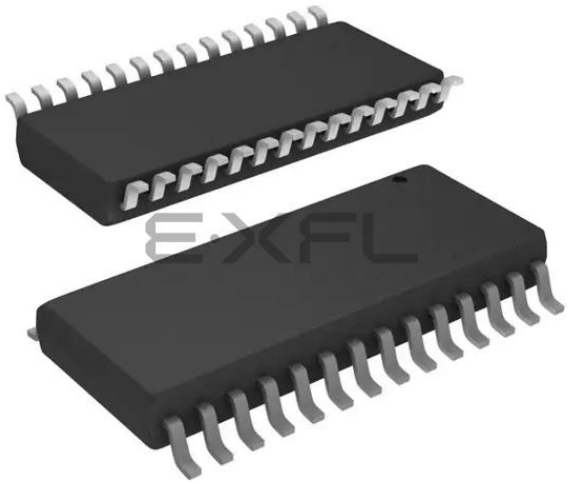


Welcome to E-XFL.COM

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	25MHz
Connectivity	CANbus, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.25K x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f2680-e-so

TABLE 1-3: PIC18F4585/4680 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTE is a bidirectional I/O port.
RE0/$\overline{RD}$/AN5	8	25	25			
RE0				I/O	ST	Digital I/O.
$\overline{RD}$				I	TTL	Read control for Parallel Slave Port (see also $\overline{WR}$ and $\overline{CS}$ pins).
AN5				I	Analog	Analog input 5.
RE1/$\overline{WR}$/AN6/C1OUT	9	26	26			
RE1				I/O	ST	Digital I/O.
$\overline{WR}$				I	TTL	Write control for Parallel Slave Port (see $\overline{CS}$ and $\overline{RD}$ pins).
AN6				I	Analog	Analog input 6.
C1OUT				O	TTL	Comparator 1 output.
RE2/$\overline{CS}$/AN7/C2OUT	10	27	27			
RE2				I/O	ST	Digital I/O.
$\overline{CS}$				I	TTL	Chip select control for Parallel Slave Port (see related $\overline{RD}$ and $\overline{WR}$).
AN7				I	Analog	Analog input 7.
C2OUT				O	TTL	Comparator 2 output.
RE3	—	—	—	—	—	See $\overline{MCLR}$/VPP/RE3 pin.
VSS	12, 31	6, 30, 31	6, 29	P	—	Ground reference for logic and I/O pins.
VDD	11, 32	7, 8, 28, 29	7, 28	P	—	Positive supply for logic and I/O pins.
NC	—	13	12, 13, 33, 34	—	—	No connect.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

REGISTER 4-1: RCON: RESET CONTROL REGISTER

R/W-0	R/W-1(1)	U-0	R/W-1	R-1	R-1	R/W-0(2)	R/W-0
IPEN	SBOREN	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

bit 7 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (16CXXX Compatibility mode)

bit 6 **SBOREN:** BOR Software Enable bit(1)
If BOREN1:BOREN0 = 01:
1 = BOR is enabled
0 = BOR is disabled
If BOREN1:BOREN0 = 00, 10 or 11:
Bit is disabled and read as '0'.

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

bit 4 **$\overline{RI}$:** RESET Instruction Flag bit
1 = The RESET instruction was not executed (set by firmware only)
0 = The RESET instruction was executed causing a device Reset (must be set in software after a Brown-out Reset occurs)

bit 3 **$\overline{TO}$:** Watchdog Time-out Flag bit
1 = Set by power-up, CLRWDT instruction or SLEEP instruction
0 = A WDT time-out occurred

bit 2 **$\overline{PD}$:** Power-down Detection Flag bit
1 = Set by power-up or by the CLRWDT instruction
0 = Set by execution of the SLEEP instruction

bit 1 **$\overline{POR}$:** Power-on Reset Status bit(2)
1 = A Power-on Reset has not occurred (set by firmware only)
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0 **$\overline{BOR}$:** Brown-out Reset Status bit
1 = A Brown-out Reset has not occurred (set by firmware only)
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

Note 1: If SBOREN is enabled, its Reset state is '1'; otherwise, it is '0'.

2: The actual Reset value of $\overline{POR}$ is determined by the type of device Reset. See the notes following this register and **Section 4.6 "Reset State of Registers"** for additional information.

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

Note 1: It is recommended that the $\overline{POR}$ bit be set after a Power-on Reset has been detected so that subsequent Power-on Resets may be detected.

2: Brown-out Reset is said to have occurred when $\overline{BOR}$ is '0' and POR is '1' (assuming that $\overline{POR}$ was set to '1' by software immediately after POR).

TABLE 5-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F2585/2680/4585/4680 DEVICES (CONTINUED)

Address	Name	Address	Name	Address	Name	Address	Name
E7Fh	CANCON_RO4	E6Fh	CANCON_RO5	E5Fh	CANCON_RO6	E4Fh	CANCON_RO7
E7Eh	CANSTAT_RO4	E6Eh	CANSTAT_RO5	E5Eh	CANSTAT_RO6	E4Eh	CANSTAT_RO7
E7Dh	B5D7(2)	E6Dh	B4D7(2)	E5Dh	B3D7(2)	E4Dh	B2D7(2)
E7Ch	B5D6(2)	E6Ch	B4D6(2)	E5Ch	B3D6(2)	E4Ch	B2D6(2)
E7Bh	B5D5(2)	E6Bh	B4D5(2)	E5Bh	B3D5(2)	E4Bh	B2D5(2)
E7Ah	B5D4(2)	E6Ah	B4D4(2)	E5Ah	B3D4(2)	E4Ah	B2D4(2)
E79h	B5D3(2)	E69h	B4D3(2)	E59h	B3D3(2)	E49h	B2D3(2)
E78h	B5D2(2)	E68h	B4D2(2)	E58h	B3D2(2)	E48h	B2D2(2)
E77h	B5D1(2)	E67h	B4D1(2)	E57h	B3D1(2)	E47h	B2D1(2)
E76h	B5D0(2)	E66h	B4D0(2)	E56h	B3D0(2)	E46h	B2D0(2)
E75h	B5DLC(2)	E65h	B4DLC(2)	E55h	B3DLC(2)	E45h	B2DLC(2)
E74h	B5EIDL(2)	E64h	B4EIDL(2)	E54h	B3EIDL(2)	E44h	B2EIDL(2)
E73h	B5EIDH(2)	E63h	B4EIDH(2)	E53h	B3EIDH(2)	E43h	B2EIDH(2)
E72h	B5SIDL(2)	E62h	B4SIDL(2)	E52h	B3SIDL(2)	E42h	B2SIDL(2)
E71h	B5SIDH(2)	E61h	B4SIDH(2)	E51h	B3SIDH(2)	E41h	B2SIDH(2)
E70h	B5CON(2)	E60h	B4CON(2)	E50h	B3CON(2)	E40h	B2CON(2)
E3Fh	CANCON_RO8	E2Fh	CANCON_RO9	E1Fh	—	E0Fh	—
E3Eh	CANSTAT_RO8	E2Eh	CANSTAT_RO9	E1Eh	—	E0Eh	—
E3Dh	B1D7(2)	E2Dh	B0D7(2)	E1Dh	—	E0Dh	—
E3Ch	B1D6(2)	E2Ch	B0D6(2)	E1Ch	—	E0Ch	—
E3Bh	B1D5(2)	E2Bh	B0D5(2)	E1Bh	—	E0Bh	—
E3Ah	B1D4(2)	E2Ah	B0D4(2)	E1Ah	—	E0Ah	—
E39h	B1D3(2)	E29h	B0D3(2)	E19h	—	E09h	—
E38h	B1D2(2)	E28h	B0D2(2)	E18h	—	E08h	—
E37h	B1D1(2)	E27h	B0D1(2)	E17h	—	E07h	—
E36h	B1D0(2)	E26h	B0D0(2)	E16h	—	E06h	—
E35h	B1DLC(2)	E25h	B0DLC(2)	E15h	—	E05h	—
E34h	B1EIDL(2)	E24h	B0EIDL(2)	E14h	—	E04h	—
E33h	B1EIDH(2)	E23h	B0EIDH(2)	E13h	—	E03h	—
E32h	B1SIDL(2)	E22h	B0SIDL(2)	E12h	—	E02h	—
E31h	B1SIDH(2)	E21h	B0SIDH(2)	E11h	—	E01h	—
E30h	B1CON(2)	E20h	B0CON(2)	E10h	—	E00h	—

