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	32MHz
Connectivity	LINbus, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	6
Program Memory Size	3.5KB (2K x 14)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 4x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	8-SOIC (0.154", 3.90mm Width)
Supplier Device Package	8-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic12lf1572t-i-sn

1.0   DEVICE OVERVIEW

The PIC12(L)F1571/2 devices are described within this data sheet. The block diagram of these devices is shown in Figure 1-1, the available peripherals are shown in Table 1-1 and the pinout descriptions are shown in Table 1-2.

TABLE 1-1:   DEVICE PERIPHERAL SUMMARY

Peripheral	PIC12(L)F1571	PIC12(L)F1572
Analog-to-Digital Converter (ADC)	●	●
Complementary Wave Generator (CWG)	●	●
Digital-to-Analog Converter (DAC)	●	●
Enhanced Universal Synchronous/Asynchronous Receiver/Transmitter (EUSART)		●
Fixed Voltage Reference (FVR)	●	●
Temperature Indicator	●	●
Comparators		
C1	●	●
PWM Modules		
PWM1	●	●
PWM2	●	●
PWM3	●	●
Timers		
Timer0	●	●
Timer1	●	●
Timer2	●	●

1.1   Register and Bit Naming Conventions

1.1.1   REGISTER NAMES

When there are multiple instances of the same peripheral in a device, the peripheral control registers will be depicted as the concatenation of a peripheral identifier, peripheral instance and control identifier. The control registers section will show just one instance of all the register names with an 'x' in the place of the peripheral instance number. This naming convention may also be applied to peripherals when there is only one instance of that peripheral in the device to maintain compatibility with other devices in the family that contain more than one.

1.1.2   BIT NAMES

There are two variants for bit names:

• Short name: Bit function abbreviation

• Long name: Peripheral abbreviation + short name

1.1.2.1   Short Bit Names

Short bit names are an abbreviation for the bit function. For example, some peripherals are enabled with the EN bit. The bit names shown in the registers are the short name variant.

Short bit names are useful when accessing bits in C programs. The general format for accessing bits by the short name is *RegisterName*bits.*ShortName*. For example, the enable bit, EN, in the COG1CON0 register can be set in C programs with the instruction, `COG1CON0bits.EN = 1`.

Short names are generally not useful in assembly programs because the same name may be used by different peripherals in different bit positions. When this occurs, during the include file generation, all instances of that short bit name are appended with an underscore, plus the name of the register in which the bit resides, to avoid naming contentions.

1.1.2.2 Long Bit Names

Long bit names are constructed by adding a peripheral abbreviation prefix to the short name. The prefix is unique to the peripheral, thereby making every long bit name unique. The long bit name for the COG1 enable bit is the COG1 prefix, G1, appended with the enable bit short name, EN, resulting in the unique bit name G1EN.

Long bit names are useful in both C and assembly programs. For example, in C, the COG1CON0 enable bit can be set with the G1EN = 1 instruction. In assembly, this bit can be set with the BSF   COG1CON0,G1EN instruction.

1.1.2.3 Bit Fields

Bit fields are two or more adjacent bits in the same register. Bit fields adhere only to the short bit naming convention. For example, the three Least Significant bits of the COG1CON0 register contain the mode control bits. The short name for this field is MD. There is no long bit name variant. Bit field access is only possible in C programs. The following example demonstrates a C program instruction for setting the COG1 to the Push-Pull mode:

```
COG1CON0bits.MD = 0x5;
```

Individual bits in a bit field can also be accessed with long and short bit names. Each bit is the field name appended with the number of the bit position within the field. For example, the Most Significant mode bit has the short bit name, MD2, and the long bit name is G1MD2. The following two examples demonstrate assembly program sequences for setting the COG1 to Push-Pull mode:

Example 1:

```
MOVLW  ~(1<<G1MD1)
ANDWF  COG1CON0,F
MOVLW  1<<G1MD2 | 1<<G1MD0
IORWF  COG1CON0,F
```

Example 2:

```
BSF    COG1CON0,G1MD2
BCF    COG1CON0,G1MD1
BSF    COG1CON0,G1MD0
```

1.1.3 REGISTER AND BIT NAMING EXCEPTIONS

1.1.3.1 Status, Interrupt and Mirror Bits

Status, interrupt enables, interrupt flags and mirror bits are contained in registers that span more than one peripheral. In these cases, the bit name shown is unique so there is no prefix or short name variant.

1.1.3.2 Legacy Peripherals

There are some peripherals that do not strictly adhere to these naming conventions. Peripherals that have existed for many years and are present in almost every device are the exceptions. These exceptions were necessary to limit the adverse impact of the new conventions on legacy code. Peripherals that do adhere to the new convention will include a table in the registers section indicating the long name prefix for each peripheral instance. Peripherals that fall into the exception category will not have this table. These peripherals include, but are not limited to, the following:

• EUSART
• MSSP

4.7   Register Definitions: Device ID

REGISTER 4-3:   DEVICEID: DEVICE ID REGISTER[1]

R	R	R	R	R	R
DEV<13:8>					
bit 13					bit 8

R	R	R	R	R	R	R	R
DEV<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit			
'0' = Bit is cleared	'1' = Bit is set	x = Bit is unknown	

bit 13-0   **DEV<13:0>:** Device ID bits

Refer to Table 4-1 to determine what these bits will read on which device. A value of 3FFFh is invalid.

Note 1:   This location cannot be written.

