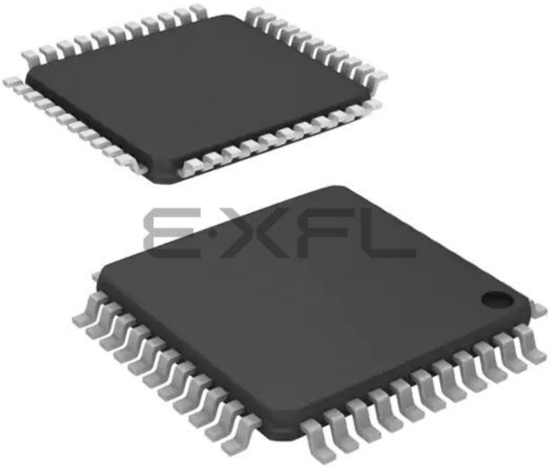
# Microchip Technology - DSPIC33EV64GM004-E/PT Datasheet



## What is "Embedded - Microcontrollers"?

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

## Applications of "Embedded - Microcontrollers"

| **Details** | |
|---|---|
| Product Status | Active |
| Core Processor | dsPIC |
| Core Size | 16-Bit |
| Speed | 60 MIPs |
| Connectivity | I²C, IrDA, LINbus, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, DMA, Motor Control PWM, POR, PWM, WDT |
| Number of I/O | 35 |
| Program Memory Size | 64KB (22K x 24) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 8K x 8 |
| Voltage - Supply (Vcc/Vdd) | 4.5V ~ 5.5V |
| Data Converters | A/D 24x10/12b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 125°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 44-TQFP |
| Supplier Device Package | 44-TQFP (10x10) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/dspic33ev64gm004-e-pt |

## TABLE 4-26: DMAC REGISTER MAP (CONTINUED)

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | All Resets |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMALCA | 0BF6 | — | — | — | — | — | — | — | — | — | — | — | — | LSTCH<3:0> | | | | 000F |
| DSADRL | 0BF8 | DSADR<15:0> | | | | | | | | | | | | | | | | 0000 |
| DSADRH | 0BFA | — | — | — | — | — | — | — | — | DSADR<23:16> | | | | | | | | 0000 |

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

## TABLE 4-27: PWM REGISTER MAP

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | All Resets |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PTCON | 0C00 | PTEN | — | PTSIDL | SESTAT | SEIEN | EIPU | SYNCPOL | SYNCOEN | SYNCEN | SYNCSRC2 | SYNCSRC1 | SYNCSRC0 | SEVTPS3 | SEVTPS2 | SEVTPS1 | SEVTPS0 | 0000 |
| PTCON2 | 0C02 | — | — | — | — | — | — | — | — | — | — | — | — | — | PCLKDIV<2:0> | | | 0000 |
| PTPER | 0C04 | PTPER<15:0> | | | | | | | | | | | | | | | | FFF8 |
| SEVTCMP | 0C06 | SEVTCMP<15:0> | | | | | | | | | | | | | | | | 0000 |
| MDC | 0C0A | MDC<15:0> | | | | | | | | | | | | | | | | 0000 |
| CHOP | 0C1A | CHPCLKEN | — | — | — | — | — | CHOPCLK9 | CHOPCLK8 | CHOPCLK7 | CHOPCLK6 | CHOPCLK5 | CHOPCLK4 | CHOPCLK3 | CHOPCLK2 | CHOPCLK1 | CHOPCLK0 | 0000 |
| PWMKEY | 0C1E | PWMKEY<15:0> | | | | | | | | | | | | | | | | 0000 |

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

## TABLE 4-28: PWM GENERATOR 1 REGISTER MAP

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | All Resets |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PWMCON1 | 0C20 | FLTSTAT | CLSTAT | TRGSTAT | FLTIEN | CLIEN | TRGIEN | ITB | MDCS | DTC1 | DTC0 | DTCP | — | — | CAM | XPRES | IUE | 0000 |
| IOCON1 | 0C22 | PENH | PENL | POLH | POLL | PMOD1 | PMOD0 | OVRENH | OVRENL | OVRDAT1 | OVRDAT0 | FLTDAT1 | FLTDAT0 | CLDAT1 | CLDAT0 | SWAP | OSYNC | 0000 |
| FCLCON1 | 0C24 | — | CLSRC4 | CLSRC3 | CLSRC2 | CLSRC1 | CLSRC0 | CLPOL | CLMOD | FLTSRC4 | FLTSRC3 | FLTSRC2 | FLTSRC1 | FLTSRC0 | FLTPOL | FLTMOD1 | FLTMOD0 | 0000 |
| PDC1 | 0C26 | PDC1<15:0> | | | | | | | | | | | | | | | | 0000 |
| PHASE1 | 0C28 | PHASE1<15:0> | | | | | | | | | | | | | | | | 0000 |
| DTR1 | 0C2A | — | — | DTR1<13:0> | | | | | | | | | | | | | | 0000 |
| ALTDTR1 | 0C2C | — | — | ALTDTR1<13:0> | | | | | | | | | | | | | | 0000 |
| TRIG1 | 0C32 | TRGCMP<15:0> | | | | | | | | | | | | | | | | 0000 |
| TRGCON1 | 0C34 | TRGDIV3 | TRGDIV2 | TRGDIV1 | TRGDIV0 | — | — | — | — | — | — | TRGSTRT5 | TRGSTRT4 | TRGSTRT3 | TRGSTRT2 | TRGSTRT1 | TRGSTRT0 | 0000 |
| PWMCAP1 | 0C38 | PWMCAP1<15:0> | | | | | | | | | | | | | | | | 0000 |
| LEBCON1 | 0C3A | PHR | PHF | PLR | PLF | FLTLEBEN | CLLEBEN | — | — | — | — | BCH | BCL | BPHH | BPHL | BPLH | BPLL | 0000 |
| LEBDLY1 | 0C3C | — | — | — | — | LEB<11:0> | | | | | | | | | | | | 0000 |
| AUXCON1 | 0C3E | — | — | — | — | BLANKSEL3 | BLANKSEL2 | BLANKSEL1 | BLANKSEL0 | — | — | CHOPSEL3 | CHOPSEL2 | CHOPSEL1 | CHOPSEL0 | CHOPHEN | CHOPLEN | 0000 |

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

### TABLE 4-31: PORTA REGISTER MAP FOR dsPIC33EVXXXGMX06 DEVICES

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | All Resets |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TRISA | 0E00 | — | — | — | TRISA<12:7> | | | | | | — | — | TRISA4 | — | — | TRISA<1:0> | | 1F93 |
| PORTA | 0E02 | — | — | — | RA<12:7> | | | | | | — | — | RA4 | — | — | RA<1:0> | | 0000 |
| LATA | 0E04 | — | — | — | LATA<12:7> | | | | | | — | — | LATA4 | — | — | LATA<1:0> | | 0000 |
| ODCA | 0E06 | — | — | — | ODCA<12:7> | | | | | | — | — | ODCA4 | — | — | ODCA<1:0> | | 0000 |
| CNENA | 0E08 | — | — | — | CNIEA<12:7> | | | | | | — | — | CNIEA4 | — | — | CNIEA<1:0> | | 0000 |
| CNPUA | 0E0A | — | — | — | CNPUA<12:7> | | | | | | — | — | CNPUA4 | — | — | CNPUA<1:0> | | 0000 |
| CNPDA | 0E0C | — | — | — | CNPDA<12:7> | | | | | | — | — | CNPDA4 | — | — | CNPDA<1:0> | | 0000 |
| ANSELA | 0E0E | — | — | — | ANSA<12:9> | | | | — | ANSA7 | — | — | ANSA4 | — | — | ANSA<1:0> | | 1E93 |
| SR1A | 0E10 | — | — | — | — | — | — | SR1A9 | — | — | — | — | SR1A4 | — | — | — | — | 0000 |
| SR0A | 0E12 | — | — | — | — | — | — | SR0A9 | — | — | — | — | SR0A4 | — | — | — | — | 0000 |

