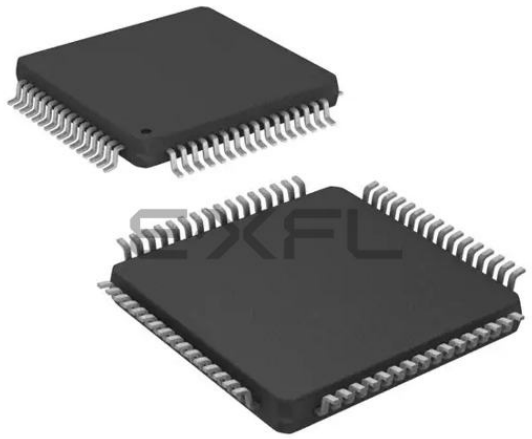


What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	64MHz
Connectivity	ECANbus, I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	54
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.6K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 11x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf66k80-i-pt

2.4 Voltage Regulator Pins (ENVREG and VCAP/VDDCORE)

The on-chip voltage regulator enable pin, ENVREG, must always be connected directly to either a supply voltage or to ground. Tying ENVREG to VDD enables the regulator, while tying it to ground disables the regulator. Refer to **Section 28.3 “On-Chip Voltage Regulator”** for details on connecting and using the on-chip regulator.

When the regulator is enabled, a low-ESR (< 5Ω) capacitor is required on the VCAP/VDDCORE pin to stabilize the voltage regulator output voltage. The VCAP/VDDCORE pin must not be connected to VDD and must use a capacitor of 10 µF connected to ground. The type can be ceramic or tantalum. Suitable examples of capacitors are shown in Table 2-1. Capacitors with equivalent specifications can be used.

Designers may use Figure 2-3 to evaluate ESR equivalence of candidate devices.

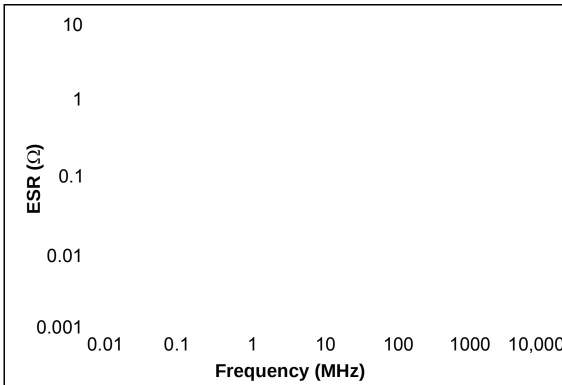
It is recommended that the trace length not exceed 0.25 inch (6 mm). Refer to **Section 31.0 “Electrical Characteristics”** for additional information.

When the regulator is disabled, the VCAP/VDDCORE pin must be tied to a voltage supply at the VDDCORE level. Refer to **Section 31.0 “Electrical Characteristics”** for information on VDD and VDDCORE.

Some PIC18FXXKXX families, or some devices within a family, do not provide the option of enabling or disabling the on-chip voltage regulator:

- Some devices (with the name, PIC18**LF**XXKXX) permanently disable the voltage regulator.
  These devices do not have the ENVREG pin and require a 0.1 µF capacitor on the VCAP/VDDCORE pin. The VDD level of these devices must comply with the “voltage regulator disabled” specification for Parameter D001, in **Section 31.0 “Electrical Characteristics”**.
- Some devices permanently enable the voltage regulator.
  These devices also do not have the ENVREG pin. The 10 µF capacitor is still required on the VCAP/VDDCORE pin.

FIGURE 2-3: FREQUENCY vs. ESR PERFORMANCE FOR SUGGESTED VCAP



Note: Typical data measurement at 25°C, 0V DC bias.

.

TABLE 2-1: SUITABLE CAPACITOR EQUIVALENTS

Make	Part #	Nominal Capacitance	Base Tolerance	Rated Voltage	Temp. Range
TDK	C3216X7R1C106K	10 µF	±10%	16V	-55 to 125ºC
TDK	C3216X5R1C106K	10 µF	±10%	16V	-55 to 85ºC
Panasonic	ECJ-3YX1C106K	10 µF	±10%	16V	-55 to 125ºC
Panasonic	ECJ-4YB1C106K	10 µF	±10%	16V	-55 to 85ºC
Murata	GRM32DR71C106KA01L	10 µF	±10%	16V	-55 to 125ºC
Murata	GRM31CR61C106KC31L	10 µF	±10%	16V	-55 to 85ºC

TABLE 5-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices			Power-on Reset, Brown-out Reset	$\overline{MCLR}$ Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
PIE5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
EEADRH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---- --00	---- --00	---- --00
EEADR	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
EEDATA	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
ECANCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0001 0000	0001 0000	uuuu uuuu
COMSTAT	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
CIOCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 ---0	0000 ---0	uuuu ---u
CANCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu
CANSTAT	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu
RXB0D7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0DLC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx x-xx	uuuu u-uu	uuuu u-uu
RXB0SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
CM1CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0001 1111	0001 1111	uuuu uuuu
CM2CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0001 1111	0001 1111	uuuu uuuu
ANCON0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1111 1111	1111 1111	uuuu uuuu
ANCON1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-111 1111	-111 1111	-uuu uuuu
WPUB	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	1111 1111	uuuu uuuu
IOCB	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 ----	0000 ----	uuuu ----
PMD0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu

Legend: u = unchanged; x = unknown; - = unimplemented bit, read as ‘0’; q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 5-3 for Reset value for specific conditions.

5: Bits 6 and 7 of PORTA, LATA and TRISA are enabled, depending on the oscillator mode selected. When not enabled as PORTA pins, they are disabled and read as ‘0’.

