Section 10. Power-Saving Modes

HIGHLIGHTS

This section of the manual contains the following topics:

10.1 INTRODUCTION

This section describes the Power-Saving modes of operation for the PIC32MX device family. The PIC32MX devices have nine Low-Power modes in two categories that allow the user to balance power consumption with device performance. In all of the modes listed below, the device can select the desired Power-Saving mode via software.

10.1.1 CPU Running Modes

In the CPU Running modes, the CPU is running and peripherals can optionally be switched ON or OFF.

- FRC RUN mode: the CPU is clocked from the FRC clock source with or without postscalers.
- LPRC RUN mode: the CPU is clocked from the LPRC clock source.
- SOSC RUN mode: the CPU is clocked from the SOSC clock source.
- Peripheral Bus Scaling mode:
  Peripherals are clocked at programmable fraction of the CPU clock (SYSCLK).

10.1.2 CPU Halted Modes

In the CPU Halted modes, the CPU is halted. Depending on the mode, peripherals can continue to operate or be halted as well.

- POSC IDLE mode: the system clock is derived from the POSC. The system clock source continues to operate.
  Peripherals continue to operate, but can optionally be individually disabled.
- FRC IDLE mode: the system clock is derived from the FRC with or without postscalers.
  Peripherals continue to operate, but can optionally be individually disabled.
- SOSC IDLE mode: the system clock is derived from the SOSC.
  Peripherals continue to operate, but can optionally be individually disabled.
- LPRC IDLE mode: the system clock is derived from the LPRC.
  Peripherals continue to operate, but can optionally be individually disabled. This is the lowest power mode for the device with a clock running.
- SLEEP Mode: the CPU, the system clock source, and any peripherals that operate from the system clock source, are halted.
  Some peripherals can operate in Sleep using specific clock sources. This is the lowest power mode for the device.

10.2 POWER-SAVING MODES CONTROL REGISTERS

Power-Saving modes control consists of the following Special Function Registers (SFRs):

- OSCCON: Control Register for the Oscillators Module

  OSCCONCLR, OSCCONSET, OSCCONINV: Atomic Bit Manipulation Write-only Registers for OSCCON
- WDTCON: Control Register for the Watchdog Timer Module

  WDTCONCLR, WDTCONSET, WDTCONINV: Atomic Bit Manipulation Write-only Registers for WDTCON
- RCON: Control Register for the Resets Module

  RCONCLR, RCONSET, RCONINV: Atomic Bit Manipulation Write-only Registers for RCON

The following table summarizes all Power-Saving-modes-related registers. Corresponding registers appear after the summary, followed by a detailed description of each register.

Table 10-1: Power-Saving Modes SFR Summary

Name		Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
OSCCON	31:24	—	—	PLLODIV<2:0>			FRCDIV<2:0>		
	23:16	—	SOSCRDY	—	PBDIV<1:0>		PLLMULT<2:0>		
	15:8	—	COSC<2:0>			—	NOSC<2:0>		
	7:0	CLKLOCK	ULOCK	LOCK	SLPEN	CF	UFRCEN	SOSCEN	OSWEN
OSCCONCLR	31:0	Write clears selected bits in OSCCON, read yields undefined value							
OSCCONSET	31:0	Write sets selected bits in OSCCON, read yields undefined value							
OSCCONINV	31:0	Write inverts selected bits in OSCCON, read yields undefined value							
WDTCON	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	ON	—	—	—	—	—	—	—
	7:0	—	SWDTPS<4:0>					—	WDTCLR
WDTCONCLR	31:0	Write clears selected bits in WDTCON; read yields undefined value							
WDTCONSET	31:0	Write sets selected bits in WDTCON; read yields undefined value							
WDTCONINV	31:0	Write inverts selected bits in WDTCON; read yields undefined value							
RCON	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	—	—	—	—	CM	VREGS
	7:0	EXTR	SWR	—	WDTO	SLEEP	IDLE	BOR	POR
RCONCLR	31:0	Write clears selected bits in RCON; read yields undefined value							
RCONSET	31:0	Write sets selected bits in RCON; read yields undefined value							
RCONINV	31:0	Write inverts selected bits in RCON; read yields undefined value							

Register 10-1: OSCCON: Oscillator Control Register

r-x	r-x	R/W-x	R/W-x	R/W-x	R/W-0	R/W-0	R/W-1
—	—	PLLODIV<2:0>			FRCDIV<2:0>		
bit 31							bit 24

r-x	R-0	r-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	SOSCRDY	—	PBDIV<1:0>		PLLMULT<2:0>		
bit 23							bit 16

r-x	R-0	R-0	R-0	r-x	R/W-x	R/W-x	R/W-x
—	COSC<2:0>			—	NOSC<2:0>		
bit 15							bit 8

