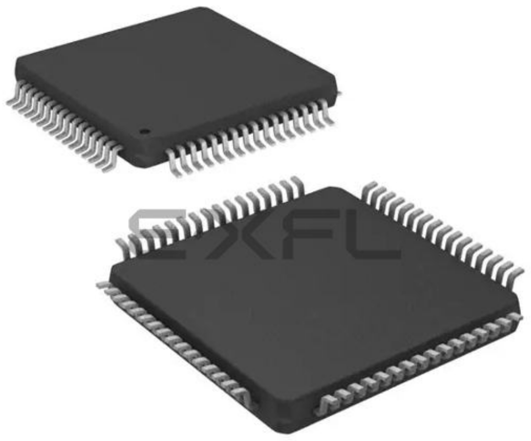


Welcome to E-XFL.COM

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	64MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	60
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.5K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 45x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f66k40t-i-pt

3.2 Register Definitions: Configuration Words

REGISTER 3-1: Configuration Word 1L (30 0000h): Oscillators

U-1	R/W-1	R/W-1	R/W-1	U-1	R/W-1	R/W-1	R/W-1
—	RSTOSC<2:0>			—	FEXTOSC<2:0>		
bit 7							bit 0

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

bit 7	**Unimplemented:** Read as ‘1’
bit 6-4	**RSTOSC<2:0>:** Power-up Default Value for COSC bits This value is the Reset default value for COSC and selects the oscillator first used by user software. Refer to COSC operation. 111 = EXTOSC operating per FEXTOSC bits (device manufacturing default) 110 = HFINTOSC with HFFRQ = 4 MHz (Register 4-5) and CDIV = 4:1 (Register 4-2) 101 = LFINTOSC 100 = SOSC 011 = Reserved 010 = EXTOSC with 4x PLL, with EXTOSC operating per FEXTOSC bits 001 = Reserved 000 = HFINTOSC with HFFRQ = 64 MHz (Register 4-5) and CDIV = 1:1 (Register 4-2). Resets COSC/NOSC to 3'b110.
bit 3	**Unimplemented:** Read as ‘1’
bit 2-0	**FEXTOSC<2:0>:** FEXTOSC External Oscillator Mode Selection bits 111 = EC (external clock) above 8 MHz; PFM set to high power (device manufacturing default) 110 = EC (external clock) for 500 kHz to 8 MHz; PFM set to medium power 101 = EC (external clock) below 500 kHz; PFM set to low power 100 = Oscillator not enabled 011 = Reserved (do not use) 010 = HS (crystal oscillator) above 8 MHz; PFM set to high power 001 = XT (crystal oscillator) above 500 kHz, below 8 MHz; PFM set to medium power 000 = LP (crystal oscillator) optimized for 32.768 kHz; PFM set to low power

REGISTER 4-3: OSCCON3: OSCILLATOR CONTROL REGISTER 3

R/W/HC-0/0	R/W-0/0	U-0	R-0/0	R-0/0	U-0	U-0	U-0
CSWHOLD	SOSCPWR	—	ORDY	NOSCR	—	—	—
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	HC = Bit is cleared by hardware

bit 7	**CSWHOLD:** Clock Switch Hold bit 1 = Clock switch will hold (with interrupt) when the oscillator selected by NOSC is ready 0 = Clock switch may proceed when the oscillator selected by NOSC is ready; NOSCR becomes ‘1’, the switch will occur
bit 6	**SOSCPWR:** Secondary Oscillator Power Mode Select bit 1 = Secondary oscillator operating in High-Power mode 0 = Secondary oscillator operating in Low-Power mode
bit 5	**Unimplemented:** Read as ‘0’
bit 4	**ORDY:** Oscillator Ready bit (read-only) 1 = OSCCON1 = OSCCON2; the current system clock is the clock specified by NOSC 0 = A clock switch is in progress
bit 3	**NOSCR:** New Oscillator is Ready bit (read-only)$^{(1)}$ 1 = A clock switch is in progress and the oscillator selected by NOSC indicates a “ready” condition 0 = A clock switch is not in progress, or the NOSC-selected oscillator is not yet ready
bit 2-0	**Unimplemented:** Read as ‘0’

Note 1: If CSWHOLD = 0, the user may not see this bit set because, when the oscillator becomes ready there may be a delay of one instruction clock before this bit is set. The clock switch occurs in the next instruction cycle and this bit is cleared.

REGISTER 6-2: CPUDOZE: DOZE AND IDLE REGISTER

R/W-0/u	R/W/HC/HS-0/0	R/W-0/0	R/W-0/0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
IDLEN	DOZEN	ROI	DOE	—	DOZE<2:0>		
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	HC = Bit is cleared by hardware

bit 7 **IDLEN:** Idle Enable bit
`1` = A `SLEEP` instruction inhibits the CPU clock, but not the peripheral clock(s)
`0` = A `SLEEP` instruction places the device into full Sleep mode

bit 6 **DOZEN:** Doze Enable bit[(1,2)]
`1` = The CPU executes instruction cycles according to DOZE setting
`0` = The CPU executes all instruction cycles (fastest, highest power operation)

bit 5 **ROI:** Recover-On-Interrupt bit
`1` = Entering the Interrupt Service Routine (ISR) makes DOZEN = `0` bit, bringing the CPU to full-speed operation
`0` = Interrupt entry does not change DOZEN

bit 4 **DOE:** Doze-On-Exit bit
`1` = Executing RETFIE makes DOZEN = `1`, bringing the CPU to reduced speed operation
`0` = RETFIE does not change DOZEN

bit 3 **Unimplemented:** Read as ‘`0`’

bit 2-0 **DOZE<2:0>:** Ratio of CPU Instruction Cycles to Peripheral Instruction Cycles
`111` =1:256
`110` =1:128
`101` =1:64
`100` =1:32
`011` =1:16
`010` =1:8
`001` =1:4
`000` =1:2

Note 1: When ROI = `1` or DOE = `1`, DOZEN is changed by hardware interrupt entry and/or exit.
2: Entering ICD overrides DOZEN, returning the CPU to full execution speed; this bit is not affected.

