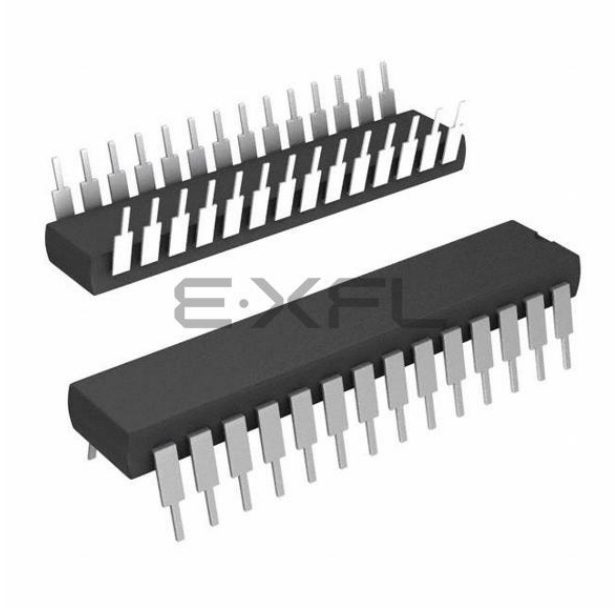


**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

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

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Active |
| Core Processor | PIC |
| Core Size | 8-Bit |
| Speed | 40MHz |
| Connectivity | CANbus, I²C, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, HLVD, POR, PWM, WDT |
| Number of I/O | 25 |
| Program Memory Size | 80KB (40K x 16) |
| Program Memory Type | FLASH |
| EEPROM Size | 1K x 8 |
| RAM Size | 3.25K x 8 |
| Voltage - Supply (Vcc/Vdd) | 2V ~ 5.5V |
| Data Converters | A/D 8x10b |
| 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/pic18lf2682-i-sp |

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

| Register | Applicable Devices | | | | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Resets, WDT Reset, RESET Instruction, Stack Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|---|---|
| ADCON2 | 2682 | 2685 | 4682 | 4685 | 0-00 0000 | 0-00 0000 | u-uu uuuu |
| CCPR1H | 2682 | 2685 | 4682 | 4685 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR1L | 2682 | 2685 | 4682 | 4685 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP1CON | 2682 | 2685 | 4682 | 4685 | --00 0000 | --00 0000 | --uu uuuu |
| ECCPR1H | 2682 | 2685 | 4682 | 4685 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ECCPR1L | 2682 | 2685 | 4682 | 4685 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ECCP1CON | 2682 | 2685 | 4682 | 4685 | 0000 0000 | 0000 0000 | uuuu uuuu |
| BAUDCON | 2682 | 2685 | 4682 | 4685 | 01-0 0-00 | 01-0 0-00 | --uu uuuu |
| ECCP1DEL | 2682 | 2685 | 4682 | 4685 | 0000 0000 | 0000 0000 | uuuu uuuu |
| ECCP1AS | 2682 | 2685 | 4682 | 4685 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CVRCON | 2682 | 2685 | 4682 | 4685 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CMCON | 2682 | 2685 | 4682 | 4685 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TMR3H | 2682 | 2685 | 4682 | 4685 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR3L | 2682 | 2685 | 4682 | 4685 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T3CON | 2682 | 2685 | 4682 | 4685 | 0000 0000 | uuuu uuuu | uuuu uuuu |
| SPBRGH | 2682 | 2685 | 4682 | 4685 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SPBRG | 2682 | 2685 | 4682 | 4685 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RCREG | 2682 | 2685 | 4682 | 4685 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXREG | 2682 | 2685 | 4682 | 4685 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXSTA | 2682 | 2685 | 4682 | 4685 | 0000 0010 | 0000 0010 | uuuu uuuu |
| RCSTA | 2682 | 2685 | 4682 | 4685 | 0000 000x | 0000 000x | uuuu uuuu |
| EEADRH | 2682 | 2685 | 4682 | 4685 | ---- --00 | ---- --00 | ---- --uu |
| EEADR | 2682 | 2685 | 4682 | 4685 | 0000 0000 | 0000 0000 | uuuu uuuu |
| EEDATA | 2682 | 2685 | 4682 | 4685 | 0000 0000 | 0000 0000 | uuuu uuuu |
| EECON2 | 2682 | 2685 | 4682 | 4685 | 0000 0000 | 0000 0000 | 0000 0000 |
| EECON1 | 2682 | 2685 | 4682 | 4685 | xx-0 x000 | uu-0 u000 | uu-0 u000 |
| IPR3 | 2682 | 2685 | 4682 | 4685 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PIR3 | 2682 | 2685 | 4682 | 4685 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PIE3 | 2682 | 2685 | 4682 | 4685 | 0000 0000 | 0000 0000 | uuuu uuuu |
| IPR2 | 2682 | 2685 | 4682 | 4685 | 11-1 1111 | 11-1 1111 | uu-u uuuu |
| | 2682 | 2685 | 4682 | 4685 | 1--1 111- | 1--1 111- | u--u uuu- |

