MSP430i2xx Family

User's Guide

Literature Number: SLAU335
August 2014

Contents

List of Figures

List of Tables

Read This First

About This Manual

This manual describes the modules and peripherals of the MSP430i2xx family of devices. Each description presents the module or peripheral in a general sense. Not all features and functions of all modules or peripherals may be present on all devices. In addition, modules or peripherals may differ in their exact implementation between device families, or may not be fully implemented on an individual device or device family.

Pin functions, internal signal connections, and operational parameters differ from device to device. The user should consult the device-specific data sheet for these details.

Related Documentation From Texas Instruments

For related documentation see the web site http://www.ti.com/msp430.

Notational Conventions

Program examples are shown in a `special typeface`.

Glossary

ACLK	Auxiliary Clock
ADC	Analog-to-Digital Converter
BOR	Brown-Out Reset
BSL	Bootstrap Loader; see www.ti.com/msp430 for application reports
CPU	Central Processing Unit
DAC	Digital-to-Analog Converter
DCO	Digitally Controlled Oscillator
dst	Destination
FLL	Frequency Locked Loop
GIE Modes	General Interrupt Enable
INT(N/2)	Integer portion of N/2
I/O	Input/Output
ISR	Interrupt Service Routine
LSB	Least-Significant Bit
LSD	Least-Significant Digit
LPM	Low-Power Mode; also named PM for Power Mode
MAB	Memory Address Bus
MCLK	Master Clock
MDB	Memory Data Bus
MSB	Most-Significant Bit
MSD	Most-Significant Digit
NMI	(Non)-Maskable Interrupt; also split to UNMI and SNMI
PC	Program Counter
PM	Power Mode
POR	Power-On Reset
PUC	Power-Up Clear

RAM	Random Access Memory
SCG	System Clock Generator
SFR	Special Function Register
SMCLK	Sub-System Master Clock
SNMI	System NMI
SP	Stack Pointer
SR	Status Register
src	Source
TOS	Top of stack
UNMI	User NMI
WDT	Watchdog Timer
z16	16-bit address space

Register Bit Conventions

Each register is shown with a key indicating the accessibility of the each individual bit, and the initial condition:

Register Bit Accessibility and Initial Condition

Key	Bit Accessibility
rw	Read/write
r	Read only
r0	Read as 0
r1	Read as 1
w	Write only
w0	Write as 0
w1	Write as 1
(w)	No register bit implemented; writing a 1 results in a pulse. The register bit is always read as 0.
h0	Cleared by hardware
h1	Set by hardware
-0,-1	Condition after PUC
-(0),-(1)	Condition after POR
-[0],-[1]	Condition after BOR
-{0},-{1}	Condition after Brownout

System Resets, Interrupts, and Operating Modes

This chapter describes the system resets, interrupts, and operating modes of MSP430i devices.

Topic **Page**

1.1 System Reset and Initialization

Figure 1-1 shows the system reset circuitry, which sources a brownout reset (BOR), a power-on reset (POR), and a power-up clear (PUC). Different events trigger these reset signals and different initial conditions exist depending on which signal is generated.

A BOR is a device reset. A BOR is generated by the following events:

- Powering up the device
- A wakeup event from LPM4.5 mode

A POR is always generated when a BOR is generated, but a BOR is not generated by a POR. The following events trigger a POR:

- A BOR signal
- A low signal on $\overline{\text{RST}}$/NMI pin when configured in the reset mode

A PUC is always generated when a POR is generated, but a POR is not generated by a PUC. The following events trigger a PUC:

- A POR signal
- Watchdog timer expiration when in watchdog mode
- Watchdog timer password violation
- A flash memory password violation
- A fetch from peripheral area

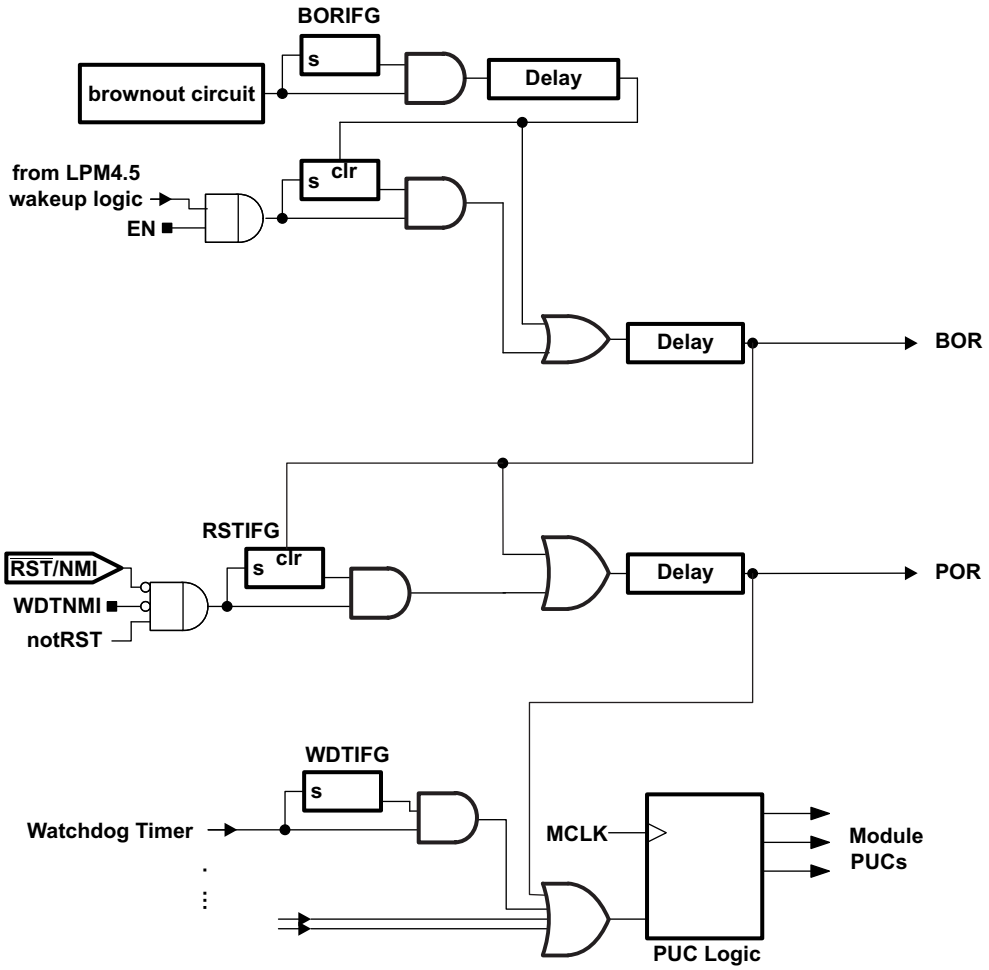


Figure 1-1. BOR, POR, and PUC Reset Circuit

1.1.1 Device Initial Conditions After System Reset

After a BOR, the initial device conditions are:

- The $\overline{\text{RST}}$/NMI pin is configured in the reset mode.
- I/O pins are set to input mode.
- Other peripheral modules and registers are initialized as described in their respective chapters.
- The Status register (SR) is reset.
- The watchdog timer powers up active in watchdog mode.
- The program counter (PC) is loaded with the address that is contained at the reset vector location (0FFFEh).

After a system reset, user software must initialize the device for the application requirements. The following must occur:

- Initialize the stack pointer (SP), typically to the top of RAM.
- Initialize the watchdog to the requirements of the application.
- Configure peripheral modules to the requirements of the application.

NOTE: A device that is unprogrammed or blank is defined as having its reset vector value, which resides at memory address FFFEh, equal to FFFFh. Upon system reset of a blank device, the device automatically enters operating mode LPM4. See Section 1.3 for information on operating modes and Section 1.2.4 for details on interrupt vectors.

1.2 Interrupts

The interrupt priorities are fixed and defined by the arrangement of the modules in the connection chain as shown in Figure 1-2. The nearer a module is to the CPU/NMIRS, the higher the priority. Interrupt priorities determine which interrupt is taken when more than one interrupt is pending simultaneously.

There are three types of interrupts:

- System reset (see Section 1.1)
- (Non)-maskable NMI (see Section 1.2.1)
- Maskable (see Section 1.2.2)

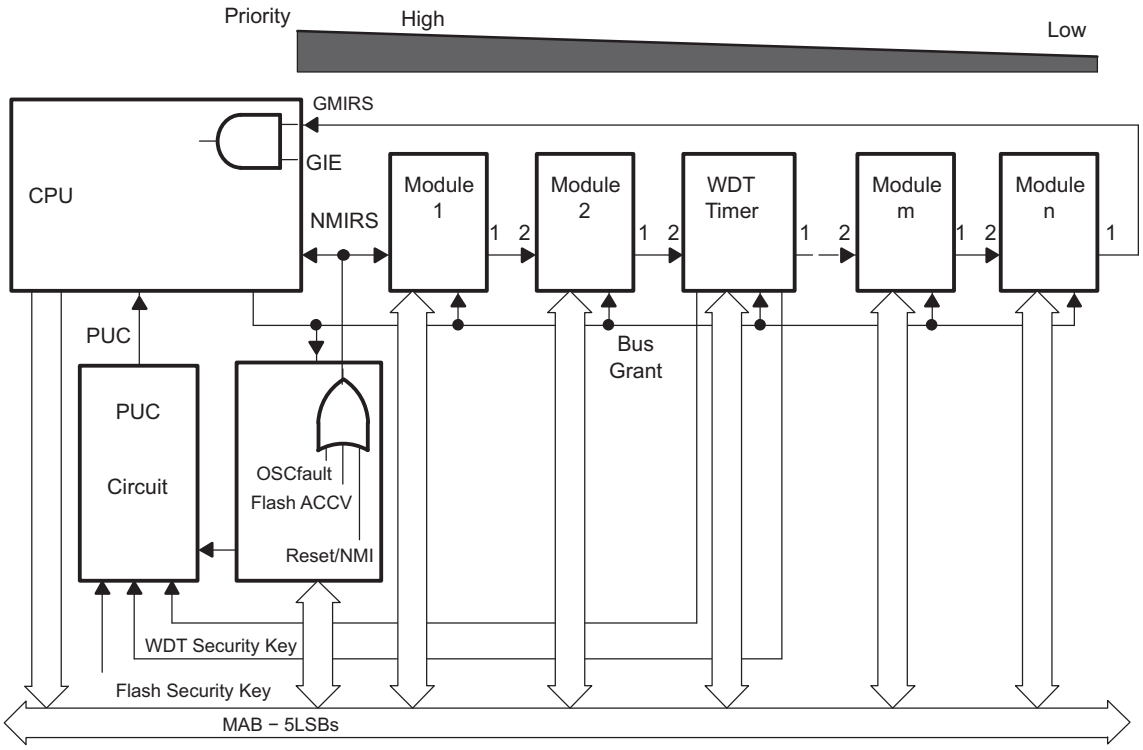


Figure 1-2. Interrupt Priority

1.2.1 (Non)-Maskable Interrupts (NMI)

(Non)-maskable NMI interrupts are not masked by the general interrupt enable bit (GIE) but are enabled by individual interrupt enable bits (NMIIE, ACCVIE, OFIE). When a NMI interrupt is accepted, all NMI interrupt enable bits are automatically reset. Program execution begins at the address stored in the (non)-maskable interrupt vector, 0FFFCh. User software must set the required NMI interrupt enable bits for the interrupt to be re-enabled. The block diagram for NMI sources is shown in Figure 1-3.

A (non)-maskable NMI interrupt can be generated by three sources:

- An edge on the $\overline{\text{RST}}$/NMI pin when configured in NMI mode
- An oscillator fault occurs
- An access violation to the flash memory

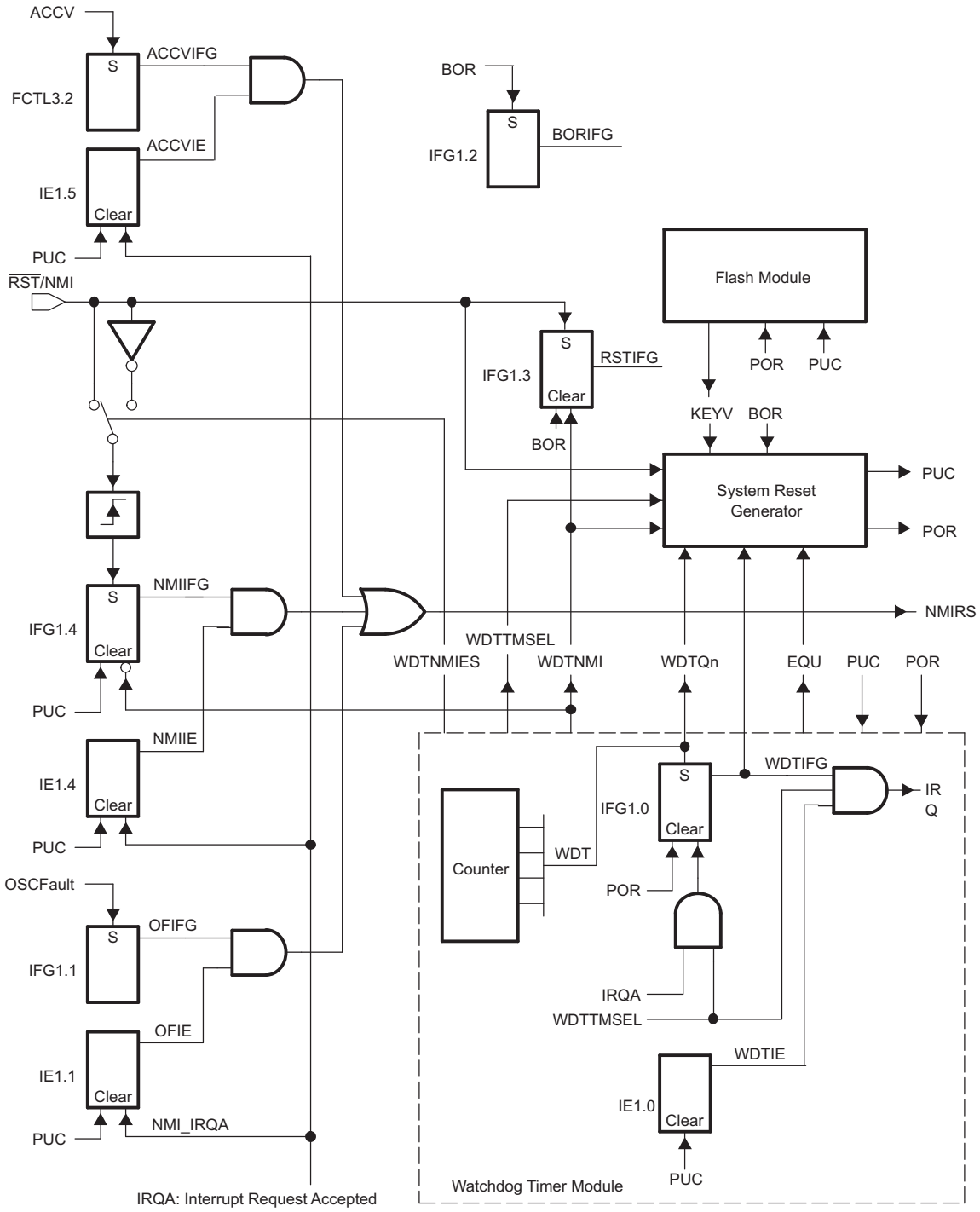


Figure 1-3. Block Diagram of (Non)-Maskable Interrupt Sources

1.2.1.1 $\overline{\text{RST}}$/NMI Pin

At power-up, the $\overline{\text{RST}}$/NMI pin is configured in the reset mode. The function of the $\overline{\text{RST}}$/NMI pin is selected in the watchdog control register WDTCTL. If the $\overline{\text{RST}}$/NMI pin is set to the reset function, the CPU is held in the reset state as long as the $\overline{\text{RST}}$/NMI pin is held low. After the input changes to a high state, the CPU starts program execution at the word address stored in the reset vector, 0FFFEh, and the RSTIFG flag is set.

If the $\overline{\text{RST}}$/NMI pin is configured by user software to the NMI function, a signal edge selected by the WDTNMIES bit generates an NMI interrupt if the NMIIE bit is set. The $\overline{\text{RST}}$/NMI flag NMIIFG is also set.

NOTE: Holding $\overline{\text{RST}}$/NMI Low

When configured in the NMI mode, a signal generating an NMI event should not hold the $\overline{\text{RST}}$/NMI pin low. If a PUC occurs from a different source while the NMI signal is low, the device is held in the reset state because a PUC changes the $\overline{\text{RST}}$/NMI pin to the reset function.

NOTE: Modifying WDTNMIES

When NMI mode is selected and the WDTNMIES bit is changed, an NMI can be generated, depending on the actual level at the $\overline{\text{RST}}$/NMI pin. When the NMI edge select bit is changed before selecting the NMI mode, no NMI is generated.

1.2.1.2 Flash Access Violation

The flash ACCVIFG flag is set when a flash access violation occurs. The flash access violation can be enabled to generate an NMI interrupt by setting the ACCVIE bit. The ACCVIFG flag can then be tested by the NMI interrupt service routine to determine if the NMI was caused by a flash access violation.

1.2.1.3 Oscillator Fault

The oscillator fault flag OFIFG is set when there is a fault with DCO in the External Resistor mode. The oscillator fault can be enabled to generate an NMI interrupt by setting the OFIE bit. The OFIFG flag can then be tested by NMI the interrupt service routine to determine if the NMI was caused by an oscillator fault.

1.2.1.4 Example of an NMI Interrupt Handler

The NMI interrupt is a multiple-source interrupt. An NMI interrupt automatically resets the NMIIE, OFIE and ACCVIE interrupt-enable bits. The user NMI service routine resets the interrupt flags and re-enables the interrupt-enable bits according to the application needs as shown in Figure 1-4.

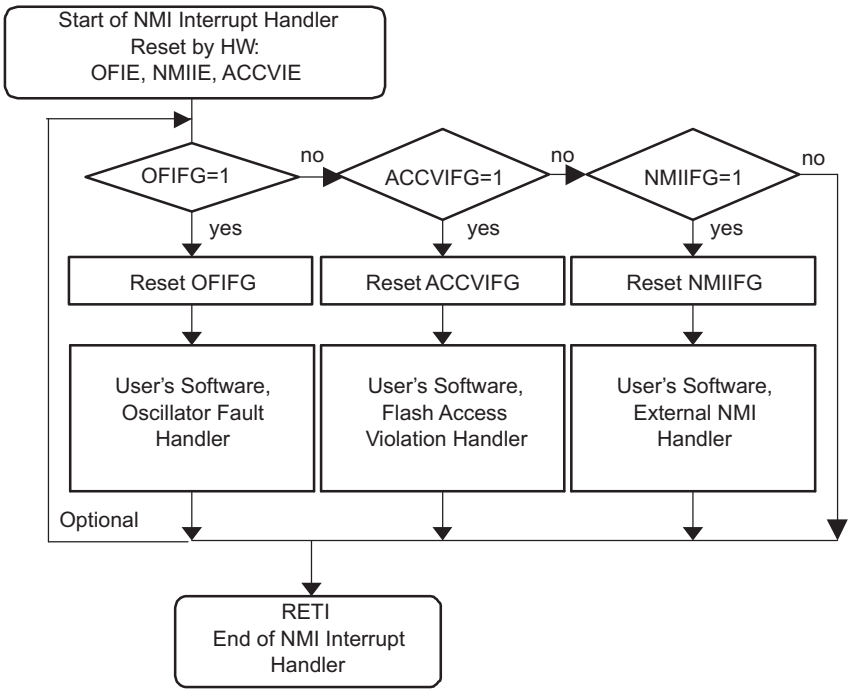


Figure 1-4. NMI Interrupt Handler

NOTE: Enabling NMI Interrupts with ACCVIE, NMIIE, and OFIE

To prevent nested NMI interrupts, the ACCVIE, NMIIE, and OFIE enable bits should not be set inside of an NMI interrupt service routine.

1.2.2 Maskable Interrupts

Maskable interrupts are caused by peripherals with interrupt capability including the watchdog timer overflow in interval-timer mode. Each maskable interrupt source can be disabled individually by an interrupt enable bit, or all maskable interrupts can be disabled by the general interrupt enable (GIE) bit in the status register (SR).

Each individual peripheral interrupt is discussed in the associated peripheral module chapter in this manual.

1.2.3 Interrupt Processing

When an interrupt is requested from a peripheral and the peripheral interrupt enable bit and GIE bit are set, the interrupt service routine is requested. Only the individual enable bit must be set for (non)-maskable interrupts to be requested.

1.2.3.1 Interrupt Acceptance

The interrupt latency is 6 cycles, starting with the acceptance of an interrupt request and lasting until the start of execution of the first instruction of the interrupt-service routine, as shown in Figure 1-5. The interrupt logic executes the following:

1. Any currently executing instruction is completed.
2. The PC, which points to the next instruction, is pushed onto the stack.
3. The SR is pushed onto the stack.
4. The interrupt with the highest priority is selected if multiple interrupts occurred during the last instruction and are pending for service.
5. The interrupt request flag resets automatically on single-source flags. Multiple source flags remain set for servicing by software.
6. The SR is cleared. This terminates any low-power mode. Because the GIE bit is cleared, further interrupts are disabled.
7. The content of the interrupt vector is loaded into the PC: the program continues with the interrupt service routine at that address.

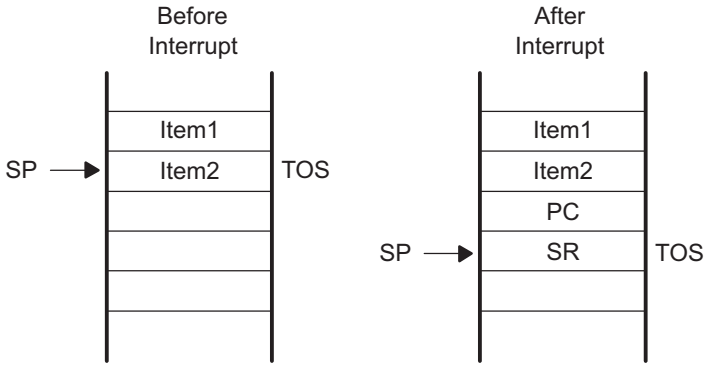


Figure 1-5. Interrupt Processing

1.2.3.2 Return From Interrupt

The interrupt handling routine terminates with the instruction:

RETI (return from an interrupt service routine)

The return from the interrupt takes 5 cycles to execute the following actions and is illustrated in Figure 1-6.

1. The SR with all previous settings pops from the stack. All previous settings of GIE, CPUOFF, etc. are now in effect, regardless of the settings used during the interrupt service routine.
2. The PC pops from the stack and begins execution at the point where it was interrupted.

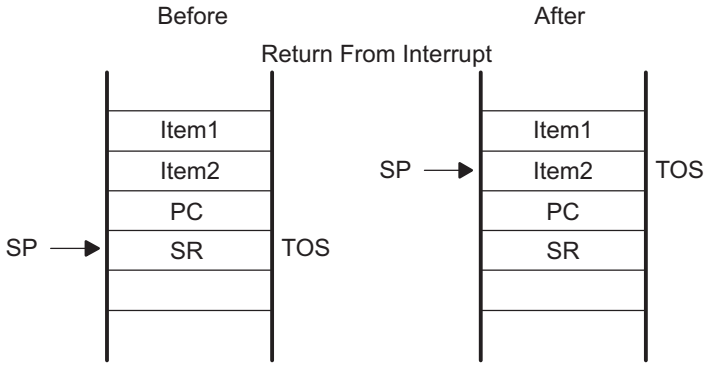


Figure 1-6. Return From Interrupt

1.2.3.3 Interrupt Nesting

Interrupt nesting is enabled if the GIE bit is set inside an interrupt service routine. When interrupt nesting is enabled, any interrupt occurring during an interrupt service routine will interrupt the routine, regardless of the interrupt priorities.

1.2.4 Interrupt Vectors

The interrupt vectors and the power-up starting address are located in the address range 0FFFFh to 0FFC0h, as described in Table 1-1. The end address of interrupt vector table is device specific. A vector is programmed by the user with the 16-bit address of the corresponding interrupt service routine. See the device-specific data sheet for the complete interrupt vector list.

It is recommended to provide an interrupt service routine for each interrupt vector that is assigned to a module. A dummy interrupt service routine can consist of just the RETI instruction and several interrupt vectors can point to it.

Unassigned interrupt vectors can be used for regular program code if necessary.

Some interrupt enable bits, and interrupt flags are located in the SFRs. The SFRs are located in the lower address range and are implemented in byte format. SFRs must be accessed using byte instructions. See the device-specific data sheet for the SFR configuration.

Table 1-1. Interrupt Sources, Flags, and Vectors

Interrupt Source	Interrupt Flag	System Interrupt	Word Address	Priority
Power-up, external reset, watchdog, flash password, illegal instruction fetch	BORIFG RSTIFG WDTIFG KEYV	Reset	0FFFEh	31, highest
NMI, oscillator fault, flash memory access violation	NMIIFG OFIFG ACCVIFG	(non)-maskable (non)-maskable (non)-maskable	0FFFCh	30
device-specific			0FFFAh	29
device-specific			0FFF8h	28
device-specific			0FFF6h	27

Table 1-1. Interrupt Sources, Flags, and Vectors (continued)

Interrupt Source	Interrupt Flag	System Interrupt	Word Address	Priority
Watchdog timer	WDTIFG	maskable	0FFF4h	26
device-specific			0FFF2h	25
device-specific			0FFF0h	24
device-specific			0FFEEh	23
device-specific			0FFECh	22
device-specific			0FFEAh	21
device-specific			0FFE8h	20
device-specific			0FFE6h	19
device-specific			0FFE4h	18
device-specific			0FFE2h	17
device-specific			0FFE0h	16
device-specific			0FFDEh	15
device-specific			0FFDCh	14
device-specific			0FFDAh	13
device-specific			0FFD8h	12
device-specific			0FFD6h	11
device-specific			0FFD4h	10
device-specific			0FFD2h	9
device-specific			0FFD0h	8
device-specific			0FFCEh	7
device-specific			0FFCCh	6
device-specific			0FFCAh	5
device-specific			0FFC8h	4
device-specific			0FFC6h	3
device-specific			0FFC4h	2
device-specific			0FFC2h	1
device-specific			0FFC0h	0, lowest

1.3 Operating Modes

The MSP430i devices have one active mode and four software selectable low power modes. These low power modes are LPM0/LPM1, LPM2/LPM3, LPM4, and LPM4.5. LPM0 and LPM1 are identical (LPM0 = LPM1) and similarly LPM2 and LPM3 are identical (LPM2 = LPM3). An interrupt event can wake up the device from the low-power modes LPM0 to LPM4, service the request, and restore back the low-power mode on return from the interrupt program.

The low-power modes 0 to 4 are configured with the CPUOFF, SCG1, and OSCOFF bits in the CPU status register. REGOFF bit in the PMM LPM45CTL register is used in addition for the LPM4.5 mode. The advantage of including the CPUOFF, SCG1, and OSCOFF mode-control bits in the status register is that the present operating mode is saved onto the stack during an interrupt service routine. Program flow returns to the previous operating mode if the saved SR value is not altered during the interrupt service routine. Program flow can be returned to a different operating mode by manipulating the saved SR value on the stack inside of the interrupt service routine. The mode-control bits and the stack can be accessed with any instruction.

When setting any of the mode-control bits, the selected operating mode takes effect immediately (see Figure 1-7). Peripherals operating with any disabled clock are disabled until the clock becomes active. The peripherals may also be disabled with their individual control register settings. All I/O port pins and RAM/registers are unchanged in low power modes LPM0 to LPM4. Wake up from LPM0 to LPM4 is possible through all enabled interrupts. Core voltage is turned off in LPM4.5 mode. RAM/registers are not retained but I/O port pin states are latched during LPM4.5 mode. Wake up from LPM4.5 is possible through the specific I/Os designated with LPM4.5 mode wake up capability. Refer to device specific datasheet for information on LPMx or LPM4.5 wake up capable I/Os.

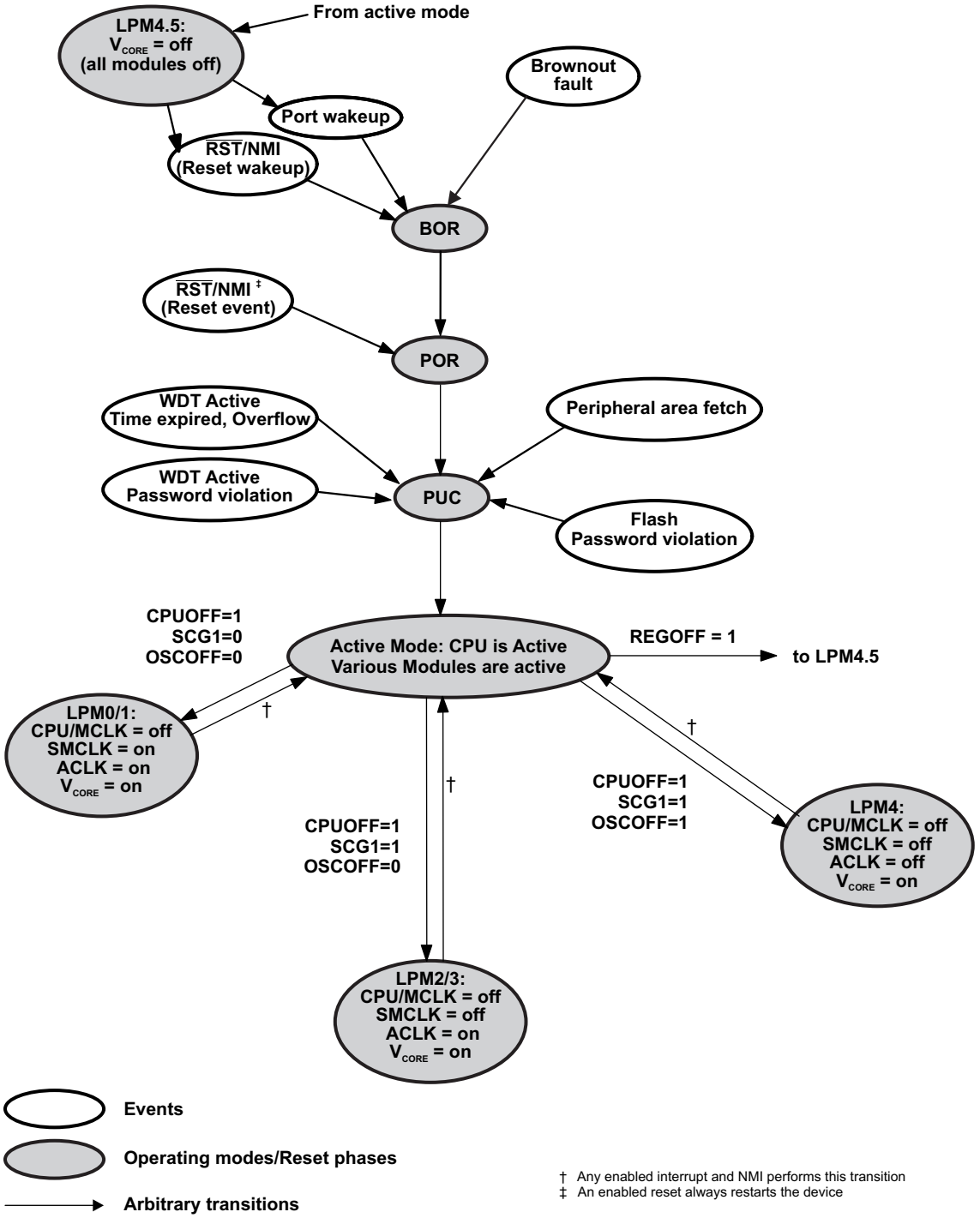


Figure 1-7. Operating Modes of MSP430i Devices

Table 1-2. Operating Modes of MSP430i Devices(1)

REGOFF	OSCOFF	SCG1	CPUOFF	Mode	CPU and Clocks Status
0	0	0	0	Active	CPU is active, DCO is enabled, all enabled clocks are active.
0	0	0	1	LPM0/LPM1	CPU is disabled, DCO is enabled, MCLK is disabled, SMCLK, ACLK are active.
0	0	1	1	LPM2/LPM3	CPU is disabled, DCO is enabled, MCLK, SMCLK are disabled, ACLK is active.
0	1	1	1	LPM4	CPU, DCO are disabled, MCLK, SMCLK, ACLK are inactive. RAM, Registers and I/O port pin states retained.
1	1	1	1	LPM4.5	Core voltage is turned off. CPU, DCO are disabled, MCLK, SMCLK, ACLK are inactive. No RAM/registers retention. I/O port pin states retained.

(1) CPUOFF, SCG1, and OSCOFF bits are in CPU Status Register. REGOFF bit is present in PMM LPM45CTL register.

1.3.1 Entering and Exiting Low-Power Modes

An enabled interrupt event wakes the MSP430 from any of the low-power operating modes. The program flow is:

- Enter interrupt service routine:
  - The PC and SR are stored on the stack
  - The CPUOFF, SCG1, and OSCOFF bits are automatically reset
- Options for returning from the interrupt service routine:
  - The original SR is popped from the stack, restoring the previous operating mode.
  - The SR bits stored on the stack can be modified within the interrupt service routine returning to a different operating mode when the RETI instruction is executed.

```
; Enter LPM0 Example
  BIS    #GIE+CPUOFF,SR            ; Enter LPM0
;  ...                             ; Program stops here
;
; Exit LPM0 Interrupt Service Routine
  BIC    #CPUOFF,0(SP)             ; Exit LPM0 on RETI
  RETI

; Enter LPM3 Example
  BIS    #GIE+CPUOFF+SCG1,SR       ; Enter LPM3
;  ...                             ; Program stops here
;
; Exit LPM3 Interrupt Service Routine
  BIC    #CPUOFF+SCG1,0(SP)        ; Exit LPM3 on RETI
  RETI
```

1.4 Principles for Low-Power Applications

The guidelines for low power consumption at application level while using MSP430i devices are given below .

- Use interrupts to wake the processor and control program flow.
- Peripherals should be switched on only when needed.
- Use low-power integrated peripheral modules in place of software driven functions. For example Timer_A can automatically generate PWM and capture external timing, with no CPU resources.
- Calculated branching and fast table look-ups should be used in place of flag polling and long software calculations.
- Avoid frequent subroutine and function calls due to overhead.
- For longer software routines, single-cycle CPU registers should be used.

1.5 Connection of Unused Pins

The correct termination of all unused pins is listed in Table 1-3.

Table 1-3. Connection of Unused Pins(1)

PIN	POTENTIAL	COMMENT
AVCC	DVCC	
AVSS	DVSS	
VREF	Open	
ROSC	AVSS	Connect ROSC pin to AVSS when DCO is used in internal resistor mode
Px.0 to Px.7	Open	Switched to port function, output direction
Ax.0+ and Ax.0-	AVSS	Short unused analog input pairs and connect them to analog ground
$\overline{\text{RST}}$/NMI	DVCC or VCC	47-kΩ pullup with 10 nF (or 2.2 nF(2)) pulldown
TEST	Open	This pin always has an internal pulldown enabled
TDO TDI TMS TCK	Open	The JTAG pins are shared with general-purpose I/O function. If not being used, these should be switched to port function, output direction. When used as JTAG pins, these pins should remain open.

(1) Any unused pin with a secondary function that is shared with general-purpose I/O should follow the Px.0 to Px.7 unused pin connection guidelines.

(2) The pulldown capacitor should not exceed 2.2 nF when using devices with Spy-Bi-Wire interface in Spy-Bi-Wire mode or in 4-wire JTAG mode with TI tools like FET interfaces or GANG programmers.

CPU

This chapter describes the MSP430 CPU, addressing modes, and instruction set.

Topic **Page**

2.1 CPU Introduction

The CPU incorporates features specifically designed for modern programming techniques such as calculated branching, table processing, and the use of high-level languages such as C. The CPU can address the complete address range without paging.

The CPU features include:

- RISC architecture with 27 instructions and 7 addressing modes.
- Orthogonal architecture with every instruction usable with every addressing mode.
- Full register access including program counter, status registers, and stack pointer.
- Single-cycle register operations.
- Large 16-bit register file reduces fetches to memory.
- 16-bit address bus allows direct access and branching throughout entire memory range.
- 16-bit data bus allows direct manipulation of word-wide arguments.
- Constant generator provides six most used immediate values and reduces code size.
- Direct memory-to-memory transfers without intermediate register holding.
- Word and byte addressing and instruction formats.

The block diagram of the CPU is shown in Figure 2-1.

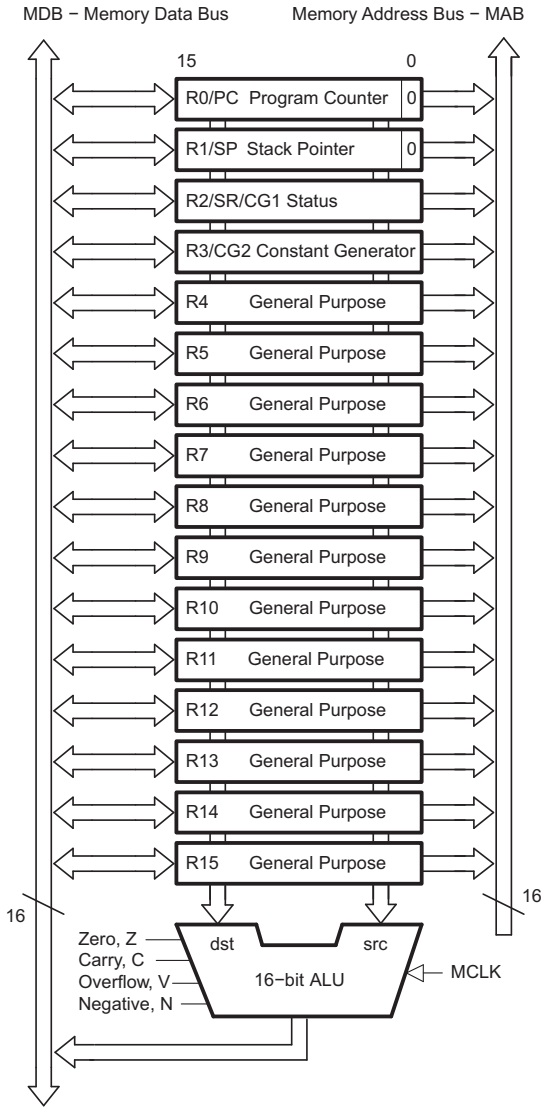


Figure 2-1. CPU Block Diagram

2.2 CPU Registers

The CPU incorporates sixteen 16-bit registers. R0, R1, R2, and R3 have dedicated functions. R4 to R15 are working registers for general use.

2.2.1 Program Counter (PC)

The 16-bit program counter (PC/R0) points to the next instruction to be executed. Each instruction uses an even number of bytes (two, four, or six), and the PC is incremented accordingly. Instruction accesses in the 64-KB address space are performed on word boundaries, and the PC is aligned to even addresses. Figure 2-2 shows the program counter.

Figure 2-2. Program Counter

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Program Counter Bits 15 to 1															0

The PC can be addressed with all instructions and addressing modes. A few examples:

```
MOV  #LABEL,PC    ; Branch to address LABEL
MOV  LABEL,PC     ; Branch to address contained in LABEL
MOV  @R14,PC      ; Branch indirect to address in R14
```

2.2.2 Stack Pointer (SP)

The stack pointer (SP/R1) is used by the CPU to store the return addresses of subroutine calls and interrupts. It uses a predecrement, postincrement scheme. In addition, the SP can be used by software with all instructions and addressing modes. Figure 2-3 shows the SP. The SP is initialized into RAM by the user, and is aligned to even addresses.

Figure 2-4 shows stack usage.

Figure 2-3. Stack Counter

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Stack Pointer Bits 15 to 1															0

```
MOV   2(SP),R6    ; Item I2 -> R6
MOV   R7,0(SP)    ; Overwrite TOS with R7
PUSH  #0123h      ; Put 0123h onto TOS
POP   R8          ; R8 = 0123h
```

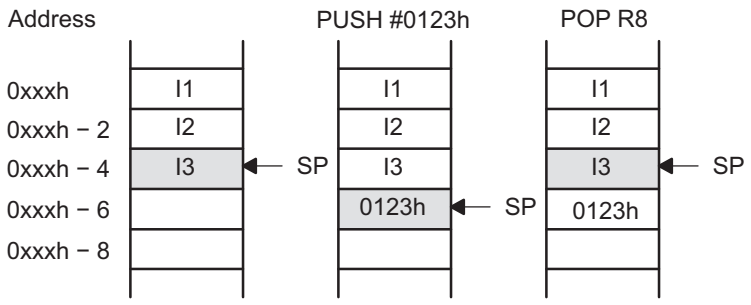


Figure 2-4. Stack Usage

The special cases of using the SP as an argument to the PUSH and POP instructions are described and shown in Figure 2-5.

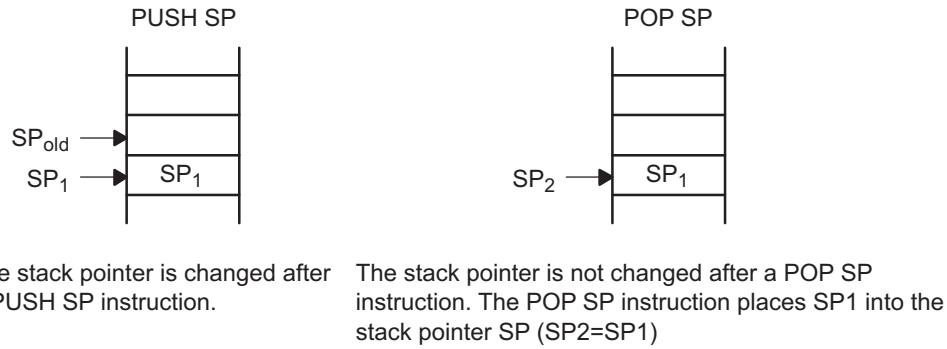


Figure 2-5. PUSH SP - POP SP Sequence

2.2.3 Status Register (SR)

The status register (SR/R2), used as a source or destination register, can be used in the register mode only addressed with word instructions. The remaining combinations of addressing modes are used to support the constant generator. Figure 2-6 shows the SR bits.

Figure 2-6. Status Register Bits

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							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

Table 2-1 describes the status register bits.

Table 2-1. Description of Status Register Bits

Bit	Description
V	Overflow bit. This bit is set when the result of an arithmetic operation overflows the signed-variable range. `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	System clock generator 1. When set, turns off the SMCLK.
SCG0	This bit is redundant and programming this bit has no functional effect.
OSCOFF	Oscillator Off. When set, turns off the ACLK.
CPUOFF	CPU off. When set, turns off the CPU.
GIE	General interrupt enable. When set, enables maskable interrupts. When reset, all maskable interrupts are disabled.
N	Negative bit. Set when the result of a byte or word operation is negative and cleared when the result is not negative. Word operation: N is set to the value of bit 15 of the result. Byte operation: N is set to the value of bit 7 of the result.
Z	Zero bit. Set when the result of a byte or word operation is 0 and cleared when the result is not 0.
C	Carry bit. Set when the result of a byte or word operation produced a carry and cleared when no carry occurred.

2.2.4 Constant Generator Registers CG1 and CG2

Six commonly-used constants are generated with the constant generator registers R2 and R3, without requiring an additional 16-bit word of program code. The constants are selected with the source-register addressing modes (As), as described in Table 2-2.

Table 2-2. Values of Constant Generators CG1, CG2

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

The constant generator advantages are:

- No special instructions required
- No additional code word for the six constants
- No code memory access required to retrieve the constant

The assembler uses the constant generator automatically if one of the six constants is used as an immediate source operand. Registers R2 and R3, used in the constant mode, cannot be addressed explicitly; they act as source-only registers.

2.2.4.1 Constant Generator - Expanded Instruction Set

The RISC instruction set of the MSP430 has only 27 instructions. However, the constant generator allows the MSP430 assembler to support 24 additional, emulated instructions. For example, the single-operand instruction

```
CLR   dst
```

is emulated by the double-operand instruction with the same length:

```
MOV   R3,dst
```

where the #0 is replaced by the assembler, and R3 is used with As=00.

```
INC   dst
```

is replaced by:

```
ADD   0(R3),dst
```

2.2.5 General-Purpose Registers R4 to R15

The twelve registers, R4-R15, are general-purpose registers. All of these registers can be used as data registers, address pointers, or index values and can be accessed with byte or word instructions as shown in Figure 2-7.

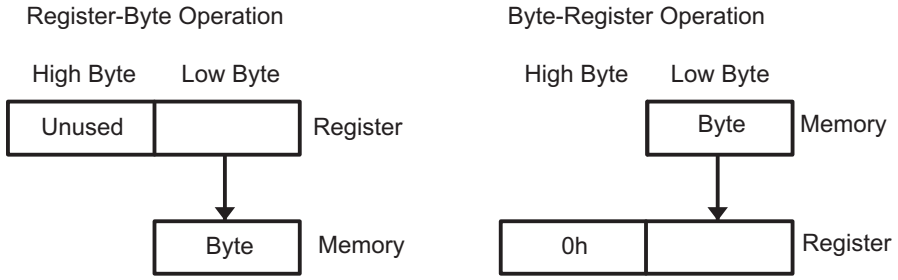


Figure 2-7. Register-Byte/Byte-Register Operations

Example Register-Byte Operation	Example Byte-Register Operation
R5 = 0A28Fh	R5 = 01202h
R6 = 0203h	R6 = 0223h
Mem(0203h) = 012h	Mem(0223h) = 05Fh
`ADD.B R5,0(R6)`	`ADD.B @R6,R5`
08Fh + 012h 0A1h	05Fh + 002h 00061h
Mem (0203h) = 0A1h	R5 = 00061h
C = 0, Z = 0, N = 1	C = 0, Z = 0, N = 0
(Low byte of register) + (Addressed byte) ->(Addressed byte)	(Addressed byte) + (Low byte of register) ->(Low byte of register, zero to High byte)

2.3 Addressing Modes

Seven addressing modes for the source operand and four addressing modes for the destination operand can address the complete address space with no exceptions. The bit numbers in Table 2-3 describe the contents of the As (source) and Ad (destination) mode bits.

Table 2-3. Source/Destination Operand Addressing Modes

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. X is stored in the next word. Indexed mode X(SR) is used.
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 by 1 for .B instructions and by 2 for .W instructions.
11/-	Immediate mode	#N	The word following the instruction contains the immediate constant N. Indirect autoincrement mode @PC+ is used.

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

NOTE: Use of Labels *EDE, TONI, TOM, and LEO*

Throughout MSP430 documentation EDE, TONI, TOM, and LEO are used as generic labels. They are only labels. They have no special meaning.

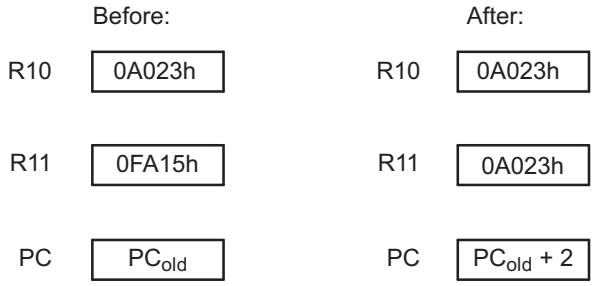
2.3.1 Register Mode

The register mode is described in Table 2-4.

Table 2-4. Register Mode Description

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

Length: One or two words
Operation: Move the content of R10 to R11. R10 is not affected.
Comment: Valid for source and destination
Example: `MOV R10,R11`



NOTE: Data in Registers

The data in the register can be accessed using word or byte instructions. If byte instructions are used, the high byte is always 0 in the result. The status bits are handled according to the result of the byte instructions.

2.3.2 Indexed Mode

The indexed mode is described in Table 2-5.

Table 2-5. Indexed Mode Description

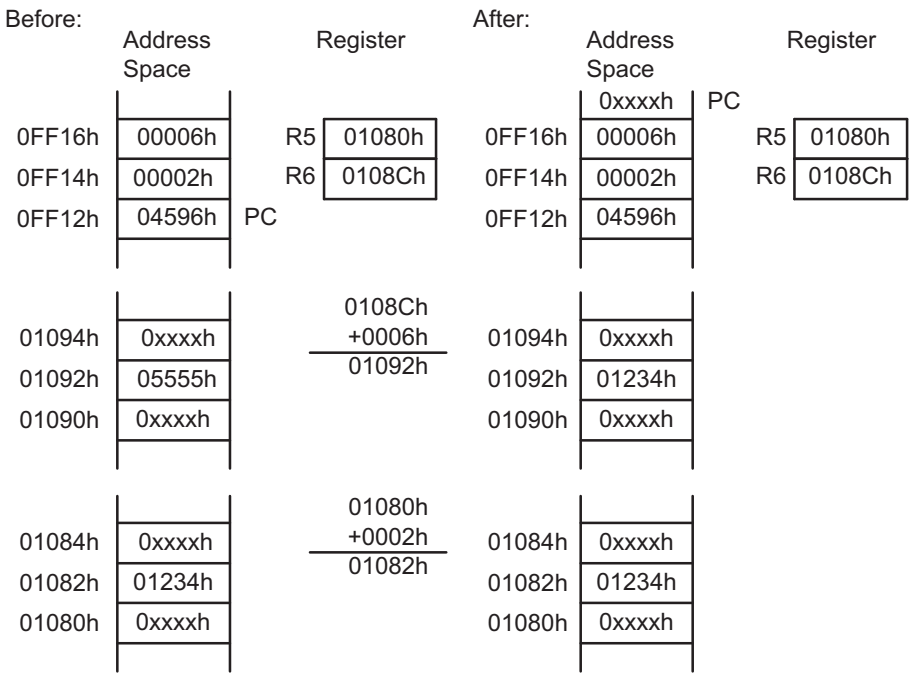
Assembler Code	Content of ROM
MOV 2(R5),6(R6)	MOV X(R5),Y(R6) X = 2 Y = 6

Length: Two or three 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. In indexed mode, the program counter is incremented automatically so that program execution continues with the next instruction.

Comment: Valid for source and destination

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



2.3.3 Symbolic Mode

The symbolic mode is described in Table 2-6.

Table 2-6. Symbolic Mode Description

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

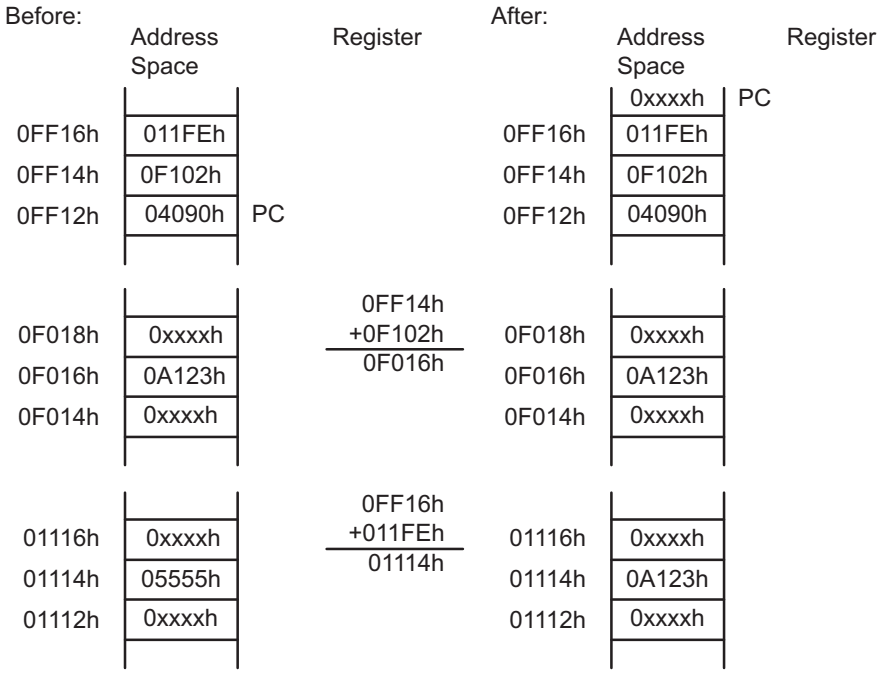
Length: Two or three words

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 between the PC and the source or destination addresses. The assembler computes and inserts offsets X and Y automatically. With symbolic mode, the program counter (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
```



2.3.4 Absolute Mode

The absolute mode is described in Table 2-7.

Table 2-7. Absolute Mode Description

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

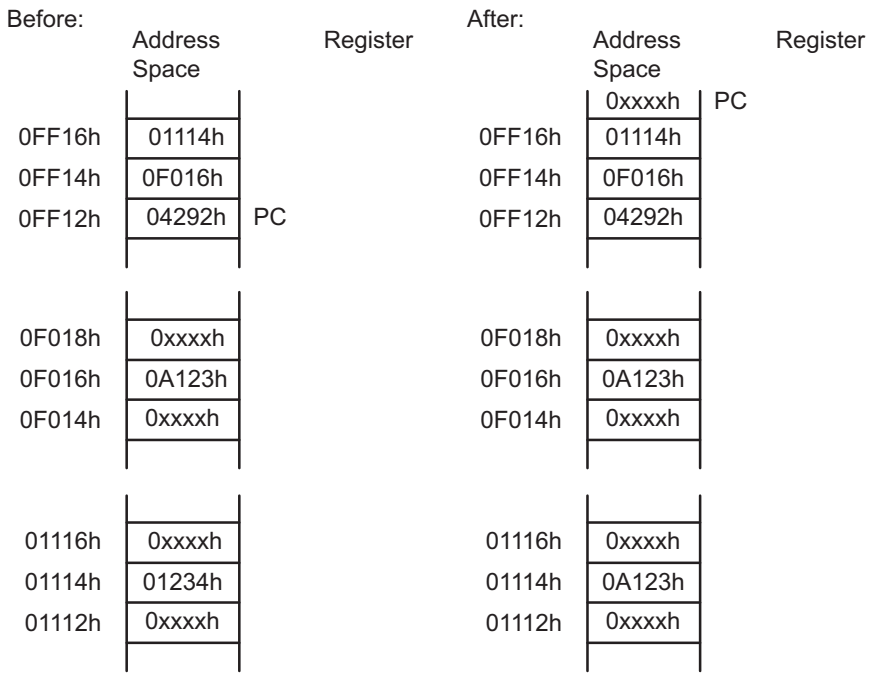
Length: Two or three words

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 and 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
```



This address mode is mainly for hardware peripheral modules that are located at an absolute, fixed address. These are addressed with absolute mode to ensure software transportability (for example, position-independent code).

2.3.5 Indirect Register Mode

The indirect register mode is described in Table 2-8.

Table 2-8. Indirect Mode Description

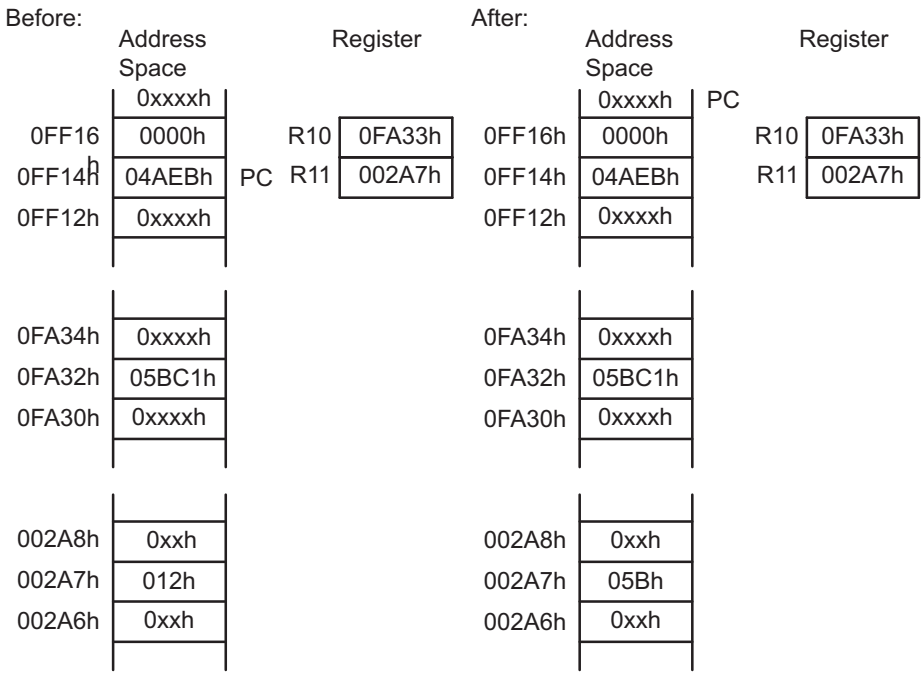
Assembler Code	Content of ROM
`MOV @R10,0(R11)`	`MOV @R10,0(R11)`

Length: One or two words

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. The substitute for destination operand is 0(Rd).

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



2.3.6 Indirect Autoincrement Mode

The indirect autoincrement mode is described in Table 2-9.

Table 2-9. Indirect Autoincrement Mode Description

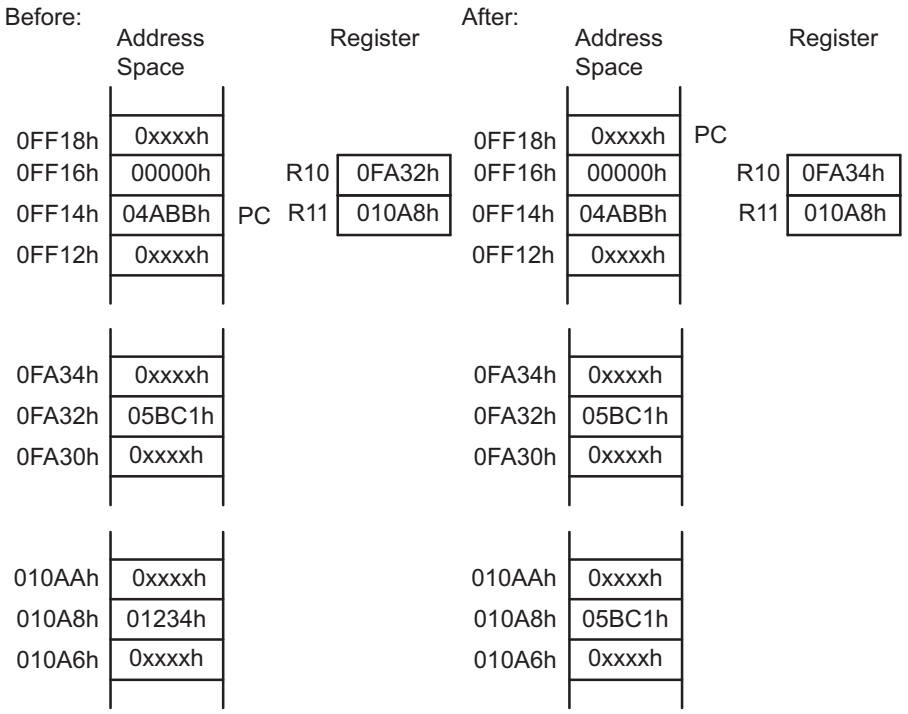
Assembler Code	Content of ROM
`MOV @R10+,0(R11)`	`MOV @R10+,0(R11)`

Length: One or two words

Operation: Move the contents of the source address (contents of R10) to the destination address (contents of R11). Register R10 is incremented by 1 for a byte operation, or 2 for a word operation after the fetch; it points to the next address without any overhead. This is useful for table processing.

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

Example: `MOV @R10+,0(R11)`



The auto-incrementing of the register contents occurs after the operand is fetched. This is shown in Figure 2-8.

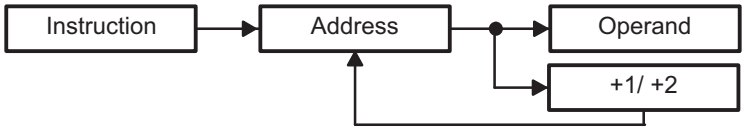


Figure 2-8. Operand Fetch Operation

2.3.7 Immediate Mode

The immediate mode is described in Table 2-10.

Table 2-10. Immediate Mode Description

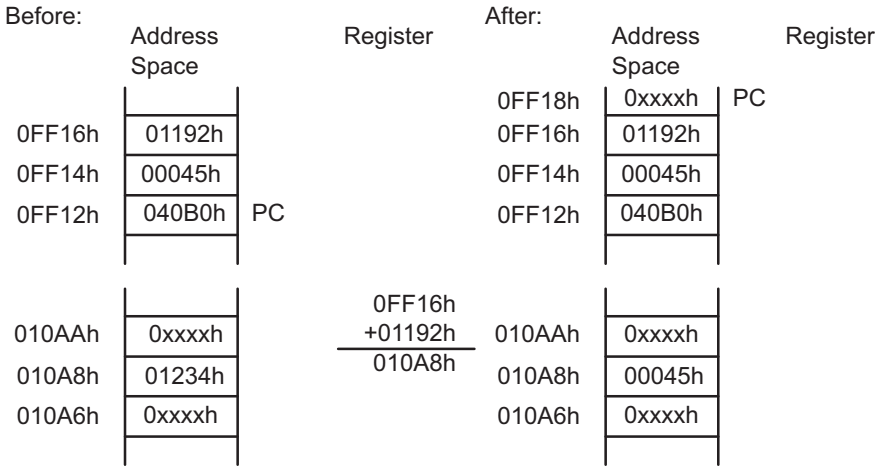
Assembler Code	Content of ROM
MOV #45h,TONI	MOV @PC+,X(PC) 45 X = TONI – PC

Length: Two or three words
It is one word less if a constant of CG1 or CG2 can be used.

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

Comment: Valid only for a source operand.

Example: `MOV #45h,TONI`



2.4 Instruction Set

The complete MSP430 instruction set consists of 27 core instructions and 24 emulated instructions. The core instructions are instructions that have unique op-codes decoded by the CPU. The emulated instructions are instructions that make code easier to write and read, but do not have op-codes themselves, instead they are replaced automatically by the assembler with an equivalent core instruction. There is no code or performance penalty for using emulated instruction.

There are three core-instruction formats:

- Dual-operand
- Single-operand
- Jump

All single-operand and dual-operand instructions can be byte or word instructions by using .B or .W extensions. Byte instructions are used to access byte data or byte peripherals. Word instructions are used to access word data or word peripherals. If no extension is used, the instruction is a word instruction.

The source and destination of an instruction are defined by the following fields:

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 working register used for the source (src)
Ad	The addressing bits responsible for the addressing mode used for the destination (dst)
D-reg	The working register used for the destination (dst)
B/W	Byte or word operation: 0: word operation 1: byte operation

NOTE: Destination Address

Destination addresses are valid anywhere in the memory map. However, when using an instruction that modifies the contents of the destination, the user must ensure the destination address is writable. For example, a masked-ROM location would be a valid destination address, but the contents are not modifiable, so the results of the instruction would be lost.

2.4.1 Double-Operand (Format I) Instructions

Figure 2-9 illustrates the double-operand instruction format.

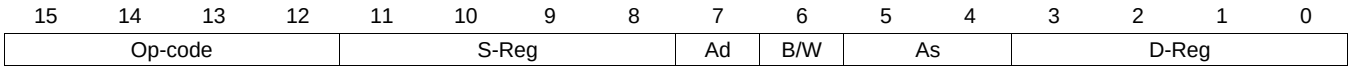


Figure 2-9. Double Operand Instruction Format

Table 2-11 lists and describes the double operand instructions.

Table 2-11. Double Operand Instructions

Mnemonic	S-Reg, D-Reg	Operation	Status Bits			
			V	N	Z	C
MOV(.B)	src,dst	src → dst	-	-	-	-
ADD(.B)	src,dst	src + dst → dst	*	*	*	*
ADDC(.B)	src,dst	src + dst + C → dst	*	*	*	*
SUB(.B)	src,dst	dst + .not.src + 1 → dst	*	*	*	*
SUBC(.B)	src,dst	dst + .not.src + C → dst	*	*	*	*
CMP(.B)	src,dst	dst - src	*	*	*	*
DADD(.B)	src,dst	src + dst + C → dst (decimally)	*	*	*	*
BIT(.B)	src,dst	src .and. dst	0	*	*	*
BIC(.B)	src,dst	not.src .and. dst → dst	-	-	-	-
BIS(.B)	src,dst	src .or. dst → dst	-	-	-	-
XOR(.B)	src,dst	src .xor. dst → dst	*	*	*	*
AND(.B)	src,dst	src .and. dst → dst	0	*	*	*

* The status bit is affected
– The status bit is not affected
0 The status bit is cleared
1 The status bit is set

NOTE: Instructions CMP and SUB

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

2.4.2 Single-Operand (Format II) Instructions

Figure 2-10 illustrates the single-operand instruction format.

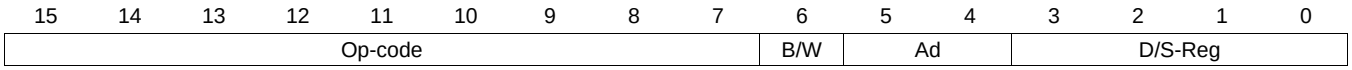


Figure 2-10. Single Operand Instruction Format

Table 2-12 lists and describes the single operand instructions.

Table 2-12. Single Operand Instructions

Mnemonic	S-Reg, D-Reg	Operation	Status Bits			
			V	N	Z	C
RRC(.B)	dst	C → MSB →.....LSB → C	*	*	*	*
RRA(.B)	dst	MSB → MSB →....LSB → C	0	*	*	*
PUSH(.B)	src	SP – 2 → SP, src → @SP	-	-	-	-
SWPB	dst	Swap bytes	-	-	-	-
CALL	dst	SP – 2 → SP, PC+2 → @SP dst → PC	-	-	-	-
RETI		TOS → SR, SP + 2 → SP TOS → PC,SP + 2 → SP	*	*	*	*
SXT	dst	Bit 7 → Bit 8.....Bit 15	0	*	*	*

*	The status bit is affected
–	The status bit is not affected
0	The status bit is cleared
1	The status bit is set

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 word that follows contains the address information.

2.4.3 Jumps

Figure 2-11 shows the conditional-jump instruction format.

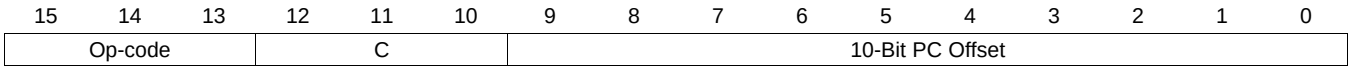


Figure 2-11. Jump Instruction Format

Table 2-13 lists and describes the jump instructions

Table 2-13. Jump Instructions

Mnemonic	S-Reg, D-Reg	Operation
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

Conditional jumps support program branching relative to the PC and do not affect the status bits. The possible jump range is from –511 to +512 words relative to the PC value at the jump instruction. The 10-bit program-counter offset is treated as a signed 10-bit value that is doubled and added to the program counter:

$PC_{new} = PC_{old} + 2 + PC_{offset} \times 2$

2.4.4 Instruction Cycles and Lengths

The number of CPU clock cycles required for an instruction depends on the instruction format and the addressing modes used - not the instruction itself. The number of clock cycles refers to the MCLK.

2.4.4.1 Interrupt and Reset Cycles

Table 2-14 lists the CPU cycles for interrupt overhead and reset.

Table 2-14. Interrupt and Reset Cycles

Action	No. of Cycles	Length of Instruction
Return from interrupt (RETI)	5	1
Interrupt accepted	6	-
WDT reset	4	-
Reset ($\overline{RST}$/NMI)	4	-

2.4.4.2 Format-II (Single Operand) Instruction Cycles and Lengths

Table 2-15 lists the length and CPU cycles for all addressing modes of format-II instructions.

Table 2-15. Format-II Instruction Cycles and Lengths

	No. of Cycles				
Addressing Mode	RRA, RRC SWPB, SXT	PUSH	CALL	Length of Instruction	Example
Rn	1	3	4	1	`SWPB R5`
@Rn	3	4	4	1	`RRC @R9`
@Rn+	3	5	5	1	`SWPB @R10+`
#N	(See note)	4	5	2	`CALL #0F000h`
X(Rn)	4	5	5	2	`CALL 2(R7)`
EDE	4	5	5	2	`PUSH EDE`
&EDE	4	5	5	2	`SXT &EDE`

NOTE: Instruction Format II Immediate Mode

Do not use instruction RRA, RRC, SWPB, and SXT with the immediate mode in the destination field. Use of these in the immediate mode results in an unpredictable program operation.

2.4.4.3 Format-III (Jump) Instruction Cycles and Lengths

All jump instructions require one code word, and take two CPU cycles to execute, regardless of whether the jump is taken or not.

2.4.4.4 Format-I (Double Operand) Instruction Cycles and Lengths

Table 2-16 lists the length and CPU cycles for all addressing modes of format-I instructions.

Table 2-16. Format 1 Instruction Cycles and Lengths

Addressing Mode			Length of		
Src	Dst	No. of Cycles	Instruction		Example
Rn	Rm	1	1	MOV	R5,R8
	PC	2	1	BR	R9
	x(Rm)	4	2	ADD	R5,4(R6)
	EDE	4	2	XOR	R8,EDE
	&EDE	4	2	MOV	R5,&EDE
@Rn	Rm	2	1	AND	@R4,R5
	PC	2	1	BR	@R8
	x(Rm)	5	2	XOR	@R5,8(R6)
	EDE	5	2	MOV	@R5,EDE
	&EDE	5	2	XOR	@R5,&EDE
@Rn+	Rm	2	1	ADD	@R5+,R6
	PC	3	1	BR	@R9+
	x(Rm)	5	2	XOR	@R5,8(R6)
	EDE	5	2	MOV	@R9+,EDE
	&EDE	5	2	MOV	@R9+,&EDE
#N	Rm	2	2	MOV	#20,R9
	PC	3	2	BR	#2AEh
	x(Rm)	5	3	MOV	#0300h,0(SP)
	EDE	5	3	ADD	#33,EDE
	&EDE	5	3	ADD	#33,&EDE
x(Rn)	Rm	3	2	MOV	2(R5),R7
	PC	3	2	BR	2(R6)
	TONI	6	3	MOV	4(R7),TONI
	x(Rm)	6	3	ADD	4(R4),6(R9)
	&TONI	6	3	MOV	2(R4),&TONI
EDE	Rm	3	2	AND	EDE,R6
	PC	3	2	BR	EDE
	TONI	6	3	CMP	EDE,TONI
	x(Rm)	6	3	MOV	EDE,0(SP)
	&TONI	6	3	MOV	EDE,&TONI
&EDE	Rm	3	2	MOV	&EDE,R8
	PC	3	2	BRA	&EDE
	TONI	6	3	MOV	&EDE,TONI
	x(Rm)	6	3	MOV	&EDE,0(SP)
	&TONI	6	3	MOV	&EDE,&TONI

2.4.5 Instruction Set Description

The instruction map is shown in Figure 2-12 and the complete instruction set is summarized in Table 2-17.

	000	040	080	0C0	100	140	180	1C0	200	240	280	2C0	300	340	380	3C0
0xxx																
4xxx																
8xxx																
Cxxx																
1xxx	RRC	RRC.B	SWPB		RRA	RRA.B	SXT		PUSH	PUSH.B	CALL		RETI			
14xx																
18xx																
1Cxx																
20xx	JNE/JNZ															
24xx	JEQ/JZ															
28xx	JNC															
2Cxx	JC															
30xx	JN															
34xx	JGE															
38xx	JL															
3Cxx	JMP															
4xxx	MOV, MOV.B															
5xxx	ADD, ADD.B															
6xxx	ADDC, ADDC.B															
7xxx	SUBC, SUBC.B															
8xxx	SUB, SUB.B															
9xxx	CMP, CMP.B															
Axxx	DADD, DADD.B															
Bxxx	BIT, BIT.B															
Cxxx	BIC, BIC.B															
Dxxx	BIS, BIS.B															
Exxx	XOR, XOR.B															
Fxxx	AND, AND.B															

Figure 2-12. Core Instruction Map

Table 2-17. MSP430 Instruction Set

Mnemonic		Description		V	N	Z	C
ADC(.B) (1)	dst	Add C to destination	dst + C → dst	*	*	*	*
ADD(.B)	src,dst	Add source to destination	src + dst → dst	*	*	*	*
ADDC(.B)	src,dst	Add source and C to destination	src + dst + C → dst	*	*	*	*
AND(.B)	src,dst	AND source and destination	src .and. dst → dst	0	*	*	*
BIC(.B)	src,dst	Clear bits in destination	not.src .and. dst → dst	-	-	-	-
BIS(.B)	src,dst	Set bits in destination	src .or. dst → dst	-	-	-	-
BIT(.B)	src,dst	Test bits in destination	src .and. dst	0	*	*	*
BR (1)	dst	Branch to destination	dst → PC	-	-	-	-
CALL	dst	Call destination	PC+2 → stack, dst → PC	-	-	-	-
CLR(.B) (1)	dst	Clear destination	0 → dst	-	-	-	-
CLRC (1)		Clear C	0 → C	-	-	-	0
CLRN (1)		Clear N	0 → N	-	0	-	-
CLRZ (1)		Clear Z	0 → Z	-	-	0	-
CMP(.B)	src,dst	Compare source and destination	dst - src	*	*	*	*
DADC(.B) (1)	dst	Add C decimally to destination	dst + C → dst (decimally)	*	*	*	*
DADD(.B)	src,dst	Add source and C decimally to dst	src + dst + C → dst (decimally)	*	*	*	*
DEC(.B) (1)	dst	Decrement destination	dst - 1 → dst	*	*	*	*

(1) Emulated Instruction

Table 2-17. MSP430 Instruction Set (continued)

Mnemonic		Description		V	N	Z	C
DECD(.B) [1]	dst	Double-decrement destination	dst - 2 → dst	*	*	*	*
DINT [1]		Disable interrupts	0 → GIE	-	-	-	-
EINT [1]		Enable interrupts	1 → GIE	-	-	-	-
INC(.B) [1]	dst	Increment destination	dst +1 → dst	*	*	*	*
INCD(.B) [1]	dst	Double-increment destination	dst+2 → dst	*	*	*	*
INV(.B) [1]	dst	Invert destination	.not.dst → dst	*	*	*	*
JC/JHS	label	Jump if C set/Jump if higher or same		-	-	-	-
JEQ/JZ	label	Jump if equal/Jump if Z set		-	-	-	-
JGE	label	Jump if greater or equal		-	-	-	-
JL	label	Jump if less		-	-	-	-
JMP	label	Jump	PC + 2 × offset → PC	-	-	-	-
JN	label	Jump if N set		-	-	-	-
JNC/JLO	label	Jump if C not set/Jump if lower		-	-	-	-
JNE/JNZ	label	Jump if not equal/Jump if Z not set		-	-	-	-
MOV(.B)	src,dst	Move source to destination	src → dst	-	-	-	-
NOP [2]		No operation		-	-	-	-
POP(.B) [2]	dst	Pop item from stack to destination	@SP → dst, SP+2 → SP	-	-	-	-
PUSH(.B)	src	Push source onto stack	SP - 2 → SP, src → @SP	-	-	-	-
RET [2]		Return from subroutine	@SP → PC, SP + 2 → SP	-	-	-	-
RETI		Return from interrupt		*	*	*	*
RLA(.B) [2]	dst	Rotate left arithmetically		*	*	*	*
RLC(.B) [2]	dst	Rotate left through C		*	*	*	*
RRA(.B)	dst	Rotate right arithmetically		0	*	*	*
RRC(.B)	dst	Rotate right through C		*	*	*	*
SBC(.B) [2]	dst	Subtract not(C) from destination	dst + 0FFFFh + C → dst	*	*	*	*
SETC [2]		Set C	1 → C	-	-	-	1
SETN [2]		Set N	1 → N	-	1	-	-
SETZ [2]		Set Z	1 → Z	-	-	1	-
SUB(.B)	src,dst	Subtract source from destination	dst + .not.src + 1 → dst	*	*	*	*
SUBC(.B)	src,dst	Subtract source and not(C) from dst	dst + .not.src + C → dst	*	*	*	*
SWPB	dst	Swap bytes		-	-	-	-
SXT	dst	Extend sign		0	*	*	*
TST(.B) [2]	dst	Test destination	dst + 0FFFFh + 1	0	*	*	1
XOR(.B)	src,dst	Exclusive OR source and destination	src .xor. dst → dst	*	*	*	*

[2] Emulated Instruction

2.4.6 Instruction Set Details

2.4.6.1 ADC

***ADC[.W]**	Add carry to destination
***ADC.B**	Add carry to destination
Syntax	`ADC dst or ADC.W dst` `ADC.B dst`
Operation	dst + C → dst
Emulation	`ADDC #0,dst` `ADDC.B #0,dst`
Description	The carry bit (C) is added to the destination operand. The previous contents of the destination are lost.
Status Bit	N: Set if result is negative, reset if positive Z: Set if result is zero, reset otherwise C: Set if dst was incremented from 0FFFFh to 0000, reset otherwise Set if dst was incremented from 0FFh to 00, reset otherwise V: Set if an arithmetic overflow occurs, otherwise reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.

Example	The 16-bit counter pointed to by R13 is added to a 32-bit counter pointed to by R12.
	`ADD @R13,0(R12) ; Add LSDs`
	`ADC 2(R12) ; Add carry to MSD`

Example	The 8-bit counter pointed to by R13 is added to a 16-bit counter pointed to by R12.
	`ADD.B @R13,0(R12) ; Add LSDs`
	`ADC.B 1(R12) ; Add carry to MSD`

2.4.6.2 ADD

ADD[.W]	Add source to destination
ADD.B	Add source to destination
Syntax	`ADD src,dst or ADD.W src,dst` `ADD.B src,dst`
Operation	src + dst → dst
Description	The source operand is added to the destination operand. The source operand is not affected. The previous contents of the destination are lost.
Status Bits	N: Set if result is negative, reset if positive Z: Set if result is zero, reset otherwise C: Set if there is a carry from the result, cleared if not V:Set if an arithmetic overflow occurs, otherwise reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.

Example	R5 is increased by 10. The jump to TONI is performed on a carry.
	`ADD #10,R5`
	`JC TONI ; Carry occurred`
	`..... ; No carry`

Example	R5 is increased by 10. The jump to TONI is performed on a carry.
	`ADD.B #10,R5 ; Add 10 to Lowbyte of R5`
	`JC TONI ; Carry occurred, if (R5) >= 246 [0Ah+0F6h]`
	`..... ; No carry`

2.4.6.3 ADDC

ADDC[.W]	Add source and carry to destination
ADDC.B	Add source and carry to destination
Syntax	`ADDC src,dst  or  ADDC.W src,dst` `ADDC.B src,dst`
Operation	src + dst + C → dst
Description	The source operand and the carry bit (C) are added to the destination operand. The source operand is not affected. The previous contents of the destination are lost.
Status Bits	N: Set if result is negative, reset if positive Z: Set if result is zero, reset otherwise C: Set if there is a carry from the MSB of the result, reset otherwise V: Set if an arithmetic overflow occurs, otherwise reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The 32-bit counter pointed to by R13 is added to a 32-bit counter, eleven words (20/2 + 2/2) above the pointer in R13.

```
ADD      @R13+,20(R13)   ; ADD LSDs with no carry in
ADDC     @R13+,20(R13)   ; ADD MSDs with carry
...                      ; resulting from the LSDs
```

Example The 24-bit counter pointed to by R13 is added to a 24-bit counter, eleven words above the pointer in R13.

```
ADD.B    @R13+,10(R13)   ; ADD LSDs with no carry in
ADDC.B   @R13+,10(R13)   ; ADD medium Bits with carry
ADDC.B   @R13+,10(R13)   ; ADD MSDs with carry
...                      ; resulting from the LSDs
```

2.4.6.4 AND

AND[.W] Source AND destination

AND.B Source AND destination

Syntax

```
AND src,dst  or  AND.W src,dst
AND.B src,dst
```

Operation

src .AND. dst → dst

Description The source operand and the destination operand are logically ANDed. The result is placed into the destination.

Status Bits

N: Set if result MSB is set, reset if not set

Z: Set if result is zero, reset otherwise

C: Set if result is not zero, reset otherwise (= .NOT. Zero)

V: Reset

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

Example The bits set in R5 are used as a mask (#0AA55h) for the word addressed by TOM. If the result is zero, a branch is taken to label TONI.

```
MOV     #0AA55h,R5   ; Load mask into register R5
AND     R5,TOM       ; mask word addressed by TOM with R5
JZ      TONI         ;
......               ; Result is not zero
;
;
;   or
;
;
AND     #0AA55h,TOM
JZ      TONI
```

Example The bits of mask #0A5h are logically ANDed with the low byte TOM. If the result is zero, a branch is taken to label TONI.

```
AND.B   #0A5h,TOM     ; mask Lowbyte TOM with 0A5h
JZ      TONI          ;
......                ; Result is not zero
```

2.4.6.5 BIC

BIC[.W]	Clear bits in destination
BIC.B	Clear bits in destination
Syntax	`BIC src,dst  or  BIC.W src,dst` `BIC.B src,dst`
Operation	.NOT.src .AND. dst → dst
Description	The inverted source operand and the destination operand are logically ANDed. The result is placed into the destination. The source operand is not affected.
Status Bits	Status bits are not affected.
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The six MSBs of the RAM word LEO are cleared. `BIC    #0FC00h,LEO  ; Clear 6 MSBs in MEM(LEO)`
Example	The five MSBs of the RAM byte LEO are cleared. `BIC.B  #0F8h,LEO    ; Clear 5 MSBs in Ram location LEO`

2.4.6.6 BIS

BIS[.W]	Set bits in destination
BIS.B	Set bits in destination
Syntax	`BIS src,dst  or  BIS.W src,dst` `BIS.B src,dst`
Operation	src .OR. dst → dst
Description	The source operand and the destination operand are logically ORed. The result is placed into the destination. The source operand is not affected.
Status Bits	Status bits are not affected.
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The six LSBs of the RAM word TOM are set. `BIS    #003Fh,TOM   ; set the six LSBs in RAM location TOM`
Example	The three MSBs of RAM byte TOM are set. `BIS.B  #0E0h,TOM    ; set the 3 MSBs in RAM location TOM`

2.4.6.7 BIT

BIT[.W] Test bits in destination

BIT.B Test bits in destination

Syntax

```
BIT src,dst  or  BIT.W src,dst
```

Operation src .AND. dst

Description The source and destination operands are logically ANDed. The result affects only the status bits. The source and destination operands are not affected.

Status Bits

N: Set if MSB of result is set, reset otherwise

Z: Set if result is zero, reset otherwise

C: Set if result is not zero, reset otherwise (.NOT. Zero)

V: Reset

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

Example If bit 9 of R8 is set, a branch is taken to label TOM.

```
BIT   #0200h,R8   ; bit 9 of R8 set?
JNZ   TOM         ; Yes, branch to TOM
...               ; No, proceed
```

Example If bit 3 of R8 is set, a branch is taken to label TOM.

```
BIT.B   #8,R8
JC      TOM
```

Example A serial communication receive bit (RCV) is tested. Because the carry bit is equal to the state of the tested bit while using the BIT instruction to test a single bit, the carry bit is used by the subsequent instruction; the read information is shifted into register RECBUF.

```
;
; Serial communication with LSB is shifted first:
                      ; xxxx   xxxx   xxxx   xxxx
BIT.B   #RCV,RCCTL    ; Bit info into carry
RRC     RECBUF        ; Carry -> MSB of RECBUF
                      ; cxxx    xxxx
......                ; repeat previous two instructions
......                ; 8 times
                      ; cccc   cccc
                      ; ^         ^
                      ; MSB       LSB
; Serial communication with MSB shifted first:
BIT.B   #RCV,RCCTL    ; Bit info into carry
RLC.B   RECBUF        ; Carry -> LSB of RECBUF
                      ; xxxx   xxxc
......                ; repeat previous two instructions
......                ; 8 times
                      ; cccc   cccc
                      ; |
                      ; MSB       LSB
```

2.4.6.8 BR, BRANCH

***BR, BRANCH** Branch to ..... destination

Syntax `BR dst`

Operation dst → PC

Emulation `MOV dst,PC`

Description An unconditional branch is taken to an address anywhere in the 64K address space. All source addressing modes can be used. The branch instruction is a word instruction.

Status Bits Status bits are not affected.

