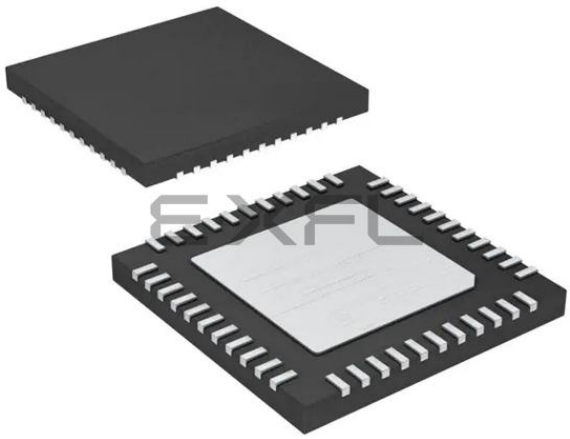


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	64MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, HLVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 35x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VQFN Exposed Pad
Supplier Device Package	44-QFN (8x8)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f46k42-e-ml

1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F26K42
- PIC18F27K42
- PIC18F45K42
- PIC18F46K42
- PIC18F47K42
- PIC18F55K42
- PIC18F56K42
- PIC18F57K42
- PIC18LF26K42
- PIC18LF27K42
- PIC18LF45K42
- PIC18LF46K42
- PIC18LF47K42
- PIC18LF55K42
- PIC18LF56K42
- PIC18LF57K42

- This family offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high-endurance Program Flash Memory, Universal Asynchronous Receiver Transmitter (UART), Serial Peripheral Interface (SPI), Inter-integrated Circuit (I^2C), Direct Memory Access (DMA), Configurable Logic Cells (CLC), Signal Measurement Timer (SMT), Numerically Controlled Oscillator (NCO), and Analog-to-Digital Converter with Computation (ADC^2).

1.1 New Features

- **Direct Memory Access Controller:** The Direct Memory Access (DMA) Controller is designed to service data transfers between different memory regions directly without intervention from the CPU. By eliminating the need for CPU-intensive management of handling interrupts intended for data transfers, the CPU now can spend more time on other tasks.
- **Vectored Interrupt Controller:** The Vectored Interrupt Controller module reduces the numerous peripheral interrupt request signals to a single interrupt request signal to the CPU. It assembles all of the interrupt request signals and resolves the interrupts based on both a fixed natural order priority and a user-assigned priority, thereby eliminating scanning of interrupt sources.
- **Universal Asynchronous Receiver Transmitter:** The Universal Asynchronous Receiver Transmitter (UART) module is a serial I/O communications peripheral. It contains all the clock generators, shift registers and data buffers necessary to perform an input or output serial data transfer, independent of device program execution. The UART can be configured as a full-duplex asynchronous system or one of several automated protocols. Full-Duplex mode is useful for communications with peripheral systems, with DMX/DALI/LIN support.
- **Serial Peripheral Interface:** The Serial Peripheral Interface (SPI) module is a synchronous serial data communication bus that operates in Full-Duplex mode. Devices communicate in a master/slave environment where the master device initiates the communication. A slave device is controlled through a Chip Select known as Slave Select. Example slave devices include serial EEPROMs, shift registers, display drivers, A/D converters, or another PIC.
- **I^2C Module:** The I^2C module provides a synchronous interface between the microcontroller and other I^2C-compatible devices using the two-wire I^2C serial bus. Devices communicate in a master/slave environment. The I^2C bus specifies two signal connections - Serial Clock (SCL) and Serial Data (SDA). Both the SCL and SDA connections are bidirectional open-drain lines, each requiring pull-up resistors to the supply voltage.
- **12-bit A/D Converter with Computation:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period and thus, reduces code overhead. It has a new module called ADC^2 with computation features, which provides a digital filter and threshold interrupt functions.

1.2 Details on Individual Family Members

Devices in the PIC18(L)F26/27/45/46/47/55/56/57K42 family are available in 28-pin and 40/44/48-pin packages. The block diagram for this device is shown in Figure 3-1.

The similarities and differences among the devices are listed in the PIC18(L)F2X/4X/5XK42 Family Types Table (page 4). The pinouts for all devices are listed in Table 1.

4.2.4 PROGRAM COUNTER

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21-bit wide and is contained in three separate 8-bit registers. The low byte, known as the PCL register, is both readable and writable. The high byte, or PCH register, contains the PC<15:8> bits; it is not directly readable or writable. Updates to the PCH register are performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits; it is also not directly readable or writable. Updates to the PCU register are performed through the PCLATU register.

The contents of PCLATH and PCLATU are transferred to the program counter by any operation that writes PCL. Similarly, the upper two bytes of the program counter are transferred to PCLATH and PCLATU by any operation that reads PCL. This is useful for computed offsets to the PC (see **Section 4.3.2.1 "Computed GOTO"**).

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the Least Significant bit of PCL is fixed to a value of '0'. The PC increments by two to address sequential instructions in the program memory.

The CALL, RCALL, GOTO and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

4.2.5 RETURN ADDRESS STACK

The return address stack allows any combination of up to 31 program calls and interrupts to occur. The PC is pushed onto the stack when a CALL or RCALL instruction is executed or an interrupt is acknowledged. The PC value is pulled off the stack on a RETURN, RETLW or a RETFIE instruction. PCLATU and PCLATH are not affected by any of the RETURN or CALL instructions.

The stack operates as a 31-word by 21-bit RAM and a 5-bit Stack Pointer. The stack space is not part of either program or data space. The Stack Pointer is readable and writable and the address on the top of the stack is readable and writable through the Top-of-Stack (TOS) Special File Registers. Data can also be pushed to, or popped from the stack, using these registers.