R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-x	R/W-0
CLKLOCK	ULOCK	LOCK	SLPEN	CF	UFRCEN	SOSCEN	OSWEN
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 29-27 **PLLODIV<2:0>:** Output Divider for PLL bits

111 = PLL output divided by 256
110 = PLL output divided by 64
101 = PLL output divided by 32
100 = PLL output divided by 16
011 = PLL output divided by 8
010 = PLL output divided by 4
001 = PLL output divided by 2
000 = PLL output divided by 1

Note: On Reset these bits are set to the value of the FPLLODIV Configuration bits (DEVCFG2<18:16>)

bit 26-24 **FRCDIV<2:0>:** Fast Internal RC Clock Divider bits

111 = FRC divided by 256
110 = FRC divided by 64
101 = FRC divided by 32
100 = FRC divided by 16
011 = FRC divided by 8
010 = FRC divided by 4
001 = FRC divided by 2 (default setting)
000 = FRC divided by 1

bit 23 **Reserved:** Write ‘0’; ignore read

bit 22 **SOSCRDY:** Secondary Oscillator Ready Indicator bit

1 = Indicates that the Secondary Oscillator is running and is stable
0 = Secondary oscillator is either turned off or is still warming up

bit 21 **Unimplemented:** Read as ‘0’

Register 10-1: OSCCON: Oscillator Control Register (Continued)

bit 20-19 **PBDIV<1:0>:** Peripheral Bus Clock Divisor

11 = PBCLK is SYSCLK divided by 8(default)
10 = PBCLK is SYSCLK divided by 4
01 = PBCLK is SYSCLK divided by 2
00 = PBCLK is SYSCLK divided by 1
Note: On Reset these bits are set to the value of the Configuration bits (DEVCFG1<13:12>).

bit 18-16 **PLLMULT<2:0>:** PLL Multiplier bits

111 = Clock is multiplied by 24
110 = Clock is multiplied by 21
101 = Clock is multiplied by 20
100 = Clock is multiplied by 19
011 = Clock is multiplied by 18
010 = Clock is multiplied by 17
001 = Clock is multiplied by 16
000 = Clock is multiplied by 15
Note: On Reset these bits are set to the value of the PLLMULT Configuration bits (DEVCFG2<6:4>)

bit 15 **Reserved:** Write ‘0’; ignore read

bit 14-12 **COSC<2:0>:** Current Oscillator Selection bits

111 = Fast Internal RC Oscillator divided by OSCCON<FRCDIV> bits
110 = Fast Internal RC Oscillator divided by 16
101 = Low-Power Internal RC Oscillator (LPRC)
100 = Secondary Oscillator (SOSC)
011 = Primary Oscillator with PLL module (XTPLL, HSPLL or ECPLL)
010 = Primary Oscillator (XT, HS or EC)
001 = Fast RC Oscillator with PLL module via Postscaler (FRCPLL)
000 = Fast RC Oscillator (FRC)
Note: On Reset these bits are set to the value of the FNOSC Configuration bits (DEVCFG1<2:0>).

bit 11 **Reserved:** Write ‘0’; ignore read

bit 10-8 **NOSC<2:0>:** New Oscillator Selection bits

111 = Fast Internal RC Oscillator divided by OSCCON (FRCDIV> bits
110 = Fast Internal RC Oscillator divided by 16
101 = Low Power Internal RC Oscillator (LPRC)
100 = Secondary Oscillator (SOSC)
011 = Primary Oscillator with PLL module (XTPLL, HSPLL or ECPLL)
010 = Primary Oscillator (XT, HS or EC)
001 = Fast Internal RC Oscillator with PLL module via Postscaler (FRCPLL)
000 = Fast Internal RC Oscillator (FRC)
On Reset these bits are set to the value of the FNOSC Configuration bits (DEVCFG1<2:0>).

bit 7 **CLKLOCK:** Clock Selection Lock Enable bit

If FSCM is enabled (FCKSM1 = 1):
1 = Clock and PLL selections are locked
0 = Clock and PLL selections are not locked and may be modified
If FSCM is disabled (FCKSM1 = 0):
Clock and PLL selections are never locked and may be modified

bit 6 **ULOCK:** USB PLL Lock Status bit

1 = Indicates that the USB PLL module is in lock or USB PLL module start-up timer is satisfied
0 = Indicates that the USB PLL module is out of lock or USB PLL module start-up timer is in progress or USB PLL is disabled

bit 5 **LOCK:** PLL Lock Status bit

1 = PLL module is in lock or PLL module start-up timer is satisfied
0 = PLL module is out of lock, PLL start-up timer is running or PLL is disabled

bit 4 **SLPEN:** SLEEP Mode Enable bit

1 = Device will enter SLEEP mode when a WAIT instruction is executed
0 = Device will enter IDLE mode when a WAIT instruction is executed