7.5 Register Definitions: Peripheral Module Disable

REGISTER 7-1: PMD0: PMD CONTROL REGISTER 0

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
SYSCMD	FVRMD	HLVDMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD
7							0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	q = Value depends on condition

bit 7 **SYSCMD:** Disable Peripheral System Clock Network bit(1)
See description in **Section 7.4 “System Clock Disable”**.
1 = System clock network disabled (FOSC)
0 = System clock network enabled

bit 6 **FVRMD:** Disable Fixed Voltage Reference bit
1 = FVR module disabled
0 = FVR module enabled

bit 5 **HLVDMD:** Disable Low-Voltage Detect bit
1 = HLVD module disabled
0 = HLVD module enabled

bit 4 **CRCMD:** Disable CRC Engine bit
1 = CRC module disabled
0 = CRC module enabled

bit 3 **SCANMD:** Disable NVM Memory Scanner bit(2)
1 = NVM Memory Scan module disabled
0 = NVM Memory Scan module enabled

bit 2 **NVMMD:** NVM Module Disable bit(3)
1 = All Memory reading and writing is disabled; NVMCON registers cannot be written
0 = NVM module enabled

bit 1 **CLKRMD:** Disable Clock Reference bit
1 = CLKR module disabled
0 = CLKR module enabled

bit 0 **IOCMD:** Disable Interrupt-on-Change bit, All Ports
1 = IOC module(s) disabled
0 = IOC module(s) enabled

Note 1: Clearing the SYSCMD bit disables the system clock (FOSC) to peripherals, however peripherals clocked by FOSC/4 are not affected.
2: Subject to SCANE bit in CONFIG4H.
3: When enabling NVM, a delay of up to 1 μs may be required before accessing data.

9.2 Independent Clock Source

The WWDT can derive its time base from either the 31 kHz LFINTOSC or 31.25 kHz MFINTOSC internal oscillators, depending on the value of WDTE<1:0> Configuration bits.

If WDTE = `2'b1x`, then the clock source will be enabled depending on the WDTCCS<2:0> Configuration bits.

If WDTE = `2'b01`, the SEN bit should be set by software to enable WWDT, and the clock source is enabled by the WDTCS bits in the WDTCON1 register.

Time intervals in this chapter are based on a minimum nominal interval of 1 ms. See **Section 37.0 "Electrical Specifications"** for LFINTOSC and MFINTOSC tolerances.

9.3 WWDT Operating Modes

The Windowed Watchdog Timer module has four operating modes controlled by the WDTE<1:0> bits in Configuration Words. See Table 9-1.

9.3.1 WWDT IS ALWAYS ON

When the WDTE bits of Configuration Words are set to '`11`', the WWDT is always on.

WWDT protection is active during Sleep.

9.3.2 WWDT IS OFF IN SLEEP

When the WDTE bits of Configuration Words are set to '`10`', the WWDT is on, except in Sleep.

WWDT protection is not active during Sleep.

9.3.3 WWDT CONTROLLED BY SOFTWARE

When the WDTE bits of Configuration Words are set to '`01`', the WWDT is controlled by the SEN bit of the WDTCON0 register.

WWDT protection is unchanged by Sleep. See Table 9-1 for more details.

TABLE 9-1: WWDT OPERATING MODES

WDTE<1:0>	SEN	Device Mode	WWDT Mode
11	X	X	Active
10	X	Awake	Active
		Sleep	Disabled
01	1	X	Active
	0	X	Disabled
00	X	X	Disabled

9.4 Time-out Period

If the WDTCPS<4:0> Configuration bits default to `5'b11111`, then the WDTPS bits of the WDTCON0 register set the time-out period from 1 ms to 256 seconds (nominal). If any value other than the default value is assigned to WDTCPS<4:0> Configuration bits, then the timer period will be based on the WDTCPS<4:0> bits in the CONFIG3L register. After a Reset, the default time-out period is 2s.

9.5 Watchdog Window

The Windowed Watchdog Timer has an optional Windowed mode that is controlled by the WDTCWS<2:0> Configuration bits and WINDOW<2:0> bits of the WDTCON1 register. In the Windowed mode, the `CLRWDT` instruction must occur within the allowed window of the WDT period. Any `CLRWDT` instruction that occurs outside of this window will trigger a window violation and will cause a WWDT Reset, similar to a WWDT time out. See Figure 9-2 for an example.

The window size is controlled by the WINDOW<2:0> Configuration bits, or the WINDOW<2:0> bits of WDTCON1, if WDTCWS<2:0> = `111`.

The five Most Significant bits of the WDTTMR register are used to determine whether the window is open, as defined by the WINDOW<2:0> bits of the WDTCON1 register.

In the event of a window violation, a Reset will be generated and the $\overline{\text{WDTWV}}$ bit of the PCON0 register will be cleared. This bit is set by a POR or can be set in firmware.

9.6 Clearing the WWDT

The WWDT is cleared when any of the following conditions occur:

- Any Reset
- Valid `CLRWDT` instruction is executed
- Device enters Sleep
- Exit Sleep by Interrupt
- WWDT is disabled
- Oscillator Start-up Timer (OST) is running
- Any write to the WDTCON0 or WDTCON1 registers

9.6.1 `CLRWDT` CONSIDERATIONS (WINDOWED MODE)

When in Windowed mode, the WWDT must be armed before a `CLRWDT` instruction will clear the timer. This is performed by reading the WDTCON0 register. Executing a `CLRWDT` instruction without performing such an arming action will trigger a window violation regardless of whether the window is open or not.