A CALL, CALLW or RCALL instruction causes a push onto the stack; the Stack Pointer is first incremented and the location pointed to by the Stack Pointer is written with the contents of the PC (already pointing to the instruction following the CALL). A RETURN type instruction causes a pop from the stack; the contents of the location pointed to by the STKPTR are transferred to the PC and then the Stack Pointer is decremented.

The Stack Pointer is initialized to '00000' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '00000'; this is only a Reset value. Status bits in the PCON0 register indicate if the stack has overflowed or underflowed.

4.2.5.1 Top-of-Stack Access

Only the top of the return address stack (TOS) is readable and writable. A set of three registers, TOSU:TOSH:TOSL, holds the contents of the stack location pointed to by the STKPTR register (Figure 4-1). This allows users to implement a software stack, if necessary. After a CALL, RCALL or interrupt, the software can read the pushed value by reading the TOSU:TOSH:TOSL registers. These values can be placed on a user-defined software stack. At return time, the software can return these values to TOSU:TOSH:TOSL and do a return.

The user must disable the Global Interrupt Enable (GIE) bits while accessing the stack to prevent inadvertent stack corruption.

4.4.2 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as either two bytes or four bytes in program memory. The Least Significant Byte of an instruction word is always stored in a program memory location with an even address (LSb = 0). To maintain alignment with instruction boundaries, the PC increments in steps of two and the LSb will always read '0' (see **Section 4.2.4 "Program Counter"**).

Figure 4-2 shows an example of how instruction words are stored in the program memory.

The CALL and GOTO instructions have the absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1>, which accesses the desired byte address in program memory. Instruction #2 in Figure 4-2 shows how the instruction GOTO 0006h is encoded in the program memory. Program branch instructions, which encode a relative address offset, operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. **Section 41.0 "Instruction Set Summary"** provides further details of the instruction set.

4.4.3 MULTI-WORD INSTRUCTIONS

The standard PIC18 instruction set has four two-word instructions: CALL, MOVFF, GOTO and LFSR and two three-word instructions: MOVFFL and MOVSFL. In all cases, the second and the third word of the instruction always has '1111' as its four Most Significant bits; the other 12 bits are literal data, usually a data memory address.

The use of '1111' in the four MSbs of an instruction specifies a special form of NOP. If the instruction is executed in proper sequence – immediately after the first word – the data in the second word is accessed and used by the instruction sequence. If the first word is skipped for some reason and the second or third word is executed by itself, a NOP is executed instead. This is necessary for cases when the multi-word instruction is preceded by a conditional instruction that changes the PC. Example 4-4 shows how this works.

FIGURE 4-2: INSTRUCTIONS IN PROGRAM MEMORY

Program Memory Byte Locations →

Instruction 1: MOVLW 055h
Instruction 2: GOTO 0006h
Instruction 3: MOVFF 123h, 456h
Instruction 4: MOVFFL 123h, 456h

LSB = 1	LSB = 0	Word Address ↓
		000000h
		000002h
		000004h
		000006h
0Fh	55h	000008h
EFh	03h	00000Ah
F0h	00h	00000Ch
C1h	23h	00000Eh
F4h	56h	000010h
00h	60h	000012h
F4h	8Ch	000014h
F4h	56h	000016h
		000018h
		00001Ah

REGISTER 7-7: OSCEN: OSCILLATOR MANUAL ENABLE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0
EXTOEN	HFOEN	MFOEN	LFOEN	SOSCEN	ADOEN	—	—

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7 **EXTOEN:** External Oscillator Manual Request Enable bit
1 = EXTOSC is explicitly enabled, operating as specified by FEXTOSC
0 = EXTOSC could be enabled by requesting peripheral

bit 6 **HFOEN:** HFINTOSC Oscillator Manual Request Enable bit
1 = HFINTOSC is explicitly enabled, operating as specified by OSCFRQ (Register 7-5)
0 = HFINTOSC could be enabled by requesting peripheral

bit 5 **MFOEN:** MFINTOSC (500 kHz/31.25 kHz) Oscillator Manual Request Enable bit (Derived from HFINTOSC)
1 = MFINTOSC is explicitly enabled
0 = MFINTOSC could be enabled by requesting peripheral

bit 4 **LFOEN:** LFINTOSC (31 kHz) Oscillator Manual Request Enable bit
1 = LFINTOSC is explicitly enabled
0 = LFINTOSC could be enabled by requesting peripheral

bit 3 **SOSCEN:** Secondary Oscillator Manual Request Enable bit
1 = Secondary Oscillator is explicitly enabled, operating as specified by SOSCPWR
0 = Secondary Oscillator could be enabled by requesting peripheral

bit 2 **ADOEN:** ADC Oscillator Manual Request Enable bit
1 = ADC oscillator is explicitly enabled
0 = ADC oscillator could be enabled by requesting peripheral

bit 1-0 **Unimplemented:** Read as '0'

13.3.5 WRITE VERIFY

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

EXAMPLE 13-5: DATA EEPROM READ

```
; Data Memory Address to read
            CLRF      NVMCON1          ; Setup Data EEPROM Access
            MOVF      EE_ADDRL, W      ;
            MOVWF     NVMADRL          ; Setup Address
            BSF       NVMCON1, RD      ; Issue EE Read
            MOVF      NVMDAT, W        ; W = EE_DATA
```