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 (PIC18F2585/2680/4585/4680) (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:
B0DLC(8) Receive mode	—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0	-xxx xxxx	56, 300
B0DLC(8) Transmit mode	—	TXRTR	—	—	DLC3	DLC2	DLC1	DLC0	-x-- xxxx	56, 301
B0EIDL(8)	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	58, 299
B0EIDH(8)	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	58, 298
B0SIDL(8) Receive mode	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	xxxx x-xx	56, 297
B0SIDL(8) Transmit mode	SID2	SID1	SID0	—	EXIDE	—	EID17	EID16	xxx- x-xx	56, 297
B0SIDH(8)	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	58, 296
B0CON(8) Receive mode	RXFUL	RXM1	RXRTRRO	FILHIT4	FILHIT3	FILHIT2	FILHIT1	FILHIT0	0000 0000	58, 295
B0CON(8) Transmit mode	TXBIF	TXABT	TXLARB	TXERR	TXREQ	RTREN	TXPRI1	TXPRI0	0000 0000	58, 295
TXBIE	—	—	—	TXB2IE	TXB1IE	TXB0IE	—	—	---0 00--	58, 318
BIE0	B5IE	B4IE	B3IE	B2IE	B1IE	B0IE	RXB1IE	RXB0IE	0000 0000	58, 318
BSEL0	B5TXEN	B4TXEN	B3TXEN	B2TXEN	B1TXEN	B0TXEN	—	—	0000 00--	59, 301
MSEL3	FIL15_1	FIL15_0	FIL14_1	FIL14_0	FIL13_1	FIL13_0	FIL12_1	FIL12_0	0000 0000	59, 310
MSEL2	FIL11_1	FIL11_0	FIL10_1	FIL10_0	FIL9_1	FIL9_0	FIL8_1	FIL8_0	0000 0000	59, 309
MSEL1	FIL7_1	FIL7_0	FIL6_1	FIL6_0	FIL5_1	FIL5_0	FIL4_1	FIL4_0	0000 0101	59, 308
MSEL0	FIL3_1	FIL3_0	FIL2_1	FIL2_0	FIL1_1	FIL1_0	FIL0_1	FIL0_0	0101 0000	59, 307
RXFBCON7	F15BP_3	F15BP_2	F15BP_1	F15BP_0	F14BP_3	F14BP_2	F14BP_1	F14BP_0	0000 0000	59, 305
RXFBCON6	F13BP_3	F13BP_2	F13BP_1	F13BP_0	F12BP_3	F12BP_2	F12BP_1	F12BP_0	0000 0000	59, 305
RXFBCON5	F11BP_3	F11BP_2	F11BP_1	F11BP_0	F10BP_3	F10BP_2	F10BP_1	F10BP_0	0000 0000	59, 305
RXFBCON4	F9BP_3	F9BP_2	F9BP_1	F9BP_0	F8BP_3	F8BP_2	F8BP_1	F8BP_0	0000 0000	59, 305
RXFBCON3	F7BP_3	F7BP_2	F7BP_1	F7BP_0	F6BP_3	F6BP_2	F6BP_1	F6BP_0	0000 0000	59, 305
RXFBCON2	F5BP_3	F5BP_2	F5BP_1	F5BP_0	F4BP_3	F4BP_2	F4BP_1	F4BP_0	0001 0001	59, 305
RXFBCON1	F3BP_3	F3BP_2	F3BP_1	F3BP_0	F2BP_3	F2BP_2	F2BP_1	F2BP_0	0001 0001	59, 305
RXFBCON0	F1BP_3	F1BP_2	F1BP_1	F1BP_0	F0BP_3	F0BP_2	F0BP_1	F0BP_0	0000 0000	59, 305
SDFLC	—	—	—	FLC4	FLC3	FLC2	FLC1	FLC0	---0 0000	59, 305
RXFCON1	RXF15EN	RXF14EN	RXF13EN	RXF12EN	RXF11EN	RXF10EN	RXF9EN	RXF8EN	0000 0000	59, 306
RXFCON0	RXF7EN	RXF6EN	RXF5EN	RXF4EN	RXF3EN	RXF2EN	RXF1EN	RXF0EN	0000 0000	59, 305
RXF15EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	59, 303
RXF15EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	59, 303
RXF15SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	59, 304
RXF15SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	59, 303
RXF14EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	59, 303
RXF14EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	59, 303
RXF14SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	59, 304
RXF14SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	59, 303
RXF13EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	59, 303
RXF13EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	59, 303

Legend: x = unknown, u = unchanged, - = unimplemented, q = value depends on condition

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 PIC18F2X8X devices and are read as '0'. Reset values are shown for PIC18F4X8X devices; individual unimplemented bits should be interpreted as '—'.

4: The PLLEN bit is only available in specific oscillator configuration; 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 are available on PIC18F4X8X devices only.

