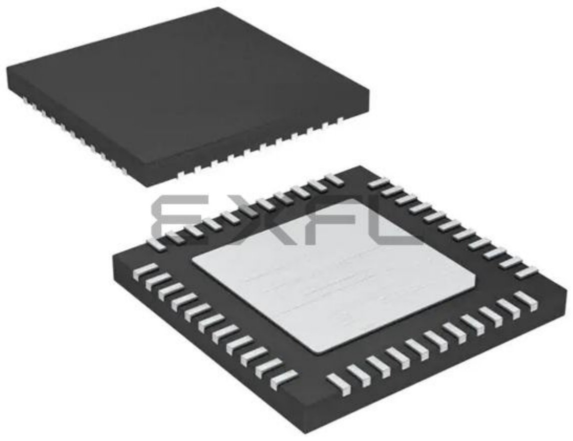
Microchip Technology - DSPIC30F4013-20I/ML Datasheet



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	dsPIC
Core Size	16-Bit
Speed	20 MIPS
Connectivity	CANbus, I²C, SPI, UART/USART
Peripherals	AC'97, Brown-out Detect/Reset, I²S, POR, PWM, WDT
Number of I/O	30
Program Memory Size	48KB (16K x 24)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	A/D 13x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°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/dspic30f4013-20i-ml

NOTES:

FIGURE 1-2: dsPIC30F4013 BLOCK DIAGRAM

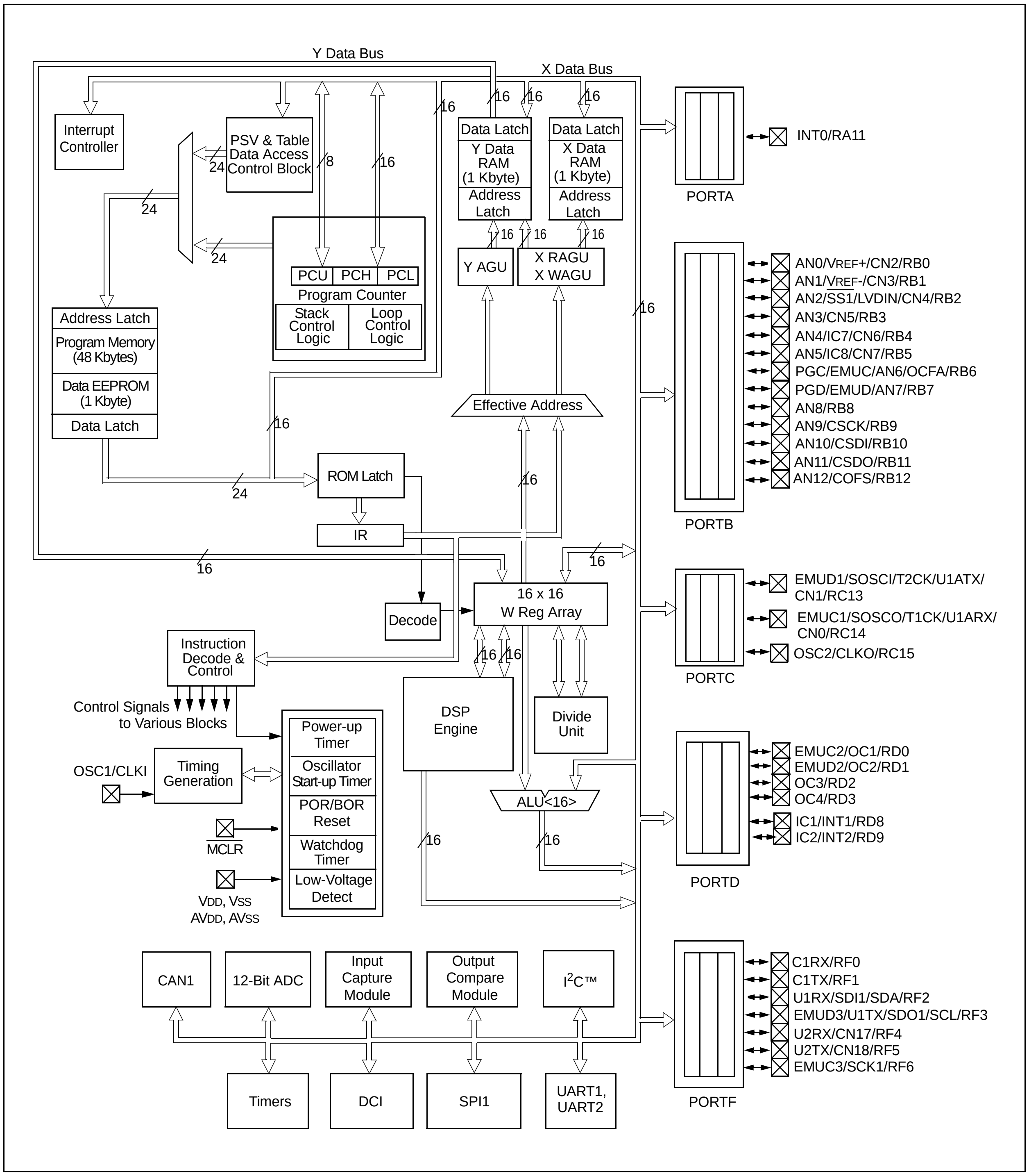
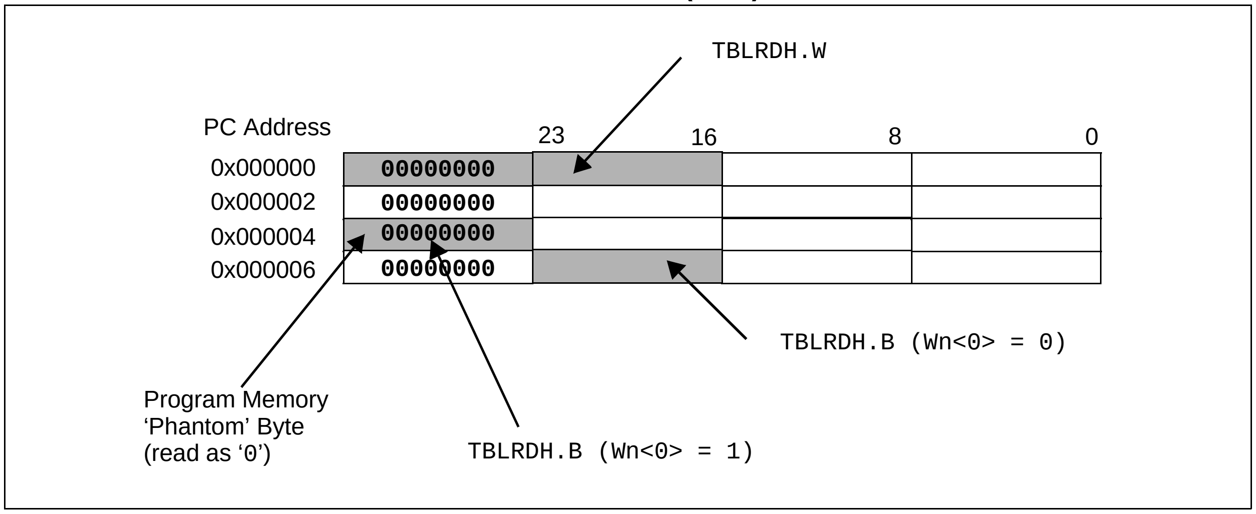


FIGURE 3-5: PROGRAM DATA TABLE ACCESS (MSB)



3.1.2 DATA ACCESS FROM PROGRAM MEMORY USING PROGRAM SPACE VISIBILITY

The upper 32 Kbytes of data space may optionally be mapped into any 16K word program space page. This provides transparent access of stored constant data from X data space without the need to use special instructions (i.e., TBLRDL/H, TBLWTL/H instructions).

