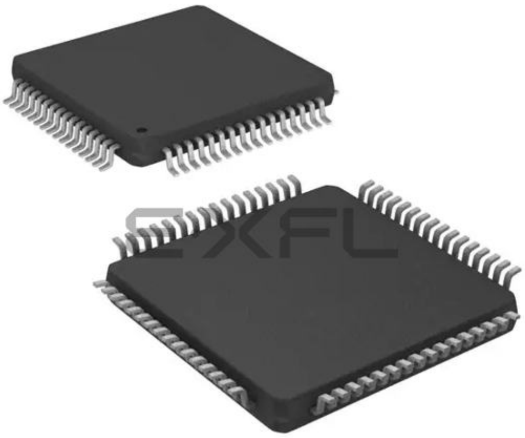


What is "Embedded - Microcontrollers"?

"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, LCD, POR, PWM, WDT
Number of I/O	53
Program Memory Size	128KB (64K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 16x12b
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/pic18f67k90t-i-pt

1.0 DEVICE OVERVIEW

This document contains device-specific information for the following devices:

- PIC18F65K90
- PIC18F66K90
- PIC18F67K90
- PIC18F85K90
- PIC18F86K90
- PIC18F87K90

This family combines the traditional advantages of all PIC18 microcontrollers – namely, high computational performance and a rich feature set – with a versatile on-chip LCD driver, while maintaining an extremely competitive price point. These features make the PIC18F87K90 family a logical choice for many high-performance applications where price is a primary consideration.

1.1 Core Features

1.1.1 nanoWatt TECHNOLOGY

All of the devices in the PIC18F87K90 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Alternate Run Modes:** By clocking the controller from the Timer1 source or the internal RC oscillator, power consumption during code execution can be reduced.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further.
- **On-the-Fly Mode Switching:** The power-managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application's software design.
- **nanoWatt XLP:** An extra low-power BOR, RTCC and low-power Watchdog Timer. Also, an ultra low-power regulator for Sleep mode is provided in regulator-enabled modes.

1.1.2 OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F87K90 family offer different oscillator options, allowing users a range of choices in developing application hardware. These include:

- External Resistor/Capacitor (RC); RA6 available
- External Resistor/Capacitor with Clock Out (RCIO)
- Three External Clock modes:
  - External Clock (EC); RA6 available
  - External Clock with Clock Out (ECIO)
  - External Crystal (XT, HS, LP)
- A Phase Lock Loop (PLL) frequency multiplier, available to the External Oscillator modes which allows clock speeds of up to 64 MHz. PLL can also be used with the internal oscillator.
- An internal oscillator block that provides a 16 MHz clock (±2% accuracy) and an INTRC source (approximately 31 kHz, stable over temperature and V_{DD})
  - Operates as HF-INTOSC or MF-INTOSC when block selected for 16 MHz or 500 kHz
  - Frees the two oscillator pins for use as additional general purpose I/O

The internal oscillator block provides a stable reference source that gives the family additional features for robust operation:

- **Fail-Safe Clock Monitor:** This option constantly monitors the main clock source against a reference signal provided by the internal oscillator. If a clock failure occurs, the controller is switched to the internal oscillator, allowing for continued low-speed operation or a safe application shutdown.
- **Two-Speed Start-up:** This option allows the internal oscillator to serve as the clock source from Power-on Reset, or wake-up from Sleep mode, until the primary clock source is available.

1.1.3 MEMORY OPTIONS

The PIC18F87K90 family provides ample room for application code, from 32 Kbytes to 128 Kbytes of code space. The Flash cells for program memory are rated to last up to 10,000 erase/write cycles. Data retention without refresh is conservatively estimated to be greater than 40 years.

The Flash program memory is readable and writable. During normal operation, the PIC18F87K90 family also provides plenty of room for dynamic application data with up to 3,828 bytes of data RAM.

1.1.4 EXTENDED INSTRUCTION SET

The PIC18F87K90 family implements the optional extension to the PIC18 instruction set, adding 8 new instructions and an Indexed Addressing mode. Enabled as a device configuration option, the extension has been specifically designed to optimize re-entrant application code originally developed in high-level languages, such as 'C'.

1.1.5 EASY MIGRATION

Regardless of the memory size, all devices share the same rich set of peripherals (except the 32-Kbyte parts, which have two less CCPs and three less Timers), allowing for a smooth migration path as applications grow and evolve.

The consistent pinout scheme, used throughout the entire family, also aids in migrating to the next larger device. This is true when moving between the 64-pin members, between the 80-pin members, or even jumping from 64-pin to 80-pin devices.

3.2 Control Registers

The OSCCON register (Register 3-1) controls the main aspects of the device clock's operation. It selects the oscillator type to be used, which of the power-managed modes to invoke and the output frequency of the INTOSC source. It also provides status on the oscillators.

The OSCTUNE register (Register 3-3) controls the tuning and operation of the internal oscillator block. It also implements the PLLEN bit which controls the operation of the Phase Locked Loop (PLL) (see **Section 3.5.2 "PLL Frequency Multiplier"**).

REGISTER 3-1: OSCCON: OSCILLATOR CONTROL REGISTER

R/W-0	R/W-1	R/W-1	R/W-0	R[1]	R-0	R/W-0	R/W-0
IDLEN	IRCF2[2]	IRCF1[2]	IRCF0[2]	OSTS	HFIOFS	SCS1[4]	SCS0[4]
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 **IDLEN:** Idle Enable bit

1 = Device enters an Idle mode when a SLEEP instruction is executed
0 = Device enters Sleep mode when a SLEEP instruction is executed

bit 6-4 **IRCF<2:0>:** Internal Oscillator Frequency Select bits[2]

111 = HF-INTOSC output frequency is used (16 MHz)
110 = HF-INTOSC/2 output frequency is used (8 MHz, default)
101 = HF-INTOSC/4 output frequency is used (4 MHz)
100 = HF-INTOSC/8 output frequency is used (2 MHz)
011 = HF-INTOSC/16 output frequency is used (1 MHz)

If INTSRC = 0 and MFIOSEL = 0:[3,5]
010 = HF-INTOSC/32 output frequency is used (500 kHz)
001 = HF-INTOSC/64 output frequency is used (250 kHz)
000 = LF-INTOSC output frequency is used (31.25 kHz)

If INTSRC = 1 and MFIOSEL = 0:[3,5]
010 = HF-INTOSC/32 output frequency is used (500 kHz)
001 = HF-INTOSC/64 output frequency is used (250 kHz)
000 = HF-INTOSC/512 output frequency is used (31.25 kHz)

If INTSRC = 0 and MFIOSEL = 1:[3,5]
010 = MF-INTOSC output frequency is used (500 kHz)
001 = MF-INTOSC/2 output frequency is used (250 kHz)
000 = LF-INTOSC output frequency is used (31.25 kHz)

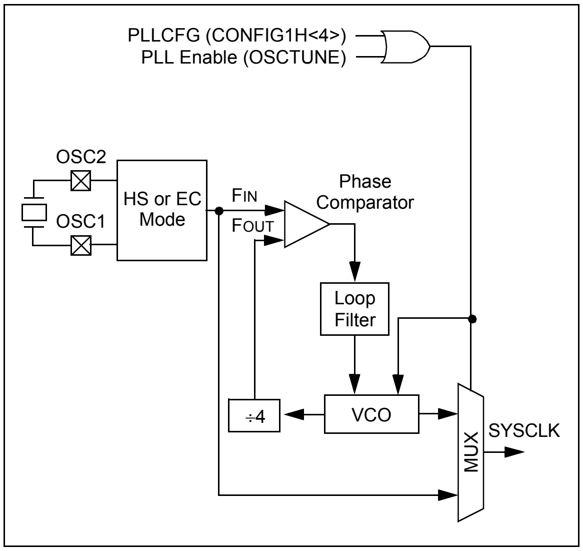
If INTSRC = 1 and MFIOSEL = 1:[3,5]
010 = MF-INTOSC output frequency is used (500 kHz)
001 = MF-INTOSC/2 output frequency is used (250 kHz)
000 = MF-INTOSC/16 output frequency is used (31.25 kHz)

bit 3 **OSTS:** Oscillator Start-up Timer Time-out Status bit[1]

