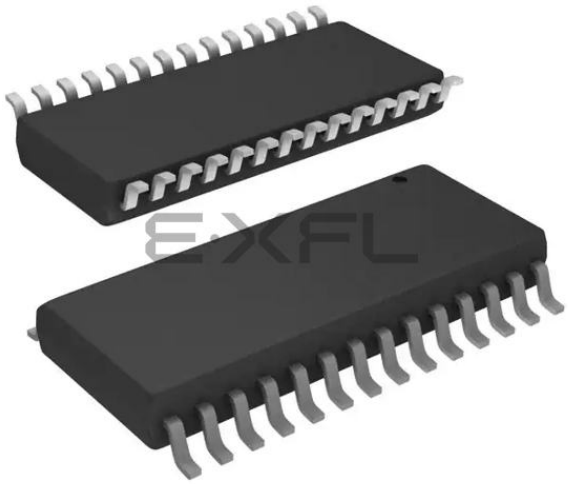


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	64MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, HLVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 24x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf25k42-i-so

4.7.2 DIRECT ADDRESSING

Direct addressing specifies all or part of the source and/or destination address of the operation within the opcode itself. The options are specified by the arguments accompanying the instruction.

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of direct addressing by default. All of these instructions include some 8-bit literal address as their Least Significant Byte. This address specifies either a register address in one of the banks of data RAM (**Section 4.5.3 "General Purpose Register File"**) or a location in the Access Bank (**Section 4.5.2 "Access Bank"**) as the data source for the instruction.

The Access RAM bit 'a' determines how the address is interpreted. When 'a' is '1', the contents of the BSR (**Section 4.5.1 "Bank Select Register (BSR)"**) are used with the address to determine the complete 14-bit address of the register. When 'a' is '0', the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

A few instructions, such as MOVFFL, include the entire 14-bit address (either source or destination) in their opcodes. In these cases, the BSR is ignored entirely.

The destination of the operation's results is determined by the destination bit 'd'. When 'd' is '1', the results are stored back in the source register, overwriting its original contents. When 'd' is '0', the results are stored in the W register. Instructions without the 'd' argument have a destination that is implicit in the instruction; their destination is either the target register being operated on or the W register.

4.7.3 INDIRECT ADDRESSING

Indirect addressing allows the user to access a location in data memory without giving a fixed address in the instruction. This is done by using File Select Registers (FSRs) as pointers to the locations which are to be read or written. Since the FSRs are themselves located in RAM as Special File Registers, they can also be directly manipulated under program control. This makes FSRs very useful in implementing data structures, such as tables and arrays in data memory.

The registers for indirect addressing are also implemented with Indirect File Operands (INDFs) that permit automatic manipulation of the pointer value with auto-incrementing, auto-decrementing or offsetting with another value. This allows for efficient code, using loops, such as the example of clearing an entire RAM bank in Example 4-6.

EXAMPLE 4-6: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

```
        LFSR   FSR0, 100h ;
NEXT    CLRF   POSTINC0   ; Clear INDF
                          ; register then
                          ; inc pointer
        BTFSS  FSR0H, 1   ; All done with
                          ; Bank1?
        BRA    NEXT       ; NO, clear next
CONTINUE                  ; YES, continue
```

4.7.3.1 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. Each FSR pair holds a 14-bit value, therefore, the two upper bits of the FSRnH register are not used. The 14-bit FSR value 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; they are mapped in the SFR space but are not physically implemented. Reading or writing to a particular INDF register actually accesses the data addressed by 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 14-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.

5.5 Device ID and Revision ID

The 16-bit device ID word is located at 3F FFFEh and the 16-bit revision ID is located at 3F FFFCh. These locations are read-only and cannot be erased or modified.

Development tools, such as device programmers and debuggers, may be used to read the Device ID, Revision ID and Configuration Words. Refer to **15.0 "Nonvolatile Memory (NVM) Control"** for more information on accessing these locations.

5.6 Register Definitions: Device ID and Revision ID

REGISTER 5-10: DEVICE ID: DEVICE ID REGISTER

R	R	R	R	R	R	R	R
DEV15	DEV14	DEV13	DEV12	DEV11	DEV10	DEV9	DEV8
bit 15							bit 8

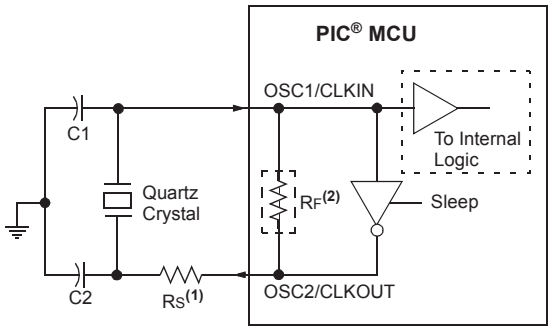
R	R	R	R	R	R	R	R
DEV7	DEV6	DEV5	DEV4	DEV3	DEV2	DEV1	DEV0
bit 7							bit 0

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

bit 15-0 **DEV<15:0>:** Device ID bits

Device	Device ID
PIC18F24K42	6CA0h
PIC18F25K42	6C80h
PIC18LF24K42	6DE0h
PIC18LF25K42	6DC0h

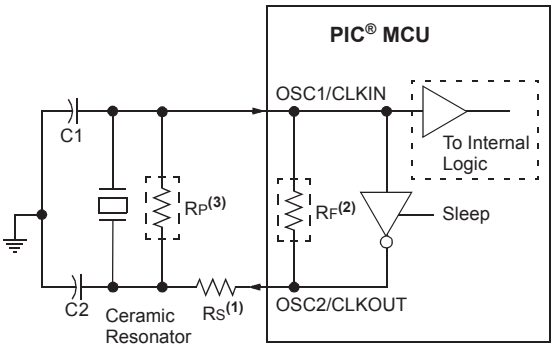
FIGURE 9-3: QUARTZ CRYSTAL OPERATION (LP, XT OR HS MODE)



Note 1: A series resistor (RS) may be required for quartz crystals with low drive level.