REGISTER 4-4:   REVISIONID: REVISION ID REGISTER[1]

R	R	R	R	R	R
REV<13:8>					
bit 13					bit 8

R	R	R	R	R	R	R	R
REV<7:0>							
bit 7							bit 0

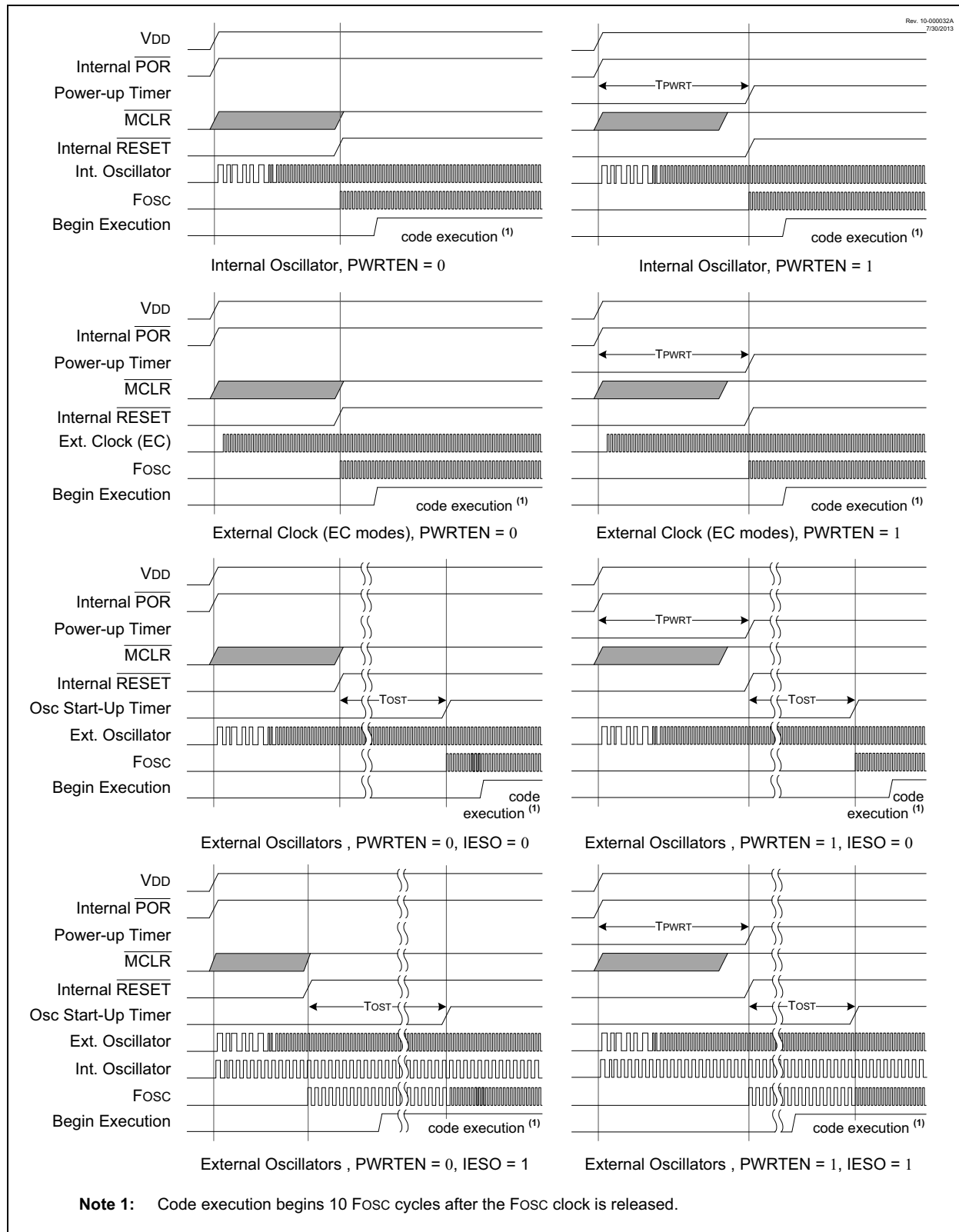| Legend:			
R = Readable bit			
'0' = Bit is cleared	'1' = Bit is set	x = Bit is unknown	

bit 13-0   **REV<13:0>:** Revision ID bits

These bits are used to identify the device revision.

Note 1:   This location cannot be written.

TABLE 4-1:   DEVICE ID VALUES

DEVICE	Device ID	Revision ID
PIC12F1571	3051h	2xxxh
PIC12LF1571	3053h	2xxxh
PIC12F1572	3050h	2xxxh
PIC12LF1572	3052h	2xxxh

PIC12(L)F1571/2

FIGURE 6-3: **RESET START-UP SEQUENCE**

Rev. 10-000032A
7/30/2013

Internal Oscillator, PWRTEN = 0

Internal Oscillator, PWRTEN = 1

External Clock (EC modes), PWRTEN = 0

External Clock (EC modes), PWRTEN = 1

External Oscillators , PWRTEN = 0, IESO = 0

External Oscillators , PWRTEN = 1, IESO = 0

External Oscillators , PWRTEN = 0, IESO = 1

External Oscillators , PWRTEN = 1, IESO = 1

Note 1: Code execution begins 10 F$_{OSC}$ cycles after the F$_{OSC}$ clock is released.

NOTES:

PIC12(L)F1571/2

9.0 WATCHDOG TIMER (WDT)

The Watchdog Timer is a system timer that generates a Reset if the firmware does not issue a `CLRWDT` instruction within the time-out period. The Watchdog Timer is typically used to recover the system from unexpected events.

The WDT has the following features:

- Independent clock source
- Multiple operating modes:
  - WDT is always on
  - WDT is off when in Sleep
  - WDT is controlled by software
  - WDT is always off
- Configurable time-out period is from 1 ms to 256 seconds (nominal)
- Multiple Reset conditions
- Operation during Sleep

FIGURE 9-1: WATCHDOG TIMER BLOCK DIAGRAM
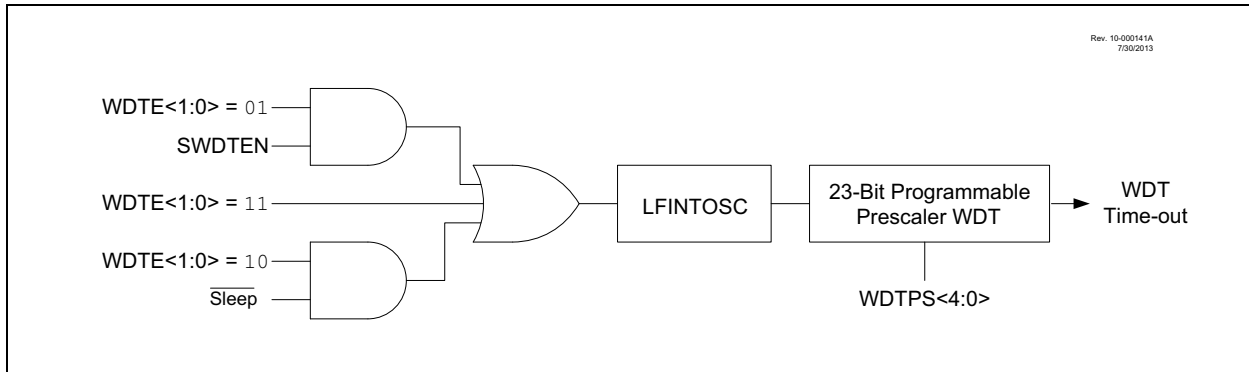
PIC12(L)F1571/2