1 = Oscillator Start-up Timer (OST) time-out has expired: primary oscillator is running as defined by OSC<3:0>
0 = Oscillator Start-up Timer (OST) time-out is running: primary oscillator is not ready; device is running from an internal oscillator (HF-INTOSC, MF-INTOSC or LF-INTOSC)

Note 1: Reset state depends on the state of the IESO Configuration bit (CONFIG1H<7>).
2: Modifying these bits will cause an immediate clock frequency switch if the internal oscillator is providing the device clocks.
3: Source selected by the INTSRC bit (OSCTUNE<7>).
4: Modifying these bits will cause an immediate clock source switch.
5: INTSRC = OSCTUNE<7> and MFIOSEL = OSCCON2<0>.

FIGURE 3-7: PLL BLOCK DIAGRAM



3.5.2.2 PLL and HF-INTOSC

The PLL is available to the internal oscillator block when the internal oscillator block is configured as the primary clock source. In this configuration, the PLL is enabled in software and generates a clock output of up to 64 MHz.

The operation of INTOSC with the PLL is described in **Section 3.6.2 "INTPLL Modes"**. Care should be taken that the PLL is enabled only if the HF-INTOSC postscaler is configured for 8 MHz or 16 MHz.

3.6 Internal Oscillator Block

The PIC18F87K90 family of devices includes an internal oscillator block which generates two different clock signals. Either clock can be used as the microcontroller's clock source, which may eliminate the need for an external oscillator circuit on the OSC1 and/or OSC2 pins.

The internal oscillator consists of three blocks, depending on the frequency of operation. They are HF-INTOSC, MF-INTOSC and LF-INTRC.

In HF-INTOSC mode, the internal oscillator can provide a frequency, ranging from 31 kHz to 16 MHz, with the postscaler deciding the selected frequency (IRCF<2:0>).

The INTSRC bit (OSCTUNE<7>) and MFIOSEL bit (OSCCON2<0>) also decide which INTOSC provides the lower frequency (500 kHz to 31 kHz). For the HF-INTOSC to provide these frequencies, INTSRC = 1 and MFI0SEL = 0.

In HF-INTOSC, the postscaler (IRCF<2:0>) provides the frequency range of 31 kHz to 16 MHz. If HF-INTOSC is used with the PLL, the input frequency to the PLL should be 8 MHz or 16 MHz (IRCF<2:0> = 111, 110).

For MF-INTOSC mode to provide a frequency range of 500 kHz to 31 kHz, INTSRC = 1 and MFIOSEL = 1. The postscaler (IRCF<2:0>), in this mode, provides the frequency range of 31 kHz to 500 kHz.

The LF-INTRC can provide only 31 kHz if INTSRC = 0.

The LF-INTRC provides 31 kHz and is enabled if it is selected as the device clock source. The mode is enabled automatically when any of the following is enabled:

- Power-up Timer
- Fail-Safe Clock Monitor
- Watchdog Timer
- Two-Speed Start-up

These features are discussed in greater detail in **Section 28.0 "Special Features of the CPU"**.

The clock source frequency (HF-INTOSC, MF-INTOSC or LF-INTRC direct) is selected by configuring the IRCF bits of the OSCCON register, as well the INTSRC and MFIOSEL bits. The default frequency on device Resets is 8 MHz.

3.6.1 INTIO MODES

Using the internal oscillator as the clock source eliminates the need for up to two external oscillator pins, which can then be used for digital I/O. Two distinct oscillator configurations, which are determined by the OSC Configuration bits, are available:

- In INTIO1 mode, the OSC2 pin (RA6) outputs FOSC/4, while OSC1 functions as RA7 (see Figure 3-8) for digital input and output.
- In INTIO2 mode, OSC1 functions as RA7 and OSC2 functions as RA6 (see Figure 3-9). Both are available as digital input and output ports.

FIGURE 3-8: INTIO1 OSCILLATOR MODE

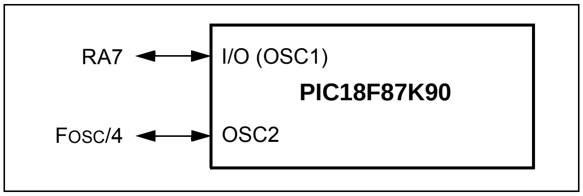
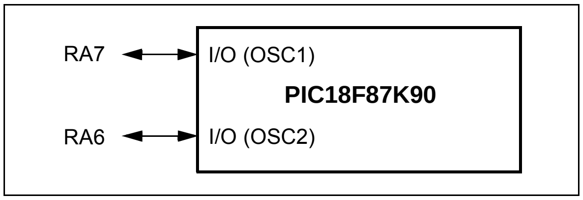


FIGURE 3-9: INTIO2 OSCILLATOR MODE



REGISTER 4-3: PMD1: PERIPHERAL MODULE DISABLE REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	CTMUMD	RTCCMD[1]	TMR4MD	TMR3MD	TMR2MD	TMR1MD	—
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 **Unimplemented:** Read as ‘0’

bit 6 **CTMUMD:** PMD CTMU Enable/Disable bit
1 = Peripheral Module Disable (PMD) is enabled for CMTU, disabling all of its clock sources
0 = PMD is disabled for CMTU

bit 5 **RTCCMD:** PMD RTCC Enable/Disable bit[1]
1 = PMD is enabled for RTCC, disabling all of its clock sources
0 = PMD is disabled for RTCC

bit 4 **TMR4MD:** TMR4MD Disable bit
1 = PMD is enabled and all TMR4MD clock sources are disabled
0 = PMD is disabled and TMR4MD is enabled

bit 3 **TMR3MD:** TMR3MD Disable bit
1 = PMD is enabled and all TMR3MD clock sources are disabled
0 = PMD is disabled and TMR3MD is enabled

bit 2 **TMR2MD:** TMR2MD Disable bit
1 = PMD is enabled and all TMR2MD clock sources are disabled
0 = PMD is disabled and TMR2MD is enabled

bit 1 **TMR1MD:** TMR1MD Disable bit
1 = PMD is enabled and all TMR1MD clock sources are disabled
0 = PMD is disabled and TMR1MD is enabled

bit 0 **Unimplemented:** Read as ‘0’

Note 1: RTCCMD can only be set to ‘1’ after an EECON2 unlock sequence. Refer to **Section 17.0 “Real-Time Clock and Calendar (RTCC)”** for the unlock sequence (see Example 17-1).

6.2 PIC18 Instruction Cycle

6.2.1 CLOCKING SCHEME

The microcontroller clock input, whether from an internal or external source, is internally divided by four to generate four non-overlapping, quadrature clocks (Q1, Q2, Q3 and Q4). Internally, the Program Counter is incremented on every Q1, with the instruction fetched from the program memory and latched into the Instruction Register (IR) during Q4.

The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 6-4.

