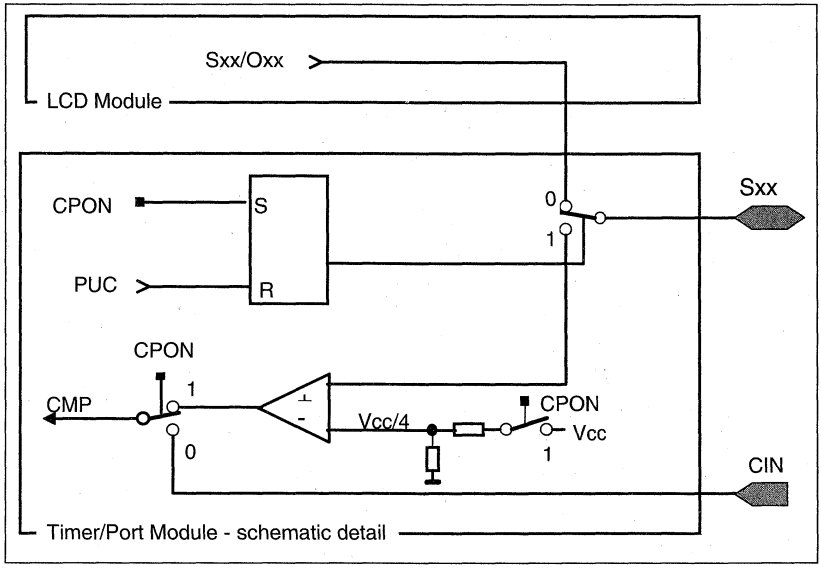


Figure 8.8: Schematic of LCD pin - Timer/Port Comparator

9 Timers

9.1 Basic Timer, Basic Timer1

The intention of the basic timer operation is to support the software and various peripheral modules with a low power consuming, low frequency reference.

Examples of software functions controlled by crystal's stability:
- real time clock RTC
- debouncing keys, keyboard
- software time incremental feature**

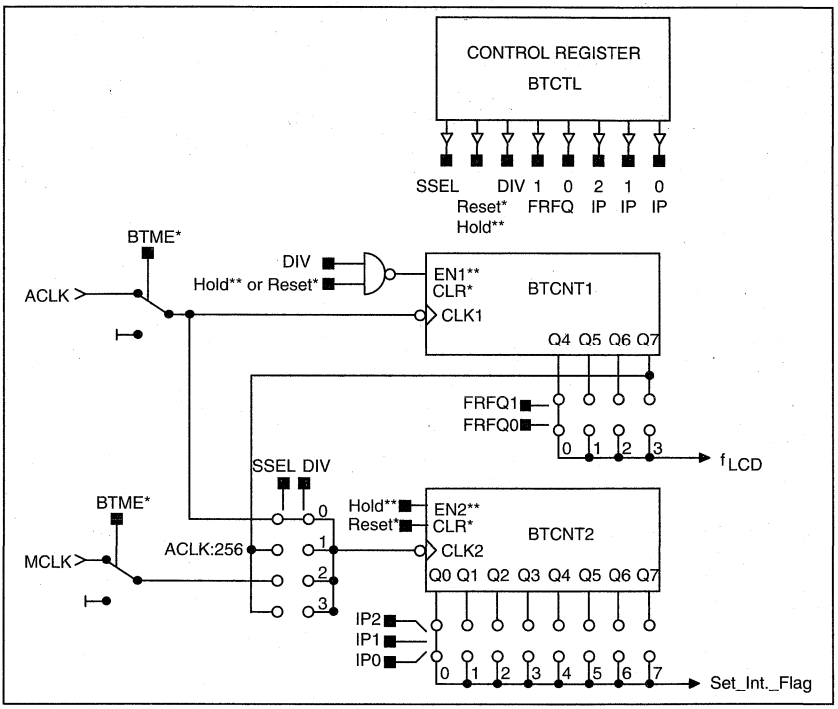


Figure 9.1: Basic Timer Configuration

Note: Basic Timer, Basic Timer1 marks

** Function implemented in Basic Timer1.
* Function implemented in Basic Timer.

These marks are valid throughout the Basic Timer section.

The Basic Timer supplies other peripheral modules with low frequency control signals. The configuration of the Basic Timer1** peripheral enables the software access to both 8-bit counters.

The control register BTCTL holds the flags to control or select the different operational functions. The register BTCTL (as well as the BTCNT1/BTCNT2**) is connected to the CPU with the peripheral standard interconnection using the MAB, MDB and control lines. When supply voltage is applied, a reset of the device or a watchdog overflow or any other operational condition occurs all bits in the register hold an undefined or unchanged status. The user's software usually configures the operational conditions on the BT during initialization.

Differencies of Basic Timer and Basic Timer1

	Basic Timer	Basic Timer1
• SFR bit BTME	Yes, stops clock source	No
• Bit 6 of BT(1)CTL	Reset CNT2 Reset CNT1 with DIV set	Hold CNT2 Hold CNT1 with DIV set
• BTCNT1/BTCNT2	Reset capability	Hold capability
• BTCNT1/BTCNT2 read/write capability	No	Yes

9.1.1 Basic Timer, Basic Timer 1 Register

The Basic Timer module hardware is connected to the processor core using access via the 8-bit MDB structure and MAB. It should be accessed using byte instructions.

Register	short form	Register type	Address	Initial state
• BT,BT1 control register	BTCTL	Type of read/write	040h	unchanged
• BT counter 1**	BTCNT1	Type of read/write	046h	unchanged
• BT counter 2**	BTCNT2	Type of read/write	047h	unchanged

Note: Basic Timer1 counter access

** Both counters can be accessed only in the Basic Timer1 Module.

Basic Timer, Basic Timer1 Control Register

The information stored in the control register determines the operation of the basic timer. The status of the different bits selects the frequency source, the interrupt frequency and the framing frequency of the LCD control circuitry.

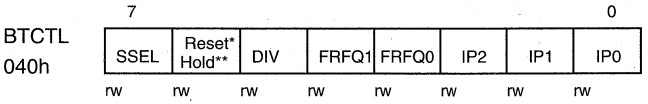


Figure 9.2: Basic Timer, Basic Timer1 Register

Bit 0 ... 2: The three least significant bits IP2..0 determine the interrupt interval time. It is the interval of consecutive settings of interrupt request flag BTIFG.

Bit 3 ... 4: The two bits FRFQ1 and FRFQ0 select the frequency f_{LCD}. Devices with LCD peripheral on-chip use this frequency to generate the timing of the common and select lines.

Bit 5: see Bit 7.

Bit 6: * Basic Timer: The Reset bit resets the counter/divider. The interrupt flag BTIFG is reset.
BTCNT2 is reset while Reset bit is set.
BTCNT1 is reset while Reset bit and DIV are set.

** Basic Timer1: The Hold bit stops the counters operation.
The BTCNT2 is held if Hold bit is set.
The BTCNT1 is held if Hold bit and DIV bit are set.

Bit 7: The SSEL bit and DIV bit select the input frequency source of BTCNT2.

SSEL	DIV	CLK2
0	0	ACLK
0	1	ACLK/256
1	0	MCLK
1	1	ACLK/256

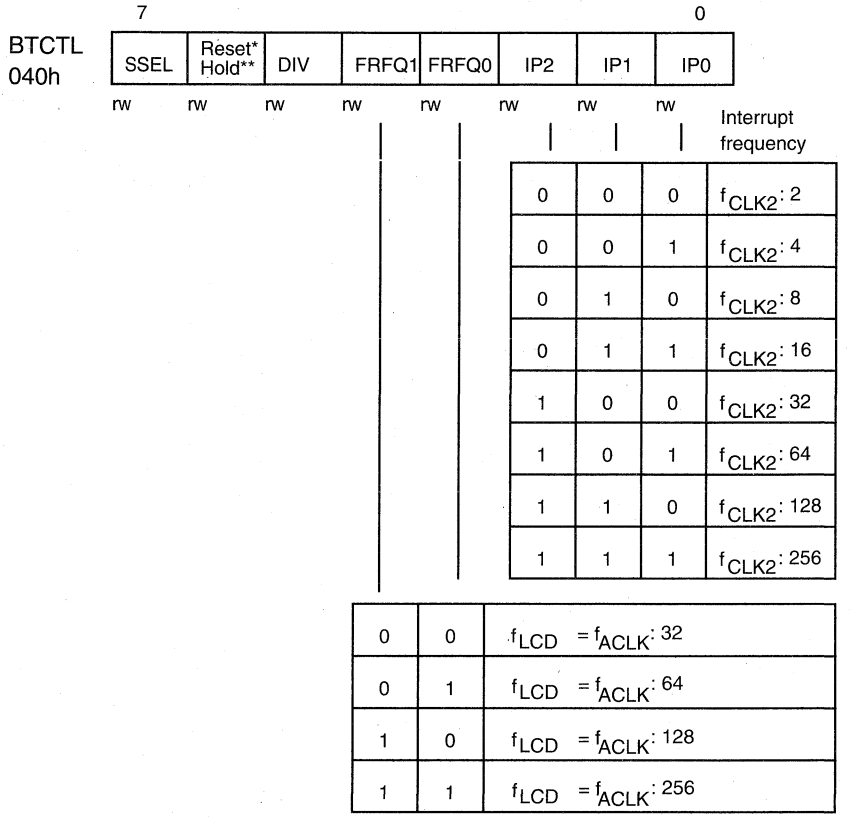


Figure 9.3: Basic Timer, Basic Timer1 Register Function

Basic Timer, Basic Timer1 Counter 1

The Basic Timer Counter BTCNT1 divides the auxiliary clock ACLK. The frame frequency for LCD-Drive is selected from four outputs of the counter FFs. The output of the most significant FF can be used for the clock input to the second counter BTCNT2. The output of the counter Q0...7 can be read and the counter Q0..7 can be written by software in Basic Timer1 only.

	7							0
BTCNT1 046h	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Basic Timer1	rw	rw	rw	rw	rw	rw	rw	rw
Basic Timer	--	--	--	--	--	--	--	--

Basic Timer, Basic Timer1 Counter 2

The Basic Timer Counter BTCNT2 divides the input clock frequency. The input clock source can be selected to be MCLK, ACLK or ACLK:256 signal. The interrupt period can be selected using IP0...2 located in the basic timer control register BTCTL and selects one of the eight FF outputs.

	7							0
BTCNT2 047h	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Basic Timer1	rw	rw	rw	rw	rw	rw	rw	
Basic Timer	--	--	--	--	--	--	--	--

The output of the counter Q0...7 can be read and the counter (Q0...7) can be written by software in Basic Timer1 only.

9.1.2 Special function register bits

Bits in the SFR address range handle the system control interaction according to the function implemented in the basic timer:

- Basic Timer Module Enable BTME (located in ME2.7), Basic Timer only
- Basic Timer Interrupt Flag BTIFG (located in IFG2.7)
- Basic Timer Interrupt Enable BTIE (located in IE2.7).

In the Basic Timer module, the module enable bit inhibits all functions of the module and reduces the power consumption to its minimum - the leakage current. In Basic Timer1 the Hold bit in the BTCTL register operates similar to the module enable bit. Since the module enable bit BTME in Basic Timer module directly stops the clock signal additional count pulses can occur.

In the Basic Timer1 module the counter itself is enabled or disabled and therefore no additional counts occur. The access of the system to the general module register BTCTL is not affected. It can be read or written in the usual manner.

The interrupt flag and interrupt enable follow the general rules of module interrupts. Beside the individual interrupt enable bit the interrupt request is also controlled by the general interrupt enable bit GIE.

The interrupt enable flag BTIE is reset on PUC. The interrupt flag BTIFG is reset when an interrupt request of the basic timer is accepted or, in the Basic Timer module when the basic timer is reset (Reset* bit in BTCTL). During reset from Reset* bit in BTCTL the interrupt flag BTIFG can not be set by software.

9.1.3 Basic Timer, Basic Timer1 Operation

The basic timer is constantly incremented by the clock ACLK or MCLK. The SSEL control signal selects either the auxiliary clock ACLK (32 768Hz) or the main clock MCLK (system clock f_{system}) for Counter 2.

An interrupt can be used to control system operation. The interrupt is a single source interrupt.

The Basic Timer can operate in two different modes:
- Two independent eight-bit timer/counters
- One sixteen-bit timer/counter

Two eight bit-modes

In the eight-bit mode the basic timer BTCNT1 is incremented constantly with ACLK. In the Basic Timer1 module, when the counter is read the asynchronous behaviour of the counter (ACLK) and the system (MCLK) should be considered. The counter can be written to in software asynchronous to the counter's clock.

The BTCNT2 clock signal can be selected for MCLK, ACLK or ACLK/256 with the control signals SSEL and DIV.

The counter BTCNT2 is incremented with the signal selected.

One of the eight counter outputs can be selected to set the basic timer interrupt flag. In the Basic Timer module the Reset* bit in the control register will reset the counter and simultaneously the interrupt flag BTIFG. In the Basic Timer1 module, read and write access can be asynchronous when a clock other than MCLK is selected.

Sixteen-bit timer/counter mode

The sixteen-bit timer/counter mode is selected with the control bit DIV set. In this mode the clock source of counter BTCNT1/BTCNT2 is ACLK signal.

Basic Timer Module: The Reset bit is active on both eight-bit counters.
Basic Timer1 Module: The Hold bit stops operation of both eight-bit counters.

9.1.4 Basic Timer, Basic Timer1 Operation: Signal f_{LCD}

The peripheral LCD module uses the signal f_{LCD} to generate the timing for common and segment lines. The frequency of the signal f_{LCD} is generated from ACLK. Using a 32 768 Hz crystal at the oscillator the frequency at f_{LCD} is 1024 Hz, 512 Hz, 256 Hz or 128 Hz. The bits FRFQ1 and FRFQ0 allow the correct choice of the frame frequency. The proper frequency f_{LCD} depends on the LCD's characteristic data for the framing frequency and the multiplex rate of the LCD.

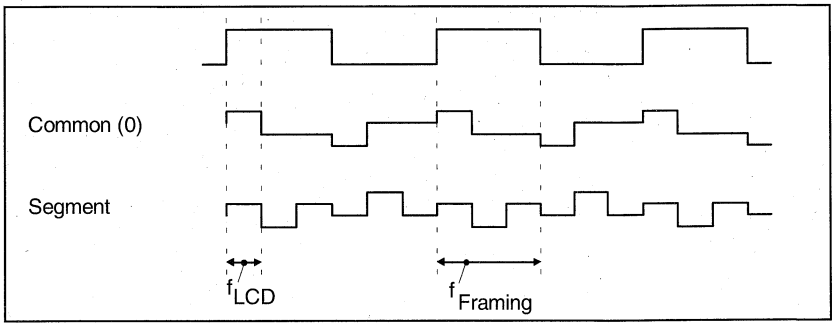


Figure 9.4: Frequency Select for LCD (Example for 3MUX)

Example for 3MUX:

LCD data sheet: $f_{Framing}$ = 100 Hz .... 30 Hz

FRFQ: $f_{LCD} = 3 \times f_{Framing}$

f_{LCD} = 3 x 100 Hz .... 3 x 30 Hz = 300 Hz .... 90 Hz

Select f_{LCD} : 1024 Hz or 512 Hz or 256 Hz or 128 Hz

f_{LCD} = 128 Hz FRQ1 = 1; FRFQ0 = 1

9.2 8-bit Interval Timer/Counter

The 8-bit interval timer supports three major functions for the application:
- serial communication or data exchange
- pulse counting or pulse accumulation
- timer

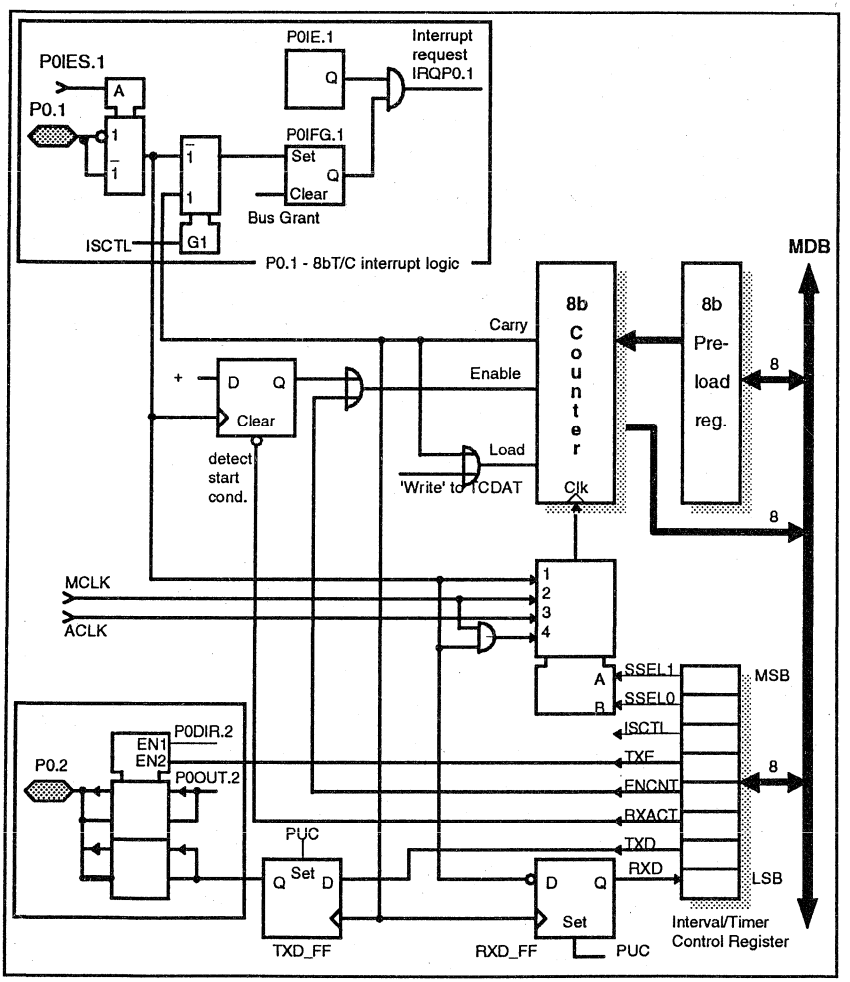


Figure 9.5: Principle Schematic of 8-bit Timer/Counter

9.2.1 Operation of 8-bit Timer/Counter

The 8-bit Timer/Counter includes following major blocks:

- 8-bit Up-Counter with pre-load register
- 8-bit Control Register
- Input clock selector
- Edge detection, e.g. Start bit at asynchronous protocols
- Input and output data latch, triggered by carry-out-signal from 8-bit counter.

8-bit Up-Counter with pre-load register

The 8-bit counter counts up with the input clock selected via two control bits (SEL0, SEL1) of the control register. Two inputs (Load, Enable) at the counter control the operation.

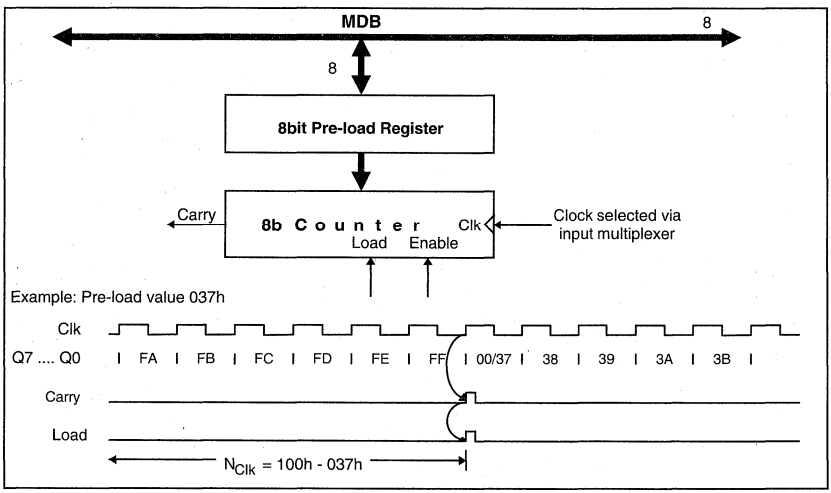


Figure 9.6: Schematic of 8-bit Counter

One of the two inputs control the load function. A load operation loads the counter with the data of the pre-load register. A write access to the counter results in loading the pre-load register contents into the counter.

The software writes or reads the pre-load register with full control with all instructions. The pre-load register acts as a buffer and can be written immediately after the load of the counter has completed.

The second of the two inputs enables the count operation. When the enable signal is set high the counter will count-up each time a positive clock edge is applied to the clock input of the counter.

8-bit Control Register
The information stored in the 8-bit control register selects the operating mode of the timer/counter and controls the function.

Input clock selector
Two signals out of the 8-bit control register select the source for the clock input of the 8-bit up-counter. The four sources are the system clock MCLK, the auxiliary clock ACLK, the external signal from pin P0.1 and the signal from the logical .AND. of MCLK and pin P0.1.

Edge detection
Serial protocols like UART protocol needs start-bit edge detection to determine the start of a data transmission at the receiver.

Input and output data latch, RXD_FF and TXD_FF
The clock to latch data into the input and output data latch is the carry signal from the 8-bit counter. Both latches are used as single bit buffers and change their outputs with the pre-defined timing.

9.2.2 8-bit Timer/Counter Registers

The Timer/Counter module hardware is controlled using access via the 8-bit MDB structure and MAB. It should be accessed using byte instructions.

Register	short form	Register type	Address	Initial state
• T/C control register:	TCCTL	Type of read/write	042h	Reset
• Pre-load register:	TCPLD	Type of read/write	043h	Unchanged
• Counter:	TCDAT	Type of read/(write)	044h	Unchanged

8-bit Timer/Counter Control Register

The information stored in the control register determines the operation of the timer/counter.

TCCTL
042h

7							0
SSEL1	SSEL0	ISCTL	TXE	ENCNT	RXACT	TXD	RXD
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r(-1)

Figure 9.7: 8-bit Timer/Counter Control Register

Bit 0: The RXD bit is read only. The signal from the external pin P0.1 is latched with the carry of the 8-bit counter. The external signal is scanned in a fixed timing sequence independent of the different run-times from software.

Bit 1: The TXD register bit is the buffer for the TXD signal clocked out with the carry from the 8-bit counter at the pin P0.2.

Bit 2: The RXACT bit controls the edge detect logic. The edge detect logic needs a reset ENCNT bit (bit 3) for proper counter enable operation.
RXACT = 0: The edge detect FF is cleared and it can not be the source of enabling the counter operation.
RXACT = 1: The edge detect FF is enabled for operation. A postive or negative edge at pin P0.1, selected by P0IES.1, sets the FF and the counter is prepared for count operation. If the FF is set it remains set.

Bit 3: The ENCNT bit sets the counter enable signal. The 8-bit counter increments its value with each rising edge at clock input.
Together with the RXACT bit (bit2, '0') this bit provides start/stop operation.

Bit 4: The TXE signal controls the 3-state output buffer for the TXD bit.
TXE = 0: 3-state
TXE = 1: Output buffer active.

Bit 5: The ISCTL signal controls the interrupt source between the I/O pin P0.1 and the carry of the 8-bit counter.
ISCTL = 0: The I/O pin P0.1 is the source of interrupt P0IFG.1.
ISCTL = 1: The carry from the 8-bit counter is the source of interrupt P0IFG.1.

BIT 6,7: The bits SSEL0 and SSEL1 select source of the clock input.

SSEL1	SSEL0	Clock source
0	0	Signal at pin P0.1 according to P0IES.1
1	0	MCLK
0	1	ACLK
1	1	Signal pin P0.1(according to P0IES.1) .AND. MCLK

8-bit Timer/Counter Pre-load Register
The information stored in the pre-load register is loaded into the 8-bit counter when a write access to the counter (TCDAT) is performed:

```
;=========   Definitions ==========================================
Dummy       .EQU   0       ; Value for dummy is not loaded into
                           ; counter
TCDAT       .EQU   044h    ; Address of 8-bit Timer/Counter
;=========   Write pre-load register content to 8-bit Timer/Counter =
;
            MOV.B  #Dummy,&TCDAT
;
```

The pre-load register (TCPLD) can be accessed using the address 043h.