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

### TABLE 4-32: PORTA REGISTER MAP FOR dsPIC33EVXXXGMX04 DEVICES

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | All Resets |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TRISA | 0E00 | — | — | — | — | — | TRISA<10:7> | | | | — | — | TRISA<4:0> | | | | | DF9F |
| PORTA | 0E02 | — | — | — | — | — | RA<10:7> | | | | — | — | RA<4:0> | | | | | 0000 |
| LATA | 0E04 | — | — | — | — | — | LATA<10:7> | | | | — | — | LATA<4:0> | | | | | 0000 |
| ODCA | 0E06 | — | — | — | — | — | ODCA<10:7> | | | | — | — | ODCA<4:0> | | | | | 0000 |
| CNENA | 0E08 | — | — | — | — | — | CNIEA<10:7> | | | | — | — | CNIEA<4:0> | | | | | 0000 |
| CNPUA | 0E0A | — | — | — | — | — | CNPUA<10:7> | | | | — | — | CNPUA<4:0> | | | | | 0000 |
| CNPDA | 0E0C | — | — | — | — | — | CNPDA<10:7> | | | | — | — | CNPDA<4:0> | | | | | 0000 |
| ANSELA | 0E0E | — | — | — | — | — | ANSA<10:9> | | — | ANSA7 | — | — | ANSA4 | — | ANSA<2:0> | | | 1813 |
| SR1A | 0E10 | — | — | — | — | — | — | SR1A9 | — | — | — | — | SR1A4 | — | — | — | — | 0000 |
| SR0A | 0E12 | — | — | — | — | — | — | SR0A9 | — | — | — | — | SR0A4 | — | — | — | — | 0000 |

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

Allocating different Page registers for read and write access allows the architecture to support data movement between different pages in the data memory. This is accomplished by setting the DSRPAG register value to the page from which you want to read, and configure the DSWPAG register to the page to which it needs to be written. Data can also be moved from different PSV to EDS pages by configuring the DSRPAG and DSWPAG registers to address PSV and EDS space, respectively. The data can be moved between pages by a single instruction.

When an EDS or PSV page overflow or underflow occurs, EA<15> is cleared as a result of the register indirect EA calculation. An overflow or underflow of the EA in the EDS or PSV pages can occur at the page boundaries when:

- The initial address, prior to modification, addresses an EDS or a PSV page.
- The EA calculation uses Pre- or Post-Modified Register Indirect Addressing. However, this does not include Register Offset Addressing.

In general, when an overflow is detected, the DSxPAG register is incremented and the EA<15> bit is set to keep the base address within the EDS or PSV window. When an underflow is detected, the DSxPAG register is decremented and the EA<15> bit is set to keep the base address within the EDS or PSV window. This creates a linear EDS and PSV address space, but only when using the Register Indirect Addressing modes.

Exceptions to the operation described above arise when entering and exiting the boundaries of Page 0, EDS and PSV spaces. Table 4-43 lists the effects of overflow and underflow scenarios at different boundaries.

In the following cases, when an overflow or underflow occurs, the EA<15> bit is set and the DSxPAG is not modified; therefore, the EA will wrap to the beginning of the current page:

- Register Indirect with Register Offset Addressing
- Modulo Addressing
- Bit-Reversed Addressing

**TABLE 4-43: OVERFLOW AND UNDERFLOW SCENARIOS AT PAGE 0, EDS AND PSV SPACE BOUNDARIES(2,3,4)**

| O/U, R/W | Operation | Before | | | After | | |
|---|---|---|---|---|---|---|---|
| | | DSxPAG | DS EA<15> | Page Description | DSxPAG | DS EA<15> | Page Description |
| O, Read | [++Wn]<br>or<br>[Wn++] | DSRPAG = 0x1FF | 1 | EDS: Last Page | DSRPAG = 0x1FF | 0 | See **Note 1** |
| O, Read | | DSRPAG = 0x2FF | 1 | PSV: Last lsw Page | DSRPAG = 0x300 | 1 | PSV: First MSB Page |
| O, Read | | DSRPAG = 0x3FF | 1 | PSV: Last MSB Page | DSRPAG = 0x3FF | 0 | See **Note 1** |
| O, Write | | DSWPAG = 0x1FF | 1 | EDS: Last Page | DSWPAG = 0x1FF | 0 | See **Note 1** |
| U, Read | [--Wn]<br>or<br>[Wn--] | DSRPAG = 0x001 | 1 | PSV Page | DSRPAG = 0x001 | 0 | See **Note 1** |
| U, Read | | DSRPAG = 0x200 | 1 | PSV: First lsw Page | DSRPAG = 0x200 | 0 | See **Note 1** |
| U, Read | | DSRPAG = 0x300 | 1 | PSV: First MSB Page | DSRPAG = 0x2FF | 1 | PSV: Last lsw Page |

**Legend:** O = Overflow, U = Underflow, R = Read, W = Write

**Note 1:** The Register Indirect Addressing now addresses a location in the Base Data Space (0x0000-0x8000).

**2:** An EDS access with DSxPAG = 0x000 will generate an address error trap.

**3:** Only reads from PS are supported using DSRPAG. An attempt to write to PS using DSWPAG will generate an address error trap.

**4:** Pseudolinear Addressing is not supported for large offsets.

**FIGURE 4-16: BIT-REVERSED ADDRESSING EXAMPLE**

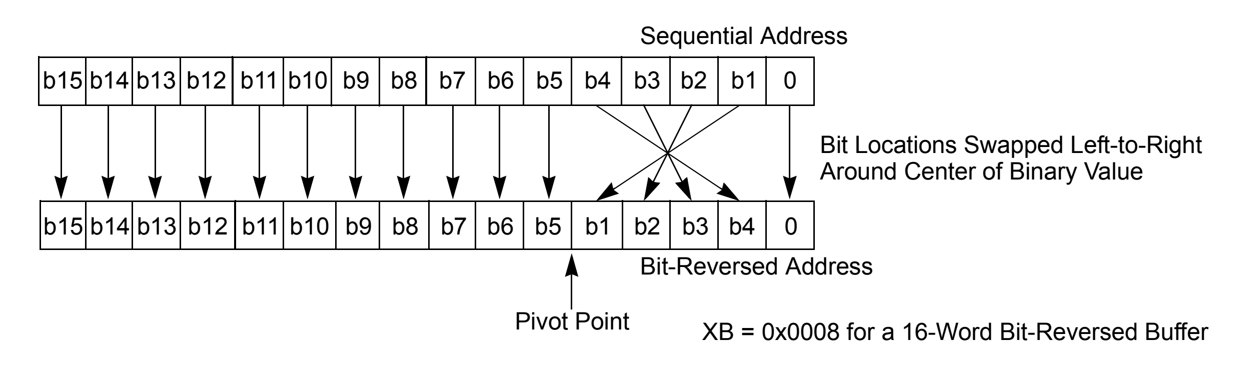

Sequential Address
b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 0
Bit Locations Swapped Left-to-Right
Around Center of Binary Value
b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b1 b2 b3 b4 0
Bit-Reversed Address
Pivot Point
XB = 0x0008 for a 16-Word Bit-Reversed Buffer

**TABLE 4-46: BIT-REVERSED ADDRESSING SEQUENCE (16-ENTRY)**

| Normal Address | | | | | Bit-Reversed Address | | | | |
|---|---|---|---|---|---|---|---|---|---|
| A3 | A2 | A1 | A0 | Decimal | A3 | A2 | A1 | A0 | Decimal |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 8 |
| 0 | 0 | 1 | 0 | 2 | 0 | 1 | 0 | 0 | 4 |
| 0 | 0 | 1 | 1 | 3 | 1 | 1 | 0 | 0 | 12 |
| 0 | 1 | 0 | 0 | 4 | 0 | 0 | 1 | 0 | 2 |
| 0 | 1 | 0 | 1 | 5 | 1 | 0 | 1 | 0 | 10 |
| 0 | 1 | 1 | 0 | 6 | 0 | 1 | 1 | 0 | 6 |
| 0 | 1 | 1 | 1 | 7 | 1 | 1 | 1 | 0 | 14 |
| 1 | 0 | 0 | 0 | 8 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 9 | 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 1 | 0 | 10 | 0 | 1 | 0 | 1 | 5 |
| 1 | 0 | 1 | 1 | 11 | 1 | 1 | 0 | 1 | 13 |
| 1 | 1 | 0 | 0 | 12 | 0 | 0 | 1 | 1 | 3 |
| 1 | 1 | 0 | 1 | 13 | 1 | 0 | 1 | 1 | 11 |
| 1 | 1 | 1 | 0 | 14 | 0 | 1 | 1 | 1 | 7 |
| 1 | 1 | 1 | 1 | 15 | 1 | 1 | 1 | 1 | 15 |