EXAMPLE 13-6: DATA EEPROM WRITE

```
; Data Memory Address to write
            CLRF      NVMCON1          ; Setup Data EEPROM Access
            MOVF      EE_ADDRL, W      ;
            MOVWF     NVMADRL          ; Setup Address
; Data Memory Value to write
            MOVF      EE_DATA, W       ;
            MOVWF     NVMDAT           ;
; Enable writes
            BSF       NVMCON1, WREN    ;
; Disable interrupts
            BCF       INTCON0, GIE     ;
; Required unlock sequence
            MOVLW     55h              ;
            MOVWF     NVMCON2          ;
            MOVLW     AAh              ;
            MOVWF     NVMCON2          ;
; Set WR bit to begin write
            BSF       NVMCON1, WR      ;
; Enable INT
            BSF       INTCON0, GIE     ;
; Wait for interrupt, write done
            SLEEP                      ;
; Disable writes
            BCF       NVMCON1, WREN    ;
```

13.3.6 OPERATION DURING CODE-PROTECT

Data EEPROM Memory has its own code-protect bits in Configuration Words. External read and write operations are disabled if code protection is enabled.

If the Data EEPROM is write-protected or if NVMADR points an invalid address location, the WR bit is cleared without any effect. WRERR is signaled in this scenario.

13.3.7 PROTECTION AGAINST SPURIOUS WRITE

There are conditions when the user may not want to write to the Data EEPROM Memory. To protect against spurious EEPROM writes, various mechanisms have been implemented. On power-up, the WREN bit is cleared. In addition, writes to the EEPROM are blocked during the Power-up Timer period (TPWRT).

The unlock sequence and the WREN bit together help prevent an accidental write during brown-out, power glitch or software malfunction.

TABLE 14-1: SCANNER OPERATING MODES(1)

TRIGEN	BURSTMD	Scanner Operation
0	0	Memory access is requested when the CRC module is ready to accept data; the request is granted if no other higher priority source request is pending.
1	0	Memory access is requested when the CRC module is ready to accept data and trigger selection is true; the request is granted if no other higher priority source request is pending.
x	1	Memory access is always requested, the request is granted if no other higher priority source request is pending.

Note 1: See **Section 3.1 "System Arbitration"** for Priority selection and **Section 3.2 "Memory Access Scheme"** for Memory Access Scheme.

REGISTER 14-12: SCANLADRU: SCAN LOW ADDRESS UPPER BYTE REGISTER

U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	LADR<21:16>(1,2)					
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **LADR<21:16>:** Scan Start/Current Address bits(1,2)
Upper bits of the current address to be fetched from, value increments on each fetch of memory.

Note 1: Registers SCANLADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).
2: While SGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 14-13: SCANLADRH: SCAN LOW ADDRESS HIGH BYTE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
LADR<15:8>(1, 2)							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-0 **LADR<15:8>**: Scan Start/Current Address bits(1, 2)
Most Significant bits of the current address to be fetched from, value increments on each fetch of memory.

Note 1: Registers SCANLADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).
2: While SGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 14-14: SCANLADRL: SCAN LOW ADDRESS LOW BYTE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
LADR<7:0>(1, 2)							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-0 **LADR<7:0>**: Scan Start/Current Address bits(1, 2)
Least Significant bits of the current address to be fetched from, value increments on each fetch of memory

Note 1: Registers SCANLADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).
2: While SGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 14-15: SCANHADRU: SCAN HIGH ADDRESS UPPER BYTE REGISTER

U-0	U-0	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
—	—	HADR<21:16>					
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **HADR<21:16>:** Scan End Address bits(1, 2)
Upper bits of the address at the end of the designated scan

Note 1: Registers SCANHADRU/H/L form a 22-bit value but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).
2: While SGO = 1 (SCANCON0 register), writing to this register is ignored.

20.3 Programmable Prescaler

A software programmable prescaler is available for exclusive use with Timer0. There are 16 prescaler options for Timer0 ranging in powers of two from 1:1 to 1:32768. The prescaler values are selected using the CKPS<3:0> bits of the T0CON1 register.

The prescaler is not directly readable or writable. Clearing the prescaler register can be done by writing to the TMR0L register or to the T0CON0/T0CON1 register or by any Reset.

20.4 Programmable Postscaler

A software programmable postscaler (output divider) is available for exclusive use with Timer0. There are 16 postscaler options for Timer0 ranging from 1:1 to 1:16. The postscaler values are selected using the OUTPS bits of the T0CON0 register.

The postscaler is not directly readable or writable. Clearing the postscaler register can be done by writing to the TMR0L register or to the T0CON0/T0CON1 register or by any Reset.

20.5 Operation During Sleep

When operating synchronously, Timer0 will halt. When operating asynchronously, Timer0 will continue to increment and wake the device from Sleep (if Timer0 interrupts are enabled) provided that the input clock source is active.

20.6 Timer0 Interrupts

The Timer0 interrupt flag bit (TMR0IF) is set when either of the following conditions occur:

- 8-bit TMR0L matches the TMR0H value
- 16-bit TMR0 rolls over from ‘FFFFh’

When the postscaler bits (OUTPS) are set to 1:1 operation (no division), the T0IF flag bit will be set with every TMR0 match or rollover. In general, the TMR0IF flag bit will be set every OUTPS +1 matches or rollovers.

If Timer0 interrupts are enabled (TMR0IE bit of the PIE3 register = ‘1’), the CPU will be interrupted and the device may wake from Sleep (see **Section 20.2 “Clock Source Selection”** for more details).