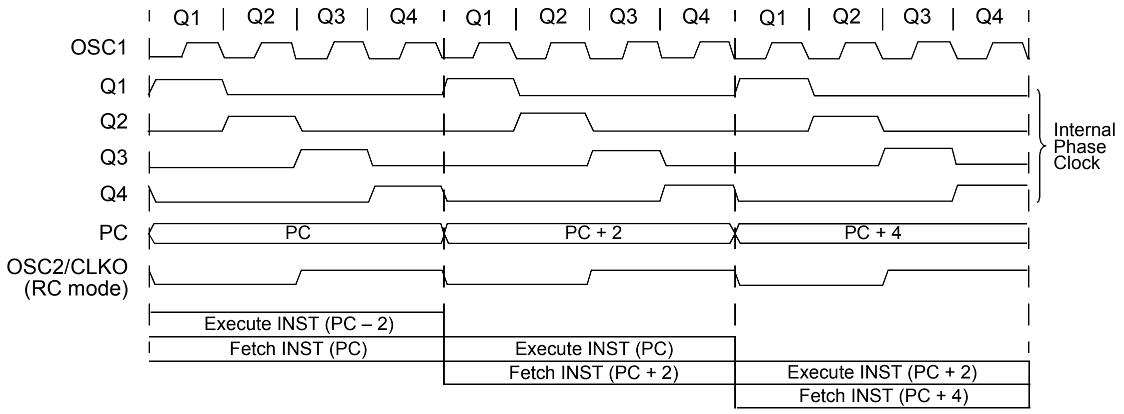
6.2.2 INSTRUCTION FLOW/PIPELINING

An “Instruction Cycle” consists of four Q cycles, Q1 through Q4. The instruction fetch and execute are pipelined in such a manner that a fetch takes one instruction cycle, while the decode and execute take another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction (such as `GOTO`) causes the Program Counter to change, two cycles are required to complete the instruction. (See Example 6-3.)

A fetch cycle begins with the Program Counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the Instruction Register (IR) in cycle Q1. This instruction is then decoded and executed during the Q2, Q3 and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

FIGURE 6-4: CLOCK/INSTRUCTION CYCLE



EXAMPLE 6-3: INSTRUCTION PIPELINE FLOW

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
`1. MOVLW 55h`	Fetch 1	Execute 1				
`2. MOVWF PORTB`		Fetch 2	Execute 2			
`3. BRA   SUB_1`			Fetch 3	Execute 3		
`4. BSF   PORTA, BIT3 (Forced NOP)`				Fetch 4	Flush (`NOP`)	
`5. Instruction @ address SUB_1`					Fetch SUB_1	Execute SUB_1

All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is “flushed” from the pipeline while the new instruction is being fetched and then executed.

6.4.3.1 FSR Registers and the INDF Operand

At the core of Indirect Addressing are three sets of registers: FSR0, FSR1 and FSR2. Each represents a pair of 8-bit registers: FSRnH and FSRnL. The four upper bits of the FSRnH register are not used, so each FSR pair holds a 12-bit value. This represents a value that can address the entire range of the data memory in a linear fashion. The FSR register pairs, then, serve as pointers to data memory locations.

Indirect Addressing is accomplished with a set of Indirect File Operands, INDF0 through INDF2. These can be thought of as "virtual" registers. The operands are mapped in the SFR space, but are not physically implemented. Reading or writing to a particular INDF register actually accesses its corresponding FSR register pair. A read from INDF1, for example, reads the data at the address indicated by FSR1H:FSR1L.

Instructions that use the INDF registers as operands actually use the contents of their corresponding FSR as a pointer to the instruction's target. The INDF operand is just a convenient way of using the pointer.

Because Indirect Addressing uses a full 12-bit address, data RAM banking is not necessary. Thus, the current contents of the BSR and the Access RAM bit have no effect on determining the target address.

FIGURE 6-8: INDIRECT ADDRESSING

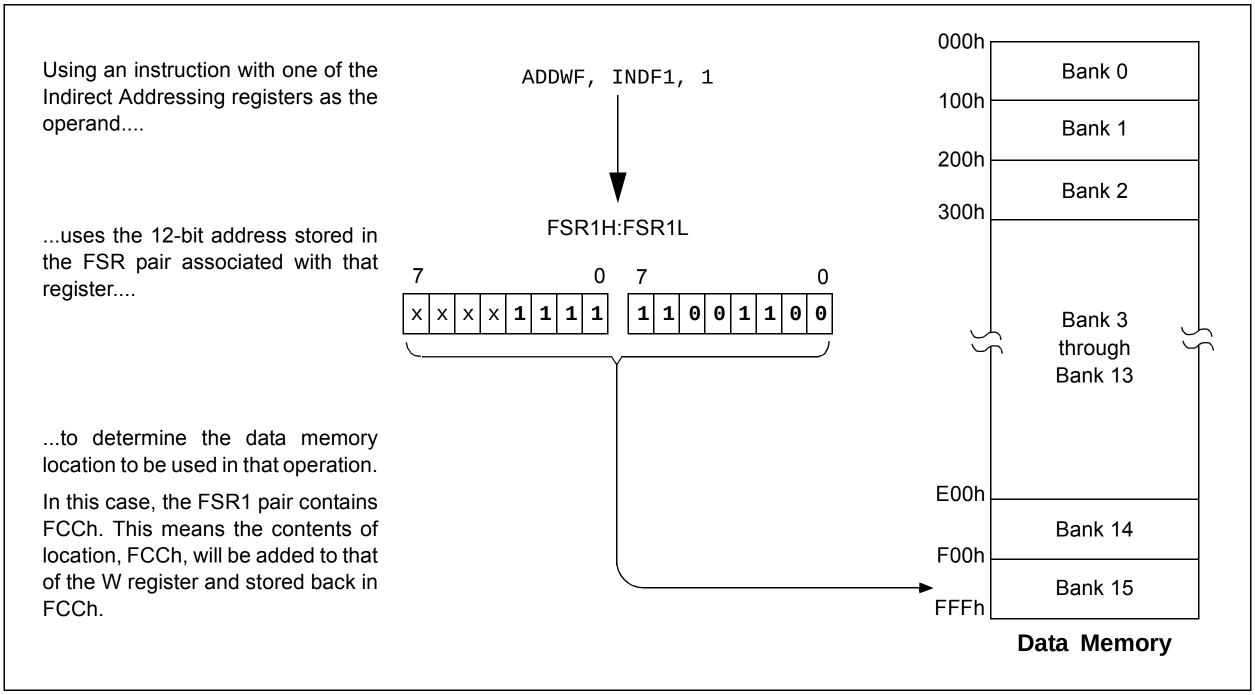

Using an instruction with one of the Indirect Addressing registers as the operand....
ADDWF, INDF1, 1
...uses the 12-bit address stored in the FSR pair associated with that register....
FSR1H:FSR1L
7
0
7
0
x x x x 1 1 1 1
1 1 0 0 1 1 0 0
...to determine the data memory location to be used in that operation.
In this case, the FSR1 pair contains FCCh. This means the contents of location, FCCh, will be added to that of the W register and stored back in FCCh.
000h
Bank 0
100h
Bank 1
200h
Bank 2
300h
Bank 3 through Bank 13
E00h
Bank 14
F00h
Bank 15
FFFh
Data Memory