### REGISTER 7-3: INTCON1: INTERRUPT CONTROL REGISTER 1 (CONTINUED)

bit 3 **ADDRERR:** Address Error Trap Status bit

1 = Address error trap has occurred
0 = Address error trap has not occurred

bit 2 **STKERR:** Stack Error Trap Status bit

1 = Stack error trap has occurred
0 = Stack error trap has not occurred

bit 1 **OSCFAIL:** Oscillator Failure Trap Status bit

1 = Oscillator failure trap has occurred
0 = Oscillator failure trap has not occurred

bit 0 **Unimplemented:** Read as ‘0’

### REGISTER 8-11: DMAPWC: DMA PERIPHERAL WRITE COLLISION STATUS REGISTER

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

| U-0 | U-0 | U-0 | U-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | PWCOL3 | PWCOL2 | PWCOL1 | PWCOL0 |
| bit 7 | | | | | | | bit 0 |

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

bit 15-4 **Unimplemented:** Read as ‘0’

bit 3 **PWCOL3:** Channel 3 Peripheral Write Collision Flag bit
1 = Write collision is detected
0 = Write collision is not detected

bit 2 **PWCOL2:** Channel 2 Peripheral Write Collision Flag bit
1 = Write collision is detected
0 = Write collision is not detected

bit 1 **PWCOL1:** Channel 1 Peripheral Write Collision Flag bit
1 = Write collision is detected
0 = Write collision is not detected

bit 0 **PWCOL0:** Channel 0 Peripheral Write Collision Flag bit
1 = Write collision is detected
0 = Write collision is not detected

### REGISTER 17-18: AUXCONx: PWMx AUXILIARY CONTROL REGISTER

| U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | BLANKSEL3 | BLANKSEL2 | BLANKSEL1 | BLANKSEL0 |
| bit 15 | | | | | | | bit 8 |

| U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | CHOPSEL3 | CHOPSEL2 | CHOPSEL1 | CHOPSEL0 | CHOPHEN | CHOPLEN |
| bit 7 | | | | | | | bit 0 |

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

bit 15-12 **Unimplemented:** Read as ‘0’

bit 11-8 **BLANKSEL<3:0>:** PWMx State Blank Source Select bits

The selected state blank signal will block the current-limit and/or Fault input signals (if enabled through the BCH and BCL bits in the LEBCONx register).
1001 = Reserved
•
•
•
0100 = Reserved
0011 = PWM3H is selected as the state blank source
0010 = PWM2H is selected as the state blank source
0001 = PWM1H is selected as the state blank source
0000 = No state blanking

bit 7-6 **Unimplemented:** Read as ‘0’

bit 5-2 **CHOPSEL<3:0>:** PWMx Chop Clock Source Select bits

The selected signal will enable and disable (Chop) the selected PWMx outputs.
1001 = Reserved
•
•
•
0100 = Reserved
0011 = PWM3H is selected as the chop clock source
0010 = PWM2H is selected as the chop clock source
0001 = PWM1H is selected as the chop clock source
0000 = Chop clock generator is selected as the chop clock source

bit 1 **CHOPHEN:** PWMxH Output Chopping Enable bit

1 = PWMxH chopping function is enabled
0 = PWMxH chopping function is disabled

bit 0 **CHOPLEN:** PWMxL Output Chopping Enable bit

1 = PWMxL chopping function is enabled
0 = PWMxL chopping function is disabled

### REGISTER 18-1: SPIxSTAT: SPIx STATUS AND CONTROL REGISTER (CONTINUED)

bit 1 **SPITBF:** SPIx Transmit Buffer Full Status bit

`1` = Transmit has not yet started, the SPIxTXB bit is full
`0` = Transmit has started, the SPIxTXB bit is empty

<u>Standard Buffer mode:</u>
Automatically set in hardware when the core writes to the SPIxBUF location, loading SPIxTXB. Automatically cleared in hardware when the SPIx module transfers data from SPIxTXB to SPIxSR.

<u>Enhanced Buffer mode:</u>
Automatically set in the hardware when the CPU writes to the SPIxBUF location, loading the last available buffer location. Automatically cleared in hardware when a buffer location is available for a CPU write operation.

bit 0 **SPIRBF:** SPIx Receive Buffer Full Status bit

`1` = Receive is complete, the SPIxRXB bit is full
`0` = Receive is incomplete, the SPIxRXB bit is empty

<u>Standard Buffer mode:</u>
Automatically set in the hardware when SPIx transfers data from SPIxSR to SPIxRXB. Automatically cleared in hardware when the core reads the SPIxBUF location, reading SPIxRXB.

<u>Enhanced Buffer mode:</u>
Automatically set in hardware when SPIx transfers data from SPIxSR to the buffer, filling the last unread buffer location. Automatically cleared in hardware when a buffer location is available for a transfer from SPIxSR.

### REGISTER 18-3: SPIxCON2: SPIx CONTROL REGISTER 2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| R/W-0 | R/W-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
| FRMEN | SPIFSD | FRMPOL | — | — | — | — | — |
| bit 15 | | | | | | | bit 8 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 |
| — | — | — | — | — | — | FRMDLY | SPIBEN |
| bit 7 | | | | | | | bit 0 |

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

bit 15 **FRMEN:** Framed SPIx Support bit

1 = Framed SPIx support is enabled ($\overline{\text{SSx}}$ pin is used as the Frame Sync pulse input/output)
0 = Framed SPIx support is disabled

bit 14 **SPIFSD:** SPIx Frame Sync Pulse Direction Control bit

1 = Frame Sync pulse input (slave)
0 = Frame Sync pulse output (master)

bit 13 **FRMPOL:** Frame Sync Pulse Polarity bit

1 = Frame Sync pulse is active-high
0 = Frame Sync pulse is active-low

bit 12-2 **Unimplemented:** Read as ‘0’

bit 1 **FRMDLY:** Frame Sync Pulse Edge Select bit

1 = Frame Sync pulse coincides with the first bit clock
0 = Frame Sync pulse precedes the first bit clock

bit 0 **SPIBEN:** SPIx Enhanced Buffer Enable bit

1 = Enhanced buffer is enabled
0 = Enhanced buffer is disabled (Standard mode)

### REGISTER 22-15: CxBUFPNT4: CANx FILTERS 12-15 BUFFER POINTER REGISTER 4

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| F15BP3 | F15BP2 | F15BP1 | F15BP0 | F14BP3 | F14BP2 | F14BP1 | F14BP0 |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| F13BP3 | F13BP2 | F13BP1 | F13BP0 | F12BP3 | F12BP2 | F12BP1 | F12BP0 |
| bit 7 | | | | | | | bit 0 |

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

bit 15-12 **F15BP<3:0>:** RX Buffer Mask for Filter 15 bits

1111 = Filter hits received in RX FIFO buffer
1110 = Filter hits received in RX Buffer 14
•
•
•
0001 = Filter hits received in RX Buffer 1
0000 = Filter hits received in RX Buffer 0

bit 11-8 **F14BP<3:0>:** RX Buffer Mask for Filter 14 bits (same values as bits 15-12)

bit 7-4 **F13BP<3:0>:** RX Buffer Mask for Filter 13 bits (same values as bits 15-12)

bit 3-0 **F12BP<3:0>:** RX Buffer Mask for Filter 12 bits (same values as bits 15-12)