Program space access through the data space occurs if the MSb of the data space, EA, is set and program space visibility is enabled by setting the PSV bit in the Core Control register (CORCON). The functions of CORCON are discussed in **Section 2.4 "DSP Engine"**.

Data accesses to this area add an additional cycle to the instruction being executed, since two program memory fetches are required.

Note that the upper half of addressable data space is always part of the X data space. Therefore, when a DSP operation uses program space mapping to access this memory region, Y data space should typically contain state (variable) data for DSP operations, whereas X data space should typically contain coefficient (constant) data.

Although each data space address, 0x8000 and higher, maps directly into a corresponding program memory address (see Figure 3-6), only the lower 16 bits of the 24-bit program word are used to contain the data. The upper 8 bits should be programmed to force an illegal instruction to maintain machine robustness. Refer to the *"16-bit MCU and DSC Programmer's Reference Manual*" (DS70157) for details on instruction encoding.

Note that by incrementing the PC by 2 for each program memory word, the 15 LSbs of data space addresses directly map to the 15 LSbs in the corresponding program space addresses. The remaining bits are provided by the Program Space Visibility Page register, PSVPAG<7:0>, as shown in Figure 3-6.

Note: PSV access is temporarily disabled during table reads/writes.

For instructions that use PSV which are executed outside a REPEAT loop:

- The following instructions require one instruction cycle in addition to the specified execution time:
  - MAC class of instructions with data operand prefetch
  - MOV instructions
  - MOV.D instructions
- All other instructions require two instruction cycles in addition to the specified execution time of the instruction.

For instructions that use PSV which are executed inside a REPEAT loop:

- The following instances require two instruction cycles in addition to the specified execution time of the instruction:
  - Execution in the first iteration
  - Execution in the last iteration
  - Execution prior to exiting the loop due to an interrupt
  - Execution upon re-entering the loop after an interrupt is serviced
- Any other iteration of the REPEAT loop allows the instruction accessing data, using PSV, to execute in a single cycle.

3.2.2 DATA SPACES

The X data space is used by all instructions and supports all addressing modes. There are separate read and write data buses. The X read data bus is the return data path for all instructions that view data space as combined X and Y address space. It is also the X address space data path for the dual operand read instructions (`MAC` class). The X write data bus is the only write path to data space for all instructions.

The X data space also supports Modulo Addressing for all instructions, subject to addressing mode restrictions. Bit-Reversed Addressing is only supported for writes to X data space.

The Y data space is used in concert with the X data space by the `MAC` class of instructions (`CLR, ED, EDAC, MAC, MOVSAC, MPY, MPY.N` and `MSC`) to provide two concurrent data read paths. No writes occur across the Y bus. This class of instructions dedicates two W register pointers, W10 and W11, to always address Y data space, independent of X data space, whereas W8 and W9 always address X data space. Note that during accumulator write-back, the data address space is considered a combination of X and Y data spaces, so the write occurs across the X bus. Consequently, the write can be to any address in the entire data space.

The Y data space can only be used for the data prefetch operation associated with the `MAC` class of instructions. It also supports Modulo Addressing for automated circular buffers. Of course, all other instructions can access the Y data address space through the X data path as part of the composite linear space.

The boundary between the X and Y data spaces is defined as shown in Figure 3-7 and is not user-programmable. Should an EA point to data outside its own assigned address space, or to a location outside physical memory, an all zero word/byte is returned. For example, although Y address space is visible by all non-`MAC` instructions using any addressing mode, an attempt by a `MAC` instruction to fetch data from that space using W8 or W9 (X Space Pointers) returns 0x0000.

TABLE 3-2: EFFECT OF INVALID MEMORY ACCESSES

Attempted Operation	Data Returned
EA = an unimplemented address	0x0000
W8 or W9 used to access Y data space in a `MAC` instruction	0x0000
W10 or W11 used to access X data space in a `MAC` instruction	0x0000

All effective addresses are 16 bits wide and point to bytes within the data space. Therefore, the data space address range is 64 Kbytes or 32K words.

3.2.3 DATA SPACE WIDTH

The core data width is 16 bits. All internal registers are organized as 16-bit wide words. Data space memory is organized in byte addressable, 16-bit wide blocks.

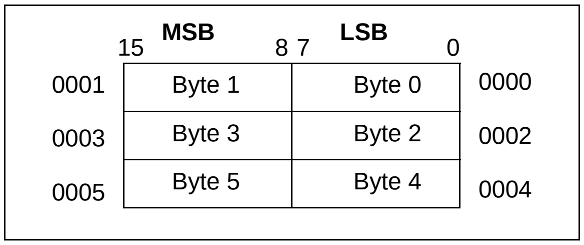
3.2.4 DATA ALIGNMENT

To help maintain backward compatibility with PIC® MCU devices and improve data space memory usage efficiency, the dsPIC30F instruction set supports both word and byte operations. Data is aligned in data memory and registers as words, but all data space EAs resolve to bytes. Data byte reads 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 X data path (no byte accesses are possible from the Y data path as the `MAC` class of instruction can only fetch words). That is, data memory and 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.

As a consequence of this byte accessibility, all effective address calculations (including those generated by the DSP operations which are restricted to word-sized data) are internally scaled to step through word-aligned memory. For example, the core would recognize 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.

All word accesses must be aligned to an even address. Misaligned word data fetches are not supported so care must be taken when mixing byte and word operations, or translating from 8-bit MCU code. Should a misaligned read or write be attempted, an address error trap is generated. If the error occurred on a read, the instruction underway is completed, whereas if it occurred on a write, the instruction is executed but the write does 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.

FIGURE 3-9: DATA ALIGNMENT



5.6 Programming Operations

A complete programming sequence is necessary for programming or erasing the internal Flash in RTSP mode. A programming operation is nominally 2 msec in duration and the processor stalls (waits) until the operation is finished. Setting the WR bit (NVMCON<15>) starts the operation and the WR bit is automatically cleared when the operation is finished.

