



Welcome to [E-XFL.COM](#)

What is "[Embedded - Microcontrollers](#)"?

"[Embedded - Microcontrollers](#)" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "[Embedded - Microcontrollers](#)"

Details

| | |
|----------------------------|---|
| Product Status | Active |
| Core Processor | PIC |
| Core Size | 8-Bit |
| Speed | 64MHz |
| Connectivity | I ² C, LINbus, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, LVD, POR, PWM, WDT |
| Number of I/O | 53 |
| Program Memory Size | 64KB (32K 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/pic18f66k22-i-ptsl |

1.0 DEVICE OVERVIEW

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

- PIC18F65K22
- PIC18F66K22
- PIC18F67K22
- PIC18F85K22
- PIC18F86K22
- PIC18F87K22

This family combines the traditional advantages of all PIC18 microcontrollers – namely, high computational performance and a rich feature set – with an extremely competitive price point. These features make the PIC18F87K22 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 PIC18F87K22 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 Sleep, BOR, RTCC and Watchdog Timer

1.1.2 OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F87K22 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 ($\pm 2\%$ accuracy) and an INTRC source (approximately 31 kHz, stable over temperature and VDD)
 - 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 PIC18F87K22 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 PIC18F87K22 family also provides plenty of room for dynamic application data with up to 3,862 bytes of data RAM.

1.1.4 EXTERNAL MEMORY BUS

Should 128 Kbytes of memory be inadequate for an application, the 80-pin members of the PIC18F87K22 family have an External Memory Bus (EMB) enabling the controller's internal Program Counter to address a memory space of up to 2 Mbytes. This is a level of data access that few 8-bit devices can claim and enables:

- Using combinations of on-chip and external memory of up to 2 Mbytes
- Using external Flash memory for reprogrammable application code or large data tables
- Using external RAM devices for storing large amounts of variable data

1.1.5 EXTENDED INSTRUCTION SET

The PIC18F87K22 family implements the optional extension to the PIC18 instruction set, adding eight 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'.

The use of the flag and control bits in the OSCCON register is discussed in more detail in **Section 4.0 “Power-Managed Modes”**.

Note 1: The Timer1/3/5/7 oscillator must be enabled to select the secondary clock source. The Timerx oscillator is enabled by setting the SOSCEN bit in the Timerx Control register (TxCON<3>). If the Timerx oscillator is not enabled, then any attempt to select a secondary clock source when executing a SLEEP instruction will be ignored.

2: It is recommended that the Timerx oscillator be operating and stable before executing the SLEEP instruction or a very long delay may occur while the Timerx oscillator starts.

3.3.2.1 System Clock Selection and Device Resets

Since the SCS bits are cleared on all forms of Reset, this means the primary oscillator, defined by the FOSC<3:0> Configuration bits, is used as the primary clock source on device Resets. This could either be the internal oscillator block by itself, or one of the other primary clock source (HS, EC, XT, LP, External RC and PLL-Enabled modes).

In those cases when the internal oscillator block, without PLL, is the default clock on Reset, the Fast RC oscillator (INTOSC) will be used as the device clock source. It will initially start at 8 MHz; the postscaler selection that corresponds to the Reset value of the IRCF<2:0> bits ('110').

Regardless of which primary oscillator is selected, INTRC will always be enabled on device power-up. It serves as the clock source until the device has loaded its configuration values from memory. It is at this point that the FOSC Configuration bits are read and the oscillator selection of the operational mode is made.

Note that either the primary clock source or the internal oscillator will have two bit setting options for the possible values of the SCS<1:0> bits, at any given time.

3.3.3 OSCILLATOR TRANSITIONS

PIC18F87K22 family devices contain circuitry to prevent clock “glitches” when switching between clock sources. A short pause in the device clock occurs during the clock switch. The length of this pause is the sum of two cycles of the old clock source and three to four cycles of the new clock source. This formula assumes that the new clock source is stable.

Clock transitions are discussed in greater detail in **Section 4.1.2 “Entering Power-Managed Modes”**.

3.4 RC Oscillator

For timing-insensitive applications, the RC and RCIO Oscillator modes offer additional cost savings. The actual oscillator frequency is a function of several factors:

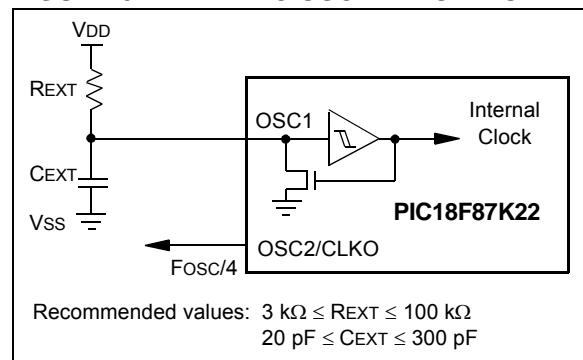
- Supply Voltage
- Values of the External Resistor (R_{EXT}) and Capacitor (C_{EXT})
- Operating Temperature

Given the same device, operating voltage and temperature, and component values, there will also be unit to unit frequency variations. These are due to factors, such as:

- Normal manufacturing variation
- Difference in lead frame capacitance between package types (especially for low C_{EXT} values)
- Variations within the tolerance of limits of R_{EXT} and C_{EXT}

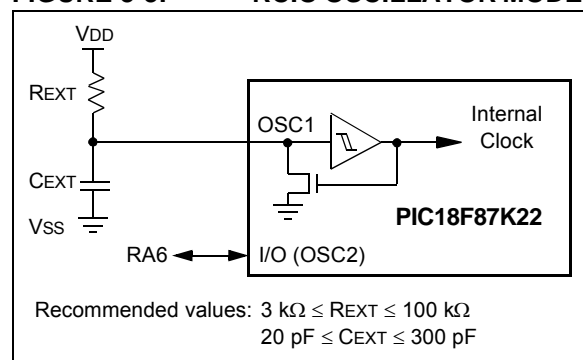
In the RC Oscillator mode, the oscillator frequency, divided by 4, is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 3-2 shows how the R/C combination is connected.

FIGURE 3-2: RC OSCILLATOR MODE



The RCIO Oscillator mode (Figure 3-3) functions like the RC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6).

FIGURE 3-3: RCIO OSCILLATOR MODE



PIC18F87K22 FAMILY

4.7 Ultra Low-Power Wake-up

The Ultra Low-Power Wake-up (ULPWU) on pin, RA0, allows a slow falling voltage to generate an interrupt without excess current consumption.

To use this feature:

1. Charge the capacitor on RA0 by configuring the RA0 pin to an output and setting it to '1'.
2. Stop charging the capacitor by configuring RA0 as an input.
3. Discharge the capacitor by setting the ULPEN and ULPSINK bits in the WDTCON register.
4. Configure Sleep mode.
5. Enter Sleep mode.

When the voltage on RA0 drops below V_{IL} , the device wakes up and executes the next instruction.