Register 10-1: OSCCON: Oscillator Control Register (Continued)

bit 3	**CF:** Clock Fail Detect bit `1` = FSCM (Fail Safe Clock Monitor) has detected a clock failure `0` = No clock failure has been detected
bit 2	**UFRCEN:** USB FRC Clock Enable bit `1` = Enable FRC as the clock source for the USB clock source `0` = Use the primary oscillator or USB PLL as the USB clock source
bit 1	**SOSCEN:** 32.768 kHz Secondary Oscillator (SOSC) Enable bit `1` = Enable Secondary Oscillator `0` = Disable Secondary Oscillator **Note:** On Reset this bit is set to the value of the FSOSCEN Configuration bit (DEVCFG1<5>).
bit 0	**OSWEN:** Oscillator Switch Enable bit `1` = Initiate an oscillator switch to selection specified by NOSC2:NOSC0 bits `0` = Oscillator switch is complete

Register 10-2: OSCCONCLR: Programming Control Clear Register

Write clears selected bits in OSCCON, read yields undefined value	
bit 31	bit 0

bit 31-0 **Clears selected bits in OSCCON**

A write of '1' in one or more bit positions clears the corresponding bit(s) in OSCCON register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `OSCCONCLR = 0x00000001` will clear bit 0 in OSCCON register.

Register 10-3: OSCCONSET: Programming Control Set Register

Write sets selected bits in OSCCON, read yields undefined value	
bit 31	bit 0

bit 31-0 **Sets selected bits in OSCCON**

A write of '1' in one or more bit positions sets the corresponding bit(s) in OSCCON register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: OSCCONSET = `0x00000001` will set bit 0 in OSCCON register.

Register 10-4: OSCCONINV: Programming Control Invert Register

Write inverts selected bits in OSCCON, read yields undefined value	
bit 31	bit 0

bit 31-0 **Inverts selected bits in OSCCON**

A write of '1' in one or more bit positions inverts the corresponding bit(s) in OSCCON register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `OSCCONINV = 0x00000001` will invert bit 0 in OSCCON register.

Register 10-5: WDTCON: WATCHDOG TIMER CONTROL REGISTER

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	r-x	r-x	r-x	r-x	r-x	r-x	r-x
ON	—	—	—	—	—	—	—
bit 15							bit 8

r-x	R-x	R-x	R-x	R-x	R-x	r-0	R/W-0
—	SWDTPS<4:0>					—	WDTCLR
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 15 **ON:** Watchdog Peripheral On bit

1 = Watchdog peripheral is enabled. The status of other bits in the register are not affected by setting this bit. The LPRC oscillator will not be disabled when entering Sleep.

0 = Watchdog peripheral is disabled and not drawing current. SFR modifications are allowed. The status of other bits in this register are not affected by clearing this bit.

Note: When using 1:1 PBCLK divisor, the user’s software should not read/write the peripheral’s SFRs in the SYSCLK cycle immediately following the instruction that clears the module’s ON bit.

Register 10-6: WDTCONCLR: Comparator Control Clear Register

Write clears selected bits in WDTCON, read yields undefined value	
bit 31	bit 0

bit 31-0 **Clear selected bits in WDTCON**

A write of '`1`' in one or more bit positions clears the corresponding bit(s) in WDTCON register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register.

Example: `WDTCONCLR = 0x00008001` clears bits 15 and 0 in WDTCON register.

Register 10-7: WDTCONSET: Comparator Control Set Register

Write sets selected bits in WDTCON, read yields undefined value	
bit 31	bit 0

bit 31-0 **Set selected bits in WDTCON**

A write of '`1`' in one or more bit positions sets the corresponding bit(s) in WDTCON register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register.

Example: `WDTCONSET = 0x00008001` sets bits 15 and 0 in WDTCON register.

Register 10-8: WDTCONINV: Comparator Control Invert Register

Write inverts selected bits in WDTCON, read yields undefined value	
bit 31	bit 0

bit 31-0 **Inverts selected bits in WDTCON**

A write of '`1`' in one or more bit positions inverts the corresponding bit(s) in WDTCON register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register.

Example: `WDTCONINV = 0x00008001` inverts bits 15 and 0 in WDTCON register.

Register 10-9: RCON: RESETS CONTROL REGISTER

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	r-x	r-x	r-x	r-x	r-0	R/W-0	R/W-0
—	—	—	—	—	—	CM	VREGS
bit 15							bit 8

R/W-0	R/W-0	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EXTR	SWR	—	WDTO	SLEEP	IDLE	BOR	POR
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 3 **SLEEP:** Wake from Sleep bit

1 = The device woke up from SLEEP mode
0 = The device did not wake from SLEEP mode
Note: Must clear this bit to detect future wake ups from SLEEP.

bit 2 **IDLE:** Wake from IDLE bit

1 = The device woke up from IDLE mode
0 = The device did not wake from IDLE mode
Note: Must clear this bit to detect future wake ups from IDLE.

