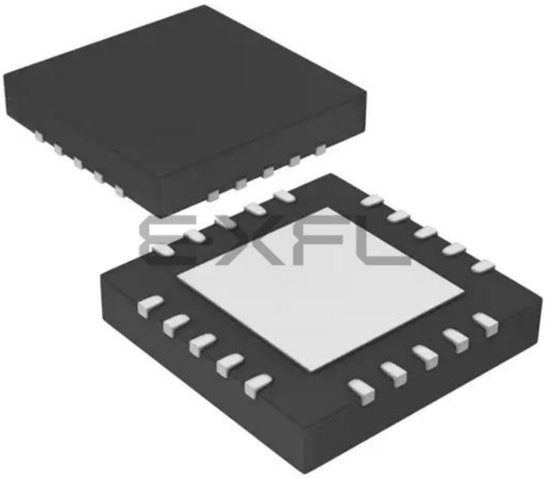


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

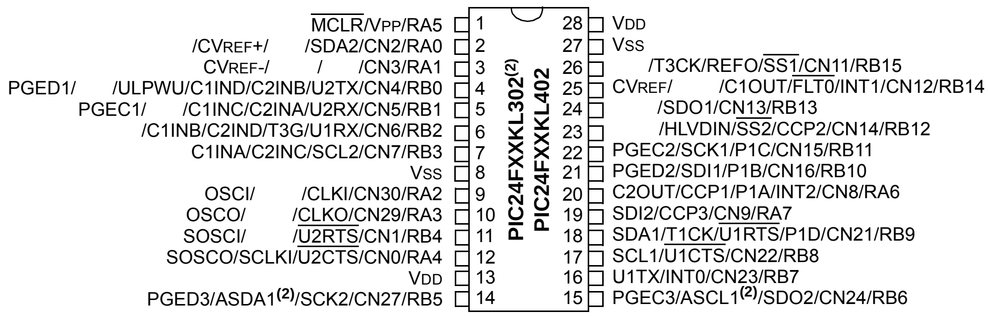
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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"

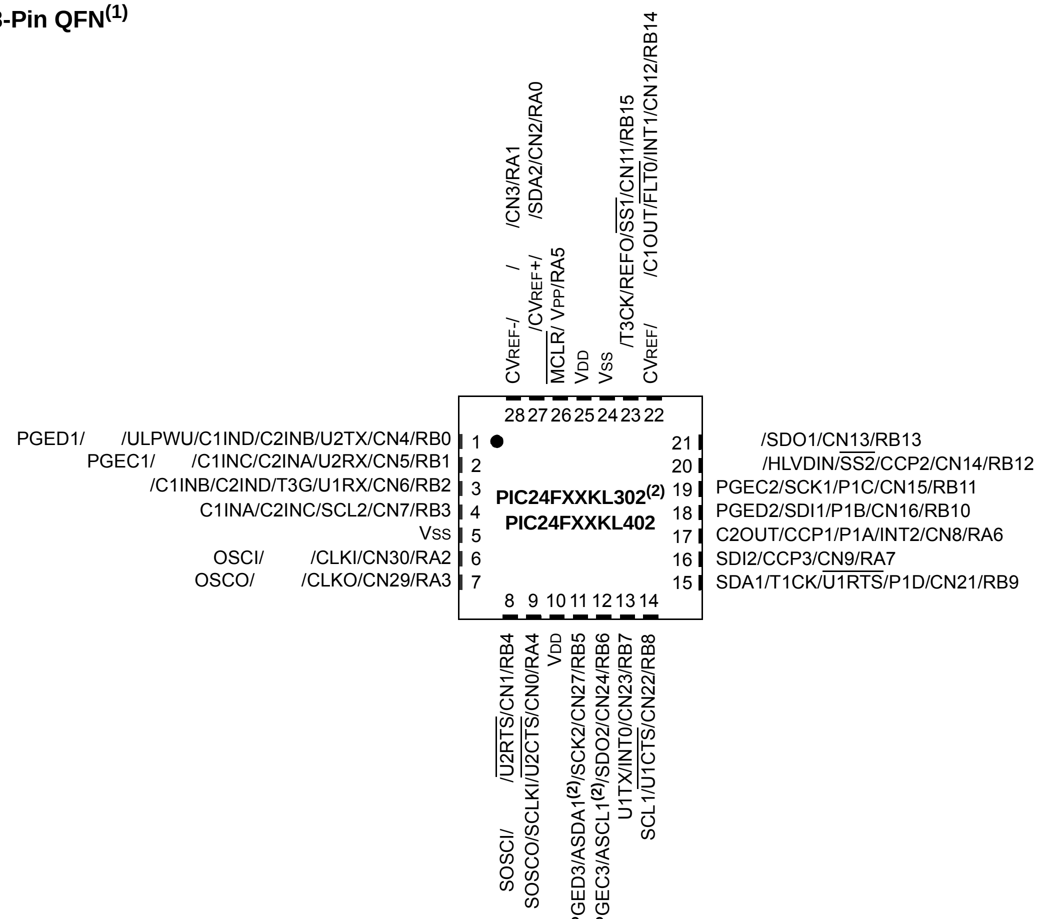
Details	
Product Status	Active
Core Processor	PIC
Core Size	16-Bit
Speed	32MHz
Connectivity	I²C, IrDA, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	18
Program Memory Size	8KB (2.75K x 24)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-VQFN Exposed Pad
Supplier Device Package	20-VQFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic24f08kl401-i-mq

Pin Diagrams: PIC24FXXKL302/402

28-Pin SPDIP/SSOP/SOIC(1)



28-Pin QFN(1)



Contact your Microchip sales team for Chip Scale Package (CSP) availability.

Note 1: Analog features (indicated in) are not available on PIC24FXXKL302 devices.

2: Alternate location for I^2C™ functionality of MSSP1, as determined by the I2C1SEL Configuration bit.

1.2 Other Special Features

- **Communications:** The PIC24F16KL402 family incorporates multiple serial communication peripherals to handle a range of application requirements. The MSSP module implements both SPI and I^2C™ protocols, and supports both Master and Slave modes of operation for each. Devices also include one of two UARTs with built-in IrDA® encoders/decoders.
- **Analog Features:** Select members of the PIC24F16KL402 family include a 10-bit A/D Converter module. The A/D module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period, as well as faster sampling speeds.
  The comparator modules are configurable for a wide range of operations and can be used as either a single or double comparator module.

1.3 Details on Individual Family Members

Devices in the PIC24F16KL402 family are available in 14-pin, 20-pin and 28-pin packages. The general block diagram for all devices is shown in Figure 1-1.

The PIC24F16KL402 family may be thought of as four different device groups, each offering a slightly different set of features. These differ from each other in multiple ways:

- The size of the Flash program memory
- The presence and size of data EEPROM
- The presence of an A/D Converter and the number of external analog channels available
- The number of analog comparators
- The number of general purpose timers
- The number and type of CCP modules (i.e., CCP vs. ECCP)
- The number of serial communications modules (both MSSPs and UARTs)

The general differences between the different sub-families are shown in Table 1-1. The feature sets for specific devices are summarized in Table 1-2 and Table 1-3.