TABLE 15-5: REGISTERS ASSOCIATED WITH TIMER3/5/7 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	75
PIR5	TMR7GIF(1)	TMR12IF(1)	TMR10IF(1)	TMR8IF	TMR7IF(1)	TMR6IF	TMR5IF	TMR4IF	77
PIE5	TMR7GIE(1)	TMR12IE(1)	TMR10IE(1)	TMR8IE	TMR7IE(1)	TMR6IE	TMR5IE	TMR4IE	77
PIR2	OSCFIF	—	SSP2IF	BCL2IF	BCL1IF	HLVDIF	TMR3IF	TMR3GIF	77
PIE2	OSCFIE	—	SSP2IE	BCL2IE	BCL1IE	HLVDIE	TMR3IE	TMR3GIE	77
PIR3	TMR5GIF(1)	LCDIF	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	RTCCIF	77
PIE3	TMR5GIE(1)	LCDIE	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	RTCCIE	77
TMR3H	Timer3 Register High Byte								77
TMR3L	Timer3 Register Low Byte								77
T3GCON	TMR3GE	T3GPOL	T3GTM	T3GSPM	T3GGO/$\overline{\text{T3DONE}}$	T3GVAL	T3GSS1	T3GSS0	77
T3CON	TMR3CS1	TMR3CS0	T3CKPS1	T3CKPS0	SOSCEN	$\overline{\text{T3SYNC}}$	RD16	TMR3ON	77
TMR5H	Timer5 Register High Byte								82
TMR5L	Timer5 Register Low Byte								82
T5GCON	TMR5GE	T5GPOL	T5GTM	T5GSPM	T5GGO/$\overline{\text{T5DONE}}$	T5GVAL	T5GSS1	T5GSS0	82
T5CON	TMR5CS1	TMR5CS0	T5CKPS1	T5CKPS0	SOSCEN	$\overline{\text{T5SYNC}}$	RD16	TMR5ON	82
TMR7H(1)	Timer7 Register High Byte								81
TMR7L(1)	Timer7 Register Low Byte								81
T7GCON(1)	TMR7GE	T7GPOL	T7GTM	T7GSPM	T7GGO/$\overline{\text{T7DONE}}$	T7GVAL	T7GSS1	T7GSS0	81
T7CON(1)	TMR7CS1	TMR7CS0	T7CKPS1	T7CKPS0	SOSCEN	$\overline{\text{T7SYNC}}$	RD16	TMR7ON	81
OSCCON2	—	SOSCRUN	—	—	SOSCGO	—	MFIOFS	MFIOSEL	79
CCPTMRS0	C3TSEL1	C3TSEL0	C2TSEL2	C2TSEL1	C2TSEL0	C1TSEL2	C1TSEL1	C1TSEL0	81
CCPTMRS1	C7TSEL1	C7TSEL0	—	C6TSEL0	—	C5TSEL0	C4TSEL1	C4TSEL0	81
CCPTMRS1	—	—	—	C10TSEL0	—	C9TSEL0	C8TSEL1	C8TSEL0	81

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used by the Timer3/5/7 modules.

Note 1: Unimplemented in devices with a program memory of 32 Kbytes (PIC18FX5K90).

17.0 REAL-TIME CLOCK AND CALENDAR (RTCC)

The key features of the Real-Time Clock and Calendar (RTCC) module are:

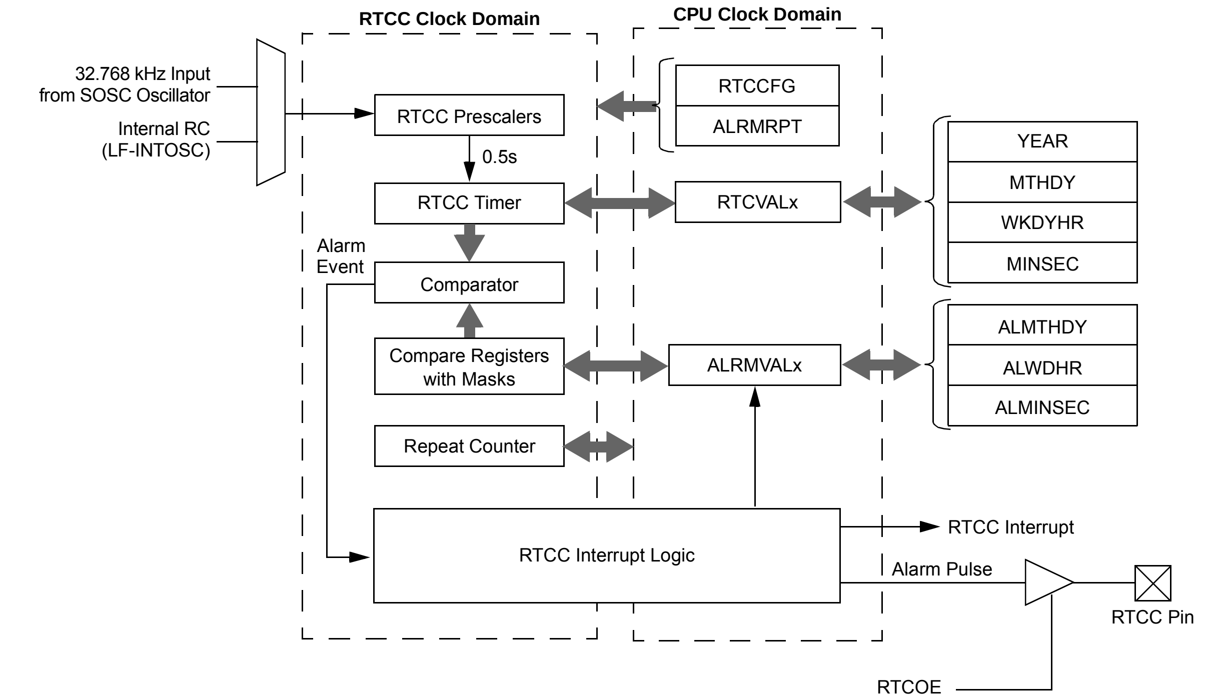
- Time: hours, minutes and seconds
- Twenty-four hour format (military time)
- Calendar: weekday, date, month and year
- Alarm configurable
- Year range: 2000 to 2099
- Leap year correction
- BCD format for compact firmware
- Optimized for low-power operation
- User calibration with auto-adjust
- Calibration range: ±2.64 seconds error per month
- Requirements: external 32.768 kHz clock crystal
- Alarm pulse or seconds clock output on RTCC pin

The RTCC module is intended for applications where accurate time must be maintained for an extended period with minimum to no intervention from the CPU. The module is optimized for low-power usage in order to provide extended battery life while keeping track of time.

The module is a 100-year clock and calendar with automatic leap year detection. The range of the clock is from 00:00:00 (midnight) on January 1, 2000 to 23:59:59 on December 31, 2099.

Hours are measured in 24-hour (military time) format. The clock provides a granularity of one second with half-second visibility to the user.

FIGURE 17-1: RTCC BLOCK DIAGRAM



NOTES:

REGISTER 18-3: CCPTMRS2: CCPx TIMER SELECT REGISTER 2

U-0	U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	C10TSEL0[1]	—	C9TSEL0[1]	C8TSEL1	C8TSEL0
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 7-5 **Unimplemented:** Read as ‘0’

bit 4 **C10TSEL0:** CCP10 Timer Selection bit[1]
0 = CCP10 is based off of TMR1/TMR2
1 = CCP10 is based off of TMR7/TMR2

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **C9TSEL0:** CCP9 Timer Selection bit[1]
0 = CCP9 is based off of TMR1/TMR2
1 = CCP9 is based off of TMR7/TMR4

bit 1-0 **C8TSEL<1:0>:** CCP8 Timer Selection bits
On non 32-Kbyte device variants:
00 = CCP8 is based off of TMR1/TMR2
01 = CCP8 is based off of TMR7/TMR4
10 = CCP8 is based off of TMR7/TMR6
11 = Reserved; do not use
On 32-Kbyte device variants (PIC18F85K90/65K90:
00 = CCP8 is based off of TMR1/TMR2
01 = CCP8 is based off of TMR1/TMR4
10 = CCP8 is based off of TMR1/TMR6
11 = Reserved; do not use

Note 1: This bit is unimplemented and reads as ‘0’ on devices with 32 Kbytes of program memory (PIC18FX5K90).