8-bit Counter Data
The data of the 8-bit counter can be read using the address 044h. Writing to the counter loads the content of the pre-load register - not the data mentioned by the instruction.

9.2.3 Special function register bits, 8-bit Timer/Counter related

The 8-bit Timer/Counter has no individual interrupt bits; it shares the interrupt bits with the port P0. One bit in the control register TCCTL, the bit ISCTL, selects the interrupt source for the interrupt flag.

The port0 signal P0/RXD.1 or the carry of the 8-bit counter is used for interrupt source. Two bits in the SFR address range and one bit in the port0 address frame handle the interrupt events on P0/RXD.1 :
- P0/RXD.1 Interrupt Enable P0IE.1 (located in IE1.3, initial state is reset)
- P0/RXD.1 Interrupt Edge Select P0IES.1(located in P0IES, initial state is reset)

The interrupt flag is a single source interrupt flag and is automatically reset when the processor system serves it. The enable bit and edge select bit remain unchanged.

9.2.4 8-bit Timer/Counter in UART Applications

The Timer/Counter peripheral incorporates some features to support serial data exchange with software. The data exchange consists of the transmit cycle and receive cycle. The peripheral hardware supports half duplex protocols.

Software operation can be seperated into three categories to support the different conditions and requirements of individual applications:
- Control the bit information immediately after each receive cycle
- Control all bits of one frame immediately after each receive cycle
- Receive the complete message, store the frames in memory and check it after completion of receive cycle.

UART Protocol

The UART protocol is a serial bit stream which includes start bit, 1 to 8 data bits, an optional parity bit, an optional address bit and 1 or 2 stop bits. The least significant data bit is sent first after the start bit.

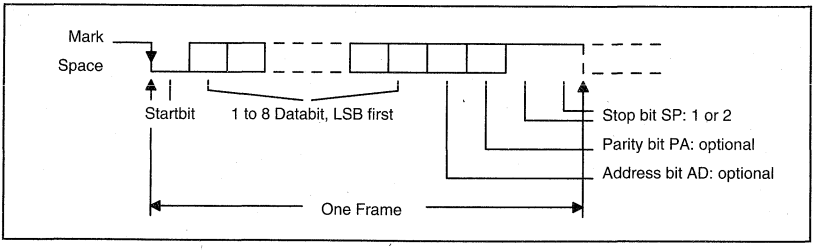


Figure 9.8: Asynchronous communication format

UART Protocol Receive Mode

The Timer/Counter acts as a timer and the carry out latches the bit information available at the pin P0.1. The negative edge - from mark to space of the start bit - indicates the start of one frame. Each bit is scanned right in the middle.

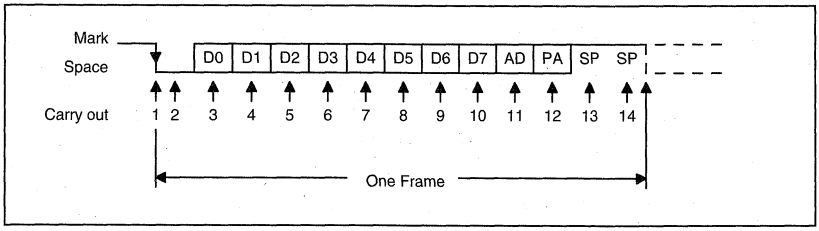


Figure 9.9: Scanning of the asynchronous bits of one frame

The software UART is closely combined with the start of a receive cycle and the baud rate. All timings vary from bit-to-bit if a reference or clock frequency clocks the timer at a frequency which is not a multiple of the selected baud rate. The example in this section should run with a baud rate of 2400 baud with a crystal frequency of 32,768 Hz. The result of these conditions is that each bit interval has got its own timing and therefore its own pre-load value.

UART Protocol Transmit Mode

The Timer/Counter acts as a timer and the carry out latches the bit information available from the control register TCCTL, bit 1: TXD. The software UART should load the TXD bit with the information which should be on I/O pin P0.2. The next carry from the timer latches the TXD bit into the TXD_FF. The transmission of data out of I/O pin P0.2 is enabled by setting of bit TXE in the control register. The reset state of the signal TXE disables the output buffer connected to pin P0.2 and sets parallel the TXD_FF output. This corresponds to the mark state defined for UART format.

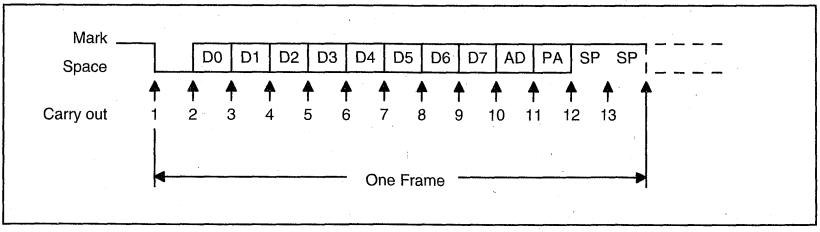


Figure 9.10: Transmitting of the asynchronous bits of one frame

The timing of the transmit part of the software UART depends on the baud rate. All timings vary from bit-to-bit if a reference or clock frequency clocks the timer in non-multiple frequency from the selected baud rate. The example in this section should run with a baud rate of 2400 baud with a crystal frequency of 32,768 Hz. The result of these conditions is that each bit interval has got its own timing and therefore its own pre-load value.

Each communication needs features to recognize errors that happen during the data transmission. Four different error conditions are defined:

- Parity Error
- Overrun Error
- Framing Error
- Break detect

In addition to the previous fundamentals there is an optional function included to support protocol handling: the identification of the start of a block of frames and the destination of the telegram. Two different modes for identifications are used in the industry, the idle line multiprocessor protocol and the address bit multiprocessor protocol.

Idle line multiprocessor mode format

The blocks of data are seperated by idle time between them. An idle receive line is detected when ten or more mark state (1's) in a row are received after the first stop bit of a character. When two stop bits are used, the second is counted as the first mark bit of the idle. The first character received after an idle period is an address character. The idle line periods detected by the receiver are illustrated with one and two stop bits:

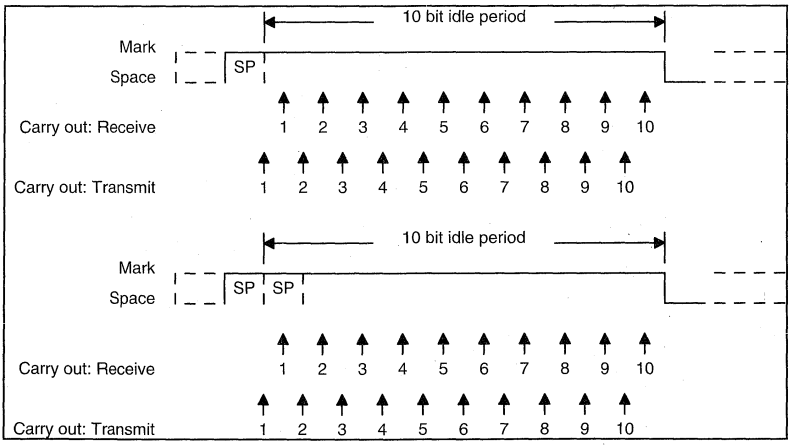


Figure 9.11: UART idle period

It is recommended to transmit an idle period of 11 bits instead of 10 bits.

The precise idle period generates an efficient address character identifier. The first character of a block of frames can be identified as an address. The idle periods of frames within a block of information should not exceed the idle period detect time of 10 bits.

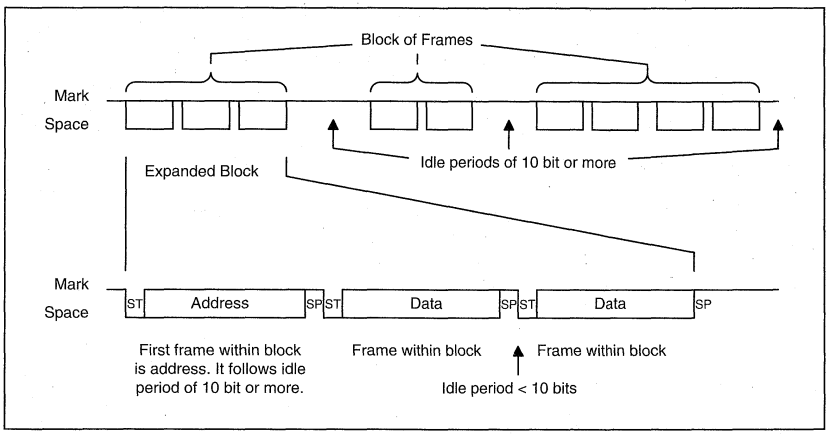


Figure 9.12: Idle line multiprocessor protocol

Address bit multiprocessor mode format

Each character contains an extra bit that represents an address indicator. The first character in a block of data (frames) carries an address bit that is set. This indicates that the character is an address.

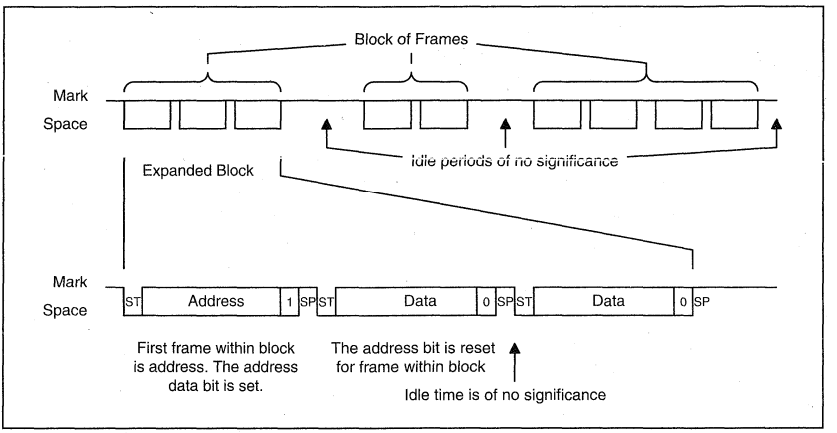


Figure 9.13: Address bit multiprocessor mode format

Transmit/Receive Application Example

This programming example of a serial UART communication protocol using the features of the 8-bit Timer/Counter has these characteristics:

- Baud rate 2400 baud
- ACLK = 32 768 Hz
- Parity Even
- Two stop bits
- Half duplex

The carry signal of the Timer/Counter module is selected for the interrupt source instead of P0.1 source. The associated vector contains the address of the transmit/receive interrupt routine. The first instructions seperate the program flow into the transmit or receive part and use the bit TXRX as an indicator for the running mode.

```
; --------     Define interrupt vector ------------------------------
             .SECT  "RXTX_vec",FFF8     ;Vector of P0.1 or carry from
                                        ;8-bit Timer/Counter
             .Word      VECRXTX         ;Address of UART
                                        ;handler
             ..........
```

The time intervals between two carry signals differs during each transmit or receive cycle. The selected baud rate of 2400 baud and the clock frequency of 32 768 Hz would require a divider of $\frac{32768}{2400}$ = 13.67. The deviation from this ideal factor is accomplished using the sequence 14-13-14 for the division.

```
; Definitions of used expressions
RXD         .EQU  1                 ; Receive data bit in control register TCCTL
TXD         .EQU  2                 ; Transmit data bit in control register TCCTL
RXACT       .EQU  4
ENCNT       .EQU  8                 ; Counter enable bit in control register TCCTL
TXE         .EQU  010h              ; 1: TX buffer active,  0: TX buffer 3-state
ISCTL       .EQU  020h
TCCTL       .EQU  042h              ; Address of Timer/Counter control register
TCPLD       .EQU  043h              ; Address of Timer/Counter pre-load register
TCDAT       .EQU  044h              ; Address of Timer/Counter
BitTime1    .EQU  0100h - 0Eh       ; 14/32 768 sec. = 427.2µs bit length
BitTime1_2  .EQU  0100h - 07h       ; Half of bitime1
BitTime2    .EQU  0100h - 0Dh       ; 13/32 768 sec. =  396.7 µs bit length
AdP0_0      .EQU  0h                ; Interrupt enable 1 register address (SFR)
IEnP0_0     .EQU  08h               ; Bit in Interrupt enable 1 register (SFR)
ParVal      .EQU  0h                ; Parity Even selected
;
(P0.1 respectively TC interrupt-enable)
; Registers or RAM used for data handling
RCstatus    .EQU  0200h             ; RAM (or Register), stores actual status of
                                    ; receive sequence
TXStatus    .EQU  0201h             ; RAM (or Register), stores actual status of
                                    ; transmit sequence
TXData      .EQU  R6                ; Register that contains the transmit data
RCData      .EQU  R6                ; Stores receive data (RXD) in HighByte
Parity      .EQU  0yyyh             ; LSB is actual status of parity. The start value
                                    ; determines odd or even parity
Bend        .EQU 2 x 12
```

The main loop in the program for both the transmit and receive function of an information sequence is demonstrated using the outputting of a table's data and receiving and storing data into a table:

```
; ---------- Transmitting of frames: a table is to be output ----------
; Ry points to the table
;
            MOV     #Table2,Ry      ; Start of table copied to Ry
L$5         CRL.B   &TXStatus
            CMP     #TabEnd+1,Ry    ; All frames transmitted?
            JEQ     TabFin          ; Yes, stop transmision and continue program
            MOV.B   @Ry+,TXData     ; Info to TXData
            CALL    #TXInit         ; No, initialize transmission
TXStat      CMP     #Bend,TXStatus  ; output of one frame completed?
            JEQ     L$5             ; Yes, transmit next data of table!
            JMP     TXStat          ; No, wait for completion
            ......
```

```
                .....
Table2          .           0xxh Byte       ; Start of table containing data for transmission
                .....
                .....
TabEnd          .           0zzh Byte       ; End of table containing data for transmission
                .....
TabFin          .....                       ; Transmission of table is completed
                                            ; Continue program here

; ---------- Prepare receiving of one frame ----------
; Rx points to the table
;
                MOV     #Table1,Rx          ; Start of receive table copied into Rx
                CALL    #RCPrep             ; Receive of next frame
                .....
; ---------- Processing part of received frame: Store frame in table1 ----------
RECCMPL         RLA     RCData              ; Adjust info to HighByte (remove parity bit)
                SWPB    RCData              ; Swap info to LowByte
                MOV.B   RCData,0(Rx)        ; Store info in table1
                INC     Rx                  ; and pre-increment of table pointer
                CALL    #RCInit             ; Prepare for next frame
                .....                       ; Continue with background program
```

Transmit Mode Application Example:

2400 Baud, ACLK, 8 data bits, Parity Even, Two Stop bits

The transmit mode uses the 8-bit timer/counter, the pre-load register, the control register, the clock selector and the TXD data latch.

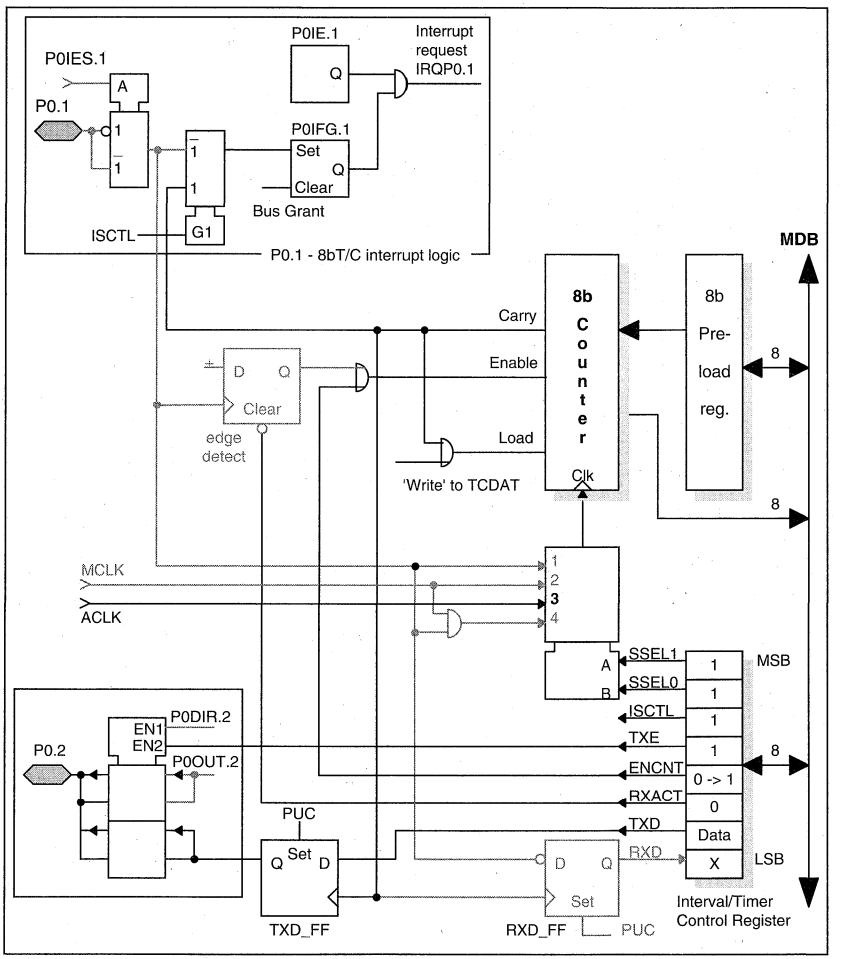


Figure 9.14: 8-bit Timer/Counter configuration for transmit example 2400Baud, ACLK clock

Before a serial communication character transmit starts there are some operation conditions to be defined:

- The output buffer should be enabled -> TXE bit is set.
- The clock input into the 8-bit timer should be selected -> ACLK is selected with SSEL0 bit is reset and SSEL1 bit is set.
- The interrupt source control bit ISCTL selects carry out of timer.

The appropriate bits regarding P0.1 direction and interrupt edge bits should be chosen properly.

- The pre-load register is loaded.
- The write access to the counter loads the pre-load value into the timer .
- The RXACT bit is reset.

```
; ---------- Prepare Transmit Cycle ----------------------------------------------------------
TXINIT    MOV.B   #BitTime1_2,&TCPLD  ; Load time until start bit starts
                                      ; Disable P0.1 O/P buffer (direction in) if
                                      ; needed
          MOV.B   #072h,&TCCTL        ; TXD = 1 : Defined Start,  ACLK selected
                                      ; TXEN = 1
          MOV.B   #0,&TCDAT           ; Dummy write to load 8b counter/timer
          MOV.B   #BitTime1,&TCPLD    ; Load bit time of first bit for transmission
                                      ; into pre-load register, time of Startbit
          BIS.B   #ENCNT,&TCCTL       ; Set transmit start condition
          BIS.B   #IEnP0_0,&AdP0_0    ; Interrupt enabled for P0.1 in SFR,
                                      ; address is 0.
          CLR.B   &TXStatus           ; Temporary register is prepared.
          MOV.B   #ParVal,Parity      ; ParVal = 0 for Even, ParVal = 1 for Odd
                                      ; Parity
          RET
```

```
; ---------- Acknowledge Transmit/Receive Cycle, UART Handler ----------
; The following two instructions decide whether to transmit or to receive data
; It is necessary because they use a common interrupt vector address
VECRCTX BIT.B   #RXACT,&TCCTL  ; Test which interrupt handler is active
        JNZ     RCINTRPT       ; Receive mode is active -> Jump
;
TXINTRPT PUSH   R5             ; RXACT = 0 --> Transmit
        MOV.B   &TXStatus,R5   ; Use TXStatus for
        BR      TXTAB(R5)      ; branching
TXTAB   .Word   TXStat0        ; Startbit     ; Bitime2, 13 clocks of ACLK
        .Word   TXStat1        ; Bit 0, LSB   ; Bitime1, 14 clocks of ACLK
        .Word   TXStat1        ; Bit 1        ; Bitime1, 14 clocks of ACLK
        .Word   TXStat2        ; Bit 2        ; Bitime2, 13 clocks of ACLK
        .Word   TXStat1        ; Bit 3        ; Bitime1, 14 clocks of ACLK
        .Word   TXStat1        ; Bit 4        ; Bitime1, 14 clocks of ACLK
        .Word   TXStat2        ; Bit 5        ; Bitime2, 13 clocks of ACLK
        .Word   TXStat1        ; Bit 6        ; Bitime1, 14 clocks of ACLK
        .Word   TXStat1        ; Bit 7        ; Bitime1, 14 clocks of ACLK
        .Word   TXPar          ; Parity bit   ; Bitime2, 13 clocks of ACLK
        .Word   TXStop         ; Stop bit 1   ; Bitime1, 14 clocks of ACLK
        .Word   TXStop         ; Stop bit 2   ; Bitime1, 14 clocks of ACLK
        .Word   TXCCmpl        ; Frame transmitted
TXStat0 BIC.B   #TXD,&TCCTL
        MOV.B   #BitTime2,&TCPLD
                               ;Load time 13/32768 [sec] into pre-load register
        JMP     TXRET
TXStat2 MOV.B   #BitTime2,&TCPLD
                               ;Load time 13/32768 [sec] into pre-load register
        JMP     L$3            ; Shift next bit out at P0.2
TXStat1 MOV.B   #BitTime1,&TCPLD
                               ;Load time 14/32768 [sec] into pre-load register
L$3     RRA     TXData         ; LSB is shifted to Carry
        JNC     L$1            ; Jump to L$1 if bit = 0
L$2     BIS.B   #TXD,&TCCTL    ; Bit =1, set TXD bit in control register TCCTL
        XOR.B   Parity         ; Count 1's for parity
        JMP     TXRET          ; Bit output completed
L$1     BIC.B   #TXD,&TCCTL    ; Bit =0, reset TXD bit in control register TCCTL
TXRET   INCD.B  &TXStatus      ; Bit output completed
TXStat12 POP    R5
        RETI                   ; Transmit of one bit completed
; ---------- Parity bit check: Count of 1's in Parity must be even ----------
TXPar   MOV.B   #BitTime2,&TCPLD
        BIT.B   #1,Parity      ; Check parity bit value
        JNZ     L$2            ; Parity bit should be Mark
        JMP     L$1            ; Parity bit should be Space
; ---------- Output of stop bit(s) ----------
TXStop  MOV.B   #BitTime1,&TCPLD
        JMP     L$2            ; Send stop bit 1 or 2
```

```
; ---------- Output of one frame completed ----------
TXCmpl  BIC.B   #IEnP0_0,&AdP0_0 ; Interrupt disabled for P0.2 in SFR,
                                 ; address is 0.
;       BIC.B   #ENCNT,&TCCTL; Stop counter to conserve power consumption
        JMP     TXStat12
; ---------- End of transmit interrupt handler ----------
```

Receive Mode Application Example:

2400 Baud, ACLK, 8 data bits, Parity Even, Two Stop bits
The receive mode uses the 8-bit timer/counter, the pre-load register, the control register, the clock selector, the edge detect logic and the RXD data latch.