9.0 INTERRUPTS

The PIC18F2585/2680/4585/4680 devices have multiple interrupt sources and an interrupt priority feature that allows each interrupt source to be assigned a high priority level or a low priority level. The high priority interrupt vector is at 000008h and the low priority interrupt vector is at 000018h. High priority interrupt events will interrupt any low priority interrupts that may be in progress.

There are ten registers which are used to control interrupt operation. These registers are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2, PIR3
- PIE1, PIE2, PIE3
- IPR1, IPR2, IPR3

It is recommended that the Microchip header files supplied with MPLAB® IDE be used for the symbolic bit names in these registers. This allows the assembler/compiler to automatically take care of the placement of these bits within the specified register.

Each interrupt source has three bits to control its operation. The functions of these bits are:

- Flag bit to indicate that an interrupt event occurred
- Enable bit that allows program execution to branch to the interrupt vector address when the flag bit is set
- Priority bit to select high priority or low priority

The interrupt priority feature is enabled by setting the IPEN bit (RCON<7>). When interrupt priority is enabled, there are two bits which enable interrupts globally. Setting the GIEH bit (INTCON<7>) enables all interrupts that have the priority bit set (high priority). Setting the GIEL bit (INTCON<6>) enables all interrupts that have the priority bit cleared (low priority). When the interrupt flag, enable bit and appropriate global interrupt enable bit are set, the interrupt will vector immediately to address 000008h or 000018h, depending on the priority bit setting. Individual interrupts can be disabled through their corresponding enable bits.

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PIC® mid-range devices. In Compatibility mode, the interrupt priority bits for each source have no effect. INTCON<6> is the PEIE bit, which enables/disables all peripheral interrupt sources. INTCON<7> is the GIE bit, which enables/disables all interrupt sources. All interrupts branch to address 000008h in Compatibility mode.

When an interrupt is responded to, the global interrupt enable bit is cleared to disable further interrupts. If the IPEN bit is cleared, this is the GIE bit. If interrupt priority levels are used, this will be either the GIEH or GIEL bit. High priority interrupt sources can interrupt a low priority interrupt. Low priority interrupts are not processed while high priority interrupts are in progress.

The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (000008h or 000018h). Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

The “return from interrupt” instruction, `RETFIE`, exits the interrupt routine and sets the GIE bit (GIEH or GIEL if priority levels are used), which re-enables interrupts.

For external interrupt events, such as the INT pins or the PORTB input change interrupt, the interrupt latency will be three to four instruction cycles. The exact latency is the same for one or two-cycle instructions. Individual interrupt flag bits are set, regardless of the status of their corresponding enable bit or the GIE bit.

Note:	Do not use the `MOVFF` instruction to modify any of the interrupt control registers while **any** interrupt is enabled. Doing so may cause erratic microcontroller behavior.

TABLE 10-3: PORTB I/O SUMMARY

Pin Name	Function	I/O	TRIS	Buffer	Description
RB0/INT0/FLT0/AN10	RB0	OUT	0	DIG	LATB<0> data output.
		IN	1	TTL	PORTB<0> data input. Weak pull-up available only in this mode.
	INT0	IN	1	ST	External interrupt 0 input.
	FLT0	IN	1	ST	Enhanced PWM Fault input.
	AN10	IN	1	ANA	A/D input channel 10. Enabled on POR, this analog input overrides the digital input (read as clear – low level).
RB1/INT1/AN8	RB1	OUT	0	DIG	LATB<1> data output.
		IN	1	TTL	PORTB<1> data input. Weak pull-up available only in this mode.
	INT1	IN	1	ST	External interrupt 1 input.
	AN8	IN	1	ANA	A/D input channel 8. Enabled on POR, this analog input overrides the digital input (read as clear – low level).
RB2/INT2/CANTX	RB2	OUT	x	DIG	LATB<2> data output.
		IN	1	TTL	PORTB<2> data input. Weak pull-up available only in this mode.
	INT2	IN	1	ST	External interrupt 2 input.
	CANTX	OUT	1	DIG	CAN transmit signal output. The CAN interface overrides the TRIS<2> control when enabled.
RB3/CANRX	RB3	OUT	0	DIG	LATB<3> data output.
		IN	1	TTL	PORTB<3> data input. Weak pull-up available only in this mode.
	CANRX	IN	1	ST	CAN receive signal input. Pin must be configured as a digital input by setting TRISB<3>.
RB4/KBI0/AN9	RB4	OUT	0	DIG	LATB<4> data output.
		IN	1	TTL	PORTB<4> data input. Weak pull-up available only in this mode.
	KBI0	IN	1	TTL	Interrupt-on-pin change.
	AN9	IN	1	ANA	A/D input channel 9. Enabled on POR, this analog input overrides the digital input (read as clear – low level).
RB5/KBI1/PGM	RB5	OUT	0	DIG	LATB<5> data output.
		IN	1	TTL	PORTB<5> data input. Weak pull-up available only in this mode.
	KBI1	IN	1	TTL	Interrupt-on-pin change.
	PGM	IN	x	ST	Low-Voltage Programming mode entry (ICSP™). Enabling this function overrides digital output.
RB6/KBI2/PGC	RB6	OUT	0	DIG	LATB<6> data output.
		IN	1	TTL	PORTB<6> data input. Weak pull-up available only in this mode.
	KBI2	IN	1	TTL	Interrupt-on-pin change.
	PGC	IN	x	ST	Low-Voltage Programming mode entry (ICSP) clock input.
RB7/KBI3/PGD	RB7	OUT	0	DIG	LATB<7> data output.
		IN	1	TTL	PORTB<7> data input. Weak pull-up available only in this mode.
	KBI3	IN	1	TTL	Interrupt-on-pin change.
	PGD	OUT	x	DIG	Low-Voltage Programming mode entry (ICSP) clock output.
		IN	x	ST	Low-Voltage Programming mode entry (ICSP) clock input.

Legend: PWR = Power Supply; OUT = Output; IN = Input; ANA = Analog Signal; DIG = Digital Output; ST = Schmitt Buffer Input; TTL – TTL Buffer Input

REGISTER 10-1: TRISE REGISTER (PIC18F4X8X DEVICES ONLY)

R-0	R-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0
bit 7							bit 0

bit 7 **IBF:** Input Buffer Full Status bit
1 = A word has been received and waiting to be read by the CPU
0 = No word has been received

bit 6 **OBF:** Output Buffer Full Status bit
1 = The output buffer still holds a previously written word
0 = The output buffer has been read

bit 5 **IBOV:** Input Buffer Overflow Detect bit (in Microprocessor mode)
1 = A write occurred when a previously input word has not been read (must be cleared in software)
0 = No overflow occurred

bit 4 **PSPMODE:** Parallel Slave Port Mode Select bit
1 = Parallel Slave Port mode
0 = General Purpose I/O mode

bit 3 **Unimplemented:** Read as '0'

bit 2 **TRISE2:** RE2 Direction Control bit
1 = Input
0 = Output

bit 1 **TRISE1:** RE1 Direction Control bit
1 = Input
0 = Output

bit 0 **TRISE0:** RE0 Direction Control bit
1 = Input
0 = Output

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

REGISTER 17-2: SSPCON1: MSSP CONTROL REGISTER 1 (SPI MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

bit 7 **WCOL:** Write Collision Detect bit (Transmit mode only)

1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)

0 = No collision

bit 6 **SSPOV:** Receive Overflow Indicator bit

SPI Slave mode:

1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. The user must read the SSPBUF, even if only transmitting data, to avoid setting overflow (must be cleared in software).