20.7 Timer0 Output

The Timer0 output can be routed to any I/O pin via the RxyPPS output selection register (see **Section 17.0 “Peripheral Pin Select (PPS) Module”** for additional information). The Timer0 output can also be used by other peripherals, such as the auto-conversion trigger of the Analog-to-Digital Converter. Finally, the Timer0 output can be monitored through software via the Timer0 output bit (OUT) of the T0CON0 register (Register 20-1).

TMR0_out will be a pulse of one postscaled clock period when a match occurs between TMR0L and PR0 (Period register for TMR0) in 8-bit mode, or when TMR0 rolls over in 16-bit mode. The Timer0 output is a 50% duty cycle that toggles on each TMR0_out rising clock edge.

TABLE 21-3: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER1/3/5 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page
TxCON	—	—	CKPS<1:0>		—	$\overline{\text{SYNC}}$	RD16	ON	313
TxGCON	GE	GPOL	GTM	GSPM	GO/$\overline{\text{DONE}}$	GVAL	—	—	314
TxCLK	—	—	—	CS<4:0>					315
TxGATE	—	—	—	GSS<4:0>					316
TMRxL	Least Significant Byte of the 16-bit TMR3 Register								317
TMRxH	Holding Register for the Most Significant Byte of the 16-bit TMR3 Register								317

Legend: — = Unimplemented location, read as '0'. Shaded cells are not used by TIMER1/3/5.

TABLE 22-3: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER2

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
TxPR	Timer2 Module Period Register								320*
TxTMR	Holding Register for the 8-bit T2TMR Register								320*
TxCON	ON	CKPS<2:0>			OUTPS<3:0>				338
TxCLK	—	—	—	—	—	CS<2:0>			335
TxRST	—	—	—	—	RSEL<3:0>				336
TxHLT	$\overline{\text{PSYNC}}$	CPOL	$\overline{\text{CSYNC}}$	MODE<4:0>					339

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for Timer2 module.

* Page provides register information.

25.6.6 GATED WINDOWED MEASURE MODE

This mode measures the duty cycle of the SMT1_signal input over a known input window. It does so by incrementing the timer on each pulse of the clock signal while the SMT1_signal input is high, updating the SMT1CPR register and resetting the timer on every rising edge of the SMTWINx input after the first. See Figure 25-12 and Figure 25-13.

FIGURE 25-19: GATED COUNTER MODE REPEAT ACQUISITION TIMING DIAGRAM

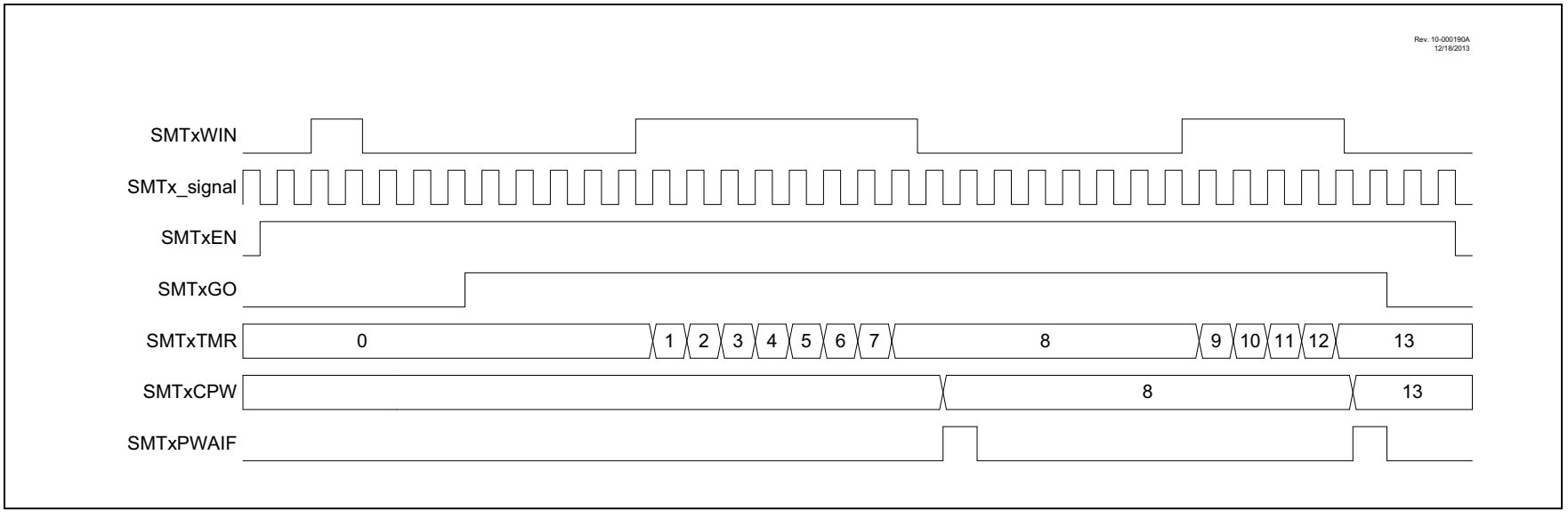
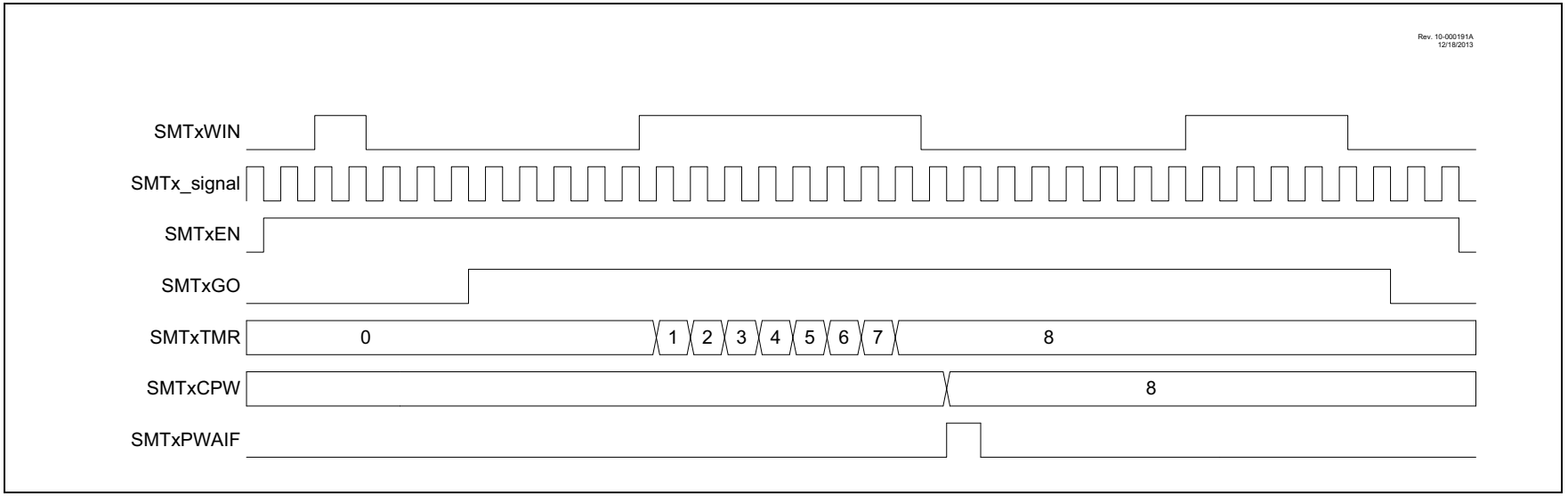