See Table 9-2 for more information.

Operations on the FSRs with POSTDEC, POSTINC and PREINC affect the entire register pair; that is, roll-overs of the FSRnL register from FFh to 00h carry over to the FSRnH register. On the other hand, results of these operations do not change the value of any flags in the STATUS register (e.g., Z, N, OV, etc.).

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

10.6.3.3 Operations by FSRs on FSRs

Indirect addressing operations that target other FSRs or virtual registers represent special cases. For example, using an FSR to point to one of the virtual registers will not result in successful operations. As a specific case, assume that FSR0H:FSR0L contains FE7h, the address of INDF1. Attempts to read the value of the INDF1 using INDF0 as an operand will return 00h. Attempts to write to INDF1 using INDF0 as the operand will result in a `NOP`.

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

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

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

10.7 Data Memory and the Extended Instruction Set

Enabling the PIC18 extended instruction set (XINST Configuration bit = `1`) significantly changes certain aspects of data memory and its addressing. Specifically, the use of the Access Bank for many of the core PIC18 instructions is different; this is due to the introduction of a new addressing mode for the data memory space.

What does not change is just as important. The size of the data memory space is unchanged, as well as its linear addressing. The SFR map remains the same. Core PIC18 instructions can still operate in both Direct and Indirect Addressing mode; inherent and literal instructions do not change at all. Indirect addressing with FSR0 and FSR1 also remain unchanged.

10.7.1 INDEXED ADDRESSING WITH LITERAL OFFSET

Enabling the PIC18 extended instruction set changes the behavior of indirect addressing using the FSR2 register pair within Access RAM. Under the proper conditions, instructions that use the Access Bank – that is, most bit-oriented and byte-oriented instructions – can invoke a form of indexed addressing using an offset specified in the instruction. This special addressing mode is known as Indexed Addressing with Literal Offset, or Indexed Literal Offset mode.

When using the extended instruction set, this addressing mode requires the following:

- The use of the Access Bank is forced (‘a’ = `0`) and
- The file address argument is less than or equal to 5Fh.

Under these conditions, the file address of the instruction is not interpreted as the lower byte of an address (used with the BSR in direct addressing), or as an 8-bit address in the Access Bank. Instead, the value is interpreted as an offset value to an Address Pointer, specified by FSR2. The offset and the contents of FSR2 are added to obtain the target address of the operation.

10.7.2 INSTRUCTIONS AFFECTED BY INDEXED LITERAL OFFSET MODE

Any of the core PIC18 instructions that can use direct addressing are potentially affected by the Indexed Literal Offset Addressing mode. This includes all byte-oriented and bit-oriented instructions, or almost one-half of the standard PIC18 instruction set. Instructions that only use Inherent or Literal Addressing modes are unaffected.

Additionally, byte-oriented and bit-oriented instructions are not affected if they do not use the Access Bank (Access RAM bit is ‘`1`’), or include a file address of 60h or above. Instructions meeting these criteria will continue to execute as before. A comparison of the different possible addressing modes when the extended instruction set is enabled is shown in Figure 10-7.

Those who desire to use byte-oriented or bit-oriented instructions in the Indexed Literal Offset mode should note the changes to assembler syntax for this mode. This is described in more detail in **Section 36.2.1 “Extended Instruction Syntax”**.

REGISTER 14-3: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0
OSCFIF	CSWIF[(1)]	—	—	—	—	ADTIF	ADIF
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	**OSCFIF:** Oscillator Fail Interrupt Flag bit `1` = Device oscillator failed, clock input has changed to HFINTOSC (must be cleared by software) `0` = Device clock operating
bit 6	**CSWIF:** Clock-Switch Interrupt Flag bit[(1)] `1` = New oscillator is ready for switch (must be cleared by software) (see Figure 4-6 and Figure 4-7) `0` = New oscillator is not ready for switch or has not been started
bit 5-2	**Unimplemented:** Read as ‘`0`’
bit 1	**ADTIF:** ADC Threshold Interrupt Flag bit `1` = ADC Threshold interrupt has occurred (must be cleared by software) `0` = ADC Threshold event is not complete or has not been started
bit 0	**ADIF:** ADC Interrupt Flag bit `1` = An A/D conversion completed (must be cleared by software) `0` = The A/D conversion is not complete or has not been started

Note 1: The CSWIF interrupt will not wake the system from Sleep. The system will sleep until another interrupt causes the wake-up.

REGISTER 14-25: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
RC2IP	TX2IP	RC1IP	TX1IP	BCL2IP	SSP2IP	BCL1IP	SSP1IP
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	**RC2IP:** EUSART2 Receive Interrupt Priority bit `1` = High priority `0` = Low priority
bit 6	**TX2IP:** EUSART2 Transmit Interrupt Priority bit `1` = High priority `0` = Low priority
bit 5	**RC1IP:** EUSART1 Receive Interrupt Priority bit `1` = High priority `0` = Low priority
bit 4	**TX1IP:** EUSART1 Transmit Interrupt Priority bit `1` = High priority `0` = Low priority
bit 3	**BCL2IP:** MSSP2 Bus Collision Interrupt Priority bit `1` = High priority `0` = Low priority
bit 2	**SSP2IP:** Synchronous Serial Port 2 Interrupt Priority bit `1` = High priority `0` = Low priority
bit 1	**BCL1IP:** MSSP1 Bus Collision Interrupt Priority bit `1` = High priority `0` = Low priority
bit 0	**SSP1IP:** Synchronous Serial Port 1 Interrupt Priority bit `1` = High priority `0` = Low priority

17.0 PERIPHERAL PIN SELECT (PPS) MODULE

The Peripheral Pin Select (PPS) module connects peripheral inputs and outputs to the device I/O pins. Only digital signals are included in the selections. All analog inputs and outputs remain fixed to their assigned pins. Input and output selections are independent as shown in the simplified block diagram Figure 17-1.

