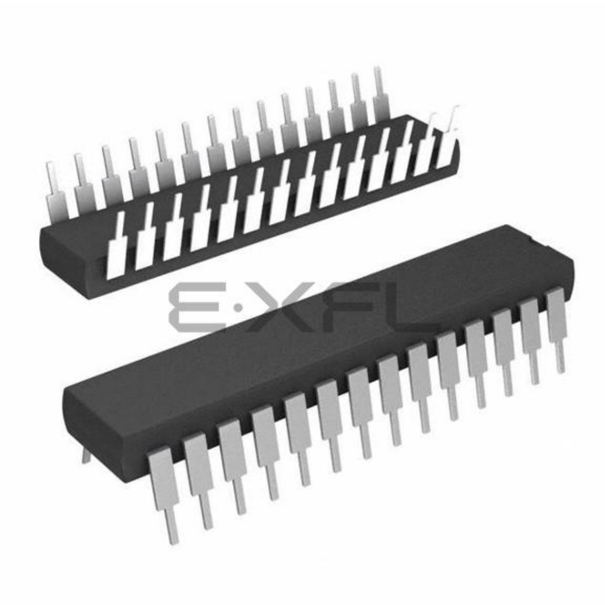


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

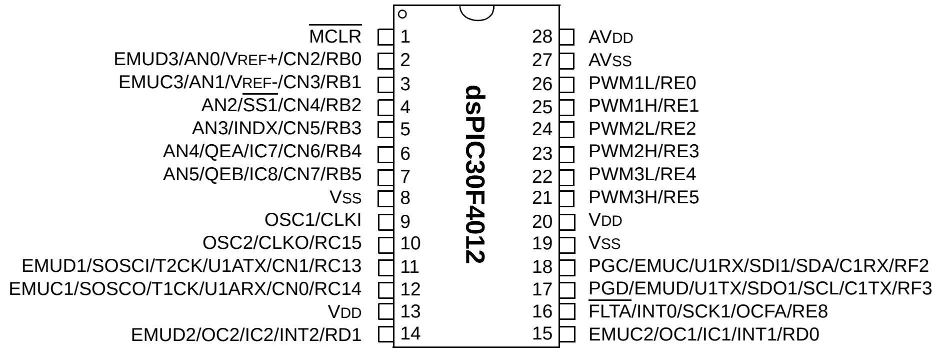
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded"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

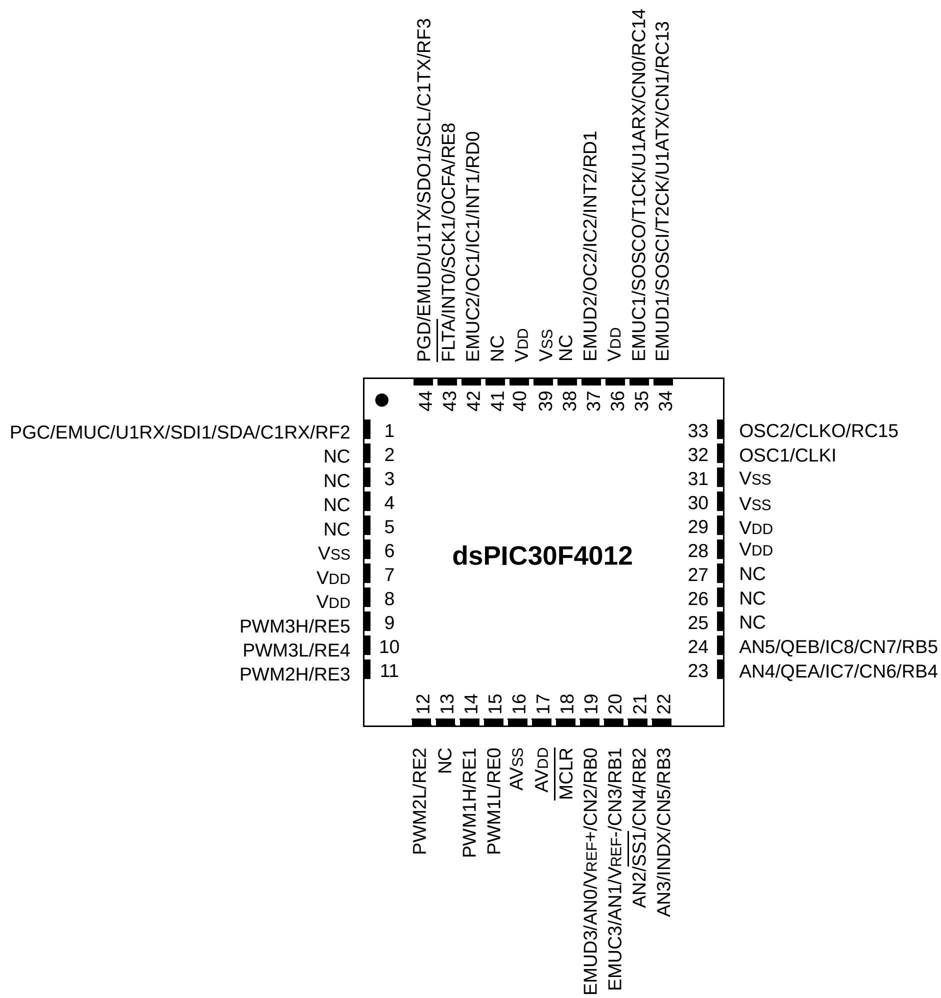
Details	
Product Status	Active
Core Processor	dsPIC
Core Size	16-Bit
Speed	30 MIPs
Connectivity	CANbus, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, Motor Control PWM, QEI, POR, PWM, WDT
Number of I/O	20
Program Memory Size	48KB (16K x 24)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	A/D 6x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.300", 7.62mm)
Supplier Device Package	28-SPDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/dspic30f4012-30i-sp

Pin Diagrams (Continued)

28-Pin SPDIP and SOIC

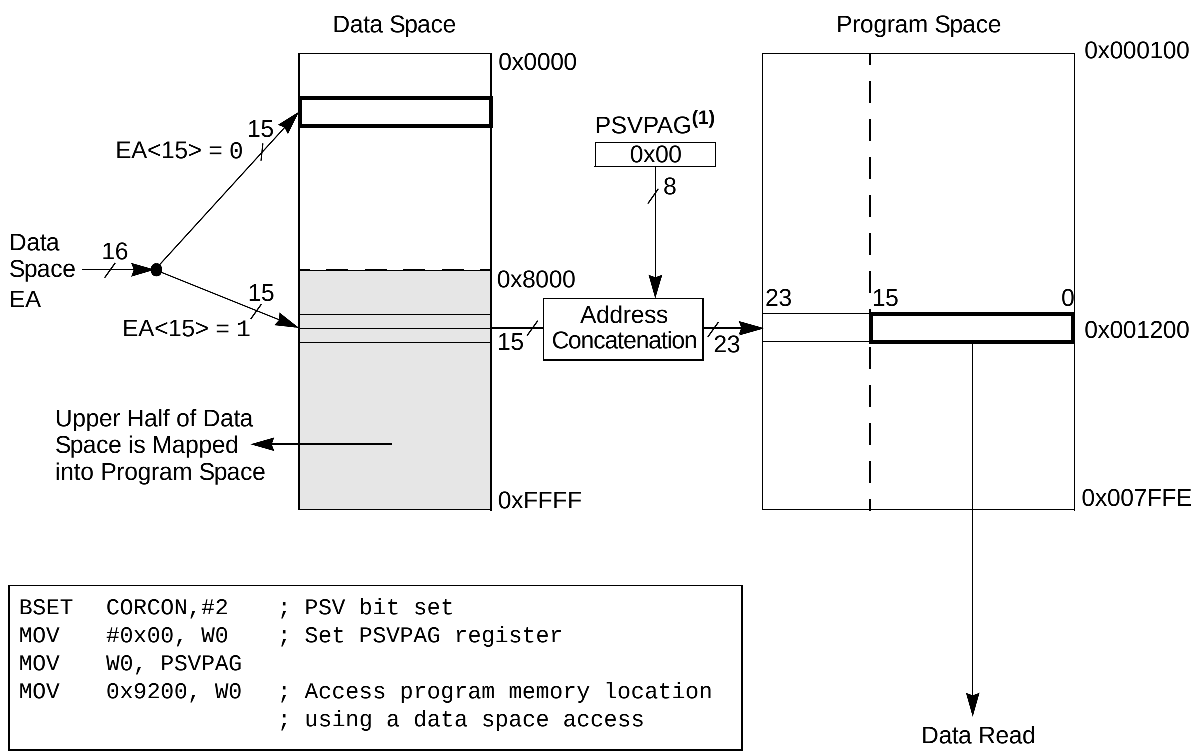


44-Pin QFN(1)



Note 1: The metal plane at the bottom of the device is not connected to any pins and is recommended to be connected to VSS externally.