TABLE 5-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices			Power-on Reset, Brown-out Reset	MCLR Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
B0EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx x-xx	uuuu u-uu	uuuu u-uu
B0SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
TXBIE	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---0 00--	---u uu--	---u uu--
BIE0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
BSEL0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 00--	0000 00--	uuuu uu--
MSEL3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
MSEL2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
MSEL1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0101	0000 0101	uuuu uuuu
MSEL0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0101 0000	0101 0000	uuuu uuuu
RXFBCON7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
RXFBCON6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
RXFBCON5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
RXFBCON4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
RXFBCON3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
RXFBCON2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0001 0001	0001 0001	uuuu uuuu
RXFBCON1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0001 0001	0001 0001	uuuu uuuu
RXFBCON0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
SDFLC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---0 0000	---0 0000	---u uuuu
RXF15EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF15EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF15SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF15SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF14EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF14EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF14SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF14SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF13EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF13EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF13SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF13SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF12EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF12EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF12SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxx- x-xx	uuu- u-uu	uuu- u-uu

Legend: u = unchanged; x = unknown; - = unimplemented bit, read as ‘0’; q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 5-3 for Reset value for specific conditions.

5: Bits 6 and 7 of PORTA, LATA and TRISA are enabled, depending on the oscillator mode selected. When not enabled as PORTA pins, they are disabled and read as ‘0’.

6.4.3.2 FSR Registers and POSTINC, POSTDEC, PREINC and PLUSW

In addition to the INDF operand, each FSR register pair also has four additional indirect operands. Like INDF, these are "virtual" registers that cannot be indirectly read or written to. Accessing these registers actually accesses the associated FSR register pair, but also performs a specific action on its stored value.

These operands are:

- POSTDEC – Accesses the FSR value, then automatically decrements it by '1' afterwards
- POSTINC – Accesses the FSR value, then automatically increments it by '1' afterwards
- PREINC – Increments the FSR value by '1', then uses it in the operation
- PLUSW – Adds the signed value of the W register (range of -127 to 128) to that of the FSR and uses the new value in the operation

In this context, accessing an INDF register uses the value in the FSR registers without changing them. Similarly, accessing a PLUSW register gives the FSR value, offset by the value in the W register, with neither value actually changed in the operation. Accessing the other virtual registers changes the value of the FSR registers.

Operations on the FSRs with POSTDEC, POSTINC and PREINC affect the entire register pair. Rollovers of the FSRnL register, from FFh to 00h, carry over to the FSRnH register. On the other hand, results of these operations do not change the value of any flags in the STATUS register (for example, Z, N and OV bits).

The PLUSW register can be used to implement a form of Indexed Addressing in the data memory space. By manipulating the value in the W register, users can reach addresses that are fixed offsets from pointer addresses. In some applications, this can be used to implement some powerful program control structure, such as software stacks, inside of data memory.

6.4.3.3 Operations by FSRs on FSRs

Indirect Addressing operations that target other FSRs or virtual registers represent special cases. For example, using an FSR to point to one of the virtual registers will not result in successful operations.

As a specific case, assume that the FSR0H:FSR0L registers contain FE7h, the address of INDF1. Attempts to read the value of the INDF1, using INDF0 as an operand, will return 00h. Attempts to write to INDF1, using INDF0 as the operand, will result in a `NOP`.

On the other hand, using the virtual registers to write to an FSR pair may not occur as planned. In these cases, the value will be written to the FSR pair, but without any incrementing or decrementing. Thus, writing to INDF2 or POSTDEC2 will write the same value to the FSR2H:FSR2L.

Since the FSRs are physical registers mapped in the SFR space, they can be manipulated through all direct operations. Users should proceed cautiously when working on these registers, however, particularly if their code uses Indirect Addressing.

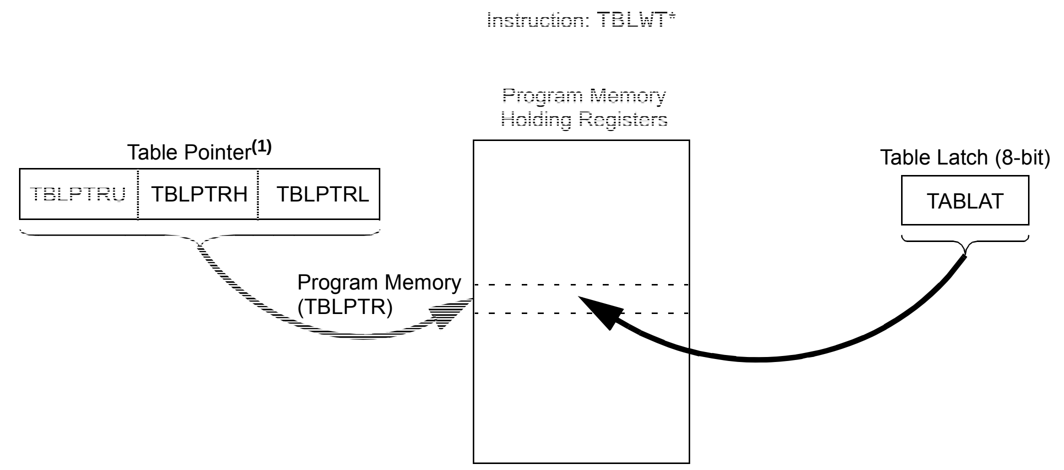
Similarly, operations by Indirect Addressing are generally permitted on all other SFRs. Users should exercise the appropriate caution, so that they do not inadvertently change settings that might affect the operation of the device.

6.5 Program Memory and the Extended Instruction Set

The operation of program memory is unaffected by the use of the extended instruction set.

Enabling the extended instruction set adds five additional two-word commands to the existing PIC18 instruction set: `ADDFSR`, `CALLW`, `MOVSF`, `MOVSS` and `SUBFSR`. These instructions are executed as described in **Section 6.2.4 "Two-Word Instructions"**.

FIGURE 7-2: TABLE WRITE OPERATION



Note 1: The Table Pointer actually points to one of 64 holding registers, the address of which is determined by TBLPTRL<5:0>. The process for physically writing data to the program memory array is discussed in **Section 7.5 "Writing to Flash Program Memory"**.

7.2 Control Registers

Several control registers are used in conjunction with the `TBLRD` and `TBLWT` instructions. These include the:

- EECON1 register
- EECON2 register
- TABLAT register
- TBLPTR registers

7.2.1 EECON1 AND EECON2 REGISTERS

The EECON1 register (Register 7-1) is the control register for memory accesses. The EECON2 register, not a physical register, is used exclusively in the memory write and erase sequences. Reading EECON2 will read all '`0`'s.