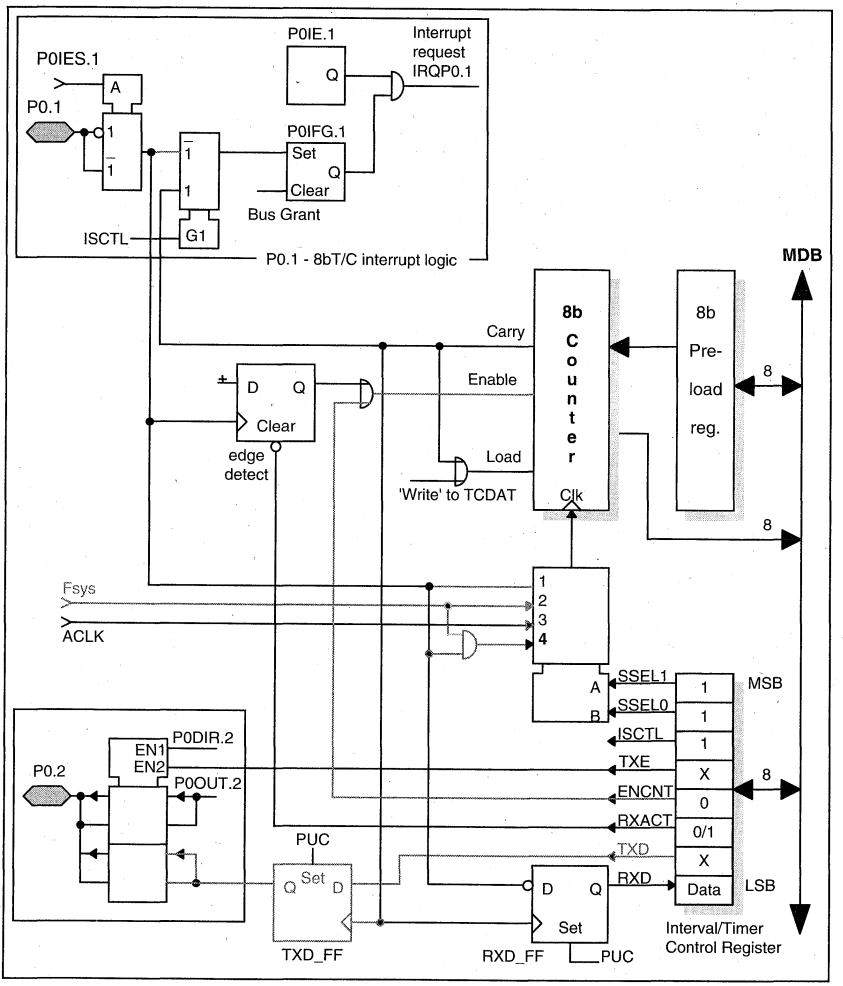


Figure 9.15: 8-bit Timer/Counter configuration for receive example 2400Baud, ACLK clock

Before a serial communication character receive starts there are some operation conditions to be defined (assuming RXACT bit is reset):

- The output buffer should be disabled -> TXE bit is reset.
- The clock input into the 8-bit timer should be selected -> ACLK is selected with SSEL0 bit is reset and SSEL1 bit is set.
- The interrupt source control bit ISCTL selects carry out of timer.

The appropriate bits regarding P0.1 direction and interrupt edge bits should be chosen properly.

- The pre-load register is loaded.
- The write access to the counter loads the pre-load value into the timer .
- The RXACT bit is set.

```
; ---------- Prepare Receive Cycle ----------------------------------------------------------------
RCPREP  MOV.B   #062h,&TCCTL    ; SSEL0 = 0, SSEL1 = ISCTL = 1,
                                ; all other bits are cleared
                                ; Select ACLK for clock source to 8-bit timer
                                ; Use #072h if TXEN should be enabled
RCINIT  MOV.B   #BitTime1_2,&TCPLD
                                ; Set Preload register with t1-2 = 0100h - 7
        MOV.B   #0,&TCDAT       ; Prepare timer interval for start bit scanning
        MOV.B   #BitTime1,&TCPLD
                                ; Set Preload register with Bittime 1 for receive
                                  of first data bit
        CLR     RCstatus        ; Prepare temporary registers
        MOV.B   #ParVal,Parity  ; Register Parity=0 for Even parity receive mode
                                ; and Parity=1 for Odd parity
        BIS.B   #RXACT,&TCCTL ; activate neg. edge detect of P0.1
                                ; ( -> RX data )
        BIS.B   #IENP0_0,&ADP0_0
                                ; Enable interrupt according to P0.1
                                ; Interrupt source is carry from 8-bit timer
                                ; according to state of ISCTL
        RET
```

As long as the RXACT and ENCNT bit are reset the timer/counter is halted. The change to the receive active state with a set of RXACT bit enables negative edge detection. The first edge of the start bit, applied to pin P0.1, set the output of the edge detect latch. It will be set until another reset of RXACT bit is performed.

Once the edge detect latch is set the time starts operation. The first time interval is started and with the elapse of the programmed time the logical level of pin P0.1 is latched into the RXD latch. After that activity the interrupt is requested.

The interrupt routine for the first bit is optional and can test the presence of a start bit. In the absence of the start bit, the receive cycle is stopped. When the receive cycle is continued the next timing should be prepared by loading the pre-load register with proper data.

All further bits follow nearly the same process in the interrupt routine..

The interrupt routine should handle:

- Store RXD bit information
- Prepare next timing
- Optional: update parity information
  decide program flow on parity error
  look for address bit information
- Test stop bit if received bit should be stop bit.

> **Note: UART protocol, LSB/MSB sequence**
>
> UART protocol shifts the LSB of the data first. In order to collect the data properly use the RRC instruction to have the correct order of bits.

```
; ---------- Receive interrupt handler ----------------------------------------------------------
RCINTRPT  PUSH    R5                ; Receiver interrupt routine
                                    ; R5 is used temporary as pointer
                                    ; of receive bit
          MOV.B   &RCstatus,R5 ;
          BR      RCTAB(R5)         ;
;
RCTAB     .Word   RCstat0           ; Receive start bit
                                    ; set receive time bit0/LSB, 13ACLK
          .Word   RCstat1           ; Receive bit 0  ; set receive time bit1,
                                    ; 14ACLK
          .Word   RCstat1           ; Receive bit 1  ; set receive time bit2,
          .Word   RCstat2           ; Receive bit 2  ; set receive time bit 3
          .Word   RCstat1           ; Receive bit 3  ; set receive time bit 4
          .Word   RCstat1           ; Receive bit 4  ; set receive time bit 5
          .Word   RCstat2           ; Receive bit 5  ; set receive time bit 6
          .Word   RCstat1           ; Receive bit 6  ; set receive time bit 7: MSB
          .Word   RCstat1           ; Receive bit 7  ; set receive time parity bit
          .Word   RCstat2           ; Receive parity bit
                                    ; set receive time stop bit 1
          .Word   RCstop1           ; Receive stop bit 1
                                    ; set receive time stop bit 2
          .Word   RCstop2           ; Receive stop bit 2
                                    ; set receive time stop bit 2
          .Word   RCCmpl            ; Frame received
; ---------- Receive start bit: Test for space --------------------------------------------------
RCstat0   BIT.B   #RXD,&TCCTL ; Check start bit
          JC      RCError           ; Error: start bit is Mark not Space
          MOV.B   #BitTime2,&TCPLD ; Start bit fine, load pre-load
                                    ; register
          JMP     RCRET
;
```

```
RCstat2     MOV.B   #BitTime2,&TCPLD
                                    ; Load pre-load register with bit time 2
            JMP     RCBit
RCstat1     MOV.B   #BitTime1,&TCPLD
                                    ; Load pre-load register with bit time 1
RCBit       BIT.B   #RXD,&TCCTL ; RXD bit -> Carry bit
            JNC     RCRET           ; RXD bit = Carry bit = 0 ?, Yes, jump
            RRC     RCData          ; RXD bit -> MSB, Negative bit
            INC.B   &Parity         ; RXD bit = 1, increment '1'-counter
JMP         RCRET1
RCRET       RRC     RCData          ; RXD bit -> MSB, Negative bit
RCRET1      INCD.B  &RCstatus
RCCmpl      POP     R5
            RETI
;
; Parity bit was received just like all other bits. During first stop bit parity is tested
;
RCstop1     BIT.B   #1,&Parity      ; Check parity bit. Bit must be zero.
            JNZ     RCError         ; Parity bit false.
;
RCstop2     MOV.B   #BitTime1,&TCPLD ; Load pre-load register with bit time 1
            BIT.B   #RXD,&TCCTL     ; Check stop bit for Mark
            JNZ     RCRET           ; Stop bit is Mark -> Ok
;
; Error handling: a new start is tried
            RCError POP R5
            CALL    RCINIT          ; Initialize receive routine again
            RETI
```

9.3 The Watchdog Timer

The primary function of the Watchdog Timer module (WDT) is to perform a controlled system restart after a software problem has occurred. If the selected time interval expires, a system reset is generated. If this watchdog function is not needed in an application, the module can work as an interval timer, which generates an interrupt after the selected time interval.

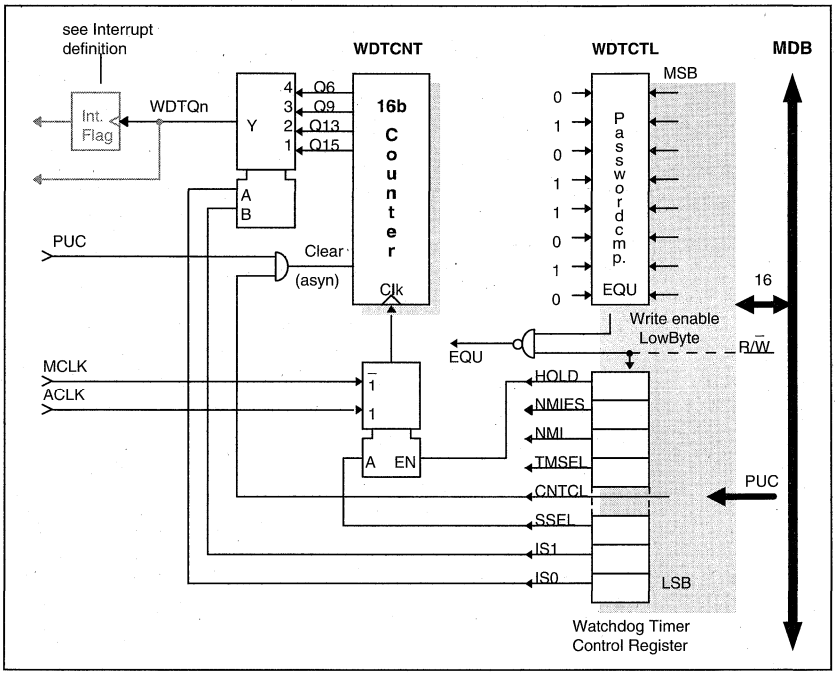


Figure 9.16: Schematic of Watchdog Timer

Features:

- eight software selectable time intervals
- two operating modes: as watchdog or interval timer
- expiration of time interval in watchdog mode generates a system reset, in timer mode an interrupt request
- for safety reasons, writing to the WDT control register is only possible using a password
- supports ultra-low power feature using hold mode

9.3.1 Watchdog Timer Register

The watchdog timer counter WDTCNT is a 16-bit up-counter which is not directly accessible by software. The WDTCNT is controlled through the watchdog timer control register WDTCTL, which is a 16-bit read/write-register located at the low byte of word address 0120h. Any read or write access should be done with word instructions, using no suffix or suffix '.w'. Writing to WDTCTL is, in both operating modes (watchdog or timer), only possible in conjunction with the correct password.

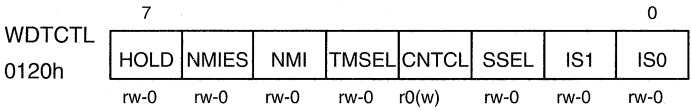


Figure 9.17: Watchdog Timer Control Register

Bits 0,1: The bits IS0,IS1 select one of four taps from the WDTCNT.
Assuming fcrystal = 32 768 Hz and fSystem = 1 MHz, the following intervals are possible:

SSEL	IS1	IS0	interval [ms]	
0	1	1	0.064	tMCLK x 26
0	1	0	0.5	tMCLK x 29
1	1	1	1.9	tACLK x 26
0	0	1	8	tMCLK x 213
1	1	0	16.0	tACLK x 29
0	0	0	32	tMCLK x 215 <- Value after PUC (Reset)
1	0	1	250	tACLK x 213
1	0	0	1000	tACLK x 215

Bit 2: The SSEL bit selects the clock source for WDTCNT.
SSEL = 0: WDTCNT is clocked by the system frequency
SSEL = 1: WDTCNT is clocked by ACLK, the crystal frequency (32 768 Hz)

Bit 3: CNTCL bit: In both operating modes writing a '1' to this bit restarts the WDTCNT at 00000h. The read value is not defined.

Bit 4: The bit TMSEL selects the operating mode: watchdog or timer.
TMSEL = 0: Watchdog mode
TMSEL = 1: Interval timer mode

BIT 5: The NMI-Bit selects the function of the RST/NMI-input pin. It is cleared after PUC.
NMI = 0: The RST/NMI input works as Reset input.
As long as the RST/NMI-pin is held 'low', the internal PUC-signal is active (level sensitive).
NMI = 1: The RST/NMI input works as edge sensitive non-maskable interrupt input.

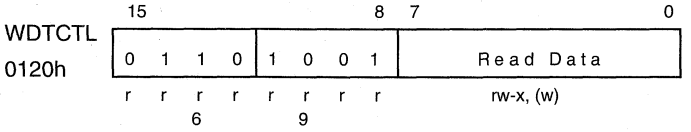
BIT 6: This bit selects the activating edge of the RST/NMI input if NMI function is selected. It is cleared after PUC.
NMIES = 0: A rising edge triggers a NMI-interrupt.
NMIES = 1: A falling edge triggers a NMI-interrupt.

Bit 7: This stops the complete operation of the watchdog counter. It is mandatory to support the ultra-low power features. The clock multiplexer is disabled and the counter stops incrementing. It holds the actual state until the HOLD bit is reset and the operation continues. It is cleared after PUC.
HOLD = 0: Function is fully active.
HOLD = 1: Clock and counter are stopped

Accessing WDTCTL Watchdog Timer Control Register

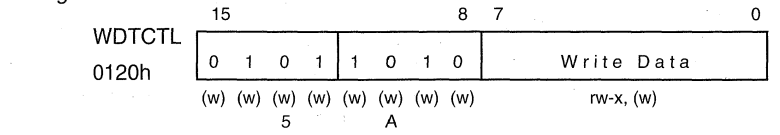
- Read access:
  WDTCTL can be read without restriction by a password. A read is performed by simply accessing word address 0120h. The Lowbyte contains the value of WDTCTL. The value of the Highbyte is 069h. The value of the Highbyte is selected to 069h and limits the effect of instructions that can alter the WDTCTL register.

  Reading WDTCTL:



- Write access:
  A write access to WDTCTL is only possible using the correct password in the high-byte. Changing the WDTCTL-register is performed by writing to word address 0120h. The low-byte contains the data to be written to WDTCTL and the high-byte has to be the password which is 05Ah. If any other value than 05Ah is written to the high-byte of address 0120h a system reset is generated.

Writing WDTCTL:



9.3.2 Watchdog Timer interrupt control functions

The Watchdog Timer uses two bits in the SFR address range:
- WDT Interrupt Flag WDTIFG (located in IFG1.0, initial state is reset)
- WDT Interrupt Enable WDTIE (located in IE1.0, initial state is reset)

The WDT interrupt flag is reset when power is applied or a reset from the $\overline{\text{RST}}$/NMI pin is performed. The signal is called POR. The watchdog interrupt flag indicates whether the watchdog was the reason for a PUC or a power/reset. The vector address is in address 0FFFEh. The enable bit is not relevant.

The Watchdog Timer operates in two different modes. When the WDT is configured to operate in watchdog mode both a watchdog overflow and a security violation trigger the PUC signal which automatically clears the appropiate register bits in the entire system. For the bits in the WDTCTL register it results in a system configuration where the WDT is set into the watchdog mode and the $\overline{\text{RST}}$/NMI pin is switched to reset configuration.

When the WDT is configured to operate in timer mode the WDTIFG flag is set after the selected time interval and it requests a standard interrupt service. The WDT interrupt flag is a single source interrupt flag and is automatically reset when the processor system serves it. The enable bit remains unchanged. The WDT interrupt enable bit and the GIE bit should be set to allow an interrupt request situation. The vector address of the interrupt in timer mode is different from that in watchdog mode.

9.3.3 Watchdog Timer Operation

The Watchdog Timer module can be configured in two modes, the watchdog mode and the interval timer mode.

Watchdog mode

After power-on reset or a system reset the Watchdog Timer module automatically enters the watchdog mode with all bits in watchdog control register WDTCTL and watchdog counter WDTCNT cleared. The initial conditions at the WDTCTL register result in a time interval of 32 ms @ f_{sys}=1 MHz. Since also the digital controlled oscillator DCO in the system clock generator is set to its lowest frequency about 32 600 cycles are available for the software to react on such a drastic event. The initial conditions were selected to run the WDCNT with the system frequency f_{sys} and to allow the application software to start operation with a compromise of the watchdog time in the middle of the time frame.

When the module is used in watchdog mode, the software must periodically reset WDTCNT by writing a '1' to bit CNTCL of WDTCTL to prevent expiry of the selected time interval. If a software problem occurs and the time interval expires because the counter is not reset anymore, a reset is generated and system power-up clear PUC is activated. The system restarts at the same program address as after power-up. The cause of reset can be determined by testing bit0 in the Interrupt Flag Register 1 in the SFR block. The appropriate time interval is selected by setting the bits SSEL, IS0 and IS1 accordingly.

Timer mode

Setting bit TMSEL in the WDTCTL register to '1' selects the timer mode. This mode provides periodic interrupts at the selected time interval. A time interval can also be started under software control by writing a '1' to bit CNTCL in the WDTCTL register.

Note: Watchdog Timer, changing the time interval

Changing the time interval without clearing the WDTCNT may result in an unexpected immediate system reset or interrupt. The time interval should be changed together with a counter clear in one instruction e.g. MOV #05A0Ah, &WDTCTL. Sequential clear and interval select may result in an unexpected immediate system reset or interrupt.

Changing the clock source during normal operation may result in additional clocks for the WDTCNT.

Operation inlow- power modes

The system check generator can run in five different modes. With three of them the MCLK and ACLK signals are active. During one mode only the ACLK signal is active and during the other remaining mode neither MCLK nor ACLK signal is active.

The application requirements set the handling of the watchdog timer in combination with the hardware reaction to the different operating conditions of ACLK and MCLK.

CPUOFF mode: Program execution is stopped. The software should define the operating conditions during this operating mode.

MCLKOFF mode/LPM2..3: The ACLK signal is active and MCLK is inactive. When ACLK (32 768 Hz) is selected, the watchdog timer continues operation and will awake the CPU through a system reset or a timer interrupt (if enabled) depending on the selected operating mode. When MCLK is selected the WDT halts operation until MCLK is restarted.

OSCOFF mode/LPM4: The MCLK and ACLK signal are inactive and the watchdog timer counter halts until the system is restarted. The software can reset the counter before entering the OscOff mode depending on the application needs.

The provision of a hold function supports ultra-low power operation. Wherean application uses various low power modes, the watchdog timermay be held.

Software example

```
; After RESET or power-up, the WDTCTL register and WDTCNT are cleared and the initial
; operating conditions are watchdog mode with a time interval of 32 ms.
;
;Constant definitions:
;
WDTCTL    .EQU    0120h       ; Address of Watchdog timer
WDTPW     .EQU    05A00h      ; Password
T250MS    .EQU    5           ; SSEL, IS0, IS1 set to 250 ms
T05MS     .EQU    2           ; SSEL, IS0, IS1 set to 0.5 ms
CNTCL     .EQU    8           ; Bit position to reset WDTCNT
TMSEL     .EQU    010h        ; Bit position to select timer mode
;
; As long as watchdog mode is selected, watchdog reset has to be done periodically
; through a instruction e.g.:
;
            .....
            .....
            MOV     #WDTPW+CNTCL,&WDTCTL
      .
      .
;
;To change to timer mode and a time interval of 250 ms, the following instruction sequence
; can be used:
;
            MOV     #WDTPW+CNTCL+TMSEL+T250MS,&WDTCTL
                                        ; Clear WDTCNT and
                                        ; select 250 ms and timer
                                        ; mode
            .....
            .....
; Note: The time interval and clear of WDTCNT should be modified within one
        instruction to avoid unexpected reset or interrupt
;
; Switching back to watchdog mode and a time interval of 0.5 ms is performed by:
;
            .....
            .....
            MOV     #WDTPW+CNTCL,&WDTCTL    ; Reset WDT counter
;
            MOV     #WDTPW+T05MS,&WDTCTL    ; Select watchdog mode
                                            ; and 0.5 ms
            .....
```

9.4 8-bit PWM Timer

Using an 8-bit timer counter, PWM peripheral generates a rectangular output pulse with a duty factor of 0 to 100%. The duty factor is specified by an 8-bit duty control register PWMDT.

The PWM timer module has the following features:
- Selection of eight clock sources
- Duty factors from 0 to 100% with 1/254 resolution
- Output with positive or negative logic

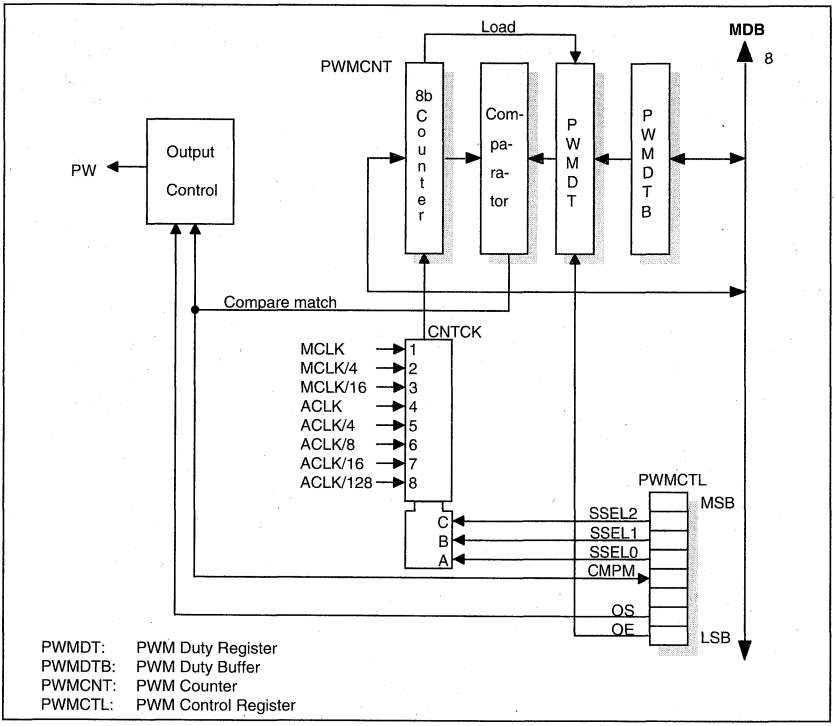


Figure 9.18: Block Diagram of PWM Timer

9.4.1 Operation

The operation of the PWM timer is described with the output polarity control signal OS reset. The value of the PWMDT register represents the number of clock pulses when PWM output is high.

When OE = 0, the timer count is held at 00h and the PWM output is reset. Any value written into the PWMDT becomes valid immediately.

When OE = 1, the timer counter begins incrementing, and the PWM output goes High (situation b in figure).

When the count reaches the PWMDT value, the PWM output goes Low (situation c in figure) with the next clock pulse.

If the PWMDT value is changed (by writing the data M in figure), the new value becomes valid after the timer count changes from FDh to 00h (situation d in figure)

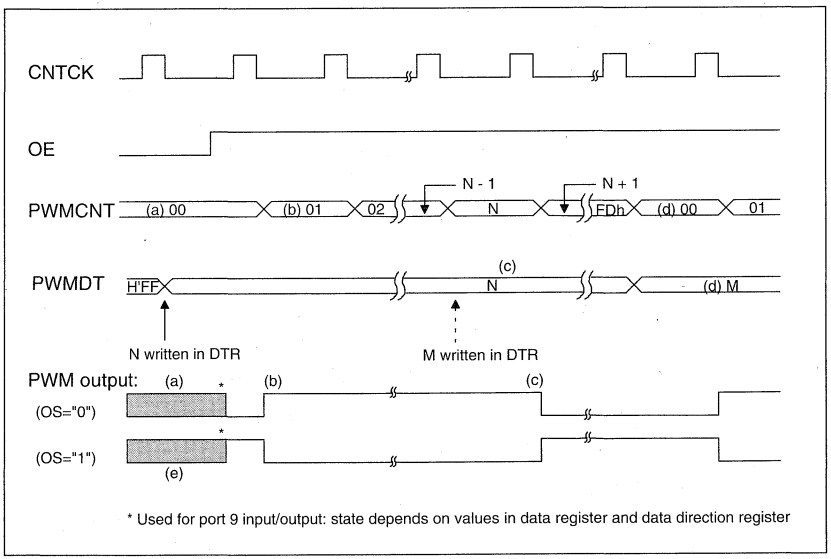


Figure 9.19: PWM timing scheme

The control flag OS in the PWMCTL register defines the polarity of the PWM output.