This feature provides a low-power technique for periodically waking up the device from Sleep mode.

The time-out is dependent on the discharge time of the RC circuit on RA0.

When the ULPWU module wakes the device from Sleep mode, the ULPLVL bit (WDTCON<5>) is set. Software can check this bit upon wake-up to determine the wake-up source.

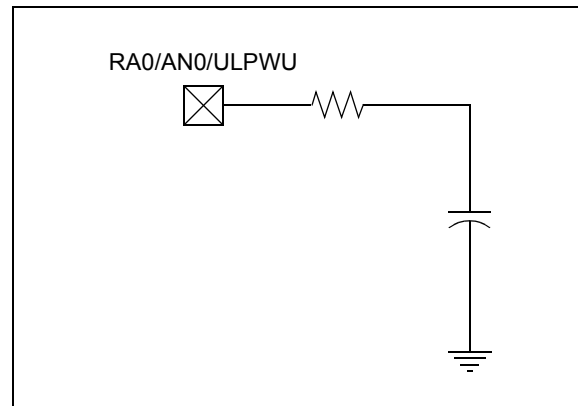
See Example 4-1 for initializing the ULPWU module.

EXAMPLE 4-1: ULTRA LOW-POWER WAKE-UP INITIALIZATION

```
//*****  
//Charge the capacitor on RA0  
//*****  
TRISAbits.TRISA0 = 0;  
PORTAbits.RA0 = 1;  
for(i = 0; i < 10000; i++) Nop();  
//*****  
//Stop Charging the capacitor  
//on RA0  
//*****  
TRISAbits.TRISA0 = 1;  
//*****  
//Enable the Ultra Low Power  
//Wakeup module and allow  
//capacitor discharge  
//*****  
WDTCONbits.ULPEN = 1;  
WDTCONbits.ULPSINK = 1;  
//For Sleep  
OSCCONbits.IDLEN = 0;  
//Enter Sleep Mode  
//  
Sleep();  
  
//for sleep, execution will  
//resume here
```

A series resistor, between RA0 and the external capacitor, provides overcurrent protection for the RA0/AN0/ULPWU pin and enables software calibration of the time-out (see Figure 4-9).

FIGURE 4-9: ULTRA LOW-POWER WAKE-UP INITIALIZATION



A timer can be used to measure the charge time and discharge time of the capacitor. The charge time can then be adjusted to provide the desired delay in Sleep. This technique compensates for the affects of temperature, voltage and component accuracy. The peripheral can also be configured as a simple Programmable Low-Voltage Detect (LVD) or temperature sensor.

Note: For more information, see AN879, "Using the Microchip Ultra Low-Power Wake-up Module" (DS00879).

PIC18F87K22 FAMILY

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.

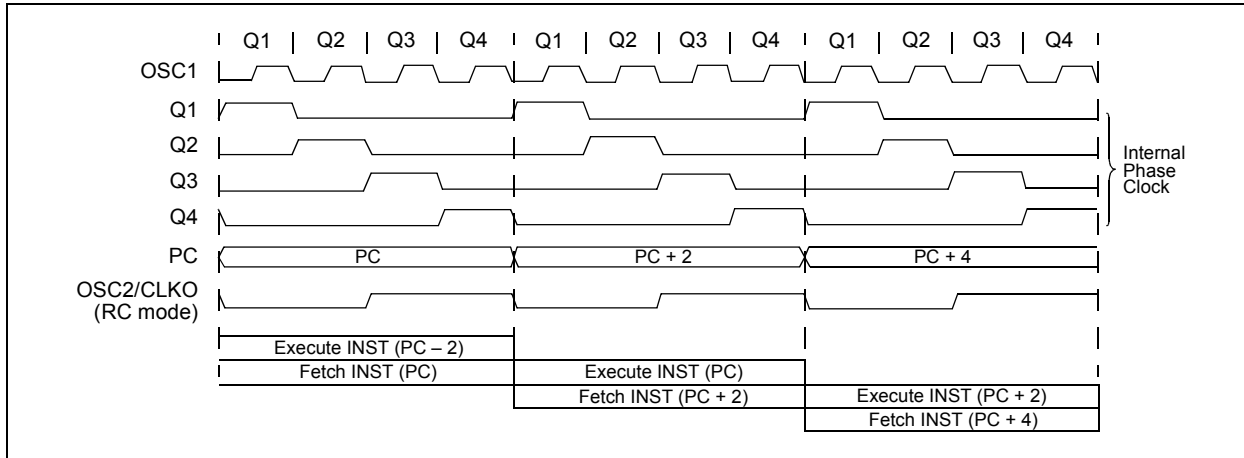
6.2.2 INSTRUCTION FLOW/PIPELINING

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.)

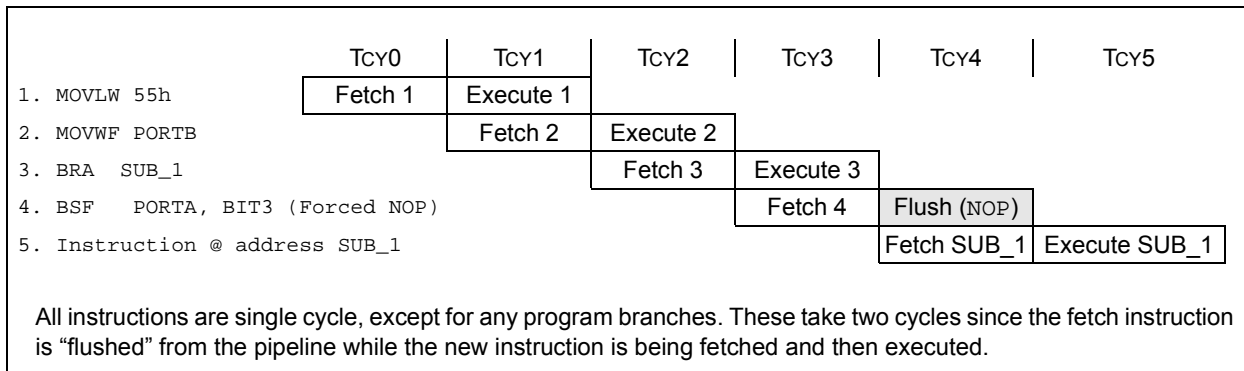
A fetch cycle begins with the Program Counter (PC) incrementing in Q1.

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



7.5 Writing to Flash Program Memory

The programming blocks are:

- PIC18FX5K22 and PIC18FX6K22 – 32 words or 64 bytes
- PIC18FX7K22 – 64 words or 128 bytes

Word or byte programming is not supported.