## 24.3 ADC Control Registers

### REGISTER 24-1: ADxCON1: ADCx CONTROL REGISTER 1

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| R/W-0 | U-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 |
| ADON | — | ADSIDL | ADDMABM | — | AD12B | FORM1 | FORM0 |
| bit 15 | | | | | | | bit 8 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0, HC, HS | R/C-0, HC, HS |
| SSRC2 | SSRC1 | SSRC0 | SSRCG | SIMSAM | ASAM | SAMP | DONE(1) |
| bit 7 | | | | | | | bit 0 |

| | | | |
|---|---|---|---|
| **Legend:** | C = Clearable bit | U = Unimplemented bit, read as ‘0’ | |
| R = Readable bit | W = Writable bit | HS = Hardware Settable bit | HC = Hardware Clearable bit |
| -n = Value at POR | ‘1’ = Bit is set | ‘0’ = Bit is cleared | x = Bit is unknown |

bit 15 **ADON:** ADCx Operating Mode bit

1 = ADCx module is operating
0 = ADCx is off

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **ADSIDL:** ADCx Stop in Idle Mode bit

1 = Discontinues module operation when the device enters Idle mode
0 = Continues module operation in Idle mode

bit 12 **ADDMABM:** ADCx DMA Buffer Build Mode bit

1 = DMA buffers are written in the order of conversion; the module provides an address to the DMA channel that is the same as the address used for the non-DMA stand-alone buffer
0 = DMA buffers are written in Scatter/Gather mode; the module provides a Scatter/Gather mode address to the DMA channel based on the index of the analog input and the size of the DMA buffer

bit 11 **Unimplemented:** Read as ‘0’

bit 10 **AD12B:** ADCx 10-Bit or 12-Bit Operation Mode bit

1 = 12-bit, 1-channel ADC operation
0 = 10-bit, 4-channel ADC operation

bit 9-8 **FORM<1:0>:** Data Output Format bits

For 10-Bit Operation:
11 = Signed fractional (DOUT = sddd dddd dd00 0000, where s = .NOT.d<9>)
10 = Fractional (DOUT = dddd dddd dd00 0000)
01 = Signed integer (DOUT = ssss sssd dddd dddd, where s = .NOT.d<9>)
00 = Integer (DOUT = 0000 00dd dddd dddd)

For 12-Bit Operation:
11 = Signed fractional (DOUT = sddd dddd dddd 0000, where s = .NOT.d<11>)
10 = Fractional (DOUT = dddd dddd dddd 0000)
01 = Signed integer (DOUT = ssss sddd dddd dddd, where s = .NOT.d<11>)
00 = Integer (DOUT = 0000 dddd dddd dddd)

**Note 1:** Do not clear the DONE bit in software if auto-sample is enabled (ASAM = 1).

## 26.0 COMPARATOR VOLTAGE REFERENCE

**Note 1:** This data sheet summarizes the features of the dsPIC33EVXXXGM00X/10X family of devices. It is not intended to be a comprehensive reference source. To complement the information in this data sheet, refer to **"Op Amp/Comparator"** (DS70000357) in the *"dsPIC33/PIC24 Family Reference Manual"*, which is available from the Microchip web site (www.microchip.com).

**2:** Some registers and associated bits described in this section may not be available on all devices. Refer to **Section 4.0 "Memory Organization"** in this data sheet for device-specific register and bit information.

### 26.1 Configuring the Comparator Voltage Reference

The comparator voltage reference module is controlled through the CVRxCON registers (Register 26-1 and Register 26-2). The comparator voltage reference provides a range of output voltages with 128 distinct levels. The comparator reference supply voltage can come from either VDD and VSS, or the external CVREF+ and AVSS pins. The voltage source is selected by the CVRSS bit (CVRxCON<11>). The settling time of the comparator voltage reference must be considered when changing the CVREF output.

### TABLE 28-2: INSTRUCTION SET OVERVIEW (CONTINUED)

| Base Instr # | Assembly Mnemonic | Assembly Syntax | Description | # of Words | # of Cycles | Status Flags Affected |
|---|---|---|---|---|---|---|
| 53 | MUL | MUL.SS Wb,Ws,Wnd | {Wnd + 1, Wnd} = signed(Wb) * signed(Ws) | 1 | 1 | None |
| | | MUL.SS Wb,Ws,Acc | Accumulator = signed(Wb) * signed(Ws) | 1 | 1 | None |
| | | MUL.SU Wb,Ws,Wnd | {Wnd + 1, Wnd} = signed(Wb) * unsigned(Ws) | 1 | 1 | None |
| | | MUL.SU Wb,Ws,Acc | Accumulator = signed(Wb) * unsigned(Ws) | 1 | 1 | None |
| | | MUL.SU Wb,#lit5,Acc | Accumulator = signed(Wb) * unsigned(lit5) | 1 | 1 | None |
| | | MUL.US Wb,Ws,Wnd | {Wnd + 1, Wnd} = unsigned(Wb) * signed(Ws) | 1 | 1 | None |
| | | MUL.US Wb,Ws,Acc | Accumulator = unsigned(Wb) * signed(Ws) | 1 | 1 | None |
| | | MUL.UU Wb,Ws,Wnd | {Wnd + 1, Wnd} = unsigned(Wb) * unsigned(Ws) | 1 | 1 | None |
| | | MUL.UU Wb,#lit5,Acc | Accumulator = unsigned(Wb) * unsigned(lit5) | 1 | 1 | None |
| | | MUL.UU Wb,Ws,Acc | Accumulator = unsigned(Wb) * unsigned(Ws) | 1 | 1 | None |
| | | MULW.SS Wb,Ws,Wnd | Wnd = signed(Wb) * signed(Ws) | 1 | 1 | None |
| | | MULW.SU Wb,Ws,Wnd | Wnd = signed(Wb) * unsigned(Ws) | 1 | 1 | None |
| | | MULW.US Wb,Ws,Wnd | Wnd = unsigned(Wb) * signed(Ws) | 1 | 1 | None |
| | | MULW.UU Wb,Ws,Wnd | Wnd = unsigned(Wb) * unsigned(Ws) | 1 | 1 | None |
| | | MUL.SU Wb,#lit5,Wnd | {Wnd + 1, Wnd} = signed(Wb) * unsigned(lit5) | 1 | 1 | None |
| | | MUL.SU Wb,#lit5,Wnd | Wnd = signed(Wb) * unsigned(lit5) | 1 | 1 | None |
| | | MUL.UU Wb,#lit5,Wnd | {Wnd + 1, Wnd} = unsigned(Wb) * unsigned(lit5) | 1 | 1 | None |
| | | MUL.UU Wb,#lit5,Wnd | Wnd = unsigned(Wb) * unsigned(lit5) | 1 | 1 | None |
| | | MUL f | W3:W2 = f * WREG | 1 | 1 | None |
| 54 | NEG | NEG Acc | Negate Accumulator | 1 | 1 | OA,OB,OAB,<br>SA,SB,SAB |
| | | NEG f | f = $\overline{f}$ + 1 | 1 | 1 | C,DC,N,OV,Z |
| | | NEG f,WREG | WREG = $\overline{f}$ + 1 | 1 | 1 | C,DC,N,OV,Z |
| | | NEG Ws,Wd | Wd = $\overline{Ws}$ + 1 | 1 | 1 | C,DC,N,OV,Z |
| 55 | NOP | NOP | No Operation | 1 | 1 | None |
| | | NOPR | No Operation | 1 | 1 | None |
| 56 | POP | POP f | Pop f from Top-of-Stack (TOS) | 1 | 1 | None |
| | | POP Wdo | Pop from Top-of-Stack (TOS) to Wdo | 1 | 1 | None |
| | | POP.D Wnd | Pop from Top-of-Stack (TOS) to W(nd):W(nd + 1) | 1 | 2 | None |
| | | POP.S | Pop Shadow Registers | 1 | 1 | All |
| 57 | PUSH | PUSH f | Push f to Top-of-Stack (TOS) | 1 | 1 | None |
| | | PUSH Wso | Push Wso to Top-of-Stack (TOS) | 1 | 1 | None |
| | | PUSH.D Wns | Push W(ns):W(ns + 1) to Top-of-Stack (TOS) | 1 | 2 | None |
| | | PUSH.S | Push Shadow Registers | 1 | 1 | None |
| 58 | PWRSAV | PWRSAV #lit1 | Go into Sleep or Idle mode | 1 | 1 | WDTO,Sleep |
| 59 | RCALL | RCALL Expr | Relative Call | 1 | 4 | SFA |
| | | RCALL Wn | Computed Call | 1 | 4 | SFA |
| 60 | REPEAT | REPEAT #lit15 | Repeat Next Instruction lit15 + 1 times | 1 | 1 | None |
| | | REPEAT Wn | Repeat Next Instruction (Wn) + 1 times | 1 | 1 | None |
| 61 | RESET | RESET | Software device Reset | 1 | 1 | None |
| 62 | RETFIE | RETFIE | Return from interrupt | 1 | 6 (5) | SFA |

**Note:** Read and Read-Modify-Write (e.g., bit operations and logical operations) on non-CPU SFRs incur an additional instruction cycle.