A list of the individual pin features available on the PIC24F16KL402 family devices, sorted by function, is provided in Table 1-4 (for PIC24FXXKL40X/30X devices) and Table 1-5 (for PIC24FXXKL20X/10X devices). Note that these tables show the pin location of individual peripheral features and not how they are multiplexed on the same pin. This information is provided in the pinout diagrams in the beginning of this data sheet. Multiplexed features are sorted by the priority given to a feature, with the highest priority peripheral being listed first.

TABLE 1-1: FEATURE COMPARISON FOR PIC24F16KL402 FAMILY GROUPS

Device Group	Program Memory (bytes)	Data EEPROM (bytes)	Timers (8/16-bit)	CCP and ECCP	Serial (MSSP/ UART)	A/D (channels)	Comparators
PIC24FXXKL10X	4K	—	1/2	2/0	1/1	—	1
PIC24FXXKL20X	8K	—	1/2	2/0	1/1	7 or 12	1
PIC24FXXKL30X	8K	256	2/2	2/1	2/2	—	2
PIC24FXXKL40X	8K or 16K	512	2/2	2/1	2/2	12	2

TABLE 1-4: PIC24F16KL40X/30X FAMILY PINOUT DESCRIPTIONS (CONTINUED)

Function	Pin Number				I/O	Buffer	Description
	20-Pin PDIP/ SSOP/ SOIC	20-Pin QFN	28-Pin SPDIP/ SSOP/ SOIC	28-Pin QFN			
CN0	10	7	12	9	I	ST	Interrupt-on-Change Inputs
CN1	9	6	11	8	I	ST	
CN2	2	19	2	27	I	ST	
CN3	3	20	3	28	I	ST	
CN4	4	1	4	1	I	ST	
CN5	5	2	5	2	I	ST	
CN6	6	3	6	3	I	ST	
CN7	—	—	7	4	I	ST	
CN8	14	11	20	17	I	ST	
CN9	—	—	19	16	I	ST	
CN11	18	15	26	23	I	ST	
CN12	17	14	25	22	I	ST	
CN13	16	13	24	21	I	ST	
CN14	15	12	23	20	I	ST	
CN15	—	—	22	19	I	ST	
CN16	—	—	21	18	I	ST	
CN21	13	10	18	15	I	ST	
CN22	12	9	17	14	I	ST	
CN23	11	8	16	13	I	ST	
CN24	—	—	15	12	I	ST	
CN27	—	—	14	11	I	ST	
CN29	8	5	10	7	I	ST	
CN30	7	4	9	6	I	ST	
CVREF	17	14	25	22	I	ANA	Comparator Voltage Reference Output
CVREF+	2	19	2	27	I	ANA	Comparator Reference Positive Input Voltage
CVREF-	3	20	3	28	I	ANA	Comparator Reference Negative Input Voltage
$\overline{\text{FLT0}}$	17	14	25	22	I	ST	ECCP1 Enhanced PWM Fault Input
HLVDIN	15	12	23	20	I	ST	High/Low-Voltage Detect Input
INT0	11	8	16	13	I	ST	Interrupt 0 Input
INT1	17	14	25	22	I	ST	Interrupt 1 Input
INT2	14	11	20	17	I	ST	Interrupt 2 Input
$\overline{\text{MCLR}}$	1	18	1	26	I	ST	Master Clear (device Reset) Input. This line is brought low to cause a Reset.
OSCI	7	4	9	6	I	ANA	Main Oscillator Input
OSCO	8	5	10	7	O	ANA	Main Oscillator Output
P1A	14	11	20	17	O	—	ECCP1 Output A (Enhanced PWM Mode)
P1B	5	2	21	18	O	—	ECCP1 Output B (Enhanced PWM Mode)
P1C	4	1	22	19	O	—	ECCP1 Output C (Enhanced PWM Mode)
P1D	16	13	18	15	O	—	ECCP1 Output D (Enhanced PWM Mode)

Legend: TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; I^2C = I^2C™/SMBus input buffer

3.0 CPU

Note:	This data sheet summarizes the features of this group of PIC24F devices. It is not intended to be a comprehensive reference source. For more information on the CPU, refer to the *"dsPIC33/PIC24 Family Reference Manual"*, **"CPU"** (DS39703).

The PIC24F CPU has a 16-bit (data) modified Harvard architecture with an enhanced instruction set and a 24-bit instruction word with a variable length opcode field. The Program Counter (PC) is 23 bits wide and addresses up to 4M instructions of user program memory space. A single-cycle instruction prefetch mechanism is used to help maintain throughput and provides predictable execution. All instructions execute in a single cycle, with the exception of instructions that change the program flow, the double-word move (`MOV.D`) instruction and the table instructions. Overhead-free program loop constructs are supported using the `REPEAT` instructions, which are interruptible at any point.

PIC24F devices have sixteen, 16-bit Working registers in the programmer's model. Each of the Working registers can act as a data, address or address offset register. The 16^{th} Working register (W15) operates as a Software Stack Pointer (SSP) for interrupts and calls.

The upper 32 Kbytes of the data space memory map can optionally be mapped into program space at any 16K word boundary of either program memory or data EEPROM memory, defined by the 8-bit Program Space Visibility Page Address (PSVPAG) register. The program to data space mapping feature lets any instruction access program space as if it were data space.

The Instruction Set Architecture (ISA) has been significantly enhanced beyond that of the PIC18, but maintains an acceptable level of backward compatibility. All PIC18 instructions and addressing modes are supported, either directly, or through simple macros. Many of the ISA enhancements have been driven by compiler efficiency needs.

The core supports Inherent (no operand), Relative, Literal, Memory Direct and three groups of addressing modes. All modes support Register Direct and various Register Indirect modes. Each group offers up to seven addressing modes. Instructions are associated with predefined addressing modes depending upon their functional requirements.

For most instructions, the core is capable of executing a data (or program data) memory read, a Working register (data) read, a data memory write and a program (instruction) memory read per instruction cycle. As a result, three parameter instructions can be supported, allowing trinary operations (i.e., A + B = C) to be executed in a single cycle.

A high-speed, 17-bit by 17-bit multiplier has been included to significantly enhance the core arithmetic capability and throughput. The multiplier supports Signed, Unsigned and Mixed mode, 16-bit by 16-bit or 8-bit by 8-bit integer multiplication. All multiply instructions execute in a single cycle.

The 16-bit ALU has been enhanced with integer divide assist hardware that supports an iterative non-restoring divide algorithm. It operates in conjunction with the `REPEAT` instruction looping mechanism and a selection of iterative divide instructions to support 32-bit (or 16-bit), divided by a 16-bit integer signed and unsigned division. All divide operations require 19 cycles to complete, but are interruptible at any cycle boundary.