FIGURE 3-5: DATA SPACE WINDOW INTO PROGRAM SPACE OPERATION




```
BSET   CORCON,#2    ; PSV bit set
MOV    #0x00, W0    ; Set PSVPAG register
MOV    W0, PSVPAG
MOV    0x9200, W0   ; Access program memory location
                    ; using a data space access
```

Note: PSVPAG is an 8-bit register containing bits <22:15> of the program space address (i.e., it defines the page in program space to which the upper half of data space is being mapped).

3.2 Data Address Space

The core has two data spaces. The data spaces can be considered either separate (for some DSP instructions), or as one unified linear address range (for MCU instructions). The data spaces are accessed using two Address Generation Units (AGUs) and separate data paths.

3.2.1 DATA SPACE MEMORY MAP

The data space memory is split into two blocks, X and Y data space. A key element of this architecture is that Y space is a subset of X space, and is fully contained within X space. In order to provide an apparent linear addressing space, X and Y spaces have contiguous addresses.

When executing any instruction other than one of the MAC class of instructions, the X block consists of the 64-Kbyte data address space (including all Y addresses). When executing one of the MAC class of instructions, the X block consists of the 64-Kbyte data address space excluding the Y address block (for data reads only). In other words, all other instructions regard the entire data memory as one composite address space. The MAC class instructions extract the Y address space from data space and address it using EAs sourced from W10 and W11. The remaining X data space is addressed using W8 and W9. Both address spaces are concurrently accessed only with the MAC class instructions.

A data space memory map is shown in Figure 3-6.

Figure 3-7 illustrates a graphical summary of how X and Y data spaces are accessed for MCU and DSP instructions.

6.0 FLASH PROGRAM MEMORY

Note:	This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *"16-bit MCU and DSC Reference Manual"* (DS70157).

The dsPIC30F family of devices contains internal program Flash memory for executing user code. There are two methods by which the user can program this memory:

1. In-Circuit Serial Programming™ (ICSP™)
2. Run-Time Self-Programming (RTSP)

6.1 In-Circuit Serial Programming (ICSP)

dsPIC30F devices can be serially programmed while in the end application circuit. This is simply done with two lines for Programming Clock and Programming Data (which are named PGC and PGD, respectively), and three other lines for Power (V_{DD}), Ground (V_{SS}) and Master Clear ($\overline{MCLR}$). This allows customers to manufacture boards with unprogrammed devices, and then program the digital signal controller just before shipping the product. This also allows the most recent firmware or a custom firmware to be programmed.

6.2 Run-Time Self-Programming (RTSP)

RTSP is accomplished using `TBLRD` (table read) and `TBLWT` (table write) instructions.

With RTSP, the user may erase program memory, 32 instructions (96 bytes) at a time, and can write program memory data, 32 instructions (96 bytes) at a time.

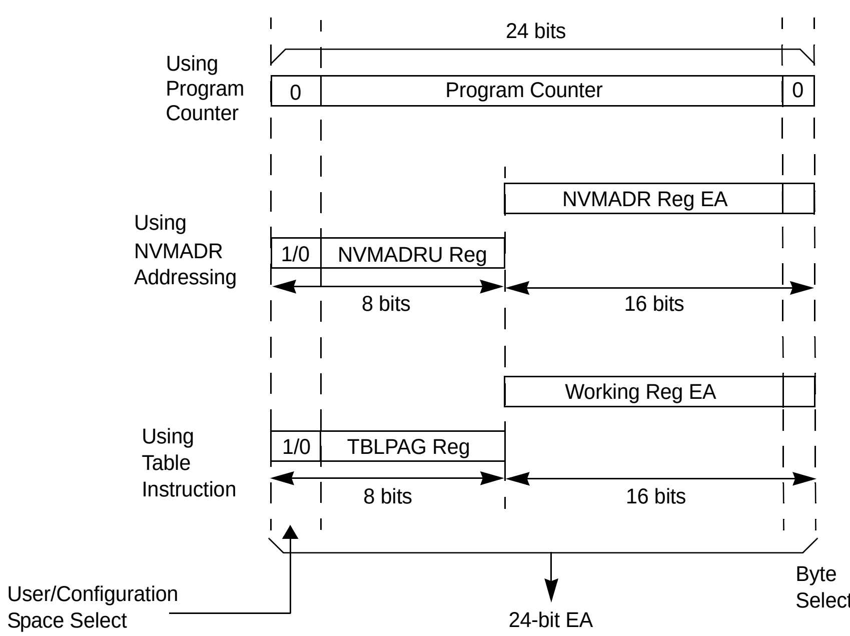
6.3 Table Instruction Operation Summary

The `TBLRDL` and the `TBLWTL` instructions are used to read or write to bits <15:0> of program memory. `TBLRDL` and `TBLWTL` can access program memory in Word or Byte mode.

The `TBLRDH` and `TBLWTH` instructions are used to read or write to bits<23:16> of program memory. `TBLRDH` and `TBLWTH` can access program memory in Word or Byte mode.

A 24-bit program memory address is formed using bits<7:0> of the TBLPAG register and the Effective Address (EA) from a W register, specified in the table instruction, as shown in Figure 6-1.

FIGURE 6-1: ADDRESSING FOR TABLE AND NVM REGISTERS



6.6 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.

6.6.1 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 will begin erase cycle.
   f) CPU will stall 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 will begin program cycle.
   e) CPU will stall 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.

6.6.2 ERASING A ROW OF PROGRAM MEMORY

Example 6-1 shows a code sequence that can be used to erase a row (32 instructions) of program memory.

EXAMPLE 6-1: 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                   ; Intialize 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
```

6.6.3 LOADING WRITE LATCHES

Example 6-2 shows a sequence of instructions that can be used to load the 96 bytes of write latches. 32 TBLWTL and 32 TBLWTH instructions are needed to load the write latches selected by the Table Pointer.

EXAMPLE 6-2: LOADING WRITE LATCHES

```
; Set up a pointer to the first program memory location to be written
; program memory selected, and writes enabled
       MOV    #0x0000,W0                ;
       MOV    W0,TBLPAG                 ; Initialize PM Page Boundary SFR
       MOV    #0x6000,W0                ; An example program memory address
; Perform the TBLWT instructions to write the latches
; 0th_program_word
       MOV    #LOW_WORD_0,W2            ;
       MOV    #HIGH_BYTE_0,W3           ;
       TBLWTL W2,[W0]                   ; Write PM low word into program latch
       TBLWTH W3,[W0++]                 ; Write PM high byte into program latch
; 1st_program_word
       MOV    #LOW_WORD_1,W2            ;
       MOV    #HIGH_BYTE_1,W3           ;
       TBLWTL W2,[W0]                   ; Write PM low word into program latch
       TBLWTH W3,[W0++]                 ; Write PM high byte into program latch
;  2nd_program_word
       MOV    #LOW_WORD_2,W2            ;
       MOV    #HIGH_BYTE_2,W3           ;
       TBLWTL W2, [W0]                  ; Write PM low word into program latch
       TBLWTH W3, [W0++]                ; Write PM high byte into program latch
       •
       •
       •
; 31st_program_word
       MOV    #LOW_WORD_31,W2           ;
       MOV    #HIGH_BYTE_31,W3          ;
       TBLWTL W2, [W0]                  ; Write PM low word into program latch
       TBLWTH W3, [W0++]                ; Write PM high byte into program latch
```

Note: In Example 6-2, the contents of the upper byte of W3 have no effect.

6.6.4 INITIATING THE PROGRAMMING SEQUENCE

For protection, 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.

EXAMPLE 6-3: INITIATING A PROGRAMMING SEQUENCE