**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 4-3 for Reset value for specific condition.

**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 '0'.

**6:** This register reads all '0's until the ECAN™ technology is set up in Mode 1 or Mode 2.

**TABLE 5-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F2682/2685/4682/4685 DEVICES (CONTINUED)**

| Address | Name | Address | Name | Address | Name | Address | Name |
|---|---|---|---|---|---|---|---|
| DFFh | — | DDFh | — | DBFh | — | D9Fh | — |
| DFEh | — | DDEh | — | DBEh | — | D9Eh | — |
| DFDh | — | DDDh | — | DBDh | — | D9Dh | — |
| DFCh | TXBIE | DDCh | — | DBCh | — | D9Ch | — |
| DFBh | — | DDBh | — | DBBh | — | D9Bh | — |
| DFAh | BIE0 | DDAh | — | DBAh | — | D9Ah | — |
| DF9h | — | DD9h | — | DB9h | — | D99h | — |
| DF8h | BSEL0 | DD8h | SDFLC | DB8h | — | D98h | — |
| DF7h | — | DD7h | — | DB7h | — | D97h | — |
| DF6h | — | DD6h | — | DB6h | — | D96h | — |
| DF5h | — | DD5h | RXFCON1 | DB5h | — | D95h | — |
| DF4h | — | DD4h | RXFCON0 | DB4h | — | D94h | — |
| DF3h | MSEL3 | DD3h | — | DB3h | — | D93h | RXF15EIDL |
| DF2h | MSEL2 | DD2h | — | DB2h | — | D92h | RXF15EIDH |
| DF1h | MSEL1 | DD1h | — | DB1h | — | D91h | RXF15SIDL |
| DF0h | MSEL0 | DD0h | — | DB0h | — | D90h | RXF15SIDH |
| DEFh | — | DCFh | — | DAFh | — | D8Fh | — |
| DEEh | — | DCEh | — | DAEh | — | D8Eh | — |
| DEDh | — | DCDh | — | DADh | — | D8Dh | — |
| DECh | — | DCCh | — | DACh | — | D8Ch | — |
| DEBh | — | DCBh | — | DABh | — | D8Bh | RXF14EIDL |
| DEAh | — | DCAh | — | DAAh | — | D8Ah | RXF14EIDH |
| DE9h | — | DC9h | — | DA9h | — | D89h | RXF14SIDL |
| DE8h | — | DC8h | — | DA8h | — | D88h | RXF14SIDH |
| DE7h | RXFBCON7 | DC7h | — | DA7h | — | D87h | RXF13EIDL |
| DE6h | RXFBCON6 | DC6h | — | DA6h | — | D86h | RXF13EIDH |
| DE5h | RXFBCON5 | DC5h | — | DA5h | — | D85h | RXF13SIDL |
| DE4h | RXFBCON4 | DC4h | — | DA4h | — | D84h | RXF13SIDH |
| DE3h | RXFBCON3 | DC3h | — | DA3h | — | D83h | RXF12EIDL |
| DE2h | RXFBCON2 | DC2h | — | DA2h | — | D82h | RXF12EIDH |
| DE1h | RXFBCON1 | DC1h | — | DA1h | — | D81h | RXF12SIDL |
| DE0h | RXFBCON0 | DC0h | — | DA0h | — | D80h | RXF12SIDH |

**Note 1:** Registers available only on PIC18F4X8X devices; otherwise, the registers read as '0'.
**2:** When any TX_ENn bit in RX_TX_SELn is set, then the corresponding bit in this register has transmit properties.
**3:** This is not a physical register.

**TABLE 5-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F2682/2685/4682/4685 DEVICES (CONTINUED)**

| Address | Name |
|---|---|
| D7Fh | — |
| D7Eh | — |
| D7Dh | — |
| D7Ch | — |
| D7Bh | RXF11EIDL |
| D7Ah | RXF11EIDH |
| D79h | RXF11SIDL |
| D78h | RXF11SIDH |
| D77h | RXF10EIDL |
| D76h | RXF10EIDH |
| D75h | RXF10SIDL |
| D74h | RXF10SIDH |
| D73h | RXF9EIDL |
| D72h | RXF9EIDH |
| D71h | RXF9SIDL |
| D70h | RXF9SIDH |
| D6Fh | — |
| D6Eh | — |
| D6Dh | — |
| D6Ch | — |
| D6Bh | RXF8EIDL |
| D6Ah | RXF8EIDH |
| D69h | RXF8SIDL |
| D68h | RXF8SIDH |
| D67h | RXF7EIDL |
| D66h | RXF7EIDH |
| D65h | RXF7SIDL |
| D64h | RXF7SIDH |
| D63h | RXF6EIDL |
| D62h | RXF6EIDH |
| D61h | RXF6SIDL |
| D60h | RXF6SIDH |

**Note 1:** Registers available only on PIC18F4X8X devices; otherwise, the registers read as '0'.
**2:** When any TX_ENn bit in RX_TX_SELn is set, then the corresponding bit in this register has transmit properties.
**3:** This is not a physical register.