The peripheral input is selected with the peripheral xxxPPS register (Register 17-1), and the peripheral output is selected with the PORT RxyPPS register (Register 17-2). For example, to select PORTC<7> as the EUSART1 RX input, set RXxPPS to `6’b01 0111`, and to select PORTC<6> as the EUSART1 TX output set RC6PPS to `6'b00 1100`.

17.1 PPS Inputs

Each peripheral has a PPS register with which the inputs to the peripheral are selected. Inputs include the device pins.

Multiple peripherals can operate from the same source simultaneously. Port reads always return the pin level regardless of peripheral PPS selection. If a pin also has analog functions associated, the ANSEL bit for that pin must be cleared to enable the digital input buffer.

Although every peripheral has its own PPS input selection register, the selections are identical for every peripheral as shown in Register 17-1.

Note: The notation “xxx” in the register name is a place holder for the peripheral identifier. For example, INT0PPS.

17.2 PPS Outputs

Each I/O pin has a PPS register with which the pin output source is selected. With few exceptions, the port TRIS control associated with that pin retains control over the pin output driver. Peripherals that control the pin output driver as part of the peripheral operation will override the TRIS control as needed. These peripherals include:

- EUSART (synchronous operation)
- MSSP (I^{2}C)

Although every pin has its own PPS peripheral selection register, the selections are identical for every pin as shown in Register 17-2.

Note: The notation “Rxy” is a place holder for the pin identifier. For example, RA0PPS.

19.0 TIMER1/3/5/7 MODULE WITH GATE CONTROL

Timer1/3/5/7 module is a 16-bit timer/counter with the following features:

- 16-bit timer/counter register pair (TMRxH:TMRxL)
- Programmable internal or external clock source
- 2-bit prescaler
- Dedicated Secondary 32 kHz oscillator circuit
- Optionally synchronized comparator out
- Multiple Timer1/3/5/7 gate (count enable) sources
- Interrupt on overflow
- Wake-up on overflow (external clock, Asynchronous mode only)
- 16-Bit Read/Write Operation
- Time base for the Capture/Compare function with the CCP modules
- Special Event Trigger (with CCP)
- Selectable Gate Source Polarity
- Gate Toggle mode
- Gate Single-pulse mode
- Gate Value Status
- Gate Event Interrupt

Figure 19-1 is a block diagram of the Timer1/3/5/7 module.

EQUATION 23-2: R-C CALCULATIONS

VPEAK = External voltage source peak voltage
f = External voltage source frequency
C = Series capacitor
R = Series resistor
V_C = Peak capacitor voltage
Φ = Capacitor induced zero crossing phase advance in radians
T_Φ = Time ZC event occurs before actual zero crossing

$$Z = \frac{V_{PEAK}}{3 \times 10^{-4}}$$

$$X_C = \frac{1}{2\pi fC}$$

$$R = \sqrt{Z^2 - X_C^2}$$

$$V_C = X_C(3 \times 10^{-4})$$

$$\Phi = \mathrm{Tan}^{-1}\left(\frac{X_C}{R}\right)$$

$$T\Phi = \frac{\Phi}{2\pi f}$$

EXAMPLE 23-1: R-C CALCULATIONS

VRMS = 120
VPEAK = VRMS *√2 = 169.7
f = 60 Hz
C = 0.1 μF

$$Z = \frac{V_{PEAK}}{3 \times 10^{-4}} = \frac{169.7}{3 \times 10^{-4}} = 565.7\text{k}\Omega$$

$$X_C = \frac{1}{2\pi fC} = \frac{1}{(2\pi \times 60 \times 1 \times 10^{-7})} = 26.53\text{k}\Omega$$

$$R = \sqrt{(Z^2 \times X_C^2)} = 565.1\text{k}\Omega \text{ (computed)}$$

$$R = 560\text{k}\Omega \text{ (used)}$$

$$Z_R = \sqrt{R^2 + X_C^2} = 560.6\text{k}\Omega \text{ (using actual resistor)}$$

$$I_{PEAK} = \frac{V_{PEAK}}{Z_R} = 302.7 \times 10^{-6}$$

$$V_C = X_C \times \text{Ipeak} = 8.0\text{V}$$

$$\Phi = \mathrm{Tan}^{-1}\left(\frac{X_C}{R}\right) = 0.047 \text{ radians}$$

$$T\Phi = \frac{\Phi}{2\pi f} = 125.6\mu\text{s}$$

23.5.2 CORRECTION BY OFFSET CURRENT

When the waveform is varying relative to VSS, then the zero cross is detected too early as the waveform falls and too late as the waveform rises. When the waveform is varying relative to VDD, then the zero cross is detected too late as the waveform rises and too early as the waveform falls. The actual offset time can be determined for sinusoidal waveforms with the corresponding equations shown in Equation 23-3.

EQUATION 23-3: ZCD EVENT OFFSET

When External Voltage Source is relative to Vss:

$$T_{OFFSET} = \frac{\operatorname{asin}\left(\frac{V_{CPINV}}{V_{PEAK}}\right)}{2\pi \bullet Freq}$$

When External Voltage Source is relative to VDD:

$$T_{OFFSET} = \frac{\operatorname{asin}\left(\frac{V_{DD} - V_{CPINV}}{V_{PEAK}}\right)}{2\pi \bullet Freq}$$

This offset time can be compensated for by adding a pull-up or pull-down biasing resistor to the ZCD pin. A pull-up resistor is used when the external voltage source is varying relative to VSS. A pull-down resistor is used when the voltage is varying relative to VDD. The resistor adds a bias to the ZCD pin so that the target external voltage source must go to zero to pull the pin voltage to the VCPINV switching voltage. The pull-up or pull-down value can be determined with the equations shown in Equation 23-4.