The EEPGD control bit determines if the access is a program or data EEPROM memory access. When clear, any subsequent operations operate on the data EEPROM memory. When set, any subsequent operations operate on the program memory.

The CFGS control bit determines if the access is to the Configuration registers or to program memory/data EEPROM memory. When set, subsequent operations operate on Configuration registers regardless of EEPGD (see **Section 28.0 "Special Features of the CPU"**). When clear, memory selection access is determined by EEPGD.

The FREE bit, when set, allows a program memory erase operation. When FREE is set, the erase operation is initiated on the next WR command. When FREE is clear, only writes are enabled.

The WREN bit, when set, allows a write operation. On power-up, the WREN bit is clear. The WRERR bit is set in hardware when the WR bit is set and cleared when the internal programming timer expires and the write operation is complete.

Note:	During normal operation, the WRERR is read as '`1`'. This can indicate that a write operation was prematurely terminated by a Reset, or a write operation was attempted improperly.

The WR control bit initiates write operations. The bit cannot be cleared, only set, in software. It is cleared in hardware at the completion of the write operation.

Note:	The EEIF interrupt flag bit (PIR4<6>) is set when the write is complete. It must be cleared in software.

7.4 Erasing Flash Program Memory

The erase blocks are 32 words or 64 bytes.

Word erase in the Flash array is not supported.

When initiating an erase sequence from the microcontroller itself, a block of 64 bytes of program memory is erased. The Most Significant 16 bits of the TBLPTR<21:6> point to the block being erased. The TBLPTR<5:0> bits are ignored.

The EECON1 register commands the erase operation. The EEPGD bit must be set to point to the Flash program memory. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation.

For protection, the write initiate sequence for EECON2 must be used.

A long write is necessary for erasing the internal Flash. Instruction execution is halted while in a long write cycle. The long write will be terminated by the internal programming timer.

7.4.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

The sequence of events for erasing a block of internal program memory location is:

1. Load the Table Pointer register with the address of row to be erased.
2. Set the EECON1 register for the erase operation:
   - Set the EEPGD bit to point to program memory
   - Clear the CFGS bit to access program memory
   - Set the WREN bit to enable writes
   - Set the FREE bit to enable the erase
3. Disable the interrupts.
4. Write 55h to EECON2.
5. Write 0AAh to EECON2.
6. Set the WR bit.

   This begins the row erase cycle.

   The CPU will stall for the duration of the erase for TIW. (See Parameter D133A.)
7. Re-enable interrupts.

EXAMPLE 7-2: ERASING A FLASH PROGRAM MEMORY ROW

```
                  MOVLW   CODE_ADDR_UPPER          ; load TBLPTR with the base
                  MOVWF   TBLPTRU                  ; address of the memory block
                  MOVLW   CODE_ADDR_HIGH
                  MOVWF   TBLPTRH
                  MOVLW   CODE_ADDR_LOW
                  MOVWF   TBLPTRL
       ERASE_ROW
                  BSF     EECON1, EEPGD            ; point to Flash program memory
                  BCF     EECON1, CFGS             ; access Flash program memory
                  BSF     EECON1, WREN             ; enable write to memory
                  BSF     EECON1, FREE             ; enable Row Erase operation
                  BCF     INTCON, GIE              ; disable interrupts
   Required       MOVLW   55h
   Sequence       MOVWF   EECON2                   ; write 55h
                  MOVLW   0AAh
                  MOVWF   EECON2                   ; write 0AAh
                  BSF     EECON1, WR               ; start erase (CPU stall)
                  BSF     INTCON, GIE              ; re-enable interrupts
```

8.3 Reading the Data EEPROM Memory

To read a data memory location, the user must write the address to the EEADRH:EEADR register pair, clear the EEPGD control bit (EECON1<7>) and then set control bit, RD (EECON1<0>). The data is available after one instruction cycle, in the EEDATA register. It can be read after one `NOP` instruction. EEDATA will hold this value until another read operation or until it is written to by the user (during a write operation).

The basic process is shown in Example 8-1.

8.4 Writing to the Data EEPROM Memory

To write an EEPROM data location, the address must first be written to the EEADRH:EEADR register pair and the data written to the EEDATA register. The sequence in Example 8-2 must be followed to initiate the write cycle.

The write will not begin if this sequence is not exactly followed (write 55h to EECON2, write 0AAh to EECON2, then set WR bit) for each byte. It is strongly recommended that interrupts be disabled during this code segment.

Additionally, the WREN bit in EECON1 must be set to enable writes. This mechanism prevents accidental writes to data EEPROM due to unexpected code execution (i.e., runaway programs). 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, EECON1, EEADRH:EEADR and EEDATA cannot be modified. 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 EEPROM Interrupt Flag bit (EEIF) is set. The user may either enable this interrupt or poll this bit; EEIF must be cleared by software.

8.5 Write Verify

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

Note:	Self-write execution to Flash and EEPROM memory cannot be done while running in LP Oscillator (low-power) mode. Executing a self-write will put the device into High-Power mode.

13.0 TIMER0 MODULE

The Timer0 module incorporates the following features:

- Software selectable operation as a timer or counter in both 8-bit or 16-bit modes
- Readable and writable registers
- Dedicated 8-bit, software programmable prescaler
- Selectable clock source (internal or external)
- Edge select for external clock
- Interrupt-on-overflow

The T0CON register (Register 13-1) controls all aspects of the module's operation, including the prescale selection. It is both readable and writable.

Figure 13-1 provides a simplified block diagram of the Timer0 module in 8-bit mode. Figure 13-2 provides a simplified block diagram of the Timer0 module in 16-bit mode.

REGISTER 13-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

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

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-Bit/16-Bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transitions on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increments on high-to-low transition on T0CKI pin
0 = Increments on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = Timer0 prescaler is not assigned; Timer0 clock input bypasses prescaler
0 = Timer0 prescaler is assigned; Timer0 clock input comes from prescaler output

bit 2-0 **T0PS<2:0>**: Timer0 Prescaler Select bits
111 = 1:256 Prescale value
110 = 1:128 Prescale value
101 = 1:64 Prescale value
100 = 1:32 Prescale value
011 = 1:16 Prescale value
010 = 1:8 Prescale value
001 = 1:4 Prescale value
000 = 1:2 Prescale value

13.1 Timer0 Operation

Timer0 can operate as either a timer or a counter. The mode is selected with the T0CS bit (T0CON<5>). In Timer mode (T0CS = 0), the module increments on every clock by default unless a different prescaler value is selected (see **Section 13.3 “Prescaler”**). If the TMR0 register is written to, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0 register.