0 = No overflow

Note: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.

bit 5 **SSPEN:** Synchronous Serial Port Enable bit

1 = Enables serial port and configures SCK, SDO, SDI and $\overline{SS}$ as serial port pins

0 = Disables serial port and configures these pins as I/O port pins

Note: When enabled, these pins must be properly configured as input or output.

bit 4 **CKP:** Clock Polarity Select bit

1 = Idle state for clock is a high level

0 = Idle state for clock is a low level

bit 3-0 **SSPM3:SSPM0:** Synchronous Serial Port Mode Select bits

0101 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control disabled, $\overline{SS}$ can be used as I/O pin

0100 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control enabled

0011 = SPI Master mode, clock = TMR2 output/2

0010 = SPI Master mode, clock = FOSC/64

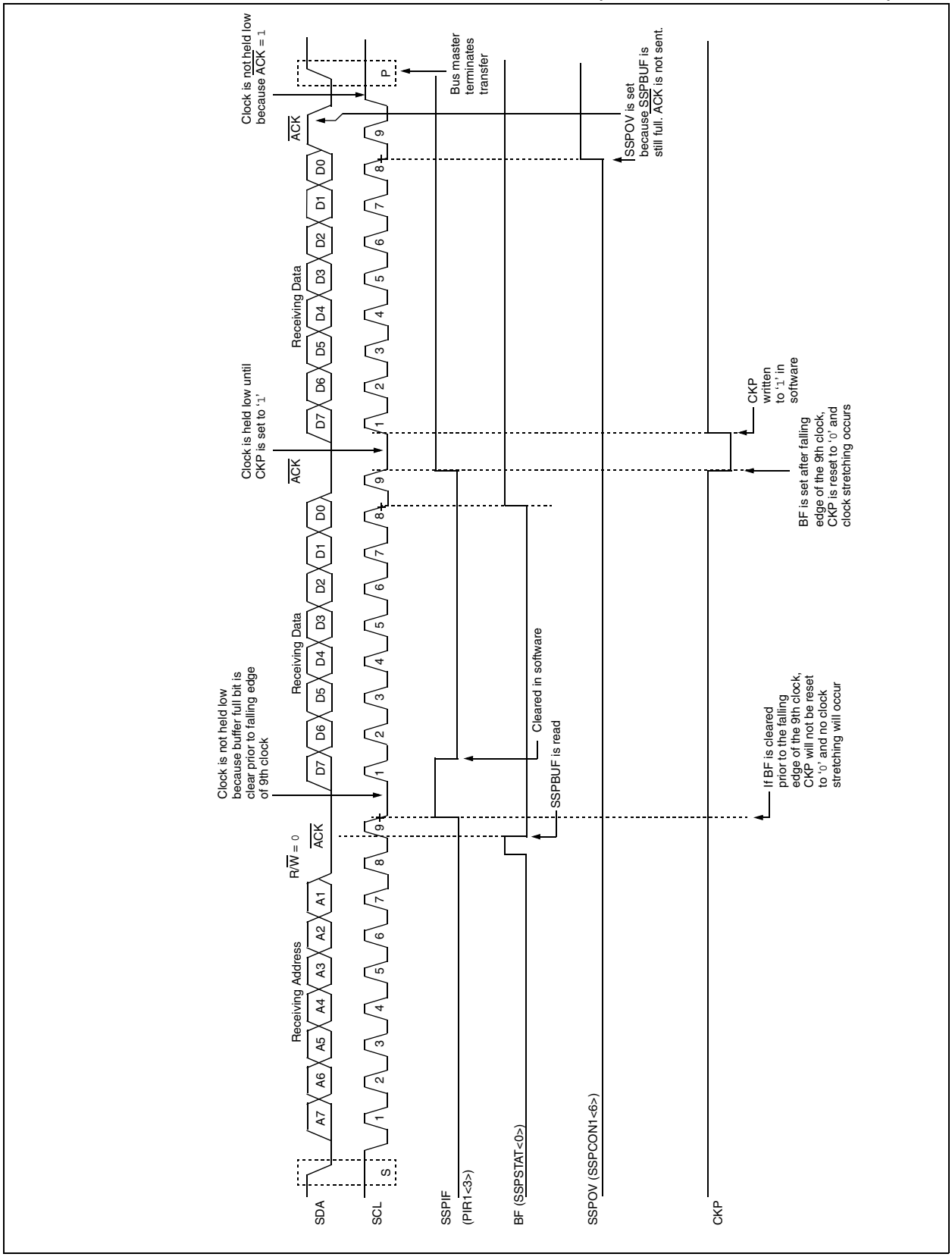
0001 = SPI Master mode, clock = FOSC/16

0000 = SPI Master mode, clock = FOSC/4

Note: Bit combinations not specifically listed here are either reserved or implemented in I^2C mode only.

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

FIGURE 17-13: I²C™ SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 7-BIT ADDRESS)



17.4.6 MASTER MODE

Master mode is enabled by setting and clearing the appropriate SSPM bits in SSPCON1 and by setting the SSPEN bit. In Master mode, the SCL and SDA lines are manipulated by the MSSP hardware.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I^2C bus may be taken when the P bit is set or the bus is Idle, with both the S and P bits clear.

In Firmware Controlled Master mode, user code conducts all I^2C bus operations based on Start and Stop bit conditions.

Once Master mode is enabled, the user has six options.