FIGURE 25-20: GATED COUNTER MODE SINGLE ACQUISITION TIMING DIAGRAM



REGISTER 25-16: SMT1PRL: SMT PERIOD REGISTER – LOW BYTE

R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1
SMT1PR<7:0>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **SMT1PR<7:0>**: Significant bits of the SMT Timer Value for Period Match – Low Byte

REGISTER 25-17: SMT1PRH: SMT PERIOD REGISTER – HIGH BYTE

R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1
SMT1PR<15:8>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **SMT1PR<15:8>**: Significant bits of the SMT Timer Value for Period Match – High Byte

REGISTER 25-18: SMT1PRU: SMT PERIOD REGISTER – UPPER BYTE

R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1
SMT1PR<23:16>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **SMT1PR<23:16>**: Significant bits of the SMT Timer Value for Period Match – Upper Byte

31.2.1.8 Asynchronous Transmission Setup

1. Initialize the UxBRGH, UxBRGL register pair and the BRGS bit to achieve the desired baud rate (see **Section 31.17 "UART Baud Rate Generator (BRG)"**).
2. Set the MODE<3:0> bits to the desired asynchronous mode.
3. Set TXPOL bit if inverted TX output is desired.
4. Enable the asynchronous serial port by setting the ON bit.
5. Enable the transmitter by setting the TXEN control bit. This will cause the UxTXIF interrupt flag to be set.
6. If the device has PPS, configure the desired I/O pin RxyPPS register with the code for TX output.
7. If interrupts are desired, set the UxTXIE interrupt enable bit in the respective PIE register. An interrupt will occur immediately provided that the GIE bits in the INTCON0 register are also set.
8. Write one byte of data into the UxTXB register. This will start the transmission.
9. Subsequent bytes may be written when the UxTXIF bit is '1'.

