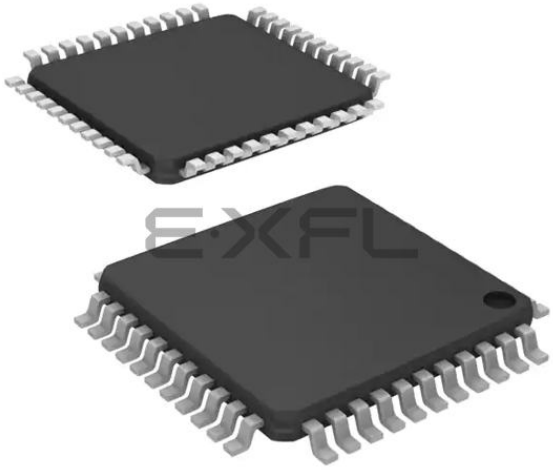


Welcome to E-XFL.COM

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	36
Program Memory Size	128KB (64K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.6K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 35x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f47k40t-i-pt

5.5 Register Definitions: Reference Clock

Long bit name prefixes for the Reference Clock peripherals are shown in Table 5-1. Refer to **Section 1.4.2.2 "Long Bit Names"** for more information.

TABLE 5-1:

Peripheral	Bit Name Prefix
CLKR	CLKR

REGISTER 5-1: CLKRCON: REFERENCE CLOCK CONTROL REGISTER

R/W-0/0	U-0	U-0	R/W-1/1	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
EN	—	—	DC<1:0>		DIV<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	

bit 7 **EN:** Reference Clock Module Enable bit
- 1 = Reference clock module enabled
- 0 = Reference clock module is disabled

bit 6-5 **Unimplemented:** Read as '0'

bit 4-3 **DC<1:0>:** Reference Clock Duty Cycle bits(1)
- 11 = Clock outputs duty cycle of 75%
- 10 = Clock outputs duty cycle of 50%
- 01 = Clock outputs duty cycle of 25%
- 00 = Clock outputs duty cycle of 0%

bit 2-0 **DIV<2:0>:** Reference Clock Divider bits
- 111 = Base clock value divided by 128
- 110 = Base clock value divided by 64
- 101 = Base clock value divided by 32
- 100 = Base clock value divided by 16
- 011 = Base clock value divided by 8
- 010 = Base clock value divided by 4
- 001 = Base clock value divided by 2
- 000 = Base clock value

Note 1: Bits are valid for reference clock divider values of two or larger, the base clock cannot be further divided.

10.2 Register Definitions: Stack Pointer

REGISTER 10-1: STKPTR: STACK POINTER REGISTER

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

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented	C = Clearable only bit
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

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

bit 4-0 **STKPTR<4:0>:** Stack Pointer Location bits

10.2.1 STACK OVERFLOW AND UNDERFLOW RESETS

Device Resets on Stack Overflow and Stack Underflow conditions are enabled by setting the STVREN bit in Configuration Register 4L. When STVREN is set, a Full or Underflow condition will set the appropriate STKOVF or STKUNF bit and then cause a device Reset. When STVREN is cleared, a Full or Underflow condition will set the appropriate STKOVF or STKUNF bit but not cause a device Reset. The STKOVF or STKUNF bits are cleared by the user software or a Power-on Reset.

10.2.2 FAST REGISTER STACK

A fast register stack is provided for the Status, WREG and BSR registers, to provide a "fast return" option for interrupts. The stack for each register is only one level deep and is neither readable nor writable. It is loaded with the current value of the corresponding register when the processor vectors for an interrupt. All interrupt sources will push values into the stack registers. The values in the registers are then loaded back into their associated registers if the `RETFIE, FAST` instruction is used to return from the interrupt.

If both low and high priority interrupts are enabled, the stack registers cannot be used reliably to return from low priority interrupts. If a high priority interrupt occurs while servicing a low priority interrupt, the stack register values stored by the low priority interrupt will be overwritten. In these cases, users must save the key registers by software during a low priority interrupt.

If interrupt priority is not used, all interrupts may use the fast register stack for returns from interrupt. If no interrupts are used, the fast register stack can be used to restore the STATUS, WREG and BSR registers at the end of a subroutine call. To use the fast register stack for a subroutine call, a `CALL label, FAST` instruction must be executed to save the `STATUS`, WREG and BSR registers to the fast register stack. A `RETURN, FAST` instruction is then executed to restore these registers from the fast register stack.

Example 10-1 shows a source code example that uses the fast register stack during a subroutine call and return.