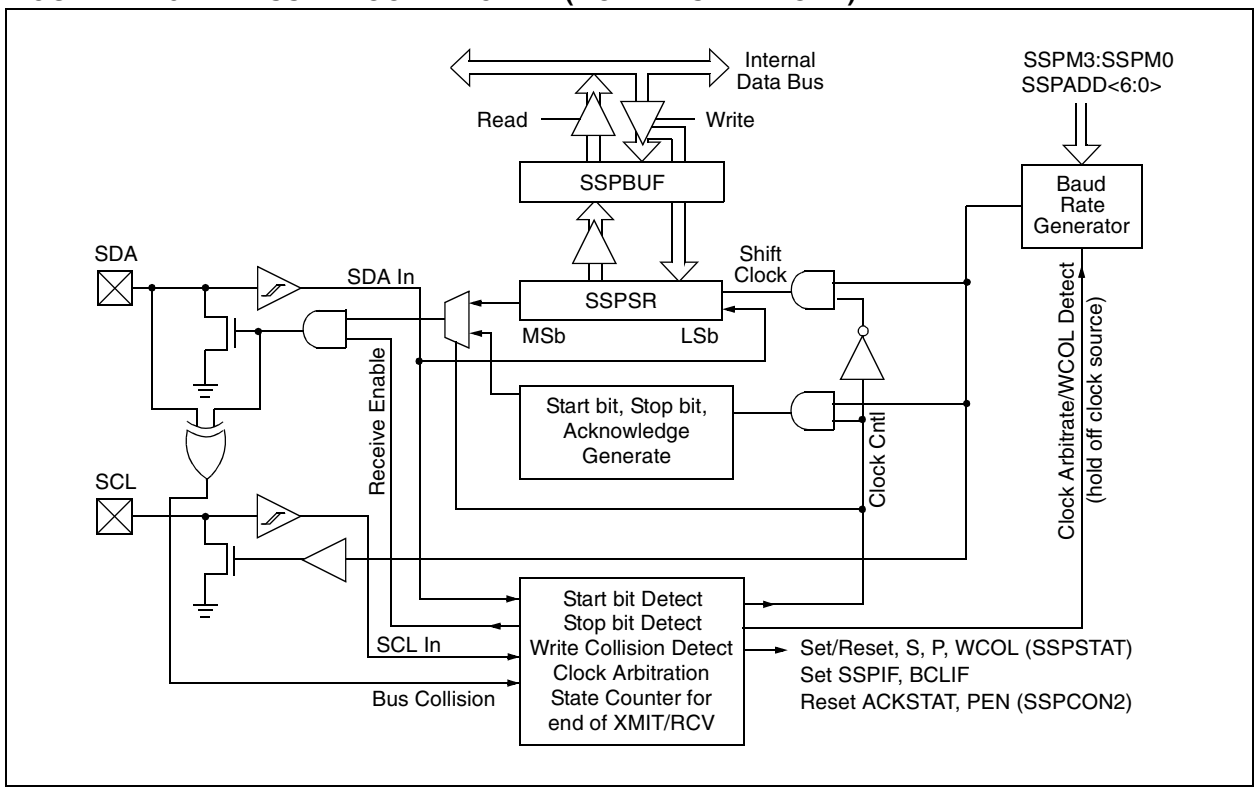
1. Assert a Start condition on SDA and SCL.
2. Assert a Repeated Start condition on SDA and SCL.
3. Write to the SSPBUF register initiating transmission of data/address.
4. Configure the I^2C port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6. Generate a Stop condition on SDA and SCL.

Note: The MSSP module, when configured in I^2C Master mode, does not allow queueing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPBUF register to initiate transmission before the Start condition is complete. In this case, the SSPBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPBUF did not occur.

The following events will cause the SSP Interrupt Flag bit, SSPIF, to be set (SSP interrupt, if enabled):

- Start condition
- Stop condition
- Data transfer byte transmitted/received
- Acknowledge transmit
- Repeated Start

FIGURE 17-16: MSSP BLOCK DIAGRAM (I^2C™ MASTER MODE)



17.4.17.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

a) SDA or SCL are sampled low at the beginning of the Start condition (Figure 17-26).

b) SCL is sampled low before SDA is asserted low (Figure 17-27).

During a Start condition, both the SDA and the SCL pins are monitored.

If the SDA pin is already low, or the SCL pin is already low, then all of the following occur:

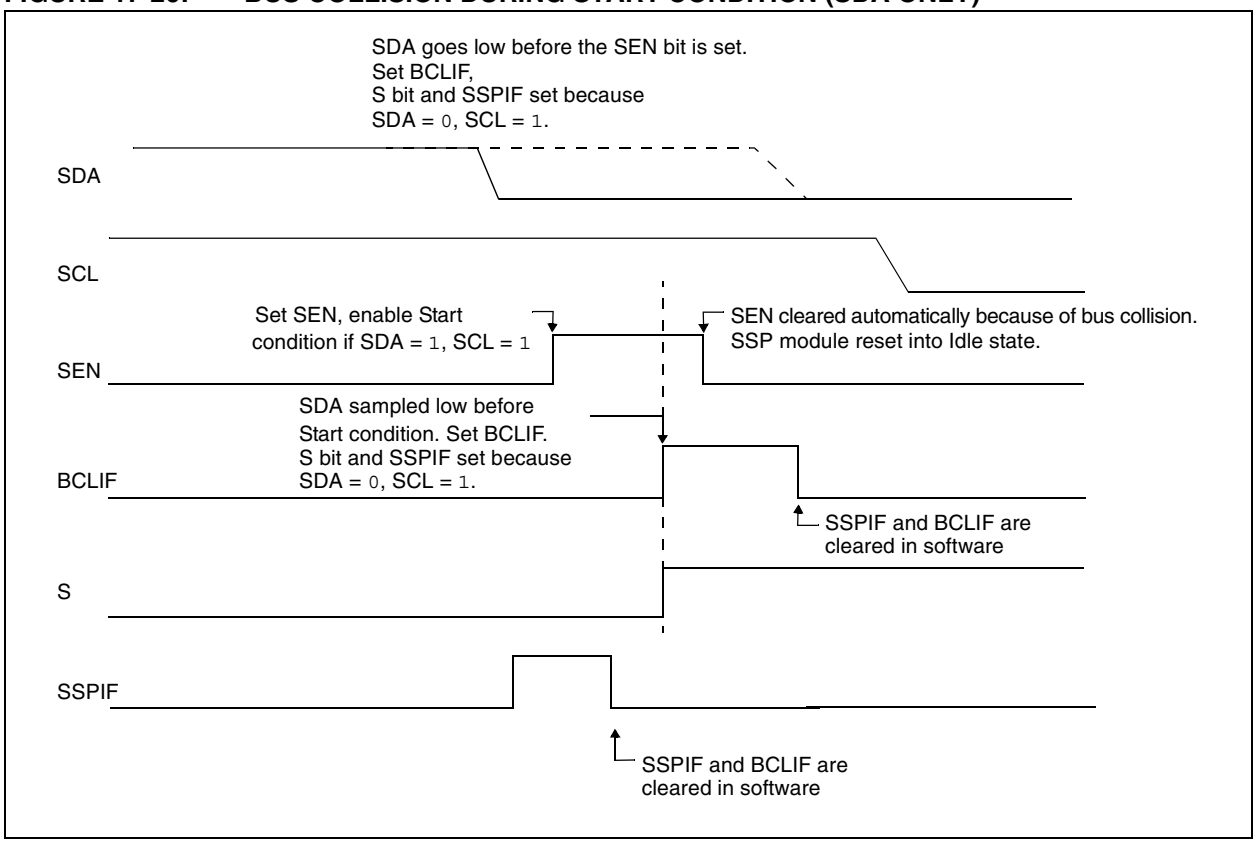
- the Start condition is aborted,
- the BCLIF flag is set; and
- the MSSP module is reset to its Idle state (Figure 17-26).