Example Examples for all addressing modes are given.

```
BR  #EXEC  ; Branch to label EXEC or direct branch (e.g. #0A4h)
           ; Core instruction MOV @PC+,PC
BR  EXEC   ; Branch to the address contained in EXEC
           ; Core instruction MOV X(PC),PC
           ; Indirect address
BR  &EXEC  ; Branch to the address contained in absolute
           ; address EXEC
           ; Core instruction MOV X(0),PC
           ; Indirect address
BR  R5     ; Branch to the address contained in R5
           ; Core instruction MOV R5,PC
           ; Indirect R5
BR  @R5    ; Branch to the address contained in the word
           ; pointed to by R5.
           ; Core instruction MOV @R5+,PC
           ; Indirect, indirect R5
BR  @R5+   ; Branch to the address contained in the word pointed
           ; to by R5 and increment pointer in R5 afterwards.
           ; The next time--S/W flow uses R5 pointer--it can
           ; alter program execution due to access to
           ; next address in a table pointed to by R5
           ; Core instruction MOV @R5,PC
           ; Indirect, indirect R5 with autoincrement
BR  X(R5)  ; Branch to the address contained in the address
           ; pointed to by R5 + X (e.g. table with address
           ; starting at X). X can be an address or a label
           ; Core instruction MOV X(R5),PC
           ; Indirect, indirect R5 + X
```

2.4.6.9 CALL

CALL	Subroutine
Syntax	`CALL dst`
Operation	dst → tmp dst is evaluated and stored SP - 2 → SP PC → @SP PC updated to TOS tmp → PC dst saved to PC
Description	A subroutine call is made to an address anywhere in the 64K address space. All addressing modes can be used. The return address (the address of the following instruction) is stored on the stack. The call instruction is a word instruction.
Status Bits	Status bits are not affected.
Example	Examples for all addressing modes are given.

```
CALL  #EXEC  ; Call on label EXEC or immediate address (e.g. #0A4h)
             ; SP-2 -> SP, PC+2 -> @SP, @PC+ -> PC
CALL  EXEC   ; Call on the address contained in EXEC
             ; SP-2 -> SP, PC+2 ->SP, X(PC) -> PC
             ; Indirect address
CALL  &EXEC  ; Call on the address contained in absolute address
             ; EXEC
             ; SP-2 -> SP, PC+2 -> @SP, X(0) -> PC
             ; Indirect address
CALL  R5     ; Call on the address contained in R5
             ; SP-2 -> SP, PC+2 -> @SP, R5 -> PC
             ; Indirect R5
CALL  @R5    ; Call on the address contained in the word
             ; pointed to by R5
             ; SP-2 -> SP, PC+2 -> @SP, @R5 -> PC
             ; Indirect, indirect R5
CALL  @R5+   ; Call on the address contained in the word
             ; pointed to by R5 and increment pointer in R5.
             ; The next time S/W flow uses R5 pointer
             ; it can alter the program execution due to
             ; access to next address in a table pointed to by R5
             ; SP-2 -> SP, PC+2 -> @SP, @R5 -> PC
             ; Indirect, indirect R5 with autoincrement
CALL  X(R5)  ; Call on the address contained in the address pointed
             ; to by R5 + X (e.g. table with address starting at X)
             ; X can be an address or a label
             ; SP-2 -> SP, PC+2 -> @SP, X(R5) -> PC
             ; Indirect, indirect R5 + X
```

2.4.6.10 CLR

***CLR[.W]**	Clear destination
***CLR.B**	Clear destination
Syntax	`CLR dst  or  CLR.W dst` `CLR.B dst`
Operation	0 → dst
Emulation	`MOV #0,dst` `MOV.B #0,dst`
Description	The destination operand is cleared.
Status Bits	Status bits are not affected.
Example	RAM word TONI is cleared. `CLR    TONI   ; 0 -> TONI`
Example	Register R5 is cleared. `CLR    R5`
Example	RAM byte TONI is cleared. `CLR.B  TONI   ; 0 -> TONI`

2.4.6.11 CLRC

***CLRC**	Clear carry bit
Syntax	`CLRC`
Operation	0 → C
Emulation	`BIC #1,SR`
Description	The carry bit (C) is cleared. The clear carry instruction is a word instruction.
Status Bits	N: Not affected Z: Not affected C: Cleared V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The 16-bit decimal counter pointed to by R13 is added to a 32-bit counter pointed to by R12.

```
CLRC                ; C=0: defines start
DADD   @R13,0(R12)  ; add 16=bit counter to low word of 32=bit counter
DADC   2(R12)       ; add carry to high word of 32=bit counter
```

2.4.6.12 CLRN

***CLRN**	Clear negative bit
Syntax	`CLRN`
Operation	0 → N or (.NOT.src .AND. dst → dst)
Emulation	`BIC  #4,SR`
Description	The constant 04h is inverted (0FFFBh) and is logically ANDed with the destination operand. The result is placed into the destination. The clear negative bit instruction is a word instruction.
Status Bits	N: Reset to 0 Z: Not affected C: Not affected V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The Negative bit in the status register is cleared. This avoids special treatment with negative numbers of the subroutine called.

```
        CLRN
        CALL   SUBR
        ......
        ......
SUBR    JN     SUBRET   ; If input is negative: do nothing and return
        ......
        ......
        ......
SUBRET  RET
```

2.4.6.13 CLRZ

***CLRZ**	Clear zero bit
Syntax	`CLRZ`
Operation	0 → Z or (.NOT.src .AND. dst → dst)
Emulation	`BIC #2,SR`
Description	The constant 02h is inverted (0FFFDh) and logically ANDed with the destination operand. The result is placed into the destination. The clear zero bit instruction is a word instruction.
Status Bits	N: Not affected Z: Reset to 0 C: Not affected V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The zero bit in the status register is cleared. `CLRZ`

2.4.6.14 CMP

CMP[.W] Compare source and destination

CMP.B Compare source and destination

Syntax

```
CMP src,dst  or  CMP.W src,dst
CMP.B src,dst
```

Operation

dst + .NOT.src + 1

or

(dst - src)

Description The source operand is subtracted from the destination operand. This is accomplished by adding the 1s complement of the source operand plus 1. The two operands are not affected and the result is not stored; only the status bits are affected.

Status Bits

N: Set if result is negative, reset if positive (src ≥ dst)

Z: Set if result is zero, reset otherwise (src = dst)

C: Set if there is a carry from the MSB of the result, reset otherwise

V: Set if an arithmetic overflow occurs, otherwise reset

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

Example R5 and R6 are compared. If they are equal, the program continues at the label EQUAL.

```
CMP   R5,R6   ; R5 = R6?
JEQ   EQUAL   ; YES, JUMP
```

Example Two RAM blocks are compared. If they are not equal, the program branches to the label ERROR.

```
     MOV   #NUM,R5      ; number of words to be compared
     MOV   #BLOCK1,R6   ; BLOCK1 start address in R6
     MOV   #BLOCK2,R7   ; BLOCK2 start address in R7
L$1  CMP   @R6+,0(R7)   ; Are Words equal? R6 increments
     JNZ   ERROR        ; No, branch to ERROR
     INCD  R7           ; Increment R7 pointer
     DEC   R5           ; Are all words compared?
     JNZ   L$1          ; No, another compare
```

Example The RAM bytes addressed by EDE and TONI are compared. If they are equal, the program continues at the label EQUAL.

```
CMP.B   EDE,TONI   ; MEM(EDE) = MEM(TONI)?
JEQ     EQUAL      ; YES, JUMP
```

2.4.6.15 DADC

***DADC[.W]**	Add carry decimally to destination
***DADC.B**	Add carry decimally to destination
Syntax	`DADC dst or DADC.W src,dst` `DADC.B dst`
Operation	dst + C → dst (decimally)
Emulation	`DADD #0,dst` `DADD.B #0,dst`
Description	The carry bit (C) is added decimally to the destination.
Status Bits	N: Set if MSB is 1 Z: Set if dst is 0, reset otherwise C: Set if destination increments from 9999 to 0000, reset otherwise Set if destination increments from 99 to 00, reset otherwise V: Undefined
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The four-digit decimal number contained in R5 is added to an eight-digit decimal number pointed to by R8.

```
CLRC                ; Reset carry
                    ; next instruction's start condition is defined
DADD     R5,0(R8)   ; Add LSDs + C
DADC     2(R8)      ; Add carry to MSD
```

Example The two-digit decimal number contained in R5 is added to a four-digit decimal number pointed to by R8.

```
CLRC                ; Reset carry
                    ; next instruction's start condition is defined
DADD.B   R5,0(R8)   ; Add LSDs + C
DADC.B   1(R8)      ; Add carry to MSDs
```

2.4.6.16 DADD

DADD[.W]	Source and carry added decimally to destination
DADD.B	Source and carry added decimally to destination
Syntax	`DADD src,dst  or  DADD.W src,dst` `DADD.B src,dst`
Operation	src + dst + C → dst (decimally)
Description	The source operand and the destination operand are treated as four binary coded decimals (BCD) with positive signs. The source operand and the carry bit (C)are added decimally to the destination operand. The source operand is not affected. The previous contents of the destination are lost. The result is not defined for non-BCD numbers.
Status Bits	N: Set if the MSB is 1, reset otherwise Z: Set if result is zero, reset otherwise C: Set if the result is greater than 9999 Set if the result is greater than 99 V: Undefined
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The eight-digit BCD number contained in R5 and R6 is added decimally to an eight-digit BCD number contained in R3 and R4 (R6 and R4 contain the MSDs).

```
CLRC                ; clear carry
DADD    R5,R3       ; add LSDs
DADD    R6,R4       ; add MSDs with carry
JC      OVERFLOW    ; If carry occurs go to error handling routine
```

Example The two-digit decimal counter in the RAM byte CNT is incremented by one.

```
CLRC              ; clear carry
DADD.B  #1,CNT
```

or

```
SETC
DADD.B  #0,CNT    ; equivalent to DADC.B CNT
```

2.4.6.17 DEC

***DEC[.W]** Decrement destination

***DEC.B** Decrement destination

Syntax

```
DEC dst  or  DEC.W dst
DEC.B dst
```

Operation dst - 1 → dst

Emulation

```
SUB #1,dst
SUB.B #1,dst
```

Description The destination operand is decremented by one. The original contents are lost.

Status Bits

N: Set if result is negative, reset if positive

Z: Set if dst contained 1, reset otherwise

C: Reset if dst contained 0, set otherwise

V: Set if an arithmetic overflow occurs, otherwise reset.

Set if initial value of destination was 08000h, otherwise reset.

Set if initial value of destination was 080h, otherwise reset.

Mode Bits OSCOFF, CPUOFF,and GIE are not affected.

Example R10 is decremented by 1.

```
     DEC     R10    ; Decrement R10

; Move a block of 255 bytes from memory location starting with EDE to
memory location starting with
; TONI. Tables should not overlap: start of destination address TONI
must not be within the range EDE
; to EDE+0FEh

     MOV     #EDE,R6
     MOV     #255,R10
L$1  MOV.B   @R6+,TONI-EDE-1(R6)
     DEC     R10
     JNZ     L$1
```

Do not transfer tables using the routine above with the overlap shown in Figure 2-13.

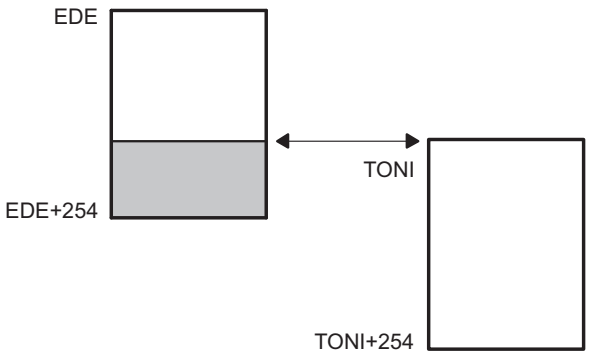


Figure 2-13. Decrement Overlap

2.4.6.18 DECD

***DECD[.W]** Double-decrement destination

***DECD.B** Double-decrement destination

Syntax

```
DECD dst  or  DECD.W dst
DECD.B dst
```

Operation dst - 2 → dst

Emulation

```
SUB #2,dst
```

Emulation

```
SUB.B #2,dst
```

Description The destination operand is decremented by two. The original contents are lost.

Status Bits

N: Set if result is negative, reset if positive

Z: Set if dst contained 2, reset otherwise

C: Reset if dst contained 0 or 1, set otherwise

V: Set if an arithmetic overflow occurs, otherwise reset.

Set if initial value of destination was 08001 or 08000h, otherwise reset.

Set if initial value of destination was 081 or 080h, otherwise reset.

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

Example R10 is decremented by 2.

```
      DECD   R10      ; Decrement R10 by two
; Move a block of 255 words from memory location starting with EDE to
; memory location starting with TONI
; Tables should not overlap: start of destination address TONI must not be
; within the range EDE to EDE+0FEh

      MOV    #EDE,R6
      MOV    #510,R10
L$1   MOV    @R6+,TONI-EDE-2(R6)
      DECD   R10
      JNZ    L$1
```

Example Memory at location LEO is decremented by two.

```
DECD.B   LEO   ; Decrement MEM(LEO)
```

Decrement status byte STATUS by two.

```
DECD.B   STATUS
```

2.4.6.19 DINT

***DINT** Disable (general) interrupts

Syntax

```
DINT
```

Operation

0 → GIE

or

(0FFF7h .AND. SR → SR / .NOT.src .AND. dst → dst)

Emulation

```
BIC #8,SR
```

Description All interrupts are disabled.

The constant 08h is inverted and logically ANDed with the status register (SR). The result is placed into the SR.

Status Bits Status bits are not affected.

Mode Bits GIE is reset. OSCOFF and CPUOFF are not affected.

Example The general interrupt enable (GIE) bit in the status register is cleared to allow a nondisrupted move of a 32-bit counter. This ensures that the counter is not modified during the move by any interrupt.

```
DINT               ; All interrupt events using the GIE bit are disabled
NOP
MOV   COUNTHI,R5   ; Copy counter
MOV   COUNTLO,R6
EINT               ; All interrupt events using the GIE bit are enabled
```

NOTE: Disable Interrupt

If any code sequence needs to be protected from interruption, the DINT should be executed at least one instruction before the beginning of the uninterruptible sequence, or should be followed by a NOP instruction.

2.4.6.20 EINT

***EINT** Enable (general) interrupts

Syntax `EINT`

Operation 1 → GIE

or

(0008h .OR. SR → SR / .src .OR. dst → dst)

Emulation `BIS #8,SR`

Description All interrupts are enabled.

The constant #08h and the status register SR are logically ORed. The result is placed into the SR.

Status Bits Status bits are not affected.

Mode Bits GIE is set. OSCOFF and CPUOFF are not affected.

Example The general interrupt enable (GIE) bit in the status register is set.

```
; Interrupt routine of ports P1.2 to P1.7
; P1IN is the address of the register where all port bits are read. P1IFG is
; the address of the register where all interrupt events are latched.

        PUSH.B   &P1IN
        BIC.B    @SP,&P1IFG   ; Reset only accepted flags
        EINT                  ; Preset port 1 interrupt flags stored on stack
                              ; other interrupts are allowed
        BIT      #Mask,@SP
        JEQ      MaskOK       ; Flags are present identically to mask: jump
        ......
MaskOK  BIC      #Mask,@SP
        ......
        INCD     SP           ; Housekeeping: inverse to PUSH instruction
                              ; at the start of interrupt subroutine. Corrects
                              ; the stack pointer.
        RETI
```

NOTE: Enable Interrupt

The instruction following the enable interrupt instruction (EINT) is always executed, even if an interrupt service request is pending when the interrupts are enable.

2.4.6.21 INC

***INC[.W]**	Increment destination
***INC.B**	Increment destination
Syntax	`INC dst  or  INC.W dst` `INC.B dst`
Operation	dst + 1 → dst
Emulation	`ADD #1,dst`
Description	The destination operand is incremented by one. The original contents are lost.
Status Bits	N: Set if result is negative, reset if positive Z: Set if dst contained 0FFFFh, reset otherwise Set if dst contained 0FFh, reset otherwise C: Set if dst contained 0FFFFh, reset otherwise Set if dst contained 0FFh, reset otherwise V: Set if dst contained 07FFFh, reset otherwise Set if dst contained 07Fh, reset otherwise
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.

Example	The status byte, STATUS, of a process is incremented. When it is equal to 11, a branch to OVFL is taken.
	`INC.B  STATUS`
	`CMP.B  #11,STATUS`
	`JEQ  OVFL`

2.4.6.22 INCD

***INCD[.W]**	Double-increment destination
***INCD.B**	Double-increment destination
Syntax	`INCD dst  or  INCD.W dst` `INCD.B dst`
Operation	dst + 2 → dst
Emulation	`ADD #2,dst` `ADD.B #2,dst`
Example	The destination operand is incremented by two. The original contents are lost.
Status Bits	N: Set if result is negative, reset if positive Z: Set if dst contained 0FFFEh, reset otherwise Set if dst contained 0FEh, reset otherwise C: Set if dst contained 0FFFEh or 0FFFFh, reset otherwise Set if dst contained 0FEh or 0FFh, reset otherwise V: Set if dst contained 07FFEh or 07FFFh, reset otherwise Set if dst contained 07Eh or 07Fh, reset otherwise
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.

Example The item on the top of the stack (TOS) is removed without using a register.

```
PUSH     R5     ; R5 is the result of a calculation, which is stored
                ; in the system stack
INCD     SP     ; Remove TOS by double-increment from stack
                ; Do not use INCD.B, SP is a word-aligned register
RET
```

Example The byte on the top of the stack is incremented by two.

```
INCD.B   0(SP)  ; Byte on TOS is increment by two
```

2.4.6.23 INV

***INV[.W]**	Invert destination
***INV.B**	Invert destination
Syntax	`INV dst` `INV.B dst`
Operation	.NOT.dst → dst
Emulation	`XOR #0FFFFh,dst` `XOR.B #0FFh,dst`
Description	The destination operand is inverted. The original contents are lost.
Status Bits	N: Set if result is negative, reset if positive Z: Set if dst contained 0FFFFh, reset otherwise Set if dst contained 0FFh, reset otherwise C: Set if result is not zero, reset otherwise (= .NOT. Zero) Set if result is not zero, reset otherwise (= .NOT. Zero) V: Set if initial destination operand was negative, otherwise reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.

Example Content of R5 is negated (twos complement).

```
MOV    #00AEh,R5  ;                      R5 = 000AEh
INV    R5         ; Invert R5,           R5 = 0FF51h
INC    R5         ; R5 is now negated,   R5 = 0FF52h
```

Example Content of memory byte LEO is negated.

```
MOV.B  #0AEh,LEO  ;                      MEM(LEO) = 0AEh
INV.B  LEO        ; Invert LEO,          MEM(LEO) = 051h
INC.B  LEO        ; MEM(LEO) is negated, MEM(LEO) = 052h
```

2.4.6.24 JC, JHS

JC Jump if carry set

JHS Jump if higher or same

Syntax

```
JC label
JHS label
```

Operation

If C = 1: PC + 2 offset → PC

If C = 0: execute following instruction

Description The status register carry bit (C) is tested. If it is set, the 10-bit signed offset contained in the instruction LSBs is added to the program counter. If C is reset, the next instruction following the jump is executed. JC (jump if carry/higher or same) is used for the comparison of unsigned numbers (0 to 65536).

Status Bits Status bits are not affected.

Example The P1IN.1 signal is used to define or control the program flow.

```
BIT.B   #02h,&P1IN   ; State of signal -> Carry
JC      PROGA        ; If carry=1 then execute program routine A
......               ; Carry=0, execute program here
```

Example R5 is compared to 15. If the content is higher or the same, branch to LABEL.

```
CMP     #15,R5
JHS     LABEL        ; Jump is taken if R5 >= 15
......               ; Continue here if R5 < 15
```

2.4.6.25 JEQ, JZ

JEQ, JZ	Jump if equal, jump if zero
Syntax	`JEQ label` `JZ label`
Operation	If Z = 1: PC + 2 offset → PC If Z = 0: execute following instruction
Description	The status register zero bit (Z) is tested. If it is set, the 10-bit signed offset contained in the instruction LSBs is added to the program counter. If Z is not set, the instruction following the jump is executed.
Status Bits	Status bits are not affected.

Example	Jump to address TONI if R7 contains zero.
	`TST R7 ; Test R7`
	`JZ TONI ; if zero: JUMP`

Example	Jump to address LEO if R6 is equal to the table contents.
	`CMP R6,Table(R5) ; Compare content of R6 with content of`
	`; MEM (table address + content of R5)`
	`JEQ LEO ; Jump if both data are equal`
	`..... ; No, data are not equal, continue here`

Example	Branch to LABEL if R5 is 0.
	`TST R5`
	`JZ LABEL`
	`.....`

2.4.6.26 JGE

JGE	Jump if greater or equal
Syntax	`JGE label`
Operation	If (N .XOR. V) = 0 then jump to label: PC + 2 P offset → PC If (N .XOR. V) = 1 then execute the following instruction
Description	The status register negative bit (N) and overflow bit (V) are tested. If both N and V are set or reset, the 10-bit signed offset contained in the instruction LSBs is added to the program counter. If only one is set, the instruction following the jump is executed. This allows comparison of signed integers.
Status Bits	Status bits are not affected.
Example	When the content of R6 is greater or equal to the memory pointed to by R7, the program continues at label EDE.

```
CMP   @R7,R6   ; R6 >= (R7)?, compare on signed numbers
JGE   EDE      ; Yes, R6 >= (R7)
......         ; No, proceed
......
......
```

2.4.6.27 JL

JL	Jump if less
Syntax	`JL label`
Operation	If (N .XOR. V) = 1 then jump to label: PC + 2 offset → PC If (N .XOR. V) = 0 then execute following instruction
Description	The status register negative bit (N) and overflow bit (V) are tested. If only one is set, the 10-bit signed offset contained in the instruction LSBs is added to the program counter. If both N and V are set or reset, the instruction following the jump is executed. This allows comparison of signed integers.
Status Bits	Status bits are not affected.
Example	When the content of R6 is less than the memory pointed to by R7, the program continues at label EDE.

```
CMP   @R7,R6   ; R6 < (R7)?,  compare on signed numbers
JL    EDE      ; Yes, R6 < (R7)
......         ; No, proceed
......
......
```

2.4.6.28 JMP

JMP	Jump unconditionally
Syntax	`JMP label`
Operation	PC + 2 × offset → PC
Description	The 10-bit signed offset contained in the instruction LSBs is added to the program counter.
Status Bits	Status bits are not affected.
Hint	This one-word instruction replaces the BRANCH instruction in the range of –511 to +512 words relative to the current program counter.

2.4.6.29 JN

JN	Jump if negative
Syntax	`JN label`
Operation	if N = 1: PC + 2 ×offset → PC if N = 0: execute following instruction
Description	The negative bit (N) of the status register is tested. If it is set, the 10-bit signed offset contained in the instruction LSBs is added to the program counter. If N is reset, the next instruction following the jump is executed.
Status Bits	Status bits are not affected.
Example	The result of a computation in R5 is to be subtracted from COUNT. If the result is negative, COUNT is to be cleared and the program continues execution in another path.

```
      SUB   R5,COUNT   ; COUNT - R5 -> COUNT
      JN    L$1        ; If negative continue with COUNT=0 at PC=L$1
      ......           ; Continue with COUNT>=0
      ......
      ......
      ......
L$1   CLR   COUNT
      ......
      ......
      ......
```

2.4.6.30 JNC, JLO

JNC Jump if carry not set

JLO Jump if lower

Syntax

```
JNC label
JLO label
```

Operation

if C = 0: PC + 2 offset → PC

if C = 1: execute following instruction

Description The status register carry bit (C) is tested. If it is reset, the 10-bit signed offset contained in the instruction LSBs is added to the program counter. If C is set, the next instruction following the jump is executed. JNC (jump if no carry/lower) is used for the comparison of unsigned numbers (0 to 65536).

Status Bits Status bits are not affected.

Example The result in R6 is added in BUFFER. If an overflow occurs, an error handling routine at address ERROR is used.

```
        ADD     R6,BUFFER   ; BUFFER + R6 -> BUFFER
        JNC     CONT        ; No carry, jump to CONT
ERROR   ......              ; Error handler start
        ......
        ......
        ......
CONT    ......              ; Continue with normal program flow
        ......
        ......
```

Example Branch to STL2 if byte STATUS contains 1 or 0.

```
        CMP.B   #2,STATUS
        JLO     STL 2       ; STATUS < 2
        ......              ; STATUS >= 2, continue here
```

2.4.6.31 JNE, JNZ

JNE	Jump if not equal
JNZ	Jump if not zero
Syntax	`JNE label` `JNZ label`
Operation	If Z = 0: PC + 2 a offset → PC If Z = 1: execute following instruction
Description	The status register zero bit (Z) is tested. If it is reset, the 10-bit signed offset contained in the instruction LSBs is added to the program counter. If Z is set, the next instruction following the jump is executed.
Status Bits	Status bits are not affected.
Example	Jump to address TONI if R7 and R8 have different contents.

```
CMP   R7,R8   ; COMPARE R7 WITH R8
JNE   TONI    ; if different: jump
......        ; if equal, continue
```

2.4.6.32 MOV

MOV[.W] Move source to destination

MOV.B Move source to destination

Syntax

```
MOV src,dst  or  MOV.W src,dst
MOV.B src,dst
```

Operation src → dst

Description The source operand is moved to the destination.

The source operand is not affected. The previous contents of the destination are lost.

Status Bits Status bits are not affected.

Mode Bits OSCOFF, CPUOFF,and GIE are not affected.

Example The contents of table EDE (word data) are copied to table TOM. The length of the tables must be 020h locations.

```
       MOV   #EDE,R10               ; Prepare pointer
       MOV   #020h,R9               ; Prepare counter
Loop   MOV   @R10+,TOM-EDE-2(R10)   ; Use pointer in R10 for both tables
       DEC   R9                     ; Decrement counter
       JNZ   Loop                   ; Counter not 0, continue copying
       ......                       ; Copying completed
       ......
       ......
```

Example The contents of table EDE (byte data) are copied to table TOM. The length of the tables should be 020h locations

```
       MOV     #EDE,R10             ; Prepare pointer
       MOV     #020h,R9             ; Prepare counter
Loop   MOV.B   @R10+,TOM-EDE-1(R10) ; Use pointer in R10 for
                                    ; both tables
       DEC   R9                     ; Decrement counter
       JNZ   Loop                   ; Counter not 0, continue
                                    ; copying
       ......                       ; Copying completed
       ......
       ......
```

2.4.6.33 NOP

***NOP**	No operation
Syntax	`NOP`
Operation	None
Emulation	`MOV #0, R3`
Description	No operation is performed. The instruction may be used for the elimination of instructions during the software check or for defined waiting times.
Status Bits	Status bits are not affected.

The NOP instruction is mainly used for two purposes:

- To fill one, two, or three memory words
- To adjust software timing

NOTE: Emulating No-Operation Instruction

Other instructions can emulate the NOP function while providing different numbers of instruction cycles and code words. Some examples are:

```
MOV  #0,R3         ; 1 cycle, 1 word
MOV  0(R4),0(R4)   ; 6 cycles, 3 words
MOV  @R4,0(R4)     ; 5 cycles, 2 words
BIC  #0,EDE(R4)    ; 4 cycles, 2 words
JMP  $+2           ; 2 cycles, 1 word
BIC  #0,R5         ; 1 cycle, 1 word
```

However, care should be taken when using these examples to prevent unintended results. For example, if `MOV 0(R4), 0(R4)` is used and the value in R4 is 120h, then a security violation occurs with the watchdog timer (address 120h), because the security key was not used.

2.4.6.34 POP

***POP[.W]** Pop word from stack to destination

***POP.B** Pop byte from stack to destination

Syntax

```
POP dst
POP.B dst
```

Operation

@SP → temp

SP + 2 → SP

temp → dst

Emulation

```
MOV @SP+,dst or MOV.W @SP+,dst
MOV.B @SP+,dst
```

Description The stack location pointed to by the stack pointer (TOS) is moved to the destination. The stack pointer is incremented by two afterwards.

Status Bits Status bits are not affected.

Example The contents of R7 and the status register are restored from the stack.

```
POP     R7      ; Restore R7
POP     SR      ; Restore status register
```

Example The contents of RAM byte LEO is restored from the stack.

```
POP.B   LEO     ; The low byte of the stack is moved to LEO.
```

Example The contents of R7 is restored from the stack.

```
POP.B   R7      ; The low byte of the stack is moved to R7,
                ; the high byte of R7 is 00h
```

Example The contents of the memory pointed to by R7 and the status register are restored from the stack.

```
POP.B   0(R7)   ; The low byte of the stack is moved to the
                ; the byte which is pointed to by R7
                ; Example:  R7 = 203h
                ;           Mem(R7) = low byte of system stack
                ; Example:  R7 = 20Ah
                ;           Mem(R7) = low byte of system stack
POP     SR      ; Last word on stack moved to the SR
```

NOTE: The System Stack Pointer

The system stack pinter (SP) is always incremented by two, independent of the byte suffix.

2.4.6.35 PUSH

PUSH[.W] Push word onto stack

PUSH.B Push byte onto stack

Syntax

```
PUSH src  or  PUSH.W src
PUSH.B src
```

Operation

SP - 2 → SP

src → @SP

Description The stack pointer is decremented by two, then the source operand is moved to the RAM word addressed by the stack pointer (TOS).

Status Bits Status bits are not affected.

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

Example The contents of the status register and R8 are saved on the stack.

```
PUSH     SR      ; save status register
PUSH     R8      ; save R8
```

Example The contents of the peripheral TCDAT is saved on the stack.

```
PUSH.B   &TCDAT  ; save data from 8-bit peripheral module,
                 ; address TCDAT, onto stack
```

NOTE: System Stack Pointer

The System stack pointer (SP) is always decremented by two, independent of the byte suffix.

2.4.6.36 RET

***RET**	Return from subroutine
Syntax	`RET`
Operation	@SP → PC SP + 2 → SP
Emulation	`MOV @SP+,PC`
Description	The return address pushed onto the stack by a CALL instruction is moved to the program counter. The program continues at the code address following the subroutine call.
Status Bits	Status bits are not affected.

2.4.6.37 RETI

RETI	Return from interrupt
Syntax	`RETI`
Operation	TOS → SR SP + 2 → SP TOS → PC SP + 2 → SP
Description	The status register is restored to the value at the beginning of the interrupt service routine by replacing the present SR contents with the TOS contents. The stack pointer (SP) is incremented by two. The program counter is restored to the value at the beginning of interrupt service. This is the consecutive step after the interrupted program flow. Restoration is performed by replacing the present PC contents with the TOS memory contents. The stack pointer (SP) is incremented.
Status Bits	N: Restored from system stack Z: Restored from system stack C: Restored from system stack V: Restored from system stack
Mode Bits	OSCOFF, CPUOFF, and GIE are restored from system stack.
Example	Figure 2-14 illustrates the main program interrupt.

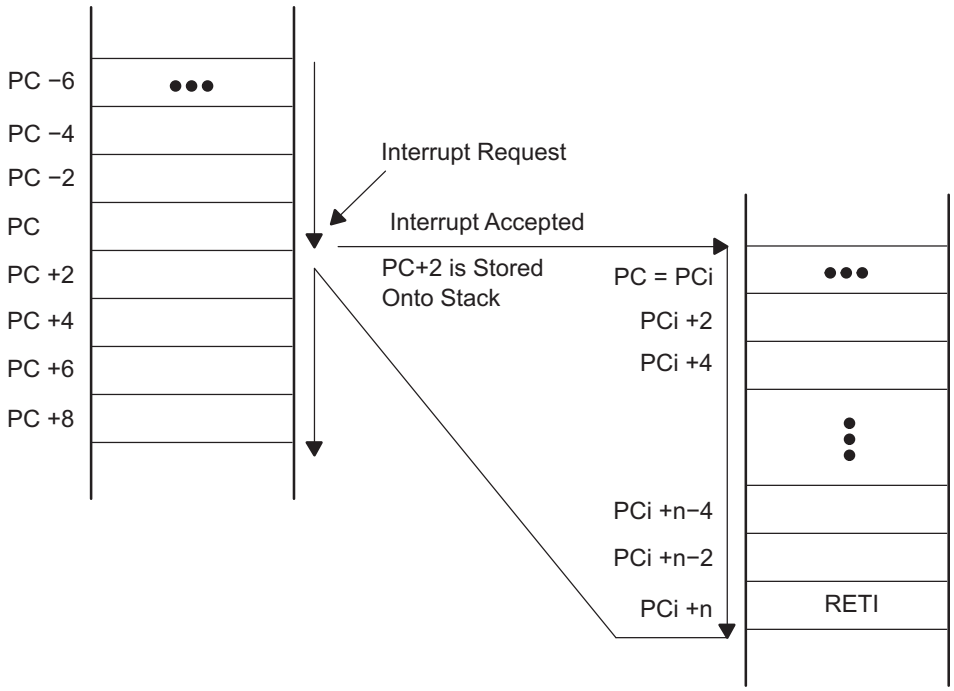


Figure 2-14. Main Program Interrupt

2.4.6.38 RLA

***RLA[.W]** Rotate left arithmetically

***RLA.B** Rotate left arithmetically

Syntax

```
RLA dst or RLA.W dst
RLA.B dst
```

Operation C <- MSB <- MSB-1 .... LSB+1 <- LSB <- 0

Emulation

```
ADD dst,dst
ADD.B dst,dst
```

Description The destination operand is shifted left one position as shown in Figure 2-15. The MSB is shifted into the carry bit (C) and the LSB is filled with 0. The RLA instruction acts as a signed multiplication by 2.

An overflow occurs if dst ≥ 04000h and dst < 0C000h before operation is performed: the result has changed sign.

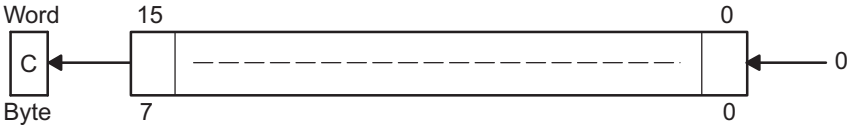


Figure 2-15. Destination Operand – Arithmetic Shift Left

An overflow occurs if dst ≥ 040h and dst < 0C0h before the operation is performed: the result has changed sign.

Status Bits N: Set if result is negative, reset if positive

Z: Set if result is zero, reset otherwise

C: Loaded from the MSB

V: Set if an arithmetic overflow occurs:

the initial value is 04000h ≤ dst < 0C000h; reset otherwise

Set if an arithmetic overflow occurs:

the initial value is 040h ≤ dst < 0C0h; reset otherwise

Mode Bits OSCOFF, CPUOFF,and GIE are not affected.

Example R7 is multiplied by 2.

```
RLA     R7   ; Shift left R7  (x 2)
```

Example The low byte of R7 is multiplied by 4.

```
RLA.B   R7   ; Shift left low byte of R7 (x 2)
RLA.B   R7   ; Shift left low byte of R7 (x 4)
```

NOTE: RLA Substitution

The assembler does not recognize the instruction:

```
RLA @R5+, RLA.B @R5+, or RLA(.B) @R5
```

It must be substituted by:

```
ADD @R5+,-2(R5), ADD.B @R5+,-1(R5), or ADD(.B) @R5
```

2.4.6.39 RLC

***RLC[.W]** Rotate left through carry

***RLC.B** Rotate left through carry

Syntax

```
RLC dst or RLC.W dst
RLC.B dst
```

Operation C <- MSB <- MSB-1 .... LSB+1 <- LSB <- C

Emulation

```
ADDC dst,dst
```

Description The destination operand is shifted left one position as shown in Figure 2-16. The carry bit (C) is shifted into the LSB and the MSB is shifted into the carry bit (C).

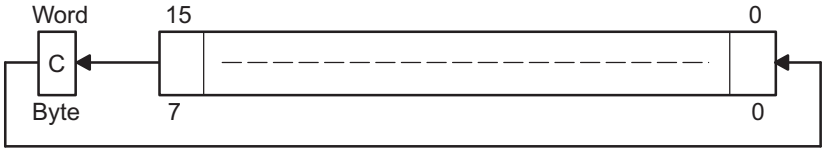


Figure 2-16. Destination Operand - Carry Left Shift

Status Bits

N: Set if result is negative, reset if positive

Z: Set if result is zero, reset otherwise

C: Loaded from the MSB

V: Set if an arithmetic overflow occurs

the initial value is 04000h ≤ dst < 0C000h; reset otherwise

Set if an arithmetic overflow occurs:

the initial value is 040h ≤ dst < 0C0h; reset otherwise

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

Example R5 is shifted left one position.

```
RLC     R5          ; (R5 x 2) + C -> R5
```

Example The input P1IN.1 information is shifted into the LSB of R5.

```
BIT.B   #2,&P1IN    ; Information -> Carry
RLC     R5          ; Carry=P0in.1 -> LSB of R5
```

Example The MEM(LEO) content is shifted left one position.

```
RLC.B   LEO         ; Mem(LEO) x 2 + C -> Mem(LEO)
```

NOTE: RLC and RLC.B Substitution

The assembler does not recognize the instruction:

```
RLC @R5+, RLC @R5, or RLC(.B) @R5
```

It must be substitued by:

```
ADDC @R5+,-2(R5), ADDC.B @R5+,-1(R5), or ADDC(.B) @R5
```

2.4.6.40 RRA

RRA[.W] Rotate right arithmetically

RRA.B Rotate right arithmetically

Syntax

```
RRA dst  or  RRA.W dst
RRA.B dst
```

Operation MSB → MSB, MSB → MSB-1, ... LSB+1 → LSB, LSB → C

Description The destination operand is shifted right one position as shown in Figure 2-17. The MSB is shifted into the MSB, the MSB is shifted into the MSB-1, and the LSB+1 is shifted into the LSB.

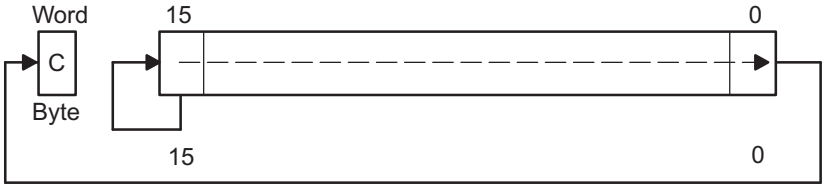


Figure 2-17. Destination Operand – Arithmetic Right Shift

Status Bits

N: Set if result is negative, reset if positive

Z: Set if result is zero, reset otherwise

C: Loaded from the LSB

V: Reset

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

Example R5 is shifted right one position. The MSB retains the old value. It operates equal to an arithmetic division by 2.

```
RRA   R5   ; R5/2 -> R5
;   The value in R5 is multiplied by 0.75 (0.5 + 0.25).
;
PUSH    R5        ; Hold R5 temporarily using stack
RRA     R5        ; R5 x 0.5  ->  R5
ADD     @SP+,R5   ; R5 x 0.5 + R5 = 1.5 x R5  -> R5
RRA     R5        ; (1.5 x R5) x 0.5 = 0.75 x R5  -> R5
......
```

Example The low byte of R5 is shifted right one position. The MSB retains the old value. It operates equal to an arithmetic division by 2.

```
RRA.B    R5       ; R5/2 -> R5: operation is on low byte only
                  ; High byte of R5 is reset
PUSH.B   R5       ; R5 x 0.5  ->  TOS
RRA.B    @SP      ; TOS x 0.5 = 0.5 x R5 x 0.5 = 0.25 x R5  -> TOS
ADD.B    @SP+,R5  ; R5 x 0.5 + R5 x 0.25 = 0.75 x R5  -> R5
......
```

2.4.6.41 RRC

RRC[.W] Rotate right through carry

RRC.B Rotate right through carry

Syntax

```
RRC dst  or  RRC.W dst
RRC dst
```

Operation C → MSB → MSB-1 .... LSB+1 → LSB → C

Description The destination operand is shifted right one position as shown in Figure 2-18. The carry bit (C) is shifted into the MSB, the LSB is shifted into the carry bit (C).

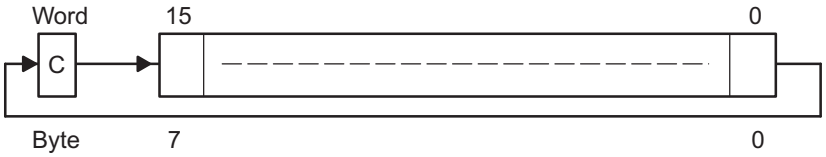


Figure 2-18. Destination Operand - Carry Right Shift

Status Bits

N: Set if result is negative, reset if positive

Z: Set if result is zero, reset otherwise

C: Loaded from the LSB

V: Reset

Mode Bits OSCOFF, CPUOFF, and GIEare not affected.

Example R5 is shifted right one position. The MSB is loaded with 1.

```
SETC          ; Prepare carry for MSB
RRC     R5    ; R5/2 + 8000h -> R5
```

Example R5 is shifted right one position. The MSB is loaded with 1.

```
SETC          ; Prepare carry for MSB
RRC.B   R5    ; R5/2 + 80h -> R5; low byte of R5 is used
```

2.4.6.42 SBC

***SBC[.W]** Subtract source and borrow/.NOT. carry from destination

***SBC.B** Subtract source and borrow/.NOT. carry from destination

Syntax

```
SBC dst  or  SBC.W dst
SBC.B dst
```

Operation

dst + 0FFFFh + C → dst

dst + 0FFh + C → dst

Emulation

```
SUBC #0,dst
SUBC.B #0,dst
```

Description The carry bit (C) is added to the destination operand minus one. The previous contents of the destination are lost.

Status Bits

N: Set if result is negative, reset if positive

Z: Set if result is zero, reset otherwise

C: Set if there is a carry from the MSB of the result, reset otherwise.

Set to 1 if no borrow, reset if borrow.

V: Set if an arithmetic overflow occurs, reset otherwise.

Mode Bits OSCOFF, CPUOFF,and GIE are not affected.

Example The 16-bit counter pointed to by R13 is subtracted from a 32-bit counter pointed to by R12.

```
SUB     @R13,0(R12)   ; Subtract LSDs
SBC     2(R12)        ; Subtract carry from MSD
```

Example The 8-bit counter pointed to by R13 is subtracted from a 16-bit counter pointed to by R12.

```
SUB.B   @R13,0(R12)   ; Subtract LSDs
SBC.B   1(R12)        ; Subtract carry from MSD
```

NOTE: Borrow Implementation

The borrow is treated as a .NOT. carry:

Borrow	Carry bit
Yes	0
No	1

2.4.6.43 SETC

***SETC**	Set carry bit
Syntax	`SETC`
Operation	1 → C
Emulation	`BIS #1,SR`
Description	The carry bit (C) is set.
Status Bits	N: Not affected Z: Not affected C: Set V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Emulation of the decimal subtraction: Subtract R5 from R6 decimally Assume that R5 = 03987h and R6 = 04137h

```
DSUB    ADD     #06666h,R5    ; Move content R5 from 0-9 to 6-0Fh
                              ; R5 = 03987h + 06666h = 09FEDh
        INV     R5            ; Invert this (result back to 0-9)
                              ; R5 = .NOT. R5 = 06012h
        SETC                  ; Prepare carry = 1
        DADD    R5,R6         ; Emulate subtraction by addition of:
                              ; (010000h - R5 - 1)
                              ; R6 = R6 + R5 + 1
                              ; R6 = 0150h
```

2.4.6.44 SETN

***SETN**	Set negative bit
Syntax	`SETN`
Operation	1 → N
Emulation	`BIS #4,SR`
Description	The negative bit (N) is set.
Status Bits	N: Set Z: Not affected C: Not affected V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.

2.4.6.45 SETZ

***SETZ**	Set zero bit
Syntax	`SETZ`
Operation	1 → Z
Emulation	`BIS #2,SR`
Description	The zero bit (Z) is set.
Status Bits	N: Not affected Z: Set C: Not affected V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.

2.4.6.46 SUB

SUB[.W]	Subtract source from destination
SUB.B	Subtract source from destination
Syntax	`SUB src,dst  or  SUB.W src,dst` `SUB.B src,dst`
Operation	dst + .NOT.src + 1 → dst or [(dst - src → dst)]
Description	The source operand is subtracted from the destination operand by adding the source operand's 1s complement and the constant 1. The source operand is not affected. The previous contents of the destination are lost.
Status Bits	N: Set if result is negative, reset if positive Z: Set if result is zero, reset otherwise C: Set if there is a carry from the MSB of the result, reset otherwise. Set to 1 if no borrow, reset if borrow. V: Set if an arithmetic overflow occurs, otherwise reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	See example at the SBC instruction.
Example	See example at the SBC.B instruction.

NOTE: Borrow Is Treated as a .NOT.

The borrow is treated as a .NOT. carry:	Borrow	Carry bit
	Yes	0
	No	1

2.4.6.47 SUBC, SBB

SUBC[.W], SBB[.W] Subtract source and borrow/.NOT. carry from destination

SUBC.B, SBB.B Subtract source and borrow/.NOT. carry from destination

Syntax

```
SUBC     src,dst    or    SUBC.W   src,dst    or
SBB      src,dst    or    SBB.W    src,dst
SUBC.B   src,dst    or    SBB.B    src,dst
```

Operation

dst + .NOT.src + C → dst

or

(dst - src - 1 + C → dst)

Description The source operand is subtracted from the destination operand by adding the source operand's 1s complement and the carry bit (C). The source operand is not affected. The previous contents of the destination are lost.

Status Bits

N: Set if result is negative, reset if positive.

Z: Set if result is zero, reset otherwise.

C: Set if there is a carry from the MSB of the result, reset otherwise.

Set to 1 if no borrow, reset if borrow.

V: Set if an arithmetic overflow occurs, reset otherwise.

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

Example Two floating point mantissas (24 bits) are subtracted.

LSBs are in R13 and R10, MSBs are in R12 and R9.

```
SUB.W    R13,R10     ; 16-bit part, LSBs
SUBC.B   R12,R9      ; 8-bit part, MSBs
```

Example The 16-bit counter pointed to by R13 is subtracted from a 16-bit counter in R10 and R11(MSD).

```
SUB.B    @R13+,R10   ; Subtract LSDs without carry
SUBC.B   @R13,R11    ; Subtract MSDs with carry
...                  ; resulting from the LSDs
```

NOTE: Borrow Implementation

The borrow is treated as a .NOT. carry:

Borrow	Carry bit
Yes	0
No	1

2.4.6.48 SWPB

SWPB Swap bytes

Syntax `SWPB dst`

Operation Bits 15 to 8 ↔ bits 7 to 0

Description The destination operand high and low bytes are exchanged as shown in Figure 2-19.

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

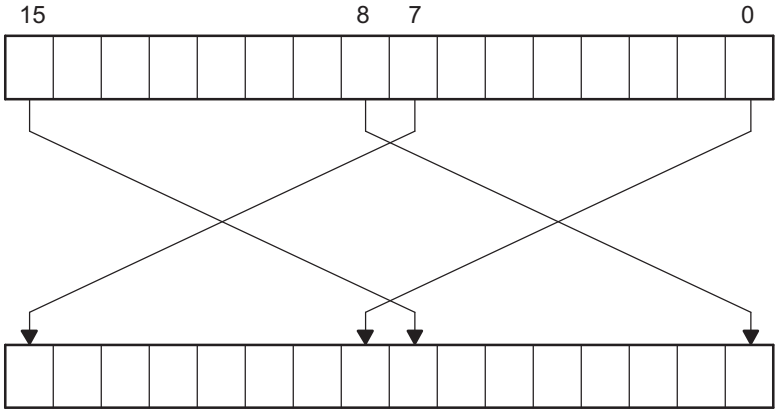


Figure 2-19. Destination Operand - Byte Swap

Example

```
MOV    #040BFh,R7   ; 0100000010111111 -> R7
SWPB   R7           ; 1011111101000000 in R7
```

Example The value in R5 is multiplied by 256. The result is stored in R5,R4.

```
SWPB   R5           ;
MOV    R5,R4        ; Copy the swapped value to R4
BIC    #0FF00h,R5   ; Correct the result
BIC    #00FFh,R4    ; Correct the result
```

2.4.6.49 SXT

SXT Extend Sign

Syntax `SXT dst`

Operation Bit 7 → Bit 8 ..... Bit 15

Description The sign of the low byte is extended into the high byte as shown in Figure 2-20.

Status Bits N: Set if result is negative, reset if positive

Z: Set if result is zero, reset otherwise

C: Set if result is not zero, reset otherwise (.NOT. Zero)

V: Reset

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

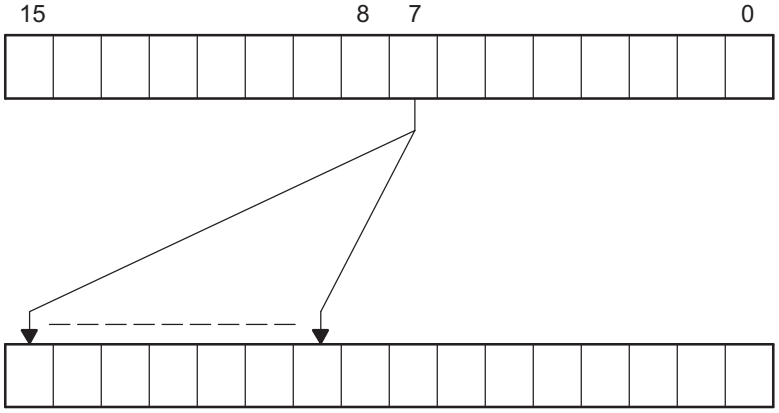


Figure 2-20. Destination Operand - Sign Extension

Example R7 is loaded with the P1IN value. The operation of the sign-extend instruction expands bit 8 to bit 15 with the value of bit 7.

R7 is then added to R6.

```
MOV.B   &P1IN,R7   ; P1IN = 080h:    .... .... 1000 0000
SXT     R7         ; R7 = 0FF80h:   1111 1111 1000 0000
```

2.4.6.50 TST

***TST[.W]**	Test destination
***TST.B**	Test destination
Syntax	`TST dst  or  TST.W dst` `TST.B dst`
Operation	dst + 0FFFFh + 1 dst + 0FFh + 1
Emulation	`CMP #0,dst` `CMP.B #0,dst`
Description	The destination operand is compared with zero. The status bits are set according to the result. The destination is not affected.
Status Bits	N: Set if destination is negative, reset if positive Z: Set if destination contains zero, reset otherwise C: Set V: Reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	R7 is tested. If it is negative, continue at R7NEG; if it is positive but not zero, continue at R7POS.

```
         TST     R7        ; Test R7
         JN      R7NEG     ; R7 is negative
         JZ      R7ZERO    ; R7 is zero
R7POS    ......            ; R7 is positive but not zero
R7NEG    ......            ; R7 is negative
R7ZERO   ......            ; R7 is zero
```

Example The low byte of R7 is tested. If it is negative, continue at R7NEG; if it is positive but not zero, continue at R7POS.

```
         TST.B   R7        ; Test low byte of R7
         JN      R7NEG     ; Low byte of R7 is negative
         JZ      R7ZERO    ; Low byte of R7 is zero
R7POS    ......            ; Low byte of R7 is positive but not zero
R7NEG    .....             ; Low byte of R7 is negative
R7ZERO   ......            ; Low byte of R7 is zero
```

2.4.6.51 XOR

XOR[.W]	Exclusive OR of source with destination
XOR.B	Exclusive OR of source with destination
Syntax	`XOR src,dst  or  XOR.W src,dst` `XOR.B src,dst`
Operation	src .XOR. dst → dst
Description	The source and destination operands are exclusive ORed. The result is placed into the destination. The source operand is not affected.
Status Bits	N: Set if result MSB is set, reset if not set Z: Set if result is zero, reset otherwise C: Set if result is not zero, reset otherwise (= .NOT. Zero) V: Set if both operands are negative
Mode Bits	OSCOFF, CPUOFF,and GIE are not affected.
Example	The bits set in R6 toggle the bits in the RAM word TONI. `XOR     R6,TONI  ; Toggle bits of word TONI on the bits set in R6`

Example	The bits set in R6 toggle the bits in the RAM byte TONI.
	`XOR.B   R6,TONI  ; Toggle bits of byte TONI on the bits set in`
	`                ; low byte of R6`

Example	Reset to 0 those bits in low byte of R7 that are different from bits in RAM byte EDE.
	`XOR.B   EDE,R7   ; Set different bit to "1s"`
	`INV.B   R7       ; Invert Lowbyte, Highbyte is 0h`

Power Management Module (PMM)

The Power Management Module (PMM) manages all functions related to the power supply and its supervision for the device. This chapter describes the PMM.

Topic **Page**

3.1 Power Management Module Introduction

PMM features include:

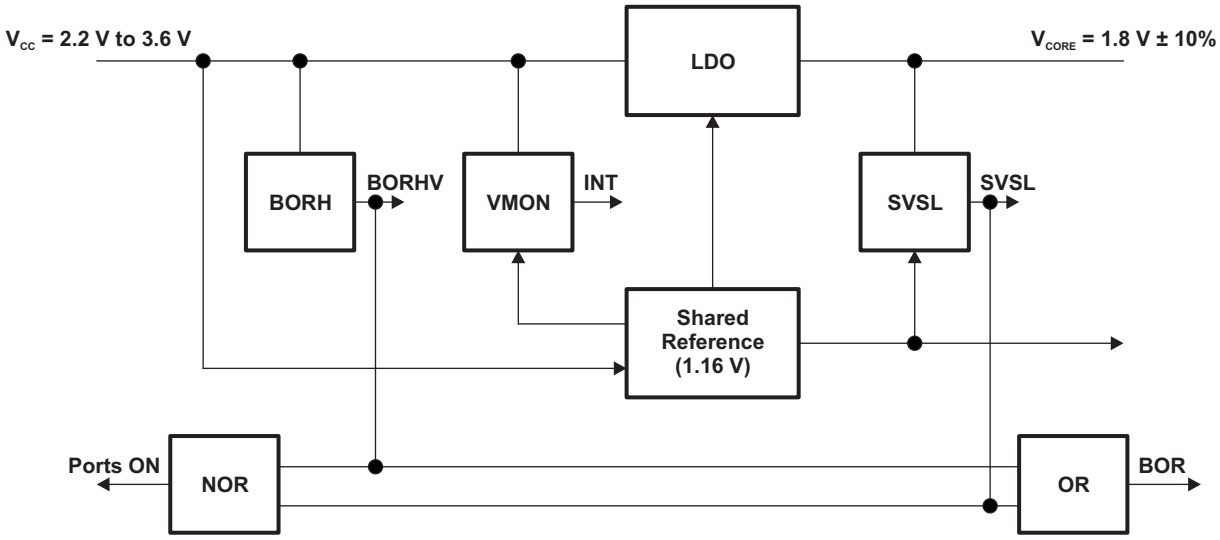
- Supply voltage (V_{CC}) range: 2.2 V to 3.6 V
- High-side brownout reset (BORH)
- Supply voltage monitor (VMON) for V_{CC} with programmable threshold levels and monitoring of external pin (VMONIN) against internal reference
- Generation of fixed voltage of 1.8 V for the device core (V_{CORE})
- Supply voltage supervisor (SVS) for V_{CORE}
- Precise 1.16-V reference for the entire device and integrated temperature sensor.

The PMM manages all functions related to the power supply and its supervision for the device. Its primary functions are:

1. To generate a supply voltage for the core logic
2. To provide mechanisms for the supervising and monitoring of both the voltage applied to the device (V_{CC}) and the voltage generated for the core (V_{CORE})

The PMM uses an integrated low-dropout voltage regulator (LDO) to produce a secondary core voltage (V_{CORE}) from the primary voltage applied to the device (V_{CC}). V_{CORE} supplies the CPU, memories (flash and RAM), and the digital modules, while V_{CC} supplies the I/Os and analog modules. The V_{CORE} output is maintained using a voltage reference that is generated by the reference block within the PMM. The input or primary side of the regulator is referred to in this chapter as its high side. The output or secondary side is referred to in this chapter as its low side.

Figure 3-1 shows the block diagram of the power management system.



NOTE: BORHV is active-high polarity (1 = V_{CC} domain is in reset). SVSL is active-high polarity (1 = V_{CORE} domain is in reset).

Figure 3-1. PMM and REF Block Diagram

3.2 Power Management Module Operation

3.2.1 Voltage Regulator

V_{CC} can be powered from a wide input voltage range, but the core logic of the device must be kept at a voltage lower than what this range allows. For this reason, a regulator has been integrated into the PMM. The regulator derives the necessary core voltage (V_{CORE}) from V_{CC}.

3.2.2 Brownout Reset (BOR) and Supply Voltage Supervisor (SVS) – Power Up

The primary function of the brownout reset (BOR) circuit occurs when the device is powering up. It is functional very early in the power-up ramp, generating a reset that initializes the system. It also functions when a brownout condition occurs after power-up. It sustains this reset until the input voltage is high enough for a proper reset of the system.

There is a supply voltage supervisor (SVSL) on the low side of the PMM, which gives an indication that the V_{CORE} is in the usable range for the digital core and the rest of the circuitry powered on the core voltage. Early in the power-up ramp, V_{CC} is low and, therefore, the PMM holds the device in BOR reset. After V_{CC} crosses the brownout level, the LDO is turned ON, and starts to ramp up V_{CORE}. When V_{CORE} rises above the SVSL level, the BOR reset is released after a small delay. Figure 3-2 shows this process.

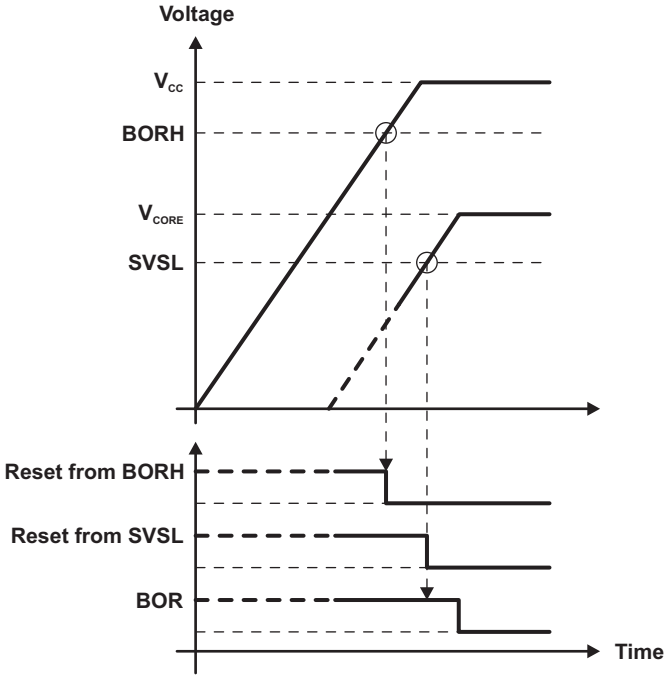


Figure 3-2. Power-Up Sequence

3.2.3 Voltage Monitor (VMON)

The voltage monitor (VMON) compares the voltage applied on either the external device pin VMONIN or the on-chip DVCC to a specified limit. If the monitored voltage falls below the threshold, VMON triggers an interrupt.

The voltage monitor is disabled after power up. The application must enable this module by programming appropriate values into the VMONLVLx bits in the VMONCTL register:

- Set VMONLVLx = 111b to monitor the voltage on the VMONIN pin against an internal reference voltage of 1.16 V.
- Set VMONLVLx = 001b, 010b, or 011b to monitor DVCC against one of the three user-selectable threshold voltages. See Section 3.3.2 for details on programmable threshold voltages for monitoring DVCC.

The interrupt flag (VMONIFG) and interrupt enable (VMONIE) are defined in VMONCTL register. When VMONIFG is set and VMONIE is enabled, the voltage monitor triggers a maskable interrupt to the CPU. VMONIFG flag indicates the status of supply being monitored. When VMON is disabled, VMONIFG is 0. Software can write into VMONIFG to emulate a VMON interrupt when the module is disabled. When VMON is enabled, software writes to VMONIFG are ignored, and VMONIFG reflects the status of the supply being monitored.

VMON requires a settling time of 0.5 µs (typical) when it is turned on or when the monitoring mode is changed from VMONIN to DVCC or from DVCC to VMONIN. VMONIFG is not reliable during this settling time. There is no settling time involved when the DVCC monitoring threshold is changed from one value to another, and VMONIFG status is always reliable.

In the external voltage monitoring mode, VMONIFG = 0 when the voltage applied on the VMONIN pin is above the internal reference voltage, and VMONIFG = 1 when the voltage on the VMONIN pin is below the internal reference voltage.

In the DVCC monitoring mode, VMONIFG = 0 when the DVCC is above the selected threshold voltage, and VMONIFG = 1 when DVCC is below the selected threshold voltage.

When the voltage monitor is enabled and VMONIFG = 1, software cannot clear the flag as long as the low-voltage condition persists. The voltage monitor is disabled and VMONIFG is cleared by hardware when the VMONLVLx bits are programmed to 000b or with any of the reserved values. When the voltage monitor is disabled and an interrupt is triggered from software by writing 1 to VMONIFG and VMONIE bits, the VMONIFG bit is cleared when the CPU enters the VMON interrupt service routine.

3.2.3.1 Voltage Monitor Timing Waveforms

The following waveforms show the operation of VMON in different use cases.

Use Case 1: VMON is configured to monitor DVCC or VMONIN and low voltage is detected.

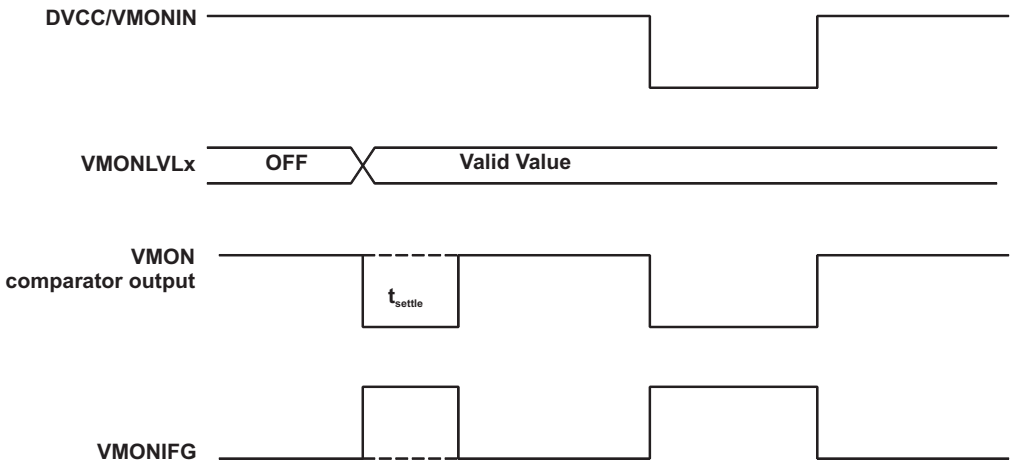


Figure 3-3. Voltage Monitor Timing, Use Case 1

Use Case 2: VMON is configured to monitor DVCC/VMONIN and low voltage is detected –> VMON disabled by software.

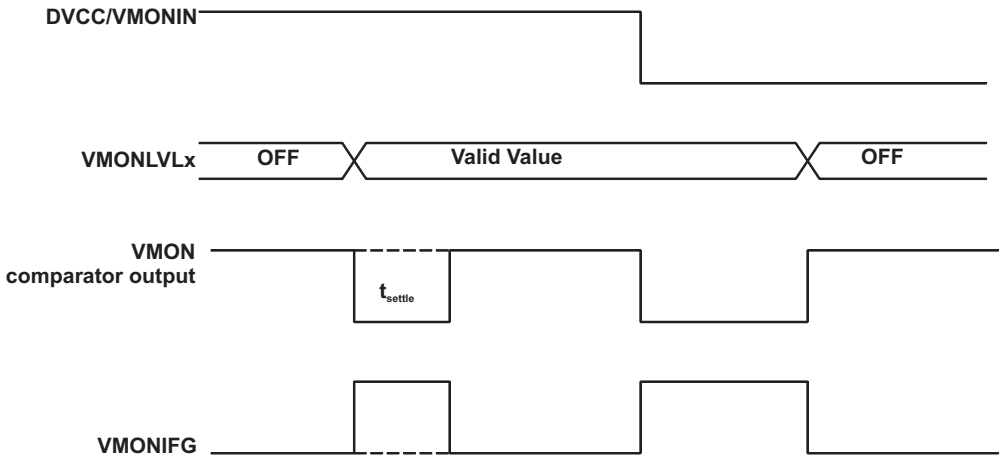


Figure 3-4. Voltage Monitor Timing, Use Case 2

Use Case 3: VMON is configured to monitor DVCC and low voltage is detected –> VMON is switched to monitor VMONIN and low voltage is detected –> VMON disabled by software during low voltage condition.

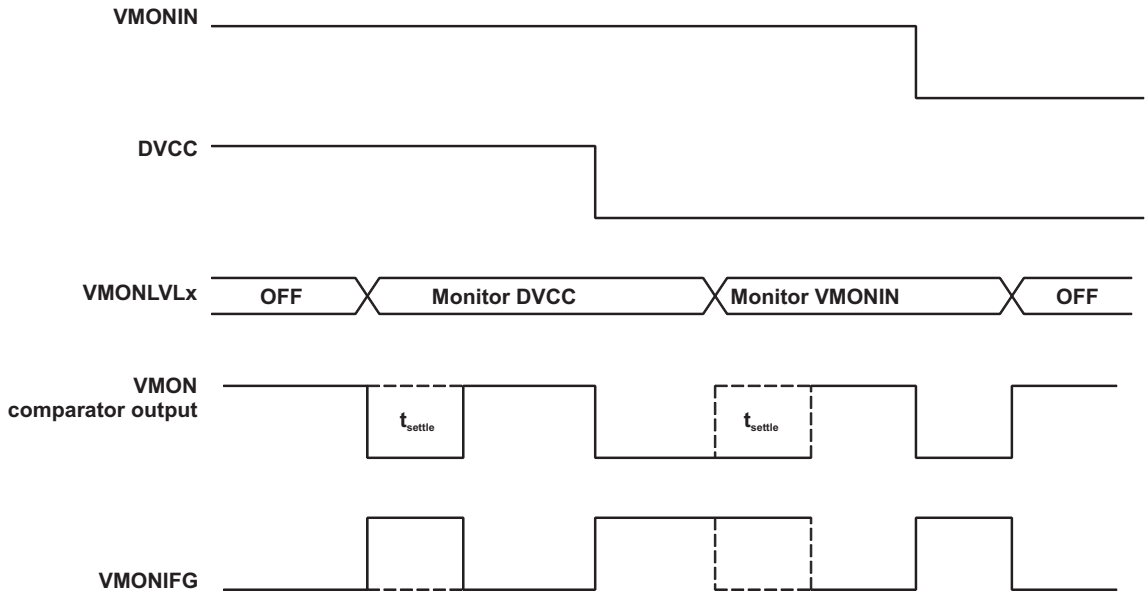


Figure 3-5. Voltage Monitor Timing, Use Case 3

3.2.4 LPM4.5

LPM4.5 is an additional low-power mode in which the regulator of the PMM is disabled to provide additional power savings. Because there is no power supplied to V_{CORE} during LPM4.5, the CPU and all digital modules including RAM are unpowered. This disables nearly the entire device, and the contents of the registers and RAM are lost. Any essential values should be stored to flash prior to entering LPM4.5. LPM4.5 mode is entered when REGOFF bit is set to 1 in the LPM45CTL register followed by LPM4 programming in the CPU status register.

Because the regulator of the PMM is disabled upon entering LPM4.5, all I/O register configurations are lost. Therefore, the configuration of I/O pins must be handled differently to ensure that all pins in the application behave in a controlled manner upon entering and exiting LPM4.5. Properly setting the I/O pins is critical to achieving the lowest possible power consumption in LPM4.5, as well as preventing any possible uncontrolled input or output I/O state in the application. The application has complete control of the I/O pin conditions that prevent unwanted spurious activity upon entry and exit from LPM4.5. The I/O pin state is held and locked based on the settings prior to LPM4.5 entry. Upon entry into LPM4.5, the LOCKLPM45 bit in the LPM45CTL register of the PMM module is set automatically. Note that only the pin condition is retained. All other port configuration register settings are lost.

Exit from LPM4.5 is possible with an active-low event on $\overline{\text{RST}}$/NMI pin, a power-on cycle, or by specific I/O. Refer to device-specific data sheet for details on LPM4.5 wakeup capable I/Os. Any exit from LPM4.5 causes a BOR. Program execution continues at the location stored in the system reset vector location 0FFFEh. The LPM45IFG bit inside the PMM module is set to indicate that the device was in LPM4.5 prior to the wakeup event. During LPM4.5, all I/O pin conditions are automatically locked to the current state. Upon exit from LPM4.5, the I/O pin conditions remain locked until the application unlocks them by writing 0 to the LOCKLPM45 bit. See Section 6.3 for complete details.

3.2.5 Shared Reference

The REF module is a general-purpose reference system that generates voltage references required for other subsystems such as digital-to-analog converters, analog-to-digital converters, and comparators.

Features of the REF include:

- Centralized bandgap with excellent PSRR, temperature coefficient, and accuracy
- Unbuffered bandgap voltage available to rest of system
- Integrated temperature sensor

Figure 3-6 shows the block diagram of the REF module.

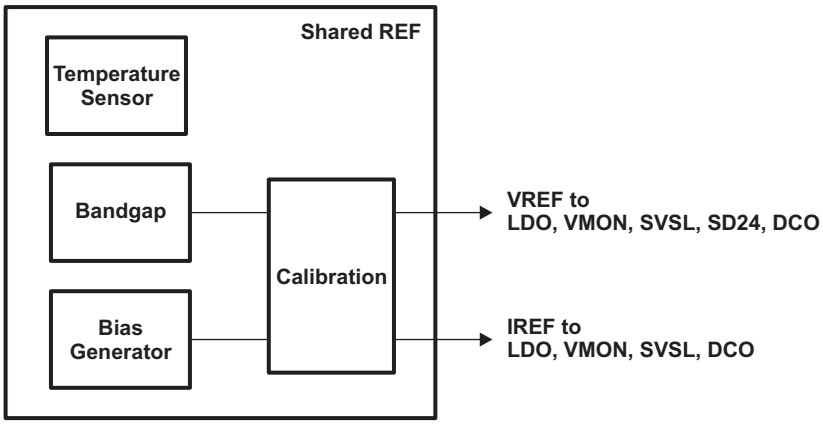


Figure 3-6. REF Block Diagram

The REF module provides all of the necessary voltage references for use by various peripheral modules throughout the system. These include, but are not limited to, devices that contain an LDO, VMON, SVSL, SD24, analog-to-digital converter (ADC), digital-to-analog converter (DAC), or digitally controlled oscillator (DCO).

The REF subsystem contains a high-performance bandgap. This bandgap has good accuracy, low temperature coefficient, and high PSRR while operating at low power. Additionally, the REF generates bias currents required for the LDO, VMON, SVSL, and DCO.

3.2.5.1 Reference Calibration

The application should calibrate the reference voltage and current for the specified accuracy by programming the REFCAL0 and REFCAL1 registers. The calibration values are stored in the flash information memory in TLV format. See the REF section in the device-specific data sheet for complete the electrical specification of the REF module.

3.2.5.2 Temperature Sensor

The REF subsystem also includes the temperature sensor circuitry, which is derived from the bandgap. The temperature sensor is used by a sigma-delta ADC (SD24) to measure a voltage proportional to temperature. The temperature sensor is not enabled by default, and it is active only when the sigma-delta ADC selects the temperature sensor as an input.

3.3 PMM Registers

Table 3-1 lists the PMM registers.

Table 3-1. PMM Registers

Address	Acronym	Register Name	Type	Access	Reset	Section
0060h	LPM45CTL	LPM4.5 Control	Read/Write	Byte	00h	Section 3.3.1
0061h	VMONCTL	Voltage Monitor Control	Read/Write	Byte	00h	Section 3.3.2
0062h	REFCAL0	Reference Calibration 0	Read/Write	Byte	00h	Section 3.3.3
0063h	REFCAL1	Reference Calibration 1	Read/Write	Byte	00h	Section 3.3.4

3.3.1 LPM45CTL Register (address = 0060h) [reset = 00h]

LPM4.5 Control Register

Figure 3-7. LPM45CTL Register

7	6	5	4	3	2	1	0
Reserved			REGOFF	Reserved		LPM45IFG	LOCKLPM45
r0	r0	r0	rw-0	r0	r0	rw-[0]	rw-[0]

Table 3-2. LPM45CTL Register Description

Bit	Field	Type	Reset	Description
7-5	Reserved	R	0h	Reserved. Always reads as 0.
4	REGOFF	RW	0h	PMM regulator off. If this bit is set followed by LPM4 programming, internal voltage regulator is turned off. 0b = PMM voltage regulator is operating 1b = PMM voltage regulator is turned off when LPM4 mode is programmed subsequently
3-2	Reserved	R	0h	Reserved. Always reads as 0.
1	LPM45IFG	RW	0h	LPM4.5 interrupt flag. This bit is set if the system was in LPM4.5 mode before. This bit is cleared by software or by a BOR. 0b = Device was not in LPM4.5 mode 1b = Device was in LPM4.5 mode
0	LOCKLPM45	RW	0h	Lock I/O pin configuration upon entry to or exit from LPM4.5. This bit can only be written as 0. When power is applied to the device, this bit, once set by hardware, can be cleared only by user software or by BOR. 0b = I/O pin configuration is not locked and defaults to its reset condition. 1b = I/O pin configuration remains locked. Pin state is held during LPM4.5 entry and exit.

3.3.2 VMONCTL Register (address = 0061h) [reset = 00h]

Voltage Monitor Control Register

Figure 3-8. VMONCTL Register

7	6	5	4	3	2	1	0
Reserved		VMONIFG	VMONIE	Reserved	VMONLVLx		
r0	r0	rw-(0)	rw-0	r0	rw-(0)	rw-(0)	rw-(0)

Table 3-3. VMONCTL Register Description

Bit	Field	Type	Reset	Description
7-6	Reserved	R	0h	Reserved. Always reads as 0.
5	VMONIFG	RW	0h	VMON interrupt flag 0b = No low-voltage condition 1b = DVCC below the selected threshold or external VMONIN below 1.16 V
4	VMONIE	RW	0h	VMON interrupt enable. If this bit it set, when DVCC or external VMONIN goes below the monitored level, an interrupt is generated
3	Reserved	R	0h	Reserved. Always reads as 0.
2-0	VMONLVLx	RW	0h	Voltage monitor level selection. These bits are used to select between DVCC and external VMONIN pin for voltage monitoring and also to define the monitoring levels when DVCC is being monitored. 000b = Disable VMON 001b = Compare DVCC to 2.35 V (typ) 010b = Compare DVCC to 2.65 V (typ) 011b = Compare DVCC to 2.85 V (typ) 100b = Reserved 101b = Reserved 110b = Reserved 111b = Compare VMONIN to 1.16 V (typ)

3.3.3 REFCAL0 Register (address = 0062h) [reset = 00h]

Reference Calibration 0 Register

Figure 3-9. REFCAL0 Register

7	6	5	4	3	2	1	0
Reserved		BGCOARSE		BGFINE			
r0	r0	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]

Table 3-4. REFCAL0 Register Description

Bit	Field	Type	Reset	Description
7-6	Reserved	R	0h	Reserved. Always reads as 0.
5-4	BGCOARSE	RW	0h	Bandgap voltage coarse calibration.
3-0	BGFINE	RW	0h	Bandgap voltage fine calibration.

3.3.4 REFCAL1 Register (address = 0063h) [reset = 00h]

Reference Calibration 1 Register

Figure 3-10. REFCAL1 Register

7	6	5	4	3	2	1	0
BIASCURRENT				BGCURVE			
rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]

Table 3-5. REFCAL1 Register Description

Bit	Field	Type	Reset	Description
7-4	BIASCURRENT	RW	0h	Bandgap bias current calibration.
3-0	BGCURVE	RW	0h	Bandgap curvature compensation.

Clock System (CS)

The clock system provides clocks for MSP430i devices. This chapter describes the operation of the clock system.

Topic **Page**

4.1 Clock System Introduction

Features of the clock system:

- 16.384-MHz fixed-frequency internal digitally controlled oscillator (DCO)
- DCO operation with internal or external resistor
- Higher clock accuracy while operating with external resistor
- Fail-safe feature in external resistor mode
- Status indication and non-maskable interrupt generation upon DCO external resistor fault
- Support for DCO bypass mode in which an external digital clock can be input on a device pin
- Programmable independent clock dividers for MCLK and SMCLK
- ACLK at fixed 32-kHz frequency
- Support for different power modes: Active mode, LPM0 to LPM4, and LPM4.5

Figure 4-1 shows the block diagram of the clock system.

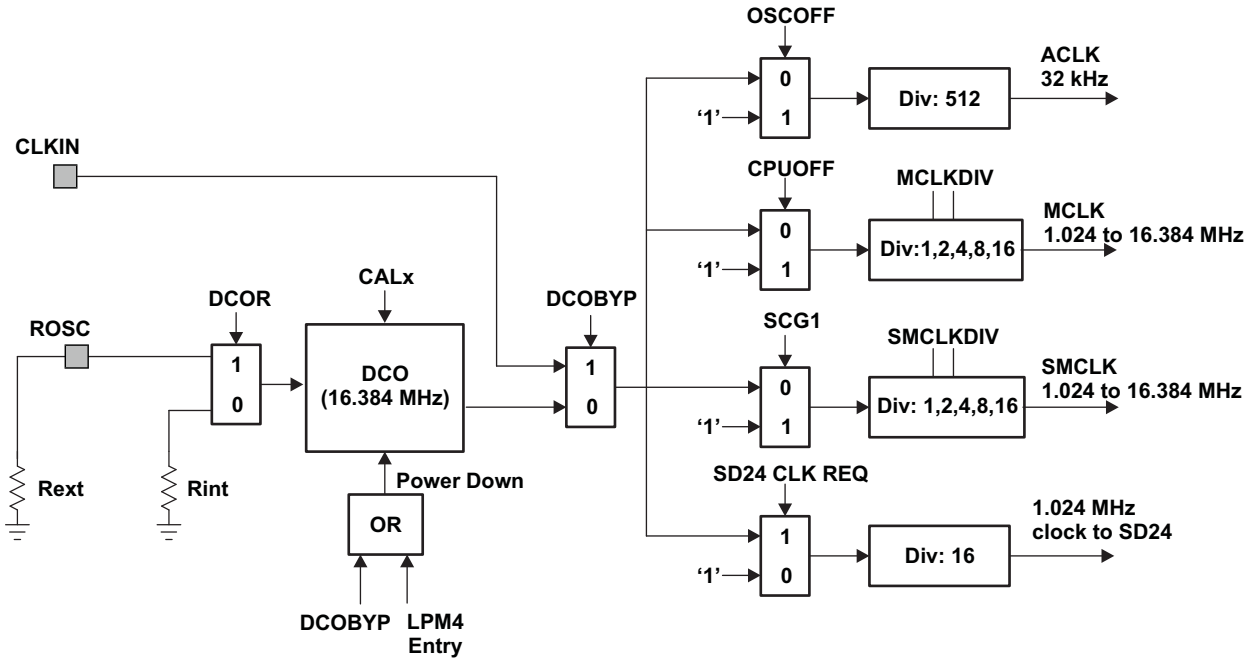


Figure 4-1. Clock System Block Diagram

4.2 Clock System Operation

The following sections describe the operation of clock system in the MSP430i family.

4.2.1 Digitally Controlled Oscillator (DCO)

The DCO is an 16.384-MHz fixed-frequency integrated digitally controlled oscillator. The DCO supports two modes of operation. It can operate with an internal resistor or with an external resistor that is connected to the ROSC pin of the device. The application selects between these modes by configuring the DCOR bit in the CSCTL0 register.

4.2.1.1 DCO Operation in Internal Resistor Mode

After power up, the DCOR bit is reset, which selects the internal resistor mode. The DCO generates a clock signal after the specified start-up time at an uncalibrated frequency in the range of 10 MHz to 14 MHz. The application should calibrate the DCO for 16.384-MHz frequency and for the specified clock accuracy by programming the CSIRFCAL and CSIRTCAL registers. The calibration values are stored in flash information memory in TLV format. See the DCO parameters in the device-specific data sheet for clock accuracy in internal resistor mode.

The MCLK and SMCLK clock divider bits in CSCTL1 register are reset after power up, which keeps MCLK and SMCLK running at 16.384-MHz frequency. The clock dividers can be programmed by the application for the desired MCLK and SMCLK clock frequency. ACLK runs at a fixed 32-kHz frequency. The fault detection circuitry in DCO is inactive in the internal resistor mode, and the DCO fault bit (DCOF) in CSCTL0 register is always reset.

4.2.1.2 DCO Operation in External Resistor Mode

When DCOR bit is set to 1 by the application, the DCO selects the external resistor mode for operation. It is recommended to connect a 20-kΩ 0.1% ±50-ppm/°C resistor at the ROSC pin of the device for operating in the external resistor mode. This mode offers higher clock accuracy in terms of absolute tolerance and temperature drift compared to internal resistor mode. After the external resistor mode is selected, the DCO generates a clock signal at an uncalibrated frequency in the range of 10 MHz to 14 MHz. The application should calibrate the DCO for 16.384-MHz frequency and for the specified clock accuracy by programming the CSERFCAL and CSERTCAL registers. See the DCO parameters in the device-specific data sheet for clock accuracy in external resistor mode. The clock dividers can be programmed to change MCLK or SMCLK frequency. ACLK runs at fixed 32-kHz frequency.

4.2.1.3 Fault Detection and Fail-Safe

When the DCOR bit is set, the fault detection circuitry in the DCO is activated and performs a health check of the external resistor that is connected. The DCOF bit is set to 1 when the external resistor mode is selected and is automatically cleared after a short delay if the external resistor is normal. If the DCO detects any abnormality (for example, if the ROSC pin is left open or shorted to ground, or if the resistance connected at the ROSC pin is far from the recommended value), then the DCOF bit remains at 1 to indicate a persistent DCO fault.

If there is a persistent DCO fault, the DCO automatically switches to the internal resistor mode as a fail-safe mechanism. The external resistor is decoupled and the internal resistor is used. If the DCO was calibrated for internal resistor mode, then the clock signal is generated at a precise 16.384-MHz frequency in the fail-safe mode of operation. The DCO fault-sensing circuitry is continuously active while the DCO operates in external resistor mode, and it can detect faults during runtime and switch to fail-safe mode.

The DCOF bit also sets the OFIFG bit in SFR IFG1 register. OFIFG can source a non-maskable interrupt to CPU when the OFIE bit is enabled in the SFR IE1 register. DCOF is a status bit and is set or cleared based on DCO fault indication. OFIFG is a sticky bit that is set when DCOF is set, and it cannot be cleared as long as DCOF is set. When DCOF is cleared, OFIFG is not cleared automatically and must be cleared by software.

4.2.1.4 DCO Bypass Mode

The clock system supports a DCO bypass mode. When DCOBYP bit in CSCTL0 register is set to 1, the DCO is powered down and completely bypassed. An external digital clock can be input on the CLKIN pin of the device to drive all clocks on the device. MCLK and SMCLK frequency can be changed by programming clock dividers in bypass mode. ACLK frequency is not programmable and is fixed to the bypass clock frequency divided by 512. If the bypass clock supplied at CLKIN pin is 16.384-MHz, ACLK runs at 32-kHz frequency.

4.2.2 Clocking Sigma-Delta ADC (SD24)

The clock system provides a 1.024-MHz fixed-frequency clock to the sigma-delta ADC (SD24) module. This clock is not free running and is delivered only when a clock request from SD24 is asserted. This clock is not affected by LPM0 to LPM4 modes and runs as long as the SD24 clock request is active. If the SD24 is not active when entering LPM4, the DCO is powered down. If the SD24 is active while entering LPM4, the DCO is not powered down and the clock system stops MCLK, SMCLK, ACLK but continues to deliver the clock to SD24. The SD24 clock is stopped and the DCO is powered down when the SD24 clock request is de-asserted. Port interrupt wakes up the device from LPM4 and powers up the DCO.

If SD24 functionality is desired in DCO bypass mode, an 16.384-MHz clock should be provided at the CLKIN pin. The SD24 module does not operate reliably if the external clock frequency in bypass mode is not 16.384 MHz.

4.2.3 Device Power Modes

The clock system in the MSP430i family supports active mode and six different low-power modes. The programming and description of the power modes and which clocks are running or stopped are given in Table 4-1. REGOFF is a bit in the LPM45CTL register of PMM that is used to shut off the LDO. CPUOFF, SCG1, OSCOFF are the low-power mode control bits in the CPU status register. The SCG0 bit in the CPU status register has no effect on power modes in the MSP430i clock system.

To enter LPM4.5 mode, the REGOFF bit must be set to 1 followed by LPM4 programming. If the SD24 clock request is not active, the DCO is turned off when entering LPM4, and the voltage regulator is turned off when entering LPM4.5 mode. If the SD24 clock request is active, the device enters LPM4 only when the SD24 clock request is deasserted, and the voltage regulator is turned off after that to enter LPM4.5 mode. The device can be waked up from LPM4.5 mode by the selected port I/O interrupt or by an external reset triggered through the $\overline{\text{RST}}$/NMI pin. See the device-specific data sheet for details on which port I/Os can be programmed to wake up the device from LPM4.5.

