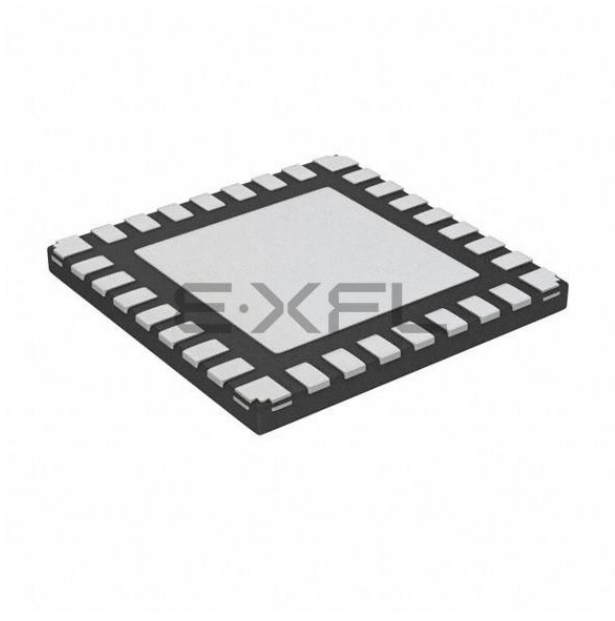


Welcome to E-XFL.COM

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	CANbus, I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	25
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 24x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	28-UQFN Exposed Pad
Supplier Device Package	28-UQFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf26k83-e-mx

REGISTER 5-5: CONFIGURATION WORD 3L (30 0004h)

U-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	WDTE<1:0>		WDTCPS<4:0>				
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '1'	
-n = Value for blank device	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **Unimplemented:** Read as '1'

bit 6-5 **WDTE<1:0>:** WDT Operating Mode bits

00 = WDT is disabled, SWDTEN is ignored
01 = WDT is enabled/disabled by the SWDTEN bit in WDTCON0
10 = WDT is enabled while Sleep = 0, suspended when Sleep = 1; SWDTEN is ignored
11 = WDT is enabled regardless of Sleep; SWDTEN is ignored

bit 4-0 **WDTCPS<4:0>:** WDT Period Select bits

WDTCPS<4:0>	WDTPS at POR				Software Control of WDTPS?
	Value	Divider Ratio		Typical Time-out (FIN = 31 kHz)	
00000	00000	1:32	2^5	1 ms	No
00001	00001	1:64	2^6	2 ms	
00010	00010	1:128	2^7	4 ms	
00011	00011	1:256	2^8	8 ms	
00100	00100	1:512	2^9	16 ms	
00101	00101	1:1024	2^{10}	32 ms	
00110	00110	1:2048	2^{11}	64 ms	
00111	00111	1:4096	2^{12}	128 ms	
01000	01000	1:8192	2^{13}	256 ms	
01001	01001	1:16384	2^{14}	512 ms	
01010	01010	1:32768	2^{15}	1s	
01011	01011	1:65536	2^{16}	2s	
01100	01100	1:131072	2^{17}	4s	
01101	01101	1:262144	2^{18}	8s	
01110	01110	1:524299	2^{19}	16s	
01111	01111	1:1048576	2^{20}	32s	
10000	10000	1:2097152	2^{21}	64s	
10001	10001	1:4194304	2^{22}	128s	
10010	10010	1:8388608	2^{23}	256s	
10011 ... 11110	10011 ... 11110	1:32	2^5	1 ms	No
11111	01011	1:65536	2^{16}	2s	Yes

REGISTER 9-25: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2

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
I2C1RXIP	SPI1IP	SPI1TXIP	SPI1RXIP	DMA1AIP	DMA1ORIP	DMA1DCNTIP	DMA1SCNTIP
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 **I2C1RXIP:** I^2C1 Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **SPI1IP:** SPI1 Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **SPI1TXIP:** I^2C1 Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **SPI1RXIP:** SPI1 Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **DMA1AIP:** DMA1 Abort Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **DMA1ORIP:** DMA1 Overrun Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **DMA1DCNTIP:** DMA1 Destination Count Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **DMA1SCNTIP:** DMA1 Source Count Interrupt Priority bit
1 = High priority
0 = Low priority

10.4 Register Definitions: Voltage Regulator Control

REGISTER 10-1: VREGCON: VOLTAGE REGULATOR CONTROL REGISTER(1)

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-1/1
—	—	—	—	—	—	VREGPM	Reserved
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 7-2 **Unimplemented:** Read as '0'