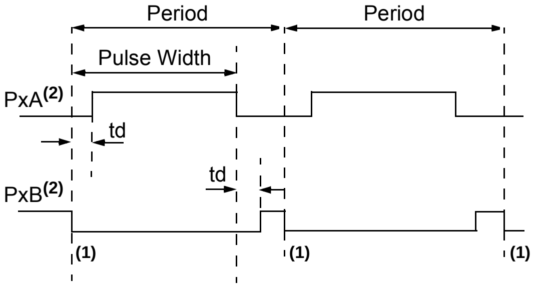
19.4.1 HALF-BRIDGE MODE

In Half-Bridge mode, two pins are used as outputs to drive push-pull loads. The PWM output signal is output on the PxA pin, while the complementary PWM output signal is output on the PxB pin (see Figure 19-6). This mode can be used for half-bridge applications, as shown in Figure 19-7, or for full-bridge applications, where four power switches are being modulated with two PWM signals.

In Half-Bridge mode, the programmable dead-band delay can be used to prevent shoot-through current in half-bridge power devices. The value of the PxDC<6:0> bits of the ECCPxDEL register sets the number of instruction cycles before the output is driven active. If the value is greater than the duty cycle, the corresponding output remains inactive during the entire cycle. For more details on the dead-band delay operations, see **Section 19.4.6 "Programmable Dead-Band Delay Mode"**.

Since the PxA and PxB outputs are multiplexed with the PORT data latches, the associated TRIS bits must be cleared to configure PxA and PxB as outputs.

FIGURE 19-6: EXAMPLE OF HALF-BRIDGE PWM OUTPUT

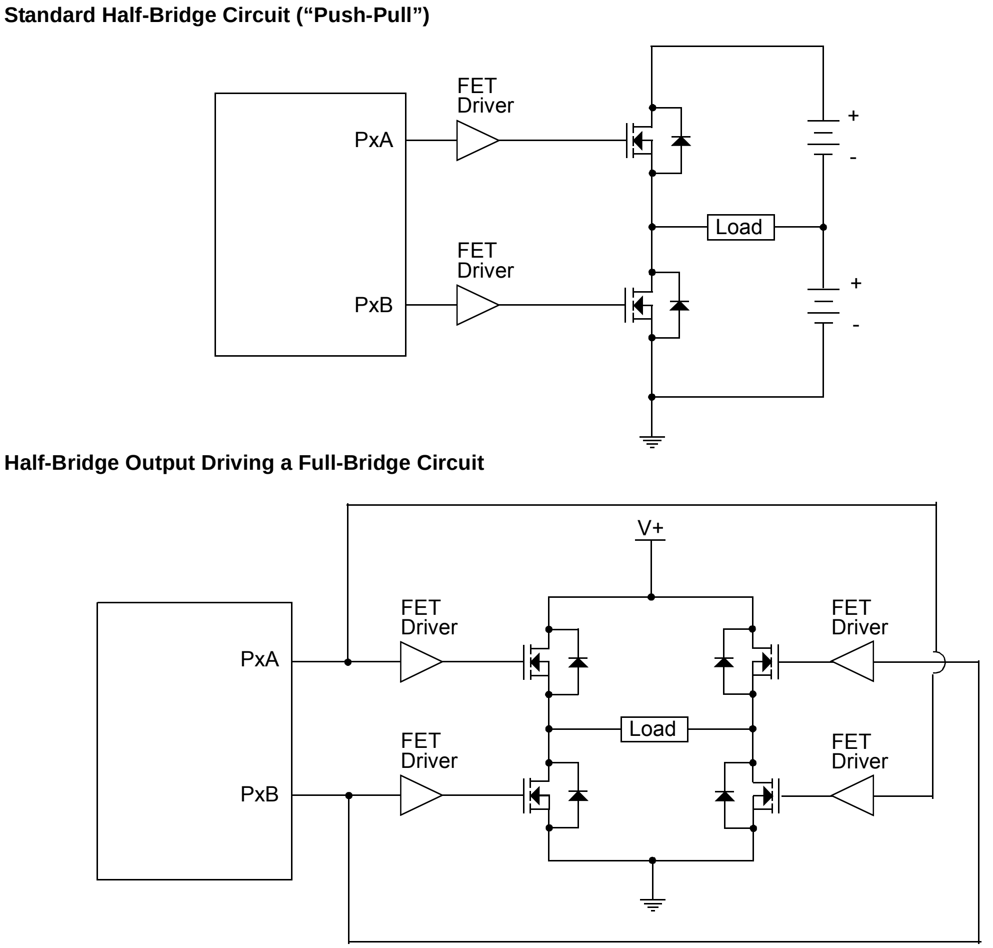


td = Dead-Band Delay

Note 1: At this time, the TMR2 register is equal to the PR2 register.

2: Output signals are shown as active-high.

FIGURE 19-7: EXAMPLE OF HALF-BRIDGE APPLICATIONS


Standard Half-Bridge Circuit ("Push-Pull")
PxA
PxB
FET Driver
FET Driver
+
-
Load
+
-
Half-Bridge Output Driving a Full-Bridge Circuit
V+
PxA
PxB
FET Driver
FET Driver
FET Driver
FET Driver
Load

The LCDSE5:LCDSE0 registers configure the functions of the port pins. Setting the segment enable bit for a particular segment configures that pin as an LCD driver. There are six LCD Segment Enable registers, as shown in Table 20-1. The prototype LCDSEx register is shown in Register 20-5.

TABLE 20-1: LCDSE REGISTERS AND ASSOCIATED SEGMENTS

Register	Segments
LCDSE0	7:0 (RD<7:0>)
LCDSE1	15:8 (RA<5:4>, RC2, RC5, RB<4:1>)
LCDSE2	23:16 (RF<5:1>, RA1, RC<4:3>)
LCDSE3	31:24 (RE7, RB0, RB5, RC<7:6>, RG4, RF<7:6>)
LCDSE4	39:32 (RJ<4:7>, RJ<3:1>, RC1)
LCDSE5	47:40 (RH<0:3>, RH<7:4>)

> **Note:** The LCDSE5:LCDSE4 registers are not implemented in PIC18F6XK90 devices.

Once the module is initialized for the LCD panel, the individual bits of the LCDDATA23:LCDDATA0 registers are cleared or set to represent a clear or dark pixel, respectively.

Specific sets of LCDDATA registers are used with specific segments and common signals. Each bit represents a unique combination of a specific segment connected to a specific common.

Individual LCDDATA bits are named by the convention, “SxxCy”, with “xx” as the segment number and “y” as the common number. The relationship is summarized in Table 20-2. The prototype LCDDATAx register is shown in Register 20-6.

> **Note:** In PIC18F6XK90 devices, writing into the registers, LCDDATA4, LCDDATA5, LCDDATA10, LCDDATA11, LCDDATA16, LCDDATA17, LCDDATA22 and LCDDATA23, will not affect the status of any pixel. These registers can be used as general purpose registers.