EXAMPLE 10-1: FAST REGISTER STACK CODE EXAMPLE

```
CALL SUB1, FAST    ;STATUS, WREG, BSR
                   ;SAVED IN FAST REGISTER
                   ;STACK
       •
       •
       •
SUB1   •
       •
       •
   RETURN, FAST    ;RESTORE VALUES SAVED
                   ;IN FAST REGISTER STACK
```

10.2.3 LOOK-UP TABLES IN PROGRAM MEMORY

There may be programming situations that require the creation of data structures, or look-up tables, in program memory. For PIC18 devices, look-up tables can be implemented in two ways:

- Computed GOTO
- Table Reads

10.2.3.1 Computed GOTO

A computed GOTO is accomplished by adding an offset to the program counter. An example is shown in Example 10-2.

A look-up table can be formed with an ADDWF PCL instruction and a group of RETLW nn instructions. The W register is loaded with an offset into the table before executing a call to that table. The first instruction of the called routine is the ADDWF PCL instruction. The next instruction executed will be one of the RETLW nn instructions that returns the value 'nn' to the calling function.

The offset value (in WREG) specifies the number of bytes that the program counter should advance and should be multiples of two (LSb = 0).

In this method, only one data byte may be stored in each instruction location and room on the return address stack is required.

EXAMPLE 10-2: COMPUTED GOTO USING AN OFFSET VALUE

```
        MOVF    OFFSET, W
        CALL    TABLE
ORG     nn00h
TABLE   ADDWF   PCL
        RETLW   nnh
        RETLW   nnh
        RETLW   nnh
        .
        .
        .
```

10.2.3.2 Table Reads and Table Writes

A better method of storing data in program memory allows two bytes of data to be stored in each instruction location.

Look-up table data may be stored two bytes per program word by using table reads and writes. The Table Pointer (TBLPTR) register specifies the byte address and the Table Latch (TABLAT) register contains the data that is read from or written to program memory. Data is transferred to or from program memory one byte at a time.

Table read and table write operations are discussed further in **Section 11.1.1 "Table Reads and Table Writes"**.

10.4.5 STATUS REGISTER

The STATUS register, shown in Register 10-2, contains the arithmetic status of the ALU. As with any other SFR, it can be the operand for any instruction.

If the STATUS register is the destination for an instruction that affects the Z, DC, C, OV or N bits, the results of the instruction are not written; instead, the STATUS register is updated according to the instruction performed. Therefore, the result of an instruction with the STATUS register as its destination may be different than intended. As an example, `CLRF STATUS` will set the Z bit and leave the remaining Status bits unchanged (‘`000u u1uu`’).

It is recommended that only `BCF`, `BSF`, `SWAPF`, `MOVFF` and `MOVWF` instructions are used to alter the STATUS register, because these instructions do not affect the Z, C, DC, OV or N bits in the STATUS register.

For other instructions that do not affect Status bits, see the instruction set summaries in **Section 35.0 “Instruction Set Summary”** and Table 35-3.

Note:	The C and DC bits operate as the borrow and digit borrow bits, respectively, in subtraction.

11.1.1 TABLE READS AND TABLE WRITES

In order to read and write program memory, there are two operations that allow the processor to move bytes between the program memory space and the data RAM:

- Table Read (TBLRD)
- Table Write (TBLWT)

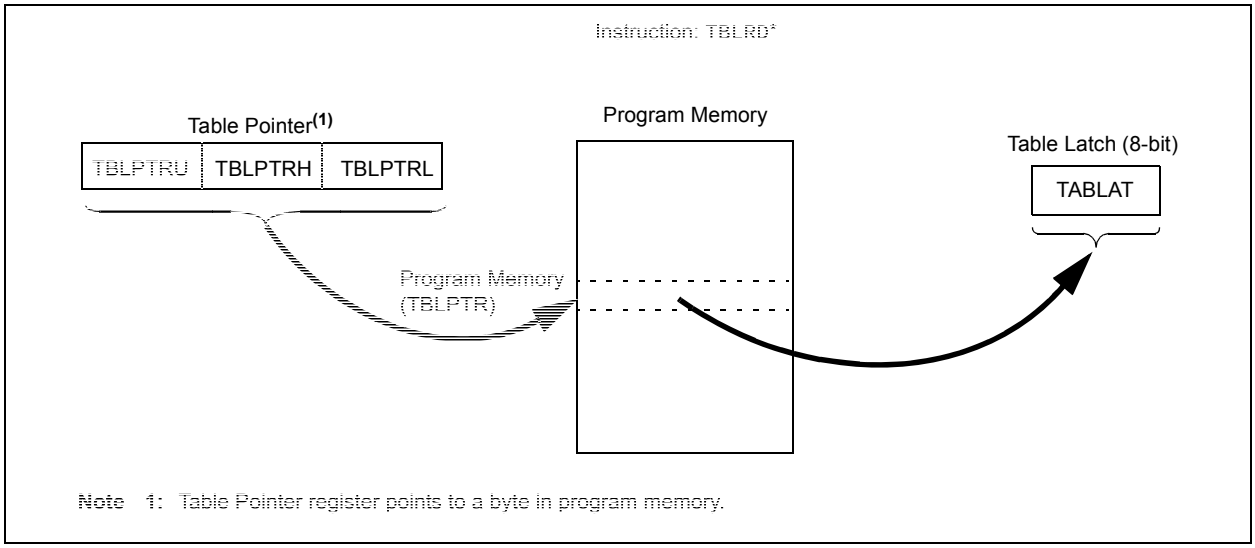
The program memory space is 16 bits wide, while the data RAM space is eight bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