```
       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
```

TABLE 6-1: NVM REGISTER MAP(1)

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
NVMCON	0760	WR	WREN	WRERR	—	—	—	—	TWRI	—	PROGOP<6:0>							`0000 0000 0000 0000`
NVMADR	0762	NVMADR<15:0>																`uuuu uuuu uuuu uuuu`
NVMADRU	0764	—	—	—	—	—	—	—	—	—	NVMADR<22:16>							`0000 0000 uuuu uuuu`
NVMKEY	0766	—	—	—	—	—	—	—	—	KEY<7:0>								`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.

7.3 Writing to the Data EEPROM

To write an EEPROM data location, the following sequence must be followed:

1. Erase data EEPROM word.
   a) Select word, data EEPROM; erase and set WREN bit in NVMCON register.
   b) Write address of word to be erased into NVMADRU/NVMADR.
   c) Enable NVM interrupt (optional).
   d) Write 0x55 to NVMKEY.
   e) Write 0xAA to NVMKEY.
   f) Set the WR bit. This will begin erase cycle.
   g) Either poll NVMIF bit or wait for NVMIF interrupt.
   h) The WR bit is cleared when the erase cycle ends.
2. Write data word into data EEPROM write latches.
3. Program 1 data word into data EEPROM.
   a) Select word, data EEPROM; program and set WREN bit in NVMCON register.
   b) Enable NVM write done interrupt (optional).
   c) Write 0x55 to NVMKEY.
   d) Write 0xAA to NVMKEY.
   e) Set the WR bit. This will begin program cycle.
   f) Either poll NVMIF bit or wait for NVM interrupt.
   g) The WR bit is cleared when the write cycle ends.

The write will not initiate if the above sequence is not exactly followed (write 0x55 to NVMKEY, write 0xAA to NVMCON, then set WR bit) for each word. It is strongly recommended that interrupts be disabled during this code segment.

Additionally, the WREN bit in NVMCON must be set to enable writes. This mechanism prevents accidental writes to data EEPROM due to unexpected code execution. The WREN bit should be kept clear at all times except when updating the EEPROM. The WREN bit is not cleared by hardware.

After a write sequence has been initiated, clearing the WREN bit will not affect the current write cycle. The WR bit will be inhibited from being set unless the WREN bit is set. The WREN bit must be set on a previous instruction. Both WR and WREN cannot be set with the same instruction.

At the completion of the write cycle, the WR bit is cleared in hardware and the Nonvolatile Memory Write Complete Interrupt Flag bit (NVMIF) is set. The user may either enable this interrupt or poll this bit. NVMIF must be cleared by software.

7.3.1 WRITING A WORD OF DATA EEPROM

Once the user has erased the word to be programmed, then a table write instruction is used to write one write latch, as shown in Example 7-4.

EXAMPLE 7-4: DATA EEPROM WORD WRITE

```
; Point to data memory
    MOV             #LOW_ADDR_WORD,W0                ; Init pointer
    MOV             #HIGH_ADDR_WORD,W1
    MOV             W1,TBLPAG
    MOV             #LOW(WORD),W2                    ; Get data
    TBLWTL          W2,[ W0]                         ; Write data
; The NVMADR captures last table access address
; Select data EEPROM for 1 word op
    MOV             #0x4004,W0
    MOV             W0,NVMCON

; Operate key to allow write operation
    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                       ; Initiate program sequence
    NOP
    NOP
; Write cycle will complete in 2mS. CPU is not stalled for the Data Write Cycle
; User can poll WR bit, use NVMIF or Timer IRQ to determine write complete
```

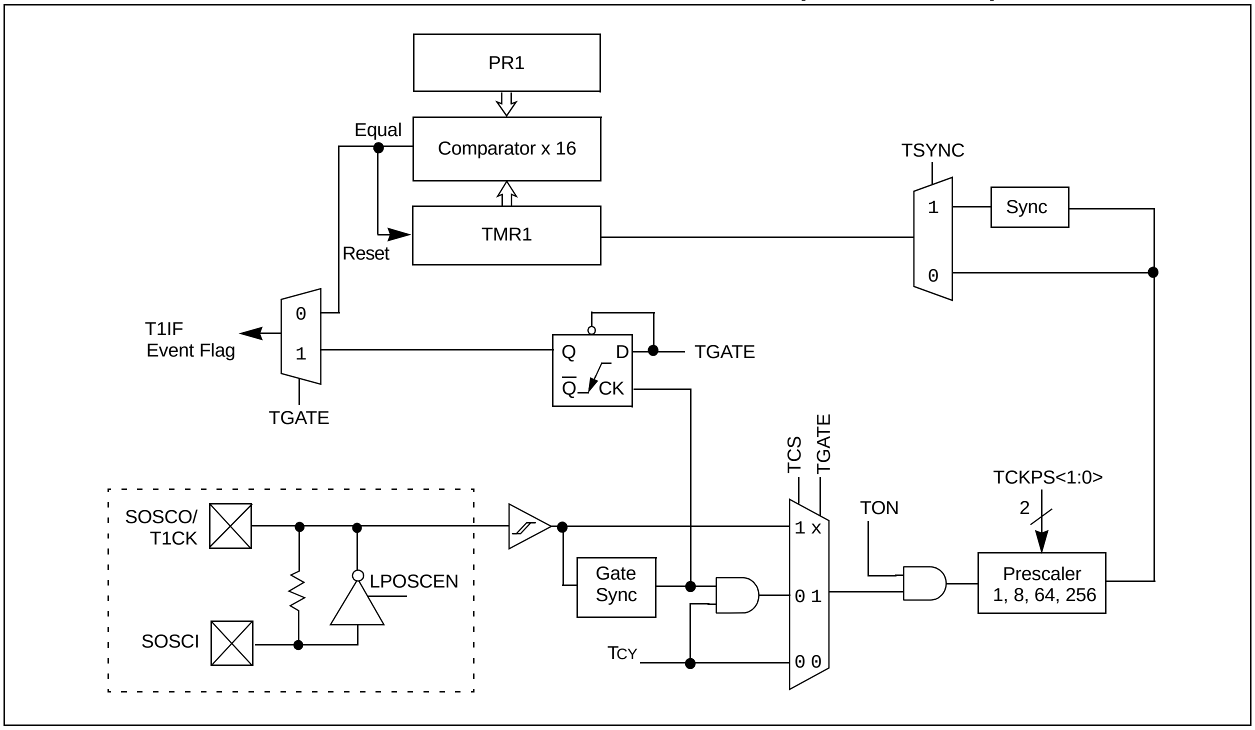
7.3.2 WRITING A BLOCK OF DATA EEPROM

To write a block of data EEPROM, write to all sixteen latches first, then set the NVMCON register and program the block.

EXAMPLE 7-5: DATA EEPROM BLOCK WRITE