Table writes are used internally to load the holding registers needed to program the Flash memory. The number of holding registers used for programming by the table writes are:

- PIC18FX5K22 and PIC18FX6K22 – 64
- PIC18FX7K22 – 128

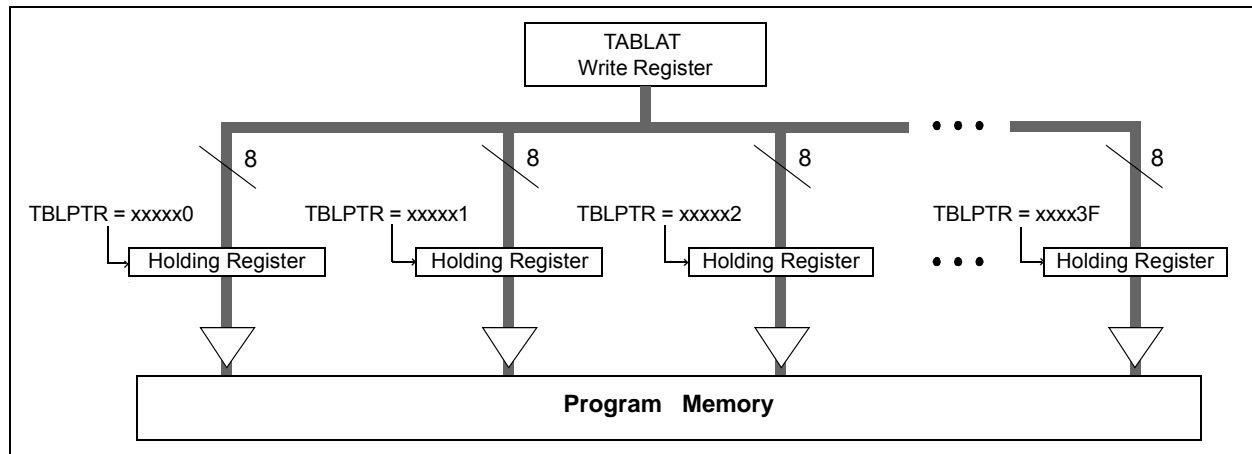
Since the Table Latch (TABLAT) is only a single byte, the TBLWT instruction may need to be executed 64 times for each programming operation. All of the table write operations will essentially be short writes because only the holding registers are written. At the end of updating the 64 or 128 holding registers, the EECON1 register must be written to in order to start the programming operation with a long write.

The long write is necessary for programming the internal Flash. Instruction execution is halted while in a long write cycle. The long write is terminated by the internal programming timer.

The EEPROM on-chip timer controls the write time. The write/erase voltages are generated by an on-chip charge pump, rated to operate over the voltage range of the device.

Note: The default value of the holding registers on device Resets, and after write operations, is FFh. A write of FFh to a holding register does not modify that byte. This means that individual bytes of program memory may be modified, provided that the change does not attempt to change any bit from a '0' to a '1'. When modifying individual bytes, it is not necessary to load all 64 or 128 holding registers before executing a write operation.

FIGURE 7-5: TABLE WRITES TO FLASH PROGRAM MEMORY



PIC18F87K22 FAMILY

REGISTER 11-14: PIE5: PERIPHERAL INTERRUPT ENABLE REGISTER 5

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|------------------------|------------------------|------------------------|--------|-----------------------|--------|--------|--------|
| TMR7GIE ⁽¹⁾ | TMR12IE ⁽¹⁾ | TMR10IE ⁽¹⁾ | TMR8IE | TMR7IE ⁽¹⁾ | TMR6IE | TMR5IE | TMR4IE |
| 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 **TMR7GIE:** TMR7 Gate Interrupt Enable bit⁽¹⁾
 1 = Enabled
 0 = Disabled
- bit 6 **TMR12IE:** TMR12 to PR12 Match Interrupt Enable bit⁽¹⁾
 1 = Enables the TMR12 to PR12 match interrupt
 0 = Disables the TMR12 to PR12 match interrupt
- bit 5 **TMR10IE:** TMR10 to PR10 Match Interrupt Enable bit⁽¹⁾
 1 = Enables the TMR10 to PR10 match interrupt
 0 = Disables the TMR10 to PR10 match interrupt
- bit 4 **TMR8IE:** TMR8 to PR8 Match Interrupt Enable bit
 1 = Enables the TMR8 to PR8 match interrupt
 0 = Disables the TMR8 to PR8 match interrupt
- bit 3 **TMR7IE:** TMR7 Overflow Interrupt Enable bit⁽¹⁾
 1 = Enables the TMR7 overflow interrupt
 0 = Disables the TMR7 overflow interrupt
- bit 2 **TMR6IE:** TMR6 to PR6 Match Interrupt Enable bit
 1 = Enables the TMR6 to PR6 match interrupt
 0 = Disables the TMR6 to PR6 match interrupt
- bit 1 **TMR5IE:** TMR5 Overflow Interrupt Enable bit
 1 = Enables the TMR5 overflow interrupt
 0 = Disables the TMR5 overflow interrupt
- bit 0 **TMR4IE:** TMR4 to PR4 Match Interrupt Enable bit
 1 = Enables the TMR4 to PR4 match interrupt
 0 = Disables the TMR4 to PR4 match interrupt

Note 1: Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).

PIC18F87K22 FAMILY

12.5 PORTD, TRISD and LATD Registers

PORTD is an 8-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISD and LATD.

All pins on PORTD are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

Note: These pins are configured as digital inputs on any device Reset.

Each of the PORTD pins has a weak internal pull-up. A single control bit can turn off all the pull-ups. This is performed by setting bit, RDPU (PADCFG1<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on all device Resets.

On 80-pin devices, PORTD is multiplexed with the system bus as part of the external memory interface. The I/O port and other functions are only available when the interface is disabled by setting the EBDIS bit (MEMCON<7>). When the interface is enabled, PORTD is the low-order byte of the multiplexed address/data bus (AD<7:0>). The TRISD bits are also overridden.

PORTD can also be configured as an 8-bit wide microprocessor port (Parallel Slave Port) by setting control bit, PSPMODE (TRISE<4>). In this mode, the input buffers are TTL. For additional information, see **Section 12.11 “Parallel Slave Port”**.

The PORTD also has the I²C and SPI functionality on RD4, RD5 and RD6. The pins for SPI are also configurable for open-drain output. Open-drain configuration is selected by setting bit, SSP2OD (ODCON1<0>).

RD0 has a CTMU functionality. RD1 has the functionality for the Timer5 clock input and Timer7 external clock gate input.