The Counter mode is selected by setting the T0CS bit (= 1). In this mode, Timer0 increments either on every rising edge or falling edge of the T0CKI pin. The incrementing edge is determined by the Timer0 Source Edge Select bit, T0SE (T0CON<4>); clearing this bit selects the rising edge. Restrictions on the external clock input are discussed below.

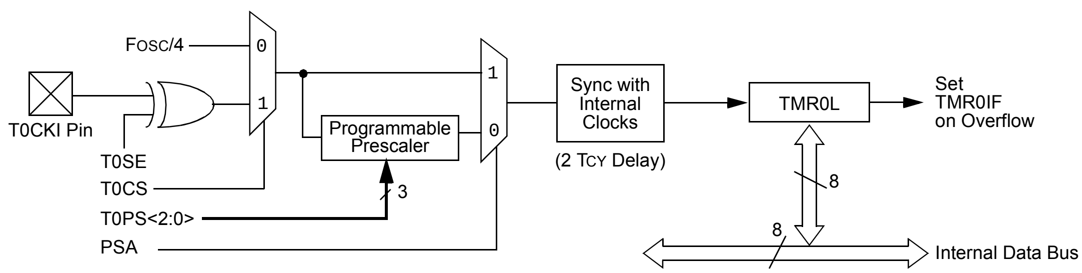
An external clock source can be used to drive Timer0; however, it must meet certain requirements to ensure that the external clock can be synchronized with the internal phase clock (TOSC). There is a delay between synchronization and the onset of incrementing the timer/counter.

13.2 Timer0 Reads and Writes in 16-Bit Mode

TMR0H is not the actual high byte of Timer0 in 16-bit mode. It is actually a buffered version of the real high byte of Timer0, which is not directly readable nor writable. (See Figure 13-2.) TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16 bits of Timer0 without having to verify that the read of the high and low byte were valid, due to a rollover between successive reads of the high and low byte.

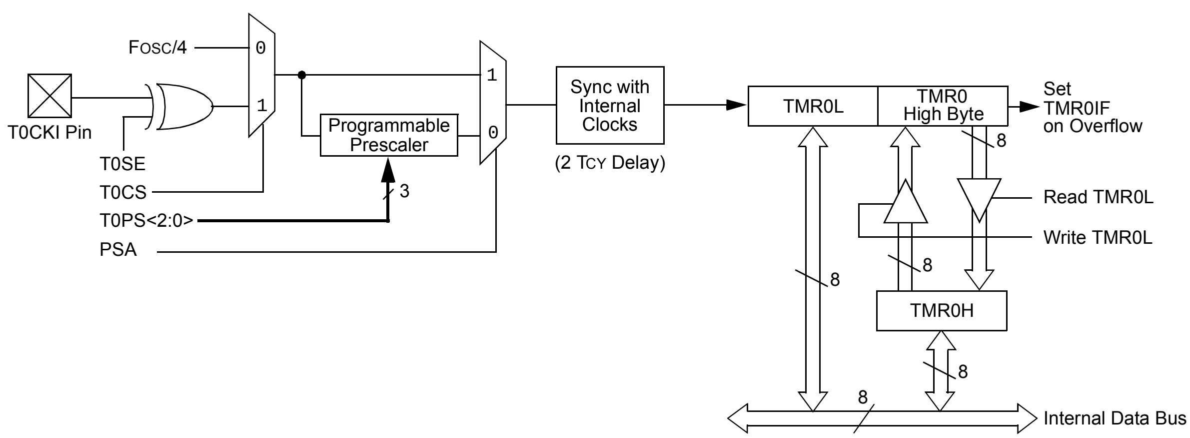
Similarly, a write to the high byte of Timer0 must also take place through the TMR0H Buffer register. The high byte is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16 bits of Timer0 to be updated at once.

FIGURE 13-1: TIMER0 BLOCK DIAGRAM (8-BIT MODE)



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

FIGURE 13-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

For more details on selecting the optimum C1 and C2 for a given crystal, see the crystal manufacture's applications information. The optimum value depends in part on the amount of parasitic capacitance in the circuit, which is often unknown. For that reason, it is highly recommended that thorough testing and validation of the oscillator be performed after values have been selected.

14.5.1 USING SOSC AS A CLOCK SOURCE

The SOSC oscillator is also available as a clock source in power-managed modes. By setting the clock select bits, SCS<1:0> (OSCCON<1:0>), to '01', the device switches to SEC_RUN mode and both the CPU and peripherals are clocked from the SOSC oscillator. If the IDLEN bit (OSCCON<7>) is cleared and a SLEEP instruction is executed, the device enters SEC_IDLE mode. Additional details are available in **Section 4.0 "Power-Managed Modes"**.

Whenever the SOSC oscillator is providing the clock source, the SOSC System Clock Status flag, SOSCRUN (OSCCON2<6>), is set. This can be used to determine the controller's current clocking mode. It can also indicate the clock source currently being used by the Fail-Safe Clock Monitor.

If the Clock Monitor is enabled and the SOSC oscillator fails while providing the clock, polling the SOCSRUN bit will indicate whether the clock is being provided by the SOSC oscillator or another source.

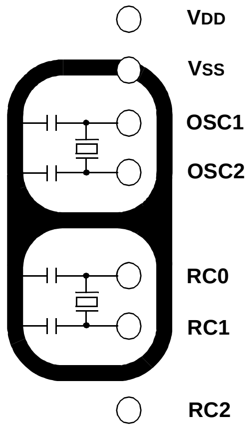
14.5.2 SOSC OSCILLATOR LAYOUT CONSIDERATIONS

The SOSC oscillator circuit draws very little power during operation. Due to the low-power nature of the oscillator, it may also be sensitive to rapidly changing signals in close proximity. This is especially true when the oscillator is configured for extremely Low-Power mode, SOSCSEL<1:0> (CONFIG1L<4:3>) = 01.