5.6.1 PROGRAMMING ALGORITHM FOR PROGRAM FLASH

The user can erase or program one row of program Flash memory at a time. The general process is:

1. Read one row of program Flash (32 instruction words) and store into data RAM as a data “image”.
2. Update the data image with the desired new data.
3. Erase program Flash row.
    a) Set up NVMCON register for multi-word, program Flash, erase, and set WREN bit.
    b) Write address of row to be erased into NVMADRU/NVMDR.
    c) Write 0x55 to NVMKEY.
    d) Write 0xAA to NVMKEY.
    e) Set the WR bit. This begins erase cycle.
    f) CPU stalls for the duration of the erase cycle.
    g) The WR bit is cleared when erase cycle ends.
4. Write 32 instruction words of data from data RAM “image” into the program Flash write latches.
5. Program 32 instruction words into program Flash.
    a) Set up NVMCON register for multi-word, program Flash, program, and set WREN bit.
    b) Write 0x55 to NVMKEY.
    c) Write 0xAA to NVMKEY.
    d) Set the WR bit. This begins program cycle.
    e) CPU stalls for duration of the program cycle.
    f) The WR bit is cleared by the hardware when program cycle ends.
6. Repeat steps 1 through 5 as needed to program desired amount of program Flash memory.

5.6.2 ERASING A ROW OF PROGRAM MEMORY

Example 5-1 shows a code sequence that can be used to erase a row (32 instructions) of program memory.

EXAMPLE 5-1: ERASING A ROW OF PROGRAM MEMORY

```
; Setup NVMCON for erase operation, multi word write
; program memory selected, and writes enabled
       MOV    #0x4041,W0                 ;
       MOV    W0,NVMCON                  ; Init NVMCON SFR
; Init pointer to row to be ERASED
       MOV    #tblpage(PROG_ADDR),W0     ;
       MOV    W0,NVMADRU                 ; Initialize PM Page Boundary SFR
       MOV    #tbloffset(PROG_ADDR),W0   ; Intialize in-page EA[15:0] pointer
       MOV    W0, NVMADR                 ; Initialize NVMADR SFR
       DISI   #5                         ; Block all interrupts with priority <7 for
                                         ; next 5 instructions
       MOV    #0x55,W0
       MOV    W0,NVMKEY                  ; Write the 0x55 key
       MOV    #0xAA,W1                   ;
       MOV    W1,NVMKEY                  ; Write the 0xAA key
       BSET   NVMCON,#WR                 ; Start the erase sequence
       NOP                               ; Insert two NOPs after the erase
       NOP                               ; command is asserted
```

FIGURE 10-2: 16-BIT TIMER2 BLOCK DIAGRAM

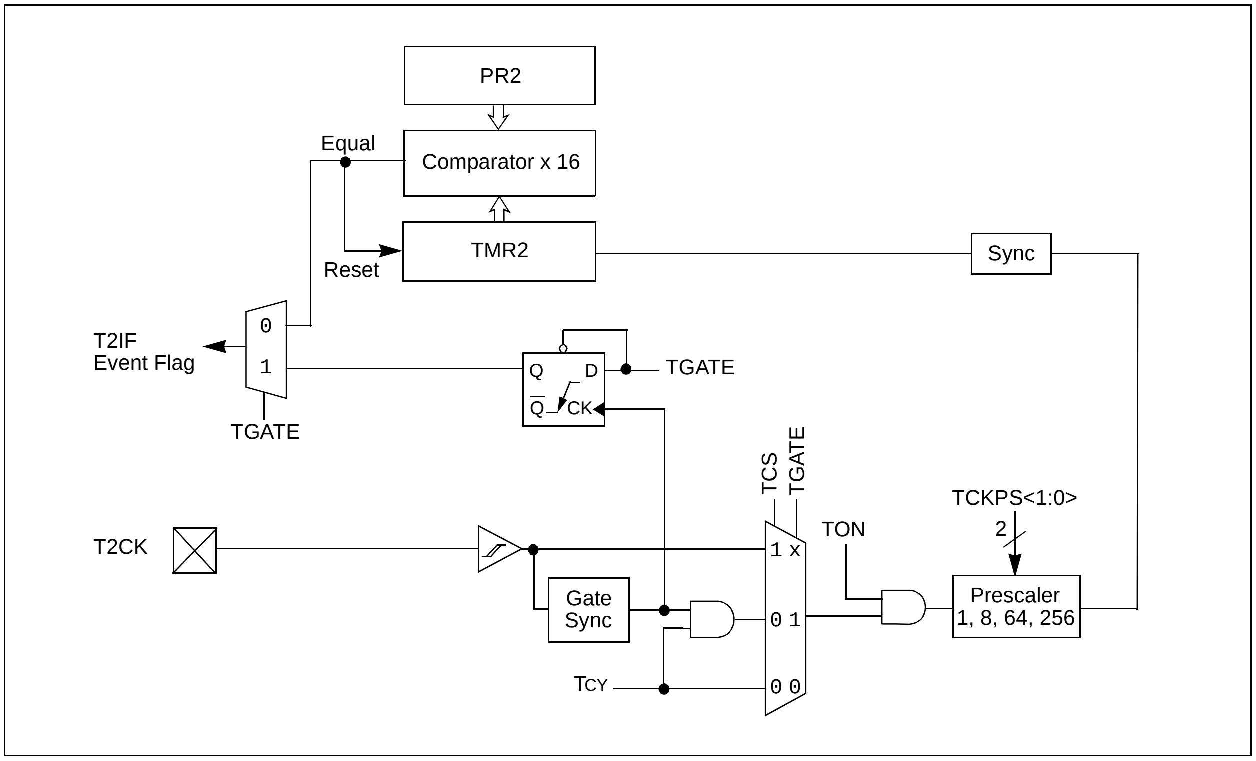
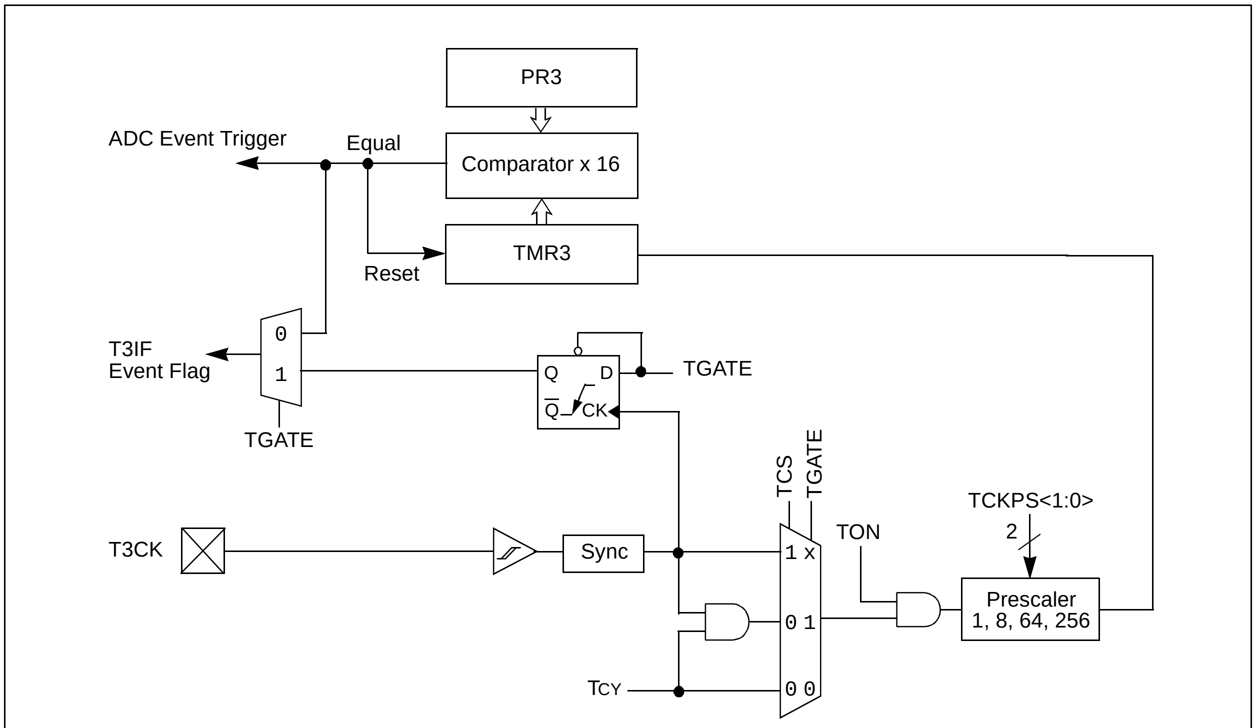


FIGURE 10-3: 16-BIT TIMER3 BLOCK DIAGRAM



Note: T3CK pin does not exist on dsPIC30F3014/4013 devices. The block diagram shown here illustrates the schematic of Timer3 as implemented on the dsPIC30F6014 device.

TABLE 10-1: dsPIC30F3014/4013 TIMER2/3 REGISTER MAP(1)

SFR 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	Reset State
TMR2	0106	Timer2 Register																uuuu uuuu uuuu uuuu
TMR3HLD	0108	Timer3 Holding Register (for 32-bit timer operations only)																uuuu uuuu uuuu uuuu
TMR3	010A	Timer3 Register																uuuu uuuu uuuu uuuu
PR2	010C	Period Register 2																1111 1111 1111 1111
PR3	010E	Period Register 3																1111 1111 1111 1111
T2CON	0110	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	T32	—	TCS	—	0000 0000 0000 0000
T3CON	0112	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	—	TCS	—	0000 0000 0000 0000

Legend: u = uninitialized bit; — = unimplemented bit, read as '0'

Note 1: Refer to the *"dsPIC30F Family Reference Manual"* (DS70046) for descriptions of register bit fields.

NOTES:

16.5.2 FRAMING ERROR (FERR)

The FERR bit (UxSTA<2>) is set if a ‘0’ is detected instead of a Stop bit. If two Stop bits are selected, both Stop bits must be ‘1’; otherwise, FERR is set. The read-only FERR bit is buffered along with the received data; it is cleared on any Reset.

16.5.3 PARITY ERROR (PERR)

The PERR bit (UxSTA<3>) is set if the parity of the received word is incorrect. This error bit is applicable only if a Parity mode (odd or even) is selected. The read-only PERR bit is buffered along with the received data bytes; it is cleared on any Reset.

16.5.4 IDLE STATUS

When the receiver is active (i.e., between the initial detection of the Start bit and the completion of the Stop bit), the RIDLE bit (UxSTA<4>) is ‘0’. Between the completion of the Stop bit and detection of the next Start bit, the RIDLE bit is ‘1’, indicating that the UART is Idle.

16.5.5 RECEIVE BREAK

The receiver counts and expects a certain number of bit times based on the values programmed in the PDSEL (UxMODE<2:1>) and STSEL (UxMODE<0>) bits.

If the break is longer than 13 bit times, the reception is considered complete after the number of bit times specified by PDSEL and STSEL. The URXDA bit is set, FERR is set, zeros are loaded into the receive FIFO, interrupts are generated if appropriate, and the RIDLE bit is set.

When the module receives a long Break signal and the receiver has detected the Start bit, the data bits and the invalid Stop bit (which sets the FERR), the receiver must wait for a valid Stop bit before looking for the next Start bit. It cannot assume that the Break condition on the line is the next Start bit.

Break is regarded as a character containing all ‘0’s with the FERR bit set. The Break character is loaded into the buffer. No further reception can occur until a Stop bit is received. Note that RIDLE goes high when the Stop bit has not yet been received.

16.6 Address Detect Mode

Setting the ADDEN bit (UxSTA<5>) enables this special mode in which a 9th bit (URX8) value of ‘1’ identifies the received word as an address, rather than data. This mode is only applicable for 9-bit data communication. The URXISEL control bit does not have any impact on interrupt generation in this mode since an interrupt (if enabled) is generated every time the received word has the 9th bit set.

16.7 Loopback Mode

Setting the LPBACK bit enables this special mode in which the UxTX pin is internally connected to the UxRX pin. When configured for the Loopback mode, the UxRX pin is disconnected from the internal UART receive logic. However, the UxTX pin still functions as in a normal operation.

To select this mode:

a) Configure UART for desired mode of operation.
b) Set LPBACK = 1 to enable Loopback mode.
c) Enable transmission as defined in **Section 16.3 “Transmitting Data”**.

16.8 Baud Rate Generator

The UART has a 16-bit Baud Rate Generator to allow maximum flexibility in baud rate generation. The Baud Rate Generator register (UxBRG) is readable and writable. The baud rate is computed as follows:

BRG = 16-bit value held in UxBRG register (0 through 65535)

F_{CY} = Instruction Clock Rate ($1/T_{CY}$)

The Baud Rate is given by Equation 16-1.

EQUATION 16-1: BAUD RATE

$$\text{Baud Rate} = F_{CY}/(16*(BRG+1))$$

Therefore, the maximum baud rate possible is:

$F_{CY}/16$ (if BRG = 0),

and the minimum baud rate possible is:

$F_{CY}/(16 * 65536)$.

With a full 16-bit Baud Rate Generator at 30 MIPS operation, the minimum baud rate achievable is 28.5 bps.

- Receive Error Interrupts:

  A receive error interrupt is indicated by the ERRIF bit. This bit shows that an error condition occurred. The source of the error can be determined by checking the bits in the CAN Interrupt register, CiINTF.

  - Invalid Message Received:

    If any type of error occurred during reception of the last message, an error is indicated by the IVRIF bit.

  - Receiver Overrun:

    The RXnOVR bit indicates that an overrun condition occurred.

  - Receiver Warning:

    The RXWAR bit indicates that the Receive Error Counter (RERRCNT<7:0>) has reached the warning limit of 96.

  - Receiver Error Passive:

    The RXEP bit indicates that the Receive Error Counter has exceeded the error passive limit of 127 and the module has gone into error passive state.