When OS=0, value in the PWMDT register represents the number of PWM counter clock pulses where the PWM output is set.

When OS=1, value in the output polarity is inverted and the PWMDT register represents the number of PWM counter clock pulses where the PWM output is reset.

9.4.2 PWM Register Descriptions

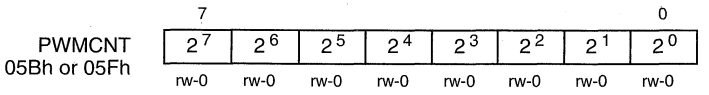
The PWM timer module is controlled using access via the 8-bit MDB structure and MAB. It should be accessed using byte instructions.

Register	short form	Register type	Address	Initial State
PWM timer control register	PWMCTL.1	R/W	58h	reset
PWM duty buffer	PWMDTB.1	R/W	59h	reset
PWM duty register	PWMDTR.1	R/W	5Ah	reset
PWM timer counter	PWMCNT.1	R/(W)*	5Bh	reset
PWM timer control register	PWMCTL.2	R/W	5Ch	reset
PWM duty buffer	PWMDTB.2	R/W	5Dh	reset
PWM duty register	PWMDTR.2	R/W	5Eh	reset
PWM timer counter	PWMCNT.2	R/(W)*	5Fh	reset

Note: Changing the timer counter

The timer counters are read/write registers, but the write function is for test purposes only.
Application programs should never write to these registers.

Timer Counter PWMCNT



The PWM timer counter PWMCNT is an 8-bit up-counter. When the output enable bit OE in the timer control register PWMCTL is set, the timer counter starts counting pulses of an internal clock source selected by clock select bits 2 to 0 (SSEL2 to SSEL0). After counting from 00h to FDh, the timer counter repeats from 00h.

The PWM timer counters can be read and written, but the write function is for test purposes only. Application software should never write to the PW timer counter, because this may have unpredictable effects.

The PWM timer counters are initialized to 00h at a PUC, and when the OE bit is cleared.

Duty Buffer Register PWMDTB

PWMDTB
059h or 05Dh

7							0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

The duty buffer register holds the value for the duty factor. This duty factor is written into the duty register when the timer counter changes from FDh to 00h.

The duty buffer register PWMDTB is initialized to 00h at a reset and in the OSCOff mode.

Note: Changing the PWM duty factor

Changing the duty factor of the PWM should be done only by writing the new value into the PWM duty buffer PWMDTB. Any write access directly to the duty register can result in a random duty cycle during the running period. The next period will run with the new duty factor now contained in the duty buffer.

Duty Register PWMDT

PWMDT
05Ah or 05Eh

7							0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

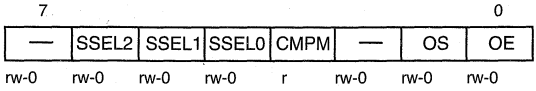
The duty register specifies the duty factor of the output pulse. Any duty factor from 0 to 100% can be selected, with a resolution of 1/254. Writing 0h in the PWMDT gives a 0% duty factor; writing 127 (07Fh) gives a 50% duty factor; writing 254 (0FEh) gives a 100% duty factor.

The timer count is continually compared with the duty register's contents. If the PWMDT value is not 0, the PWM output signal is set when the count increments from 00h to 01h. When the count increments to the PWMDT value, the PWM output returns to 0. If the PWMDT value is 0 (duty factor 0%), the PWM output remains constant at 0.

The PWMDT is double-buffered. A new value written in the PWMDTB while the timer counter is running does not become valid until after the count changes from FDh to 00h. After the PUC of the OE bit is reset the timer counter is stopped and new values become valid as soon as written. When the PWMDT is read, the value read is the currrently valid value.

The duty register PWMDT is initialized to 0FFh at a reset and in the OSCOff mode.

PWM Timer Control Register PWMCTL



The PWM control register is an 8-bit readable/writable register that selects the clock source and controls the PWM outputs.

Bit 0: Output Enable (OE): This bit enables the PWM counter and the PWM output.
OE = 0: PWM output is disabled. PWMTC is cleared to 00h and stopped.
OE = 1 PWM output is enabled. PWMTC runs.

Bit 1: Output Select (OS): This bit selects a true or inverted signal for the PWM output.
OS =0 Positive logic; positive going PWM pulse, 1 = High (Initial value)
OS =1 Negative logic; negative going PWM pulse, 1 = Low

Bits 2 and 7: Reserved: These bits cannot be modified and are always read as 0.

Bit 3: CMPM: This bit is of read-only type. The output of the compare match signal can be deteced. It is read as '1' as long as the timer counter PWMCNT and the duty register PWMDT are identical.

Bits 4 - 6: Clock Select: These bits select one of eight clock sources obtained by dividing the system clock MCLK or the auxilary clock ACLK.

Bit 6 SSEL2	Bit 5 SSEL1	Bit 4 SSEL0	Clock source
0	0	0	MCLK
0	0	1	MCLK/4
0	1	0	MCLK/16
0	1	1	ACLK
1	0	0	ACLK/4
1	0	1	ACLK/8
1	1	0	ACLK/16
1	1	1	ACLK/128

From the clock source frequency, the resolution, period, and frequency of the PWM output can be calculated.

Resolution	=	1/clock source frequency
PWM period	=	resolution x 254 = 254 / clock source frequency
PWM frequency	=	1/PWM period = clock source frequency / 254
Duty cycle	=	PWMDT/254

10 Liquid Crystal Display Drive

10.1 Basics of LCD Drive

Liquid crystal displays use the ambient luminescence to display information and they do not send out light actively. This results in low power consumption. The requirement for visible displayed information is enough ambient luminescence.

The liquid crystal should be driven with alternating voltage. DC drive would destroy the liquid crystal. This AC drive requirement is the main factor for any power consumption. The electrical equivalence for the driving stage is a capacitor. Its electrodes are the back plane or common plane, controlled by signal COMn and the segment driven by SEGn. The frequency of the AC drive is low - in the range of 1000 Hz to 30 Hz. The data sheets of the LCD manufacturer give defined ranges for this frequency.

Different methods of controlling LC displays were developed in the past. The different driving methods are applied as a compromise between number of segments, number of pins at display and driving source, LCD contrast, temperature range, ...
Multiplexing methods reduce the number of pins needed.

The MSP430 Family's LCD module supports four driving methods:
- Static
- 2MUX or 1/2 duty, 1/2 bias
- 3MUX or 1/3 duty, 1/3 bias
- 4MUX or 1/4 duty, 1/3 bias.

The static method needs one pin for common plane (COM0) and one pin for each segment:

#-of-pins = 1 + #-of-segments

The 2MUX method needs two pins for common plane (COM0, COM1) and one pin for two segments:

#-of-pins = Integer [2 + (#-of-segments/2)]

The 3MUX method needs three pins for common plane (COM0, COM1, COM2) and one pin for three segments:

#-of-pins = Integer [3 + (#-of-segments/3)]

The 4MUX method needs four pins for common plane (COM0, COM1, COM2, COM3) and one pin for four segments:

#-of-pins = Integer [4 + (#-of-segments/4)]

The increase of the multiplex rate reduces the number of pins required. The continuous reduction of pin counts is demonstrated by an application that uses 80 segments:

Static method:	#-of-pins = (1 + 80) = 81
2MUX:	#-of-pins = (2 + 80/2) = 42
3MUX:	#-of-pins = (3 + 80/3) = 30
4MUX:	#-of-pins = (4 + 80/4) = 24

Static Driving Method

In the static drive method each segment line drives one segment.
The example shows one digit on the liquid crystal display including connection example, displaying '5' and the output wave forms.

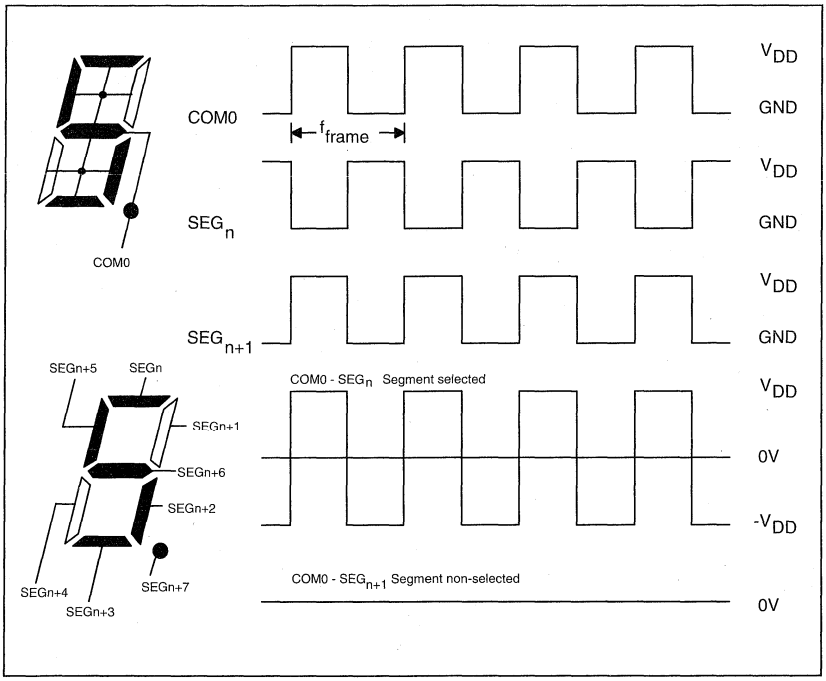


Figure 10.1: Example of static wave form drive

Two MUX, ½ Bias

In the 2MUX drive each segment line drives two segments.
The example shows one digit on the liquid crystal display including connection example, displaying '5' and the output wave forms.

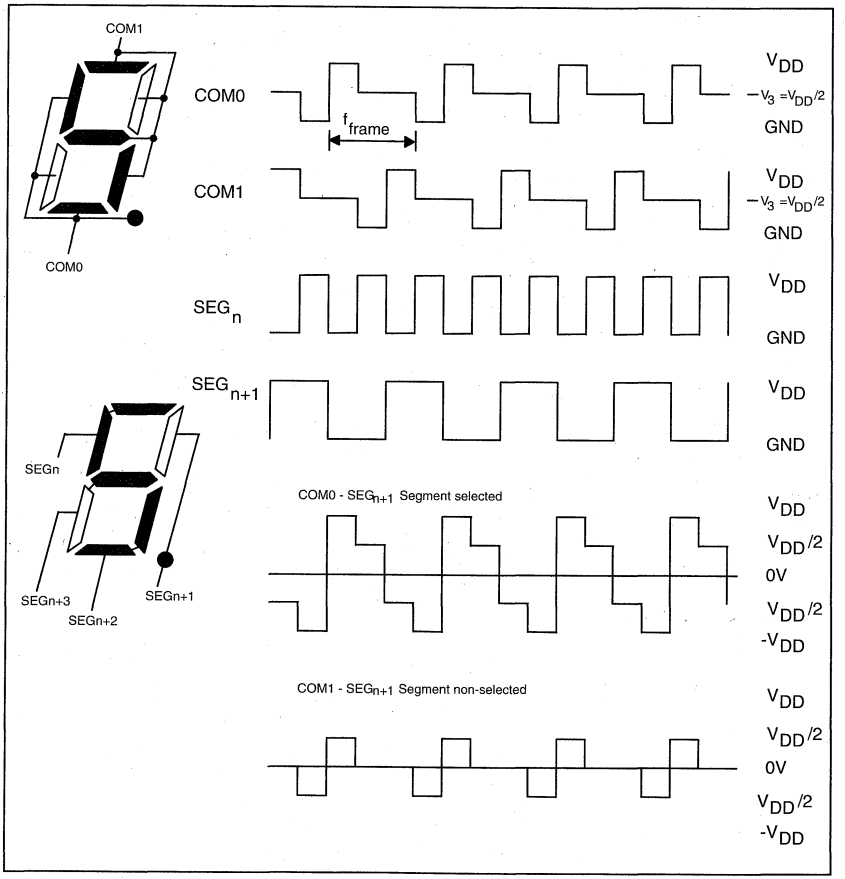


Figure 10.2: Example of 2MUX wave form drive

Three MUX, $^1/_3$ Bias

In the 3MUX drive each segment line drives three segments.
The example shows one digit on the liquid crystal display including connection example, displaying '5' and the output wave forms.

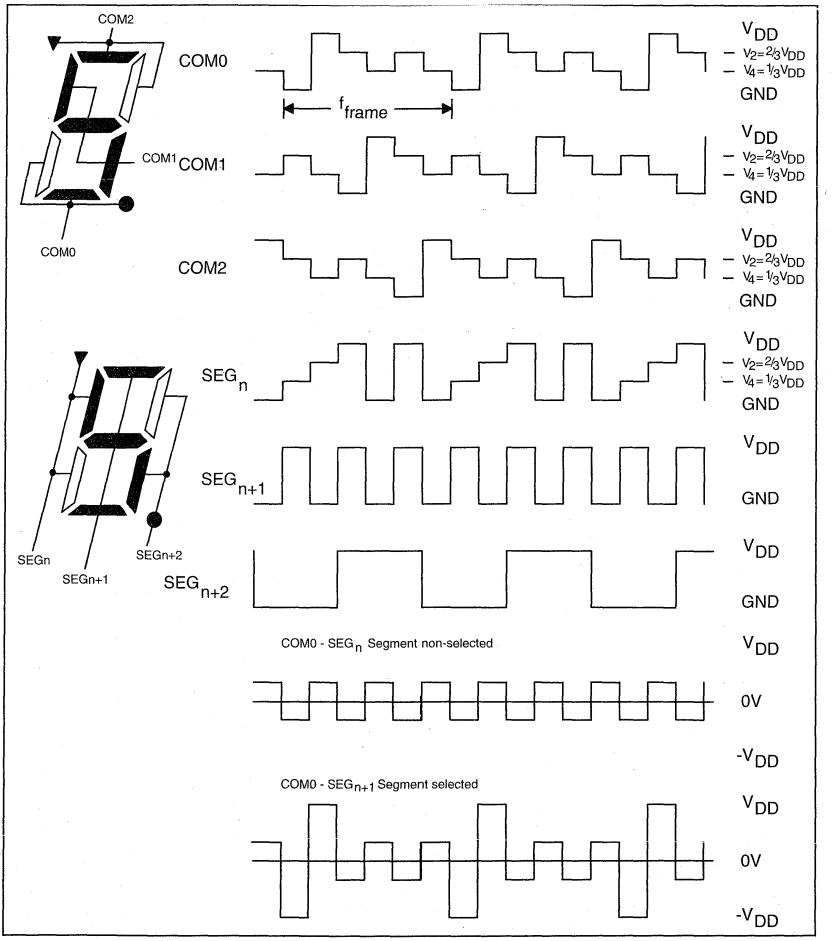


Figure 10.3: Example of 3MUX wave form drive

Four MUX, $^1/_3$ Bias

In the 4MUX drive each segment line drives four segments.
The example shows one digit on the liquid crystal display including connection example, displaying '5' and the output wave forms.

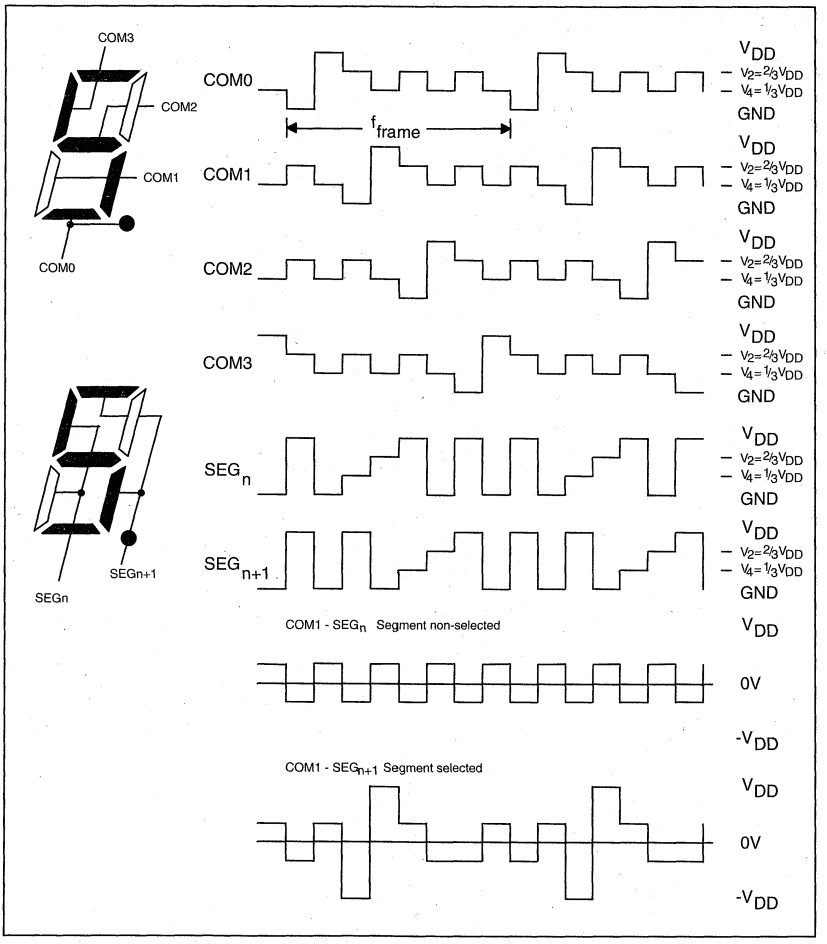


Figure 10.4: Example of 4MUX wave form drive

10.2 LCD Controller/Driver

The LCD controller/driver peripheral generates the segment and common signals according to data in the display data memory. It contains all functional blocks to drive an external directly connected LCD. The main blocks in the LCD peripherals are:

- Data memory containing the segment information
- Timing generator
- Module bus interface
- LCD Module Analog voltage applied externally
- LCD+ Module only: Analog voltage generator internally

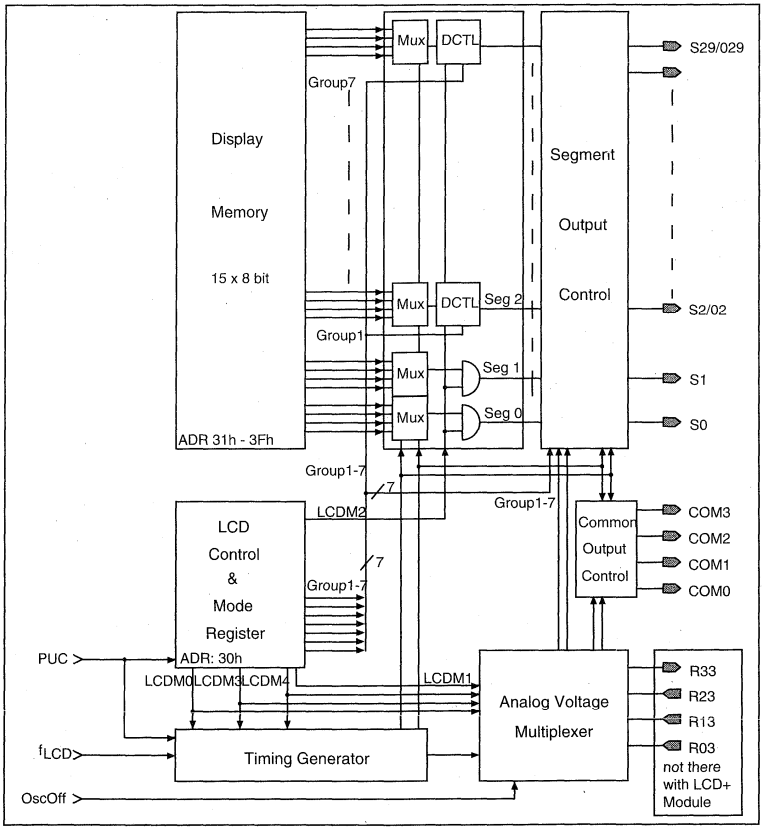


Figure 10.5: LCD Controller/Driver Block Diagram

Differencies of LCD Module and LCD+ Module:

	LCD Module	LCD+ Module
• Analog Voltage Generation	external options: • 2 inputs R23, R13 V1 = VCC V5 = VSS • 3 inputs R23, R13, R03 V1 = VCC • 4 inputs R33, R23, R13, R03	internal
• Control bit LCDM1	unused	selects impedance of R-Ladder
• Control bit LCDM0	• stops timing generator	• stops timing generator • stops current through R-Ladder

10.2.1 LCD Controller/Driver Functions

The functions of the LCD Controller/Driver are:

- Reads automatically data from the display memory and generates the segment and common signals
- Four different display modes are selectable:
  Static mode
  2MUX , 1/2 bias
  3MUX , 1/3 bias
  4MUX , 1/3 bias.
  Within the basic timer BT there are two bits to select one of four different frame frequencies.
- Segment signal outputs can be switched to an output port
- Display memory not used for segment information can be used as a normal memory.
- Operation via the basic timer with the auxiliary clock (ACLK).
- LCD+ Module only:
  Resistive network to supply the analog voltage levels for LCD drive
  One bit in the control register LCDCTL controls the switch through which the resistive network is connected with V1.

The frame frequency of the LCD lines is:

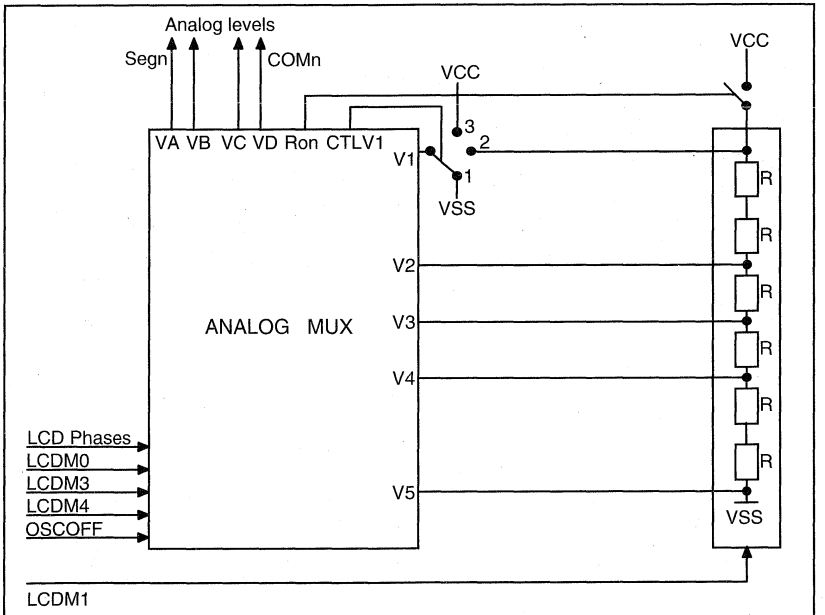
- Static method: $f_{frame} = \frac{1}{2} \times f_{LCD}$
- 2MUX: $f_{frame} = \frac{1}{4} \times f_{LCD}$
- 3MUX: $f_{frame} = \frac{1}{6} \times f_{LCD}$
- 4MUX: $f_{frame} = \frac{1}{8} \times f_{LCD}$

LCD+ Module:

The analog voltage is generated internally.

When the OSCOff bit in the status register is set the power supply to the resistor network is switched off independent of LCDM0 bit.

During static mode the analog generator is switched inactive since the static mode use only V1 and V5 levels. Supply current consumption is reduced.