10.4 User ID, Device ID and Configuration Word Access

Instead of accessing program memory, the User IDs, Device ID/Revision ID and Configuration Words can be accessed when CFGS = 1 in the PMCON1 register. This is the region that would be pointed to by PC<15> = 1, but not all addresses are accessible. Different access may exist for reads and writes. Refer to Table 10-2.

When read access is initiated on an address outside the parameters listed in Table 10-2, the PMDATH:PMDATL register pair is cleared, reading back '0's.

TABLE 10-2: USER ID, DEVICE ID AND CONFIGURATION WORD ACCESS (CFGS = 1)

Address	Function	Read Access	Write Access
8000h-8003h	User IDs	Yes	Yes
8006h/8005h	Device ID/Revision ID	Yes	No
8007h-8008h	Configuration Words 1 and 2	Yes	No

EXAMPLE 10-4: CONFIGURATION WORD AND DEVICE ID ACCESS

```
* This code block will read 1 word of program memory at the memory address:
*    PROG_ADDR_LO (must be 00h-08h) data will be returned in the variables;
*    PROG_DATA_HI, PROG_DATA_LO

    BANKSEL  PMADRL              ; Select correct Bank
    MOVLW    PROG_ADDR_LO        ;
    MOVWF    PMADRL              ; Store LSB of address
    CLRF     PMADRH              ; Clear MSB of address

    BSF      PMCON1,CFGS         ; Select Configuration Space
    BCF      INTCON,GIE          ; Disable interrupts
    BSF      PMCON1,RD           ; Initiate read
    NOP                          ; Executed (See Figure 10-2)
    NOP                          ; Ignored (See Figure 10-2)
    BSF      INTCON,GIE          ; Restore interrupts

    MOVF     PMDATL,W            ; Get LSB of word
    MOVWF    PROG_DATA_LO        ; Store in user location
    MOVF     PMDATH,W            ; Get MSB of word
    MOVWF    PROG_DATA_HI        ; Store in user location
```

11.3 PORTA Registers

11.3.1 DATA REGISTER

PORTA is a 6-bit wide, bidirectional port. The corresponding Data Direction register is TRISA (Register 11-3). Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., disable the output driver). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., enables output driver and puts the contents of the output latch on the selected pin). The exception is RA3, which is input-only and its TRISA bit will always read as '1'. Example 11-1 shows how to initialize an I/O port.

Reading the PORTA register (Register 11-2) reads the status of the pins, whereas writing to it will write to the PORT latch. All write operations are Read-Modify-Write operations. Therefore, a write to a port implies that the port pins are read, this value is modified and then written to the Port Data Latch (LATA).

11.3.2 DIRECTION CONTROL

The TRISA register (Register 11-3) controls the PORTA pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISA register are maintained set when using them as analog inputs. I/O pins configured as analog input always read '0'.

11.3.3 OPEN-DRAIN CONTROL

The ODCONA register (Register 11-7) controls the open-drain feature of the port. Open-drain operation is independently selected for each pin. When an ODCONA bit is set, the corresponding port output becomes an open-drain driver capable of sinking current only. When an ODCONA bit is cleared, the corresponding port output pin is the standard push-pull drive capable of sourcing and sinking current.

11.3.4 SLEW RATE CONTROL

The SLRCONA register (Register 11-8) controls the slew rate option for each port pin. Slew rate control is independently selectable for each port pin. When an SLRCONA bit is set, the corresponding port pin drive is slew rate limited. When an SLRCONA bit is cleared, the corresponding port pin drive slews at the maximum rate possible.

11.3.5 INPUT THRESHOLD CONTROL

The INLVLA register (Register 11-9) controls the input voltage threshold for each of the available PORTA input pins. A selection between the Schmitt Trigger CMOS or the TTL compatible thresholds is available. The input threshold is important in determining the value of a read of the PORTA register and also the level at which an Interrupt-On-Change occurs, if that feature is enabled. See **Section 26.3 "DC Characteristics"** for more information on threshold levels.

Note:	Changing the input threshold selection should be performed while all peripheral modules are disabled. Changing the threshold level during the time a module is active may inadvertently generate a transition associated with an input pin, regardless of the actual voltage level on that pin.

11.3.6 ANALOG CONTROL

The ANSELA register (Register 11-5) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELA bit high will cause all digital reads on the pin to be read as '0' and allow analog functions on the pin to operate correctly.

The state of the ANSELA bits has no effect on digital output functions. A pin with TRIS clear and ANSELA set will still operate as a digital output, but the Input mode will be analog. This can cause unexpected behavior when executing Read-Modify-Write instructions on the affected port.

Note:	The ANSELA bits default to the Analog mode after Reset. To use any pins as digital general purpose or peripheral inputs, the corresponding ANSELA bits must be initialized to '0' by user software.