Register 10-10: RCONCLR: RCON Clear Register

Write clears selected bits in RCON, read yields undefined value	
bit 31	bit 0

bit 31-0	**Clears selected bits in RCON** A write of '`1`' in one or more bit positions clears the corresponding bit(s) in RCON register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register. **Example:** `RCONCLR = 0x0000000C` will clear bits 3 and 2 in RCON register.

Register 10-11: RCONSET: RCON Set Register

Write sets selected bits in RCON, read yields undefined value	
bit 31	bit 0

bit 31-0	**Sets selected bits in RCON** A write of '`1`' in one or more bit positions sets the corresponding bit(s) in RCON register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register. **Example:** `RCONSET = 0x0000000C` will set bits 3 and 2 in RCON register.

Register 10-12: RCONINV: RCON Invert Register

Write inverts selected bits in RCON, read yields undefined value	
bit 31	bit 0

bit 31-0	**Inverts selected bits in RCON** A write of '`1`' in one or more bit positions inverts the corresponding bit(s) in RCON register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register. **Example:** `RCONINV = 0x0000000C` will invert bits 3 and 2 in RCON register.

10.3 OPERATION OF POWER-SAVING MODES

Note:	In this manual, a distinction is made between a power mode as it is used in a specific module, and a power mode as it is used by the device, e.g., Sleep mode of the Comparator and SLEEP mode of the CPU. To indicate which type of power mode is intended, uppercase and lowercase letters (Sleep, Idle, Debug) signify a module power mode, and all uppercase letters (SLEEP, IDLE, DEBUG) signify a device power mode.

The PIC32MX device family has nine Power-Saving modes. The purpose of all the Power-Saving modes is to reduce power consumption by reducing the device clock frequency. To achieve this, multiple low-frequency clock sources can be selected. In addition, the peripherals and CPU can be halted or disabled to further reduce power consumption.

10.3.1 SLEEP Mode

SLEEP mode has the lowest power consumption of the device Power-Saving operating modes. The CPU and most peripherals are halted. Select peripherals can continue to operate in Sleep mode and can be used to wake the device from Sleep. See the individual peripheral module sections for descriptions of behavior in Sleep.

Some of the characteristics of SLEEP mode are as follows:

- The CPU is halted.
- The system clock source is typically shut down. See **10.3.1.1 “Oscillator Shutdown in SLEEP Mode”** for specific information.
- There can be a wake-up delay based on the oscillator selection (refer to Table 10-2).
- The Fail-Safe Clock Monitor (FSCM) does not operate during Sleep mode.
- The BOR circuit, if enabled, remains operative during SLEEP mode.
- The WDT, if enabled, is not automatically cleared prior to entering SLEEP mode.
- Some peripherals can continue to operate in SLEEP mode. These peripherals include I/O pins that detect a change in the input signal, WDT, RTCC, ADC, UART, and peripherals that use an external clock input or the internal LPRC oscillator.
- I/O pins continue to sink or source current in the same manner as they do when the device is not in Sleep.
- The USB module can override the disabling of the POSC or FRC. Refer to the USB section for specific details.
- Modules can be individually disabled by software prior to entering SLEEP in order to further reduce consumption.

The processor will exit, or ‘wake-up’, from SLEEP on one of the following events:

- On any interrupt from an enabled source that is operating in Sleep. The interrupt priority must be greater than the current CPU priority.
- On any form of device Reset.
- On a WDT time-out. See **10.4.2 “Wake-up from SLEEP or IDLE on Watchdog Time-out (NMI)”**.

If the interrupt priority is lower than or equal to current priority, the CPU will remain halted, but the PBCLK will start running and the device will enter in IDLE mode.

Refer Example 10-1 for example code.

Note:	There is no FRZ mode for this module.

10.3.1.1 Oscillator Shutdown in SLEEP Mode

The criteria for the device disabling the clock source in SLEEP are: the oscillator type, peripherals using the clock source, and (for select sources) the clock enable bit.

- If the CPU clock source is POSC, it is turned off in SLEEP. See Table 10-2 for applicable delays when waking from SLEEP. The USB module can override the disabling of the POSC or FRC. Refer to the USB section for specific details.
- If the CPU clock source is FRC, it is turned off in SLEEP. See Table 10-2 for applicable delays when waking from SLEEP. The USB module can override the disabling of the POSC or FRC. Refer to the USB section for specific details.
- If the CPU clock source is SOSC, it will be turned off if the SOSCEN bit is not set. See Table 10-2 for applicable delays when waking from SLEEP.
- If the CPU clock source is LPRC, it will be turned off if the clock source is not being used by a peripheral that will be operating SLEEP such as the WDT. See Table 10-2 for applicable delays when waking from SLEEP.