The PIC24F has a vectored exception scheme, with up to eight sources of non-maskable traps and up to 118 interrupt sources. Each interrupt source can be assigned to one of seven priority levels.

A block diagram of the CPU is illustrated in Figure 3-1.

3.1 Programmer's Model

Figure 3-2 displays the programmer's model for the PIC24F. All registers in the programmer's model are memory mapped and can be manipulated directly by instructions.

Table 3-1 provides a description of each register. All registers associated with the programmer's model are memory mapped.

4.2.2 DATA MEMORY ORGANIZATION AND ALIGNMENT

To maintain backward compatibility with PIC® devices and improve data space memory usage efficiency, the PIC24F instruction set supports both word and byte operations. As a consequence of byte accessibility, all Effective Address (EA) calculations are internally scaled to step through word-aligned memory. For example, the core recognizes that Post-Modified Register Indirect Addressing mode [Ws++] will result in a value of Ws + 1 for byte operations and Ws + 2 for word operations.

Data byte reads will read the complete word, which contains the byte, using the LSB of any EA to determine which byte to select. The selected byte is placed onto the LSB of the data path. That is, data memory and the registers are organized as two parallel, byte-wide entities with shared (word) address decode, but separate write lines. Data byte writes only write to the corresponding side of the array or register, which matches the byte address.

All word accesses must be aligned to an even address. Mis-aligned word data fetches are not supported, so care must be taken when mixing byte and word operations, or translating from 8-bit MCU code. If a mis-aligned read or write is attempted, an address error trap will be generated. If the error occurred on a read, the instruction underway is completed; if it occurred on a write, the instruction will be executed, but the write will not occur. In either case, a trap is then executed, allowing the system and/or user to examine the machine state prior to execution of the address Fault.

All byte loads into any W register are loaded into the LSB; the MSB is not modified.

A Sign-Extend (`SE`) instruction is provided to allow the users to translate 8-bit signed data to 16-bit signed values. Alternatively, for 16-bit unsigned data, users can clear the MSB of any W register by executing a Zero-Extend (`ZE`) instruction on the appropriate address.

Although most instructions are capable of operating on word or byte data sizes, it should be noted that some instructions operate only on words.

4.2.3 NEAR DATA SPACE

The 8-Kbyte area between 0000h and 1FFFh is referred to as the Near Data Space (NDS). Locations in this space are directly addressable via a 13-bit absolute address field within all memory direct instructions. The remainder of the data space is addressable indirectly. Additionally, the whole data space is addressable using `MOV` instructions, which support Memory Direct Addressing (MDA) with a 16-bit address field. For PIC24F16KL402 family devices, the entire implemented data memory lies in Near Data Space.

4.2.4 SFR SPACE

The first 2 Kbytes of the Near Data Space, from 0000h to 07FFh, are primarily occupied with Special Function Registers (SFRs). These are used by the PIC24F core and peripheral modules for controlling the operation of the device.

SFRs are distributed among the modules that they control and are generally grouped together by the module. Much of the SFR space contains unused addresses; these are read as '`0`'. The SFR space, where the SFRs are actually implemented, is provided in Table 4-2. Each implemented area indicates a 32-byte region, where at least one address is implemented as an SFR. A complete listing of implemented SFRs, including their addresses, is provided in Table 4-3 through Table 4-18.

TABLE 4-2: IMPLEMENTED REGIONS OF SFR DATA SPACE

	SFR Space Address							
	xx00	**xx20**	**xx40**	**xx60**	**xx80**	**xxA0**	**xxC0**	**xxE0**
000h	Core			ICN	Interrupts			—
100h	Timers	— / TMR	—	—	— / CCP	CCP / —	—	—
200h	MSSP	UART	—	—	—	—	I/O	—
300h	A/D		—	—	—	—	—	—
400h	—	—	—	—	—	—	— / ANSEL	ANSEL / —
500h	—	—	—	—	—	—	—	—
600h	—	CMP / —	—	—	—	—	—	—
700h	—	—	System	NVM/PMD	—	—	—	—

Legend: — = No implemented SFRs in this block.

TABLE 4-5: INTERRUPT CONTROLLER REGISTER MAP

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
INTCON1	0080	NSTDIS	—	—	—	—	—	—	—	—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—	0000
INTCON2	0082	ALTIVT	DISI	—	—	—	—	—	—	—	—	—	—	—	INT2EP	INT1EP	INT0EP	0000
IFS0	0084	NVMIF	—	AD1IF	U1TXIF	U1RXIF	—	—	T3IF	T2IF	CCP2IF	—	—	T1IF	CCP1IF	—	INT0IF	0000
IFS1	0086	U2TXIF	U2RXIF	INT2IF	—	T4IF(1)	—	CCP3IF(1)	—	—	—	—	INT1IF	CNIF	CMIF	BCL1IF	SSP1IF	0000
IFS2	0088	—	—	—	—	—	—	—	—	—	—	T3GIF	—	—	—	—	—	0000
IFS3	008A	—	—	—	—	—	—	—	—	—	—	—	—	—	BCL2IF(1)	SSP2IF(1)	—	0000
IFS4	008C	—	—	—	—	—	—	—	HLVDIF	—	—	—	—	—	U2ERIF	U1ERIF	—	0000
IFS5	008E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	ULPWUIF	0000
IEC0	0094	NVMIE	—	AD1IE	U1TXIE	U1RXIE	—	—	T3IE	T2IE	CCP2IE	—	—	T1IE	CCP1IE	—	INT0IE	0000
IEC1	0096	U2TXIE	U2RXIE	INT2IE	—	T4IE(1)	—	CCP3IE(1)	—	—	—	—	INT1IE	CNIE	CMIE	BCL1IE	SSP1IE	0000
IEC2	0098	—	—	—	—	—	—	—	—	—	—	T3GIE	—	—	—	—	—	0000
IEC3	009A	—	—	—	—	—	—	—	—	—	—	—	—	—	BCL2IE(1)	SSP2IE(1)	—	0000
IEC4	009C	—	—	—	—	—	—	—	HLVDIE	—	—	—	—	—	U2ERIE	U1ERIE	—	0000
IEC5	009E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	ULPWUIE	0000
IPC0	00A4	—	T1IP2	T1IP1	T1IP0	—	CCP1IP2	CCP1IP1	CCP1IP0	—	—	—	—	—	INT0IP2	INT0IP1	INT0IP0	4404
IPC1	00A6	—	T2IP2	T2IP1	T2IP0	—	CCP2IP2	CCP2IP1	CCP2IP0	—	—	—	—	—	—	—	—	4400
IPC2	00A8	—	U1RXIP2	U1RXIP1	U1RXIP0	—	—	—	—	—	—	—	—	—	T3IP2	T3IP1	T3IP0	4004
IPC3	00AA	—	NVMIP2	NVMIP1	NVMIP0	—	—	—	—	—	AD1IP2	AD1IP1	AD1IP0	—	U1TXIP2	U1TXIP1	U1TXIP0	4044
IPC4	00AC	—	CNIP2	CNIP1	CNIP0	—	CMIP2	CMIP1	CMIP0	—	BCL1IP2	BCL1IP1	BCL1IP0	—	SSP1IP2	SSP1IP1	SS1IP0	4444
IPC5	00AE	—	—	—	—	—	—	—	—	—	—	—	—	—	INT1IP2	INT1IP1	INT1IP0	0004
IPC6	00B0	—	T4IP2(1)	T4IP1(1)	T4IP0(1)	—	—	—	—	—	CCP3IP2(1)	CCP3IP1(1)	CCP3IP0(1)	—	—	—	—	4040
IPC7	00B2	—	U2TXIP2	U2TXIP1	U2TXIP0	—	U2RXIP2	U2RXIP1	U2RXIP0	—	INT2IP2	INT2IP1	INT2IP0	—	—	—	—	4440
IPC9	00B6	—	—	—	—	—	—	—	—	—	T3GIP2	T3GIP1	T3GIP0	—	—	—	—	0040
IPC12	00BC	—	—	—	—	—	BCL2IP2(1)	BCL2IP1(1)	BCL2IP0(1)	—	SSP2IP2(1)	SSP2IP1(1)	SSP2IP0(1)	—	—	—	—	0440
IPC16	00C4	—	—	—	—	—	U2ERIP2	U2ERIP1	U2ERIP0	—	U1ERIP2	U1ERIP1	U1ERIP0	—	—	—	—	0440
IPC18	00C8	—	—	—	—	—	—	—	—	—	—	—	—	—	HLVDIP2	HLVDIP1	HLVDIP0	0004
IPC20	00CC	—	—	—	—	—	—	—	—	—	—	—	—	—	ULPWUIP2	ULPWUIP1	ULPWUIP0	0004
INTTREG	00E0	CPUIRQ	r	VHOLD	—	ILR3	ILR2	ILR1	ILR0	—	VECNUM6	VECNUM5	VECNUM4	VECNUM3	VECNUM2	VECNUM1	VECNUM0	0000