The table read operation retrieves one byte of data directly from program memory and places it into the TABLAT register. Figure 11-1 shows the operation of a table read.

The table write operation stores one byte of data from the TABLAT register into a write block holding register. The procedure to write the contents of the holding registers into program memory is detailed in **Section 11.1.6 "Writing to Program Flash Memory"**. Figure 11-2 shows the operation of a table write with program memory and data RAM.

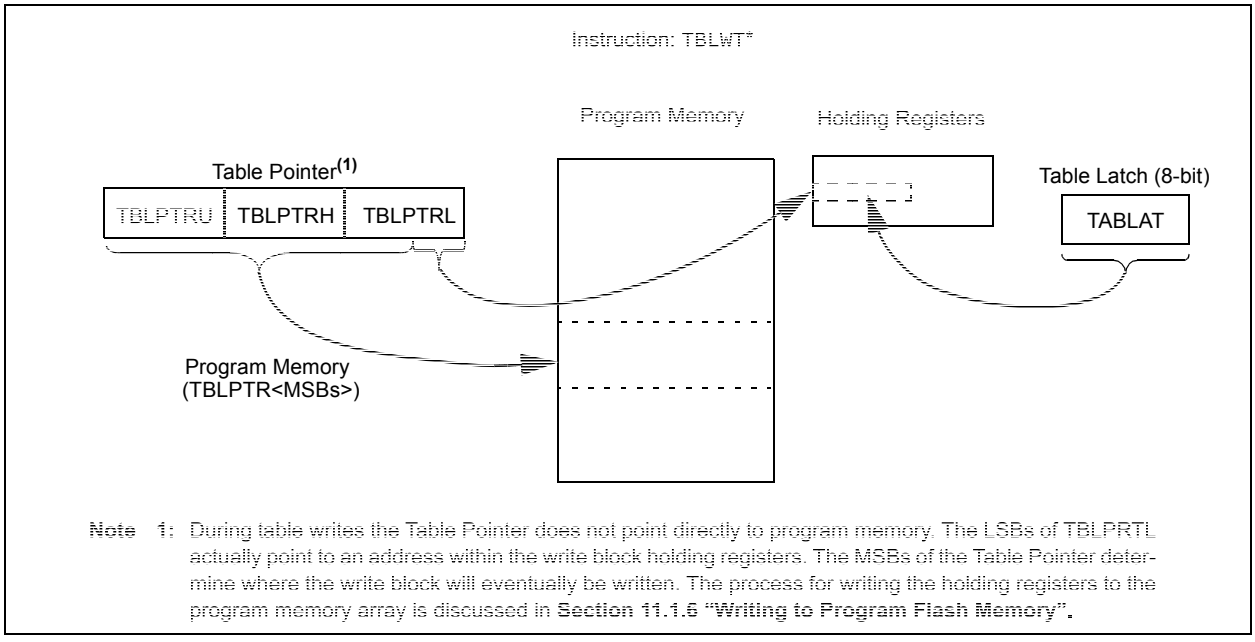
Table operations work with byte entities. Tables containing data, rather than program instructions, are not required to be word aligned. Therefore, a table can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions will need to be word aligned.

FIGURE 11-1: TABLE READ OPERATION



Note 1: Table Pointer register points to a byte in program memory.

FIGURE 11-2: TABLE WRITE OPERATION



Note 1: During table writes the Table Pointer does not point directly to program memory. The LSBs of TBLPRTL actually point to an address within the write block holding registers. The MSBs of the Table Pointer determine where the write block will eventually be written. The process for writing the holding registers to the program memory array is discussed in **Section 11.1.6 "Writing to Program Flash Memory"**.