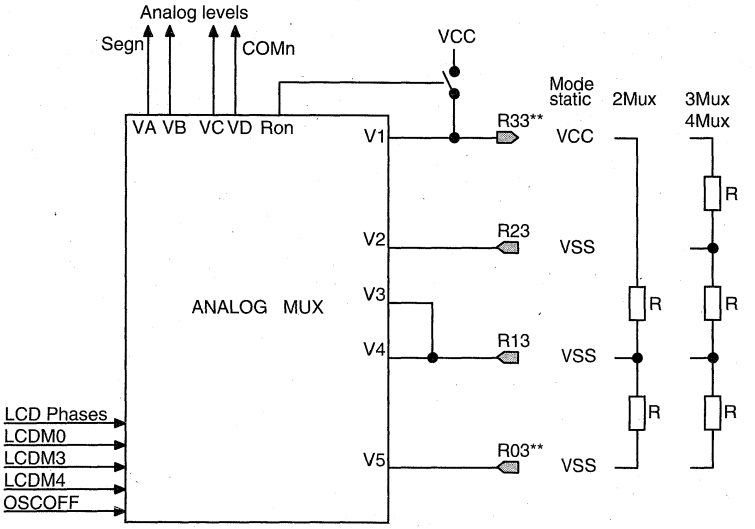
OSCOFF	LCDM4	LCDM3	LCDM0	VA	VB	VC	VD	RON	CTLV1
X	X	X	0	0	0	0	0	OFF	1*
1	X	X	X	0	0	0	0	OFF	1*
0	0	0	1	V5/V1	V1/V5	V5/V1	V1/V5	OFF	3*
0	0	1	1	V5/V1	V1/V5	V3/V3	V1/V5	ON	2*
0	1	X	1	V5/V1	V2/V4	V4/V2	V1/V5	ON	2*

* ≡ to the position of the switch

Figure 10.6: Internal analog voltage generated by LCD+ Module

LCD Module:

The analog voltage is supplied from external, applied on pins R33**, R23, R13, R03**.



OSCOFF	LCDM4	LCDM3	LCDM0	VA	VB	VC	VD	RON
X	X	X	0	0	0	0	0	OFF
1	X	X	X	0	0	0	0	OFF
0	0	0	1	V5/V1	V1/V5	V5/V1	V1/V5	OFF
0	0	1	1	V5/V1	V1/V5	V3/V3	V1/V5	ON
0	1	X	1	V5/V1	V2/V4	V4/V2	V1/V5	ON

* ≡ to the position of the switch

** Supply pins for V1 and V5 are optional. Devices without R33 and R03 pins will have V1 tied to VCC and V5 tied to VSS. External resistor ladder should be connected to VCC and VSS.

Figure 10.7: External analog voltage applied to LCD Module

Note: ** Supply pins R33 and R03 are optional.

10.2.2 LCD Control & Mode Register

The content of the LCD control & mode register defines the different operating conditions. It is connected to the 8-bit memory bus with the standard module interface using the MAB, MDB, MS, MSFR and Read/Write lines. It can be read and written with the entire instruction set. It should be accessed using byte instructions.

LCDCTL
030h

7							0
LCDM7	LCDM6	LCDM5	LCDM4	LCDM3	LCDM2	LCDM1	LCDM0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

LCDM0: LCDM0 = 0: The timing generator is switched off.
Common and segment lines are "L".
Outputs selected for port output lines are not affected.
LCD+ Module: Power supply to resistor network is off.
LCDM0 = 1: Common and segment lines output the signal corresponding to the display memory.
Outputs selected for port output lines are not affected.
LCD+ Module: Power supply to resistor network is switched on at 2MUX, 3MUX and 4MUX not at static mode.

LCDM1: This bit selects the LCD driving strength by selecting the internal resistance of the Analog Generator. It is only valid along with the LCD+ Module.
LCDM1 = 0 : High impedance of Analog Generator.
LCDM1 = 1 : Low impedance of Analog Generator.

LCDM2,3,4: These three bits select the display mode and can switch the segment output to non-selected level.

LCDM4	LCDM3	LCDM2	Display mode	Bias, LCD +	Bias, LCD
X	X	0	Not affected, Display is off - all Segment signals are non-selected level. The port outputs remain stable		
0	0	1	Static mode	1/1	R33*, R03*
0	1	1	2MUX mode	1/2	R33*, R13, R03*
1	0	1	3MUX mode	1/3	R33*, R23, R13, R03*
1	1	1	4Mux mode	1/3	R33*, R23, R13, R03*

* optional pins

The signal LCDM2 disables (0) or enables (1) the segment lines. This is done with an AND combination with each individual segment information. It is located in the parallel serial conversion block between the output of the display memory and the segment output control. The segment information in the display memory remains.
The major purpose of this function is to support applications with flashing displays.

LCDM5,6,7: The information of the three bits selects groups of outputs to carry segment information or bit port information. The outputs selected for port function are driven with the state of display memory bit and are no longer part of the LCD segment lines.

LCDM7	LCDM6	LCDM5	Group0	Group1	Group2	Group3	Group4	Group5	Group6	Group7	
0	0	0	S0-S1	O2-O5	O6-O9	O10-O13	O14-O17	O18-O21	O22-O25	O26-O29	← reset condition
0	0	1	S0-S1	S2-S5	O6-O9	O10-O13	O14-O17	O18-O21	O22-O25	O26-O29	
0	1	0	S0-S1	S2-S5	S6-S9	O10-O13	O14-O17	O18-O21	O22-O25	O26-O29	
0	1	1	S0-S1	S2-S5	S6-S9	S10-S13	O14-O17	O18-O21	O22-O25	O26-O29	
1	0	0	S0-S1	S2-S5	S6-S9	S10-S13	S14-S17	O18-O21	O22-O25	O26-O29	
1	0	1	S0-S1	S2-S5	S6-S9	S10-S13	S14-S17	S18-S21	O22-O25	O26-O29	
1	1	0	S0-S1	S2-S5	S6-S9	S10-S13	S14-S17	S18-S21	S22-S25	O26-O29	
1	1	1	S0-S1	S2-S5	S6-S9	S10-S13	S14-S17	S18-S21	S22-S25	S26-S29	

Note: LCD control bits

The control bits LCDM5 ... LCDM7 are reset with PUC.

Function Seg: The Sxx signals are part of the display driving signals and carry modulated voltage levels according to the time frame of the common lines.

Function Port: The signals selected for 'port' function are static signals. They take two digital levels according to bits in the display memory. The logical state of the bits is taken from bit 0 to bit 3 for even S-lines (n=3,5,.....) and from bit 4 to bit 7 for odd S-lines (n=2,4,.....).

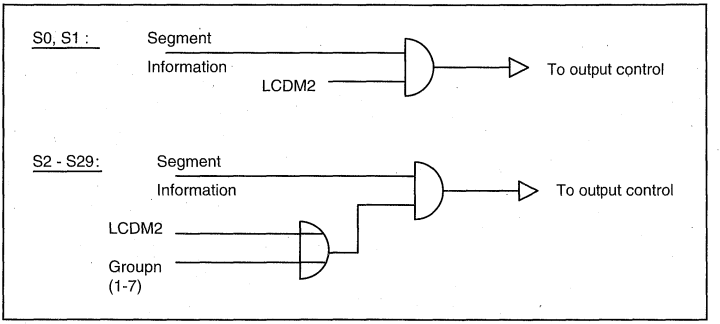


Figure 10.8: Information control

10.2.3 LC Display Memory

The LC Display Memory holds the information to be displayed during all operating and power down modes. The bits in the memory are directly attached to the segments of the liquid crystal display. The figures displayed at the LCD are decoded by the software from the BCD or binary representation to the segment/common combination of the individual display. The bit information in the memory matches with one common line and one segment line. The bit information in the memory corresponds to the selection of segments - a bit set in the memory is identical with segment selection 'on' and reverse.

One segment line carries the on/off state of one to four segments depending on the multiplex rate:

Static drive -> state of one segment/segment line
2MUX drive -> state of two segment/segment line
3MUX drive -> state of three segment/segment line
4MUX drive -> state of four segment/segment line

The timing generator of the LCD controller/driver drives the conversion of the parallel information stored in the LC Display Memory into the serial information required for the segment line signal. The bits of the LC Display Memory are hard wired to the common lines:

Static drive -> COM0: Bit 0 to Sn, Bit 4 to Sn+1
2MUX drive -> COM0: Bit 0 to Sn, Bit 4 to Sn+1, COM1: Bit 1 to Sn, Bit 5 to Sn+1
3MUX drive -> COM0: Bit 0 to Sn, Bit 4 to Sn+1, COM1: Bit 1 to Sn, Bit 5 to Sn+1
COM2: Bit 2 to Sn, Bit 6 to Sn+1
4MUX drive -> COM0: Bit 0 to Sn, Bit 4 to Sn+1, COM1: Bit 1 to Sn, Bit 5 to Sn+1
COM2: Bit 2 to Sn, Bit 6 to Sn+1, COM3: Bit 3 to Sn, Bit 7 to Sn+1

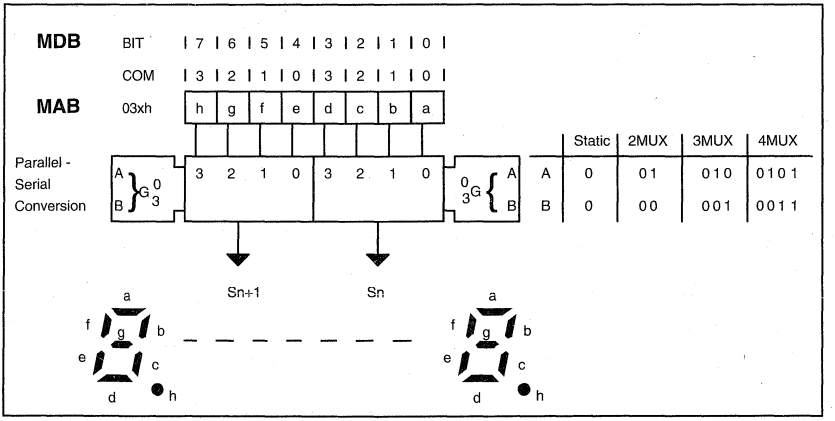


Figure 10.9: Bits of Display Memory attached to Segment lines

Display memory using the static driving method

The static driving method uses one common line. The active common line is COM0. In this mode BIT0 and BIT4 are used for segment information. The other bits can be used like any other memory.

The maximum number of segments is 30.

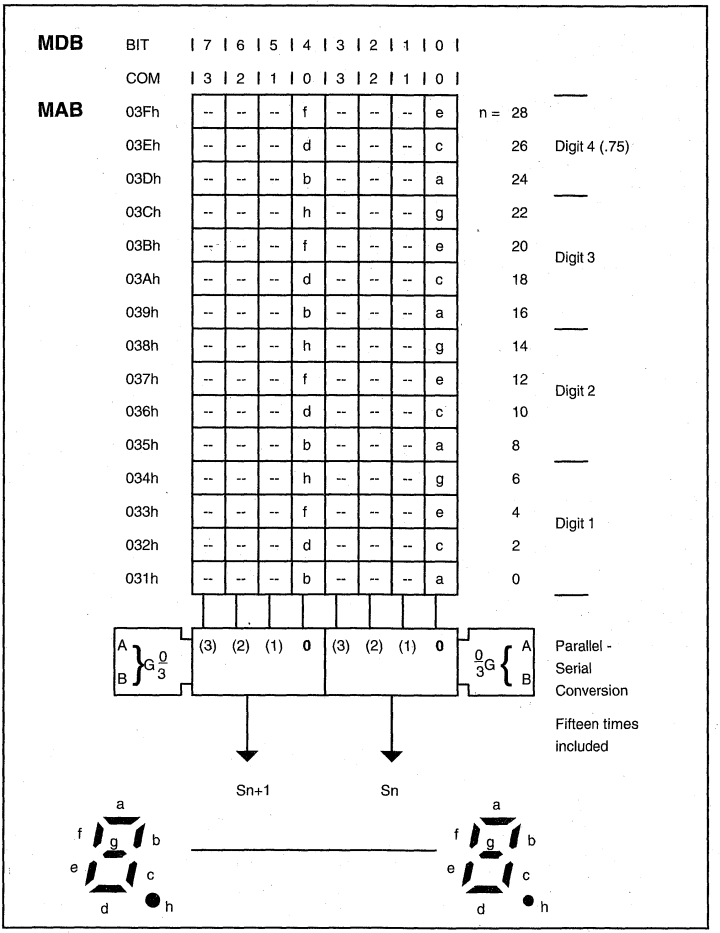


Figure 10.10: Use of Display Memory with the static driving method

Display memory using 2MUX, 1/2 bias driving method

The 2MUX driving method uses two common lines. The active common lines are COM0 and COM1. In this mode the BIT0, BIT1, BIT4 and BIT5 are used for segment information. The other bits can be used like any other memory.

The maximum number of segments is 60.

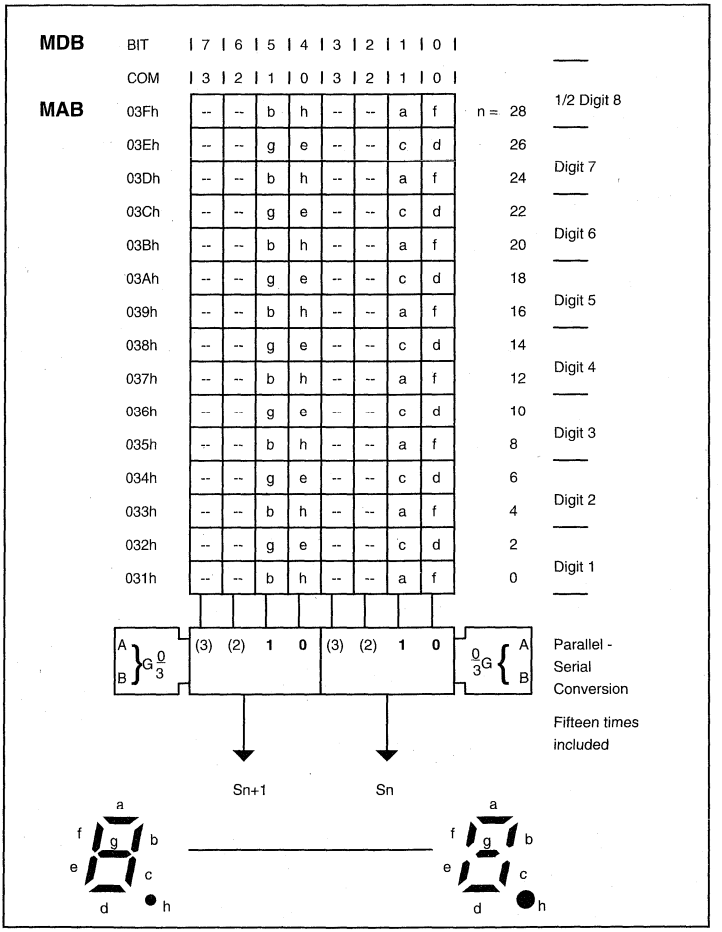


Figure 10.11: Use of Display Memory with the 2MUX method

Display memory using 3MUX, 1/3 bias driving method

The 3MUX driving method uses three common lines. The active common lines are COM0, COM1 and COM2. In this mode BIT0, BIT1, BIT2, BIT4, BIT5 and BIT6 are used for segment information. The other bits can be used like any other memory.

The maximum number of segments is 90.

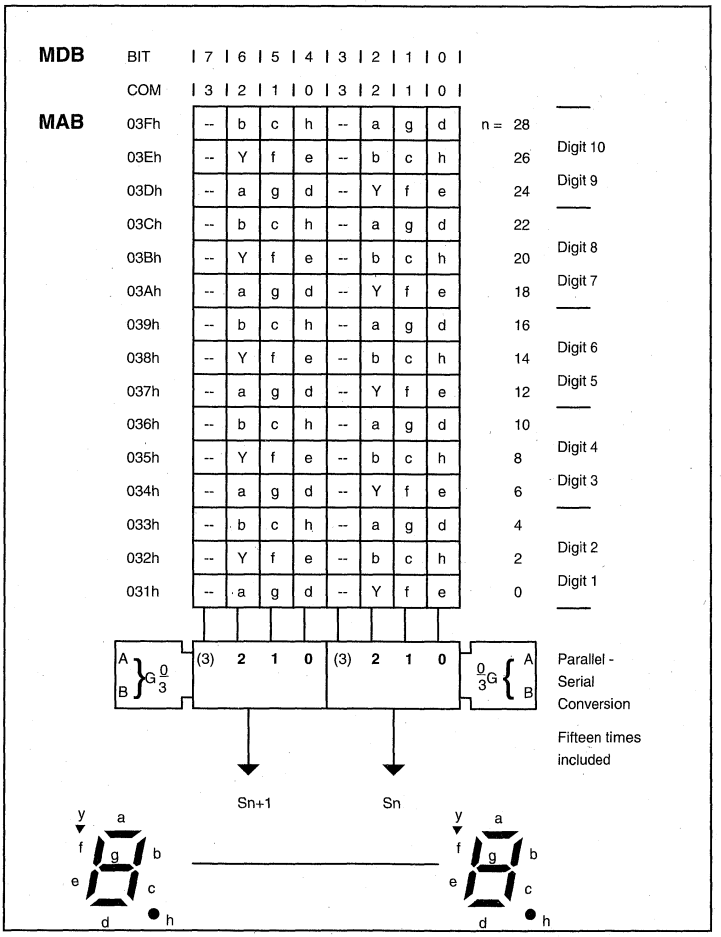


Figure 10.12: Use of Display Memory with the 3MUX method

Display memory using 4MUX, 1/3 bias driving method

The 4MUX driving method uses four common lines. The active common lines are COM0, COM1, COM2 and COM3. In this mode BIT0, BIT1, BIT2, BIT3, BIT4, BIT5, BIT6 and BIT7 are used for segment information.

The maximum number of segments is 120.

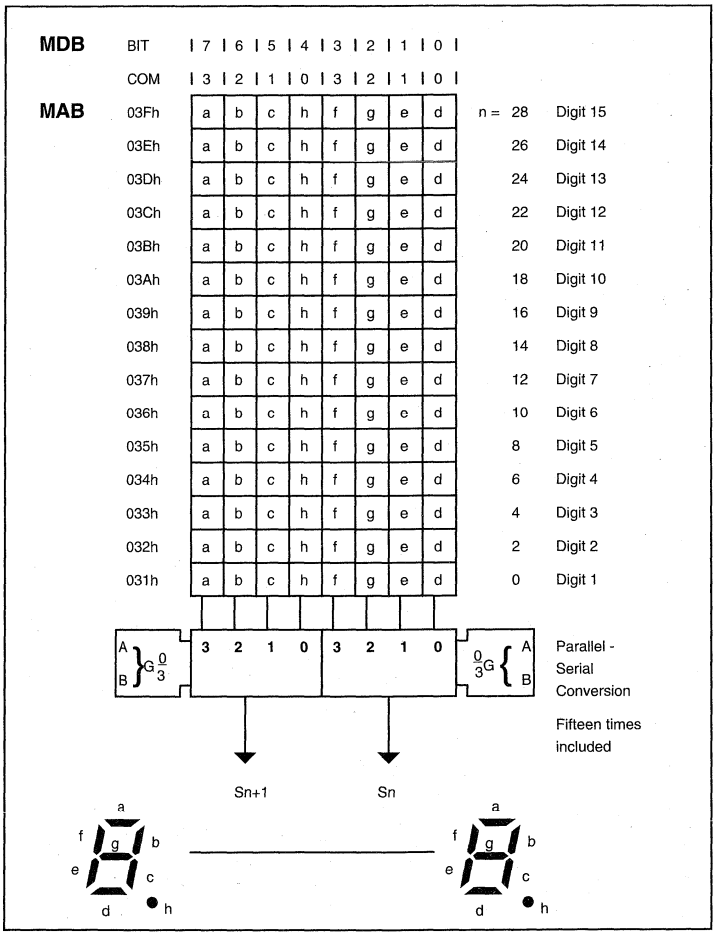


Figure 10.13: Use of Display Memory with the 4MUX method

10.2.4 Software Examples for LCD Operation

The examples in this paragraph demonstrate the software to display digits on the LCD. They used the standard nomenclature of 7-segment digit.

Software for 4MUX, 1/3 bias LCD

```
                .sect "lcd4mux",0f000h
;   The 4MUX rate is the most easy to handle display rate. All eight segments of a digit
;   are located in one display memory byte
;
a               .EQU    080h
b               .EQU    040h
c               .EQU    020h
d               .EQU    001h
e               .EQU    002h
f               .EQU    008h
g               .EQU    004h
h               .EQU    010h
;
;   The  LSDigit of register Rx (000m) should be displayed.
;   The Table represents the 'on'-segments according to the content of Rx.
;
                ..........
LCD1            .EQU    00031h          ; Address of LC Display Memory
                ..........
                ..........
LCD15           .EQU    0003Fh
                ..........

;               ..........
                ..........
                MOV.B   Table(Rx),&LCDn ; n = 1 ..... 15
                                        ; all eight segments are written to the
                                        ; display memory
                ..........
                ..........
;
Table           .BYTE   a+b+c+d+e+f     ; displays "0"
                .BYTE   b+c             ; displays "1"
                ..........
                ..........
                .BYTE   b+c+d+e+g       ; displays "d"
                .BYTE   a+d+e+f+g       ; displays "E"
                .BYTE   a+e+f+g         ; displays "F"
```

Software for 3MUX, 1/3 bias LCD

```
        .sect "lcd3mux",0f000h
;       The 3MUX rate supports nine segments instead of eight segments for each digit.
;       The nine segments of a digit are located in 1 ½ display memory bytes.
;
a               .EQU    0040h
b               .EQU    0400h
c               .EQU    0200h
d               .EQU    0010h
e               .EQU    0001h
f               .EQU    0002h
g               .EQU    0020h
h               .EQU    0100h
Y               .EQU    0004h
;       The  LSDigit of register Rx (000m) should be displayed.
;       The Table represents the 'on'-segments according to the LSDigit of register of Rx.
;       The register Ry is used for temporary memory
;
LCD1            .EQU    00031h
                .....
LCD15           .EQU    0003Fh
                .....
                .....
ODDDIG  RLA     Rx
                MOV     Table(Rx),Ry    ; Load segment information to
                                        ; temporary mem.
                                        ; (Ry) = 0000 0bch 0agd 0Yfe
                MOV.B   Ry,&LCDn        ; write 'a, b, c, d, e, f' of Digit n
                                        ; (LowByte)
                SWPB    Ry              ; (Ry) = 0agd 0Yfe 0000 0bch
                BIC.B   #07h,&LCDn+1    ; write 'b, c, h' of Digit n (HighByte)
                BIS.B   Ry,&LCDn+1
                .....
EVNDIG  RLA     Rx
                MOV     Table(Rx),Ry    ; Load segment information to
                                        ; temporary mem.
                                        ; (Ry) = 0000 0bch 0agd 0Yfe
                RLA     Ry              ; (Ry) = 0000 bch0 agd0 Yfe0
                RLA     Ry              ; (Ry) = 000b ch0a gd0Y fe00
                RLA     Ry              ; (Ry) = 00bc h0ag d0Yf e000
                RLA     Ry              ; (Ry) = 0bch 0agd 0Yfe 0000
                BIC.B   #070h,&LCDn+1
                BIS.B   Ry,&LCDn+1      ; write 'Y, f, e' of Digit n+1 (LowByte)
                SWPB    Ry              ; (Ry) = 0Yfe 0000 0bch 0agd
                MOV.B   Ry,&LCDn+2      ; write 'b, c, h, a, g, d' of Digit n+1
                                        ; (HighByte)
                .....
```

```
Table   .WORD   a+b+c+d+e+f    ; displays "0"
        .WORD   b+c            ; displays "1"
        .....
        .....
        .WORD   a+e+f+g        ; displays "F"
```

Software for 2MUX, 1/2 bias LCD

```
            .sect "lcd2mux",0f000h
;           All eight segments of a digit are located in two display memory bytes with the
;           2MUX display rate
;
a                   .EQU    002h
b                   .EQU    020h
c                   .EQU    008h
d                   .EQU    004h
e                   .EQU    040h
f                   .EQU    001h
g                   .EQU    080h
h                   .EQU    010h
;           The register content of Rx (000m) should be displayed.
;           The Table represents the 'on'-segments according to the content of Rx.
;
                    .....
;
LCD1                .EQU    00031h
                    .....
                    .....
LCD15               .EQU    0003Fh
;
                    .....
                    .....
                    MOV.B   Table(Rx),Ry    ; Load segment information to
                                            ; temporary mem.
                    MOV.B   Ry,&LCDn        ; (Ry) = 0000 0000 gebh cdaf
                                            ; Note:
                                            ; All bits of an LCD memory byte are
                                            ; written
                    RRA     Ry              ; (Ry) = 0000 0000 0geb hcda
                    RRA     Ry              ; (Ry) = 0000 0000 00ge bhcd
                    MOV.B   Ry,&LCDn+1      ; Note:
                                            ; All bits of an LCD memory byte are
                                            ; written
                    .....
                    .....
;
Table               .BYTE   a+b+c+d+e+f     ; displays "0"
                    .....
                    .BYTE   a+b+c+d+e+f+g+h ; displays "8"
                    .....
                    .....
                    .BYTE
                    .....
;
```

Software for static LCD

```
            .sect "lcd1mux",0f000h
;       All eight segments of a digit are located in four display memory bytes with the
;       static display method.
;
a           .EQU    001h
b           .EQU    010h
c           .EQU    002h
d           .EQU    020h
e           .EQU    004h
f           .EQU    040h
g           .EQU    008h
h           .EQU    080h
;       The register content of Rx should be displayed.
:       The Table represents the 'on'-segments according to the content of Rx.
;
            .....
;
LCD1        .EQU    00031h
            .....
            .....
LCD15       .EQU    0003Fh
;
            .....
            .....
            MOV.B   Table(Rx),Ry    ; Load segment information to temporary
                                    ; mem.
                                    ; (Ry) = 0000 0000 hfdb geca
            MOV.B   Ry,&LCDn        ; Note:
                                    ; All bits of an LCD memory byte are written
            RRA     Ry              ; (Ry) = 0000 0000 0hfd bgec
            MOV.B   Ry,&LCDn+1      ; Note:
                                    ; All bits of an LCD memory byte are written
            RRA     Ry              ; (Ry) = 0000 0000 00hf dbge
            MOV.B   Ry,&LCDn+2      ; Note:
                                    ; All bits of an LCD memory byte are written
            RRA     Ry              ; (Ry) = 0000 0000 000h fdbg
            MOV.B   Ry,&LCDn+3      ; Note:
                                    ; All bits of an LCD memory byte are written
            .....
            .....
;
```

```
Table   .BYTE   a+b+c+d+e+f   ; displays "0"
        .BYTE   b+c           ; displays "1"
        .....
        .....
        .BYTE
        .....
```

10.3 LCD Port Function

The large number of LCD common and segment lines together with the fixed number of pins of the package version could limit the integration. To support applications which require a reduced number of segments, the signals LCDM5 to LCDM7 can switch the function from segment lines to output lines in groups of four bits. These outputs can be used with the application for various functions. Bits in the display memory define the logical state of the signals. The output signals are digital switched either near to ground GND or near to supply voltage VCC.