REGISTER 20-5: LCDSEx: LCD SEGMENTx ENABLE REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SE(n + 7)	SE(n + 6)	SE(n + 5)	SE(n + 4)	SE(n + 3)	SE(n + 2)	SE(n + 1)	SE(n)
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 7-0 **SE(n + 7):SE(n):** Segment Enable bits

For LCDSE0: n = 0

For LCDSE1: n = 8

For LCDSE2: n = 16

For LCDSE3: n = 24

For LCDSE4: n = 32

For LCDSE5: n = 40

`1` = Segment function of the pin is enabled, digital I/O is disabled
`0` = I/O function of the pin is enabled

FIGURE 20-10: TYPE-A WAVEFORMS IN 1/2 MUX, 1/3 BIAS DRIVE

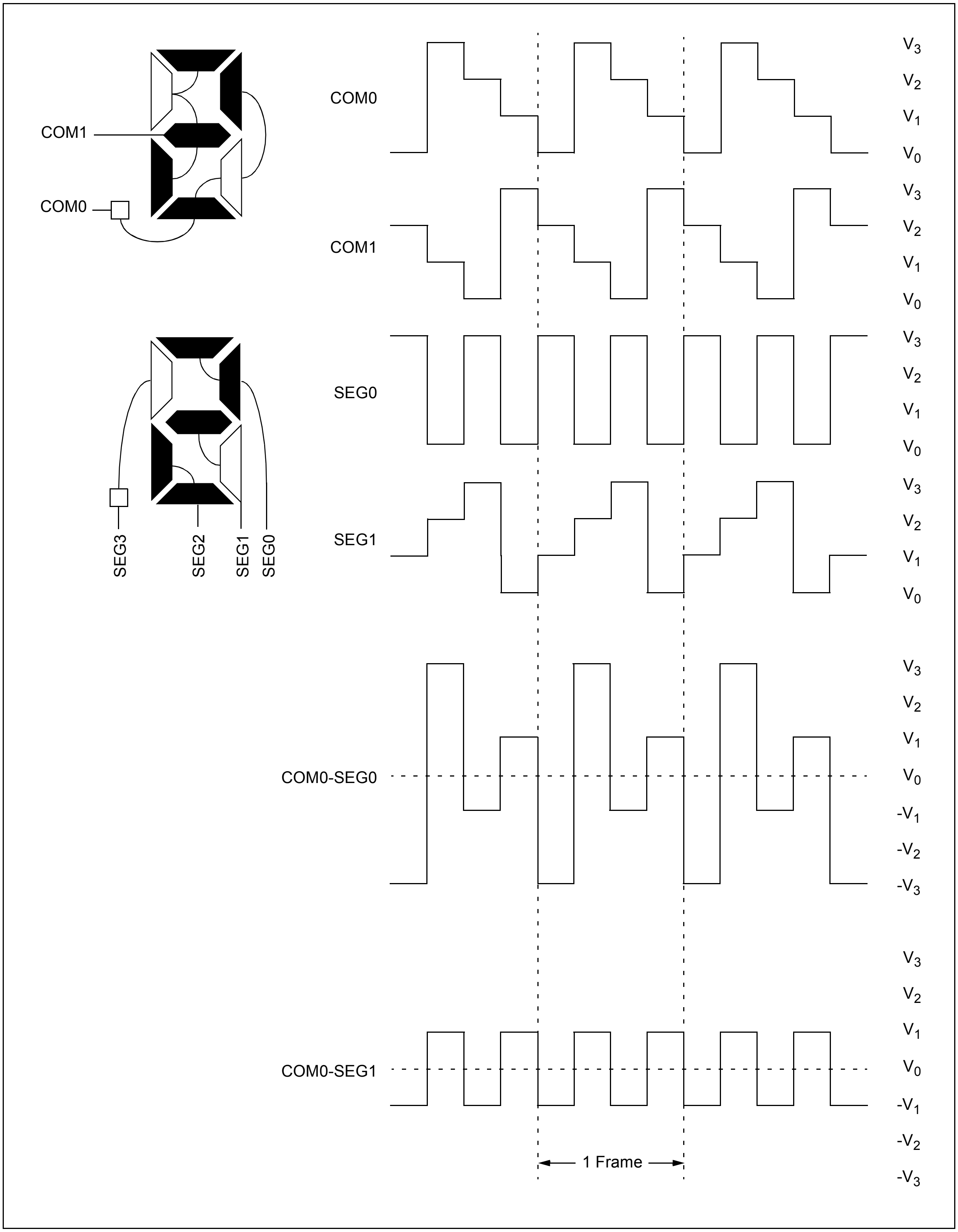


TABLE 20-7: REGISTERS ASSOCIATED WITH LCD OPERATION

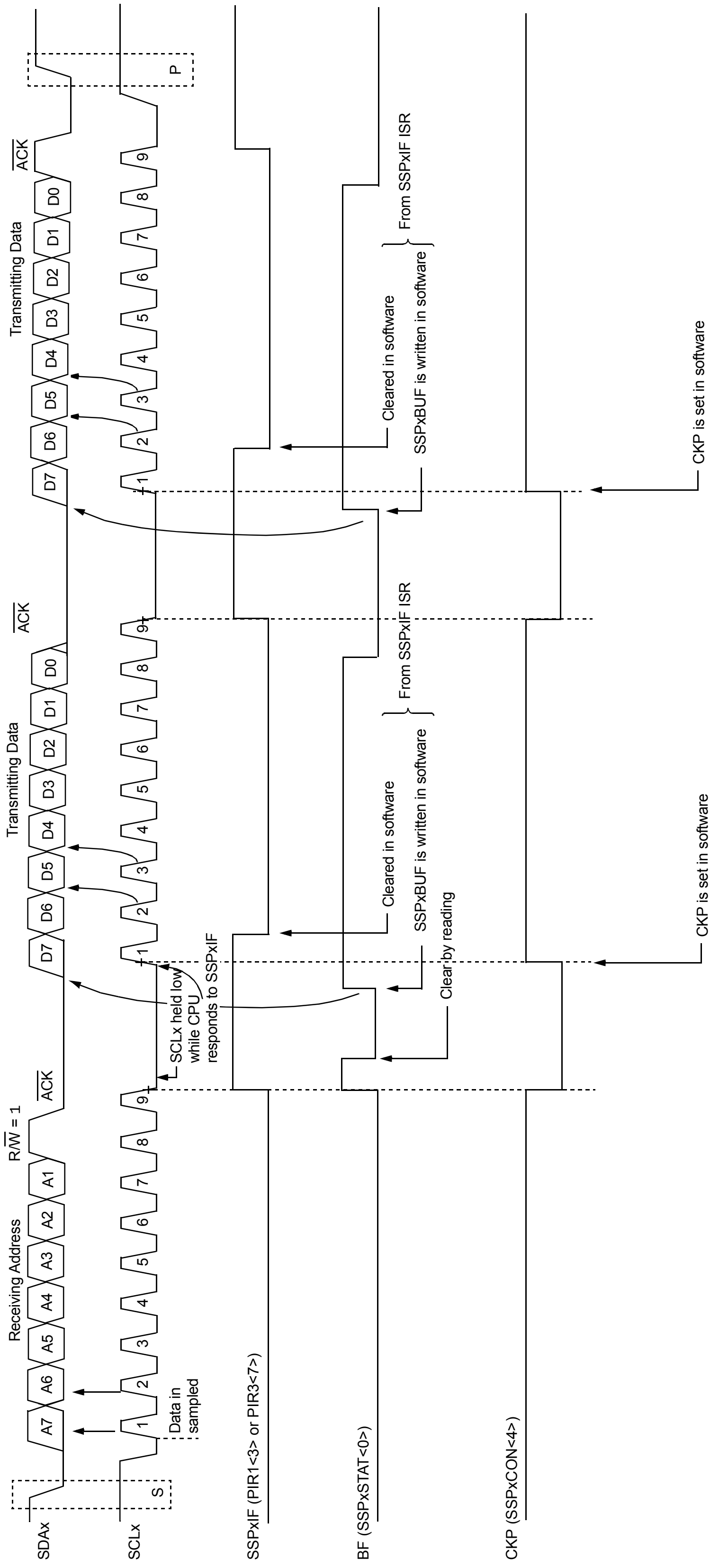
Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	75
PIR3	TMR5GIF	LCDIF	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	RTCCIF	77
PIE3	TMR5GIE	LCDIE	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	RTCCIE	77
IPR3	TMR5GIP	LCDIP	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	RTCCIP	77
RCON	IPEN	SBOREN	$\overline{\text{CM}}$	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	76
LCDDATA23(1)	S47C3	S46C3	S45C3	S44C3	S43C3	S42C3	S41C3	S40C3	79
LCDDATA22(1)	S39C3	S38C3	S37C3	S36C3	S35C3	S34C3	S33C3	S32C3	79
LCDDATA21	S31C3	S30C3	S29C3	S28C3	S27C3	S26C3	S25C3	S24C3	79
LCDDATA20	S23C3	S22C3	S21C3	S20C3	S19C3	S18C3	S17C3	S16C3	79
LCDDATA19	S15C3	S14C3	S13C3	S12C3	S11C3	S10C3	S09C3	S08C3	79
LCDDATA18	S07C3	S06C3	S05C3	S04C3	S03C3	S02C3	S01C3	S00C3	79
LCDDATA17(1)	S47C2	S46C2	S45C2	S44C2	S43C2	S42C2	S41C2	S40C2	79
LCDDATA16(1)	S39C2	S38C2	S37C2	S36C2	S35C2	S34C2	S33C2	S32C2	79
LCDDATA15	S31C2	S30C2	S29C2	S28C2	S27C2	S26C2	S25C2	S24C2	79
LCDDATA14	S23C2	S22C2	S21C2	S20C2	S19C2	S18C2	S17C2	S16C2	79
LCDDATA13	S15C2	S14C2	S13C2	S12C2	S11C2	S10C2	S09C2	S08C2	79
LCDDATA12	S07C2	S06C2	S05C2	S04C2	S03C2	S02C2	S01C2	S00C2	79
LCDDATA11(1)	S47C1	S46C1	S45C1	S44C1	S43C1	S42C1	S41C1	S40C1	79
LCDDATA10(1)	S39C1	S38C1	S37C1	S36C1	S35C1	S34C1	S33C1	S32C1	79
LCDDATA9	S31C1	S30C1	S29C1	S28C1	S27C1	S26C1	S25C1	S24C1	79
LCDDATA8	S23C1	S22C1	S21C1	S20C1	S19C1	S18C1	S17C1	S16C1	79
LCDDATA7	S15C1	S14C1	S13C1	S12C1	S11C1	S10C1	S09C1	S08C1	79
LCDDATA6	S07C1	S06C1	S05C1	S04C1	S03C1	S02C1	S01C1	S00C1	79
LCDDATA5(1)	S47C0	S46C0	S45C0	S44C0	S43C0	S42C0	S41C0	S40C0	79
LCDDATA4(1)	S39C0	S38C0	S37C0	S36C0	S35C0	S34C0	S33C0	S32C0	79
LCDDATA3	S31C0	S30C0	S29C0	S28C0	S27C0	S26C0	S25C0	S24C0	79
LCDDATA2	S23C0	S22C0	S21C0	S20C0	S19C0	S18C0	S17C0	S16C0	79
LCDDATA1	S15C0	S14C0	S13C0	S12C0	S11C0	S10C0	S09C0	S08C0	79
LCDDATA0	S07C0	S06C0	S05C0	S04C0	S03C0	S02C0	S01C0	S00C0	79
LCDSE5(2)	SE47	SE46	SE45	SE44	SE43	SE42	SE41	SE40	83
LCDSE4(2)	SE39	SE38	SE37	SE36	SE35	SE34	SE33	SE32	83
LCDSE3	SE31	SE30	SE29	SE28	SE27	SE26	SE25	SE24	83
LCDSE2	SE23	SE22	SE21	SE20	SE19	SE18	SE17	SE16	83
LCDSE1	SE15	SE14	SE13	SE12	SE11	SE10	SE9	SE8	83
LCDSE0	SE7	SE6	SE5	SE4	SE3	SE2	SE1	SE0	83
LCDCON	LCDEN	SLPEN	WERR	—	CS1	CS0	LMUX1	LMUX0	83
LCDPS	WFT	BIASMD	LCDA	WA	LP3	LP2	LP1	LP0	83
LCDREF	LCDIRE	LCDIRS	LCDCST2	LCDCST1	LCDCST0	VLCD3PE	VLCD2PE	VLCD1PE	83
LCDRL	LRLAP1	LRLAP0	LRLBP1	LRLBP0	—	LRLAT2	LRLAT1	LRLAT0	83

Legend: — = unimplemented, read as '0'. Shaded cells are not used for LCD operations.

Note 1: These registers are implemented, but unused on 64-pin devices, and may be used as general purpose data RAM.

2: These registers are unimplemented in 64-pin devices.

FIGURE 21-10: I^{2}C™ SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)



21.4.17.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

a) SDAx or SCLx is sampled low at the beginning of the Start condition (Figure 21-28).

b) SCLx is sampled low before SDAx is asserted low (Figure 21-29).

During a Start condition, both the SDAx and the SCLx pins are monitored.

If the SDAx pin is already low, or the SCLx pin is already low, then all of the following occur:

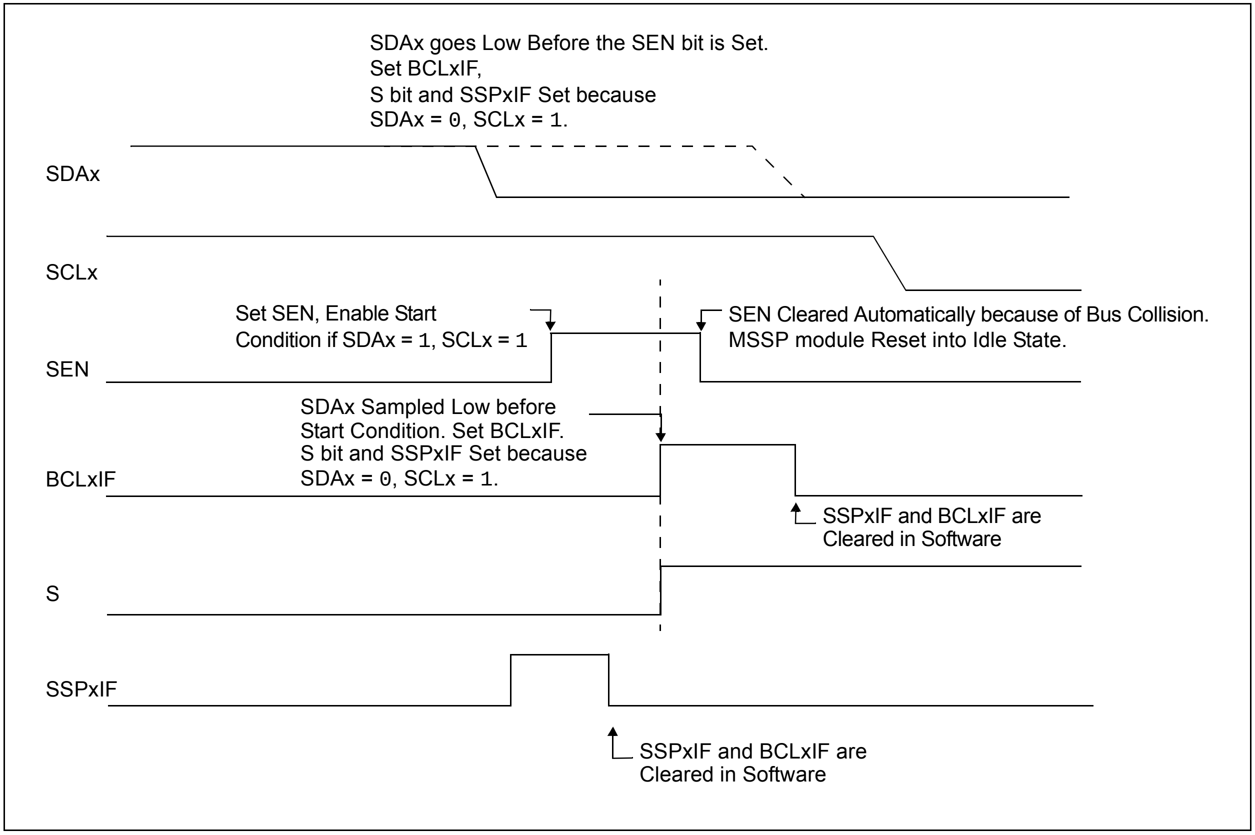
- The Start condition is aborted
- The BCLxIF flag is set
- The MSSP module is reset to its inactive state (see Figure 21-28)

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

If the SDAx pin is sampled low during this count, the BRG is reset and the SDAx line is asserted early (Figure 21-30). If, however, a ‘1’ is sampled on the SDAx pin, the SDAx pin is asserted low at the end of the BRG count. The Baud Rate Generator is then reloaded and counts down to 0. If the SCLx pin is sampled as ‘0’ during this time, a bus collision does not occur. At the end of the BRG count, the SCLx pin is asserted low.

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

FIGURE 21-28: BUS COLLISION DURING START CONDITION (SDAx ONLY)


SDAx goes Low Before the SEN bit is Set.
Set BCLxIF,
S bit and SSPxIF Set because
SDAx = 0, SCLx = 1.
SDAx
SCLx
Set SEN, Enable Start
Condition if SDAx = 1, SCLx = 1
SEN Cleared Automatically because of Bus Collision.
MSSP module Reset into Idle State.
SEN
SDAx Sampled Low before
Start Condition. Set BCLxIF.
S bit and SSPxIF Set because
SDAx = 0, SCLx = 1.
BCLxIF
SSPxIF and BCLxIF are
Cleared in Software
S
SSPxIF
SSPxIF and BCLxIF are
Cleared in Software

REGISTER 23-4: ADRESH: A/D RESULT HIGH BYTE REGISTER, LEFT JUSTIFIED (ADFM = 0)

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
ADRES11	ADRES10	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4
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 7-0 **ADRES<11:4>:** A/D Result High Byte bits

REGISTER 23-5: ADRESL: A/D RESULT LOW BYTE REGISTER, LEFT JUSTIFIED (ADFM = 0)

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
ADRES3	ADRES2	ADRES1	ADRES0	ADSGN	ADSGN	ADSGN	ADSGN
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 7-4 **ADRES<3:0>:** A/D Result Low Byte bits

bit 3-0 **ADSGN:** A/D Result Sign bits
1 = A/D result is negative
0 = A/D result is positive

REGISTER 28-12: CONFIG7L: CONFIGURATION REGISTER 7 LOW (BYTE ADDRESS 30000Ch)(3)

R/C-1	R/C-1	R/C-1	R/C-1	R/C-1	R/C-1	R/C-1	R/C-1
EBTR7(1)	EBTR6(1)	EBTR5(1)	EBTR4(1)	EBTR3	EBTR2	EBTR1	EBTR0
bit 7							bit 0

Legend:	C = Clearable bit		
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 **EBTR7:** Table Read Protection bit(1)
1 = Block 7 is not protected from table reads executed in other blocks
0 = Block 7 is protected from table reads executed in other blocks

bit 6 **EBTR6:** Table Read Protection bit(1)
1 = Block 6 is not protected from table reads executed in other blocks
0 = Block 6 is protected from table reads executed in other blocks

bit 5 **EBTR5:** Table Read Protection bit(1)
1 = Block 5 is not protected from table reads executed in other blocks
0 = Block 5 is protected from table reads executed in other blocks

bit 4 **EBTR4:** Table Read Protection bit(1)
1 = Block 4 is not protected from table reads executed in other blocks
0 = Block 4 is protected from table reads executed in other blocks

bit 3 **EBTR3:** Table Read Protection bit
1 = Block 3 is not protected from table reads executed in other blocks
0 = Block 3 is protected from table reads executed in other blocks

bit 2 **EBTR2**: Table Read Protection bit
1 = Block 2 is not protected from table reads executed in other blocks
0 = Block 2 is protected from table reads executed in other blocks

bit 1 **EBTR1:** Table Read Protection bit
1 = Block 1 is not protected from table reads executed in other blocks
0 = Block 1 is protected from table reads executed in other blocks

bit 0 **EBTR0:** Table Read Protection bit
1 = Block 0 is not protected from table reads executed in other blocks
0 = Block 0 is protected from table reads executed in other blocks

Note 1: This bit is only available on PIC18F67K90 and PIC18F87K90.
2: This bit is only available on PIC18F66K90, PIC18F67K90, PIC18F86K90 and PIC18F87K90 devices.
3: For the memory size of the blocks, refer to Figure 28-6.

BZ Branch if Zero

Syntax: BZ n

Operands: $-128 \le n \le 127$

Operation: if Zero bit is ‘1’,
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0000	nnnn	nnnn

Description: If the Zero bit is ‘1’, then the program will branch.

The 2’s complement number ‘2n’ is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal ‘n’	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal ‘n’	Process Data	No operation

Example:
```
        HERE      BZ   Jump

Before Instruction
    PC          =   address (HERE)
After Instruction
    If Zero     =   1;
        PC      =   address (Jump)
    If Zero     =   0;
        PC      =   address (HERE + 2)
```

CALL Subroutine Call

Syntax: CALL k {,s}

Operands: $0 \le k \le 1048575$
$s \in [0,1]$

Operation: (PC) + 4 → TOS,
k → PC<20:1>;
if s = 1,
(W) → WS,
(STATUS) → STATUSS,
(BSR) → BSRS

Status Affected: None

Encoding:

1st word (k<7:0>)	1110	110s	$k_7$kkk	kkkk$_0$
2nd word(k<19:8>)	1111	k_{19}kkk	kkkk	kkkk$_8$

Description: Subroutine call of entire 2-Mbyte memory range. First, return address (PC+ 4) is pushed onto the return stack. If ‘s’ = 1, the W, STATUS and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If ‘s’ = 0, no update occurs. Then, the 20-bit value ‘k’ is loaded into PC<20:1>. CALL is a two-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal ‘k’<7:0>,	Push PC to stack	Read literal ’k’<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example:
```
        HERE      CALL   THERE,1

Before Instruction
    PC        =   address (HERE)
After Instruction
    PC        =   address (THERE)
    TOS       =   address (HERE + 4)
    WS        =   W
    BSRS      =   BSR
    STATUSS   =   STATUS
```

PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PART NO.	X	/XX	XXX
Device	**Temperature Range**	**Package**	**Pattern**

Device[1,2]	PIC18F65K90, PIC18F65K90T PIC18F66K90, PIC18F66K90T PIC18F67K90, PIC18F67K90T PIC18F85K90, PIC18F85K90T PIC18F86K90, PIC18F86K90T PIC18F87K90, PIC18F87K90T

Temperature Range	I = -40°C to +85°C (Industrial)
	E = -40°C to +125°C (Extended)

Package	PT = TQFP (Plastic Thin Quad Flatpack)
	MR = QFN (Plastic Quad Flat)

Pattern	QTP, SQTP, Code or Special Requirements (blank otherwise)

Examples:

a) PIC18F87K90-I/PT 301 = Industrial temperature, TQFP package, QTP pattern #301.

b) PIC18F87K90T-I/PT = Tape and reel, Industrial temperature, TQFP package.

c) PIC18F87K22T-E/PT = Tape and reel, Extended temperature, TQFP package.

Note 1: F = Standard Voltage Range
2: T = In tape and reel
3: RSL = Silicon revision A3