11.3 Data EEPROM Memory

The data EEPROM is a nonvolatile memory array, separate from the data RAM and program memory, which is used for long-term storage of program data. It is not directly mapped in either the register file or program memory space but is indirectly addressed through the Special Function Registers (SFRs). The EEPROM is readable and writable during normal operation over the entire VDD range.

Four SFRs are used to read and write to the data EEPROM as well as the program memory. They are:

- NVMCON1
- NVMCON2
- NVMDAT
- NVMADRL
- NVMADRH

The data EEPROM allows byte read and write. When interfacing to the data memory block, NVMDAT holds the 8-bit data for read/write and the NVMADRH:NVMADRL register pair hold the address of the EEPROM location being accessed.

The EEPROM data memory is rated for high erase/write cycle endurance. A byte write automatically erases the location and writes the new data (erase-before-write). The write time is controlled by an internal programming timer; it will vary with voltage and temperature as well as from chip-to-chip. Please refer to the Data EEPROM Memory parameters in **Section 37.0 "Electrical Specifications"** for limits.

11.3.1 NVMADRL AND NVMADRH REGISTERS

The NVMADRH:NVMADRL registers are used to address the data EEPROM for read and write operations.

11.3.2 NVMCON1 AND NVMCON2 REGISTERS

Access to the data EEPROM is controlled by two registers: NVMCON1 and NVMCON2. These are the same registers which control access to the program memory and are used in a similar manner for the data EEPROM.

The NVMCON1 register (Register 11-1) is the control register for data and program memory access. Control bits NVMREG<1:0> determine if the access will be to program, Data EEPROM Memory or the User IDs, Configuration bits, Revision ID and Device ID.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear.

The WRERR bit is set by hardware when the WR bit is set and cleared when the internal programming timer expires and the write operation is complete.

The WR control bit initiates write operations. The bit can be set but not cleared by software. It is cleared only by hardware at the completion of the write operation.

The NVMIF interrupt flag bit of the PIR7 register is set when the write is complete. It must be cleared by software.

Control bits, RD and WR, start read and erase/write operations, respectively. These bits are set by firmware and cleared by hardware at the completion of the operation.

The RD bit cannot be set when accessing program memory (NVMREG<1:0> = 0x10). Program memory is read using table read instructions. See **Section 11.1.1 "Table Reads and Table Writes"** regarding table reads.

TABLE 16-1: IOC REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IOCAP	IOCAP7	IOCAP6	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0
IOCAN	IOCAN7	IOCAN6	IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0
IOCAF	IOCAF7	IOCAF6	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0
IOCBP	IOCBP7	IOCBP6	IOCBP5	IOCBP4	IOCBP3	IOCBP2	IOCBP1	IOCBP0
IOCBN	IOCBN7	IOCBN6	IOCBN5	IOCBN4	IOCBN3	IOCBN2	IOCBN1	IOCBN0
IOCBF	IOCBF7	IOCBF6	IOCBF5	IOCBF4	IOCBF3	IOCBF2	IOCBF1	IOCBF0
IOCCP	IOCCP7	IOCCP6	IOCCP5	IOCCP4	IOCCP3	IOCCP2	IOCCP1	IOCCP0
IOCCN	IOCCN7	IOCCN6	IOCCN5	IOCCN4	IOCCN3	IOCCN2	IOCCN1	IOCCN0
IOCCF	IOCCF7	IOCCF6	IOCCF5	IOCCF4	IOCCF3	IOCCF2	IOCCF1	IOCCF0
IOCEP	—	—	—	—	IOCEP3(1)	—	—	—
IOCEN	—	—	—	—	IOCEN3(1)	—	—	—
IOCEF	—	—	—	—	IOCEF3(1)	—	—	—

Note 1: If MCLRE = 1 or LVP = 1, RE3 port functionality is disabled and IOC on RE3 is not available.

TABLE 16-2: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPT-ON-CHANGE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
INTCON	GIE/GIEH	PEIE/GIEL	IPEN	—	—	INT2EDG	INT1EDG	INT0EDG	170
IOCxF	IOCxF7	IOCxF6	IOCxF5	IOCxF4	IOCxF3	IOCxF2	IOCxF1	IOCxF0	211
IOCxN	IOCxN7	IOCxN6	IOCxN5	IOCxN4	IOCxN3	IOCxN2	IOCxN1	IOCxN0	211
IOCxP	IOCxP7	IOCxP6	IOCxP5	IOCxP4	IOCxP3	IOCxP2	IOCxP1	IOCxP0	211

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by interrupt-on-change.