EQUATION 23-4: ZCD PULL-UP/DOWN

When External Signal is relative to Vss:

$$R_{PULLUP} = \frac{R_{SERIES}(V_{PULLUP} - V_{CPINV})}{V_{CPINV}}$$

When External Signal is relative to VDD:

$$R_{PULLDOWN} = \frac{R_{SERIES}(V_{CPINV})}{(V_{DD} - V_{CPINV})}$$

FIGURE 24-12: DEAD-BAND OPERATION, CWG1DBR = 0x01, CWG1DBF = 0x02

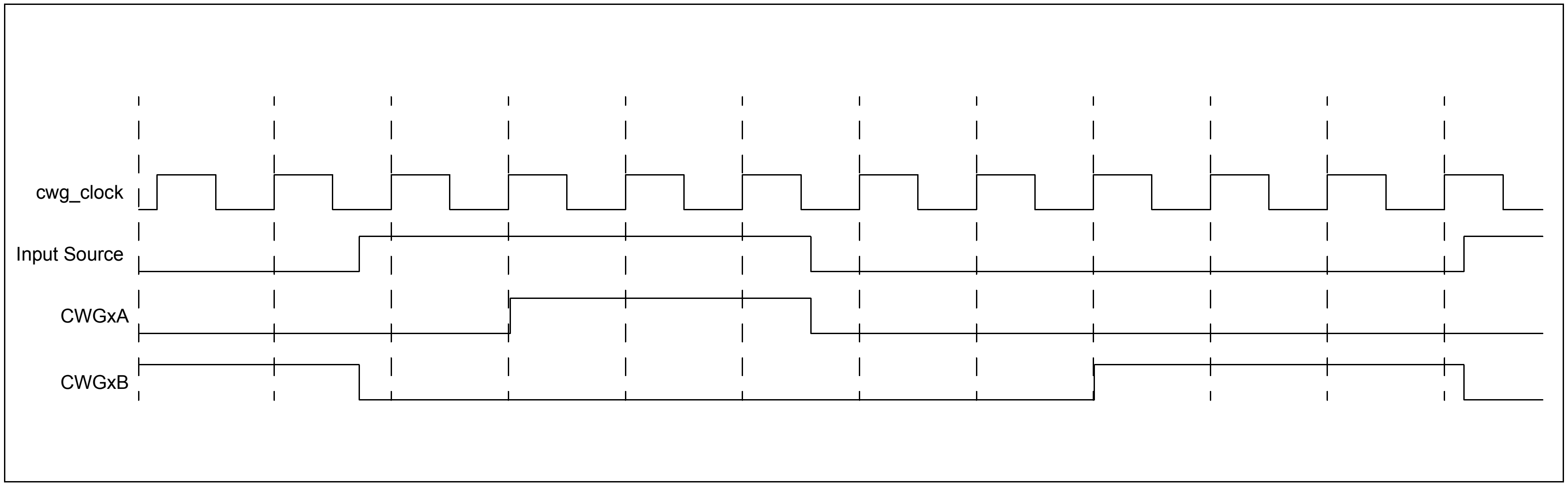


FIGURE 24-13: DEAD-BAND OPERATION, CWG1DBR = 0x03, CWG1DBF = 0x06, SOURCE SHORTER THAN DEAD BAND

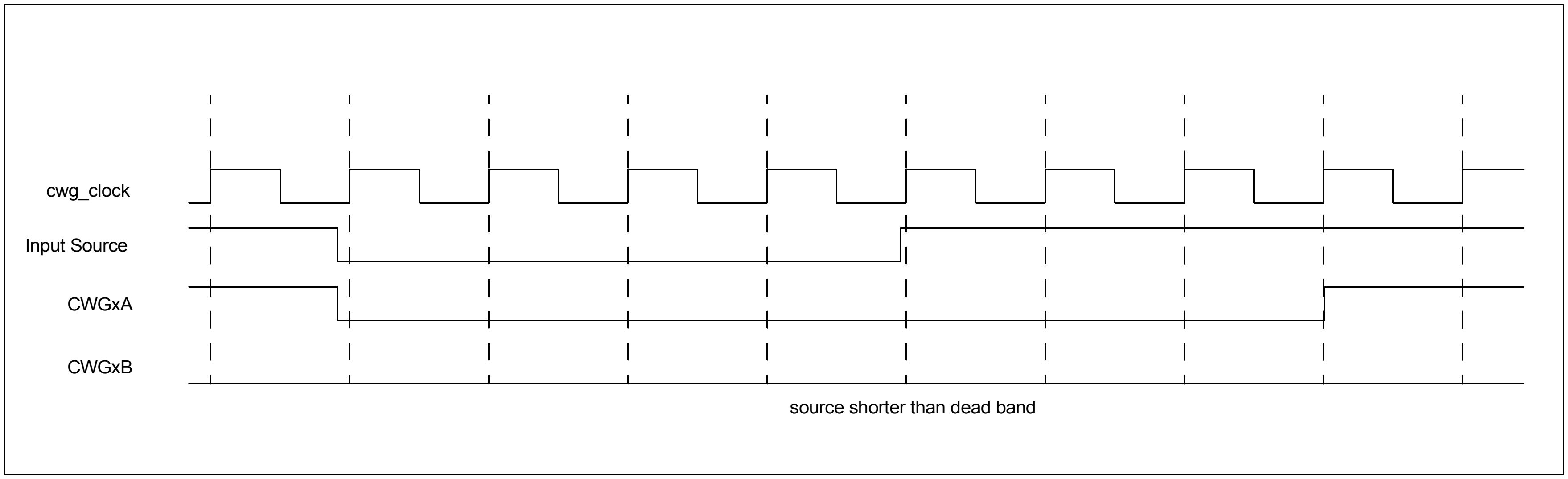


FIGURE 25-4: GATED TIMER MODE REPEAT ACQUISITION TIMING DIAGRAM

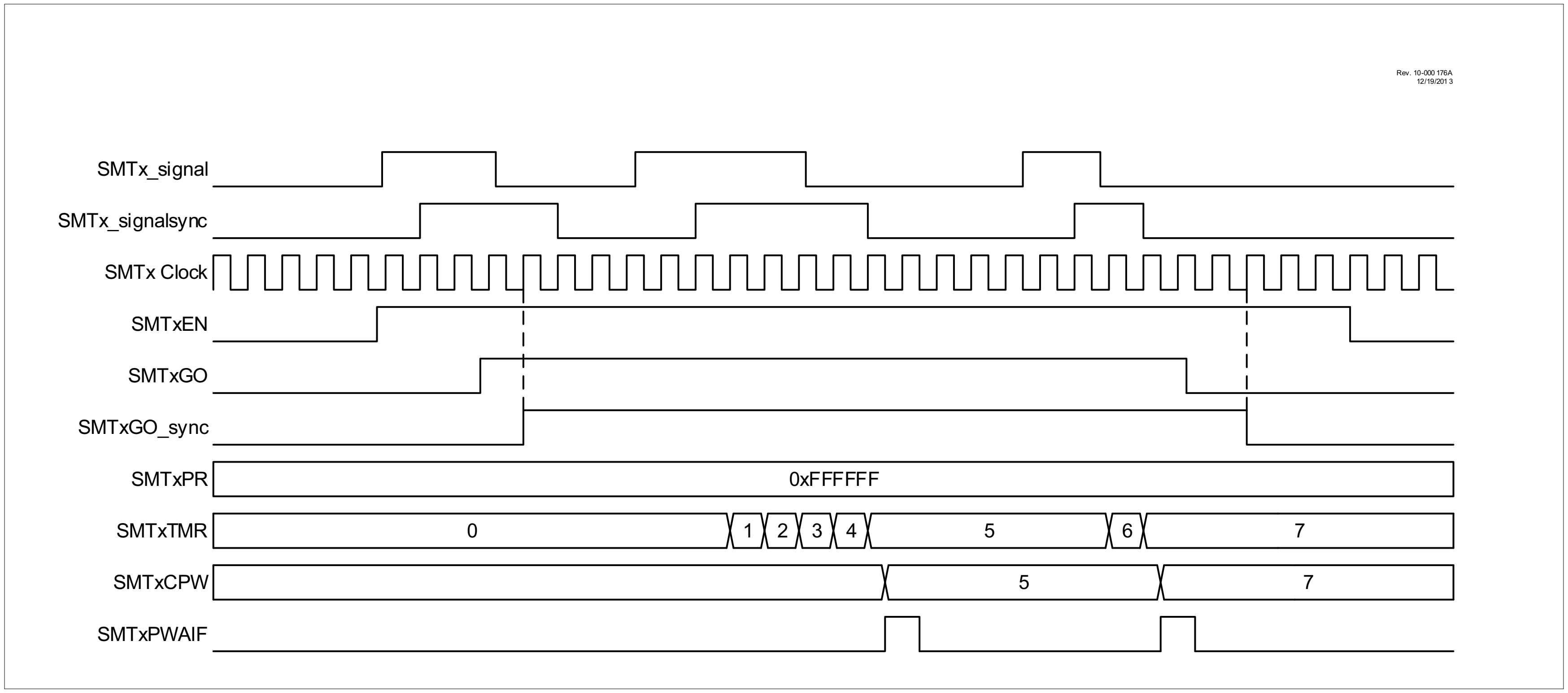
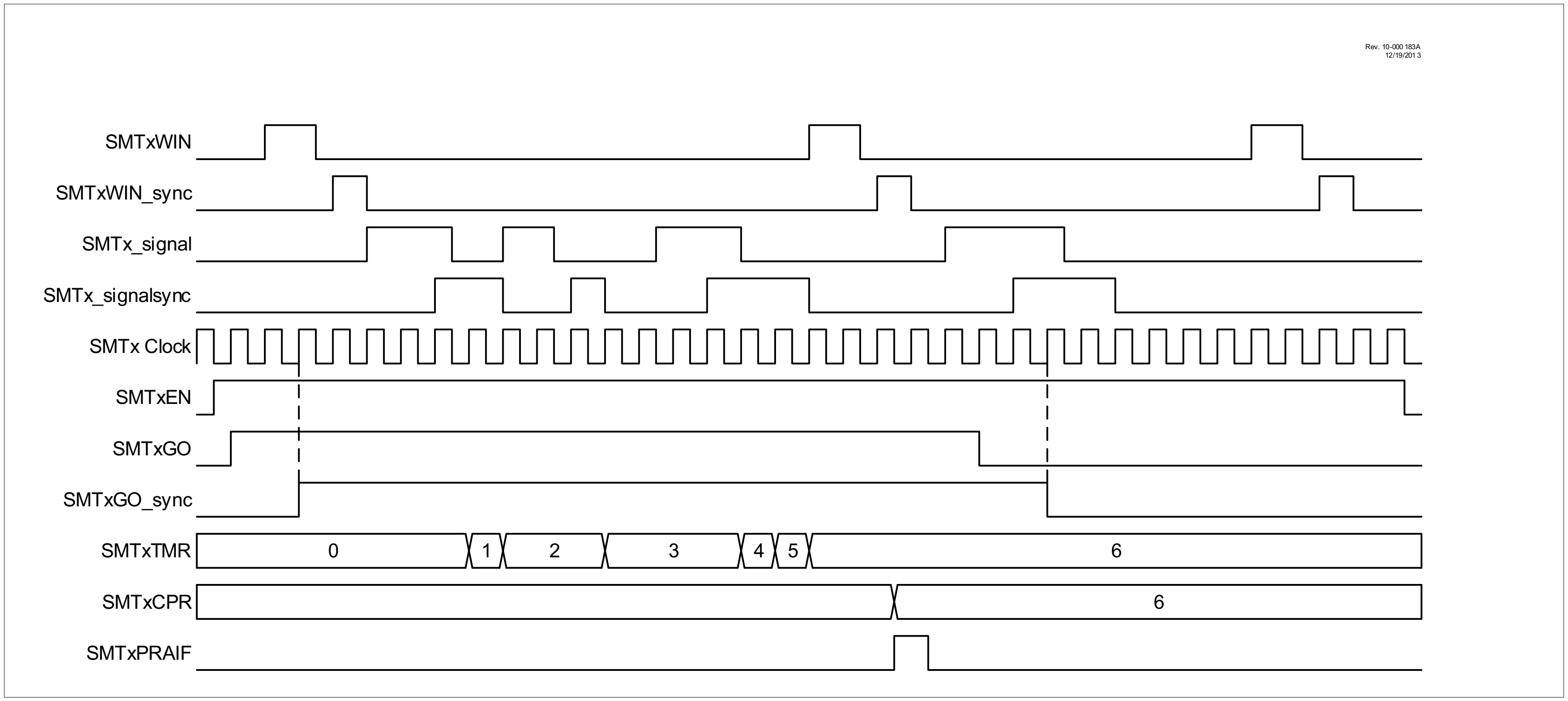


FIGURE 25-13: GATED WINDOWED MEASURE MODE SINGLE ACQUISITION TIMING DIAGRAMS



25.8 Interrupts

The SMT can trigger an interrupt under three different conditions:

- PW Acquisition Complete
- PR Acquisition Complete
- Counter Period Match

The interrupts are controlled by the PIR9 and PIE9 registers of the device.

25.8.1 PW AND PR ACQUISITION INTERRUPTS

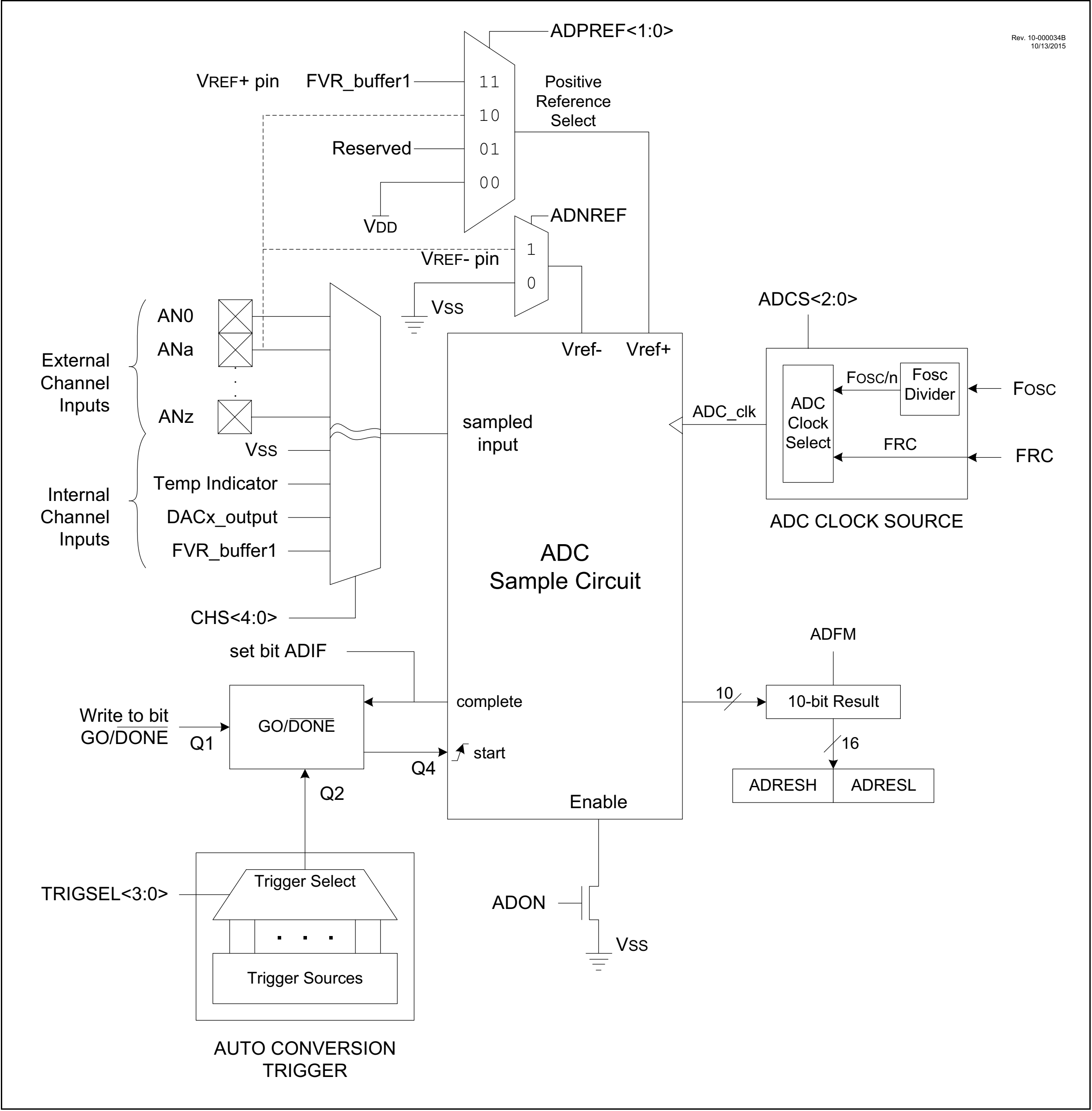
The SMT can trigger interrupts whenever it updates the SMTxCPW and SMTxCPR registers, the circumstances for which are dependent on the SMT mode, and are discussed in each mode's specific section. The SMTxCPW interrupt is controlled by SMTxPWAIF and SMTxPWAIE bits in registers PIR9 and PIE9, respectively. The SMTxCPR interrupt is controlled by the SMTxPRAIF and SMTxPRAIE bits, also located in registers PIR9 and PIE9, respectively.