Legend: — = unimplemented, read as '0', r = reserved. Reset values are shown in hexadecimal.

Note 1: These bits are unimplemented on PIC24FXXKL10X and PIC24FXXKL20X family devices; read as '0'.

TABLE 4-10: PORTA REGISTER MAP

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7[1]	Bit 6	Bit 5[2]	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
TRISA	02C0	—	—	—	—	—	—	—	—	TRISA7	TRISA6	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	00DF
PORTA	02C2	—	—	—	—	—	—	—	—	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxx
LATA	02C4	—	—	—	—	—	—	—	—	LATA7	LATA6	—	LATA4	LATA3	LATA2	LATA1	LATA0	xxxx
ODCA	02C6	—	—	—	—	—	—	—	—	ODA7	ODA6	—	ODA4	ODA3	ODA2	ODA1	ODA0	0000

Legend: — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

Note 1: These ports and their associated bits are unimplemented on 14-pin and 20-pin devices; read as '0'.

2: PORTA<5> is unavailable when $\overline{\text{MCLR}}$ functionality is enabled (MCLRE Configuration bit = 1).

TABLE 4-11: PORTB REGISTER MAP

File Name	Addr	Bit 15	Bit 14	Bit 13[1]	Bit 12[1]	Bit 11[2]	Bit 10[2]	Bit 9	Bit 8	Bit 7[1]	Bit 6[2]	Bit 5[2]	Bit 4	Bit 3[2]	Bit 2[1]	Bit 1[1]	Bit 0	All Resets
TRISB	02C8	TRISB15	TRISB14	TRISB13	TRISB12	TRISB11	TRISB10	TRISB9	TRISB8	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	FFFF
PORTB	02CA	RB15	RB14	RB13	RB12	RB11	RB10	RB9	RB8	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx
LATB	02CC	LATB15	LATB14	LATB13	LATB12	LATB11	LATB10	LATB9	LATB8	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	xxxx
ODCB	02CE	ODB15	ODB14	ODB13	ODB12	ODB11	ODB10	ODB9	ODB8	ODB7	ODB6	ODB5	ODB4	ODB3	ODB2	ODB1	ODB0	0000

Legend: — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

Note 1: These ports and their associated bits are unimplemented on 14-pin and 20-pin devices.

2: These ports and their associated bits are unimplemented in 14-pin devices.

TABLE 4-12: PAD CONFIGURATION REGISTER MAP

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
PADCFG1	02FC	—	—	—	—	SDO2DIS[1]	SCK2DIS[1]	SDO1DIS	SCK1DIS	—	—	—	—	—	—	—	—	0000

Legend: — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

Note 1: These bits are unimplemented on PIC24FXXKL10X and PIC24FXXKL20X family devices; read as '0'.

5.5.1 PROGRAMMING ALGORITHM FOR FLASH PROGRAM MEMORY

The user can program one row of Flash program memory at a time by erasing the programmable row. The general process is as follows:

1. Read a row of program memory (32 instructions) and store in data RAM.
2. Update the program data in RAM with the desired new data.
3. Erase a row (see Example 5-1):
    a) Set the NVMOPx bits (NVMCON<5:0>) to '011000' to configure for row erase. Set the ERASE (NVMCON<6>) and WREN (NVMCON<14>) bits.
    b) Write the starting address of the block to be erased into the TBLPAG and W registers.
    c) Write 55h to NVMKEY.
    d) Write AAh to NVMKEY.
    e) Set the WR bit (NVMCON<15>). The erase cycle begins and the CPU stalls for the duration of the erase cycle. When the erase is done, the WR bit is cleared automatically.
4. Write the first 32 instructions from data RAM into the program memory buffers (see Example 5-1).
5. Write the program block to Flash memory:
    a) Set the NVMOPx bits to '000100' to configure for row programming. Clear the ERASE bit and set the WREN bit.
    b) Write 55h to NVMKEY.
    c) Write AAh to NVMKEY.
    d) Set the WR bit. The programming cycle begins and the CPU stalls for the duration of the write cycle. When the write to Flash memory is done, the WR bit is cleared automatically.

For protection against accidental operations, the write initiate sequence for NVMKEY must be used to allow any erase or program operation to proceed. After the programming command has been executed, the user must wait for the programming time until programming is complete. The two instructions following the start of the programming sequence should be NOPs, as shown in Example 5-5.