EXAMPLE 11-1:   INITIALIZING PORTA

```
BANKSEL   PORTA         ;
CLRF      PORTA         ;Init PORTA
BANKSEL   LATA          ;Data Latch
CLRF      LATA          ;
BANKSEL   ANSELA        ;
CLRF      ANSELA        ;digital I/O
BANKSEL   TRISA         ;
MOVLW     B'00111000'   ;Set RA<5:3> as inputs
MOVWF     TRISA         ;and set RA<2:0> as
                        ;outputs
```

REGISTER 12-3: IOCAF: INTERRUPT-ON-CHANGE PORTA FLAG REGISTER

U-0	U-0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0
—	—			IOCAF<5:0>			
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	HS - Bit is set in hardware

bit 7-6   **Unimplemented:** Read as '0'

bit 5-0   **IOCAF<5:0>:** Interrupt-On-Change PORTA Flag bits

    1 = An enabled change was detected on the associated pin
         Set when IOCAPx = 1 and a rising edge was detected on RAx, or when IOCANx = 1 and a falling
         edge was detected on RAx.
    0 = No change was detected or the user cleared the detected change

TABLE 12-1: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPT-ON-CHANGE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
ANSELA	—	—	—	ANSA4	—	ANSA<2:0>			114
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	74
IOCAF	—	—	IOCAF<5:0>						122
IOCAN	—	—	IOCAN<5:0>						121
IOCAP	—	—	IOCAP<5:0>						121
TRISA	—	—	TRISA<5:4>		—(1)	TRISA<2:0>			113

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by Interrupt-On-Change.

Note 1: Unimplemented, read as '1'.

TABLE 14-2:   SUMMARY OF REGISTERS ASSOCIATED WITH THE TEMPERATURE INDICATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
FVRCON	FVREN	FVRRDY	TSEN	TSRNG	CDAFVR<1:0>		ADFVR<1:0>		118

Legend: Shaded cells are unused by the temperature indicator module.

18.1.3 SOFTWARE PROGRAMMABLE PRESCALER

A software programmable prescaler is available for exclusive use with Timer0. The prescaler is enabled by clearing the PSA bit of the OPTION_REG register.

> **Note:** The Watchdog Timer (WDT) uses its own independent prescaler.

There are eight prescaler options for the Timer0 module, ranging from 1:2 to 1:256. The prescale values are selectable via the PS<2:0> bits of the OPTION_REG register. In order to have a 1:1 prescaler value for the Timer0 module, the prescaler must be disabled by setting the PSA bit of the OPTION_REG register.

The prescaler is not readable or writable. All instructions writing to the TMR0 register will clear the prescaler.

18.1.4 TIMER0 INTERRUPT

Timer0 will generate an interrupt when the TMR0 register overflows from FFh to 00h. The TMR0IF interrupt flag bit of the INTCON register is set every time the TMR0 register overflows, regardless of whether or not the Timer0 interrupt is enabled. The TMR0IF bit can only be cleared in software. The Timer0 interrupt enable is the TMR0IE bit of the INTCON register.

> **Note:** The Timer0 interrupt cannot wake the processor from Sleep since the timer is frozen during Sleep.

18.1.5 8-BIT COUNTER MODE SYNCHRONIZATION

When in 8-Bit Counter mode, the incrementing edge on the T0CKI pin must be synchronized to the instruction clock. Synchronization can be accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the instruction clock. The high and low periods of the external clocking source must meet the timing requirements as shown in **Section 26.0 "Electrical Specifications"**.

18.1.6 OPERATION DURING SLEEP

Timer0 cannot operate while the processor is in Sleep mode. The contents of the TMR0 register will remain unchanged while the processor is in Sleep mode.

NOTES:

PIC12(L)F1571/2

TABLE 21-5: **BAUD RATES FOR ASYNCHRONOUS MODES**

BAUD RATE	SYNC = 0, BRGH = 0, BRG16 = 0											
	Fosc = 20.000 MHz			Fosc = 18.432 MHz			Fosc = 16.000 MHz			Fosc = 11.0592 MHz		
	Actual Rate	% Error	SPBRG Value (decimal)	Actual Rate	% Error	SPBRG Value (decimal)	Actual Rate	% Error	SPBRG Value (decimal)	Actual Rate	% Error	SPBRG Value (decimal)
300	—	—	—	—	—	—	—	—	—	—	—	—
1200	1221	1.73	255	1200	0.00	239	1202	0.16	207	1200	0.00	143
2400	2404	0.16	129	2400	0.00	119	2404	0.16	103	2400	0.00	71
9600	9470	-1.36	32	9600	0.00	29	9615	0.16	25	9600	0.00	17
10417	10417	0.00	29	10286	-1.26	27	10417	0.00	23	10165	-2.42	16
19.2k	19.53k	1.73	15	19.20k	0.00	14	19.23k	0.16	12	19.20k	0.00	8
57.6k	—	—	—	57.60k	0.00	7	—	—	—	57.60k	0.00	2
115.2k	—	—	—	—	—	—	—	—	—	—	—	—

BAUD RATE	SYNC = 0, BRGH = 0, BRG16 = 0											
	Fosc = 8.000 MHz			Fosc = 4.000 MHz			Fosc = 3.6864 MHz			Fosc = 1.000 MHz		
	Actual Rate	% Error	SPBRG Value (decimal)	Actual Rate	% Error	SPBRG Value (decimal)	Actual Rate	% Error	SPBRG Value (decimal)	Actual Rate	% Error	SPBRG Value (decimal)
300	—	—	—	300	0.16	207	300	0.00	191	300	0.16	51
1200	1202	0.16	103	1202	0.16	51	1200	0.00	47	1202	0.16	12
2400	2404	0.16	51	2404	0.16	25	2400	0.00	23	—	—	—
9600	9615	0.16	12	—	—	—	9600	0.00	5	—	—	—
10417	10417	0.00	11	10417	0.00	5	—	—	—	—	—	—
19.2k	—	—	—	—	—	—	19.20k	0.00	2	—	—	—
57.6k	—	—	—	—	—	—	57.60k	0.00	0	—	—	—
115.2k	—	—	—	—	—	—	—	—	—	—	—	—

BAUD RATE	SYNC = 0, BRGH = 1, BRG16 = 0											
	Fosc = 20.000 MHz			Fosc = 18.432 MHz			Fosc = 16.000 MHz			Fosc = 11.0592 MHz		
	Actual Rate	% Error	SPBRG Value (decimal)	Actual Rate	% Error	SPBRG Value (decimal)	Actual Rate	% Error	SPBRG Value (decimal)	Actual Rate	% Error	SPBRG Value (decimal)
300	—	—	—	—	—	—	—	—	—	—	—	—
1200	—	—	—	—	—	—	—	—	—	—	—	—
2400	—	—	—	—	—	—	—	—	—	—	—	—
9600	9615	0.16	129	9600	0.00	119	9615	0.16	103	9600	0.00	71
10417	10417	0.00	119	10378	-0.37	110	10417	0.00	95	10473	0.53	65
19.2k	19.23k	0.16	64	19.20k	0.00	59	19.23k	0.16	51	19.20k	0.00	35
57.6k	56.82k	-1.36	21	57.60k	0.00	19	58.82k	2.12	16	57.60k	0.00	11
115.2k	113.64k	-1.36	10	115.2k	0.00	9	111.1k	-3.55	8	115.2k	0.00	5

22.3   Offset Modes

The Offset modes provide the means to adjust the waveform of a slave PWM module relative to the waveform of a master PWM module in the same device.

22.3.1   INDEPENDENT RUN MODE

In Independent Run mode (OFM<1:0> = 00), the PWM module is unaffected by the other PWM modules in the device. The PWMxTMR associated with the PWM module in this mode starts counting as soon as the EN bit associated with this PWM module is set and continues counting until the EN bit is cleared. Period events reset the PWMxTMR to zero, after which, the timer continues to count.

A detailed timing diagram of this mode used with Standard PWM mode is shown in Figure 22-8.

22.3.2   SLAVE RUN MODE WITH SYNC START

In Slave Run mode with Sync Start (OFM<1:0> = 01), the slave PWMxTMR waits for the master's OFx_match event. When this event occurs, if the EN bit is set, the PWMxTMR begins counting and continues to count until software clears the EN bit. Slave period events reset the PWMxTMR to zero, after which, the timer continues to count.

A detailed timing diagram of this mode used with Standard PWM mode is shown in Figure 22-9.

22.3.3   ONE-SHOT SLAVE MODE WITH SYNC START

In One-Shot Slave mode with Synchronous Start (OFM<1:0> = 10), the slave PWMxTMR waits until the master's OFx_match event. The timer then begins counting, starting from the value that is already in the timer, and continues to count until the period match event. When the period event occurs, the timer resets to zero and stops counting. The timer then waits until the next master OFx_match event, after which, it begins counting again to repeat the cycle. An OFx_match event that occurs before the slave PWM has completed the previously triggered period will be ignored. A slave period that is greater than the master period, but less than twice the master period, will result in a slave output every other master period.

Note:	During the time the slave timers are resetting to zero, if another offset match event is received, it is possible that the slave PWM would not recognize this match event and the slave timers would fail to begin counting again. This would result in missing duty cycles from the output of the slave PWM. To prevent this from happening, avoid using the same period for both the master and slave PWMs.

A detailed timing diagram of this mode used with Standard PWM mode is shown in Figure 22-10.

22.3.4   CONTINUOUS RUN SLAVE MODE WITH SYNC START AND TIMER RESET

In Continuous Run Slave mode with Synchronous Start and Timer Reset (OFM<1:0> = 11), the slave PWMxTMR is inhibited from counting after the slave PWM enable is set. The first master OFx_match event starts the slave PWMxTMR. Subsequent master OFx_match events reset the slave PWMxTMR timer value back to 1, after which, the slave PWMxTMR continues to count. The next master OFx_match event resets the slave PWMxTMR back to 1 to repeat the cycle. Slave period events that occur before the master's OFx_match event will reset the slave PWMxTMR to zero, after which, the timer will continue to count. Slaves operating in this mode must have a PWMxPH register pair value equal to or greater than 1; otherwise, the phase match event will not occur precluding the start of the PWM output duty cycle.

The offset timing will persist If both the master and slave PWMxPR values are the same, and the Slave Offset mode is changed to Independent Run mode while the PWM module is operating.

A detailed timing diagram of this mode used in Standard PWM mode is shown in Figure 22-11.

Note:	Unexpected results will occur if the slave PWM_clock is a higher frequency than the master PWM_clock.

22.3.5   OFFSET MATCH IN CENTER-ALIGNED MODE

When a master is operating in Center-Aligned mode, the offset match event depends on which direction the PWMxTMR is counting. Clearing the OFO bit of the PWMxOFCON register will cause the OFx_match event to occur when the timer is counting up. Setting the OFO bit of the PWMxOFCON register will cause the OFx_match event to occur when the timer is counting down. The OFO bit is ignored in non-Center-Aligned modes.

The OFO bit is double-buffered and requires setting the LDA bit to take effect when the PWM module is operating.

Detailed timing diagrams of Center-Aligned mode using offset match control in Independent Slave with Sync Start mode can be seen in Figure 22-12 and Figure 22-13.

23.10 Operation During Sleep

The CWG module operates independently from the system clock, and will continue to run during Sleep provided that the clock and input sources selected remain active.

The HFINTOSC remains active during Sleep, provided that the CWG module is enabled, the input source is active and the HFINTOSC is selected as the clock source, regardless of the system clock source selected.

In other words, if the HFINTOSC is simultaneously selected as the system clock and the CWG clock source, when the CWG is enabled and the input source is active, the CPU will go idle during Sleep, but the CWG will continue to operate and the HFINTOSC will remain active.

This will have a direct effect on the Sleep mode current.

23.11 Configuring the CWG

The following steps illustrate how to properly configure the CWG to ensure a synchronous start:

1. Ensure that the TRISx control bits corresponding to CWGxA and CWGxB are set so that both are configured as inputs.
2. Clear the GxEN bit if not already cleared.
3. Set desired dead-band times with the CWGxDBR and CWGxDBF registers.
4. Set up the following controls in the CWGxCON2 auto-shutdown register:
   • Select desired shutdown source.
   • Select both output overrides to the desired levels (this is necessary even if not using auto-shutdown because start-up will be from a shutdown state).
   • Set the GxASE bit and clear the GxARSEN bit.
5. Select the desired input source using the CWGxCON1 register.
6. Configure the following controls in the CWGxCON0 register:
   • Select desired clock source.
   • Select the desired output polarities.
   • Set the output enables for the outputs to be used.
7. Set the GxEN bit.
8. Clear the TRISx control bits corresponding to CWGxA and CWGxB to be used to configure those pins as outputs.
9. If auto-restart is to be used, set the GxARSEN bit and the GxASE bit will be cleared automatically. Otherwise, clear the GxASE bit to start the CWG.