The oscillator circuit, displayed in Figure 14-2, should be located as close as possible to the microcontroller. There should be no circuits passing within the oscillator circuit boundaries other than VSS or VDD.

If a high-speed circuit must be located near the oscillator, it may help to have a grounded guard ring around the oscillator circuit. The guard, as displayed in Figure 14-3, could be used on a single-sided PCB or in addition to a ground plane. (Examples of a high-speed circuit include the ECCP1 pin, in Output Compare or PWM mode, or the primary oscillator, using the OSC2 pin.)

FIGURE 14-3: OSCILLATOR CIRCUIT WITH GROUNDED GUARD RING



Note: Not drawn to scale.

In the Low Drive Level mode, SOSCSEL<1:0> = 01, it is critical that RC2 I/O pin signals be kept away from the oscillator circuit. Configuring RC2 as a digital output, and toggling it, can potentially disturb the oscillator circuit, even with a relatively good PCB layout. If possible, either leave RC2 unused or use it as an input pin with a slew rate limited signal source. If RC2 must be used as a digital output, it may be necessary to use the Higher Drive Level Oscillator mode (SOSCSEL<1:0> = 11) with many PCB layouts.

Even in the Higher Drive Level mode, careful layout procedures should still be followed when designing the oscillator circuit.

In addition to dV/dt induced noise considerations, it is important to ensure that the circuit board is clean. Even a very small amount of conductive, soldering flux residue can cause PCB leakage currents that can overwhelm the oscillator circuit.

14.6 Timer1 Interrupt

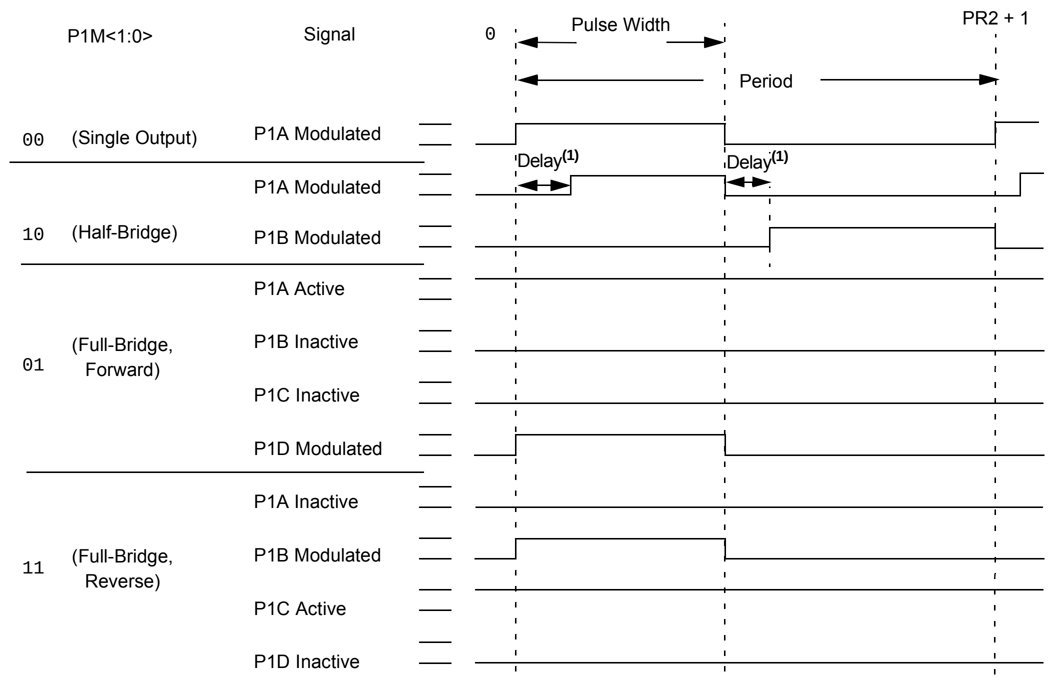
The TMR1 register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The Timer1 interrupt, if enabled, is generated on overflow which is latched in interrupt flag bit, TMR1IF (PIR1<0>). This interrupt can be enabled or disabled by setting or clearing the Timer1 Interrupt Enable bit, TMR1IE (PIE1<0>).

TABLE 20-2: EXAMPLE PIN ASSIGNMENTS FOR VARIOUS PWM ENHANCED MODES

ECCP Mode	P1M<1:0>	P1A	P1B	P1C	P1D
Single	00	Yes[1]	Yes[1]	Yes[1]	Yes[1]
Half-Bridge	10	Yes	Yes	No	No
Full-Bridge, Forward	01	Yes	Yes	Yes	Yes
Full-Bridge, Reverse	11	Yes	Yes	Yes	Yes

Note 1: Outputs are enabled by pulse steering in Single mode (see Register 20-5).

FIGURE 20-4: EXAMPLE PWM (ENHANCED MODE) OUTPUT RELATIONSHIPS (ACTIVE-HIGH STATE)



Relationships:
- Period = 4 * TOSC * (PR2 + 1) * (TMR2 Prescale Value)
- Pulse Width = TOSC * (CCPR1L<7:0>:CCP1CON<5:4>) * (TMR2 Prescale Value)
- Delay = 4 * TOSC * (ECCP1DEL<6:0>)

Note 1: Dead-band delay is programmed using the ECCP1DEL register (**Section 20.4.6 "Programmable Dead-Band Delay Mode"**).

23.0 12-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) Converter module in the PIC18F66K80 family of devices. It is a 13-bit differential A/D with 12-bit single-ended compatibility. It has inputs eight inputs for the 28-pin devices, 11 inputs for the 40/44-pin and 64-pin devices. This module allows conversion of an analog input signal to a corresponding 12-bit digital number.