## TABLE 28-2: INSTRUCTION SET OVERVIEW (CONTINUED)

| Base Instr # | Assembly Mnemonic | Assembly Syntax | Description | # of Words | # of Cycles | Status Flags Affected |
|---|---|---|---|---|---|---|
| 63 | `RETLW` | `RETLW #lit10,Wn` | Return with literal in Wn | 1 | 6 (5) | SFA |
| 64 | `RETURN` | `RETURN` | Return from Subroutine | 1 | 6 (5) | SFA |
| 65 | `RLC` | `RLC f` | f = Rotate Left through Carry f | 1 | 1 | C,N,Z |
| | | `RLC f,WREG` | WREG = Rotate Left through Carry f | 1 | 1 | C,N,Z |
| | | `RLC Ws,Wd` | Wd = Rotate Left through Carry Ws | 1 | 1 | C,N,Z |
| 66 | `RLNC` | `RLNC f` | f = Rotate Left (No Carry) f | 1 | 1 | N,Z |
| | | `RLNC f,WREG` | WREG = Rotate Left (No Carry) f | 1 | 1 | N,Z |
| | | `RLNC Ws,Wd` | Wd = Rotate Left (No Carry) Ws | 1 | 1 | N,Z |
| 67 | `RRC` | `RRC f` | f = Rotate Right through Carry f | 1 | 1 | C,N,Z |
| | | `RRC f,WREG` | WREG = Rotate Right through Carry f | 1 | 1 | C,N,Z |
| | | `RRC Ws,Wd` | Wd = Rotate Right through Carry Ws | 1 | 1 | C,N,Z |
| 68 | `RRNC` | `RRNC f` | f = Rotate Right (No Carry) f | 1 | 1 | N,Z |
| | | `RRNC f,WREG` | WREG = Rotate Right (No Carry) f | 1 | 1 | N,Z |
| | | `RRNC Ws,Wd` | Wd = Rotate Right (No Carry) Ws | 1 | 1 | N,Z |
| 69 | `SAC` | `SAC Acc,#Slit4,Wdo` | Store Accumulator | 1 | 1 | None |
| | | `SAC.R Acc,#Slit4,Wdo` | Store Rounded Accumulator | 1 | 1 | None |
| 70 | `SE` | `SE Ws,Wnd` | Wnd = sign-extended Ws | 1 | 1 | C,N,Z |
| 71 | `SETM` | `SETM f` | f = 0xFFFF | 1 | 1 | None |
| | | `SETM WREG` | WREG = 0xFFFF | 1 | 1 | None |
| | | `SETM Ws` | Ws = 0xFFFF | 1 | 1 | None |
| 72 | `SFTAC` | `SFTAC Acc,Wn` | Arithmetic Shift Accumulator by (Wn) | 1 | 1 | OA,OB,OAB, SA,SB,SAB |
| | | `SFTAC Acc,#Slit6` | Arithmetic Shift Accumulator by Slit6 | 1 | 1 | OA,OB,OAB, SA,SB,SAB |
| 73 | `SL` | `SL f` | f = Left Shift f | 1 | 1 | C,N,OV,Z |
| | | `SL f,WREG` | WREG = Left Shift f | 1 | 1 | C,N,OV,Z |
| | | `SL Ws,Wd` | Wd = Left Shift Ws | 1 | 1 | C,N,OV,Z |
| | | `SL Wb,Wns,Wnd` | Wnd = Left Shift Wb by Wns | 1 | 1 | N,Z |
| | | `SL Wb,#lit5,Wnd` | Wnd = Left Shift Wb by lit5 | 1 | 1 | N,Z |
| 74 | `SUB` | `SUB Acc` | Subtract Accumulators | 1 | 1 | OA,OB,OAB, SA,SB,SAB |
| | | `SUB f` | f = f – WREG | 1 | 1 | C,DC,N,OV,Z |
| | | `SUB f,WREG` | WREG = f – WREG | 1 | 1 | C,DC,N,OV,Z |
| | | `SUB #lit10,Wn` | Wn = Wn – lit10 | 1 | 1 | C,DC,N,OV,Z |
| | | `SUB Wb,Ws,Wd` | Wd = Wb – Ws | 1 | 1 | C,DC,N,OV,Z |
| | | `SUB Wb,#lit5,Wd` | Wd = Wb – lit5 | 1 | 1 | C,DC,N,OV,Z |
| 75 | `SUBB` | `SUBB f` | f = f – WREG – ($\overline{C}$) | 1 | 1 | C,DC,N,OV,Z |
| | | `SUBB f,WREG` | WREG = f – WREG – ($\overline{C}$) | 1 | 1 | C,DC,N,OV,Z |
| | | `SUBB #lit10,Wn` | Wn = Wn – lit10 – ($\overline{C}$) | 1 | 1 | C,DC,N,OV,Z |
| | | `SUBB Wb,Ws,Wd` | Wd = Wb – Ws – ($\overline{C}$) | 1 | 1 | C,DC,N,OV,Z |
| | | `SUBB Wb,#lit5,Wd` | Wd = Wb – lit5 – ($\overline{C}$) | 1 | 1 | C,DC,N,OV,Z |
| 76 | `SUBR` | `SUBR f` | f = WREG – f | 1 | 1 | C,DC,N,OV,Z |
| | | `SUBR f,WREG` | WREG = WREG – f | 1 | 1 | C,DC,N,OV,Z |
| | | `SUBR Wb,Ws,Wd` | Wd = Ws – Wb | 1 | 1 | C,DC,N,OV,Z |
| | | `SUBR Wb,#lit5,Wd` | Wd = lit5 – Wb | 1 | 1 | C,DC,N,OV,Z |
| 77 | `SUBBR` | `SUBBR f` | f = WREG – f – ($\overline{C}$) | 1 | 1 | C,DC,N,OV,Z |
| | | `SUBBR f,WREG` | WREG = WREG – f – ($\overline{C}$) | 1 | 1 | C,DC,N,OV,Z |
| | | `SUBBR Wb,Ws,Wd` | Wd = Ws – Wb – ($\overline{C}$) | 1 | 1 | C,DC,N,OV,Z |
| | | `SUBBR Wb,#lit5,Wd` | Wd = lit5 – Wb – ($\overline{C}$) | 1 | 1 | C,DC,N,OV,Z |

**Note:** Read and Read-Modify-Write (e.g., bit operations and logical operations) on non-CPU SFRs incur an additional instruction cycle.