**TABLE 5-2: REGISTER FILE SUMMARY (PIC18F2682/2685/4682/4685) (CONTINUED)**

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Details on page: |
|---|---|---|---|---|---|---|---|---|---|---|
| RXB1D5 | RXB1D57 | RXB1D56 | RXB1D55 | RXB1D54 | RXB1D53 | RXB1D52 | RXB1D51 | RXB1D50 | xxxx xxxx | 55, 295 |
| RXB1D4 | RXB1D47 | RXB1D46 | RXB1D45 | RXB1D44 | RXB1D43 | RXB1D42 | RXB1D41 | RXB1D40 | xxxx xxxx | 55, 295 |
| RXB1D3 | RXB1D37 | RXB1D36 | RXB1D35 | RXB1D34 | RXB1D33 | RXB1D32 | RXB1D31 | RXB1D30 | xxxx xxxx | 55, 295 |
| RXB1D2 | RXB1D27 | RXB1D26 | RXB1D25 | RXB1D24 | RXB1D23 | RXB1D22 | RXB1D21 | RXB1D20 | xxxx xxxx | 55, 295 |
| RXB1D1 | RXB1D17 | RXB1D16 | RXB1D15 | RXB1D14 | RXB1D13 | RXB1D12 | RXB1D11 | RXB1D10 | xxxx xxxx | 55, 295 |
| RXB1D0 | RXB1D07 | RXB1D06 | RXB1D05 | RXB1D04 | RXB1D03 | RXB1D02 | RXB1D01 | RXB1D00 | xxxx xxxx | 55, 295 |
| RXB1DLC | — | RXRTR | RB1 | RB0 | DLC3 | DLC2 | DLC1 | DLC0 | -xxx xxxx | 55, 295 |
| RXB1EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 55, 294 |
| RXB1EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 55, 294 |
| RXB1SIDL | SID2 | SID1 | SID0 | SRR | EXID | — | EID17 | EID16 | xxxx xxxx | 55, 294 |
| RXB1SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 56, 293 |
| RXB1CON Mode 0 | RXFUL | RXM1 | RXM0(7) | —(7) | RXRTRRO(7) | FILHIT2(7) | FILHIT1(7) | FILHIT0(7) | 000- 0000 | 56, 292 |
| RXB1CON Mode 1, 2 | RXFUL | RXM1 | RTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 | 0000 0000 | 56, 292 |
| TXB0D7 | TXB0D77 | TXB0D76 | TXB0D75 | TXB0D74 | TXB0D73 | TXB0D72 | TXB0D71 | TXB0D70 | xxxx xxxx | 56, 286 |
| TXB0D6 | TXB0D67 | TXB0D66 | TXB0D65 | TXB0D64 | TXB0D63 | TXB0D62 | TXB0D61 | TXB0D60 | xxxx xxxx | 56, 286 |
| TXB0D5 | TXB0D57 | TXB0D56 | TXB0D55 | TXB0D54 | TXB0D53 | TXB0D52 | TXB0D51 | TXB0D50 | xxxx xxxx | 56, 286 |
| TXB0D4 | TXB0D47 | TXB0D46 | TXB0D45 | TXB0D44 | TXB0D43 | TXB0D42 | TXB0D41 | TXB0D40 | xxxx xxxx | 56, 286 |
| TXB0D3 | TXB0D37 | TXB0D36 | TXB0D35 | TXB0D34 | TXB0D33 | TXB0D32 | TXB0D31 | TXB0D30 | xxxx xxxx | 56, 286 |
| TXB0D2 | TXB0D27 | TXB0D26 | TXB0D25 | TXB0D24 | TXB0D23 | TXB0D22 | TXB0D21 | TXB0D20 | xxxx xxxx | 56, 286 |
| TXB0D1 | TXB0D17 | TXB0D16 | TXB0D15 | TXB0D14 | TXB0D13 | TXB0D12 | TXB0D11 | TXB0D10 | xxxx xxxx | 56, 286 |
| TXB0D0 | TXB0D07 | TXB0D06 | TXB0D05 | TXB0D04 | TXB0D03 | TXB0D02 | TXB0D01 | TXB0D00 | xxxx xxxx | 56, 286 |
| TXB0DLC | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | -x-- xxxx | 56, 287 |
| TXB0EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 56, 286 |
| TXB0EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 56, 285 |
| TXB0SIDL | SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | xxx- x-xx | 56, 285 |
| TXB0SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 56, 285 |
| TXB0CON | TXBIF | TXABT | TXLARB | TXERR | TXREQ | — | TXPRI1 | TXPRI0 | 0000 0-00 | 56, 284 |
| TXB1D7 | TXB1D77 | TXB1D76 | TXB1D75 | TXB1D74 | TXB1D73 | TXB1D72 | TXB1D71 | TXB1D70 | xxxx xxxx | 56, 286 |
| TXB1D6 | TXB1D67 | TXB1D66 | TXB1D65 | TXB1D64 | TXB1D63 | TXB1D62 | TXB1D61 | TXB1D60 | xxxx xxxx | 56, 286 |
| TXB1D5 | TXB1D57 | TXB1D56 | TXB1D55 | TXB1D54 | TXB1D53 | TXB1D52 | TXB1D51 | TXB1D50 | xxxx xxxx | 56, 286 |
| TXB1D4 | TXB1D47 | TXB1D46 | TXB1D45 | TXB1D44 | TXB1D43 | TXB1D42 | TXB1D41 | TXB1D40 | xxxx xxxx | 56, 286 |
| TXB1D3 | TXB1D37 | TXB1D36 | TXB1D35 | TXB1D34 | TXB1D33 | TXB1D32 | TXB1D31 | TXB1D30 | xxxx xxxx | 56, 286 |
| TXB1D2 | TXB1D27 | TXB1D26 | TXB1D25 | TXB1D24 | TXB1D23 | TXB1D22 | TXB1D21 | TXB1D20 | xxxx xxxx | 56, 286 |
| TXB1D1 | TXB1D17 | TXB1D16 | TXB1D15 | TXB1D14 | TXB1D13 | TXB1D12 | TXB1D11 | TXB1D10 | xxxx xxxx | 56, 286 |
| TXB1D0 | TXB1D07 | TXB1D06 | TXB1D05 | TXB1D04 | TXB1D03 | TXB1D02 | TXB1D01 | TXB1D00 | xxxx xxxx | 56, 286 |
| TXB1DLC | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | -x-- xxxx | 56, 287 |
| TXB1EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 56, 286 |
| TXB1EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 56, 285 |
| TXB1SIDL | SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | xxx- x-xx | 56, 285 |
| TXB1SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 56, 285 |

**Legend:** x = unknown, u = unchanged, - = unimplemented, q = value depends on condition. Shaded cells are unimplemented, read as '0'.

**Note 1:** Bit 21 of the PC is only available in Test mode and Serial Programming modes.

**2:** The SBOREN bit is only available when CONFIG2L<1:0> = 01; otherwise, it is disabled and reads as '0'. See **Section 4.4 "Brown-out Reset (BOR)"**.

**3:** These registers and/or bits are not implemented on PIC18F2682/2685 devices and are read as '0'. Reset values are shown for PIC18F4682/4685 devices; individual unimplemented bits should be interpreted as '—'.

**4:** The PLLEN bit is only available in specific oscillator configurations; otherwise, it is disabled and reads as '0'. See **Section 2.6.4 "PLL in INTOSC Modes"**.

**5:** The RE3 bit is only available when Master Clear Reset is disabled (CONFIG3H<7> = 0); otherwise, RE3 reads as '0'. This bit is read-only.

**6:** RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**7:** CAN bits have multiple functions depending on the selected mode of the CAN module.

**8:** This register reads all '0's until the ECAN™ technology is set up in Mode 1 or Mode 2.

**9:** These registers and/or bits are available on PIC18F4682/4685 devices only.

### 17.3.5 MASTER MODE

The master can initiate the data transfer at any time because it controls the SCK. The master determines when the slave (Processor 2, Figure 17-2) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPBUF register as if a normal received byte (interrupts and status bits appropriately set). This could be useful in receiver applications as a “Line Activity Monitor” mode.