In synchronous SMT modes, the interrupt trigger is synchronized to the SMTxCLK. In Asynchronous modes, the interrupt trigger is asynchronous. In either mode, once triggered, the interrupt will be synchronized to the CPU clock.

25.8.2 COUNTER PERIOD MATCH INTERRUPT

As described in **Section 25.2.2 "Period Match interrupt"**, the SMT will also interrupt upon SMTxTMR, matching SMTxPR with its period match limit functionality described in **Section 25.4 "Halt Operation"**. The period match interrupt is controlled by SMTxIF and SMTxIE.

FIGURE 32-1: ADC^2 BLOCK DIAGRAM



REGISTER 32-11: ADCAP: ADC ADDITIONAL SAMPLE CAPACITOR SELECTION REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	ADCAP<4:0>				
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-5 **Unimplemented**: Read as ‘0’

bit 4-0 **ADCAP<4:0>**: ADC Additional Sample Capacitor Selection bits
`11111` = 31 pF
`11110` = 30 pF
`11101` = 29 pF
•
•
•
`00011` = 3 pF
`00010` = 2 pF
`00001` = 1 pF
`00000` = No additional capacitance

REGISTER 32-12: ADRPT: ADC REPEAT SETTING REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
ADRPT<7:0>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

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

bit 7-0 **ADRPT<7:0>**: ADC Repeat Threshold bits
Counts the number of times that the ADC has been triggered and is used along with ADCNT to determine when the error threshold is checked when the computation is Low-pass Filter, Burst Average, or Average modes. See Table 32-3 for more details.

REGISTER 32-27: ADERRL: ADC SETPOINT ERROR LOW BYTE REGISTER

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
ADERR<7:0>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **ADERR<7:0>**: ADC Setpoint Error LSB. Lower byte of ADC Setpoint Error calculation is determined by ADCALC bits of ADCON3, see Register 23-1 for more details.

REGISTER 32-28: ADLTHH: ADC LOWER THRESHOLD HIGH BYTE REGISTER

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
ADLTH<15:8>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **ADLTH<15:8>**: ADC Lower Threshold MSB. ADLTH and ADUTH are compared with ADERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.

REGISTER 32-29: ADLTHL: ADC LOWER THRESHOLD LOW BYTE REGISTER

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
ADLTH<7:0>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **ADLTH<7:0>**: ADC Lower Threshold LSB. ADLTH and ADUTH are compared with ADERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.

MOVLW — Move literal to W

Syntax:	MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → W
Status Affected:	None
Encoding:	`0000` `1110` `kkkk` `kkkk`
Description:	The 8-bit literal ‘k’ is loaded into W.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `MOVLW 5Ah`

After Instruction

W = 5Ah

MOVWF — Move W to f

Syntax:	MOVWF f {,a}
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]
Operation:	(W) → f
Status Affected:	None
Encoding:	`0110` `111a` `ffff` `ffff`
Description:	Move data from W to register ‘f’. Location ‘f’ can be anywhere in the 256-byte bank. If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank. 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 36.2.3 “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 register ‘f’

Example: `MOVWF REG, 0`

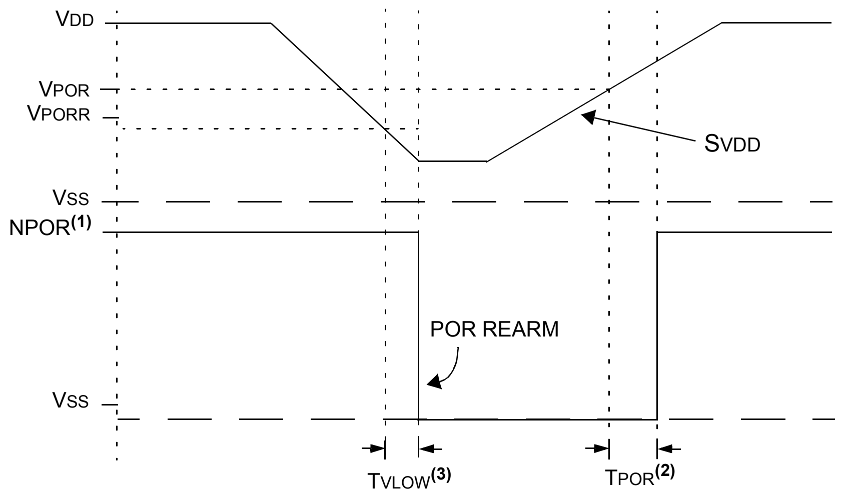
Before Instruction

W = 4Fh
REG = FFh

After Instruction

W = 4Fh
REG = 4Fh

FIGURE 38-3: POR AND POR REARM WITH SLOW RISING VDD



Note 1: When NPOR is low, the device is held in Reset.
2: TPOR 1 μs typical.
3: TVLOW 2.7 μs typical.