```
    MOV       #LOW_ADDR_WORD,W0  ; Init pointer
    MOV       #HIGH_ADDR_WORD,W1
    MOV       W1,TBLPAG
    MOV       #data1,W2           ; Get 1st data
    TBLWTL    W2,[W0]++           ; write data
    MOV       #data2,W2           ; Get 2nd data
    TBLWTL    W2,[W0]++           ; write data
    MOV       #data3,W2           ; Get 3rd data
    TBLWTL    W2,[W0]++           ; write data
    MOV       #data4,W2           ; Get 4th data
    TBLWTL    W2,[W0]++           ; write data
    MOV       #data5,W2           ; Get 5th data
    TBLWTL    W2,[W0]++           ; write data
    MOV       #data6,W2           ; Get 6th data
    TBLWTL    W2,[W0]++           ; write data
    MOV       #data7,W2           ; Get 7th data
    TBLWTL    W2,[W0]++           ; write data
    MOV       #data8,W2           ; Get 8th data
    TBLWTL    W2,[W0]++           ; write data
    MOV       #data9,W2           ; Get 9th data
    TBLWTL    W2,[W0]++           ; write data
    MOV       #data10,W2          ; Get 10th data
    TBLWTL    W2,[W0]++           ; write data
    MOV       #data11,W2          ; Get 11th data
    TBLWTL    W2,[W0]++           ; write data
    MOV       #data12,W2          ; Get 12th data
    TBLWTL    W2,[W0]++           ; write data
    MOV       #data13,W2          ; Get 13th data
    TBLWTL    W2,[W0]++           ; write data
    MOV       #data14,W2          ; Get 14th data
    TBLWTL    W2,[W0]++           ; write data
    MOV       #data15,W2          ; Get 15th data
    TBLWTL    W2,[W0]++           ; write data
    MOV       #data16,W2          ; Get 16th data
    TBLWTL    W2,[W0]++           ; write data. The NVMADR captures last table access address.
    MOV       #0x400A,W0          ; Select data EEPROM for multi word op
    MOV       W0,NVMCON           ; Operate Key to allow program operation
    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 write cycle
    NOP
    NOP
```

FIGURE 9-1: 16-BIT TIMER1 MODULE BLOCK DIAGRAM (TYPE A TIMER)



9.1 Timer Gate Operation

The 16-bit timer can be placed in the Gated Time Accumulation mode. This mode allows the internal TCY to increment the respective timer when the gate input signal (T1CK pin) is asserted high. Control bit, TGATE (T1CON<6>), must be set to enable this mode. The timer must be enabled (TON = 1) and the timer clock source set to internal (TCS = 0).

When the CPU goes into the Idle mode, the timer will stop incrementing unless TSIDL = 0. If TSIDL = 1, the timer will resume the incrementing sequence upon termination of the CPU Idle mode.

9.2 Timer Prescaler

The input clock (FOSC/4 or external clock) to the 16-bit Timer has a prescale option of 1:1, 1:8, 1:64 and 1:256 selected by control bits, TCKPS<1:0> (T1CON<5:4>). The prescaler counter is cleared when any of the following occurs:

- A write to the TMR1 register
- Clearing of the TON bit (T1CON<15>)
- A device Reset, such as a POR and BOR

However, if the timer is disabled (TON = 0), then the timer prescaler cannot be reset since the prescaler clock is halted.

The TMR1 register is not cleared when the T1CON register is written. It is cleared by writing to the TMR1 register.

9.3 Timer Operation During Sleep Mode

During CPU Sleep mode, the timer will operate if:

- The timer module is enabled (TON = 1) and
- The timer clock source is selected as external (TCS = 1) and
- The TSYNC bit (T1CON<2>) is asserted to a logic '0', which defines the external clock source as asynchronous

When all three conditions are true, the timer will continue to count up to the Period register and be reset to 0x0000.

When a match between the timer and the Period register occurs, an interrupt can be generated if the respective timer interrupt enable bit is asserted.

11.0 TIMER4/5 MODULE

Note:	This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *“16-bit MCU and DSC Reference Manual”* (DS70157).

This section describes the second 32-bit general purpose timer module (Timer4/5) and associated operational modes. Figure 11-1 depicts the simplified block diagram of the 32-bit Timer4/5 module. Figure 11-2 and Figure 11-3 illustrate Timer4/5 configured as two independent 16-bit timers, Timer4 and Timer5, respectively.

Note:	Timer4 is a ‘Type B’ timer and Timer5 is a ‘Type C’ timer. Please refer to the appropriate timer type in **Section 24.0 “Electrical Characteristics”** of this document.

The Timer4/5 module is similar in operation to the Timer 2/3 module. However, there are some differences, which are as follows:

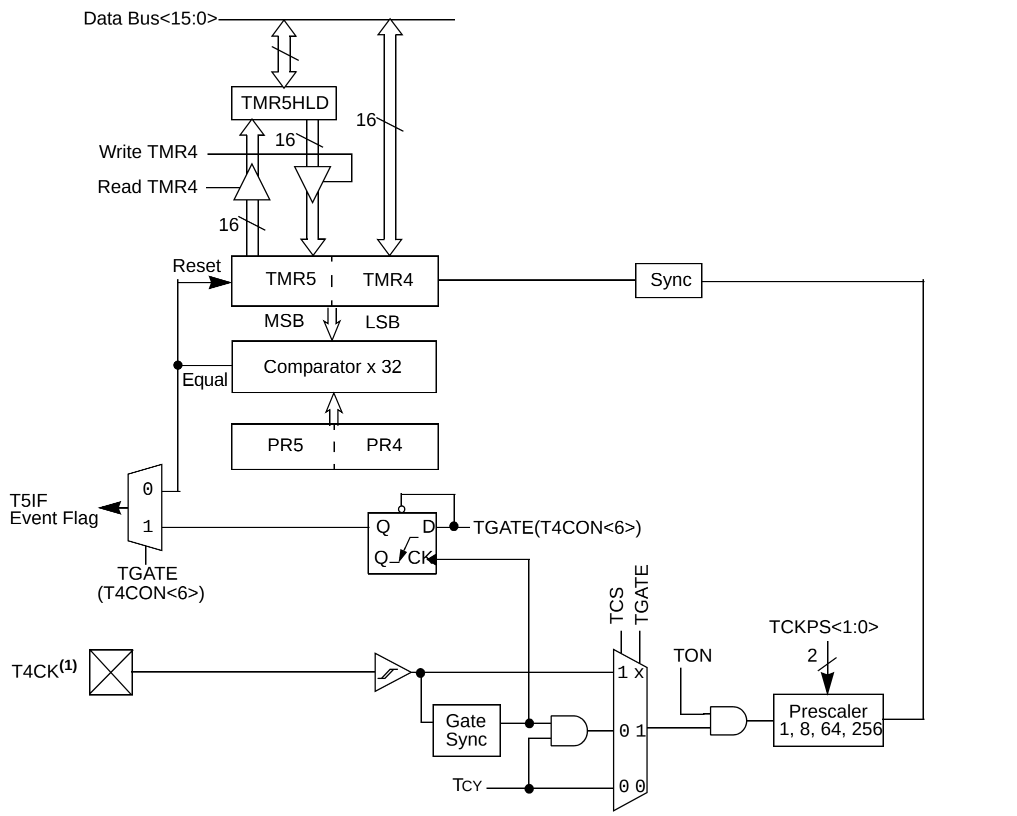
- The Timer4/5 module does not support the ADC event trigger feature
- Timer4/5 can not be utilized by other peripheral modules such as input capture and output compare

The operating modes of the Timer4/5 module are determined by setting the appropriate bit(s) in the 16-bit T4CON and T5CON SFRs.

For 32-bit timer/counter operation, Timer4 is the least significant word and Timer5 is the most significant word of the 32-bit timer.

Note:	For 32-bit timer operation, T5CON control bits are ignored. Only T4CON control bits are used for setup and control. Timer4 clock and gate inputs are utilized for the 32-bit timer module, but an interrupt is generated with the Timer5 Interrupt Flag (T5IF) and the interrupt is enabled with the Timer5 Interrupt Enable bit (T5IE).

FIGURE 11-1: 32-BIT TIMER4/5 BLOCK DIAGRAM



Note 1: T4CK is not implemented and this line is tied to VSS.

2: Timer configuration bit, T32 (T2CON<3>), must be set to ‘1’ for a 32-bit timer/counter operation. All control bits are respective to the T4CON register.

12.3 Input Capture Interrupts

The input capture channels have the ability to generate an interrupt, based upon the selected number of capture events. The selection number is set by control bits, ICI<1:0> (ICxCON<6:5>).

Each channel provides an interrupt flag (ICxIF) bit. The respective capture channel interrupt flag is located in the corresponding IFSx register.

Enabling an interrupt is accomplished via the respective Capture Channel Interrupt Enable (ICxIE) bit. The capture interrupt enable bit is located in the corresponding IECx register.

15.1.4 DOUBLE UPDATE MODE

In the Double Update mode (PTMOD<1:0> = 11), an interrupt event is generated each time the PTMR register is equal to zero, as well as each time a period match occurs. The postscaler selection bits have no effect in this mode of the timer.

The Double Update mode provides two additional functions to the user. First, the control loop bandwidth is doubled because the PWM duty cycles can be updated, twice per period. Second, asymmetrical center-aligned PWM waveforms can be generated which are useful for minimizing output waveform distortion in certain motor control applications.

Note:	Programming a value of 0x0001 in the Period register could generate a continuous interrupt pulse, and hence, must be avoided.

15.1.5 PWM TIME BASE PRESCALER

The input clock to PTMR (FOSC/4) has prescaler options of 1:1, 1:4, 1:16 or 1:64, selected by control bits, PTCKPS<1:0>, in the PTCON SFR. The prescaler counter is cleared when any of the following occurs:

- A write to the PTMR register
- A write to the PTCON register
- Any device Reset

The PTMR register is not cleared when PTCON is written.

15.1.6 PWM TIME BASE POSTSCALER

The match output of PTMR can optionally be post-scaled through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling).

The postscaler counter is cleared when any of the following occurs:

- A write to the PTMR register
- A write to the PTCON register
- Any device Reset

The PTMR register is not cleared when the PTCON register is written.

15.2 PWM Period

PTPER is a 15-bit register and is used to set the counting period for the PWM time base. PTPER is a double-buffered register. The PTPER buffer contents are loaded into the PTPER register at the following instants:

- Free-Running and Single-Shot modes: When the PTMR register is reset to zero after a match with the PTPER register.
- Continuous Up/Down Count modes: When the PTMR register is zero.

The value held in the PTPER buffer is automatically loaded into the PTPER register when the PWM time base is disabled (PTEN = 0).

The PWM period can be determined using Equation 15-1:

EQUATION 15-1: PWM PERIOD

$$T_{PWM} = T_{CY} \bullet (PTPER + 1) \bullet PTMR\ Prescale\ Value$$

If the PWM time base is configured for one of the Continuous Up/Down Count modes, the PWM period is provided by Equation 15-2.

EQUATION 15-2: PWM PERIOD (CENTER-ALIGNED MODE)

$$T_{PWM} = T_{CY} \bullet 2 \bullet PTPER + 1) \bullet PTMR\ Prescale\ Value$$

The maximum resolution (in bits) for a given device oscillator and PWM frequency can be determined using Equation 15-3:

EQUATION 15-3: PWM RESOLUTION

$$Resolution = \frac{\log (2 \bullet T_{PWM}/T_{CY})}{\log (2)}$$

19.3 Modes of Operation

The CAN module can operate in one of several operation modes selected by the user. These modes include:

- Initialization Mode
- Disable Mode
- Normal Operation Mode
- Listen Only Mode
- Loopback Mode
- Error Recognition Mode

Modes are requested by setting the REQOP<2:0> bits (C1CTRL<10:8>). Entry into a mode is acknowledged by monitoring the OPMODE<2:0> bits (C1CTRL<7:5>). The module will not change the mode and the OPMODE bits until a change in mode is acceptable, generally during bus Idle time which is defined as at least 11 consecutive recessive bits.

19.3.1 INITIALIZATION MODE

In the Initialization mode, the module will not transmit or receive. The error counters are cleared and the interrupt flags remain unchanged. The programmer will have access to Configuration registers that are access restricted in other modes. The module will protect the user from accidentally violating the CAN protocol through programming errors. All registers which control the configuration of the module can not be modified while the module is online. The CAN module will not be allowed to enter the Configuration mode while a transmission is taking place. The Configuration mode serves as a lock to protect the following registers.

- All Module Control Registers
- Baud Rate and Interrupt Configuration Registers
- Bus Timing Registers
- Identifier Acceptance Filter Registers
- Identifier Acceptance Mask Registers

19.3.2 DISABLE MODE

In Disable mode, the module will not transmit or receive. The module has the ability to set the WAKIF bit due to bus activity, however, any pending interrupts will remain and the error counters will retain their value.

If the REQOP<2:0> bits (C1CTRL<10:8>) = `001`, the module will enter the Disable mode. If the module is active, the module will wait for 11 recessive bits on the CAN bus, detect that condition as an Idle bus, then accept the disable command. When the OPMODE<2:0> bits (C1CTRL<7:5>) = `001`, that indicates whether the module successfully went into Disable mode. The I/O pins will revert to normal I/O function when the module is in the Disable mode.

The module can be programmed to apply a low-pass filter function to the C1RX input line while the module or the CPU is in Sleep mode. The WAKFIL bit (C1CFG2<14>) enables or disables the filter.

> **Note:** Typically, if the CAN module is allowed to transmit in a particular mode of operation and a transmission is requested immediately after the CAN module has been placed in that mode of operation, the module waits for 11 consecutive recessive bits on the bus before starting transmission. If the user switches to Disable mode within this 11-bit period, then this transmission is aborted and the corresponding TXABT bit is set and TXREQ bit is cleared.

19.3.3 NORMAL OPERATION MODE

Normal Operation mode is selected when REQOP<2:0> = `000`. In this mode, the module is activated and the I/O pins will assume the CAN bus functions. The module will transmit and receive CAN bus messages via the C1TX and C1RX pins.

19.3.4 LISTEN ONLY MODE

If the Listen Only mode is activated, the module on the CAN bus is passive. The transmitter buffers revert to the port I/O function. The receive pins remain inputs. For the receiver, no error flags or Acknowledge signals are sent. The error counters are deactivated in this state. The Listen Only mode can be used for detecting the baud rate on the CAN bus. To use this, it is necessary that there are at least two further nodes that communicate with each other.

19.3.5 ERROR RECOGNITION MODE

The module can be set to ignore all errors and receive any message. In this mode, the data which is in the message assembly buffer until the time an error occurred, is copied in the receive buffer and can be read via the CPU interface.

19.3.6 LOOPBACK MODE

If the Loopback mode is activated, the module will connect the internal transmit signal to the internal receive signal at the module boundary. The transmit and receive pins revert to their port I/O function.

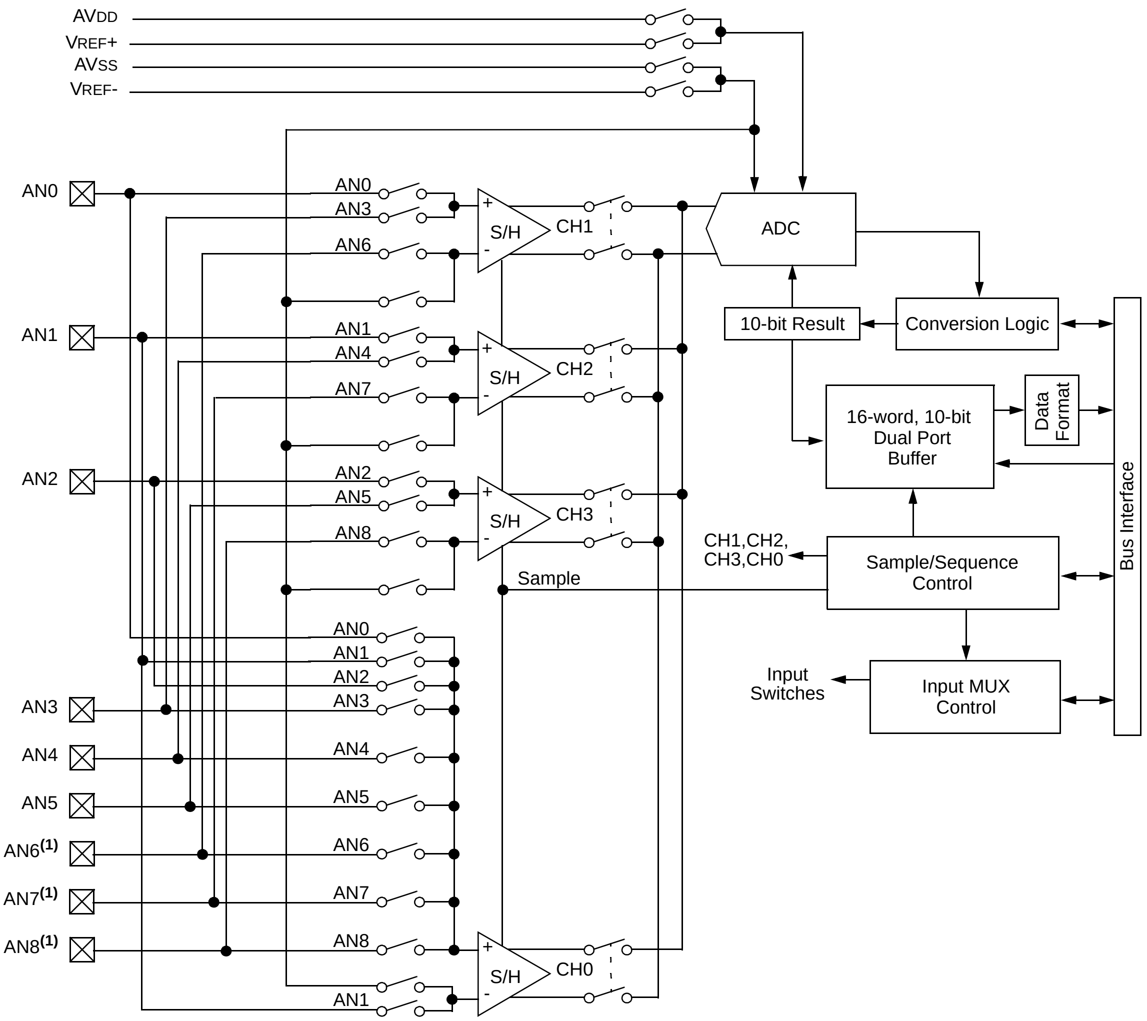
TABLE 19-1: CAN1 REGISTER MAP(1)

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
C1RXF0SID	0300	—	—	—	Receive Acceptance Filter 0 Standard Identifier<10:0>											—	EXIDE	000u uuuu uuuu uu0u
C1RXF0EIDH	0302	—	—	—	—	Receive Acceptance Filter 0 Extended Identifier<17:6>												0000 uuuu uuuu uuuu
C1RXF0EIDL	0304	Receive Acceptance Filter 0 Extended Identifier<5:0>						—	—	—	—	—	—	—	—	—	—	uuuu uu00 0000 0000
C1RXF1SID	0308	—	—	—	Receive Acceptance Filter 1 Standard Identifier<10:0>											—	EXIDE	000u uuuu uuuu uu0u
C1RXF1EIDH	030A	—	—	—	—	Receive Acceptance Filter 1 Extended Identifier<17:6>												0000 uuuu uuuu uuuu
C1RXF1EIDL	030C	Receive Acceptance Filter 1 Extended Identifier<5:0>						—	—	—	—	—	—	—	—	—	—	uuuu uu00 0000 0000
C1RXF2SID	0310	—	—	—	Receive Acceptance Filter 2 Standard Identifier<10:0>											—	EXIDE	000u uuuu uuuu uu0u
C1RXF2EIDH	0312	—	—	—	—	Receive Acceptance Filter 2 Extended Identifier<17:6>												0000 uuuu uuuu uuuu
C1RXF2EIDL	0314	Receive Acceptance Filter 2 Extended Identifier<5:0>						—	—	—	—	—	—	—	—	—	—	uuuu uu00 0000 0000
C1RXF3SID	0318	—	—	—	Receive Acceptance Filter 3 Standard Identifier<10:0>											—	EXIDE	000u uuuu uuuu uu0u
C1RXF3EIDH	031A	—	—	—	—	Receive Acceptance Filter 3 Extended Identifier<17:6>												0000 uuuu uuuu uuuu
C1RXF3EIDL	031C	Receive Acceptance Filter 3 Extended Identifier<5:0>						—	—	—	—	—	—	—	—	—	—	uuuu uu00 0000 0000
C1RXF4SID	0320	—	—	—	Receive Acceptance Filter 4 Standard Identifier<10:0>											—	EXIDE	000u uuuu uuuu uu0u
C1RXF4EIDH	0322	—	—	—	—	Receive Acceptance Filter 4 Extended Identifier<17:6>												0000 uuuu uuuu uuuu
C1RXF4EIDL	0324	Receive Acceptance Filter 4 Extended Identifier<5:0>						—	—	—	—	—	—	—	—	—	—	uuuu uu00 0000 0000
C1RXF5SID	0328	—	—	—	Receive Acceptance Filter 5 Standard Identifier<10:0>											—	EXIDE	000u uuuu uuuu uu0u
C1RXF5EIDH	032A	—	—	—	—	Receive Acceptance Filter 5 Extended Identifier<17:6>												0000 uuuu uuuu uuuu
C1RXF5EIDL	032C	Receive Acceptance Filter 5 Extended Identifier<5:0>						—	—	—	—	—	—	—	—	—	—	uuuu uu00 0000 0000
C1RXM0SID	0330	—	—	—	Receive Acceptance Mask 0 Standard Identifier<10:0>											—	MIDE	000u uuuu uuuu uu0u
C1RXM0EIDH	0332	—	—	—	—	Receive Acceptance Mask 0 Extended Identifier<17:6>												0000 uuuu uuuu uuuu
C1RXM0EIDL	0334	Receive Acceptance Mask 0 Extended Identifier<5:0>						—	—	—	—	—	—	—	—	—	—	uuuu uu00 0000 0000
C1RXM1SID	0338	—	—	—	Receive Acceptance Mask 1 Standard Identifier<10:0>											—	MIDE	000u uuuu uuuu uu0u
C1RXM1EIDH	033A	—	—	—	—	Receive Acceptance Mask 1 Extended Identifier<17:6>												0000 uuuu uuuu uuuu
C1RXM1EIDL	033C	Receive Acceptance Mask 1 Extended Identifier<5:0>						—	—	—	—	—	—	—	—	—	—	uuuu uu00 0000 0000
C1TX2SID	0340	Transmit Buffer 2 Standard Identifier<10:6>					—	—	—	Transmit Buffer 2 Standard Identifier<5:0>						SRR	TXIDE	uuuu u000 uuuu uuuu
C1TX2EID	0342	Transmit Buffer 2 Extended Identifier<17:14>				—	—	—	—	Transmit Buffer 2 Extended Identifier<13:6>								uuuu 0000 uuuu uuuu
C1TX2DLC	0344	Transmit Buffer 2 Extended Identifier<5:0>						TXRTR	TXRB1	TXRB0	DLC<3:0>				—	—	—	uuuu uuuu uuuu u000
C1TX2B1	0346	Transmit Buffer 2 Byte 1								Transmit Buffer 2 Byte 0								uuuu uuuu uuuu uuuu
C1TX2B2	0348	Transmit Buffer 2 Byte 3								Transmit Buffer 2 Byte 2								uuuu uuuu uuuu uuuu
C1TX2B3	034A	Transmit Buffer 2 Byte 5								Transmit Buffer 2 Byte 4								uuuu uuuu uuuu uuuu
C1TX2B4	034C	Transmit Buffer 2 Byte 7								Transmit Buffer 2 Byte 6								uuuu uuuu uuuu uuuu
C1TX2CON	034E	—	—	—	—	—	—	—	—	—	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI<1:0>		0000 0000 0000 0000
C1TX1SID	0350	Transmit Buffer 1 Standard Identifier<10:6>					—	—	—	Transmit Buffer 1 Standard Identifier<5:0>						SRR	TXIDE	uuuu u000 uuuu uuuu
C1TX1EID	0352	Transmit Buffer 1 Extended Identifier<17:14>				—	—	—	—	Transmit Buffer 1 Extended Identifier<13:6>								uuuu 0000 uuuu uuuu
C1TX1DLC	0354	Transmit Buffer 1 Extended Identifier<5:0>						TXRTR	TXRB1	TXRB0	DLC<3:0>				—	—	—	uuuu uuuu uuuu u000
C1TX1B1	0356	Transmit Buffer 1 Byte 1								Transmit Buffer 1 Byte 0								uuuu uuuu uuuu uuuu

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.

FIGURE 20-1: 10-BIT, HIGH-SPEED ADC FUNCTIONAL BLOCK DIAGRAM



Note 1: Not available on dsPIC30F4012 devices.

TABLE 21-1: OSCILLATOR OPERATING MODES

Oscillator Mode	Description
XTL	200 kHz-4 MHz crystal on OSC1:OSC2
XT	4 MHz-10 MHz crystal on OSC1:OSC2
XT w/PLL 4x	4 MHz-10 MHz crystal on OSC1:OSC2, 4x PLL enabled
XT w/PLL 8x	4 MHz-10 MHz crystal on OSC1:OSC2, 8x PLL enabled
XT w/PLL 16x	4 MHz-10 MHz crystal on OSC1:OSC2, 16x PLL enabled(1)
LP	32 kHz crystal on SOSCO:SOSCI(2)
HS	10 MHz-25 MHz crystal
EC	External clock input (0-40 MHz)
ECIO	External clock input (0-40 MHz), OSC2 pin is I/O
EC w/PLL 4x	External clock input (0-40 MHz), OSC2 pin is I/O, 4x PLL enabled(1)
EC w/PLL 8x	External clock input (0-40 MHz), OSC2 pin is I/O, 8x PLL enabled(1)
EC w/PLL 16x	External clock input (0-40 MHz), OSC2 pin is I/O, 16x PLL enabled(1)
ERC	External RC oscillator, OSC2 pin is FOSC/4 output(3)
ERCIO	External RC oscillator, OSC2 pin is I/O(3)
FRC	8 MHz internal RC oscillator
FRC w/PLL 4x	7.37 MHz Internal RC oscillator, 4x PLL enabled
FRC w/PLL 8x	7.37 MHz Internal RC oscillator, 8x PLL enabled
FRC w/PLL 16x	7.37 MHz Internal RC oscillator, 16x PLL enabled
LPRC	512 kHz internal RC oscillator

Note 1: The dsPIC30F maximum operating frequency of 120 MHz must be met.
2: LP oscillator can be conveniently shared as a system clock, as well as a Real-Time Clock for Timer1.
3: Requires external R and C. Frequency operation up to 4 MHz.

TABLE 22-2: INSTRUCTION SET OVERVIEW

Base Instr #	Assembly Mnemonic	Assembly Syntax		Description	# of words	# of cycles	Status Flags Affected
1	ADD	ADD	Acc	Add Accumulators	1	1	OA, OB, SA, SB
		ADD	f	f = f + WREG	1	1	C, DC, N, OV, Z
		ADD	f,WREG	WREG = f + WREG	1	1	C, DC, N, OV, Z
		ADD	#lit10,Wn	Wd = lit10 + Wd	1	1	C, DC, N, OV, Z
		ADD	Wb,Ws,Wd	Wd = Wb + Ws	1	1	C, DC, N, OV, Z
		ADD	Wb,#lit5,Wd	Wd = Wb + lit5	1	1	C, DC, N, OV, Z
		ADD	Wso,#Slit4,Acc	16-bit Signed Add to Accumulator	1	1	OA, OB, SA, SB
2	ADDC	ADDC	f	f = f + WREG + (C)	1	1	C, DC, N, OV, Z
		ADDC	f,WREG	WREG = f + WREG + (C)	1	1	C, DC, N, OV, Z
		ADDC	#lit10,Wn	Wd = lit10 + Wd + (C)	1	1	C, DC, N, OV, Z
		ADDC	Wb,Ws,Wd	Wd = Wb + Ws + (C)	1	1	C, DC, N, OV, Z
		ADDC	Wb,#lit5,Wd	Wd = Wb + lit5 + (C)	1	1	C, DC, N, OV, Z
3	AND	AND	f	f = f .AND. WREG	1	1	N, Z
		AND	f,WREG	WREG = f .AND. WREG	1	1	N, Z
		AND	#lit10,Wn	Wd = lit10 .AND. Wd	1	1	N, Z
		AND	Wb,Ws,Wd	Wd = Wb .AND. Ws	1	1	N, Z
		AND	Wb,#lit5,Wd	Wd = Wb .AND. lit5	1	1	N, Z
4	ASR	ASR	f	f = Arithmetic Right Shift f	1	1	C, N, OV, Z
		ASR	f,WREG	WREG = Arithmetic Right Shift f	1	1	C, N, OV, Z
		ASR	Ws,Wd	Wd = Arithmetic Right Shift Ws	1	1	C, N, OV, Z
		ASR	Wb,Wns,Wnd	Wnd = Arithmetic Right Shift Wb by Wns	1	1	N, Z
		ASR	Wb,#lit5,Wnd	Wnd = Arithmetic Right Shift Wb by lit5	1	1	N, Z
5	BCLR	BCLR	f,#bit4	Bit Clear f	1	1	None
		BCLR	Ws,#bit4	Bit Clear Ws	1	1	None
6	BRA	BRA	C,Expr	Branch if Carry	1	1 (2)	None
		BRA	GE,Expr	Branch if greater than or equal	1	1 (2)	None
		BRA	GEU,Expr	Branch if unsigned greater than or equal	1	1 (2)	None
		BRA	GT,Expr	Branch if greater than	1	1 (2)	None
		BRA	GTU,Expr	Branch if unsigned greater than	1	1 (2)	None
		BRA	LE,Expr	Branch if less than or equal	1	1 (2)	None
		BRA	LEU,Expr	Branch if unsigned less than or equal	1	1 (2)	None
		BRA	LT,Expr	Branch if less than	1	1 (2)	None
		BRA	LTU,Expr	Branch if unsigned less than	1	1 (2)	None
		BRA	N,Expr	Branch if Negative	1	1 (2)	None
		BRA	NC,Expr	Branch if Not Carry	1	1 (2)	None
		BRA	NN,Expr	Branch if Not Negative	1	1 (2)	None
		BRA	NOV,Expr	Branch if Not Overflow	1	1 (2)	None
		BRA	NZ,Expr	Branch if Not Zero	1	1 (2)	None
		BRA	OA,Expr	Branch if Accumulator A overflow	1	1 (2)	None
		BRA	OB,Expr	Branch if Accumulator B overflow	1	1 (2)	None
		BRA	OV,Expr	Branch if Overflow	1	1 (2)	None
		BRA	SA,Expr	Branch if Accumulator A saturated	1	1 (2)	None
		BRA	SB,Expr	Branch if Accumulator B saturated	1	1 (2)	None
		BRA	Expr	Branch Unconditionally	1	2	None
		BRA	Z,Expr	Branch if Zero	1	1 (2)	None
		BRA	Wn	Computed Branch	1	2	None
7	BSET	BSET	f,#bit4	Bit Set f	1	1	None
		BSET	Ws,#bit4	Bit Set Ws	1	1	None
8	BSW	BSW.C	Ws,Wb	Write C bit to Ws<Wb>	1	1	None
		BSW.Z	Ws,Wb	Write Z bit to Ws<Wb>	1	1	None

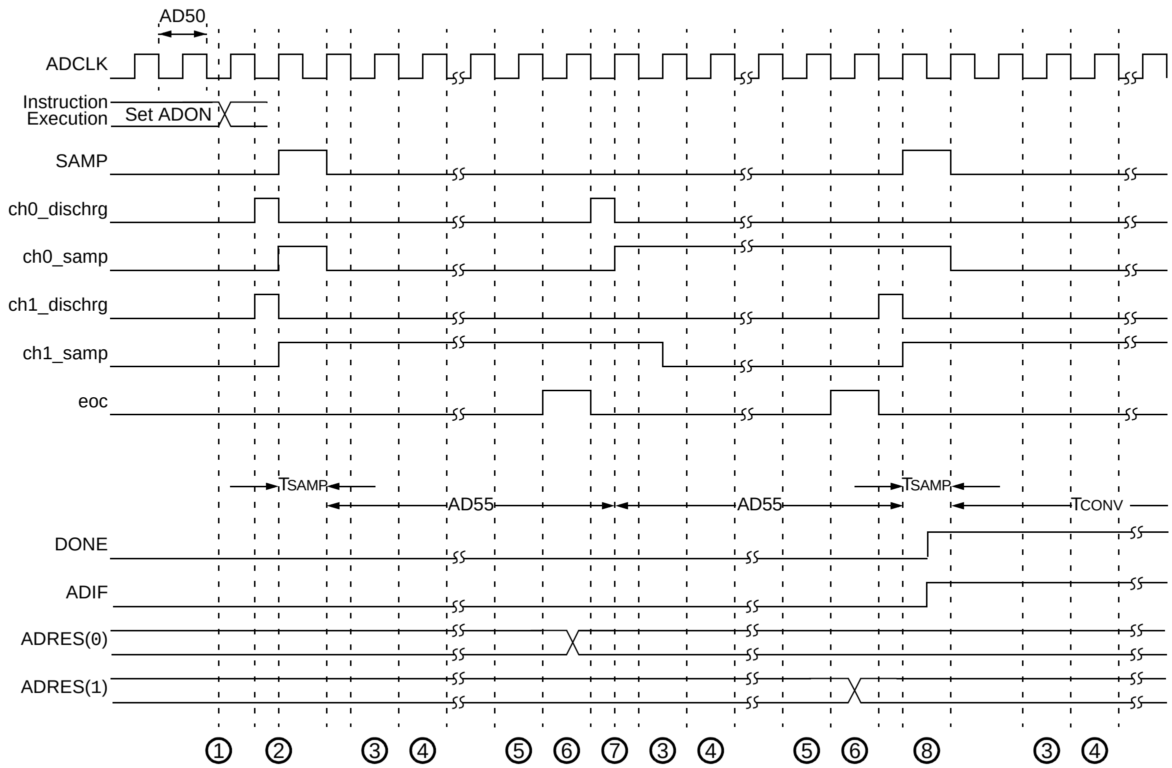
TABLE 24-36: I²C™ BUS DATA TIMING REQUIREMENTS (MASTER MODE)

AC 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(1)	Max	Units	Conditions
IM10	TLO:SCL	Clock Low Time	100 kHz mode	TCY/2 (BRG + 1)	—	µs	
			400 kHz mode	TCY/2 (BRG + 1)	—	µs	
			1 MHz mode(2)	TCY/2 (BRG + 1)	—	µs	
IM11	THI:SCL	Clock High Time	100 kHz mode	TCY/2 (BRG + 1)	—	µs	
			400 kHz mode	TCY/2 (BRG + 1)	—	µs	
			1 MHz mode(2)	TCY/2 (BRG + 1)	—	µs	
IM20	TF:SCL	SDA and SCL Fall Time	100 kHz mode	—	300	ns	CB is specified to be from 10 to 400 pF
			400 kHz mode	20 + 0.1 CB	300	ns	
			1 MHz mode(2)	—	100	ns	
IM21	TR:SCL	SDA and SCL Rise Time	100 kHz mode	—	1000	ns	CB is specified to be from 10 to 400 pF
			400 kHz mode	20 + 0.1 CB	300	ns	
			1 MHz mode(2)	—	300	ns	
IM25	TSU:DAT	Data Input Setup Time	100 kHz mode	250	—	ns	
			400 kHz mode	100	—	ns	
			1 MHz mode(2)	—	—	ns	
IM26	THD:DAT	Data Input Hold Time	100 kHz mode	0	—	ns	
			400 kHz mode	0	0.9	µs	
			1 MHz mode(2)	—	—	ns	
IM30	TSU:STA	Start Condition Setup Time	100 kHz mode	TCY/2 (BRG + 1)	—	µs	Only relevant for Repeated Start condition
			400 kHz mode	TCY/2 (BRG + 1)	—	µs	
			1 MHz mode(2)	TCY/2 (BRG + 1)	—	µs	
IM31	THD:STA	Start Condition Hold Time	100 kHz mode	TCY/2 (BRG + 1)	—	µs	After this period, the first clock pulse is generated
			400 kHz mode	TCY/2 (BRG + 1)	—	µs	
			1 MHz mode(2)	TCY/2 (BRG + 1)	—	µs	
IM33	TSU:STO	Stop Condition Setup Time	100 kHz mode	TCY/2 (BRG + 1)	—	µs	
			400 kHz mode	TCY/2 (BRG + 1)	—	µs	
			1 MHz mode(2)	TCY/2 (BRG + 1)	—	µs	
IM34	THD:STO	Stop Condition Hold Time	100 kHz mode	TCY/2 (BRG + 1)	—	ns	
			400 kHz mode	TCY/2 (BRG + 1)	—	ns	
			1 MHz mode(2)	TCY/2 (BRG + 1)	—	ns	
IM40	TAA:SCL	Output Valid From Clock	100 kHz mode	—	3500	ns	
			400 kHz mode	—	1000	ns	
			1 MHz mode(2)	—	—	ns	
IM45	TBF:SDA	Bus Free Time	100 kHz mode	4.7	—	µs	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	µs	
			1 MHz mode(2)	—	—	µs	
IM50	CB	Bus Capacitive Loading		—	400	pF	

Note 1: BRG is the value of the I²C Baud Rate Generator. Refer to **Section 21. "Inter-Integrated Circuit (I²C™)"** in the "*dsPIC30F Family Reference Manual*" (DS70046).

2: Maximum pin capacitance = 10 pF for all I²C pins (for 1 MHz mode only).

FIGURE 24-26: 10-BIT HIGH-SPEED A/D CONVERSION TIMING CHARACTERISTICS (CHPS<1:0> = 01, SIMSAM = 0, ASAM = 1, SSRC<2:0> = 111, SAMC<4:0> = 00001)



1 - Software sets ADCONx. ADON to start AD operation.

2 - Sampling starts after discharge period. TSAMP is described in **Section 17. 10-Bit A/D Converter"** in the "*dsPIC30F Family Reference Manual*" (DS70046).

3 - Convert bit 9.

4 - Convert bit 8.

5 - Convert bit 0.

6 - One TAD for end of conversion.

7 - Begin conversion of next channel

8 - Sample for time specified by SAMC<4:0>.