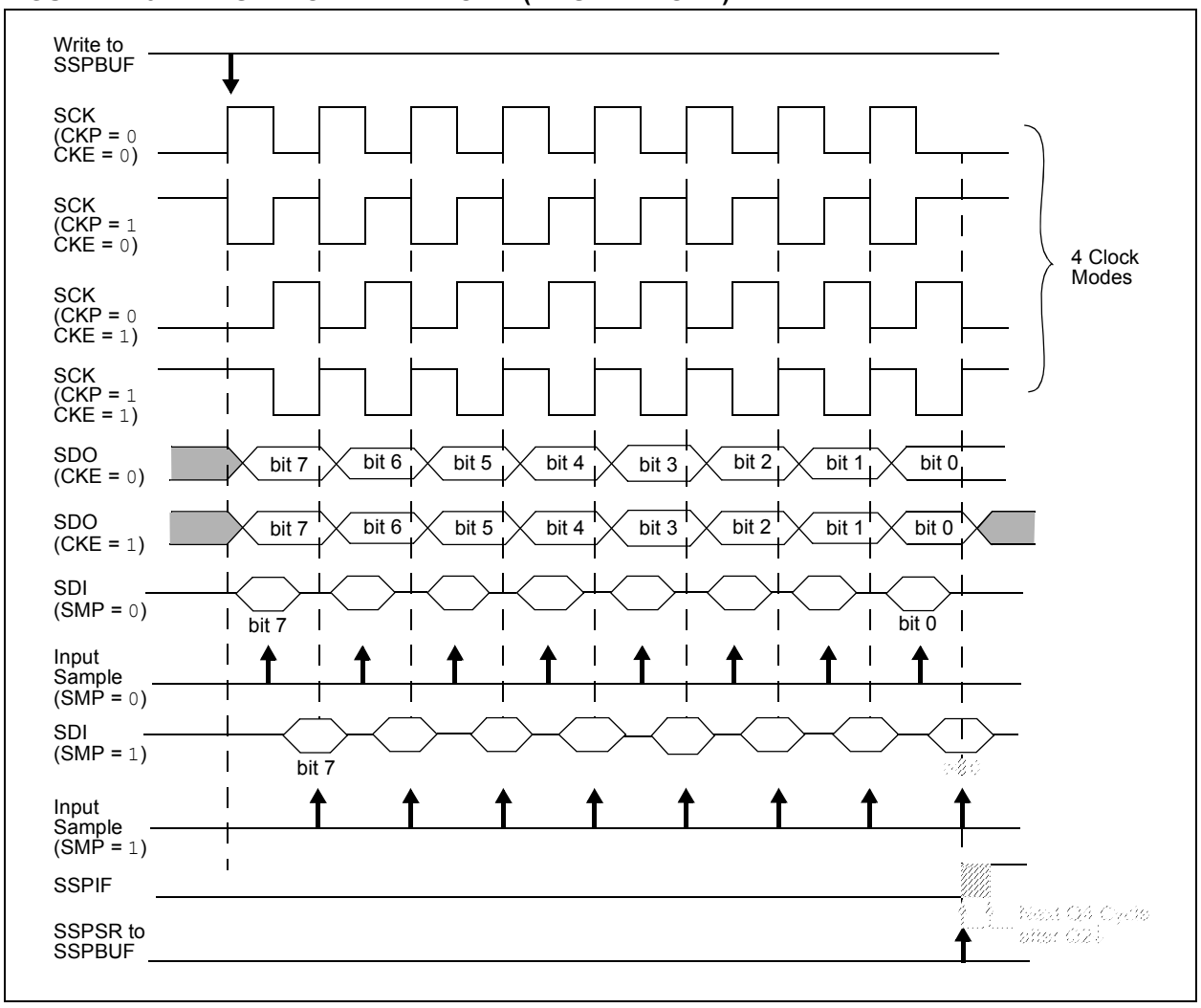
The clock polarity is selected by appropriately programming the CKP bit (SSPCON1<4>). This then, would give waveforms for SPI communication as shown in Figure 17-3, Figure 17-5 and Figure 17-6, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user-programmable to be one of the following:

- FOSC/4 (or TCY)
- FOSC/16 (or 4 • TCY)
- FOSC/64 (or 16 • TCY)
- Timer2 output/2

This allows a maximum data rate (at 40 MHz) of 10.00 Mbps.

Figure 17-3 shows the waveforms for Master mode. When the CKE bit is set, the SDO data is valid before there is a clock edge on SCK. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPBUF is loaded with the received data is shown.

**FIGURE 17-3: SPI MODE WAVEFORM (MASTER MODE)**



**REGISTER 18-3: BAUDCON: BAUD RATE CONTROL REGISTER**

| R/W-0 | R-1 | U-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN |
| 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 **ABDOVF**: Auto-Baud Acquisition Rollover Status bit
1 = A BRG rollover has occurred during Auto-Baud Rate Detect mode (must be cleared in software)
0 = No BRG rollover has occurred

bit 6 **RCIDL**: Receive Operation Idle Status bit
1 = Receive operation is Idle
0 = Receive operation is active

bit 5 **Unimplemented:** Read as '0'

bit 4 **SCKP**: Synchronous Clock Polarity Select bit
Asynchronous mode:
Unused in this mode.
Synchronous mode:
1 = Idle state for clock (CK) is a high level
0 = Idle state for clock (CK) is a low level

bit 3 **BRG16:** 16-Bit Baud Rate Register Enable bit
1 = 16-bit Baud Rate Generator – SPBRGH and SPBRG
0 = 8-bit Baud Rate Generator – SPBRG only (Compatible mode), SPBRGH value ignored

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

bit 1 **WUE:** Wake-up Enable bit
Asynchronous mode:
1 = EUSART will continue to sample the RX pin – interrupt generated on falling edge; bit cleared in hardware on following rising edge
0 = RX pin not monitored or rising edge detected
Synchronous mode:
Unused in this mode.

bit 0 **ABDEN**: Auto-Baud Detect Enable bit
Asynchronous mode:
1 = Enable baud rate measurement on the next character. Requires reception of a Sync field (55h); cleared in hardware upon completion.
0 = Baud rate measurement disabled or completed
Synchronous mode:
Unused in this mode.

### 18.2.5 BREAK CHARACTER SEQUENCE

The Enhanced EUSART module has the capability of sending the special Break character sequences that are required by the LIN bus standard. The Break character transmit consists of a Start bit, followed by twelve '0' bits and a Stop bit. The frame Break character is sent whenever the SENDB and TXEN bits (TXSTA<3> and TXSTA<5>) are set while the Transmit Shift register is loaded with data. Note that the value of data written to TXREG will be ignored and all '0's will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN specification).

Note that the data value written to the TXREG for the Break character is ignored. The write simply serves the purpose of initiating the proper sequence.

The TRMT bit indicates when the transmit operation is active or Idle, just as it does during normal transmission. See Figure 18-10 for the timing of the Break character sequence.

#### 18.2.5.1 Break and Sync Transmit Sequence

The following sequence will send a message frame header made up of a Break, followed by an Auto-Baud Sync byte. This sequence is typical of a LIN bus master.

1. Configure the EUSART for the desired mode.
2. Set the TXEN and SENDB bits to set up the Break character.
3. Load the TXREG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREG to load the Sync character into the transmit FIFO buffer.
5. After the Break has been sent, the SENDB bit is reset by hardware. The Sync character now transmits in the preconfigured mode.

When the TXREG becomes empty, as indicated by the TXIF, the next data byte can be written to TXREG.

### 18.2.6 RECEIVING A BREAK CHARACTER

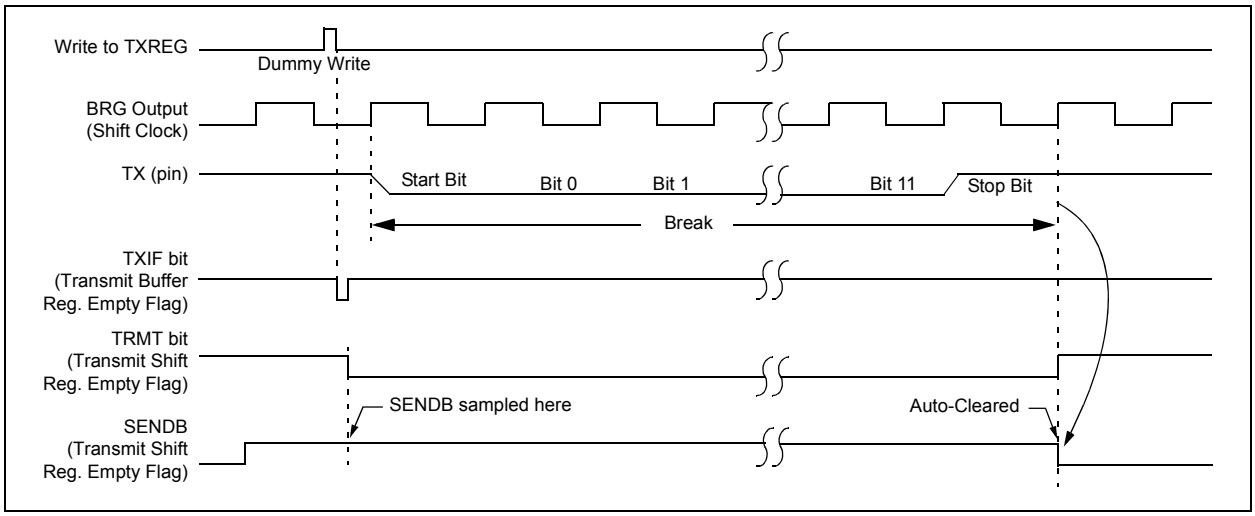
The Enhanced USART module can receive a Break character in two ways.

The first method forces configuration of the baud rate at a frequency of 9/13 the typical speed. This allows for the Stop bit transition to be at the correct sampling location (13 bits for Break versus Start bit and 8 data bits for typical data).

The second method uses the auto-wake-up feature described in **Section 18.2.4 "Auto-Wake-up on Sync Break Character"**. By enabling this feature, the EUSART will sample the next two transitions on RX/DT, cause an RCIF interrupt and receive the next data byte followed by another interrupt.

Note that following a Break character, the user will typically want to enable the Auto-Baud Rate Detect feature. For both methods, the user can set the ABD bit once the TXIF interrupt is observed.

**FIGURE 18-10: SEND BREAK CHARACTER SEQUENCE**



## 22.0 HIGH/LOW-VOLTAGE DETECT (HLVD)

PIC18F2682/2685/4682/4685 devices have a High/Low-Voltage Detect module (HLVD). This is a programmable circuit that allows the user to specify both a device voltage trip point and the direction of change from that point. If the device experiences an excursion past the trip point in that direction, an interrupt flag is set. If the interrupt is enabled, the program execution will branch to the interrupt vector address and the software can then respond to the interrupt.

The High/Low-Voltage Detect Control register (Register 22-1) completely controls the operation of the HLVD module. This allows the circuitry to be "turned off" by the user under software control, which minimizes the current consumption for the device.

The block diagram for the HLVD module is shown in Figure 22-1.

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

| R/W-0 | U-0 | R-0 | R/W-0 | R/W-0 | R/W-1 | R/W-0 | R/W-1 |
|---|---|---|---|---|---|---|---|
| VDIRMAG | — | IRVST | HLVDEN | HLVDL3(1) | HLVDL2(1) | HLVDL1(1) | HLVDL0(1) |
| 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 **VDIRMAG:** Voltage Direction Magnitude Select bit
1 = Event occurs when voltage equals or exceeds trip point (HLVDL3:HLDVL0)
0 = Event occurs when voltage equals or falls below trip point (HLVDL3:HLVDL0)

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

bit 5 **IRVST:** Internal Reference Voltage Stable Flag bit
1 = Indicates that the voltage detect logic will generate the interrupt flag at the specified voltage range
0 = Indicates that the voltage detect logic will not generate the interrupt flag at the specified voltage range and the HLVD interrupt should not be enabled

bit 4 **HLVDEN:** High/Low-Voltage Detect Power Enable bit
1 = HLVD enabled
0 = HLVD disabled

bit 3-0 **HLVDL3:HLVDL0:** High/Low-Voltage Detection Limit bits(1)
1111 = External analog input is used (input comes from the HLVDIN pin)
1110 = 4.48V-4.69V
1101 = 4.23V-4.43V
1100 = 4.01V-4.20V
1011 = 3.81V-3.99V
1010 = 3.63V-3.80V
1001 = 3.46V-3.63V
1000 = 3.31V-3.47V
0111 = 3.05V-3.19V
0110 = 2.82V-2.95V
0101 = 2.72V-2.85V
0100 = 2.54V-2.66V
0011 = 2.38V-2.49V
0010 = 2.31V-2.42V
0001 = 2.18V-2.28V
0000 = 2.12V-2.22V

**Note 1:** HLVDL3:HLVDL0 modes that result in a trip point below the valid operating voltage of the device are not tested.

The module is enabled by setting the HLVDEN bit. Each time that the HLVD module is enabled, the circuitry requires some time to stabilize. The IRVST bit is a read-only bit and is used to indicate when the circuit is stable. The module can only generate an interrupt after the circuit is stable and IRVST is set.

The VDIRMAG bit determines the overall operation of the module. When VDIRMAG is cleared, the module monitors for drops in VDD below a predetermined set point. When the bit is set, the module monitors for rises in VDD above the set point.

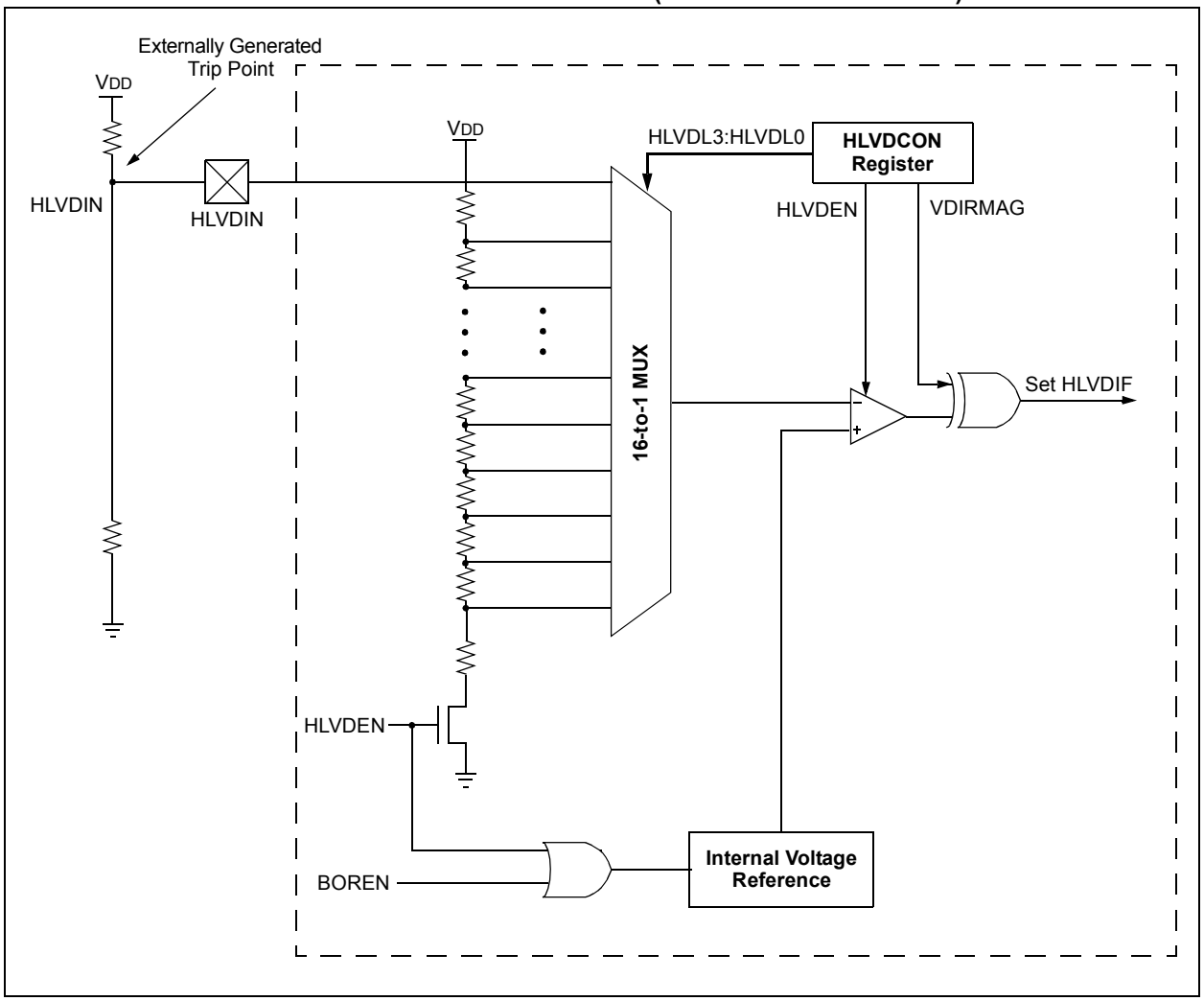
## 22.1 Operation

When the HLVD module is enabled, a comparator uses an internally generated reference voltage as the set point. The set point is compared with the trip point, where each node in the resistor divider represents a trip point voltage. The “trip point” voltage is the voltage level at which the device detects a high or low-voltage event, depending on the configuration of the module. When the supply voltage is equal to the trip point, the voltage tapped off of the resistor array is equal to the internal reference voltage generated by the voltage reference module. The comparator then generates an interrupt signal by setting the HLVDIF bit.

The trip point voltage is software programmable to any one of sixteen values. The trip point is selected by programming the HLVDL3:HLVDL0 bits (HLVDCON<3:0>).

The HLVD module has an additional feature that allows the user to supply the trip voltage to the module from an external source. This mode is enabled when bits HLVDL3:HLVDL0 are set to ‘1111’. In this state, the comparator input is multiplexed from the external input pin, HLVDIN. This gives users flexibility because it allows them to configure the High/Low-Voltage Detect interrupt to occur at any voltage in the valid operating range.

**FIGURE 22-1: HLVD MODULE BLOCK DIAGRAM (WITH EXTERNAL INPUT)**



# 23.0 ECAN™ TECHNOLOGY

PIC18F2682/2685/4682/4685 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.

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 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

## 23.1 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
- Interframe Space Generation/Detection

The CAN module uses the RB2/CANTX and RB3/CANRX pins to interface with the CAN bus. In normal mode, the CAN module automatically overrides TRISB<2>. The user must ensure that TRISB<3> is set.

### 23.1.1 MODULE FUNCTIONALITY

The CAN bus module consists of a protocol engine, message buffering and control (see Figure 23-1). 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. Ensure that the ECAN module is in Configuration mode.
2. Select ECAN operational mode.
3. Set up the baud rate registers.
4. Set up the filter and mask registers.
5. Set the ECAN module to normal mode or any other mode required by the application logic.

**REGISTER 23-1: CANCON: CAN CONTROL REGISTER**

| **Mode 0** | R/W-1 | R/W-0 | R/W-0 | R/S-0 | R/W-0 | R/W-0 | R/W-0 | U-0 |
|---|---|---|---|---|---|---|---|---|
| | REQOP2 | REQOP1 | REQOP0 | ABAT | WIN2 | WIN1 | WIN0 | — |

| **Mode 1** | R/W-1 | R/W-0 | R/W-0 | R/S-0 | U0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|---|
| | REQOP2 | REQOP1 | REQOP0 | ABAT | — | — | — | — |

| **Mode 2** | R/W-1 | R/W-0 | R/W-0 | R/S-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|---|
| | REQOP2 | REQOP1 | REQOP0 | ABAT | FP3 | FP2 | FP1 | FP0 |
| | bit 7 | | | | | | | bit 0 |

| **Legend:** | S = Settable bit | | |
|---|---|---|---|
| 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 **REQOP2:REQOP0:** Request CAN Operation Mode bits

1xx = Request Configuration mode
011 = Request Listen Only mode
010 = Request Loopback mode
001 = Request Disable mode
000 = Request Normal mode

bit 4 **ABAT:** Abort All Pending Transmissions bit

1 = Abort all pending transmissions (in all transmit buffers)
0 = Transmissions proceeding as normal

bit 3-1 Mode 0:

**WIN2:WIN0:** Window Address bits

These bits select which of the CAN buffers to switch into the access bank area. This allows access to the buffer registers from any data memory bank. After a frame has caused an interrupt, the ICODE3:ICODE0 bits can be copied to the WIN3:WIN0 bits to select the correct buffer. See Example 23-2 for a code example.

111 = Receive Buffer 0
110 = Receive Buffer 0
101 = Receive Buffer 1
100 = Transmit Buffer 0
011 = Transmit Buffer 1
010 = Transmit Buffer 2
001 = Receive Buffer 0
000 = Receive Buffer 0

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

bit 4-0 Mode 1:

**Unimplemented:** Read as '0'

Mode 2:

**FP3:FP0:** FIFO Read Pointer bits

These bits point to the message buffer to be read.
0111:0000 = Message buffer to be read
1111:1000 = Reserved

**REGISTER 23-4: COMSTAT: COMMUNICATION STATUS REGISTER**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Mode 0** | R/C-0 | R/C-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| | RXB0OVFL | RXB1OVFL | TXBO | TXBP | RXBP | TXWARN | RXWARN | EWARN |
| **Mode 1** | R/C-0 | R/C-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| | — | RXBnOVFL | TXB0 | TXBP | RXBP | TXWARN | RXWARN | EWARN |
| **Mode 2** | R/C-0 | R/C-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| | $\overline{\text{FIFOEMPTY}}$ | RXBnOVFL | TXBO | TXBP | RXBP | TXWARN | RXWARN | EWARN |
| | bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | C = Clearable 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 Mode 0:
**RXB0OVFL:** Receive Buffer 0 Overflow bit
1 = Receive Buffer 0 overflowed
0 = Receive Buffer 0 has not overflowed
Mode 1:
**Unimplemented:** Read as ‘0’
Mode 2:
**$\overline{\text{FIFOEMPTY}}$:** FIFO Not Empty bit
1 = Receive FIFO is not empty
0 = Receive FIFO is empty

bit 6 Mode 0:
**RXB1OVFL:** Receive Buffer 1 Overflow bit
1 = Receive Buffer 1 overflowed
0 = Receive Buffer 1 has not overflowed
Mode 1, 2:
**RXBnOVFL:** Receive Buffer n Overflow bit
1 = Receive Buffer n has overflowed
0 = Receive Buffer n has not overflowed

bit 5 **TXBO:** Transmitter Bus-Off bit
1 = Transmit error counter > 255
0 = Transmit error counter ≤ 255

bit 4 **TXBP:** Transmitter Bus Passive bit
1 = Transmit error counter > 127
0 = Transmit error counter ≤ 127

bit 3 **RXBP:** Receiver Bus Passive bit
1 = Receive error counter > 127
0 = Receive error counter ≤ 127

bit 2 **TXWARN:** Transmitter Warning bit
1 = Transmit error counter > 95
0 = Transmit error counter ≤ 95

bit 1 **RXWARN:** Receiver Warning bit
1 = 127 ≥ Receive error counter > 95
0 = Receive error counter ≤ 95

bit 0 **EWARN:** Error Warning bit
This bit is a flag of the RXWARN and TXWARN bits.
1 = The RXWARN or the TXWARN bits are set
0 = Neither the RXWARN or the TXWARN bits are set

**ANDWF** **AND W with f**

Syntax: ANDWF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) .AND. (f) → dest

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0001 | 01da | ffff | ffff |

Description: The contents of W are ANDed with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `ANDWF REG, 0, 0`

Before Instruction
W = 17h
REG = C2h
After Instruction
W = 02h
REG = C2h

**BC** **Branch if Carry**

Syntax: BC n

Operands: $-128 \le n \le 127$

Operation: if Carry bit is '1'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0010 | nnnn | nnnn |

Description: If the Carry bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example:
```
HERE    BC   5
```
Before Instruction
PC = address (HERE)
After Instruction
If Carry = 1;
PC = address (HERE + 12)
If Carry = 0;
PC = address (HERE + 2)

**TBLRD** **Table Read**

Syntax: TBLRD ( *; *+; *-; +*)

Operands: None

Operation: if TBLRD *,
(Prog Mem (TBLPTR)) → TABLAT,
TBLPTR – No Change;
if TBLRD *+,
(Prog Mem (TBLPTR)) → TABLAT,
(TBLPTR) + 1 → TBLPTR;
if TBLRD *-,
(Prog Mem (TBLPTR)) → TABLAT,
(TBLPTR) – 1 → TBLPTR;
if TBLRD +*,
(TBLPTR) + 1 → TBLPTR,
(Prog Mem (TBLPTR)) → TABLAT

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 10nn<br>nn=0 *<br>=1 *+<br>=2 *–<br>=3 +* |

Description: This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer, called Table Pointer (TBLPTR), is used.

The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word
TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLRD instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |
| No operation | No operation (Read Program Memory) | No operation | No operation (Write TABLAT) |

**TBLRD** **Table Read (Continued)**

Example 1:
```
TBLRD  *+ ;
```
Before Instruction
TABLAT = 55h
TBLPTR = 00A356h
MEMORY(00A356h) = 34h
After Instruction
TABLAT = 34h
TBLPTR = 00A357h

Example 2:
```
TBLRD  +* ;
```
Before Instruction
TABLAT = 0AAh
TBLPTR = 01A357h
MEMORY(01A357h) = 12h
MEMORY(01A358h) = 34h
After Instruction
TABLAT = 34h
TBLPTR = 01A358h

**SUBFSR** **Subtract Literal from FSR**

Syntax: SUBFSR f, k

Operands: $0 \le k \le 63$
$f \in [0, 1, 2]$

Operation: FSRf – k → FSRf

Status Affected: None

Encoding:

| 1110 | 1001 | ffkk | kkkk |
|---|---|---|---|

Description: The 6-bit literal 'k' is subtracted from the contents of the FSR specified by 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: SUBFSR 2, 23h

Before Instruction
FSR2 = 03FFh

After Instruction
FSR2 = 03DCh

**SUBULNK** **Subtract Literal from FSR2 and Return**

Syntax: SUBULNK k

Operands: $0 \le k \le 63$

Operation: FSR2 – k → FSR2,
(TOS) → PC

Status Affected: None

Encoding:

| 1110 | 1001 | 11kk | kkkk |
|---|---|---|---|

Description: The 6-bit literal 'k' is subtracted from the contents of the FSR2. A RETURN is then executed by loading the PC with the TOS.

The instruction takes two cycles to execute; a NOP is performed during the second cycle.

This may be thought of as a special case of the SUBFSR instruction, where f = 3 (binary '11'); it operates only on FSR2.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |
| No Operation | No Operation | No Operation | No Operation |

Example: SUBULNK 23h

Before Instruction
FSR2 = 03FFh
PC = 0100h

After Instruction
FSR2 = 03DCh
PC = (TOS)

### 25.2.5 SPECIAL CONSIDERATIONS WITH MICROCHIP MPLAB® IDE TOOLS

The latest versions of Microchip's software tools have been designed to fully support the extended instruction set of the PIC18F2682/2685/4682/4685 family of devices. This includes the MPLAB C18 C compiler, MPASM assembly language and MPLAB Integrated Development Environment (IDE).