EXAMPLE 12-4: INITIALIZING PORTD

```
CLRF    PORTD    ; Initialize PORTD by
                  ; clearing output
                  ; data latches
CLRF    LATD      ; Alternate method
                  ; to clear output
                  ; data latches
MOVLW   0CFh     ; Value used to
                  ; initialize data
                  ; direction
MOVWF   TRISD    ; Set RD<3:0> as inputs
                  ; RD<5:4> as outputs
                  ; RD<7:6> as inputs
```

TABLE 12-7: PORTD FUNCTIONS

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|--------------------------------|---------------------|--------------|-----|----------|---------------------------------|
| RD0/PSP0/ AD0/CTPLS | RD0 | 0 | O | DIG | LATD<0> data output. |
| | | 1 | I | ST | PORTD<0> data input. |
| | PSP0 ⁽¹⁾ | x | I/O | TTL | Parallel Slave Port data. |
| | AD0 ⁽²⁾ | x | I/O | TTL | External Memory Address/Data 0. |
| | CTPLS | x | O | DIG | CTMU pulse generator output. |
| RD1/PSP1/ AD1/T5CKI/ T7G | RD1 | 0 | O | DIG | LATD<1> data output. |
| | | 1 | I | ST | PORTD<1> data input. |
| | PSP1 ⁽¹⁾ | x | I/O | TTL | Parallel Slave Port data. |
| | AD1 ⁽²⁾ | x | I/O | TTL | External Memory Address/Data 1. |
| | T5CKI | x | I | ST | Timer5 clock input. |
| RD2/PSP2/AD2 | RD2 | 0 | O | DIG | LATD<2> data output. |
| | | 1 | I | ST | PORTD<2> data input. |
| | PSP2 ⁽¹⁾ | x | I/O | TTL | Parallel Slave Port data. |
| | AD2 ⁽²⁾ | x | I/O | TTL | External Memory Address/Data 2. |

Legend: O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger Buffer Input, I²C = I²C™/SMBus Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

Note 1: The Parallel Slave Port (PSP) is available only in Microcontroller mode.

Note 2: This feature is available only on PIC18F8XK22 devices.

12.11 Parallel Slave Port

PORTD can function as an 8-bit-wide Parallel Slave Port (PSP), or microprocessor port, when control bit, PSPMODE (PSPCON<4>), is set. The port is asynchronously readable and writable by the external world through the \overline{RD} control input pin (RE0/P2D/ \overline{RD} /AD8) and \overline{WR} control input pin (RE1/P2C/ \overline{WR} /AD9).

Note: The Parallel Slave Port is available only in Microcontroller mode.

The PSP can directly interface to an 8-bit microprocessor data bus. The external microprocessor can read or write the PORTD latch as an eight-bit latch.

Setting bit, PSPMODE, enables port pin, RE0/P2D/ \overline{RD} /AD8, to be the \overline{RD} input, RE1/P2C/ \overline{WR} /AD9 to be the \overline{WR} input and RE2/P2B/CCP10/ \overline{CS} /AD10 to be the \overline{CS} (Chip Select) input. For this functionality, the corresponding data direction bits of the TRISE register (TRISE<2:0>) must be configured as inputs (= 111).

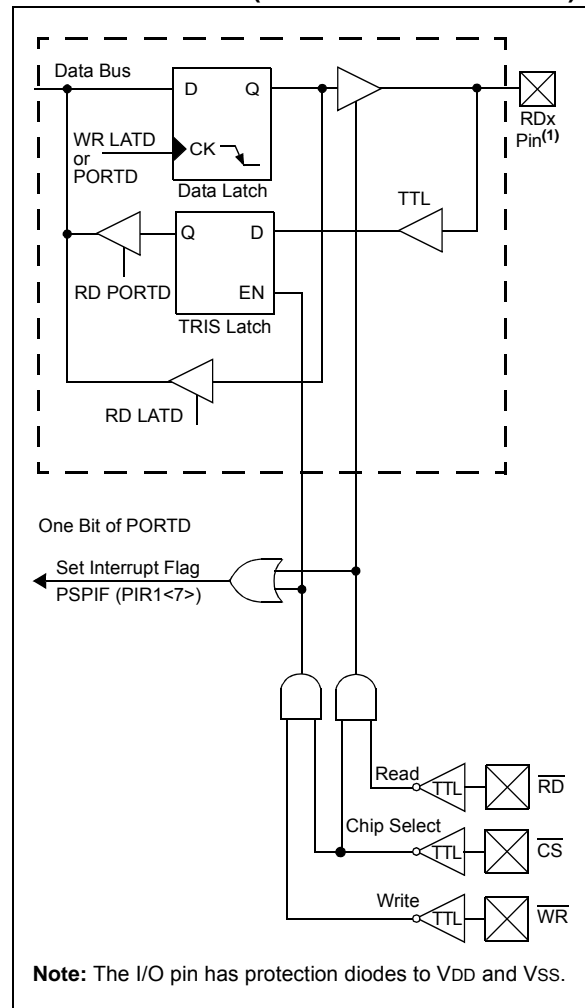
A write to the PSP occurs when both the \overline{CS} and \overline{WR} lines are first detected low and ends when either are detected high. The PSPIF and IBF flag bits (PIR1<7> and PSPCON<7>, respectively) are set when the write ends.

A read from the PSP occurs when both the \overline{CS} and \overline{RD} lines are first detected low. The data in PORTD is read out and the OBF bit (PSPCON<6>) is set. If the user writes new data to PORTD to set OBF, the data is immediately read out, but the OBF bit is not set.

When either the \overline{CS} or \overline{RD} line is detected high, the PORTD pins return to the input state and the PSPIF bit is set. User applications should wait for PSPIF to be set before servicing the PSP. When this happens, the IBF and OBF bits can be polled and the appropriate action taken.

The timing for the control signals in Write and Read modes is shown in Figure 12-4 and Figure 12-5, respectively.

FIGURE 12-3: PORTD AND PORTE BLOCK DIAGRAM (PARALLEL SLAVE PORT)



PIC18F87K22 FAMILY

REGISTER 19-3: CCPTMRS2: CCP 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-Byte 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-Byte Device Variants (PIC18F65K22 and PIC18F85K22):

00 = CCP8 is based off of TMR1/TMR2

01 = CCP8 is based off of TMR1/TMR4

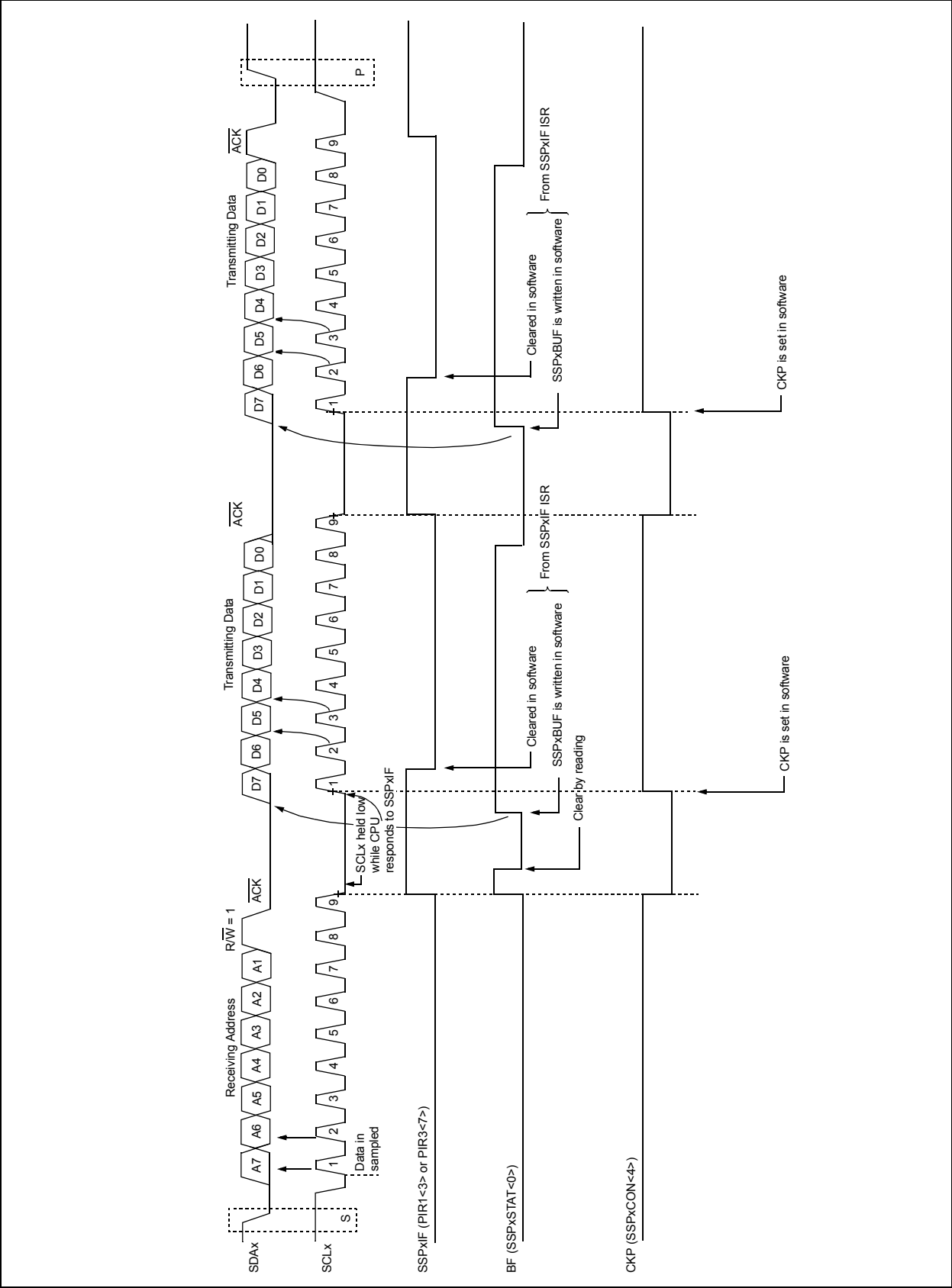
10 = CCP8 is based off of TMR1/TMR6

11 = Reserved; do not use

Note 1: This bit is unimplemented and reads as '0' on devices with 32 Kbytes of program memory (PIC18FX5K22).

PIC18F87K22 FAMILY

FIGURE 21-10: I²C™ SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)



21.4.7 BAUD RATE

In I²C Master mode, the Baud Rate Generator (BRG) reload value is placed in the lower 7 bits of the SSPxADD register (Figure 21-19). When a write occurs to SSPxBUF, the Baud Rate Generator will automatically begin counting. The BRG counts down to 0 and stops until another reload has taken place. The BRG count is decremented twice per instruction cycle (Tcy) on the Q2 and Q4 clocks. In I²C Master mode, the BRG is reloaded automatically.

Once the given operation is complete (i.e., transmission of the last data bit is followed by ACK), the internal clock will automatically stop counting and the SCLx pin will remain in its last state.

Table 21-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPxADD. The SSPxADD BRG value of 0x00 is not supported.

21.4.7.1 Baud Rate and Module Interdependence

Because MSSP1 and MSSP2 are independent, they can operate simultaneously in I²C Master mode at different baud rates. This is done by using different BRG reload values for each module.

Because this mode derives its basic clock source from the system clock, any changes to the clock will affect both modules in the same proportion. It may be possible to change one or both baud rates back to a previous value by changing the BRG reload value.

FIGURE 21-19: BAUD RATE GENERATOR BLOCK DIAGRAM

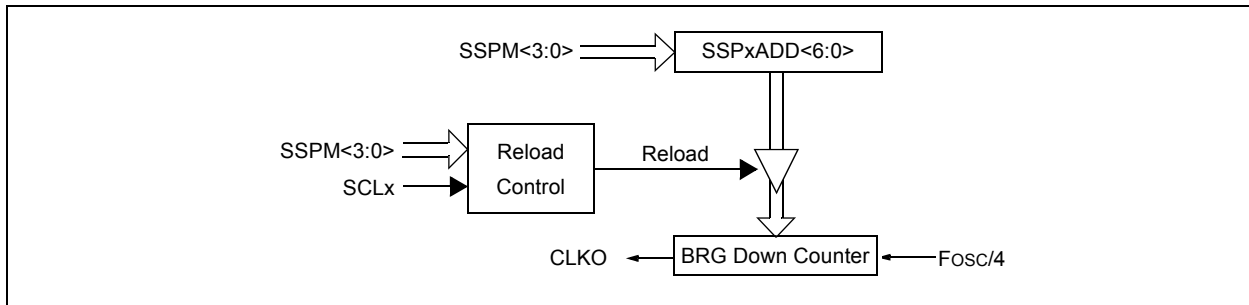


TABLE 21-3: I²C™ CLOCK RATE w/BRG

| Fosc | Fcy | Fcy * 2 | BRG Value | Fscl (2 Rollovers of BRG) |
|--------|--------|---------|-----------|------------------------------|
| 40 MHz | 10 MHz | 20 MHz | 18h | 400 kHz |
| 40 MHz | 10 MHz | 20 MHz | 1Fh | 312.5 kHz |
| 40 MHz | 10 MHz | 20 MHz | 63h | 100 kHz |
| 16 MHz | 4 MHz | 8 MHz | 09h | 400 kHz |
| 16 MHz | 4 MHz | 8 MHz | 0Ch | 308 kHz |
| 16 MHz | 4 MHz | 8 MHz | 27h | 100 kHz |
| 4 MHz | 1 MHz | 2 MHz | 02h | 333 kHz |
| 4 MHz | 1 MHz | 2 MHz | 09h | 100 kHz |
| 16 MHz | 4 MHz | 8 MHz | 03h | 1 MHz ⁽¹⁾ |

Note 1: A minimum of 16 MHz Fosc is required to get 1 MHz I²C.

PIC18F87K22 FAMILY

REGISTER 23-10: ANCON2: A/D PORT CONFIGURATION REGISTER 2

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|------------------------|------------------------|------------------------|------------------------|---------|---------|---------|---------|
| ANSEL23 ⁽¹⁾ | ANSEL22 ⁽¹⁾ | ANSEL21 ⁽¹⁾ | ANSEL20 ⁽¹⁾ | ANSEL19 | ANSEL18 | ANSEL17 | ANSEL16 |
| 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 **ANSEL<23:16>:** Analog Port Configuration bits (AN23 through AN16)⁽¹⁾

1 = Pin is configured as an analog channel; digital input is disabled and any inputs read as '0'

0 = Pin is configured as a digital port

Note 1: AN15 through AN12 and AN23 through AN20 are implemented only on 80-pin devices. For 64-pin devices, the corresponding ANSELx bits are still implemented for these channels, but have no effect.

The analog reference voltage is software-selectable to either the device's positive and negative supply voltage (AVDD and AVSS) or the voltage level on the RA3/AN3/VREF+ and RA2/AN2/VREF- pins. VREF+ has two additional Internal Reference Voltage selections: 2.048V and 4.096V.

The A/D Converter can uniquely operate while the device is in Sleep mode. To operate in Sleep, the A/D conversion clock must be derived from the A/D Converter's internal RC oscillator.

The output of the Sample-and-Hold (S/H) is the input into the converter, which generates the result via successive approximation.

Each port pin associated with the A/D Converter can be configured as an analog input or a digital I/O. The ADRESH and ADRESL registers contain the result of the A/D conversion. When the A/D conversion is complete, the result is loaded into the ADRESH:ADRESL register pair, the GO/DONE bit (ADCON0<1>) is cleared and the A/D Interrupt Flag bit, ADIF (PIR1<6>), is set.

A device Reset forces all registers to their Reset state. This forces the A/D module to be turned off and any conversion in progress is aborted. The value in the ADRESH:ADRESL register pair is not modified for a Power-on Reset. These registers will contain unknown data after a Power-on Reset.

The block diagram of the A/D module is shown in Figure 23-4.

24.6 Comparator Interrupts

The comparator interrupt flag is set whenever any of the following occurs:

- Low-to-high transition of the comparator output
- High-to-low transition of the comparator output
- Any change in the comparator output

The comparator interrupt selection is done by the $\text{EVPOL}<1:0>$ bits in the CMxCON register ($\text{CMxCON}<4:3>$).

In order to provide maximum flexibility, the output of the comparator may be inverted using the CPOL bit in the CMxCON register ($\text{CMxCON}<5>$). This is functionally identical to reversing the inverting and non-inverting inputs of the comparator for a particular mode.

An interrupt is generated on the low-to-high or high-to-low transition of the comparator output. This mode of interrupt generation is dependent on $\text{EVPOL}<1:0>$ in the CMxCON register. When $\text{EVPOL}<1:0> = 01$ or 10 , the interrupt is generated on a low-to-high or high-to-low transition of the comparator output. Once the interrupt is generated, it is required to clear the interrupt flag by software.

When $\text{EVPOL}<1:0> = 11$, the comparator interrupt flag is set whenever there is a change in the output value of either comparator. Software will need to maintain information about the status of the output bits, as read from $\text{CMSTAT}<7:5>$, to determine the actual change that occurred.

The CMPxIF bits ($\text{PIR6}<2:0>$) are the Comparator Interrupt Flags. The CMPxIF bits must be reset by clearing them. Since it is also possible to write a '1' to this register, a simulated interrupt may be initiated. Table 24-2 shows the interrupt generation with respect to comparator input voltages and EVPOL bit settings.

Both the CMPxIE bits ($\text{PIE6}<2:0>$) and the PEIE bit ($\text{INTCON}<6>$) must be set to enable the interrupt. In addition, the GIE bit ($\text{INTCON}<7>$) must also be set. If any of these bits are clear, the interrupt is not enabled, though the CMPxIF bits will still be set if an interrupt condition occurs.

A simplified diagram of the interrupt section is shown in Figure 24-3.

Note: The CMPxIF bits will not be set when $\text{EVPOL}<1:0> = 00$.

TABLE 24-2: COMPARATOR INTERRUPT GENERATION

| CPOL | EVPOL<1:0> | Comparator Input Change | CxOUT Transition | Interrupt Generated |
|------|------------|-------------------------|------------------|---------------------|
| 0 | 00 | $V_{IN+} > V_{IN-}$ | Low-to-High | No |
| | | $V_{IN+} < V_{IN-}$ | High-to-Low | No |
| | 01 | $V_{IN+} > V_{IN-}$ | Low-to-High | Yes |
| | | $V_{IN+} < V_{IN-}$ | High-to-Low | No |
| | 10 | $V_{IN+} > V_{IN-}$ | Low-to-High | No |
| | | $V_{IN+} < V_{IN-}$ | High-to-Low | Yes |
| | 11 | $V_{IN+} > V_{IN-}$ | Low-to-High | Yes |
| | | $V_{IN+} < V_{IN-}$ | High-to-Low | Yes |
| 1 | 00 | $V_{IN+} > V_{IN-}$ | High-to-Low | No |
| | | $V_{IN+} < V_{IN-}$ | Low-to-High | No |
| | 01 | $V_{IN+} > V_{IN-}$ | High-to-Low | No |
| | | $V_{IN+} < V_{IN-}$ | Low-to-High | Yes |
| | 10 | $V_{IN+} > V_{IN-}$ | High-to-Low | Yes |
| | | $V_{IN+} < V_{IN-}$ | Low-to-High | No |
| | 11 | $V_{IN+} > V_{IN-}$ | High-to-Low | Yes |
| | | $V_{IN+} < V_{IN-}$ | Low-to-High | Yes |

28.0 SPECIAL FEATURES OF THE CPU

The PIC18F87K22 family of devices includes several features intended to maximize reliability and minimize cost through elimination of external components. These include:

- Oscillator Selection
- Resets:
 - Power-on Reset (POR)
 - Power-up Timer (PWRT)
 - Oscillator Start-up Timer (OST)
 - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT) and On-chip Regulator
- Fail-Safe Clock Monitor
- Two-Speed Start-up
- Code Protection
- ID Locations
- In-Circuit Serial Programming™ (ICSP™)

The oscillator can be configured for the application depending on frequency, power, accuracy and cost. All of the options are discussed in detail in **Section 3.0 “Oscillator Configurations”**.

A complete discussion of device Resets and interrupts is available in previous sections of this data sheet.

In addition to their Power-up and Oscillator Start-up Timers provided for Resets, the PIC18F87K22 family of devices has a Watchdog Timer, which is either permanently enabled via the Configuration bits or software controlled (if configured as disabled).

The inclusion of an internal RC oscillator (LF-INTOSC) also provides the additional benefits of a Fail-Safe Clock Monitor (FSCM) and Two-Speed Start-up. FSCM provides for background monitoring of the peripheral clock and automatic switchover in the event of its failure. Two-Speed Start-up enables code to be executed almost immediately on start-up, while the primary clock source completes its start-up delays.

All of these features are enabled and configured by setting the appropriate Configuration register bits.

28.1 Configuration Bits

The Configuration bits can be programmed (read as ‘0’) or left unprogrammed (read as ‘1’) to select various device configurations. These bits are mapped starting at program memory location, 300000h.

The user will note that address, 300000h, is beyond the user program memory space. In fact, it belongs to the configuration memory space (300000h-3FFFFh), which can only be accessed using table reads and table writes.

Software programming of the Configuration registers is done in a manner similar to programming the Flash memory. The WR bit in the EECON1 register starts a self-timed write to the Configuration register. In normal Operation mode, a TBLWT instruction, with the TBLPTR pointing to the Configuration register, sets up the address and the data for the Configuration register write. Setting the WR bit starts a long write to the Configuration register. The Configuration registers are written a byte at a time. To write or erase a configuration cell, a TBLWT instruction can write a ‘1’ or a ‘0’ into the cell. For additional details on Flash programming, refer to **Section 7.5 “Writing to Flash Program Memory”**.

PIC18F87K22 FAMILY

TABLE 29-2: PIC18F87K22 FAMILY INSTRUCTION SET

| Mnemonic, Operands | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes | |
|--------------------------|---------------------------------|--|-------------------------|------|------|------|--------------------|-----------------|------------|
| | | | MSb | | LSb | | | | |
| BYTE-ORIENTED OPERATIONS | | | | | | | | | |
| ADDWF | f, d, a | Add WREG and f | 1 | 0010 | 01da | ffff | ffff | C, DC, Z, OV, N | 1, 2 |
| ADDWFC | f, d, a | Add WREG and Carry bit to f | 1 | 0010 | 00da | ffff | ffff | C, DC, Z, OV, N | 1, 2 |
| ANDWF | f, d, a | AND WREG with f | 1 | 0001 | 01da | ffff | ffff | Z, N | 1, 2 |
| CLRF | f, a | Clear f | 1 | 0110 | 101a | ffff | ffff | Z | 2 |
| COMF | f, d, a | Complement f | 1 | 0001 | 11da | ffff | ffff | Z, N | 1, 2 |
| CPFSEQ | f, a | Compare f with WREG, Skip = | 1 (2 or 3) | 0110 | 001a | ffff | ffff | None | 4 |
| CPFSGT | f, a | Compare f with WREG, Skip > | 1 (2 or 3) | 0110 | 010a | ffff | ffff | None | 4 |
| CPFSLT | f, a | Compare f with WREG, Skip < | 1 (2 or 3) | 0110 | 000a | ffff | ffff | None | 1, 2 |
| DECf | f, d, a | Decrement f | 1 | 0000 | 01da | ffff | ffff | C, DC, Z, OV, N | 1, 2, 3, 4 |
| DECFSZ | f, d, a | Decrement f, Skip if 0 | 1 (2 or 3) | 0010 | 11da | ffff | ffff | None | 1, 2, 3, 4 |
| DCFSNZ | f, d, a | Decrement f, Skip if Not 0 | 1 (2 or 3) | 0100 | 11da | ffff | ffff | None | 1, 2 |
| INCF | f, d, a | Increment f | 1 | 0010 | 10da | ffff | ffff | C, DC, Z, OV, N | 1, 2, 3, 4 |
| INCFSZ | f, d, a | Increment f, Skip if 0 | 1 (2 or 3) | 0011 | 11da | ffff | ffff | None | 4 |
| INFSNZ | f, d, a | Increment f, Skip if Not 0 | 1 (2 or 3) | 0100 | 10da | ffff | ffff | None | 1, 2 |
| IORWF | f, d, a | Inclusive OR WREG with f | 1 | 0001 | 00da | ffff | ffff | Z, N | 1, 2 |
| MOVF | f, d, a | Move f | 1 | 0101 | 00da | ffff | ffff | Z, N | 1 |
| MOVFF | f _s , f _d | Move f _s (source) to f _d (destination) | 2 | 1100 | ffff | ffff | ffff | None | |
| | | 1st word | | 1111 | ffff | ffff | ffff | | |
| | | 2nd word | | | | | | | |
| MOVWF | f, a | Move WREG to f | 1 | 0110 | 111a | ffff | ffff | None | |
| MULWF | f, a | Multiply WREG with f | 1 | 0000 | 001a | ffff | ffff | None | 1, 2 |
| NEGF | f, a | Negate f | 1 | 0110 | 110a | ffff | ffff | C, DC, Z, OV, N | |
| RLCF | f, d, a | Rotate Left f through Carry | 1 | 0011 | 01da | ffff | ffff | C, Z, N | 1, 2 |
| RLNCF | f, d, a | Rotate Left f (No Carry) | 1 | 0100 | 01da | ffff | ffff | Z, N | |
| RRCF | f, d, a | Rotate Right f through Carry | 1 | 0011 | 00da | ffff | ffff | C, Z, N | |
| RRNCF | f, d, a | Rotate Right f (No Carry) | 1 | 0100 | 00da | ffff | ffff | Z, N | |
| SETF | f, a | Set f | 1 | 0110 | 100a | ffff | ffff | None | 1, 2 |
| SUBFWB | f, d, a | Subtract f from WREG with Borrow | 1 | 0101 | 01da | ffff | ffff | C, DC, Z, OV, N | |
| SUBWF | f, d, a | Subtract WREG from f | 1 | 0101 | 11da | ffff | ffff | C, DC, Z, OV, N | 1, 2 |
| SUBWFB | f, d, a | Subtract WREG from f with Borrow | 1 | 0101 | 10da | ffff | ffff | C, DC, Z, OV, N | |
| SWAPF | f, d, a | Swap Nibbles in f | 1 | 0011 | 10da | ffff | ffff | None | 4 |
| TSTFSZ | f, a | Test f, Skip if 0 | 1 (2 or 3) | 0110 | 011a | ffff | ffff | None | 1, 2 |
| XORWF | f, d, a | Exclusive OR WREG with f | 1 | 0001 | 10da | ffff | ffff | Z, N | |

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

- 2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.
- 3: If the Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.
- 4: Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

PIC18F87K22 FAMILY

ADDWFC ADD W and Carry bit to f

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

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

Operation: $(W) + (f) + (C) \rightarrow \text{dest}$

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

Encoding:

| | | | |
|------|------|------|------|
| 0010 | 00da | ffff | ffff |
|------|------|------|------|

Description: Add W, the Carry flag and data memory location 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in data memory location 'f'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See **Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: ADDWFC REG, 0, 1

Before Instruction

Carry bit = 1
REG = 02h
W = 4Dh

After Instruction

Carry bit = 0
REG = 02h
W = 50h

ANDLW AND Literal with W

Syntax: ANDLW k

Operands: $0 \leq k \leq 255$

Operation: $(W) .\text{AND}. k \rightarrow W$

Status Affected: N, Z

Encoding:

| | | | |
|------|------|------|------|
| 0000 | 1011 | kkkk | kkkk |
|------|------|------|------|

Description: The contents of W are ANDed with the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: ANDLW 05Fh

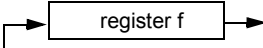
Before Instruction

W = A3h

After Instruction

W = 03h

PIC18F87K22 FAMILY

| RRNCF | | Rotate Right f (No Carry) | | | | | | | | | | | |
|-------------------|---|---------------------------|----------------------|--|--|------|------|------|------|--------|-------------------|--------------|----------------------|
| Syntax: | RRNCF f {,d {,a}} | | | | | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$ | | | | | | | | | | | | |
| Operation: | $(f < n) \rightarrow \text{dest} < n - 1 >$, $(f < 0) \rightarrow \text{dest} < 7 >$ | | | | | | | | | | | | |
| Status Affected: | N, Z | | | | | | | | | | | | |
| Encoding: | <table><tr><td>0100</td><td>00da</td><td>ffff</td><td>ffff</td></tr></table> | | | | | 0100 | 00da | ffff | ffff | | | | |
| 0100 | 00da | ffff | ffff | | | | | | | | | | |
| Description: | <p>The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'.</p> <p>If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 29.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.</p> <div></div> | | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | | |
| Cycles: | 1 | | | | | | | | | | | | |
| Q Cycle Activity: | <table><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr></table> | | | | | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| Decode | Read register 'f' | Process Data | Write to destination | | | | | | | | | | |

Example 1: RRNCF REG, 1, 0

Before Instruction
 REG = 1101 0111
 After Instruction
 REG = 1110 1011

Example 2: RRNCF REG, 0, 0

Before Instruction
 W = ?
 REG = 1101 0111
 After Instruction
 W = 1110 1011
 REG = 1101 0111

| SETF | | Set f | | | | | | | |
|-------------------|--|-------------------|--------------|--------------------|--|------|------|------|------|
| Syntax: | SETF f {,a} | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $a \in [0,1]$ | | | | | | | | |
| Operation: | $\text{FFh} \rightarrow f$ | | | | | | | | |
| Status Affected: | None | | | | | | | | |
| Encoding: | <table border="1"><tr><td>0110</td><td>100a</td><td>ffff</td><td>ffff</td></tr></table> | | | | | 0110 | 100a | ffff | ffff |
| 0110 | 100a | ffff | ffff | | | | | | |
| Description: | <p>The contents of the specified register are set to FFh.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 29.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | |
| | Q1 | Q2 | Q3 | Q4 | | | | | |
| | Decode | Read register 'f' | Process Data | Write register 'f' | | | | | |

Example: SETF REG, 1

Before Instruction
 REG = 5Ah
 After Instruction
 REG = FFh

PIC18F87K22 FAMILY

MOVSS Move Indexed to Indexed

| | |
|-------------------|---|
| Syntax: | MOVSS [z _s], [z _d] |
| Operands: | 0 ≤ z _s ≤ 127 0 ≤ z _d ≤ 127 |
| Operation: | ((FSR2) + z _s) → ((FSR2) + z _d) |
| Status Affected: | None |
| Encoding: | |
| 1st word (source) | 1110 1011 1zzz zzzz _s |
| 2nd word (dest.) | 1111 xxxx xzzz zzzz _d |

Description

The contents of the source register are moved to the destination register. The addresses of the source and destination registers are determined by adding the 7-bit literal offsets, 'z_s' or 'z_d', respectively, to the value of FSR2. Both registers can be located anywhere in the 4096-byte data memory space (000h to FFFh).

The MOVSS instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

If the resultant source address points to an Indirect Addressing register, the value returned will be 00h. If the resultant destination address points to an Indirect Addressing register, the instruction will execute as a NOP.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-----------------------|-----------------------|-------------------|
| Decode | Determine source addr | Determine source addr | Read source reg |
| Decode | Determine dest addr | Determine dest addr | Write to dest reg |

Example: MOVSS [05h], [06h]

Before Instruction

FSR2 = 80h
 Contents of 85h = 33h
 Contents of 86h = 11h

After Instruction

FSR2 = 80h
 Contents of 85h = 33h
 Contents of 86h = 33h

PUSHL Store Literal at FSR2, Decrement FSR2

| | |
|------------------|--------------------------------|
| Syntax: | PUSHL k |
| Operands: | 0 ≤ k ≤ 255 |
| Operation: | k → (FSR2), FSR2 – 1 → FSR2 |
| Status Affected: | None |
| Encoding: | 1110 1010 kkkk kkkk |
| Description: | |

The 8-bit literal 'k' is written to the data memory address specified by FSR2. FSR2 is decremented by 1 after the operation.

This instruction allows users to push values onto a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|----------|--------------|----------------------|
| Decode | Read 'k' | Process data | Write to destination |

Example: PUSHL 08h

Before Instruction

FSR2H:FSR2L = 01ECh
 Memory (01ECh) = 00h

After Instruction

FSR2H:FSR2L = 01EBh
 Memory (01ECh) = 08h

PIC18F87K22 FAMILY

31.1 DC Characteristics: Supply Voltage PIC18F87K22 Family (Industrial/Extended)

| PIC18F87K22 Family (Industrial/Extended) | | | Standard Operating Conditions (unless otherwise stated) | | | | |
|---|--------|--|---|--------------------------|------------------------------|--------|--|
| | | | Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended | | | | |
| Param No. | Symbol | Characteristic | Min | Typ | Max | Units | Conditions |
| D001 | VDD | Supply Voltage | 1.8 1.8 | — — | 3.6 5.5 | V V | ENVREG tied to Vss ENVREG tied to VDD |
| D001C | AVDD | Analog Supply Voltage | VDD – 0.3 | — | VDD + 0.3 | V | |
| D001D | AVSS | Analog Ground Potential | VSS – 0.3 | — | VSS + 0.3 | V | |
| D002 | VDR | RAM Data Retention Voltage ⁽¹⁾ | 1.5 | — | — | V | |
| D003 | VPOR | VDD Start Voltage to Ensure Internal Power-on Reset Signal | — | — | 0.7 | V | See Section 5.3 “Power-on Reset (POR)” for details |
| D004 | SVDD | VDD Rise Rate to Ensure Internal Power-on Reset Signal | 0.05 | — | — | V/ms | See Section 5.3 “Power-on Reset (POR)” for details |
| D005 | BVDD | Brown-out Reset Voltage (High/Medium/Low-Power mode) ⁽²⁾ BORV<1:0> = 11 ⁽³⁾ BORV<1:0> = 10 BORV<1:0> = 01 BORV<1:0> = 00 | 1.69 1.88 2.53 2.82 | 1.8 2.0 2.7 3.0 | 1.91 2.12 2.86 3.18 | | |

- Note 1:** This is the limit to which VDD can be lowered in Sleep mode, or during a device Reset, without losing RAM data.
- 2:** The following values are taken in HP-BOR mode.
- 3:** The device will operate normally until Brown-out Reset occurs, even though VDD may be below VDDMIN.