The module has these registers:

- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Control Register 2 (ADCON2)
- A/D Port Configuration Register 1 (ANCON0)
- A/D Port Configuration Register 2 (ANCON1)
- ADRESH (the upper, A/D Results register)
- ADRESL (the lower, A/D Results register)

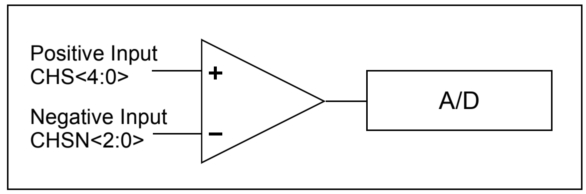
The ADCON0 register, shown in Register 23-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 23-2, configures the voltage reference and special trigger selection. The ADCON2 register, shown in Register 23-3, configures the A/D clock source and programmed acquisition time and justification.

23.1 Differential A/D Converter

The converter in PIC18F66K80 family devices is implemented as a differential A/D where the differential voltage between two channels is measured and converted to digital values (see Figure 23-1).

The converter also can be configured to measure a voltage from a single input by clearing the CHSNx bits (ADCON1<2:0>). With this configuration, the negative channel input is connected internally to AVSS (see Figure 23-2).

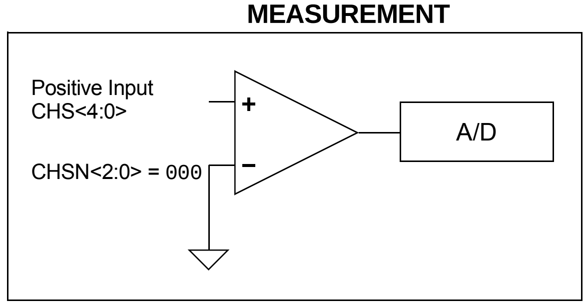
FIGURE 23-1: DIFFERENTIAL CHANNEL MEASUREMENT



Differential conversion feeds the two input channels to a unity gain differential amplifier. The positive channel input is selected using the CHSx bits (ADCON0<6:2>) and the negative channel input is selected using the CHSNx bits (ADCON1<2:0>).

The output from the amplifier is fed to the A/D Converter, as shown in Figure 23-1. The 12-bit result is available on the ADRESH and ADRESL registers. An additional bit indicates if the 12-bit result is a positive or negative value.

FIGURE 23-2: SINGLE CHANNEL MEASUREMENT



In the Single Channel Measurement mode, the negative input is connected to AVSS by clearing the CHSNx bits (ADCON1<2:0>).

24.6 Comparator Interrupts

The comparator interrupt flag is set whenever any of the following occurs:

- Low-to-high transition of the comparator output
- High-to-low transition of the comparator output
- Any change in the comparator output

The comparator interrupt selection is done by the EVPOL<1:0> bits in the CMxCON register (CMxCON<4:3>).

In order to provide maximum flexibility, the output of the comparator may be inverted using the CPOL bit in the CMxCON register (CMxCON<5>). This is functionally identical to reversing the inverting and non-inverting inputs of the comparator for a particular mode.

An interrupt is generated on the low-to-high or high-to-low transition of the comparator output. This mode of interrupt generation is dependent on EVPOL<1:0> in the CMxCON register. When EVPOL<1:0> = `01` or `10`, the interrupt is generated on a low-to-high or high-to-low transition of the comparator output. Once the interrupt is generated, it is required to clear the interrupt flag by software.

When EVPOL<1:0> = `11`, the comparator interrupt flag is set whenever there is a change in the output value of either comparator. Software will need to maintain information about the status of the output bits, as read from CMSTAT<7:6>, to determine the actual change that occurred.

The CMPxIF<2:0> (PIR4<5:4) bits are the Comparator Interrupt Flags. The CMPxIF bits must be reset by clearing them. Since it is also possible to write a '`1`' to this register, a simulated interrupt may be initiated. Table 24-2 shows the interrupt generation with respect to comparator input voltages and EVPOL bit settings.

Both the CMPxIE bits (PIE4<5:4>) and the PEIE bit (INTCON<6>) must be set to enable the interrupt. In addition, the GIE bit (INTCON<7>) must also be set. If any of these bits are clear, the interrupt is not enabled, though the CMPxIF bits will still be set if an interrupt condition occurs.

A simplified diagram of the interrupt section is shown in Figure 24-3.

> **Note:** CMPxIF will not be set when EVPOL<1:0> = `00`.

TABLE 24-2: COMPARATOR INTERRUPT GENERATION

CPOL	EVPOL<1:0>	Comparator Input Change	CxOUT Transition	Interrupt Generated
0	00	$V_{IN+} > V_{IN-}$	Low-to-High	No
		$V_{IN+} < V_{IN-}$	High-to-Low	No
	01	$V_{IN+} > V_{IN-}$	Low-to-High	Yes
		$V_{IN+} < V_{IN-}$	High-to-Low	No
	10	$V_{IN+} > V_{IN-}$	Low-to-High	No
		$V_{IN+} < V_{IN-}$	High-to-Low	Yes
	11	$V_{IN+} > V_{IN-}$	Low-to-High	Yes
		$V_{IN+} < V_{IN-}$	High-to-Low	Yes
1	00	$V_{IN+} > V_{IN-}$	High-to-Low	No
		$V_{IN+} < V_{IN-}$	Low-to-High	No
	01	$V_{IN+} > V_{IN-}$	High-to-Low	No
		$V_{IN+} < V_{IN-}$	Low-to-High	Yes
	10	$V_{IN+} > V_{IN-}$	High-to-Low	Yes
		$V_{IN+} < V_{IN-}$	Low-to-High	No
	11	$V_{IN+} > V_{IN-}$	High-to-Low	Yes
		$V_{IN+} < V_{IN-}$	Low-to-High	Yes

27.0 ECAN MODULE

PIC18F66K80 family devices contain an Enhanced Controller Area Network (ECAN) module. The ECAN module is fully backward compatible with the CAN module available in PIC18CXX8 and PIC18FXX8 devices and the ECAN module in PIC18Fxx80 devices.

The Controller Area Network (CAN) module is a serial interface which is useful for communicating with other peripherals or microcontroller devices. This interface, or protocol, was designed to allow communications within noisy environments.

The ECAN module is a communication controller, implementing the CAN 2.0A or B protocol as defined in the BOSCH specification. The module will support CAN 1.2, CAN 2.0A, CAN 2.0B Passive and CAN 2.0B Active versions of the protocol. The module implementation is a full CAN system; however, the CAN specification is not covered within this data sheet. Refer to the BOSCH CAN specification for further details.

The module features are as follows:

- Implementation of the CAN protocol, CAN 1.2, CAN 2.0A and CAN 2.0B
- DeviceNet™ data bytes filter support
- Standard and extended data frames
- 0-8 bytes data length
- Programmable bit rate up to 1 Mbit/sec
- Fully backward compatible with the PIC18XXX8 CAN module
- Three modes of operation:
  - Mode 0 – Legacy mode
  - Mode 1 – Enhanced Legacy mode with DeviceNet support
  - Mode 2 – FIFO mode with DeviceNet support
- Support for remote frames with automated handling
- Double-buffered receiver with two prioritized received message storage buffers
- Six buffers programmable as RX and TX message buffers
- 16 full (standard/extended identifier) acceptance filters that can be linked to one of four masks
- Two full acceptance filter masks that can be assigned to any filter
- One full acceptance filter that can be used as either an acceptance filter or acceptance filter mask
- Three dedicated transmit buffers with application specified prioritization and abort capability
- Programmable wake-up functionality with integrated low-pass filter
- Programmable Loopback mode supports self-test operation
- Signaling via interrupt capabilities for all CAN receiver and transmitter error states
- Programmable clock source
- Programmable link to timer module for time-stamping and network synchronization
- Low-power Sleep mode

27.1 Module Overview

The CAN bus module consists of a protocol engine and message buffering and control. The CAN protocol engine automatically handles all functions for receiving and transmitting messages on the CAN bus. Messages are transmitted by first loading the appropriate data registers. Status and errors can be checked by reading the appropriate registers. Any message detected on the CAN bus is checked for errors and then matched against filters to see if it should be received and stored in one of the two receive registers.

The CAN module supports the following frame types:

- Standard Data Frame
- Extended Data Frame
- Remote Frame
- Error Frame
- Overload Frame Reception

The CAN module uses the RB2/CANTX and RB3/CANRX pins to interface with the CAN bus. The CANTX and CANRX pins can be placed on alternate I/O pins by setting the CANMX (CONFIG3H<0>) Configuration bit.

For the PIC18F2XK80 and PIC18F4XK80, the alternate pin locations are RC6/CANTX and RC7/CANRX. For the PIC18F6XK80, the alternate pin locations are RE4/CANRX and RE5/CANTX.

In normal mode, the CAN module automatically overrides the appropriate TRIS bit for CANTX. The user must ensure that the appropriate TRIS bit for CANRX is set.

27.1.1 MODULE FUNCTIONALITY

The CAN bus module consists of a protocol engine, message buffering and control (see Figure 27-1). The protocol engine can best be understood by defining the types of data frames to be transmitted and received by the module.

The following sequence illustrates the necessary initialization steps before the ECAN module can be used to transmit or receive a message. Steps can be added or removed depending on the requirements of the application.

1. Initial LAT and TRIS bits for RX and TX CAN.
2. Ensure that the ECAN module is in Configuration mode.
3. Select ECAN Operational mode.
4. Set up the Baud Rate registers.
5. Set up the Filter and Mask registers.
6. Set the ECAN module to normal mode or any other mode required by the application logic.

REGISTER 27-16: RXBnSIDL: RECEIVE BUFFER ‘n’ STANDARD IDENTIFIER REGISTERS, LOW BYTE [$0 \leq n \leq 1$]

R-x	R-x	R-x	R-x	R-x	U-0	R-x	R-x
SID2	SID1	SID0	SRR	EXID	—	EID17	EID16
bit 7							bit 0

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

bit 7-5 **SID<2:0>:** Standard Identifier bits (if EXID = 0)
Extended Identifier bits, EID<20:18> (if EXID = 1).

bit 4 **SRR:** Substitute Remote Request bit

bit 3 **EXID:** Extended Identifier bit
1 = Received message is an extended data frame, SID<10:0> are EID<28:18>
0 = Received message is a standard data frame

bit 2 **Unimplemented:** Read as ‘0’

bit 1-0 **EID<17:16>:** Extended Identifier bits

REGISTER 27-17: RXBnEIDH: RECEIVE BUFFER ‘n’ EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [$0 \leq n \leq 1$]

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
bit 7							bit 0

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

bit 7-0 **EID<15:8>:** Extended Identifier bits

REGISTER 27-18: RXBnEIDL: RECEIVE BUFFER ‘n’ EXTENDED IDENTIFIER REGISTERS, LOW BYTE [$0 \leq n \leq 1$]

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
bit 7							bit 0

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

bit 7-0 **EID<7:0>:** Extended Identifier bits

27.3.4 LISTEN ONLY MODE

Listen Only mode provides a means for the PIC18F66K80 family devices to receive all messages, including messages with errors. This mode can be used for bus monitor applications or for detecting the baud rate in 'hot plugging' situations. For auto-baud detection, it is necessary that there are at least two other nodes which are communicating with each other. The baud rate can be detected empirically by testing different values until valid messages are received. The Listen Only mode is a silent mode, meaning no messages will be transmitted while in this state, including error flags or Acknowledge signals. In Listen Only mode, both valid and invalid messages will be received, regardless of RXMn bit settings. The filters and masks can still be used to allow only particular valid messages to be loaded into the Receive registers, or the filter masks can be set to all zeros to allow a message with any identifier to pass. All invalid messages will be received in this mode, regardless of filters and masks or RXMn Receive Buffer mode bits.The error counters are reset and deactivated in this state. The Listen Only mode is activated by setting the mode request bits in the CANCON register.

27.3.5 LOOPBACK MODE

This mode will allow internal transmission of messages from the transmit buffers to the receive buffers without actually transmitting messages on the CAN bus. This mode can be used in system development and testing. In this mode, the ACK bit is ignored and the device will allow incoming messages from itself, just as if they were coming from another node. The Loopback mode is a silent mode, meaning no messages will be transmitted while in this state, including error flags or Acknowledge signals. The TXCAN pin will revert to port I/O while the device is in this mode. The filters and masks can be used to allow only particular messages to be loaded into the receive registers. The masks can be set to all zeros to provide a mode that accepts all messages. The Loopback mode is activated by setting the mode request bits in the CANCON register.