The Start condition begins with the SDA and SCL pins deasserted. When the SDA pin is sampled high, the Baud Rate Generator is loaded from SSPADD<6:0> and counts down to 0. If the SCL pin is sampled low while SDA is high, a bus collision occurs because it is assumed that another master is attempting to drive a data '1' during the Start condition.

If the SDA pin is sampled low during this count, the BRG is reset and the SDA line is asserted early (Figure 17-28). If, however, a '1' is sampled on the SDA pin, the SDA pin is asserted low at the end of the BRG count. The Baud Rate Generator is then reloaded and counts down to 0 and during this time, if the SCL pins are sampled as '0', a bus collision does not occur. At the end of the BRG count, the SCL pin is asserted low.

Note:	The reason that bus collision is not a factor during a Start condition is that no two bus masters can assert a Start condition at the exact same time. Therefore, one master will always assert SDA before the other. This condition does not cause a bus collision because the two masters must be allowed to arbitrate the first address following the Start condition. If the address is the same, arbitration must be allowed to continue into the data portion, Repeated Start or Stop conditions.

FIGURE 17-26: BUS COLLISION DURING START CONDITION (SDA ONLY)



22.2 HLVD Setup

The following steps are needed to set up the HLVD module:

1. Disable the module by clearing the HLVDEN bit (HLVDCON<4>).
2. Write the value to the HLVDL3:HLVDL0 bits that select the desired HLVD trip point.
3. Set the VDIRMAG bit to detect high voltage (VDIRMAG = 1) or low voltage (VDIRMAG = 0).
4. Enable the HLVD module by setting the HLVDEN bit.
5. Clear the HLVD interrupt flag (PIR2<2>), which may have been set from a previous interrupt.
6. Enable the HLVD interrupt if interrupts are desired by setting the HLVDIE and GIE bits (PIE<2> and INTCON<7>). An interrupt will not be generated until the IRVST bit is set.

22.3 Current Consumption

When the module is enabled, the HLVD comparator and voltage divider are enabled and will consume static current. The total current consumption, when enabled, is specified in electrical specification parameter D022B.

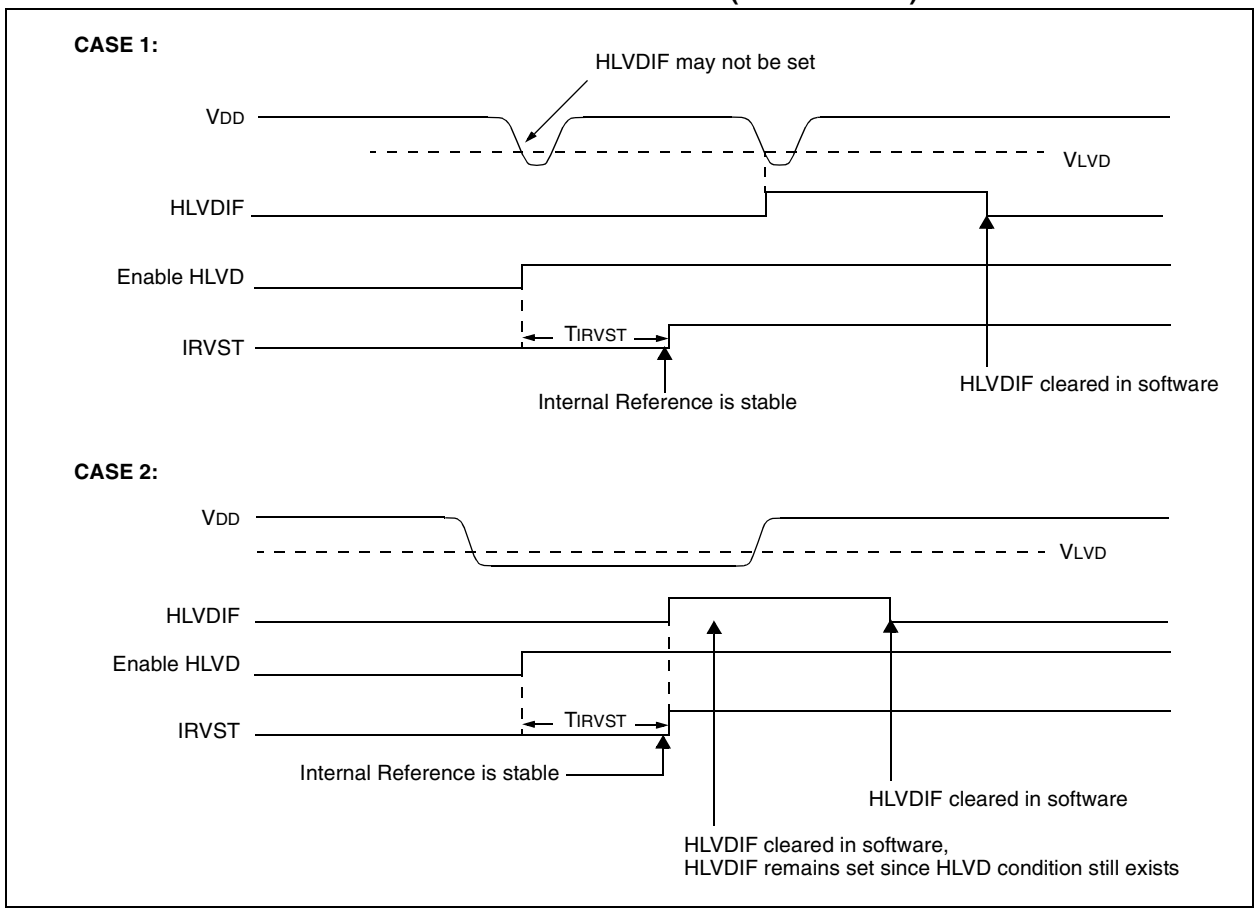
Depending on the application, the HLVD module does not need to be operating constantly. To decrease the current requirements, the HLVD circuitry may only need to be enabled for short periods where the voltage is checked. After doing the check, the HLVD module may be disabled.