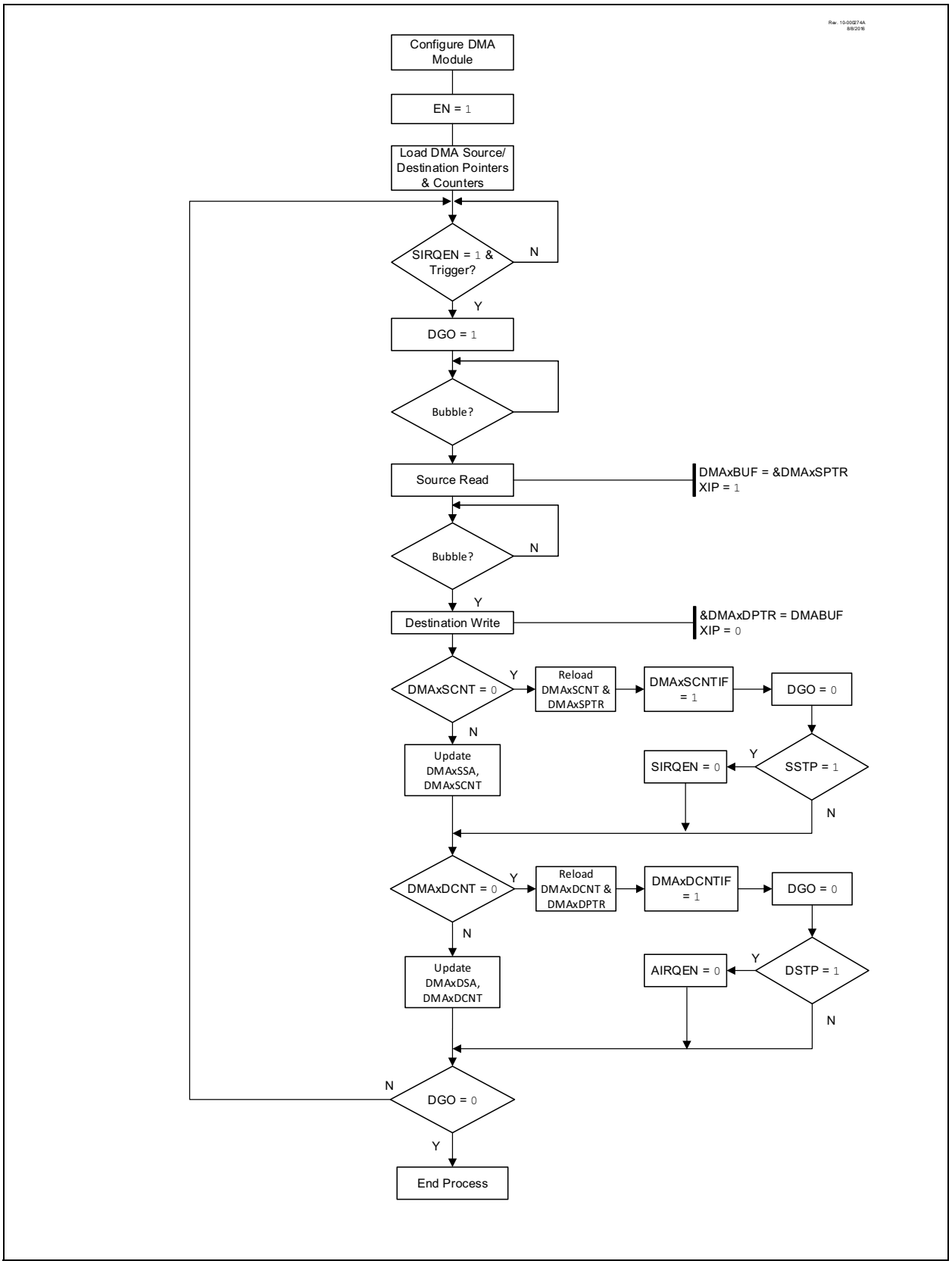
bit 1 **VREGPM:** Voltage Regulator Power Mode Selection bit
- 1 = Low-Power Sleep mode enabled in Sleep(2)
  Draws lowest current in Sleep, slower wake-up
- 0 = Normal Power mode enabled in Sleep(2)
  Draws higher current in Sleep, faster wake-up

bit 0 **Reserved:** Read as '1'. Maintain this bit set.

Note 1: Not present in LF parts.

2: See **Section 45.0 "Electrical Specifications"**.

FIGURE 15-4: DMA OPERATION WITH HARDWARE TRIGGER



REGISTER 19-3: PMD2: PMD CONTROL REGISTER 2

U-0	R/W-0/0	R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	DACMD	ADCMD	—	—	CMP2MD	CMP1MD	ZCDMD(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	q = Value depends on condition

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

bit 6 **DACMD:** Disable DAC bit
- 1 = DAC module disabled
- 0 = DAC module enabled

bit 5 **ADCMD:** Disable ADCC bit
- 1 = ADCC module disabled
- 0 = ADCC module enabled

bit 4-3 **Unimplemented:** Read as '0'

bit 2 **CMP2MD:** Disable Comparator CMP2 bit
- 1 = CMP2 module disabled
- 0 = CMP2 module enabled

bit 1 **CMP1MD:** Disable Comparator CMP1 bit
- 1 = CMP1 module disabled
- 0 = CMP1 module enabled

bit 0 **ZCDMD:** Disable Zero-Cross Detect module bit(1)
- 1 = ZCD module disabled
- 0 = ZCD module enabled

Note 1: Subject to $\overline{\text{ZCD}}$ bit in CONFIG2H.

REGISTER 20-2: 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

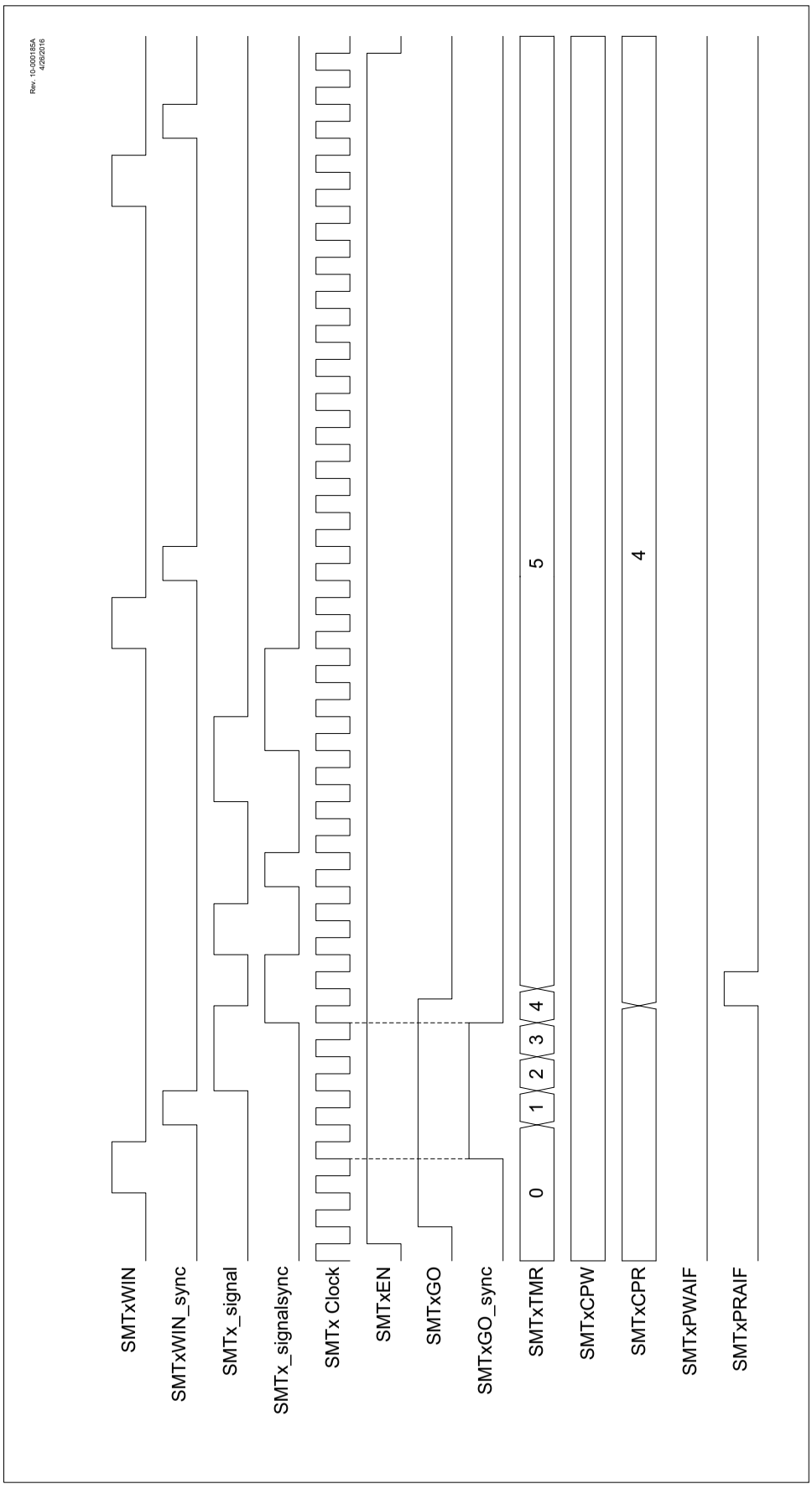
TABLE 22-3: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER2

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
TxPR	Timer2 Module Period Register								305*
TxTMR	Holding Register for the 8-bit T2TMR Register								305*
TxCON	ON	CKPS<2:0>			OUTPS<3:0>				323
TxCLK	—	—	—	—	—	CS<2:0>			320
TxRST	—	—	—	—	RSEL<3:0>				321
TxHLT	$\overline{\text{PSYNC}}$	CPOL	$\overline{\text{CSYNC}}$	MODE<4:0>					324

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for Timer2 module.

* Page provides register information.

FIGURE 25-15: TIME OF FLIGHT MODE SINGLE ACQUISITION TIMING DIAGRAM



REGISTER 26-8: CWGxDBR: CWG RISING DEAD-BAND COUNT REGISTER

U-0	U-0	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
—	—	DBR<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	q = Value depends on condition

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

bit 5-0 **DBR<5:0>:** CWG Rising Edge Triggered Dead-Band Count bits

```
11 1111 =  63-64 CWG clock periods
11 1110 =  62-63 CWG clock periods
.
.
.
00 0010 =  2-3 CWG clock periods
00 0001 =  1-2 CWG clock periods
00 0000 =  0 CWG clock periods. Dead-band generation is by-passed
```

REGISTER 26-9: CWGxDBF: CWG FALLING DEAD-BAND COUNT REGISTER

U-0	U-0	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
—	—	DBF<5:0>					
bit 7							bit 0

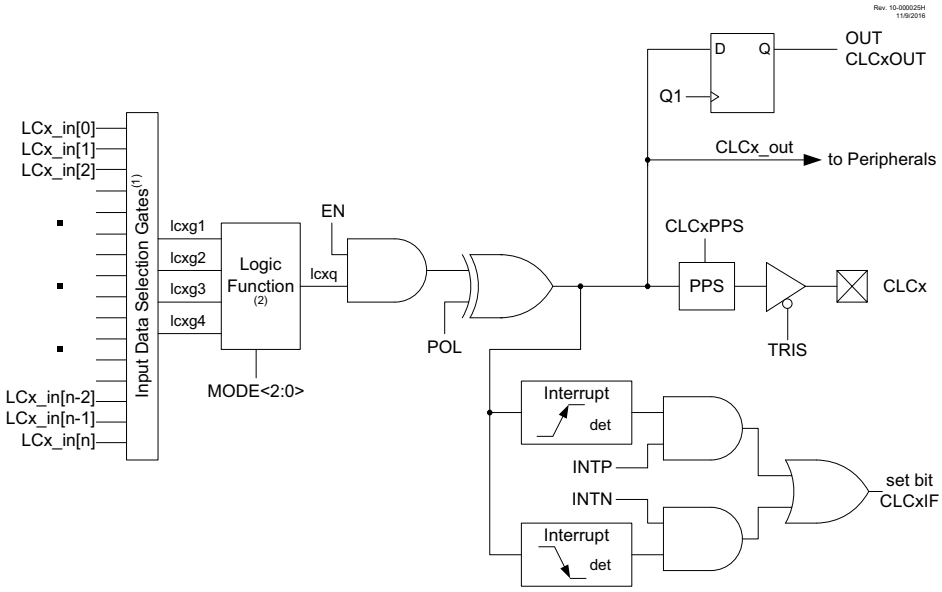
Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	q = Value depends on condition

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

bit 5-0 **DBF<5:0>:** CWG Falling Edge Triggered Dead-Band Count bits

```
11 1111 =  63-64 CWG clock periods
11 1110 =  62-63 CWG clock periods
.
.
.
00 0010 =  2-3 CWG clock periods
00 0001 =  1-2 CWG clock periods
00 0000 =  0 CWG clock periods. Dead-band generation is by-passed.
```

FIGURE 27-1: CLCx SIMPLIFIED BLOCK DIAGRAM



Note 1: See Figure 27-2: Input Data Selection and Gating

2: See Figure 27-3: Programmable Logic Functions.

27.1 CLCx Setup

Programming the CLCx module is performed by configuring the four stages in the logic signal flow. The four stages are:

- Data selection
- Data gating
- Logic function selection
- Output polarity

Each stage is setup at run time by writing to the corresponding CLCx Special Function Registers. This has the added advantage of permitting logic reconfiguration on-the-fly during program execution.

27.1.1 DATA SELECTION

There are 32 signals available as inputs to the configurable logic. Four 32-input multiplexers are used to select the inputs to pass on to the next stage.

Data selection is through four multiplexers as indicated on the left side of Figure 27-2. Data inputs in the figure are identified by a generic numbered input name.

Table 27-1 correlates the generic input name to the actual signal for each CLC module. The column labeled 'DyS<4:0> Value' indicates the MUX selection code for the selected data input. DyS is an abbreviation for the MUX select input codes: D1S<4:0> through D4S<4:0>.

Data inputs are selected with CLCxSEL0 through CLCxSEL3 registers (Register 27-3 through Register 27-6).

Note:	Data selections are undefined at power-up.

32.4.2 VARIABLE TRANSFER SIZE MODE (BMODE = 1)

In this mode, SPIxTWIDTH specifies the width of every individual piece of the data transfer in bits. SPIxTCTH/SPIxTCTL specifies the number of transfers of this bit length. If SPIxTWIDTH = 0, each piece is a full byte of data. If SPIxTWIDTH ≠ 0, then only the specified number of bits from the transmit FIFO are shifted out, with the unused bits ignored. Received data is padded with zeros in the unused bit areas when transfered into the receive FIFO. The LSBF bit of SPIxCON0 determines whether the Most Significant or Least Significant bits of the transfers are ignored/padded. In this mode, the transfer counter being zero only stops messages from being sent/received when in "Receive only" mode.

Note:	With BMODE = 1, it is possible for the transfer counter (SPIxTCTH/L) to decrement below zero, although when in "Receive Only" Master mode, transfer clocks will cease when the transfer counter reaches zero.

32.4.3 TRANSFER COUNTER IN SLAVE MODE

In Slave mode, the transfer counter will still decrement as data is shifted in and out of the SPI module, but it will not control data transfers. In addition, in Slave mode, the BMODE bit along with the transfer counter is used to determine when the device should look for Slave Select faults. If BMODE = 0, the SSFLT bit will be set if Slave Select transitions from its active to inactive state during bytes of data, as well as if it transitions before the last bit sent during the final byte (if SPIxTWIDTH≠0). If BMODE=1, the SSFLT bit will be set if Slave Select transitions from its active to inactive state before the final bit of each individual transfer is completed. Note that SSFLT does not have an associated interrupt, so it should be checked in software. An ideal time to do this is when the End of Slave Select Interrupt (EOSIF) is triggered (see **Section 32.8.3.3 "Start of Slave Select and End of Slave Select Interrupts"**).

32.5 Master mode

In Master mode, the device controls the SCK line, and as such, initiates data transfers and determines when any slaves broadcast data onto the SPI bus.

Master mode of this device can be configured in four different modes, configured by the TXR and RXR bits:

- Full-Duplex mode
- Receive Only mode
- Transmit Only mode
- Transfer Off mode

The modes are illustrated in Table 32-1, below:

33.1 I²C Features

- Inter-Integrated Circuit (I²C) interface supports the following modes in hardware:
  - Master mode
  - Slave mode with byte NACKing
  - Multi-Master mode
- Dedicated Address, Receive and Transmit buffers
- Up to four slave addresses matching
- General Call address matching
- 7-bit and 10-bit addressing with masking
- Start, Restart, Stop, Address, Write, and ACK Interrupts
- Clock Stretching hardware for:
  - RX Buffer Full
  - TX Buffer Empty
  - After Address, Write, and ACK
- Bus Collision Detection with arbitration
- Bus Timeout Detection
- SDA hold time selection
- I²C, SMBus 2.0, and SMBus 3.0 input level selections

33.2 I²C Module Overview

The I²C module provides a synchronous interface between the microcontroller and other I²C-compatible devices using the two-wire I²C serial bus. Devices communicate in a master/slave environment. The I²C bus specifies two signal connections:

- Serial Clock (SCL)
- Serial Data (SDA)

Both the SCL and SDA connections are bidirectional open-drain lines, each requiring pull-up resistors to the supply voltage. Pulling the line to ground is considered a logical zero and letting the line float is considered a logical one. Every transaction on the I²C bus has to be initiated by the master.

Figure 33-2 shows a typical connection between a master and more than one slave.

FIGURE 33-2: I²C MASTER/SLAVE CONNECTIONS

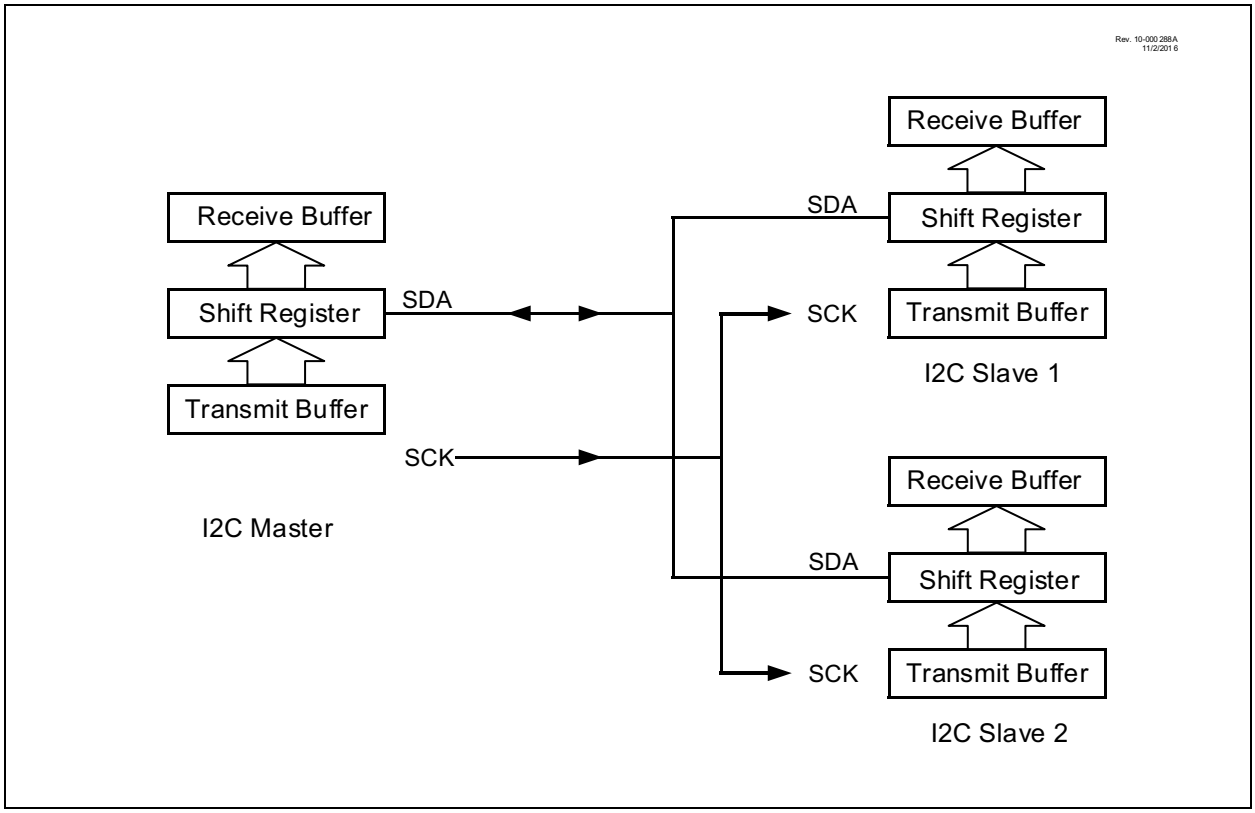
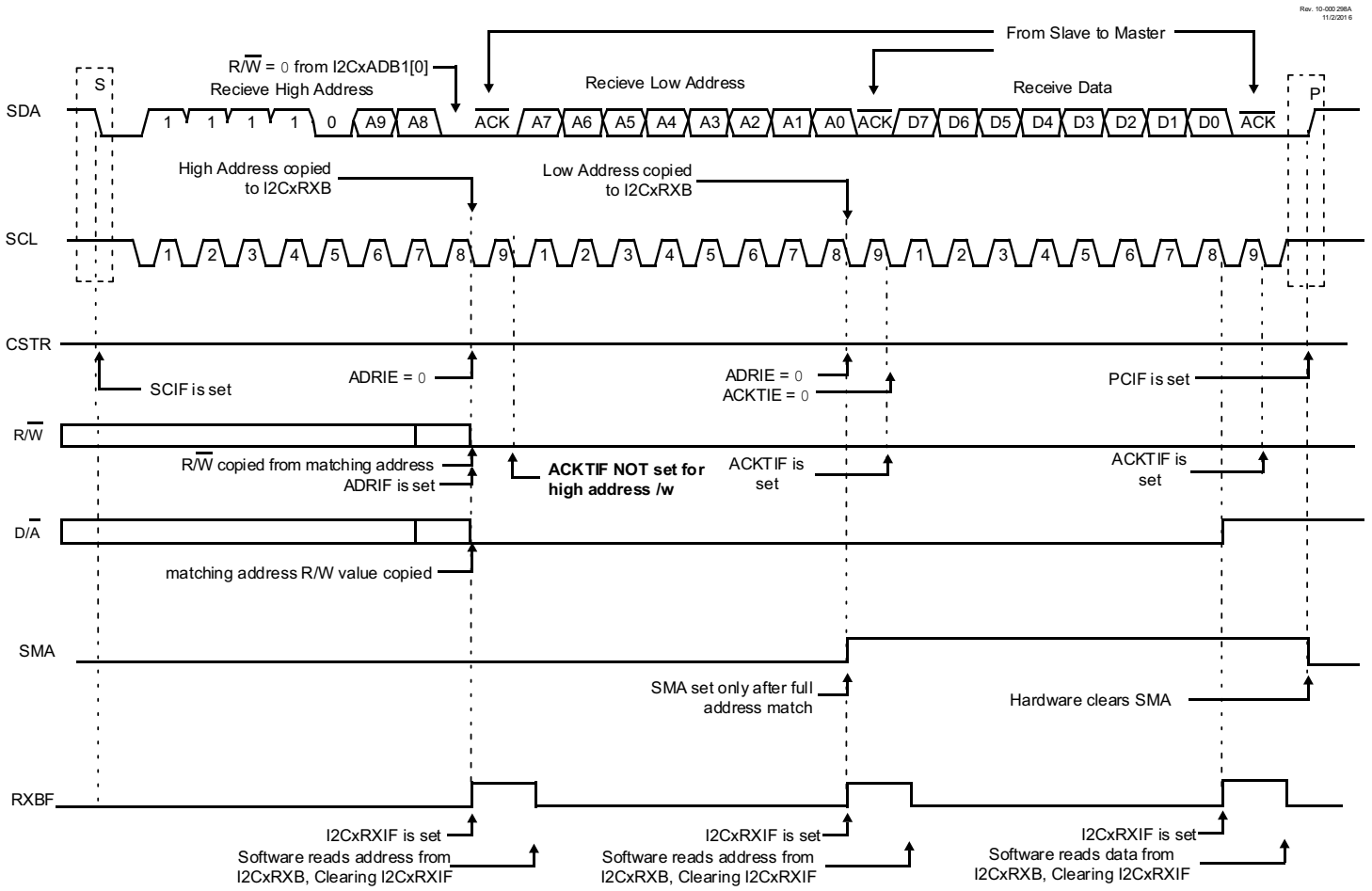


FIGURE 33-11: I^2C SLAVE, 10-BIT ADDRESS, RECEPTION WITH STOP (ADB = 1)



messages regardless of the values of the acceptance filters. Also, if a message has an error before the end of frame, that portion of the message assembled in the MAB before the error frame will be loaded into the buffer. This mode may serve as a valuable debugging tool for a given CAN network. It should not be used in an actual system environment as the actual system will always have some bus errors and all nodes on the bus are expected to ignore them.

In Mode 1 and 2, when a programmable buffer is configured as a transmit buffer and one or more acceptance filters are associated with it, all incoming messages matching this acceptance filter criteria will be discarded. To avoid this scenario, user firmware must make sure that there are no acceptance filters associated with a buffer configured as a transmit buffer.

34.6.2 RECEIVE PRIORITY

When in Mode 0, RXB0 is the higher priority buffer and has two message acceptance filters associated with it. RXB1 is the lower priority buffer and has four acceptance filters associated with it. The lower number of acceptance filters makes the match on RXB0 more restrictive and implies a higher priority for that buffer. Additionally, the RXB0CON register can be configured such that if RXB0 contains a valid message and another valid message is received, an overflow error will not occur and the new message will be moved into RXB1 regardless of the acceptance criteria of RXB1. There are also two programmable acceptance filter masks available, one for each receive buffer (see **Section 34.4 "CAN Message Buffers"**).

In Mode 1 and 2, there are a total of 16 acceptance filters available and each can be dynamically assigned to any of the receive buffers. A buffer with a lower number has higher priority. Given this, if an incoming message matches with two or more receive buffer acceptance criteria, the buffer with the lower number will be loaded with that message.

34.6.3 ENHANCED FIFO MODE

When configured for Mode 2, two of the dedicated receive buffers in combination with one or more programmable transmit/receive buffers, are used to create a maximum of an eight buffers deep FIFO buffer. In this mode, there is no direct correlation between filters and receive buffer registers. Any filter that has been enabled can generate an acceptance. When a message has been accepted, it is stored in the next available receive buffer register and an internal Write Pointer is incremented. The FIFO can be a maximum of eight buffers deep. The entire FIFO must consist of contiguous receive buffers. The FIFO head begins at RXB0 buffer and its tail spans toward B5. The maximum length of the FIFO is limited by the presence or absence of the first transmit buffer starting from B0. If a buffer is configured as a transmit buffer, the FIFO length is reduced accordingly. For instance, if B3 is configured as a transmit buffer, the actual FIFO will consist of RXB0, RXB1, B0, B1 and B2, a total of five buffers. If B0 is configured as a transmit buffer, the FIFO length will be two. If none of the programmable buffers are configured as a transmit buffer, the FIFO will be eight buffers deep. A system that requires more transmit buffers should try to locate transmit buffers at the very end of B0-B5 buffers to maximize available FIFO length.

When a message is received in FIFO mode, the Interrupt Flag Code bits (EICODE<4:0>) in the CANSTAT register will have a value of '10000', indicating the FIFO has received a message. FIFO Pointer bits, FP<3:0> in the CANCON register, point to the buffer that contains data not yet read. The FIFO Pointer bits, in this sense, serve as the FIFO Read Pointer. The user should use the FP bits and read corresponding buffer data. When receive data is no longer needed, the RXFUL bit in the current buffer must be cleared, causing FP<3:0> to be updated by the module.

To determine whether FIFO is empty or not, the user may use the FP<3:0> bits to access the RXFUL bit in the current buffer. If RXFUL is cleared, the FIFO is considered to be empty. If it is set, the FIFO may contain one or more messages. In Mode 2, the module also provides a bit called FIFO High Water Mark (FIFOWM) in the ECANCON register. This bit can be used to cause an interrupt whenever the FIFO contains only one or four empty buffers. The FIFO high water mark interrupt can serve as an early warning to a full FIFO condition.

34.6.4 TIME-STAMPING

The CAN module can be programmed to generate a time-stamp for every message that is received. When enabled, the module generates a capture signal for the CCP modules, which in turn captures the value of Timer1, Timer3 or Timer5. This value can be used as the message time-stamp.

To use the time-stamp capability, set the CTS<3:0> bits of the appropriate CCPxCAP register to '1000' to configure the CCP module capture input to the CAN_rx_timestamp signal.

In addition, the CAN_rx_timestamp can be chosen as a signal input for the Signal Measurement Timer, which can be used for a variety of other timing applications.

39.2 Comparator Control

Each comparator has two control registers: CMxCON0 and CMxCON1.

The CMxCON0 register (see Register 39-1) contains Control and Status bits for the following:

- Enable
- Output
- Output polarity
- Hysteresis enable
- Timer1 output synchronization

The CMxCON1 register (see Register 39-2) contains Control bits for the following:

- Interrupt on positive/negative edge enables

The CMxPCH and CMxNCH registers are used to select the positive and negative input channels, respectively.

39.2.1 COMPARATOR ENABLE

Setting the EN bit of the CMxCON0 register enables the comparator for operation. Clearing the EN bit disables the comparator resulting in minimum current consumption.

39.2.2 COMPARATOR OUTPUT

The output of the comparator can be monitored by reading either the CxOUT bit of the CMxCON0 register or the CxOUT bit of the CMOUT register.

The comparator output can also be routed to an external pin through the RxyPPS register (Register 17-2). The corresponding TRIS bit must be clear to enable the pin as an output.

Note 1:	The internal output of the comparator is latched with each instruction cycle. Unless otherwise specified, external outputs are not latched.

39.2.3 COMPARATOR OUTPUT POLARITY

Inverting the output of the comparator is functionally equivalent to swapping the comparator inputs. The polarity of the comparator output can be inverted by setting the POL bit of the CMxCON0 register. Clearing the POL bit results in a noninverted output.

Table 39-1 shows the output state versus input conditions, including polarity control.

TABLE 39-1: COMPARATOR OUTPUT STATE VS. INPUT CONDITIONS

Input Condition	POL	CxOUT
CxVN > CxVP	0	0
CxVN < CxVP	0	1
CxVN > CxVP	1	1
CxVN < CxVP	1	0

42.1.1 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 42.2.3 "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).

CPFSEQ **Compare f with W, skip if f = W**

Syntax: CPFSEQ f {,a}

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

Operation: (f) – (W),
skip if (f) = (W)
(unsigned comparison)

Status Affected: None

Encoding:

0110	001a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.
If 'f' = W, then the fetched instruction is discarded and a NOP is executed instead, making this 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 $f \le 95$ (5Fh). See **Section 42.2.3 "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	No operation

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     CPFSEQ REG, 0
NEQUAL   :
EQUAL    :
```
Before Instruction
PC Address = HERE
W = ?
REG = ?
After Instruction
If REG = W;
PC = Address (EQUAL)
If REG ≠ W;
PC = Address (NEQUAL)

CPFSGT **Compare f with W, skip if f > W**

Syntax: CPFSGT f {,a}

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

Operation: (f) – (W),
skip if (f) > (W)
(unsigned comparison)

Status Affected: None

Encoding:

0110	010a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of the W by performing an unsigned subtraction.
If the contents of 'f' are greater than the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this 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 $f \le 95$ (5Fh). See **Section 42.2.3 "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	No operation

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     CPFSGT REG, 0
NGREATER :
GREATER  :
```
Before Instruction
PC = Address (HERE)
W = ?
After Instruction
If REG > W;
PC = Address (GREATER)
If REG ≤ W;
PC = Address (NGREATER)

TABLE 42-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected
				MSb			LSb	
ADDULNK	k	Add FSR2 with (k) & return	2	1110	1000	11kk	kkkk	None
MOVSF	z_s, f_d	Move z_s (source) to 1st word	2	1110	1011	0zzz	zzzz	None
		f_d (destination) 2nd word	2	1111	ffff	ffff	ffff	
MOVSFL	z_s, f_d	Opcode 1st word		0000	0000	0000	0010	None
		Move z_s (source) to 2nd word	3	1111	xxxz	zzzz	zzff	
		f_d (full destination) 3rd word		1111	ffff	ffff	ffff	
MOVSS	z_s, z_d	Move z_s (source) to 1st word		1110	1011	1zzz	zzzz	None
		z_d (destination) 2nd word	2	1111	xxxx	xzzz	zzzz	
PUSHL	k	Push literal to POSTDEC2	1	1110	1010	kkkk	kkkk	None
SUBULNK	k	Subtract (k) from FSR2 & return	2	1110	1001	11kk	kkkk	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: Only available when extended instruction set is enabled.

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

44.11 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.

44.12 Third-Party Development Tools

Microchip also offers a great collection of tools from third-party vendors. These tools are carefully selected to offer good value and unique functionality.

- Device Programmers and Gang Programmers from companies, such as SoftLog and CCS
- Software Tools from companies, such as Gimpel and Trace Systems
- Protocol Analyzers from companies, such as Saleae and Total Phase
- Demonstration Boards from companies, such as MikroElektronika, Digilent® and Olimex
- Embedded Ethernet Solutions from companies, such as EZ Web Lynx, WIZnet and IPLogika®

TABLE 45-2: SUPPLY CURRENT (I_{DD})(1,2,4)

PIC18LF25/26K83			Standard Operating Conditions (unless otherwise stated)					
PIC18F25/26K83								
Param. No.	Symbol	Device Characteristics	Min.	Typ. †	Max.	Units	Conditions	
							V_{DD}	Note
D100	$I_{DD_{XT4}}$	XT = 4 MHz	—	590	1200	µA	3.0V	
D100	$I_{DD_{XT4}}$	XT = 4 MHz	—	770	1300	µA	3.0V	
D100A	$I_{DD_{XT4}}$	XT = 4 MHz	—	390	850	µA	3.0V	PMD's all 1's
D100A	$I_{DD_{XT4}}$	XT = 4 MHz	—	620	950	µA	3.0V	PMD's all 1's
D101	$I_{DD_{HFO16}}$	HFINTOSC = 16 MHz	—	2.3	5.0	mA	3.0V	
D101	$I_{DD_{HFO16}}$	HFINTOSC = 16 MHz	—	2.4	5.1	mA	3.0V	
D101A	$I_{DD_{HFO16}}$	HFINTOSC = 16 MHz	—	1.5	3.2	mA	3.0V	PMD's all 1's
D101A	$I_{DD_{HFO16}}$	HFINTOSC = 16 MHz	—	1.5	3.5	mA	3.0V	PMD's all 1's
D102	$I_{DD_{HFOPLL}}$	HFINTOSC = 64 MHz	—	8.4	18.5	mA	3.0V	
D102	$I_{DD_{HFOPLL}}$	HFINTOSC = 64 MHz	—	8.4	19	mA	3.0V	
D102A	$I_{DD_{HFOPLL}}$	HFINTOSC = 64 MHz	—	5	11	mA	3.0V	PMD's all 1's
D102A	$I_{DD_{HFOPLL}}$	HFINTOSC = 64 MHz	—	5	11.5	mA	3.0V	PMD's all 1's
D103	$I_{DD_{HSPLL64}}$	HS+PLL = 64 MHz	—	8.4	19	mA	3.0V	
D103	$I_{DD_{HSPLL64}}$	HS+PLL = 64 MHz	—	8.4	20	mA	3.0V	
D103A	$I_{DD_{HSPLL64}}$	HS+PLL = 64 MHz	—	5	10	mA	3.0V	PMD's all 1's
D103A	$I_{DD_{HSPLL64}}$	HS+PLL = 64 MHz	—	5	10	mA	3.0V	PMD's all 1's
D104	$I_{DD_{IDLE}}$	IDLE mode, HFINTOSC = 16 MHz	—	1.9	4.5	mA	3.0V	
D104	$I_{DD_{IDLE}}$	IDLE mode, HFINTOSC = 16 MHz	—	1.9	4.5	mA	3.0V	
D105	$I_{DD_{DOZE}}$(3)	DOZE mode, HFINTOSC = 16 MHz, Doze Ratio = 16	—	1.9	—	mA	3.0V	
D105	$I_{DD_{DOZE}}$(3)	DOZE mode, HFINTOSC = 16 MHz, Doze Ratio = 16	—	1.9	—	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.