When selecting a target device for software development, MPLAB IDE will automatically set default Configuration bits for that device. The default setting for the XINST Configuration bit is '0', disabling the extended instruction set and Indexed Literal Offset Addressing mode. For proper execution of applications developed to take advantage of the extended instruction set, XINST must be set during programming.

To develop software for the extended instruction set, the user must enable support for the instructions and the Indexed Addressing mode in their language tool(s). Depending on the environment being used, this may be done in several ways:

- A menu option, or dialog box within the environment, that allows the user to configure the language tool and its settings for the project
- A command line option
- A directive in the source code

These options vary between different compilers, assemblers and development environments. Users are encouraged to review the documentation accompanying their development systems for the appropriate information.

## 26.11 PICkit 2 Development Programmer/Debugger and PICkit 2 Debug Express

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

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

## 26.12 MPLAB PM3 Device Programmer

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

## 26.13 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

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

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

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

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

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

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

27.4.3 TIMING DIAGRAMS AND SPECIFICATIONS

**FIGURE 27-5: EXTERNAL CLOCK TIMING (ALL MODES EXCEPT PLL)**

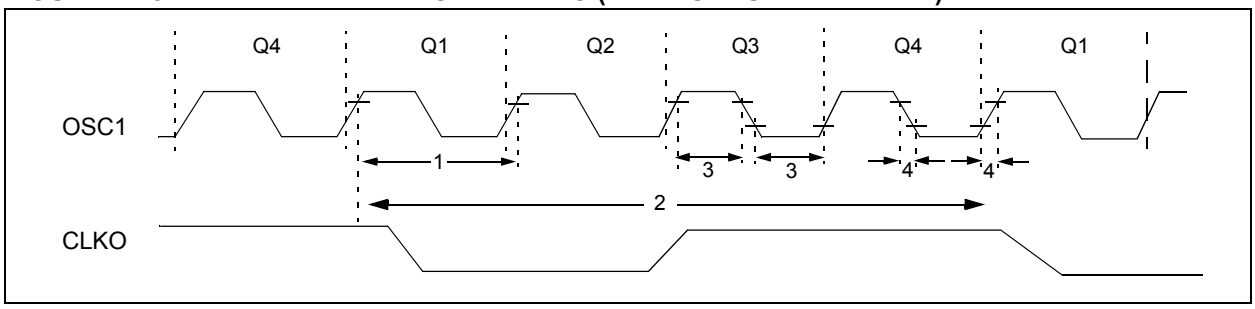



**TABLE 27-6: EXTERNAL CLOCK TIMING REQUIREMENTS**

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|
| 1A | FOSC | External CLKI Frequency(1) | DC | 1 | MHz | XT, RC Oscillator modes |
| | | | DC | 25 | MHz | HS Oscillator mode |
| | | | DC | 31.25 | kHz | LP Oscillator mode |
| | | | DC | 40 | MHz | EC Oscillator mode |
| | | Oscillator Frequency(1) | DC | 4 | MHz | RC Oscillator mode |
| | | | 0.1 | 4 | MHz | XT Oscillator mode |
| | | | 4 | 25 | MHz | HS Oscillator mode |
| | | | 4 | 10 | MHz | HSPLL Oscillator mode |
| | | | 5 | 200 | kHz | LP Oscillator mode |
| 1 | TOSC | External CLKI Period(1) | 1000 | — | ns | XT, RC Oscillator modes |
| | | | 40 | — | ns | HS Oscillator mode |
| | | | 32 | — | μs | LP Oscillator mode |
| | | | 25 | — | ns | EC Oscillator mode |
| | | Oscillator Period(1) | 250 | — | ns | RC Oscillator mode |
| | | | 250 | 1 | μs | XT Oscillator mode |
| | | | 40 | 250 | ns | HS Oscillator mode |
| | | | 100 | 250 | ns | HSPLL Oscillator mode |
| | | | 5 | 200 | μs | LP Oscillator mode |
| 2 | TCY | Instruction Cycle Time(1) | 100 | — | ns | TCY = 4/FOSC |
| 3 | TOSL, TOSH | External Clock in (OSC1) High or Low Time | 30 | — | ns | XT Oscillator mode |
| | | | 2.5 | — | μs | LP Oscillator mode |
| | | | 10 | — | ns | HS Oscillator mode |
| 4 | TOSR, TOSF | External Clock in (OSC1) Rise or Fall Time | — | 20 | ns | XT Oscillator mode |
| | | | — | 50 | ns | LP Oscillator mode |
| | | | — | 7.5 | ns | HS Oscillator mode |

**Note 1:** Instruction cycle period (TCY) equals four times the input oscillator time base period for all configurations except PLL. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at “min.” values with an external clock applied to the OSC1/CLKI pin. When an external clock input is used, the “max.” cycle time limit is “DC” (no clock) for all devices.

## APPENDIX A: REVISION HISTORY

### Revision A (February 2006)

Original data sheet for PIC18F2682/2685/4682/4685 devices.

### Revision B (January 2007)

Major edits to **Section 27.0 "Electrical Characteristics"**. Packaging diagrams have been updated and minor edits to text have been made throughout document.

### Revision C (October 2009)

Updated to remove Preliminary status.

## APPENDIX B: DEVICE DIFFERENCES

The differences between the devices listed in this data sheet are shown in Table B-1.

**TABLE B-1: DEVICE DIFFERENCES**

| Features | PIC18F2682 | PIC18F2685 | PIC18F4682 | PIC18F4685 |
|---|---|---|---|---|
| Program Memory (Bytes) | 80K | 96K | 80K | 96K |
| Program Memory (Instructions) | 40960 | 49152 | 40960 | 49152 |
| Interrupt Sources | 27 | 27 | 28 | 28 |
| I/O Ports | Ports A, B, C, (E) | Ports A, B, C, (E) | Ports A, B, C, D, E | Ports A, B, C, D, E |
| Capture/Compare/PWM Modules | 1 | 1 | 1 | 1 |
| Enhanced Capture/Compare/ PWM Modules | 0 | 0 | 1 | 1 |
| Parallel Slave Port Communications (PSP) | No | No | Yes | Yes |
| 10-Bit Analog-to-Digital Module | 8 input channels | 8 input channels | 11 input channels | 11 input channels |
| Packages | 28-pin PDIP<br>28-pin SOIC | 28-pin PDIP<br>28-pin SOIC | 40-pin PDIP<br>44-pin TQFP<br>44-pin QFN | 40-pin PDIP<br>44-pin TQFP<br>44-pin QFN |