## 29.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip's 8, 16, and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

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

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. 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. MPLAB XC Compiler uses the assembler to produce its object 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 X IDE compatibility

## 29.3 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 X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

## 29.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. 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

## 29.5 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC 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 X IDE compatibility

### TABLE 30-13: DC CHARACTERISTICS: PROGRAM MEMORY

| **DC CHARACTERISTICS** | | | **Standard Operating Conditions: 4.5V to 5.5V (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for Industrial<br>-40°C ≤ TA ≤ +125°C for Extended | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Symbol** | **Characteristic** | **Min.** | **Typ.[1]** | **Max.** | **Units** | **Conditions** |
| | | **Program Flash Memory** | | | | | |
| D130 | EP | Cell Endurance | 10,000 | — | — | E/W | -40°C to +125°C |
| D131 | VPR | VDD for Read | 4.5 | — | 5.5 | V | |
| D132b | VPEW | VDD for Self-Timed Write | 4.5 | — | 5.5 | V | |
| D134 | TRETD | Characteristic Retention | 20 | — | — | Year | Provided no other specifications are violated, -40°C to +125°C |
| D135 | IDDP | Supply Current During Programming | — | 10 | — | mA | |
| D136a | TRW | Row Write Cycle Time | 0.657 | — | 0.691 | ms | TRW = 4965 FRC cycles, TA = +85°C (see **Note 2)** |
| D136b | TRW | Row Write Cycle Time | 0.651 | — | 0.698 | ms | TRW = 4965 FRC cycles, TA = +125°C (see **Note 2)** |
| D137a | TPE | Page Erase Time | 19.44 | — | 20.44 | ms | TPE = 146893 FRC cycles, TA = +85°C (see **Note 2)** |
| D137b | TPE | Page Erase Time | 19.24 | — | 20.65 | ms | TPE = 146893 FRC cycles, TA = +125°C (see **Note 2)** |
| D138a | TWW | Word Write Cycle Time | 45.78 | — | 48.15 | µs | TWW = 346 FRC cycles, TA = +85°C (see **Note 2)** |
| D138b | TWW | Word Write Cycle Time | 45.33 | — | 48.64 | µs | TWW = 346 FRC cycles, TA = +125°C (see **Note 2)** |

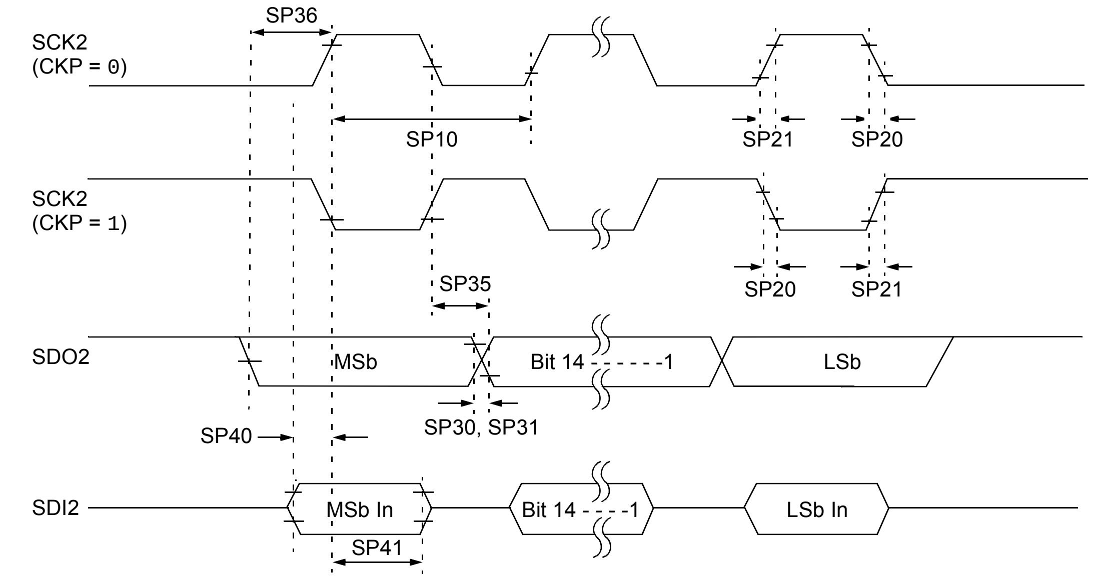
**Note 1:** Data in “Typ.” column is at 5.0V, +25°C unless otherwise stated.

**2:** Other conditions: FRC = 7.3728 MHz, TUN<5:0> = `b'011111` (for Min), TUN<5:0> = `b'100000` (for Max). This parameter depends on the FRC accuracy (see Table 30-20) and the value of the FRC Oscillator Tuning register.

### TABLE 30-14: ELECTRICAL CHARACTERISTICS: INTERNAL BAND GAP REFERENCE VOLTAGE

| **DC CHARACTERISTICS** | | | **Standard Operating Conditions: 4.5V to 5.5V (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for Industrial<br>-40°C ≤ TA ≤ +125°C for Extended | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Symbol** | **Characteristic** | **Min.** | **Typ.** | **Max.** | **Units** | **Conditions** |
| DVR10 | VBG | Internal Band Gap Reference Voltage | 1.14 | 1.2 | 1.26 | V | |

### FIGURE 30-14: SPI2 MASTER MODE (FULL-DUPLEX, CKE = 1, CKP = x, SMP = 1) TIMING CHARACTERISTICS



**Note:** Refer to Figure 30-1 for load conditions.

### TABLE 30-32: SPI2 MASTER MODE (FULL-DUPLEX, CKE = 1, CKP = x, SMP = 1) TIMING REQUIREMENTS

| AC CHARACTERISTICS | | | **Standard Operating Conditions: 4.5V to 5.5V (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for Industrial<br>-40°C ≤ TA ≤ +125°C for Extended | | | | |
|---|---|---|---|---|---|---|---|
| **Param.** | **Symbol** | **Characteristic[1]** | **Min.** | **Typ.[2]** | **Max.** | **Units** | **Conditions** |
| SP10 | FscP | Maximum SCK2 Frequency | — | — | 9 | MHz | See **Note 3** |
| SP20 | TscF | SCK2 Output Fall Time | — | — | — | ns | See Parameter DO32 and **Note 4** |
| SP21 | TscR | SCK2 Output Rise Time | — | — | — | ns | See Parameter DO31 and **Note 4** |
| SP30 | TdoF | SDO2 Data Output Fall Time | — | — | — | ns | See Parameter DO32 and **Note 4** |
| SP31 | TdoR | SDO2 Data Output Rise Time | — | — | — | ns | See Parameter DO31 and **Note 4** |
| SP35 | TscH2doV,<br>TscL2doV | SDO2 Data Output Valid after SCK2 Edge | — | 6 | 20 | ns | |
| SP36 | TdoV2sc,<br>TdoV2scL | SDO2 Data Output Setup to First SCK2 Edge | 30 | — | — | ns | |
| SP40 | TdiV2scH,<br>TdiV2scL | Setup Time of SDI2 Data Input to SCK2 Edge | 30 | — | — | ns | |
| SP41 | TscH2diL,<br>TscL2diL | Hold Time of SDI2 Data Input to SCK2 Edge | 30 | — | — | ns | |

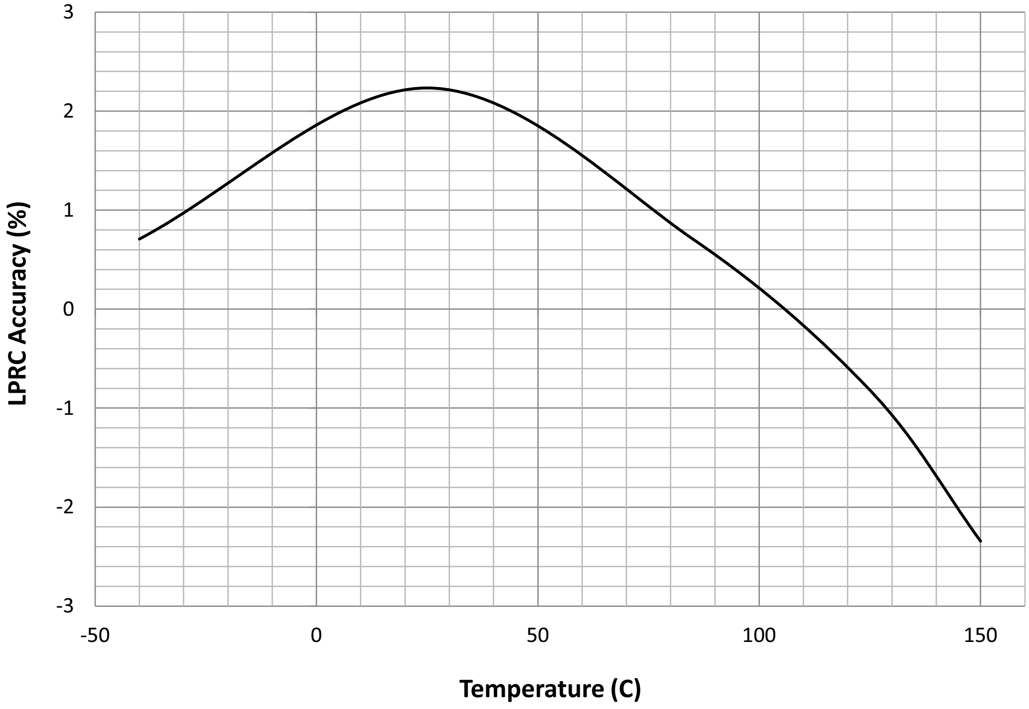
**Note 1:** These parameters are characterized but not tested in manufacturing.