EXAMPLE 5-1: ERASING A PROGRAM MEMORY ROW – ASSEMBLY LANGUAGE CODE

```
; Set up NVMCON for row erase operation
       MOV     #0x4058, W0                 ;
       MOV     W0, NVMCON                  ; Initialize NVMCON
; Init pointer to row to be ERASED
       MOV     #tblpage(PROG_ADDR), W0     ;
       MOV     W0, TBLPAG                  ; Initialize PM Page Boundary SFR
       MOV     #tbloffset(PROG_ADDR), W0   ; Initialize in-page EA[15:0] pointer
       TBLWTL  W0, [W0]                    ; Set base address of erase block
       DISI    #5                          ; Block all interrupts
                                             for next 5 instructions
       MOV     #0x55, W0
       MOV     W0, NVMKEY                  ; Write the 55 key
       MOV     #0xAA, W1                   ;
       MOV     W1, NVMKEY                  ; Write the AA key
       BSET    NVMCON, #WR                 ; Start the erase sequence
       NOP                                 ; Insert two NOPs after the erase
       NOP                                 ; command is asserted
```

8.0 INTERRUPT CONTROLLER

Note:	This data sheet summarizes the features of this group of PIC24F devices. It is not intended to be a comprehensive reference source. For more information on the Interrupt Controller, refer to the *"dsPIC33/PIC24 Family Reference Manual"*, **"Interrupts"** (DS39707).

The PIC24F interrupt controller reduces the numerous peripheral interrupt request signals to a single interrupt request signal to the CPU. It has the following features:

- Up to eight processor exceptions and software traps
- Seven user-selectable priority levels
- Interrupt Vector Table (IVT) with up to 118 vectors
- Unique vector for each interrupt or exception source
- Fixed priority within a specified user priority level
- Alternate Interrupt Vector Table (AIVT) for debug support
- Fixed interrupt entry and return latencies

8.1 Interrupt Vector Table (IVT)

The IVT is shown in Figure 8-1. The IVT resides in the program memory, starting at location, 000004h. The IVT contains 126 vectors, consisting of eight non-maskable trap vectors, plus up to 118 sources of interrupt. In general, each interrupt source has its own vector. Each interrupt vector contains a 24-bit wide address. The value programmed into each interrupt vector location is the starting address of the associated Interrupt Service Routine (ISR).

Interrupt vectors are prioritized in terms of their natural priority; this is linked to their position in the vector table. All other things being equal, lower addresses have a higher natural priority. For example, the interrupt associated with vector 0 will take priority over interrupts at any other vector address.

PIC24F16KL402 family devices implement 32 non-maskable traps and unique interrupts; these are summarized in Table 8-1 and Table 8-2.

8.1.1 ALTERNATE INTERRUPT VECTOR TABLE (AIVT)

The Alternate Interrupt Vector Table (AIVT) is located after the IVT, as shown in Figure 8-1. Access to the AIVT is provided by the ALTIVT control bit (INTCON2<15>). If the ALTIVT bit is set, all interrupt and exception processes will use the alternate vectors instead of the default vectors. The alternate vectors are organized in the same manner as the default vectors.

The AIVT supports emulation and debugging efforts by providing a means to switch between an application and a support environment without requiring the interrupt vectors to be reprogrammed. This feature also enables switching between applications for evaluation of different software algorithms at run time. If the AIVT is not needed, the AIVT should be programmed with the same addresses used in the IVT.

8.2 Reset Sequence

A device Reset is not a true exception, because the interrupt controller is not involved in the Reset process. The PIC24F devices clear their registers in response to a Reset, which forces the Program Counter (PC) to zero. The microcontroller then begins program execution at location, 000000h. The user programs a GOTO instruction at the Reset address, which redirects the program execution to the appropriate start-up routine.

Note:	Any unimplemented or unused vector locations in the IVT and AIVT should be programmed with the address of a default interrupt handler routine that contains a RESET instruction.

REGISTER 8-25: IPC9: INTERRUPT PRIORITY CONTROL REGISTER 9

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	T3GIP2	T3GIP1	T3GIP0	—	—	—	—
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 15-7 **Unimplemented:** Read as ‘0’

bit 6-4 **T3GIP<2:0>:** Timer3 External Gate Interrupt Priority bits
111 = Interrupt is Priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is Priority 1
000 = Interrupt source is disabled

bit 3-0 **Unimplemented:** Read as ‘0’

10.4 Doze Mode

Generally, changing clock speed and invoking one of the power-saving modes are the preferred strategies for reducing power consumption. There may be circumstances, however, where this is not practical. For example, it may be necessary for an application to maintain uninterrupted, synchronous communication, even while it is doing nothing else. Reducing system clock speed may introduce communication errors, while using a power-saving mode may stop communications completely.

Doze mode is a simple and effective alternative method to reduce power consumption while the device is still executing code. In this mode, the system clock continues to operate from the same source and at the same speed. Peripheral modules continue to be clocked at the same speed, while the CPU clock speed is reduced. Synchronization between the two clock domains is maintained, allowing the peripherals to access the SFRs while the CPU executes code at a slower rate.

Doze mode is enabled by setting the DOZEN bit (CLKDIV<11>). The ratio between peripheral and core clock speed is determined by the DOZE<2:0> bits (CLKDIV<14:12>). There are eight possible configurations, from 1:1 to 1:128, with 1:1 being the default.

It is also possible to use Doze mode to selectively reduce power consumption in event driven applications. This allows clock-sensitive functions, such as synchronous communications, to continue without interruption. Meanwhile, the CPU Idles, waiting for something to invoke an interrupt routine. Enabling the automatic return to full-speed CPU operation on interrupts is enabled by setting the ROI bit (CLKDIV<15>). By default, interrupt events have no effect on Doze mode operation.

10.5 Selective Peripheral Module Control

Idle and Doze modes allow users to substantially reduce power consumption by slowing or stopping the CPU clock. Even so, peripheral modules still remain clocked and thus, consume power. There may be cases where the application needs what these modes do not provide: the allocation of power resources to CPU processing, with minimal power consumption from the peripherals.

PIC24F devices address this requirement by allowing peripheral modules to be selectively disabled, reducing or eliminating their power consumption. This can be done with two control bits:

- The Peripheral Enable bit, generically named, “XXXEN”, located in the module’s main control SFR.
- The Peripheral Module Disable (PMD) bit, generically named, “XXXMD”, located in one of the PMD Control registers.

Both bits have similar functions in enabling or disabling its associated module. Setting the PMD bit for a module disables all clock sources to that module, reducing its power consumption to an absolute minimum. In this state, the control and status registers associated with the peripheral will also be disabled, so writes to those registers will have no effect, and read values will be invalid. Many peripheral modules have a corresponding PMD bit.

In contrast, disabling a module by clearing its XXXEN bit, disables its functionality, but leaves its registers available to be read and written to. Power consumption is reduced, but not by as much as when the PMD bits are used.

To achieve more selective power savings, peripheral modules can also be selectively disabled when the device enters Idle mode. This is done through the control bit of the generic name format, “XXXIDL”. By default, all modules that can operate during Idle mode will do so. Using the disable on Idle feature disables the module while in Idle mode, allowing further reduction of power consumption during Idle mode. This enhances power savings for extremely critical power applications.

REGISTER 17-2: SSPxSTAT: MSSPx STATUS REGISTER (I²C™ MODE)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{\text{A}}$	P(1)	S(1)	R/$\overline{\text{W}}$	UA	BF
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 15-8 **Unimplemented:** Read as ‘0’

bit 7 **SMP:** Slew Rate Control bit
In Master or Slave mode:
1 = Slew rate control is disabled for Standard Speed mode (100 kHz and 1 MHz)
0 = Slew rate control is enabled for High-Speed mode (400 kHz)

bit 6 **CKE:** SMBus Select bit
In Master or Slave mode:
1 = Enables SMBus specific inputs
0 = Disables SMBus specific inputs

bit 5 **D/$\overline{\text{A}}$:** Data/$\overline{\text{Address}}$ bit
In Master mode:
Reserved.
In Slave mode:
1 = Indicates that the last byte received or transmitted was data
0 = Indicates that the last byte received or transmitted was address

bit 4 **P:** Stop bit(1)
1 = Indicates that a Stop bit has been detected last
0 = Stop bit was not detected last

bit 3 **S:** Start bit(1)
1 = Indicates that a Start bit has been detected last
0 = Start bit was not detected last

bit 2 **R/$\overline{\text{W}}$:** Read/$\overline{\text{Write}}$ Information bit
In Slave mode:(2)
1 = Read
0 = Write
In Master mode:(3)
1 = Transmit is in progress
0 = Transmit is not in progress

bit 1 **UA:** Update Address bit (10-Bit Slave mode only)
1 = Indicates that the user needs to update the address in the SSPxADD register
0 = Address does not need to be updated

Note 1: This bit is cleared on RESET and when SSPEN is cleared.
2: This bit holds the R/$\overline{\text{W}}$ bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit or not $\overline{\text{ACK}}$ bit.
3: ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSPx is in Active mode.

REGISTER 17-4: SSPxCON1: MSSPx CONTROL REGISTER 1 (I^{2}C™ MODE)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN(1)	CKP	SSPM3(2)	SSPM2(2)	SSPM1(2)	SSPM0(2)
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 15-8 **Unimplemented:** Read as ‘0’

bit 7 **WCOL:** Write Collision Detect bit

In Master Transmit mode:
1 = A write to the SSPxBUF register was attempted while the I^{2}C conditions were not valid for a transmission to be started (must be cleared in software)
0 = No collision

In Slave Transmit mode:
1 = The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

In Receive mode (Master or Slave modes):
This is a “don’t care” bit.

bit 6 **SSPOV:** MSSPx Receive Overflow Indicator bit

In Receive mode:
1 = A byte is received while the SSPxBUF register is still holding the previous byte (must be cleared in software)
0 = No overflow

In Transmit mode:
This is a “don’t care” bit in Transmit mode.

bit 5 **SSPEN:** MSSPx Enable bit(1)

1 = Enables the serial port and configures the SDAx and SCLx pins as the serial port pins
0 = Disables the serial port and configures these pins as I/O port pins

bit 4 **CKP:** SCLx Release Control bit

In Slave mode:
1 = Releases clock
0 = Holds clock low (clock stretch); used to ensure data setup time

In Master mode:
Unused in this mode.

bit 3-0 **SSPM<3:0>:** MSSPx Mode Select bits(2)

1111 = I^{2}C Slave mode, 10-bit address with Start and Stop bit interrupts is enabled
1110 = I^{2}C Slave mode, 7-bit address with Start and Stop bit interrupts is enabled
1011 = I^{2}C Firmware Controlled Master mode (Slave Idle)
1000 = I^{2}C Master mode, Clock = $F_{OSC}/(2 * ([SSPxADD] + 1))$(3)
0111 = I^{2}C Slave mode, 10-bit address
0110 = I^{2}C Slave mode, 7-bit address

Note 1: When enabled, the SDAx and SCLx pins must be configured as inputs.
2: Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.
3: SSPxADD values of 0, 1 or 2 are not supported when the Baud Rate Generator is used with I^{2}C mode.

NOTES:

REGISTER 22-1: HLVDCON: HIGH/LOW-VOLTAGE DETECT CONTROL REGISTER

R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
HLVDEN	—	HLSIDL	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
VDIR	BGVST	IRVST	—	HLVDL3	HLVDL2	HLVDL1	HLVDL0
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 15 **HLVDEN:** High/Low-Voltage Detect Power Enable bit
1 = HLVD is enabled
0 = HLVD is disabled

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **HLSIDL:** HLVD Stop in Idle Mode bit
1 = Discontinues module operation when the device enters Idle mode
0 = Continues module operation in Idle mode

bit 12-8 **Unimplemented:** Read as ‘0’

bit 7 **VDIR:** Voltage Change Direction Select bit
1 = Event occurs when the voltage equals or exceeds the trip point (HLVDL<3:0>)
0 = Event occurs when the voltage equals or falls below the trip point (HLVDL<3:0>)

bit 6 **BGVST:** Band Gap Voltage Stable Flag bit
1 = Indicates that the band gap voltage is stable
0 = Indicates that the band gap voltage is unstable

bit 5 **IRVST:** Internal Reference Voltage Stable Flag bit
1 = Indicates that the internal reference voltage is stable and the High-Voltage Detect logic generates the interrupt flag at the specified voltage range
0 = Indicates that the internal reference voltage is unstable and the High-Voltage Detect logic will not generate the interrupt flag at the specified voltage range, and the HLVD interrupt should not be enabled

bit 4 **Unimplemented:** Read as ‘0’

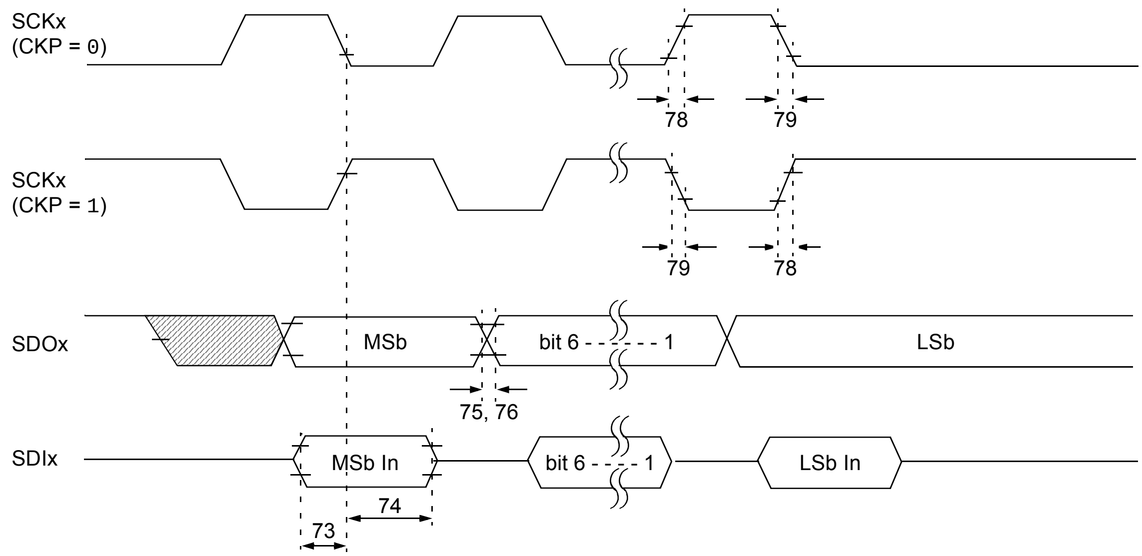
bit 3-0 **HLVDL<3:0>:** High/Low-Voltage Detection Limit bits
1111 = External analog input is used (input comes from the HLVDIN pin)
1110 = Trip Point 14(1)
1101 = Trip Point 13(1)
1100 = Trip Point 12(1)
.
.
.
0000 = Trip Point 0(1)

Note 1: For the actual trip point, see **Section 26.0 “Electrical Characteristics”**.

TABLE 25-2: INSTRUCTION SET OVERVIEW (CONTINUED)

Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles	Status Flags Affected
GOTO	GOTO Expr	Go to Address	2	2	None
	GOTO Wn	Go to Indirect	1	2	None
INC	INC f	f = f + 1	1	1	C, DC, N, OV, Z
	INC f,WREG	WREG = f + 1	1	1	C, DC, N, OV, Z
	INC Ws,Wd	Wd = Ws + 1	1	1	C, DC, N, OV, Z
INC2	INC2 f	f = f + 2	1	1	C, DC, N, OV, Z
	INC2 f,WREG	WREG = f + 2	1	1	C, DC, N, OV, Z
	INC2 Ws,Wd	Wd = Ws + 2	1	1	C, DC, N, OV, Z
IOR	IOR f	f = f .IOR. WREG	1	1	N, Z
	IOR f,WREG	WREG = f .IOR. WREG	1	1	N, Z
	IOR #lit10,Wn	Wd = lit10 .IOR. Wd	1	1	N, Z
	IOR Wb,Ws,Wd	Wd = Wb .IOR. Ws	1	1	N, Z
	IOR Wb,#lit5,Wd	Wd = Wb .IOR. lit5	1	1	N, Z
LNK	LNK #lit14	Link Frame Pointer	1	1	None
LSR	LSR f	f = Logical Right Shift f	1	1	C, N, OV, Z
	LSR f,WREG	WREG = Logical Right Shift f	1	1	C, N, OV, Z
	LSR Ws,Wd	Wd = Logical Right Shift Ws	1	1	C, N, OV, Z
	LSR Wb,Wns,Wnd	Wnd = Logical Right Shift Wb by Wns	1	1	N, Z
	LSR Wb,#lit5,Wnd	Wnd = Logical Right Shift Wb by lit5	1	1	N, Z
MOV	MOV f,Wn	Move f to Wn	1	1	None
	MOV [Wns+Slit10],Wnd	Move [Wns+Slit10] to Wnd	1	1	None
	MOV f	Move f to f	1	1	N, Z
	MOV f,WREG	Move f to WREG	1	1	None
	MOV #lit16,Wn	Move 16-bit Literal to Wn	1	1	None
	MOV.b #lit8,Wn	Move 8-bit Literal to Wn	1	1	None
	MOV Wn,f	Move Wn to f	1	1	None
	MOV Wns,[Wns+Slit10]	Move Wns to [Wns+Slit10]	1	1	None
	MOV Wso,Wdo	Move Ws to Wd	1	1	None
	MOV WREG,f	Move WREG to f	1	1	None
	MOV.D Wns,Wd	Move Double from W(ns):W(ns+1) to Wd	1	2	None
	MOV.D Ws,Wnd	Move Double from Ws to W(nd+1):W(nd)	1	2	None
MUL	MUL.SS Wb,Ws,Wnd	{Wnd+1, Wnd} = Signed(Wb) * Signed(Ws)	1	1	None
	MUL.SU Wb,Ws,Wnd	{Wnd+1, Wnd} = Signed(Wb) * Unsigned(Ws)	1	1	None
	MUL.US Wb,Ws,Wnd	{Wnd+1, Wnd} = Unsigned(Wb) * Signed(Ws)	1	1	None
	MUL.UU Wb,Ws,Wnd	{Wnd+1, Wnd} = Unsigned(Wb) * Unsigned(Ws)	1	1	None
	MUL.SU Wb,#lit5,Wnd	{Wnd+1, Wnd} = Signed(Wb) * Unsigned(lit5)	1	1	None
	MUL.UU Wb,#lit5,Wnd	{Wnd+1, Wnd} = Unsigned(Wb) * Unsigned(lit5)	1	1	None
	MUL f	W3:W2 = f * WREG	1	1	None
NEG	NEG f	f = $\overline{f}$ + 1	1	1	C, DC, N, OV, Z
	NEG f,WREG	WREG = $\overline{f}$ + 1	1	1	C, DC, N, OV, Z
	NEG Ws,Wd	Wd = $\overline{Ws}$ + 1	1	1	C, DC, N, OV, Z
NOP	NOP	No Operation	1	1	None
	NOPR	No Operation	1	1	None
POP	POP f	Pop f from Top-of-Stack (TOS)	1	1	None
	POP Wdo	Pop from Top-of-Stack (TOS) to Wdo	1	1	None
	POP.D Wnd	Pop from Top-of-Stack (TOS) to W(nd):W(nd+1)	1	2	None
	POP.S	Pop Shadow Registers	1	1	All
PUSH	PUSH f	Push f to Top-of-Stack (TOS)	1	1	None
	PUSH Wso	Push Wso to Top-of-Stack (TOS)	1	1	None
	PUSH.D Wns	Push W(ns):W(ns+1) to Top-of-Stack (TOS)	1	2	None
	PUSH.S	Push Shadow Registers	1	1	None

FIGURE 26-7: EXAMPLE SPI MASTER MODE TIMING (CKE = 0)

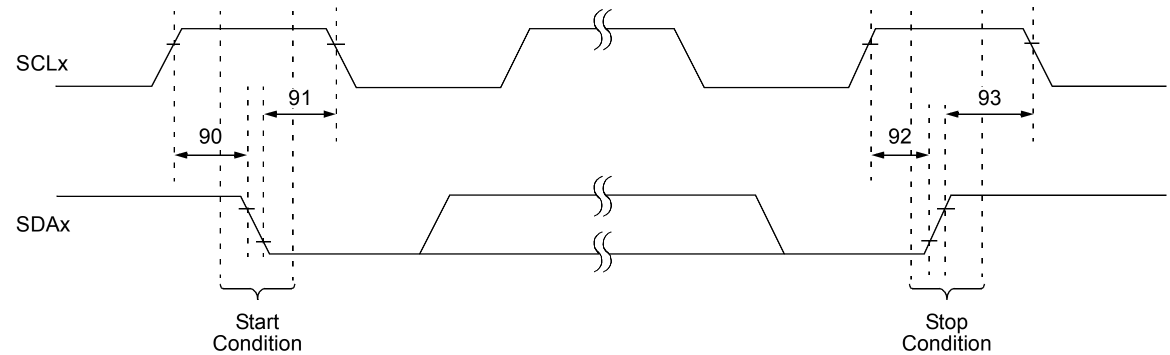


Note: Refer to Figure 26-3 for load conditions.

TABLE 26-27: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 0)

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
73	TDIV2SCH, TDIV2SCL	Setup Time of SDIx Data Input to SCKx Edge	20	—	ns	
74	TSCH2DIL, TSCL2DIL	Hold Time of SDIx Data Input to SCKx Edge	40	—	ns	
75	TDOR	SDOx Data Output Rise Time	—	25	ns	
76	TDOF	SDOx Data Output Fall Time	—	25	ns	
78	TSCR	SCKx Output Rise Time (Master mode)	—	25	ns	
79	TSCF	SCKx Output Fall Time (Master mode)	—	25	ns	
	FSCK	SCKx Frequency	—	10	MHz	

FIGURE 26-11: I^2C™ BUS START/STOP BITS TIMING

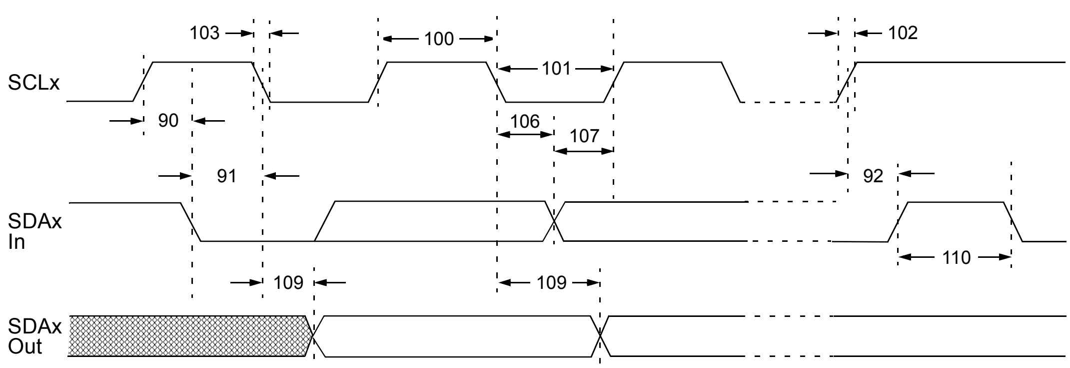


Note: Refer to Figure 26-3 for load conditions.

TABLE 26-31: I^2C™ BUS START/STOP BITS REQUIREMENTS (SLAVE MODE)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
90	TSU:STA	Start Condition Setup Time	100 kHz mode	4700	—	ns	Only relevant for Repeated Start condition
			400 kHz mode	600	—		
91	THD:STA	Start Condition Hold Time	100 kHz mode	4000	—	ns	After this period, the first clock pulse is generated
			400 kHz mode	600	—		
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	4700	—	ns	
			400 kHz mode	600	—		
93	THD:STO	Stop Condition Hold Time	100 kHz mode	4000	—	ns	
			400 kHz mode	600	—		

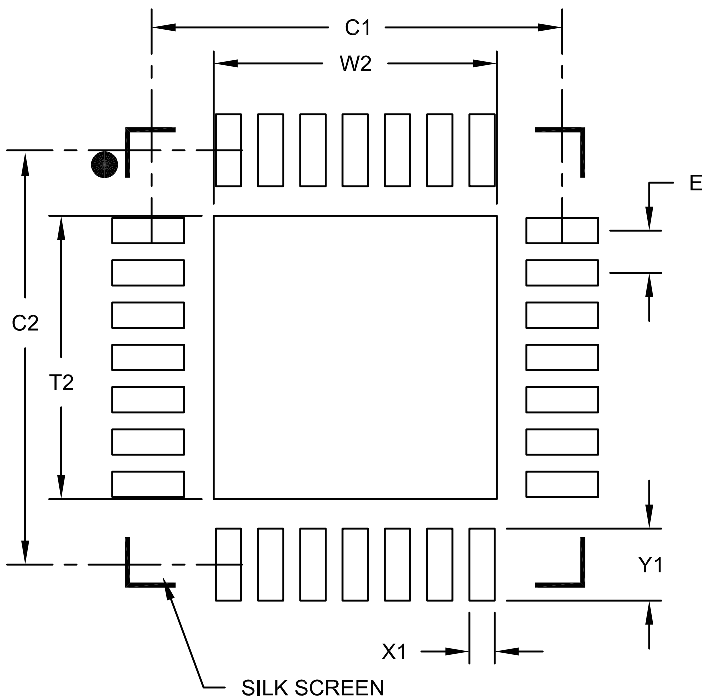
FIGURE 26-12: I^2C™ BUS DATA TIMING



Note: Refer to Figure 26-3 for load conditions.

28-Lead Plastic Quad Flat, No Lead Package (MQ) – 5x5 mm Body [QFN] Land Pattern With 0.55 mm Contact Length

Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.50 BSC	
Optional Center Pad Width	W2			3.35
Optional Center Pad Length	T2			3.35
Contact Pad Spacing	C1		4.90	
Contact Pad Spacing	C2		4.90	
Contact Pad Width (X28)	X1			0.30
Contact Pad Length (X28)	Y1			0.85

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-2140A

PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PIC 24 F 16 KL4 02 T - I / PT - XXX

- **Microchip Trademark**
- **Architecture**
- **Flash Memory Family**
- **Program Memory Size (Kbytes)**
- **Product Group**
- **Pin Count**
- **Tape and Reel Flag (if applicable)**
- **Temperature Range**
- **Package**
- **Pattern**

Architecture	24	= 16-bit modified Harvard without DSP
Flash Memory Family	F	= Standard voltage range Flash program memory
Product Group	KL4 KL3 KL2 KL1	= General purpose microcontrollers
Pin Count	00 01 02	= 14-pin = 20-pin = 28-pin
Temperature Range	I E	= -40°C to +85°C (Industrial) = -40°C to +125°C (Extended)
Package	SP SO SS ST ML, MQ P	= SPDIP = SOIC = SSOP = TSSOP = QFN = PDIP
Pattern	Three-digit QTP, SQTP, Code or Special Requirements (blank otherwise) ES = Engineering Sample	

Examples:

a) PIC24F16KL402-I/ML: General Purpose, 16-Kbyte Program Memory, 28-Pin, Industrial Temperature, QFN Package

b) PIC24F04KL101T-I/SS: General Purpose, 4-Kbyte Program Memory, 20-Pin, Industrial Temperature, SSOP Package, Tape-and-Reel