The naming convention for signals used as segment lines is Sxx and as port functions is Oxx. A pin is identified equally with the same xx representation. The letter S or O states the function of that pin.

LCDM7	LCDM6	LCDM5	Group0	Group1	Group2	Group3	Group4	Group5	Group6	Group7	
0	0	0	S0-S1	O2-O5	O6-O9	O10-O13	O14-O17	O18-O21	O22-O25	O26-O29	← Reset Condition
0	0	1	S0-S1	S2-S5	O6-O9	O10-O13	O14-O17	O18-O21	O22-O25	O26-O29	
0	1	0	S0-S1	S2-S5	S6-S9	O10-O13	O14-O17	O18-O21	O22-O25	O26-O29	
0	1	1	S0-S1	S2-S5	S6-S9	S10-S13	O14-O17	O18-O21	O22-O25	O26-O29	
1	0	0	S0-S1	S2-S5	S6-S9	S10-S13	S14-S17	O18-O21	O22-O25	O26-O29	
1	0	1	S0-S1	S2-S5	S6-S9	S10-S13	S14-S17	S18-S21	O22-O25	O26-O29	
1	1	0	S0-S1	S2-S5	S6-S9	S10-S13	S14-S17	S18-S21	S22-S25	O26-O29	
1	1	1	S0-S1	S2-S5	S6-S9	S10-S13	S14-S17	S18-S21	S22-S25	S26-S29	

Figure 10.14: Groups of Segment and Output Lines

> **Note: Control bits**
>
> The control bits LCDM5 ... LCDM7 are reset with PUC.

The segment signals Sxx are part of the display driving signals and carry modulated voltage levels according to the time frame of the common lines.

The output signals Oxx selected for 'port' function are static signals. They take two digital level according to bits in the display memory. The logical state of the bits is taken from bit 0 to bit 3* for odd S-lines (n=3,5,.....) and from bit 4 to bit 7* for even S-lines (n=2,4,.....).

* Taken bits are dependent on the MUX rate.

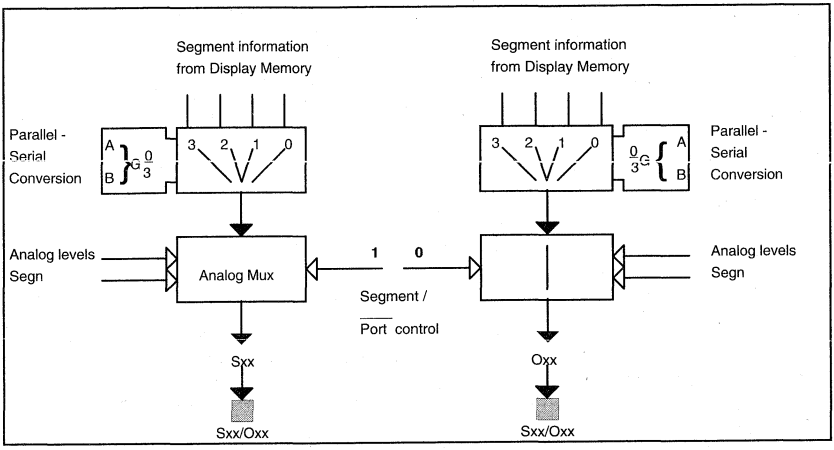


Figure 10.15: Segment Line or Output Line

The logical information of an output Oxx is defined in the display memory. Its location is either bit0 to bit3 or bit4 to bit7 pending on odd or even assignment:

- xx = 2,4, ... 28: Oxx is defined with bit0 to bit3
- xx = 3,5, ... 29: Oxx is defined with bit4 to bit7

10.4 Application Example showing mixed LCD and Port Mode

The example uses the mixed mode: 4MUX LCD drive for 13 digits and one port group with four digital outputs.

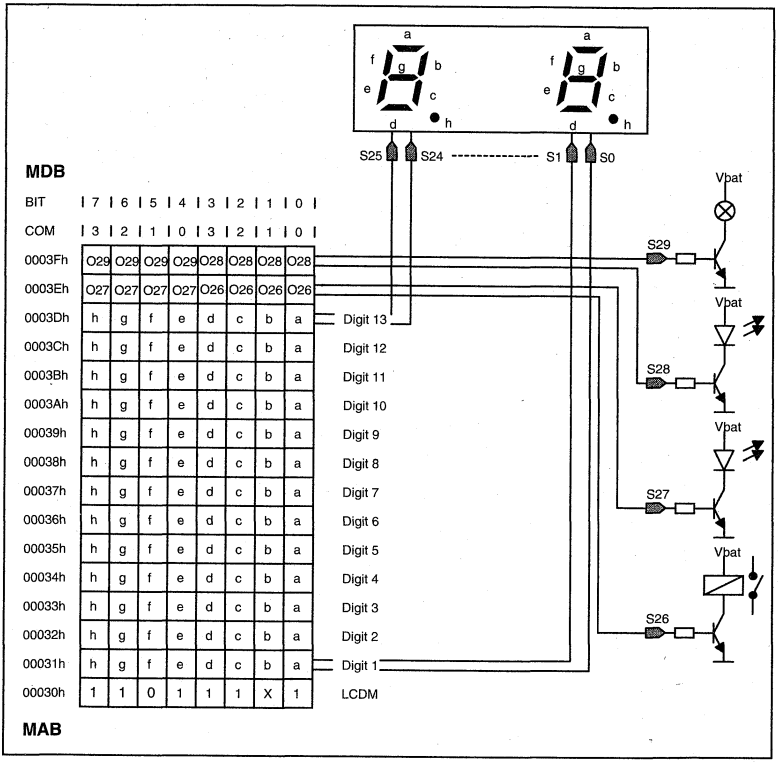


Figure 10.16: Application Example

> **Note: LCD port output**
>
> Any LCD port output is defined through four bits. All four bits of the group should have the same logical level otherwise the output is not statical. Assume O28 should be 'H', all the bit0 to bit3 are 'H'.

Software example to set O28, O29 should be unchanged

```
LCD15       .EQU    0003Fh
            BIS.B   00Fh,&LCD15
```

11 Analog-To-Digital Converter

Features of the A/D module:

- Eight Analog or Digital input channels
- Programmable (via external resistor REXT) constant current source on four analog pins
- Ratiometric or Absolute measurement
- Built in Sample and Hold
- End-Of-Conversion (EOC) interrupt flag
- ADAT register holds conversion results until next Start-Of-Conversion (SOC)
- Low power consumption
- Truly standalone without further CPU processing overhead
- Programmable 12-bit or 14-bit resolution
- Four programmable ranges give 14-bit dynamic range
- Fast conversion time
- Large supply voltage range
- Monotonic over the entire A/D conversion range

11.1 Overview

The (12+2)-bit Analog-to-Digital converter is a peripheral module, connected to the 16-bit memory data bus MDB. The result of the converter is available on this 16-bit wide bus by reading the ADAT register. It must be noted that when a conversion is started, the bits as they are resolved by the converter are latched by the successive approximation register (SAR). They are available immediately at the ADAT register and are not cleared until the next conversion is initiated by setting the Start-Of-Conversion (SOC) bit in the ACTL register. Since the SAR is transparent to the MDB the conversion progress can be monitored by reading data via the read only ADAT register. The SOC bit clears the SAR register for the new result as well as starting the clock of the A/D converter for another conversion.

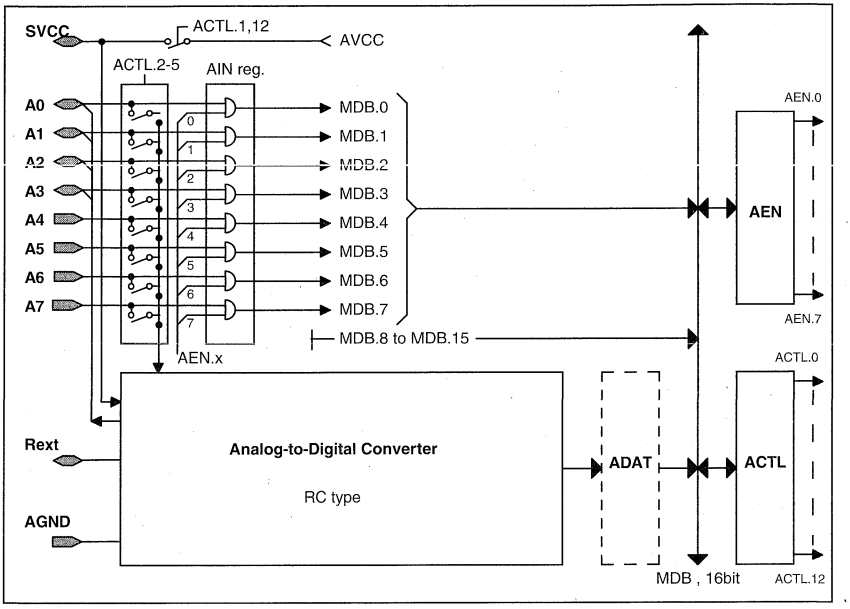


Figure 11.1: ADC Module Configuration

The module has eight individually selectable analog input channels that are multiplexed to the converter's input circuitry such that a conversion can be made on any one of these channels at any time. Four of these channels, A0, A1, A2 and A3 are also configured as four constant current source outputs whose values can be programmed by an external resistor R_{ext}. Any of these current source outputs can be turned on (one at a time) to drive external sensors in order to make ratiometric measurements. An absolute measurement can also be made to the accuracy of the reference voltage, by applying an external stable voltage source to the SVCC pin.

The eight channels can also be configured as analog or digital inputs. The digital data can be presented at all eight channels or individually selected channels by writing to the respective bits of the AEN register. The digital data presented at the channels can be read from the AIN register. When a sensitive analog conversion is being done, any digital activity on adjacent channels will cause cross-talk and interference, leading to noise and incorrect output codes.

The converter has two modes of operation depending on the status of bit11 of the ACTL register, either a 12-bit or a (12+2)-bit conversion is possible. When the range of the input signal is known two bits from the ACTL register can be used to define the range required with bit11 reset. The converter will sample the input and then convert it to 12 bits of resolution, within any one of these four ranges. In this manual mode, this yields,

effectively a 14 bit dynamic range of operation for the converter. However in the Auto Mode, selected by setting bit11 of the ACTL register, the range is automatically selected by the converter to resolve to effectively 14 bits. The input is sampled twice, once for the 2-bit range selection and lastly for the remaining 12-bits of the conversion to give a (12+2)-bit result. In both modes, when a conversion is completed (End-Of-Conversion EOC), the interrupt flag is set automatically to signal the microprocessor that a conversion has been completed. The EOC signal also disables the clock for the A/D converter to conserve power until the next SOC bit is set.

Note: ADC, Start-of-Conversion

After starting a conversion it should always be completed before the next conversion is initiated. Otherwise unpredictable conversion data will result.

The microprocessor core communicates to the A/D via the internal bus system by applying the correct address for the module and supplying the required conditions for the ACTL and AEN registers. It reads the conversion results back via the ADAT registers.

Under power-down the whole Analog-to-Digital converter shuts down to stop current consumption. This is valid while SVCC is not externally driven. Upon a conversion start or power-up signal, the converter wakes up, but may take up to 6 µs to reach steady state conditions for an accurate conversion.

11.2 Analog-to-Digital Operation

11.2.1 A/D Conversion

After power-up, the ACTL register should be programmed to decide whether to make a ratiometeric or absolute measurement, and whether the range is to be manually or automatically selected. In manual mode, once the range bits have been selected these bits can not be changed during the conversion, as this will invalidate the results.

Setting the Start-of-Conversion (SOC) bit in the ACTL register activates the clock for the A/D converter for a new conversion to begin. The converter is based on a successive approximation technique utilising a resistor array to resolve the M most significant bits (MSBs) first and a switched capacitor array to resolve the remaining L least significant bits (LSBs).

The resistor array consisting of 2^M individual, equally weighted resistors forms the DAC, and the capacitor array consisting of L capacitors forms a charge redistribution A/D. The capacitors are binary-weighted; that is they increase from the smallest value in powers of two. The number of capacitors corresponds to the range of the converter or L bits of the digital output code.

The sequence starts by selecting the analog channel of interest and sampling the analog input voltage onto the top plates of the capacitor array, the analog mux is then disconnected from the A/D and the analog input need not be present anymore after this sample period.

A successive approximation search is done on the resistor string to find the tap that corresponds to being within 2^L LSBs of VIN, this then gives the VH and VL voltages across one element and has resolved the M MSBs. The capacitor array then resolves this (VH-VL) difference voltage to L bits of resolution using a similar successive approximation search on the capacitor array starting with the MSB capacitor.

This switching procedure proceeds with the MSB or largest capacitor to the smallest (LSB) capacitor in the capacitor array, thereby the initial charge is redistributed among the capacitors. The particular setting of the switches both in the resistor array and those connected to the bottom plates of the capacitors has then induced a change on the top plate that is as close to the input voltage VIN as possible, and the switch settings then correspond to the binary code (12-bit or (12+2)-bit) that represents the fraction VIN/VREF.

The top plate voltage is monitored by a comparator with built-in input offset cancellation circuitry which senses whether the input voltage is less than or greater than the voltage on the top plate and generates a digital output which determines which way the successive approximation search is to be performed.

The smallest voltage change (LSB) occurs when the smallest capacitor is switched in and this is the resolution of the converter, or VREF/2^n where n is the number of bits.

When this sequence is completed, the top plate voltage is as close to zero as the resolution of the converter allows and the LSB has been determined. An End-of-Conversion (EOC) signal is then sent to indicate that a 12-bit or (12+2)-bit conversion result is available for reading from the ADAT register for further processing.

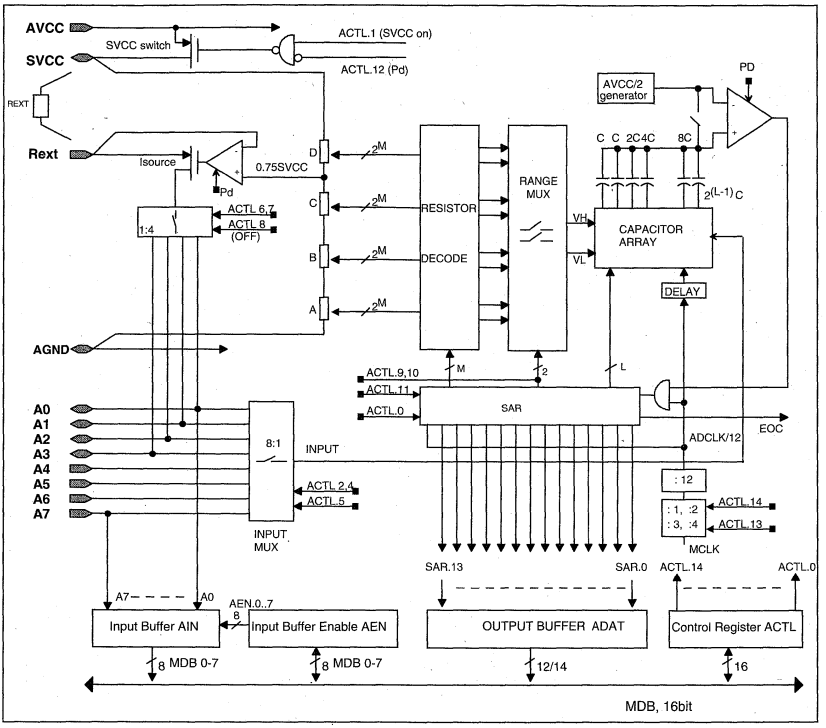


Figure 11.2: ADC Schematic

A/D conversion timing

After the ADC module has been activated with the Power Down bit reset, at least 6 µs must elapse before a new conversion is attempted in order to allow the correct internal biases to be established.

The A/D converter always runs at a clock rate set to one twelfth of the ADCLK. The frequency of ADCLK should be chosen to meet the conversion time defined in the actual electrical charactaristics. If the ADCLK is too fast an accurate conversion to 12 bits cannot guaranteed, due to internal time constants associated with sampling the analog input and the conversion network. If the ADCLK is too slow a conversion accurate to 12 bits cannot be guaranteed, due to charge loss within the capacitor array of the A/D even if the input signal is valid and steady for the required acquisition time.

Sampling the analog input signal takes twelve ADCLK clock pulses and the 12-bit conversion takes another twelve times seven (84) ADCLK clock cycles. This is true for a 12-bit conversion with pre-selected range, ACTL.11 is reset. All together the 12-bit conversion takes 96 ADCLK cycles.

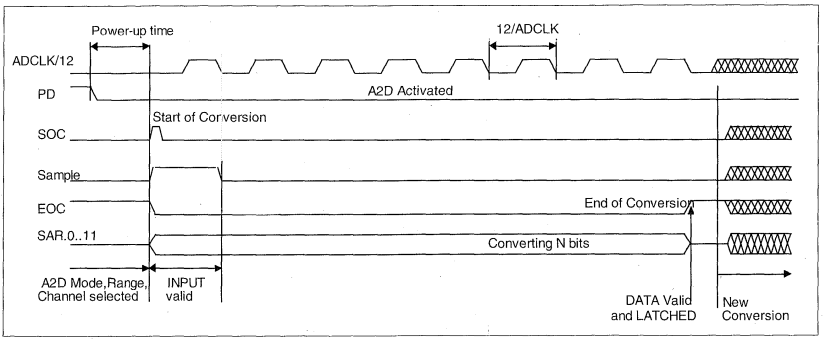


Figure 11.3: ADC Timing, 12-bit conversion

When ACTL.11 is set a (12+2)-bit conversion with auto range selected takes place. The analog input signal is sampled twice, each taking twelve ADCLK clock pulses. After the first sampling of the input signal the range conversion is done and takes 24 ADCLK clocks. After the second sampling of the input signal the 12-bit conversion is done and takes another 84 (12*7) ADCLK clock cycles. All together the (12+2)-bit conversion takes 132 ADCLK cycles.

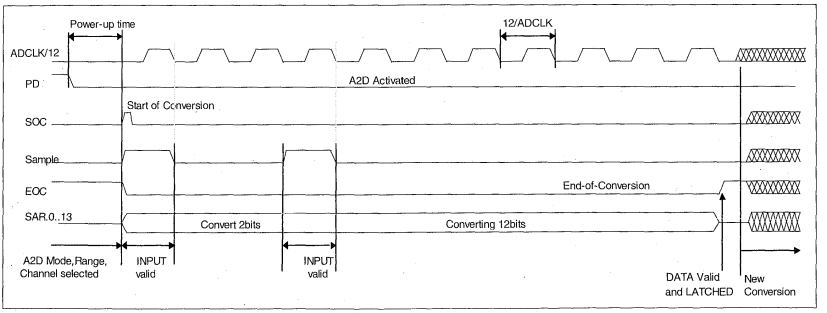


Figure 11.4: ADC Timing, (12+2)-bit conversion

The input signal must be valid and steady during this sampling period in order to obtain an accurate conversion. It is also desirable not to have any digital activity on any adjacent digital channels during the whole of the conversion period to ensure errors due to supply glitching and ground bounce or cross-talk interference do not corrupt the results.

The A/D converter uses the charge redistribution method and thus when the inputs are internally switched to sample the input, the switching action causes displacement currents to flow into and out of the analog inputs.

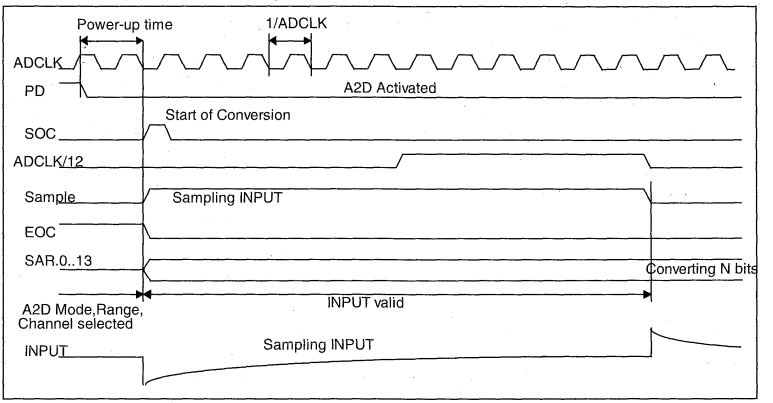


Figure 11.5: ADC, input sampling timing

These current spikes or transients occur at the leading and falling edge of the internal sample pulse, and quickly decay and settle before causing any problems because the time constant is less than that given by the internal 'effective RC'. Internally the analog inputs see a nominal RC of effectively a 32 pF (C-array) capacitor in series with a 2-kΩ resistor (Ron of switches). However if the external dynamic source impedance is large, then these transients may not settle within the allocated sampling time to ensure 12 or (12+2) bits of accuracy.

11.2.2 A/D Interrupt

When an A/D conversion is complete, the EOC signal goes high and activates the A/D interrupt circuit by setting the interrupt flag ADIFG, which informs the rest of the system when a conversion has completed. An interrupt is requested when the enable bit ADIE is set.

11.2.3 A/D Ranges

One of four ranges can be selected to yield a result with 12 bits of resolution within any one given range. If the bit ACTL.11 is reset, effectively 14 bits of dynamic range are possible. The range is defined prior to conversion start with the bits ACTL.9 and ACTL.10. However if bit11 is set then the converter will find the appropriate range during the conversion by sampling the input twice, once for the range selection and secondly for the 12-bit conversion, thereby giving overall a (12+2)-bit conversion result.

The ranges are:

$$0.00 \times VREF \leq VIN < 0.25 \times VREF \quad \text{Range A}$$
$$0.25 \times VREF \leq VIN < 0.50 \times VREF \quad \text{Range B}$$
$$0.50 \times VREF \leq VIN < 0.75 \times VREF \quad \text{Range C}$$
$$0.75 \times VREF \leq VIN < 1.00 \times VREF \quad \text{Range D}$$

Where VREF is the voltage at the SVCC pin, either applied externally or that voltage (close to AVCC) derived by closing the SVCC switch with bit12 of the ACTL register.