FIGURE 31-3: ASYNCHRONOUS TRANSMISSION

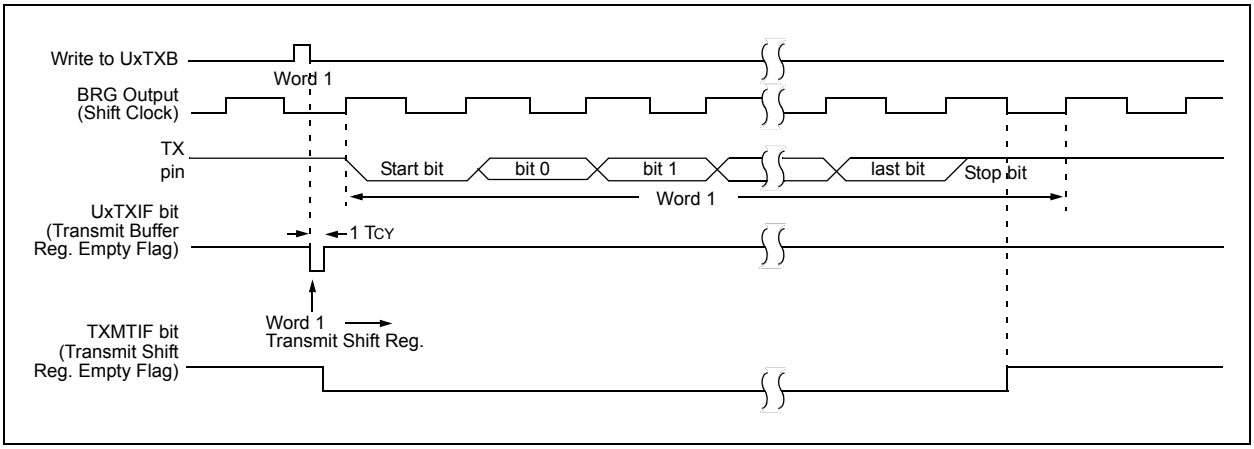
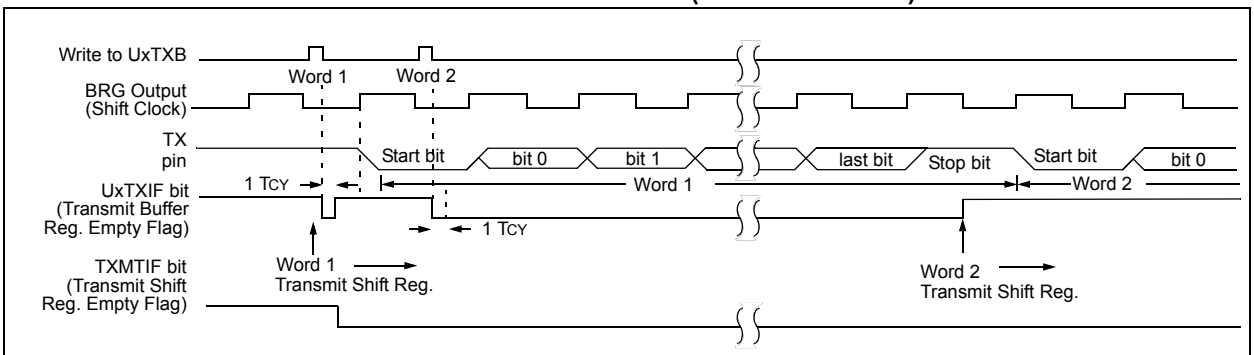


FIGURE 31-4: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)



Note: This timing diagram shows the first transmission and the start of the second consecutive transmission.

31.4 DMX Mode (UART1 only)

DMX is a protocol used in stage and show equipment. This includes lighting, fog machines, motors, etc. The protocol consists of a controller that sends out commands, and receiver such as theater lights that receive these commands. DMX protocol is usually unidirectional, but can be a bidirectional protocol in either Half or Full-duplex modes. An example of Half-duplex mode is the RDM (Remote Device Management) protocol that sits on DMX512A. The controller transmits commands and the receiver receives them. Also there are no error conditions or re-transmit mechanisms.

DMX, or DMX512A as it is known, consists of a “Universe” of 512 channels. This means that one controller can output up to 512 bytes on a single DMX link. Each equipment on the line is programmed to listen to a consecutive sequence of one or more of these bytes.

For example, a fog machine connected to one of the universes may be programmed to receive one byte, starting at byte number 10, and a lighting unit may be programmed to receive four bytes starting at byte number 22.

31.4.1 DMX CONTROLLER

DMX Controller mode is configured with the following settings:

- MODE<3:0> = 1010
- TXEN = 1
- RXEN = 0
- TXPOL = 0
- UxP1 = One less than the number of bytes to transmit (excluding the Start code)
- UxBRGH:L = Value to achieve 250K baud rate
- STP<1:0> = 10 for 2 Stop bits
- RxyPPS = TX pin output code
- ON = 1

Each DMX transmission begins with a Break followed by a byte called the ‘Start Code’. The width of the BREAK is fixed at 25 bit times. The Break is followed by a “Mark After Break” (MAB) Idle period. After this Idle period, the 1st through ‘n’th byte is transmitted, where ‘n-1’ is the value in UxP1. See Figure 31-6.

Software sends the Start Code and the ‘n’ data bytes by writing the UxTXB register with each byte to be sent in the desired order. A UxTXIF value of ‘1’ indicates when the UxTXB is ready to accept the next byte.

The internal byte counter is not accessible to software. Software needs to keep track of the number of bytes written to UxTXB to ensure that no more and no less than ‘n’ bytes are sent because the DMX state machine will automatically insert a Break and reset its internal counter after ‘n’ bytes are written. One way to ensure synchronization between hardware and software is to toggle TXEN after the last byte of the universe is completely free of the transmit shift register as indicated by the TXMTIF bit.

31.4.2 DMX RECEIVER

DMX Receiver mode is configured with the following settings:

- MODE<3:0> = 1010
- TXEN = 0
- RXEN = 1
- RXPOL = 0
- UxP2 = number of first byte to receive
- UxP3 = number of last byte to receive
- UxBRGH:L = Value to achieve 250K baud rate
- STP<1:0> = 10 for 2 Stop bits
- ON = 1
- UxRXPPS = code for desired input pin
- Input pin ANSEL bit = 0

When configured as DMX Receiver, the UART listens for a Break character that is at least 23 bit periods wide. If the Break is shorter than 23 bit times, the Break is ignored and the DMX state machine remains in Idle mode. Upon receiving the Break, the DMX counters will be reset to align with the incoming data stream. Immediately after the Break, the UART will see the “Mark after Break” (MAB). This space is ignored by the UART. The Start Code follows the MAB and will always be stored in the receive FIFO.

After the Start Code, the 1st through 512th byte will be received, but not all of them are stored in the receive FIFO. The UART ignores all received bytes until the ones of interest are received. This is done using the UxP2 and UxP3 registers. The UxP2 register holds the value of the byte number to start the receive process. The byte counter starts at 0 for the first byte after the Start Code. For example, to receive four bytes starting at the 10th byte after the Start Code, write 009h (9 decimal) to UxP2H:L and 00Ch (12 decimal) to UxP3H:L. The receive FIFO is only 2 bytes deep, therefore the bytes must be retrieved by reading UxRXB as they come in to avoid a receive FIFO overrun condition.