10.3.1.2 Clock Selection on Wake-up from SLEEP

The processor will resume code execution and use the same clock source that was active when SLEEP mode was entered. The device is subject to a start-up delay if a crystal oscillator and/or PLL is used as a clock source when the device exits SLEEP.

10.3.1.3 Delay on Wake-up from SLEEP

The oscillator start-up and Fail-Safe Clock Monitor delays, if enabled associated with waking up from SLEEP mode are shown in Table 10-2.

Table 10-2: Delay Times for Exit from Sleep Mode

Clock Source	Oscillator Delay	FSCM Delay
EC, EXTRC	—	—
EC + PLL	TLOCK	TFSCM
XT + PLL	TOST + TLOCK	TFSCM
XT, HS, XTL	TOST	TFSCM
LP (OFF during Sleep)	TOST	TFSCM
LP (ON during Sleep)	—	—
FRC, LPRC	—	—

Note: Please refer to the “Electrical Specifications” section of the PIC32MX device data sheet for TPOR, TFSCM and TLOCK specification values.

10.3.1.4 Wake-up from SLEEP Mode with Crystal Oscillator or PLL

If the system clock source is derived from a crystal oscillator and/or the PLL, then the Oscillator Start-up Timer (OST) and/or PLL lock times will be applied before the system clock source is made available to the device. As an exception to this rule, no oscillator delays are applied if the system clock source is the POSC oscillator and it was running while in SLEEP mode.

Note: In spite of the various delays applied the crystal oscillator (and PLL) may not be up and running at the end of the Tost, or Tlock delays. For proper operation the user must design the external oscillator circuit such that reliable oscillation will occur within the delay period.

10.3.1.5 Fail-Safe Clock Monitor (FSCM) Delay and SLEEP Mode

The FSCM does not operate while the device is in Sleep. If the FSCM is enabled it will resume operation when the device wakes from Sleep. A delay of T_{FSCM} is applied to allow the oscillator source to stabilize before the FSCM resumes monitoring.

the following conditions are true, a delay of T_{FSCM} will be applied when waking from SLEEP mode:

- The oscillator was shutdown while in SLEEP mode.
- The system clock is derived from a crystal oscillator source and/or the PLL.

In most cases, the T_{FSCM} delay provides time for the OST to expire and the PLL to stabilize before device execution resumes. If the FSCM is enabled, it will begin to monitor the system clock source after the T_{FSCM} delay expires.

10.3.1.6 Slow Oscillator Start-up

When an oscillator starts slowly, the OST and PLL lock times may not have expired before FSCM time out.

If the FSCM is enabled, then the device will detect this condition as a clock failure and a clock fail trap will occur. The device will switch to the FRC oscillator and the user can re-enable the crystal oscillator source in the clock failure Interrupt Service Routine.

If the FSCM is not enabled, then the device will simply not start executing code until the clock is stable. From the user’s perspective, the device will appear to be in SLEEP until the oscillator clock has started.

10.3.1.7 USB Peripheral Control of Oscillators in SLEEP Mode

The USB module, when active, will prevent the clock source it is using from being disabled when the device enters sleep. Though the oscillator remains active the CPU and peripherals will remain halted.

Example 10-1: Put Device in SLEEP, then Wake with WDT

```
// Code example to put the Device in sleep and then Wake the device
// with the WDT

OSCCONSET = 0x10;      // set Power-Saving mode to Sleep

WDTCONCLR = 0x0002;    // Disable WDT window mode
WDTCONSET = 0x8000;    // Enable WDT
                       // WDT timeout period is set in the device configuration

while (1)
{
    ... user code ...

    WDTCONSET = 0x01;  // service the WDT
    asm volatile( “wait” );// put device in selected Power-Saving mode

                       // code execution will resume here after wake

    ... user code ...
}

// The following code fragment is at the beginning of the ‘C’ start-up code

if ( RCON & 0x18 )
{
                       // The WDT caused a wake from Sleep
    asm volatile( “eret” );// return from interrupt
}
```

10.3.2 Peripheral Bus Scaling

Most of the peripherals on the device are clocked using the PBCLK. The peripheral bus can be scaled relative to the SYSCLK to minimize the dynamic power consumed by the peripherals. The PBCLK divisor is controlled by PBDIV<1:0> (OSCCON<20:19>), allowing SYSCLK-to-PBCLK ratios of 1:1, 1:2, 1:4, and 1:8. All peripherals using PBCLK are affected when the divisor is changed. Peripherals such as the Interrupt Controller, DMA, Bus Matrix, and Prefetch Cache are clocked directly from SYSCLK, as a result, they are not affected by PBCLK divisor changes.