17.5 Message Transmission

17.5.1 TRANSMIT BUFFERS

The CAN module has three transmit buffers. Each of the three buffers occupies 14 bytes of data. Eight of the bytes are the maximum 8 bytes of the transmitted message. Five bytes hold the standard and extended identifiers and other message arbitration information.

17.5.2 TRANSMIT MESSAGE PRIORITY

Transmit priority is a prioritization within each node of the pending transmittable messages. There are 4 levels of transmit priority. If TXPRI<1:0> (CiTXnCON<1:0>, where n = 0, 1 or 2, represents a particular transmit buffer) for a particular message buffer is set to ‘`11`’, that buffer has the highest priority. If TXPRI<1:0> for a particular message buffer is set to ‘`10`’ or ‘`01`’, that buffer has an intermediate priority. If TXPRI<1:0> for a particular message buffer is ‘`00`’, that buffer has the lowest priority.

17.5.3 TRANSMISSION SEQUENCE

To initiate transmission of the message, the TXREQ bit (CiTXnCON<3>) must be set. The CAN bus module resolves any timing conflicts between setting of the TXREQ bit and the Start-of-Frame (SOF), ensuring that if the priority was changed, it is resolved correctly before the SOF occurs. When TXREQ is set, the TXABT (CiTXnCON<6>), TXLARB (CiTXnCON<5>) and TXERR (CiTXnCON<4>) flag bits are automatically cleared.

Setting TXREQ bit simply flags a message buffer as enqueued for transmission. When the module detects an available bus, it begins transmitting the message which has been determined to have the highest priority.

If the transmission completes successfully on the first attempt, the TXREQ bit is cleared automatically and an interrupt is generated if TX1IE was set.

If the message transmission fails, one of the error condition flags is set, and the TXREQ bit remains set, indicating that the message is still pending for transmission. If the message encountered an error condition during the transmission attempt, the TXERR bit is set, and the error condition may cause an interrupt. If the message loses arbitration during the transmission attempt, the TXLARB bit is set. No interrupt is generated to signal the loss of arbitration.

17.5.4 ABORTING MESSAGE TRANSMISSION

The system can also abort a message by clearing the TXREQ bit associated with each message buffer. Setting the ABAT bit (CiCTRL<12>) requests an abort of all pending messages. If the message has not yet started transmission, or if the message started but is interrupted by loss of arbitration or an error, the abort is processed. The abort is indicated when the module sets the TXABT bit and the TXnIF flag is not automatically set.

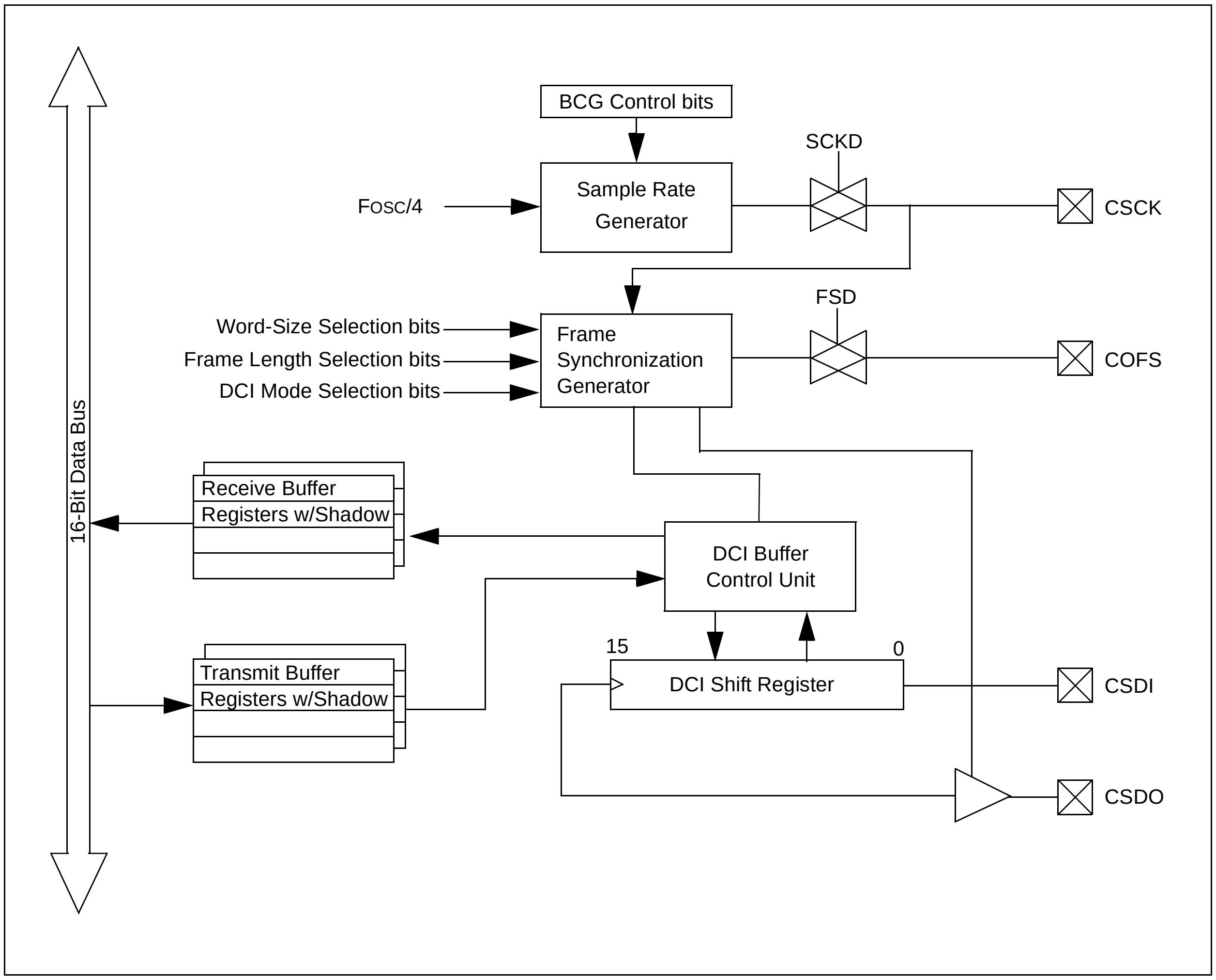
17.5.5 TRANSMISSION ERRORS

The CAN module detects the following transmission errors:

- Acknowledge error
- Form error
- Bit error

These transmission errors do not necessarily generate an interrupt but are indicated by the transmission error counter. However, each of these errors causes the transmission error counter to be incremented by one. Once the value of the error counter exceeds the value of 96, the ERRIF (CiINTF<5>) and the TXWAR bit (CiINTF<10>) are set. Once the value of the error counter exceeds the value of 96, an interrupt is generated and the TXWAR bit in the Error Flag register is set.

FIGURE 18-1: DCI MODULE BLOCK DIAGRAM



19.1 A/D Result Buffer

The module contains a 16-word, dual port read-only buffer, called ADCBUF0...ADCBUFF, to buffer the A/D results. The RAM is 12 bits wide but the data obtained is represented in one of four different 16-bit data formats. The contents of the sixteen A/D Conversion Result Buffer registers, ADCBUF0 through ADCBUFF, cannot be written by user software.

19.2 Conversion Operation

After the A/D module has been configured, the sample acquisition is started by setting the SAMP bit. Various sources, such as a programmable bit, timer time-outs and external events, terminate acquisition and start a conversion. When the A/D conversion is complete, the result is loaded into ADCBUF0...ADCBUFF, and the DONE bit and the A/D Interrupt Flag, ADIF, are set after the number of samples specified by the SMPI bit. The ADC module can be configured for different interrupt rates as described in **Section 19.3 “Selecting the Conversion Sequence”**.

The following steps should be followed for doing an A/D conversion:

1. Configure the A/D module:
   - Configure analog pins, voltage reference and digital I/O
   - Select A/D input channels
   - Select A/D conversion clock
   - Select A/D conversion trigger
   - Turn on A/D module
2. Configure A/D interrupt (if required):
   - Clear ADIF bit
   - Select A/D interrupt priority
   - Set ADIE bit (for ISR processing)
3. Start sampling
4. Wait the required acquisition time
5. Trigger acquisition end, start conversion:
6. Wait for A/D conversion to complete, by either:
   - Waiting for the A/D interrupt, or
   - Waiting for the DONE bit to get set.
7. Read A/D result buffer, clear ADIF if required

19.3 Selecting the Conversion Sequence

Several groups of control bits select the sequence in which the A/D connects inputs to the sample/hold channel, converts a channel, writes the buffer memory and generates interrupts.

The sequence is controlled by the sampling clocks.

The SMPI bits select the number of acquisition/conversion sequences that would be performed before an interrupt occurs. This can vary from 1 sample per interrupt to 16 samples per interrupt.

The BUFM bit splits the 16-word results buffer into two 8-word groups. Writing to the 8-word buffers is alternated on each interrupt event.

Use of the BUFM bit depends on how much time is available for moving the buffers after the interrupt.

If the processor can quickly unload a full buffer within the time it takes to acquire and convert one channel, the BUFM bit can be ‘`0`’ and up to 16 conversions (corresponding to the 16 input channels) may be done per interrupt. The processor has one acquisition and conversion time to move the sixteen conversions.

If the processor cannot unload the buffer within the acquisition and conversion time, the BUFM bit should be ‘`1`’. For example, if SMPI<3:0> (ADCON2<5:2>) = `0111`, then eight conversions are loaded into 1/2 of the buffer, following which an interrupt occurs. The next eight conversions are loaded into the other 1/2 of the buffer. The processor has the entire time between interrupts to move the eight conversions.

The ALTS bit can be used to alternate the inputs selected during the sampling sequence. The input multiplexer has two sets of sample inputs: MUX A and MUX B. If the ALTS bit is ‘`0`’, only the MUX A inputs are selected for sampling. If the ALTS bit is ‘`1`’ and SMPI<3:0> = `0000` on the first sample/convert sequence, the MUX A inputs are selected and on the next acquire/convert sequence, the MUX B inputs are selected.

The CSCNA bit (ADCON2<10>) allows the S/H input to be sequentially scanned across a selected number of analog inputs for the MUX A group. The inputs are selected by the ADCSSL register. If a particular bit in the ADCSSL register is ‘`1`’, the corresponding input is selected. The inputs are always scanned from lower to higher numbered inputs, starting after each interrupt. If the number of inputs selected is greater than the number of samples taken per interrupt, the higher numbered inputs are unused.

Note:	The ADCHS, ADPCFG and ADCSSL registers allow the application to configure AN13-AN15 as analog input pins. Since these pins are not physically present on the device, conversion results from these pins read ‘`0`’.

TABLE 21-2: INSTRUCTION SET OVERVIEW (CONTINUED)

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles	Status Flags Affected
48	`MPY`	`MPY Wm*Wn,Acc,Wx,Wxd,Wy,Wyd`	Multiply Wm by Wn to Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
		`MPY Wm*Wm,Acc,Wx,Wxd,Wy,Wyd`	Square Wm to Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
49	`MPY.N`	`MPY.N Wm*Wn,Acc,Wx,Wxd,Wy,Wyd`	-(Multiply Wm by Wn) to Accumulator	1	1	None
50	`MSC`	`MSC Wm*Wm,Acc,Wx,Wxd,Wy,Wyd, AWB`	Multiply and Subtract from Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
51	`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
52	`NEG`	`NEG Acc`	Negate Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
		`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
53	`NOP`	`NOP`	No Operation	1	1	None
		`NOPR`	No Operation	1	1	None
54	`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
55	`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
56	`PWRSAV`	`PWRSAV #lit1`	Go into Sleep or Idle mode	1	1	WDTO, Sleep
57	`RCALL`	`RCALL Expr`	Relative Call	1	2	None
		`RCALL Wn`	Computed Call	1	2	None
58	`REPEAT`	`REPEAT #lit14`	Repeat Next Instruction lit14+1 Times	1	1	None
		`REPEAT Wn`	Repeat Next Instruction (Wn)+1 Times	1	1	None
59	`RESET`	`RESET`	Software Device Reset	1	1	None
60	`RETFIE`	`RETFIE`	Return from Interrupt	1	3 (2)	None
61	`RETLW`	`RETLW #lit10,Wn`	Return with Literal in Wn	1	3 (2)	None
62	`RETURN`	`RETURN`	Return from Subroutine	1	3 (2)	None
63	`RLC`	`RLC f`	f = Rotate Left through Carry f	1	1	C,N,Z
		`RLC f,WREG`	WREG = Rotate Left through Carry f	1	1	C,N,Z
		`RLC Ws,Wd`	Wd = Rotate Left through Carry Ws	1	1	C,N,Z
64	`RLNC`	`RLNC f`	f = Rotate Left (No Carry) f	1	1	N,Z
		`RLNC f,WREG`	WREG = Rotate Left (No Carry) f	1	1	N,Z
		`RLNC Ws,Wd`	Wd = Rotate Left (No Carry) Ws	1	1	N,Z
65	`RRC`	`RRC f`	f = Rotate Right through Carry f	1	1	C,N,Z
		`RRC f,WREG`	WREG = Rotate Right through Carry f	1	1	C,N,Z
		`RRC Ws,Wd`	Wd = Rotate Right through Carry Ws	1	1	C,N,Z

22.11 PICkit 2 Development Programmer/Debugger and PICkit 2 Debug Express

The PICkit™ 2 Development Programmer/Debugger is a low-cost development tool with an easy to use interface for programming and debugging Microchip's Flash families of microcontrollers. The full featured Windows® programming interface supports baseline (PIC10F, PIC12F5xx, PIC16F5xx), midrange (PIC12F6xx, PIC16F), PIC18F, PIC24, dsPIC30, dsPIC33, and PIC32 families of 8-bit, 16-bit, and 32-bit microcontrollers, and many Microchip Serial EEPROM products. With Microchip's powerful MPLAB Integrated Development Environment (IDE) the PICkit™ 2 enables in-circuit debugging on most PIC® microcontrollers. In-Circuit-Debugging runs, halts and single steps the program while the PIC microcontroller is embedded in the application. When halted at a breakpoint, the file registers can be examined and modified.

The PICkit 2 Debug Express include the PICkit 2, demo board and microcontroller, hookup cables and CDROM with user's guide, lessons, tutorial, compiler and MPLAB IDE software.

22.12 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at VDDMIN and VDDMAX for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages and a modular, detachable socket assembly to support various package types. The ICSP™ cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices and incorporates an MMC card for file storage and data applications.

22.13 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.

TABLE 23-5: DC CHARACTERISTICS: OPERATING CURRENT (IDD)

DC CHARACTERISTICS			**Standard Operating Conditions: 2.5V to 5.5V (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for Industrial; -40°C ≤ TA ≤ +125°C for Extended			
Parameter No.	**Typical**	**Max**	**Units**	**Conditions**		
Operating Current (IDD)[1]						
DC31a	2	4	mA	25°C	3.3V	0.128 MIPS LPRC (512 kHz)
DC31b	2	4	mA	85°C		
DC31c	2	4	mA	125°C		
DC31e	4	6	mA	25°C	5V	
DC31f	4	6	mA	85°C		
DC31g	4	6	mA	125°C		
DC30a	6	11	mA	25°C	3.3V	1.8 MIPS FRC (7.37 MHz)
DC30b	6	11	mA	85°C		
DC30c	7	11	mA	125°C		
DC30e	11	16	mA	25°C	5V	
DC30f	11	16	mA	85°C		
DC30g	11	16	mA	125°C		
DC23a	13	20	mA	25°C	3.3V	4 MIPS
DC23b	13	20	mA	85°C		
DC23c	14	20	mA	125°C		
DC23e	22	31	mA	25°C	5V	
DC23f	22	31	mA	85°C		
DC23g	22	31	mA	125°C		
DC24a	27	39	mA	25°C	3.3V	10 MIPS
DC24b	28	39	mA	85°C		
DC24c	28	39	mA	125°C		
DC24e	46	64	mA	25°C	5V	
DC24f	46	64	mA	85°C		
DC24g	46	64	mA	125°C		
DC27d	86	120	mA	25°C	5V	20 MIPS
DC27e	85	120	mA	85°C		
DC27f	85	120	mA	125°C		
DC29a	123	170	mA	25°C	5V	30 MIPS
DC29b	122	170	mA	85°C		

Note 1: The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption. The test conditions for all IDD measurements are as follows: OSC1 driven with external square wave from rail-to-rail. All I/O pins are configured as inputs and pulled to VDD. $\overline{\text{MCLR}}$ = VDD, WDT, FSCM, LVD and BOR are disabled. CPU, SRAM, program memory and data memory are operational. No peripheral modules are operating.

TABLE 23-8: DC CHARACTERISTICS: I/O PIN INPUT SPECIFICATIONS

DC CHARACTERISTICS			**Standard Operating Conditions: 2.5V to 5.5V (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for Industrial -40°C ≤ TA ≤ +125°C for Extended				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ(1)**	**Max**	**Units**	**Conditions**
	VIL	**Input Low Voltage(2)**					
DI10		I/O Pins: with Schmitt Trigger Buffer	VSS	—	0.2 VDD	V	
DI15		$\overline{\text{MCLR}}$	VSS	—	0.2 VDD	V	
DI16		OSC1 (in XT, HS and LP modes)	VSS	—	0.2 VDD	V	
DI17		OSC1 (in RC mode)(3)	VSS	—	0.3 VDD	V	
DI18		SDA, SCL	VSS	—	0.3 VDD	V	SM bus disabled
DI19		SDA, SCL	VSS	—	0.8	V	SM bus enabled
	VIH	**Input High Voltage(2)**					
DI20		I/O Pins: with Schmitt Trigger Buffer	0.8 VDD	—	VDD	V	
DI25		$\overline{\text{MCLR}}$	0.8 VDD	—	VDD	V	
DI26		OSC1 (in XT, HS and LP modes)	0.7 VDD	—	VDD	V	
DI27		OSC1 (in RC mode)(3)	0.9 VDD	—	VDD	V	
DI28		SDA, SCL	0.7 VDD	—	VDD	V	SM bus disabled
DI29		SDA, SCL	2.1	—	VDD	V	SM bus enabled
	ICNPU	**CNxx Pull-up Current(2)**					
DI30			50	250	400	μA	VDD = 5V, VPIN = VSS
	IIL	**Input Leakage Current(2,4,5)**					
DI50		I/O Ports	—	0.01	±1	μA	VSS ≤ VPIN ≤ VDD, Pin at high-impedance
DI51		Analog Input Pins	—	0.50	—	μA	VSS ≤ VPIN ≤ VDD, Pin at high-impedance
DI55		$\overline{\text{MCLR}}$	—	0.05	±5	μA	VSS ≤ VPIN ≤ VDD
DI56		OSC1	—	0.05	±5	μA	VSS ≤ VPIN ≤ VDD, XT, HS and LP Osc mode

Note 1: Data in “Typ” column is at 5V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

2: These parameters are characterized but not tested in manufacturing.

3: In RC oscillator configuration, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the dsPIC30F device be driven with an external clock while in RC mode.

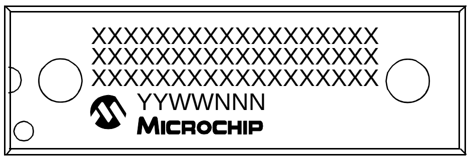
4: The leakage current on the $\overline{\text{MCLR}}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

5: Negative current is defined as current sourced by the pin.

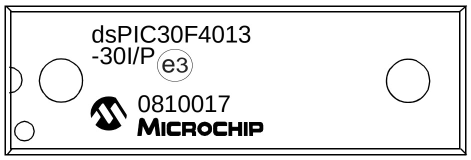
24.0 PACKAGING INFORMATION

24.1 Package Marking Information

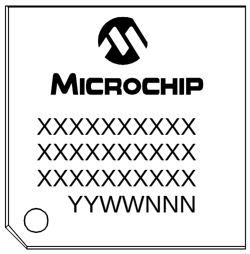
40-Lead PDIP



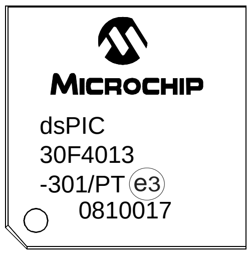
Example



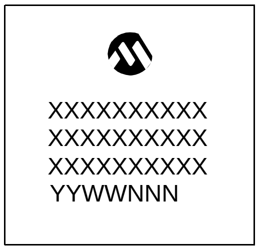
44-Lead TQFP



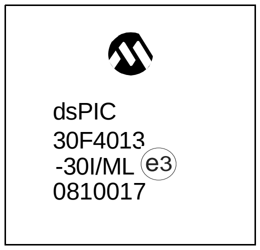
Example



44-Lead QFN



Example

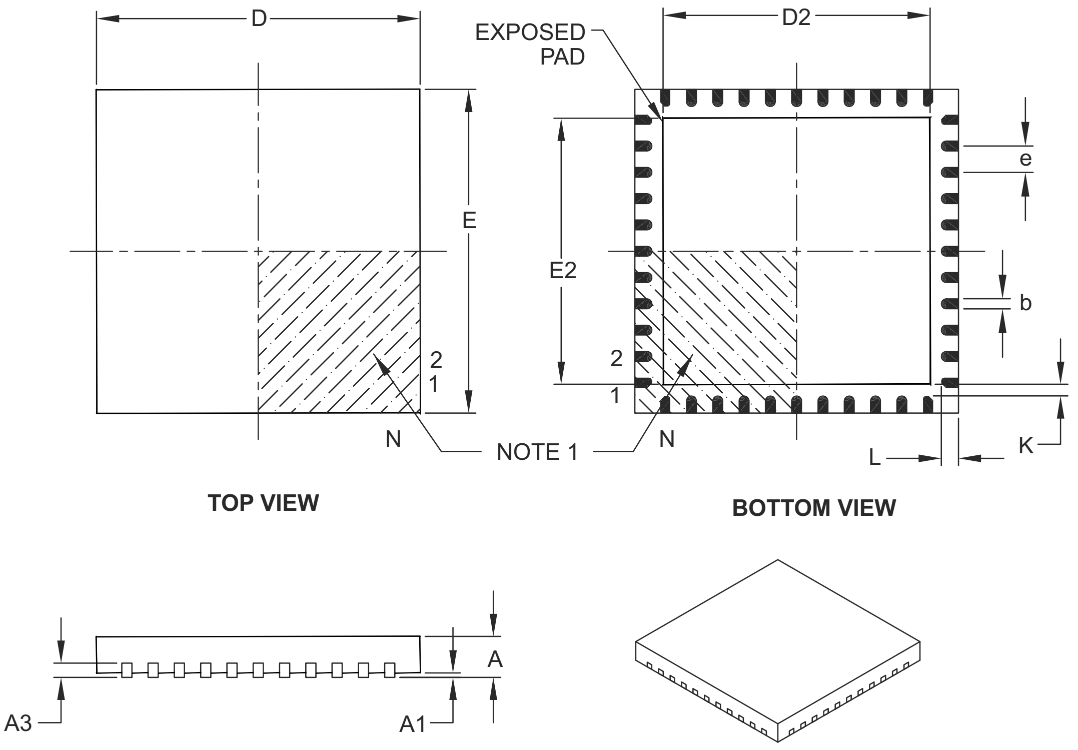


Legend:	XX...X	Customer-specific information
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week '01')
	NNN	Alphanumeric traceability code
	e3	Pb-free JEDEC designator for Matte Tin (Sn)
	*	This package is Pb-free. The Pb-free JEDEC designator (e3) can be found on the outer packaging for this package.

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

44-Lead Plastic Quad Flat, No Lead Package (ML) – 8x8 mm Body [QFN]

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



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		44	
Pitch	e		0.65 BSC	
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Contact Thickness	A3		0.20 REF	
Overall Width	E		8.00 BSC	
Exposed Pad Width	E2	6.30	6.45	6.80
Overall Length	D		8.00 BSC	
Exposed Pad Length	D2	6.30	6.45	6.80
Contact Width	b	0.25	0.30	0.38
Contact Length	L	0.30	0.40	0.50
Contact-to-Exposed Pad	K	0.20	–	–

Notes:
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. Dimensioning and tolerancing per ASME Y14.5M.
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-103B

APPENDIX A: REVISION HISTORY

Revision D (June 2006)

Previous versions of this data sheet contained Advance or Preliminary Information. They were distributed with incomplete characterization data.

This revision reflects these changes:

- Revised I^{2}C Slave Addresses (see Table 14-1)
- Updated example for ADC Conversion Clock selection (see **Section 19.0 “12-bit Analog-to-Digital Converter (ADC) Module”**)
- Base instruction CP1 eliminated from instruction set (seeTable 21-2)
- Revised electrical characteristics:
  - Operating Current (IDD) Specifications (see Table 23-5)
  - Idle Current (IIDLE) Specifications (see Table 23-6)
  - Power-down Current (IPD) Specifications (see Table 23-7)
  - I/O pin Input Specifications (see Table 23-8)
  - Brown Out Reset (BOR) Specifications (see Table 23-11)
  - Watchdog Timer time-out limits (see Table 23-20)

Revision E (January 2007)

This revision includes updates to the packaging diagrams.

Revision F (April 2008)

This revision reflects these updates:

- Added FUSE Configuration Register (FICD) details (see **Section 20.8 “Device Configuration Registers”** and Table 20-8)
- Added Note 2 in Device Configuration Registers table (Table 20-8)
- Removed erroneous statement regarding generation of CAN receive errors (see **Section 17.4.5 “Receive Errors”**)
- Updated ADC Conversion Clock and Sampling Rate Calculation (see Example 19-1). Minimum TAD is 334 nsec.
- Updated details related to the Input Change Notification module:
  - Updated last sentence in the first paragraph of **Section 7.3 “Input Change Notification Module”**
  - Updated Table 7-2
  - Removed Table 7-3, Table 7-4, and Table 7-5
- Electrical Specifications:
  - Resolved TBD values for parameters DO10, DO16, DO20, and DO26 (see Table 23-9)
  - 10-bit High-Speed ADC tPDU timing parameter (time to stabilize) has been updated from 20 µs typical to 20 µs maximum (see Table 23-38)
  - Parameter OS65 (Internal RC Accuracy) has been expanded to reflect multiple Min and Max values for different temperatures (see Table 23-18)
  - Parameter DC12 (RAM Data Retention Voltage) has been updated to include a Min value (see Table 23-4)
  - Parameter D134 (Erase/Write Cycle Time) has been updated to include Min and Max values and the Typ value has been removed (see Table 23-12)
  - Removed parameters OS62 (Internal FRC Jitter) and OS64 (Internal FRC Drift) and Note 2 from AC Characteristics (see Table 23-17)
  - Parameter OS63 (Internal FRC Accuracy) has been expanded to reflect multiple Min and Max values for different temperatures (see Table 23-17)
  - Removed parameters DC27a, DC27b, DC47a, and DC47b (references to IDD, 20 MIPs @ 3.3V) in Table 23-5 and Table 23-6
  - Removed parameters CS77 and CS78 (references to TRACL and TFACL @ 3.3V) in Table 23-29
  - Updated Min and Max values and Conditions for parameter SY11 and updated Min, Typ, and Max values and Conditions for parameter SY20 (see Table 23-20)
- Additional minor corrections throughout the document