23.11.1 PIN OVERRIDE LEVELS

The levels driven to the output pins, while the shutdown input is true, are controlled by the GxASDLA and GxASDLB bits of the CWGxCON1 register (Register 23-3). GxASDLA controls the CWG1A override level and GxASDLB controls the CWG1B override level. The control bit logic level corresponds to the output logic drive level while in the shutdown state. The polarity control does not apply to the override level.

23.11.2 AUTO-SHUTDOWN RESTART

After an auto-shutdown event has occurred, there are two ways to resume operation:

• Software controlled
• Auto-restart

The restart method is selected with the GxARSEN bit of the CWGxCON2 register. Waveforms of software controlled and automatic restarts are shown in Figure 23-5 and Figure 23-6.

23.11.2.1 Software Controlled Restart

When the GxARSEN bit of the CWGxCON2 register is cleared, the CWG must be restarted after an auto-shutdown event by software.

Clearing the shutdown state requires all selected shutdown inputs to be low, otherwise, the GxASE bit will remain set. The overrides will remain in effect until the first rising edge event after the GxASE bit is cleared. The CWG will then resume operation.

23.11.2.2 Auto-Restart

When the GxARSEN bit of the CWGxCON2 register is set, the CWG will restart from the auto-shutdown state automatically.

The GxASE bit will clear automatically when all shutdown sources go low. The overrides will remain in effect until the first rising edge event after the GxASE bit is cleared. The CWG will then resume operation.

MOVIW	**Move INDFn to W**
Syntax:	[*label*] MOVIW ++FSRn [*label*] MOVIW --FSRn [*label*] MOVIW FSRn++ [*label*] MOVIW FSRn-- [*label*] MOVIW k[FSRn]
Operands:	n ∈ [0,1] mm ∈ [00,01, 10, 11] -32 ≤ k ≤ 31
Operation:	INDFn → W Effective address is determined by • FSR + 1 (preincrement) • FSR - 1 (predecrement) • FSR + k (relative offset) After the Move, the FSR value will be either: • FSR + 1 (all increments) • FSR - 1 (all decrements) • Unchanged
Status Affected:	Z

Mode	Syntax	mm
Preincrement	++FSRn	00
Predecrement	--FSRn	01
Postincrement	FSRn++	10
Postdecrement	FSRn--	11

Description:	This instruction is used to move data between W and one of the indirect registers (INDFn). Before/after this move, the pointer (FSRn) is updated by pre/post incrementing/decrementing it. **Note:** The INDFn registers are not physical registers. Any instruction that accesses an INDFn register actually accesses the register at the address specified by the FSRn. FSRn is limited to the range 0000h - FFFFh. Incrementing/decrementing it beyond these bounds will cause it to wraparound.

MOVLB	**Move literal to BSR**
Syntax:	[*label*] MOVLB k
Operands:	0 ≤ k ≤ 31
Operation:	k → BSR
Status Affected:	None
Description:	The 5-bit literal 'k' is loaded into the Bank Select Register (BSR).

MOVLP	**Move literal to PCLATH**
Syntax:	[*label*] MOVLP k
Operands:	0 ≤ k ≤ 127
Operation:	k → PCLATH
Status Affected:	None
Description:	The 7-bit literal 'k' is loaded into the PCLATH register.

MOVLW	**Move literal to W**
Syntax:	[*label*] MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W)
Status Affected:	None
Description:	The 8-bit literal 'k' is loaded into W register. The "don't cares" will assemble as '0's.
Words:	1
Cycles:	1
Example:	MOVLW 0x5A

After Instruction
```
        W   =   0x5A
```

MOVWF	**Move W to f**
Syntax:	[*label*] MOVWF f
Operands:	0 ≤ f ≤ 127
Operation:	(W) → (f)
Status Affected:	None
Description:	Move data from W register to register 'f'.
Words:	1
Cycles:	1
Example:	MOVWF OPTION_REG

Before Instruction
```
        OPTION_REG  =  0xFF
        W           =  0x4F
```
After Instruction
```
        OPTION_REG  =  0x4F
        W           =  0x4F
```

MOVWI	**Move W to INDFn**
Syntax:	[*label*] MOVWI ++FSRn [*label*] MOVWI --FSRn [*label*] MOVWI FSRn++ [*label*] MOVWI FSRn-- [*label*] MOVWI k[FSRn]
Operands:	n ∈ [0,1] mm ∈ [00,01, 10, 11] -32 ≤ k ≤ 31
Operation:	W → INDFn Effective address is determined by • FSR + 1 (preincrement) • FSR - 1 (predecrement) • FSR + k (relative offset) After the Move, the FSR value will be either: • FSR + 1 (all increments) • FSR - 1 (all decrements) Unchanged
Status Affected:	None

Mode	Syntax	mm
Preincrement	++FSRn	00
Predecrement	--FSRn	01
Postincrement	FSRn++	10
Postdecrement	FSRn--	11

Description:	This instruction is used to move data between W and one of the indirect registers (INDFn). Before/after this move, the pointer (FSRn) is updated by pre/post incrementing/decrementing it. **Note:** The INDFn registers are not physical registers. Any instruction that accesses an INDFn register actually accesses the register at the address specified by the FSRn. FSRn is limited to the range 0000h - FFFFh. Incrementing/decrementing it beyond these bounds will cause it to wraparound. The increment/decrement operation on FSRn WILL NOT affect any Status bits.

NOP	**No Operation**
Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Description:	No operation.
Words:	1
Cycles:	1
<u>Example:</u>	NOP

OPTION	**Load OPTION_REG Register with W**
Syntax:	[*label*] OPTION
Operands:	None
Operation:	(W) → OPTION_REG
Status Affected:	None
Description:	Move data from W register to OPTION_REG register.

RESET	**Software Reset**
Syntax:	[*label*] RESET
Operands:	None
Operation:	Execute a device Reset. Resets the nRI flag of the PCON register.
Status Affected:	None
Description:	This instruction provides a way to execute a hardware Reset by software.

FIGURE 27-19: I_{DD} TYPICAL, HFINTOSC, PIC12F1571/2 ONLY
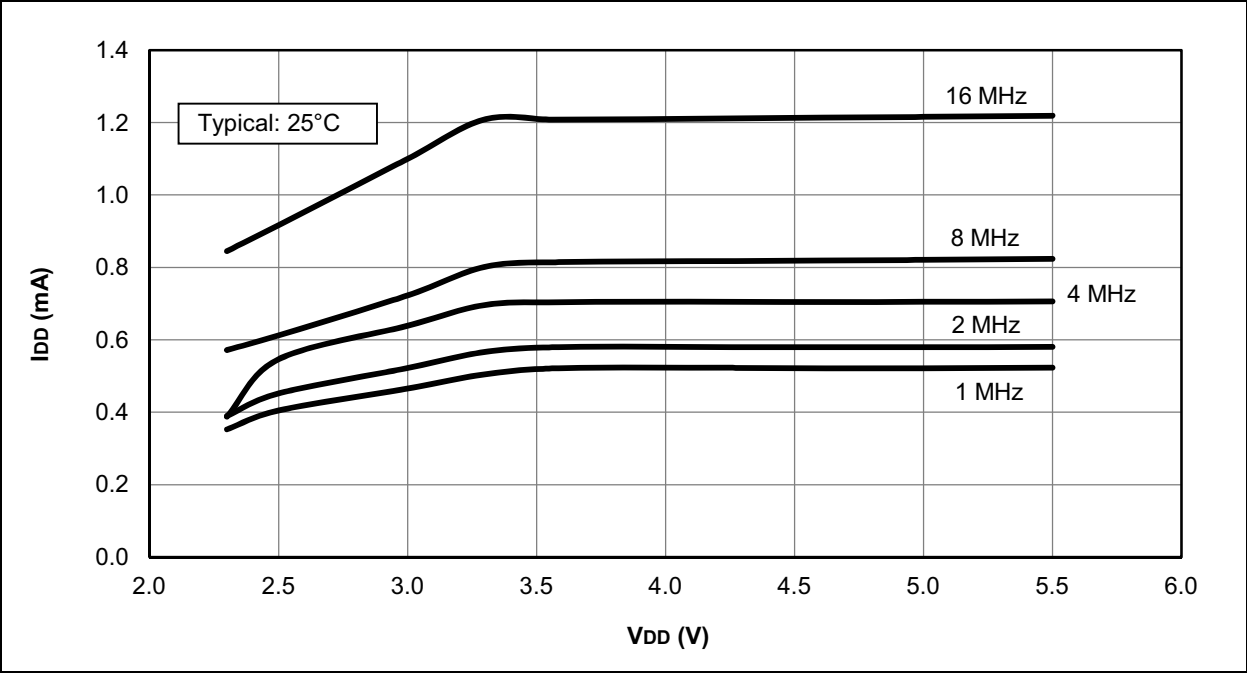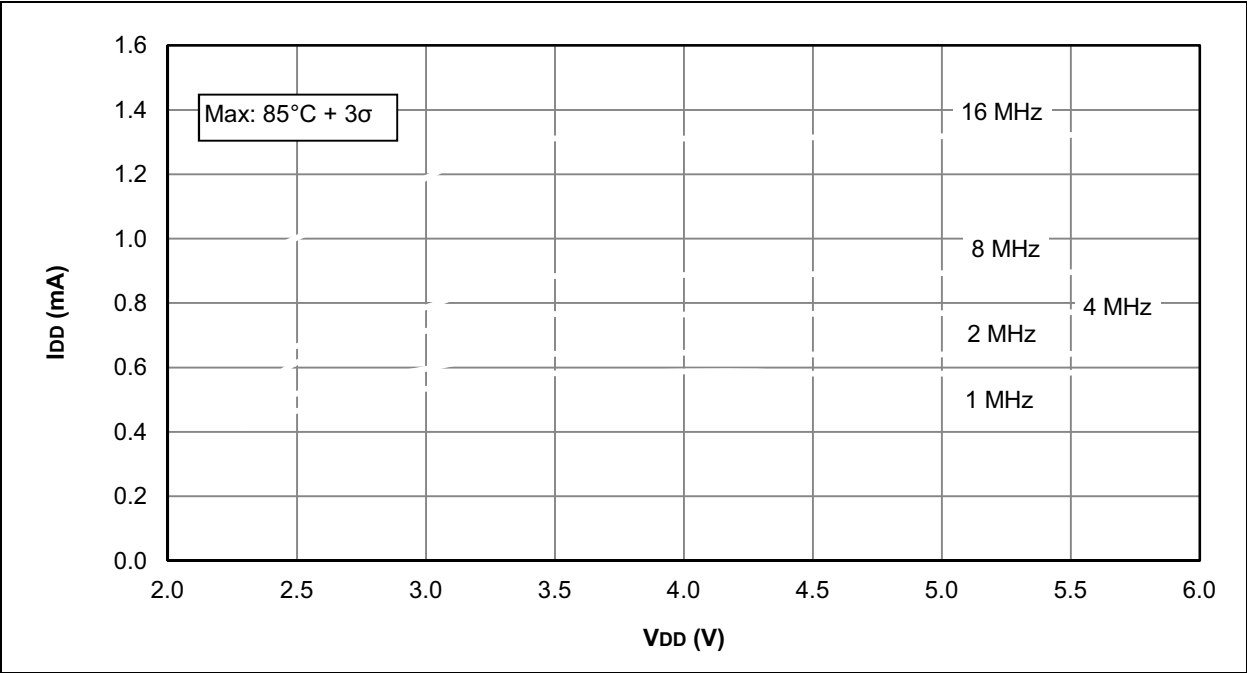

FIGURE 27-20: I_{DD} MAXIMUM, HFINTOSC, PIC12F1571/2 ONLY

PIC12(L)F1571/2

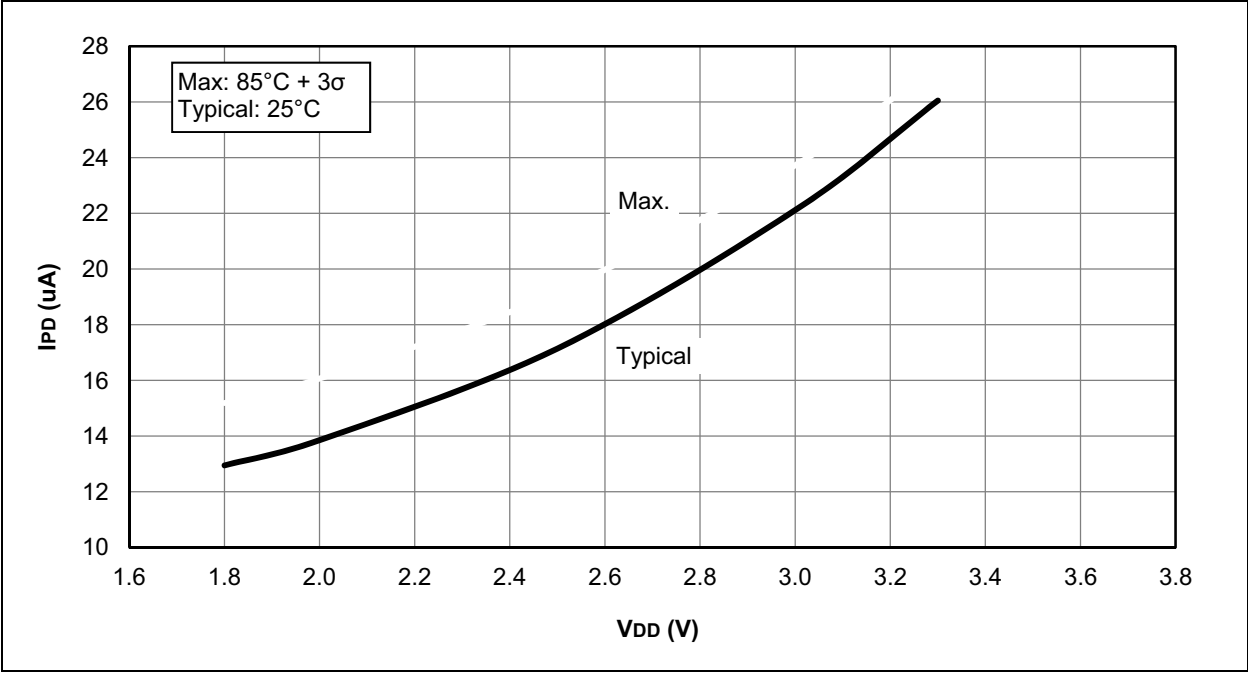**FIGURE 27-25:** **I**PD**, FIXED VOLTAGE REFERENCE (FVR), PIC12LF1571/2 ONLY**
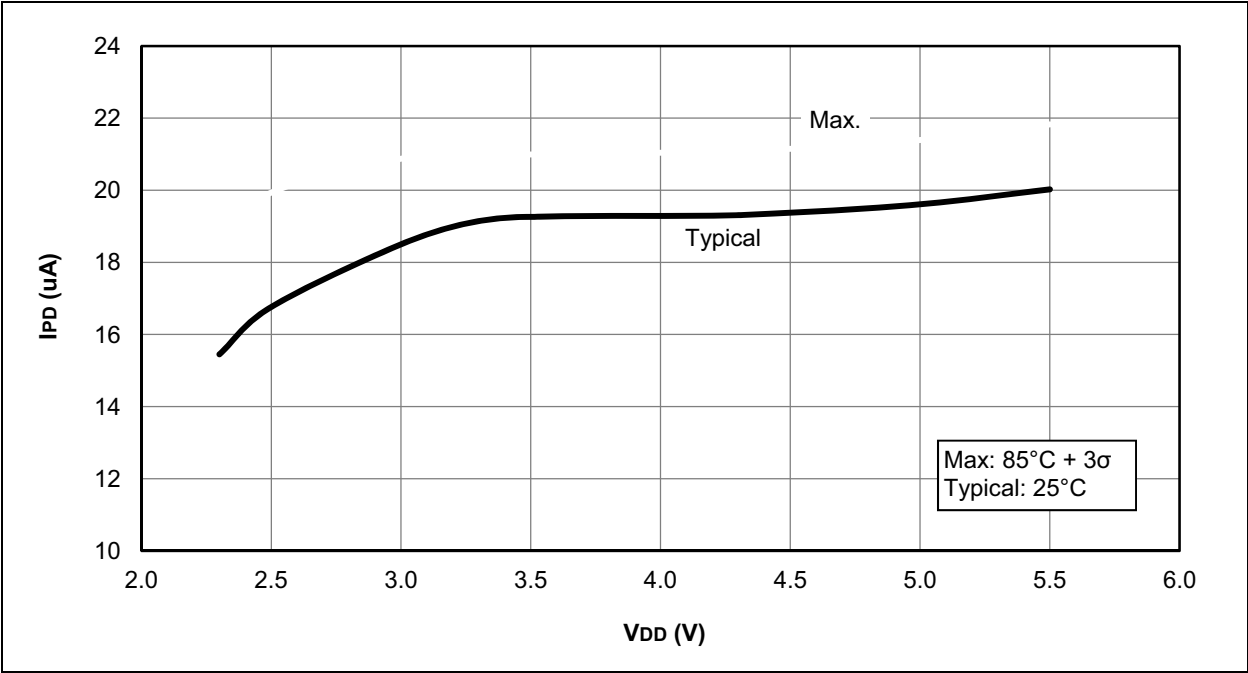

FIGURE 27-26: **I**PD**, FIXED VOLTAGE REFERENCE (FVR), PIC12F1571/2 ONLY**

FIGURE 27-38: FVR STABILIZATION PERIOD

Max: Typical + 3σ
Typical: statistical mean @ 25°C

Note:
The FVR Stabilization Period applies when:
1) coming out of RESET or exiting Sleep mode for PIC12/16LFxxxx devices.
2) when exiting sleep mode with VREGPM = 1 for PIC12/16Fxxxx devices
In all other cases, the FVR is stable when released from RESET.