2: The value of RF varies with the Oscillator mode selected (typically between 2 MΩ to 10 MΩ).

FIGURE 9-4: CERAMIC RESONATOR OPERATION (XT OR HS MODE)



Note 1: A series resistor (RS) may be required for ceramic resonators with low drive level.

2: The value of RF varies with the Oscillator mode selected (typically between 2 MΩ to 10 MΩ).

3: An additional parallel feedback resistor (RP) may be required for proper ceramic resonator operation.

9.2.1.3 Oscillator Start-up Timer (OST)

If the oscillator module is configured for LP, XT or HS modes, the Oscillator Start-up Timer (OST) counts 1024 oscillations from OSC1. This occurs following a Power-on Reset (POR), or a wake-up from Sleep. The OST ensures that the oscillator circuit, using a quartz crystal resonator or ceramic resonator, has started and is providing a stable system clock to the oscillator module.

9.2.1.4 4x PLL

The oscillator module contains a 4x PLL that can be used with the external clock sources to provide a system clock source. The input frequency for the PLL must fall within specifications. See the PLL Clock Timing Specifications in Table 46-9.

The PLL can be enabled for use by one of two methods:

1. Program the RSTOSC bits in the Configuration Word 1 to 010 (enable EXTOSC with 4x PLL).
2. Write the NOSC bits in the OSCCON1 register to 010 (enable EXTOSC with 4x PLL).

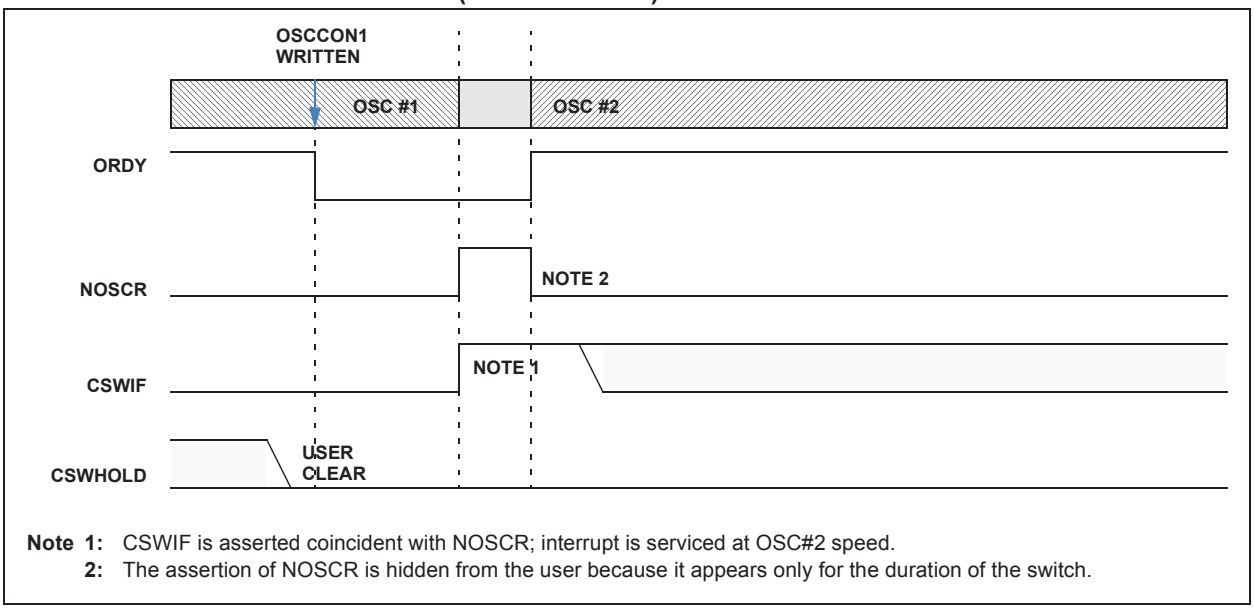
9.3.2 CLOCK SWITCH AND SLEEP

If OSCCON1 is written with a new value and the device is put to Sleep before the switch completes, the switch will not take place and the device will enter Sleep mode.

When the device wakes from Sleep and the CSWHOLD bit is clear, the device will wake with the 'new' clock active, and the Clock Switch Interrupt flag bit (CSWIF) will be set.

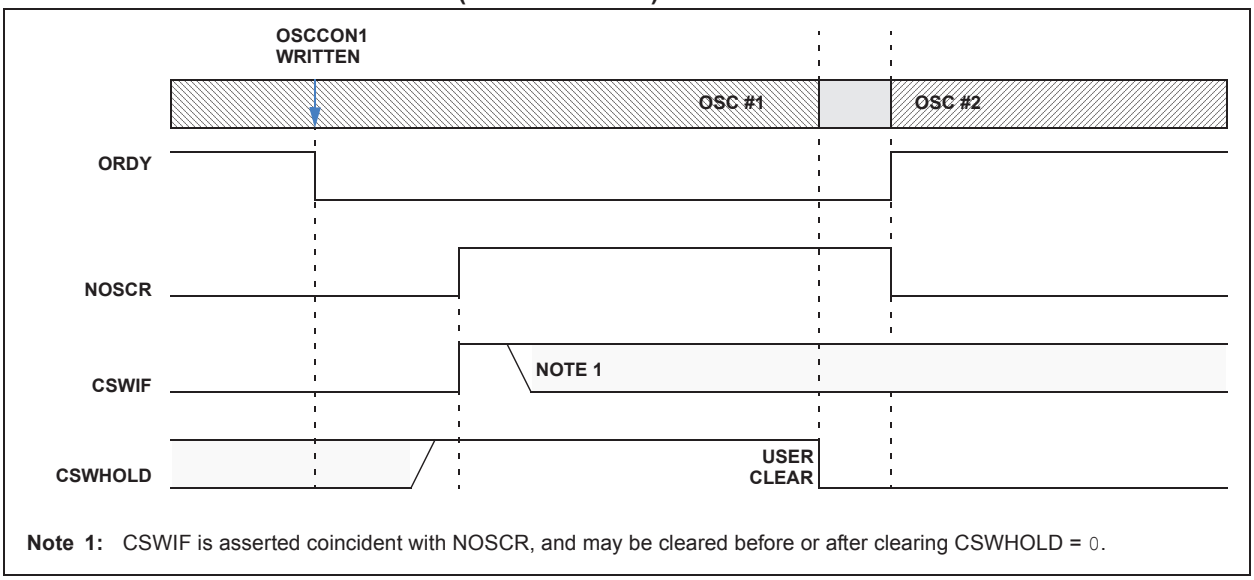
When the device wakes from Sleep and the CSWHOLD bit is set, the device will wake with the 'old' clock active and the new clock will be requested again.

FIGURE 9-6: CLOCK SWITCH (CSWHOLD = 0)



Note 1: CSWIF is asserted coincident with NOSCR; interrupt is serviced at OSC#2 speed.
2: The assertion of NOSCR is hidden from the user because it appears only for the duration of the switch.

FIGURE 9-7: CLOCK SWITCH (CSWHOLD = 1)



Note 1: CSWIF is asserted coincident with NOSCR, and may be cleared before or after clearing CSWHOLD = 0.

13.7 Register Definitions: Windowed Watchdog Timer Control

REGISTER 13-1: WDTCON0: WATCHDOG TIMER CONTROL REGISTER 0

U-0	U-0	R/W(3)-q/q(2)	R/W(3)-q/q(2)	R/W(3)-q/q(2)	R/W(3)-q/q(2)	R/W(3)-q/q(2)	R/W-0/0
—	—	PS<4:0>					SEN
bit 7							bit 0

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

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

bit 5-1 **PS<4:0>:** Watchdog Timer Prescale Select bits(1)

Bit Value = Prescale Rate

11111 = Reserved. Results in minimum interval (1:32)
•
•
•
10011 = Reserved. Results in minimum interval (1:32)

10010 = 1:8388608 (2^{23}) (Interval 256s nominal)
10001 = 1:4194304 (2^{22}) (Interval 128s nominal)
10000 = 1:2097152 (2^{21}) (Interval 64s nominal)
01111 = 1:1048576 (2^{20}) (Interval 32s nominal)
01110 = 1:524288 (2^{19}) (Interval 16s nominal)
01101 = 1:262144 (2^{18}) (Interval 8s nominal)
01100 = 1:131072 (2^{17}) (Interval 4s nominal)
01011 = 1:65536 (Interval 2s nominal) (Reset value)
01010 = 1:32768 (Interval 1s nominal)
01001 = 1:16384 (Interval 512 ms nominal)
01000 = 1:8192 (Interval 256 ms nominal)
00111 = 1:4096 (Interval 128 ms nominal)
00110 = 1:2048 (Interval 64 ms nominal)
00101 = 1:1024 (Interval 32 ms nominal)
00100 = 1:512 (Interval 16 ms nominal)
00011 = 1:256 (Interval 8 ms nominal)
00010 = 1:128 (Interval 4 ms nominal)
00001 = 1:64 (Interval 2 ms nominal)
00000 = 1:32 (Interval 1 ms nominal)

bit 0 **SEN:** Software Enable/Disable for Watchdog Timer bit

If WDTE<1:0> = 1x:
This bit is ignored.
If WDTE<1:0> = 01:
1 = WDT is turned on
0 = WDT is turned off
If WDTE<1:0> = 00:
This bit is ignored.

Note 1: Times are approximate. WDT time is based on 31 kHz LFINTOSC.

2: When WDTCPS <4:0> in CONFIG3L = 11111, the Reset value of PS<4:0> is 01011. Otherwise, the Reset value of PS<4:0> is equal to WDTCPS<4:0> in CONFIG3L.

3: When WDTCPS <4:0> in CONFIG3L ≠ 11111, these bits are read-only.

4: When the WWDT is configured to run using the SOSC as a clock source and the device is allowed to undergo a Reset, as triggered by a WDT time-out, the SOSC would also undergo a Reset. That means the SOSC will execute its start-up sequence which requires 1024 SOSC clock counts before it is made available for peripherals to use. So for example, if the WDT is set for a 1 ms time-out and the device is allowed to undergo a WDT Reset, then the actual WDT Reset period will be: WDT_PERIOD = (1/(SOSC_FREQUENCY) * 1024) + 1 ms.

EXAMPLE 17-1: SETUP DMA1 TO MOVE DATA FROM PROGRAM FLASH MEMORY TO UART1 TRANSMIT BUFFER USING HARDWARE TRIGGERS

```
//This code example illustrates using DMA1 to transfer
//10 bytes of data from 0x1000 in PFM to U1TXB 0x3DEA

void main() {
    //System Initialize
    initializeSystem();

    //Setup UART1
    initializeUART1();

    //Setup DMA1
    //DMA1CON1 - DPTR remains, Source Memory Region PFM, SPTR increments, SSTP
    DMA1CON1 = 0x0B;

    //Source registers
    //Source size
    DMA1SSZH = 0x00;
    DMA1SSZL = 0x0A;

    //Source start address, 0x1000
    DMA1SSAU = 0x00;
    DMA1SSAH = 0x10;
    DMA1SSAL = 0x00;

    //Destination registers
    //Destination size
    DMA1DSZH = 0x00;
    DMA1DSZL = 0x01;

    //Destination start address, 0x3DEA
    DMA1DSAH = 0x3D;
    DMA1DSAL = 0xEA;

    //Start trigger source U1TX
    DMA1SIRQ = 0x1C;

    //Enable & Start DMA transfer
    DMA1CON0 = 0xC0;

    while (1) {
        doSomething();
    }
}
```

17.13 Register definitions: DMA

Long bit name prefixes for the DMA peripherals are shown in Table 17-7. Refer to **Section 1.4 "Register and Bit naming conventions"** for more information.

TABLE 17-7: REGISTER AND BIT NAMING

Peripheral	Bit Name Prefix
DMA 1	DMA1
DMA 2	DMA2

REGISTER 17-12: DMAxSCNTL – DMAx SOURCE COUNT LOW REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
SCNT<7:0>							
bit 7							bit 0

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

bit 7-0 **SCNT<7:0>:** Current Source Byte Count

REGISTER 17-13: DMAxSCNTH – DMAx SOURCE COUNT HIGH REGISTER

U-0	U-0	U-0	U-0	R-0	R-0	R-0	R-0
—	—	—	—	SCNT<11:8>			
bit 7							bit 0

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

bit 7-4 **Unimplemented**: Read as '0'

bit 3-0 **SCNT<11:8>:** Current Source Byte Count

REGISTER 17-14: DMAxDSAL – DMAx DESTINATION START ADDRESS LOW REGISTER

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

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

bit 7-0 **DSA<7:0>:** Destination Start Address bits

18.4 PORTE Functionality

18.4.1 MASTER CLEAR INPUT $\overline{\text{MCLR}}$

For PIC18(L)F2xK42 devices, PORTE is only available when Master Clear functionality is disabled (MCLRE = 0). In this case, PORTE is a single bit, input-only port comprised of RE3 only. The pin operates as previously described. RE3 in PORTE register is a read-only bit and will read '1' when MCLRE = 1 (i.e., Master Clear enabled).

18.4.2 RE3 WEAK PULL-UP

The port RE3 pin has an individually controlled weak internal pull-up. When set, the WPUE3 bit enables the RE3 pin pull-up. When the RE3 port pin is configured as $\overline{\text{MCLR}}$, (CONFIG2L, MCLRE = 1 and CONFIG4H, LVP = 0), or configured for Low-Voltage Programming, (MCLRE = x and LVP = 1), the pull-up is always enabled and the WPUE3 bit has no effect.

18.4.3 INTERRUPT-ON-CHANGE

The interrupt-on-change feature is available only on the RE3 pin of PORTE for all devices. If MCLRE = 1 or LVP = 1, RE3 port functionality is disabled and interrupt-on-change on RE3 is not available. For further details refer to **Section 9.11 "Interrupt-on-Change"**.

19.3 Bidirectional Pins

PPS selections for peripherals with bidirectional signals on a single pin must be made so that the PPS input and PPS output select the same pin. Peripherals that have bidirectional signals include:

- I^2C
- CCP module

Note:	Refer to **Section TABLE 19-1: "PPS Input Register Details"** for pins that are I^2C compatible. Clock and data signals can be routed to any pin, however pins without I^2C compatibility will operate at standard TTL/ST logic levels as selected by the INVLV register.

19.4 PPS Lock

The PPS includes a mode in which all input and output selections can be locked to prevent inadvertent changes. PPS selections are locked by setting the PPSLOCKED bit of the PPSLOCK register. Setting and clearing this bit requires a special sequence as an extra precaution against inadvertent changes. Examples of setting and clearing the PPSLOCKED bit are shown in Example 19-1.

EXAMPLE 19-1: PPS LOCK SEQUENCE

```
; Disable interrupts:
    BCF     INTCON0,GIE

; Bank to PPSLOCK register
    BANKSEL  PPSLOCK
    MOVLB    PPSLOCK
    MOVLW    55h

; Required sequence, next 4 instructions
    MOVWF    PPSLOCK
    MOVLW    AAh
    MOVWF    PPSLOCK

; Set PPSLOCKED bit to disable writes
; Only a BSF instruction will work
    BSF     PPSLOCK,0

; Enable Interrupts
    BSF     INTCON0,GIE
```

EXAMPLE 19-2: PPS UNLOCK SEQUENCE

```
; Disable interrupts:
    BCF     INTCON0,GIE

; Bank to PPSLOCK register
    BANKSEL  PPSLOCK
    MOVLB    PPSLOCK
    MOVLW    55h

; Required sequence, next 4 instructions
    MOVWF    PPSLOCK
    MOVLW    AAh
    MOVWF    PPSLOCK

; Clear PPSLOCKED bit to enable writes
; Only a BCF instruction will work
    BCF     PPSLOCK,0

; Enable Interrupts
    BSF     INTCON0,GIE
```

19.5 PPS One-way Lock

When this bit is set, the PPSLOCKED bit can only be cleared and set one time after a device Reset. This allows for clearing the PPSLOCKED bit so that the input and output selections can be made during initialization. When the PPSLOCKED bit is set after all selections have been made, it will remain set and cannot be cleared until after the next device Reset event.

19.6 Operation During Sleep

PPS input and output selections are unaffected by Sleep.

19.7 Effects of a Reset

A device Power-on-Reset (POR) clears all PPS input and output selections to their default values. All other Resets leave the selections unchanged. Default input selections are shown in pin allocation . The PPS one-way lock is also removed.

REGISTER 21-2: PMD1: PMD CONTROL REGISTER 1

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
NCO1MD	TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD
bit 7							bit 0

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

bit 7 **NCO1MD:** Disable NCO1 Module bit
1 = NCO1 module disabled
0 = NCO1 module enabled

bit 6 **TMR6MD:** Disable Timer TMR6 bit
1 = TMR6 module disabled
0 = TMR6 module enabled

bit 5 **TMR5MD:** Disable Timer TMR5 bit
1 = TMR5 module disabled
0 = TMR5 module enabled

bit 4 **TMR4MD:** Disable Timer TMR4 bit
1 = TMR4 module disabled
0 = TMR4 module enabled

bit 3 **TMR3MD:** Disable Timer TMR3 bit
1 = TMR3 module disabled
0 = TMR3 module enabled

bit 2 **TMR2MD:** Disable Timer TMR2 bit
1 = TMR2 module disabled
0 = TMR2 module enabled

bit 1 **TMR1MD:** Disable Timer TMR1 bit
1 = TMR1 module disabled
0 = TMR1 module enabled

bit 0 **TMR0MD:** Disable Timer TMR0 bit
1 = TMR0 module disabled
0 = TMR0 module enabled

REGISTER 22-2: T0CON1: TIMER0 CONTROL REGISTER 1

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
CS<2:0>			ASYNC	CKPS<3:0>			
bit 7							bit 0

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

bit 7-5 **CS<2:0>:** Timer0 Clock Source Select bits
111 = CLC1
110 = SOSC
101 = MFINTOSC (500 kHz)
100 = LFINTOSC
011 = HFINTOSC
010 = FOSC/4
001 = Pin selected by T0CKIPPS (Inverted)
000 = Pin selected by T0CKIPPS (Non-inverted)

bit 4 **ASYNC:** TMR0 Input Asynchronization Enable bit
1 = The input to the TMR0 counter is not synchronized to system clocks
0 = The input to the TMR0 counter is synchronized to FOSC/4

bit 3-0 **CKPS<3:0>:** Prescaler Rate Select bit
1111 = 1:32768
1110 = 1:16384
1101 = 1:8192
1100 = 1:4096
1011 = 1:2048
1010 = 1:1024
1001 = 1:512
1000 = 1:256
0111 = 1:128
0110 = 1:64
0101 = 1:32
0100 = 1:16
0011 = 1:8
0010 = 1:4
0001 = 1:2
0000 = 1:1

REGISTER 23-4: TxGATE: TIMERx GATE ISM REGISTER

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

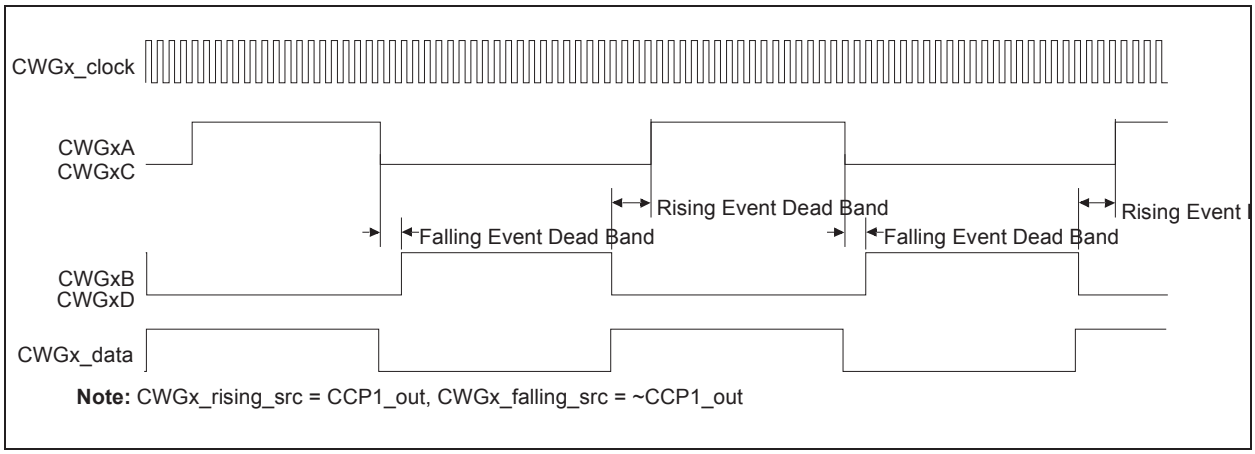
Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	u = unchanged

bit 7-5 **Unimplemented:** Read as '0'

bit 4-0 **GSS<4:0>:** Timerx Gate Source Selection bits

GSS	Timer1	Timer3	Timer5
	Gate Source	Gate Source	Gate Source
11111-11011	Reserved	Reserved	Reserved
11010	CLC4_out	CLC4_out	CLC4_out
11001	CLC3_out	CLC3_out	CLC3_out
11000	CLC2_out	CLC2_out	CLC2_out
10111	CLC1_out	CLC1_out	CLC1_out
10110	ZCDOUT	ZCDOUT	ZCDOUT
10101	CMP2OUT	CMP2OUT	CMP2OUT
10100	CMP1OUT	CMP1OUT	CMP1OUT
10011	NCO1OUT	NCO1OUT	NCO1OUT
10010-10001	Reserved	Reserved	Reserved
10000	PWM8OUT	PWM8OUT	PWM8OUT
01111	PWM7OUT	PWM7OUT	PWM7OUT
01110	PWM6OUT	PWM6OUT	PWM6OUT
01101	PWM5OUT	PWM5OUT	PWM5OUT
01100	CCP4OUT	CCP4OUT	CCP4OUT
01011	CCP3OUT	CCP3OUT	CCP3OUT
01010	CCP2OUT	CCP2OUT	CCP2OUT
01001	CCP1OUT	CCP1OUT	CCP1OUT
01000	SMT1_match	SMT1_match	SMT1_match
00111	TMR6OUT (post-scaled)	TMR6OUT (post-scaled)	TMR6OUT (post-scaled)
00110	TMR5 overflow	TMR5 overflow	Reserved
00101	TMR4OUT (post-scaled)	TMR4OUT (post-scaled)	TMR4OUT (post-scaled)
00100	TMR3 overflow	Reserved	TMR3 overflow
00011	TMR2OUT (post-scaled)	TMR2OUT (post-scaled)	TMR2OUT (post-scaled)
00010	Reserved	TMR1 overflow	TMR1 overflow
00001	TMR0 overflow	TMR0 overflow	TMR0 overflow
00000	Pin selected by T1GPPS	Pin selected by T3GPPS	Pin selected by T5GPPS

FIGURE 28-2: CWGx HALF-BRIDGE MODE OPERATION



Note: CWGx_rising_src = CCP1_out, CWGx_falling_src = ~CCP1_out

28.2.2 PUSH-PULL MODE

In Push-Pull mode, two output signals are generated, alternating copies of the input as illustrated in Figure 28-4. This alternation creates the push-pull effect required for driving some transformer-based power supply designs. Steering modes are not used in Push-Pull mode. A basic block diagram for the Push-Pull mode is shown in Figure 28-3.

The push-pull sequencer is reset whenever EN = 0 or if an auto-shutdown event occurs. The sequencer is clocked by the first input pulse, and the first output appears on CWGxA.

The unused outputs CWGxC and CWGxD drive copies of CWGxA and CWGxB, respectively, but with polarity controlled by the POLC and POLD bits of the CWGxCON1 register, respectively.

REGISTER 29-8: CLCxGLS1: GATE 1 LOGIC SELECT REGISTER

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
bit 7							bit 0

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

bit 7 **G2D4T:** Gate 1 Data 4 True (non-inverted) bit
1 = CLCIN3 (true) is gated into CLCx Gate 1
0 = CLCIN3 (true) is not gated into CLCx Gate 1

bit 6 **G2D4N:** Gate 1 Data 4 Negated (inverted) bit
1 = CLCIN3 (inverted) is gated into CLCx Gate 1
0 = CLCIN3 (inverted) is not gated into CLCx Gate 1

bit 5 **G2D3T:** Gate 1 Data 3 True (non-inverted) bit
1 = CLCIN2 (true) is gated into CLCx Gate 1
0 = CLCIN2 (true) is not gated into CLCx Gate 1

bit 4 **G2D3N:** Gate 1 Data 3 Negated (inverted) bit
1 = CLCIN2 (inverted) is gated into CLCx Gate 1
0 = CLCIN2 (inverted) is not gated into CLCx Gate 1

bit 3 **G2D2T:** Gate 1 Data 2 True (non-inverted) bit
1 = CLCIN1 (true) is gated into CLCx Gate 1
0 = CLCIN1 (true) is not gated into CLCx Gate 1

bit 2 **G2D2N:** Gate 1 Data 2 Negated (inverted) bit
1 = CLCIN1 (inverted) is gated into CLCx Gate 1
0 = CLCIN1 (inverted) is not gated into CLCx Gate 1

bit 1 **G2D1T:** Gate 1 Data 1 True (non-inverted) bit
1 = CLCIN0 (true) is gated into CLCx Gate 1
0 = CLCIN0 (true) is not gated into CLCx Gate1

bit 0 **G2D1N:** Gate 1 Data 1 Negated (inverted) bit
1 = CLCIN0 (inverted) is gated into CLCx Gate 1
0 = CLCIN0 (inverted) is not gated into CLCx Gate 1

REGISTER 34-6: SPIxBAUD: SPI BAUD RATE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
BAUD7	BAUD6	BAUD5	BAUD4	BAUD3	BAUD2	BAUD1	BAUD0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'

bit 7-0 BAUD<7:0>: Baud Clock Prescaler Select bits

SCK high or low time: TSC=SPI Clock Period*(BAUD+1)

SCK toggle frequency: FSCK=FBAUD= SPI Clock Frequency/(2*(BAUD+1))

Note: This register should not be written while the SPI is enabled (EN bit of SPIxCON0=1)

REGISTER 34-7: SPIxCON0: SPI CONFIGURATION REGISTER 0

R/W-0/0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
EN	—	—	—	—	LSBF	MST	BMODE
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'

bit 7 **EN**: SPI Module Enable Control bit

1 =SPI is enabled

0 = SPI is disabled,

bit 6-3 **Unimplemented**: Read as '0'

bit 2 **LSBF**: LSb-First Data Exchange bit

1 = Data is exchanged LSb first

0 = Data is exchanged MSb first (traditional SPI operation)

bit 1 **MST**: SPI Operating Mode Master Select bit

1 = SPI module operates as the bus master

0 = SPI module operates as a bus slave

bit 0 **BMODE**: Bit-Length Mode Select bit

1 = SPIxTWIDTH setting applies to every byte: total bits sent is SPIxTWIDTH*SPIxTCNT, end-of-packet occurs when SPIxTCNT = 0

0 = SPIxTWIDTH setting applies only to the last byte exchanged; total bits sent is SPIxTWIDTH + (SPIxTCNT*8)

Note: This register should only be written when the EN bit is cleared, or to clear the EN bit.

REGISTER 35-18: RxyI2C: I²C PAD Rxy CONTROL REGISTER

U-0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0
—	SLEW	PU<1:0>		—	—	TH<1:0>	
bit 7							bit 0

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

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

bit 6 **SLEW**: I²C Specific Slew Rate Limiting is Enabled
- 1 = I²C specific slew rate limiting is enabled. Standard pad slew limiting is disabled. The SLRxy bit is ignored.
- 0 = Standard GPIO Slew Rate; enabled/disabled via SLRxy bit.

bit 5-4 **PU<1:0>**: I²C Pull-Up Selection bits
- 11 = Reserved
- 10 = 10x current of standard weak pull-up
- 01 = 2x current of standard weak pull-up
- 00 = Standard GPIO weak pull-up, enabled via WPUxy bit

bit 3-2 **Unimplemented**: Read as '0'

bit 1-0 **TH<1:0>**: I²C Input Threshold Selection bits
- 11 = SMBus 3.0 (1.35 V) input threshold
- 10 = SMBus 2.0 (2.1 V) input threshold
- 01 = I²C specific input thresholds
- 00 = Standard GPIO Input pull-up, enabled via INLVLxy registers

38.4 ADC Charge Pump

The ADC module has a dedicated charge pump which can be controlled through the ADCP register (Register 38-36). The primary purpose of the charge pump is to supply a constant voltage to the gates of transistor devices in the A/D converter, signal and reference input pass-gates, to prevent degradation of transistor performance at low operating voltage.

The charge pump can be enabled by setting the CPON bit in the ADC register. Once enabled, the pump will undergo a start-up time to stabilize the charge pump output. Once the output stabilizes and is ready for use, the CPRDY bit of the ADCP register will be set.

38.5 Capacitive Voltage Divider (CVD) Features

The ADC module contains several features that allow the user to perform a relative capacitance measurement on any ADC channel using the internal ADC sample and hold capacitance as a reference. This relative capacitance measurement can be used to implement capacitive touch or proximity sensing applications. Figure 38-6 shows the basic block diagram of the CVD portion of the ADC module.

FIGURE 38-6: HARDWARE CAPACITIVE VOLTAGE DIVIDER BLOCK DIAGRAM

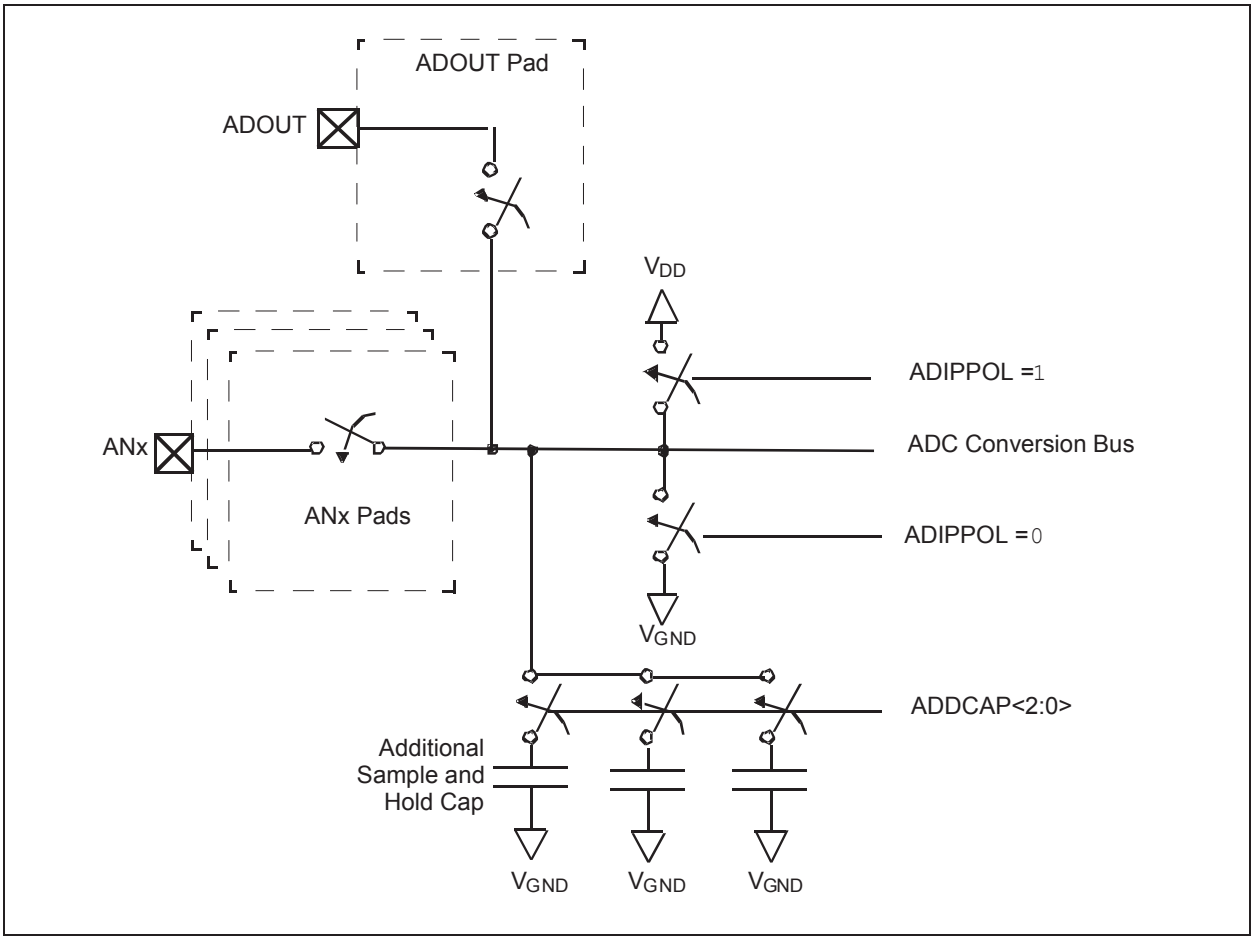


TABLE 38-2: COMPUTATION MODES

		Bit Clear Conditions	Value after Trigger completion		Threshold Operations			Value at ADTIF interrupt		
Mode	ADMD	ACC and CNT	ACC	CNT	Retrigger	Threshold Test	Interrupt	ADAOV	FLTR	CNT
Basic	0	ADACLR = 1	Unchanged	Unchanged	No	Every Sample	If threshold=true	N/A	N/A	count
Accumulate	1	ADACLR = 1	S + ACC or (S2-S1) + ACC	If (CNT=0xFF): CNT, otherwise: CNT+1	No	Every Sample	If threshold=true	ACC Overflow	$ACC/2^{ADCRS}$	count
Average	2	ADACLR = 1 or CNT>=RPT at GO or retrigger	S + ACC or (S2-S1) + ACC	If (CNT=0xFF): CNT, otherwise: CNT+1	No	If CNT>=RPT	If threshold=true	ACC Overflow	$ACC/2^{ADCRS}$	count
Burst Average	3	ADACLR = 1 or GO set or retrigger	Each repetition: same as Average End with sum of all samples	Each repetition: same as Average End with CNT=RPT	Repeat while CNT<RPT	If CNT>=RPT	If threshold=true	ACC Overflow	$ACC/2^{ADCRS}$	RPT
Low-pass Filter	4	ADACLR = 1	$S+ACC-ACC/2^{ADCRS}$ or $(S2-S1)+ACC-ACC/2^{ADCRS}$	Count up, stop counting when CNT = 0xFF	No	If CNT>=RPT	If threshold=true	ACC Overflow	Filtered Value	count

Note: S1 and S2 are abbreviations for Sample 1 and Sample 2, respectively. When ADDSEN = 0, S1 = ADRES; When ADDSEN = 1, S1 = PREV and S2 = ADRES.

MOVLW **Move literal to W**

Syntax: MOVLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow W$

Status Affected: None

Encoding:

0000	1110	kkkk	kkkk

Description: The 8-bit literal ‘k’ is loaded into W.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `MOVLW 5Ah`

After Instruction

W = 5Ah

MOVWF **Move W to f**

Syntax: MOVWF f {,a}

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

Operation: $(W) \rightarrow f$

Status Affected: None

Encoding:

0110	111a	ffff	ffff

Description: Move data from W to register ‘f’. Location ‘f’ can be anywhere in the 256-byte bank.
If ‘a’ is ‘0’, the Access Bank is selected.
If ‘a’ is ‘1’, the BSR is used to select the GPR bank.
If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 43.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write register ‘f’

Example: `MOVWF REG, 0`

Before Instruction

W = 4Fh
REG = FFh

After Instruction

W = 4Fh
REG = 4Fh

SWAPF	Swap f
Syntax:	SWAPF f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f<3:0>) → dest<7:4>, (f<7:4>) → dest<3:0>
Status Affected:	None

Encoding:

0011	10da	ffff	ffff

Description: The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in register 'f' (default).
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 43.2.3 "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:
```
SWAPF   REG, 1, 0
```
Before Instruction
REG = 53h
After Instruction
REG = 35h