**2:** Data in “Typ.” column is at 5.0V, +25°C unless otherwise stated.

**3:** The minimum clock period for SCK2 is 111 ns. The clock generated in Master mode must not violate this specification.

**4:** Assumes 50 pF load on all SPI2 pins.

FIGURE 32-23: TYPICAL LPRC ACCURACY vs. TEMPERATURE (5.5V VDD)



## 32.7 Leakage Current

FIGURE 32-24: TYPICAL IIL vs. TEMPERATURE ($\overline{\text{MCLR}}$)

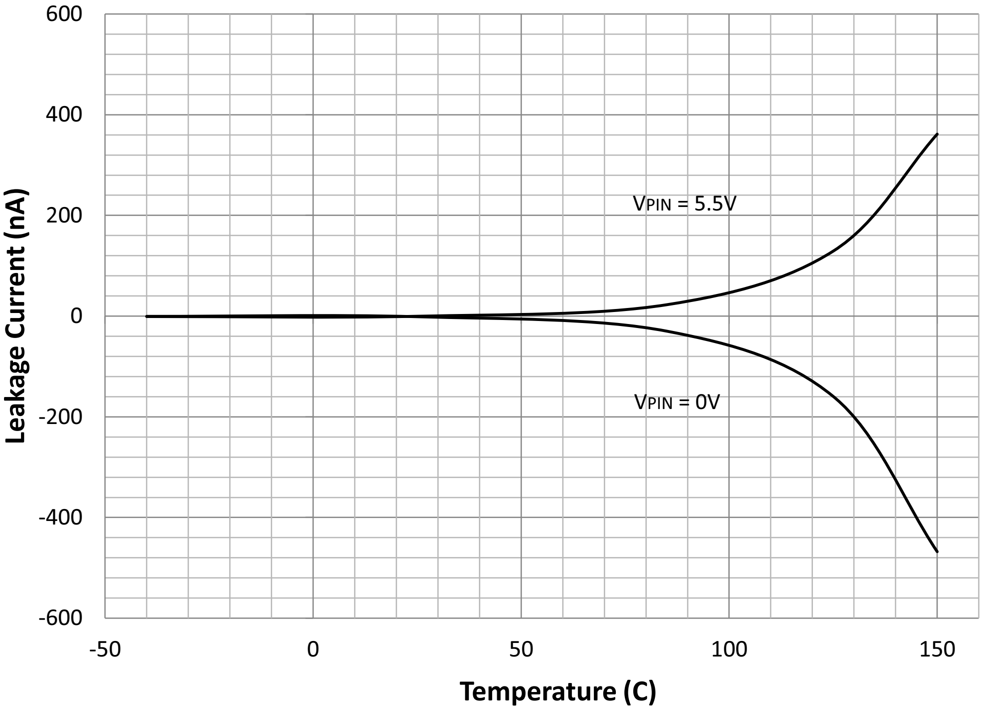


FIGURE 33-27: TYPICAL VOH 4x DRIVER PINS vs. IOH (GENERAL PURPOSE I/Os, TEMPERATURES AS NOTED)

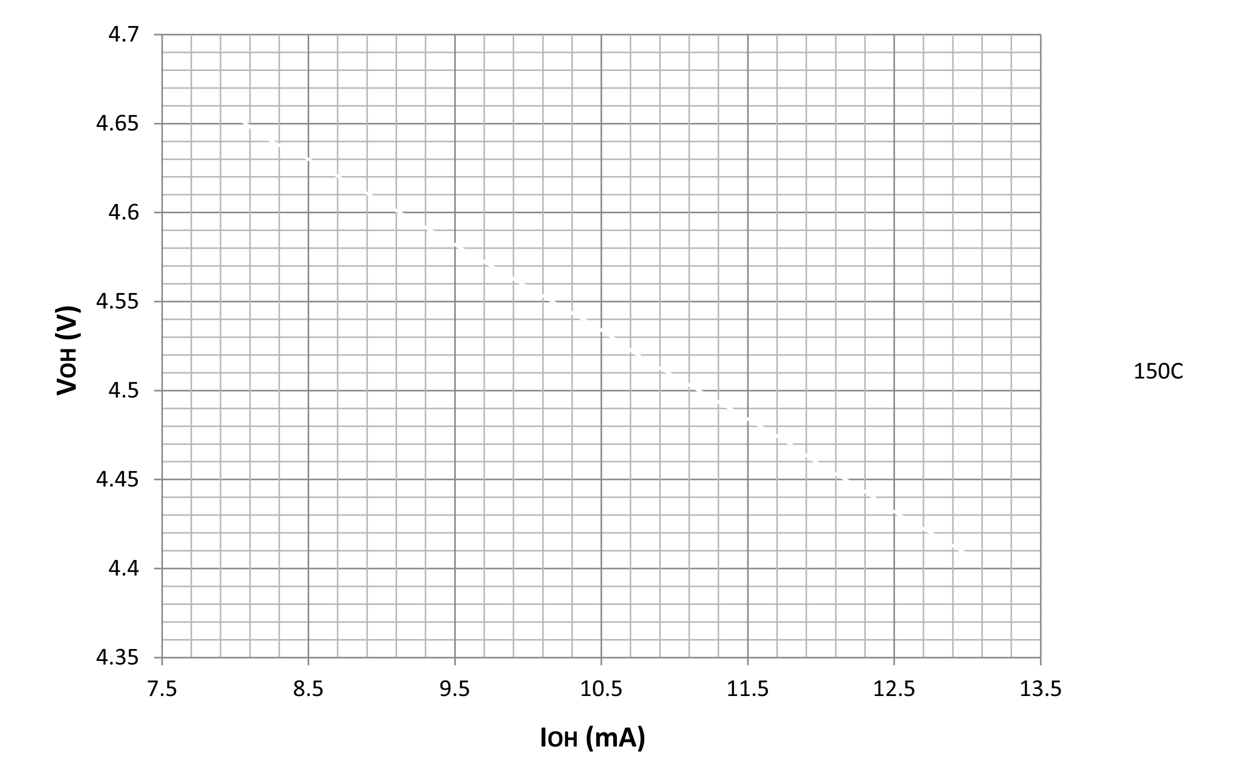
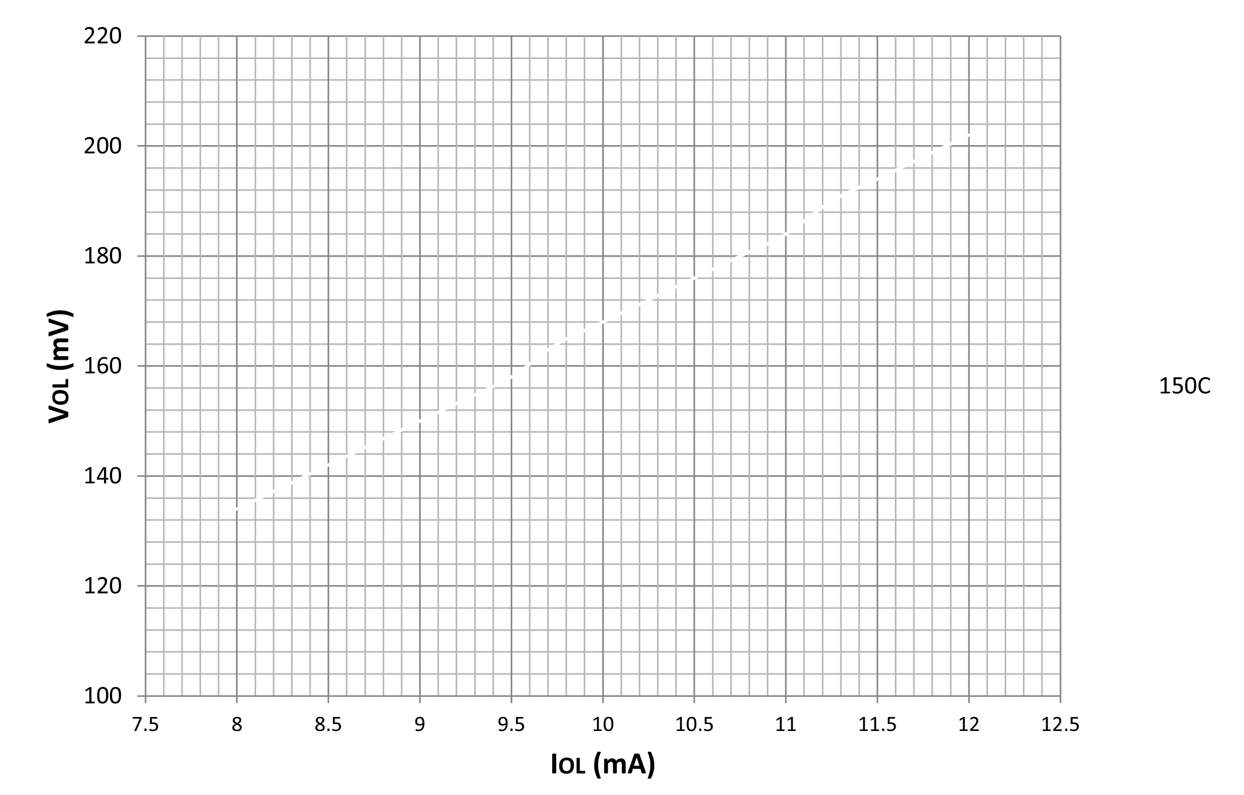
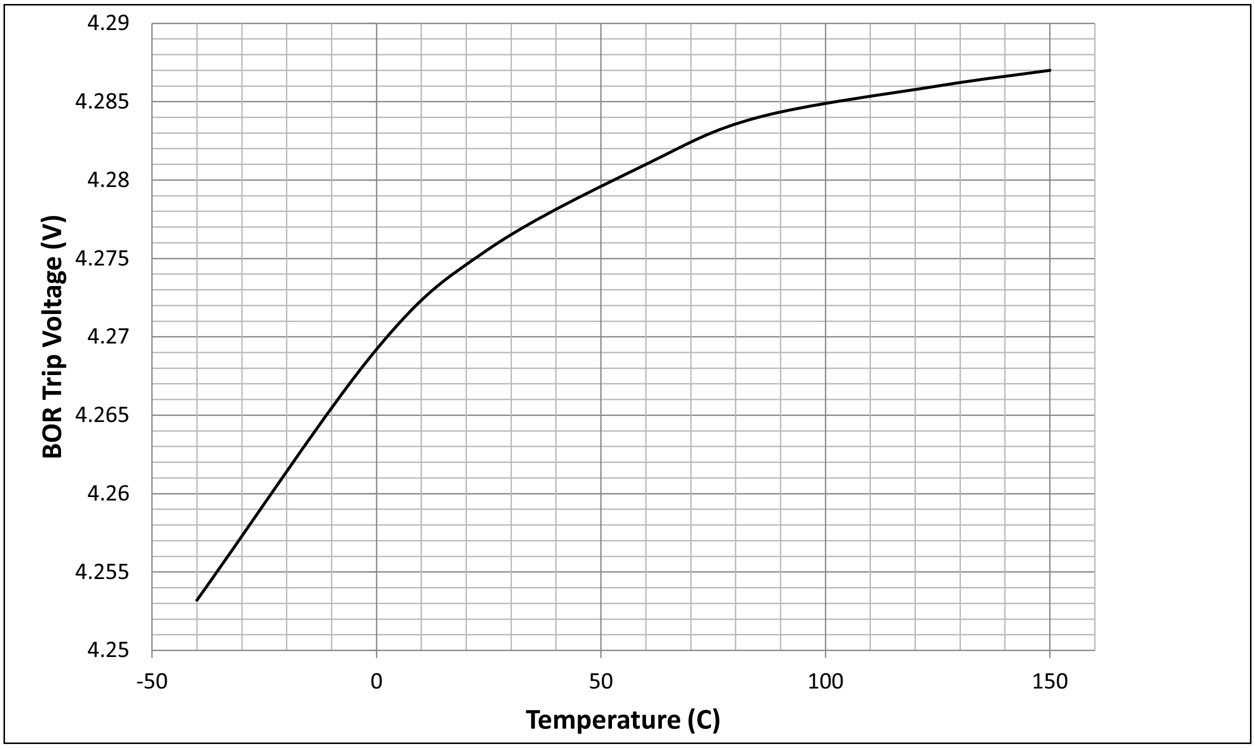