Typically two Stop bits are inserted between bytes. If either Stop bit is detected as a ‘0’ then the framing error for that byte will be set.

Since the DMX sequence always starts with a Break, the software can verify that it is in sync with the sequence by monitoring the RXBKIF flag to ensure that the next byte received after the RXBKIF is processed as the Start Code and subsequent bytes are processed as the expected data.

32.5.6 MASTER MODE SPI CLOCK CONFIGURATION

32.5.6.1 SPI Clock Selection

The clock source for SPI master modes is selected by the SPIxCLK register. Selections include the following:

- FOSC
- HFINTOSC
- CLKREF
- Timer0_overflow
- Timer2_Postscaled
- Timer4_Postscaled
- Timer6_Postscaled
- SMT_match

The SPIxBAUD register allows for dividing this clock. The frequency of the SCK output is defined by Equation 32-1:

EQUATION 32-1: FREQUENCY OF SCK OUTPUT SIGNAL

$$F_{BAUD} = \frac{F_{CSEL}}{(2 \cdot (BAUD + 1))}$$

where FBAUD is the baud rate frequency output on the SCK pin, FCSEL is the frequency of the input clock selected by the SPIxCLK register, and BAUD is the value contained in the SPIxBAUD register.

32.5.6.2 CKE, CKP and SMP

The CKP, CKE, and SMP bits control the relationship between the SCK clock output, SDO output data changes, and SDI input data sampling. The bit functions are as follows:

- CKP - SCK output polarity
- CKE - SDO output change relative to the SCK clock
- SMP - SDI input sampling relative to the clock edges

The CKE bit, when set, inverts the low Idle state of the SCK output to a high Idle state.

Figure 32-7 through Figure 32-10 illustrate the eight possible combinations of the CKP, CKE, and SMP bit selections.

When the CKE bit is set, the SDO data is valid before there is a clock edge on SCK. When the CKE bit is cleared, the SDO data is undefined prior to the first SCK edge.

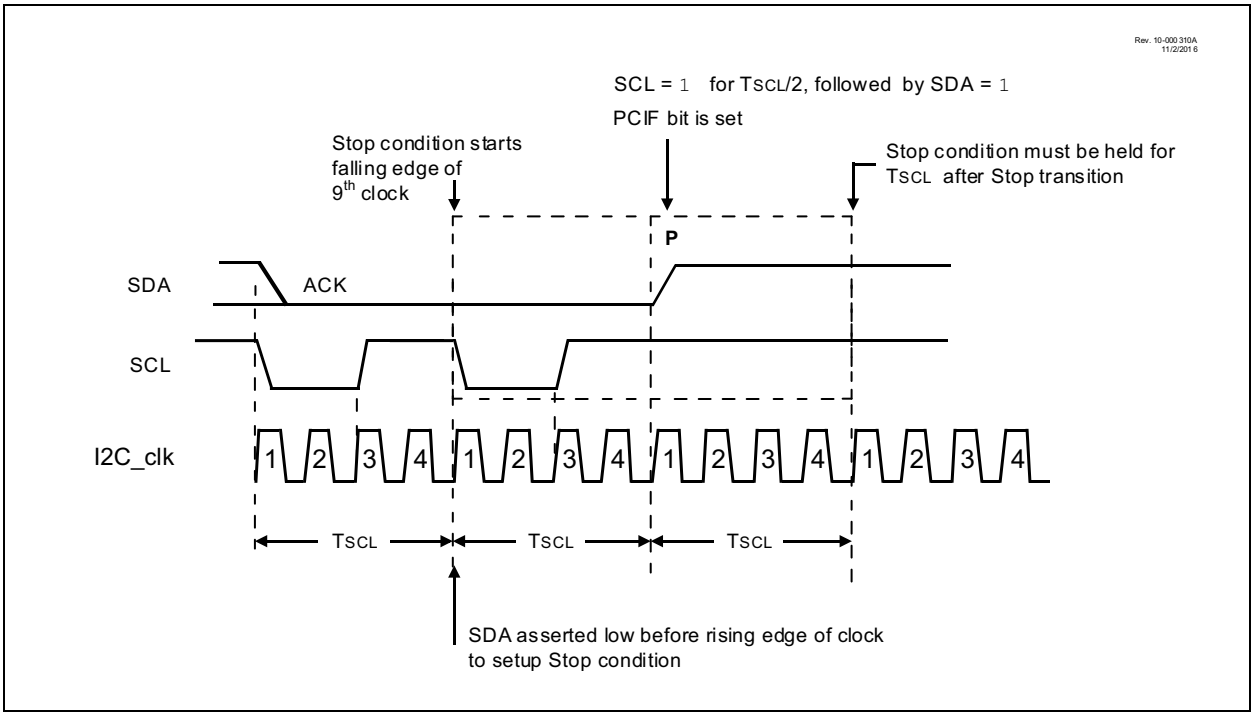
Note:	All timing diagrams assume the LSBF bit of SPIxCON0 is cleared.