Most of the peripherals on the device are clocked using the PBCLK. The peripheral bus can be scaled relative to the SYSCLK to minimize the dynamic power consumed by the peripherals. The PBCLK divisor is controlled by PBDIV<1:0> (OSCCON<20:19>), allowing SYSCLK-to-PBCLK ratios of 1:1, 1:2, 1:4, and 1:8. All peripherals using PBCLK are affected when the divisor is changed. Peripherals such as USB, Interrupt Controller, DMA, Bus Matrix, and Prefetch Cache are clocked directly from SYSCLK, as a result, they are not affected by PBCLK divisor changes

Changing the PBCLK divisor affects:

- The CPU to peripheral access latency. The CPU has to wait for next PBCLK edge for a read to complete. In 1:8 mode this results in a latency of one to seven SYSCLKs.
- The power consumption of the peripherals. Power consumption is directly proportional to the frequency at which the peripherals are clocked. The greater the divisor, the lower the power consumed by the peripherals.

To minimize dynamic power the PB divisor should be chosen to run the peripherals at the lowest frequency that provides acceptable system performance. When selecting a PBCLK divider, peripheral clock requirements such as baud rate accuracy should be taken into account. For example, the UART peripheral may not be able to achieve all baud rate values at some PBCLK divider depending on the SYSCLK value.

10.3.2.1 Dynamic Peripheral Bus Scaling

The PBCLK can be scaled dynamically, by software, to save additional power when the device is in a low activity mode. The following issues need to be taken into account when scaling the PBCLK:

- All the peripherals clocked from PBCLK will scale at the same ratio, at the same time. This needs to be accounted in peripherals which need to maintain a constant baud rate, or pulse period even in low-power modes.
- Any communication through a peripheral on the peripheral bus that is in progress when the PBCLK changes may cause a data or protocol error due to a frequency change during transmission or reception.

The following steps are recommended, if the user intends to scale the PBCLK divisor dynamically:

- Disable all communication peripherals whose baud rate will be affected. Care should be taken to ensure that no communication is currently in progress before disabling the peripherals as it may result in protocol errors.
- Update the Baud Rate Generator (BRG) settings for peripherals as required for operation at the new PBCLK frequency.
- Change the peripheral bus ratio to the desired value.
- Enable all communication peripherals whose baud rate were affected.

Note:	Modifying the peripheral baud rate is done by writing to the associated peripheral SFRs. To minimize latency, the peripherals should be modified in the mode where the PBCLK is running at its highest frequency.

Example 10-2: Changing the PB Clock Divisor

```
                                         // Code example to change the PBCLK divisor
                                         // This example is for a device running at 40 MHz
                                         // Make sure that there is no UART send/receive in
progress
... user code ...
U1BRG = 0x81;                            // set baud rate for UART1 for 9600
... user code ...
SYSKEY = 0x0;                            // write invalid key to force lock
SYSKEY = 0xAA996655;                     // Write Key1 to SYSKEY
SYSKEY = 0x556699AA;                     // Write Key2 to SYSKEY
OSCCONCLR = 0x3 << 19;                   // set PB divisor to minimum (1:1)
SYSKEY = 0x0;                            // write invalid key to force lock

... user code ...
                                         // Change Peripheral Clock value
U1BRG = 0x0F;                            // set baud rate for UART1 for 9600 based on

                                         // new PB clock frequency
SYSKEY = 0x0;                            // write invalid key to force lock
SYSKEY = 0xAA996655;                     // Write Key1 to SYSKEY
SYSKEY = 0x556699AA;                     // Write Key2 to SYSKEY
OSCCONSET = 0x3 << 19;                   // set PB divisor to maximum (1:8)
SYSKEY = 0x0;                            // write invalid key to force lock

                                         // Reset Peripheral Clock
SYSKEY = 0x0;                            // write invalid key to force lock
SYSKEY = 0xAA996655;                     // Write Key1 to SYSKEY
SYSKEY = 0x556699AA;                     // Write Key2 to SYSKEY
OSCCONCLR = 0x3 << 19;                   // set PB divisor to minimum (1:1)
SYSKEY = 0x0;                            // write invalid key to force lock

U1BRG = 0x81;                            // restore baud rate for UART1 to 9600 based
                                         // on new PB clock frequency
```

10.3.3 IDLE Modes

In the IDLE modes, the CPU is halted but the System clock (SYSCLK) source is still enabled. This allows peripherals to continue to operate when the CPU is halted. Peripherals can be individually configured to halt when entering IDLE by setting their respective SIDL bit. Latency when exiting Idle mode is very low due to the CPU oscillator source remaining active.

There are four Idle modes of operation: POSC IDLE, FRC IDLE, SOSC IDLE, and LPRC IDLE.

- POSC IDLE mode: The SYSCLK is derived from the POSC. The CPU is halted, but the SYSCLK source continues to operate. Peripherals continue to operate, but can optionally be individually disabled. If the PLL is used, the Multiplier value, PLLMULT<2:0> (OSCCON<18:16>), can also be lowered to reduce power consumption by peripherals.
- FRC IDLE mode: The SYSCLK is derived from the FRC. The CPU is halted. Peripherals continue to operate, but can optionally be individually disabled. If the PLL is used, the Multiplier value, PLLMULT<2:0> (OSCCON<18:16>), can also be lowered to reduce power consumption by peripherals. The FRC clock can be further divided by a postscaler using RCDIV<2:0> (OSCCON<26:24>).
- SOSC IDLE mode: The SYSCLK is derived from the SOSC. The CPU is halted. Peripherals continue to operate, but can optionally be individually disabled.
- LPRC IDLE. The SYSCLK is derived from the LPRC. The CPU is halted. Peripherals continue to operate, but can optionally be individually disabled.

Note: Changing the PBCLK divider ratio requires recalculation of peripheral timing. For example, assume the UART is configured for 9600 baud with a PB clock ratio of 1:1 and a POSC of 8 MHz. When the PB clock divisor of 1:2 is used, the input frequency to the baud clock is cut in half; therefore, the baud rate is reduced to 1/2 its former value. Due to numeric truncation in calculations (such as the baud rate divisor), the actual baud rate may be a tiny percentage different than expected. For this reason, any timing calculation required for a peripheral should be performed with the new PB clock frequency instead of scaling the previous value based on a change in PB divisor ratio.

Note: Oscillator start-up and PLL lock delays are applied when switching to a clock source that was disabled and that uses a crystal and/or the PLL. For example, assume the clock source is switched from POSC to LPRC just prior to entering Sleep in order to save power. No oscillator start-up delay would be applied when exiting Idle. However, when switching back to POSC, the appropriate PLL and or Oscillator startup/lock delays would be applied.

The device enters IDLE mode when the SLPEN (OSCCON<4>) bit is clear and a `WAIT` instruction is executed.

The processor will wake or exit from Idle mode on the following events:

- On any interrupt event for which the interrupt source is enabled. The priority of the interrupt event must be greater than the current priority of CPU. If the priority of the interrupt event is lower than or equal to current priority of CPU, the CPU will remain halted and the device will remain in IDLE mode.
- On any source of device Reset.
- On a WDT time-out interrupt. See **10.4.2 "Wake-up from SLEEP or IDLE on Watchdog Time-out (NMI)"** and **Section 9. "Watchdog Timer and Power-up Timer"**.

Example 10-3: Placing Device in IDLE and Waking by ADC Event

```
                          // Code example to put the Device in Idle and then Wake the device
                          // when the ADC completes a conversion
SYSKEY = 0x0;             // write invalid key to force lock
SYSKEY = 0xAA996655;      // Write Key1 to SYSKEY
SYSKEY = 0x556699AA;      // Write Key2 to SYSKEY
OSCCONCLR = 0x10;         // set Power-Saving mode to Idle
SYSKEY = 0x0;             // write invalid key to force lock

asm volatile ( "wait" );  // put device in selected Power-Saving mode
                          // code execution will resume here after wake and the ISR is
                             complete
... user code ...

                          // interrupt handler
void __ISR(27_ADC_VECTOR, ipl7) ADC_HANDLER(void)
{
                          // interrupt handler
unsigned long int result;

result = ADC1BUF0;        // read the result
IFS1CLR = 2;              // Clear ADC conversion interrupt flag
}
```

10.4 INTERRUPTS

There are two sources of interrupts that will wake the device from a Power-Saving mode: Peripheral interrupts and an Non-Maskable Interrupt (NMI) generated by the WDT in Power-Saving mode.

10.4.1 Wake-up from SLEEP or IDLE on Peripheral Interrupt

Any source of interrupt that is individually enabled using the corresponding IE control bit in the IECx register and is operational in the current Power-Saving mode will be able to wake up the processor from SLEEP or IDLE mode. When the device wakes, one of two events will occur based on the interrupt priority:

- If the assigned priority for the interrupt is less than or equal to the current CPU priority, the CPU will remain halted and the device enters or remains in IDLE mode.
- If the assigned priority level for the interrupt source is greater than the current CPU priority, the device will wake-up and the CPU will jump to the corresponding interrupt vector. Upon completion of the ISR, CPU will start executing the next instruction after `WAIT`.

The IDLE Status bit (RCON<2>) is set upon wake-up from IDLE mode. The SLEEP Status bit (RCON<3>) is set upon wake-up from SLEEP mode.

Note: A peripheral with an interrupt priority setting of Zero cannot wake the device.

Note: Any applicable oscillator start-up delays are applied before the CPU resumes code execution.

10.4.2 Wake-up from SLEEP or IDLE on Watchdog Time-out (NMI)

When the WDT times out in SLEEP or IDLE mode, an NMI is generated. The NMI causes the CPU code execution to jump to the device Reset vector. Although CPU executes Reset vector, it is not a device Reset – peripherals and most CPU registers do not change their states.

Note: Any applicable oscillator start-up delays are applied before the CPU resumes code execution.

To detect a wake from a Power-Saving mode caused by WDT expiration, the WDTO (RCON<4>), SLEEP (RCON<3>) and IDLE (RCON<2>) bits must be tested. If the WDTO bit is a ‘`1`’ the event was due to a WDT time-out. The SLEEP and IDLE bits can then be tested to determine if the WDT event occurred in Sleep or Idle.

To use a WDT time-out during SLEEP mode as a wake-up interrupt, a return from interrupt (`ERET`) instruction must be used in the start-up code after the event was determined to be a WDT wake-up. This will cause code execution to continue from the instruction following the `WAIT` instruction that put the device in Power-Saving mode.

Note: If a peripheral interrupt and WDT event occur simultaneously, or in close proximity, the NMI may not occur, due to the device being woken-up by the peripheral interrupt. To avoid unexpected WDT Reset in this scenario, the WDT is automatically cleared when the device awakens.

See **Section 9. “Watchdog Timer and Power-up Timer”** for detailed information on the WDT operation.

10.4.3 Interrupts Coincident with Power-Saving Instruction

Any peripheral interrupt that coincides with the execution of a `WAIT` instruction will be held off until entry into SLEEP or IDLE mode has completed. The device will then wake-up from SLEEP or IDLE mode.

10.5 I/O PINS ASSOCIATED WITH POWER-SAVING MODES

No device pins are associated with Power-Saving modes.

10.6 OPERATION IN DEBUG MODE

The user cannot change Clock modes when the debugger is active. Clock source changes due to the Fail-Safe Clock Monitor (FSCM) will still occur when the debugger is active.

10.7 RESETS

The behavior of the device after a Reset is determined by the type of Reset that occurred. For behavior related to the Power-Saving modes, Resets can be categorized into two groups: Power-on Reset (POR) and all other Resets (non POR).

10.7.1 Resets Other than POR During SLEEP or IDLE

The CPU will wake and code execution will begin at the device Reset vector. Any applicable oscillator delays will apply. The IDLE Status bit (RCON<2>) or SLEEP Status bit (RCON<3>) will be set to indicate the device was in a Power-Saving mode prior to the Reset.

10.7.2 POR Reset During SLEEP or IDLE

The CPU will wake and code execution will begin at the device Reset vector. Any applicable oscillator delays will apply. The IDLE Status bit (RCON<2>) or SLEEP Status bit (RCON<3>) will be forced clear. The power-saving state prior to the POR event is lost.

10.8 DESIGN TIPS

Question 1: ***What should my software do before entering SLEEP or IDLE mode?***

Answer: Make sure that the sources intended to wake the device have their IE bits set. In addition, make sure that the particular source of interrupt has the ability to wake the device. Some sources do not function when the device is in SLEEP mode.

If the device is to be placed in Idle mode, make sure that the ‘stop-in-idle’ control bit for each device peripheral is properly set. These control bits determine whether the peripheral will continue operation in IDLE mode. See the individual peripheral sections of this manual for further details.

Clear the WDT before entering SLEEP. If in Window mode, the WDT can only be cleared within the window period to prevent a device Reset.

Question 2: ***How do I determine which peripheral woke the device from SLEEP or IDLE mode?***

Answer: Most peripherals have a unique interrupt vector. If needed, you can poll the IF bits for each enabled interrupt source to determine the source of wake-up.

10.9 RELATED APPLICATION NOTES

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the PIC32MX device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Power-Saving modes include the following:

Title	**Application Note #**
Low-Power Design using PIC® Microcontrollers	AN606

Note:	Please visit the Microchip web site (www.microchip.com) for additional application notes and code examples for the PIC32MX family of devices.

10.10 REVISION HISTORY

Revision A (October 2007)

This is the initial released version of this document.

Revision B (October 2007)

Updated document to remove Confidential status.

Revision C (April 2008)

Revised status to Preliminary; Revised U-0 to r-x.

Revision D (July 2008)

Revised Example 10-1 and 10-3; Revised Table 10-1; Revised Register 10-5 and 10-9; Revised Section 10.3.2 (2nd para.); Change Reserved bits from “Maintain as” to “Write”; Added Note to ON bit (WDTCON Register).

Revision E (July 2008)

Revised Examples 10-2, 10-3.

NOTES: