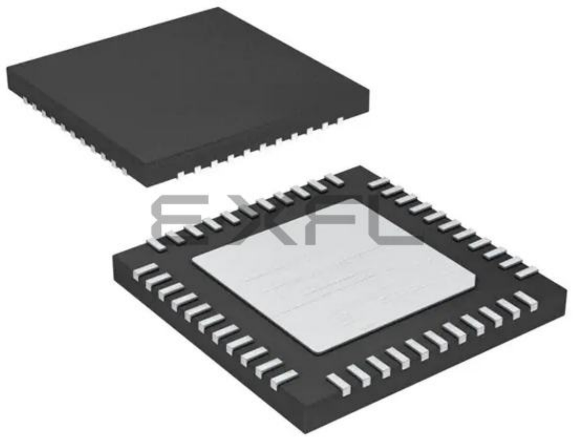
Microchip Technology - PIC18F45K42-I/ML Datasheet



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	36
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)	2.3V ~ 5.5V
Data Converters	A/D 35x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VQFN Exposed Pad
Supplier Device Package	44-QFN (8x8)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f45k42-i-ml

Pin Allocation Tables

TABLE 1: 28-PIN ALLOCATION TABLE (PIC18(L)F2XK42)

I/O	28-Pin SPDIP/SOIC/SSOP	28-Pin (U)QFN	ADC	Voltage Reference	DAC	Comparators	Zero Cross Detect	I^{2}C	SPI	UART	DSM	Timers/SMT	CCP and PWM	CWG	CLC	NCO	Clock Reference (CLKR)	Interrupt-on-Change	Basic
RA0	2	27	ANA0	—	—	C1IN0- C2IN0-	—	—	—	—	—	—	—	—	CLCIN0$^{(1)}$	—	—	IOCA0	—
RA1	3	28	ANA1	—	—	C1IN1- C2IN1-	—	—	—	—	—	—	—	—	CLCIN1$^{(1)}$	—	—	IOCA1	—
RA2	4	1	ANA2	VREF-	DAC1OUT1	C1IN0+ C2IN0+	—	—	—	—	—	—	—	—	—	—	—	IOCA2	—
RA3	5	2	ANA3	VREF+	—	C1IN1+	—	—	—	—	MDCARL$^{(1)}$	—	—	—	—	—	—	IOCA3	—
RA4	6	3	ANA4	—	—	—	—	—	—	—	MDCARH$^{(1)}$	T0CKI$^{(1)}$	—	—	—	—	—	IOCA4	—
RA5	7	4	ANA5	—	—	—	—	—	SS1$^{(1)}$	—	MDSRC$^{(1)}$	—	—	—	—	—	—	IOCA5	—
RA6	10	7	ANA6	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IOCA6	OSC2 CLKOUT
RA7	9	6	ANA7	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IOCA7	OSC1 CLKIN
RB0	21	18	ANB0	—	—	C2IN1+	ZCD	—	—	—	—	—	CCP4$^{(1)}$	CWG1IN$^{(1)}$	—	—	—	INT0$^{(1)}$ IOCB0	—
RB1	22	19	ANB1	—	—	C1IN3- C2IN3-	—	SCL2$^{(3,4)}$	—	—	—	—	—	CWG2IN$^{(1)}$	—	—	—	INT1$^{(1)}$ IOCB1	—
RB2	23	20	ANB2	—	—	—	—	SDA2$^{(3,4)}$	—	—	—	—	—	CWG3IN$^{(1)}$	—	—	—	INT2$^{(1)}$ IOCB2	—
RB3	24	21	ANB3	—	—	C1IN2- C2IN2-	—	—	—	—	—	—	—	—	—	—	—	IOCB3	—
RB4	25	22	ANB4 ADCACT$^{(1)}$	—	—	—	—	—	—	—	—	T5G$^{(1)}$	—	—	—	—	—	IOCB4	—
RB5	26	23	ANB5	—	—	—	—	—	—	—	—	T1G$^{(1)}$	CCP3$^{(1)}$	—	—	—	—	IOCB5	—
RB6	27	24	ANB6	—	—	—	—	—	—	CTS2$^{(1)}$	—	—	—	—	CLCIN2$^{(1)}$	—	—	IOCB6	ICSPCLK
RB7	28	25	ANB7	—	DAC1OUT2	—	—	—	—	RX2$^{(1)}$	—	T6IN(1)	—	—	CLCIN3$^{(1)}$	—	—	IOCB7	ICSPDAT

Note **1:** This is a PPS remappable input signal. The input function may be moved from the default location shown to one of several other PORTx pins.
2: All output signals shown in this row are PPS remappable.
3: This is a bidirectional signal. For normal module operation, the firmware should map this signal to the same pin in both the PPS input and PPS output registers.
4: These pins can be configured for I^{2}C and SMB™ 3.0/2.0 logic levels; The SCLx/SDAx signals may be assigned to any of the RB1/RB2/RC3/RC4 pins. PPS assignments to the other pins (e.g., RA5) will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the I^{2}C specific or SMBUS input buffer thresholds.

EXAMPLE 9-3: SETTING UP VECTORED INTERRUPTS USING MPASM

```
ISR_TMR0:   CODE        0x8C0                   ; ISR code at 0x08C0 in PFM
       BANKSEL         PIR0                    ; Select bank for PIR0
       BCF             PIR3, TMR0IF            ; Clear TMR0IF
       BTG             LATC, 0, ACCESS         ; Code to execute in ISR
       RETFIE          1                       ; Return from ISR

InterruptInit:
       BANKSEL         INTCON0                 ; Select bank for INTCON0
       BSF             INTCON0, GIEH           ; Enable high priority interrupts
       BSF             INTCON0, GIEL           ; Enable low priority interrupts
       BSF             INTCON0, IPEN_INTCON0   ; Enable interrupt priority

       BANKSEL         PIE0                    ; Select bank for PIE0
       BSF             PIE3, TMR0IE            ; Enable TMR0 interrupt
       BSF             PIE4, TMR1IE            ; Enable TMR1 interrupt

       BCF             IPR3, TMR0IP            ; Make TMR0 interrupt low priority
       RETURN          1

VectorTableInit:
       ; Set IVTBASE (optional - default is 0x000008)
       MOVLW           0x00                    ; This is optional
       MOVWF           IVTBASEU, ACCESS        ; If not included, then the
       MOVLW           0x40                    ; hardware default value of
       MOVWF           IVTBASEH, ACCESS        ; 0x0008 will be taken.
       MOVLW           0x08
       MOVWF           IVTBASEL, ACCESS

       ; TMR0 vector at IVTBASE + 2*(TMR0 vector number i.e. 31) = 0x4046
       MOVLW           0x00                    ; Load TBLPTR with the
       MOVWF           TBLPTRU, ACCESS         ; PFM memory location to be
       MOVLW           0x40                    ; written to.
       MOVWF           TBLPTRH, ACCESS
       MOVLW           0x46
       MOVWF           TBLPTRL, ACCESS

       ; Write the contents of TMR0 vector location
       ; ISR_TMR0_ADDRESS >> 2 = 0x08C0 >> 2 = 0x0230
       MOVLW           0x30                    ; Low byte first
       MOVWF           TABLAT, ACCESS
       TBLWT*+                                 ; Write to temp table latch

       MOVLW           0x02                    ; High byte next
       MOVWF           TABLAT, ACCESS
       TBLWT*+                                 ; Write to temp table latch

       ; Write to PFM now using NVMCON
       BANKSEL         NVMCON1                 ; Select bank for NVMCON1
       MOVLW           0x84                    ; Setting to write to PFM
       MOVWF           NVMCON1

       MOVLW           0x55                    ; Required unlock sequence
       MOVWF           NVMCON2
       MOVLW           0xAA
       MOVWF           NVMCON2
       BSF             NVMCON1, WR             ; Start writing to PFM

       BTFSC           NVMCON1, WR             ; Wait for write to complete
       GOTO            $-2

       RETURN          1
```

REGISTER 9-28: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
TMR0IP	U1IP	U1EIP	U1TXIP	U1RXIP	I2C1EIP	I2C1IP	I2C1TXIP
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 **TMR0IP:** TMR0 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **U1IP:** UART1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **U1EIP:** UART1 Framing Error Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **U1TXIP:** UART1 Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **U1RXIP:** UART1 Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **I2C1EIP:** I^{2}C1 Error Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **I2C1IP:** I^{2}C1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **I2C1TXIP:** I^{2}C1 Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

11.0 WINDOWED WATCHDOG TIMER (WWDT)

The Watchdog Timer (WDT) is a system timer that generates a Reset if the firmware does not issue a `CLRWDT` instruction within the time-out period. The Watchdog Timer is typically used to recover the system from unexpected events. The Windowed Watchdog Timer (WWDT) differs in that `CLRWDT` instructions are only accepted when they are performed within a specific window during the time-out period.

The WWDT has the following features:

- Selectable clock source
- Multiple operating modes
  - WWDT is always On
  - WWDT is off when in Sleep
  - WWDT is controlled by software
  - WWDT is always Off
- Configurable time-out period is from 1 ms to 256s (nominal)
- Configurable window size from 12.5% to 100% of the time-out period
- Multiple Reset conditions

REGISTER 11-2: WDTCON1: WATCHDOG TIMER CONTROL REGISTER 1

U-0	R/W(3)-q/q(1)	R/W(3)-q/q(1)	R/W(3)-q/q(1)	U-0	R/W(4)-q/q(2)	R/W(4)-q/q(2)	R/W(4)-q/q(2)
—	CS<2:0>			—	WINDOW<2:0>		
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	q = Value depends on condition

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **CS<2:0>:** Watchdog Timer Clock Select bits

111 = Reserved
•
•
•
011 = Reserved
010 = SOSC
001 = MFINTOSC 31.25 kHz
000 = LFINTOSC 31 kHz

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **WINDOW<2:0>:** Watchdog Timer Window Select bits

WINDOW<2:0>	Window delay Percent of time	Window opening Percent of time
111	N/A	100
110	12.5	87.5
101	25	75
100	37.5	62.5
011	50	50
010	62.5	37.5
001	75	25
000	87.5	12.5

Note 1: If WDTCCS <2:0> in CONFIG3H = 111, the Reset value of CS<2:0> is 000.
2: The Reset value of WINDOW<2:0> is determined by the value of WDTCWS<2:0> in the CONFIG3H register.
3: If WDTCCS<2:0> in CONFIG3H ≠ 111, these bits are read-only.
4: If WDTCWS<2:0> in CONFIG3H ≠ 111, these bits are read-only.

REGISTER 14-6: CRCACCL: CRC ACCUMULATOR LOW BYTE 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
ACC<7:0>							
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 7-0 **ACC<7:0>**: CRC Accumulator Register bits

REGISTER 14-7: CRCSHIFTH: CRC SHIFT HIGH BYTE REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
SHIFT<15:8>							
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 7-0 **SHIFT<15:8>**: CRC Shifter Register bits
Reading from this register reads the CRC Shifter.

REGISTER 14-8: CRCSHIFTL: CRC SHIFT LOW BYTE REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
SHIFT<7:0>							
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 7-0 **SHIFT<7:0>**: CRC Shifter Register bits
Reading from this register reads the CRC Shifter.

15.5 DMA Message Transfers

Once the Enable bit is set to start DMA message transfers, the Source/Destination pointer and counter registers are initialized to the conditions shown in Table 15-3.

TABLE 15-3: DMA INITIAL CONDITIONS

Register	Value loaded
DMAxSPTR<21:0>	DMAxSSA<21:0>
DMAxSCNT<11:0>	DMAxSSZ<11:0>
DMAxDPTR<15:0>	DMAxDSA<15:0>
DMAxDCNT<11:0>	DMAxDSZ<11:0>

During the DMA Operation after each transaction, Table 15-4 and Table 15-5 indicate how the Source/ Destination pointer and counter registers are modified.

TABLE 15-4: DMA SOURCE POINTER/COUNTER DURING OPERATION

Register	Modified Source Counter/Pointer Value
DMAxSCNT<11:0> != 1	DMAxSCNT = DMAxSCNT -1
	SMODE = 00: DMAxSPTR = DMAxSPTR
	SMODE = 01: DMAxSPTR = DMAxSPTR + 1
	SMODE = 10: DMAxSPTR = DMAxSPTR - 1
DMAxSCNT<11:0> == 1	DMAxSCNT = DMAxSSZ
	DMAxSPTR = DMAxSSA

TABLE 15-5: DMA DESTINATION POINTER/COUNTER DURING OPERATION

Register	Modified Destination Counter/Pointer Value
DMAxDCNT<11:0>!= 1	DMAxDCNT = DMAxDCNT -1
	DMODE = 00: DMAxDPTR = DMAxDPTR
	DMODE = 01: DMAxDPTR = DMAxDPTR + 1
	DMODE = 10: DMAxDPTR = DMAxDPTR - 1
DMAxDCNT<11:0> == 1	DMAxDCNT = DMAxDSZ
	DMAxDPTR = DMAxDSA

The following sections discuss how to initiate and terminate DMA transfers.

15.5.1 STARTING DMA MESSAGE TRANSFERS

The DMA can initiate data transactions by either of the following two conditions:

1. User software control
2. Hardware trigger, SIRQ

15.5.1.1 User Software Control

Software starts or stops DMA transaction by setting/ clearing the DGO bit. The DGO bit is also used to indicate whether a DMA hardware trigger has been received and a message is in progress.

Note 1: Software start can only occur if the EN bit (DMAxCON1) is set.

2: If the CPU writes to the DGO bit while it is already set, there is no effect on the system, the DMA will continue to operate normally.

15.5.3.1 Clearing the SIRQEN bit

Clearing the SIRQEN bit (DMAxCON1 register) stops the sampling of external start interrupt triggers, hence preventing further DMA Message transfers.

An example would be a communications peripheral with a level-triggered interrupt. The peripheral will continue to request data (because its buffer is empty) even though there is no more data to be moved. Disabling the SIRQEN bit prevents the DMA from processing these requests.

15.5.3.2 Source/Destination Stop

The SSTP and DSTP bits (DMAxCON0 register) determine whether or not to disable the hardware triggers (SIRQEN = 0) once a DMA message has completed.

When the SSTP bit is set and the DMAxSCNT = 0, then the SIRQEN bit will be cleared. Similarly, when the DSTP bit is set and the DMAxDCNT = 0, the SIRQEN bit will be cleared.

Note: The SSTP and DSTP bits are independent functions and do not depend on each other. It is possible for a message to be stopped by either counter at message end or both counters at message end.

15.6 Types of Hardware Triggers

The DMA has two different trigger inputs namely the Source trigger and the abort trigger. Each of these trigger sources is user configurable using the DMAxSIRQ and DMAxAIRQ registers.

Based on the source selected for each trigger, there are two types of requests that can be sent to the DMA.

- Edge triggers
- Level triggers

15.6.1 EDGE TRIGGER REQUESTS

An Edge request occurs only once when a given module interrupt requirements are true.

15.6.2 LEVEL TRIGGER REQUESTS

A level request is asserted as long as the condition that causes the interrupt is true.

15.7 Types of Data Transfers

Based on the memory access capabilities of the DMA (See Table 15-1), the following sections discuss the different types of data movement between the Source and Destination Memory regions.

- N: 1

This type of transfer is common when sending predefined data packets (such as strings) through a single interface point (such as communications modules transmit registers).

- N: N

This type of transfer is useful for moving information out of the Program Flash or Data EEPROM to SRAM for manipulation by the CPU or other peripherals.

- 1: N

This type of transfer is common when bridging two different modules data streams together (communications bridge).

- 1: N

This type of transfer is useful for moving information from a single data source into a memory buffer (communications receive registers).

15.8 DMA Interrupts

Each DMA has its own set of four interrupt flags, used to indicate a range of conditions during data transfers. The interrupt flag bits can be accessed using the corresponding PIR registers (Refer to the Interrupt Section).

15.8.1 DMA SOURCE COUNT INTERRUPT

The DMAxSCNTIF source count interrupt flag is set every time the DMAxSCNT<11:0> reaches zero and is reloaded to its starting value.

15.8.2 DMA DESTINATION COUNT INTERRUPT

The DMAxDCNTIF destination count interrupt flag is set every time the DMAxDCNT<11:0> reaches zero and is reloaded to its starting value.

The DMA Source Count zero and Destination Count zero interrupts are used in conjunction to determine when to signal the CPU when the DMA Messages are completed.

15.8.3 ABORT INTERRUPT

The DMAxAIF abort interrupt flag is used to signal that the DMA has halted activity due to an abort signal from one of the abort sources. This is used to indicate that the transaction has been halted for some reason.

15.8.4 OVERRUN INTERRUPT

When the DMA receives a trigger to start a new message before the current message is completed, then the DMAxORIF Overrun interrupt flag is set.

This condition indicates that the DMA is being requested before its current transaction is finished. This implies that the active DMA may not be able to keep up with the demands from the peripheral module being serviced, which may result in data loss.

The DMAxORIF flag being set does not cause the current DMA transfer to terminate.

The Overrun interrupt is only available for trigger sources that are edge based and not available for sources that are level-based. Therefore a level-based interrupt source does not trigger a DMA overrun error due to the potential latency issues in the system.

An example of an interrupt that could use the overrun interrupt would be a timer overflow (or period match) interrupt. This event only happens every time the timer rolls over and is not dependent on any other system conditions.

An example of an interrupt that does not allow the overrun interrupt would be the UARTTX buffer. The UART will continue to assert the interrupt until the DMA is able to process the MSG. Due to latency issues, the DMA may not be able to service an empty buffer immediately, but the UART continues to assert its transmit interrupt until it is serviced. If overrun was allowed in this case, the overrun would occur almost immediately as the module samples the interrupt sources every instruction cycle.

15.9 DMA Setup and Operation

The following steps illustrate how to configure the DMA for data transfer:

1. Program the appropriate Source and Destination addresses for the transaction into the DMAxSSA and DMAxDSA registers
2. Select the source memory region that is being addressed by DMAxSSA register, using the SMR<1:0> bits.
3. Program the SMODE and DMODE bits to select the addressing mode.
4. Program the Source size DMAxSSZ and Destination size DMAxDSZ registers with the number of bytes to be transferred. It is recommended for proper operation that the size registers be a multiple of each other.
5. If the user desires to disable data transfers once the message has completed, then the SSTP and DSTP bits in DMAxCON0 register need to be set. (see **Section 15.5.3.2 "Source/Destination Stop"**).
6. If using hardware triggers for data transfer, setup the hardware trigger interrupt sources for the starting and aborting DMA transfers (DMAxSIRQ and DMAxAIRQ), and set the corresponding interrupt request enable bits (SIRQEN and AIRQEN).
7. Select the priority level for the DMA (see **Section 3.1 "System Arbitration"**) and lock the priorities (see **Section 3.1.1 "Priority Lock"**)
8. Enable the DMA (DMAxCON1bits. EN = `1`)
9. If using software control for data transfer, set the DGO bit, else this bit will be set by the hardware trigger.

Once the DMA is set up, the following flow chart describes the sequence of operation when the DMA uses hardware triggers and utilizes the unused CPU cycles (bubble) for DMA transfers.

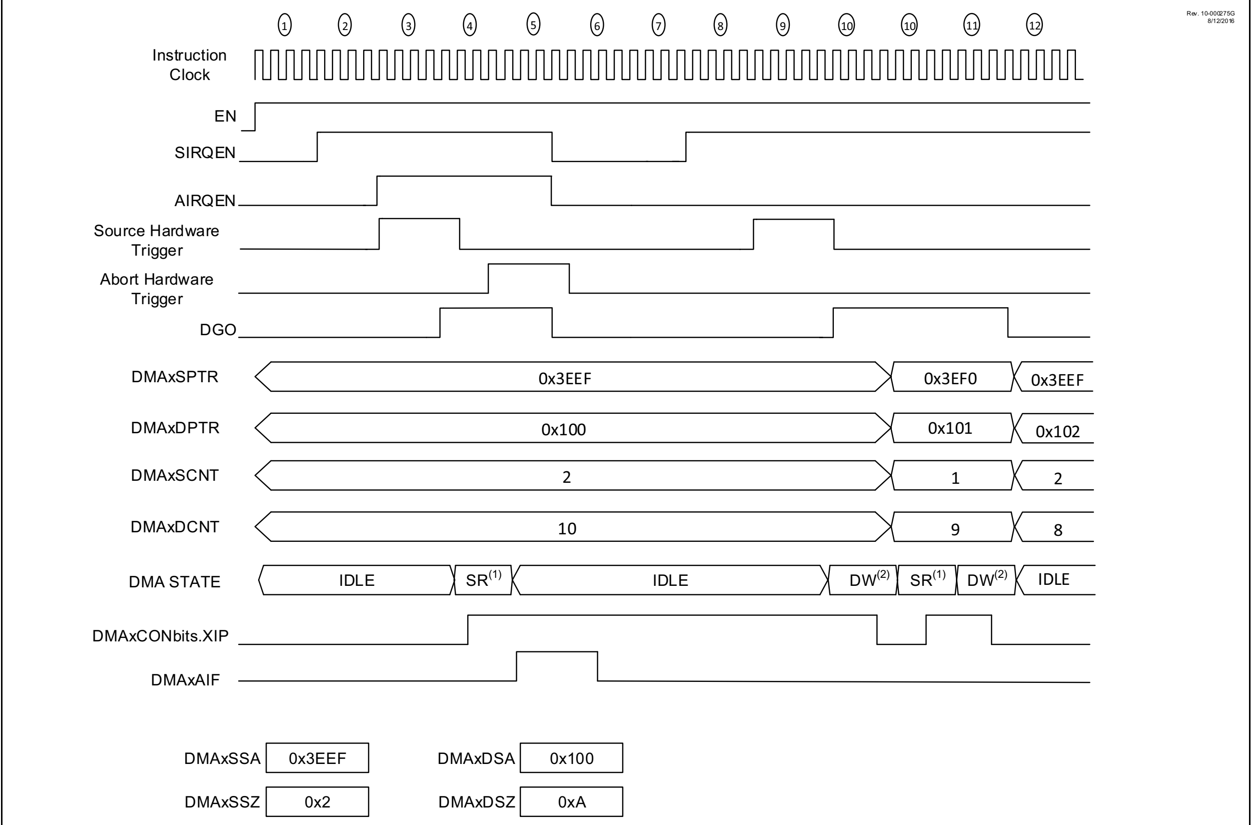
15.9.7 ABORT TRIGGER, MESSAGE IN PROGRESS

When an abort interrupt request is received in a DMA transaction, the DMA will perform a soft-stop by clearing the DGO (i.e., if the DMA was reading the source register, it will complete the read operation and then clear the DGO bit).

The SIREQEN bit is cleared to prevent any overrun and the AIRQEN bit is cleared to prevent any false aborts.

When the DGO bit is set again the DMA will resume operation from where it left off after the soft-stop.

FIGURE 15-11: ABORT DURING MESSAGE TRANSFER



Note 1: SR - Source Read

2: DW - Destination Write

The following table contains some of the cases in which the DMA module can be configured to.

22.5.6 EDGE-TRIGGERED ONE-SHOT MODE

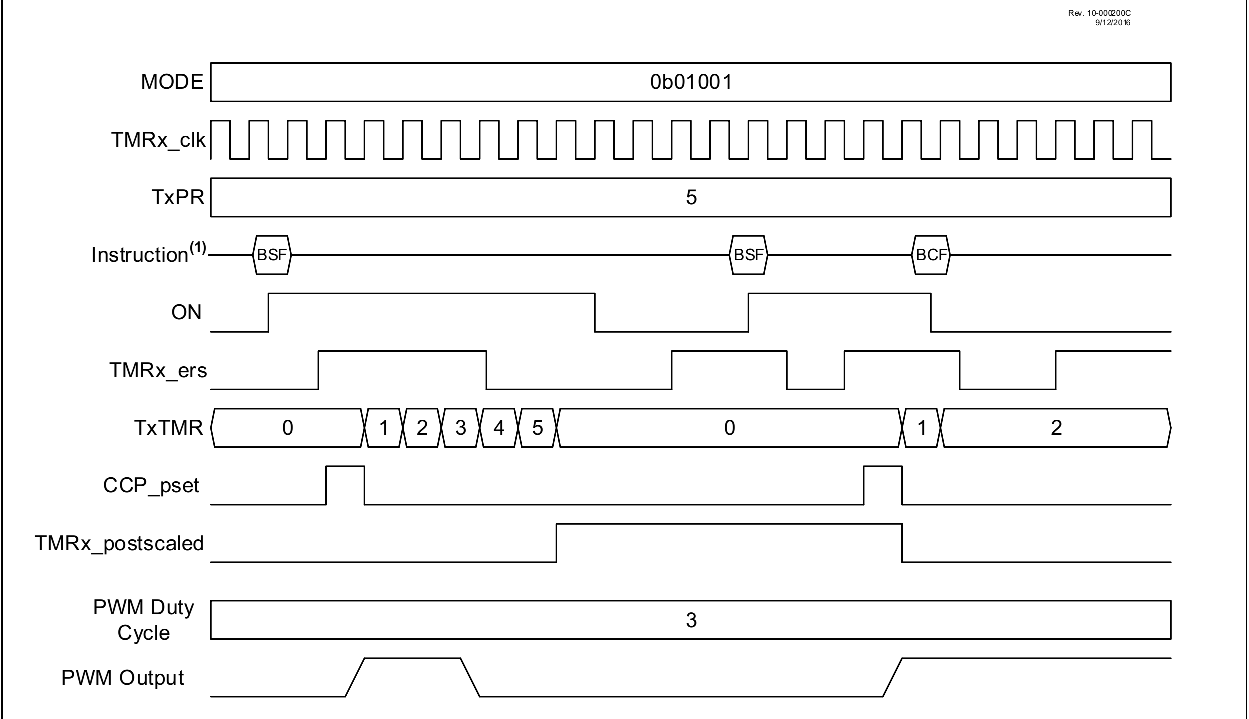
The Edge-Triggered One-Shot modes start the timer on an edge from the external signal input, after the ON bit is set, and clear the ON bit when the timer matches the T2PR period value. The following edges will start the timer:

- Rising edge (MODE<4:0> = `01001`)
- Falling edge (MODE<4:0> = `01010`)
- Rising or Falling edge (MODE<4:0>=‘`01011`’)

If the timer is halted by clearing the ON bit then another TMRx_ers edge is required after the ON bit is set to resume counting. Figure 22-9 illustrates operation in the rising edge One-Shot mode.

When Edge-Triggered One-Shot mode is used in conjunction with the CCP then the edge-trigger will activate the PWM drive and the PWM drive will deactivate when the timer matches the CCPRx pulse width value and stay deactivated when the timer halts at the T2PR period count match.

FIGURE 22-9: EDGE TRIGGERED ONE-SHOT MODE TIMING DIAGRAM (MODE = `01001`)



Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

REGISTER 27-8: 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 (noninverted) 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 (noninverted) 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 (noninverted) 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 (noninverted) 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

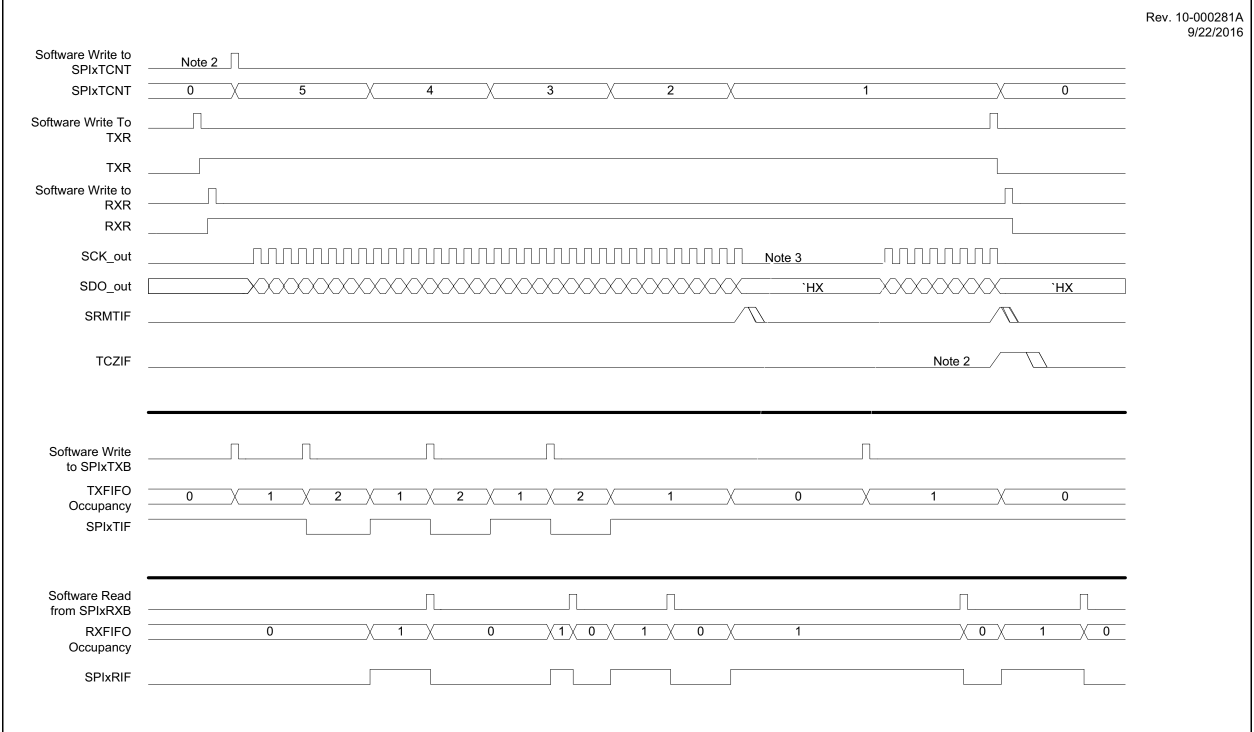
32.5.1 FULL DUPLEX MODE

When both TXR and RXR are set, the SPI master is in Full Duplex mode. In this mode, data transfer triggering is affected by the BMODE bit of SPIxCON0.

When BMODE = `1`, data transfers will occur whenever both the RXFIFO is not full and there is data present in the TXFIFO. In practice, as long as the RXFIFO is not full, data will be transmitted/received as soon as the SPIxTxB register is written to, matching functionality of SPI (MSSP) modules on older 8-bit Microchip devices. The SPIxTCNT will decrement with each transfer. However, when SPIxTCNT is zero the next transfer is not inhibited and the corresponding SPIxTCNT decrement will cause the count to roll over to the maximum value. Figure 32-3 shows an example of a communication using this mode.

When BMODE = `0`, the transfer counter (SPIxTCNTH/SPIxTCNTL) must also be written to before transfers will occur, and transfers will cease when the transfer counter reaches '`0`'. For example, if SPIxTXB is written twice and then SPIxTCTL is written with '3' then the transfer will start with the SPIxTCTL write. The two bytes in the TXFIFO will be sent after which the transfer will suspend until the third and last byte is written to SPIxTXB.

FIGURE 32-3: SPI MASTER OPERATION – DATA EXCHANGE, TXR/RXR = `1/1`



Note: 1. SS(out) is not shown on this diagram
2. SPIxTCNT write is optional TXR/RXR = 1/1 and BMODE=1. If BMODE=0, a write to SPIxTCNT is required to start transmission; TCZIF signals the end of the transmission.
3. Transmission gap occurs while waiting for transmitter data.

32.8.3.4 Receiver Overflow and Transmitter Underflow Interrupts

The receiver overflow interrupt triggers if data is received when the RXFIFO is already full and RXR = `1`. In this case, the data will be discarded and the RXOIF bit will be set. The receiver overflow interrupt flag is the RXOIF bit of SPIxINTF. The receiver overflow interrupt enable bit is the RXOIE bit of SPIxINTE.

The Transmitter Underflow interrupt flag triggers if a data transfer begins when the TXFIFO is empty and TXR = `1`. In this case, the most recently received data will be transmitted and the TXUIF bit will be set. The transmitter underflow interrupt flag is the TXUIF bit of SPIxINTF. The transmitter underflow interrupt enable bit is the TXUIE bit of SPIxINTE.

Both of these interrupts will only occur in Slave mode, as Master mode will not allow the RXFIFO to overflow or the TXFIFO to underflow.

REGISTER 32-9: SPIxCON2: SPI CONFIGURATION REGISTER 2

R-0/0	R-0/0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
BUSY	SSFLT	—	—	—	SSET	TXR[(1)]	RXR[(1)]
bit 7							bit 0

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

bit 7 **BUSY**: SPI Module Busy Status bit
1 = Data exchange is busy
0 = Data exchange is not taking place

bit 6 **SSFLT**: SS(in) Fault Status bit
If SSET = 0
1 = SS(in) ended the transaction unexpectedly, and the data byte being received was lost
0 = SS(in) ended normally
If SSET = 1
This bit is unchanged.

bit 5-3 **Unimplemented**: Read as ‘0’

bit 2 **SSET**: Slave Select Enable bit
Master mode:
1 = SS(out) is driven to the active state continuously
0 = SS(out) is driven to the active state while the transmit counter is not zero
Slave mode:
1 = SS(in) is ignored and data is clocked on all SCK(in) (as though SS = TRUE at all times)
0 = SS(in) enables/disables data input and tri-states SDO if the TRIS bit associated with the SDO pin is set (see Table 32-2 for details)

bit 1 **TXR**: Transmit Data-Required Control bit[(1)]
1 = TxFIFO data is required for a transfer
0 = TxFIFO data is not required for a transfer

bit 0 **RXR**: Receive FIFO Space-Required Control bit[(1)]
1 = Data transfers are suspended if the RxFIFO is full
0 = Received data is not stored in the FIFO

Note 1: See Table 32-1 as well as **Section 32.5 “Master mode”** and **Section 32.6 “Slave Mode”** for more details pertaining to TXR and RXR function.

2: This register should not be written to while a transfer is in progress (BUSY bit of SPIxCON2 is set).

REGISTER 33-17: I2CxADB1: I^{2}C ADDRESS DATA BUFFER 1 REGISTER(1)

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
ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
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 HC = Hardware clear

bit 7-0 MODE<2:0> = 00x
Unused in this mode; bit state is a don’t care
MODE<2:0> = 01x
ADB<7:1>: 10-bit Address High byte
Received matching 10-bit high address data
R/$\overline{\text{W}}$: Read/not-Write Data bit
Received read/write value from matching 10-bit high address
MODE<2:0> = 100
ADB<7:1>: Address Data byte
7-bit address value copied to transmit shift register
R/$\overline{\text{W}}$: Read/not-Write Data bit
Read/write value copied to transmit shift register
Master hardware uses this bit to produce read versus write operations.
MODE<2:0> = 101
ADB<7:1>: 10-bit Address High Data byte
10-bit high address value copied to transmit shift register
R/$\overline{\text{W}}$: Read/not-Write Data bit
Read/write value copied to transmit shift register
Master hardware uses this bit to produce read versus write operations.
MODE<2:0> = 11x
ADB<7:1>: Address Data byte
7-bit address value copied to transmit shift register
R/$\overline{\text{W}}$: Read/not-Write Data bit
Read/write value copied to transmit shift register
Master hardware uses this bit to produce read versus write operations

Note 1: This register is read only in slave, 7-bit Addressing modes (MODE<2:0> = 0xx)

REGISTER 36-8: ADPCH: ADC POSITIVE CHANNEL SELECTION REGISTER

U-0	U-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
—	—	ADPCH<5:0>					
bit 7							bit 0

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

bit 7-6 **Unimplemented**: Read as ‘0’

bit 5-0 **ADPCH<5:0>**: ADC Positive Input Channel Selection bits

111111 = FVR buffer 2[2]
111110 = FVR buffer 1[2]
111101 = DAC1 output[1]
111100 = Temperature indicator[3]
111011 = Vss (Analog Ground)
111010 = Reserved. No channel connected.
•
•
•
110000 = Reserved. No channel connected.
101111 = ANF7[4]
101110 = ANF6[4]
101101 = ANF5[4]
101100 = ANF4[4]
101011 = ANF3[4]
101010 = ANF2[4]
101001 = ANF1[4]
101000 = ANF0[4]
100111 = Reserved. No channel connected.
•
•
100011 = Reserved. No channel connected.
100010 = ANE2[5]
100001 = ANE1[5]
100000 = ANE0[5]
011111 = AND7[5]
011110 = AND6[5]
011101 = AND5[5]
011100 = AND4[5]
011011 = AND3[5]
011010 = AND2[5]
011001 = AND1[5]
011000 = AND0[5]
010111 = ANC7
010110 = ANC6
010101 = ANC5
010100 = ANC4
010011 = ANC3
010010 = ANC2
010001 = ANC1
010000 = ANC0
001111 = ANB7
001110 = ANB6
001101 = ANB5
001100 = ANB4
001011 = ANB3
001010 = ANB2
001001 = ANB1
001000 = ANB0
000111 = ANA7
000110 = ANA6
000101 = ANA5
000100 = ANA4
000011 = ANA3
000010 = ANA2
000001 = ANA1
000000 = ANA0

Note 1: See **Section 37.0 “5-Bit Digital-to-Analog Converter (DAC) Module”** for more information.
2: See **Section 34.0 “Fixed Voltage Reference (FVR)”** for more information.
3: See **Section 35.0 “Temperature Indicator Module”** for more information.
4: Reserved on PIC18(L)F26/27/45/46/47K42 parts.
5: Reserved on PIC18(L)F26/27K42 parts.

INCF — Increment f

Syntax:	INCF f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f) + 1 → dest
Status Affected:	C, DC, N, OV, Z
Encoding:	`0010` `10da` `ffff` `ffff`
Description:	The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back 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 41.2.3 "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: `INCF CNT, 1, 0`

Before Instruction

CNT	=	FFh
Z	=	0
C	=	?
DC	=	?

After Instruction

CNT	=	00h
Z	=	1
C	=	1
DC	=	1

INCFSZ — Increment f, skip if 0

Syntax:	INCFSZ f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f) + 1 → dest, skip if result = 0
Status Affected:	None
Encoding:	`0011` `11da` `ffff` `ffff`
Description:	The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is '0', the next instruction, which is already fetched, is discarded and a `NOP` is executed instead, making it a 2-cycle instruction. 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 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

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

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE    INCFSZ   CNT, 1, 0
NZERO   :
ZERO    :
```

Before Instruction

PC	=	Address `(HERE)`

After Instruction

CNT	=	CNT + 1
If CNT	=	0;
PC	=	Address `(ZERO)`
If CNT	≠	0;
PC	=	Address `(NZERO)`

MOVF — Move f

Syntax:	MOVF f {,d {,a}}
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	$f \rightarrow$ dest
Status Affected:	N, Z

Encoding:

`0101`	`00da`	`ffff`	`ffff`

Description: The contents of register 'f' are moved to a destination dependent upon the status of 'd'. If 'd' is '`0`', the result is placed in W. If 'd' is '`1`', the result is placed back in register 'f' (default). 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 $f \leq 95$ (5Fh). See **Section 41.2.3 "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 W

Example: `MOVF   REG, 0, 0`

Before Instruction

REG	=	22h
W	=	FFh

After Instruction

REG	=	22h
W	=	22h

MOVFF — Move f to f

Syntax:	MOVFF f_s,f_d
Operands:	$0 \leq f_s \leq 4095$ $0 \leq f_d \leq 4095$
Operation:	$(f_s) \rightarrow f_d$
Status Affected:	None

Encoding:

1st word (source)	`1100`	`ffff`	`ffff`	`ffff`$_s$
2nd word (destin.)	`1111`	`ffff`	`ffff`	`ffff`$_d$

Description: The contents of source register 'f_s' are moved to destination register 'f_d'. Location of source 'f_s' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination 'f_d' can also be anywhere from 000h to FFFh.
`MOVFF` has curtailed the source and destination range to the lower 4 Kbyte space of memory (Banks 1 through 15). For everything else, use `MOVFFL`.

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

Example: `MOVFF   REG1, REG2`

Before Instruction

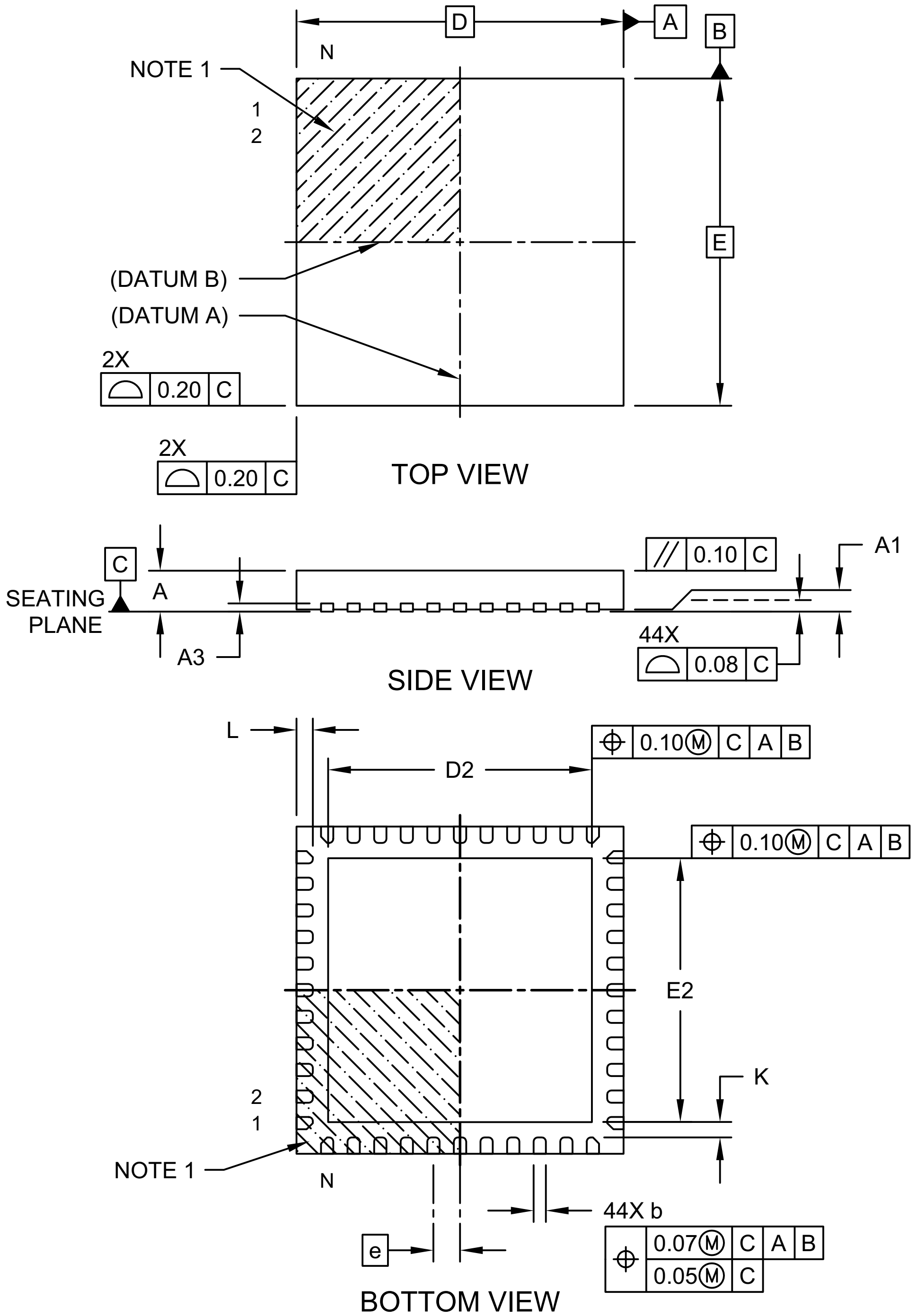
REG1	=	33h
REG2	=	11h

After Instruction

REG1	=	33h
REG2	=	33h

44-Lead Plastic Quad Flat, No Lead Package (ML) - 8x8 mm Body [QFN or VQFN]

Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-103D Sheet 1 of 2