22.4 HLVD Start-up Time

The internal reference voltage of the HLVD module, specified in electrical specification parameter D420, may be used by other internal circuitry, such as the Programmable Brown-out Reset. If the HLVD or other circuits using the voltage reference are disabled to lower the device's current consumption, the reference voltage circuit will require time to become stable before a low or high-voltage condition can be reliably detected. This start-up time, TIRVST, is an interval that is independent of device clock speed. It is specified in electrical specification parameter 36.

The HLVD interrupt flag is not enabled until TIRVST has expired and a stable reference voltage is reached. For this reason, brief excursions beyond the set point may not be detected during this interval. Refer to Figure 22-2 or Figure 22-3.

FIGURE 22-2: LOW-VOLTAGE DETECT OPERATION (VDIRMAG = 0)



23.2.3 DEDICATED CAN RECEIVE BUFFER REGISTERS

This section shows the dedicated CAN Receive Buffer registers with their associated control registers.

REGISTER 23-13: RXB0CON: RECEIVE BUFFER 0 CONTROL REGISTER

Mode 0

R/C-0	R/W-0	R/W-0	U-0	R-0	R/W-0	R-0	R-0
RXFUL(1)	RXM1	RXM0	—	RXRTRRO	RXB0DBEN	JTOFF(2)	FILHIT0

Mode 1, 2

R/C-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
RXFUL(1)	RXM1	RTRRO	FILHIT4	FILHIT3	FILHIT2	FILHIT1	FILHIT0
bit 7							bit 0

bit 7 **RXFUL:** Receive Full Status bit(1)
1 = Receive buffer contains a received message
0 = Receive buffer is open to receive a new message

bit 6 Mode 0:
RXM1: Receive Buffer Mode bit 1
Combines with RXM0 to form RXM<1:0> bits (see bit 5).
11 = Receive all messages (including those with errors); filter criteria is ignored
10 = Receive only valid messages with extended identifier; EXIDEN in RXFnSIDL must be '1'
01 = Receive only valid messages with standard identifier; EXIDEN in RXFnSIDL must be '0'
00 = Receive all valid messages as per EXIDEN bit in RXFnSIDL register
Mode 1, 2:
RXM1: Receive Buffer Mode bit 1
1 = Receive all messages (including those with errors); acceptance filters are ignored
0 = Receive all valid messages as per acceptance filters

bit 5 Mode 0:
RXM0: Receive Buffer Mode bit 0
Combines with RXM1 to form RXM<1:0> bits (see bit 6).
Mode 1, 2:
RTRRO: Remote Transmission Request bit for Received Message (read-only)
1 = A remote transmission request is received
0 = A remote transmission request is not received

bit 4 Mode 0:
Unimplemented: Read as '0'
Mode 1, 2:
FILHIT4: Filter Hit bit 4
This bit combines with other bits to form filter acceptance bits <4:0>.

bit 3 Mode 0:
RXRTRRO: Remote Transmission Request bit for Received Message (read-only)
1 = A remote transmission request is received
0 = A remote transmission request is not received
Mode 1, 2:
FILHIT3: Filter Hit bit 3
This bit combines with other bits to form filter acceptance bits <4:0>.

REGISTER 23-48: MSEL0: MASK SELECT REGISTER 0(1)

R/W-0	R/W-1	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
FIL3_1	FIL3_0	FIL2_1	FIL2_0	FIL1_1	FIL1_0	FIL0_1	FIL0_0
bit 7							bit 0

bit 7-6 **FIL3_1:FIL3_0:** Filter 3 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 5-4 **FIL2_1:FIL2_0:** Filter 2 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 3-2 **FIL1_1:FIL1_0:** Filter 1 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 1-0 **FIL0_1:FIL0_0:** Filter 0 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

Note 1: This register is available in Mode 1 and 2 only.

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

23.10 Synchronization

To compensate for phase shifts between the oscillator frequencies of each of the nodes on the bus, each CAN controller must be able to synchronize to the relevant signal edge of the incoming signal. When an edge in the transmitted data is detected, the logic will compare the location of the edge to the expected time (Sync_Seg). The circuit will then adjust the values of Phase Segment 1 and Phase Segment 2 as necessary. There are two mechanisms used for synchronization.

23.10.1 HARD SYNCHRONIZATION

Hard synchronization is only done when there is a recessive to dominant edge during a bus Idle condition, indicating the start of a message. After hard synchronization, the bit time counters are restarted with Sync_Seg. Hard synchronization forces the edge which has occurred to lie within the synchronization segment of the restarted bit time. Due to the rules of synchronization, if a hard synchronization occurs, there will not be a resynchronization within that bit time.

23.10.2 RESYNCHRONIZATION

As a result of resynchronization, Phase Segment 1 may be lengthened or Phase Segment 2 may be shortened. The amount of lengthening or shortening of the phase buffer segments has an upper bound given by the Synchronization Jump Width (SJW). The value of the SJW will be added to Phase Segment 1 (see Figure 23-6) or subtracted from Phase Segment 2 (see Figure 23-7). The SJW is programmable between 1 TQ and 4 TQ.

Clocking information will only be derived from recessive to dominant transitions. The property, that only a fixed maximum number of successive bits have the same value, ensures resynchronization to the bit stream during a frame.

The phase error of an edge is given by the position of the edge relative to Sync_Seg, measured in TQ. The phase error is defined in magnitude of TQ as follows:

- e = 0 if the edge lies within Sync_Seg.
- e > 0 if the edge lies before the sample point.
- e < 0 if the edge lies after the sample point of the previous bit.

If the magnitude of the phase error is less than, or equal to, the programmed value of the Synchronization Jump Width, the effect of a resynchronization is the same as that of a hard synchronization.

If the magnitude of the phase error is larger than the Synchronization Jump Width and if the phase error is positive, then Phase Segment 1 is lengthened by an amount equal to the Synchronization Jump Width.

If the magnitude of the phase error is larger than the resynchronization jump width and if the phase error is negative, then Phase Segment 2 is shortened by an amount equal to the Synchronization Jump Width.

23.10.3 SYNCHRONIZATION RULES

- Only one synchronization within one bit time is allowed.
- An edge will be used for synchronization only if the value detected at the previous sample point (previously read bus value) differs from the bus value immediately after the edge.
- All other recessive to dominant edges fulfilling rules 1 and 2 will be used for resynchronization, with the exception that a node transmitting a dominant bit will not perform a resynchronization as a result of a recessive to dominant edge with a positive phase error.

DAW — Decimal Adjust W Register

Syntax: DAW

Operands: None

Operation: If [W<3:0> >9] or [DC = 1] then
(W<3:0>) + 6 → W<3:0>;
else
(W<3:0>) → W<3:0>;

If [W<7:4> >9] or [C = 1] then
(W<7:4>) + 6 → W<7:4>;
C = 1;
else
(W<7:4>) → W<7:4>;

Status Affected: C

Encoding:

0000	0000	0000	0111

Description: DAW adjusts the eight-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register W	Process Data	Write W

Example 1:

```
DAW
```

Before Instruction
W = A5h
C = 0
DC = 0
After Instruction
W = 05h
C = 1
DC = 0

Example 2:

Before Instruction
W = CEh
C = 0
DC = 0
After Instruction
W = 34h
C = 1
DC = 0

DECF — Decrement f

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

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f) – 1 → dest

Status Affected: C, DC, N, OV, Z

Encoding:

0000	01da	ffff	ffff

Description: Decrement 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 ≤ 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: DECF CNT, 1, 0

Before Instruction
CNT = 01h
Z = 0
After Instruction
CNT = 00h
Z = 1

SUBLW **Subtract W from Literal**

Syntax: SUBLW k

Operands: $0 \le k \le 255$

Operation: $k - (W) \rightarrow W$

Status Affected: N, OV, C, DC, Z

Encoding:

0000	1000	kkkk	kkkk

Description: W is subtracted from the eight-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example 1:
```
SUBLW   02h
```
Before Instruction
W = 01h
C = ?
After Instruction
W = 01h
C = 1 ; result is positive
Z = 0
N = 0

Example 2:
```
SUBLW   02h
```
Before Instruction
W = 02h
C = ?
After Instruction
W = 00h
C = 1 ; result is zero
Z = 1
N = 0

Example 3:
```
SUBLW   02h
```
Before Instruction
W = 03h
C = ?
After Instruction
W = FFh; (2's complement)
C = 0 ; result is negative
Z = 0
N = 1

SUBWF **Subtract W from f**

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

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

Operation: $(f) - (W) \rightarrow dest$

Status Affected: N, OV, C, DC, Z

Encoding:

0101	11da	ffff	ffff

Description: Subtract W from register 'f' (2's complement method). 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 1:
```
SUBWF   REG, 1, 0
```
Before Instruction
REG = 3
W = 2
C = ?
After Instruction
REG = 1
W = 2
C = 1 ; result is positive
Z = 0
N = 0

Example 2:
```
SUBWF   REG, 0, 0
```
Before Instruction
REG = 2
W = 2
C = ?
After Instruction
REG = 2
W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3:
```
SUBWF   REG, 1, 0
```
Before Instruction
REG = 1
W = 2
C = ?
After Instruction
REG = FFh ;(2's complement)
W = 2
C = 0 ; result is negative
Z = 0
N = 1

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 nn=0 * =1 *+ =2 *- =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

TBLWT Table Write

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

Operands: None

Operation: if TBLWT*,
(TABLAT) → Holding Register;
TBLPTR – No Change;
if TBLWT*+,
(TABLAT) → Holding Register;
(TBLPTR) + 1 → TBLPTR;
if TBLWT*-,
(TABLAT) → Holding Register;
(TBLPTR) – 1 → TBLPTR;
if TBLWT+*,
(TBLPTR) + 1 → TBLPTR;
(TABLAT) → Holding Register;

Status Affected: None

Encoding:

0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*

Description: This instruction uses the 3 LSBs of the TBLPTR to determine which of the 8 holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to **Section 6.0 "Flash Program Memory"** for additional details on programming Flash memory.)

The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-MBtye address range. The LSb of the TBLPTR selects which byte of the program memory location to access.

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

The TBLWT 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 TABLAT)	No operation	No operation (Write to Holding Register)

TBLWT Table Write (Continued)

Example 1: TBLWT *+;

Before Instruction
TABLAT = 55h
TBLPTR = 00A356h
HOLDING REGISTER (00A356h) = FFh
After Instructions (table write completion)
TABLAT = 55h
TBLPTR = 00A357h
HOLDING REGISTER (00A356h) = 55h

Example 2: TBLWT +*;

Before Instruction
TABLAT = 34h
TBLPTR = 01389Ah
HOLDING REGISTER (01389Ah) = FFh
HOLDING REGISTER (01389Bh) = FFh
After Instruction (table write completion)
TABLAT = 34h
TBLPTR = 01389Bh
HOLDING REGISTER (01389Ah) = FFh
HOLDING REGISTER (01389Bh) = 34h

26.7 MPLAB ICE 2000 High-Performance In-Circuit Emulator

The MPLAB ICE 2000 In-Circuit Emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PIC microcontrollers. Software control of the MPLAB ICE 2000 In-Circuit Emulator is advanced by the MPLAB Integrated Development Environment, which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICE 2000 is a full-featured emulator system with enhanced trace, trigger and data monitoring features. Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The architecture of the MPLAB ICE 2000 In-Circuit Emulator allows expansion to support new PIC microcontrollers.

The MPLAB ICE 2000 In-Circuit Emulator system has been designed as a real-time emulation system with advanced features that are typically found on more expensive development tools. The PC platform and Microsoft® Windows® 32-bit operating system were chosen to best make these features available in a simple, unified application.

26.8 MPLAB REAL ICE In-Circuit Emulator System

MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC® and MCU devices. It debugs and programs PIC® and dsPIC® Flash microcontrollers with the easy-to-use, powerful graphical user interface of the MPLAB Integrated Development Environment (IDE), included with each kit.

The MPLAB REAL ICE probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with the popular MPLAB ICD 2 system (RJ11) or with the new high speed, noise tolerant, low-voltage differential signal (LVDS) interconnection (CAT5).

MPLAB REAL ICE is field upgradeable through future firmware downloads in MPLAB IDE. In upcoming releases of MPLAB IDE, new devices will be supported, and new features will be added, such as software breakpoints and assembly code trace. MPLAB REAL ICE offers significant advantages over competitive emulators including low-cost, full-speed emulation, real-time variable watches, trace analysis, complex breakpoints, a ruggedized probe interface and long (up to three meters) interconnection cables.

26.9 MPLAB ICD 2 In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB ICD 2, is a powerful, low-cost, run-time development tool, connecting to the host PC via an RS-232 or high-speed USB interface. This tool is based on the Flash PIC MCUs and can be used to develop for these and other PIC MCUs and dsPIC DSCs. The MPLAB ICD 2 utilizes the in-circuit debugging capability built into the Flash devices. This feature, along with Microchip's In-Circuit Serial Programming™ (ICSP™) protocol, offers cost-effective, in-circuit Flash debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by setting breakpoints, single stepping and watching variables, and CPU status and peripheral registers. Running at full speed enables testing hardware and applications in real time. MPLAB ICD 2 also serves as a development programmer for selected PIC devices.

26.10 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 SD/MMC card for file storage and secure data applications.