27.3.6 ERROR RECOGNITION MODE

The module can be set to ignore all errors and receive any message. In functional Mode 0, the Error Recognition mode is activated by setting the RXM<1:0> bits in the RXBnCON registers to '11'. In this mode, the data which is in the message assembly buffer until the error time, is copied in the receive buffer and can be read via the CPU interface.

27.4 CAN Module Functional Modes

In addition to CAN modes of operation, the ECAN module offers a total of 3 functional modes. Each of these modes are identified as Mode 0, Mode 1 and Mode 2.

27.4.1 MODE 0 – LEGACY MODE

Mode 0 is designed to be fully compatible with CAN modules used in PIC18CXX8 and PIC18FXX8 devices. This is the default mode of operation on all Reset conditions. As a result, module code written for the PIC18XX8 CAN module may be used on the ECAN module without any code changes.

The following is the list of resources available in Mode 0:

- Three transmit buffers: TXB0, TXB1 and TXB2
- Two receive buffers: RXB0 and RXB1
- Two acceptance masks, one for each receive buffer: RXM0, RXM1
- Six acceptance filters, 2 for RXB0 and 4 for RXB1: RXF0, RXF1, RXF2, RXF3, RXF4, RXF5

27.4.2 MODE 1 – ENHANCED LEGACY MODE

Mode 1 is similar to Mode 0, with the exception that more resources are available in Mode 1. There are 16 acceptance filters and two acceptance mask registers. Acceptance Filter 15 can be used as either an acceptance filter or an acceptance mask register. In addition to three transmit and two receive buffers, there are six more message buffers. One or more of these additional buffers can be programmed as transmit or receive buffers. These additional buffers can also be programmed to automatically handle RTR messages.

Fourteen of sixteen acceptance filter registers can be dynamically associated to any receive buffer and acceptance mask register. One can use this capability to associate more than one filter to any one buffer.

When a receive buffer is programmed to use standard identifier messages, part of the full acceptance filter register can be used as a data byte filter. The length of the data byte filter is programmable from 0 to 18 bits. This functionality simplifies implementation of high-level protocols, such as the DeviceNet™ protocol.

The following is the list of resources available in Mode 1:

- Three transmit buffers: TXB0, TXB1 and TXB2
- Two receive buffers: RXB0 and RXB1
- Six buffers programmable as TX or RX: B0-B5
- Automatic RTR handling on B0-B5
- Sixteen dynamically assigned acceptance filters: RXF0-RXF15
- Two dedicated acceptance mask registers; RXF15 programmable as third mask: RXM0-RXM1, RXF15
- Programmable data filter on standard identifier messages: SDFLC

27.15.6.1 Receiver Overflow

An overflow condition occurs when the MAB has assembled a valid received message (the message meets the criteria of the acceptance filters) and the receive buffer associated with the filter is not available for loading of a new message. The associated RXBnOVFL bit in the COMSTAT register will be set to indicate the overflow condition. This bit must be cleared by the MCU.

27.15.6.2 Receiver Warning

The receive error counter has reached the MCU warning limit of 96.

27.15.6.3 Transmitter Warning

The transmit error counter has reached the MCU warning limit of 96.

27.15.6.4 Receiver Bus Passive

This will occur when the device has gone to the error-passive state because the receive error counter is greater or equal to 128.

27.15.6.5 Transmitter Bus Passive

This will occur when the device has gone to the error-passive state because the transmit error counter is greater or equal to 128.

27.15.6.6 Bus-Off

The transmit error counter has exceeded 255 and the device has gone to bus-off state.

FIGURE 29-1: GENERAL FORMAT FOR INSTRUCTIONS

Example Instruction

Byte-oriented file register operations

15 10	9	8	7 0
OPCODE	d	a	f (FILE #)

Example: `ADDWF MYREG, W, B`

d = `0` for result destination to be WREG register
d = `1` for result destination to be file register (f)
a = `0` to force Access Bank
a = `1` for BSR to select bank
f = 8-bit file register address

Byte to Byte move operations (2-word)

15 12	11 0
OPCODE	f (Source FILE #)
`1111`	f (Destination FILE #)

Example: `MOVFF MYREG1, MYREG2`

f = 12-bit file register address

Bit-oriented file register operations

15 12	11 9	8	7 0
OPCODE	b (BIT #)	a	f (FILE #)

Example: `BSF MYREG, bit, B`

b = 3-bit position of bit in file register (f)
a = `0` to force Access Bank
a = `1` for BSR to select bank
f = 8-bit file register address

Literal operations

15 8	7 0
OPCODE	k (literal)

Example: `MOVLW 7Fh`

k = 8-bit immediate value

Control operations

`CALL`, `GOTO` and Branch operations

15 8	7 0
OPCODE	n<7:0> (literal)

15 12	11 0
`1111`	n<19:8> (literal)

Example: `GOTO Label`

n = 20-bit immediate value

15 8		7 0
OPCODE	S	n<7:0> (literal)

15 12	11 0
`1111`	n<19:8> (literal)

Example: `CALL MYFUNC`

S = Fast bit

15 11	10 0
OPCODE	n<10:0> (literal)

Example: `BRA MYFUNC`

15 8	7 0
OPCODE	n<7:0> (literal)

Example: `BC MYFUNC`

30.2 MPLAB C Compilers for Various Device Families

The MPLAB C Compiler code development systems are complete ANSI C compilers for Microchip's PIC18, PIC24 and PIC32 families of microcontrollers and the dsPIC30 and dsPIC33 families of digital signal controllers. These compilers provide powerful integration capabilities, superior code optimization and ease of use.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

30.3 HI-TECH C for Various Device Families

The HI-TECH C Compiler code development systems are complete ANSI C compilers for Microchip's PIC family of microcontrollers and the dsPIC family of digital signal controllers. These compilers provide powerful integration capabilities, omniscient code generation and ease of use.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

The compilers include a macro assembler, linker, preprocessor, and one-step driver, and can run on multiple platforms.

30.4 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multi-purpose source files
- Directives that allow complete control over the assembly process

30.5 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler and the MPLAB C18 C Compiler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

30.6 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC devices. MPLAB C Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility