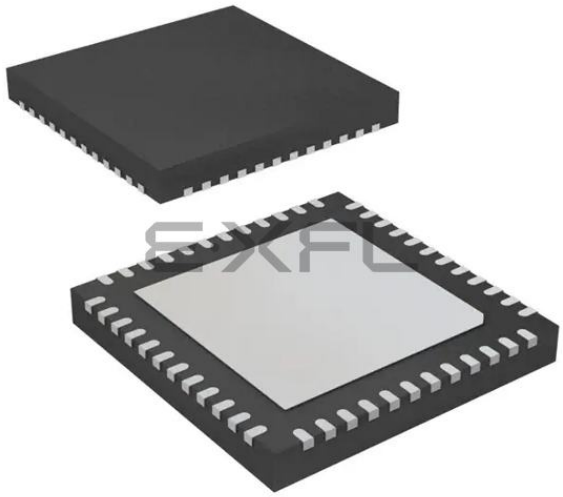


Welcome to E-XFL.COM

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, LVD, POR, PWM, WDT
Number of I/O	44
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)	2.3V ~ 5.5V
Data Converters	A/D 43x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	48-UFQFN Exposed Pad
Supplier Device Package	48-UQFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f56k42-i-mv

EXAMPLE 4-4: TWO-WORD INSTRUCTIONS

CASE 1:				
Object Code	**Source Code**			
0110 0110 0000 0000	TSTFSZ	REG1	;	is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	;	Yes, skip this word
1111 0100 0101 0110			;	Execute this word as a NOP
0010 0100 0000 0000	ADDWF	REG3	;	continue code
CASE 2:				
Object Code	**Source Code**			
0110 0110 0000 0000	TSTFSZ	REG1	;	is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	;	No, execute this word
1111 0100 0101 0110			;	2nd word of instruction
0010 0100 0000 0000	ADDWF	REG3	;	continue code

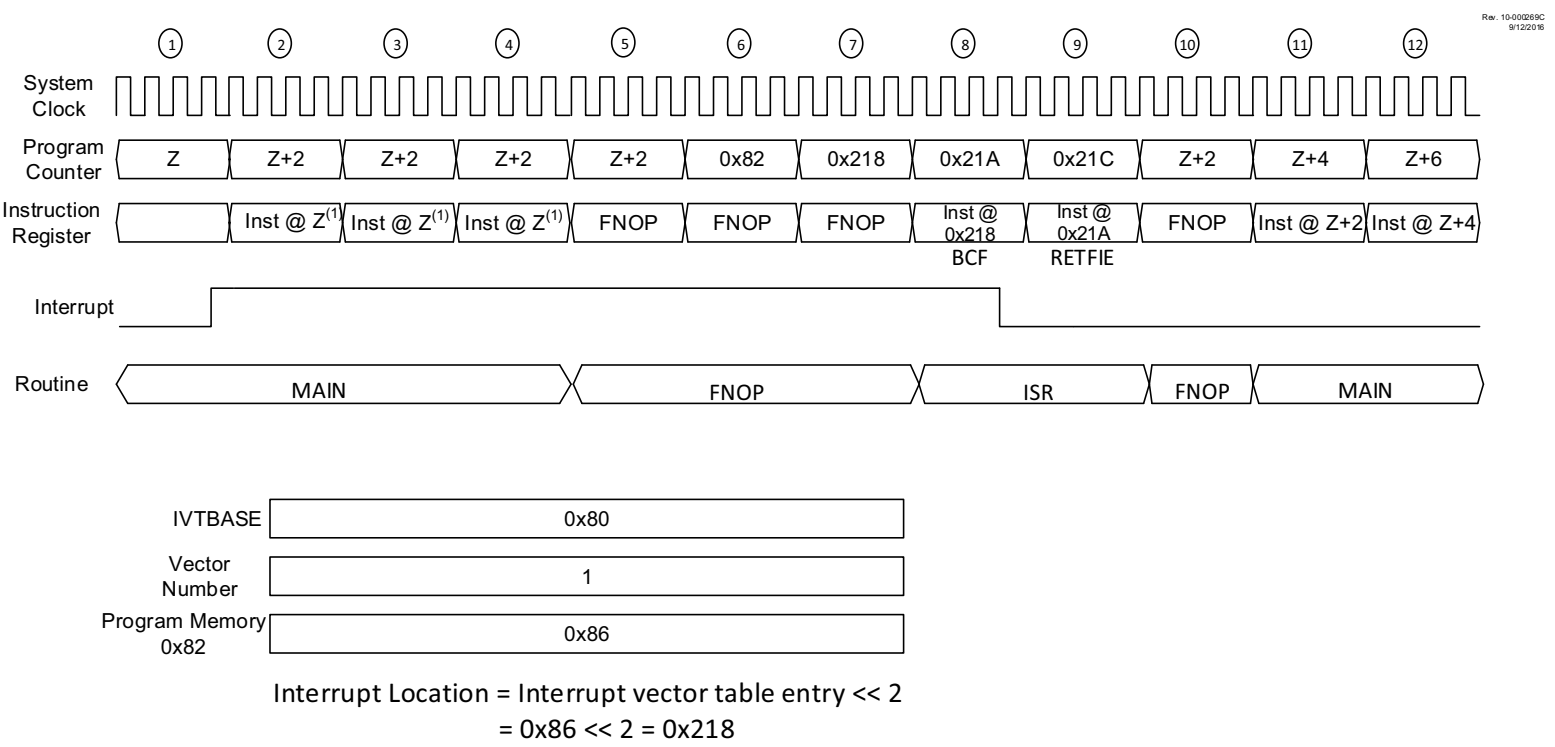
EXAMPLE 4-5: THREE-WORD INSTRUCTIONS

CASE 1:				
Object Code	**Source Code**			
0110 0110 0000 0000	TSTFSZ	REG1	;	is RAM location 0?
0000 0000 0110 0000	MOVFFL	REG1, REG2	;	Yes, skip this word
1111 0100 1000 1100			;	Execute this word as a NOP
1111 0100 0101 0110			;	Execute this word as a NOP
0010 0100 0000 0000	ADDWF	REG3	;	continue code
CASE 2:				
Object Code	**Source Code**			
0110 0110 0000 0000	TSTFSZ	REG1	;	is RAM location 0?
0000 0000 0110 0000	MOVFFL	REG1, REG2	;	No, execute this word
1111 0100 1000 1100			;	2nd word of instruction
1111 0100 0101 0110			;	3rd word of instruction
0010 0100 0000 0000	ADDWF	REG3	;	continue code

FIGURE 4-4: DATA MEMORY MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

Bank	BSR<5:0>	Address addr<7:0>	PIC18(L)F45K42 PIC18(L)F55K42	PIC18(L)F26K42 PIC18(L)F46K42 PIC18(L)F56K42	PIC18(L)F27K42 PIC18(L)F47K42 PIC18(L)F57K42	Address addr<13:0>
Bank 0	00 0000	00h	Access RAM	Access RAM	Access RAM	0000h 005Fh
		FFh	GPR	GPR	GPR	0060h 00FFh
Bank 1	00 0001	00h FFh	GPR	GPR	GPR	0100h 03FFh
Bank 2	00 0010	00h FFh				
Bank 3	00 0011	00h FFh				
Banks 4 to 7	00 0100 – 00 0111	00h FFh	GPR	GPR	GPR	0400h 07FFh
Banks 8 to 15	00 1000 – 00 1111	00h FFh	Unimplemented	GPR	GPR	0800h 0FFFh
Banks 16 to 31	01 0000 – 01 1111	00h FFh		Unimplemented		1000h 1FFFh
Banks 32 to 55	10 0000 – 11 0111	00h FFh			Unimplemented	2000h 37FFh
Banks 56 to 62	11 1000 – 11 1110	00h FFh	SFR	SFR	SFR	3800h 3EFFh
Bank 63	11 1111	00h FFh	SFR	SFR	SFR	3800h 3EFFh 3F60h 3FFFh

FIGURE 9-9: INTERRUPT TIMING DIAGRAM - THREE CYCLE INSTRUCTION



Note 1: Instruction @ Z is a Three-cycle instruction.

REGISTER 9-17: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3

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
TMR0IE	U1IE	U1EIE	U1TXIE	U1RXIE	I2C1EIE	I2C1IE	I2C1TXIE
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 **TMR0IE:** TMR0 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **U1IE:** UART1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5 **U1EIE:** UART1 Framing Error Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 4 **U1TXIE:** UART1 Transmit Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **U1RXIE:** UART1 Receive Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **I2C1EIE:** I^2C1 Error Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **I2C1IE:** I^2C1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **I2C1TXIE:** I^2C1 Transmit Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 9-22: PIE8: PERIPHERAL INTERRUPT ENABLE REGISTER 8

R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0	U-0	U-0
TMR5GIE	TMR5IE	—	—	—	—	—	—
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 **TMR5GIE:** TMR5 Gate Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **TMR5IE:** TMR5 Interrupt Enable bit
1 = Enabled
0 = Disabled

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

REGISTER 9-23: PIE9: PERIPHERAL INTERRUPT ENABLE REGISTER 9

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	CLC3IE	CWG3IE	CCP3IE	TMR6IE
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 7-4 **Unimplemented:** Read as ‘0’

bit 3 **CLC3IE:** CLC3 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **CWG3IE:** CWG3 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **CCP3IE:** CCP3 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **TMR6IE:** TMR6 Interrupt Enable bit
1 = Enabled
0 = Disabled

EXAMPLE 13-4: WRITING TO PROGRAM FLASH MEMORY

```
                MOVLW     D'64'                 ; number of bytes in erase block
                MOVWF     COUNTER
                MOVLW     BUFFER_ADDR_HIGH      ; point to buffer
                MOVWF     FSR0H
                MOVLW     BUFFER_ADDR_LOW
                MOVWF     FSR0L
                MOVLW     CODE_ADDR_UPPER       ; Load TBLPTR with the base
                MOVWF     TBLPTRU               ; address of the memory block
                MOVLW     CODE_ADDR_HIGH
                MOVWF     TBLPTRH
                MOVLW     CODE_ADDR_LOW
                MOVWF     TBLPTRL
READ_BLOCK
                TBLRD*+                         ; read into TABLAT, and inc
                MOVF      TABLAT, W             ; get data
                MOVWF     POSTINC0              ; store data
                DECFSZ    COUNTER               ; done?
                BRA       READ_BLOCK            ; repeat
MODIFY_WORD
                MOVLW     BUFFER_ADDR_HIGH      ; point to buffer
                MOVWF     FSR0H
                MOVLW     BUFFER_ADDR_LOW
                MOVWF     FSR0L
                MOVLW     NEW_DATA_LOW          ; update buffer word
                MOVWF     POSTINC0
                MOVLW     NEW_DATA_HIGH
                MOVWF     INDF0
ERASE_BLOCK
                MOVLW     CODE_ADDR_UPPER       ; load TBLPTR with the base
                MOVWF     TBLPTRU               ; address of the memory block
                MOVLW     CODE_ADDR_HIGH
                MOVWF     TBLPTRH
                MOVLW     CODE_ADDR_LOW
                MOVWF     TBLPTRL
                BCF       NVMCON1, REG0         ; point to Program Flash Memory
                BSF       NVMCON1, REG1         ; point to Program Flash Memory
                BSF       NVMCON1, WREN         ; enable write to memory
                BSF       NVMCON1, FREE         ; enable Erase operation
                BCF       INTCON0, GIE          ; disable interrupts
                MOVLW     55h
Required        MOVWF     NVMCON2               ; write 55h
Sequence        MOVLW     AAh
                MOVWF     NVMCON2               ; write 0AAh
                BSF       NVMCON1, WR           ; start erase (CPU stall)
                BSF       INTCON0, GIE          ; re-enable interrupts
                TBLRD*-                         ; dummy read decrement
                MOVLW     BUFFER_ADDR_HIGH      ; point to buffer
                MOVWF     FSR0H
                MOVLW     BUFFER_ADDR_LOW
                MOVWF     FSR0L
WRITE_BUFFER_BACK
                MOVLW     BlockSize             ; number of bytes in holding register
                MOVWF     COUNTER
                MOVLW     D'64'/BlockSize       ; number of write blocks in 64 bytes
                MOVWF     COUNTER2
```

REGISTER 15-9: DMAxSPTRU: DMAx SOURCE POINTER UPPER REGISTER

U-0	U-0	R-0	R-0	R-0	R-0	R-0	R-0
—	—	SPTR<21:16>					
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 7-6 **Unimplemented:** Read as '0'

bit 5-0 **SPTR<21:16>:** Current Source Address Pointer

REGISTER 15-10: DMAxSSZL: DMAx SOURCE SIZE 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
SSZ<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 **SSZ<7:0>:** Source Message Size bits

REGISTER 15-11: DMAxSSZH: DMAx SOURCE SIZE HIGH REGISTER

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	SSZ<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 **SSZ<11:8>:** Source Message Size bits

REGISTER 24-2: CCPTMRS1: CCP TIMERS CONTROL REGISTER 1

R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1
P8TSEL<1:0>		P7TSEL<1:0>		P6TSEL<1:0>		P5TSEL<1:0>	
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	x = Bit is unknown

bit 7-6 **P8TSEL<1:0>:** PWM8 Timer Selection bits
11 = PWM8 based on TMR6
10 = PWM8 based on TMR4
01 = PWM8 based on TMR2
00 = Reserved

bit 5-4 **P7TSEL<1:0>:** PWM7 Timer Selection bits
11 = PWM7 based on TMR6
10 = PWM7 based on TMR4
01 = PWM7 based on TMR2
00 = Reserved

bit 3-2 **P6TSEL<1:0>:** PWM6 Timer Selection bits
11 = PWM6 based on TMR6
10 = PWM6 based on TMR4
01 = PWM6 based on TMR2
00 = Reserved

bit 1-0 **P5TSEL<1:0>:** PWM5 Timer Selection bits
11 = PWM5 based on TMR6
10 = PWM5 based on TMR4
01 = PWM5 based on TMR2
00 = Reserved

25.6.8 CAPTURE MODE

This mode captures the Timer value based on a rising or falling edge on the SMTWINx input and triggers an interrupt. This mimics the capture feature of a CCP module. The timer begins incrementing upon the GO bit being set, and updates the value of the SMT1CPR register on each rising edge of SMTWINx, and updates the value of the CPW register on each falling edge of the SMTWINx. The timer is not reset by any hardware conditions in this mode and must be reset by software, if desired. See Figure 25-16 and Figure 25-17.

FIGURE 25-16: CAPTURE MODE REPEAT ACQUISITION TIMING DIAGRAM

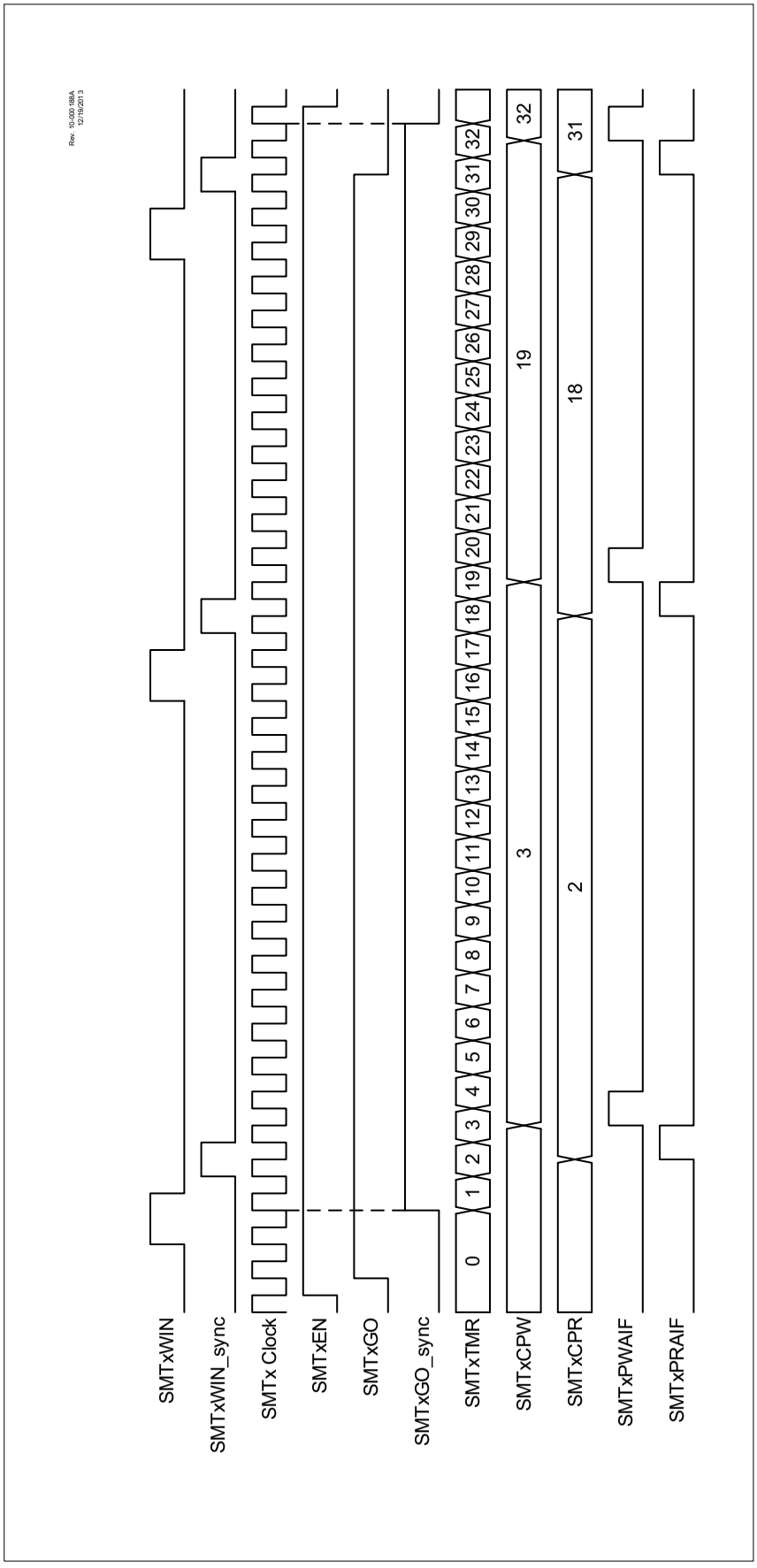
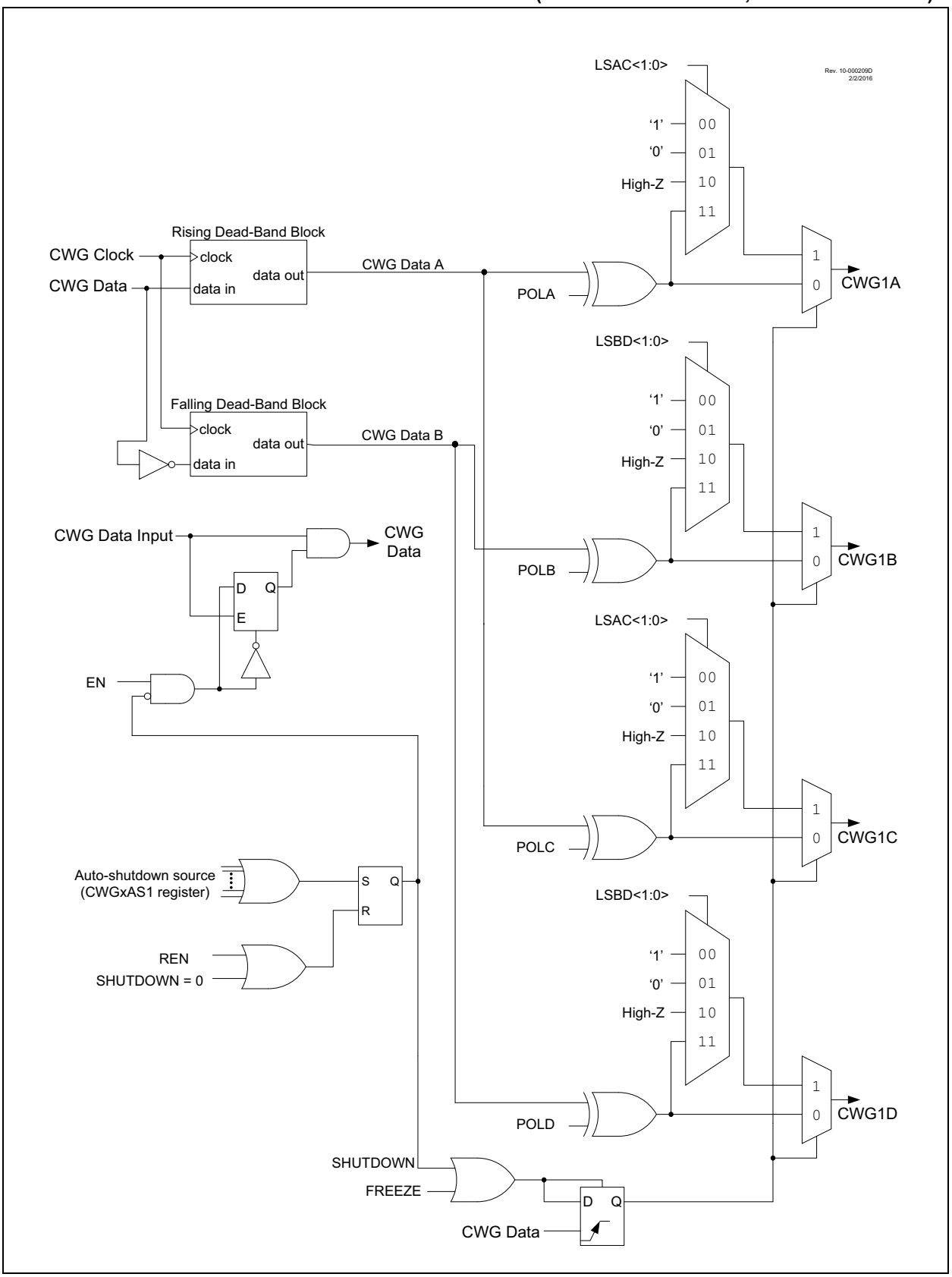


FIGURE 26-1: SIMPLIFIED CWG BLOCK DIAGRAM (HALF-BRIDGE MODE, MODE<2:0> = 100)



REGISTER 30-3: MD1CARH: MODULATION HIGH CARRIER CONTROL REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	CH<4:0>(1)				
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 **Unimplemented:** Read as '0'

bit 4-0 **CH<4:0>:** Modulator Carrier High Selection bits(1)
See Table 30-1 for signal list

Note 1: Unused selections provide an input value.

REGISTER 30-4: MD1CARL: MODULATION LOW CARRIER CONTROL REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	CL<4:0>(1)				
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 **Unimplemented:** Read as '0'

bit 4-0 **CL<4:0>:** Modulator Carrier Low Input Selection bits(1)
See Table 30-1 for signal list

Note 1: Unused selections provide a zero as the input value.

31.0 UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER (UART) WITH PROTOCOL SUPPORT

The Universal Asynchronous Receiver Transmitter (UART) module is a serial I/O communications peripheral. It contains all the clock generators, shift registers and data buffers necessary to perform an input or output serial data transfer, independent of device program execution. The UART, also known as a Serial Communications Interface (SCI), can be configured as a full-duplex asynchronous system or one of several automated protocols. Full-Duplex mode is useful for communications with peripheral systems, such as CRT terminals and personal computers.

Supported protocols include:

- LIN Master and Slave
- DMX mode
- DALI control gear and control device

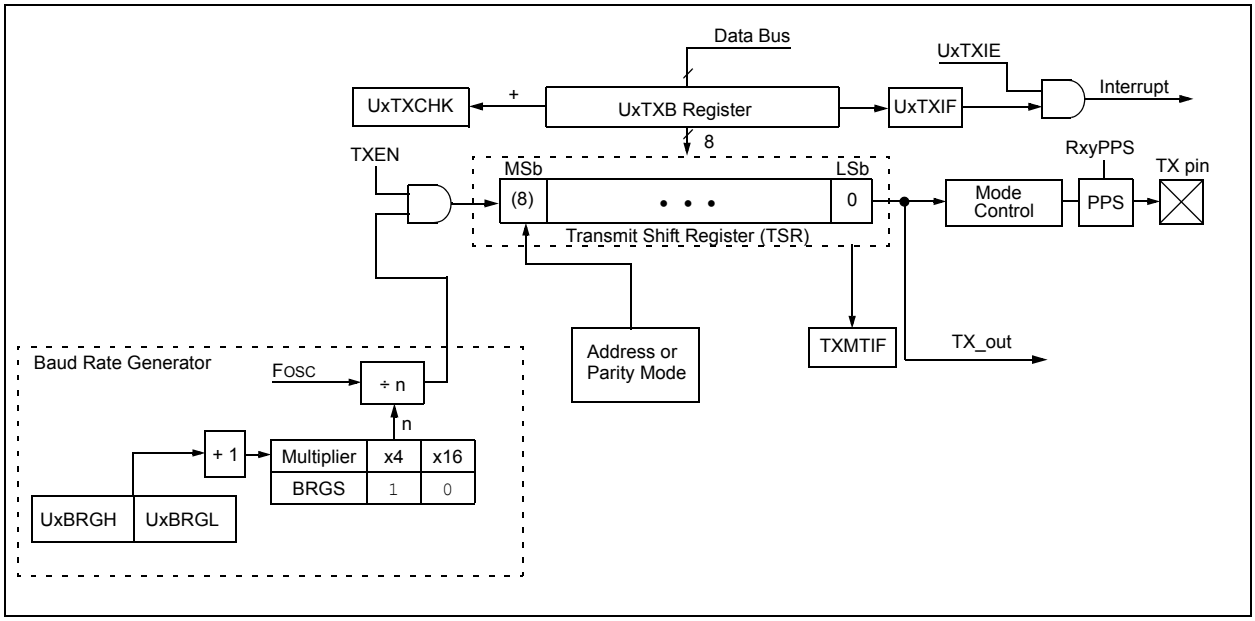
The UART module includes the following capabilities:

- Full-duplex asynchronous transmit and receive
- Two-character input buffer
- One-character output buffer
- Programmable 7-bit or 8-bit character length
- 9th bit Address detection
- 9th bit even or odd parity
- Input buffer overrun error detection
- Received character framing error detection
- Hardware and software flow control
- Automatic checksums
- Programmable 1, 1.5, and 2 Stop bits
- Programmable data polarity
- Manchester encoder/decoder
- Operation in Sleep
- Automatic detection and calibration of the baud rate
- Wake-up on Break reception
- Automatic and user timed Break period generation
- RX and TX inactivity timeouts (with Timer2)

Block diagrams of the UART transmitter and receiver are shown in Figure 31-1 and Figure 31-2.

The UART transmit output (TX_out) is available to the TX pin and internally to various peripherals.

FIGURE 31-1: UART TRANSMIT BLOCK DIAGRAM



REGISTER 32-6: 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 32-7: 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 36-2: ADCON1: ADC CONTROL REGISTER 1

R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0	R/W-0/0
PPOL	IPEN	GPOL	—	—	—	—	DSEN
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 **PPOL:** Precharge Polarity bit
If PRE>0x00:

PPOL	Action During 1st Precharge Stage	
	External (selected analog I/O pin)	Internal (AD sampling capacitor)
1	Connected to V_{DD}	C_{HOLD} connected to V_{SS}
0	Connected to V_{SS}	C_{HOLD} connected to V_{DD}

Otherwise:
The bit is ignored

bit 6 **IPEN:** A/D Inverted Precharge Enable bit
If DSEN = 1
1 = The precharge and guard signals in the second conversion cycle are the opposite polarity of the first cycle
0 = Both Conversion cycles use the precharge and guards specified by ADPPOL and ADGPOL
Otherwise:
The bit is ignored

bit 5 **GPOL:** Guard Ring Polarity Selection bit
1 = ADC guard Ring outputs start as digital high during Precharge stage
0 = ADC guard Ring outputs start as digital low during Precharge stage

bit 4-1 **Unimplemented:** Read as '0'

bit 0 **DSEN:** Double-sample enable bit
1 = Two conversions are performed on each trigger. Data from the first conversion appears in PREV
0 = One conversion is performed for each trigger

REGISTER 36-13: ADCAP: ADC ADDITIONAL SAMPLE CAPACITOR SELECTION REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	ADCAP<4:0>				
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 **Unimplemented**: Read as ‘0’

bit 4-0 **ADCAP<4:0>**: ADC Additional Sample Capacitor Selection bits
11111 = 31 pF
11110 = 30 pF
11101 = 29 pF
•
•
•
00011 = 3 pF
00010 = 2 pF
00001 = 1 pF
00000 = No additional capacitance

REGISTER 36-14: ADRPT: ADC REPEAT SETTING 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
RPT<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 **RPT<7:0>**: ADC Repeat Threshold bits
Determines the number of times that the ADC is triggered before the threshold is checked when the computation is Low-pass Filter, Burst Average, or Average modes. See Table 36-2 for more details.

TABLE 41-2: INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
LITERAL INSTRUCTIONS									
ADDLW	k	Add literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW	k	AND literal with WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW	k	Inclusive OR literal with WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR	f_n, k	Load FSR(f_n) with a 14-bit literal (k)	2	1110	1110	00ff	kkkk	None	
				1111	00kk	kkkk	kkkk		
ADDFSR	f_n, k	Add FSR(f_n) with (k)	1	1110	1000	ffkk	kkkk	None	
SUBFSR	f_n, k	Subtract (k) from FSR(f_n)	1	1110	1001	ffkk	kkkk	None	
MOVLB	k	Move literal to BSR<5:0>	1	0000	0001	00kk	kkkk	None	
MOVLW	k	Move literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW	k	Multiply literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW	k	Subtract WREG from literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	
DATA MEMORY – PROGRAM MEMORY INSTRUCTIONS									
TBLRD*		Table Read	2 - 5	0000	0000	0000	1000	None	
TBLRD*+		Table Read with post-increment		0000	0000	0000	1001	None	
TBLRD*-		Table Read with post-decrement		0000	0000	0000	1010	None	
TBLRD+*		Table Read with pre-increment		0000	0000	0000	1011	None	
TBLWT*		Table Write	2 - 5	0000	0000	0000	1100	None	
TBLWT*+		Table Write with post-increment		0000	0000	0000	1101	None	
TBLWT*-		Table Write with post-decrement		0000	0000	0000	1110	None	
TBLWT+*		Table Write with pre-increment		0000	0000	0000	1111	None	

Note 1: If Program Counter (PC) is modified or a conditional test is true, the instruction requires an additional cycle. The extra cycle is executed as a NOP.

2: Some instructions are multi word instructions. The second/third words of these instructions will be decoded 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.

3: f_s and f_d do not cover the full memory range. 2 MSBs of bank selection are forced to ‘b00 to limit the range of these instructions to lower 4k addressing space.

41.1.1 STANDARD INSTRUCTION SET

ADDFSR **Add Literal to FSR**

Syntax: ADDFSR f, k

Operands: $0 \le k \le 63$
$f \in [0, 1, 2]$

Operation: FSR(f) + k → FSR(f)

Status Affected: None

Encoding:

1110	1000	ffkk	kkkk

Description: The 6-bit literal 'k' is added to the contents of the FSR specified by 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to FSR
Decode	Read literal 'k'	Process Data	Write to FSR

Example: `ADDFSR 2, 23h`

Before Instruction
FSR2 = 03FFh
After Instruction
FSR2 = 0422h

ADDLW **ADD literal to W**

Syntax: ADDLW k

Operands: $0 \le k \le 255$

Operation: (W) + k → W

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

Encoding:

0000	1111	kkkk	kkkk

Description: The contents of W are added to the 8-bit literal 'k' and 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: `ADDLW 15h`

Before Instruction
W = 10h
After Instruction
W = 25h

ADDWF **ADD W to f**

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

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

Operation: (W) + (f) → dest

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

Encoding:

0010	01da	ffff	ffff

Description: Add W to register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored 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 $f \le 95$ (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: `ADDWF REG, 0, 0`

Before Instruction
W = 17h
REG = 0C2h
After Instruction
W = 0D9h
REG = 0C2h

Note: All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction format then becomes: {label} instruction argument(s).

BNC **Branch if Not Carry**

Syntax: BNC n

Operands: -128 ≤ n ≤ 127

Operation: if CARRY bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0011	nnnn	nnnn

Description: If the CARRY bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

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

If No Jump:

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

Example: `HERE BNC Jump`

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

BNN **Branch if Not Negative**

Syntax: BNN n

Operands: -128 ≤ n ≤ 127

Operation: if NEGATIVE bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0111	nnnn	nnnn

Description: If the NEGATIVE bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

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

If No Jump:

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

Example: `HERE BNN Jump`

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

TABLE 44-3: SUPPLY CURRENT (I_{DD})(1,2,4)

PIC18LF27/47/57K42			Standard Operating Conditions (unless otherwise stated)					
PIC18F27/47/57K42								
Param. No.	Symbol	Device Characteristics	Min.	Typ.†	Max.	Units	Conditions	
							V_{DD}	Note
D100	$I_{DD_{XT4}}$	XT = 4 MHz	—	625	1200	µA	3.0V	
D100	$I_{DD_{XT4}}$	XT = 4 MHz	—	825	1400	µA	3.0V	
D100A	$I_{DD_{XT4}}$	XT = 4 MHz	—	425	—	µA	3.0V	PMD's all 1's
D100A	$I_{DD_{XT4}}$	XT = 4 MHz	—	665	—	µA	3.0V	PMD's all 1's
D101	$I_{DD_{HFO16}}$	HFINTOSC = 16 MHz	—	2.9	5	mA	3.0V	
D101	$I_{DD_{HFO16}}$	HFINTOSC = 16 MHz	—	3	5.1	mA	3.0V	
D101A	$I_{DD_{HFO16}}$	HFINTOSC = 16 MHz	—	2.0	—	mA	3.0V	PMD's all 1's
D101A	$I_{DD_{HFO16}}$	HFINTOSC = 16 MHz	—	2.1	—	mA	3.0V	PMD's all 1's
D102	$I_{DD_{HFOPLL}}$	HFINTOSC = 64 MHz	—	10.7	17.5	mA	3.0V	
D102	$I_{DD_{HFOPLL}}$	HFINTOSC = 64 MHz	—	11	18	mA	3.0V	
D102A	$I_{DD_{HFOPLL}}$	HFINTOSC = 64 MHz	—	6.7	—	mA	3.0V	PMD's all 1's
D102A	$I_{DD_{HFOPLL}}$	HFINTOSC = 64 MHz	—	6.9	—	mA	3.0V	PMD's all 1's
D103	$I_{DD_{HSPLL64}}$	HS+PLL = 64 MHz	—	10.7	17.5	mA	3.0V	
D103	$I_{DD_{HSPLL64}}$	HS+PLL = 64 MHz	—	11	18	mA	3.0V	
D103A	$I_{DD_{HSPLL64}}$	HS+PLL = 64 MHz	—	6.7	—	mA	3.0V	PMD's all 1's
D103A	$I_{DD_{HSPLL64}}$	HS+PLL = 64 MHz	—	6.9	—	mA	3.0V	PMD's all 1's
D104	$I_{DD_{IDLE}}$	IDLE mode, HFINTOSC = 16 MHz	—	2.0	—	mA	3.0V	
D104	$I_{DD_{IDLE}}$	IDLE mode, HFINTOSC = 16 MHz	—	2.1	—	mA	3.0V	
D105	$I_{DD_{DOZE}}$(3)	DOZE mode, HFINTOSC = 16 MHz, Doze Ratio = 16	—	2.0	—	mA	3.0V	
D105	$I_{DD_{DOZE}}$(3)	DOZE mode, HFINTOSC = 16 MHz, Doze Ratio = 16	—	2.1	—	mA	3.0V	

† Data in "Typ." column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: The test conditions for all I_{DD} measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins are outputs driven low; $\overline{MCLR}$ = V_{DD}; WDT disabled.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.

3: $I_{DD_{DOZE}}$ = [$I_{DD_{IDLE}}$*(N-1)/N] + $I_{DD_{HFO}}$16/N where N = DOZE Ratio (Register 10-2).

4: PMD bits are all in the default state, no modules are disabled.