Table 4-1. Power Modes

REGOFF	CPUOFF	SCG1	OSCOFF	Power Mode and Description
0	0	0	0	Active mode. LDO and DCO on. MCLK, SMCLK, ACLK running.
0	0	0	1	LDO and DCO on. MCLK, SMCLK running. ACLK stopped.
0	0	1	0	LDO and DCO on. MCLK, ACLK running. SMCLK stopped.
0	0	1	1	LDO and DCO on. MCLK running. SMCLK, ACLK stopped.
0	1	0	0	LPM0 or LPM1 mode. LDO and DCO on. SMCLK and ACLK running. MCLK stopped.
0	1	0	1	LDO and DCO on. SMCLK running. MCLK, ACLK stopped.
0	1	1	0	LPM2 or LPM3 mode (standby mode). LDO and DCO on. ACLK running. MCLK, SMCLK stopped.
0	1	1	1	LPM4 mode (off mode). LDO on. DCO off. MCLK, SMCLK, ACLK stopped.
1	1	1	1	LPM4.5 mode (shutdown mode). LDO and DCO off. MCLK, SMCLK, ACLK stopped.

4.2.4 Application Use Cases

4.2.4.1 Use Case 1: DCO Operation With Internal Resistor

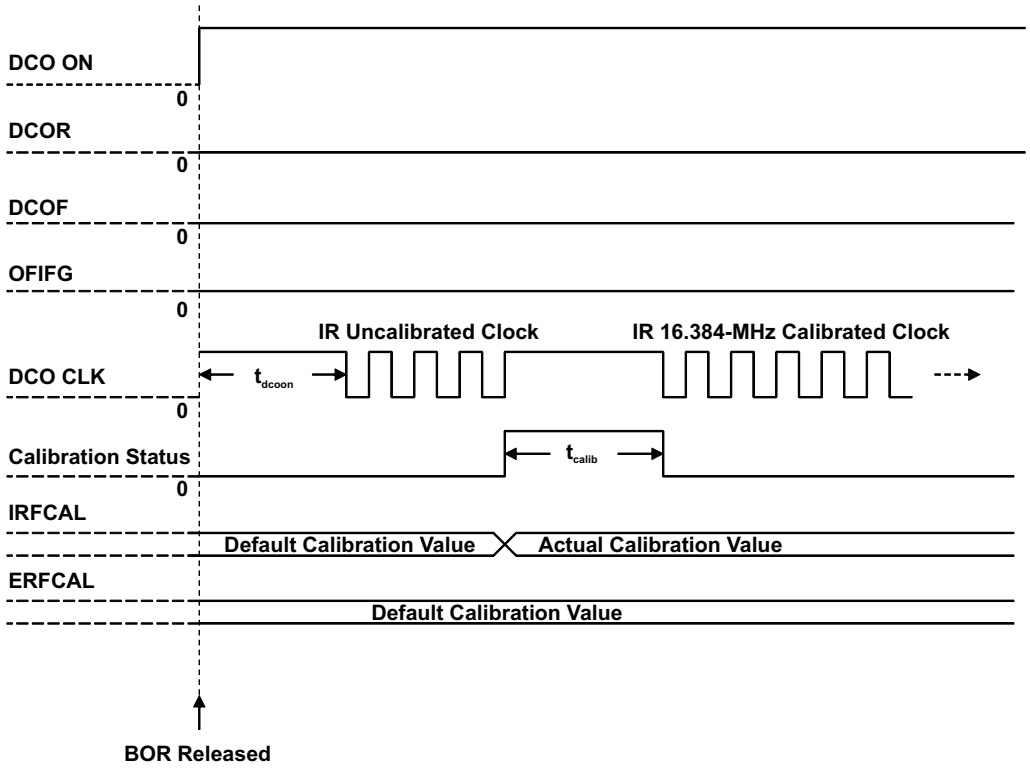


Figure 4-2. DCO Operation With Internal Resistor

Description

1. After BOR, the DCO starts in internal resistor mode.
2. The DCO produces a clock signal at an uncalibrated frequency after the specified DCO startup time.
3. User code loads internal resistor frequency calibration value from TLV table in flash information memory into the CSIRFCAL register.
4. The clock is stopped and the calibration process is initiated within the DCO.
5. When calibration is complete, the DCO produces a calibrated 16.384-MHz clock.
6. DCO fault detection circuitry is off in the internal resistor mode and, therefore, the DCOF and OFIFG bits are always 0.

4.2.4.2 Use Case 2: DCO Operation With External Resistor – No Fault

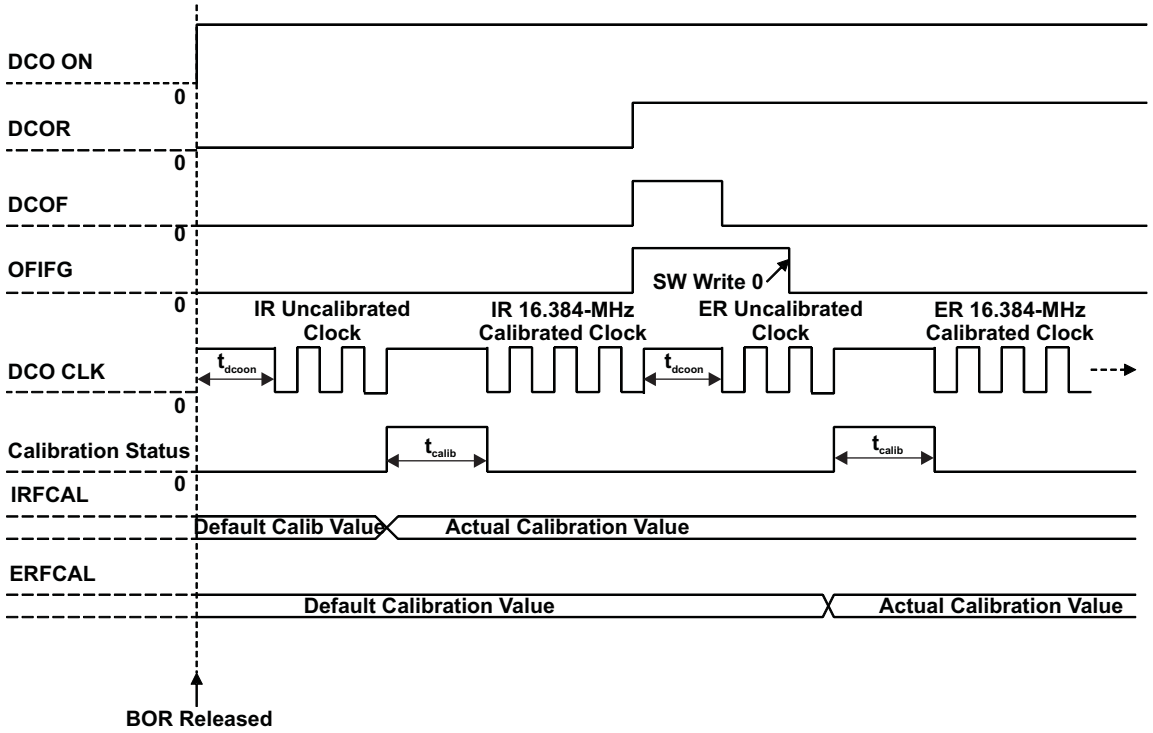


Figure 4-3. DCO Operation With External Resistor – No Fault

Description

1. After BOR, the DCO starts in internal resistor mode.
2. The DCO produces a clock signal at an uncalibrated frequency after the specified DCO startup time.
3. User code loads the internal resistor frequency calibration value from TLV table in flash information memory into the CSIRFCAL register.

   NOTE: Steps 3 through 5 make sure that the 16.384-MHz calibrated clock is available in the internal resistor mode during fail-safe operation.
4. The clock is stopped and the calibration process is initiated within the DCO when CSIRFCAL is written.
5. When calibration is complete, the DCO produces a calibrated 16.384-MHz clock with the internal resistor.
6. User code selects external resistor mode by setting the DCOR bit.
7. DCO fault signal is asserted immediately, which sets the DCOF and OFIFG bits.
8. Clock is stopped and fault detection circuitry is turned on within the DCO.
9. The DCO performs a health check of the resistor that is connected to the ROSC pin.

   In this use case, assume that the recommended resistor is connected properly on the ROSC pin.
10. After the DCO startup delay (t_{dcoon}), the DCO produces an uncalibrated clock in the external resistor mode.
11. The DCO fault signal is deasserted by the DCO, which clears the DCOF bit. The OFIFG bit remains set until cleared by software.
12. User code checks the DCOF bit and proceeds to load external resistor frequency calibration value from TLV table in flash information memory into the CSERFCAL register.
13. Clock is stopped and calibration process is initiated in the DCO when CSERFCAL is written.
14. When the calibration is complete, the DCO produces an 16.384-MHz clock with the external resistor.

4.2.4.3 Use Case 3: DCO Operation With External Resistor – Fault at Startup

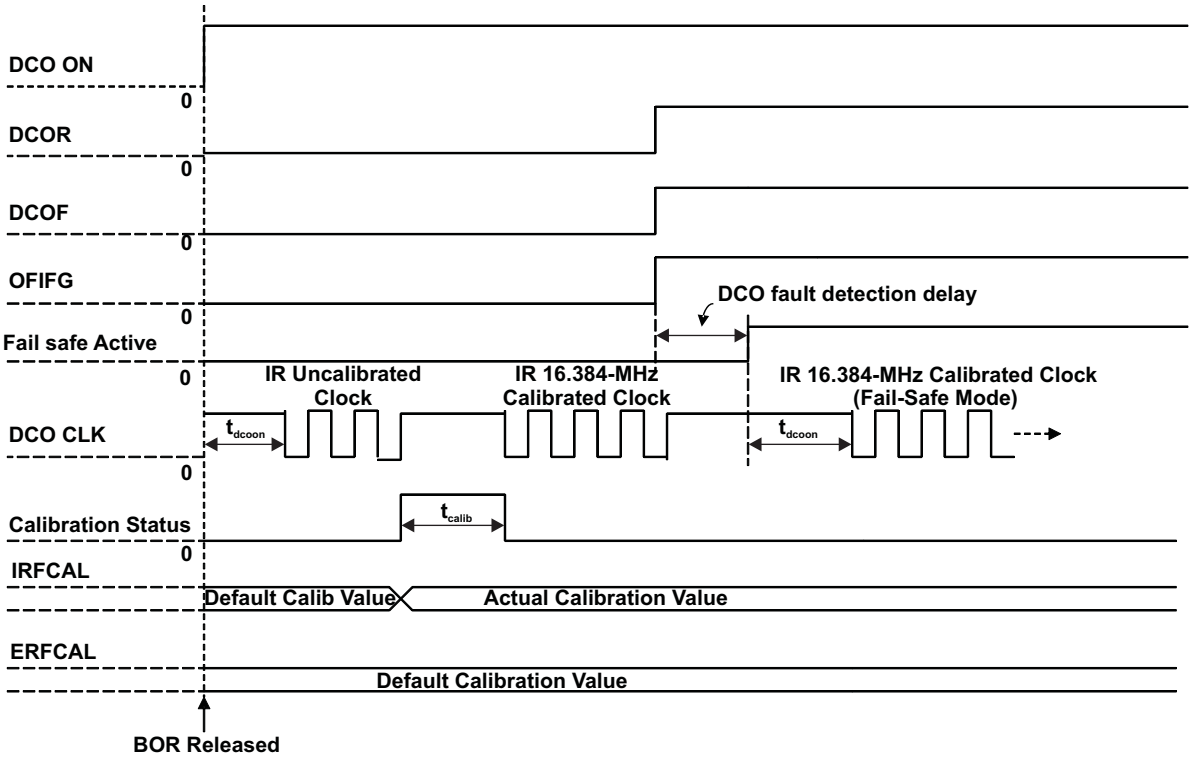


Figure 4-4. DCO Operation With External Resistor – Fault at Startup

Description

1. After BOR, the DCO starts in internal resistor mode.
2. The DCO produces a clock signal at an uncalibrated frequency after the specified DCO startup time.
3. User code loads the internal resistor frequency calibration value from TLV table in flash information memory into the CSIRFCAL register.

   NOTE: Steps 3 through 5 make sure that the 16.384-MHz calibrated clock is available in the internal resistor mode during fail-safe operation.
4. The clock is stopped and the calibration process is initiated within the DCO when CSIRFCAL is written.
5. When calibration is complete, the DCO produces an 16.384-MHz clock with the internal resistor.
6. User code selects external resistor mode by setting DCOR bit.
7. DCO fault signal is asserted immediately, which sets the DCOF and OFIFG bits.
8. Clock is stopped and fault detection circuitry is turned on within the DCO.
9. The DCO performs a health check of the resistor that is connected to the ROSC pin.

   In this use case, assume that the DCO finds a fault with the external resistor. For example, the resistor is not connected or is shorted to ground, or the resistance is far from the recommended value.
10. After the DCO fault detection delay, the DCO asserts the fault signal and sets the DCOF and OFIFG bits.
11. The fail-safe active signal is generated, which switches the DCO from the external resistor mode to the internal resistor mode.
12. After the DCO startup delay (t_{dcoon}), the DCO produces an 16.384-MHz calibrated clock in the internal resistor mode (because the internal resistor was already calibrated in step 3 before selecting the external resistor mode).
13. The DCO continuously asserts the fault signal, which keeps the DCOF and OFIFG bits set to 1.

4.2.4.4 Use Case 4: DCO Operation With External Resistor – Fault During Run Time

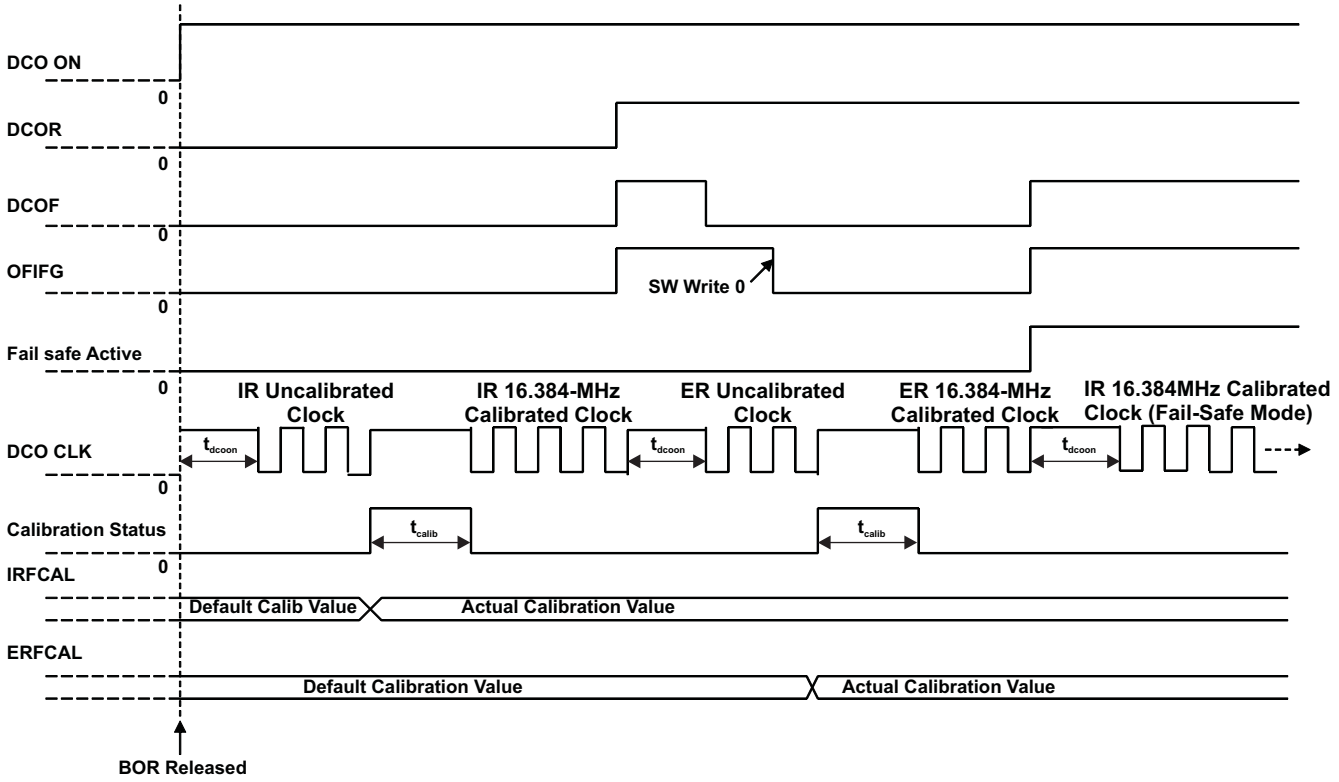


Figure 4-5. DCO Operation With External Resistor – Fault During Run Time

Description

1. After BOR, the DCO starts in internal resistor mode.
2. The DCO produces a clock signal at an uncalibrated frequency after the specified DCO startup time.
3. User code loads the internal resistor frequency calibration value from TLV table in flash information memory into the CSIRFCAL register.

   NOTE: Steps 3 through 5 make sure that the 16.384-MHz calibrated clock is available in the internal resistor mode during fail-safe operation.
4. The clock is stopped and the calibration process is initiated within the DCO when CSIRFCAL is written.
5. When calibration is complete, the DCO produces an 16.384-MHz clock with the internal resistor.
6. User code selects external resistor mode by setting DCOR bit.
7. DCO fault signal is asserted immediately, which sets the DCOF and OFIFG bits.
8. Clock is stopped and fault detection circuitry is turned on within the DCO.
9. The DCO performs a health check of the resistor that is connected to the ROSC pin.

   In this use case, assume that the recommended resistor is connected properly at the ROSC pin.
10. After the DCO startup delay (t_{dcoon}), the DCO produces an uncalibrated clock in the external resistor mode.
11. The DCO fault signal is deasserted by the DCO, which clears the DCOF bit. The OFIFG bit remains set until cleared by software.
12. User code checks for DCOF bit and proceeds to load external resistor frequency calibration value from TLV table in flash information memory into the CSERFCAL register.
13. Clock is stopped and calibration process is initiated within the DCO when CSERFCAL is written.
14. When the calibration is complete, the DCO produces an 16.384-MHz clock with external resistor.
15. DCO fault detection circuit is continuously active in the DCO and checks the health of the external

resistor during operation.

16. In this example, the DCO detects a fault with the external resistor during run time and sets the DCOF and OFIFG bits.
17. The fail-safe active signal is generated, which switches the DCO from the external resistor mode to the internal resistor mode.
18. After the DCO startup delay (t_{dcoon}), the DCO produces an 16.384-MHz calibrated clock in the internal resistor mode (because the internal resistor was already calibrated in step 3 before selecting external resistor mode).
19. The DCO continuously asserts the fault signal, which keeps the DCOF and OFIFG bits set to 1.

4.2.4.5 Use Case 5: DCO Operation in Bypass Mode

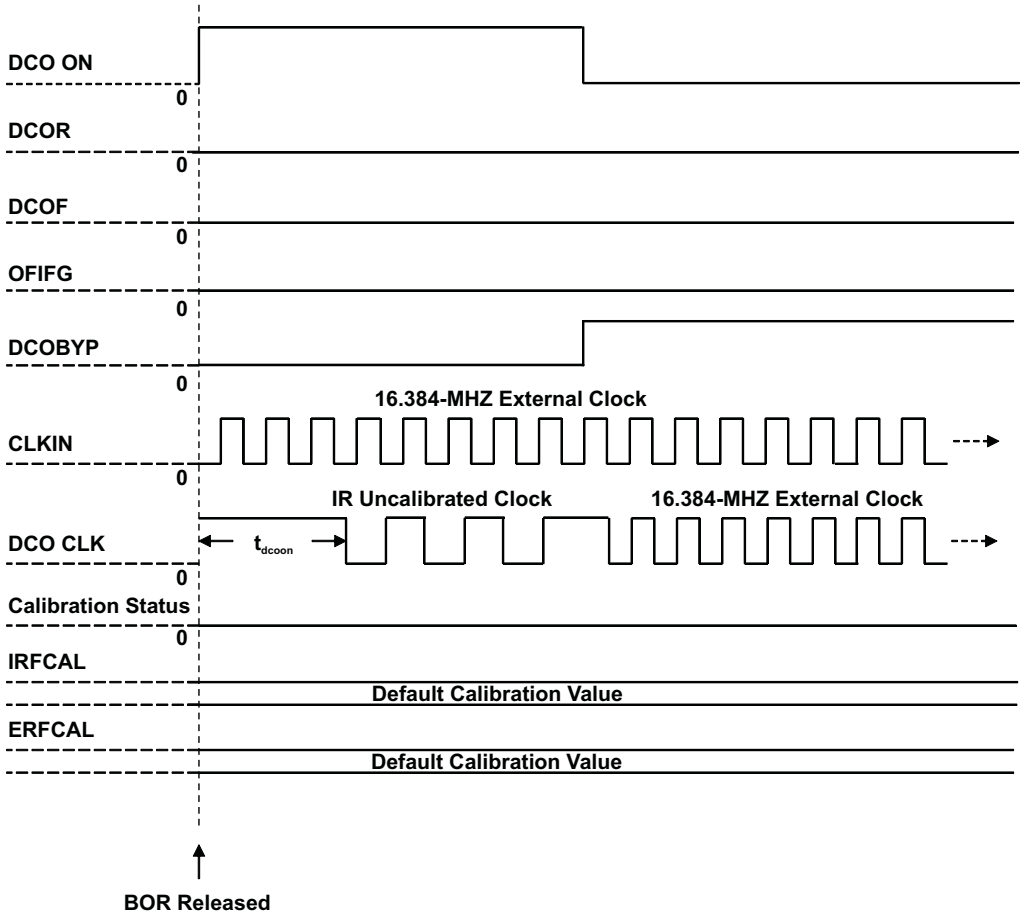


Figure 4-6. DCO Operation in Bypass Mode

Description

1. After BOR, the DCO starts in internal resistor mode.
2. The DCO produces a clock signal at an uncalibrated frequency after the specified DCO startup time.
3. In this example, consider that 16.384-MHz external clock is provided at CLKIN pin of the device.
4. Once the DCOBYP bit in CSCTL0 register is set together with the necessary IO programming, DCO gets powered down if there is no active peripheral clock request.
5. Clock system output switches from DCO IR mode uncalibrated clock to 16.384-MHz external clock provided at CLKIN pin.
6. DCOF and OFIFG bits remain cleared throughout after switching to bypass mode.
7. Writing any values to calibration registers in bypass mode will not take any effect since DCO is powered down.

4.3 Clock System Registers

Clock System registers are listed in Table 4-2.

Table 4-2. Clock System Registers

Address	Acronym	Register Name	Type	Access	Reset	Section
0050h	CSCTL0	Clock System Control 0	Read/Write	Byte	00h	Section 4.3.1
0051h	CSCTL1	Clock System Control 1	Read/Write	Byte	00h	Section 4.3.2
0052h	CSIRFCAL	Clock System Internal Resistor Frequency Calibration	Read/Write	Byte	00h	Section 4.3.3
0053h	CSIRTCAL	Clock System Internal Resistor Temperature Calibration	Read/Write	Byte	00h	Section 4.3.4
0054h	CSERFCAL	Clock System External Resistor Frequency Calibration	Read/Write	Byte	00h	Section 4.3.5
0055h	CSERTCAL	Clock System External Resistor Temperature Calibration	Read/Write	Byte	00h	Section 4.3.6

4.3.1 CSCTL0 Register (address = 0050h) [reset = 00h]

Clock System Control 0 Register

Figure 4-7. CSCTL0 Register

7	6	5	4	3	2	1	0
DCOF	Reserved					DCOBYP	DCOR
r-0	r0	r0	r0	r0	r0	rw-0	rw-0

Table 4-3. CSCTL0 Register Description

Bit	Field	Type	Reset	Description
7	DCOF	R	0h	DCO fault flag. This bit is set when the fault is detected when DCO operates in the external resistor mode. DCO switches to internal resistor mode as fail-safe feature. 0b = No DCO fault 1b = DCO fault detected
6-2	Reserved	R	0h	Reserved. Always read as 0.
1	DCOBYP	RW	0h	DCO bypass mode. When this bit is set, the DCO is bypassed. An external digital clock can be input on the CLKIN pin. 0b = DCO bypass mode not selected 1b = DCO bypass mode selected
0	DCOR	RW	0h	DCO resistor select. This bit selects internal or external resistor for DCO operation. 0b = DCO operation with internal resistor 1b = DCO operation with external resistor

4.3.2 CSCTL1 Register (address = 0051h) [reset = 00h]

Clock System Control 1 Register

Figure 4-8. CSCTL1 Register

7	6	5	4	3	2	1	0
Reserved	SMCLKDIV			Reserved	MCLKDIV		
r0	rw-0	rw-0	rw-0	r0	rw-0	rw-0	rw-0

Table 4-4. CSCTL1 Register Description

Bit	Field	Type	Reset	Description
7	Reserved	R	0h	Reserved. Always read as 0.
6-4	SMCLKDIV	RW	0h	SMCLK divider bits. These bits select the divider setting for SMCLK. 000b = Divide by 1 001b = Divide by 2 010b = Divide by 4 011b = Divide by 8 100b = Divide by 16 101b = Reserved 110b = Reserved 111b = Reserved
3	Reserved	R	0h	Reserved. Always read as 0.
2-0	MCLKDIV	RW	0h	MCLK divider bits. These bits select the divider setting for MCLK. 000b = Divide by 1 001b = Divide by 2 010b = Divide by 4 011b = Divide by 8 100b = Divide by 16 101b = Reserved 110b = Reserved 111b = Reserved

4.3.3 CSIRFCAL Register (address = 0052h) [reset = 00h]

Clock System Internal Resistor Frequency Calibration Register

Figure 4-9. CSIRFCAL Register

7	6	5	4	3	2	1	0
IRFCAL							
rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]

Table 4-5. CSIRFCAL Register Description

Bit	Field	Type	Reset	Description
7-0	IRFCAL	RW	0h	DCO internal resistor frequency calibration value. These bits are reset on BOR.

4.3.4 CSIRTCAL Register (address = 0053h) [reset = 00h]

Clock System Internal Resistor Temperature Calibration Register

Figure 4-10. CSIRTCAL Register

7	6	5	4	3	2	1	0
IRTCAL							
rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]

Table 4-6. CSIRTCAL Register Description

Bit	Field	Type	Reset	Description
7-0	IRTCAL	RW	0h	DCO internal resistor temperature calibration value. These bits are reset on BOR.

4.3.5 CSERFCAL Register (address = 0054h) [reset = 00h]

Clock System External Resistor Frequency Calibration Register

Figure 4-11. CSERFCAL Register

7	6	5	4	3	2	1	0
ERFCAL							
rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]

Table 4-7. CSERFCAL Register Description

Bit	Field	Type	Reset	Description
7-0	ERFCAL	RW	0h	DCO external resistor frequency calibration value. These bits are reset on BOR.

4.3.6 CSERTCAL Register (address = 0055h) [reset = 00h]

Clock System External Resistor Temperature Calibration Register

Figure 4-12. CSERTCAL Register

7	6	5	4	3	2	1	0
ERTCAL							
rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]

Table 4-8. CSERTCAL Register Description

Bit	Field	Type	Reset	Description
7-0	ERTCAL	RW	0h	DCO external resistor temperature calibration value. These bits are reset on BOR.

Flash Memory Controller (FCTL)

This chapter describes the operation of the flash memory controller.

Topic **Page**

5.1 Flash Memory Introduction

The flash memory is bit-, byte-, and word-addressable and programmable. The flash memory module has an integrated controller that controls programming and erase operations. The controller has three registers, a timing generator, and a voltage generator to supply program and erase voltages.

Flash memory features include:

- Internal programming voltage generation
- Bit, byte, or word programmable
- Segment erase and mass erase

Figure 5-1 shows the block diagram of the flash memory and controller.

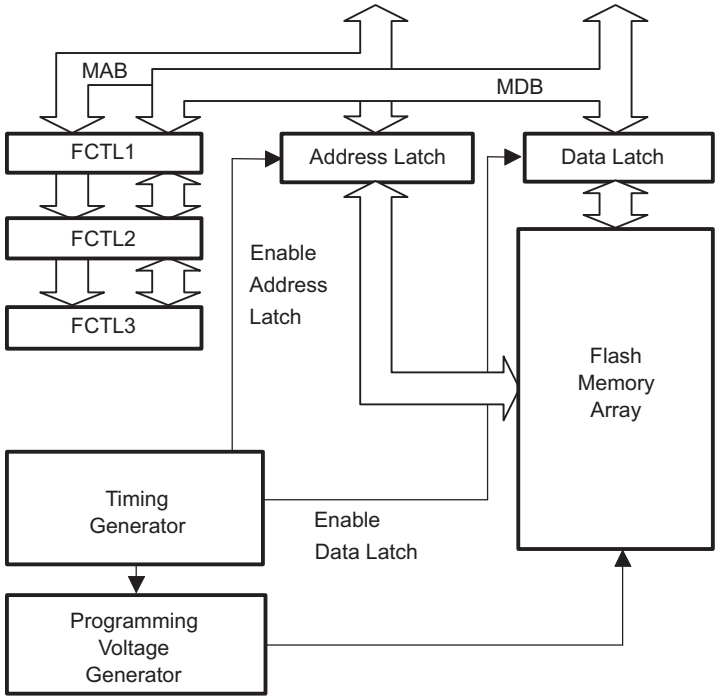


Figure 5-1. Flash Memory Module Block Diagram

5.2 Flash Memory Segmentation

The flash memory is partitioned into segments. Single bits, bytes, or words can be written to flash memory, but the segment is the smallest size of flash memory that can be erased.

The flash memory is partitioned into main and information memory sections. There is no difference in the operation of the main and information memory sections. Code or data can be located in either section. But the physical addresses are different for main and information memory.

The main memory has one or more 1KB segments. The information memory has one segment of size 1KB. See the device-specific data sheet for the complete memory map of a device.

The segments are further divided into blocks.

Figure 5-2 shows the flash segmentation using an example of 16KB of flash that has sixteen main segments and one information segment.

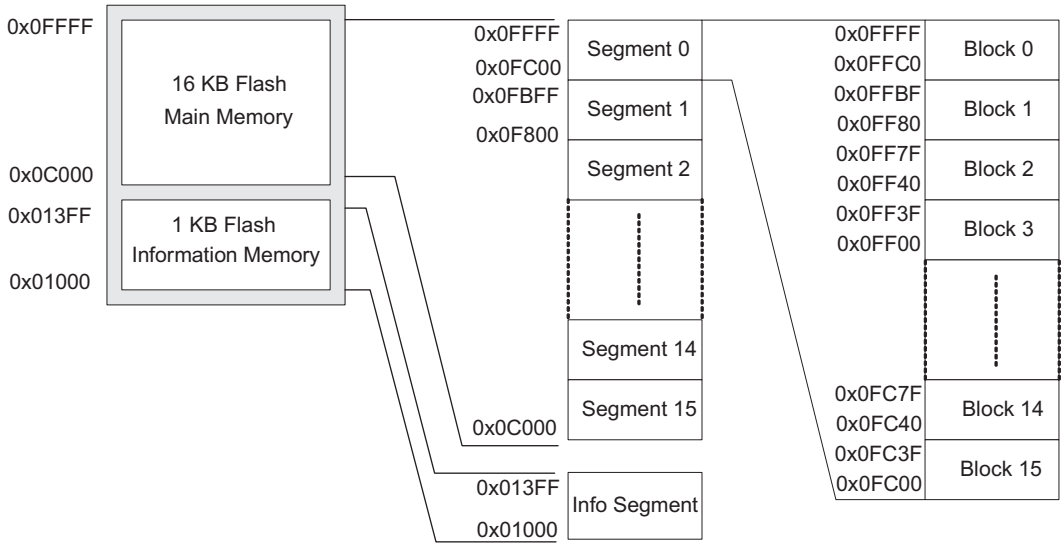


Figure 5-2. Flash Memory Segments, 16KB Example

5.2.1 Information Memory Segment

The flash information memory segment is locked with LOCKSEG bit. When LOCKSEG = 1, the information memory cannot be written or erased, and it is protected from erasure during a mass erase or production programming. When LOCKSEG = 0, the information memory can be erased and written as any other flash memory segment, and it is erased during a mass erase or production programming.

The state of the LOCKSEG bit is toggled when a 1 is written to it. Writing a 0 to LOCKSEG has no effect. This allows existing flash programming routines to be used unchanged.

```
; Unlock Information Memory Segment
   BIT   #LOCKSEG,&FCTL3              ; Test LOCKSEG
   JZ    INFOSEG_UNLOCKED             ; Already unlocked?
   MOV   #FWKEY+LOCKSEG,&FCTL3        ; No, unlock Info Segment
INFOSEG_UNLOCKED                      ; Yes, continue
; Info Segment is unlocked

; Lock Info Segment
   BIT   #LOCKSEG,&FCTL3              ; Test LOCKSEG
   JNZ   INFOSEG_LOCKED               ; Already locked?
   MOV   #FWKEY+LOCKSEG,&FCTL3        ; No, lock Info Segment
INFOSEG_LOCKED                        ; Yes, continue
; Info Segment is locked
```

5.3 Flash Memory Operation

The default mode of the flash memory is read mode. In read mode, the flash memory is not being erased or written, the flash timing generator and voltage generator are off, and the memory operates identically to ROM.

The flash memory is in-system programmable (ISP) without the need for additional external voltage. The CPU can program its own flash memory. The flash memory write and erase modes are selected with the BLKWRT, WRT, MERAS, and ERASE bits and are:

- Byte or word write
- Block write
- Segment erase
- Mass erase (all main memory segments)
- All erase (all segments)

Reading or writing to flash memory while it is being programmed or erased is prohibited. If CPU execution is required during the write or erase, the code to be executed must be in RAM. Any flash update can be initiated from within flash memory or RAM.

5.3.1 Flash Memory Timing Generator

Write and erase operations are controlled by the flash timing generator shown in Figure 5-3. The flash timing generator operating frequency, f_{FTG}, must be in the range from approximately 257 kHz to approximately 476 kHz (see device-specific data sheet).

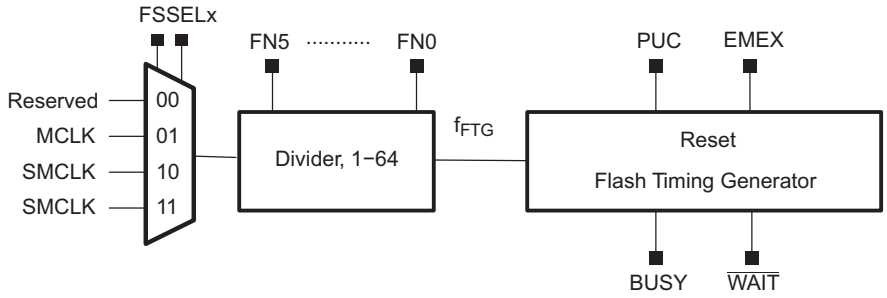


Figure 5-3. Flash Memory Timing Generator Block Diagram

5.3.1.1 Flash Timing Generator Clock Selection

The flash timing generator can be sourced from SMCLK or MCLK. The selected clock source should be divided using the FNx bits to meet the frequency requirements for f_{FTG}. If the f_{FTG} frequency deviates from the specification during the write or erase operation, the result of the write or erase may be unpredictable, or the flash memory may be stressed above the limits of reliable operation.

While a write or erase operation is active, the selected clock source cannot be disabled by putting the device into a low-power mode. The selected clock source remains active until the operation is completed before being disabled.

5.3.2 Erasing Flash Memory

The erased level of a flash memory bit is 1. Each bit can be programmed from 1 to 0 individually but to reprogram from 0 to 1 requires an erase cycle. The smallest amount of flash that can be erased is a segment. There are three erase modes selected with the ERASE and MERAS bits listed in Table 5-1.

Table 5-1. Erase Modes

MERAS	ERASE	Erase Mode
0	1	Segment erase
1	0	Mass erase (all main memory segments)
1	1	LOCKSEG = 0: Erase main and information flash memory. LOCKSEG = 1: Erase only main flash memory.

Any erase is initiated by a dummy write into the address range to be erased. The dummy write starts the flash timing generator and the erase operation. Figure 5-4 shows the erase cycle timing. The BUSY bit is set immediately after the dummy write and remains set throughout the erase cycle. BUSY, MERAS, and ERASE are automatically cleared when the cycle completes. The erase cycle timing is not dependent on the amount of flash memory present on a device. Erase cycle times are equivalent for all MSP430i devices.

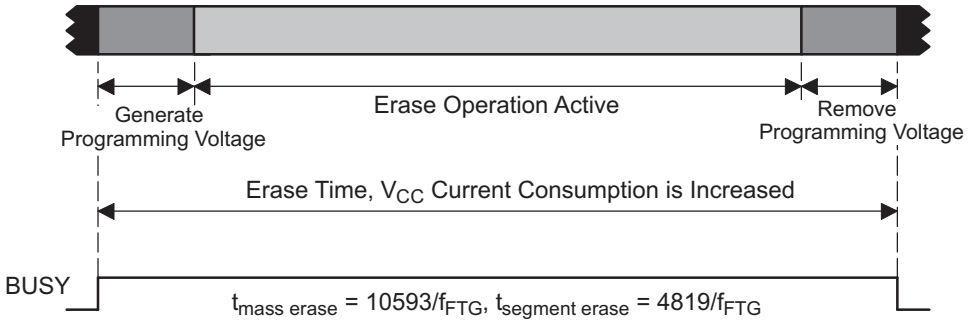


Figure 5-4. Erase Cycle Timing

A dummy write to an address not in the range to be erased does not start the erase cycle, does not affect the flash memory, and is not flagged in any way. This errant dummy write is ignored.

5.3.2.1 Initiating an Erase From Within Flash Memory

Any erase cycle can be initiated from within flash memory or from RAM. When a flash segment erase operation is initiated from within flash memory, all timing is controlled by the flash controller, and the CPU is held while the erase cycle completes. After the erase cycle completes, the CPU resumes code execution with the instruction following the dummy write.

When initiating an erase cycle from within flash memory, it is possible to erase the code needed for execution after the erase. If this occurs, CPU execution is unpredictable after the erase cycle.

Figure 5-5 shows the flow to initiate an erase from flash.

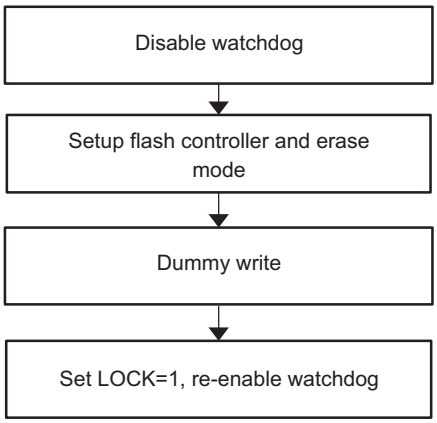


Figure 5-5. Erase Cycle From Within Flash Memory

```
; Segment Erase from flash. 514 kHz < SMCLK < 952 kHz
; Assumes ACCVIE = NMIIE = OFIE = 0.
   MOV      #WDTPW+WDTHOLD,&WDTCTL     ; Disable WDT
   MOV      #FWKEY+FSSEL1+FN0,&FCTL2   ; SMCLK/2
   MOV      #FWKEY, &FCTL3             ; Clear LOCK
   MOV      #FWKEY+ERASE, &FCTL1       ; Enable segment erase
   CLR      &0F810h                    ; Dummy write, erase S1
   MOV      #FWKEY+LOCK, &FCTL3        ; Done, set LOCK
   ...                                 ; Re-enable WDT?
```

5.3.2.2 Initiating an Erase From RAM

Any erase cycle may be initiated from RAM. In this case, the CPU is not held and can continue to execute code from RAM. The BUSY bit must be polled to determine the end of the erase cycle before the CPU can access any flash address again. If a flash access occurs while BUSY = 1, it is an access violation, ACCVIFG is set, and the erase results are unpredictable.

Figure 5-6 shows the flow to initiate an erase from flash from RAM.

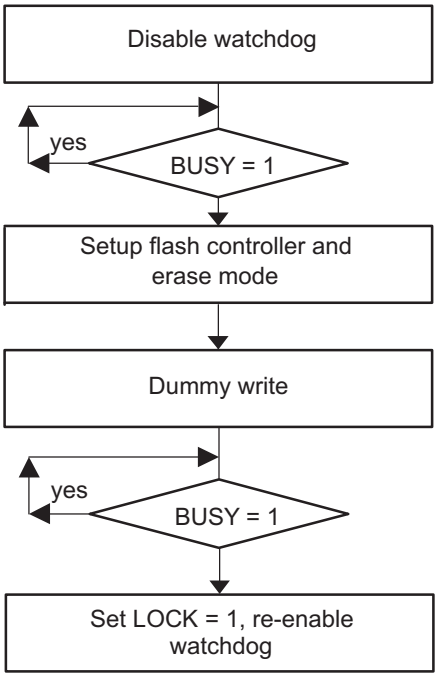


Figure 5-6. Erase Cycle from Within RAM

```
; Segment Erase from RAM. 514 kHz < SMCLK < 952 kHz
; Assumes ACCVIE = NMIIE = OFIE = 0.
     MOV   #WDTPW+WDTHOLD,&WDTCTL      ; Disable WDT
L1   BIT   #BUSY, &FCTL3               ; Test BUSY
     JNZ   L1                          ; Loop while busy
     MOV   #FWKEY+FSSEL1+FN0, &FCTL2   ; SMCLK/2
     MOV   #FWKEY&FCTL3                ; Clear LOCK
     MOV   #FWKEY+ERASE, &FCTL1        ; Enable erase
     CLR   &0F810h                     ; Dummy write, erase S1
L2   BIT   #BUSY, &FCTL3               ; Test BUSY
     JNZ   L2                          ; Loop while busy
     MOV   #FWKEY+LOCK&FCTL3           ; Done, set LOCK
     ...                               ; Re-enable WDT?
```

5.3.3 Writing Flash Memory

Table 5-2 lists the write modes, which are selected by the WRT and BLKWRT bits.

Table 5-2. Write Modes

BLKWRT	WRT	Write Mode
0	1	Byte or word write
1	1	Block write

Both write modes use a sequence of individual write instructions, but using the block write mode is approximately twice as fast as byte/word mode, because the voltage generator remains on for the complete block write. Any instruction that modifies a destination can be used to modify a flash location in either byte or word mode or block write mode. A flash word (low + high byte) must not be written more than twice between erasures. Otherwise, damage can occur.

The BUSY bit is set while a write operation is active and cleared when the operation completes. If the write operation is initiated from RAM, the CPU must not access flash while BUSY = 1. Otherwise, an access violation occurs, ACCVIFG is set, and the flash write is unpredictable.

5.3.3.1 Byte or Word Write

A byte or word write operation can be initiated from within flash memory or from RAM. When initiating from within flash memory, all timing is controlled by the flash controller, and the CPU is held while the write completes. After the write completes, the CPU resumes code execution with the instruction following the write. The byte or word write timing is shown in Figure 5-7.

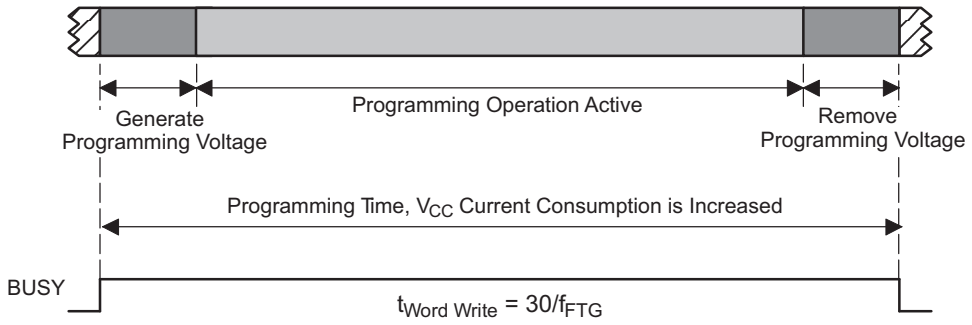


Figure 5-7. Byte or Word Write Timing

When a byte or word write is executed from RAM, the CPU continues to execute code from RAM. The BUSY bit must be zero before the CPU accesses flash again, otherwise an access violation occurs, ACCVIFG is set, and the write result is unpredictable.

In byte/word mode, the internally generated programming voltage is applied to the complete 64-byte block, each time a byte or word is written, for 27 of the 30 f_{FTG} cycles. With each byte or word write, the amount of time the block is subjected to the programming voltage accumulates. The cumulative programming time, t_{CPT}, must not be exceeded for any block. If the cumulative programming time is met, the block must be erased before performing any further writes to any address within the block. See the device-specific data sheet for specifications.

5.3.3.2 Initiating a Byte or Word Write From Flash Memory

Figure 5-8 shows the flow to initiate a byte or word write from flash.

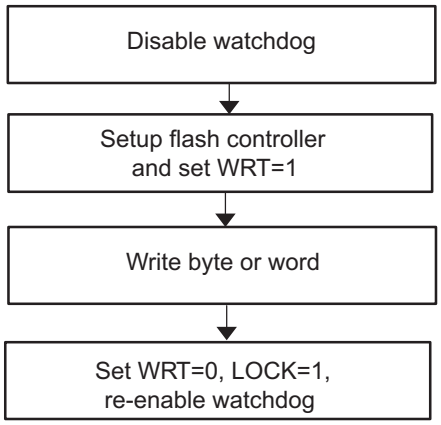


Figure 5-8. Initiating a Byte or Word Write From Flash

```
; Byte/word write from flash. 514 kHz < SMCLK < 952 kHz
; Assumes 0FF1Eh is already erased
; Assumes ACCVIE = NMIIE = OFIE = 0.
   MOV   #WDTPW+WDTHOLD,&WDTCTL     ; Disable WDT
   MOV   #FWKEY+FSSEL1+FN0,&FCTL2   ; SMCLK/2
   MOV   #FWKEY,&FCTL3              ; Clear LOCK
   MOV   #FWKEY+WRT,&FCTL1          ; Enable write
   MOV   #0123h,&0FF1Eh             ; 0123h    -> 0FF1Eh
   MOV   #FWKEY,&FCTL1              ; Done. Clear WRT
   MOV   #FWKEY+LOCK,&FCTL3         ; Set LOCK
   ...                              ; Re-enable WDT?
```

5.3.3.3 Initiating a Byte or Word Write From RAM

Figure 5-9 shows the flow to initiate a byte or word write from RAM.

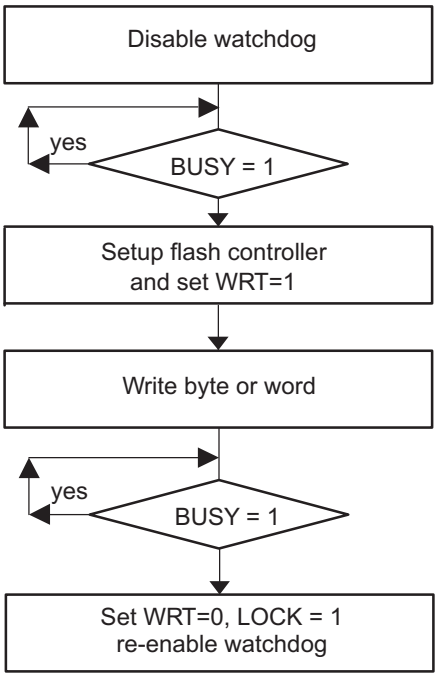


Figure 5-9. Initiating a Byte or Word Write from RAM

```
; Byte/word write from RAM. 514 kHz < SMCLK < 952 kHz
; Assumes 0FF1Eh is already erased
; Assumes ACCVIE = NMIIE = OFIE = 0.
     MOV   #WDTPW+WDTHOLD,&WDTCTL     ; Disable WDT
L1   BIT   #BUSY,&FCTL3               ; Test BUSY
     JNZ   L1                         ; Loop while busy
     MOV   #FWKEY+FSSEL1+FN0,&FCTL2   ; SMCLK/2
     MOV   #FWKEY,&FCTL3              ; Clear LOCK
     MOV   #FWKEY+WRT,&FCTL1          ; Enable write
     MOV   #0123h,&0FF1Eh             ; 0123h -> 0FF1Eh
L2   BIT   #BUSY,&FCTL3               ; Test BUSY
     JNZ   L2                         ; Loop while busy
     MOV   #FWKEY,&FCTL1              ; Clear WRT
     MOV   #FWKEY+LOCK,&FCTL3         ; Set LOCK
     ...                              ; Re-enable WDT?
```

5.3.3.4 Block Write

The block write can be used to accelerate the flash write process when many sequential bytes or words need to be programmed. The flash programming voltage remains on for the duration of writing the 64-byte block. The cumulative programming time t_{CPT} must not be exceeded for any block during a block write.

A block write cannot be initiated from within flash memory. The block write must be initiated from RAM only. The BUSY bit remains set throughout the duration of the block write. The WAIT bit must be checked between writing each byte or word in the block. When WAIT is set the next byte or word of the block can be written. When writing successive blocks, the BLKWRT bit must be cleared after the current block is complete. BLKWRT can be set initiating the next block write after the required flash recovery time given by t_{end}. BUSY is cleared following each block write completion indicating that the next block can be written. Figure 5-10 shows the block write timing.

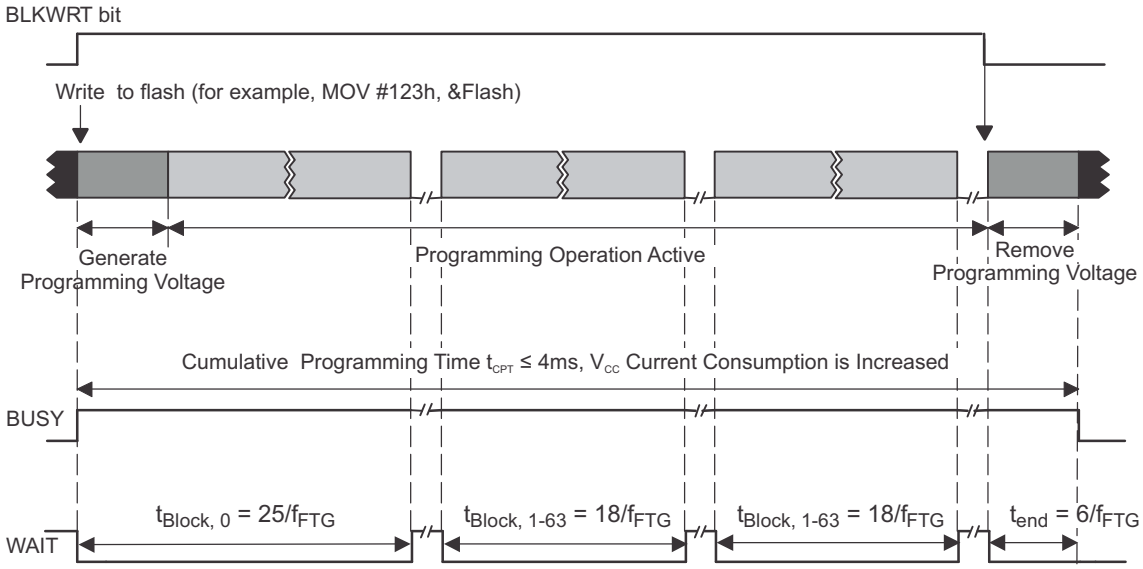


Figure 5-10. Block-Write Cycle Timing

5.3.3.5 Block Write Flow and Example

A block write flow is shown in Figure 5-11 and the following example.

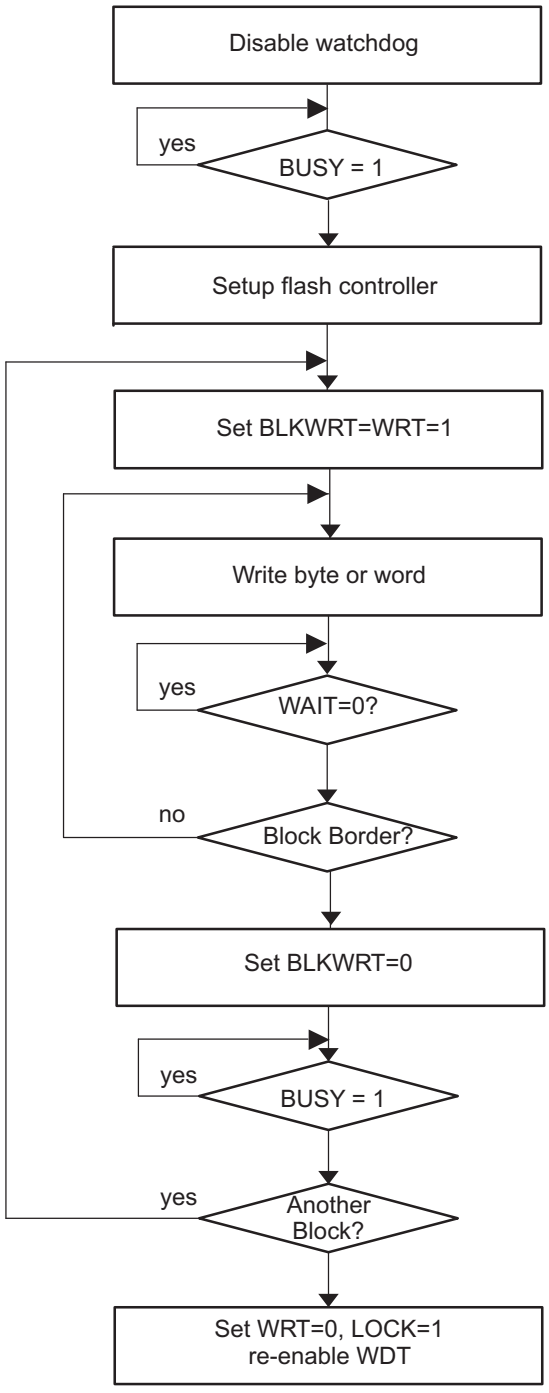


Figure 5-11. Block Write Flow

```
; Write one block starting at 0F000h.
; Must be executed from RAM, Assumes Flash is already erased.
; 514 kHz < SMCLK < 952 kHz
; Assumes ACCVIE = NMIIE = OFIE = 0.
     MOV    #32,R5                    ; Use as write counter
     MOV    #0F000h,R6                ; Write pointer
     MOV    #WDTPW+WDTHOLD,&WDTCTL    ; Disable WDT
L1   BIT    #BUSY,&FCTL3              ; Test BUSY
     JNZ    L1                        ; Loop while busy
     MOV    #FWKEY+FSSEL1+FN0,&FCTL2  ; SMCLK/2
     MOV    #FWKEY,&FCTL3             ; Clear LOCK
     MOV    #FWKEY+BLKWRT+WRT,&FCTL1  ; Enable block write
L2   MOV    Write_Value,0(R6)         ; Write location
L3   BIT    #WAIT,&FCTL3              ; Test WAIT
     JZ     L3                        ; Loop while WAIT = 0
     INCD   R6                        ; Point to next word
     DEC    R5                        ; Decrement write counter
     JNZ    L2                        ; End of block?
     MOV    #FWKEY,&FCTL1             ; Clear WRT,BLKWRT
L4   BIT    #BUSY,&FCTL3              ; Test BUSY
     JNZ    L4                        ; Loop while busy
     MOV    #FWKEY+LOCK,&FCTL3        ; Set LOCK
     ...                              ; Re-enable WDT if needed
```

5.3.4 Flash Memory Access During Write or Erase

When any write or any erase operation is initiated from RAM and while BUSY = 1, the CPU may not read from or write to any flash location. Otherwise, an access violation occurs, ACCVIFG is set, and the result is unpredictable. Also if a write to flash is attempted with WRT = 0, the ACCVIFG interrupt flag is set, and the flash memory is unaffected.

When a byte or word write or any erase operation is initiated from within flash memory, the flash controller returns op-code 03FFFh to the CPU at the next instruction fetch. Op-code 03FFFh is the JMP PC instruction. This causes the CPU to loop until the flash operation is finished. When the operation is finished and BUSY = 0, the flash controller allows the CPU to fetch the proper op-code, and program execution resumes.

Table 5-3 lists the flash access conditions while BUSY = 1.

Table 5-3. Flash Access While BUSY = 1

Flash Operation	Flash Access	WAIT	Result
Any erase or byte or word write	Read	0	ACCVIFG = 0. 03FFFh is the value read.
	Write	0	ACCVIFG = 1. Write is ignored.
	Instruction fetch	0	ACCVIFG = 0. CPU fetches 03FFFh. This is the JMP PC instruction.
Block write	Any	0	ACCVIFG = 1, LOCK = 1
	Read	1	ACCVIFG = 0. 03FFFh is the value read.
	Write	1	ACCVIFG = 0. Write is written.
	Instruction fetch	1	ACCVIFG = 1, LOCK = 1

When flash all erase (MERAS = ERASE = 1) is initiated by application code executing from flash information memory with LOCKSEG = 1, the access violation flag (ACCVIFG) is set, flash main memory is erased, and flash information memory content is retained.

Interrupts are automatically disabled during any flash operation. After the flash operation has completed, interrupts are automatically re-enabled. Any interrupt that occurred during the operation has its associated flag set and generates an interrupt request when re-enabled.

The watchdog timer (in watchdog mode) should be disabled before a flash erase cycle. A reset aborts the erase and the result is unpredictable. After the erase cycle has completed, the watchdog may be re-enabled.

5.3.5 Stopping a Write or Erase Cycle

Any write or erase operation can be stopped before its normal completion by setting the emergency exit bit EMEX. Setting the EMEX bit stops the active operation immediately and stops the flash controller. All flash operations cease, the flash returns to read mode, and all bits in the FCTL1 register are reset. The result of the intended operation is unpredictable.

5.3.6 Configuring and Accessing the Flash Memory Controller

The FCTLx registers are 16-bit password-protected read/write registers. Any read or write access must use word instructions and write accesses must include the write password 0A5h in the upper byte. Any write to any FCTLx register with any value other than 0A5h in the upper byte is a security key violation, sets the KEYV flag, and triggers a PUC system reset. Any read of any FCTLx registers reads 096h in the upper byte.

Any write to FCTL1 during an erase or a byte or word write operation is an access violation and sets ACCVIFG. Writing to FCTL1 is allowed in block write mode when WAIT = 1, but writing to FCTL1 in block write mode when WAIT = 0 is an access violation and sets ACCVIFG.

Any write to FCTL2 when the BUSY = 1 is an access violation.

Any FCTLx register may be read when BUSY = 1. A read does not cause an access violation.

5.3.7 Flash Memory Controller Interrupts

The flash controller has two interrupt sources, KEYV and ACCVIFG. ACCVIFG is set when an access violation occurs. When the ACCVIE bit is re-enabled after a flash write or erase, a set ACCVIFG flag generate an interrupt request. ACCVIFG sources the NMI interrupt vector, so it is not necessary for GIE to be set for ACCVIFG to request an interrupt. ACCVIFG may also be checked by software to determine if an access violation occurred. ACCVIFG must be reset by software.

The key violation flag KEYV is set when any of the flash control registers are written with an incorrect password. When this occurs, a PUC is generated immediately and the device is reset.

5.3.8 Programming Flash Memory Devices

There are two options for programming an MSP430i flash device. Both of these options support in-system programming:

- Program via JTAG
- Program via a custom solution

5.3.8.1 Programming Flash Memory via JTAG

MSP430i devices can be programmed via the JTAG port. The JTAG interface requires four signals, ground and, optionally, V_{CC} and $\overline{RST}$/NMI.

The application can disable JTAG port by programming SYSJTAGDIS register in SYS module. In this case, the device is secured and further access to the device via JTAG is not possible. For details, see the *MSP430 Programming Via the JTAG Interface User's Guide* (SLAU320).

5.3.8.2 Programming Flash Memory via a Custom Solution

The ability of the MSP430i CPU to write to its own flash memory allows for in-system and external custom programming solutions as shown in Figure 5-12. The user can choose to provide data to the MSP430i through any means available (for example, UART or SPI). User-developed software can receive the data and program the flash memory. Because this type of solution is developed by the user, it can be completely customized to fit the application needs for programming, erasing, or updating the flash memory.

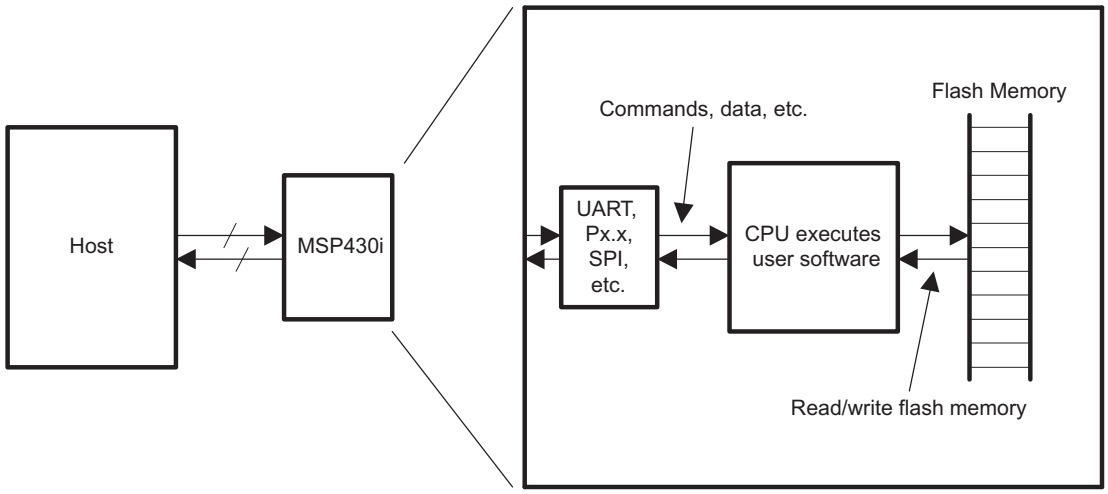


Figure 5-12. User-Developed Programming Solution

5.4 Flash Memory Registers

Table 5-4 lists the flash memory registers.

Table 5-4. FCTL Registers

Address	Acronym	Register Name	Type	Access	Reset	Section
0128h	FCTL1	Flash memory control 1 register	Read/write	Word	9600h	Section 5.4.1
012Ah	FCTL2	Flash memory control 2 register	Read/write	Word	9642h	Section 5.4.2
012Ch	FCTL3	Flash memory control 3 register	Read/write	Word	9658h	Section 5.4.3
0000h	IE1	Interrupt Enable 1 register	Read/write	Byte	0000h	Section 5.4.4

5.4.1 FCTL1 Register (address = 0128h) [reset = 9600h]

Flash Memory Control 1 Register

Figure 5-13. FCTL1 Register

15	14	13	12	11	10	9	8
FWKEY							
rw-1	rw-0	rw-0	rw-1	rw-0	rw-1	rw-1	rw-0
7	6	5	4	3	2	1	0
BLKWRT	WRT	Reserved			MERAS	ERASE	Reserved
rw-0	rw-0	r0	r0	r0	rw-0	rw-0	r0

Table 5-5. FCTL1 Register Description

Bit	Field	Type	Reset	Description
15-8	FWKEY	RW	96h	FCTLx password. Always read as 096h. Must be written as 0A5h or a PUC is generated.
7	BLKWRT	RW	0h	Block write mode. WRT must also be set for block write mode. BLKWRT is automatically reset when EMEX is set. 0b = Block-write mode is off 1b = Block-write mode is on
6	WRT	RW	0h	Write. This bit is used to select any write mode. WRT is automatically reset when EMEX is set. 0b = Write mode is off 1b = Write mode is on
5-3	Reserved	R	0h	Reserved. Always reads as 0.
2	MERAS	RW	0h	Mass erase. Used together with ERASE to select the erase mode. MERAS and ERASE are automatically reset when EMEX is set. MERAS = 0, ERASE = 0: No erase MERAS = 0, ERASE = 1: Erase individual segment only MERAS = 1, ERASE = 0: Erase all main memory segments MERAS = 1, ERASE = 1: If LOCKSEG = 0, then erase main and information flash memory. If LOCKSEG = 1, then erase only main flash memory.
1	ERASE	RW	0h	Erase. Used together with MERAS to select the erase mode. MERAS and ERASE are automatically reset when EMEX is set. MERAS = 0, ERASE = 0: No erase MERAS = 0, ERASE = 1: Erase individual segment only MERAS = 1, ERASE = 0: Erase all main memory segments MERAS = 1, ERASE = 1: If LOCKSEG = 0, then erase main and information flash memory. If LOCKSEG = 1, then erase only main flash memory.
0	Reserved	R	0h	Reserved. Always reads as 0.

5.4.2 FCTL2 Register (address = 012Ah) [reset = 9642h]

Flash Memory Control 2 Register

Figure 5-14. FCTL2 Register

15	14	13	12	11	10	9	8
FWKEYx							
rw-1	rw-0	rw-0	rw-1	rw-0	rw-1	rw-1	rw-0
7	6	5	4	3	2	1	0
FSSELx		FNx					
rw-0	rw-1	rw-0	rw-0	rw-0	rw-0	rw-1	rw-0

Table 5-6. FCTL2 Register Description

Bit	Field	Type	Reset	Description
15-8	FWKEYx	RW	96h	FCTLx password. Always read as 096h. Must be written as 0A5h or a PUC is generated.
7-6	FSSELx	RW	1h	Flash controller clock source select 00b = Reserved 01b = MCLK 10b = SMCLK 11b = SMCLK
5-0	FNx	RW	2h	Flash controller clock divider. These six bits select the divider for the flash controller clock. The divisor value is FNx + 1. For example, when FNx = 00h, the divisor is 1. When FNx = 03Fh, the divisor is 64.

5.4.3 FCTL3 Register (address = 012Ch) [reset = 9658h]

Flash Memory Control 3 Register

Figure 5-15. FCTL3 Register

15	14	13	12	11	10	9	8
FWKEYx							
rw-1	rw-0	rw-0	rw-1	rw-0	rw-1	rw-1	rw-0
7	6	5	4	3	2	1	0
Reserved	LOCKSEG	EMEX	LOCK	WAIT	ACCVIFG	KEYV	BUSY
r0	r(w)-1	rw-0	rw-1	r-1	rw-0	rw-[0]	r(w)-0

Table 5-7. FCTL3 Register Description

Bit	Field	Type	Reset	Description
15-8	FWKEYx	RW	96h	FCTLx password. Always reads as 096h. Must be written as 0A5h or a PUC is generated.
7	Reserved	R	0h	Reserved. Always reads as 0.
6	LOCKSEG	RW	1h	Flash information memory segment lock. Write a 1 to this bit to change its state. Writing 0 has no effect. 0b = Flash information memory segment is unlocked and it is erased during a mass erase. 1b = Flash information memory segment is locked and it is protected from erasure during a mass erase.
5	EMEX	RW	0h	Emergency exit 0b = No emergency exit 1b = Emergency exit
4	LOCK	RW	1h	Lock. This bit unlocks the flash memory for writing or erasing. The LOCK bit can be set anytime during a byte or word write or an erase operation, and the operation completes normally. In the block write mode, if the LOCK bit is set while BLKWRT = WAIT = 1, then BLKWRT and WAIT are reset and the mode ends normally. 0b = Unlocked 1b = Locked
3	WAIT	R	1h	Wait. Indicates the flash memory is being written. 0b = The flash memory is not ready for the next byte or word write 1b = The flash memory is ready for the next byte or word write
2	ACCVIFG	RW	0h	Access violation interrupt flag 0b = No interrupt pending 1b = Interrupt pending
1	KEYV	RW	0h	Flash security key violation. This bit indicates an incorrect FCTLx password was written to any flash control register and generates a PUC when set. KEYV must be reset with software. KEYV is reset with BOR. 0b = FCTLx password was written correctly 1b = FCTLx password was written incorrectly
0	BUSY	RW	0h	Busy. This bit indicates the status of the flash timing generator. 0b = Not Busy 1b = Busy

5.4.4 IE1 Register (address = 0000h) [reset = 00h]

Interrupt Enable 1 Register

Figure 5-16. IE1 Register

7	6	5	4	3	2	1	0
Reserved		ACCVIE	Reserved				
		rw-0					

Table 5-8. IE1 Register Description

Bit	Field	Type	Reset	Description
7-6	Reserved			These bits may be used by other modules. See the device-specific data sheet.
5	ACCVIE	RW	0h	Flash memory access violation interrupt enable. This bit enables the ACCVIFG interrupt. Because other bits in IE1 may be used for other modules, it is recommended to set or clear this bit using BIS.B or BIC.B instructions, rather than MOV.B or CLR.B instructions. 0b = Interrupt not enabled 1b = Interrupt enabled
4-0	Reserved			These bits may be used by other modules. See the device-specific data sheet.

Digital I/O

This chapter describes the operation of the digital I/O ports.

Topic **Page**

6.1 Digital I/O Introduction

The digital I/O features include:

- Independently programmable individual I/Os
- Any combination of input or output
- Individually configurable P1 and P2 interrupts.
- Independent input and output data registers

Devices within the MSP430i family may have many digital I/O ports implemented. Most ports contain eight I/O lines; however, some ports may contain fewer line (see the device-specific data sheet for ports available). Each I/O line is individually configurable for input or output direction, and each can be individually read or written.

Ports P1 and P2 always have interrupt capability. Each interrupt for the P1 and P2 I/O lines can be individually enabled and configured to provide an interrupt on a rising or falling edge of an input signal. All P1 I/O lines source a single interrupt vector (P1IV), and all P2 I/O lines source a different single interrupt vector (P2IV). On some devices, additional ports with interrupt capability may be available (see the device-specific data sheet for details) and contain their own respective interrupt vectors.

Individual ports can be accessed as byte-wide ports or can be combined into word-wide ports and accessed via word formats. Port pairs P1/P2, P3/P4, P5/P6, P7/P8, and so on, are associated with the names PA, PB, PC, PD, and so on, respectively. All port registers are handled in this manner with this naming convention except for the interrupt vector registers, P1IV and P2IV; that is, PAIV does not exist.

When writing to port PA with word operations, all 16 bits are written to the port. When writing to the lower byte of the PA port using byte operations, the upper byte remains unchanged. Similarly, writing to the upper byte of the PA port using byte instructions leaves the lower byte unchanged. When writing to a port that contains less than the maximum number of bits possible, the unused bits are a "don't care". Ports PB, PC, PD, PE, and PF behave similarly.

Reading of the PA port using word operations causes all 16 bits to be transferred to the destination. Reading the lower or upper byte of the PA port (P1 or P2) and storing to memory using byte operations causes only the lower or upper byte to be transferred to the destination, respectively. Reading of the PA port and storing to a general-purpose register using byte operations causes the byte transferred to be written to the least significant byte of the register. The upper significant byte of the destination register is cleared automatically. Ports PB, PC, PD, PE, and PF behave similarly. When reading from ports that contain fewer than the maximum bits possible, unused bits are read as zeros.

6.2 Digital I/O Operation

The digital I/O are configured with user software. The setup and operation of the digital I/O are discussed in the following sections.

6.2.1 Input Registers PxIN

Each bit in each PxIN register reflects the value of the input signal at the corresponding I/O pin when the pin is configured as I/O function. These registers are read only.

- Bit = 0: Input is low
- Bit = 1: Input is high

NOTE: Writing to read-only registers PxIN

Writing to these read-only registers results in increased current consumption while the write attempt is active.

6.2.2 Output Registers PxOUT

Each bit in each PxOUT register is the value to be output on the corresponding I/O pin when the pin is configured as I/O function, output direction.

- Bit = 0: Output is low
- Bit = 1: Output is high

6.2.3 Direction Registers PxDIR

Each bit in each PxDIR register selects the direction of the corresponding I/O pin, regardless of the selected function for the pin. PxDIR bits for I/O pins that are selected for other functions must be set as required by the other function.

- Bit = 0: Port pin is switched to input direction
- Bit = 1: Port pin is switched to output direction

6.2.4 Function Select Registers (PxSEL0, PxSEL1)

Port pins are often multiplexed with other peripheral module functions. See the device-specific data sheet to determine pin functions. Each port pin uses two bits to select the pin function – I/O port or one of the three possible peripheral module function. Table 6-1 shows how to select the various module functions. See the device-specific data sheet to determine pin functions. Each PxSEL bit is used to select the pin function – I/O port or peripheral module function.

Table 6-1. I/O Function Selection

PxSEL1	PxSEL0	I/O Function
0	0	General purpose I/O is selected
0	1	Primary module function is selected
1	0	Secondary module function is selected
1	1	Tertiary module function is selected

Setting the PxSEL1 or PxSEL0 bits to a module function does not automatically set the pin direction. Other peripheral module functions may require the PxDIR bits to be configured according to the direction needed for the module function. See the pin schematics in the device-specific data sheet.

When a port pin is selected as an input to peripheral modules, the input signal to those peripheral modules is a latched representation of the signal at the device pin. While PxSEL1 and PxSEL0 is other than 00, the internal input signal follows the signal at the pin for all connected modules. However, if PxSEL1 and PxSEL0 = 00, the input to the peripherals maintain the value of the input signal at the device pin before the PxSEL1 and PxSEL0 bits were reset.

NOTE: Interrupts are disabled when PxSEL1 = 1 or PxSEL0 = 1

When any PxSEL bit is set, the corresponding pin interrupt function is disabled. Therefore, signals on these pins do not generate interrupts, regardless of the state of the corresponding PxIE bit.

6.2.5 P1 and P2 Interrupts, Port Interrupts

Each pin in ports P1 and P2 have interrupt capability, configured with the PxIFG, PxIE, and PxIES registers. All P1 interrupt flags are prioritized, with P1IFG.0 being the highest, and combined to source a single interrupt vector. The highest priority enabled interrupt generates a number in the P1IV register. This number can be evaluated or added to the program counter to automatically enter the appropriate software routine. Disabled P1 interrupts do not affect the P1IV value. The same functionality exists for P2. The PxIV registers are word access only. Some devices may contain additional port interrupts besides P1 and P2. See the device specific data sheet to determine which port interrupts are available.

Each PxIFG bit is the interrupt flag for its corresponding I/O pin and is set when the selected input signal edge occurs at the pin. All PxIFG interrupt flags request an interrupt when their corresponding PxIE bit and the GIE bit are set. Software can also set each PxIFG flag, providing a way to generate a software-initiated interrupt.

- Bit = 0: No interrupt is pending
- Bit = 1: An interrupt is pending

Only transitions, not static levels, cause interrupts. If any PxIFG flag becomes set during a Px interrupt service routine or is set after the RETI instruction of a Px interrupt service routine is executed, the set PxIFG flag generates another interrupt. This ensures that each transition is acknowledged.

NOTE: PxIFG flags when changing PxOUT or PxDIR

Writing to P1OUT, P1DIR, P2OUT, or P2DIR can result in setting the corresponding P1IFG or P2IFG flags.

Any read access of the P1IV register automatically resets the highest pending interrupt flag. A write access to the P1IV register automatically resets all pending interrupt flags. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt. For example, assume that P1IFG.0 has the highest priority. If the P1IFG.0 and P1IFG.2 flags are set when the interrupt service routine accesses the P1IV register, P1IFG.0 is reset automatically. After the RETI instruction of the interrupt service routine is executed, the P1IFG.2 generates another interrupt.

Port P2 interrupts behave similarly, and source a separate single interrupt vector and utilize the P2IV register.

6.2.5.1 P1IV, P2IV Software Example

The following software example shows the recommended use of P1IV and the handling overhead. The P1IV value is added to the PC to automatically jump to the appropriate routine. The P2IV is similar.

The numbers at the right margin show the necessary CPU cycles for each instruction. The software overhead for different interrupt sources includes interrupt latency and return-from-interrupt cycles, but not the task handling itself.

```
;Interrupt handler for P1                                  Cycles
P1_HND   ...                  ; Interrupt latency            6
         ADD      &P1IV,PC    ; Add offset to Jump table     3
         RETI                 ; Vector  0: No interrupt      5
         JMP      P1_0_HND    ; Vector  2: Port 1 bit 0      2
         JMP      P1_1_HND    ; Vector  4: Port 1 bit 1      2
         JMP      P1_2_HND    ; Vector  6: Port 1 bit 2      2
         JMP      P1_3_HND    ; Vector  8: Port 1 bit 3      2
         JMP      P1_4_HND    ; Vector 10: Port 1 bit 4      2
         JMP      P1_5_HND    ; Vector 12: Port 1 bit 5      2
         JMP      P1_6_HND    ; Vector 14: Port 1 bit 6      2
```

```
          JMP      P1_7_HND    ; Vector 16: Port 1 bit 7      2

P1_7_HND                       ; Vector 16: Port 1 bit 7
          ...                  ; Task starts here
          RETI                 ; Back to main program         5

P1_6_HND                       ; Vector 14: Port 1 bit 6
          ...                  ; Task starts here
          RETI                 ; Back to main program         5

P1_5_HND                       ; Vector 12: Port 1 bit 5
          ...                  ; Task starts here
          RETI                 ; Back to main program         5

P1_4_HND                       ; Vector 10: Port 1 bit 4
          ...                  ; Task starts here
          RETI                 ; Back to main program         5

P1_3_HND                       ; Vector 8: Port 1 bit 3
          ...                  ; Task starts here
          RETI                 ; Back to main program         5

P1_2_HND                       ; Vector 6: Port 1 bit 2
          ...                  ; Task starts here
          RETI                 ; Back to main program         5

P1_1_HND                       ; Vector 4: Port 1 bit 1
          ...                  ; Task starts here
          RETI                 ; Back to main program         5
P1_0_HND                       ; Vector 2: Port 1 bit 0
          ...                  ; Task starts here
          RETI                 ; Back to main program         5
```

6.2.5.2 Interrupt Edge Select Registers P1IES, P2IES

Each PxIES bit selects the interrupt edge for the corresponding I/O pin.

- Bit = 0: Respective PxIFG flag is set with a low-to-high transition
- Bit = 1: Respective PxIFG flag is set with a high-to-low transition

NOTE: Writing to PxIES

Writing to P1IES or P2IES for each corresponding I/O can result in setting the corresponding interrupt flags.

PxIES	PxIN	PxIFG
0 → 1	0	May be set
0 → 1	1	Unchanged
1 → 0	0	Unchanged
1 → 0	1	May be set

6.2.5.3 Interrupt Enable P1IE, P2IE

Each PxIE bit enables the associated PxIFG interrupt flag.

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

6.2.6 Configuring Unused Port Pins

Unused I/O pins should be configured as I/O function, output direction, and left unconnected on the PC board, to prevent a floating input and reduce power consumption. The value of the PxOUT bit is don't care, because the pin is unconnected. See the System Resets, Interrupts, and Operating Modes chapter for termination of unused pins.

6.3 I/O Configuration and LPM4.5 Low-Power Mode

The regulator of the Power Management Module (PMM) is disabled upon entering LPM4.5, which causes all I/O register configurations to be lost. Because the I/O register configurations are lost, the configuration of I/O pins must be handled differently to ensure that all pins in the application behave in a controlled manner upon entering and exiting LPM4.5. Properly setting the I/O pins is critical to achieving the lowest possible power consumption in LPM4.5, as well as preventing any possible uncontrolled input or output I/O state in the application. The application has complete control of the I/O pin conditions preventing the possibility of unwanted spurious activity upon entry and exit from LPM4.5. The detailed flow for entering and exiting LPM4.5 with respect to the I/O operation is as follows:

1. Set all I/Os to general purpose I/Os and configure as needed. Each I/O can be set to input high impedance, output high, or output low. It is critical that no inputs are left floating in the application, otherwise excess current may be drawn in LPM4.5. Configuring the I/O in this manner ensures that each pin is in a safe condition prior to entering LPM4.5.

   Optionally, configure input interrupt pins for wake-up from LPM4.5. To wake up the device from LPM4.5, a general-purpose I/O port must contain an input port with interrupt capability. Not all devices include wakeup from LPM4.5 via I/O, and not all inputs with interrupt capability offer wakeup from LPM4.5. See the device-specific data sheet for availability. To configure a port to wake up the device, it should be configured properly prior to entering LPM4.5. Each port should be configured as general-purpose input. Setting the PxIES bit of the corresponding register determines the edge transition that wakes the device. Lastly, the PxIE for the port must be enabled, as well as the general interrupt enable.
2. Enter LPM4.5 with LPM4.5 entry sequence, enable general interrupts for wake-up:

```
BIS.B #REGOFF, &LPM45CTL
BIS   #GIE+CPUOFF+OSCOFF+SCG1,SR   ; Enter LPM4.5 when REGOFF is set
```

3. Upon entry into LPM4.5, LOCKLPM45 residing in the LPM45CTL register of the PMM module is set automatically. The I/O pin states are held and locked based on the settings prior to LPM4.5 entry. Note that only the pin conditions are retained. All other port configuration register settings such as PxDIR, PxOUT, PxIES, and PxIE contents are lost.
4. An LPM4.5 wakeup event (for example, an edge on a configured wakeup input pin) starts the BOR entry sequence together with the regulator. All peripheral registers are set to their default conditions. Upon exit from LPM4.5, the I/O pins remain locked while LOCKLPM45 remains set. Keeping the I/O pins locked ensures that all pin conditions remain stable upon entering the active mode regardless of the default I/O register settings.
5. Once in active mode, the I/O configuration and I/O interrupt configuration that were not retained during LPM4.5 should be restored to the values prior to entering LPM4.5. It is recommended to reconfigure the PxIES and PxIE to their previous settings to prevent a false port interrupt from occurring. The LOCKLPM45 bit can then be cleared, which releases the I/O pin conditions and I/O interrupt configuration. Any changes to the port configuration registers while LOCKLPM45 is set, have no effect on the I/O pins.
6. After enabling the I/O interrupts, the I/O interrupt that caused the wakeup can be serviced indicated by the PxIFG flags. These flags can be used directly, or the corresponding PxIV register may be used. Note that the PxIFG flag cannot be cleared until the LOCKLPM45 bit has been cleared.

NOTE: It is possible that multiple events occurred on various ports. In these cases, multiple PxIFG flags will be set, and it cannot be determined which port has caused the I/O wakeup.

6.4 Digital I/O Registers

The digital I/O registers are listed in Table 6-2.

NOTE: All registers have word or byte register access. For a generic register *ANYREG*, the suffix "_L" (*ANYREG_L*) refers to the lower byte of the register (bits 0 through 7). The suffix "_H" (*ANYREG_H*) refers to the upper byte of the register (bits 8 through 15).