After the proper range has been selected, the input channel, selected by the appropriate bits in the control register, is connected to the input of the converter. The A/D converter processes the signal at the selected input channel and the software can then access the result of the conversion via the ADAT register.

The digital code (Decimal) expected within any one range is:

$$N_{typ} = INT \left| \frac{VIN \times 2^{14}}{VREF} - 2^{13} \times ACTL.10 - 2^{12} \times ACTL.9 \right|$$

where ACTL.10 and ACTL.9 are bits 10 and 9 respectively in the ACTL register.

Thus for a 12-bit conversion:

$$0000h \leq N \leq 0FFFh \quad \text{Range A}$$
$$0000h \leq N \leq 0FFFh \quad \text{Range B}$$
$$0000h \leq N \leq 0FFFh \quad \text{Range C}$$
$$0000h \leq N \leq 0FFFh \quad \text{Range D}$$

and a (12+2)-bit conversion:

$$0000h \leq N \leq 3FFFh$$

Note: ADC Offset voltage

Any offset voltage (Vio) due to voltage drops at the bottom or top of the resistor array, caused by parasitic impedances to the SVCC pin or the ground AGND pin, will distort the digital code output and the formula.

11.2.4 A/D Current Source

One of four analog I/Os can be used for the current source output. The current out of the current source (Isource) can be programmed by an external resistor REXT and is then available on pins A0, A1, A2 and A3, with the value:

$I_{source} = (0.25 \times SVCC)/R_{ext}$ where SVCC is the voltage at pin SVCC and R_{ext} is the external resistor between pins SVCC and R_{ext}.

Therefore for ratiometric measurements the voltage (Vin) developed at the input to the channel with the resistive elements (Channels A0, A1, A2 and A3 only) is:

$Vin = (0.25 \times SVCC) \times (R_{sens}/R_{ext})$ where R_{sens} is the external resistive element.

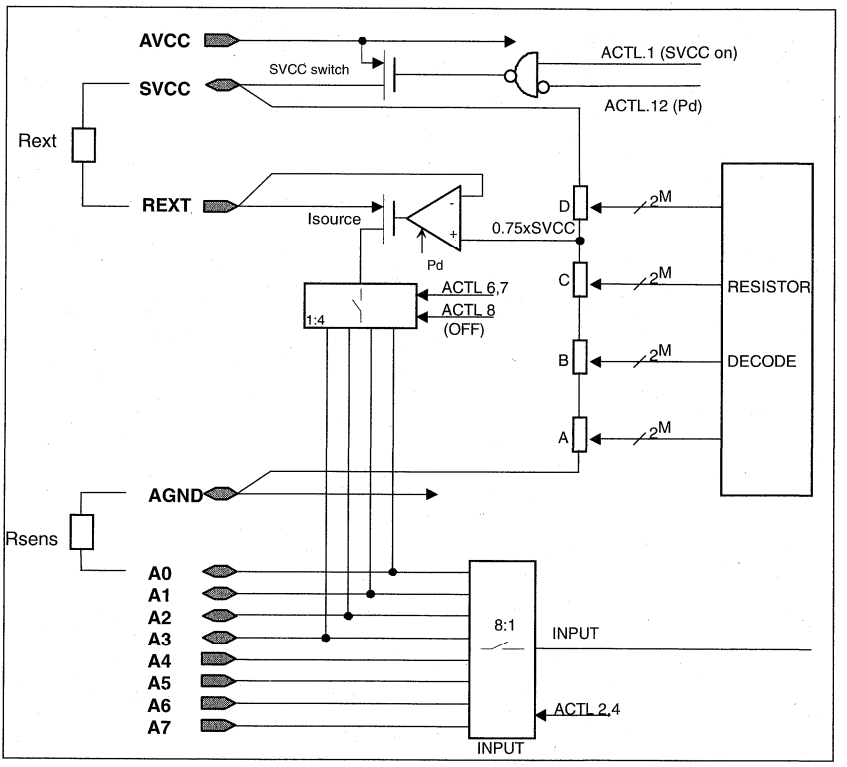


Figure 11.6: A/D Current Source

When the A/D converter is used in conjunction with resistive elements in sensor applications, current sources of precise value are required so that the input signal can be referred back to the supply voltage or voltage reference in the same manner as the current source, thereby allowing a ratiometric measurement to take place independent of the accuracy of the stable reference.

11.2.5 Analog Inputs and Multiplexer

Analog Inputs

The analog input signal is sampled onto an internal capacitor and held during the conversion. Since the charge of the capacitance is supplied by the source and the time to charge it up is defined by the sampling time of twelve ADCLK clocks the external source resistances and dynamic impedances must be limited so that the RC time constant is short enough to allow the analog inputs to completely settle within the allocated sampling time to a 12-bit accuracy. This is typically a time constant less than $0.8/f_{ADCLK}$. High source impedances have an adverse effect on the accuracy of the converter, not only due to the RC settling behaviour, but also due to voltage drops at the inputs due to leakage current or averaged DC input currents (due to input switching currents). Typically for a 12-bit converter, the error in LSBs due to leakage current is:

Error(LSBs) = 4*(µA of leakage current)*(kΩ of source resistance)/(volt of VREF)

Example: 50 nA leakage, 10 kΩ of source resistance, 3-V VREF gives 0.7 LSBs of error.

This also applies equally to the output impedance of the voltage reference source VREF as well. It must be low enough to enable the transients to settle within $(0.2/_{ADCLK})$ seconds, and generate leakage current induced errors of <<1LSB.

Analog Multiplexer

The analog multiplexer selects one of eight single-ended input channels, as determined by the bits in the ACTL register. It is based on a 'T-switch' to minimise the coupling between channels corrupting the analog input. Channels that are not selected are isolated from the A/D and the intermediate node connected to the analog ground AGND so that the stray capacitance is 'grounded' to eliminate cross-talk.

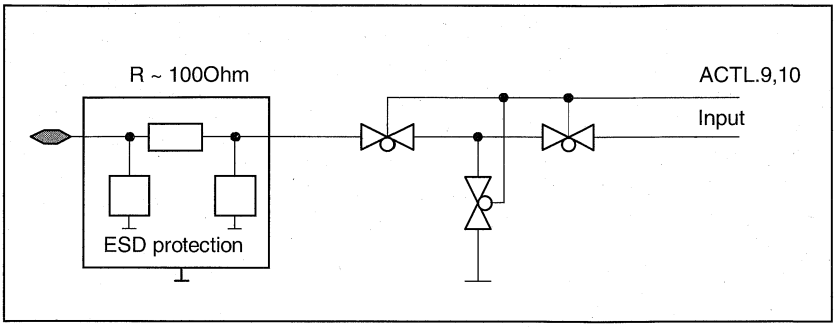


Figure 11.7: Analog Multiplexer

Cross-talk exists because there is always some parasitic coupling capacitance across the switch and between switches. This can take several forms, such as coupling from the input to the output of an 'OFF' switch, or coupling from an 'OFF' analog input channel to the output of another adjacent 'ON' output channel causing errors to creep into the digital code output. So for high accuracy conversions cross-talk interference must be minimised altogether by shielding and other well known printed circuit board (PCB) layout techniques.

11.2.6 A/D Grounding and Noise Considerations

As with any high resolution converter (≥ 12 bits) care and special attention has to be paid to the printed circuit board layout and the grounding scheme to eliminate ground loops and any unwanted parasitic components/effects and noise. There are standard techniques which are well documented in application notes that address these issues.

Ground Loops are formed when the return current from the resistor divider of the A/D flows through tracks that are common with other analog or digital circuitry. If care is not taken this current can generate small unwanted offset voltages that can add to or subtract from the reference or input voltages of the A/D converter. One way to avoid ground loops is to use the scheme where a 'star connection' is used for the AGND, thereby the ground current or reference currents do not flow through any common input leads, eliminating any error voltages.

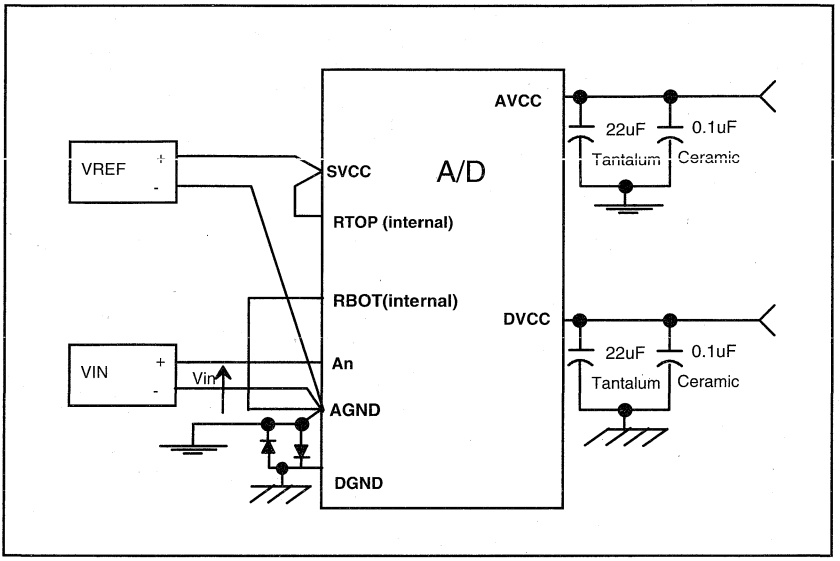


Figure 11.8: A/D Grounding and Noise Considerations

The digital ground DGND and analog ground AGND can also be star connected together, but if separate supplies are used then two reverse biased diodes limit the voltage difference to less than ±700mV.

Furthermore ripple and noise spikes on the power supply lines due to digital switching or switching power supplies are especially troublesome.

Normally the internal noise is very small and the total input referred noise is far less than one LSB so the output code is fairly stable. However as noise couples into the device via the supply and ground changes, the noise margin reduces and code uncertainty and jitter can creep in which may mean taking several readings to average out the noise effects. Another consequence is that as the reference voltage SVCC or VREF is reduced, the absolute value of the LSB also reduces, and therefore the noise becomes even more dominant as the noise margin reduces. Thus a clean, totally noise-free setup becomes of even more paramount importance to achieve the accuracies desired.

Adding carefully placed bypass capacitors returned to the respective ground planes helps in stabilising the supply current and minimising the 'noise' .

11.2.7 A/D Converter Input and Output Pins

Input Pins

There are two different types of input signals, the inputs of analog signals A0, A1, A2, A3, A4, A5, A6, A7 and REXT, SVCC.

The input signals coming from channel A0 to A7 can be treated as analog signals that should be handled with the A/D converter or as digital inputs to be read into the processing unit.

An external resistor between REXT and SVCC determines the amount of current flowing through an activated current source operation. The pin SVCC is then used as an output or input. The SVCC pin is an input when the internal SVCC switch is off and the Vref is applied externally. It is an output when the internal SVCC switch is on.

Output Pins

There are two different types of output signals, the outputs A0, A1, A2, A3 and the SVCC.
Current will flow out of one of the analog pins A0, A1, A2, A3 if the current source function is selected. The SVCC pin will then have a voltage just below the AVCC when the SVCC switch is on.

Supply Pins

There are four supply pins to split the digital and the analog current pathes:

AVCC,DVCC,AGND,DGND.

Some of the MSP430 family members will have all four supply pins bonded out for high analog resolution while others will have analog and digital VCC and/or GND rails internally connected.

11.3 ADC Control Registers

Four control registers are implemented:

Register	short form	Register type	Address	Initial state
• Input register:	AIN	Type of read only	0110h	- - -
• Input enable register:	AEN	Type of read/write	0112h	reset
• ADC control register:	ACTL	Type of read/write	0114h	→see figure
• Reserved				0116h
• ADC data register:	ADAT	Type of read	0118h	- - -

All registers may be accessed by any instruction subject to register read/write restrictions.

Input register AIN

The signals at the pins A0 to A7 can be signals from an analog source or a digital source. The value of digital sources can be read with access to the input register. The reading of the digital sources is enabled by a selection done in the input enable register.

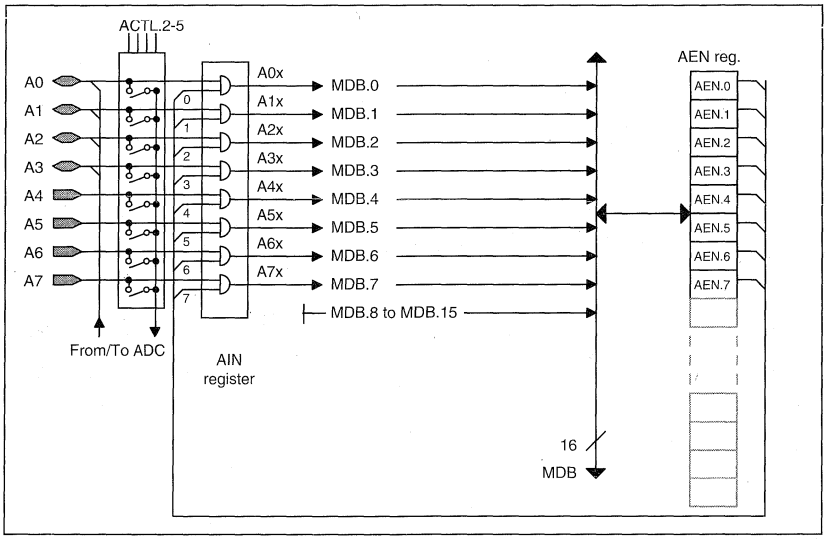
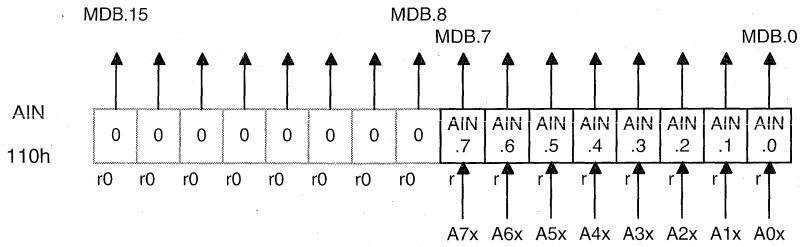


Figure 11.9: ADC Input Register, Input Register Enable

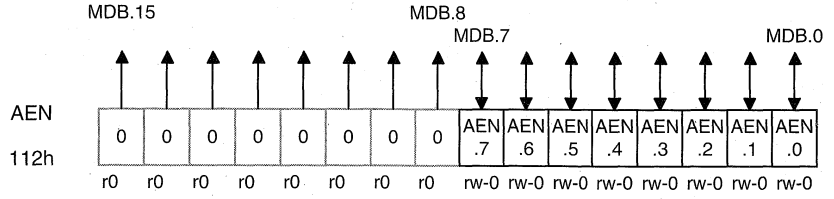
The input register AIN is a read only register connected to the 16-bit MDB. The LowByte of the register is implemented and MDB.0 to MDB.7 corresponds to A0 to A7. The HighByte of the register is read as 00h.



The signal at the corresponding input is logically gated with the appropriate enable signal, Ax .AND. AEN.x. Unselected (disabled) bits are read as '0'.

Input Enable Register AEN

The input enable register AEN is a read/write register connected to the 16-bit MDB. The LowByte of the register is implemented and MDB.0 to MDB.7 corresponds to A0 to A7. The HighByte of the register is read as 00h.



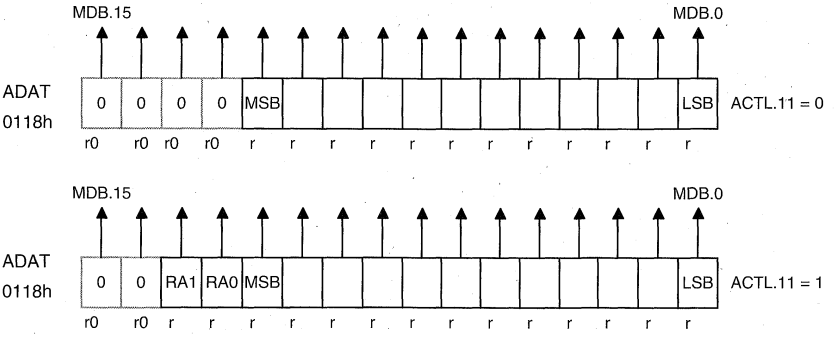
The input enable register bits control the definition of the individual bit:

AEN.x = 0 :	Analog input. The bit read at an access to the AIN register is 0.
AEN.x = 1 :	Digital input. The bit read at an access to the AIN represents the logical level at the appropriate pin.

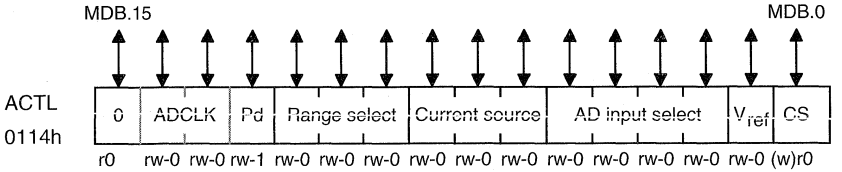
The initial state of all bits is reset.

ADC Data Register ADAT

The ADC data register holds the result of the analog-to-digital conversion. The conversion data are loaded into the register at the end of a conversion and remain until replaced by another conversion.



ADC Control Register ACTL



Bit 0, ACTL.0 : Convert Start

Bit 1, ACTL.1 : Source of Vref.

ACTL.1 = 0 : Switch SVCC is off. The output pin SVCC is not connected to VCC. The reference voltage of the ADC should be supplied from an external source.

ACTL.1 = 1 : Switch SVCC is on. The output pin SVCC is connected to VCC. The reference voltage of the ADC should not be supplied from an external source.

Bit 2-5, ACTL.2-.5 : AD input select.
These bits select the channel used for conversion. Channels should be changed only after completion of a conversion. Changing the channel while a conversion is active invalidates the conversion in progress.

ACTL.5	ACTL.4	ACTL.3	ACTL.2	Channel
0	0	0	0	A0
0	0	0	1	A1
0	0	1	0	A2
0	0	1	1	A3
0	1	0	0	A4
0	1	0	1	A5
0	1	1	0	A6
0	1	1	1	A7
1	X	X	X	NONE

Bit 6-8, ACTL.6-.8 : AD current source output select.
These bits select the channel used for output of the current source. Channels should be changed only after completion of a conversion. Changing the channel while a conversion is active invalidates the conversion in progress.

ACTL.8	ACTL.7	ACTL.6	Channel
0	0	0	A0
0	0	1	A1
0	1	0	A2
0	1	1	A3
1	X	X	NONE

Bit9-11, ACTL.9,.11: Range Select
These bits must not be changed once conversion has started. Any manipulation of these bits during a conversion will result in incorrect conversion data in ADAT.

ACTL.11	ACTL.10	ACTL.9	Range
0	0	0	A
0	0	1	B
0	1	0	C
0	1	1	D
1	X	X	Auto

Bit11, ACTL.11: Range Select Mode
ACTL.11 = 0 : The range select bits ACTL.9 and ACTL.10 have to be applied for manual range select.
ACTL.11 = 1 : The automatic range selection for a (12+2)-bit conversion is active. Since the manual range select is inactive the states of the range select bits ACTL.9 and ACTL.10 are don't care.

Bit12, ACTL.12: Power Down (Pd)
ACTL.12 = 1: SVCC switch is off,
COMPARATOR is powered down
Current source is off

Bits 13, 14 ADCLK
The clock frequency of the ADC should be adjusted to the maximum frequency described in the device's datasheet.

ACTL.14	ACTL.13	ADCLK
0	0	MCLK
0	1	MCLK/2
1	0	MCLK/3
1	1	MCLK/4

Bits 13-15 Reserved.

Test underflow and overflow condition by software

; Since the ADAT register is implemented in the 16-bit peripheral address space and
; integrated to handle data in word mode, no implementation of overflow or underflow
error ; detection is mandatory. It is reduced to simple commands in program flow:

```
; Bit11 of the ACTL register is reset, a 12-bit conversion was active
;
    CMP   #0,&ADAT          ; test for 12-bit ADC underflow
    JEQ   UndFlow           ; Yes, continue with underflow handling

    CMP   #0FFFh,&ADAT      ; test for 12-bit ADC overflow
;
; The MSBits, not implemented in the converter's hardware are read as 0s
    JEQ   OverFlow          ; Yes, continue with overflow handling


; Bit11 of the ACTL register is set, a (12+2)-bit conversion was active
; The conversion range should be limited to Range A to C
;
    CMP   #0,&ADAT          ; test for (12+2)-bit ADC underflow
    JEQ   UndFlow           ; Yes, continue with underflow handling

    CMP   #2FFFh,&ADAT      ; test only for Range A to C overflow
;
; The MSBits, not implemented in the converter's hardware are read as 0s
    JHS   OverFlow          ; Yes, continue with overflow handling
```

12 Miscellaneous Modules

12.1 FET switch

The FET switch is a transistor whose on- and off-state can be controlled by applying an external digital signal to the appropriate control pin FETI. This control pin is designed to be driven even with voltage levels above the positive supply voltage.

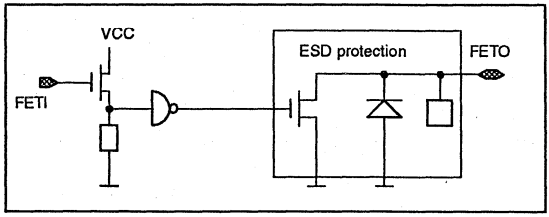


Figure 12.1: FET switch schematic

> **Note: Put FETI to GND**
>
> Put FETI to GND if FET switch is unused. A floating input will increase current consumption by VCC to GND current path.

FET switch application, Meter-Bus (M-Bus) mixed supply mode

The MSP430 in this configuration is supplied either from the M-Bus via the TSS721 or from the battery. As long as the bus lines are powered the VS signal at the TSS721 is high and the FET transistor in the MSP430 is off.

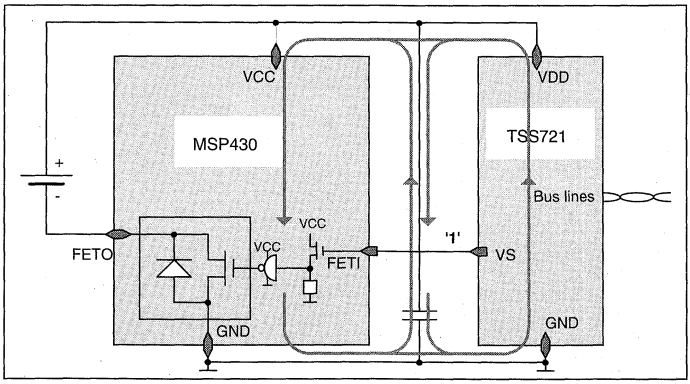


Figure 12.2: Remote Power Operation

The TSS721 is an interface circuit to connect the two wire Meter-Bus system to a slave system. It is designed to support battery operating systems, systems that are remote supplied from the Meter-Bus master or system with mixed battery/remote supply.
When the remote power from the M-Bus drops down the VS signal will change to ground potential and the FET switch in the MSP430 is turned on.

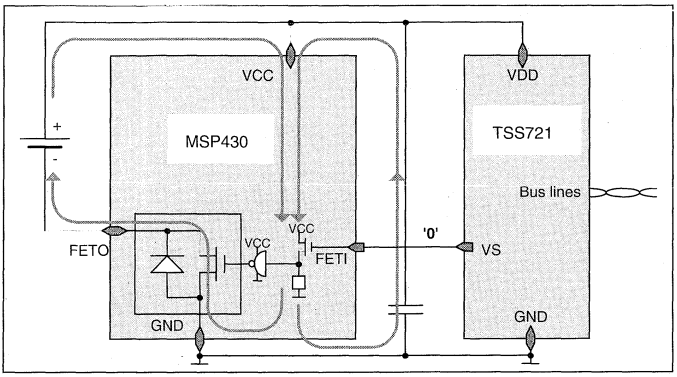


Figure 12.3: Battery Operation

12.2 Crystal Oscillator

All elements for crystal operation are integrated into the MSP430 - no additional external components are necessary for operation. Since the oscillator is designed for ultra-low power dissipation the PWB layout should provide short connections between the crystal and the MSP430 device.