TABLE 32-3: SUMMARY OF REGISTERS ASSOCIATED WITH SPI

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
SPIxINTF	SRMTIF	TCZIF	SOSIF	EOSIF	—	RXOIF	TXUIF	—	535
SPIxINTE	SRMTIE	TCZIE	SOSIE	EOSIE	—	RXOIE	TXUIE	—	536
SPIxTCNTH	—	—	—	—	—	TCNT10	TCNT9	TCNT8	537
SPIxTCNTL	TCNT7	TCNT6	TCNT5	TCNT4	TCNT3	TCNT2	TCNT1	TCNT0	536
SPIxTWIDTH	—	—	—	—	—	TWIDTH2	TWIDTH1	TWITDH0	537
SPIxBAUD	BAUD7	BAUD6	BAUD5	BAUD4	BAUD3	BAUD2	BAUD1	BAUD0	538
SPIxCON0	EN	—	—	—	—	LSBF	MST	BMODE	538
SPIxCON1	SMP	CKE	CKP	FST	—	SSP	SDIP	SDOP	539
SPIxCON2	BUSY	SSFLT	—	—	—	SSET	TXR	RXR	540
SPIxSTATUS	TXWE	—	TXBE	—	RXRE	CLRBF	—	RXBF	541
SPIxRXB	RXB7	RXB6	RXB5	RXB4	RXB3	RXB2	RXB1	RXB0	541
SPIxTXB	TXB7	TXB6	TXB5	TXB4	TXB3	TXB2	TXB1	TXB0	542
SPIxCLK	—	—	—	—	CLKSEL3	CLKSEL2	CLKSEL1	CLKSEL0	542

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the SPI module.

FIGURE 33-18: STOP CONDITION DURING RECEIVE OR TRANSMIT



33.5.9 MASTER TRANSMISSION IN 7-BIT ADDRESSING MODE

This section describes the sequence of events for the I^2C module configured as an I^2C master in 7-bit Addressing mode and is transmitting data. Figure 33-19 is used as a visual reference for this description.

1. If ABD = 0; i.e., Address buffers are enabled

Master software loads number of bytes to be transmitted in one sequence in I2CxCNT, slave address in I2CxADB1 with R/W = 0 and the first byte of data in I2CxTXB. Master software has to set the Start (S) bit to initiate communication.

If ABD = 1; i.e., Address buffers are disabled

Master software loads the number of bytes to be transmitted in one sequence in I2CxCNT and the slave address with R/W = 0 into the I2CxTXB register. Writing to the I2CxTXB will assert the start condition on the bus and sets the S bit. Software writes to the S bit are ignored in this case.

2. Master hardware waits for BFRE bit to be set; then shifts out start and address.
3. If the transmit buffer is empty (i.e., TXBE = 1) and I2CxCNT!= 0, the I2CxTXIF and MDR bits are set and the clock is stretched on the 8th falling SCL edge. Clock can be started by loading the next data byte in I2CxTXB register.
4. Master sends out the 9th SCL pulse for ACK.
5. If the Master hardware receives ACK from Slave device, it loads the next byte from the transmit buffer (I2CxTXB) into the shift register and the value of I2CxCNT register is decremented.
6. If a NACK was received, Master hardware asserts Stop or Restart
7. If ABD = 0; i.e., Address buffers are enabled

If I2CxCNT = 0, Master hardware sends Stop or sets MDR if RSEN = 1 and waits for the software to set the Start bit again to issue a restart condition.

If ABD = 1; i.e., Address buffers are disabled

If I2CxCNT = 0, Master hardware sends Stop or sets MDR if RSEN = 1 and waits for the software to write the new address to the I2CxTXB register. Software writes to the S bit are ignored in this case.

8. Master hardware outputs data on SDA.
9. If TXBE = 1 and I2CxCNT! = 0, I2CxTXIF and MDR bits are set and the clock is stretched on 8th falling SCL edge. The user can release the clock by writing the next data byte to I2CxTXB register.
10. Master hardware clocks in ACK from Slave, and loads the next data byte from I2CTXB to the shift register. The value of I2CxCNT is decremented.
11. Go to step 7.

REGISTER 36-18: ADRESH: ADC RESULT REGISTER HIGH, FM = 0

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
ADRES<11:4>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **ADRES<11:4>**: ADC Result Register bits
Upper eight bits of 12-bit conversion result.

REGISTER 36-19: ADRESL: ADC RESULT REGISTER LOW, FM = 0

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	U-0	U-0	U-0	U-0
ADRES<3:0>				—	—	—	—
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-4 **ADRES<3:0>**: ADC Result Register bits. Lower four bits of 12-bit conversion result.
bit 3-0 **Reserved**

REGISTER 38-4: CMxPCH: COMPARATOR x NON-INVERTING CHANNEL SELECT REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	PCH<2:0>		
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-3 **Unimplemented:** Read as '0'

bit 2-0 **PCH<2:0>:** Comparator Non-Inverting Input Channel Select bits

111 = Vss
110 = FVR_Buffer2
101 = DAC_Output
100 = PCH not connected
011 = PCH not connected
010 = PCH not connected
001 = CxIN1+
000 = CxIN0+

REGISTER 38-5: CMOUT: COMPARATOR OUTPUT REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	R-0/0	R-0/0
—	—	—	—	—	—	C2OUT	C1OUT
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-2 **Unimplemented:** Read as '0'

bit 1 **C2OUT:** Mirror copy of C2OUT bit

bit 0 **C1OUT:** Mirror copy of C1OUT bit

TABLE 38-3: SUMMARY OF REGISTERS ASSOCIATED WITH COMPARATOR MODULE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
CMxCON0	EN	OUT	—	POL	—	—	HYS	SYNC	648
CMxCON1	—	—	—	—	—	—	INTP	INTN	649
CMxNCH	—	—	—	—	—	NCH<2:0>			649
CMxPCH	—	—	—	—	—	PCH<2:0>			650
CMOUT	—	—	—	—	—	—	C2OUT	C1OUT	650

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the comparator module.