FIGURE 33-28: TYPICAL VOL 8x DRIVER PINS vs. IOL (GENERAL PURPOSE I/Os, TEMPERATURES AS NOTED)



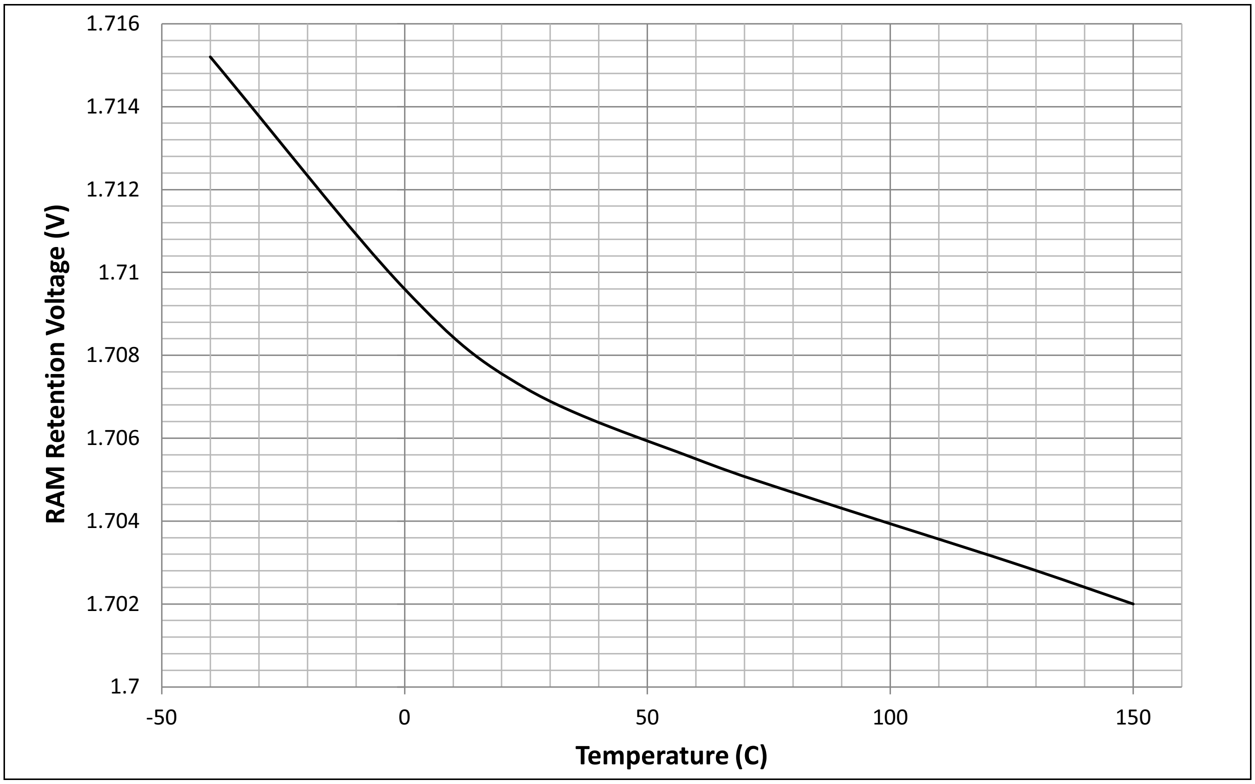
## 33.12 VBOR

FIGURE 33-31: TYPICAL BOR TRIP RANGE vs. TEMPERATURE



## 33.13 RAM Retention

FIGURE 33-32: TYPICAL RAM RETENTION VOLTAGE vs. TEMPERATURE


RAM Retention Voltage (V)
Temperature (C)