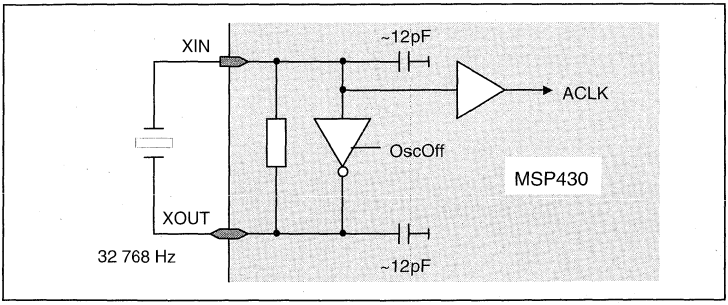


Figure 12.4: Crystal Oscillator schematic

When OscOff mode is selected the ACLK signal is held to high.

12.3 Power-on Circuitry

The power-on circuitry is part of the system reset scheme and consists of two parts, the power-on reset detection and the power-on reset delay. The output of the POR delay is fed into the POR latch and the PUC latch to set both latches in order to supply the system with the reset condition.

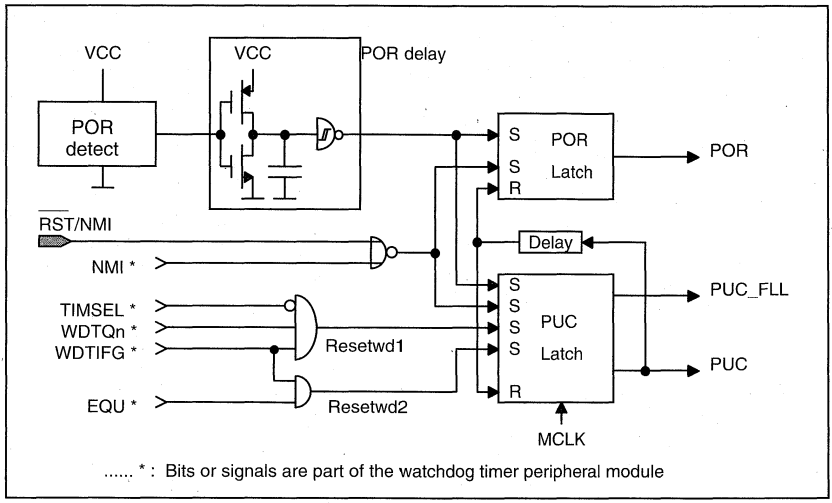


Figure 12.5: Power-on reset and Power-up clear schematic
When the VCC supply provides a fast VCC rise time the POR delay gives enough active time on the POR signal to allow it to initialize the circuit correctly after power-up.

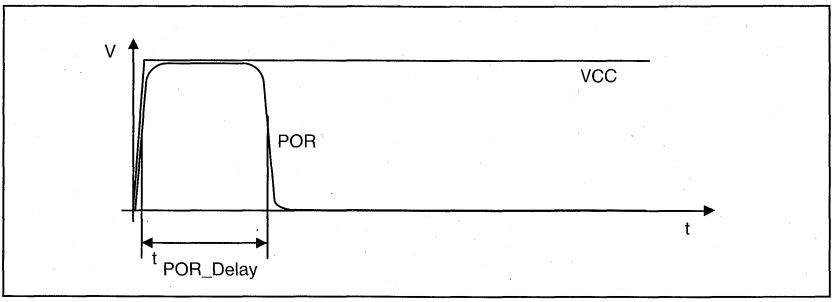


Figure 12.6: Power-on reset timing on fast VCC rise time

When the VCC supply provides a 'slow' VCC rise condition the POR detect defines the POR signal to allow it to initialize the circuit correctly after power-up.

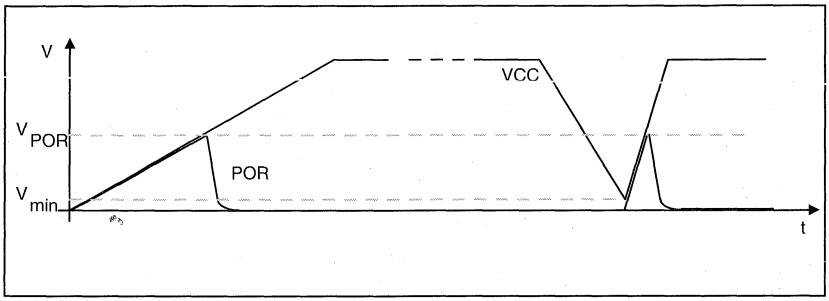


Figure 12.7: Power-on reset timing on slow VCC rise time

The supply voltage VCC should fall below V_{min} to ensure another POR signal occurs with the next increase in supply voltage. If V_{CC} does not fall below V_{min} a POR will not be generated and power-up conditions will not be set properly.

12.4 Crystal Buffer Output

The frequency of the buffer output is selected via the control register CBCTL.

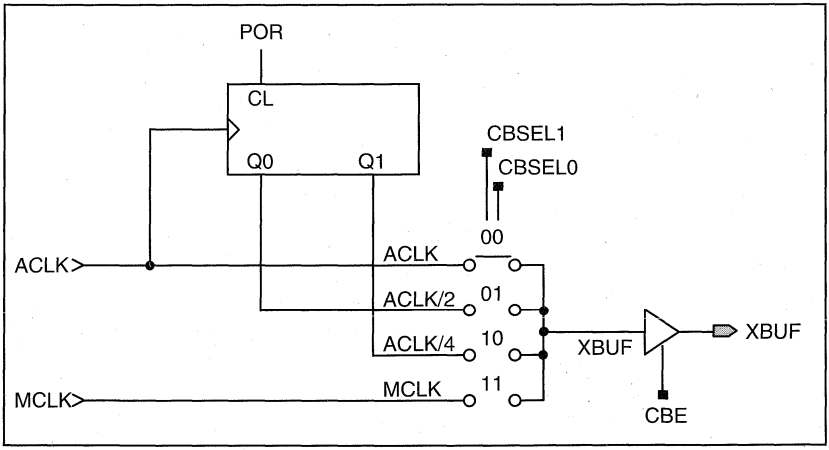


Figure 12.8: Schematic of Crystal Buffer

The control register CBCTL of the clock buffer output peripheral has bits that control the frequency applied to pin XBUF and one bit that controls the 3-state condition of the output buffer.

The divider runs with the minimum of logic necessary for correct operation. For example it is halted when ACLK or MCLK is selected or if the CBE bit is set.

The three bits in the control register CBSEL1, CBSEL0 and CBE are reset with POR signal. The POR signal is active either during switching on V_{CC} or when $\overline{RST}$/NMI-pin is tied to V_{SS} when reset function is selected

CBCTL
053h

---	---	---	---	---	CBSEL 1	CBSEL 0	CBE
					w-(0)	w-(0)	w-(0)

Bit 0: The bit CBE controls the 3-state condition of the output buffer.
CBE = 1: Output buffer enabled
CBE = 0: Output buffer disabled
During power-on reset (POR) the output buffer is always enabled to support external components with the selected frequency regardless of the source of POR.

Bit 1,2: The bits CBSEL1 and CBSEL0 select the frequency that can be put onto output pin XBUF.

CBSEL1	CBSEL0	XBUF1	
0	0	ACLK	← State after POR
0	1	ACLK/2	
1	0	ACLK/4	
1	1	MCLK	

A. Peripheral File Map

This appendix summarize the Peripheral File (PF) and control bit information into a single location for reference.

Each PF register is presented as a row of boxes containing the control or status bits belonging to the register. The register symbol (e.g. P0IN) and the PF hex address are to the left of each register.

The accessability and/or hardware conditions of each bit are indicated below each bit symbol, with the following definition:

- rw: read/write
- r: read only
- r0: read as '0'
- r1: read as '1'
- w: write only
- w0: write a '0'
- w1: write a '1'
- (w): no register bit implemented; writing a '1' will result in a puls the register bit is always read as '0'
- h0: cleared by hardware
- h1: set by hardware
- -0,-1: condition after PUC signal active (Reset condition).

Special function register, byte access

Bit # -	7	6	5	4	3	2	1	0
000Fh								
Module enable 2, ME2 0005h	BTME [1] rw							
Module enable 1, ME1 0004h			SPIME [4] rw-0					
Interrupt flag 2, IFG2 0003h	BTIFG rw					ADIFG rw-0		
Interrupt flag 1, IFG1 0002h			SPIIFG rw-0	NMIIFG rw-0	P0IFG.1 rw-0	P0IFG.0 rw-0	OFIFG rw-1	WDTIFG rw *)
Interrupt enable 2,IE2 0001h	BTIE rw-0				3)	2) rw-0		
Interrupt enable 1, IE1 0000h			SPIIE rw-0		P0IE.1 rw-0	P0IE.0 rw-0	OFIE rw-0	WDTIE rw-0

*) The WDTIFG is reset on POR signal and set with WDT overflow or WDT password violation.
1) Only valid in Basic Timer Module, not Basic Timer1
2) Configuration '201, '320: ADIE for 12+2b ADC (type: rw-0)
Configuration '310: TPIE for Timer/Port Module (type: rw-0)
3) Configuration '320: TPIE for Timer/Port Module (type: rw-0)
4) Future Module

Digital I/O frame, byte access

Bit # -	7	6	5	4	3	2	1	0
001Fh								
Interrupt Enable, P0IE 0015h	P0IE.7 rw-0	P0IE.6 rw-0	P0IE.5 rw-0	P0IE.4 rw-0	P0IE.3 rw-0	P0IE.2 rw-0	*) r0	*) r0
Int. Edge Sel., P0IES 0014h	P0IES.7 rw	P0IES.6 rw	P0IES.5 rw	P0IES.4 rw	P0IES.3 rw	P0IES.2 rw	P0IES.1 rw	P0IES.0 rw
Interrupt Flags, P0IFG 0013h	P0IFG.7 rw-0	P0IFG.6 rw-0	P0IFG.5 rw-0	P0IFG.4 rw-0	P0IFG.3 rw-0	P0IFG.2 rw-0	*) r0	*) r0
Direction reg., P0DIR 0012h	P0DIR.7 rw-0	P0DIR.6 rw-0	P0DIR.5 rw-0	P0DIR.4 rw-0	P0DIR.3 rw-0	P0DIR.2 rw-0	P0DIR.1 rw-0	P0DIR.0 rw-0
Output reg., P0OUT 0011h	P0OUT.7 rw	P0OUT.6 rw	P0OUT.5 rw	P0OUT.4 rw	P0OUT.3 rw	P0OUT.2 rw	P0OUT.1 rw	P0OUT.0 rw
Input register, P0IN 0010h	P0IN.7 r	P0IN.6 r	P0IN.5 r	P0IN.4 r	P0IN.3 r	P0IN.2 r	P0IN.1 r	P0IN.0 r

*) These interrupt enable bits and flags are included in the SFR frame.

Bit # -	7	6	5	4	3	2	1	0
002Fh - 0020h	Reserved for future use							

LCD register frame, byte access

Bit # -	7	6	5	4	3	2	1	0
LCD Memory 15 003Fh	rw	rw	rw	rw	rw	rw	rw	rw
LCD Memory 14 003Eh	rw	rw	rw	rw	rw	rw	rw	rw
LCD Memory 13 003Dh	rw	rw	rw	rw	rw	rw	rw	rw
LCD Memory 12 003Ch	rw	rw	rw	rw	rw	rw	rw	rw
LCD Memory 11 003Bh	rw	rw	rw	rw	rw	rw	rw	rw
LCD Memory 10 003Ah	rw	rw	rw	rw	rw	rw	rw	rw
LCD Memory 9 0039h	rw	rw	rw	rw	rw	rw	rw	rw
LCD Memory 8 0038h	rw	rw	rw	rw	rw	rw	rw	rw
LCD Memory 7 0037h	rw	rw	rw	rw	rw	rw	rw	rw
LCD Memory 6 0036h	rw	rw	rw	rw	rw	rw	rw	rw
LCD Memory 5 0035h	rw	rw	rw	rw	rw	rw	rw	rw
LCD Memory 4 0034h	rw	rw	rw	rw	rw	rw	rw	rw
LCD Memory 3 0033h	rw	rw	rw	rw	rw	rw	rw	rw
LCD Memory 2 0032h	rw	rw	rw	rw	rw	rw	rw	rw
LCD Memory 1 0031h	rw	rw	rw	rw	rw	rw	rw	rw
LCD Cntl&Mode, LCDC 0030h	LCDM7 rw-0	LCDM6 rw-0	LCDM5 rw-0	LCDM4 rw-0	LCDM3 rw-0	LCDM2 rw-0	LCDM1 rw-0	LCDM0 rw-0

8bit Timer/Counter frame, Basic Timer frame, Timer/Port frame, byte access

Bit # -	7	6	5	4	3	2	1	0
Timer/Port Enable reg., TPE 04Fh	TPSSEL3 rw-0	TPSSEL2 rw-0	TPE.5 rw-0	TPE.4 rw-0	TPE.3 rw-0	TPE.2 rw-0	TPE.1 rw-0	TPE.0 rw-0
Timer/Port Data reg., TPD 04Eh	B16 rw-0	CPON rw-0	TPD.5 rw-0	TPD.4 rw-0	TPD.3 rw-0	TPD.2 rw-0	TPD.1 rw-0	TPD.0 rw-0
Timer/Port Counter1, TPCNT2 04Dh	2^7 rw	2^6 rw	2^5 rw	2^4 rw	2^3 rw	2^2 rw	2^1 rw	2^0 rw
Timer/Port Counter1, TPCNT1 04Ch	2^7 rw	2^6 rw	2^5 rw	2^4 rw	2^3 rw	2^2 rw	2^1 rw	2^0 rw
Timer/Port control reg., TPCTL 04Bh	TPSSEL1 rw-0	TPSSEL0 rw-0	ENB rw-0	ENA rw-0	EN1 r-0	RC2FG rw-0	RC1FG rw-0	EN1FG rw-0
Counter Data ***), 8bit Basic Timer, BTCNT2 0047h	2^7 rw	2^6 rw	2^5 rw	2^4 rw	2^3 rw	2^2 rw	2^1 rw	2^0 rw
Counter Data ***), 8bit Basic Timer, BTCNT1 0046h	2^7 rw	2^6 rw	2^5 rw	2^4 rw	2^3 rw	2^2 rw	2^1 rw	2^0 rw
045h								
Counter Data, 8bit Timer/Counter, TCDAT 0044h	TCDAT.7 rw	TCDAT.6 rw	TCDAT.5 rw	TCDAT.4 rw	TCDAT.3 rw	TCDAT.2 rw	TCDAT.1 rw	TCDAT.0 rw
Pre-load Register, 8bit Timer/Counter, TCPLD 0043h	TCPLD.7 rw	TCPLD.6 rw	TCPLD.5 rw	TCPLD.4 rw	TCPLD.3 rw	TCPLD.2 rw	TCPLD.1 rw	TCPLD.0 rw
Control Register, 8bit Timer/Counter, TCCTL 0042h	SSEL1 rw-0	SSEL0 rw-0	ISCTL rw-0	TXEN rw-0	ENCNT rw-0	RXACT rw-0	TXD rw-0	RXD r(-1)
0041h								
Basic Timer, BTCTL 0040h	SSEL rw	Reset * Hold ** rw	DIV rw	FRFQ1 rw	FRFQ0 rw	IP2 rw	IP1 rw	IP0 rw

*) Reset function in Basic Timer Module.
**) Hold function in Basic Timer1 Module.
***) The counter data register CTCNT1 and BTCNT2 can be accessed in Basic Timer1 Module not in Basic Timer Module.

PWM Timer, EPROM control register and System Clock Generator frame, byte access

Bit # -	7	6	5	4	3	2	1	0
PWM timer counter PWMCNT.2 005Fh	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
PWM duty register PWMDTR.2 005Eh	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
PWM duty buffer PWMDTB.2 005Dh	2^7	2^6	2^5	2^4	2^3	2^2	21	2^0
	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
PWM timer control register PWMCTL.2 005Ch	----	SSEL2	SSEL1	SSEL0	CMPM	----	OS	OE
	rw-0	rw-0	rw-0	rw-0	r	rw-0	rw-0	rw-0
PWM timer counter PWMCNT.1 005Bh	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
PWM duty register PWMDTR.1 005Ah	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
PWM duty buffer PWMDTB.1 0059h	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
PWM timer control register PWMCTL.1 0058h	----	SSEL2	SSEL1	SSEL0	CMPM	----	OS	OE
	rw-0	rw-0	rw-0	rw-0	r	rw-0	rw-0	rw-0
EPROM control register EPCTL 0054h							VPPS	EXE
	r-0	r-0	r-0	r-0	r-0	r-0	rw-0	rw-0
Crystal Buffer ctl. reg., CBCTL *) 053h						CBSEL1	CBSEL0	CBE
						w-(0)	w-(0)	w-(0)
System Clock Gen., Freq. Cntl. SCFQCTL 0052h	M	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	rw-0	rw-0	rw-0	rw-1	rw-1	rw-1	rw-1	rw-1
System Clock Gen., Freq. Integrator SCFI1 0051h	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2
	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
System Clock Gen., Freq. Integrator SCFI0 0050h	0	0	0	FN_4	FN_3	FN_2	2^1	2^0
	r	r	r	rw-0	rw-0	rw-0	rw-0	rw-0

*) CBSel1, CBSEL0 and CBE bit are reset with POR signal.

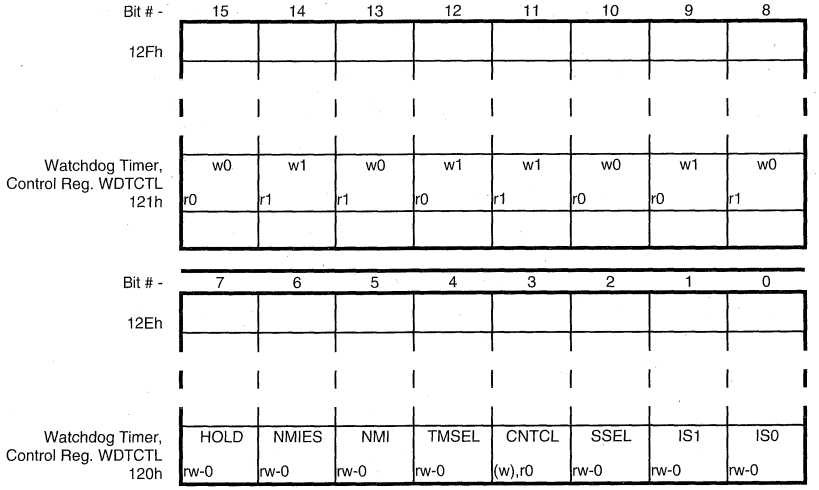
A/D converter register frame, word access

Bit # -	15	14	13	12	11	10	9	8
11Fh								
AD Converter, Data Register ADAT 118h	r0	r0	R1 *) r0	R0 *) r0	2^{11} r	2^{10} r	2^{9} r	2^{8} r
reserved 116h								
AD Converter, Control Register ACTL 114h	ACTL.15 r0	ACTL.14 r0	ACTL.13 r0	ACTL.12 rw-1	ACTL.11 rw-0	ACTL.10 rw-0	ACTL.9 rw-0	ACTL.8 rw-0
AD Converter, Input Enable Reg. AEN 112h	r0	r0	r0	r0	r0	r0	r0	r0
AD Converter, Input Data Reg. AIN 110h	r0	r0	r0	r0	r0	r0	r0	r0

*) The bits ADAT.12 and ADAT.13 are read as 0 when ACTL.11=0 otherwise signals R0 and R1 are read.

Bit # -	7	6	5	4	3	2	1	0
11Eh								
AD Converter, Data Register ADAT 118h	2^{7} r	2^{6} r	2^{5} r	2^{4} r	2^{3} r	2^{2} r	2^{1} r	2^{0} r
reserved 116h								
AD Converter, Control Register ACTL 114h	ACTL.7 rw-0	ACTL.6 rw-0	ACTL.5 rw-0	ACTL.4 rw-0	ACTL.3 rw-0	ACTL.2 rw-0	ACTL.1 rw-0	ACTL.0 (w)r0
AD Converter, Input Enable Reg. AEN 112h	AEN.7 rw-0	AEN.6 rw-0	AEN.5 rw-0	AEN.4 rw-0	AEN.3 rw-0	AEN.2 rw-0	AEN.1 rw-0	AEN.0 rw-0
AD Converter, Input Data Reg. AIN 110h	AIN.7 r	AIN.6 r	AIN.5 r	AIN.4 r	AIN.3 r	AIN.2 r	AIN.1 r	AIN.0 r

Watchdog/Timer register frame, word access



INDEX

TI SC Sales Offices in Europe

Belgium
Texas Instruments S.A./N.V.
Brussels
Tel.: (02) 7 26 75 80
Fax: (02) 7 26 72 76

Finland
Texas Instruments OY
Espoo
Tel.: (0) 43 54 20 33
Fax: (0) 46 73 23

France,
Middle-East & Africa
Texas Instruments
Velizy Villacoublay
Tel.: (1) 30 70 10 01
Fax: (1) 30 70 10 54

Germany
Texas Instruments GmbH
Freising
Tel.: (0 81 61) 80-0
Fax: (0 81 61) 80 45 16

Hannover
Tel.: (05 11) 90 49 60
Fax: (05 11) 6 49 03 31

Ostfildern
Tel.: (07 11) 3 40 30
Fax: (07 11) 3 40 32 57

Holland
Texas Instruments B.V.
Amstelveen
Tel.: (0 20) 6 40 04 16
Fax: (0 20) 5 45 06 60
(0 20) 6 40 38 46

Hungary
TI Representation:
Budapest
Tel.: (1) 2 09 22 11
Fax: (1) 2 67 13 57

Italy
Texas Instruments S.p.A.
Agrate Brianza (Mi)
Tel.: (0 39) 6 84 21
Fax: (0 39) 6 84 29 12

Republic of Ireland
Texas Instruments Ltd.
Dublin
Tel.: (01) 4 75 52 33
Fax: (01) 4 78 14 63

Spain
Texas Instruments S.A.
Madrid
Tel.: (1) 3 72 80 51
Fax: (1) 3 72 82 66

Sweden
Texas Instruments
International Trade Corporation
Kista
Tel.: (08) 7 52 58 00
Fax: (08) 7 51 97 15

United Kingdom
Texas Instruments Ltd.
Northampton
Tel.: (0 16 04) 66 30 00
Fax: (0 16 04) 66 30 01

TEXAS INSTRUMENTS

TI Technology Centres

France
Texas Instruments
Velizy Villacoublay
Tel.: Standard:
(1) 30 70 10 01
Technical Service:
(1) 30 70 11 33

Holland
Texas Instruments B.V.
Amstelveen
Tel.: (0 20) 5 45 06 00
Fax: (0 20) 6 40 38 46

Italy
Texas Instruments S.p.A.
Agrate Brianza (Mi)
Tel.: (0 39) 6 84 21
Fax: (0 39) 6 84 29 12

Sweden
Texas Instruments
International Trade Corporation
Kista
Tel.: (08) 7 52 58 00
Fax: (08) 7 51 97 15

European SC Information Centre

Telephone:
Dutch (33) 1 30 70 11 66
English (33) 1 30 70 11 65
French (33) 1 30 70 11 64
German (33) 1 30 70 11 68
Italian (33) 1 30 70 11 67

Fax: (33) 1 30 70 10 32

Printed in Germany
by Sellier Druck, Angerstr. 54, 85354, Freising

Chlorine free paper – to protect our environment

Errata Sheet

MSP430 Architecture User's Guide and Module Library

Please correct the below part on page 3-12:

Interrupt Source	Interrupt flag	System Interrupt	Word Adress	Priority
Timer/Port[1)]		maskable	0FFECh	6
(Timer B)**		maskable	0FFE8h	4

into:

Interrupt Source	Interrupt flag	System Interrupt	Word Adress	Priority
Timer/Port[1)]		maskable	0FFE8h	4
(Timer B)**				

1) Timer/Port vector in '320 configuration
**) Preliminary definition

Table 3.1: Interrupt sources, flags and vector

All other parts of the above table remain unchanged.

SLAUE10A