Table 6-2. Digital I/O Registers

Address	Acronym	Register Name	Type	Access	Reset	Section
001Eh	P1IV	Port 1 Interrupt Vector	Read only	Word	0000h	Section 6.4.1
001Eh	P1IV_L		Read only	Byte	00h	
001Fh	P1IV_H		Read only	Byte	00h	
002Eh	P2IV	Port 2 Interrupt Vector	Read only	Word	0000h	Section 6.4.2
002Eh	P2IV_L		Read only	Byte	00h	
002Fh	P2IV_H		Read only	Byte	00h	
0010h	P1IN	Port 1 Input	Read only	Byte	undefined	Section 6.4.9
0012h	P1OUT	Port 1 Output	Read/write	Byte	undefined	Section 6.4.10
0014h	P1DIR	Port 1 Direction	Read/write	Byte	00h	Section 6.4.11
001Ah	P1SEL0	Port 1 Select 0	Read/write	Byte	00h	Section 6.4.12
001Ch	P1SEL1	Port 1 Select 1	Read/write	Byte	00h	Section 6.4.13
0028h	P1IES	Port 1 Interrupt Edge Select	Read/write	Byte	undefined	Section 6.4.3
002Ah	P1IE	Port 1 Interrupt Enable	Read/write	Byte	00h	Section 6.4.4
002Ch	P1IFG	Port 1 Interrupt Flag	Read/write	Byte	00h	Section 6.4.5
0011h	P2IN	Port 2 Input	Read only	Byte	undefined	Section 6.4.14
0013h	P2OUT	Port 2 Output	Read/write	Byte	undefined	Section 6.4.15
0015h	P2DIR	Port 2 Direction	Read/write	Byte	00h	Section 6.4.16
001Bh	P2SEL0	Port 2 Select 0	Read/write	Byte	00h	Section 6.4.17
001Dh	P2SEL1	Port 2 Select 1	Read/write	Byte	00h	Section 6.4.18
0029h	P2IES	Port 2 Interrupt Edge Select	Read/write	Byte	undefined	Section 6.4.6
002Bh	P2IE	Port 2 Interrupt Enable	Read/write	Byte	00h	Section 6.4.7
002Dh	P2IFG	Port 2 Interrupt Flag	Read/write	Byte	00h	Section 6.4.8
0030h	P3IN	Port 3 Input	Read only	Byte	undefined	Section 6.4.19
0032h	P3OUT	Port 3 Output	Read/write	Byte	undefined	Section 6.4.20
0034h	P3DIR	Port 3 Direction	Read/write	Byte	00h	Section 6.4.21
003Ah	P3SEL0	Port 3 Select 0	Read/write	Byte	00h	Section 6.4.22
003Ch	P3SEL1	Port 3 Select 1	Read/write	Byte	00h	Section 6.4.23
0031h	P4IN	Port 4 Input	Read only	Byte	undefined	Section 6.4.24
0033h	P4OUT	Port 4 Output	Read/write	Byte	undefined	Section 6.4.25
0035h	P4DIR	Port 4 Direction	Read/write	Byte	00h	Section 6.4.26
003Bh	P4SEL0	Port 4 Select 0	Read/write	Byte	00h	Section 6.4.27
003Dh	P4SEL1	Port 4 Select 1	Read/write	Byte	00h	Section 6.4.28
0040h	P5IN	Port 5 Input	Read only	Byte	undefined	Section 6.4.29
0042h	P5OUT	Port 5 Output	Read/write	Byte	undefined	Section 6.4.30
0044h	P5DIR	Port 5 Direction	Read/write	Byte	00h	Section 6.4.31
004Ah	P5SEL0	Port 5 Select 0	Read/write	Byte	00h	Section 6.4.32
004Ch	P5SEL1	Port 5 Select 1	Read/write	Byte	00h	Section 6.4.33
0041h	P6IN	Port 6 Input	Read only	Byte	undefined	Section 6.4.34
0043h	P6OUT	Port 6 Output	Read/write	Byte	undefined	Section 6.4.35
0045h	P6DIR	Port 6 Direction	Read/write	Byte	00h	Section 6.4.36

Table 6-2. Digital I/O Registers (continued)

Address	Acronym	Register Name	Type	Access	Reset	Section
004Bh	P6SEL0	Port 6 Select 0	Read/write	Byte	00h	Section 6.4.37
004Dh	P6SEL1	Port 6 Select 1	Read/write	Byte	00h	Section 6.4.38
0010h	PAIN	Port A Input	Read only	Word	undefined	
0010h	PAIN_L		Read only	Byte	undefined	
0011h	PAIN_H		Read only	Byte	undefined	
0012h	PAOUT	Port A Output	Read/write	Word	undefined	
0012h	PAOUT_L		Read/write	Byte	undefined	
0013h	PAOUT_H		Read/write	Byte	undefined	
0014h	PADIR	Port A Direction	Read/write	Word	0000h	
0014h	PADIR_L		Read/write	Byte	00h	
0015h	PADIR_H		Read/write	Byte	00h	
001Ah	PASEL0	Port A Select 0	Read/write	Word	0000h	
001Ah	PASEL0_L		Read/write	Byte	00h	
001Bh	PASEL0_H		Read/write	Byte	00h	
001Ch	PASEL1	Port A Select 1	Read/write	Word	0000h	
001Ch	PASEL1_L		Read/write	Byte	00h	
001Dh	PASEL1_H		Read/write	Byte	00h	
0028h	PAIES	Port A Interrupt Edge Select	Read/write	Word	undefined	
0028h	PAIES_L		Read/write	Byte	undefined	
0029h	PAIES_H		Read/write	Byte	undefined	
002Ah	PAIE	Port A Interrupt Enable	Read/write	Word	0000h	
002Ah	PAIE_L		Read/write	Byte	00h	
002Bh	PAIE_H		Read/write	Byte	00h	
002Ch	PAIFG	Port A Interrupt Flag	Read/write	Word	0000h	
002Ch	PAIFG_L		Read/write	Byte	00h	
002Dh	PAIFG_H		Read/write	Byte	00h	
0030h	PBIN	Port B Input	Read only	Word	undefined	
0030h	PBIN_L		Read only	Byte	undefined	
0031h	PBIN_H		Read only	Byte	undefined	
0032h	PBOUT	Port B Output	Read/write	Word	undefined	
0032h	PBOUT_L		Read/write	Byte	undefined	
0033h	PBOUT_H		Read/write	Byte	undefined	
0034h	PBDIR	Port B Direction	Read/write	Word	0000h	
0034h	PBDIR_L		Read/write	Byte	00h	
0035h	PBDIR_H		Read/write	Byte	00h	
003Ah	PBSEL0	Port B Select 0	Read/write	Word	0000h	
003Ah	PBSEL0_L		Read/write	Byte	00h	
003Bh	PBSEL0_H		Read/write	Byte	00h	
003Ch	PBSEL1	Port B Select 1	Read/write	Word	0000h	
003Ch	PBSEL1_L		Read/write	Byte	00h	
003Dh	PBSEL1_H		Read/write	Byte	00h	
0040h	PCIN	Port C Input	Read only	Word	undefined	
0040h	PCIN_L		Read only	Byte	undefined	
0041h	PCIN_H		Read only	Byte	undefined	
0042h	PCOUT	Port C Output	Read/write	Word	undefined	
0042h	PCOUT_L		Read/write	Byte	undefined	
0043h	PCOUT_H		Read/write	Byte	undefined	

Table 6-2. Digital I/O Registers (continued)

Address	Acronym	Register Name	Type	Access	Reset	Section
0044h	PCDIR	Port C Direction	Read/write	Word	0000h	
0044h	PCDIR_L		Read/write	Byte	00h	
0045h	PCDIR_H		Read/write	Byte	00h	
004Ah	PCSEL0	Port C Select 0	Read/write	Word	0000h	
004Ah	PCSEL0_L		Read/write	Byte	00h	
004Bh	PCSEL0_H		Read/write	Byte	00h	
004Ch	PCSEL1	Port C Select 1	Read/write	Word	0000h	
004Ch	PCSEL1_L		Read/write	Byte	00h	
004Dh	PCSEL1_H		Read/write	Byte	00h	

6.4.1 P1IV Register (address = 001Eh) [reset = 0000h]

Port 1 Interrupt Vector Register

Figure 6-1. P1IV Register

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
0	0	0	P1IV				0
r0	r0	r0	r-0	r-0	r-0	r-0	r0

Table 6-3. P1IV Register Description

Bit	Field	Type	Reset	Description
15-0	P1IV	R	0h	Port 1 interrupt vector value 00h = No interrupt pending 02h = Interrupt Source = Port 1.0 interrupt; Interrupt Flag = P1IFG.0; Interrupt Priority = Highest 04h = Interrupt Source = Port 1.1 interrupt; Interrupt Flag = P1IFG.1 06h = Interrupt Source = Port 1.2 interrupt; Interrupt Flag = P1IFG.2 08h = Interrupt Source = Port 1.3 interrupt; Interrupt Flag = P1IFG.3 0Ah = Interrupt Source = Port 1.4 interrupt; Interrupt Flag = P1IFG.4 0Ch = Interrupt Source = Port 1.5 interrupt; Interrupt Flag = P1IFG.5 0Eh = Interrupt Source = Port 1.6 interrupt; Interrupt Flag = P1IFG.6 10h = Interrupt Source = Port 1.7 interrupt; Interrupt Flag = P1IFG.7; Interrupt Priority = Lowest

6.4.2 P2IV Register (address = 002Eh) [reset = 0000h]

Port 2 Interrupt Vector Register

Figure 6-2. P2IV Register

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
0	0	0	P2IV				0
r0	r0	r0	r-0	r-0	r-0	r-0	r0

Table 6-4. P2IV Register Description

Bit	Field	Type	Reset	Description
15-0	P2IV	R	0h	Port 2 interrupt vector value 00h = No interrupt pending 02h = Interrupt Source = Port 2.0 interrupt; Interrupt Flag = P2IFG.0; Interrupt Priority = Highest 04h = Interrupt Source = Port 2.1 interrupt; Interrupt Flag = P2IFG.1 06h = Interrupt Source = Port 2.2 interrupt; Interrupt Flag = P2IFG.2 08h = Interrupt Source = Port 2.3 interrupt; Interrupt Flag = P2IFG.3 0Ah = Interrupt Source = Port 2.4 interrupt; Interrupt Flag = P2IFG.4 0Ch = Interrupt Source = Port 2.5 interrupt; Interrupt Flag = P2IFG.5 0Eh = Interrupt Source = Port 2.6 interrupt; Interrupt Flag = P2IFG.6 10h = Interrupt Source = Port 2.7 interrupt; Interrupt Flag = P2IFG.7; Interrupt Priority = Lowest

6.4.3 P1IES Register (address = 0028h) [reset = undefined]

Port 1 Interrupt Edge Select Register

Figure 6-3. P1IES Register

7	6	5	4	3	2	1	0
P1IES							
rw	rw	rw	rw	rw	rw	rw	rw

Table 6-5. P1IES Register Description

Bit	Field	Type	Reset	Description
7-0	P1IES	RW	Undefined	Port 1 interrupt edge select 0b = P1IFG flag is set with a low-to-high transition 1b = P1IFG flag is set with a high-to-low transition

6.4.4 P1IE Register (address = 002Ah) [reset = 00h]

Port 1 Interrupt Enable Register

Figure 6-4. P1IE Register

7	6	5	4	3	2	1	0
P1IE							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-6. P1IE Register Description

Bit	Field	Type	Reset	Description
7-0	P1IE	RW	0h	Port 1 interrupt enable 0b = Corresponding port interrupt disabled 1b = Corresponding port interrupt enabled

6.4.5 P1IFG Register (address = 002Ch) [reset = 00h]

Port 1 Interrupt Flag Register

Figure 6-5. P1IFG Register

7	6	5	4	3	2	1	0
P1IFG							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-7. P1IFG Register Description

Bit	Field	Type	Reset	Description
7-0	P1IFG	RW	0h	Port 1 interrupt flag 0b = No interrupt is pending 1b = Interrupt is pending

6.4.6 P2IES Register (address = 0029h) [reset = undefined]

Port 2 Interrupt Edge Select Register

Figure 6-6. P2IES Register

7	6	5	4	3	2	1	0
P2IES							
rw	rw	rw	rw	rw	rw	rw	rw

Table 6-8. P2IES Register Description

Bit	Field	Type	Reset	Description
7-0	P2IES	RW	Undefined	Port 2 interrupt edge select 0b = P2IFG flag is set with a low-to-high transition 1b = P2IFG flag is set with a high-to-low transition

6.4.7 P2IE Register (address = 002Bh) [reset = 00h]

Port 2 Interrupt Enable Register

Figure 6-7. P2IE Register

7	6	5	4	3	2	1	0
P2IE							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-9. P2IE Register Description

Bit	Field	Type	Reset	Description
7-0	P2IE	RW	0h	Port 2 interrupt enable 0b = Corresponding port interrupt disabled 1b = Corresponding port interrupt enabled

6.4.8 P2IFG Register (address = 002Dh) [reset = 00h]

Port 2 Interrupt Flag Register

Figure 6-8. P2IFG Register

7	6	5	4	3	2	1	0
P2IFG							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-10. P2IFG Register Description

Bit	Field	Type	Reset	Description
7-0	P2IFG	RW	0h	Port 2 interrupt flag 0b = No interrupt is pending 1b = Interrupt is pending

6.4.9 P1IN Register (address = 0010h) [reset = undefined]

Port 1 Input Register

Figure 6-9. P1IN Register

7	6	5	4	3	2	1	0
P1IN							
r	r	r	r	r	r	r	r

Table 6-11. P1IN Register Description

Bit	Field	Type	Reset	Description
7-0	P1IN	R	Undefined	Port 1 input 0b = Input is low 1b = Input is high

6.4.10 P1OUT Register (address = 0012h) [reset = undefined]

Port 1 Output Register

Figure 6-10. P1OUT Register

7	6	5	4	3	2	1	0
P1OUT							
rw	rw	rw	rw	rw	rw	rw	rw

Table 6-12. P1OUT Register Description

Bit	Field	Type	Reset	Description
7-0	P1OUT	RW	Undefined	Port 1 output when I/O configured to output mode. 0b = Output is low 1b = Output is high

6.4.11 P1DIR Register (address = 0014h) [reset = 00h]

Port 1 Direction Register

Figure 6-11. P1DIR Register

7	6	5	4	3	2	1	0
P1DIR							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-13. P1DIR Register Description

Bit	Field	Type	Reset	Description
7-0	P1DIR	RW	0h	Port 1 direction 0b = Port configured as input 1b = Port configured as output

6.4.12 P1SEL0 Register (address = 001Ah) [reset = 00h]

Port 1 Function Selection Register 0

Figure 6-12. P1SEL0 Register

7	6	5	4	3	2	1	0
P1SEL0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-14. P1SEL0 Register Description

Bit	Field	Type	Reset	Description
7-0	P1SEL0	RW	0h	Port function selection. Each bit corresponds to one channel on Port 1. The values of each bit position in P1SEL1 and P1SEL0 are combined to specify the function. For example, if P1SEL1.5 = 1 and P1SEL0.5 = 0, then the secondary module function is selected for P1.5. See P1SEL1 for the definition of each value.

6.4.13 P1SEL1 Register (address = 001Ch) [reset = 00h]

Port 1 Function Selection Register 1

Figure 6-13. P1SEL1 Register

7	6	5	4	3	2	1	0
P1SEL1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-15. P1SEL1 Register Description

Bit	Field	Type	Reset	Description
7-0	P1SEL1	RW	0h	Port function selection. Each bit corresponds to one channel on Port 1. The values of each bit position in P1SEL1 and P1SEL0 are combined to specify the function. For example, if P1SEL1.5 = 1 and P1SEL0.5 = 0, then the secondary module function is selected for P1.5. 00b = General-purpose I/O is selected 01b = Primary module function is selected 10b = Secondary module function is selected 11b = Tertiary module function is selected

6.4.14 P2IN Register (address = 0011h) [reset = undefined]

Port 2 Input Register

Figure 6-14. P2IN Register

7	6	5	4	3	2	1	0
P2IN							
r	r	r	r	r	r	r	r

Table 6-16. P2IN Register Description

Bit	Field	Type	Reset	Description
7-0	P2IN	R	Undefined	Port 2 input 0b = Input is low 1b = Input is high

6.4.15 P2OUT Register (address = 0013h) [reset = undefined]

Port 2 Output Register

Figure 6-15. P2OUT Register

7	6	5	4	3	2	1	0
P2OUT							
rw	rw	rw	rw	rw	rw	rw	rw

Table 6-17. P2OUT Register Description

Bit	Field	Type	Reset	Description
7-0	P2OUT	RW	Undefined	Port 2 output when I/O configured to output mode. 0b = Output is low 1b = Output is high

6.4.16 P2DIR Register (address = 0015h) [reset = 00h]

Port 2 Direction Register

Figure 6-16. P2DIR Register

7	6	5	4	3	2	1	0
P2DIR							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-18. P2DIR Register Description

Bit	Field	Type	Reset	Description
7-0	P2DIR	RW	0h	Port 2 direction 0b = Port configured as input 1b = Port configured as output

6.4.17 P2SEL0 Register (address = 001Bh) [reset = 00h]

Port 2 Function Selection Register 0

Figure 6-17. P2SEL0 Register

7	6	5	4	3	2	1	0
P2SEL0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-19. P2SEL0 Register Description

Bit	Field	Type	Reset	Description
7-0	P2SEL0	RW	0h	Port function selection. Each bit corresponds to one channel on Port 2. The values of each bit position in P2SEL1 and P2SEL0 are combined to specify the function. For example, if P2SEL1.5 = 1 and P2SEL0.5 = 0, then the secondary module function is selected for P2.5. See P2SEL1 for the definition of each value.

6.4.18 P2SEL1 Register (address = 001Dh) [reset = 00h]

Port 2 Function Selection Register 1

Figure 6-18. P2SEL1 Register

7	6	5	4	3	2	1	0
P2SEL1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-20. P2SEL1 Register Description

Bit	Field	Type	Reset	Description
7-0	P2SEL1	RW	0h	Port function selection. Each bit corresponds to one channel on Port 2. The values of each bit position in P2SEL1 and P2SEL0 are combined to specify the function. For example, if P2SEL1.5 = 1 and P2SEL0.5 = 0, then the secondary module function is selected for P2.5. 00b = General-purpose I/O is selected 01b = Primary module function is selected 10b = Secondary module function is selected 11b = Tertiary module function is selected

6.4.19 P3IN Register (address = 0030h) [reset = undefined]

Port 3 Input Register

Figure 6-19. P3IN Register

7	6	5	4	3	2	1	0
P3IN							
r	r	r	r	r	r	r	r

Table 6-21. P3IN Register Description

Bit	Field	Type	Reset	Description
7-0	P3IN	R	Undefined	Port 3 input 0b = Input is low 1b = Input is high

6.4.20 P3OUT Register (address = 0032h) [reset = undefined]

Port 3 Output Register

Figure 6-20. P3OUT Register

7	6	5	4	3	2	1	0
P3OUT							
rw	rw	rw	rw	rw	rw	rw	rw

Table 6-22. P3OUT Register Description

Bit	Field	Type	Reset	Description
7-0	P3OUT	RW	Undefined	Port 3 output when I/O configured to output mode. 0b = Output is low 1b = Output is high

6.4.21 P3DIR Register (address = 0034h) [reset = 00h]

Port 3 Direction Register

Figure 6-21. P3DIR Register

7	6	5	4	3	2	1	0
P3DIR							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-23. P3DIR Register Description

Bit	Field	Type	Reset	Description
7-0	P3DIR	RW	0h	Port 3 direction 0b = Port configured as input 1b = Port configured as output

6.4.22 P3SEL0 Register (address = 003Ah) [reset = 00h]

Port 3 Function Selection Register 0

Figure 6-22. P3SEL0 Register

7	6	5	4	3	2	1	0
P3SEL0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-24. P3SEL0 Register Description

Bit	Field	Type	Reset	Description
7-0	P3SEL0	RW	0h	Port function selection. Each bit corresponds to one channel on Port 3. The values of each bit position in P3SEL1 and P3SEL0 are combined to specify the function. For example, if P3SEL1.5 = 1 and P3SEL0.5 = 0, then the secondary module function is selected for P3.5. See P3SEL1 for the definition of each value.

6.4.23 P3SEL1 Register (address = 003Ch) [reset = 00h]

Port 3 Function Selection Register 1

Figure 6-23. P3SEL1 Register

7	6	5	4	3	2	1	0
P3SEL1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-25. P3SEL1 Register Description

Bit	Field	Type	Reset	Description
7-0	P3SEL1	RW	0h	Port function selection. Each bit corresponds to one channel on Port 3. The values of each bit position in P3SEL1 and P3SEL0 are combined to specify the function. For example, if P3SEL1.5 = 1 and P3SEL0.5 = 0, then the secondary module function is selected for P3.5. 00b = General-purpose I/O is selected 01b = Primary module function is selected 10b = Secondary module function is selected 11b = Tertiary module function is selected

6.4.24 P4IN Register (address = 0031h) [reset = undefined]

Port 4 Input Register

Figure 6-24. P4IN Register

7	6	5	4	3	2	1	0
P4IN							
r	r	r	r	r	r	r	r

Table 6-26. P4IN Register Description

Bit	Field	Type	Reset	Description
7-0	P4IN	R	Undefined	Port 4 input 0b = Input is low 1b = Input is high

6.4.25 P4OUT Register (address = 0033h) [reset = undefined]

Port 4 Output Register

Figure 6-25. P4OUT Register

7	6	5	4	3	2	1	0
P4OUT							
rw	rw	rw	rw	rw	rw	rw	rw

Table 6-27. P4OUT Register Description

Bit	Field	Type	Reset	Description
7-0	P4OUT	RW	Undefined	Port 4 output when I/O configured to output mode. 0b = Output is low 1b = Output is high

6.4.26 P4DIR Register (address = 0035h) [reset = 00h]

Port 4 Direction Register

Figure 6-26. P4DIR Register

7	6	5	4	3	2	1	0
P4DIR							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-28. P4DIR Register Description

Bit	Field	Type	Reset	Description
7-0	P4DIR	RW	0h	Port 4 direction 0b = Port configured as input 1b = Port configured as output

6.4.27 P4SEL0 Register (address = 003Bh) [reset = 00h]

Port 4 Function Selection Register 0

Figure 6-27. P4SEL0 Register

7	6	5	4	3	2	1	0
P4SEL0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-29. P4SEL0 Register Description

Bit	Field	Type	Reset	Description
7-0	P4SEL0	RW	0h	Port function selection. Each bit corresponds to one channel on Port 4. The values of each bit position in P4SEL1 and P4SEL0 are combined to specify the function. For example, if P4SEL1.5 = 1 and P4SEL0.5 = 0, then the secondary module function is selected for P4.5. See P4SEL1 for the definition of each value.

6.4.28 P4SEL1 Register (address = 003Dh) [reset = 00h]

Port 4 Function Selection Register 1

Figure 6-28. P4SEL1 Register

7	6	5	4	3	2	1	0
P4SEL1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-30. P4SEL1 Register Description

Bit	Field	Type	Reset	Description
7-0	P4SEL1	RW	0h	Port function selection. Each bit corresponds to one channel on Port 4. The values of each bit position in P4SEL1 and P4SEL0 are combined to specify the function. For example, if P4SEL1.5 = 1 and P4SEL0.5 = 0, then the secondary module function is selected for P4.5. 00b = General-purpose I/O is selected 01b = Primary module function is selected 10b = Secondary module function is selected 11b = Tertiary module function is selected

6.4.29 P5IN Register (address = 0040h) [reset = undefined]

Port 5 Input Register

Figure 6-29. P5IN Register

7	6	5	4	3	2	1	0
P5IN							
r	r	r	r	r	r	r	r

Table 6-31. P5IN Register Description

Bit	Field	Type	Reset	Description
7-0	P5IN	R	Undefined	Port 5 input 0b = Input is low 1b = Input is high

6.4.30 P5OUT Register (address = 0042h) [reset = undefined]

Port 5 Output Register

Figure 6-30. P5OUT Register

7	6	5	4	3	2	1	0
P5OUT							
rw	rw	rw	rw	rw	rw	rw	rw

Table 6-32. P5OUT Register Description

Bit	Field	Type	Reset	Description
7-0	P5OUT	RW	Undefined	Port 5 output when I/O configured to output mode. 0b = Output is low 1b = Output is high

6.4.31 P5DIR Register (address = 0044h) [reset = 00h]

Port 5 Direction Register

Figure 6-31. P5DIR Register

7	6	5	4	3	2	1	0
P5DIR							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-33. P5DIR Register Description

Bit	Field	Type	Reset	Description
7-0	P5DIR	RW	0h	Port 5 direction 0b = Port configured as input 1b = Port configured as output

6.4.32 P5SEL0 Register (address = 004Ah) [reset = 00h]

Port 5 Function Selection Register 0

Figure 6-32. P5SEL0 Register

7	6	5	4	3	2	1	0
P5SEL0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-34. P5SEL0 Register Description

Bit	Field	Type	Reset	Description
7-0	P5SEL0	RW	0h	Port function selection. Each bit corresponds to one channel on Port 5. The values of each bit position in P5SEL1 and P5SEL0 are combined to specify the function. For example, if P5SEL1.5 = 1 and P5SEL0.5 = 0, then the secondary module function is selected for P5.5. See P5SEL1 for the definition of each value.

6.4.33 P5SEL1 Register (address = 004Ch) [reset = 00h]

Port 5 Function Selection Register 1

Figure 6-33. P5SEL1 Register

7	6	5	4	3	2	1	0
P5SEL1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-35. P5SEL1 Register Description

Bit	Field	Type	Reset	Description
7-0	P5SEL1	RW	0h	Port function selection. Each bit corresponds to one channel on Port 5. The values of each bit position in P5SEL1 and P5SEL0 are combined to specify the function. For example, if P5SEL1.5 = 1 and P5SEL0.5 = 0, then the secondary module function is selected for P5.5. 00b = General-purpose I/O is selected 01b = Primary module function is selected 10b = Secondary module function is selected 11b = Tertiary module function is selected

6.4.34 P6IN Register (address = 0041h) [reset = undefined]

Port 6 Input Register

Figure 6-34. P6IN Register

7	6	5	4	3	2	1	0
P6IN							
r	r	r	r	r	r	r	r

Table 6-36. P6IN Register Description

Bit	Field	Type	Reset	Description
7-0	P6IN	R	Undefined	Port 6 input 0b = Input is low 1b = Input is high

6.4.35 P6OUT Register (address = 0043h) [reset = undefined]

Port 6 Output Register

Figure 6-35. P6OUT Register

7	6	5	4	3	2	1	0
P6OUT							
rw	rw	rw	rw	rw	rw	rw	rw

Table 6-37. P6OUT Register Description

Bit	Field	Type	Reset	Description
7-0	P6OUT	RW	Undefined	Port 6 output when I/O configured to output mode. 0b = Output is low 1b = Output is high

6.4.36 P6DIR Register (address = 0045h) [reset = 00h]

Port 6 Direction Register

Figure 6-36. P6DIR Register

7	6	5	4	3	2	1	0
P6DIR							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-38. P6DIR Register Description

Bit	Field	Type	Reset	Description
7-0	P6DIR	RW	0h	Port 6 direction 0b = Port configured as input 1b = Port configured as output

6.4.37 P6SEL0 Register (address = 004Bh) [reset = 00h]

Port 6 Function Selection Register 0

Figure 6-37. P6SEL0 Register

7	6	5	4	3	2	1	0
P6SEL0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-39. P6SEL0 Register Description

Bit	Field	Type	Reset	Description
7-0	P6SEL0	RW	0h	Port function selection. Each bit corresponds to one channel on Port 6. The values of each bit position in P6SEL1 and P6SEL0 are combined to specify the function. For example, if P6SEL1.5 = 1 and P6SEL0.5 = 0, then the secondary module function is selected for P6.5. See P6SEL1 for the definition of each value.

6.4.38 P6SEL1 Register (address = 004Dh) [reset = 00h]

Port 6 Function Selection Register 1

Figure 6-38. P6SEL1 Register

7	6	5	4	3	2	1	0
P6SEL1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 6-40. P6SEL1 Register Description

Bit	Field	Type	Reset	Description
7-0	P6SEL1	RW	0h	Port function selection. Each bit corresponds to one channel on Port 6. The values of each bit position in P6SEL1 and P6SEL0 are combined to specify the function. For example, if P6SEL1.5 = 1 and P6SEL0.5 = 0, then the secondary module function is selected for P6.5. 00b = General-purpose I/O is selected 01b = Primary module function is selected 10b = Secondary module function is selected 11b = Tertiary module function is selected

Watchdog Timer (WDT)

The watchdog timer (WDT) is a 16-bit timer that can be used as a watchdog or as an interval timer. This chapter describes the WDT module.

Topic **Page**

7.1 Watchdog Timer (WDT) Introduction

The primary function of the WDT module is to perform a controlled system restart after a software problem occurs. If the selected time interval expires, a system reset is generated. If the watchdog function is not needed in an application, the module can be configured as an interval timer and can generate interrupts at selected time intervals.

Features of the watchdog timer module include:

- Four software-selectable time intervals
- Watchdog mode
- Interval mode
- Access to WDT control register is password protected
- Control of $\overline{\text{RST}}$/NMI pin function
- Selectable clock source
- Can be stopped to conserve power

Figure 7-1 shows the WDT block diagram.

NOTE: Watchdog Timer Powers Up Active

After a PUC, the WDT module is automatically configured in the watchdog mode with an initial 32768 clock cycle reset interval. The user must setup or halt the WDT prior to the expiration of the initial reset interval.

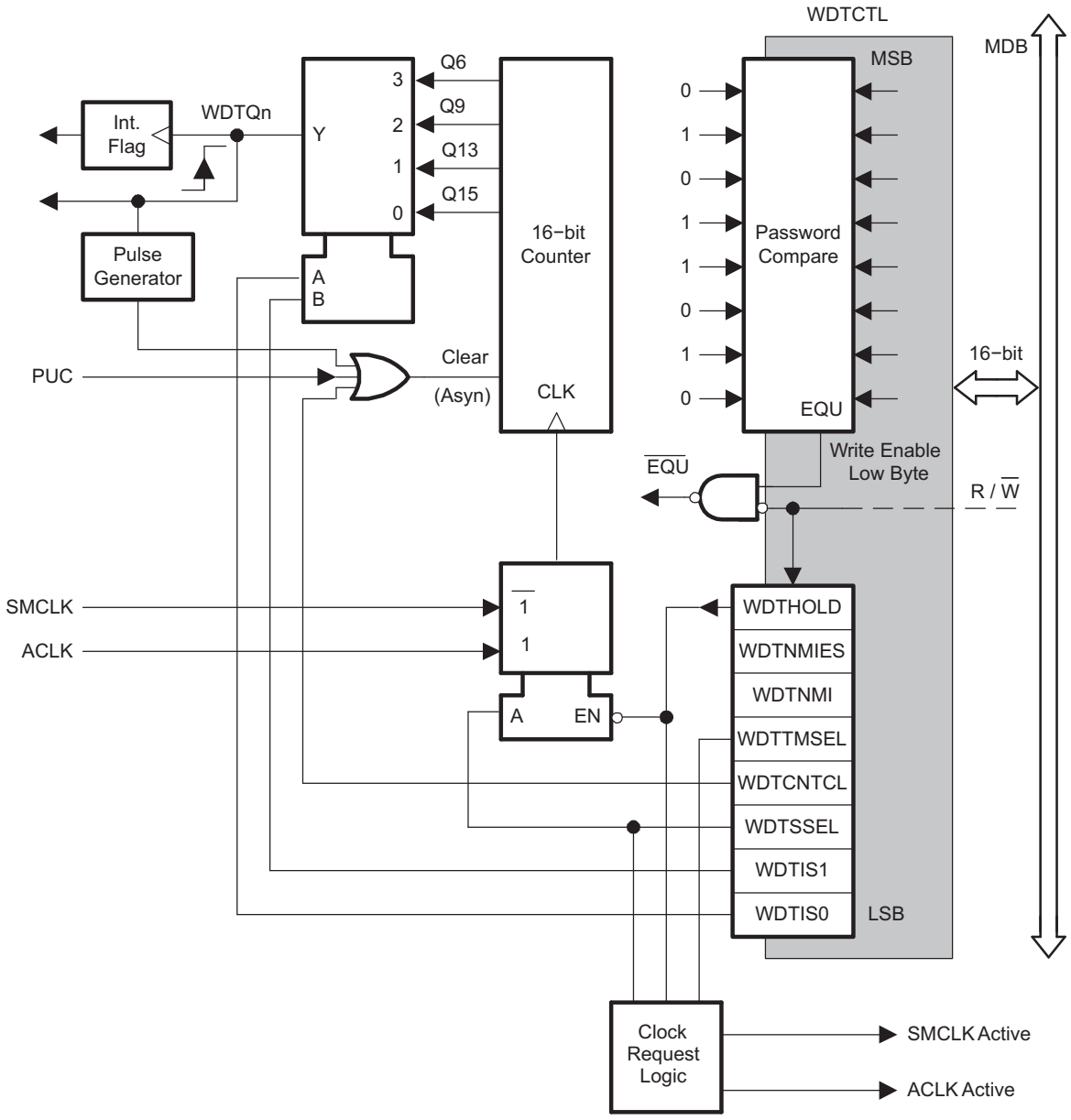


Figure 7-1. Watchdog Timer Block Diagram

7.2 Watchdog Timer Operation

The WDT module can be configured as either a watchdog or interval timer with the WDTCTL register. The WDTCTL register also contains control bits to configure the $\overline{\text{RST}}$/NMI pin. WDTCTL is a 16-bit password-protected read/write register. Any read or write access must use word instructions and write accesses must include the write password 05Ah in the upper byte. Any write to WDTCTL with any value other than 05Ah in the upper byte is a security key violation and triggers a PUC system reset regardless of timer mode. Any read of WDTCTL reads 069h in the upper byte. The WDT counter clock should be slower than or equal to the system (MCLK) frequency.

7.2.1 Watchdog Timer Counter

The watchdog timer counter (WDTCNT) is a 16-bit up-counter that is not directly accessible by software. The WDTCNT is controlled and time intervals selected through the watchdog timer control register WDTCTL.

The WDTCNT can be sourced from ACLK or SMCLK. The clock source is selected with the WDTSSEL bit.

7.2.2 Watchdog Mode

After a PUC condition, the WDT module is configured in the watchdog mode with an initial 32768-cycle reset interval. The user must setup, halt, or clear the WDT prior to the expiration of the initial reset interval, or another PUC is generated. When the WDT is configured to operate in watchdog mode, either writing to WDTCTL with an incorrect password or expiration of the selected time interval triggers a PUC. A PUC resets the WDT to its default condition and configures the $\overline{\text{RST}}$/NMI pin to reset mode.

7.2.3 Interval Timer Mode

Setting the WDTTMSEL bit to 1 selects the interval timer mode. This mode can be used to provide periodic interrupts. In interval timer mode, the WDTIFG flag is set at the expiration of the selected time interval. A PUC is not generated in interval timer mode at expiration of the selected timer interval and the WDTIFG enable bit WDTIE remains unchanged.

When the WDTIE bit and the GIE bit are set, the WDTIFG flag requests an interrupt. The WDTIFG interrupt flag is automatically reset when its interrupt request is serviced, or it can be reset by software. The interrupt vector address in interval timer mode is different from the address in watchdog mode.

NOTE: Modifying the Watchdog Timer

The WDT interval should be changed together with WDTCNTCL = 1 in a single instruction to avoid an unexpected immediate PUC or interrupt.

The WDT should be halted before changing the clock source to avoid a possible incorrect interval.

7.2.4 Watchdog Timer Interrupts

The WDT uses two bits in the SFRs for interrupt control.

- The WDT interrupt flag, WDTIFG, located in IFG1.0
- The WDT interrupt enable, WDTIE, located in IE1.0

When using the WDT in the watchdog mode, the WDTIFG flag sources a reset vector interrupt. The WDTIFG can be used by the reset interrupt service routine to determine if the watchdog caused the device to reset. If the flag is set, then the watchdog timer initiated the reset condition either by timing out or by a security key violation. If WDTIFG is cleared, the reset was caused by a different source.

When using the WDT in interval timer mode, the WDTIFG flag is set after the selected time interval and requests a WDT interval timer interrupt if the WDTIE and the GIE bits are set. The interval timer interrupt vector is different from the reset vector used in watchdog mode. In interval timer mode, the WDTIFG flag is reset automatically when the interrupt is serviced, or it can be reset with software.

7.2.5 Watchdog Timer Clock Request

The WDT module implements clock request feature assuring the clock to the WDT cannot be disabled while in watchdog mode. This means that the low-power modes may be affected by the choice for the WDT clock. For example, if ACLK is the WDT clock source, LPM4 is not available, because the WDT prevents ACLK from being disabled.

The clock request feature is not available when the WDT module is used in interval timer mode.

7.2.6 Operation in Low-Power Modes

The MSP430i devices have several low-power modes. Different clock signals are available in different low-power modes. The requirements of the user's application and the types of clocking that are used determine how the WDT should be configured. For example, the WDT should not be configured in watchdog mode with SMCLK as its clock source if the user wants to use low-power mode 3 because the WDT would keep SMCLK enabled for its clock source, increasing the current consumption of LPM3. When the watchdog timer is not required, the WDTHOLD bit can be used to hold the WDTCNT and reduce power consumption.

7.2.7 Software Examples

Any write operation to WDTCTL must be a word operation with 05Ah (WDTPW) in the upper byte:

```
; Periodically clear an active watchdog
  MOV #WDTPW+WDTCNTCL,&WDTCTL
;
; Change watchdog timer interval
  MOV #WDTPW+WDTCNTL+WDTSSEL,&WDTCTL
;
; Stop the watchdog
  MOV #WDTPW+WDTHOLD,&WDTCTL
;
; Change WDT to interval timer mode, clock/8192 interval
  MOV #WDTPW+WDTCNTCL+WDTTMSEL+WDTIS0,&WDTCTL
```

7.3 WDT Registers

Table 7-1 lists the WDT registers.

Table 7-1. WDT Registers

Address	Acronym	Register Name	Type	Access	Reset	Section
0120h	WDTCTL	Watchdog Timer Control	Read/write	Word	6900h	Section 7.3.1
0000h	IE1	SFR interrupt enable register 1	Read/write	Byte	See (1)	Section 7.3.2
0002h	IFG1	SFR interrupt flag register 1	Read/write	Byte	See (1)	Section 7.3.3

(1) Refer to the *Special Function Registers* section in the device-specific data sheet for the reset value.

7.3.1 WDTCTL Register (address = 0120h) [reset = 6900h]

Watchdog Timer Control Register

Table 7-2. WDTCTL Register

15	14	13	12	11	10	9	8
WDTPW							
rw-0	rw-1	rw-1	rw-0	rw-1	rw-0	rw-0	rw-1
7	6	5	4	3	2	1	0
WDTHOLD	WDTNMIES	WDTNMI	WDTTMSEL	WDTCNTCL	WDTSSEL	WDTIS	
rw-0	rw-0	rw-0	rw-0	r0(w)	rw-0	rw-0	rw-0

Table 7-3. WDTCTL Register Description

Bit	Field	Type	Reset	Description
15-8	WDTPW	RW	69h	Watchdog timer password. Always reads as 69h. Must be written as 5Ah; writing any other value generates a PUC.
7	WDTHOLD	RW	0h	Watchdog timer hold. This bit stops the watchdog timer. Setting WDTHOLD = 1 when the WDT is not in use conserves power. 0b = Watchdog timer is not stopped. 1b = Watchdog timer is stopped.
6	WDTNMIES	RW	0h	Watchdog timer NMI edge select. This bit selects the interrupt edge for the NMI interrupt when WDTNMI = 1. Modifying this bit can trigger an NMI. Modify this bit when WDTIE = 0 to avoid triggering an accidental NMI. 0b = NMI on rising edge 1b = NMI on falling edge
5	WDTNMI	RW	0h	Watchdog timer NMI select. This bit selects the function for the RST/NMI pin. 0b = Reset function 1b = NMI function
4	WDTTMSEL	RW	0h	Watchdog timer mode select 0b = Watchdog mode 1b = Interval timer mode
3	WDTCNTCL	RW	0h	Watchdog timer counter clear. Setting WDTCNTCL = 1 clears the count value to 0000h. WDTCNTCL is automatically reset. 0b = No action 1b = Sets WDTCNT = 0000h
2	WDTSSEL	RW	0h	Watchdog timer clock source select 0b = SMCLK 1b = ACLK
1-0	WDTIS	RW	0h	Watchdog timer interval select. These bits select the watchdog timer interval to set the WDTIFG flag or generate a PUC. 00b = Watchdog clock source /32768 01b = Watchdog clock source /8192 10b = Watchdog clock source /512 11b = Watchdog clock source /64

7.3.2 IE1 Register (address = 0000h) [reset = 00h]

Interrupt Enable Register 1

Figure 7-2. IE1 Register

7	6	5	4	3	2	1	0
Reserved			NMIIE	Reserved			WDTIE
			rw-0				rw-0

Table 7-4. IE1 Register Description

Bit	Field	Type	Reset	Description
7-5	Reserved			These bits may be used by other modules. See the device-specific data sheet.
4	NMIIE	RW	0h	NMI interrupt enable. This bit enables the NMI interrupt. Because other bits in IE1 may be used for other modules, it is recommended to set or clear this bit using BIS.B or BIC.B instructions, rather than MOV.B or CLR.B instructions. 0b = Interrupt not enabled 1b = Interrupt enabled
3-1	Reserved			These bits may be used by other modules. See the device-specific data sheet.
0	WDTIE	RW	0h	Watchdog timer interrupt enable. This bit enables the WDTIFG interrupt for interval timer mode. It is not necessary to set this bit for watchdog mode. Because other bits in IE1 may be used for other modules, it is recommended to set or clear this bit using BIS.B or BIC.B instructions, rather than MOV.B or CLR.B instructions. 0b = Interrupt not enabled 1b = Interrupt enabled

7.3.3 IFG1 Register (address = 0002h) [reset = 00h]

Interrupt Flag Register 1

Figure 7-3. IFG1 Register

7	6	5	4	3	2	1	0
Reserved			NMIIFG	Reserved			WDTIFG
			rw-0				rw-(0)

Table 7-5. IFG1 Register Description

Bit	Field	Type	Reset	Description
7-5	Reserved			These bits may be used by other modules. See the device-specific data sheet.
4	NMIIFG	RW	0h	NMI interrupt flag. NMIIFG must be reset by software. Because other bits in IFG1 may be used for other modules, it is recommended to clear NMIIFG by using BIS.B or BIC.B instructions, rather than MOV.B or CLR.B instructions. 0b = No interrupt pending 1b = Interrupt pending
3-1	Reserved			These bits may be used by other modules. See the device-specific data sheet.
0	WDTIFG	RW	0h	Watchdog timer interrupt flag. In watchdog mode, WDTIFG remains set until reset by software. In interval mode, WDTIFG is reset automatically by servicing the interrupt, or it can be reset by software. Because other bits in IFG1 may be used for other modules, it is recommended to clear WDTIFG by using BIS.B or BIC.B instructions, rather than MOV.B or CLR.B instructions. WDTIFG is reset with POR. 0b = No interrupt pending 1b = Interrupt pending

Hardware Multiplier (MPY)

This chapter describes the hardware multiplier.

Topic **Page**

8.1 Hardware Multiplier Introduction

The hardware multiplier is a peripheral and is not part of the MSP430 CPU. This means that its activities do not interfere with the CPU activities. The multiplier registers are peripheral registers that are loaded and read with CPU instructions.

The hardware multiplier supports:

- Unsigned multiply
- Signed multiply
- Unsigned multiply accumulate
- Signed multiply accumulate
- 16x16 bits, 16x8 bits, 8x16 bits, 8x8 bits

Figure 8-1 shows the hardware multiplier block diagram.

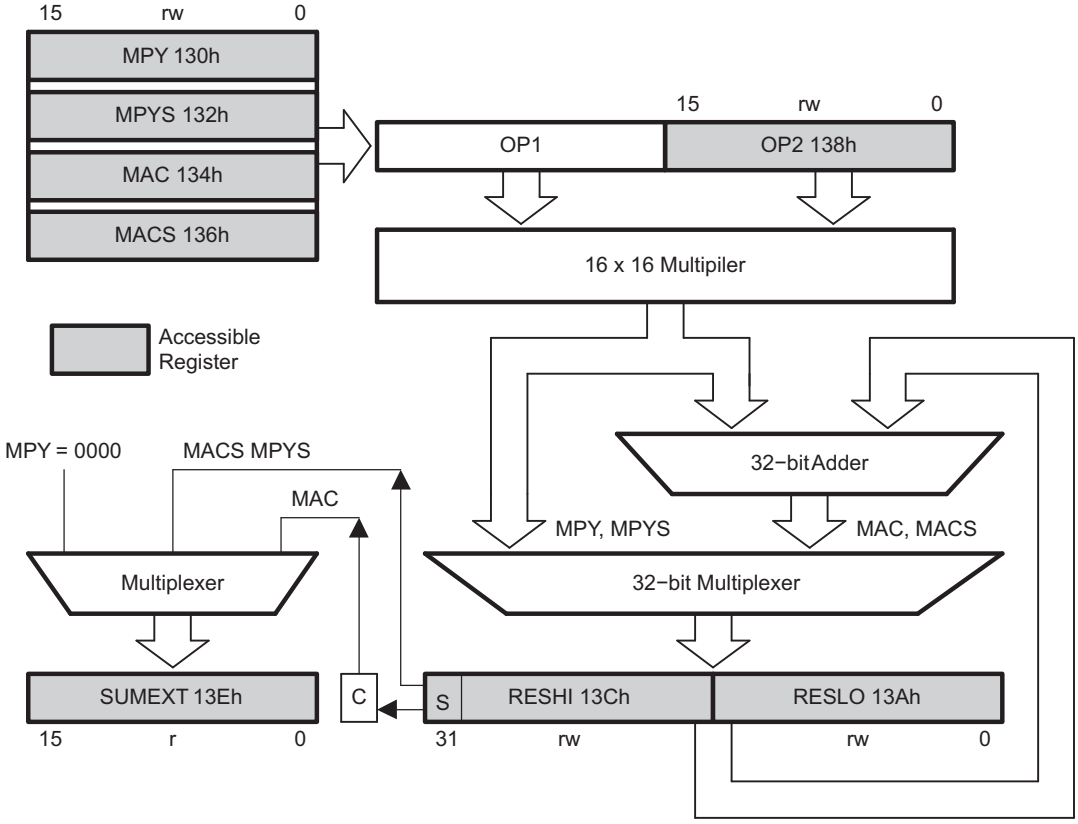


Figure 8-1. Hardware Multiplier Block Diagram

8.2 Hardware Multiplier Operation

The hardware multiplier supports unsigned multiply, signed multiply, unsigned multiply-and-accumulate, and signed multiply-and-accumulate operations. The type of operation is selected by the address to which the first operand is written.

The hardware multiplier has two 16-bit operand registers (OP1 and OP2) and three result registers (RESLO, RESHI, and SUMEXT). RESLO stores the low word of the result, RESHI stores the high word of the result, and SUMEXT stores information about the result. The result is ready in three MCLK cycles and can be read with the next instruction after writing to OP2, except when using an indirect addressing mode to access the result. When using indirect addressing for the result, a NOP is required before the result is ready.

8.2.1 Operand Registers

The operand one register (OP1) has four addresses, shown in Table 8-1, which are used to select the multiply mode. Writing the first operand to the desired address selects the type of multiply operation but does not start any operation. Writing the second operand to the operand two register (OP2) initiates the multiply operation. Writing OP2 starts the selected operation with the values stored in OP1 and OP2. The result is written into the three result registers RESLO, RESHI, and SUMEXT.

Repeated multiply operations may be performed without reloading OP1 if the OP1 value is used for successive operations. It is not necessary to re-write the OP1 value to perform the operations.

Table 8-1. OP1 Addresses

OP1 Address	Register Name	Operation
0130h	MPY	Unsigned multiply
0132h	MPYS	Signed multiply
0134h	MAC	Unsigned multiply-and-accumulate
0136h	MACS	Signed multiply-and-accumulate

8.2.2 Result Registers

The result low register (RESLO) holds the lower 16 bits of the calculation result. The contents of the result high register (RESHI) depend on the multiply operation and are listed in Table 8-2.

Table 8-2. RESHI Contents

Mode	RESHI Contents
MPY	Upper 16-bits of the result
MPYS	The MSB is the sign of the result. The remaining bits are the upper 15-bits of the result. Twos-complement notation is used for the result.
MAC	Upper 16-bits of the result
MACS	Upper 16-bits of the result. Twos-complement notation is used for the result.

The contents of the sum extension register SUMEXT depend on the multiply operation and are listed in Table 8-3.

Table 8-3. SUMEXT Contents

Mode	SUMEXT
MPY	SUMEXT is always 0000h
MPYS	SUMEXT contains the extended sign of the result 00000h = Result was positive or zero 0FFFFh = Result was negative
MAC	SUMEXT contains the carry of the result 0000h = No carry for result 0001h = Result has a carry
MACS	SUMEXT contains the extended sign of the result 00000h = Result was positive or zero 0FFFFh = Result was negative

8.2.2.1 MACS Underflow and Overflow

The multiplier does not automatically detect underflow or overflow in the MACS mode. The accumulator range for positive numbers is 0 to 7FFFFFFFh and for negative numbers is 0FFFFFFFFh to 80000000h. An underflow occurs when the sum of two negative numbers yields a result that is in the range for a positive number. An overflow occurs when the sum of two positive numbers yields a result that is in the range for a negative number. In both of these cases, the SUMEXT register contains the sign of the result, 0FFFFh for overflow and 0000h for underflow. User software must detect and handle these conditions appropriately.

8.2.3 Software Examples

Examples for all multiplier modes follow. All 8x8 modes use the absolute address for the registers because the assembler will not allow .B access to word registers when using the labels from the standard definitions file.

There is no sign extension necessary in software. Accessing the multiplier with a byte instruction during a signed operation automatically causes a sign extension of the byte within the multiplier module.

```
; 16x16 Unsigned Multiply
   MOV     #01234h,&MPY   ; Load first operand
   MOV     #05678h,&OP2   ; Load second operand
;  ...                    ; Process results

; 8x8 Unsigned Multiply. Absolute addressing.
   MOV.B   #012h,&0130h   ; Load first operand
   MOV.B   #034h,&0138h   ; Load 2nd operand
;  ...                    ; Process results

; 16x16 Signed Multiply
   MOV     #01234h,&MPYS  ; Load first operand
   MOV     #05678h,&OP2   ; Load 2nd operand
;  ...                    ; Process results

; 8x8 Signed Multiply. Absolute addressing.
   MOV.B   #012h,&0132h   ; Load first operand
   MOV.B   #034h,&0138h   ; Load 2nd operand
;  ...                    ; Process results

; 16x16 Unsigned Multiply Accumulate
   MOV     #01234h,&MAC   ; Load first operand
   MOV     #05678h,&OP2   ; Load 2nd operand
;  ...                    ; Process results

; 8x8 Unsigned Multiply Accumulate. Absolute addressing
   MOV.B   #012h,&0134h   ; Load first operand
   MOV.B   #034h,&0138h   ; Load 2nd operand
;  ...                    ; Process results

; 16x16 Signed Multiply Accumulate
   MOV     #01234h,&MACS  ; Load first operand
   MOV     #05678h,&OP2   ; Load 2nd operand
;  ...                    ; Process results

; 8x8 Signed Multiply Accumulate. Absolute addressing
   MOV.B   #012h,&0136h   ; Load first operand
   MOV.B   #034h,R5       ; Temp. location for 2nd operand
   MOV     R5,&OP2        ; Load 2nd operand
;  ...                    ; Process results
```

8.2.4 Indirect Addressing of RESLO

When using indirect or indirect autoincrement addressing mode to access the result registers, At least one instruction is needed between loading the second operand and accessing one of the result registers:

```
; Access multiplier results with indirect addressing
   MOV   #RESLO,R5     ; RESLO address in R5 for indirect
   MOV   &OPER1,&MPY   ; Load 1st operand
   MOV   &OPER2,&OP2   ; Load 2nd operand
   NOP                 ; Need one cycle
   MOV   @R5+,&xxx     ; Move RESLO
   MOV   @R5,&xxx      ; Move RESHI
```

8.2.5 Using Interrupts

If an interrupt occurs after writing OP1, but before writing OP2, and the multiplier is used in servicing that interrupt, the original multiplier mode selection is lost and the results are unpredictable. To avoid this, disable interrupts before using the hardware multiplier or do not use the multiplier in interrupt service routines.

```
; Disable interrupts before using the hardware multiplier
   DINT               ; Disable interrupts
   NOP                ; Required for DINT
   MOV   #xxh,&MPY    ; Load 1st operand
   MOV   #xxh,&OP2    ; Load 2nd operand
   EINT               ; Interrupts may be enable before
                      ; Process results
```

8.3 MPY Registers

The hardware multiplier registers are listed in Table 8-4.

Table 8-4. MPY Registers

Address	Acronym	Register Name	Type	Reset
0130h	MPY	Operand one - multiply	Read/write	Unchanged
0132h	MPYS	Operand one - signed multiply	Read/write	Unchanged
0134h	MAC	Operand one - multiply accumulate	Read/write	Unchanged
0136h	MACS	Operand one - signed multiply accumulate	Read/write	Unchanged
0138h	OP2	Operand two	Read/write	Unchanged
013Ah	RESLO	Result low word	Read/write	Undefined
013Ch	RESHI	Result high word	Read/write	Undefined
013Eh	SUMEXT	Sum extension register	Read	Undefined

Timer_A

Timer_A is a 16-bit timer/counter with multiple capture/compare registers. Multiple Timer_A modules can be present on a given device (see the device-specific data sheet). This chapter describes the operation and use of the Timer_A module.

Topic **Page**

9.1 Timer_A Introduction

Timer_A is a 16-bit timer/counter with up to seven capture/compare registers. Timer_A can support multiple capture/compares, PWM outputs, and interval timing. Timer_A also has extensive interrupt capabilities. Interrupts may be generated from the counter on overflow conditions and from each of the capture/compare registers.

Timer_A features include:

- Asynchronous 16-bit timer/counter with four operating modes
- Selectable and configurable clock source
- Up to seven configurable capture/compare registers
- Configurable outputs with pulse width modulation (PWM) capability
- Asynchronous input and output latching
- Interrupt vector register for fast decoding of all Timer_A interrupts

Figure 9-1 shows the block diagram of Timer_A.

NOTE: Use of the word *count*

Count is used throughout this chapter. It means the counter must be in the process of counting for the action to take place. If a particular value is directly written to the counter, an associated action does not take place.

NOTE: Nomenclature

There may be multiple instances of Timer_A on a given device. The prefix TAx is used, where x is a greater than or equal to zero that indicates the Timer_A instance. For devices with one instantiation, x = 0. The suffix n, where n = 0 to 6, represents the specific capture/compare register associated with the Timer_A instantiation.

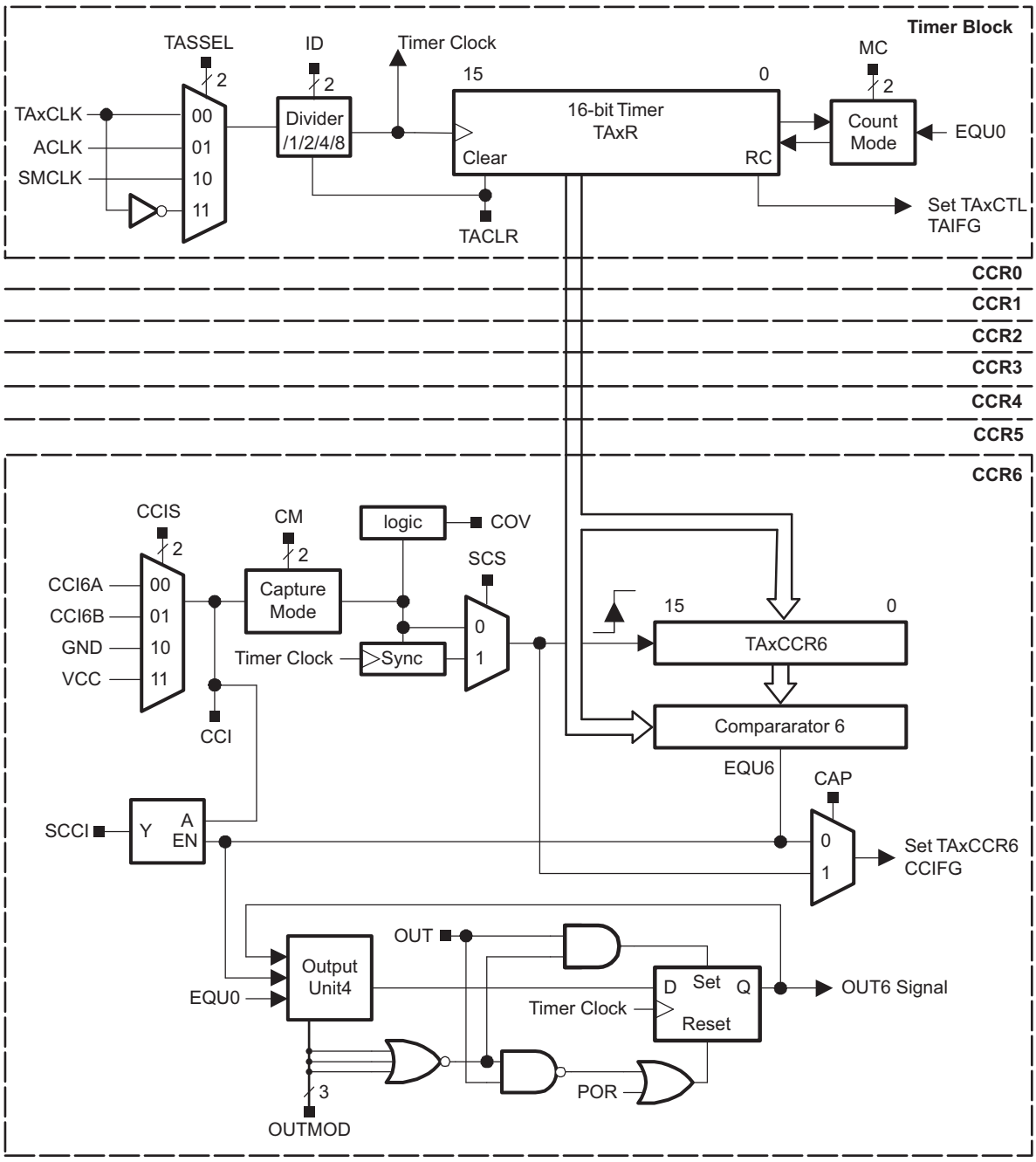


Figure 9-1. Timer_A Block Diagram

9.2 Timer_A Operation

The Timer_A module is configured with user software. The setup and operation of Timer_A are discussed in the following sections.

9.2.1 16-Bit Timer Counter

The 16-bit timer/counter register, TAxR, increments or decrements (depending on mode of operation) with each rising edge of the clock signal. TAxR can be read or written with software. Additionally, the timer can generate an interrupt when it overflows.

TAxR may be cleared by setting the TACLR bit. Setting TACLR also clears the clock divider and count direction for up/down mode.

NOTE: Modifying Timer_A registers

It is recommended to stop the timer before modifying its operation (with exception of the interrupt enable, interrupt flag, and TACLR) to avoid errant operating conditions.

When the timer clock is asynchronous to the CPU clock, any read from TAxR should occur while the timer is not operating or the results may be unpredictable. Alternatively, the timer may be read multiple times while operating, and a majority vote taken in software to determine the correct reading. Any write to TAxR takes effect immediately.

9.2.1.1 Clock Source Select and Divider

The timer clock can be sourced from ACLK, SMCLK, or externally via TAxCLK. The clock source is selected with the TASSEL bits. The selected clock source may be passed directly to the timer or divided by 2, 4, or 8, using the ID bits. The timer clock dividers are reset when TACLR is set.

NOTE: Timer_A dividers

Setting the TACLR bit clears the contents of TAxR and the clock dividers. The clock dividers are implemented as down counters. Therefore, when the TACLR bit is cleared, the timer clock immediately begins clocking at the first rising edge of the Timer_A clock source selected with the TASSEL bits and continues clocking at the divider settings set by the ID bits.

9.2.2 Starting the Timer

The timer can be started or restarted in the following ways:

- The timer counts when MC > { 0 } and the clock source is active.
- When the timer mode is either up or up/down, the timer can be stopped by writing 0 to TAxCCR0. The timer can then be restarted by writing a nonzero value to TAxCCR0. In this case, the timer starts incrementing in the up direction from zero.

9.2.3 Timer Mode Control

The timer has four modes of operation: stop, up, continuous, and up/down (see Table 9-1). The operating mode is selected with the MC bits.

Table 9-1. Timer Modes

MC	Mode	Description
00	Stop	The timer is halted.
01	Up	The timer repeatedly counts from zero to the value of TAxCCR0
10	Continuous	The timer repeatedly counts from zero to 0FFFFh.
11	Up/down	The timer repeatedly counts from zero up to the value of TAxCCR0 and back down to zero.

9.2.3.1 Up Mode

The up mode is used if the timer period must be different from 0FFFFh counts. The timer repeatedly counts up to the value of compare register TAxCCR0, which defines the period (see Figure 9-2). The number of timer counts in the period is TAxCCR0 + 1. When the timer value equals TAxCCR0, the timer restarts counting from zero. If up mode is selected when the timer value is greater than TAxCCR0, the timer immediately restarts counting from zero.

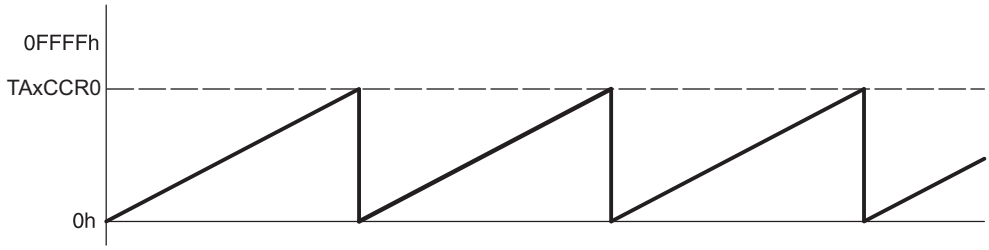


Figure 9-2. Up Mode

The TAxCCR0 CCIFG interrupt flag is set when the timer *counts* to the TAxCCR0 value. The TAIFG interrupt flag is set when the timer *counts* from TAxCCR0 to zero. Figure 9-3 shows the flag set cycle.

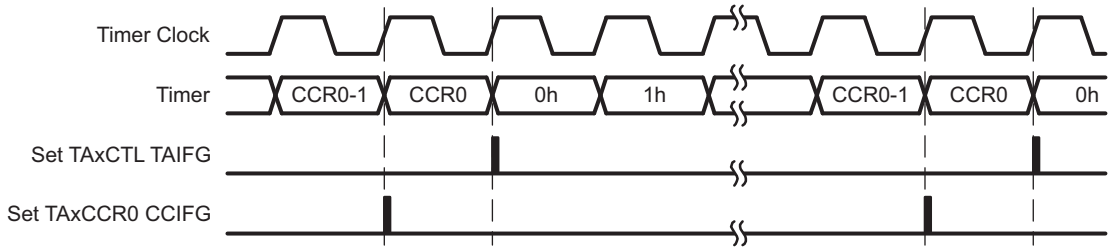


Figure 9-3. Up Mode Flag Setting

9.2.3.1.1 Changing Period Register TAxCCR0

When changing TAxCCR0 while the timer is running, if the new period is greater than or equal to the old period or greater than the current count value, the timer counts up to the new period. If the new period is less than the current count value, the timer rolls to zero. However, one additional count may occur before the counter rolls to zero.

9.2.3.2 Continuous Mode

In the continuous mode, the timer repeatedly counts up to 0FFFFh and restarts from zero as shown in Figure 9-4. The capture/compare register TAxCCR0 works the same way as the other capture/compare registers.

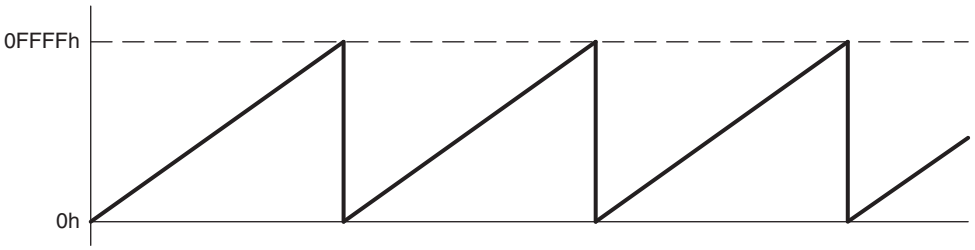


Figure 9-4. Continuous Mode

The TAIFG interrupt flag is set when the timer *counts* from 0FFFFh to zero. Figure 9-5 shows the flag set cycle.

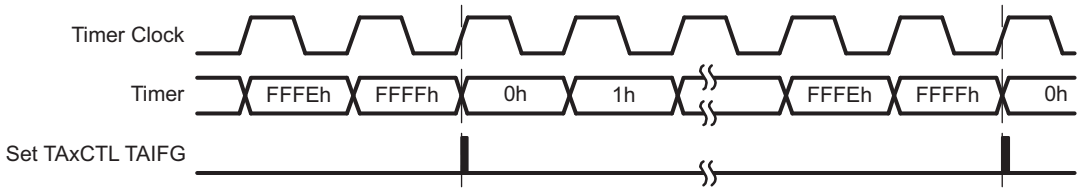


Figure 9-5. Continuous Mode Flag Setting

9.2.3.3 Use of Continuous Mode

The continuous mode can be used to generate independent time intervals and output frequencies. Each time an interval is completed, an interrupt is generated. The next time interval is added to the TAxCCRn register in the interrupt service routine. Figure 9-6 shows two separate time intervals, t_0 and t_1, being added to the capture/compare registers. In this usage, the time interval is controlled by hardware, not software, without impact from interrupt latency. Up to n (where n = 0 to 6), independent time intervals or output frequencies can be generated using capture/compare registers.

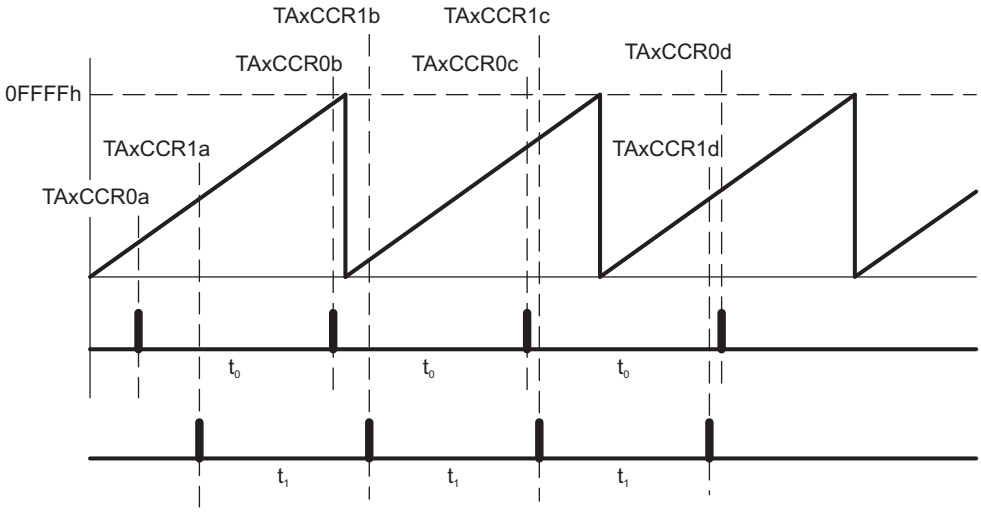


Figure 9-6. Continuous Mode Time Intervals

Time intervals can be produced with other modes as well, where TAxCCR0 is used as the period register. Their handling is more complex because the sum of the old TAxCCRn data and the new period can be higher than the TAxCCR0 value. When the previous TAxCCRn value plus t_x is greater than the TAxCCR0 data, the TAxCCR0 value must be subtracted to obtain the correct time interval.

9.2.3.4 Up/Down Mode

The up/down mode is used if the timer period must be different from 0FFFFh counts, and if symmetrical pulse generation is needed. The timer repeatedly counts up to the value of compare register TAxCCR0 and back down to zero (see Figure 9-7). The period is twice the value in TAxCCR0.

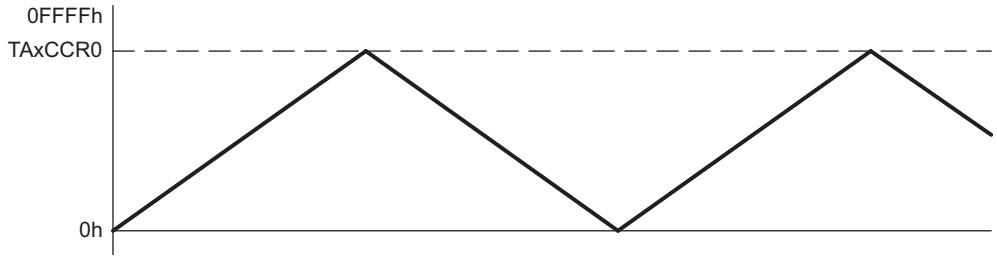


Figure 9-7. Up/Down Mode

The count direction is latched. This allows the timer to be stopped and then restarted in the same direction it was counting before it was stopped. If this is not desired, the TACLR bit must be set to clear the direction. The TACLR bit also clears the TAxR value and the timer clock divider.

In up/down mode, the TAxCCR0 CCIFG interrupt flag and the TAIFG interrupt flag are set only once during a period, separated by one-half the timer period. The TAxCCR0 CCIFG interrupt flag is set when the timer *counts* from TAxCCR0-1 to TAxCCR0, and TAIFG is set when the timer completes *counting* down from 0001h to 0000h. Figure 9-8 shows the flag set cycle.

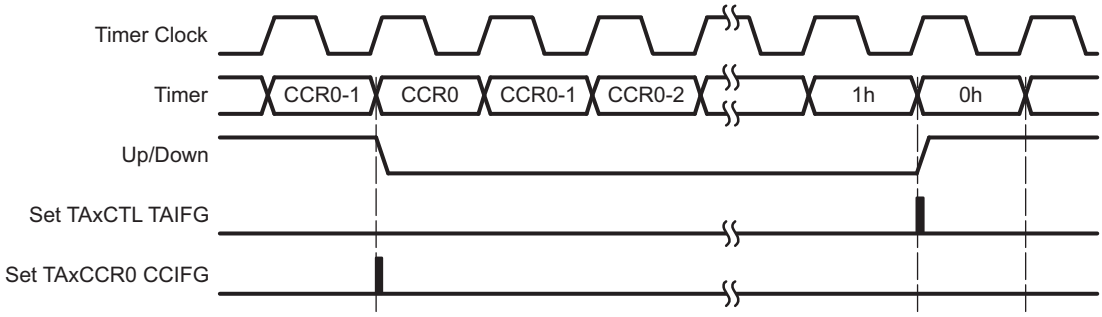


Figure 9-8. Up/Down Mode Flag Setting

9.2.3.4.1 Changing Period Register TAxCCR0

When changing TAxCCR0 while the timer is running and counting in the down direction, the timer continues its descent until it reaches zero. The new period takes affect after the counter counts down to zero.

When the timer is counting in the up direction, and the new period is greater than or equal to the old period or greater than the current count value, the timer counts up to the new period before counting down. When the timer is counting in the up direction and the new period is less than the current count value, the timer begins counting down. However, one additional count may occur before the counter begins counting down.

9.2.3.5 Use of Up/Down Mode

The up/down mode supports applications that require dead times between output signals (see Section 9.2.5). For example, to avoid overload conditions, two outputs that drive an H-bridge must never be in a high state simultaneously. In the example shown in Figure 9-9, the t_{dead} is:

$t_{dead} = t_{timer} \times (TAxCCR1 – TAxCCR2)$

Where:

t_{dead} = Time during which both outputs need to be inactive
t_{timer} = Cycle time of the timer clock
TAxCCRn = Content of capture/compare register n

The TAxCCRn registers are not buffered. They update immediately when written. Therefore, any required dead time is not maintained automatically.

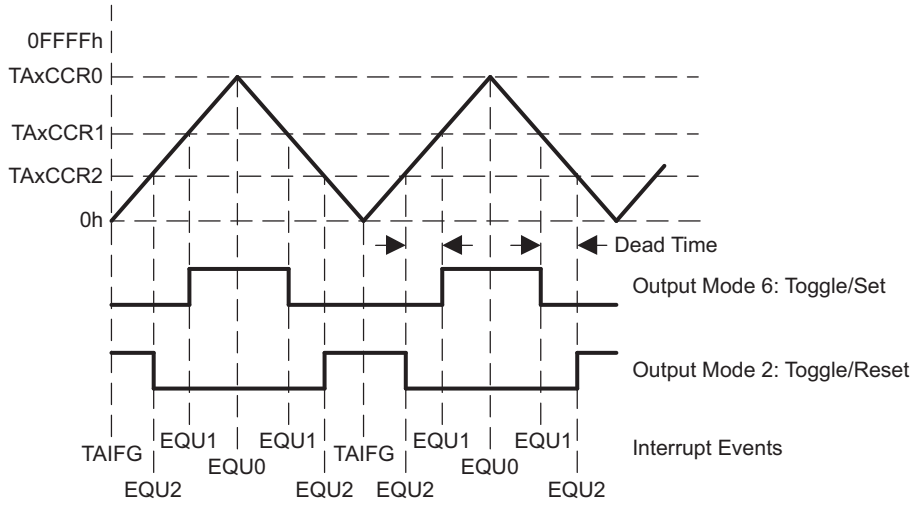


Figure 9-9. Output Unit in Up/Down Mode

9.2.4 Capture/Compare Blocks

Up to seven identical capture/compare blocks, TAxCCRn (where n = 0 to 7), are present in Timer_A. Any of the blocks may be used to capture the timer data or to generate time intervals.

9.2.4.1 Capture Mode

The capture mode is selected when CAP = 1. Capture mode is used to record time events. It can be used for speed computations or time measurements. The capture inputs CCIxA and CCIxB are connected to external pins or internal signals and are selected with the CCIS bits. The CM bits select the capture edge of the input signal as rising, falling, or both. A capture occurs on the selected edge of the input signal. If a capture occurs:

- The timer value is copied into the TAxCCRn register.
- The interrupt flag CCIFG is set.

The input signal level can be read at any time via the CCI bit. Devices may have different signals connected to CCIxA and CCIxB. See the device-specific data sheet for the connections of these signals.

The capture signal can be asynchronous to the timer clock and cause a race condition. Setting the SCS bit synchronizes the capture with the next timer clock. Setting the SCS bit to synchronize the capture signal with the timer clock is recommended (see Figure 9-10).

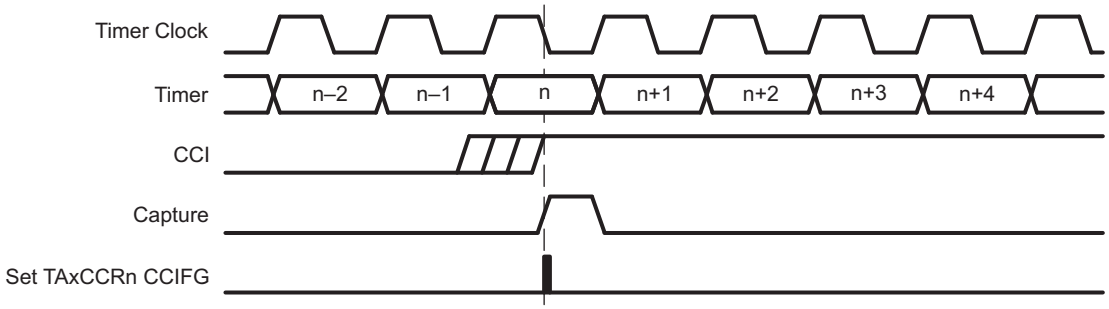


Figure 9-10. Capture Signal (SCS = 1)

NOTE: Changing Capture Inputs

Changing capture inputs while in capture mode may cause unintended capture events. To avoid this scenario, capture inputs should only be changed when capture mode is disabled (CM = {0} or CAP = 0).

Overflow logic is provided in each capture/compare register to indicate if a second capture was performed before the value from the first capture was read. Bit COV is set when this occurs as shown in Figure 9-11. COV must be reset with software.

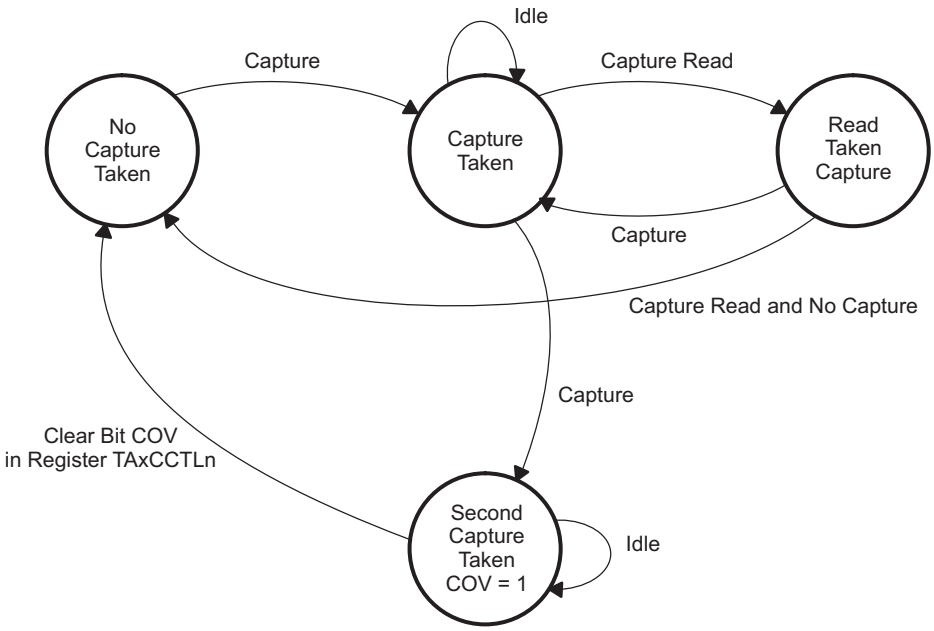


Figure 9-11. Capture Cycle

9.2.4.1.1 Capture Initiated by Software

Captures can be initiated by software. The CMx bits can be set for capture on both edges. Software then sets CCIS1 = 1 and toggles bit CCIS0 to switch the capture signal between V_{CC} and GND, initiating a capture each time CCIS0 changes state:

```
MOV  #CAP+SCS+CCIS1+CM_3,&TA0CCTL1  ; Set up TA0CCTL1, synch. capture mode
                                    ; Event trigger on both edges of capture input.
XOR  #CCIS0,&TA0CCTL1               ; TA0CCR1 = TA0R
```

NOTE: Capture Initiated by Software

In general, changing capture inputs while in capture mode may cause unintended capture events. For this scenario, switching the capture input between VCC and GND, disabling the capture mode is not required.

9.2.4.2 Compare Mode

The compare mode is selected when CAP = 0. The compare mode is used to generate PWM output signals or interrupts at specific time intervals. When TAxR *counts* to the value in a TAxCCRn, where n represents the specific capture/compare register.

- Interrupt flag CCIFG is set.
- Internal signal EQUn = 1.
- EQUn affects the output according to the output mode.
- The input signal CCI is latched into SCCI.

9.2.5 Output Unit

Each capture/compare block contains an output unit. The output unit is used to generate output signals, such as PWM signals. Each output unit has eight operating modes that generate signals based on the EQU0 and EQUn signals.

9.2.5.1 Output Modes

The output modes are defined by the OUTMOD bits and are described in Table 9-2. The OUTn signal is changed with the rising edge of the timer clock for all modes except mode 0. Output modes 2, 3, 6, and 7 are not useful for output unit 0 because EQUn = EQU0.

Table 9-2. Output Modes

OUTMODx	Mode	Description
000	Output	The output signal OUTn is defined by the OUT bit. The OUTn signal updates immediately when OUT is updated.
001	Set	The output is set when the timer *counts* to the TAxCCRn value. It remains set until a reset of the timer, or until another output mode is selected and affects the output.
010	Toggle/Reset	The output is toggled when the timer *counts* to the TAxCCRn value. It is reset when the timer *counts* to the TAxCCR0 value.
011	Set/Reset	The output is set when the timer *counts* to the TAxCCRn value. It is reset when the timer *counts* to the TAxCCR0 value.
100	Toggle	The output is toggled when the timer *counts* to the TAxCCRn value. The output period is double the timer period.
101	Reset	The output is reset when the timer *counts* to the TAxCCRn value. It remains reset until another output mode is selected and affects the output.
110	Toggle/Set	The output is toggled when the timer *counts* to the TAxCCRn value. It is set when the timer *counts* to the TAxCCR0 value.
111	Reset/Set	The output is reset when the timer *counts* to the TAxCCRn value. It is set when the timer *counts* to the TAxCCR0 value.

9.2.5.1.1 Output Example—Timer in Up Mode

The OUTn signal is changed when the timer *counts* up to the TAxCCRn value and rolls from TAxCCR0 to zero, depending on the output mode. An example is shown in Figure 9-12 using TAxCCR0 and TAxCCR1.

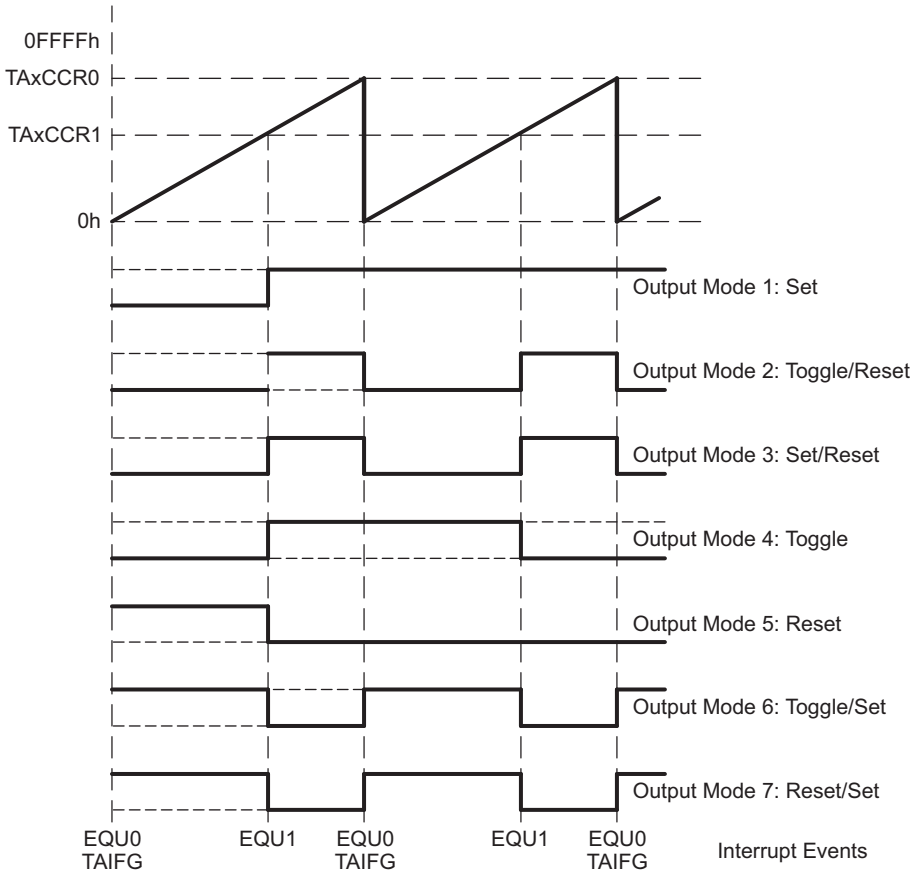


Figure 9-12. Output Example – Timer in Up Mode

9.2.5.1.2 Output Example – Timer in Continuous Mode

The OUTn signal is changed when the timer reaches the TAxCCRn and TAxCCR0 values, depending on the output mode. An example is shown in Figure 9-13 using TAxCCR0 and TAxCCR1.

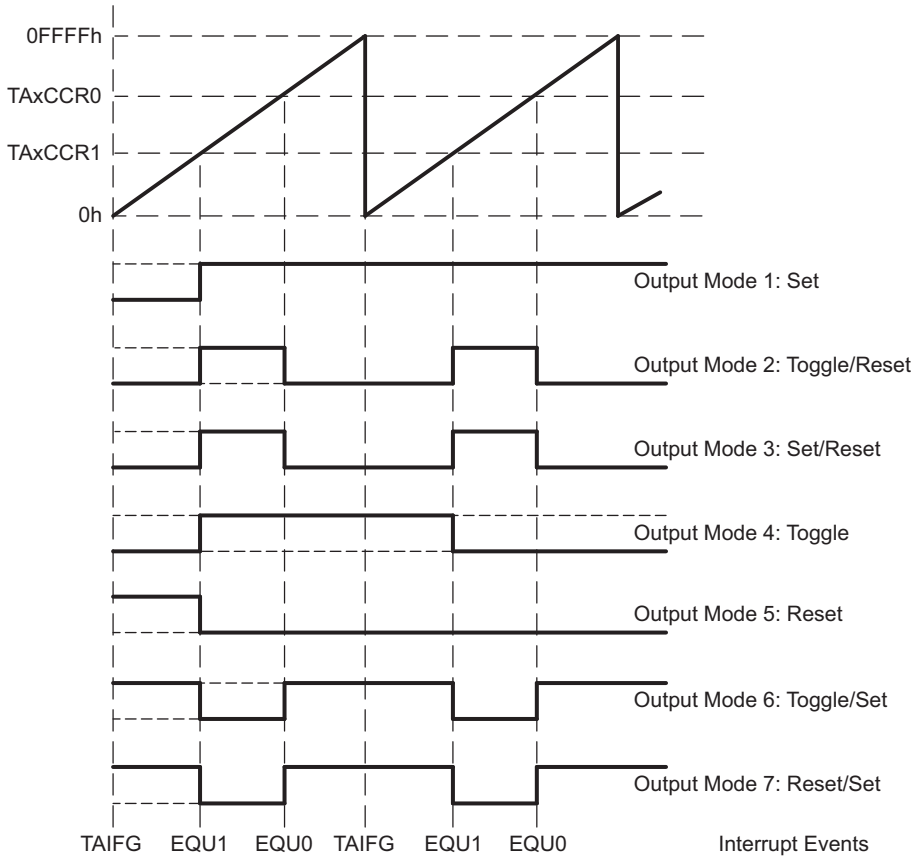


Figure 9-13. Output Example – Timer in Continuous Mode

9.2.5.1.3 Output Example – Timer in Up/Down Mode

The OUTn signal changes when the timer equals TAxCCRn in either count direction and when the timer equals TAxCCR0, depending on the output mode. An example is shown in Figure 9-14 using TAxCCR0 and TAxCCR2.

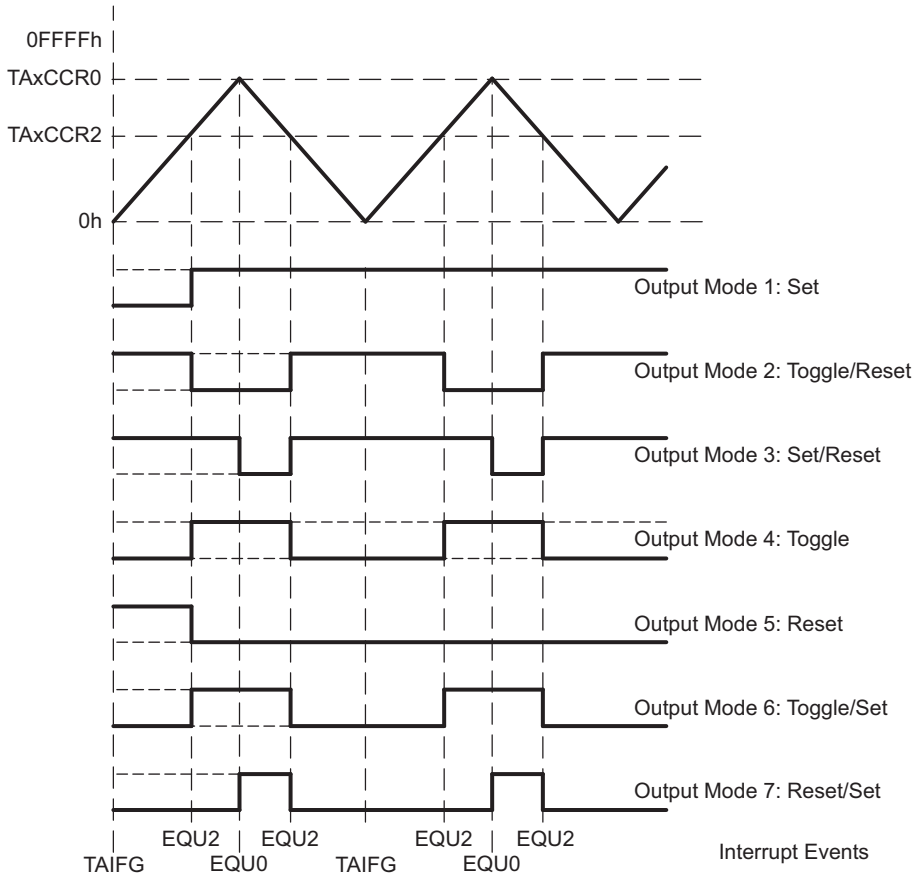


Figure 9-14. Output Example – Timer in Up/Down Mode

NOTE: Switching between output modes

When switching between output modes, one of the OUTMOD bits should remain set during the transition, unless switching to mode 0. Otherwise, output glitching can occur, because a NOR gate decodes output mode 0. A safe method for switching between output modes is to use output mode 7 as a transition state:

```
BIS   #OUTMOD_7,&TA0CCTL1     ; Set output mode=7
BIC   #OUTMOD,&TA0CCTL1       ; Clear unwanted bits
```

9.2.6 Timer_A Interrupts

Two interrupt vectors are associated with the 16-bit Timer_A module:

- TAxCCR0 interrupt vector for TAxCCR0 CCIFG
- TAxIV interrupt vector for all other CCIFG flags and TAIFG

In capture mode, any CCIFG flag is set when a timer value is captured in the associated TAxCCRn register. In compare mode, any CCIFG flag is set if TAxR *counts* to the associated TAxCCRn value. Software may also set or clear any CCIFG flag. All CCIFG flags request an interrupt when their corresponding CCIE bit and the GIE bit are set.

9.2.6.1 TAxCCR0 Interrupt

The TAxCCR0 CCIFG flag has the highest Timer_A interrupt priority and has a dedicated interrupt vector as shown in Figure 9-15. The TAxCCR0 CCIFG flag is automatically reset when the TAxCCR0 interrupt request is serviced.

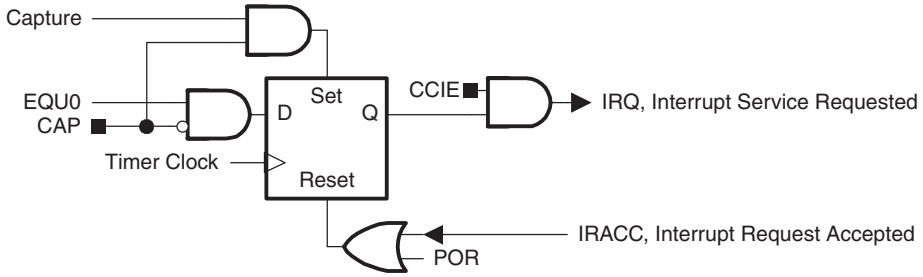


Figure 9-15. Capture/Compare TAxCCR0 Interrupt Flag

9.2.6.2 TAxIV, Interrupt Vector Generator

The TAxCCRy CCIFG flags and TAIFG flags are prioritized and combined to source a single interrupt vector. The interrupt vector register TAxIV is used to determine which flag requested an interrupt.

The highest-priority enabled interrupt generates a number in the TAxIV register (see register description). This number can be evaluated or added to the program counter to automatically enter the appropriate software routine. Disabled Timer_A interrupts do not affect the TAxIV value.

Any access, read or write, of the TAxIV register automatically resets the highest-pending interrupt flag. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt. For example, if the TAxCCR1 and TAxCCR2 CCIFG flags are set when the interrupt service routine accesses the TAxIV register, TAxCCR1 CCIFG is reset automatically. After the RETI instruction of the interrupt service routine is executed, the TAxCCR2 CCIFG flag generates another interrupt.

9.2.6.2.1 TAxIV Software Example

The following software example shows the recommended use of TAxIV and the handling overhead. The TAxIV value is added to the PC to automatically jump to the appropriate routine. The example assumes a single instantiation of the largest timer configuration available.

The numbers at the right margin show the necessary CPU cycles for each instruction. The software overhead for different interrupt sources includes interrupt latency and return-from-interrupt cycles, but not the task handling itself. The latencies are:

- Capture/compare block TA0CCR0: 11 cycles
- Capture/compare blocks TA0CCR1, TA0CCR2, TA0CCR3, TA0CCR4, TA0CCR5, TA0CCR6: 16 cycles
- Timer overflow TA0IFG: 14 cycles

```
; Interrupt handler for TA0CCR0 CCIFG.                    Cycles
CCIFG_0_HND
;        ...          ; Start of handler Interrupt latency    6
         RETI                                                 5

; Interrupt handler for TA0IFG, TA0CCR1 through TA0CCR6 CCIFG.

TA0_HND      ...                ; Interrupt latency           6
         ADD      &TA0IV,PC     ; Add offset to Jump table    3
         RETI                   ; Vector  0: No interrupt     5
         JMP      CCIFG_1_HND   ; Vector  2: TA0CCR1          2
         JMP      CCIFG_2_HND   ; Vector  4: TA0CCR2          2
         JMP      CCIFG_3_HND   ; Vector  6: TA0CCR3          2
         JMP      CCIFG_4_HND   ; Vector  8: TA0CCR4          2
         JMP      CCIFG_5_HND   ; Vector 10: TA0CCR5          2
         JMP      CCIFG_6_HND   ; Vector 12: TA0CCR6          2

TA0IFG_HND                     ; Vector 14: TA0IFG Flag
         ...                   ; Task starts here
         RETI                                                 5

CCIFG_6_HND                    ; Vector 12: TA0CCR6
         ...                   ; Task starts here
         RETI                  ; Back to main program        5

CCIFG_5_HND                    ; Vector 10: TA0CCR5
         ...                   ; Task starts here
         RETI                  ; Back to main program        5

CCIFG_4_HND                    ; Vector 8: TA0CCR4
         ...                   ; Task starts here
         RETI                  ; Back to main program        5

CCIFG_3_HND                    ; Vector 6: TA0CCR3
         ...                   ; Task starts here
         RETI                  ; Back to main program        5

CCIFG_2_HND                    ; Vector 4: TA0CCR2
         ...                   ; Task starts here
         RETI                  ; Back to main program        5

CCIFG_1_HND                    ; Vector 2: TA0CCR1
         ...                   ; Task starts here
         RETI                  ; Back to main program        5
```

9.3 Timer_A Registers

Timer_A registers are listed in Table 9-3 for the largest configuration available. The base address can be found in the device-specific data sheet.

Table 9-3. Timer_A Registers

Address	Acronym	Register Name	Type	Access	Reset	Section
0160h	TAxCTL	Timer_Ax Control	Read/write	Word	0000h	Section 9.3.1
0162h	TAxCCTL0	Timer_Ax Capture/Compare Control 0	Read/write	Word	0000h	Section 9.3.3
0164h	TAxCCTL1	Timer_Ax Capture/Compare Control 1	Read/write	Word	0000h	Section 9.3.4
0166h	TAxCCTL2	Timer_Ax Capture/Compare Control 2	Read/write	Word	0000h	Section 9.3.5
0168h	TAxCCTL3	Timer_Ax Capture/Compare Control 3	Read/write	Word	0000h	Section 9.3.6
016Ah	TAxCCTL4	Timer_Ax Capture/Compare Control 4	Read/write	Word	0000h	Section 9.3.7
016Ch	TAxCCTL5	Timer_Ax Capture/Compare Control 5	Read/write	Word	0000h	Section 9.3.8
016Eh	TAxCCTL6	Timer_Ax Capture/Compare Control 6	Read/write	Word	0000h	Section 9.3.9
0170h	TAxR	Timer_Ax Counter	Read/write	Word	0000h	Section 9.3.2
0172h	TAxCCR0	Timer_Ax Capture/Compare 0	Read/write	Word	0000h	Section 9.3.10
0174h	TAxCCR1	Timer_Ax Capture/Compare 1	Read/write	Word	0000h	Section 9.3.11
0176h	TAxCCR2	Timer_Ax Capture/Compare 2	Read/write	Word	0000h	Section 9.3.12
0178h	TAxCCR3	Timer_Ax Capture/Compare 3	Read/write	Word	0000h	Section 9.3.13
017Ah	TAxCCR4	Timer_Ax Capture/Compare 4	Read/write	Word	0000h	Section 9.3.14
017Ch	TAxCCR5	Timer_Ax Capture/Compare 5	Read/write	Word	0000h	Section 9.3.15
017Eh	TAxCCR6	Timer_Ax Capture/Compare 6	Read/write	Word	0000h	Section 9.3.16
012Eh	TAxIV	Timer_Ax Interrupt Vector	Read only	Word	0000h	Section 9.3.17

9.3.1 TAxCTL Register (address = 0160h) [reset = 0000h]

Timer_Ax Control Register

Figure 9-16. TAxCTL Register

15	14	13	12	11	10	9	8
Reserved						TASSEL	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ID		MC		Reserved	TACLR	TAIE	TAIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	w-(0)	rw-(0)	rw-(0)

Table 9-4. TAxCTL Register Description

Bit	Field	Type	Reset	Description
15-10	Reserved	RW	0h	Reserved
9-8	TASSEL	RW	0h	Timer_A clock source select 00b = TAxCLK 01b = ACLK 10b = SMCLK 11b = Inverted TAxCLK
7-6	ID	RW	0h	Input divider. These bits select the divider for the input clock. 00b = /1 01b = /2 10b = /4 11b = /8
5-4	MC	RW	0h	Mode control. Setting MCx = 00h when Timer_A is not in use conserves power. 00b = Stop mode: Timer is halted 01b = Up mode: Timer counts up to TAxCCR0 10b = Continuous mode: Timer counts up to 0FFFFh 11b = Up/down mode: Timer counts up to TAxCCR0 then down to 0000h
3	Reserved	RW	0h	Reserved
2	TACLR	RW	0h	Timer_A clear. Setting this bit resets TAxR, the timer clock divider, and the count direction. The TACLR bit is automatically reset and is always read as zero.
1	TAIE	RW	0h	Timer_A interrupt enable. This bit enables the TAIFG interrupt request. 0b = Interrupt disabled 1b = Interrupt enabled
0	TAIFG	RW	0h	Timer_A interrupt flag 0b = No interrupt pending 1b = Interrupt pending

9.3.2 TAxR Register (address = 0170h) [reset = 0000h]

Timer_Ax Counter Register

Figure 9-17. TAxR Register

15	14	13	12	11	10	9	8
TAxR							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
TAxR							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 9-5. TAxR Register Description

Bit	Field	Type	Reset	Description
15-0	TAxR	RW	0h	Timer_A register. The TAxR register is the count of Timer_A.

9.3.3 TAxCCTL0 Register (address = 0162h) [reset = 0000h]

Timer_Ax Capture/Compare Control 0 Register

Figure 9-18. TAxCCTL0 Register

15	14	13	12	11	10	9	8
CM		CCIS		SCS	SCCI	Reserved	CAP
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)	r-(0)	rw-(0)
7	6	5	4	3	2	1	0
OUTMOD			CCIE	CCI	OUT	COV	CCIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)

Table 9-6. TAxCCTL0 Register Description

Bit	Field	Type	Reset	Description
15-14	CM	RW	0h	Capture mode 00b = No capture 01b = Capture on rising edge 10b = Capture on falling edge 11b = Capture on both rising and falling edges
13-12	CCIS	RW	0h	Capture/compare input select. These bits select the TAxCCR0 input signal. See the device-specific data sheet for specific signal connections. 00b = CCIxA 01b = CCIxB 10b = GND 11b = VCC
11	SCS	RW	0h	Synchronize capture source. This bit is used to synchronize the capture input signal with the timer clock. 0b = Asynchronous capture 1b = Synchronous capture
10	SCCI	RW	0h	Synchronized capture/compare input. The selected CCI input signal is latched with the EQUx signal and can be read via this bit.
9	Reserved	R	0h	Reserved. Reads as 0.
8	CAP	RW	0h	Capture mode 0b = Compare mode 1b = Capture mode
7-5	OUTMOD	RW	0h	Output mode. Modes 2, 3, 6, and 7 are not useful for TAxCCR0 because EQUx = EQU0. 000b = OUT bit value 001b = Set 010b = Toggle/reset 011b = Set/reset 100b = Toggle 101b = Reset 110b = Toggle/set 111b = Reset/set
4	CCIE	RW	0h	Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag. 0b = Interrupt disabled 1b = Interrupt enabled
3	CCI	R	0h	Capture/compare input. The selected input signal can be read by this bit.
2	OUT	RW	0h	Output. For output mode 0, this bit directly controls the state of the output. 0b = Output low 1b = Output high

Table 9-6. TAxCCTL0 Register Description (continued)

Bit	Field	Type	Reset	Description
1	COV	RW	0h	Capture overflow. This bit indicates a capture overflow occurred. COV must be reset with software. 0b = No capture overflow occurred 1b = Capture overflow occurred
0	CCIFG	RW	0h	Capture/compare interrupt flag 0b = No interrupt pending 1b = Interrupt pending

9.3.4 TAxCCTL1 Register (address = 0164h) [reset = 0000h]

Timer_Ax Capture/Compare Control 1 Register

Figure 9-19. TAxCCTL1 Register

15	14	13	12	11	10	9	8
CM		CCIS		SCS	SCCI	Reserved	CAP
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)	r-(0)	rw-(0)
7	6	5	4	3	2	1	0
OUTMOD			CCIE	CCI	OUT	COV	CCIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)

Table 9-7. TAxCCTL1 Register Description

Bit	Field	Type	Reset	Description
15-14	CM	RW	0h	Capture mode 00b = No capture 01b = Capture on rising edge 10b = Capture on falling edge 11b = Capture on both rising and falling edges
13-12	CCIS	RW	0h	Capture/compare input select. These bits select the TAxCCR1 input signal. See the device-specific data sheet for specific signal connections. 00b = CCIxA 01b = CCIxB 10b = GND 11b = VCC
11	SCS	RW	0h	Synchronize capture source. This bit is used to synchronize the capture input signal with the timer clock. 0b = Asynchronous capture 1b = Synchronous capture
10	SCCI	RW	0h	Synchronized capture/compare input. The selected CCI input signal is latched with the EQUx signal and can be read via this bit.
9	Reserved	R	0h	Reserved. Reads as 0.
8	CAP	RW	0h	Capture mode 0b = Compare mode 1b = Capture mode
7-5	OUTMOD	RW	0h	Output mode 000b = OUT bit value 001b = Set 010b = Toggle/reset 011b = Set/reset 100b = Toggle 101b = Reset 110b = Toggle/set 111b = Reset/set
4	CCIE	RW	0h	Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag. 0b = Interrupt disabled 1b = Interrupt enabled
3	CCI	R	0h	Capture/compare input. The selected input signal can be read by this bit.
2	OUT	RW	0h	Output. For output mode 0, this bit directly controls the state of the output. 0b = Output low 1b = Output high

Table 9-7. TAxCCTL1 Register Description (continued)

Bit	Field	Type	Reset	Description
1	COV	RW	0h	Capture overflow. This bit indicates a capture overflow occurred. COV must be reset with software. 0b = No capture overflow occurred 1b = Capture overflow occurred
0	CCIFG	RW	0h	Capture/compare interrupt flag 0b = No interrupt pending 1b = Interrupt pending

9.3.5 TAxCCTL2 Register (address = 0166h) [reset = 0000h]

Timer_Ax Capture/Compare Control 2 Register

Figure 9-20. TAxCCTL2 Register

15	14	13	12	11	10	9	8
CM		CCIS		SCS	SCCI	Reserved	CAP
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)	r-(0)	rw-(0)
7	6	5	4	3	2	1	0
OUTMOD			CCIE	CCI	OUT	COV	CCIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)

Table 9-8. TAxCCTL2 Register Description

Bit	Field	Type	Reset	Description
15-14	CM	RW	0h	Capture mode 00b = No capture 01b = Capture on rising edge 10b = Capture on falling edge 11b = Capture on both rising and falling edges
13-12	CCIS	RW	0h	Capture/compare input select. These bits select the TAxCCR2 input signal. See the device-specific data sheet for specific signal connections. 00b = CCIxA 01b = CCIxB 10b = GND 11b = VCC
11	SCS	RW	0h	Synchronize capture source. This bit is used to synchronize the capture input signal with the timer clock. 0b = Asynchronous capture 1b = Synchronous capture
10	SCCI	RW	0h	Synchronized capture/compare input. The selected CCI input signal is latched with the EQUx signal and can be read via this bit.
9	Reserved	R	0h	Reserved. Reads as 0.
8	CAP	RW	0h	Capture mode 0b = Compare mode 1b = Capture mode
7-5	OUTMOD	RW	0h	Output mode 000b = OUT bit value 001b = Set 010b = Toggle/reset 011b = Set/reset 100b = Toggle 101b = Reset 110b = Toggle/set 111b = Reset/set
4	CCIE	RW	0h	Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag. 0b = Interrupt disabled 1b = Interrupt enabled
3	CCI	R	0h	Capture/compare input. The selected input signal can be read by this bit.
2	OUT	RW	0h	Output. For output mode 0, this bit directly controls the state of the output. 0b = Output low 1b = Output high

Table 9-8. TAxCCTL2 Register Description (continued)

Bit	Field	Type	Reset	Description
1	COV	RW	0h	Capture overflow. This bit indicates a capture overflow occurred. COV must be reset with software. 0b = No capture overflow occurred 1b = Capture overflow occurred
0	CCIFG	RW	0h	Capture/compare interrupt flag 0b = No interrupt pending 1b = Interrupt pending

9.3.6 TAxCCTL3 Register (address = 0168h) [reset = 0000h]

Timer_Ax Capture/Compare Control 3 Register

Figure 9-21. TAxCCTL3 Register

15	14	13	12	11	10	9	8
CM		CCIS		SCS	SCCI	Reserved	CAP
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)	r-(0)	rw-(0)
7	6	5	4	3	2	1	0
OUTMOD			CCIE	CCI	OUT	COV	CCIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)

Table 9-9. TAxCCTL3 Register Description

Bit	Field	Type	Reset	Description
15-14	CM	RW	0h	Capture mode 00b = No capture 01b = Capture on rising edge 10b = Capture on falling edge 11b = Capture on both rising and falling edges
13-12	CCIS	RW	0h	Capture/compare input select. These bits select the TAxCCR3 input signal. See the device-specific data sheet for specific signal connections. 00b = CCIxA 01b = CCIxB 10b = GND 11b = VCC
11	SCS	RW	0h	Synchronize capture source. This bit is used to synchronize the capture input signal with the timer clock. 0b = Asynchronous capture 1b = Synchronous capture
10	SCCI	RW	0h	Synchronized capture/compare input. The selected CCI input signal is latched with the EQUx signal and can be read via this bit.
9	Reserved	R	0h	Reserved. Reads as 0.
8	CAP	RW	0h	Capture mode 0b = Compare mode 1b = Capture mode
7-5	OUTMOD	RW	0h	Output mode 000b = OUT bit value 001b = Set 010b = Toggle/reset 011b = Set/reset 100b = Toggle 101b = Reset 110b = Toggle/set 111b = Reset/set
4	CCIE	RW	0h	Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag. 0b = Interrupt disabled 1b = Interrupt enabled
3	CCI	R	0h	Capture/compare input. The selected input signal can be read by this bit.
2	OUT	RW	0h	Output. For output mode 0, this bit directly controls the state of the output. 0b = Output low 1b = Output high

Table 9-9. TAxCCTL3 Register Description (continued)

Bit	Field	Type	Reset	Description
1	COV	RW	0h	Capture overflow. This bit indicates a capture overflow occurred. COV must be reset with software. 0b = No capture overflow occurred 1b = Capture overflow occurred
0	CCIFG	RW	0h	Capture/compare interrupt flag 0b = No interrupt pending 1b = Interrupt pending

9.3.7 TAxCCTL4 Register (address = 016Ah) [reset = 0000h]

Timer_Ax Capture/Compare Control 4 Register

Figure 9-22. TAxCCTL4 Register

15	14	13	12	11	10	9	8
CM		CCIS		SCS	SCCI	Reserved	CAP
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)	r-(0)	rw-(0)
7	6	5	4	3	2	1	0
OUTMOD			CCIE	CCI	OUT	COV	CCIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)

Table 9-10. TAxCCTL4 Register Description

Bit	Field	Type	Reset	Description
15-14	CM	RW	0h	Capture mode 00b = No capture 01b = Capture on rising edge 10b = Capture on falling edge 11b = Capture on both rising and falling edges
13-12	CCIS	RW	0h	Capture/compare input select. These bits select the TAxCCR4 input signal. See the device-specific data sheet for specific signal connections. 00b = CCIxA 01b = CCIxB 10b = GND 11b = VCC
11	SCS	RW	0h	Synchronize capture source. This bit is used to synchronize the capture input signal with the timer clock. 0b = Asynchronous capture 1b = Synchronous capture
10	SCCI	RW	0h	Synchronized capture/compare input. The selected CCI input signal is latched with the EQUx signal and can be read via this bit.
9	Reserved	R	0h	Reserved. Reads as 0.
8	CAP	RW	0h	Capture mode 0b = Compare mode 1b = Capture mode
7-5	OUTMOD	RW	0h	Output mode 000b = OUT bit value 001b = Set 010b = Toggle/reset 011b = Set/reset 100b = Toggle 101b = Reset 110b = Toggle/set 111b = Reset/set
4	CCIE	RW	0h	Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag. 0b = Interrupt disabled 1b = Interrupt enabled
3	CCI	R	0h	Capture/compare input. The selected input signal can be read by this bit.
2	OUT	RW	0h	Output. For output mode 0, this bit directly controls the state of the output. 0b = Output low 1b = Output high

Table 9-10. TAxCCTL4 Register Description (continued)

Bit	Field	Type	Reset	Description
1	COV	RW	0h	Capture overflow. This bit indicates a capture overflow occurred. COV must be reset with software. 0b = No capture overflow occurred 1b = Capture overflow occurred
0	CCIFG	RW	0h	Capture/compare interrupt flag 0b = No interrupt pending 1b = Interrupt pending

9.3.8 TAxCCTL5 Register (address = 016Ch) [reset = 0000h]

Timer_Ax Capture/Compare Control 5 Register

Figure 9-23. TAxCCTL5 Register

15	14	13	12	11	10	9	8
CM		CCIS		SCS	SCCI	Reserved	CAP
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)	r-(0)	rw-(0)
7	6	5	4	3	2	1	0
OUTMOD			CCIE	CCI	OUT	COV	CCIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)

Table 9-11. TAxCCTL5 Register Description

Bit	Field	Type	Reset	Description
15-14	CM	RW	0h	Capture mode 00b = No capture 01b = Capture on rising edge 10b = Capture on falling edge 11b = Capture on both rising and falling edges
13-12	CCIS	RW	0h	Capture/compare input select. These bits select the TAxCCR5 input signal. See the device-specific data sheet for specific signal connections. 00b = CCIxA 01b = CCIxB 10b = GND 11b = VCC
11	SCS	RW	0h	Synchronize capture source. This bit is used to synchronize the capture input signal with the timer clock. 0b = Asynchronous capture 1b = Synchronous capture
10	SCCI	RW	0h	Synchronized capture/compare input. The selected CCI input signal is latched with the EQUx signal and can be read via this bit.
9	Reserved	R	0h	Reserved. Reads as 0.
8	CAP	RW	0h	Capture mode 0b = Compare mode 1b = Capture mode
7-5	OUTMOD	RW	0h	Output mode 000b = OUT bit value 001b = Set 010b = Toggle/reset 011b = Set/reset 100b = Toggle 101b = Reset 110b = Toggle/set 111b = Reset/set
4	CCIE	RW	0h	Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag. 0b = Interrupt disabled 1b = Interrupt enabled
3	CCI	R	0h	Capture/compare input. The selected input signal can be read by this bit.
2	OUT	RW	0h	Output. For output mode 0, this bit directly controls the state of the output. 0b = Output low 1b = Output high

Table 9-11. TAxCCTL5 Register Description (continued)

Bit	Field	Type	Reset	Description
1	COV	RW	0h	Capture overflow. This bit indicates a capture overflow occurred. COV must be reset with software. 0b = No capture overflow occurred 1b = Capture overflow occurred
0	CCIFG	RW	0h	Capture/compare interrupt flag 0b = No interrupt pending 1b = Interrupt pending

9.3.9 TAxCCTL6 Register (address = 016Eh) [reset = 0000h]

Timer_Ax Capture/Compare Control 6 Register

Figure 9-24. TAxCCTL6 Register

15	14	13	12	11	10	9	8
CM		CCIS		SCS	SCCI	Reserved	CAP
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)	r-(0)	rw-(0)
7	6	5	4	3	2	1	0
OUTMOD			CCIE	CCI	OUT	COV	CCIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)

Table 9-12. TAxCCTL6 Register Description

Bit	Field	Type	Reset	Description
15-14	CM	RW	0h	Capture mode 00b = No capture 01b = Capture on rising edge 10b = Capture on falling edge 11b = Capture on both rising and falling edges
13-12	CCIS	RW	0h	Capture/compare input select. These bits select the TAxCCR6 input signal. See the device-specific data sheet for specific signal connections. 00b = CCIxA 01b = CCIxB 10b = GND 11b = VCC
11	SCS	RW	0h	Synchronize capture source. This bit is used to synchronize the capture input signal with the timer clock. 0b = Asynchronous capture 1b = Synchronous capture
10	SCCI	RW	0h	Synchronized capture/compare input. The selected CCI input signal is latched with the EQUx signal and can be read via this bit.
9	Reserved	R	0h	Reserved. Reads as 0.
8	CAP	RW	0h	Capture mode 0b = Compare mode 1b = Capture mode
7-5	OUTMOD	RW	0h	Output mode 000b = OUT bit value 001b = Set 010b = Toggle/reset 011b = Set/reset 100b = Toggle 101b = Reset 110b = Toggle/set 111b = Reset/set
4	CCIE	RW	0h	Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag. 0b = Interrupt disabled 1b = Interrupt enabled
3	CCI	R	0h	Capture/compare input. The selected input signal can be read by this bit.
2	OUT	RW	0h	Output. For output mode 0, this bit directly controls the state of the output. 0b = Output low 1b = Output high

Table 9-12. TAxCCTL6 Register Description (continued)

Bit	Field	Type	Reset	Description
1	COV	RW	0h	Capture overflow. This bit indicates a capture overflow occurred. COV must be reset with software. 0b = No capture overflow occurred 1b = Capture overflow occurred
0	CCIFG	RW	0h	Capture/compare interrupt flag 0b = No interrupt pending 1b = Interrupt pending

9.3.10 TAxCCR0 Register (address = 0172h) [reset = 0000h]

Timer_A Capture/Compare 0 Register

Figure 9-25. TAxCCR0 Register

15	14	13	12	11	10	9	8
TAxCCR0							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
TAxCCR0							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 9-13. TAxCCR0 Register Description

Bit	Field	Type	Reset	Description
15-0	TAxCCR0	RW	0h	Compare mode: TAxCCR0 holds the data for the comparison to the timer value in the Timer_A Register, TAR. Capture mode: The Timer_A Register, TAR, is copied into the TAxCCR0 register when a capture is performed.

9.3.11 TAxCCR1 Register (address = 0174h) [reset = 0000h]

Timer_A Capture/Compare 1 Register

Figure 9-26. TAxCCR1 Register

15	14	13	12	11	10	9	8
TAxCCR1							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
TAxCCR1							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 9-14. TAxCCR1 Register Description

Bit	Field	Type	Reset	Description
15-0	TAxCCR1	RW	0h	Compare mode: TAxCCR1 holds the data for the comparison to the timer value in the Timer_A Register, TAR. Capture mode: The Timer_A Register, TAR, is copied into the TAxCCR1 register when a capture is performed.

9.3.12 TAxCCR2 Register (address = 0176h) [reset = 0000h]

Timer_A Capture/Compare 2 Register

Figure 9-27. TAxCCR2 Register

15	14	13	12	11	10	9	8
TAxCCR2							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
TAxCCR2							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 9-15. TAxCCR2 Register Description

Bit	Field	Type	Reset	Description
15-0	TAxCCR2	RW	0h	Compare mode: TAxCCR2 holds the data for the comparison to the timer value in the Timer_A Register, TAR. Capture mode: The Timer_A Register, TAR, is copied into the TAxCCR2 register when a capture is performed.

9.3.13 TAxCCR3 Register (address = 0178h) [reset = 0000h]

Timer_A Capture/Compare 3 Register

Figure 9-28. TAxCCR3 Register

15	14	13	12	11	10	9	8
TAxCCR3							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
TAxCCR3							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 9-16. TAxCCR3 Register Description

Bit	Field	Type	Reset	Description
15-0	TAxCCR3	RW	0h	Compare mode: TAxCCR3 holds the data for the comparison to the timer value in the Timer_A Register, TAR. Capture mode: The Timer_A Register, TAR, is copied into the TAxCCR3 register when a capture is performed.

9.3.14 TAxCCR4 Register (address = 017Ah) [reset = 0000h]

Timer_A Capture/Compare 4 Register

Figure 9-29. TAxCCR4 Register

15	14	13	12	11	10	9	8
TAxCCR4							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
TAxCCR4							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 9-17. TAxCCR4 Register Description

Bit	Field	Type	Reset	Description
15-0	TAxCCR4	RW	0h	Compare mode: TAxCCR4 holds the data for the comparison to the timer value in the Timer_A Register, TAR. Capture mode: The Timer_A Register, TAR, is copied into the TAxCCR4 register when a capture is performed.

9.3.15 TAxCCR5 Register (address = 017Ch) [reset = 0000h]

Timer_A Capture/Compare 5 Register

Figure 9-30. TAxCCR5 Register

15	14	13	12	11	10	9	8
TAxCCR5							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
TAxCCR5							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 9-18. TAxCCR5 Register Description

Bit	Field	Type	Reset	Description
15-0	TAxCCR5	RW	0h	Compare mode: TAxCCR5 holds the data for the comparison to the timer value in the Timer_A Register, TAR. Capture mode: The Timer_A Register, TAR, is copied into the TAxCCR5 register when a capture is performed.

9.3.16 TAxCCR6 Register (address = 017Eh) [reset = 0000h]

Timer_A Capture/Compare 6 Register

Figure 9-31. TAxCCR6 Register

15	14	13	12	11	10	9	8
TAxCCR6							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
TAxCCR6							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 9-19. TAxCCR6 Register Description

Bit	Field	Type	Reset	Description
15-0	TAxCCR6	RW	0h	Compare mode: TAxCCR6 holds the data for the comparison to the timer value in the Timer_A Register, TAR. Capture mode: The Timer_A Register, TAR, is copied into the TAxCCR6 register when a capture is performed.

9.3.17 TAxIV Register (address = 012Eh) [reset = 0000h]

Timer_Ax Interrupt Vector Register

Figure 9-32. TAxIV Register

15	14	13	12	11	10	9	8
TAIV							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
TAIV							
r0	r0	r0	r0	r-(0)	r-(0)	r-(0)	r0

Table 9-20. TAxIV Register Description

Bit	Field	Type	Reset	Description
15-0	TAIV	R	0h	Timer_A interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: Capture/compare 1; Interrupt Flag: TAxCCR1 CCIFG; Interrupt Priority: Highest 04h = Interrupt Source: Capture/compare 2; Interrupt Flag: TAxCCR2 CCIFG 06h = Interrupt Source: Capture/compare 3; Interrupt Flag: TAxCCR3 CCIFG 08h = Interrupt Source: Capture/compare 4; Interrupt Flag: TAxCCR4 CCIFG 0Ah = Interrupt Source: Capture/compare 5; Interrupt Flag: TAxCCR5 CCIFG 0Ch = Interrupt Source: Capture/compare 6; Interrupt Flag: TAxCCR6 CCIFG 0Eh = Interrupt Source: Timer overflow; Interrupt Flag: TAxCTL TAIFG; Interrupt Priority: Lowest

Enhanced Universal Serial Communication Interface (eUSCI) – UART Mode

The enhanced universal serial communication interface A (eUSCI_A) supports multiple serial communication modes with one hardware module. This chapter discusses the operation of the asynchronous UART mode.

Topic **Page**

10.1 Enhanced Universal Serial Communication Interface A (eUSCI_A) Overview

The eUSCI_A module supports two serial communication modes:

- UART mode
- SPI mode

10.2 eUSCI_A Introduction – UART Mode

In asynchronous mode, the eUSCI_Ax modules connect the device to an external system through two external pins, UCAxRXD and UCAxTXD. UART mode is selected when the UCSYNC bit is cleared.

UART mode features include:

- 7-bit or 8-bit data with odd, even, or non-parity
- Independent transmit and receive shift registers
- Separate transmit and receive buffer registers
- LSB-first or MSB-first data transmit and receive
- Built-in idle-line and address-bit communication protocols for multiprocessor systems
- Receiver start-edge detection for auto wake up from LPMx modes
- Programmable baud rate with modulation for fractional baud-rate support
- Status flags for error detection and suppression
- Status flags for address detection
- Independent interrupt capability for receive, transmit, start bit received, and transmit complete

Figure 10-1 shows the eUSCI_Ax when configured for UART mode.

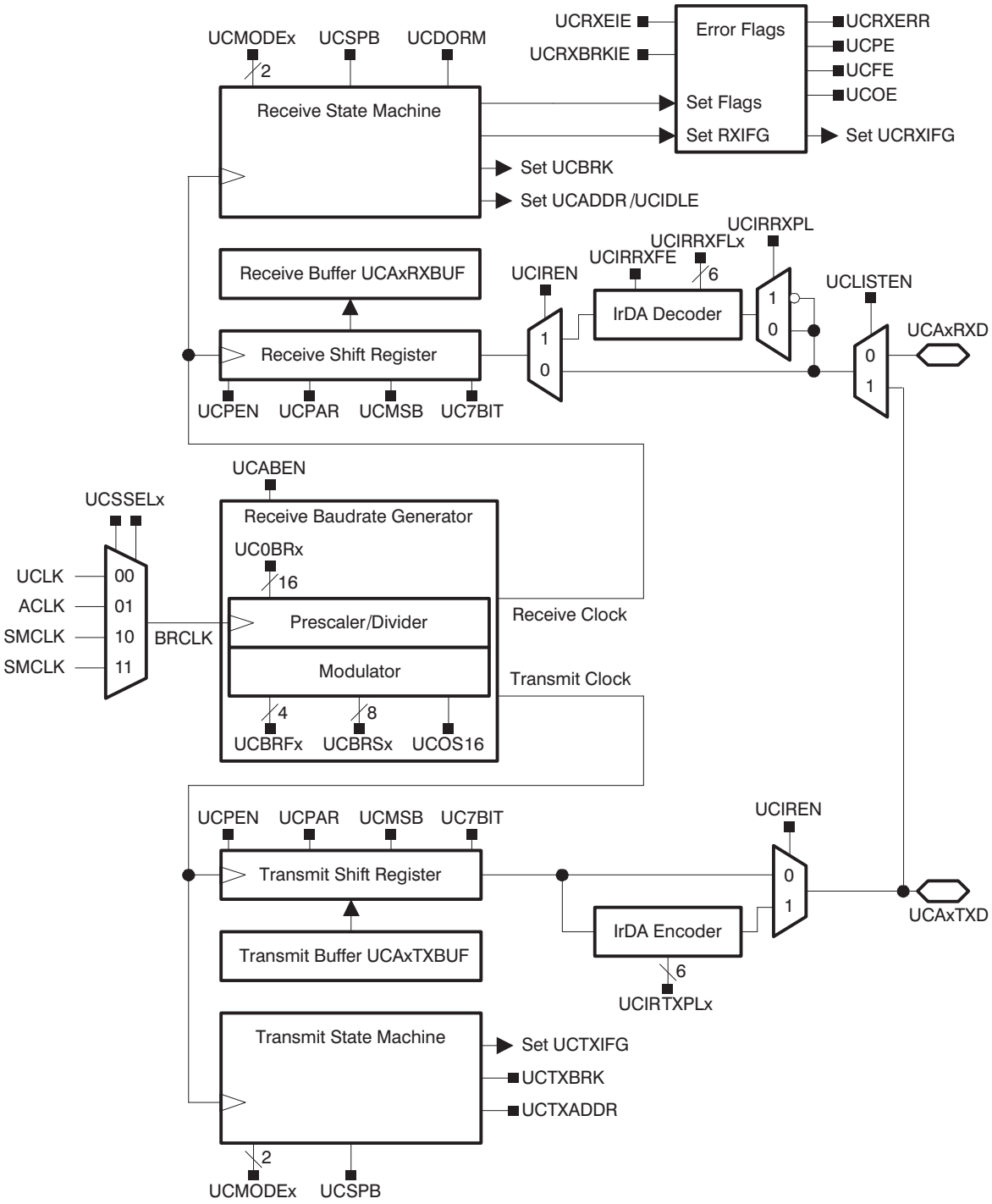


Figure 10-1. eUSCI_Ax Block Diagram – UART Mode (UCSYNC = 0)

10.3 eUSCI_A Operation – UART Mode

In UART mode, the eUSCI_A transmits and receives characters at a bit rate asynchronous to another device. Timing for each character is based on the selected baud rate of the eUSCI_A. The transmit and receive functions use the same baud-rate frequency.

10.3.1 eUSCI_A Initialization and Reset

The eUSCI_A is reset by a PUC or by setting the UCSWRST bit. After a PUC, the UCSWRST bit is automatically set, keeping the eUSCI_A in a reset condition. When set, the UCSWRST bit sets the UCTXIFG bit and resets the UCRXIE, UCTXIE, UCRXIFG, UCRXERR, UCBRK, UCPE, UCOE, UCFE, UCSTOE, and UCBTOE bits. Clearing UCSWRST releases the eUSCI_A for operation.

NOTE: Initializing or reconfiguring the eUSCI_A module

The recommended eUSCI_A initialization or reconfiguration process is:

1. Set UCSWRST (`BIS.B #UCSWRST,&UCAxCTL1`).
2. Initialize all eUSCI_A registers with UCSWRST = 1 (including UCAxCTL1).
3. Configure ports.
4. Clear UCSWRST via software (`BIC.B #UCSWRST,&UCAxCTL1`).
5. Enable interrupts (optional) via UCRXIE and/or UCTXIE.

10.3.2 Character Format

The UART character format (see Figure 10-2) consists of a start bit, seven or eight data bits, an even, odd, or no parity bit, an address bit (address-bit mode), and one or two stop bits. The UCMSB bit controls the direction of the transfer and selects LSB or MSB first. LSB first is typically required for UART communication.

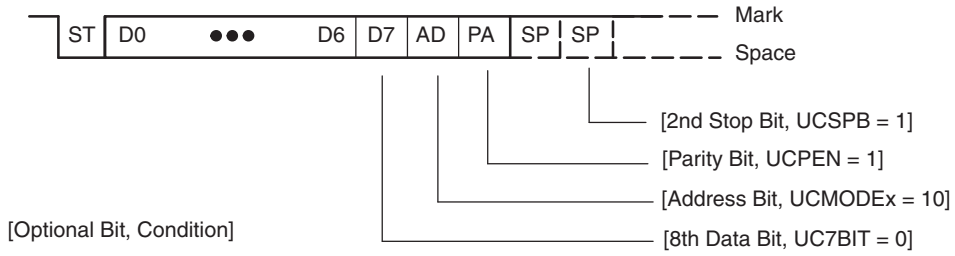


Figure 10-2. Character Format

10.3.3 Asynchronous Communication Format

When two devices communicate asynchronously, no multiprocessor format is required for the protocol. When three or more devices communicate, the eUSCI_A supports the idle-line and address-bit multiprocessor communication formats.

10.3.3.1 Idle-Line Multiprocessor Format

When UCMODEx = 01, the idle-line multiprocessor format is selected. Blocks of data are separated by an idle time on the transmit or receive lines (see Figure 10-3). An idle receive line is detected when ten or more continuous ones (marks) are received after the one or two stop bits of a character. The baud-rate generator is switched off after reception of an idle line until the next start edge is detected. When an idle line is detected, the UCIDLE bit is set.

The first character received after an idle period is an address character. The UCIDLE bit is used as an address tag for each block of characters. In idle-line multiprocessor format, this bit is set when a received character is an address.

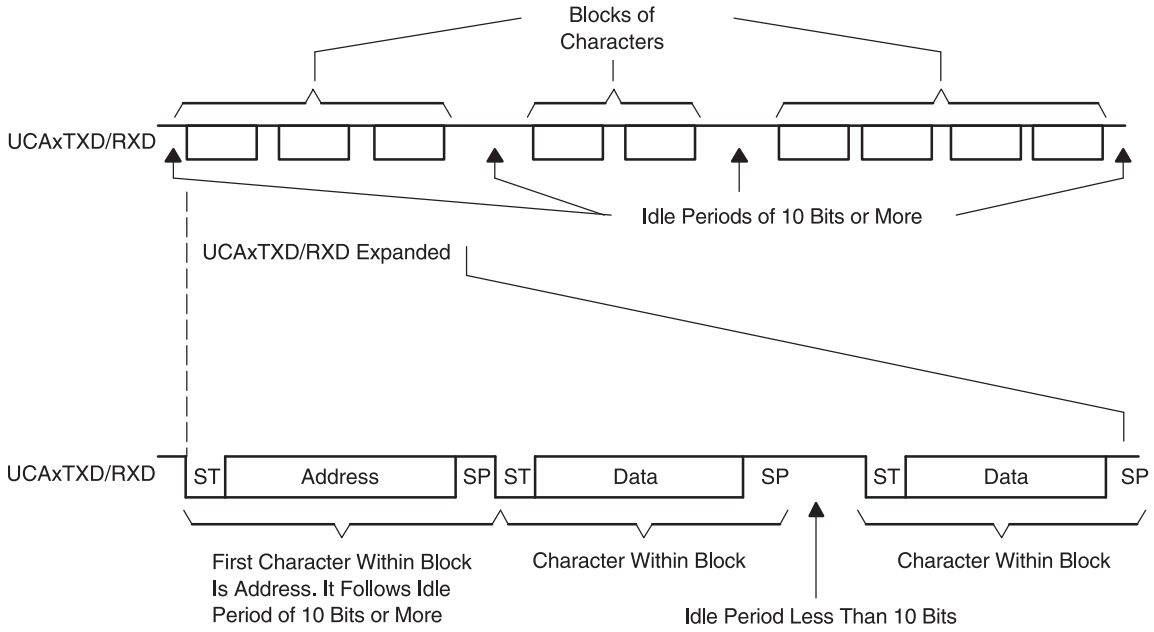


Figure 10-3. Idle-Line Format

The UCDORM bit is used to control data reception in the idle-line multiprocessor format. When UCDORM = 1, all non-address characters are assembled but not transferred into the UCAxRXBUF, and interrupts are not generated. When an address character is received, the character is transferred into UCAxRXBUF, UCRXIFG is set, and any applicable error flag is set when UCRXEIE = 1. When UCRXEIE = 0 and an address character is received but has a framing error or parity error, the character is not transferred into UCAxRXBUF and UCRXIFG is not set.

If an address is received, user software can validate the address and must reset UCDORM to continue receiving data. If UCDORM remains set, only address characters are received. When UCDORM is cleared during the reception of a character, the receive interrupt flag is set after the reception completed. The UCDORM bit is not modified automatically by the eUSCI_A hardware.

For address transmission in idle-line multiprocessor format, a precise idle period can be generated by the eUSCI_A to generate address character identifiers on UCAxTXD. The double-buffered UCTXADDR flag indicates if the next character loaded into UCAxTXBUF is preceded by an idle line of 11 bits. UCTXADDR is automatically cleared when the start bit is generated.

10.3.3.1.1 Transmitting an Idle Frame

The following procedure sends out an idle frame to indicate an address character followed by associated data:

1. Set UCTXADDR, then write the address character to UCAxTXBUF. UCAxTXBUF must be ready for new data (UCTXIFG = 1).

   This generates an idle period of exactly 11 bits followed by the address character. UCTXADDR is reset automatically when the address character is transferred from UCAxTXBUF into the shift register.
2. Write desired data characters to UCAxTXBUF. UCAxTXBUF must be ready for new data (UCTXIFG = 1).

   The data written to UCAxTXBUF is transferred to the shift register and transmitted as soon as the shift register is ready for new data.

   The idle-line time must not be exceeded between address and data transmission or between data transmissions. Otherwise, the transmitted data is misinterpreted as an address.

10.3.3.2 Address-Bit Multiprocessor Format

When UCMODEx = 10, the address-bit multiprocessor format is selected. Each processed character contains an extra bit used as an address indicator (see Figure 10-4). The first character in a block of characters carries a set address bit that indicates that the character is an address. The eUSCI_A UCADDR bit is set when a received character has its address bit set and is transferred to UCAxRXBUF.

The UCDORM bit is used to control data reception in the address-bit multiprocessor format. When UCDORM is set, data characters with address bit = 0 are assembled by the receiver but are not transferred to UCAxRXBUF and no interrupts are generated. When a character containing a set address bit is received, the character is transferred into UCAxRXBUF, UCRXIFG is set, and any applicable error flag is set when UCRXEIE = 1. When UCRXEIE = 0 and a character containing a set address bit is received but has a framing error or parity error, the character is not transferred into UCAxRXBUF and UCRXIFG is not set.

If an address is received, user software can validate the address and must reset UCDORM to continue receiving data. If UCDORM remains set, only address characters with address bit = 1 are received. The UCDORM bit is not modified by the eUSCI_A hardware automatically.

When UCDORM = 0, all received characters set the receive interrupt flag UCRXIFG. If UCDORM is cleared during the reception of a character, the receive interrupt flag is set after the reception is completed.

For address transmission in address-bit multiprocessor mode, the address bit of a character is controlled by the UCTXADDR bit. The value of the UCTXADDR bit is loaded into the address bit of the character transferred from UCAxTXBUF to the transmit shift register. UCTXADDR is automatically cleared when the start bit is generated.

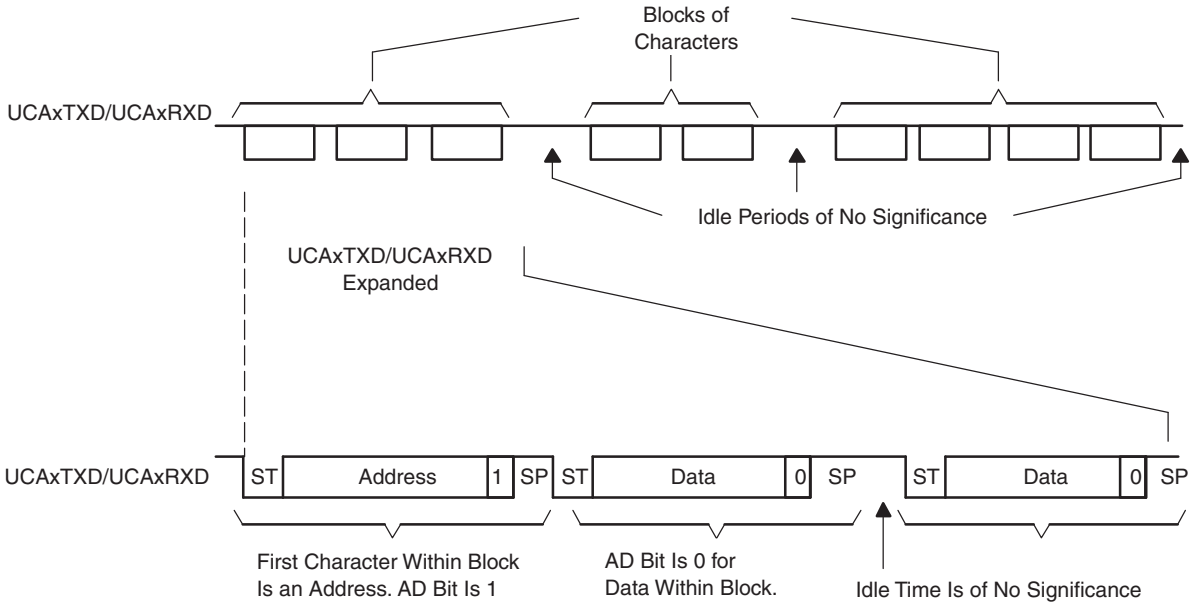


Figure 10-4. Address-Bit Multiprocessor Format

10.3.3.2.1 Break Reception and Generation

When UCMODEx = 00, 01, or 10, the receiver detects a break when all data, parity, and stop bits are low, regardless of the parity, address mode, or other character settings. When a break is detected, the UCBRK bit is set. If the break interrupt enable bit (UCBRKIE) is set, the receive interrupt flag UCRXIFG is also set. In this case, the value in UCAxRXBUF is 0h, because all data bits were zero.

To transmit a break, set the UCTXBRK bit, then write 0h to UCAxTXBUF. UCAxTXBUF must be ready for new data (UCTXIFG = 1). This generates a break with all bits low. UCTXBRK is automatically cleared when the start bit is generated.

10.3.4 Automatic Baud-Rate Detection

When UCMODEx = 11, UART mode with automatic baud-rate detection is selected. For automatic baud-rate detection, a data frame is preceded by a synchronization sequence that consists of a break and a synch field. A break is detected when 11 or more continuous zeros (spaces) are received. If the length of the break exceeds 21 bit times, the break timeout error flag UCBTOE is set. The eUSCI_A cannot transmit data while receiving the break/sync field. The synch field follows the break as shown in Figure 10-5.

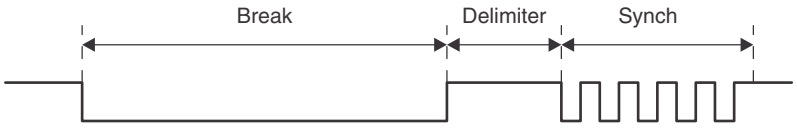


Figure 10-5. Auto Baud-Rate Detection – Break/Synch Sequence

For LIN conformance, the character format should be set to eight data bits, LSB first, no parity, and one stop bit. No address bit is available.

The synch field consists of the data 055h inside a byte field (see Figure 10-6). The synchronization is based on the time measurement between the first falling edge and the last falling edge of the pattern. The transmit baud-rate generator is used for the measurement if automatic baud-rate detection is enabled by setting UCABDEN. Otherwise, the pattern is received but not measured. The result of the measurement is transferred into the baud-rate control registers (UCAxBRW and UCAxMCTLW). If the length of the synch field exceeds the measurable time, the synch timeout error flag UCSTOE is set.

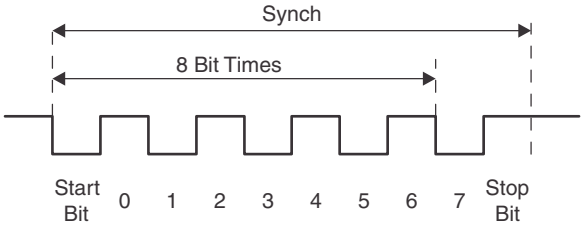


Figure 10-6. Auto Baud-Rate Detection – Synch Field

The UCDORM bit is used to control data reception in this mode. When UCDORM is set, all characters are received but not transferred into the UCAxRXBUF, and interrupts are not generated. When a break/synch field is detected, the UCBRK flag is set. The character following the break/synch field is transferred into UCAxRXBUF and the UCRXIFG interrupt flag is set. Any applicable error flag is also set. If the UCBRKIE bit is set, reception of the break/synch sets the UCRXIFG. The UCBRK bit is reset by user software or by reading the receive buffer UCAxRXBUF.

When a break/synch field is received, user software must reset UCDORM to continue receiving data. If UCDORM remains set, only the character after the next reception of a break/synch field is received. The UCDORM bit is not modified by the eUSCI_A hardware automatically.

When UCDORM = 0, all received characters set the receive interrupt flag UCRXIFG. If UCDORM is cleared during the reception of a character, the receive interrupt flag is set after the reception is complete.

The counter used to detect the baud rate is limited to 0FFFFh (2^{16}) counts. This means the minimum baud rate detectable is 244 baud in oversampling mode and 15 baud in low-frequency mode. The highest detectable baud rate is 1 Mbaud.

The automatic baud-rate detection mode can be used in a full-duplex communication system with some restrictions. The eUSCI_A cannot transmit data while receiving the break/sync field and, if a 0h byte with framing error is received, any data transmitted during this time is corrupted. The latter case can be discovered by checking the received data and the UCFE bit.

10.3.4.1 Transmitting a Break/Synch Field

The following procedure transmits a break/synch field:

1. Set UCTXBRK with UMODEx = 11.
2. Write 055h to UCAxTXBUF. UCAxTXBUF must be ready for new data (UCTXIFG = 1).

   This generates a break field of 13 bits followed by a break delimiter and the synch character. The length of the break delimiter is controlled with the UCDELIMx bits. UCTXBRK is reset automatically when the synch character is transferred from UCAxTXBUF into the shift register.
3. Write desired data characters to UCAxTXBUF. UCAxTXBUF must be ready for new data (UCTXIFG = 1).

   The data written to UCAxTXBUF is transferred to the shift register and transmitted as soon as the shift register is ready for new data.

10.3.5 IrDA Encoding and Decoding

When UCIREN is set, the IrDA encoder and decoder are enabled and provide hardware bit shaping for IrDA communication.

10.3.5.1 IrDA Encoding

The encoder sends a pulse for every zero bit in the transmit bit stream coming from the UART (see Figure 10-7). The pulse duration is defined by UCIRTXPLx bits specifying the number of one-half clock periods of the clock selected by UCIRTXCLK.

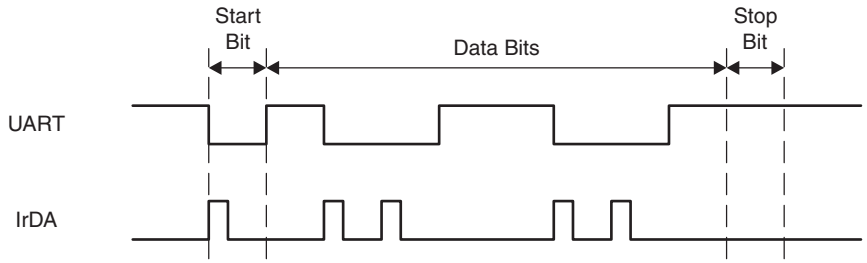


Figure 10-7. Comparison of UART and IrDA Data Formats

To set the pulse time of 3/16 bit period required by the IrDA standard, the BITCLK16 clock is selected with UCIRTXCLK = 1, and the pulse length is set to six one-half clock cycles with UCIRTXPLx = 6 – 1 = 5.

When UCIRTXCLK = 0, the pulse length t_{PULSE} is based on BRCLK and is calculated as:

$$UCIRTXPLx = t_{PULSE} \times 2 \times f_{BRCLK} - 1$$

When UCIRTXCLK = 0, the prescaler UCBRx must be set to a value greater than or equal to 5.

10.3.5.2 IrDA Decoding

The decoder detects high pulses when UCIRRXPL = 0. Otherwise, it detects low pulses. In addition to the analog deglitch filter, an additional programmable digital filter stage can be enabled by setting UCIRRXFE. When UCIRRXFE is set, only pulses longer than the programmed filter length are passed. Shorter pulses are discarded. The equation to program the filter length UCIRRXFLx is:

$$UCIRRXFLx = (t_{PULSE} - t_{WAKE}) \times 2 \times f_{BRCLK} - 4$$

Where:

t_{PULSE} = Minimum receive pulse duration

t_{WAKE} = Wake time from any low-power mode. Zero when the device is in active mode.

10.3.6 Automatic Error Detection

Glitch suppression prevents the eUSCI_A from being accidentally started. Any pulse on UCAxRXD shorter than the deglitch time t_t (selected by UCGLITx) is ignored (see the device-specific data sheet for parameters).

When a low period on UCAxRXD exceeds t_t, a majority vote is taken for the start bit. If the majority vote fails to detect a valid start bit, the eUSCI_A halts character reception and waits for the next low period on UCAxRXD. The majority vote is also used for each bit in a character to prevent bit errors.

The eUSCI_A module automatically detects framing errors, parity errors, overrun errors, and break conditions when receiving characters. The bits UCFE, UCPE, UCOE, and UCBRK are set when their respective condition is detected. When the error flags UCFE, UCPE, or UCOE are set, UCRXERR is also set. The error conditions are described in Table 10-1.

Table 10-1. Receive Error Conditions

Error Condition	Error Flag	Description
Framing error	UCFE	A framing error occurs when a low stop bit is detected. When two stop bits are used, both stop bits are checked for framing error. When a framing error is detected, the UCFE bit is set.
Parity error	UCPE	A parity error is a mismatch between the number of 1s in a character and the value of the parity bit. When an address bit is included in the character, it is included in the parity calculation. When a parity error is detected, the UCPE bit is set.
Receive overrun	UCOE	An overrun error occurs when a character is loaded into UCAxRXBUF before the prior character has been read. When an overrun occurs, the UCOE bit is set.
Break condition	UCBRK	When not using automatic baud-rate detection, a break is detected when all data, parity, and stop bits are low. When a break condition is detected, the UCBRK bit is set. A break condition can also set the interrupt flag UCRXIFG if the break interrupt enable UCBRKIE bit is set.

When UCRXEIE = 0 and a framing error or parity error is detected, no character is received into UCAxRXBUF. When UCRXEIE = 1, characters are received into UCAxRXBUF and any applicable error bit is set.

When any of the UCFE, UCPE, UCOE, UCBRK, or UCRXERR bit is set, the bit remains set until user software resets it or UCAxRXBUF is read. UCOE must be reset by reading UCAxRXBUF. Otherwise, it does not function properly. To detect overflows reliably, the following flow is recommended. After a character is received and UCAxRXIFG is set, first read UCAxSTAT to check the error flags including the overflow flag UCOE. Read UCAxRXBUF next. This clears all error flags except UCOE, if UCAxRXBUF was overwritten between the read access to UCAxSTAT and to UCAxRXBUF. Therefore, the UCOE flag should be checked after reading UCAxRXBUF to detect this condition. Note that, in this case, the UCRXERR flag is not set.

10.3.7 eUSCI_A Receive Enable

The eUSCI_A module is enabled by clearing the UCSWRST bit and the receiver is ready and in an idle state. The receive baud rate generator is in a ready state but is not clocked nor producing any clocks.

The falling edge of the start bit enables the baud rate generator and the UART state machine checks for a valid start bit. If no valid start bit is detected, the UART state machine returns to its idle state and the baud rate generator is turned off again. If a valid start bit is detected, a character is received.

When the idle-line multiprocessor mode is selected with UCMODEx = 01, the UART state machine checks for an idle line after receiving a character. If a start bit is detected, another character is received. Otherwise, the UCIDLE flag is set after 10 ones are received, the UART state machine returns to its idle state, and the baud rate generator is turned off.

10.3.7.1 Receive Data Glitch Suppression

Glitch suppression prevents the eUSCI_A from being accidentally started. Any glitch on UCAxRXD shorter than the deglitch time t_t is ignored by the eUSCI_A, and further action is initiated as shown in Figure 10-8 (see the device-specific data sheet for parameters). The deglitch time t_t can be set to four different values using the UCGLITx bits.

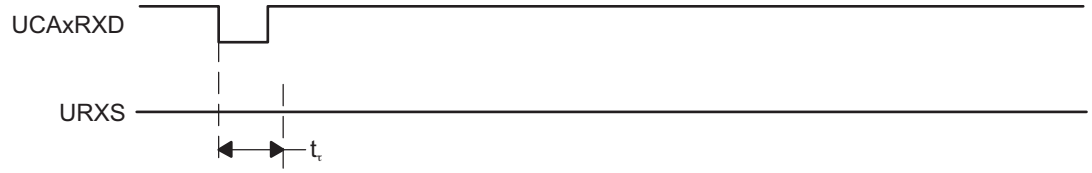


Figure 10-8. Glitch Suppression, eUSCI_A Receive Not Started

When a glitch is longer than t_t, or a valid start bit occurs on UCAxRXD, the eUSCI_A receive operation is started and a majority vote is taken (see Figure 10-9). If the majority vote fails to detect a start bit, the eUSCI_A halts character reception.

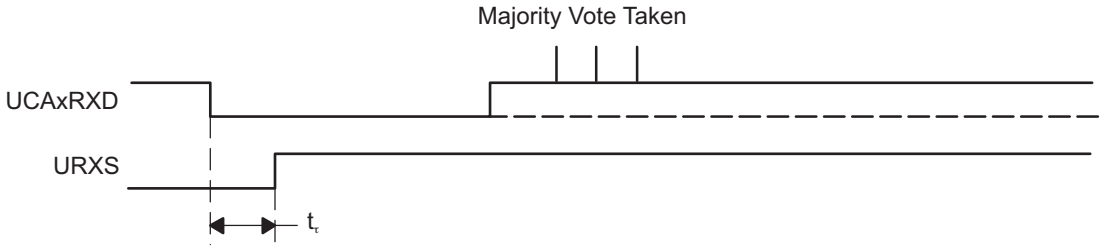


Figure 10-9. Glitch Suppression, eUSCI_A Activated

10.3.8 eUSCI_A Transmit Enable

The eUSCI_A module is enabled by clearing the UCSWRST bit and the transmitter is ready and in an idle state. The transmit baud-rate generator is ready but is not clocked nor producing any clocks.

A transmission is initiated by writing data to UCAxTXBUF. When this occurs, the baud-rate generator is enabled, and the data in UCAxTXBUF is moved to the transmit shift register on the next BITCLK after the transmit shift register is empty. UCTXIFG is set when new data can be written into UCAxTXBUF.

Transmission continues as long as new data is available in UCAxTXBUF at the end of the previous byte transmission. If new data is not in UCAxTXBUF when the previous byte has transmitted, the transmitter returns to its idle state and the baud-rate generator is turned off.

10.3.9 UART Baud-Rate Generation

The eUSCI_A baud-rate generator is capable of producing standard baud rates from nonstandard source frequencies. It provides two modes of operation selected by the UCOS16 bit.

A quick setup for finding the correct baud-rate settings for the eUSCI_A can be found in Section 10.3.10.

10.3.9.1 Low-Frequency Baud-Rate Generation

The low-frequency mode is selected when UCOS16 = 0. This mode allows generation of baud rates from low-frequency clock sources (for example, 9600 baud from a 32768-Hz crystal). By using a lower input frequency, the power consumption of the module is reduced. Using this mode with higher frequencies and higher prescaler settings causes the majority votes to be taken in an increasingly smaller window and, thus, decrease the benefit of the majority vote.

In low-frequency mode, the baud-rate generator uses one prescaler and one modulator to generate bit clock timing. This combination supports fractional divisors for baud-rate generation. In this mode, the maximum eUSCI_A baud rate is one-third the UART source clock frequency BRCLK.

Timing for each bit is shown in Figure 10-10. For each bit received, a majority vote is taken to determine the bit value. These samples occur at the N/2 – 1/2, N/2, and N/2 + 1/2 BRCLK periods, where N is the number of BRCLKs per BITCLK.

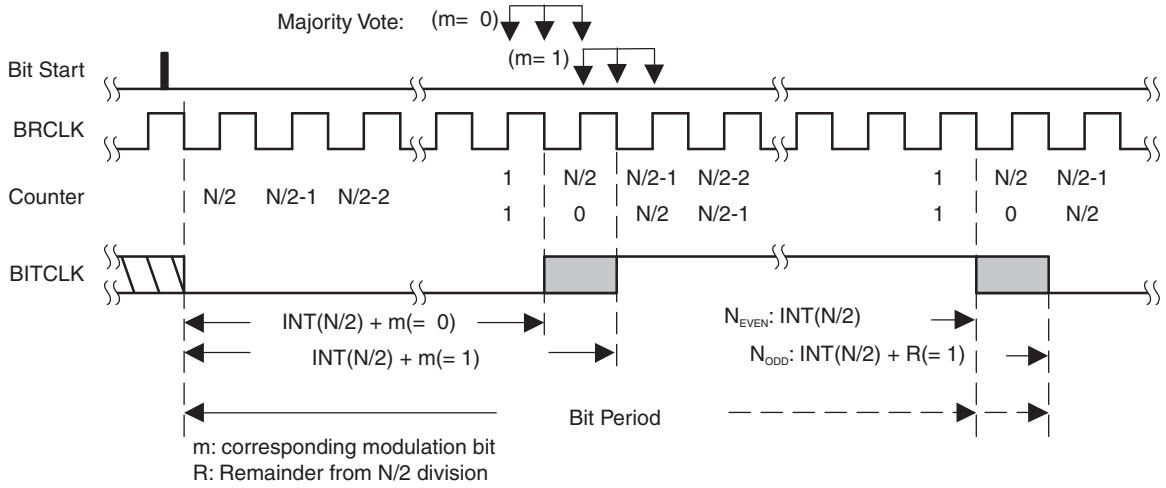


Figure 10-10. BITCLK Baud-Rate Timing With UCOS16 = 0

Modulation is based on the UCBRSx setting as shown in Table 10-2. A 1 in the table indicates that m = 1 and the corresponding BITCLK period is one BRCLK period longer than a BITCLK period with m = 0. The modulation wraps around after 8 bits but restarts with each new start bit.

The correct setting of UCBRSx is described in Section 10.3.10.

Table 10-2. Modulation Pattern Examples

UCBRSx	Bit 0 (Start Bit)	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
00h	0	0	0	0	0	0	0	0
01h	0	0	0	0	0	0	0	1
				⋮				
35h	0	0	1	1	0	1	0	1
36h	0	0	1	1	0	1	1	0
37h	0	0	1	1	0	1	1	1
				⋮				
FFh	1	1	1	1	1	1	1	1

10.3.9.2 Oversampling Baud-Rate Generation

The oversampling mode is selected when UCOS16 = 1. This mode supports sampling a UART bit stream with higher input clock frequencies. This results in majority votes that are always 1/16 of a bit clock period apart. This mode also easily supports IrDA pulses with a 3/16 bit time when the IrDA encoder and decoder are enabled.

This mode uses one prescaler and one modulator to generate the BITCLK16 clock that is 16 times faster than the BITCLK. An additional divider by 16 and modulator stage generates BITCLK from BITCLK16. This combination supports fractional divisions of both BITCLK16 and BITCLK for baud-rate generation. In this mode, the maximum eUSCI_A baud rate is 1/16 the UART source clock frequency BRCLK.

Modulation for BITCLK16 is based on the UCBRFx setting (see Table 10-3). A 1 in the table indicates that the corresponding BITCLK16 period is one BRCLK period longer than the periods m = 0. The modulation restarts with each new bit timing.

Modulation for BITCLK is based on the UCBRSx setting as previously described.

Table 10-3. BITCLK16 Modulation Pattern

UCBRFx	Number of BITCLK16 Clocks After Last Falling BITCLK Edge															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01h	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02h	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
03h	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1
04h	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1
05h	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1
06h	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1
07h	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1
08h	0	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1
09h	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1
0Ah	0	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1
0Bh	0	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1
0Ch	0	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1
0Dh	0	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
0Eh	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0Fh	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

10.3.10 Setting a Baud Rate

For a given BRCLK clock source, the baud rate used determines the required division factor N:

$N = f_{BRCLK}$ / baud rate

The division factor N is often a noninteger value, thus, at least one divider and one modulator stage is used to meet the factor as closely as possible.

If N is equal or greater than 16, it is recommended to use the oversampling baud-rate generation mode by setting UCOS16.

NOTE: Baud rate settings quick set up

To calculate the correct the correct settings for the baud-rate generation, perform these steps:

1. Calculate $N = f_{BRCLK}$ / baud rate [if N > 16 continue with step 3, otherwise with step 2]
2. OS16 = 0, UCBRx = INT(N) [continue with step 4]
3. OS16 = 1, UCBRx = INT(N/16), UCBRFx = INT([(N/16) – INT(N/16)] × 16)
4. UCBRSx can be found by looking up the fractional part of N (= N - INT(N)) in Table 10-4
5. If OS16 = 0 was chosen, a detailed error calculation is recommended to be performed

Table 10-4 can be used as a lookup table for finding the correct UCBRSx modulation pattern for the corresponding fractional part of N. The values there are optimized for transmitting.

Table 10-4. UCBRSx Settings for Fractional Portion of N = f_{BRCLK}/baud rate

Fractional Portion of N	UCBRSx(1)	Fractional Portion of N	UCBRSx(1)
0.0000	00h	0.5002	AAh
0.0529	01h	0.5715	6Bh
0.0715	02h	0.6003	ADh
0.0835	04h	0.6254	B5h
0.1001	08h	0.6432	B6h
0.1252	10h	0.6667	D6h
0.1430	20h	0.7001	B7h
0.1670	11h	0.7147	BBh
0.2147	21h	0.7503	DDh
0.2224	22h	0.7861	EDh
0.2503	44h	0.8004	EEh
0.3000	25h	0.8333	BFh
0.3335	49h	0.8464	DFh
0.3575	4Ah	0.8572	EFh
0.3753	52h	0.8751	F7h
0.4003	92h	0.9004	FBh
0.4286	53h	0.9170	FDh
0.4378	55h	0.9288	FEh

(1) The UCBRSx setting in one row is valid from the fractional portion given in that row until the one in the next row

10.3.10.1 Low-Frequency Baud-Rate Mode Setting

In low-frequency mode, the integer portion of the divisor is realized by the prescaler:

UCBRx = INT(N)

The fractional portion is realized by the modulator with its UCBRSx setting. The recommended way to determine the correct UCBRSx is to perform a detailed error calculation as explained in the following sections. However it is also possible to look up the correct settings in a table with typical crystals (see Table 10-5).

10.3.10.2 Oversampling Baud-Rate Mode Setting

In the oversampling mode, the prescaler is set to:

UCBRx = INT(N/16)

and the first stage modulator is set to:

UCBRFx = INT([(N/16) – INT(N/16)] × 16)

The second modulation stage setting (UCBRSx) can be found by performing a detailed error calculation or by using Table 10-4 and the fractional part of N = f_{BRCLK} / baud rate.

10.3.11 Transmit Bit Timing - Error calculation

The timing for each character is the sum of the individual bit timings. Using the modulation features of the baud-rate generator reduces the cumulative bit error. The individual bit error can be calculated using the following steps.

10.3.11.1 Low-Frequency Baud-Rate Mode Bit Timing

In low-frequency mode, calculation of the length of bit i $T_{bit,TX}[i]$ is based on the UCBRx and UCBRSx settings:

$T_{bit,TX}[i] = (1/f_{BRCLK})(UCBRx + m_{UCBRSx}[i])$

Where:

$m_{UCBRSx}[i]$ = Modulation of bit i of UCBRSx

10.3.11.2 Oversampling Baud-Rate Mode Bit Timing

In oversampling baud-rate mode, calculation of the length of bit i $T_{bit,TX}[i]$ is based on the baud-rate generator UCBRx, UCBRFx and UCBRSx settings:

$$T_{bit,TX}[i] = \frac{1}{f_{BRCLK}} \left((16 * UCBRx) + \sum_{j=0}^{15} m_{UCBRFx}[j] + m_{UCBRSx}[i] \right) \quad (1)$$

Where:

$\sum_{j=0}^{15} m_{UCBRFx}[j]$ = Sum of ones from the corresponding row in Table 10-3

$m_{UCBRSx}[i]$ = Modulation of bit i of UCBRSx

This results in an end-of-bit time $t_{bit,TX}[i]$ equal to the sum of all previous and the current bit times:

$$T_{bit,TX}[i] = \sum_{j=0}^{i} T_{bit,TX}[j] \quad (2)$$

To calculate bit error, this time is compared to the ideal bit time $t_{bit,ideal,TX}[i]$:

$t_{bit,ideal,TX}[i]$ = (1 / baud rate)(i + 1)

This results in an error normalized to one ideal bit time (1 / baud rate):

$Error_{TX}[i] = (t_{bit,TX}[i] – t_{bit,ideal,TX}[i]) \times$ baud rate × 100%

10.3.12 Receive Bit Timing – Error Calculation

Receive timing error consists of two error sources. The first is the bit-to-bit timing error similar to the transmit bit timing error. The second is the error between a start edge occurring and the start edge being accepted by the eUSCI_A module. Figure 10-11 shows the asynchronous timing errors between data on the UCAxRXD pin and the internal baud-rate clock. This results in an additional synchronization error. The synchronization error t_{SYNC} is between –0.5 BRCLKs and +0.5 RCLKs, independent of the selected baud-rate generation mode.

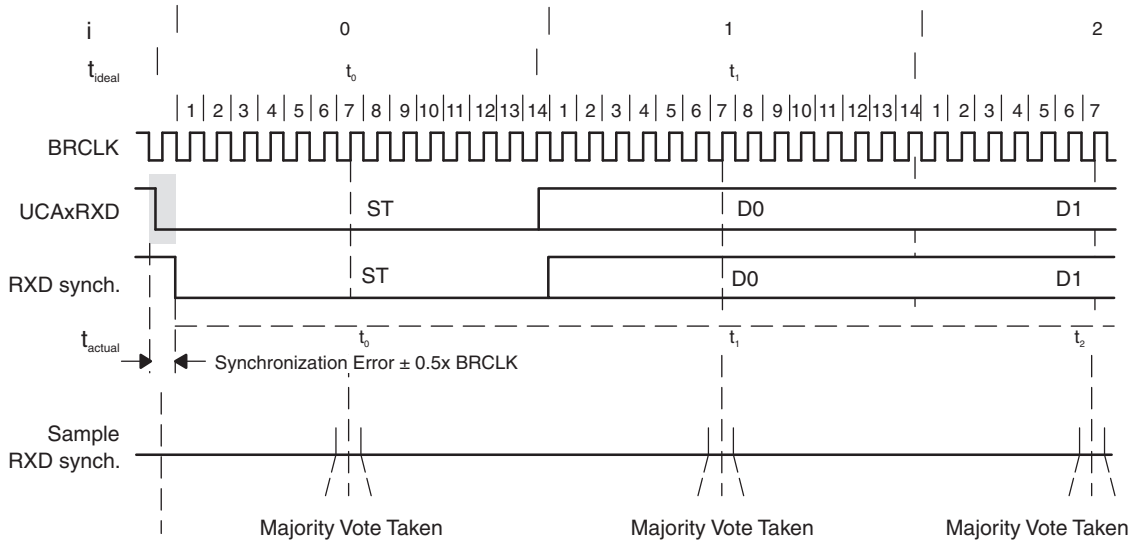


Figure 10-11. Receive Error

The ideal sampling time $t_{bit,ideal,RX}[i]$ is in the middle of a bit period:

$t_{bit,ideal,RX}[i] = (1 / \text{baud rate})(i + 0.5)$

The real sampling time, $t_{bit,RX}[i]$, is equal to the sum of all previous bits according to the formulas shown in the transmit timing section, plus one-half BITCLK for the current bit i, plus the synchronization error t_{SYNC}.

This results in the following $t_{bit,RX}[i]$ for the low-frequency baud-rate mode:

$$t_{bit,RX}[i] = t_{SYNC} + \sum_{j=0}^{i-1} T_{bit,RX}[j] + \frac{1}{f_{BRCLK}} \left(\text{INT}(½\text{UCBRx}) + m_{UCBRSx}[i] \right) \quad (3)$$

Where:

$T_{bit,RX}[i] = (1/f_{BRCLK})(\text{UCBRx} + m_{UCBRSx}[i])$

$m_{UCBRSx}[i]$ = Modulation of bit i of UCBRSx

For the oversampling baud-rate mode, the sampling time $t_{bit,RX}[i]$ of bit i is calculated by:

$$t_{bit,RX}[i] = t_{SYNC} + \sum_{j=0}^{i-1} T_{bit,RX}[j] + \frac{1}{f_{BRCLK}} \left((8 * \text{UCBRx}) + \sum_{j=0}^{7} m_{UCBRFx}[j] + m_{UCBRSx}[i] \right) \quad (4)$$

Where:

$$T_{bit,RX}[i] = \frac{1}{f_{BRCLK}} \left((16 * \text{UCBRx}) + \sum_{j=0}^{15} m_{UCBRFx}[j] + m_{UCBRSx}[i] \right)$$

$\sum_{j=0}^{7 + m_{UCBRSx}[i]} m_{UCBRFx}[j]$ = Sum of ones from columns 0 to $(7 + m_{UCBRSx}[i])$ from the corresponding row in Table 10-3.

$m_{UCBRSx}[i]$ = Modulation of bit i of UCBRSx

This results in an error normalized to one ideal bit time (1 / baud rate) according to the following formula:

$\text{Error}_{RX}[i] = (t_{bit,RX}[i] – t_{bit,ideal,RX}[i]) \times \text{baud rate} \times 100\%$

10.3.13 Typical Baud Rates and Errors

Standard baud-rate data for UCBRx, UCBRSx, and UCBRFx are listed in Table 10-5 for a 32768-Hz crystal sourcing ACLK and typical SMCLK frequencies. Make sure that the selected BRCLK frequency does not exceed the device specific maximum eUSCI_A input frequency (see the device-specific data sheet).

The receive error is the accumulated time versus the ideal scanning time in the middle of each bit. The worst-case error is given for the reception of an 8-bit character with parity and one stop bit including synchronization error.

The transmit error is the accumulated timing error versus the ideal time of the bit period. The worst-case error is given for the transmission of an 8-bit character with parity and stop bit.

Table 10-5. Recommended Settings for Typical Crystals and Baud Rates

BRCLK	Baud Rate	UCOS16	UCBRx	UCBRFx	UCBRSx	TX Error (%)		RX Error (%)	
						neg	pos	neg	pos
32768	1200	1	1	11	25h	-2.29	2.25	-2.56	5.35
32768	2400	0	13	-	B6h	-3.12	3.91	-5.52	8.84
32768	4800	0	6	-	EEh	-7.62	8.98	-21	10.25
32768	9600	0	3	-	92h	-17.19	16.02	-23.24	37.3
1000000	9600	1	6	8	20h	-0.48	0.64	-1.04	1.04
1000000	19200	1	3	4	2h	-0.8	0.96	-1.84	1.84
1000000	38400	1	1	10	0h	0	1.76	0	3.44
1000000	57600	0	17	-	4Ah	-2.72	2.56	-3.76	7.28
1000000	115200	0	8	-	D6h	-7.36	5.6	-17.04	6.96
1048576	9600	1	6	13	22h	-0.46	0.42	-0.48	1.23
1048576	19200	1	3	6	ADh	-0.88	0.83	-2.36	1.18
1048576	38400	1	1	11	25h	-2.29	2.25	-2.56	5.35
1048576	57600	0	18	-	11h	-2	3.37	-5.31	5.55
1048576	115200	0	9	-	08h	-5.37	4.49	-5.93	14.92
4000000	9600	1	26	0	B6h	-0.08	0.16	-0.28	0.2
4000000	19200	1	13	0	84h	-0.32	0.32	-0.64	0.48
4000000	38400	1	6	8	20h	-0.48	0.64	-1.04	1.04
4000000	57600	1	4	5	55h	-0.8	0.64	-1.12	1.76
4000000	115200	1	2	2	BBh	-1.44	1.28	-3.92	1.68
4000000	230400	0	17	-	4Ah	-2.72	2.56	-3.76	7.28
4194304	9600	1	27	4	FBh	-0.11	0.1	-0.33	0
4194304	19200	1	13	10	55h	-0.21	0.21	-0.55	0.33
4194304	38400	1	6	13	22h	-0.46	0.42	-0.48	1.23
4194304	57600	1	4	8	EEh	-0.75	0.74	-2	0.87
4194304	115200	1	2	4	92h	-1.62	1.37	-3.56	2.06
4194304	230400	0	18	-	11h	-2	3.37	-5.31	5.55
8000000	9600	1	52	1	49h	-0.08	0.04	-0.1	0.14
8000000	19200	1	26	0	B6h	-0.08	0.16	-0.28	0.2
8000000	38400	1	13	0	84h	-0.32	0.32	-0.64	0.48
8000000	57600	1	8	10	F7h	-0.32	0.32	-1	0.36
8000000	115200	1	4	5	55h	-0.8	0.64	-1.12	1.76
8000000	230400	1	2	2	BBh	-1.44	1.28	-3.92	1.68
8000000	460800	0	17	-	4Ah	-2.72	2.56	-3.76	7.28
8388608	9600	1	54	9	EEh	-0.06	0.06	-0.11	0.13
8388608	19200	1	27	4	FBh	-0.11	0.1	-0.33	0
8388608	38400	1	13	10	55h	-0.21	0.21	-0.55	0.33
8388608	57600	1	9	1	B5h	-0.31	0.31	-0.53	0.78
8388608	115200	1	4	8	EEh	-0.75	0.74	-2	0.87
8388608	230400	1	2	4	92h	-1.62	1.37	-3.56	2.06
8388608	460800	0	18	-	11h	-2	3.37	-5.31	5.55
12000000	9600	1	78	2	00h	0	0	0	0.04

Table 10-5. Recommended Settings for Typical Crystals and Baud Rates (continued)

BRCLK	Baud Rate	UCOS16	UCBRx	UCBRFx	UCBRSx	TX Error (%)		RX Error (%)	
						neg	pos	neg	pos
12000000	19200	1	39	1	00h	0	0	0	0.16
12000000	38400	1	19	8	65h	-0.16	0.16	-0.4	0.24
12000000	57600	1	13	0	25h	-0.16	0.32	-0.48	0.48
12000000	115200	1	6	8	20h	-0.48	0.64	-1.04	1.04
12000000	230400	1	3	4	02h	-0.8	0.96	-1.84	1.84
12000000	460800	1	1	10	00h	0	1.76	0	3.44
16000000	9600	1	104	2	D6h	-0.04	0.02	-0.09	0.03
16000000	19200	1	52	1	49h	-0.08	0.04	-0.1	0.14
16000000	38400	1	26	0	B6h	-0.08	0.16	-0.28	0.2
16000000	57600	1	17	5	DDh	-0.16	0.2	-0.3	0.38
16000000	115200	1	8	10	F7h	-0.32	0.32	-1	0.36
16000000	230400	1	4	5	55h	-0.8	0.64	-1.12	1.76
16000000	460800	1	2	2	BBh	-1.44	1.28	-3.92	1.68
16777216	9600	1	109	3	B5h	-0.03	0.02	-0.05	0.06
16777216	19200	1	54	9	EEh	-0.06	0.06	-0.11	0.13
16777216	38400	1	27	4	FBh	-0.11	0.1	-0.33	0
16777216	57600	1	18	3	44h	-0.16	0.15	-0.2	0.45
16777216	115200	1	9	1	B5h	-0.31	0.31	-0.53	0.78
16777216	230400	1	4	8	EEh	-0.75	0.74	-2	0.87
16777216	460800	1	2	4	92h	-1.62	1.37	-3.56	2.06
20000000	9600	1	130	3	25h	-0.02	0.03	0	0.07
20000000	19200	1	65	1	D6h	-0.06	0.03	-0.1	0.1
20000000	38400	1	32	8	EEh	-0.1	0.13	-0.27	0.14
20000000	57600	1	21	11	22h	-0.16	0.13	-0.16	0.38
20000000	115200	1	10	13	ADh	-0.29	0.26	-0.46	0.66
20000000	230400	1	5	6	EEh	-0.67	0.51	-1.71	0.62
20000000	460800	1	2	11	92h	-1.38	0.99	-1.84	2.8

10.3.14 Using the eUSCI_A Module in UART Mode With Low-Power Modes

The eUSCI_A module provides automatic clock activation for use with low-power modes. When the eUSCI_A clock source is inactive because the device is in a low-power mode, the eUSCI_A module automatically activates it when needed, regardless of the control-bit settings for the clock source. The clock remains active until the eUSCI_A module returns to its idle condition. After the eUSCI_A module returns to the idle condition, control of the clock source reverts to the settings of its control bits.

10.3.15 eUSCI_A Interrupts

The eUSCI_A has only one interrupt vector that is shared for transmission and for reception.

10.3.15.1 eUSCI_A Transmit Interrupt Operation

The UCTXIFG interrupt flag is set by the transmitter to indicate that UCAxTXBUF is ready to accept another character. An interrupt request is generated if UCTXIE and GIE are also set. UCTXIFG is automatically reset if a character is written to UCAxTXBUF.

UCTXIFG is set after a PUC or when UCSWRST = 1. UCTXIE is reset after a PUC or when UCSWRST = 1.

10.3.15.2 eUSCI_A Receive Interrupt Operation

The UCRXIFG interrupt flag is set each time a character is received and loaded into UCAxRXBUF. An interrupt request is generated if UCRXIE and GIE are also set. UCRXIFG and UCRXIE are reset by a system reset PUC signal or when UCSWRST = 1. UCRXIFG is automatically reset when UCAxRXBUF is read.

Additional interrupt control features include:

- When UCRXEIE = 0, erroneous characters do not set UCRXIFG.
- When UCDORM = 1, nonaddress characters do not set UCRXIFG in multiprocessor modes. In plain UART mode, no characters are set UCRXIFG.
- When UCBRKIE = 1, a break condition sets the UCBRK bit and the UCRXIFG flag.

10.3.15.3 eUSCI_A Receive Interrupt Operation

Table 10-6 describes the I^2C state change interrupt flags.

Table 10-6. UART State Change Interrupt Flags

Interrupt Flag	Interrupt Condition
UCSTTIFG	START byte received interrupt. This flag is set when the UART module receives a START byte.
UCTXCPTIFG	Transmit complete interrupt. This flag is set, after the complete UART byte in the internal shift register including STOP bit got shifted out and UCAxTXBUF is empty.

10.3.15.4 UCAxIV, Interrupt Vector Generator

The eUSCI_A interrupt flags are prioritized and combined to source a single interrupt vector. The interrupt vector register UCAxIV is used to determine which flag requested an interrupt. The highest-priority enabled interrupt generates a number in the UCAxIV register that can be evaluated or added to the program counter to automatically enter the appropriate software routine. Disabled interrupts do not affect the UCAxIV value.

Read access of the UCAxIV register automatically resets the highest-pending Interrupt condition and flag. Write access of the UCAxIV register clears all pending Interrupt conditions and flags. If another interrupt flag is set, another interrupt is generated immediately after servicing the initial interrupt.

Example 10-1 shows the recommended use of UCAxIV. The UCAxIV value is added to the PC to automatically jump to the appropriate routine. The following example is given for eUSCI_A0.

Example 10-1. UCAxIV Software Example

```
#pragma vector = USCI_A0_VECTOR __interrupt void USCI_A0_ISR(void) {
    switch(__even_in_range(UCA0IV,18)) {
        case 0x00:      // Vector 0: No interrupts
                 break;
        case 0x02: ...  // Vector 2: UCSTTIFG
                 break;
        case 0x04: ...  // Vector 4: UCTXCPTIFG
                 break;
        case 0x06: ...  // Vector 6: UCTXIFG
                 break;
        case 0x08: ...  // Vector 8: UCRXIFG
                 break;
        default: break;
    }
}
```

10.4 eUSCI_A UART Registers

The eUSCI_A registers applicable in UART mode and their addresses are listed in Table 10-7.

Table 10-7. eUSCI_A UART Registers

Address	Acronym	Register Name	Type	Access	Reset	Section
0140h	UCAxCTLW0	eUSCI_Ax Control Word 0	Read/write	Word	0001h	Section 10.4.1
0141h	UCAxCTL0(1)	eUSCI_Ax Control 0	Read/write	Byte	00h	
0140h	UCAxCTL1	eUSCI_Ax Control 1	Read/write	Byte	01h	
0142h	UCAxCTLW1	eUSCI_Ax Control Word 1	Read/write	Word	0003h	Section 10.4.2
0146h	UCAxBRW	eUSCI_Ax Baud Rate Control Word	Read/write	Word	0000h	Section 10.4.3
0146h	UCAxBR0(1)	eUSCI_Ax Baud Rate Control 0	Read/write	Byte	00h	
0147h	UCAxBR1	eUSCI_Ax Baud Rate Control 1	Read/write	Byte	00h	
0148h	UCAxMCTLW	eUSCI_Ax Modulation Control Word	Read/write	Word	00h	Section 10.4.4
014Ah	UCAxSTATW	eUSCI_Ax Status	Read/write	Word	00h	Section 10.4.5
014Ch	UCAxRXBUF	eUSCI_Ax Receive Buffer	Read/write	Word	00h	Section 10.4.6
014Eh	UCAxTXBUF	eUSCI_Ax Transmit Buffer	Read/write	Word	00h	Section 10.4.7
0150h	UCAxABCTL	eUSCI_Ax Auto Baud Rate Control	Read/write	Word	00h	Section 10.4.8
0152h	UCAxIRCTL	eUSCI_Ax IrDA Control	Read/write	Word	0000h	Section 10.4.9
0152h	UCAxIRTCTL	eUSCI_Ax IrDA Transmit Control	Read/write	Byte	00h	
0153h	UCAxIRRCTL	eUSCI_Ax IrDA Receive Control	Read/write	Byte	00h	
015Ah	UCAxIE	eUSCI_Ax Interrupt Enable	Read/write	Word	00h	Section 10.4.10
015Ch	UCAxIFG	eUSCI_Ax Interrupt Flag	Read/write	Word	00h	Section 10.4.11
015Eh	UCAxIV	eUSCI_Ax Interrupt Vector	Read	Word	0000h	Section 10.4.12

(1) It is recommended to access these registers using 16-bit access. If 8-bit access is used, the corresponding bit names must be followed by "_H".

10.4.1 UCAxCTLW0 Register (address = 0140h) [reset = 0001h]

eUSCI_Ax Control Word Register 0

Figure 10-12. UCAxCTLW0 Register

15	14	13	12	11	10	9	8
UCPEN	UCPAR	UCMSB	UC7BIT	UCSPB	UCMODEx		UCSYNC
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
UCSSELx		UCRXEIE	UCBRKIE	UCDORM	UCTXADDR	UCTXBRK	UCSWRST
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

Table 10-8. UCAxCTLW0 Register Description

Bit	Field	Type	Reset	Description
15	UCPEN	RW	0h	Parity enable 0b = Parity disabled 1b = Parity enabled. Parity bit is generated (UCAxTXD) and expected (UCAxRXD). In address-bit multiprocessor mode, the address bit is included in the parity calculation.
14	UCPAR	RW	0h	Parity select. UCPAR is not used when parity is disabled. 0b = Odd parity 1b = Even parity
13	UCMSB	RW	0h	MSB first select. Controls the direction of the receive and transmit shift register. 0b = LSB first 1b = MSB first
12	UC7BIT	RW	0h	Character length. Selects 7-bit or 8-bit character length. 0b = 8-bit data 1b = 7-bit data
11	UCSPB	RW	0h	Stop bit select. Number of stop bits. 0b = One stop bit 1b = Two stop bits
10-9	UCMODEx	RW	0h	eUSCI_A mode. The UCMODEx bits select the asynchronous mode when UCSYNC = 0. 00b = UART mode 01b = Idle-line multiprocessor mode 10b = Address-bit multiprocessor mode 11b = UART mode with automatic baud-rate detection
8	UCSYNC	RW	0h	Synchronous mode enable 0b = Asynchronous mode 1b = Synchronous mode
7-6	UCSSELx	RW	0h	eUSCI_A clock source select. These bits select the BRCLK source clock. 00b = UCLK (external eUSCI clock) 01b = ACLK 10b = SMCLK 11b = SMCLK
5	UCRXEIE	RW	0h	Receive erroneous-character interrupt enable 0b = Erroneous characters rejected and UCRXIFG is not set. 1b = Erroneous characters received set UCRXIFG.
4	UCBRKIE	RW	0h	Receive break character interrupt enable 0b = Received break characters do not set UCRXIFG. 1b = Received break characters set UCRXIFG.

Table 10-8. UCAxCTLW0 Register Description (continued)

Bit	Field	Type	Reset	Description
3	UCDORM	RW	0h	Dormant. Puts eUSCI_A into sleep mode. 0b = Not dormant. All received characters set UCRXIFG. 1b = Dormant. Only characters that are preceded by an idle-line or with address bit set UCRXIFG. In UART mode with automatic baud-rate detection, only the combination of a break and synch field sets UCRXIFG.
2	UCTXADDR	RW	0h	Transmit address. Next frame to be transmitted is marked as address, depending on the selected multiprocessor mode. 0b = Next frame transmitted is data. 1b = Next frame transmitted is an address.
1	UCTXBRK	RW	0h	Transmit break. Transmits a break with the next write to the transmit buffer. In UART mode with automatic baud-rate detection, 055h must be written into UCAxTXBUF to generate the required break/synch fields. Otherwise, 0h must be written into the transmit buffer. 0b = Next frame transmitted is not a break. 1b = Next frame transmitted is a break or a break/synch.
0	UCSWRST	RW	1h	Software reset enable 0b = Disabled. eUSCI_A reset released for operation. 1b = Enabled. eUSCI_A logic held in reset state.

10.4.2 UCAxCTLW1 Register (address = 0142h) [reset = 0003h]

eUSCI_Ax Control Word Register 1

Figure 10-13. UCAxCTLW1 Register

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved						UCGLITx	
r-0	r-0	r-0	r-0	r-0	r-0	rw-1	rw-1

Table 10-9. UCAxCTLW1 Register Description

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1-0	UCGLITx	RW	3h	Deglitch time 00b = approximately 2 ns 01b = approximately 50 ns 10b = approximately 100 ns 11b = approximately 200 ns

10.4.3 UCAxBRW Register (address = 0146h) [reset = 0000h]

eUSCI_Ax Baud Rate Control Word Register

Figure 10-14. UCAxBRW Register

15	14	13	12	11	10	9	8
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw

Table 10-10. UCAxBRW Register Description

Bit	Field	Type	Reset	Description
15-0	UCBRx	RW	0h	Clock prescaler setting of the baud rate generator

10.4.4 UCAxMCTLW Register (address = 0148h) [reset = 0000h]

eUSCI_Ax Modulation Control Word Register

Figure 10-15. UCAxMCTLW Register

15	14	13	12	11	10	9	8
UCBRSx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
UCBRFx				Reserved			UCOS16
rw-0	rw-0	rw-0	rw-0	r0	r0	r0	rw-0

Table 10-11. UCAxMCTLW Register Description

Bit	Field	Type	Reset	Description
15-8	UCBRSx	RW	0h	Second modulation stage select. These bits hold a free modulation pattern for BITCLK.
7-4	UCBRFx	RW	0h	First modulation stage select. These bits determine the modulation pattern for BITCLK16 when UCOS16 = 1. Ignored with UCOS16 = 0. The "Oversampling Baud-Rate Generation" section shows the modulation pattern.
3-1	Reserved	R	0h	Reserved
0	UCOS16	RW	0h	Oversampling mode enabled 0b = Disabled 1b = Enabled

10.4.5 UCAxSTATW Register (address = 014Ah) [reset = 0000h]

eUSCI_Ax Status Register

Figure 10-16. UCAxSTATW Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCLISTEN	UCFE	UCOE	UCPE	UCBRK	UCRXERR	UCADDR UCIDLE	UCBUSY
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r-0

Table 10-12. UCAxSTATW Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7	UCLISTEN	RW	0h	Listen enable. The UCLISTEN bit selects loopback mode. 0b = Disabled 1b = Enabled. UCAxTXD is internally fed back to the receiver.
6	UCFE	RW	0h	Framing error flag 0b = No error 1b = Character received with low stop bit
5	UCOE	RW	0h	Overrun error flag. This bit is set when a character is transferred into UCAxRXBUF before the previous character was read. UCOE is cleared automatically when UCxRXBUF is read, and must not be cleared by software. Otherwise, it does not function correctly. 0b = No error 1b = Overrun error occurred.
4	UCPE	RW	0h	Parity error flag. When UCPEN = 0, UCPE is read as 0. 0b = No error 1b = Character received with parity error
3	UCBRK	RW	0h	Break detect flag 0b = No break condition 1b = Break condition occurred.
2	UCRXERR	RW	0h	Receive error flag. This bit indicates a character was received with one or more errors. When UCRXERR = 1, on or more error flags, UCFE, UCPE, or UCOE is also set. UCRXERR is cleared when UCAxRXBUF is read. 0b = No receive errors detected 1b = Receive error detected
1	UCADDR UCIDLE	RW	0h	UCADDR: Address received in address-bit multiprocessor mode. UCADDR is cleared when UCAxRXBUF is read. UCIDLE: Idle line detected in idle-line multiprocessor mode. UCIDLE is cleared when UCAxRXBUF is read. 0b = UCADDR: Received character is data. UCIDLE: No idle line detected 1b = UCADDR: Received character is an address. UCIDLE: Idle line detected
0	UCBUSY	R	0h	eUSCI_A busy. This bit indicates if a transmit or receive operation is in progress. 0b = eUSCI_A inactive 1b = eUSCI_A transmitting or receiving

10.4.6 UCAxRXBUF Register (address = 014Ch) [reset = 0000h]

eUSCI_Ax Receive Buffer Register

Figure 10-17. UCAxRXBUF Register

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
UCRXBUFx							
r	r	r	r	r	r	r	r

Table 10-13. UCAxRXBUF Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCRXBUFx	R	0h	The receive-data buffer is user accessible and contains the last received character from the receive shift register. Reading UCAxRXBUF resets the receive-error bits, the UCADDR or UCIDLE bit, and UCRXIFG. In 7-bit data mode, UCAxRXBUF is LSB justified and the MSB is always reset.

10.4.7 UCAxTXBUF Register (address = 014Eh) [reset = 0000h]

eUSCI_Ax Transmit Buffer Register

Figure 10-18. UCAxTXBUF Register

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
UCTXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

Table 10-14. UCAxTXBUF Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCTXBUFx	RW	0h	The transmit data buffer is user accessible and holds the data waiting to be moved into the transmit shift register and transmitted on UCAxTXD. Writing to the transmit data buffer clears UCTXIFG. The MSB of UCAxTXBUF is not used for 7-bit data and is reset.

10.4.8 UCAxABCTL Register (address = 0150h) [reset = 0000h]

eUSCI_Ax Auto Baud Rate Control Register

Figure 10-19. UCAxABCTL Register

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved		UCDELIMx		UCSTOE	UCBTOE	Reserved	UCABDEN
r-0	r-0	rw-0	rw-0	rw-0	rw-0	r-0	rw-0

Table 10-15. UCAxABCTL Register Description

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	Reserved
5-4	UCDELIMx	RW	0h	Break/synch delimiter length 00b = 1 bit time 01b = 2 bit times 10b = 3 bit times 11b = 4 bit times
3	UCSTOE	RW	0h	Synch field time out error 0b = No error 1b = Length of synch field exceeded measurable time.
2	UCBTOE	RW	0h	Break time out error 0b = No error 1b = Length of break field exceeded 22 bit times.
1	Reserved	R	0h	Reserved
0	UCABDEN	RW	0h	Automatic baud-rate detect enable 0b = Baud-rate detection disabled. Length of break and synch field is not measured. 1b = Baud-rate detection enabled. Length of break and synch field is measured and baud-rate settings are changed accordingly.

10.4.9 UCAxIRCTL Register (address = 0152h) [reset = 0000h]

eUSCI_Ax IrDA Control Word Register

Figure 10-20. UCAxIRCTL Register

15	14	13	12	11	10	9	8
UCIRRXFLx						UCIRRXPL	UCIRRXFE
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
UCIRTXPLx						UCIRTXCLK	UCIREN
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 10-16. UCAxIRCTL Register Description

Bit	Field	Type	Reset	Description
15-10	UCIRRXFLx	RW	0h	Receive filter length. The minimum pulse length for receive is given by: t_{MIN} = (UCIRRXFLx + 4) / (2 × $f_{IRTXCLK}$)
9	UCIRRXPL	RW	0h	IrDA receive input UCAxRXD polarity 0b = IrDA transceiver delivers a high pulse when a light pulse is seen. 1b = IrDA transceiver delivers a low pulse when a light pulse is seen.
8	UCIRRXFE	RW	0h	IrDA receive filter enabled 0b = Receive filter disabled 1b = Receive filter enabled
7-2	UCIRTXPLx	RW	0h	Transmit pulse length. Pulse length t_{PULSE} = (UCIRTXPLx + 1) / (2 × $f_{IRTXCLK}$)
1	UCIRTXCLK	RW	0h	IrDA transmit pulse clock select 0b = BRCLK 1b = BITCLK16 when UCOS16 = 1. Otherwise, BRCLK.
0	UCIREN	RW	0h	IrDA encoder/decoder enable 0b = IrDA encoder/decoder disabled 1b = IrDA encoder/decoder enabled

10.4.10 UCAxIE Register (address = 015Ah) [reset = 0000h]

eUSCI_Ax Interrupt Enable Register

Figure 10-21. UCAxIE Register

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved				UCTXCPTIE	UCSTTIE	UCTXIE	UCRXIE
r-0	r-0	r-0	r-0	rw-0	rw-0	rw-0	rw-0

Table 10-17. UCAxIE Register Description

Bit	Field	Type	Reset	Description
15-4	Reserved	R	0h	Reserved
3	UCTXCPTIE	RW	0h	Transmit complete interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
2	UCSTTIE	RW	0h	Start bit interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
1	UCTXIE	RW	0h	Transmit interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
0	UCRXIE	RW	0h	Receive interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled

10.4.11 UCAxIFG Register (address = 015Ch) [reset = 0000h]

eUSCI_Ax Interrupt Flag Register

Figure 10-22. UCAxIFG Register

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved				UCTXCPTIFG	UCSTTIFG	UCTXIFG	UCRXIFG
r-0	r-0	r-0	r-0	rw-0	rw-0	rw-0	rw-0

Table 10-18. UCAxIFG Register Description

Bit	Field	Type	Reset	Description
15-4	Reserved	R	0h	Reserved
3	UCTXCPTIFG	RW	0h	Transmit ready interrupt enable. UCTXRDYIFG is set when the entire byte in the internal shift register got shifted out and UCAxTXBUF is empty. 0b = No interrupt pending 1b = Interrupt pending
2	UCSTTIFG	RW	0h	Start bit interrupt flag. UCSTTIFG is set after a Start bit was received 0b = No interrupt pending 1b = Interrupt pending
1	UCTXIFG	RW	0h	Transmit interrupt flag. UCTXIFG is set when UCAxTXBUF empty. 0b = No interrupt pending 1b = Interrupt pending
0	UCRXIFG	RW	0h	Receive interrupt flag. UCRXIFG is set when UCAxRXBUF has received a complete character. 0b = No interrupt pending 1b = Interrupt pending

10.4.12 UCAxIV Register (address = 015Eh) [reset = 0000h]

eUSCI_Ax Interrupt Vector Register

Figure 10-23. UCAxIV Register

15	14	13	12	11	10	9	8
UCIVx							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCIVx							
r0	r0	r0	r0	r-(0)	r-(0)	r-(0)	r0

Table 10-19. UCAxIV Register Description

Bit	Field	Type	Reset	Description
15-0	UCIVx	R	0h	eUSCI_A interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: Receive buffer full; Interrupt Flag: UCRXIFG; Interrupt Priority: Highest 04h = Interrupt Source: Transmit buffer empty; Interrupt Flag: UCTXIFG 06h = Interrupt Source: Start bit received; Interrupt Flag: UCSTTIFG 08h = Interrupt Source: Transmit complete; Interrupt Flag: UCTXCPTIFG; Interrupt Priority: Lowest

Enhanced Universal Serial Communication Interface (eUSCI) – SPI Mode

The enhanced universal serial communication interfaces, eUSCI_A and eUSCI_B, support multiple serial communication modes with one hardware module. This chapter discusses the operation of the synchronous peripheral interface (SPI) mode.

Topic **Page**

11.1 Enhanced Universal Serial Communication Interfaces (eUSCI_A, eUSCI_B) Overview

Both the eUSCI_A and the eUSCI_B support serial communication in SPI mode.

11.2 eUSCI Introduction – SPI Mode

In synchronous mode, the eUSCI connects the device to an external system via three or four pins: UCxSIMO, UCxSOMI, UCxCLK, and UCxSTE. SPI mode is selected when the UCSYNC bit is set, and SPI mode (3-pin or 4-pin) is selected with the UCMODEx bits.

SPI mode features include:

- 7-bit or 8-bit data length
- LSB first or MSB first data transmit and receive
- 3-pin and 4-pin SPI operation
- Master or slave modes
- Independent transmit and receive shift registers
- Separate transmit and receive buffer registers
- Continuous transmit and receive operation
- Selectable clock polarity and phase control
- Programmable clock frequency in master mode
- Independent interrupt capability for receive and transmit
- Slave operation in LPM4

Figure 11-1 shows the eUSCI block diagram when it is configured for SPI mode.

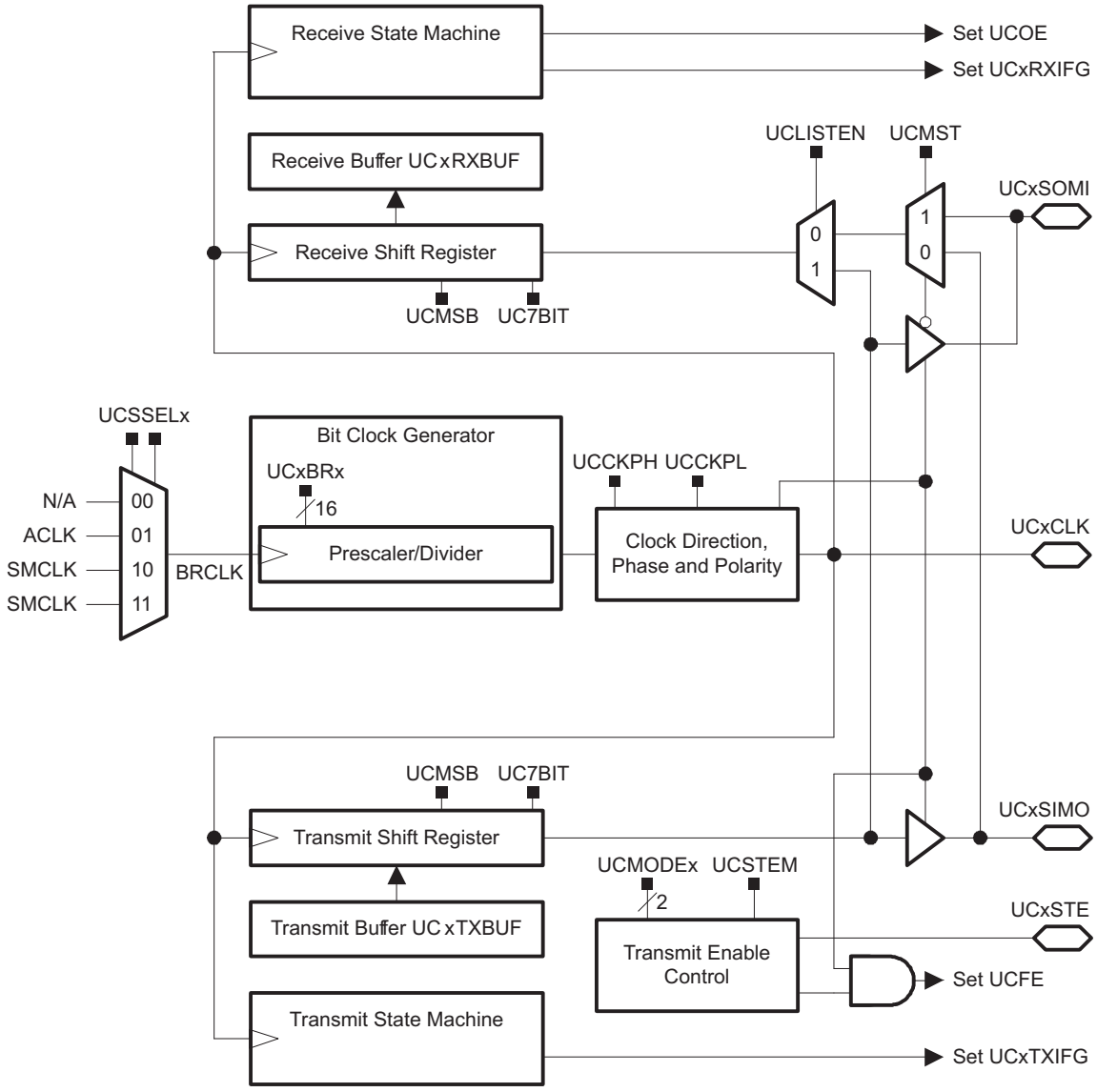


Figure 11-1. eUSCI Block Diagram – SPI Mode

11.3 eUSCI Operation – SPI Mode

In SPI mode, serial data is transmitted and received by multiple devices using a shared clock provided by the master. An additional pin controlled by the master, UCxSTE, is provided to enable a device to receive and transmit data.

Three or four signals are used for SPI data exchange:

- UCxSIMO – slave in, master out

  Master mode: UCxSIMO is the data output line.
  Slave mode: UCxSIMO is the data input line.
- UCxSOMI – slave out, master in

  Master mode: UCxSOMI is the data input line.
  Slave mode: UCxSOMI is the data output line.
- UCxCLK – eUSCI SPI clock

  Master mode: UCxCLK is an output.
  Slave mode: UCxCLK is an input.
- UCxSTE – slave transmit enable.

  Used in 4-pin mode to allow multiple masters on a single bus. Not used in 3-pin mode. Table 11-1 describes the UCxSTE operation.

Table 11-1. UCxSTE Operation

UCMODEx	UCxSTE Active State	UCxSTE	Slave	Master
01	High	0	Inactive	Active
		1	Active	Inactive
10	Low	0	Active	Inactive
		1	Inactive	Active

11.3.1 eUSCI Initialization and Reset

The eUSCI is reset by a PUC or by the UCSWRST bit. After a PUC, the UCSWRST bit is automatically set, keeping the eUSCI in a reset condition. When set, the UCSWRST bit resets the UCRXIE, UCTXIE, UCRXIFG, UCOE, and UCFE bits, and sets the UCTXIFG flag. Clearing UCSWRST releases the eUSCI for operation.

NOTE: Initializing or reconfiguring the eUSCI module

The recommended eUSCI initialization/reconfiguration process is:

1. Set UCSWRST.
   ```
   BIS.B #UCSWRST,&UCxCTL1
   ```
2. Initialize all eUSCI registers with UCSWRST = 1 (including UCxCTL1).
3. Configure ports.
4. Clear UCSWRST via software.
   ```
   BIC.B #UCSWRST,&UCxCTL1
   ```
5. Enable interrupts (optional) by setting UCRXIE and UCTXIE.

11.3.2 Character Format

The eUSCI module in SPI mode supports 7-bit and 8-bit character lengths selected by the UC7BIT bit. In 7-bit data mode, UCxRXBUF is LSB justified and the MSB is always reset. The UCMSB bit controls the direction of the transfer and selects LSB or MSB first.

NOTE: Default character format

The default SPI character transmission is LSB first. For communication with other SPI interfaces, MSB-first mode may be required.

NOTE: Character format for figures

Figures throughout this chapter use MSB-first format.

11.3.3 Master Mode

Figure 11-2 shows the eUSCI as a master in both 3-pin and 4-pin configurations. The eUSCI initiates data transfer when data is moved to the transmit data buffer UCxTXBUF. The UCxTXBUF data is moved to the transmit (TX) shift register when the TX shift register is empty, initiating data transfer on UCxSIMO starting with either the MSB or LSB, depending on the UCMSB setting. Data on UCxSOMI is shifted into the receive shift register on the opposite clock edge. When the character is received, the receive data is moved from the receive (RX) shift register to the received data buffer UCxRXBUF and the receive interrupt flag UCRXIFG is set, indicating the RX/TX operation is complete.

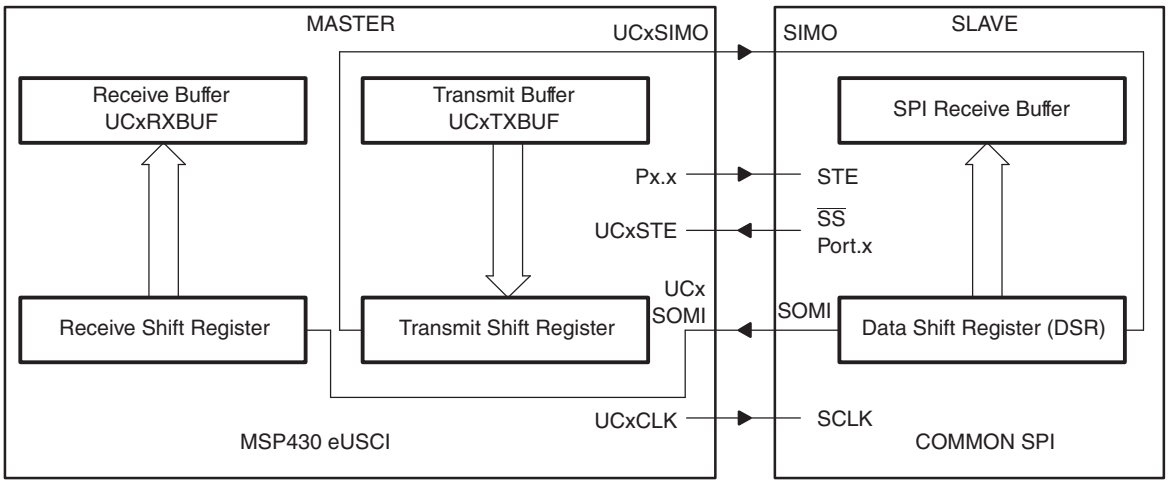


Figure 11-2. eUSCI Master and External Slave

A set transmit interrupt flag, UCTXIFG, indicates that data has moved from UCxTXBUF to the TX shift register and UCxTXBUF is ready for new data. It does not indicate RX/TX completion.

To receive data into the eUSCI in master mode, data must be written to UCxTXBUF, because receive and transmit operations operate concurrently.

There two different options for configuring the eUSCI as a 4-pin master, and the UCSTEM bit is used to select the corresponding mode.

- The fourth pin is used to prevent conflicts with other masters (UCSTEM = 0) (see Section 11.3.3.1)
- The fourth pin is used to generate a slave enable signal (UCSTEM = 1) (see Section 11.3.3.2)

11.3.3.1 4-Pin SPI Master Mode (UCSTEM = 0)

In 4-pin master mode with UCSTEM = 0, UCxSTE can be used to prevent conflicts with another master and controls the master as described in Table 11-1. When UCxSTE is in the master-inactive state and UCSTEM = 0:

- UCxSIMO and UCxCLK are set to inputs and no longer drive the bus.
- The error bit UCFE is set, indicating a communication integrity violation to be handled by the user.
- The internal state machines are reset and the shift operation is aborted.

If data is written into UCxTXBUF while the master is held inactive by UCxSTE, it is transmit as soon as UCxSTE transitions to the master-active state. If an active transfer is aborted by UCxSTE transitioning to the master-inactive state, the data must be rewritten into UCxTXBUF to be transferred when UCxSTE transitions back to the master-active state. The UCxSTE input signal is not used in 3-pin master mode.

11.3.3.2 4-Pin SPI Master Mode (UCSTEM = 1)

If UCSTEM = 1 in 4-pin master mode, the slave enable signal for a single slave is automatically generated. The corresponding behavior can be seen in Figure 11-4.

If multiple slaves are desired, this feature is not applicable and the software needs to use general-purpose I/O pins instead.

11.3.4 Slave Mode

Figure 11-3 shows the eUSCI as a slave in both 3-pin and 4-pin configurations. UCxCLK is used as the input for the SPI clock and must be supplied by the external master. The data-transfer rate is determined by this clock and not by the internal bit clock generator. Data written to UCxTXBUF and moved to the TX shift register before the start of UCxCLK is transmitted on UCxSOMI. Data on UCxSIMO is shifted into the receive shift register on the opposite edge of UCxCLK and moved to UCxRXBUF when the set number of bits are received. When data is moved from the RX shift register to UCxRXBUF, the UCRXIFG interrupt flag is set, indicating that data has been received. The overrun error bit UCOE is set when the previously received data is not read from UCxRXBUF before new data is moved to UCxRXBUF.

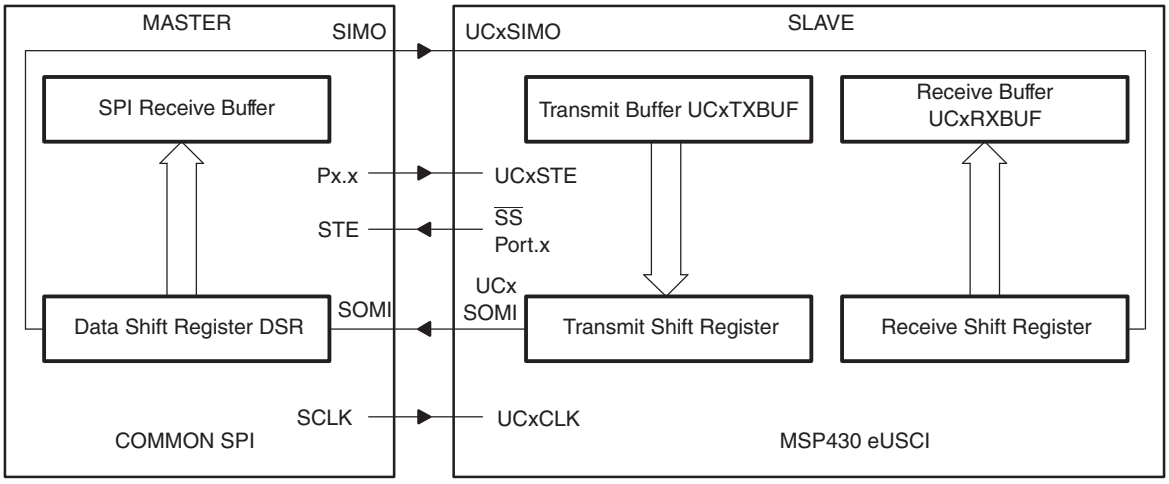


Figure 11-3. eUSCI Slave and External Master

11.3.4.1 4-Pin SPI Slave Mode

In 4-pin slave mode, UCxSTE is used by the slave to enable the transmit and receive operations and is provided by the SPI master. When UCxSTE is in the slave-active state, the slave operates normally. When UCxSTE is in the slave- inactive state:

- Any receive operation in progress on UCxSIMO is halted.
- UCxSOMI is set to the input direction.
- The shift operation is halted until the UCxSTE line transitions into the slave transmit active state.

The UCxSTE input signal is not used in 3-pin slave mode.

11.3.5 SPI Enable

When the eUSCI module is enabled by clearing the UCSWRST bit, it is ready to receive and transmit. In master mode, the bit clock generator is ready, but is not clocked nor producing any clocks. In slave mode, the bit clock generator is disabled and the clock is provided by the master.

A transmit or receive operation is indicated by UCBUSY = 1.

A PUC or set UCSWRST bit disables the eUSCI immediately and any active transfer is terminated.

11.3.5.1 Transmit Enable

In master mode, writing to UCxTXBUF activates the bit clock generator, and the data begins to transmit.

In slave mode, transmission begins when a master provides a clock and, in 4-pin mode, when the UCxSTE is in the slave-active state.

11.3.5.2 Receive Enable

The SPI receives data when a transmission is active. Receive and transmit operations operate concurrently.

11.3.6 Serial Clock Control

UCxCLK is provided by the master on the SPI bus. When UCMST = 1, the bit clock is provided by the eUSCI bit clock generator on the UCxCLK pin. The clock used to generate the bit clock is selected with the UCSSELx bits. When UCMST = 0, the eUSCI clock is provided on the UCxCLK pin by the master, the bit clock generator is not used, and the UCSSELx bits are don't care. The SPI receiver and transmitter operate in parallel and use the same clock source for data transfer.

The 16-bit value of UCBRx in the bit rate control registers (UCxxBR1 and UCxxBR0) is the division factor of the eUSCI clock source, BRCLK. The maximum bit clock that can be generated in master mode is BRCLK. Modulation is not used in SPI mode, and UCAxMCTL should be cleared when using SPI mode for eUSCI_A. The UCAxCLK or UCBxCLK frequency is given by:

$f_{BitClock} = f_{BRCLK} / UCBRx$

If UCBRx = 0, then $f_{BitClock} = f_{BRCLK}$.

11.3.6.1 Serial Clock Polarity and Phase

The polarity and phase of UCxCLK are independently configured via the UCCKPL and UCCKPH control bits of the eUSCI. Timing for each case is shown in Figure 11-4.

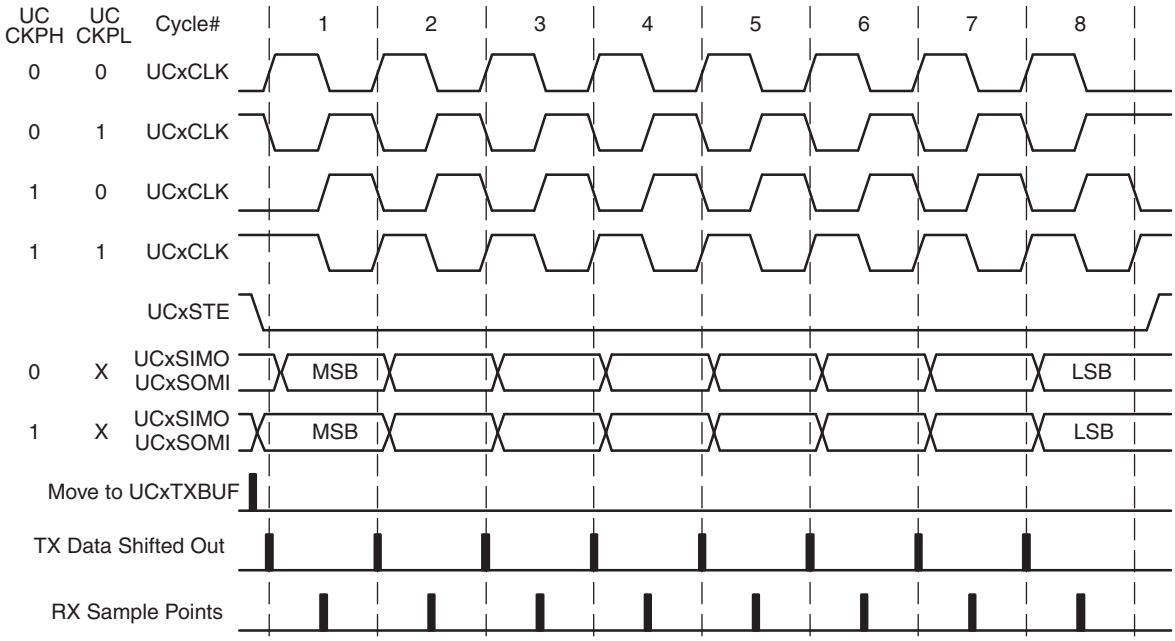


Figure 11-4. eUSCI SPI Timing With UCMSB = 1

11.3.7 Using the SPI Mode With Low-Power Modes

The eUSCI module provides automatic clock activation for use with low-power modes. When the eUSCI clock source is inactive because the device is in a low-power mode, the eUSCI module automatically activates it when needed, regardless of the control-bit settings for the clock source. The clock remains active until the eUSCI module returns to its idle condition. After the eUSCI module returns to the idle condition, control of the clock source reverts to the settings of its control bits.

In SPI slave mode, no internal clock source is required because the clock is provided by the external master. It is possible to operate the eUSCI in SPI slave mode while the device is in LPM4 and all clock sources are disabled. The receive or transmit interrupt can wake up the CPU from any low-power mode.

11.3.8 SPI Interrupts

The eUSCI has only one interrupt vector that is shared for transmission and for reception. eUSCI_Ax and eUSCI_Bx do not share the same interrupt vector.

11.3.8.1 SPI Transmit Interrupt Operation

The UCTXIFG interrupt flag is set by the transmitter to indicate that UCxTXBUF is ready to accept another character. An interrupt request is generated if UCTXIE and GIE are also set. UCTXIFG is automatically reset if a character is written to UCxTXBUF. UCTXIFG is set after a PUC or when UCSWRST = 1. UCTXIE is reset after a PUC or when UCSWRST = 1.

NOTE: Writing to UCxTXBUF in SPI mode

Data written to UCxTXBUF when UCTXIFG = 0 may result in erroneous data transmission.

11.3.8.2 SPI Receive Interrupt Operation

The UCRXIFG interrupt flag is set each time a character is received and loaded into UCxRXBUF. An interrupt request is generated if UCRXIE and GIE are also set. UCRXIFG and UCRXIE are reset by a system reset PUC signal or when UCSWRST = 1. UCRXIFG is automatically reset when UCxRXBUF is read.

11.3.8.3 UCxIV, Interrupt Vector Generator

The eUSCI interrupt flags are prioritized and combined to source a single interrupt vector. The interrupt vector register UCxIV is used to determine which flag requested an interrupt. The highest-priority enabled interrupt generates a number in the UCxIV register that can be evaluated or added to the program counter (PC) to automatically enter the appropriate software routine. Disabled interrupts do not affect the UCxIV value.

Any access, read or write, of the UCxIV register automatically resets the highest-pending interrupt flag. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt.

Example 11-1. UCxIV Software Example

The following software example shows the recommended use of UCxIV. The UCxIV value is added to the PC to automatically jump to the appropriate routine. The following example is given for eUSCI_B0.

```
USCI_SPI_ISR
         ADD      &UCB0IV, PC  ; Add offset to jump table
         RETI                  ; Vector 0: No interrupt
         JMP      RXIFG_ISR    ; Vector 2: RXIFG
TXIFG_ISR                      ; Vector 4: TXIFG
         ...                   ; Task starts here
         RETI                  ; Return
RXIFG_ISR                      ; Vector 2
         ...                   ; Task starts here
         RETI                  ; Return
```

11.4 eUSCI_A SPI Registers

The eUSCI_A registers applicable in SPI mode and their addresses are listed in Table 11-2.

Table 11-2. eUSCI_A SPI Registers

Address	Acronym	Register Name	Type	Access	Reset	Section
0140h	UCAxCTLW0	eUSCI_Ax Control Word 0	Read/write	Word	0001h	Section 11.4.1
0140h	UCAxCTL1	eUSCI_Ax Control 1	Read/write	Byte	01h	
0141h	UCAxCTL0	eUSCI_Ax Control 0	Read/write	Byte	00h	
0146h	UCAxBRW	eUSCI_Ax Bit Rate Control Word	Read/write	Word	0000h	Section 11.4.2
0146h	UCAxBR0	eUSCI_Ax Bit Rate Control 0	Read/write	Byte	00h	
0147h	UCAxBR1	eUSCI_Ax Bit Rate Control 1	Read/write	Byte	00h	
014Ah	UCAxSTATW	eUSCI_Ax Status	Read/write	Word	00h	Section 11.4.3
014Ch	UCAxRXBUF	eUSCI_Ax Receive Buffer	Read/write	Word	00h	Section 11.4.4
014Eh	UCAxTXBUF	eUSCI_Ax Transmit Buffer	Read/write	Word	00h	Section 11.4.5
015Ah	UCAxIE	eUSCI_Ax Interrupt Enable	Read/write	Word	00h	Section 11.4.6
015Ch	UCAxIFG	eUSCI_Ax Interrupt Flag	Read/write	Word	02h	Section 11.4.7
015Eh	UCAxIV	eUSCI_Ax Interrupt Vector	Read	Word	0000h	Section 11.4.8

11.4.1 UCAxCTLW0 Register (address = 0140h) [reset = 0001h]

eUSCI_Ax Control Register 0

Figure 11-5. UCAxCTLW0 Register

15	14	13	12	11	10	9	8
UCCKPH	UCCKPL	UCMSB	UC7BIT	UCMST	UCMODEx		UCSYNC
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
UCSSELx		Reserved				UCSTEM	UCSWRST
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

Table 11-3. UCAxCTLW0 Register Description

Bit	Field	Type	Reset	Description
15	UCCKPH	RW	0h	Clock phase select 0b = Data is changed on the first UCLK edge and captured on the following edge. 1b = Data is captured on the first UCLK edge and changed on the following edge.
14	UCCKPL	RW	0h	Clock polarity select 0b = The inactive state is low. 1b = The inactive state is high.
13	UCMSB	RW	0h	MSB first select. Controls the direction of the receive and transmit shift register. 0b = LSB first 1b = MSB first
12	UC7BIT	RW	0h	Character length. Selects 7-bit or 8-bit character length. 0b = 8-bit data 1b = 7-bit data
11	UCMST	RW	0h	Master mode select 0b = Slave mode 1b = Master mode
10-9	UCMODEx	RW	0h	eUSCI mode. The UCMODEx bits select the synchronous mode when UCSYNC = 1. 00b = 3-pin SPI 01b = 4-pin SPI with UCxSTE active high: Slave enabled when UCxSTE = 1 10b = 4-pin SPI with UCxSTE active low: Slave enabled when UCxSTE = 0 11b = I2C mode
8	UCSYNC	RW	0h	Synchronous mode enable 0b = Asynchronous mode 1b = Synchronous mode
7-6	UCSSELx	RW	0h	eUSCI clock source select. These bits select the BRCLK source clock in master mode. UCxCLK is always used in slave mode. 00b = Reserved 01b = ACLK 10b = SMCLK 11b = SMCLK
5-2	Reserved	R	0h	Reserved
1	UCSTEM	RW	0h	STE mode select in master mode. This byte is ignored in slave or 3-wire mode. 0b = STE pin is used to prevent conflicts with other masters 1b = STE pin is used to generate the enable signal for a 4-wire slave
0	UCSWRST	RW	1h	Software reset enable 0b = Disabled. eUSCI reset released for operation. 1b = Enabled. eUSCI logic held in reset state.

11.4.2 UCAxBRW Register (address = 0146h) [reset = 0000h]

eUSCI_Ax Bit Rate Control Register 1

Figure 11-6. UCAxBRW Register

15	14	13	12	11	10	9	8
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw

Table 11-4. UCAxBRW Register Description

Bit	Field	Type	Reset	Description
15-0	UCBRx	RW	0h	Bit clock prescaler setting.

11.4.3 UCAxSTATW Register (address = 0148h) [reset = 0000h]

eUSCI_Ax Status Register

Figure 11-7. UCAxSTATW Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCLISTEN	UCFE	UCOE	Reserved				UCBUSY
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r-0

Table 11-5. UCAxSTATW Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7	UCLISTEN	RW	0h	Listen enable. The UCLISTEN bit selects loopback mode. 0b = Disabled 1b = Enabled. The transmitter output is internally fed back to the receiver.
6	UCFE	RW	0h	Framing error flag. This bit indicates a bus conflict in 4-wire master mode. UCFE is not used in 3-wire master or any slave mode. 0b = No error 1b = Bus conflict occurred
5	UCOE	RW	0h	Overrun error flag. This bit is set when a character is transferred into UCxRXBUF before the previous character was read. UCOE is cleared automatically when UCxRXBUF is read, and must not be cleared by software. Otherwise, it does not function correctly. 0b = No error 1b = Overrun error occurred
4-1	Reserved	RW	0h	Reserved
0	UCBUSY	R	0h	eUSCI busy. This bit indicates if a transmit or receive operation is in progress. 0b = eUSCI inactive 1b = eUSCI transmitting or receiving

11.4.4 UCAxRXBUF Register (address = 014Ch) [reset = 0000h]

eUSCI_Ax Receive Buffer Register

Figure 11-8. UCAxRXBUF Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCRXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

Table 11-6. UCAxRXBUF Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCRXBUFx	R	0h	The receive-data buffer is user accessible and contains the last received character from the receive shift register. Reading UCxRXBUF resets the receive-error bits and UCRXIFG. In 7-bit data mode, UCxRXBUF is LSB justified and the MSB is always reset.

11.4.5 UCAxTXBUF Register (address = 014Eh) [reset = 0000h]

eUSCI_Ax Transmit Buffer Register

Figure 11-9. UCAxTXBUF Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCTXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

Table 11-7. UCAxTXBUF Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCTXBUFx	RW	0h	The transmit data buffer is user accessible and holds the data waiting to be moved into the transmit shift register and transmitted. Writing to the transmit data buffer clears UCTXIFG. The MSB of UCxTXBUF is not used for 7-bit data and is reset.

11.4.6 UCAxIE Register (address = 015Ah) [reset = 0000h]

eUSCI_Ax Interrupt Enable Register

Figure 11-10. UCAxIE Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	rw-0	rw-0

Table 11-8. UCAxIE Register Description

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	UCTXIE	RW	0h	Transmit interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
0	UCRXIE	RW	0h	Receive interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled

11.4.7 UCAxIFG Register (address = 015Ch) [reset = 02h]

eUSCI_Ax Interrupt Flag Register

Figure 11-11. UCAxIFG Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved						UCTXIFG	UCRXIFG
r-0	r-0	r-0	r-0	r-0	r-0	rw-1	rw-0

Table 11-9. UCAxIFG Register Description

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	UCTXIFG	RW	1h	Transmit interrupt flag. UCTXIFG is set when UCAxTXBUF is empty. 0b = No interrupt pending 1b = Interrupt pending
0	UCRXIFG	RW	0h	Receive interrupt flag. UCRXIFG is set when UCAxRXBUF has received a complete character. 0b = No interrupt pending 1b = Interrupt pending

11.4.8 UCAxIV Register (address = 015Eh) [reset = 0000h]

eUSCI_Ax Interrupt Vector Register

Figure 11-12. UCAxIV Register

15	14	13	12	11	10	9	8
UCIVx							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCIVx							
r0	r0	r0	r-0	r-0	r-0	r-0	r0

Table 11-10. UCAxIV Register Description

Bit	Field	Type	Reset	Description
15-0	UCIVx	R	0h	eUSCI interrupt vector value 000h = No interrupt pending 002h = Interrupt Source: Data received; Interrupt Flag: UCRXIFG; Interrupt Priority: Highest 004h = Interrupt Source: Transmit buffer empty; Interrupt Flag: UCTXIFG; Interrupt Priority: Lowest

11.5 eUSCI_B SPI Registers

The eUSCI_B registers applicable in SPI mode and their addresses are listed in Table 11-11.

Table 11-11. eUSCI_B SPI Registers

Address	Acronym	Register Name	Type	Access	Reset	Section
01C0h	UCBxCTLW0	eUSCI_Bx Control Word 0	Read/write	Word	01C1h	Section 11.5.1
01C0h	UCBxCTL1	eUSCI_Bx Control 1	Read/write	Byte	C1h	
01C1h	UCBxCTL0	eUSCI_Bx Control 0	Read/write	Byte	01h	
01C6h	UCBxBRW	eUSCI_Bx Bit Rate Control Word	Read/write	Word	0000h	Section 11.5.2
01C6h	UCBxBR0	eUSCI_Bx Bit Rate Control 0	Read/write	Byte	00h	
01C7h	UCBxBR1	eUSCI_Bx Bit Rate Control 1	Read/write	Byte	00h	
01C8h	UCBxSTATW	eUSCI_Bx Status	Read/write	Word	00h	Section 11.5.3
01CCh	UCBxRXBUF	eUSCI_Bx Receive Buffer	Read/write	Word	00h	Section 11.5.4
01CEh	UCBxTXBUF	eUSCI_Bx Transmit Buffer	Read/write	Word	00h	Section 11.5.5
01EAh	UCBxIE	eUSCI_Bx Interrupt Enable	Read/write	Word	00h	Section 11.5.6
01ECh	UCBxIFG	eUSCI_Bx Interrupt Flag	Read/write	Word	02h	Section 11.5.7
01EEh	UCBxIV	eUSCI_Bx Interrupt Vector	Read	Word	0000h	Section 11.5.8

11.5.1 UCBxCTLW0 Register (address = 01C0h) [reset = 01C1h]

eUSCI_Bx Control Register 0

Figure 11-13. UCBxCTLW0 Register

15	14	13	12	11	10	9	8
UCCKPH	UCCKPL	UCMSB	UC7BIT	UCMST	UCMODEx		UCSYNC
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1
7	6	5	4	3	2	1	0
UCSSELx		Reserved				UCSTEM	UCSWRST
rw-1	rw-1	r0	rw-0	rw-0	rw-0	rw-0	rw-1

Table 11-12. UCBxCTLW0 Register Description

Bit	Field	Type	Reset	Description
15	UCCKPH	RW	0h	Clock phase select 0b = Data is changed on the first UCLK edge and captured on the following edge. 1b = Data is captured on the first UCLK edge and changed on the following edge.
14	UCCKPL	RW	0h	Clock polarity select 0b = The inactive state is low. 1b = The inactive state is high.
13	UCMSB	RW	0h	MSB first select. Controls the direction of the receive and transmit shift register. 0b = LSB first 1b = MSB first
12	UC7BIT	RW	0h	Character length. Selects 7-bit or 8-bit character length. 0b = 8-bit data 1b = 7-bit data
11	UCMST	RW	0h	Master mode select 0b = Slave mode 1b = Master mode
10-9	UCMODEx	RW	0h	eUSCI mode. The UCMODEx bits select the synchronous mode when UCSYNC = 1. 00b = 3-pin SPI 01b = 4-pin SPI with UCxSTE active high: Slave enabled when UCxSTE = 1 10b = 4-pin SPI with UCxSTE active low: Slave enabled when UCxSTE = 0 11b = I2C mode
8	UCSYNC	RW	1h	Synchronous mode enable 0b = Asynchronous mode 1b = Synchronous mode
7-6	UCSSELx	RW	3h	eUSCI clock source select. These bits select the BRCLK source clock in master mode. UCxCLK is always used in slave mode. 00b = Reserved 01b = ACLK 10b = SMCLK 11b = SMCLK
5-2	Reserved	R	0h	Reserved
1	UCSTEM	RW	0h	STE mode select in master mode. This byte is ignored in slave or 3-wire mode. 0b = STE pin is used to prevent conflicts with other masters 1b = STE pin is used to generate the enable signal for a 4-wire slave
0	UCSWRST	RW	1h	Software reset enable 0b = Disabled. eUSCI reset released for operation. 1b = Enabled. eUSCI logic held in reset state.

11.5.2 UCBxBRW Register (address = 01C6h) [reset = 0000h]

eUSCI_Bx Bit Rate Control Register 1

Figure 11-14. UCBxBRW Register

15	14	13	12	11	10	9	8
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw

Table 11-13. UCBxBRW Register Description

Bit	Field	Type	Reset	Description
15-0	UCBRx	RW	0h	Bit clock prescaler setting.

11.5.3 UCBxSTATW Register (address = 01C8h) [reset = 0000h]

eUSCI_Bx Status Register

Figure 11-15. UCBxSTATW Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCLISTEN	UCFE	UCOE	Reserved				UCBUSY
rw-0	rw-0	rw-0	r0	r0	r0	r0	r-0

Table 11-14. UCBxSTATW Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7	UCLISTEN	RW	0h	Listen enable. The UCLISTEN bit selects loopback mode. 0b = Disabled 1b = Enabled. The transmitter output is internally fed back to the receiver.
6	UCFE	RW	0h	Framing error flag. This bit indicates a bus conflict in 4-wire master mode. UCFE is not used in 3-wire master or any slave mode. 0b = No error 1b = Bus conflict occurred
5	UCOE	RW	0h	Overrun error flag. This bit is set when a character is transferred into UCxRXBUF before the previous character was read. UCOE is cleared automatically when UCxRXBUF is read, and must not be cleared by software. Otherwise, it does not function correctly. 0b = No error 1b = Overrun error occurred
4-1	Reserved	R	0h	Reserved
0	UCBUSY	R	0h	eUSCI busy. This bit indicates if a transmit or receive operation is in progress. 0b = eUSCI inactive 1b = eUSCI transmitting or receiving

11.5.4 UCBxRXBUF Register (address = 01CCh) [reset = 0000h]

eUSCI_Bx Receive Buffer Register

Figure 11-16. UCBxRXBUF Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCRXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

Table 11-15. UCBxRXBUF Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCRXBUFx	R	0h	The receive-data buffer is user accessible and contains the last received character from the receive shift register. Reading UCxRXBUF resets the receive-error bits and UCRXIFG. In 7-bit data mode, UCxRXBUF is LSB justified and the MSB is always reset.

11.5.5 UCBxTXBUF Register (address = 01CEh) [reset = 0000h]

eUSCI_Bx Transmit Buffer Register

Figure 11-17. UCBxTXBUF Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCTXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

Table 11-16. UCBxTXBUF Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCTXBUFx	RW	0h	The transmit data buffer is user accessible and holds the data waiting to be moved into the transmit shift register and transmitted. Writing to the transmit data buffer clears UCTXIFG. The MSB of UCxTXBUF is not used for 7-bit data and is reset.

11.5.6 UCBxIE Register (address = 01EAh) [reset = 0000h]

eUSCI_Bx Interrupt Enable Register

Figure 11-18. UCBxIE Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	rw-0	rw-0

Table 11-17. UCBxIE Register Description

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	UCTXIE	RW	0h	Transmit interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
0	UCRXIE	RW	0h	Receive interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled

11.5.7 UCBxIFG Register (address = 01ECh) [reset = 02h]

eUSCI_Bx Interrupt Flag Register

Figure 11-19. UCBxIFG Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved						UCTXIFG	UCRXIFG
r-0	r-0	r-0	r-0	r-0	r-0	rw-1	rw-0

Table 11-18. UCBxIFG Register Description

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	UCTXIFG	RW	1h	Transmit interrupt flag. UCTXIFG is set when UCBxTXBUF empty. 0b = No interrupt pending 1b = Interrupt pending
0	UCRXIFG	RW	0h	Receive interrupt flag. UCRXIFG is set when UCBxRXBUF has received a complete character. 0b = No interrupt pending 1b = Interrupt pending

11.5.8 UCBxIV Register (address = 01EEh) [reset = 0000h]

eUSCI_Bx Interrupt Vector Register

Figure 11-20. UCBxIV Register

15	14	13	12	11	10	9	8
UCIVx							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCIVx							
r0	r0	r0	r-0	r-0	r-0	r-0	r0

Table 11-19. UCBxIV Register Description

Bit	Field	Type	Reset	Description
15-0	UCIVx	R	0h	eUSCI interrupt vector value 0000h = No interrupt pending 0002h = Interrupt Source: Data received; Interrupt Flag: UCRXIFG; Interrupt Priority: Highest 0004h = Interrupt Source: Transmit buffer empty; Interrupt Flag: UCTXIFG; Interrupt Priority: Lowest

Enhanced Universal Serial Communication Interface (eUSCI) – I²C Mode

The enhanced universal serial communication interface B (eUSCI_B) supports multiple serial communication modes with one hardware module. This chapter discusses the operation of the I²C mode.

Topic **Page**

12.1 Enhanced Universal Serial Communication Interface B (eUSCI_B) Overview

The eUSCI_B module supports two serial communication modes:

- I^2C mode
- SPI mode

If more than one eUSCI_B module is implemented on one device, those modules are named with incrementing numbers. For example, if one device has two eUSCI_B modules, they are named eUSCI0_B and eUSCI1_B.

12.2 eUSCI_B Introduction – I^2C Mode

In I^2C mode, the eUSCI_B module provides an interface between the device and I^2C-compatible devices connected by the two-wire I^2C serial bus. External components attached to the I^2C bus serially transmit data to and receive data from the eUSCI_B module through the 2-wire I^2C interface.

The eUSCI_B I^2C mode features include:

- 7-bit and 10-bit device addressing modes
- General call
- START/RESTART/STOP
- Multi-master transmitter/receiver mode
- Slave receiver/transmitter mode
- Standard mode up to 100 kbps and fast mode up to 400 kbps support
- Programmable UCxCLK frequency in master mode
- Designed for low power
- 8-bit byte counter with interrupt capability and automatic STOP assertion
- Up to four hardware slave addresses, each having its own interrupt and DMA trigger
- Mask register for slave address and address received interrupt
- Clock low timeout interrupt to avoid bus stalls
- Slave operation in LPM4
- Slave receiver START detection for automatic wake-up from LPMx modes (not LPM4.5)

Figure 12-1 shows the eUSCI_B when configured in I^2C mode.

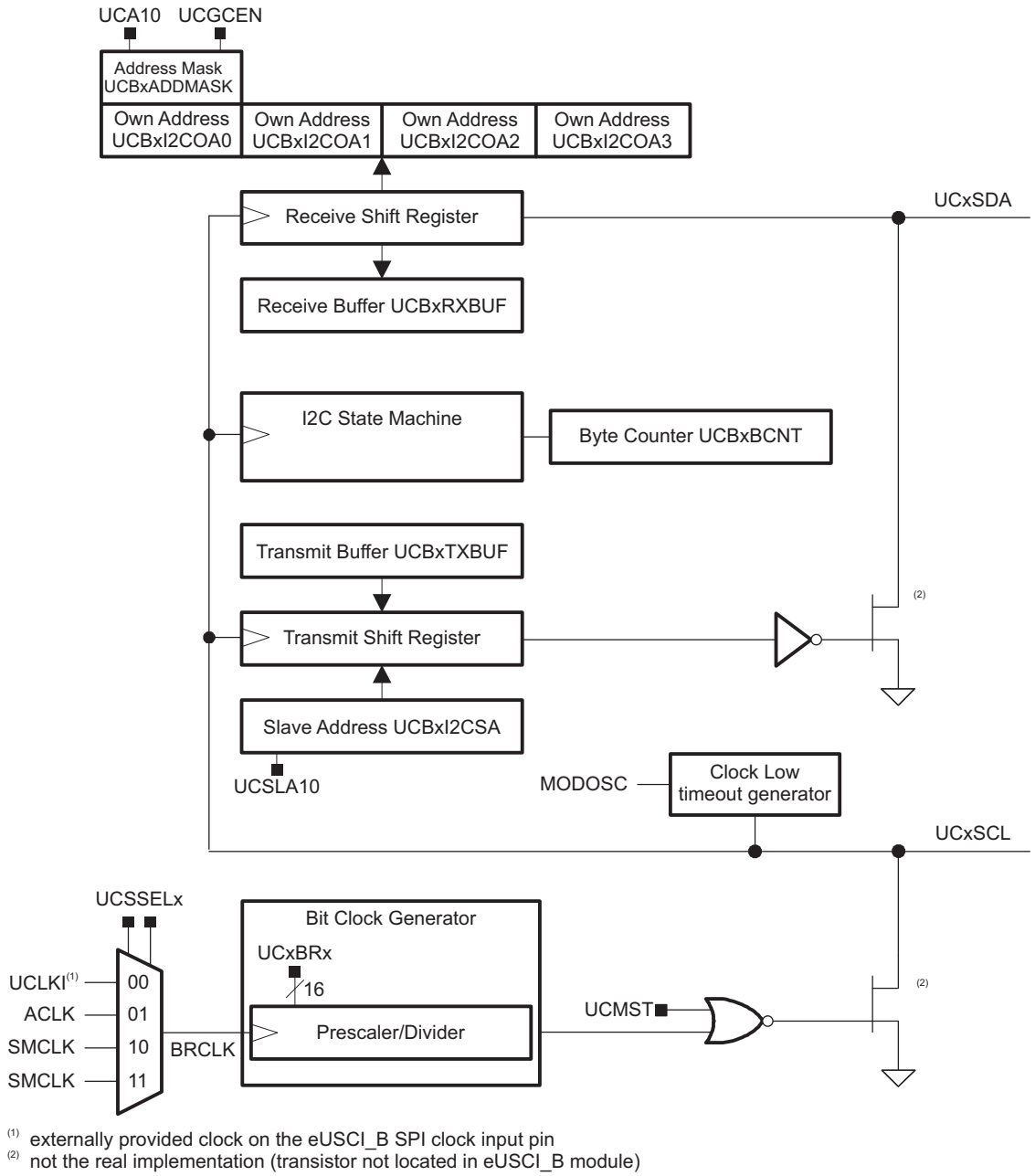


Figure 12-1. eUSCI_B Block Diagram – I²C Mode

12.3 eUSCI_B Operation – I²C Mode

The I²C mode supports any slave or master I²C-compatible device. Figure 12-2 shows an example of an I²C bus. Each I²C device is recognized by a unique address and can operate as either a transmitter or a receiver. A device connected to the I²C bus can be considered as the master or the slave when performing data transfers. A master initiates a data transfer and generates the clock signal SCL. Any device addressed by a master is considered a slave.

I²C data is communicated using the serial data (SDA) pin and the serial clock (SCL) pin. Both SDA and SCL are bidirectional and must be connected to a positive supply voltage using a pullup resistor.

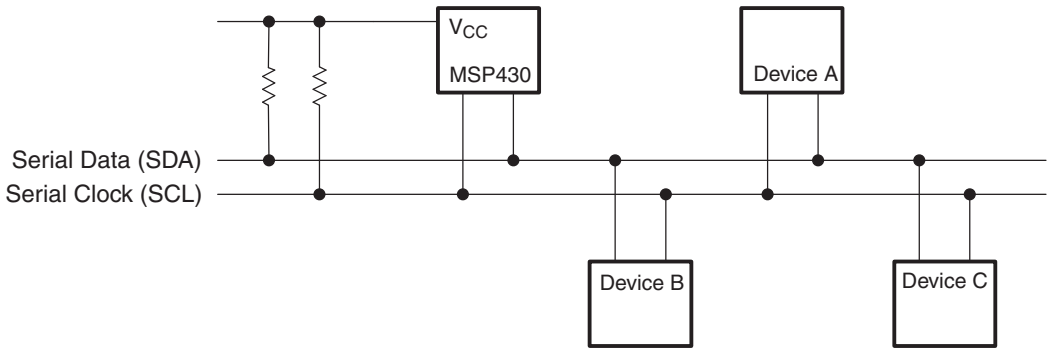


Figure 12-2. I²C Bus Connection Diagram

NOTE: SDA and SCL levels

The SDA and SCL pins must not be pulled up above the device V_{CC} level.

12.3.1 eUSCI_B Initialization and Reset

The eUSCI_B is reset by a PUC or by setting the UCSWRST bit. After a PUC, the UCSWRST bit is automatically set, which keeps the eUSCI_B in a reset condition. To select I²C operation, the UCMODEx bits must be set to 11. After module initialization, it is ready for transmit or receive operation. Clearing UCSWRST releases the eUSCI_B for operation.

Configuring and reconfiguring the eUSCI_B module should be done when UCSWRST is set to avoid unpredictable behavior. Setting UCSWRST in I²C mode has the following effects:

- I²C communication stops.
- SDA and SCL are high impedance.
- UCBxSTATW bits 15-8 and 6-4 are cleared.
- Registers UCBxIE and UCBxIFG are cleared.
- All other bits and registers remain unchanged.

NOTE: Initializing or reconfiguring the eUSCI_B module

The recommended eUSCI_B initialization or reconfiguration process is:

1. Set UCSWRST: `BIS.B #UCSWRST,&UCxCTL1`
2. Initialize all eUSCI_B registers with UCSWRST = 1 (including UCxCTL1)
3. Configure ports
4. Clear UCSWRST: `BIC.B #UCSWRST,&UCxCTL1`
5. Enable interrupts (optional)

12.3.2 I²C Serial Data

One clock pulse is generated by the master device for each data bit transferred. The I²C mode operates with byte data. Data is transferred MSB first as shown in Figure 12-3.

The first byte after a START condition consists of a 7-bit slave address and the R/$\overline{W}$ bit. When R/$\overline{W}$ = 0, the master transmits data to a slave. When R/$\overline{W}$ = 1, the master receives data from a slave. The ACK bit is sent from the receiver after each byte on the ninth SCL clock.

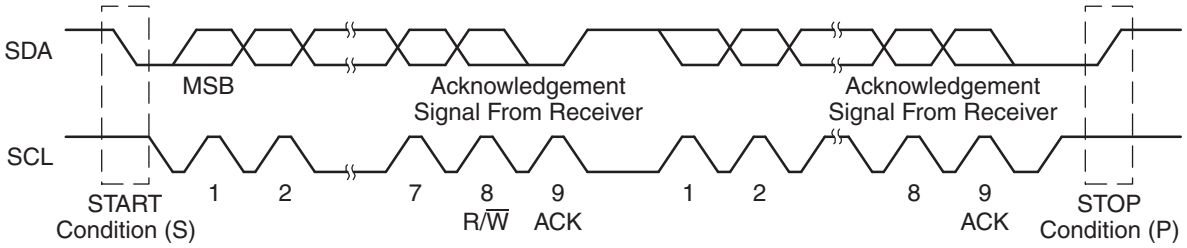


Figure 12-3. I²C Module Data Transfer

START and STOP conditions are generated by the master and are shown in Figure 12-3. A START condition is a high-to-low transition on the SDA line while SCL is high. A STOP condition is a low-to-high transition on the SDA line while SCL is high. The bus busy bit, UCBBUSY, is set after a START and cleared after a STOP.

Data on SDA must be stable during the high period of SCL (see Figure 12-4). The high and low state of SDA can change only when SCL is low, otherwise START or STOP conditions are generated.

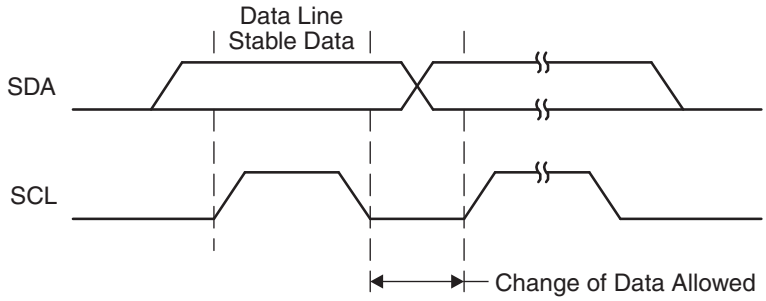


Figure 12-4. Bit Transfer on I²C Bus

12.3.3 I²C Addressing Modes

The I²C mode supports 7-bit and 10-bit addressing modes.

12.3.3.1 7-Bit Addressing

In the 7-bit addressing format (see Figure 12-5), the first byte is the 7-bit slave address and the R/$\overline{W}$ bit. The ACK bit is sent from the receiver after each byte.

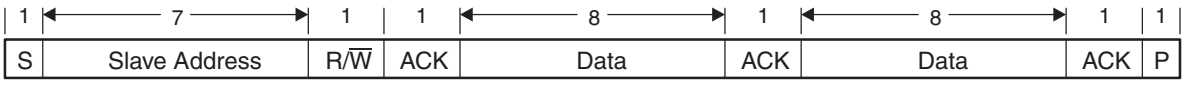


Figure 12-5. I²C Module 7-Bit Addressing Format

12.3.3.2 10-Bit Addressing

In the 10-bit addressing format (see Figure 12-6), the first byte is made up of 11110b plus the two MSBs of the 10-bit slave address and the R/$\overline{W}$ bit. The ACK bit is sent from the receiver after each byte. The next byte is the remaining eight bits of the 10-bit slave address, followed by the ACK bit and the 8-bit data. See I2C Slave 10-bit Addressing Mode and I2C Master 10-bit Addressing Mode for details how to use the 10-bit addressing mode with the eUSCI_B module.

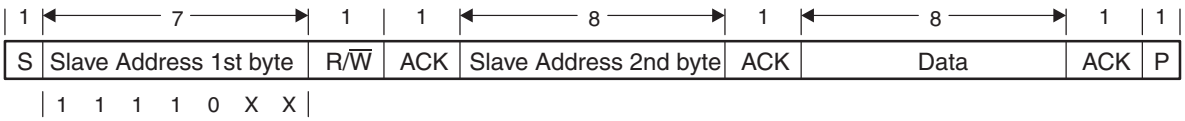


Figure 12-6. I²C Module 10-Bit Addressing Format

12.3.3.3 Repeated Start Conditions

The direction of data flow on SDA can be changed by the master, without first stopping a transfer, by issuing a repeated START condition. This is called a RESTART. After a RESTART is issued, the slave address is again sent out with the new data direction specified by the R/$\overline{W}$ bit. The RESTART condition is shown in Figure 12-7.

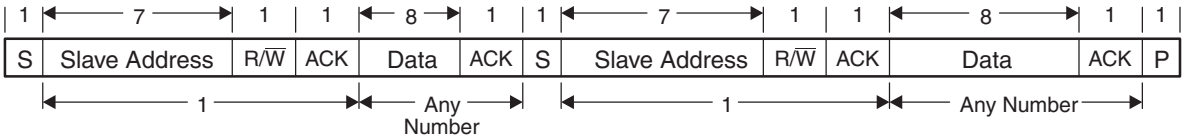


Figure 12-7. I²C Module Addressing Format With Repeated START Condition

12.3.4 I²C Quick Setup

This section is a short introduction to the operation of the eUSCI_B in I²C mode. The basic steps to start communication are described and shown as a software example. More detailed information about the possible configurations and details can be found in Section 12.3.5.

The latest code examples can be found on the MSP430 web under "Code Examples".

To set up the eUSCI_B as a master transmitter that transmits to a slave with the address 12h, only a few steps are needed (see Example 12-1).

Example 12-1. Master TX With 7-Bit Address

```
UCBxCTL1 |= UCSWRST;           // put eUSCI_B in reset state
UCBxCTLW0 |= UCMODE_3 + UCMST; // I2C master mode
UCBxBRW = 0x0008;              // baud rate = SMCLK / 8
UCBxCTLW1 = UCASTP_2;          // autom. STOP assertion
UCBxTBCNT = 0x07;              // TX 7 bytes of data
UCBxI2CSA = 0x0012;            // address slave is 12hex
P2SEL |= 0x03;                 // configure I2C pins (device specific)
UCBxCTL1 &= ^UCSWRST;          // eUSCI_B in operational state
UCBxIE |= UCTXIE;              // enable TX-interrupt
GIE;                           // general interrupt enable
...
// inside the eUSCI_B TX interrupt service routine
UCBxTXBUF = 0x77;              // fill TX buffer
```

As shown in the code example, all configurations must be done while UCSWRST is set. To select the I²C operation of the eUSCI_B, UCMODE must be set accordingly. The baud rate of the transmission is set by writing the correct divider in the UCBxBRW register. The default clock selected is SMCLK. How many bytes are transmitted in one frame is controlled by the byte counter threshold register UCBxTBCNT together with the UCASTPx bits.

The slave address to send to is specified in the UCBxI2CSA register. Finally, the ports must be configured. This step is device dependent; see the data sheet for the pins that must be used.

Each byte that is to be transmitted must be written to the UCBxTXBUF inside the interrupt service routine. The recommended structure of the interrupt service routine can be found in Example 12-3.

Example 12-2 shows the steps needed to set up the eUSCI_B as a slave with the address 12h that is able to receive and transmit data to the master.

Example 12-2. Slave RX With 7-Bit Address

```
UCBxCTL1 |= UCSWRST;        // eUSCI_B in reset state
UCBxCTLW0 |= UCMODE_3;      // I2C slave mode
UCBxI2COA0 = 0x0012;        // own address is 12hex
P2SEL |= 0x03;              // configure I2C pins (device specific)
UCBxCTL1 &= ^UCSWRST;       // eUSCI_B in operational state
UCBxIE |= UCTXIE + UCRXIE;  // enable TX&RX-interrupt
GIE;                        // general interrupt enable
...
// inside the eUSCI_B TX interrupt service routine
UCBxTXBUF = 0x77;           // send 077h
...
// inside the eUSCI_B RX interrupt service routine
data = UCBxRXBUF;           // data is the internal variable
```

As shown in Example 12-2, all configurations must be done while UCSWRST is set. For the slave, I²C operation is selected by setting UCMODE. The slave address is specified in the UCBxI2COA0 register. To enable the interrupts for receive and transmit requests, the according bits in UCBxIE and, at the end, GIE need to be set. Finally the ports must be configured. This step is device dependent; see the data sheet for the pins that are used.

The RX interrupt service routine is called for every byte received by a master device. The TX interrupt service routine is executed each time the master requests a byte. The recommended structure of the interrupt service routine can be found in Example 12-3.

12.3.5 I²C Module Operating Modes

In I²C mode, the eUSCI_B module can operate in master transmitter, master receiver, slave transmitter, or slave receiver mode. The modes are discussed in the following sections. Time lines are used to illustrate the modes.

Figure 12-8 shows how to interpret the time-line figures. Data transmitted by the master is represented by grey rectangles; data transmitted by the slave is represented by white rectangles. Data transmitted by the eUSCI_B module, either as master or slave, is shown by rectangles that are taller than the others.

Actions taken by the eUSCI_B module are shown in grey rectangles with an arrow indicating where in the data stream the action occurs. Actions that must be handled with software are indicated with white rectangles with an arrow pointing to where in the data stream the action must take place.

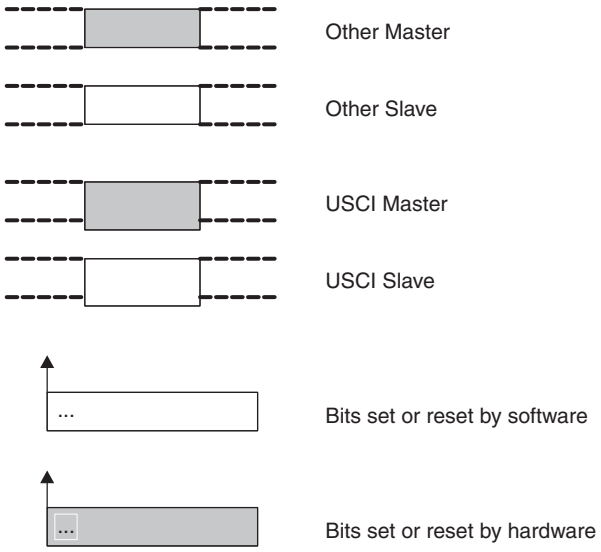


Figure 12-8. I²C Time-Line Legend

12.3.5.1 Slave Mode

The eUSCI_B module is configured as an I²C slave by selecting the I²C mode with UCMODEx = 11 and UCSYNC = 1 and clearing the UCMST bit.

Initially, the eUSCI_B module must be configured in receiver mode by clearing the UCTR bit to receive the I²C address. Afterwards, transmit and receive operations are controlled automatically, depending on the R/$\overline{W}$ bit received together with the slave address.

The eUSCI_B slave address is programmed with the UCBxI2COA0 register. Support for multiple slave addresses is explained in Section 12.3.9. When UCA10 = 0, 7-bit addressing is selected. When UCA10 = 1, 10-bit addressing is selected. The UCGCEN bit selects if the slave responds to a general call.

When a START condition is detected on the bus, the eUSCI_B module receives the transmitted address and compares it against its own address stored in UCBxI2COA0. The UCSTTIFG flag is set when address received matches the eUSCI_B slave address.

12.3.5.1.1 I²C Slave Transmitter Mode

Slave transmitter mode is entered when the slave address transmitted by the master is identical to its own address with a set R/$\overline{W}$ bit. The slave transmitter shifts the serial data out on SDA with the clock pulses that are generated by the master device. The slave device does not generate the clock, but it does hold SCL low while intervention of the CPU is required after a byte has been transmitted.

If the master requests data from the slave, the eUSCI_B module is automatically configured as a transmitter and UCTR and UCTXIFG0 become set. The SCL line is held low until the first data to be sent is written into the transmit buffer UCBxTXBUF. Then the address is acknowledged and the data is transmitted. As soon as the data is transferred into the shift register, the UCTXIFG0 is set again. After the data is acknowledged by the master, the next data byte written into UCBxTXBUF is transmitted or, if the buffer is empty, the bus is stalled during the acknowledge cycle by holding SCL low until new data is written into UCBxTXBUF. If the master sends a NACK followed by a STOP condition, the UCSTPIFG flag is set. If the NACK is followed by a repeated START condition, the eUSCI_B I²C state machine returns to its address-reception state.

Figure 12-9 shows the slave transmitter operation.

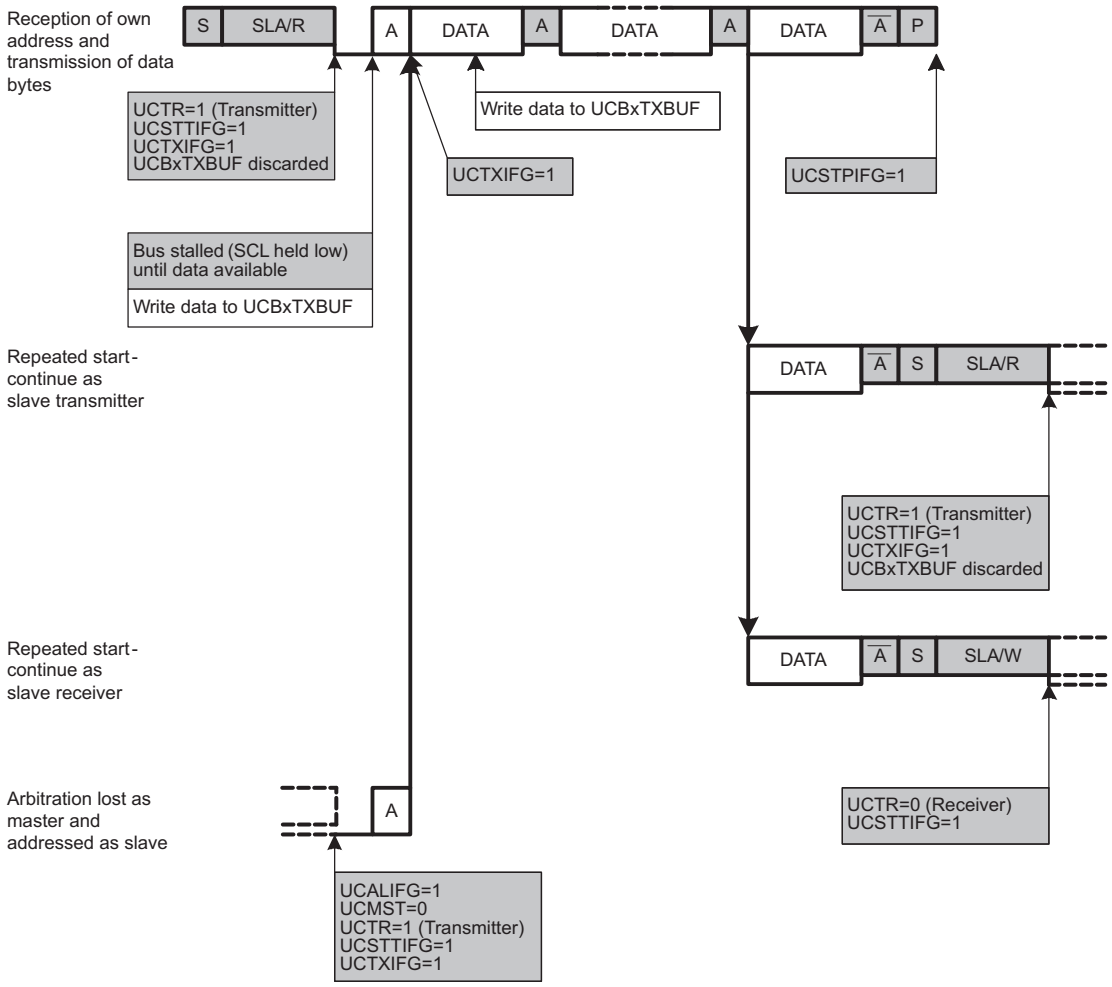


Figure 12-9. I²C Slave Transmitter Mode

12.3.5.1.2 I²C Slave Receiver Mode

Slave receiver mode is entered when the slave address transmitted by the master is identical to its own address and a cleared R/$\overline{W}$ bit is received. In slave receiver mode, serial data bits received on SDA are shifted in with the clock pulses that are generated by the master device. The slave device does not generate the clock, but it can hold SCL low if intervention of the CPU is required after a byte has been received.

If the slave receives data from the master, the eUSCI_B module is automatically configured as a receiver and UCTR is cleared. After the first data byte is received, the receive interrupt flag UCRXIFG0 is set. The eUSCI_B module automatically acknowledges the received data and can receive the next data byte.

If the previous data was not read from the receive buffer UCBxRXBUF at the end of a reception, the bus is stalled by holding SCL low. As soon as UCBxRXBUF is read, the new data is transferred into UCBxRXBUF, an acknowledge is sent to the master, and the next data can be received.

Setting the UCTXNACK bit causes a NACK to be transmitted to the master during the next acknowledgment cycle. A NACK is sent even if UCBxRXBUF is not ready to receive the latest data. If the UCTXNACK bit is set while SCL is held low, the bus is released, a NACK is transmitted immediately, and UCBxRXBUF is loaded with the last received data. Because the previous data was not read, that data is lost. To avoid loss of data, the UCBxRXBUF must be read before UCTXNACK is set**.**

When the master generates a STOP condition, the UCSTPIFG flag is set.

If the master generates a repeated START condition, the eUSCI_B I²C state machine returns to its address-reception state.

Figure 12-10 shows the I²C slave receiver operation.

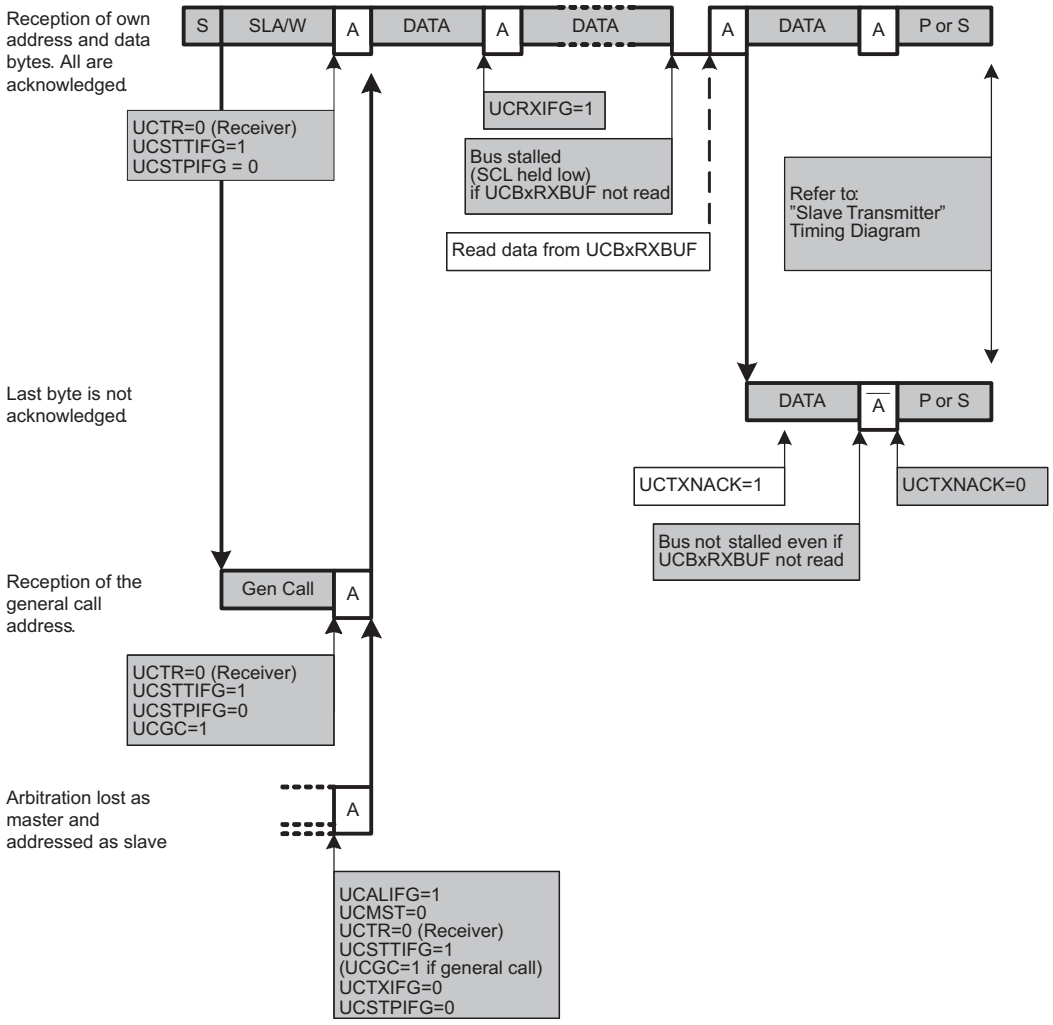


Figure 12-10. I²C Slave Receiver Mode

12.3.5.1.3 I²C Slave 10-Bit Addressing Mode

The 10-bit addressing mode is selected when UCA10 = 1 and is as shown in Figure 12-11. In 10-bit addressing mode, the slave is in receive mode after the full address is received. The eUSCI_B module indicates this by setting the UCSTTIFG flag while the UCTR bit is cleared. To switch the slave into transmitter mode, the master sends a repeated START condition together with the first byte of the address but with the R/$\overline{W}$ bit set. This sets the UCSTTIFG flag if it was previously cleared by software, and the eUSCI_B modules switches to transmitter mode with UCTR = 1.

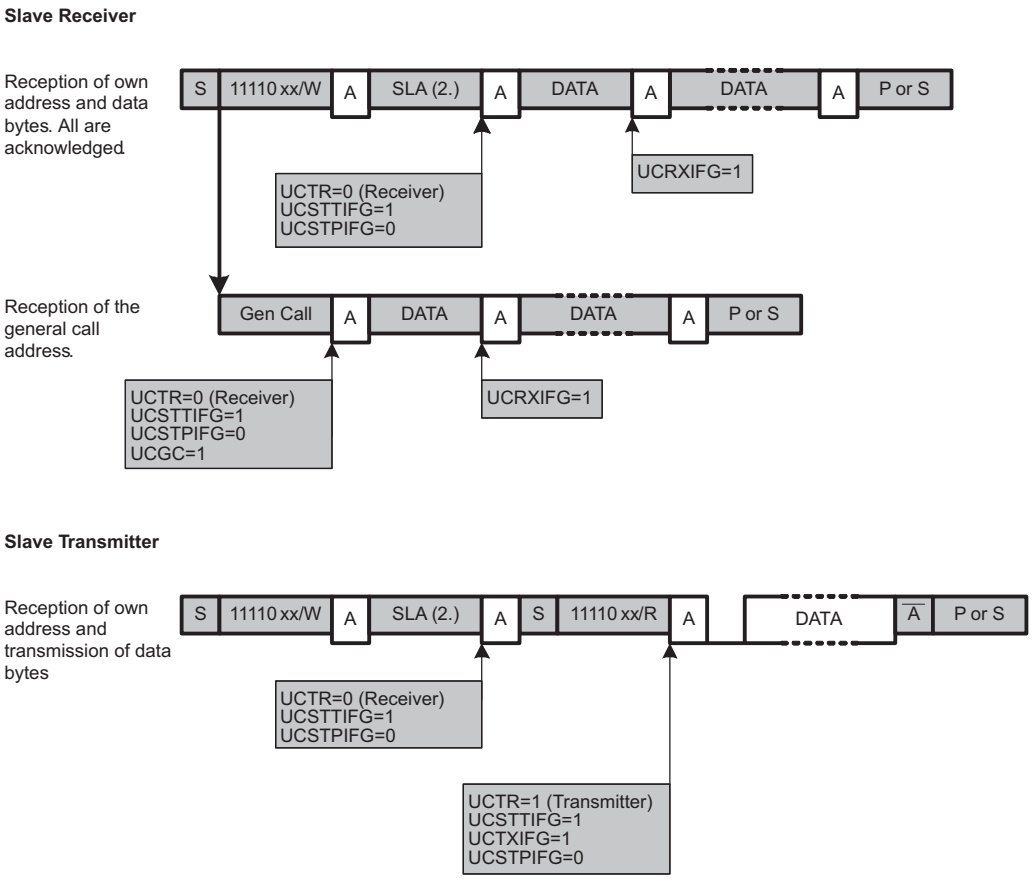


Figure 12-11. I²C Slave 10-Bit Addressing Mode

12.3.5.2 Master Mode

The eUSCI_B module is configured as an I²C master by selecting the I²C mode with UCMODEx = 11 and UCSYNC = 1 and setting the UCMST bit. When the master is part of a multi-master system, UCMM must be set and its own address must be programmed into the UCBxI2COA0 register. Support for multiple slave addresses is explained in Section 12.3.9. When UCA10 = 0, 7-bit addressing is selected. When UCA10 = 1, 10-bit addressing is selected. The UCGCEN bit selects if the eUSCI_B module responds to a general call.

NOTE: Addresses and multi-master systems

In master mode with own-address detection enabled (UCOAEN = 1)—especially in multi-master systems—it is not allowed to specify the same address in the own address and slave address register (UCBxI2CSA = UCBxI2COAx). This would mean that the eUSCI_B addresses itself.

The user software must ensure that this situation does not occur. There is no hardware detection for this case, and the consequence is unpredictable behavior of the eUSCI_B.

12.3.5.2.1 I²C Master Transmitter Mode

After initialization, master transmitter mode is initiated by writing the desired slave address to the UCBxI2CSA register, selecting the size of the slave address with the UCSLA10 bit, setting UCTR for transmitter mode, and setting UCTXSTT to generate a START condition.

The eUSCI_B module waits until the bus is available, then generates the START condition, and transmits the slave address. The UCTXIFG0 bit is set when the START condition is generated and the first data to be transmitted can be written into UCBxTXBUF. **The UCTXSTT flag is cleared as soon as the complete address is sent.**

The data written into UCBxTXBUF is transmitted if arbitration is not lost during transmission of the slave address. UCTXIFG0 is set again as soon as the data is transferred from the buffer into the shift register. If there is no data loaded to UCBxTXBUF before the acknowledge cycle, the bus is held during the acknowledge cycle with SCL low until data is written into UCBxTXBUF. Data is transmitted or the bus is held, as long as:

- No automatic STOP is generated
- The UCTXSTP bit is not set
- The UCTXSTT bit is not set

Setting UCTXSTP generates a STOP condition after the next acknowledge from the slave. If UCTXSTP is set during the transmission of the slave address or while the eUSCI_B module waits for data to be written into UCBxTXBUF, a STOP condition is generated, even if no data was transmitted to the slave. **In this case, the UCSTPIFG is set.** When transmitting a single byte of data, the UCTXSTP bit must be set while the byte is being transmitted or any time after transmission begins, without writing new data into UCBxTXBUF. Otherwise, only the address is transmitted. When the data is transferred from the buffer to the shift register, UCTXIFG0 is set, indicating data transmission has begun, and the UCTXSTP bit may be set. When UCASTPx = 10 is set, the byte counter is used for STOP generation and the user does not need to set the UCTXSTP. **This is recommended when transmitting only one byte.**

Setting UCTXSTT generates a repeated START condition. In this case, UCTR may be set or cleared to configure transmitter or receiver, and a different slave address may be written into UCBxI2CSA, if desired.

If the slave does not acknowledge the transmitted data, the not-acknowledge interrupt flag UCNACKIFG is set. The master must react with either a STOP condition or a repeated START condition. If data was already written into UCBxTXBUF, it is discarded. If this data should be transmitted after a repeated START, it must be written into UCBxTXBUF again. Any set UCTXSTT or UCTXSTP is also discarded.

Figure 12-12 shows the I²C master transmitter operation.

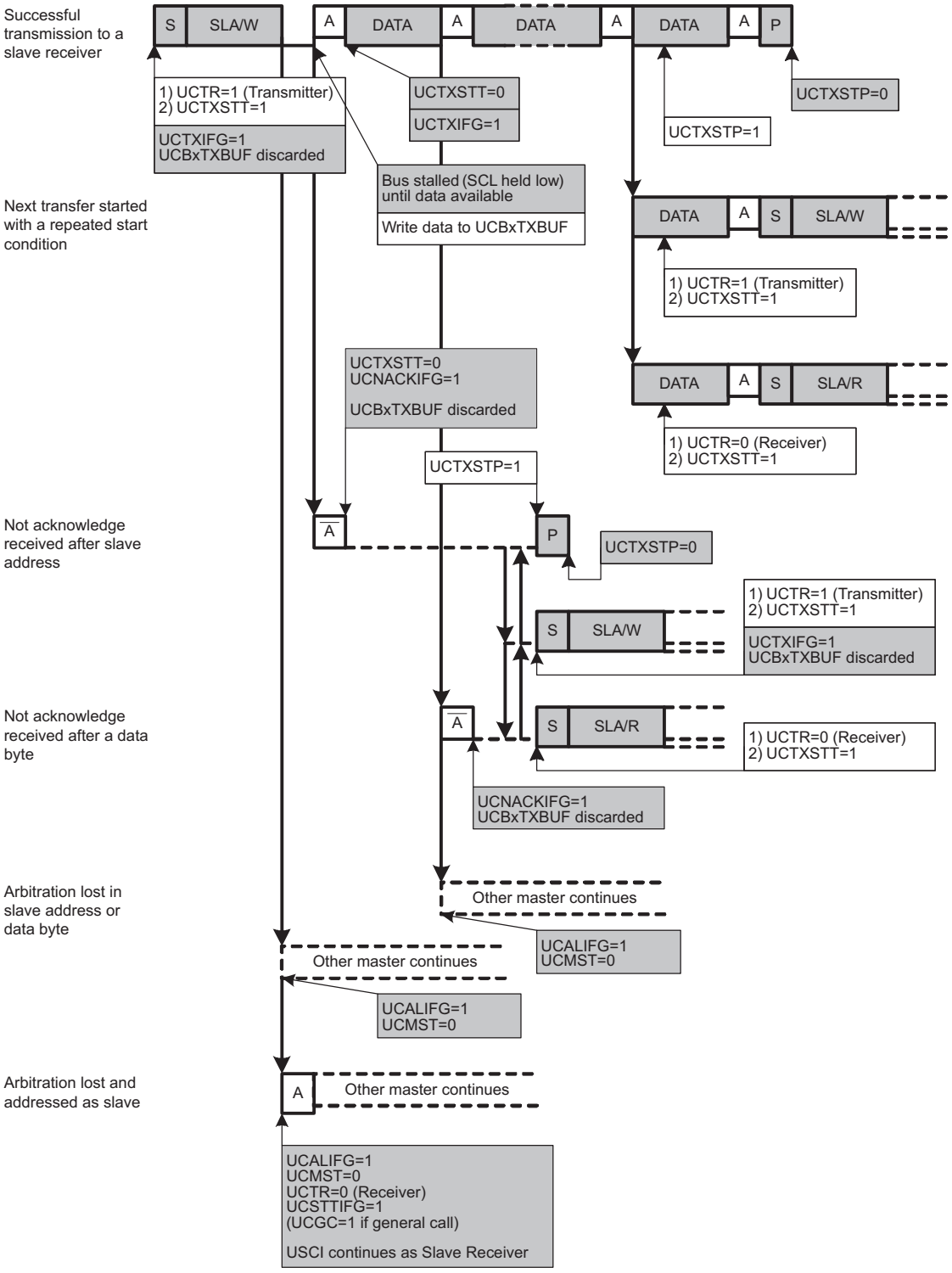


Figure 12-12. I²C Master Transmitter Mode

12.3.5.2.2 I²C Master Receiver Mode

After initialization, master receiver mode is initiated by writing the desired slave address to the UCBxI2CSA register, selecting the size of the slave address with the UCSLA10 bit, clearing UCTR for receiver mode, and setting UCTXSTT to generate a START condition.

The eUSCI_B module checks if the bus is available, generates the START condition, and transmits the slave address. The UCTXSTT flag is cleared as soon as the complete address is sent.

After the acknowledge of the address from the slave, the first data byte from the slave is received and acknowledged and the UCRXIFG flag is set. Data is received from the slave, as long as:

- No automatic STOP is generated
- The UCTXSTP bit is not set
- The UCTXSTT bit is not set

If a STOP condition was generated by the eUSCI_B module, the UCSTPIFG is set. If UCBxRXBUF is not read, the master holds the bus during reception of the last data bit and until the UCBxRXBUF is read.

If the slave does not acknowledge the transmitted address, the not-acknowledge interrupt flag UCNACKIFG is set. The master must react with either a STOP condition or a repeated START condition.

A STOP condition is either generated by the automatic STOP generation or by setting the UCTXSTP bit. The next byte received from the slave is followed by a NACK and a STOP condition. This NACK occurs immediately if the eUSCI_B module is currently waiting for UCBxRXBUF to be read.

If a RESTART is sent, UCTR may be set or cleared to configure transmitter or receiver, and a different slave address may be written into UCBxI2CSA if desired.

Figure 12-13 shows the I²C master receiver operation.

NOTE: Consecutive master transactions without repeated START

When performing multiple consecutive I²C master transactions without the repeated START feature, the current transaction must be completed before the next one is initiated. This can be done by ensuring that the transmit STOP condition flag UCTXSTP is cleared before the next I²C transaction is initiated with setting UCTXSTT = 1. Otherwise, the current transaction might be affected.

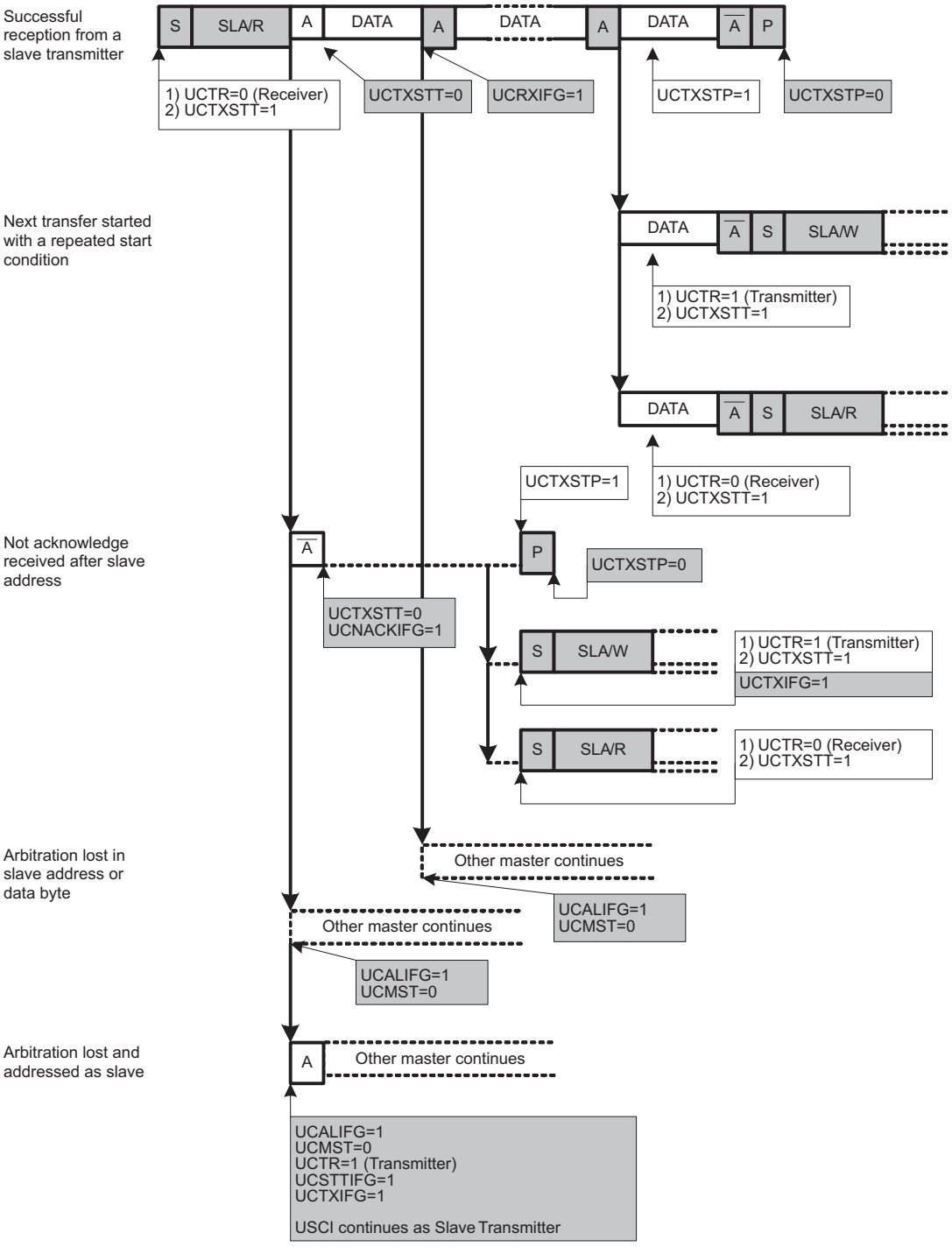


Figure 12-13. I²C Master Receiver Mode

12.3.5.2.3 I²C Master 10-Bit Addressing Mode

The 10-bit addressing mode is selected when UCSLA10 = 1 and is shown in Figure 12-14.

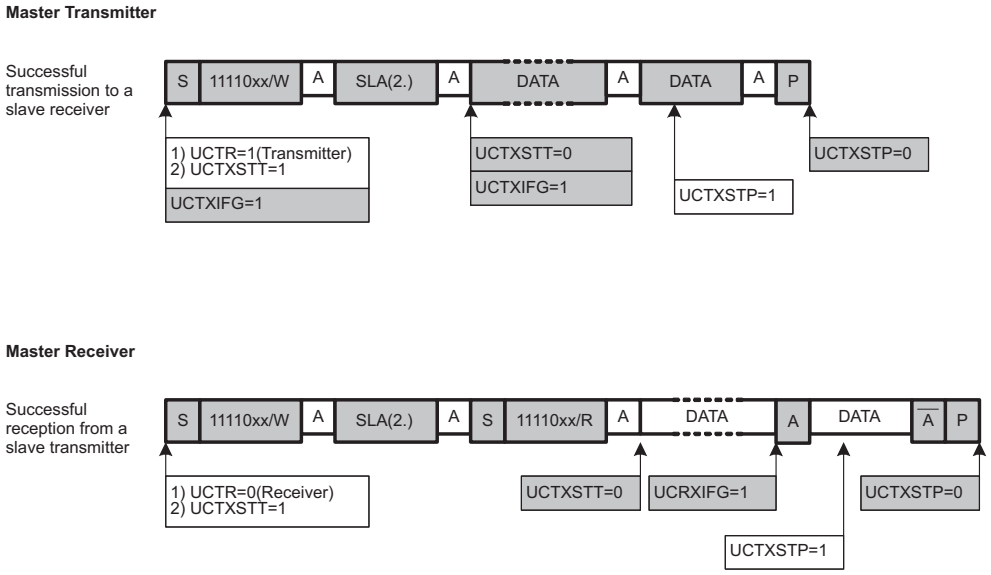


Figure 12-14. I²C Master 10-Bit Addressing Mode

12.3.5.3 Arbitration

If two or more master transmitters simultaneously start a transmission on the bus, an arbitration procedure is invoked. Figure 12-15 shows the arbitration procedure between two devices. The arbitration procedure uses the data presented on SDA by the competing transmitters. The first master transmitter that generates a logic high is overruled by the opposing master generating a logic low. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. The master transmitter that lost arbitration switches to the slave receiver mode and sets the arbitration lost flag UCALIFG. If two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

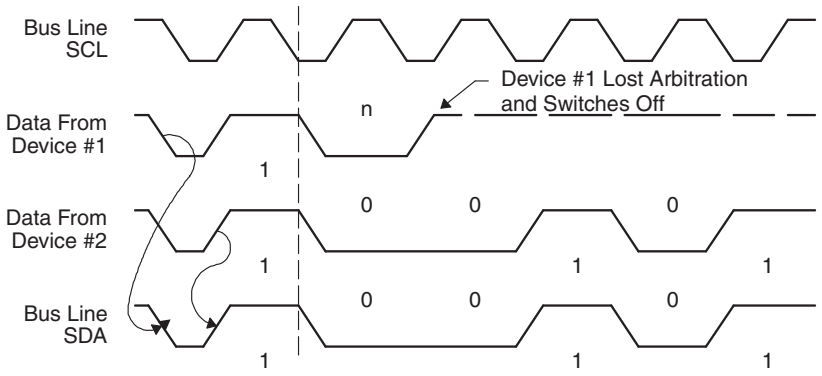


Figure 12-15. Arbitration Procedure Between Two Master Transmitters

There is an undefined condition if the arbitration procedure is still in progress when one master sends a repeated START or a STOP condition while the other master is still sending data. In other words, the following combinations result in an undefined condition:

- Master 1 sends a repeated START condition and master 2 sends a data bit.
- Master 1 sends a STOP condition and master 2 sends a data bit.
- Master 1 sends a repeated START condition and master 2 sends a STOP condition.

12.3.6 Glitch Filtering

According to the I²C standard, both the SDA and the SCL line need to be glitch filtered. The eUSCI_B module provides the UCGLITx bits to configure the length of this glitch filter:

Table 12-1. Glitch Filter Length Selection Bits

UCGLITx	Corresponding Glitch Filter Length on SDA and SCL	According to I²C Standard
00	Pulses of maximum 50-ns length are filtered	yes
01	Pulses of maximum 25-ns length are filtered.	no
10	Pulses of maximum 12.5-ns length are filtered.	no
11	Pulses of maximum 6.25-ns length are filtered.	no

12.3.7 I²C Clock Generation and Synchronization

The I²C clock SCL is provided by the master on the I²C bus. When the eUSCI_B is in master mode, BITCLK is provided by the eUSCI_B bit clock generator and the clock source is selected with the UCSSELx bits. In slave mode, the bit clock generator is not used and the UCSSELx bits are don't care.

The 16-bit value of UCBRx in registers UCBxBR1 and UCBxBR0 is the division factor of the eUSCI_B clock source, BRCLK. The maximum bit clock that can be used in single master mode is f_{BRCLK} / 4. In multi-master mode, the maximum bit clock is f_{BRCLK} / 8. The BITCLK frequency is given by:

$f_{BitClock} = f_{BRCLK}$ / UCBRx

The minimum high and low periods of the generated SCL are:

$t_{LOW,MIN} = t_{HIGH,MIN} = (UCBRx / 2) / f_{BRCLK}$ when UCBRx is even

$t_{LOW,MIN} = t_{HIGH,MIN} = ((UCBRx – 1) / 2) / f_{BRCLK}$ when UCBRx is odd

The eUSCI_B clock source frequency and the prescaler setting UCBRx must to be chosen such that the minimum low and high period times of the I²C specification are met.

During the arbitration procedure the clocks from the different masters must be synchronized. A device that first generates a low period on SCL overrules the other devices, forcing them to start their own low periods. SCL is then held low by the device with the longest low period. The other devices must wait for SCL to be released before starting their high periods. Figure 12-16 shows the clock synchronization. This allows a slow slave to slow down a fast master.

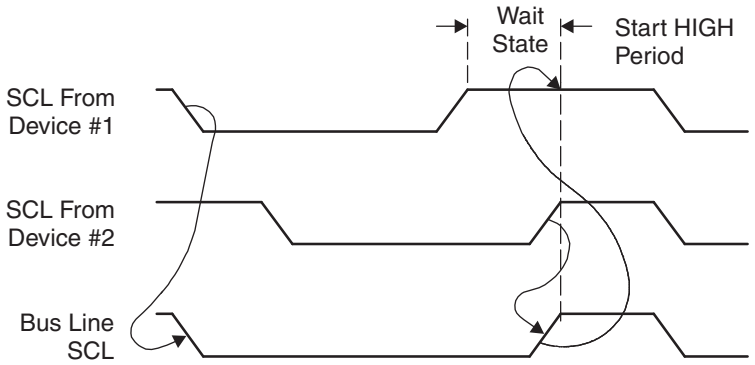


Figure 12-16. Synchronization of Two I²C Clock Generators During Arbitration

12.3.7.1 Clock Stretching

The eUSCI_B module supports clock stretching and also makes use of this feature as described in the Operation Mode sections.

The UCSCLLOW bit can be used to observe if another device pulls SCL low while the eUSCI_B module already released SCL due to the following conditions:

- eUSCI_B is acting as master and a connected slave drives SCL low.

- eUSCI_B is acting as master and another master drives SCL low during arbitration.

The UCSCLLOW bit is also active if the eUSCI_B holds SCL low because it is waiting as transmitter for data being written into UCBxTXBUF or as receiver for the data being read from UCBxRXBUF. The UCSCLLOW bit might be set for a short time with each rising SCL edge because the logic observes the external SCL and compares it to the internally generated SCL.

12.3.7.2 Avoiding Clock Stretching

Even though clock stretching is part of the I2C specification, there are applications in which clock stretching should be avoided.

The clock is stretched by the eUSCI_B under the following conditions:

- The internal shift register is expecting data, but the TXIFG is still pending
- The internal shift register is full, but the RXIFG is still pending
- The arbitration lost interrupt is pending
- UCSWACK is selected and UCBxI2COA0 did cause a match

To avoid clock stretching, all of these situations for clock stretch either need to be avoided or the corresponding interrupt flags need to be processed before the actual clock stretch can occur.

Using the DMA (on devices that contain a DMA) is the most secure way to avoid clock stretching. If no DMA is available, the software must ensure that the corresponding interrupts are serviced in time before the clock is stretched.

In slave transmitter mode, the TXIFG is set only after the reception of the direction bit; therefore, there is only a short amount of time for the software to write the TXBUF before a clock stretch occurs. This situation can be remedied by using the early Transmit Interrupt (see Section 12.3.11.2).

12.3.7.3 Clock Low Timeout

The UCCLTOIFG interrupt allows the software to react if the clock is low longer than a defined time. It is possible to detect the situation, when a clock is stretched by a master or slave for a too long time. The user can then, for example, reset the eUSCI_B module by using the UCSWRST bit.

The clock low timeout feature is enabled using the UCCLTO bits. It is possible to select one of three predefined times for the clock low timeout. If the clock has been low longer than the time defined with the UCCLTO bits and the eUSCI_B was actively receiving or transmitting, the UCCLTOIFG is set and an interrupt request is generated if UCCLTOIE and GIE are set as well. The UCCLTOIFG is set only once, even if the clock is stretched a multiple of the time defined in UCCLTO.

12.3.8 Byte Counter

The eUSCI_B module supports hardware counting of the bytes received or transmitted. The counter is automatically active and counts up for each byte seen on the bus in both master and slave mode.

The byte counter is incremented at the second bit position of each byte independently of the following ACK or NACK. A START or RESTART condition resets the counter value to zero. Address bytes do not increment the counter. The byte counter is also incremented at the second byte position, if an arbitration lost occurs during the first bit of data.

12.3.8.1 Byte Counter Interrupt

If UCASTPx = 01 or 10 the UCBCNTIFG is set when the byte counter threshold value UCBxTBCNT is reached in both master- and slave-mode. Writing zero to UCBxTBCNT does not generate an interrupt.

Because the UCBCNTIFG has a lower interrupt priority than the UCBTXIFG and UCBRXIFG, it is recommended to only use it for protocol control together with the DMA handling the received and transmitted bytes. Otherwise the application must have enough processor bandwidth to ensure that the UCBCNT interrupt routine is executed in time to generate for example a RESTART.

12.3.8.2 Automatic STOP Generation

When the eUSCI_B module is configured as a master, the byte counter can be used for automatic STOP generation by setting the UCASTPx = 10. Before starting the transmission using UCTXSTT, the byte counter threshold UCBxTBCNT must be set to the number of bytes that are to be transmitted or received. After the number of bytes that are configured in UCBxTBCNT have been transmitted, the eUSCI_B automatically generates a STOP condition.

UCBxTBCNT cannot be used if the user wants to transmit the slave address only without any data. In this case, it is recommended to set UCTXSTT and UCTXSTP at the same time.

12.3.9 Multiple Slave Addresses

The eUSCI_B module supports two different ways of implementing multiple slave addresses at the same time:

- Hardware support for up to 4 different slave addresses, each with its own interrupt flag and DMA trigger
- Software support for up to 2^{10} different slave addresses all sharing one interrupt

12.3.9.1 Multiple Slave Address Registers

The registers UCBxI2COA0, UCBxI2COA1, UCBxI2COA2, and UCBxI2COA3 contain four slave addresses. Up to four address registers are compared against a received 7- or 10-bit address. Each slave address must be activated by setting the UCAOEN bit in the corresponding UCBxI2COAx register. Register UCBxI2COA3 has the highest priority if the address received on the bus matches more than one of the slave address registers. The priority decreases with the index number of the address register, so that UCBxI2COA0 in combination with the address mask has the lowest priority.

When one of the slave registers matches the 7- or 10-bit address seen on the bus, the address is acknowledged. In the following the corresponding receive- or transmit-interrupt flag (UCTXIFGx or UCRXIFGx) to the received address is updated. The state change interrupt flags are independent of the address comparison result. They are updated according to the bus condition.

12.3.9.2 Address Mask Register

The Address Mask Register can be used when the eUSCI_B is configured in slave or in multiple-master mode. To activate this feature, at least one bit of the address mask in register UCBxADDMASK must be cleared.

If the received address matches the own address in UCBxI2COA0 on all bit positions not masked by UCBxADDMASK the eUSCI_B considers the seen address as its own address and sends an acknowledge. The user has the choice to either automatically acknowledge the address seen on the bus or to evaluate this address and send the acknowledge in software using UCTXACK. The selection between these options is done using the UCSWACK bit. If the software is used for generation of the ACK of the slave address, it is recommended to use the UCSTTIFG. The received address can be found in the UCBxADDRX register.

A slave address seen on the bus is automatically acknowledged by the eUSCI_B module, if it matches any of the slave addresses defined in UCBxI2COA1 to UCBxI2COA3.

NOTE: UCSWACK and slave-transmitter

If the user selects manual acknowledge of slave addresses, the TXIFG is set if the slave is addressed as a transmitter. If the user decides not to acknowledge the address, the TXIFG0 also must be reset.

12.3.10 Using the eUSCI_B Module in I²C Mode With Low-Power Modes

The eUSCI_B module provides automatic clock activation for use with low-power modes. When the eUSCI_B clock source is inactive because the device is in a low-power mode, the eUSCI_B module automatically activates it when needed, regardless of the control-bit settings for the clock source. The clock remains active until the eUSCI_B module returns to its idle condition. After the eUSCI_B module returns to the idle condition, control of the clock source reverts to the settings of its control bits.

In I²C slave mode, no internal clock source is required because the clock is provided by the external master. It is possible to operate the eUSCI_B in I²C slave mode while the device is in LPM4 and all internal clock sources are disabled. The receive or transmit interrupts can wake up the CPU from any low-power mode.

12.3.11 eUSCI_B Interrupts in I²C Mode

The eUSCI_B has only one interrupt vector that is shared for transmission, reception, and the state change.

Each interrupt flag has its own interrupt enable bit. When an interrupt is enabled and the GIE bit is set, the interrupt flag generates an interrupt request. DMA transfers are controlled by the UCTXIFGx and UCRXIFGx flags on devices with a DMA controller. It is possible to react on each slave address with an individual DMA channel.

All interrupt flags are not cleared automatically, but they need to be cleared together by user interactions (for example, reading the UCRXBUF clears UCRXIFGx). If the user wants to use an interrupt flag he needs to ensure that the flag has the correct state before the corresponding interrupt is enabled.

12.3.11.1 I²C Transmit Interrupt Operation

The UCTXIFG0 interrupt flag is set whenever the transmitter is able to accept a new byte. When operating as a slave with multiple slave addresses, the UCTXIFGx flags are set corresponding to which address was received before. If, for example, the slave address specified in register UCBxI2COA3 did match the address seen on the bus, the UCTXIFG3 indicates that the UCBxTXBUF is ready to accept a new byte.

When operating in master mode with automatic STOP generation (UCASTPx = 10), the UCTXIFG0 is set as many times as defined in UCBxTBCNT.

An interrupt request is generated if UCTXIEx and GIE are also set. UCTXIFGx is automatically reset if a write to UCBxTXBUF occurs or if the UCALIFG is cleared. UCTXIFGx is set when:

- Master mode: UCTXSTT was set by the user
- Slave mode: own address was received(UCETXINT = 0) or START was received (UCETXINT = 1)

UCTXIEx is reset after a PUC or when UCSWRST = 1.

12.3.11.2 Early I²C Transmit Interrupt

Setting the UCETXINT causes UCTXIFG0 to be sent out automatically when a START condition is sent and the eUSCI_B is configured as slave. In this case, it is not allowed to enable the other slave addresses UCBxI2COA1-UCBxI2COA3. This allows the software more time to handle the UCTXIFG0 compared to the normal situation, when UCTXIFG0 is sent out after the slave address match was detected. Situations where the UCTXIFG0 was set and afterward no slave address match occurred need to be handled in software. The use of the byte counter is recommended to handle this.

12.3.11.3 I²C Receive Interrupt Operation

The UCRXIFG0 interrupt flag is set when a character is received and loaded into UCBxRXBUF. When operating as a slave with multiple slave addresses, the UCRXIFGx flag is set corresponding to which address was received before.

An interrupt request is generated if UCRXIEx and GIE are also set. UCRXIFGx and UCRXIEx are reset after a PUC signal or when UCSWRST = 1. UCRXIFGx is automatically reset when UCxRXBUF is read.

12.3.11.4 I²C State Change Interrupt Operation

Table 12-2 describes the I²C state change interrupt flags.

Table 12-2. I²C State Change Interrupt Flags

Interrupt Flag	Interrupt Condition
UCALIFG	Arbitration-lost. Arbitration can be lost when two or more transmitters start a transmission simultaneously, or when the eUSCI_B operates as master but is addressed as a slave by another master in the system. The UCALIFG flag is set when arbitration is lost. When UCALIFG is set, the UCMST bit is cleared and the I²C controller becomes a slave.
UCNACKIFG	Not-acknowledge interrupt. This flag is set when an acknowledge is expected but is not received. UCNACKIFG is used in master mode only.
UCCLTOIFG	Clock low timeout. This interrupt flag is set, if the clock is held low longer than defined by the UCCLTO bits.
UCBIT9IFG	This interrupt flag is generated each time the eUSCI_B is transferring 9th clock cycle of a byte of data. This gives the user the possibility to follow the I²C communication in software if wanted. The UCBIT9IFG is not set for address information.
UCBCNTIFG	Byte counter interrupt. This flag is set when the byte counter value reaches the value defined in UCBxTBCNT and UCASTPx = 01 or 10. This bit allows to organize following communications, especially if a RESTART will be issued.
UCSTTIFG	START condition detected interrupt. This flag is set when the I²C module detects a START condition together with its own address(1). UCSTTIFG is used in slave mode only.
UCSTPIFG	STOP condition detected interrupt. This flag is set when the I²C module detects a STOP condition on the bus. UCSTPIFG is used in slave and master mode.

(1) The address evaluation includes the address mask register if it is used.

12.3.11.5 UCBxIV, Interrupt Vector Generator

The eUSCI_B interrupt flags are prioritized and combined to source a single interrupt vector. The interrupt vector register UCBxIV is used to determine which flag requested an interrupt. The highest-priority enabled interrupt generates a number in the UCBxIV register that can be evaluated or added to the PC to automatically enter the appropriate software routine. Disabled interrupts do not affect the UCBxIV value.

Read access of the UCBxIV register automatically resets the highest-pending interrupt flag. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt.

Write access of the UCBxIV register clears all pending Interrupt conditions and flags.

Example 12-3 shows the recommended use of UCBxIV. The UCBxIV value is added to the PC to automatically jump to the appropriate routine. The example is given for eUSCI0_B.

Example 12-3. UCBxIV Software Example

```
#pragma vector = USCI_B0_VECTOR __interrupt void USCI_B0_ISR(void) {
    switch(__even_in_range(UCB0IV,0x1e))    {
        case 0x00:     // Vector 0: No interrupts break;
        case 0x02: ... // Vector 2: ALIFG break;
        case 0x04: ... // Vector 4: NACKIFG break;
        case 0x06: ... // Vector 6: STTIFG break;
        case 0x08: ... // Vector 8: STPIFG break;
        case 0x0a: ... // Vector 10: RXIFG3 break;
        case 0x0c: ... // Vector 14: TXIFG3 break;
        case 0x0e: ... // Vector 16: RXIFG2 break;
        case 0x10: ... // Vector 18: TXIFG2 break;
        case 0x12: ... // Vector 20: RXIFG1 break;
        case 0x14: ... // Vector 22: TXIFG1 break;
        case 0x16: ... // Vector 24: RXIFG0 break;
        case 0x18: ... // Vector 26: TXIFG0 break;
        case 0x1a: ... // Vector 28: BCNTIFG break;
        case 0x1c: ... // Vector 30: clock low timeout break;
        case 0x1e: ... // Vector 32: 9th bit break;
        default: break;
    }
}
```

12.4 eUSCI_B I2C Registers

The eUSCI_B registers applicable in I2C mode and their addresses are listed in Table 12-3.

Table 12-3. eUSCI_B Registers

Address	Acronym	Register Name	Type	Access	Reset	Section
01C0h	UCBxCTLW0	eUSCI_Bx Control Word 0	Read/write	Word	01C1h	Section 12.4.1
01C0h	UCBxCTL1	eUSCI_Bx Control 1	Read/write	Byte	C1h	
01C1h	UCBxCTL0	eUSCI_Bx Control 0	Read/write	Byte	01h	
01C2h	UCBxCTLW1	eUSCI_Bx Control Word 1	Read/write	Word	0000h	Section 12.4.2
01C6h	UCBxBRW	eUSCI_Bx Bit Rate Control Word	Read/write	Word	0000h	Section 12.4.3
01C6h	UCBxBR0	eUSCI_Bx Bit Rate Control 0	Read/write	Byte	00h	
01C7h	UCBxBR1	eUSCI_Bx Bit Rate Control 1	Read/write	Byte	00h	
01C8h	UCBxSTATW	eUSCI_Bx Status Word	Read	Word	0000h	Section 12.4.4
01C8h	UCBxSTAT	eUSCI_Bx Status	Read	Byte	00h	
01C9h	UCBxBCNT	eUSCI_Bx Byte Counter Register	Read	Byte	00h	
01CAh	UCBxTBCNT	eUSCI_Bx Byte Counter Threshold Register	Read/Write	Word	00h	Section 12.4.5
01CCh	UCBxRXBUF	eUSCI_Bx Receive Buffer	Read/write	Word	00h	Section 12.4.6
01CEh	UCBxTXBUF	eUSCI_Bx Transmit Buffer	Read/write	Word	00h	Section 12.4.7
01D4h	UCBxI2COA0	eUSCI_Bx I2C Own Address 0	Read/write	Word	0000h	Section 12.4.8
01D6h	UCBxI2COA1	eUSCI_Bx I2C Own Address 1	Read/write	Word	0000h	Section 12.4.9
01D8h	UCBxI2COA2	eUSCI_Bx I2C Own Address 2	Read/write	Word	0000h	Section 12.4.10
01DAh	UCBxI2COA3	eUSCI_Bx I2C Own Address 3	Read/write	Word	0000h	Section 12.4.11
01DCh	UCBxADDRX	eUSCI_Bx Received Address Register	Read	Word		Section 12.4.12
01DEh	UCBxADDMASK	eUSCI_Bx Address Mask Register	Read/write	Word	03FFh	Section 12.4.13
01E0h	UCBxI2CSA	eUSCI_Bx I2C Slave Address	Read/write	Word	0000h	Section 12.4.14
01EAh	UCBxIE	eUSCI_Bx Interrupt Enable	Read/write	Word	0000h	Section 12.4.15
01ECh	UCBxIFG	eUSCI_Bx Interrupt Flag	Read/write	Word	2A02h	Section 12.4.16
01EEh	UCBxIV	eUSCI_Bx Interrupt Vector	Read	Word	0000h	Section 12.4.17

12.4.1 UCBxCTLW0 Register (address = 01C0h) [reset = 01C1h]

eUSCI_Bx Control Word Register 0

UCBxCTLW0 is shown in Figure 12-17 and described in Table 12-4.

Figure 12-17. UCBxCTLW0 Register

15	14	13	12	11	10	9	8
UCA10	UCSLA10	UCMM	Reserved	UCMST	UCMODEx		UCSYNC
rw-0	rw-0	rw-0	r0	rw-0	rw-0	rw-0	r1
7	6	5	4	3	2	1	0
UCSSELx		UCTXACK	UCTR	UCTXNACK	UCTXSTP	UCTXSTT	UCSWRST
rw-1	rw-1	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

Can be modified only when eUSCI_B is in reset state (UCSWRST = 1)

Table 12-4. UCBxCTLW0 Register Description

Bit	Field	Type	Reset	Description
15	UCA10	RW	0h	Own addressing mode select. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1). 0b = Own address is a 7-bit address. 1b = Own address is a 10-bit address.
14	UCSLA10	RW	0h	Slave addressing mode select 0b = Address slave with 7-bit address 1b = Address slave with 10-bit address
13	UCMM	RW	0h	Multi-master environment select. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1). 0b = Single master environment. There is no other master in the system. The address compare unit is disabled. 1b = Multi-master environment
12	Reserved	R	0h	Reserved
11	UCMST	RW	0h	Master mode select. When a master loses arbitration in a multi-master environment (UCMM = 1), the UCMST bit is automatically cleared and the module acts as slave. 0b = Slave mode 1b = Master mode
10-9	UCMODEx	RW	0h	eUSCI_B mode. The UCMODEx bits select the synchronous mode when UCSYNC = 1. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1). 00b = 3-pin SPI 01b = 4-pin SPI (master/slave enabled if STE = 1) 10b = 4-pin SPI (master/slave enabled if STE = 0) 11b = I2C mode
8	UCSYNC	RW	1h	Synchronous mode enable. For eUSCI_B always read and write as 1.
7-6	UCSSELx	RW	3h	eUSCI_B clock source select. These bits select the BRCLK source clock. These bits are ignored in slave mode. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1). 00b = UCLKI 01b = ACLK 10b = SMCLK 11b = SMCLK

Table 12-4. UCBxCTLW0 Register Description (continued)

Bit	Field	Type	Reset	Description
5	UCTXACK	RW	0h	Transmit ACK condition in slave mode with enabled address mask register. After the UCSTTIFG has been set, the user needs to set or reset the UCTXACK flag to continue with the I2C protocol. The clock is stretched until the UCBxCTL1 register has been written. This bit is cleared automatically after the ACK has been send. 0b = Do not acknowledge the slave address 1b = Acknowledge the slave address
4	UCTR	RW	0h	Transmitter/receiver 0b = Receiver 1b = Transmitter
3	UCTXNACK	RW	0h	Transmit a NACK. UCTXNACK is automatically cleared after a NACK is transmitted. Only for slave receiver mode. 0b = Acknowledge normally 1b = Generate NACK
2	UCTXSTP	RW	0h	Transmit STOP condition in master mode. Ignored in slave mode. In master receiver mode, the STOP condition is preceded by a NACK. UCTXSTP is automatically cleared after STOP is generated. This bit is a don't care, if automatic UCASTPx is different from 01 or 10. 0b = No STOP generated 1b = Generate STOP
1	UCTXSTT	RW	0h	Transmit START condition in master mode. Ignored in slave mode. In master receiver mode, a repeated START condition is preceded by a NACK. UCTXSTT is automatically cleared after START condition and address information is transmitted. Ignored in slave mode. 0b = Do not generate START condition 1b = Generate START condition
0	UCSWRST	RW	1h	Software reset enable. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1). 0b = Disabled. eUSCI_B released for operation. 1b = Enabled. eUSCI_B logic held in reset state.

12.4.2 UCBxCTLW1 Register (address = 01C2h) [reset = 0000h]

eUSCI_Bx Control Word Register 1

UCBxCTLW1 is shown in Figure 12-18 and described in Table 12-5.

Figure 12-18. UCBxCTLW1 Register

15	14	13	12	11	10	9	8
Reserved							UCETXINT
r0	r0	r0	r0	r0	r0	r0	rw-0
7	6	5	4	3	2	1	0
UCCLTO		UCSTPNACK	UCSWACK	UCASTPx		UCGLITx	
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when eUSCI_B is in reset state (UCSWRST = 1).

Table 12-5. UCBxCTLW1 Register Description

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0h	Reserved
8	UCETXINT	RW	0h	Early UCTXIFG0. Only in slave mode. When this bit is set, the slave addresses defined in UCxI2COA1 to UCxI2COA3 must be disabled. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1). 0b = UCTXIFGx is set after an address match with UCxI2COAx and the direction bit indicating slave transmit 1b = UCTXIFG0 is set for each START condition.
7-6	UCCLTO	RW	0h	Clock low timeout select. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1). 00b = Disable clock low timeout counter 01b = 135 000 MODCLK cycles (≠28 ms) 10b = 150 000 MODCLK cycles (≠31 ms) 11b = 165 000 MODCLK cycles (≠34 ms)
5	UCSTPNACK	RW	0h	The UCSTPNACK bit allows to make the eUSCI_B master acknowledge the last byte in master receiver mode as well. This is not conform to the I2C specification and should only be used for slaves, which automatically release the SDA after a fixed packet length. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1). 0b = Send a non-acknowledge before the STOP condition as a master receiver (conform to I2C standard) 1b = All bytes are acknowledged by the eUSCI_B when configured as master receiver.
4	UCSWACK	RW	0h	Using this bit it is possible to select, whether the eUSCI_B module triggers the sending of the ACK of the address or if it is controlled by software. 0b = The address acknowledge of the slave is controlled by the eUSCI_B module 1b = The user needs to trigger the sending of the address ACK by issuing UCTXACK.
3-2	UCASTPx	RW	0h	Automatic STOP condition generation. In slave mode only UCBCNTIFG is available. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1). 00b = No automatic STOP generation. The STOP condition is generated after the user sets the UCTXSTP bit. The value in UCBxTBCNT is a don't care. 01b = UCBCNTIFG is set with the byte counter reaches the threshold defined in UCBxTBCNT. 10b = A STOP condition is generated automatically after the byte counter value reached UCBxTBCNT. UCBCNTIFG is set with the byte counter reaching the threshold. 11b = Reserved

Table 12-5. UCBxCTLW1 Register Description (continued)

Bit	Field	Type	Reset	Description
1-0	UCGLITx	RW	0h	Deglitch time 00b = 50 ns 01b = 25 ns 10b = 12.5 ns 11b = 6.25 ns

12.4.3 UCBxBRW Register (address = 01C6h) [reset = 0000h]

eUSCI_Bx Bit Rate Control Word Register

UCBxBRW is shown in Figure 12-19 and described in Table 12-6.

Figure 12-19. UCBxBRW Register

15	14	13	12	11	10	9	8
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw

Can be modified only when eUSCI_B is in reset state (UCSWRST = 1).

Table 12-6. UCBxBRW Register Description

Bit	Field	Type	Reset	Description
15-0	UCBRx	RW	0h	Bit clock prescaler. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1).

12.4.4 UCBxSTATW Register (address = 01C8h) [reset = 0000h]

eUSCI_Bx Status Word Register

UCBxSTATW is shown in Figure 12-20 and described in Table 12-7.

Figure 12-20. UCBxSTATW Register

15	14	13	12	11	10	9	8
UCBCNTx							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved	UCSCLLOW	UCGC	UCBBUSY	Reserved			
r0	r-0	r-0	r-0	r-0	r0	r0	r0

Table 12-7. UCBxSTATW Register Description

Bit	Field	Type	Reset	Description
15-8	UCBCNTx	R	0h	Hardware byte counter value. Reading this register returns the number of bytes received or transmitted on the I2C-Bus since the last START or RESTART. There is no synchronization of this register done. When reading UCBxBCNT during the first bit position, a faulty readback can occur.
7	Reserved	R	0h	Reserved
6	UCSCLLOW	R	0h	SCL low 0b = SCL is not held low. 1b = SCL is held low.
5	UCGC	R	0h	General call address received. UCGC is automatically cleared when a START condition is received. 0b = No general call address received 1b = General call address received
4	UCBBUSY	R	0h	Bus busy 0b = Bus inactive 1b = Bus busy
3-0	Reserved	R	0h	Reserved

12.4.5 UCBxTBCNT Register (address = 01CAh) [reset = 0000h]

eUSCI_Bx Byte Counter Threshold Register

UCBxTBCNT is shown in Figure 12-21 and described in Table 12-8.

Figure 12-21. UCBxTBCNT Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCTBCNTx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when eUSCI_B is in reset state (UCSWRST = 1).

Table 12-8. UCBxTBCNT Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCTBCNTx	RW	0h	The byte counter threshold value is used to set the number of I2C data bytes after which the automatic STOP or the UCSTPIFG should occur. This value is evaluated only if UCASTPx is different from 00. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1).

12.4.6 UCBxRXBUF Register (address = 01CCh) [reset = 0000h]

eUSCI_Bx Receive Buffer Register

UCBxRXBUF is shown in Figure 12-22 and described in Table 12-9.

Figure 12-22. UCBxRXBUF Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCRXBUFx							
r	r	r	r	r	r	r	r

Table 12-9. UCBxRXBUF Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCRXBUFx	R	0h	The receive-data buffer is user accessible and contains the last received character from the receive shift register. Reading UCBxRXBUF resets the UCRXIFGx flags.

12.4.7 UCBxTXBUF Register (address = 01CEh) [reset = 0000h]

eUSCI_Bx Transmit Buffer Register

UCBxTXBUF is shown in Figure 12-23 and described in Table 12-10.

Figure 12-23. UCBxTXBUF Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCTXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

Table 12-10. UCBxTXBUF Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCTXBUFx	RW	0h	The transmit data buffer is user accessible and holds the data waiting to be moved into the transmit shift register and transmitted. Writing to the transmit data buffer clears the UCTXIFGx flags.

12.4.8 UCBxI2COA0 Register (address = 01D4h) [reset = 0000h]

eUSCI_Bx I2C Own Address 0 Register

UCBxI2COA0 is shown in Figure 12-24 and described in Table 12-11.

Figure 12-24. UCBxI2COA0 Register

15	14	13	12	11	10	9	8
UCGCEN	Reserved				UCOAEN	I2COA0	
rw-0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
I2COA0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when eUSCI_B is in reset state (UCSWRST = 1).

Table 12-11. UCBxI2COA0 Register Description

Bit	Field	Type	Reset	Description
15	UCGCEN	RW	0h	General call response enable. This bit is only available in UCBxI2COA0. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1). 0b = Do not respond to a general call 1b = Respond to a general call
14-11	Reserved	R	0h	Reserved
10	UCOAEN	RW	0h	Own Address enable register. With this register it can be selected if the I2C slave-address related to this register UCBxI2COA0 is evaluated or not. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1). 0b = The slave address defined in I2COA0 is disabled 1b = The slave address defined in I2COA0 is enabled
9-0	I2COAx	RW	0h	I2C own address. The I2COA0 bits contain the local address of the eUSCIx_B I2C controller. The address is right justified. In 7-bit addressing mode, bit 6 is the MSB and bits 9-7 are ignored. In 10-bit addressing mode, bit 9 is the MSB. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1).

12.4.9 UCBxI2COA1 Register (address = 01D6h) [reset = 0000h]

eUSCI_Bx I2C Own Address 1 Register

UCBxI2COA1 is shown in Figure 12-25 and described in Table 12-12.

Figure 12-25. UCBxI2COA1 Register

15	14	13	12	11	10	9	8
Reserved					UCOAEN	I2COA1	
rw-0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
I2COA1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when eUSCI_B is in reset state (UCSWRST = 1)

Table 12-12. UCBxI2COA1 Register Description

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved
10	UCOAEN	RW	0h	Own Address enable register. With this register it can be selected if the I2C slave-address related to this register UCBxI2COA1 is evaluated or not. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1). 0b = The slave address defined in I2COA1 is disabled 1b = The slave address defined in I2COA1 is enabled
9-0	I2COA1	RW	0h	I2C own address. The I2COAx bits contain the local address of the eUSCIx_B I2C controller. The address is right justified. In 7-bit addressing mode, bit 6 is the MSB and bits 9-7 are ignored. In 10-bit addressing mode, bit 9 is the MSB. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1).

12.4.10 UCBxI2COA2 Register (address = 01D8h) [reset = 0000h]

eUSCI_Bx I2C Own Address 2 Register

UCBxI2COA2 is shown in Figure 12-26 and described in Table 12-13.

Figure 12-26. UCBxI2COA2 Register

15	14	13	12	11	10	9	8
Reserved					UCOAEN	I2COA2	
rw-0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
I2COA2							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when eUSCI_B is in reset state (UCSWRST = 1)

Table 12-13. UCBxI2COA2 Register Description

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved
10	UCOAEN	RW	0h	Own Address enable register. With this register it can be selected if the I2C slave-address related to this register UCBxI2COA2 is evaluated or not. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1). 0b = The slave address defined in I2COA2 is disabled 1b = The slave address defined in I2COA2 is enabled
9-0	I2COA2	RW	0h	I2C own address. The I2COAx bits contain the local address of the eUSCIx_B I2C controller. The address is right justified. In 7-bit addressing mode, bit 6 is the MSB and bits 9-7 are ignored. In 10-bit addressing mode, bit 9 is the MSB. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1).

12.4.11 UCBxI2COA3 Register (address = 01DAh) [reset = 0000h]

eUSCI_Bx I2C Own Address 3 Register

UCBxI2COA3 is shown in Figure 12-27 and described in Table 12-14.

Figure 12-27. UCBxI2COA3 Register

15	14	13	12	11	10	9	8
Reserved					UCOAEN	I2COA3	
rw-0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
I2COA3							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when eUSCI_B is in reset state (UCSWRST = 1)

Table 12-14. UCBxI2COA3 Register Description

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved
10	UCOAEN	RW	0h	Own Address enable register. With this register it can be selected if the I2C slave-address related to this register UCBxI2COA3 is evaluated or not. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1). 0b = The slave address defined in I2COA3 is disabled 1b = The slave address defined in I2COA3 is enabled
9-0	I2COA3	RW	0h	I2C own address. The I2COA3 bits contain the local address of the eUSCIx_B I2C controller. The address is right justified. In 7-bit addressing mode, bit 6 is the MSB and bits 9-7 are ignored. In 10-bit addressing mode, bit 9 is the MSB. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1).

12.4.12 UCBxADDRX Register (address = 01DCh) [reset = 0000h]

eUSCI_Bx I2C Received Address Register

UCBxADDRX is shown in Figure 12-28 and described in Table 12-15.

Figure 12-28. UCBxADDRX Register

15	14	13	12	11	10	9	8
Reserved						ADDRXx	
r-0	r0	r0	r0	r0	r0	r-0	r-0
7	6	5	4	3	2	1	0
ADDRXx							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

Table 12-15. UCBxADDRX Register Description

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	ADDRXx	R	0h	Received Address Register. This register contains the last received slave address on the bus. Using this register and the address mask register it is possible to react on more than one slave address using one eUSCI_B module.

12.4.13 UCBxADDMASK Register (address = 01DEh) [reset = 03FFh]

eUSCI_Bx I2C Address Mask Register

UCBxADDMASK is shown in Figure 12-29 and described in Table 12-16.

Figure 12-29. UCBxADDMASK Register

15	14	13	12	11	10	9	8
Reserved						ADDMASKx	
r-0	r0	r0	r0	r0	r0	rw-1	rw-1
7	6	5	4	3	2	1	0
ADDMASKx							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

Can be modified only when eUSCI_B is in reset state (UCSWRST = 1).

Table 12-16. UCBxADDMASK Register Description

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	ADDMASKx	RW	3FFh	Address Mask Register. By clearing the corresponding bit of the own address, this bit is a don't care when comparing the address on the bus to the own address. Using this method, it is possible to react on more than one slave address. When all bits of ADDMASKx are set, the address mask feature is deactivated. Can be modified only when eUSCI_B is in reset state (UCSWRST = 1).

12.4.14 UCBxI2CSA Register (address = 01E0h) [reset = 0000h]

eUSCI_Bx I2C Slave Address Register

UCBxI2CSA is shown in Figure 12-30 and described in Table 12-17.

Figure 12-30. UCBxI2CSA Register

15	14	13	12	11	10	9	8
Reserved						I2CSAx	
r-0	r0	r0	r0	r0	r0	rw-0	rw-0
7	6	5	4	3	2	1	0
I2CSAx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-17. UCBxI2CSA Register Description

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	I2CSAx	RW	0h	I2C slave address. The I2CSAx bits contain the slave address of the external device to be addressed by the eUSCIx_B module. It is only used in master mode. The address is right justified. In 7-bit slave addressing mode, bit 6 is the MSB and bits 9-7 are ignored. In 10-bit slave addressing mode, bit 9 is the MSB.

12.4.15 UCBxIE Register (address = 01EAh) [reset = 0000h]

eUSCI_Bx I2C Interrupt Enable Register

UCBxIE is shown in Figure 12-31 and described in Table 12-18.

Figure 12-31. UCBxIE Register

15	14	13	12	11	10	9	8
Reserved	UCBIT9IE	UCTXIE3	UCRXIE3	UCTXIE2	UCRXIE2	UCTXIE1	UCRXIE1
r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
UCCLTOIE	UCBCNTIE	UCNACKIE	UCALIE	UCSTPIE	UCSTTIE	UCTXIE0	UCRXIE0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-18. UCBxIE Register Description

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved
14	UCBIT9IE	RW	0h	Bit position 9 interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
13	UCTXIE3	RW	0h	Transmit interrupt enable 3 0b = Interrupt disabled 1b = Interrupt enabled
12	UCRXIE3	RW	0h	Receive interrupt enable 3 0b = Interrupt disabled 1b = Interrupt enabled
11	UCTXIE2	RW	0h	Transmit interrupt enable 2 0b = Interrupt disabled 1b = Interrupt enabled
10	UCRXIE2	RW	0h	Receive interrupt enable 2 0b = Interrupt disabled 1b = Interrupt enabled
9	UCTXIE1	RW	0h	Transmit interrupt enable 1 0b = Interrupt disabled 1b = Interrupt enabled
8	UCRXIE1	RW	0h	Receive interrupt enable 1 0b = Interrupt disabled 1b = Interrupt enabled
7	UCCLTOIE	RW	0h	Clock low timeout interrupt enable. 0b = Interrupt disabled 1b = Interrupt enabled
6	UCBCNTIE	RW	0h	Byte counter interrupt enable. 0b = Interrupt disabled 1b = Interrupt enabled
5	UCNACKIE	RW	0h	Not-acknowledge interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
4	UCALIE	RW	0h	Arbitration lost interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
3	UCSTPIE	RW	0h	STOP condition interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled

Table 12-18. UCBxIE Register Description (continued)

Bit	Field	Type	Reset	Description
2	UCSTTIE	RW	0h	START condition interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
1	UCTXIE0	RW	0h	Transmit interrupt enable 0 0b = Interrupt disabled 1b = Interrupt enabled
0	UCRXIE0	RW	0h	Receive interrupt enable 0 0b = Interrupt disabled 1b = Interrupt enabled

12.4.16 UCBxIFG Register (address = 01ECh) [reset = 2A02h]

eUSCI_Bx I2C Interrupt Flag Register

UCBxIFG is shown in Figure 12-32 and described in Table 12-19.

Figure 12-32. UCBxIFG Register

15	14	13	12	11	10	9	8
Reserved	UCBIT9IFG	UCTXIFG3	UCRXIFG3	UCTXIFG2	UCRXIFG2	UCTXIFG1	UCRXIFG1
r0	rw-0	rw-1	rw-0	rw-1	rw-0	rw-1	rw-0
7	6	5	4	3	2	1	0
UCCLTOIFG	UCBCNTIFG	UCNACKIFG	UCALIFG	UCSTPIFG	UCSTTIFG	UCTXIFG0	UCRXIFG0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1	rw-0

Table 12-19. UCBxIFG Register Description

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved
14	UCBIT9IFG	RW	0h	Bit position 9 interrupt flag 0b = No interrupt pending 1b = Interrupt pending
13	UCTXIFG3	RW	1h	eUSCI_B transmit interrupt flag 3. UCTXIFG3 is set when UCBxTXBUF is empty in slave mode, if the slave address defined in UCBxI2COA3 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
12	UCRXIFG3	RW	0h	Receive interrupt flag 2. UCRXIFG2 is set when UCBxRXBUF has received a complete byte in slave mode and if the slave address defined in UCBxI2COA2 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
11	UCTXIFG2	RW	0h	eUSCI_B transmit interrupt flag 2. UCTXIFG2 is set when UCBxTXBUF is empty in slave mode, if the slave address defined in UCBxI2COA2 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
10	UCRXIFG2	RW	0h	Receive interrupt flag 2. UCRXIFG2 is set when UCBxRXBUF has received a complete byte in slave mode and if the slave address defined in UCBxI2COA2 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
9	UCTXIFG1	RW	1h	eUSCI_B transmit interrupt flag 1. UCTXIFG1 is set when UCBxTXBUF is empty in slave mode, if the slave address defined in UCBxI2COA1 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
8	UCRXIFG1	RW	0h	Receive interrupt flag 1. UCRXIFG1 is set when UCBxRXBUF has received a complete byte in slave mode and if the slave address defined in UCBxI2COA1 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
7	UCCLTOIFG	RW	0h	Clock low timeout interrupt flag 0b = No interrupt pending 1b = Interrupt pending
6	UCBCNTIFG	RW	0h	Byte counter interrupt flag. When using this interrupt the user needs to ensure enough processing bandwidth (see the Byte Counter Interrupt section). 0b = No interrupt pending 1b = Interrupt pending

Table 12-19. UCBxIFG Register Description (continued)

Bit	Field	Type	Reset	Description
5	UCNACKIFG	RW	0h	Not-acknowledge received interrupt flag. This flag only is updated when operating in master mode. 0b = No interrupt pending 1b = Interrupt pending
4	UCALIFG	RW	0h	Arbitration lost interrupt flag 0b = No interrupt pending 1b = Interrupt pending
3	UCSTPIFG	RW	0h	STOP condition interrupt flag 0b = No interrupt pending 1b = Interrupt pending
2	UCSTTIFG	RW	0h	START condition interrupt flag 0b = No interrupt pending 1b = Interrupt pending
1	UCTXIFG0	RW	0h	eUSCI_B transmit interrupt flag 0. UCTXIFG0 is set when UCBxTXBUF is empty in master mode or in slave mode, if the slave address defined in UCBxI2COA0 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
0	UCRXIFG0	RW	0h	eUSCI_B receive interrupt flag 0. UCRXIFG0 is set when UCBxRXBUF has received a complete character in master mode or in slave mode, if the slave address defined in UCBxI2COA0 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending

12.4.17 UCBxIV Register (address = 01EEh) [reset = 0000h]

eUSCI_Bx Interrupt Vector Register

UCBxIV is shown in Figure 12-33 and described in Table 12-20.

Figure 12-33. UCBxIV Register

15	14	13	12	11	10	9	8
UCIVx							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCIVx							
r0	r0	r0	r0	r-0	r-0	r-0	r0

Table 12-20. UCBxIV Register Description

Bit	Field	Type	Reset	Description
15-0	UCIVx	R	0h	eUSCI_B interrupt vector value. It generates an value that can be used as address offset for fast interrupt service routine handling. Writing to this register clears all pending interrupt flags. 00h = No interrupt pending 02h = Interrupt Source: Arbitration lost; Interrupt Flag: UCALIFG; Interrupt Priority: Highest 04h = Interrupt Source: Not acknowledgment; Interrupt Flag: UCNACKIFG 06h = Interrupt Source: Start condition received; Interrupt Flag: UCSTTIFG 08h = Interrupt Source: Stop condition received; Interrupt Flag: UCSTPIFG 0Ah = Interrupt Source: Slave 3 Data received; Interrupt Flag: UCRXIFG3 0Ch = Interrupt Source: Slave 3 Transmit buffer empty; Interrupt Flag: UCTXIFG3 0Eh = Interrupt Source: Slave 2 Data received; Interrupt Flag: UCRXIFG2 10h = Interrupt Source: Slave 2 Transmit buffer empty; Interrupt Flag: UCTXIFG2 12h = Interrupt Source: Slave 1 Data received; Interrupt Flag: UCRXIFG1 14h = Interrupt Source: Slave 1 Transmit buffer empty; Interrupt Flag: UCTXIFG1 16h = Interrupt Source: Data received; Interrupt Flag: UCRXIFG0 18h = Interrupt Source: Transmit buffer empty; Interrupt Flag: UCTXIFG0 1Ah = Interrupt Source: Byte counter zero; Interrupt Flag: UCBCNTIFG 1Ch = Interrupt Source: Clock low timeout; Interrupt Flag: UCCLTOIFG 1Eh = Interrupt Source: Nineth bit position; Interrupt Flag: UCBIT9IFG; Priority: Lowest

SD24

The SD24 is a multi-input multi-converter sigma-delta analog-to-digital conversion module. This chapter describes the operation of the SD24 module.

Topic **Page**

13.1 SD24 Introduction

The SD24 module consists of up to seven independent sigma-delta analog-to-digital converters, referred to as channels. The converters are based on second-order oversampling sigma-delta modulators and digital decimation filters. The decimation filters are comb-type filters with selectable oversampling ratios of up to 256. Additional filtering can be done in software.

Features of the SD24 include:

- Second-order sigma-delta architecture
- Up to seven independent simultaneously sampling ADCs. (The number of channels is device dependent. See the device-specific data sheet.)
- Fixed 1.024-MHz modulator input frequency
- Software selectable internal or external voltage reference
- Software selectable temperature sensor accessible by all channels

Figure 13-1 shows the block diagram of the SD24 module.

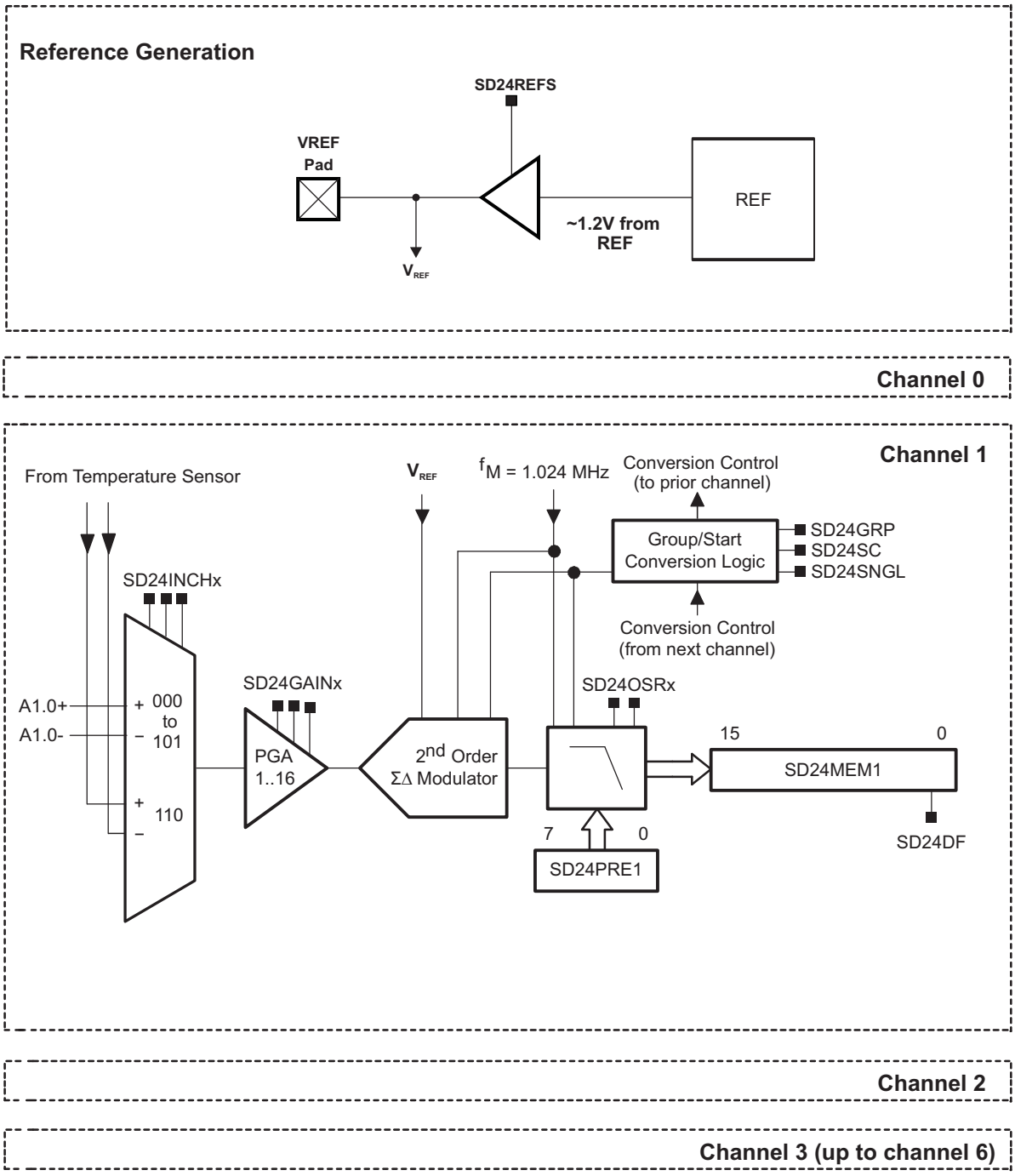


Figure 13-1. SD24 Block Diagram

13.2 SD24 Operation

The SD24 module is configured with user software. The setup and operation of the SD24 is described in the following sections.

13.2.1 Principle of Operation

A sigma-delta analog-to-digital converter basically consists of two parts: the analog part (the modulator) and the digital part (a decimation filter). The modulator of the SD24 provides a bit stream of zeros and ones to the digital decimation filter. The digital filter averages the bitstream from the modulator over a given number of bits (specified by the oversampling rate) and provides samples at a reduced rate for further processing to the CPU.

Averaging can be used to increase the signal-to-noise performance of a conversion [see Figure 13-2 a) and b)]. With a conventional ADC, each factor-of-4 oversampling can improve the SNR by approximately 6 dB or 1 bit. To achieve a 16-bit resolution out of a simple 1-bit ADC would require an impractical oversampling rate of 4^{15} = 1 073 741 824. To overcome this limitation, the sigma-delta modulator implements a technique called noise-shaping—a feedback loop and integrators push the quantization noise to higher frequencies, and thus much lower oversampling rates can achieve high resolutions [see Figure 13-2 c)].

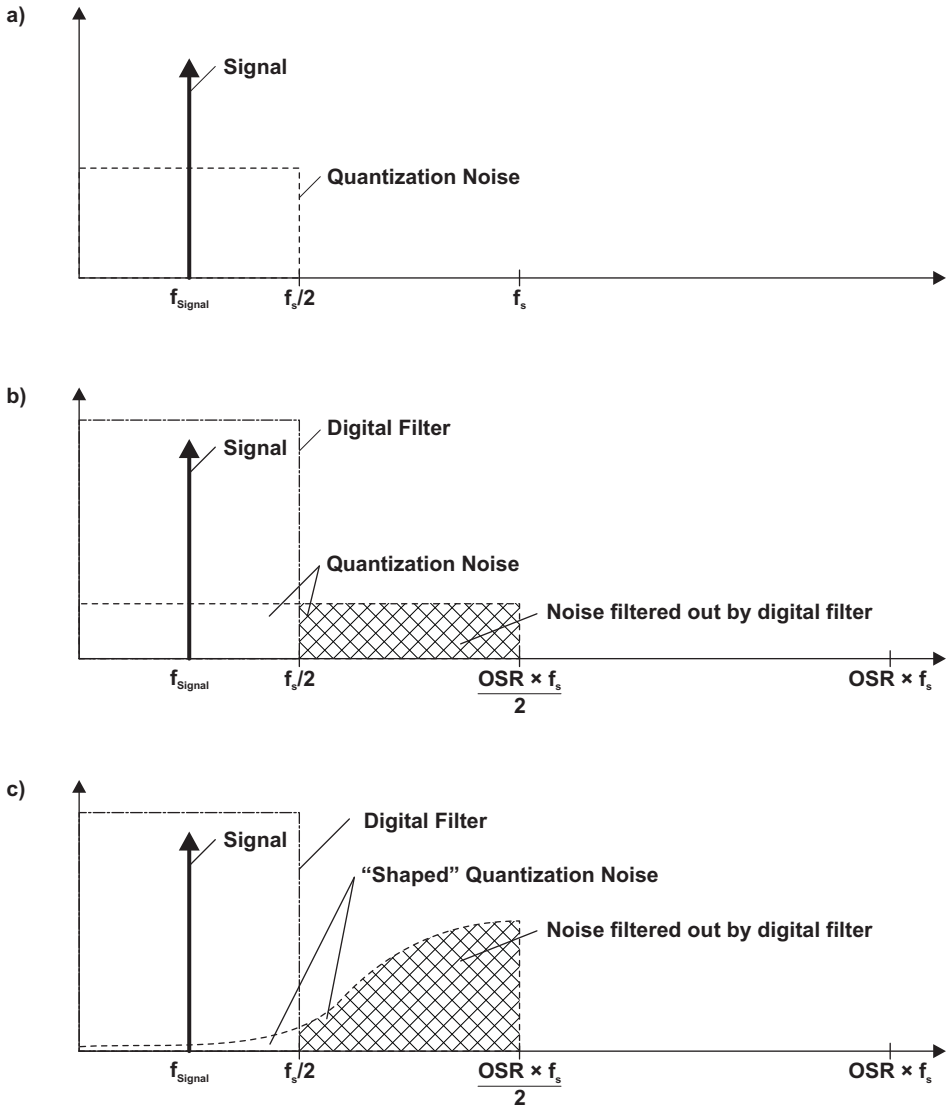


Figure 13-2. Sigma-Delta Principle

13.2.2 ADC Core

The analog-to-digital conversion is performed by a 1-bit second-order sigma-delta modulator. A single-bit comparator within the modulator quantizes the input signal with the modulator frequency f_M. The resulting 1-bit data stream is averaged by the digital filter for the conversion result.

13.2.3 Voltage Reference

The SD24 module can use an external reference voltage with SD24REFS = 0 or an internal reference with SD24REFS = 1. It uses the shared reference (REF) module as its internal voltage reference source. REF is used for each SD24 channel when requested with SD24REFS = 1. When the internal reference is used, an external 100-nF capacitor connected from VREF to AVSS is recommended to reduce noise. See device-specific data sheet for parameters.

An external voltage reference must be applied to the VREF input when SD24REFS = 0.

NOTE: When SD24 operates with internal reference (SD24REFS = 1) the VREF pin must not be loaded externally. Only the recommended capacitor value, C_{VREF} must be connected at VREF pin to AVSS.

13.2.4 Modulator Clock

The clock system generates a 1.024-MHz fixed frequency clock (f_M) to the SD24 module. This clock is supplied to all modulators so that all modulators convert synchronously. This is not a free running clock to the SD24 module—it runs only when an SD24 channel conversion is active.

13.2.5 Auto Power-Down

The SD24 is designed for low-power applications. When a SD24 channel is not actively converting, it is automatically disabled; it is automatically re-enabled when a conversion is started. When a channel is disabled, it consumes no current.

13.2.6 Analog Inputs

13.2.6.1 Analog Input Range and PGA

The full-scale input voltage range for each analog input pair depends on the gain setting of each channel. See the device-specific data sheet for full-scale input specifications.

13.2.6.2 Analog Input Setup

The analog input of each channel is configured using the SD24INCTLx register. These settings are independently configured for each SD24 channel.

The gain for each PGA is selected by the SD24GAINx bits. A total of five gain settings are available.

During conversion, any modification of the SD24INCTLx register becomes effective with the next decimation step of the digital filter. After these bits are modified, the next three conversions may be invalid due to the settling time of the digital filter. This can be handled automatically with the SD24INTDLY bit. When SD24INTDLY = 0b, conversion interrupt requests do not begin until the fourth conversion after a start condition.

An external RC anti-aliasing filter is recommended for the SD24 to prevent aliasing of the input signal. The cutoff frequency should be less than 10 kHz for a 1-MHz modulator clock and OSR = 256. The cutoff frequency may be set to a lower frequency for applications that have lower bandwidth requirements.

13.2.7 Digital Filter

The digital filter processes the 1-bit data stream from the modulator using a $SINC^3$ comb filter.

13.2.7.1 SINC³ Filter

Figure 13-3 show the structure of a SINC³ filter.

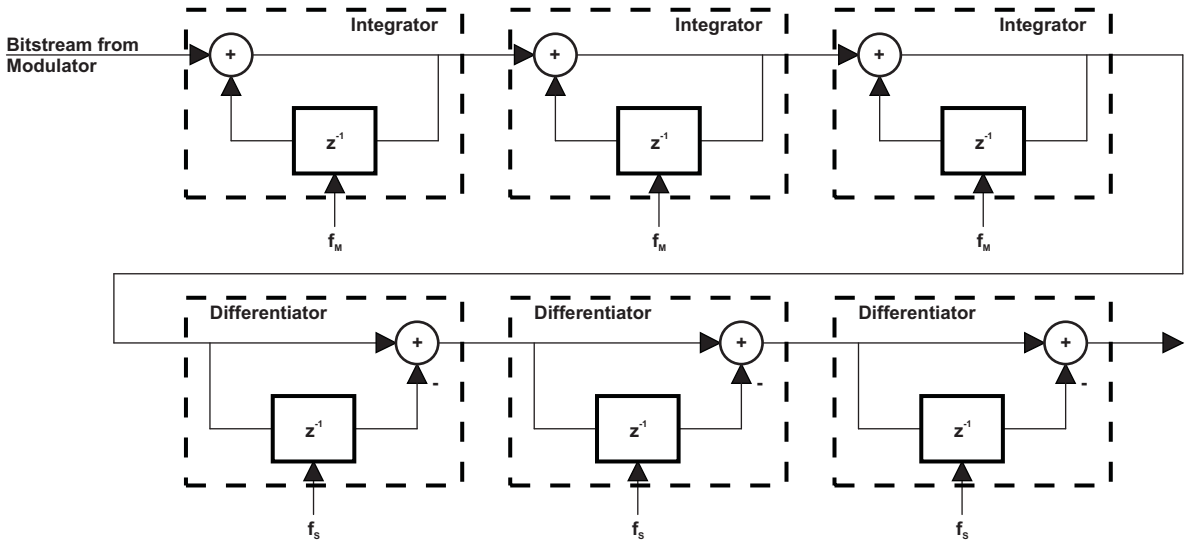


Figure 13-3. SINC³ Filter Structure

The transfer function is described in the z-domain by Equation 5.

$$H(z) = \left(\frac{1}{OSR} \times \frac{1 - z^{-OSR}}{1 - z^{-1}} \right)^3 \tag{5}$$

The transfer function is described in the frequency domain by Equation 6.

$$H(f) = \left(\frac{\text{sinc}\left(OSR\pi \frac{f}{f_M} \right)}{\text{sinc}\left(\pi \frac{f}{f_M} \right)} \right)^3 = \left(\frac{1}{OSR} \times \frac{\sin\left(OSR \times \pi \times \frac{f}{f_M} \right)}{\sin\left(\pi \times \frac{f}{f_M} \right)} \right)^3 \tag{6}$$

Where the oversampling rate, OSR, is the ratio of the modulator frequency f_M to the sample frequency f_S. Figure 13-4 shows the filter's frequency response for an OSR of 32. The first filter notch is always at $f_S = f_M/OSR$. The notch's frequency can be adjusted by changing the oversampling rate using the SD24OSRx bits.

The digital filter for each enabled ADC channel completes the decimation of the digital bit-stream and outputs new conversion results to the corresponding SD24MEMx register at the sample frequency f_S.

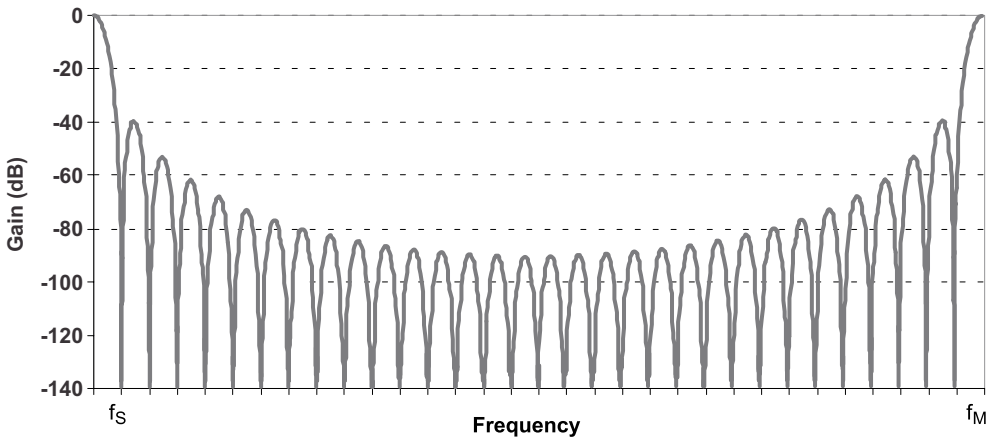


Figure 13-4. Comb Filter's Frequency Response With OSR = 32

Figure 13-5 shows the digital filter step response and conversion points. For step changes at the input after start of conversion, a settling time must be allowed before a valid conversion result is available. The SD24INTDLY bit can provide sufficient filter settling time for a full-scale change at the ADC input. If the step occurs synchronously to the decimation of the digital filter, the valid data is available on the third conversion. An asynchronous step requires four conversions before valid data is available.

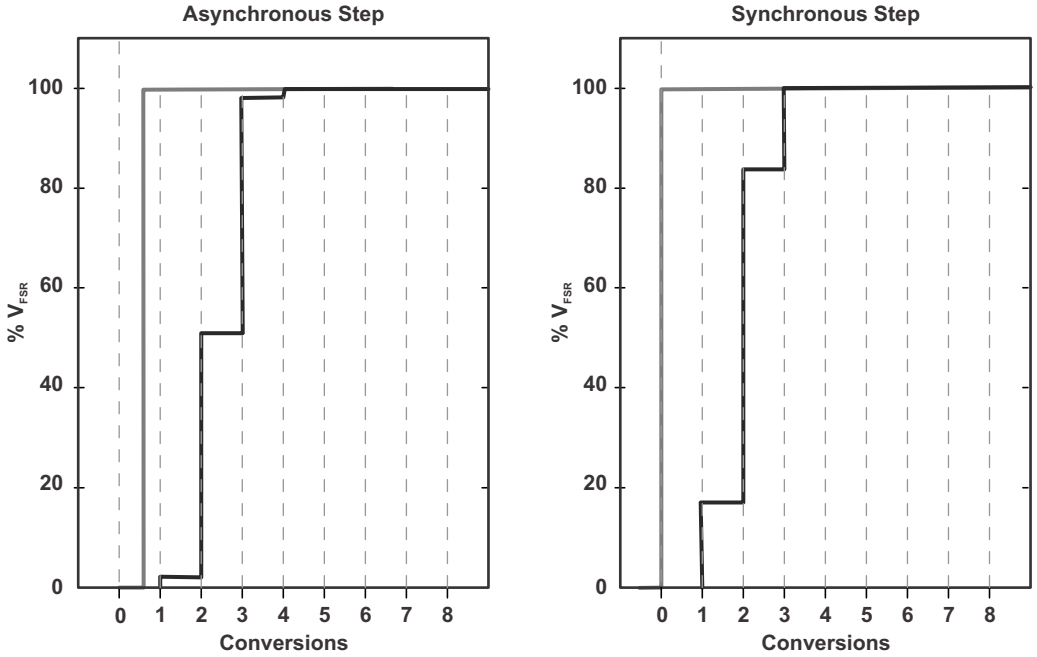


Figure 13-5. Digital Filter Step Response and Conversion Points Digital Filter Output

13.2.7.2 Digital Filter Output

The number of bits output by each digital filter depends on the oversampling ratio and ranges from 16 to 24 bits. Figure 13-6 shows the digital filter output bits and their relation to SD24MEMx for each OSR. For example, for OSR = 256 and LSBACC = 0, the SD24MEMx register contains bits 23-8 of the digital filter output. When OSR = 32, the SD24MEMx LSB is always zero.

The SD24LSBACC and SD24LSBTOG bits give access to the least significant bits of the digital filter output. When SD24LSBACC = 1, the 16 least significant bits of the digital filter's output are read from SD24MEMx using word instructions. The SD24MEMx register can also be accessed with byte instructions returning only the 8 least significant bits of the digital filter output.

When SD24LSBTOG = 1, the SD24LSBACC bit is automatically toggled each time the corresponding channel's SD24MEMx register is read. This allows the complete digital filter output result to be read with two read accesses of SD24MEMx. Setting or clearing SD24LSBTOG does not change SD24LSBACC until the next SD24MEMx access.

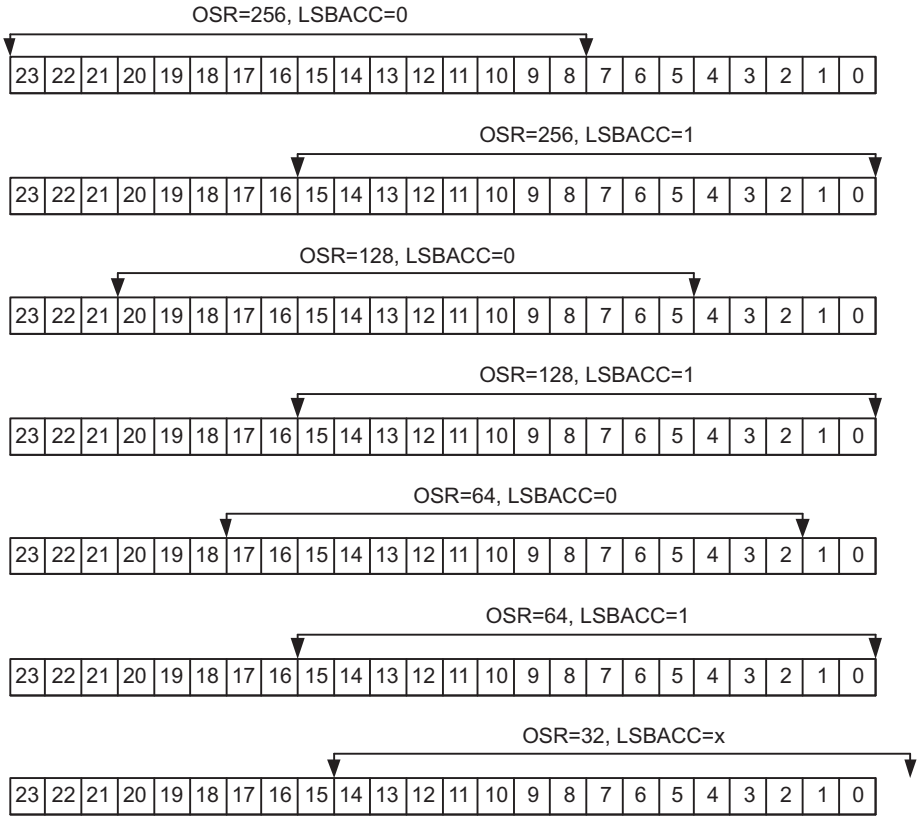


Figure 13-6. Used Bits of Digital Filter Output

13.2.8 Conversion Memory Registers: SD24MEMx

One SD24MEMx register is associated with each SD24 channel. Conversion results for each channel are moved to the corresponding SD24MEMx register with each decimation step of the digital filter. The SD24IFG bit for a given channel is set when new data is written to SD24MEMx. SD24IFG is automatically cleared when SD24MEMx is read by the CPU, or it can be cleared with software.

13.2.8.1 Output Data Format

The output data format is configurable in twos complement or offset binary as shown in Table 13-1. The data format is selected by the SD24DF bit.

Table 13-1. Data Format

SD24DF	Format	Analog Input	SD24MEMx[1]	Digital Filter Output (OSR = 256)
0	Offset binary	+FSR	FFFF	FFFFFF
		Zero	8000	800000
		-FSR	0000	000000
1	Twos complement	+FSR	7FFF	7FFFFF
		Zero	0000	000000
		-FSR	8000	800000

(1) Independent of SD24OSRx setting; SD24LSBACC = 0

Figure 13-7 shows the relationship between the full-scale input voltage range from $-V_{FSR}$ to $+V_{FSR}$ and the conversion result. The digital values for both data formats are shown.

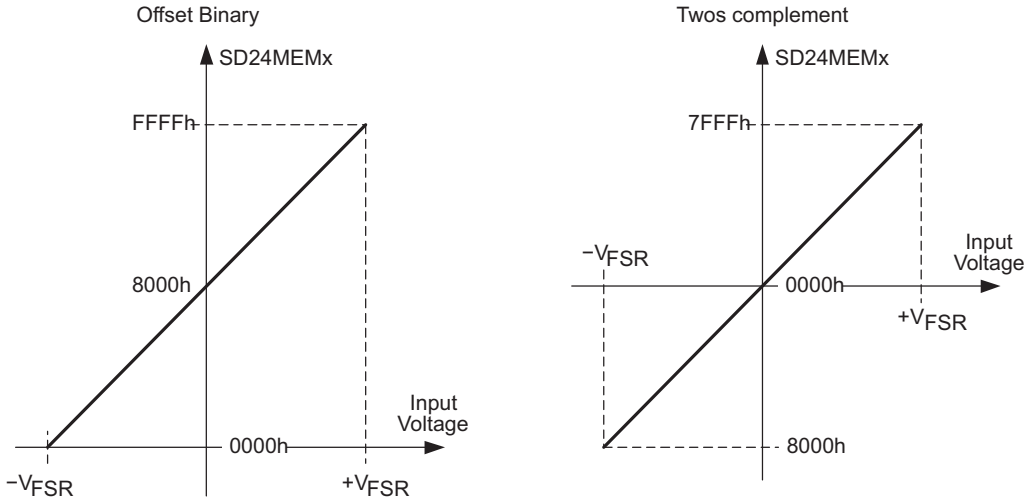


Figure 13-7. Input Voltage vs Digital Output

13.2.9 Conversion Modes

The SD24 module can be configured for four modes of operation, as shown in Table 13-2. The SD24SNGL and SD24GRP bits for each channel select the conversion mode.

Table 13-2. Conversion Mode Summary

SD24SNGL	SD24GRP[1]	Mode	Operation
1	0	Single channel, single conversion	A single channel is converted once.
0	0	Single channel, continuous conversion	A single channel is converted continuously.
1	1	Group of channels, single conversion	A group of channels is converted once.
0	1	Group of channels, continuous conversion	A group of channels is converted continuously.

(1) A channel is grouped and is the master channel of the group when SD24GRP = 0 if SD24GRP for the prior channel is set.

13.2.9.1 Single Channel, Single Conversion

Setting the SD24SC bit of a channel initiates one conversion on that channel when SD24SNGL = 1 and when the channel is not grouped with any other channels. The SD24SC bit is automatically cleared after conversion is complete.

Clearing SD24SC before the conversion is completed immediately stops conversion of the selected channel, the channel is powered down, and the corresponding digital filter is turned off. The value in SD24MEMx can change when SD24SC is cleared. It is recommended that the conversion data in SD24MEMx be read prior to clearing SD24SC to avoid reading an invalid result.

13.2.9.2 Single Channel, Continuous Conversion

When SD24SNGL = 0, continuous conversion mode is selected. Conversion of the selected channel begins when SD24SC is set and continues until the SD24SC bit is cleared by software when the channel is not grouped with any other channel.

Clearing SD24SC immediately stops conversion of the selected channel, the channel is powered down, and the corresponding digital filter is turned off. The value in SD24MEMx can change when SD24SC is cleared. It is recommended that the conversion data in SD24MEMx be read prior to clearing SD24SC to avoid reading an invalid result.

Figure 13-8 shows single channel operation for single conversion mode and continuous conversion mode.

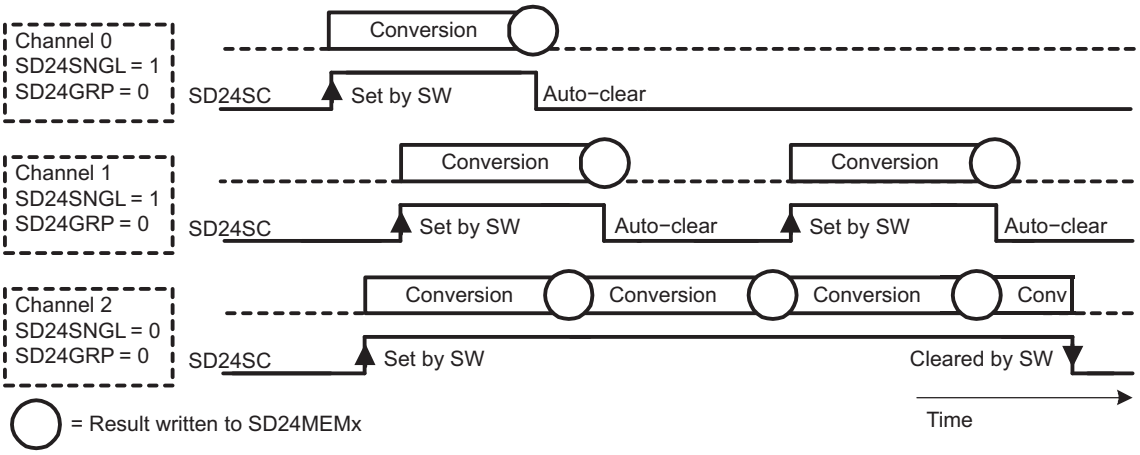


Figure 13-8. Single Channel Operation Example

13.2.9.3 Group of Channels, Single Conversion

Consecutive SD24 channels can be grouped together with the SD24GRP bit to synchronize conversions. Setting SD24GRP for a channel groups that channel with the next channel in the module. For example, setting SD24GRP for channel 0 groups that channel with channel 1. In this case, channel 1 is the master channel, enabling and disabling conversion of all channels in the group with its SD24SC bit. The SD24GRP bit of the master channel is always 0. The SD24GRP bit of the last channel in SD24 has no function and is always 0.

When SD24SNGL = 1 for a channel in a group, single conversion mode is selected. A single conversion of that channel occurs synchronously when the master channel SD24SC bit is set. The SD24SC bit of all channels in the group are automatically set and cleared by the SD24SC bit of the master channel. SD24SC for each channel can also be cleared in software independently.

Clearing SD24SC of the master channel before the conversions are completed immediately stops conversions of all channels in the group, the channels are powered down, and the corresponding digital filters are turned off. Values in SD24MEMx can change when SD24SC is cleared. It is recommended that the conversion data in SD24MEMx be read prior to clearing SD24SC to avoid reading an invalid result.

13.2.9.4 Group of Channels, Continuous Conversion

When SD24SNGL = 0 for a channel in a group, continuous conversion mode is selected. Continuous conversion of that channel occurs synchronously when the master channel SD24SC bit is set. SD24SC bits for all grouped channels are automatically set and cleared with the master channel's SD24SC bit. SD24SC for each channel in the group can also be cleared in software independently.

When SD24SC of a grouped channel is set by software independently of the master, conversion of that channel automatically synchronizes to conversions of the master channel. This ensures that conversions for grouped channels are always synchronous to the master.

Clearing SD24SC of the master channel immediately stops conversions of all channels in the group the channels are powered down and the corresponding digital filters are turned off. Values in SD24MEMx can change when SD24SC is cleared. It is recommended that the conversion data in SD24MEMx be read prior to clearing SD24SC to avoid reading an invalid result.

Figure 13-9 shows grouped channel operation for three SD24 channels. Channel 0 is configured for single conversion mode, SD24SNGL = 1, and channels 1 and 2 are in continuous conversion mode, SD24SNGL = 0. Channel two, the last channel in the group, is the master channel. Conversions of all channels in the group occur synchronously to the master channel regardless of when each SD24SC bit is set using software.

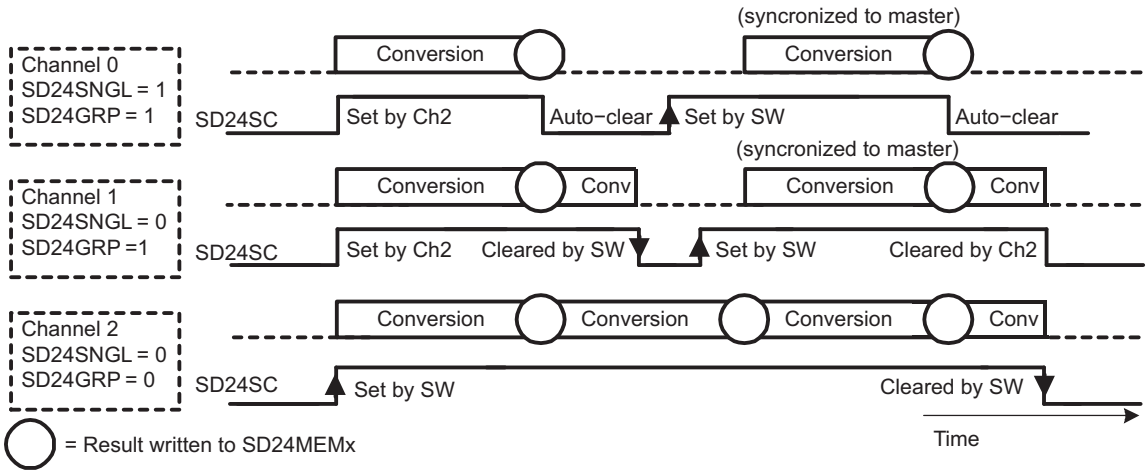


Figure 13-9. Grouped Channel Operation Example

13.2.10 Conversion Operation Using Preload

When multiple channels are grouped, the SD24PREx registers can be used to delay the conversion time frame for each channel. Using SD24PREx, the decimation time of the digital filter is increased by the specified number of f_M clock cycles. SD24PREx can range from 0 to 255. Figure 13-10 shows an example using SD24PREx.

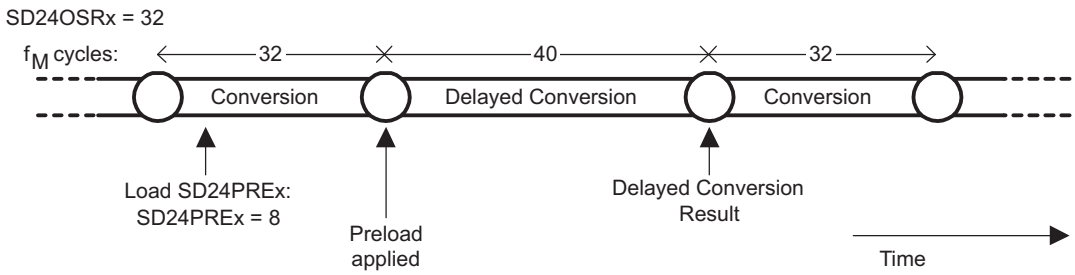


Figure 13-10. Conversion Delay Using Preload Example

The SD24PREx delay is applied to the beginning of the next conversion cycle after being written. The delay is used on the first conversion after SD24SC is set and on the conversion cycle following each write to SD24PREx. Following conversions are not delayed. After modifying SD24PREx, the next write to SD24PREx should not occur until the next conversion cycle is completed, otherwise the conversion results may be incorrect.

The accuracy of the result for the delayed conversion cycle using SD24PREx is dependent on the length of the delay and the frequency of the analog signal being sampled. For example, when measuring a dc signal, SD24PREx delay has no effect on the conversion result regardless of the duration. The user must determine when the delayed conversion result is useful in the application.

Figure 13-11 shows the operation of grouped channels 0 and 1. The preload register of channel 1 is loaded with zero, which results in immediate conversion. The conversion cycle of channel 0 is delayed by setting SD24PRE0 = 8. The first channel 0 conversion uses SD24PREx = 8, shifting all subsequent conversions by 8 f_M clock cycles.

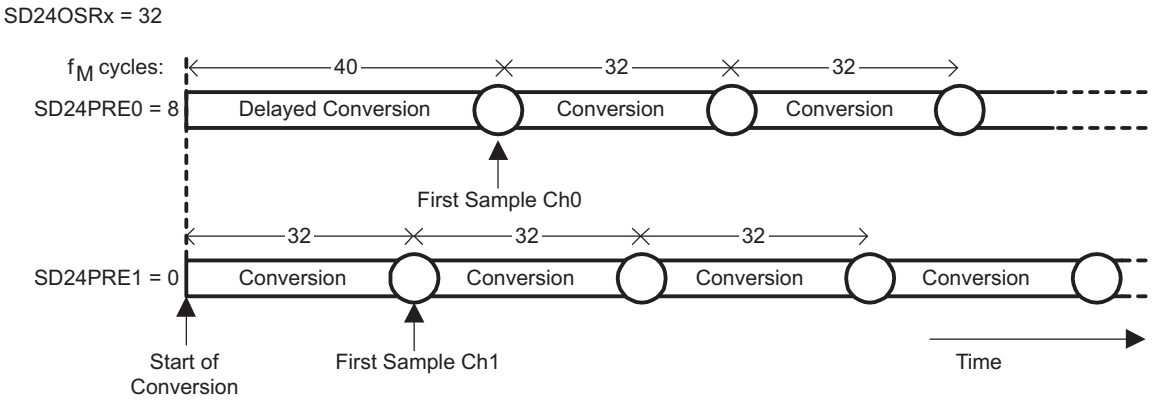


Figure 13-11. Start of Conversion Using Preload Example

When channels are grouped, care must be taken when a channel or channels operate in single conversion mode or are disabled in software while the master channel remains active. Each time channels in the group are re-enabled and resynchronized with the master channel, the preload delay for that channel is used. Figure 13-12 shows the re-synchronization and preload delays for channels in a group. It is recommended that SD24PREx = 0 for the master channel to maintain a consistent delay between the master and remaining channels in the group when they are re-enabled.

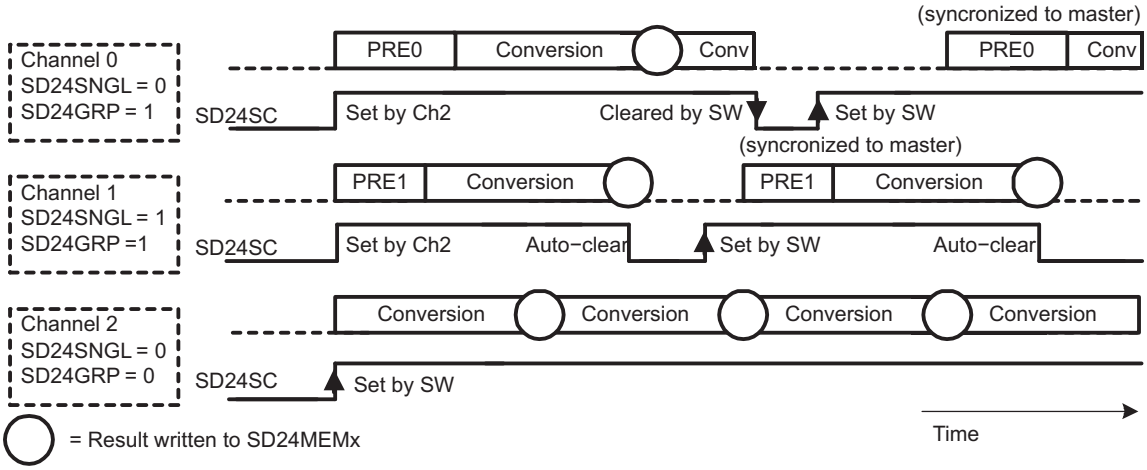


Figure 13-12. Preload and Channel Synchronization

13.2.11 Integrated Temperature Sensor

To use the on-chip temperature sensor, select the analog input pair SD24INCHx = 110. All other configuration is done as if an external analog input pair were selected, including SD24INTDLYx and SD24GAINx settings. It is possible to use the temperature sensor together with any of the available SD24 channels. However it is not allowed to use the temperature sensor with more than one SD24 channel at a time. The temperature measurement results will not be correct if more than one SD24 channels select the temperature sensor for conversion. SD24 can operate with either internal or external reference while using the temperature sensor.

Figure 13-13 shows the typical temperature sensor transfer function. When switching inputs of an SD24 channel to the temperature sensor, adequate delay must be provided using SD24INTDLYx to allow the digital filter to settle and assure that conversion results are valid. The temperature sensor offset error can be large and may need to be calibrated for most applications. See the device-specific data sheet for temperature sensor parameters.

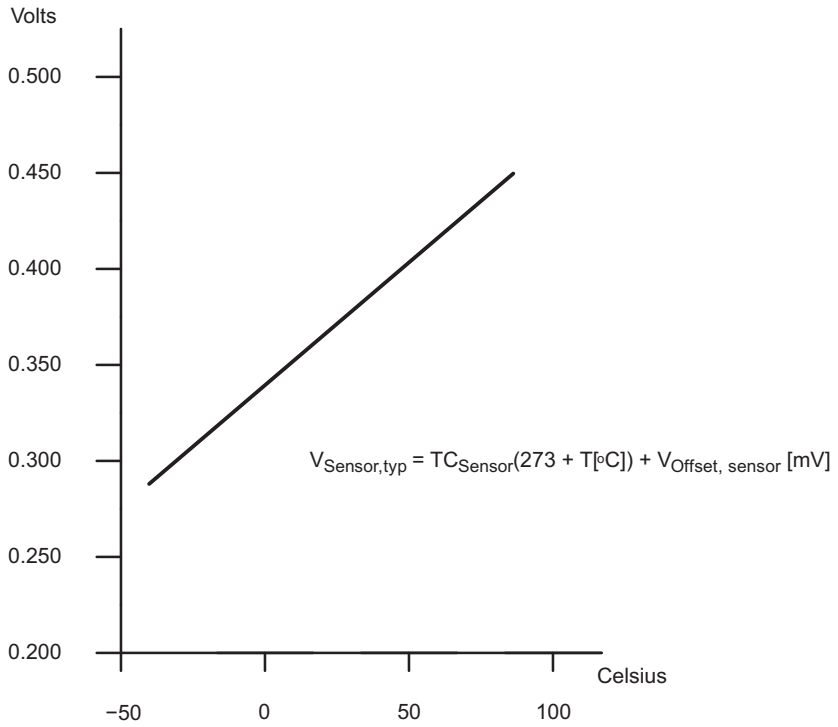


Figure 13-13. Typical Temperature Sensor Transfer Function

13.2.12 Interrupt Handling

The SD24 has two interrupt sources for each channel:

- SD24IFG
- SD24OVIFG

The SD24IFG bits are set when their corresponding SD24MEMx memory register is written with a conversion result. An interrupt request is generated if the corresponding SD24IE bit and the GIE bit are set. The SD24 overflow condition (SD24OVIFG) occurs when a conversion result is written to any SD24MEMx location before the previous conversion result was read.

13.2.12.1 SD24IV, Interrupt Vector Generator

All SD24 interrupt sources are prioritized and combined to source a single interrupt vector. SD24IV is used to determine which enabled SD24 interrupt source requested an interrupt. The highest priority SD24 interrupt request that is enabled generates a number in the SD24IV register (see register description). This number can be evaluated or added to the program counter to automatically enter the appropriate software routine. Disabled SD24 interrupts do not affect the SD24IV value.

Any access, read or write, of the SD24IV register has no effect on the SD24OVIFG or SD24IFG flags. The SD24IFG flags are reset by reading the associated SD24MEMx register or by clearing the flags in software. SD24OVIFG bits can only be reset with software.

If another interrupt is pending after servicing of an interrupt, another interrupt is generated. For example, if the SD24OVIFG and one or more SD24IFG interrupts are pending when the interrupt service routine accesses the SD24IV register, the SD24OVIFG interrupt condition is serviced first and the corresponding flag must be cleared in software. After the RETI instruction of the interrupt service routine is executed, the highest priority SD24IFG pending generates another interrupt request.

13.2.12.2 Interrupt Delay Operation

The SD24INTDLY bit controls the timing for the first interrupt service request for the corresponding channel. This feature delays the interrupt request for a completed conversion by up to four conversion cycles, which allows the digital filter to settle prior to generating an interrupt request. The delay is applied each time the SD24SC bit is set or when the SD24GAINx or SD24INCHx bits for the channel are modified. SD24INTDLY disables overflow interrupt generation for the channel for the selected number of delay cycles. Interrupt requests for the delayed conversions are not generated during the delay.

Example 13-1 shows the recommended use of SD24IV and the handling overhead. The SD24IV value is added to the PC to automatically jump to the appropriate routine.

Example 13-1. SD24 Interrupt Handling Software Example

The numbers at the right margin show the necessary CPU cycles for each instruction. The software overhead for different interrupt sources includes interrupt latency and return-from-interrupt cycles, but not the task handling itself. The latencies are:

- SD24OVIFG, CH0 SD24IFG, CH1 SD24IFG: 16 cycles
- CH2 SD24IFG: 14 cycles

The interrupt handler for channel 2 SD24IFG shows a way to check immediately if a higher prioritized interrupt occurred during the processing of the ISR. This saves nine cycles if another SD24 interrupt is pending.

```
; Interrupt handler for SD24.

INT_SD24                ; Enter Interrupt Service Routine   6
   ADD   &SD24IV,PC     ; Add offset to PC                  3
   RETI                 ; Vector 0: No interrupt            5
   JMP   ADOV           ; Vector 2: ADC overflow            2
   JMP   ADM0           ; Vector 4: CH_0 SD24IFG            2
   JMP   ADM1           ; Vector 6: CH_1 SD24IFG            2
;
; Handler for CH_2 SD24IFG starts here. No JMP required.
;
ADM2     MOV   &SD24MEM2,xxx   ; Move result, flag is reset
         ...                   ; Other instruction needed?
         JMP   INT_SD24        ; Check other int pending     2
;
; Remaining Handlers
;
ADM1     MOV   &SD24MEM1,xxx  ; Move result, flag is reset
         ...                  ; Other instruction needed?
         RETI                 ; Return                      5
;
ADM0     MOV   &SD24MEM0,xxx  ; Move result, flag is reset
         RETI                 ; Return                      5
```

Example 13-1. SD24 Interrupt Handling Software Example (continued)

```
;
ADOV    ...                 ; Handle SD24MEMx overflow
        RETI                ; Return                        5
```

13.3 SD24 Registers

Table 13-3 lists the SD24 registers.

Table 13-3. SD24 Registers

Address	Acronym	Register Name	Type	Access	Reset	Section
0100h	SD24CTL	SD24 Control	Read/write	Word	0000h	Section 13.3.1
0102h	SD24CCTL0	SD24 Channel 0 Control	Read/write	Word	0000h	Section 13.3.2
0110h	SD24MEM0	SD24 Channel 0 Conversion Memory	Read	Word	0000h	Section 13.3.3
00B0h	SD24INCTL0	SD24 Channel 0 Input Control	Read/write	Byte	00h	Section 13.3.4
00B8h	SD24PRE0	SD24 Channel 0 Preload	Read/write	Byte	00h	Section 13.3.5
0104h	SD24CCTL1	SD24 Channel 1 Control	Read/write	Word	0000h	Section 13.3.6
0112h	SD24MEM1	SD24 Channel 1 Conversion Memory	Read	Word	0000h	Section 13.3.7
00B1h	SD24INCTL1	SD24 Channel 1 Input Control	Read/write	Byte	00h	Section 13.3.8
00B9h	SD24PRE1	SD24 Channel 1 Preload	Read/write	Byte	00h	Section 13.3.9
0106h	SD24CCTL2	SD24 Channel 2 Control	Read/write	Word	0000h	Section 13.3.10
0114h	SD24MEM2	SD24 Channel 2 Conversion Memory	Read	Word	0000h	Section 13.3.11
00B2h	SD24INCTL2	SD24 Channel 2 Input Control	Read/write	Byte	00h	Section 13.3.12
00BAh	SD24PRE2	SD24 Channel 2 Preload	Read/write	Byte	00h	Section 13.3.13
0108h	SD24CCTL3	SD24 Channel 3 Control	Read/write	Word	0000h	Section 13.3.14
0116h	SD24MEM3	SD24 Channel 3 Conversion Memory	Read	Word	0000h	Section 13.3.15
00B3h	SD24INCTL3	SD24 Channel 3 Input Control	Read/write	Byte	00h	Section 13.3.16
00BBh	SD24PRE3	SD24 Channel 3 Preload	Read/write	Byte	00h	Section 13.3.17
010Ah	SD24CCTL4	SD24 Channel 4 Control	Read/write	Word	0000h	Section 13.3.18
0118h	SD24MEM4	SD24 Channel 4 Conversion Memory	Read	Word	0000h	Section 13.3.19
00B4h	SD24INCTL4	SD24 Channel 4 Input Control	Read/write	Byte	00h	Section 13.3.20
00BCh	SD24PRE4	SD24 Channel 4 Preload	Read/write	Byte	00h	Section 13.3.21
010Ch	SD24CCTL5	SD24 Channel 5 Control	Read/write	Word	0000h	Section 13.3.22
011Ah	SD24MEM5	SD24 Channel 5 Conversion Memory	Read	Word	0000h	Section 13.3.23
00B5h	SD24INCTL5	SD24 Channel 5 Input Control	Read/write	Byte	00h	Section 13.3.24
00BDh	SD24PRE5	SD24 Channel 5 Preload	Read/write	Byte	00h	Section 13.3.25
010Eh	SD24CCTL6	SD24 Channel 6 Control	Read/write	Word	0000h	Section 13.3.26
011Ch	SD24MEM6	SD24 Channel 6 Conversion Memory	Read	Word	0000h	Section 13.3.27
00B6h	SD24INCTL6	SD24 Channel 6 Input Control	Read/write	Byte	00h	Section 13.3.28
00BEh	SD24PRE6	SD24 Channel 6 Preload	Read/write	Byte	00h	Section 13.3.29
01AEh	SD24IV	SD24 Interrupt Vector	Read/write	Word	0000h	Section 13.3.30
00BFh	SD24TRIM	SD24 Trim	Read/write	Byte	0Ch	Section 13.3.31

13.3.1 SD24CTL Register (address = 0100h) [reset = 0000h]

SD24 Control Register

Figure 13-14. SD24CTL Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved					SD24REFS	SD24OVIE	Reserved
r0	r0	r0	r0	r0	rw-0	rw-0	r0

Table 13-4. SD24CTL Register Description

Bit	Field	Type	Reset	Description
15-3	Reserved	R	0h	Reserved. Always reads as 0.
2	SD24REFS	RW	0h	SD24 reference select. 0b = External reference selected. Internal reference voltage buffer disabled. 1b = Internal reference from shared REF selected and buffered internally to SD24
1	SD24OVIE	RW	0h	SD24 overflow interrupt enable. The GIE bit must also be set to enable the interrupt. 0b = Overflow interrupt disabled 1b = Overflow interrupt enabled
0	Reserved	R	0h	Reserved. Always reads as 0.

13.3.2 SD24CCTL0 Register (address = 0102h) [reset = 0000h]

SD24 Channel 0 Control Register

Figure 13-15. SD24CCTL0 Register

15	14	13	12	11	10	9	8
Reserved					SD24SNGL	SD24OSRx	
r0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
SD24LSBTOG	SD24LSBACC	SD24OVIFG	SD24DF	SD24IE	SD24IFG	SD24SC	SD24GRP
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r(w)-0

Table 13-5. SD24CCTL0 Register Description

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved. Always reads as 0.
10	SD24SNGL	RW	0h	Single conversion mode select 0b = Continuous conversion mode 1b = Single conversion mode
9-8	SD24OSRx	RW	0h	Oversampling ratio 00b = 256 01b = 128 10b = 64 11b = 32
7	SD24LSBTOG	RW	0h	LSB toggle. This bit, when set, causes SD24LSBACC to toggle each time the SD24MEM0 register is read. 0b = SD24LSBACC does not toggle with each SD24MEM0 read 1b = SD24LSBACC toggles with each SD24MEM0 read
6	SD24LSBACC	RW	0h	LSB access. This bit allows access to the upper or lower 16-bits of the SD24 conversion result. 0b = SD24MEM0 contains the most significant 16-bits of the conversion. 1b = SD24MEM0 contains the least significant 16-bits of the conversion.
5	SD24OVIFG	RW	0h	SD24 overflow interrupt flag 0b = No overflow interrupt pending 1b = Overflow interrupt pending
4	SD24DF	RW	0h	SD24 data format 0b = Offset binary 1b = 2s complement
3	SD24IE	RW	0h	SD24 interrupt enable 0b = Disabled 1b = Enabled
2	SD24IFG	RW	0h	SD24 interrupt flag. SD24IFG is set when new conversion results are available. SD24IFG is automatically reset when the SD24MEM0 register is read, or may be cleared with software. 0b = No interrupt pending 1b = Interrupt pending
1	SD24SC	RW	0h	SD24 start conversion 0b = No conversion start 1b = Start conversion
0	SD24GRP	RW	0h	SD24 group. Groups SD24 channel with next higher channel. Not used for the last channel. 0b = Not grouped 1b = Grouped

13.3.3 SD24MEM0 Register (address = 0110h) [reset = 0000h]

SD24 Channel 0 Conversion Memory Register

Figure 13-16. SD24MEM0 Register

15	14	13	12	11	10	9	8
ConversionResults							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
ConversionResults							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

Table 13-6. SD24MEM0 Register Description

Bit	Field	Type	Reset	Description
15-0	ConversionResults	R	0h	Conversion Results. The SD24MEM0 register holds the upper or lower 16-bits of the digital filter output, depending on the SD24LSBACC bit.

13.3.4 SD24INCTL0 Register (address = 00B0h) [reset = 00h]

SD24 Channel 0 Input Control Register

Figure 13-17. SD24INCTL0 Register

7	6	5	4	3	2	1	0
Reserved	SD24INTDLY	SD24GAINx			SD24INCHx		
r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 13-7. SD24INCTL0 Register Description

Bit	Field	Type	Reset	Description
7	Reserved	R	0h	Reserved. Always reads as 0.
6	SD24INTDLY	RW	0h	Interrupt delay generation after conversion start. This bit selects the delay for the first interrupt after conversion start. 0b = Fourth sample causes interrupt 1b = First sample causes interrupt
5-3	SD24GAINx	RW	0h	SD24 preamplifier gain 000b = x1 001b = x2 010b = x4 011b = x8 100b = x16 101b = Reserved 110b = Reserved 111b = Reserved
2-0	SD24INCHx	RW	0h	SD24 channel differential pair input 000b = A0.0 001b = A0.0 010b = A0.0 011b = A0.0 100b = A0.0 101b = A0.0 110b = A0.6, Temperature Sensor 111b = Reserved

13.3.5 SD24PRE0 Register (address = 00B8h) [reset = 00h]

SD24 Channel 0 Preload Register

Figure 13-18. SD24PRE0 Register

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

Table 13-8. SD24PRE0 Register Description

Bit	Field	Type	Reset	Description
7-0	PreloadValue	RW	0h	SD24 digital filter preload value.

13.3.6 SD24CCTL1 Register (address = 0104h) [reset = 0000h]

SD24 Channel 1 Control Register

Figure 13-19. SD24CCTL1 Register

15	14	13	12	11	10	9	8
Reserved					SD24SNGL	SD24OSRx	
r0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
SD24LSBTOG	SD24LSBACC	SD24OVIFG	SD24DF	SD24IE	SD24IFG	SD24SC	SD24GRP
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r(w)-0

Table 13-9. SD24CCTL1 Register Description

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved. Always reads as 0.
10	SD24SNGL	RW	0h	Single conversion mode select 0b = Continuous conversion mode 1b = Single conversion mode
9-8	SD24OSRx	RW	0h	Oversampling ratio 00b = 256 01b = 128 10b = 64 11b = 32
7	SD24LSBTOG	RW	0h	LSB toggle. This bit, when set, causes SD24LSBACC to toggle each time the SD24MEM1 register is read. 0b = SD24LSBACC does not toggle with each SD24MEM1 read 1b = SD24LSBACC toggles with each SD24MEM1 read
6	SD24LSBACC	RW	0h	LSB access. This bit allows access to the upper or lower 16-bits of the SD24 conversion result. 0b = SD24MEM1 contains the most significant 16-bits of the conversion. 1b = SD24MEM1 contains the least significant 16-bits of the conversion.
5	SD24OVIFG	RW	0h	SD24 overflow interrupt flag 0b = No overflow interrupt pending 1b = Overflow interrupt pending
4	SD24DF	RW	0h	SD24 data format 0b = Offset binary 1b = 2s complement
3	SD24IE	RW	0h	SD24 interrupt enable 0b = Disabled 1b = Enabled
2	SD24IFG	RW	0h	SD24 interrupt flag. SD24IFG is set when new conversion results are available. SD24IFG is automatically reset when the SD24MEM1 register is read, or may be cleared with software. 0b = No interrupt pending 1b = Interrupt pending
1	SD24SC	RW	0h	SD24 start conversion 0b = No conversion start 1b = Start conversion
0	SD24GRP	RW	0h	SD24 group. Groups SD24 channel with next higher channel. Not used for the last channel. 0b = Not grouped 1b = Grouped

13.3.7 SD24MEM1 Register (address = 0112h) [reset = 0000h]

SD24 Channel 1 Conversion Memory Register

Figure 13-20. SD24MEM1 Register

15	14	13	12	11	10	9	8
ConversionResults							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
ConversionResults							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

Table 13-10. SD24MEM1 Register Description

Bit	Field	Type	Reset	Description
15-0	ConversionResults	R	0h	Conversion Results. The SD24MEM1 register holds the upper or lower 16-bits of the digital filter output, depending on the SD24LSBACC bit.

13.3.8 SD24INCTL1 Register (address = 00B1h) [reset = 00h]

SD24 Channel 1 Input Control Register

Figure 13-21. SD24INCTL1 Register

7	6	5	4	3	2	1	0
Reserved	SD24INTDLY	SD24GAINx			SD24INCHx		
r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 13-11. SD24INCTL1 Register Description

Bit	Field	Type	Reset	Description
7	Reserved	R	0h	Reserved. Always reads as 0.
6	SD24INTDLY	RW	0h	Interrupt delay generation after conversion start. This bit selects the delay for the first interrupt after conversion start. 0b = Fourth sample causes interrupt 1b = First sample causes interrupt
5-3	SD24GAINx	RW	0h	SD24 preamplifier gain 000b = x1 001b = x2 010b = x4 011b = x8 100b = x16 101b = Reserved 110b = Reserved 111b = Reserved
2-0	SD24INCHx	RW	0h	SD24 channel differential pair input 000b = A1.0 001b = A1.0 010b = A1.0 011b = A1.0 100b = A1.0 101b = A1.0 110b = A1.6, Temperature Sensor 111b = Reserved

13.3.9 SD24PRE1 Register (address = 00B9h) [reset = 00h]

SD24 Channel 1 Preload Register

Figure 13-22. SD24PRE1 Register

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

Table 13-12. SD24PRE1 Register Description

Bit	Field	Type	Reset	Description
7-0	PreloadValue	RW	0h	SD24 digital filter preload value.

13.3.10 SD24CCTL2 Register (address = 0106h) [reset = 0000h]

SD24 Channel 2 Control Register

Figure 13-23. SD24CCTL2 Register

15	14	13	12	11	10	9	8
Reserved					SD24SNGL	SD24OSRx	
r0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
SD24LSBTOG	SD24LSBACC	SD24OVIFG	SD24DF	SD24IE	SD24IFG	SD24SC	SD24GRP
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r(w)-0

Table 13-13. SD24CCTL2 Register Description

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved. Always reads as 0.
10	SD24SNGL	RW	0h	Single conversion mode select 0b = Continuous conversion mode 1b = Single conversion mode
9-8	SD24OSRx	RW	0h	Oversampling ratio 00b = 256 01b = 128 10b = 64 11b = 32
7	SD24LSBTOG	RW	0h	LSB toggle. This bit, when set, causes SD24LSBACC to toggle each time the SD24MEM2 register is read. 0b = SD24LSBACC does not toggle with each SD24MEM2 read 1b = SD24LSBACC toggles with each SD24MEM2 read
6	SD24LSBACC	RW	0h	LSB access. This bit allows access to the upper or lower 16-bits of the SD24 conversion result. 0b = SD24MEM2 contains the most significant 16-bits of the conversion. 1b = SD24MEM2 contains the least significant 16-bits of the conversion.
5	SD24OVIFG	RW	0h	SD24 overflow interrupt flag 0b = No overflow interrupt pending 1b = Overflow interrupt pending
4	SD24DF	RW	0h	SD24 data format 0b = Offset binary 1b = 2s complement
3	SD24IE	RW	0h	SD24 interrupt enable 0b = Disabled 1b = Enabled
2	SD24IFG	RW	0h	SD24 interrupt flag. SD24IFG is set when new conversion results are available. SD24IFG is automatically reset when the SD24MEM2 register is read, or may be cleared with software. 0b = No interrupt pending 1b = Interrupt pending
1	SD24SC	RW	0h	SD24 start conversion 0b = No conversion start 1b = Start conversion
0	SD24GRP	RW	0h	SD24 group. Groups SD24 channel with next higher channel. Not used for the last channel. 0b = Not grouped 1b = Grouped

13.3.11 SD24MEM2 Register (address = 0114h) [reset = 0000h]

SD24 Channel 2 Conversion Memory Register

Figure 13-24. SD24MEM2 Register

15	14	13	12	11	10	9	8
ConversionResults							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
ConversionResults							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

Table 13-14. SD24MEM2 Register Description

Bit	Field	Type	Reset	Description
15-0	ConversionResults	R	0h	Conversion Results. The SD24MEM2 register holds the upper or lower 16-bits of the digital filter output, depending on the SD24LSBACC bit.

13.3.12 SD24INCTL2 Register (address = 00B2h) [reset = 00h]

SD24 Channel 2 Input Control Register

Figure 13-25. SD24INCTL2 Register

7	6	5	4	3	2	1	0
Reserved	SD24INTDLY	SD24GAINx			SD24INCHx		
r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 13-15. SD24INCTL2 Register Description

Bit	Field	Type	Reset	Description
7	Reserved	R	0h	Reserved. Always reads as 0.
6	SD24INTDLY	RW	0h	Interrupt delay generation after conversion start. This bit selects the delay for the first interrupt after conversion start. 0b = Fourth sample causes interrupt 1b = First sample causes interrupt
5-3	SD24GAINx	RW	0h	SD24 preamplifier gain 000b = x1 001b = x2 010b = x4 011b = x8 100b = x16 101b = Reserved 110b = Reserved 111b = Reserved
2-0	SD24INCHx	RW	0h	SD24 channel differential pair input 000b = A2.0 001b = A2.0 010b = A2.0 011b = A2.0 100b = A2.0 101b = A2.0 110b = A2.6, Temperature Sensor 111b = Reserved

13.3.13 SD24PRE2 Register (address = 00BAh) [reset = 00h]

SD24 Channel 2 Preload Register

Figure 13-26. SD24PRE2 Register

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

Table 13-16. SD24PRE2 Register Description

Bit	Field	Type	Reset	Description
7-0	PreloadValue	RW	0h	SD24 digital filter preload value.

13.3.14 SD24CCTL3 Register (address = 0108h) [reset = 0000h]

SD24 Channel 3 Control Register

Figure 13-27. SD24CCTL3 Register

15	14	13	12	11	10	9	8
Reserved					SD24SNGL	SD24OSRx	
r0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
SD24LSBTOG	SD24LSBACC	SD24OVIFG	SD24DF	SD24IE	SD24IFG	SD24SC	SD24GRP
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r(w)-0

Table 13-17. SD24CCTL3 Register Description

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved. Always reads as 0.
10	SD24SNGL	RW	0h	Single conversion mode select 0b = Continuous conversion mode 1b = Single conversion mode
9-8	SD24OSRx	RW	0h	Oversampling ratio 00b = 256 01b = 128 10b = 64 11b = 32
7	SD24LSBTOG	RW	0h	LSB toggle. This bit, when set, causes SD24LSBACC to toggle each time the SD24MEM3 register is read. 0b = SD24LSBACC does not toggle with each SD24MEM3 read 1b = SD24LSBACC toggles with each SD24MEM3 read
6	SD24LSBACC	RW	0h	LSB access. This bit allows access to the upper or lower 16-bits of the SD24 conversion result. 0b = SD24MEM3 contains the most significant 16-bits of the conversion. 1b = SD24MEM3 contains the least significant 16-bits of the conversion.
5	SD24OVIFG	RW	0h	SD24 overflow interrupt flag 0b = No overflow interrupt pending 1b = Overflow interrupt pending
4	SD24DF	RW	0h	SD24 data format 0b = Offset binary 1b = 2s complement
3	SD24IE	RW	0h	SD24 interrupt enable 0b = Disabled 1b = Enabled
2	SD24IFG	RW	0h	SD24 interrupt flag. SD24IFG is set when new conversion results are available. SD24IFG is automatically reset when the SD24MEM3 register is read, or may be cleared with software. 0b = No interrupt pending 1b = Interrupt pending
1	SD24SC	RW	0h	SD24 start conversion 0b = No conversion start 1b = Start conversion
0	SD24GRP	RW	0h	SD24 group. Groups SD24 channel with next higher channel. Not used for the last channel. 0b = Not grouped 1b = Grouped

13.3.15 SD24MEM3 Register (address = 0116h) [reset = 0000h]

SD24 Channel 3 Conversion Memory Register

Figure 13-28. SD24MEM3 Register

15	14	13	12	11	10	9	8
ConversionResults							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
ConversionResults							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

Table 13-18. SD24MEM3 Register Description

Bit	Field	Type	Reset	Description
15-0	ConversionResults	R	0h	Conversion Results. The SD24MEM3 register holds the upper or lower 16-bits of the digital filter output, depending on the SD24LSBACC bit.

13.3.16 SD24INCTL3 Register (address = 00B3h) [reset = 00h]

SD24 Channel 3 Input Control Register

Figure 13-29. SD24INCTL3 Register

7	6	5	4	3	2	1	0
Reserved	SD24INTDLY	SD24GAINx			SD24INCHx		
r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 13-19. SD24INCTL3 Register Description

Bit	Field	Type	Reset	Description
7	Reserved	R	0h	Reserved. Always reads as 0.
6	SD24INTDLY	RW	0h	Interrupt delay generation after conversion start. This bit selects the delay for the first interrupt after conversion start. 0b = Fourth sample causes interrupt 1b = First sample causes interrupt
5-3	SD24GAINx	RW	0h	SD24 preamplifier gain 000b = x1 001b = x2 010b = x4 011b = x8 100b = x16 101b = Reserved 110b = Reserved 111b = Reserved
2-0	SD24INCHx	RW	0h	SD24 channel differential pair input 000b = A3.0 001b = A3.0 010b = A3.0 011b = A3.0 100b = A3.0 101b = A3.0 110b = A3.6, Temperature Sensor 111b = Reserved

13.3.17 SD24PRE3 Register (address = 00BBh) [reset = 00h]

SD24 Channel 3 Preload Register

Figure 13-30. SD24PRE3 Register

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

Table 13-20. SD24PRE3 Register Description

Bit	Field	Type	Reset	Description
7-0	PreloadValue	RW	0h	SD24 digital filter preload value.

13.3.18 SD24CCTL4 Register (address = 010Ah) [reset = 0000h]

SD24 Channel 4 Control Register

Figure 13-31. SD24CCTL4 Register

15	14	13	12	11	10	9	8
Reserved					SD24SNGL	SD24OSRx	
r0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
SD24LSBTOG	SD24LSBACC	SD24OVIFG	SD24DF	SD24IE	SD24IFG	SD24SC	SD24GRP
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r(w)-0

Table 13-21. SD24CCTL4 Register Description

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved. Always reads as 0.
10	SD24SNGL	RW	0h	Single conversion mode select 0b = Continuous conversion mode 1b = Single conversion mode
9-8	SD24OSRx	RW	0h	Oversampling ratio 00b = 256 01b = 128 10b = 64 11b = 32
7	SD24LSBTOG	RW	0h	LSB toggle. This bit, when set, causes SD24LSBACC to toggle each time the SD24MEM4 register is read. 0b = SD24LSBACC does not toggle with each SD24MEM4 read 1b = SD24LSBACC toggles with each SD24MEM4 read
6	SD24LSBACC	RW	0h	LSB access. This bit allows access to the upper or lower 16-bits of the SD24 conversion result. 0b = SD24MEM4 contains the most significant 16-bits of the conversion. 1b = SD24MEM4 contains the least significant 16-bits of the conversion.
5	SD24OVIFG	RW	0h	SD24 overflow interrupt flag 0b = No overflow interrupt pending 1b = Overflow interrupt pending
4	SD24DF	RW	0h	SD24 data format 0b = Offset binary 1b = 2s complement
3	SD24IE	RW	0h	SD24 interrupt enable 0b = Disabled 1b = Enabled
2	SD24IFG	RW	0h	SD24 interrupt flag. SD24IFG is set when new conversion results are available. SD24IFG is automatically reset when the SD24MEM4 register is read, or may be cleared with software. 0b = No interrupt pending 1b = Interrupt pending
1	SD24SC	RW	0h	SD24 start conversion 0b = No conversion start 1b = Start conversion
0	SD24GRP	RW	0h	SD24 group. Groups SD24 channel with next higher channel. Not used for the last channel. 0b = Not grouped 1b = Grouped

13.3.19 SD24MEM4 Register (address = 0118h) [reset = 0000h]

SD24 Channel 4 Conversion Memory Register

Figure 13-32. SD24MEM4 Register

15	14	13	12	11	10	9	8
ConversionResults							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
ConversionResults							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

Table 13-22. SD24MEM4 Register Description

Bit	Field	Type	Reset	Description
15-0	ConversionResults	R	0h	Conversion Results. The SD24MEM4 register holds the upper or lower 16-bits of the digital filter output, depending on the SD24LSBACC bit.

13.3.20 SD24INCTL4 Register (address = 00B4h) [reset = 00h]

SD24 Channel 4 Input Control Register

Figure 13-33. SD24INCTL4 Register

7	6	5	4	3	2	1	0
Reserved	SD24INTDLY	SD24GAINx			SD24INCHx		
r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 13-23. SD24INCTL4 Register Description

Bit	Field	Type	Reset	Description
7	Reserved	R	0h	Reserved. Always reads as 0.
6	SD24INTDLY	RW	0h	Interrupt delay generation after conversion start. This bit selects the delay for the first interrupt after conversion start. 0b = Fourth sample causes interrupt 1b = First sample causes interrupt
5-3	SD24GAINx	RW	0h	SD24 preamplifier gain 000b = x1 001b = x2 010b = x4 011b = x8 100b = x16 101b = Reserved 110b = Reserved 111b = Reserved
2-0	SD24INCHx	RW	0h	SD24 channel differential pair input 000b = A4.0 001b = A4.0 010b = A4.0 011b = A4.0 100b = A4.0 101b = A4.0 110b = A4.6, Temperature Sensor 111b = Reserved

13.3.21 SD24PRE4 Register (address = 00BCh) [reset = 00h]

SD24 Channel 4 Preload Register

Figure 13-34. SD24PRE4 Register

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

Table 13-24. SD24PRE4 Register Description

Bit	Field	Type	Reset	Description
7-0	PreloadValue	RW	0h	SD24 digital filter preload value.

13.3.22 SD24CCTL5 Register (address = 010Ch) [reset = 0000h]

SD24 Channel 5 Control Register

Figure 13-35. SD24CCTL5 Register

15	14	13	12	11	10	9	8
Reserved					SD24SNGL	SD24OSRx	
r0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
SD24LSBTOG	SD24LSBACC	SD24OVIFG	SD24DF	SD24IE	SD24IFG	SD24SC	SD24GRP
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r(w)-0

Table 13-25. SD24CCTL5 Register Description

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved. Always reads as 0.
10	SD24SNGL	RW	0h	Single conversion mode select 0b = Continuous conversion mode 1b = Single conversion mode
9-8	SD24OSRx	RW	0h	Oversampling ratio 00b = 256 01b = 128 10b = 64 11b = 32
7	SD24LSBTOG	RW	0h	LSB toggle. This bit, when set, causes SD24LSBACC to toggle each time the SD24MEM5 register is read. 0b = SD24LSBACC does not toggle with each SD24MEM5 read 1b = SD24LSBACC toggles with each SD24MEM5 read
6	SD24LSBACC	RW	0h	LSB access. This bit allows access to the upper or lower 16-bits of the SD24 conversion result. 0b = SD24MEM5 contains the most significant 16-bits of the conversion. 1b = SD24MEM5 contains the least significant 16-bits of the conversion.
5	SD24OVIFG	RW	0h	SD24 overflow interrupt flag 0b = No overflow interrupt pending 1b = Overflow interrupt pending
4	SD24DF	RW	0h	SD24 data format 0b = Offset binary 1b = 2s complement
3	SD24IE	RW	0h	SD24 interrupt enable 0b = Disabled 1b = Enabled
2	SD24IFG	RW	0h	SD24 interrupt flag. SD24IFG is set when new conversion results are available. SD24IFG is automatically reset when the SD24MEM5 register is read, or may be cleared with software. 0b = No interrupt pending 1b = Interrupt pending
1	SD24SC	RW	0h	SD24 start conversion 0b = No conversion start 1b = Start conversion
0	SD24GRP	RW	0h	SD24 group. Groups SD24 channel with next higher channel. Not used for the last channel. 0b = Not grouped 1b = Grouped

13.3.23 SD24MEM5 Register (address = 011Ah) [reset = 0000h]

SD24 Channel 5 Conversion Memory Register

Figure 13-36. SD24MEM5 Register

15	14	13	12	11	10	9	8
ConversionResults							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
ConversionResults							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

Table 13-26. SD24MEM5 Register Description

Bit	Field	Type	Reset	Description
15-0	ConversionResults	R	0h	Conversion Results. The SD24MEM5 register holds the upper or lower 16-bits of the digital filter output, depending on the SD24LSBACC bit.

13.3.24 SD24INCTL5 Register (address = 00B5h) [reset = 00h]

SD24 Channel 5 Input Control Register

Figure 13-37. SD24INCTL5 Register

7	6	5	4	3	2	1	0
Reserved	SD24INTDLY	SD24GAINx			SD24INCHx		
r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 13-27. SD24INCTL5 Register Description

Bit	Field	Type	Reset	Description
7	Reserved	R	0h	Reserved. Always reads as 0.
6	SD24INTDLY	RW	0h	Interrupt delay generation after conversion start. This bit selects the delay for the first interrupt after conversion start. 0b = Fourth sample causes interrupt 1b = First sample causes interrupt
5-3	SD24GAINx	RW	0h	SD24 preamplifier gain 000b = x1 001b = x2 010b = x4 011b = x8 100b = x16 101b = Reserved 110b = Reserved 111b = Reserved
2-0	SD24INCHx	RW	0h	SD24 channel differential pair input 000b = A5.0 001b = A5.0 010b = A5.0 011b = A5.0 100b = A5.0 101b = A5.0 110b = A5.6, Temperature Sensor 111b = Reserved

13.3.25 SD24PRE5 Register (address = 00BDh) [reset = 00h]

SD24 Channel 5 Preload Register

Figure 13-38. SD24PRE5 Register

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

Table 13-28. SD24PRE5 Register Description

Bit	Field	Type	Reset	Description
7-0	PreloadValue	RW	0h	SD24 digital filter preload value.

13.3.26 SD24CCTL6 Register (address = 010Eh) [reset = 0000h]

SD24 Channel 6 Control Register

Figure 13-39. SD24CCTL6 Register

15	14	13	12	11	10	9	8
Reserved					SD24SNGL	SD24OSRx	
r0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
SD24LSBTOG	SD24LSBACC	SD24OVIFG	SD24DF	SD24IE	SD24IFG	SD24SC	SD24GRP
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r(w)-0

Table 13-29. SD24CCTL6 Register Description

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved. Always reads as 0.
10	SD24SNGL	RW	0h	Single conversion mode select 0b = Continuous conversion mode 1b = Single conversion mode
9-8	SD24OSRx	RW	0h	Oversampling ratio 00b = 256 01b = 128 10b = 64 11b = 32
7	SD24LSBTOG	RW	0h	LSB toggle. This bit, when set, causes SD24LSBACC to toggle each time the SD24MEM6 register is read. 0b = SD24LSBACC does not toggle with each SD24MEM6 read 1b = SD24LSBACC toggles with each SD24MEM6 read
6	SD24LSBACC	RW	0h	LSB access. This bit allows access to the upper or lower 16-bits of the SD24 conversion result. 0b = SD24MEM6 contains the most significant 16-bits of the conversion. 1b = SD24MEM6 contains the least significant 16-bits of the conversion.
5	SD24OVIFG	RW	0h	SD24 overflow interrupt flag 0b = No overflow interrupt pending 1b = Overflow interrupt pending
4	SD24DF	RW	0h	SD24 data format 0b = Offset binary 1b = 2s complement
3	SD24IE	RW	0h	SD24 interrupt enable 0b = Disabled 1b = Enabled
2	SD24IFG	RW	0h	SD24 interrupt flag. SD24IFG is set when new conversion results are available. SD24IFG is automatically reset when the SD24MEM6 register is read, or may be cleared with software. 0b = No interrupt pending 1b = Interrupt pending
1	SD24SC	RW	0h	SD24 start conversion 0b = No conversion start 1b = Start conversion
0	SD24GRP	RW	0h	SD24 group. Groups SD24 channel with next higher channel. Not used for the last channel. 0b = Not grouped 1b = Grouped

13.3.27 SD24MEM6 Register (address = 011Ch) [reset = 0000h]

SD24 Channel 6 Conversion Memory Register

Figure 13-40. SD24MEM6 Register

15	14	13	12	11	10	9	8
ConversionResults							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
ConversionResults							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

Table 13-30. SD24MEM6 Register Description

Bit	Field	Type	Reset	Description
15-0	ConversionResults	R	0h	Conversion Results. The SD24MEM6 register holds the upper or lower 16-bits of the digital filter output, depending on the SD24LSBACC bit.

13.3.28 SD24INCTL6 Register (address = 00B6h) [reset = 00h]

SD24 Channel 6 Input Control Register

Figure 13-41. SD24INCTL6 Register

7	6	5	4	3	2	1	0
Reserved	SD24INTDLY	SD24GAINx			SD24INCHx		
r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 13-31. SD24INCTL6 Register Description

Bit	Field	Type	Reset	Description
7	Reserved	R	0h	Reserved. Always reads as 0.
6	SD24INTDLY	RW	0h	Interrupt delay generation after conversion start. This bit selects the delay for the first interrupt after conversion start. 0b = Fourth sample causes interrupt 1b = First sample causes interrupt
5-3	SD24GAINx	RW	0h	SD24 preamplifier gain 000b = x1 001b = x2 010b = x4 011b = x8 100b = x16 101b = Reserved 110b = Reserved 111b = Reserved
2-0	SD24INCHx	RW	0h	SD24 channel differential pair input 000b = A6.0 001b = A6.0 010b = A6.0 011b = A6.0 100b = A6.0 101b = A6.0 110b = A6.6, Temperature Sensor 111b = Reserved

13.3.29 SD24PRE6 Register (address = 00BEh) [reset = 00h]

SD24 Channel 6 Preload Register

Figure 13-42. SD24PRE6 Register

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

Table 13-32. SD24PRE6 Register Description

Bit	Field	Type	Reset	Description
7-0	PreloadValue	RW	0h	SD24 digital filter preload value.

13.3.30 SD24IV Register (address = 01AEh) [reset = 0000h]

SD24 Interrupt Vector Register

Figure 13-43. SD24IV Register

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
0	0	0	SD24IVx				
r0	r0	r0	r-0	r-0	r-0	r-0	r-0

Table 13-33. SD24IV Register Description

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved. Always reads as 0.
4-0	SD24IVx	R	0h	SD24 interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: SD24MEMx overflow; Interrupt Flag: SD24CCTLx SD24OVIFG(1); Interrupt Priority = Highest. 04h = Interrupt Source: SD24_0 Interrupt; Interrupt Flag: SD24CCTL0 SD24IFG 06h = Interrupt Source: SD24_1 Interrupt; Interrupt Flag: SD24CCTL1 SD24IFG 08h = Interrupt Source: SD24_2 Interrupt; Interrupt Flag: SD24CCTL2 SD24IFG 0Ah = Interrupt Source: SD24_3 Interrupt; Interrupt Flag: SD24CCTL3 SD24IFG 0Ch = Interrupt Source: SD24_4 Interrupt; Interrupt Flag: SD24CCTL4 SD24IFG 0Eh = Interrupt Source: SD24_5 Interrupt; Interrupt Flag: SD24CCTL5 SD24IFG 10h = Interrupt Source: SD24_6 Interrupt; Interrupt Flag: SD24CCTL6 SD24IFG; Interrupt Priority = Lowest

(1) When an SD24 overflow occurs, the user must check all SD24CCTLx SD24OVIFG flags to determine which channel overflowed.

13.3.31 SD24TRIM Register (address = 00BFh) [reset = 0Ch]

SD24 TRIM Register

Figure 13-44. SD24TRIM Register

7	6	5	4	3	2	1	0
Reserved			SD24REFDLYTRIM		SD24IBTRIM		
r0	r0	r0	rw-[0]	rw-[1]	rw-[1]	rw-[0]	rw-[0]

Table 13-34. SD24TRIM Register Description

Bit	Field	Type	Reset	Description
7-5	Reserved	R	0h	Reserved. Always reads as 0.
4-3	SD24REFDLYTRIM	RW	0h	These bits are used for trimming the SD24 reference voltage buffer start up delay in the internal reference mode (REFS = 1). These bits are don't care when external reference mode is selected (REFS = 0). These bits are reset on BOR.
2-0	SD24IBTRIM	RW	0h	These bits are used for trimming the SD24 bias current. These bits are reset on BOR.

Tag-Length-Value (TLV) and Start-Up Code (SUC)

This chapter describes the Tag-Length-Value (TLV) and Start-Up Code (SUC) of the MSP430i family.

Topic **Page**

14.1 Tag-Length-Value (TLV)

Tag-Length-Value (TLV) is a device descriptor data structure stored in the flash information memory. It contains information about device and peripheral calibration data. The TLV is organized as different blocks with tag, length, and values specified for each block.

The flash information memory on MSP430i family is organized as one segment of size 1KB that is mapped to the address range of 1000h to 13FFh. The size of TLV is 64 bytes, and it is stored in flash information memory in the address range of 13C0h to 13FFh. The first two bytes of the TLV at addresses 13C0h and 13C1h contain the checksum of the data over the address range of 13C2h to 13FFh (62 bytes). The checksum is calculated using 2s-compliment of a XOR in which two bytes of data are extracted for each intermediate XOR operation. The checksum can be useful for flash integrity check during the execution of start-up code.

The die record block contains information on lot and wafer ID, X and Y position of die on the wafer, and the production test results.

Table 14-1 lists different tags defined for MSP430i family and their description and values.

Table 14-1. TLV Tags

Tag	Description	Value
TAG_EMPTY	Tag to identify an unused memory area	FEh
TAG_DIEREC	Tag for Die Record	01h
TAG_REF	Tag for Shared Reference	02h
TAG_DCO	Tag for DCO	03h
TAG_SD24	Tag for SD24	04h

The Table 14-2 lists the set of peripheral registers to be calibrated on MSP430i family.

Table 14-2. Peripheral Registers for Calibration

Registers	Description	Absolute Address
REFCAL0	Shared Reference calibration-0	62h
REFCAL1	Shared Reference calibration-1	63h
CSIRFCAL	DCO frequency calibration in Internal Resistor mode	52h
CSIRTCAL	DCO temperature calibration in Internal Resistor mode	53h
CSERFCAL	DCO frequency calibration in External Resistor mode	54h
CSERTCAL	DCO temperature calibration in External Resistor mode	55h
SD24TRIM	SD24 trim	BFh

Table 14-3 represents the TLV device descriptor for MSP430i family.

Table 14-3. MSP430i TLV Device Descriptor

	Description	Address	Size (Bytes)	Value
Checksum	TLV checksum	013C0h	2	per unit
Die Record	Die Record Tag	013C2h	1	01h
	Die Record Length	013C3h	1	0Ah
	Lot/Wafer ID	013C4h	4	per unit
	Die X position	013C8h	2	per unit
	Die Y position	013CAh	2	per unit
	Test results	013CCh	2	per unit

Table 14-3. MSP430i TLV Device Descriptor (continued)

	Description	Address	Size (Bytes)	Value
REF Calibration	REF Calibration Tag	013CEh	1	02h
	REF Calibration Length	013CFh	1	02h
	Calibrate REF – for REFCAL1 register	013D0h	1	per unit
	Calibrate REF – for REFCAL0 register	013D1h	1	per unit
DCO Calibration	DCO Calibration Tag	013D2h	1	03h
	DCO Calibration Length	013D3h	1	04h
	Calibrate DCO – for CSIRFCAL register	013D4h	1	per unit
	Calibrate DCO – for CSIRTCAL register	013D5h	1	per unit
	Calibrate DCO – for CSERFCAL register	013D6h	1	per unit
	Calibrate DCO – for CSERTCAL register	013D7h	1	per unit
SD24 Calibration	SD24 Calibration Tag	013D8h	1	04h
	SD24 Calibration Length	013D9h	1	02h
	Calibrate SD24 – for SD24TRIM register	013DAh	1	per unit
	Empty	013DBh	1	FFh
Empty	Tag Empty	013DCh	1	FEh
	Empty Length	013DDh	1	22h
	Empty	013DEh	34	FFh

14.2 Start-Up Code (SUC)

The device security and peripheral calibration are handled through the start-up code on MSP430i devices. The user-provided start-up code needs to be executed each time after a reset (BOR, POR, or PUC). The reset vector at FFFEh must point to start-up code.

NOTE: The start-up code must be included in the application by the user and it is not present on the device by default.

Device Security

- JTAG is disabled after BOR or POR reset. The SUC must set the device security function (the decision to secure or unsecure the device) within the first 64 MCLK clock cycles after a BOR or POR reset.
- Four bytes over the address range of FFDCh to FFDFh (below the device interrupt vector table) are designated for the JTAG disable password.
- The SUC should read the JTAG disable password from these addresses and compare it with the expected value.
- If the value read is different from all zeros or all ones (00000000h or FFFFFFFFh), then the SUC should write A5A5h into the SYSJTAGDIS register. This disables JTAG and secures the device. This step must be completed within 64 MCLK clock cycles after a BOR or POR reset.
- If the value read is either all zeros or all ones (00000000h or FFFFFFFFh), then SUC should not take any action and immediately enter the peripheral calibration routine. JTAG is enabled and the device is unsecured automatically in this case after 64 MCLK clock cycles.
- Having all zeros (00000000h) as JTAG enable password can be useful for unsecuring a previously secured device. The user may set a password to disable JTAG after the application is developed. But the application can be written in such a way that it can receive commands through the serial ports that can overwrite the JTAG password to make it all zeros. At the next reset, the start-up code would see all zeros as the JTAG password, enable JTAG, and unsecure the device.
- The Watchdog module should be stopped at the beginning of start-up code and re-enabled at the end, if it is needed by the application, to avoid any erroneous resets during the execution of start-up code.

NOTE: If the application programs the device to any of the low-power modes within the first 64 MCLK clock cycles after a BOR or POR reset, the device is locked for any JTAG/SBW access.

Peripherals Calibration

- The SUC should check for the BORIFG flag in the SFR IFG1 register. If the BORIFG flag is not set, the whole calibration routine is skipped, and the SUC transfers control to the start of the user main program after re-enabling the watchdog.

  The reason for this step is to perform calibration only after power cycle, which sets the BORIFG flag. There is no need to recalibrate peripherals upon POR or PUC reset, as the peripheral calibration registers are reset only on BOR reset.
- The BORIFG flag is not cleared by POR or PUC reset and hence the start-up code must clear the BORIFG flag after completing the calibration routine. This ensures that the BORIFG flag is seen set during the execution of start-up code only due to a power cycle.
- If the BORIFG flag is set, then the SUC should proceed to the TLV device descriptor integrity check in flash information memory. The SUC should calculate the checksum of the TLV over the address range of 13C2h to 13FFh (62 bytes) and compare the result with the stored checksum at addresses 13C0h and 13C1h.
  - If the checksum does not match, then the SUC should program the device into LPM4 mode.
  - If the checksum matches, then the SUC should proceed with peripheral calibration.
- The SUC must follow a defined order for peripheral calibration.
  1. The shared reference (REF) should be calibrated. Both REFCAL1 and REFCAL0 registers must be written with calibration values fromthe TLV. The order must be REFCAL1 write followed by REFCAL0. The writes can be back to back, and no explicit delay is necessary in the code.
  2. The DCO must be calibrated. The DCO calibration programming can start right after REF calibration. There is no explicit delay necessary in the code between REF calibration and DCO calibration to allow for REF settling.

     To operate DCO in internal resistor mode, only CSIRFCAL and CSIRTCAL registers need to be written with appropriate calibration values.

     To operate DCO in external resistor mode, then DCOR bit must be set to 1 in the CSCTL0 register. Then the CSIRFCAL, CSIRTCAL, CSERFCAL, and CSERTCAL registers must be loaded with calibration values.

     Programming CSIRFCAL and CSIRTCAL registers even in DCO external resistor mode allows the DCO to generate a 16.384-MHz clock upon external resistor fault conditions, when the DCO switches to internal resistor mode as fail-safe mechanism.

     Note: When DCO external resistor mode is selected (DCOR: 0 → 1) or when the CSIRFCAL or CSERFCAL registers are programmed, the DCO clock stops for approximately 40 µs (typical) and resumes automatically. Refer to the *Clock System* chapter for more details.
  3. The SUC should then calibrate the SD24 by writing the calibration value into the SD24TRIM register.
- The SUC must clear the BORIFG flag, re-enable watchdog if needed by the application, and jump to the start of user application code.

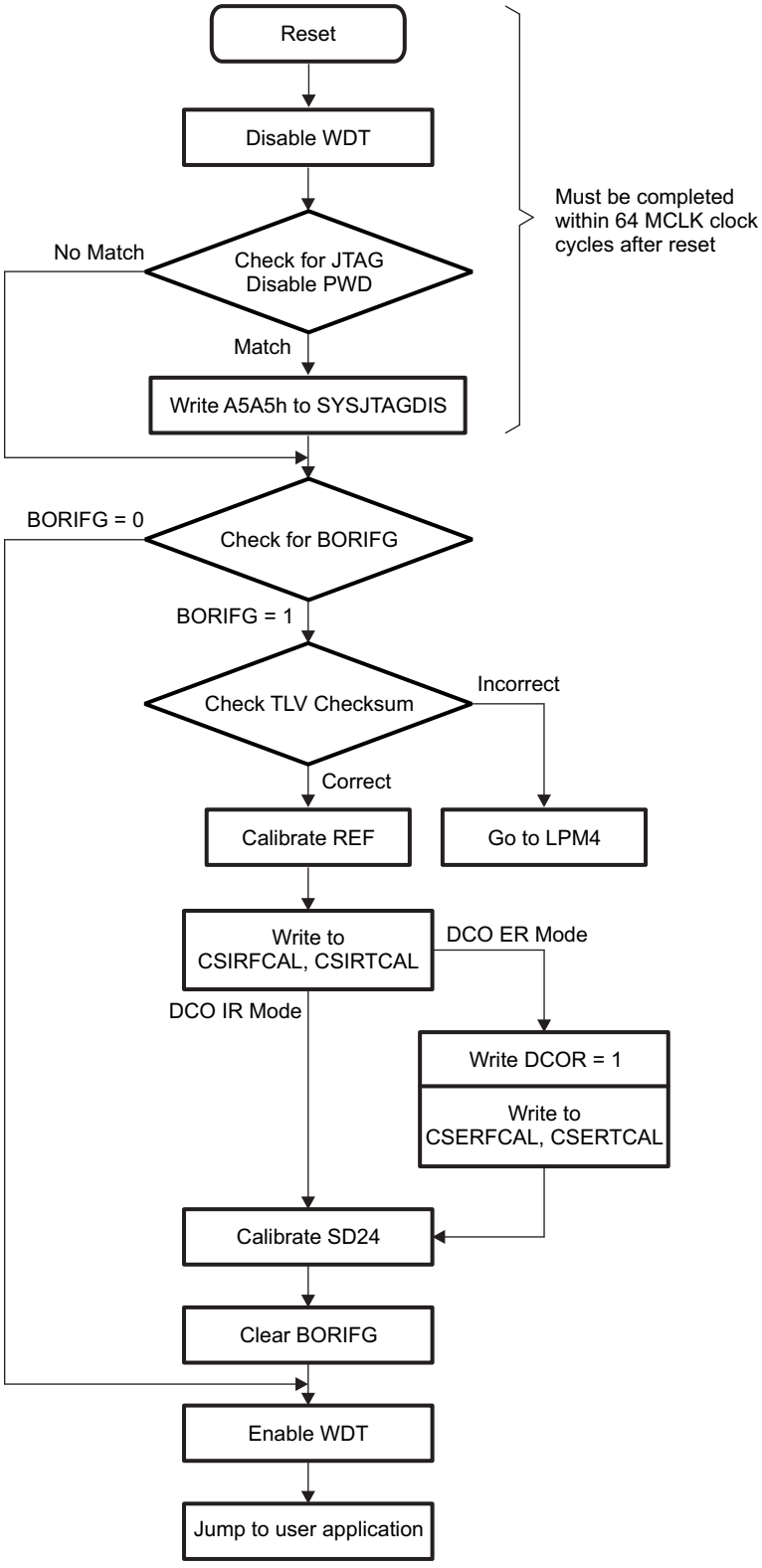


Figure 14-1. Start-Up Code Flow Chart

Embedded Emulation Module (EEM)

This chapter describes the Embedded Emulation Module (EEM) that is implemented in all MSP430i flash devices.

Topic **Page**

15.1 EEM Introduction

Every MSP430 flash-based microcontroller implements an embedded emulation module (EEM). It is accessed and controlled through JTAG. Each implementation is device dependent and is described in the device-specific data sheet.

In general, the following features are available:

- Non-intrusive code execution with real-time breakpoint control
- Single step, step into and step over functionality
- Full support of all low-power modes
- Support for all system frequencies, for all clock sources
- Up to eight (device dependent) hardware triggers/breakpoints on memory address bus (MAB) or memory data bus (MDB)
- Up to two (device dependent) hardware triggers/breakpoints on CPU register write accesses
- MAB, MDB ,and CPU register access triggers can be combined to form up to eight (device dependent) complex triggers/breakpoints
- Trigger sequencing (device dependent)
- Storage of internal bus and control signals using an integrated trace buffer (device dependent)
- Clock control for timers, communication peripherals, and other modules on a global device level or on a per-module basis during an emulation stop

Figure 15-1 shows a simplified block diagram of the largest currently available EEM implementation.

For more details on how the features of the EEM can be used together with the IAR Embedded Workbench™ debugger see the application report *Advanced Debugging Using the Enhanced Emulation Module* (SLAA263) at www.msp430.com. Code Composer Studio (CCS) and most other debuggers supporting MSP430 have the same or a similar feature set. For details see the user's guide of the applicable debugger.

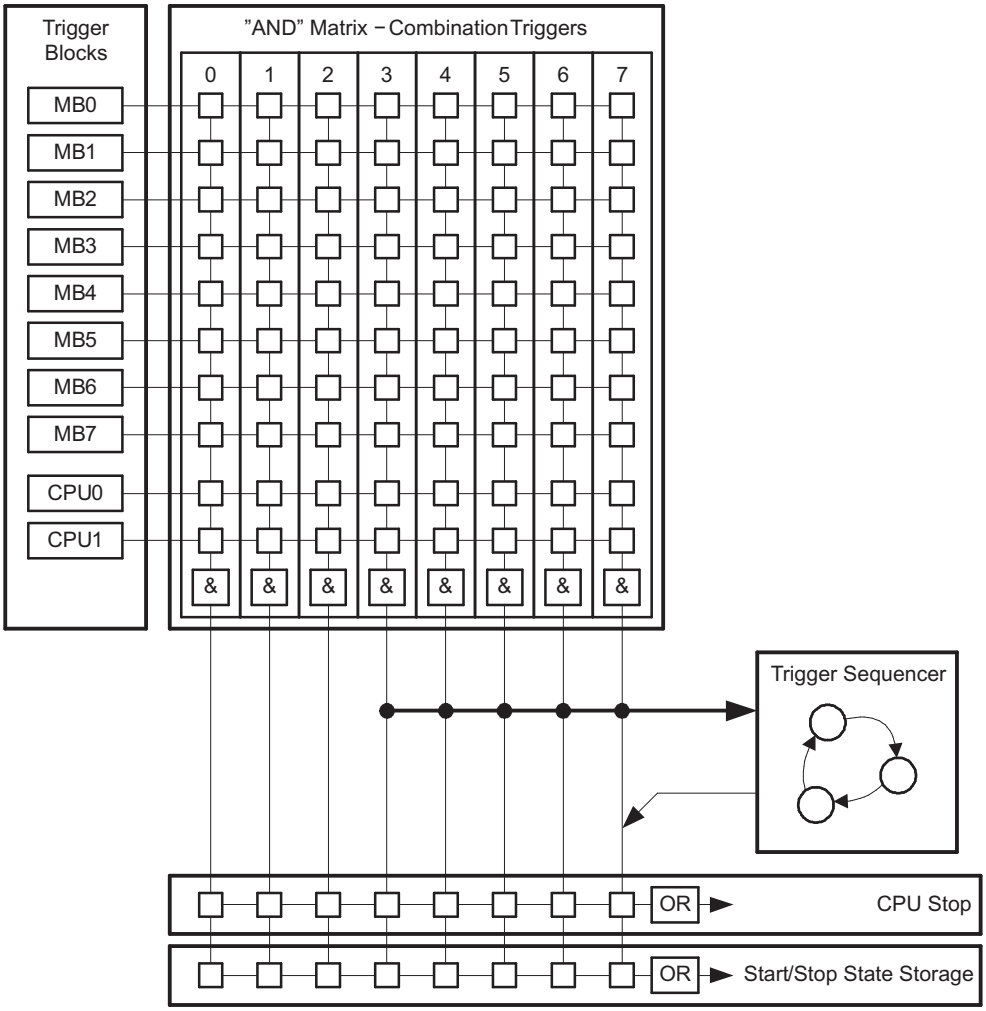


Figure 15-1. Large Implementation of the Embedded Emulation Module (EEM)

15.2 EEM Building Blocks

15.2.1 Triggers

The event control in the EEM of the MSP430i system consists of triggers, which are internal signals indicating that a certain event has happened. These triggers may be used as simple breakpoints, but it is also possible to combine two or more triggers to allow detection of complex events and trigger various reactions besides stopping the CPU.

In general, the triggers can be used to control the following functional blocks of the EEM:

- Breakpoints (CPU stop)
- State storage
- Sequencer

There are two different types of triggers, the memory trigger and the CPU register write trigger.

Each memory trigger block can be independently selected to compare either the MAB or the MDB with a given value. Depending on the implemented EEM the comparison can be =, ≠, ≥, or ≤. The comparison can also be limited to certain bits with the use of a mask. The mask is either bit-wise or byte-wise, depending upon the device. In addition to selecting the bus and the comparison, the condition under which the trigger is active can be selected. The conditions include read access, write access, DMA access, and instruction fetch.

Each CPU register write trigger block can be independently selected to compare what is written into a selected register with a given value. The observed register can be selected for each trigger independently. The comparison can be =, ≠, ≥, or ≤. The comparison can also be limited to certain bits with the use of a bit mask.

Both types of triggers can be combined to form more complex triggers. For example, a complex trigger can signal when a particular value is written into a user-specified address.

15.2.2 Trigger Sequencer

The trigger sequencer allows the definition of a certain sequence of trigger signals before an event is accepted for a break or state storage event. Within the trigger sequencer, it is possible to use the following features:

- Four states (State 0 to State 3)
- Two transitions per state to any other state
- Reset trigger that resets the sequencer to State 0.

The Trigger sequencer always starts at State 0 and must execute to State 3 to generate an action. If State 1 or State 2 are not required, they can be bypassed.

15.2.3 State Storage (Internal Trace Buffer)

The state storage function uses a built-in buffer to store MAB, MDB, and CPU control signal information (that is, read, write, or instruction fetch) in a non-intrusive manner. The built-in buffer can hold up to eight entries. The flexible configuration allows the user to record the information of interest very efficiently.

15.2.4 Clock Control

The EEM provides device dependent flexible clock control. This is useful in applications where a running clock is needed for peripherals after the CPU is stopped (for example, to allow a UART module to complete its transfer of a character or to allow a timer to continue generating a PWM signal).

The clock control is flexible and supports both modules that need a running clock and modules that must be stopped when the CPU is stopped due to a breakpoint.

15.3 EEM Configurations

Table 15-1 gives an overview of the EEM configurations in the MSP430i family. The implemented configuration is device dependent - see the device data sheet.

Table 15-1. EEM Configurations

Feature	XS	S	M	L
Memory Bus Triggers	2(=, ≠ only)	3	5	8
Memory Bus Trigger Mask for	1) Low byte 2) High byte	1) Low byte 2) High byte	1) Low byte 2) High byte	All 16 or 20 bits
CPU Register-Write Triggers	0	1	1	2
Combination Triggers	2	4	6	8
Sequencer	No	No	Yes	Yes
State Storage	No	No	No	Yes

In general the following features can be found on any MSP430i device:

- At least two MAB/MDB triggers supporting:
  - Distinction between CPU, DMA, read, and write accesses
  - =, ≠, ≥, or ≤ comparison (in XS only =, ≠)
- At least two trigger Combination registers
- Hardware breakpoints using the CPU Stop reaction
- Clock control with individual control of module clocks (in some XS configurations the control of module clocks is hardwired)

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not 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 relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265