REGISTER 19-3: TMRxCLK: TIMERx CLOCK REGISTER

U-0	U-0	U-0	U-0	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u
—	—	—	—	CS<3:0>			
bit 7							bit 0

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

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

bit 3-0 **CS<3:0>:** Timerx Clock Source Selection bits

CS	Timer1	Timer3	Timer5
	Clock Source	**Clock Source**	**Clock Source**
1111-1100	Reserved	Reserved	Reserved
1011	TMR5 overflow	TMR5 overflow	Reserved
1010	TMR3 overflow	Reserved	TMR3 overflow
1001	Reserved	TMR1 overflow	TMR1 overflow
1000	TMR0 overflow	TMR0 overflow	TMR0 overflow
0111	CLKREF	CLKREF	CLKREF
0110	SOSC	SOSC	SOSC
0101	MFINTOSC (500 kHz)	MFINTOSC (500 kHz)	MFINTOSC (500 kHz)
0100	LFINTOSC	LFINTOSC	LFINTOSC
0011	HFINTOSC	HFINTOSC	HFINTOSC
0010	Fosc	Fosc	Fosc
0001	Fosc/4	Fosc/4	Fosc/4
0000	T1CKIPPS	T3CKIPPS	T5CKIPPS

20.5.2 HARDWARE GATE MODE

The Hardware Gate modes operate the same as the Software Gate mode except the TMRx_ers external signal can also gate the timer. When used with the CCP the gating extends the PWM period. If the timer is stopped when the PWM output is high then the duty cycle is also extended.

When MODE<4:0> = 00001 then the timer is stopped when the external signal is high. When MODE<4:0> = 00010 then the timer is stopped when the external signal is low.

Figure 20-5 illustrates the Hardware Gating mode for MODE<4:0> = 00001 in which a high input level starts the counter.

FIGURE 20-5: HARDWARE GATE MODE TIMING DIAGRAM (MODE = 00001)

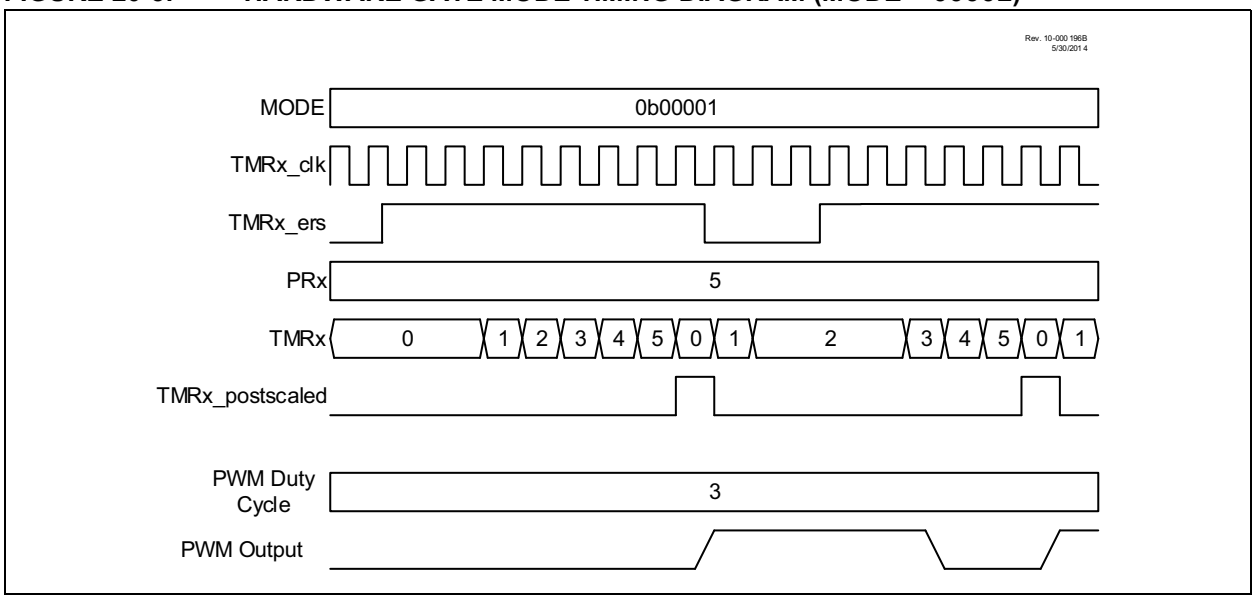
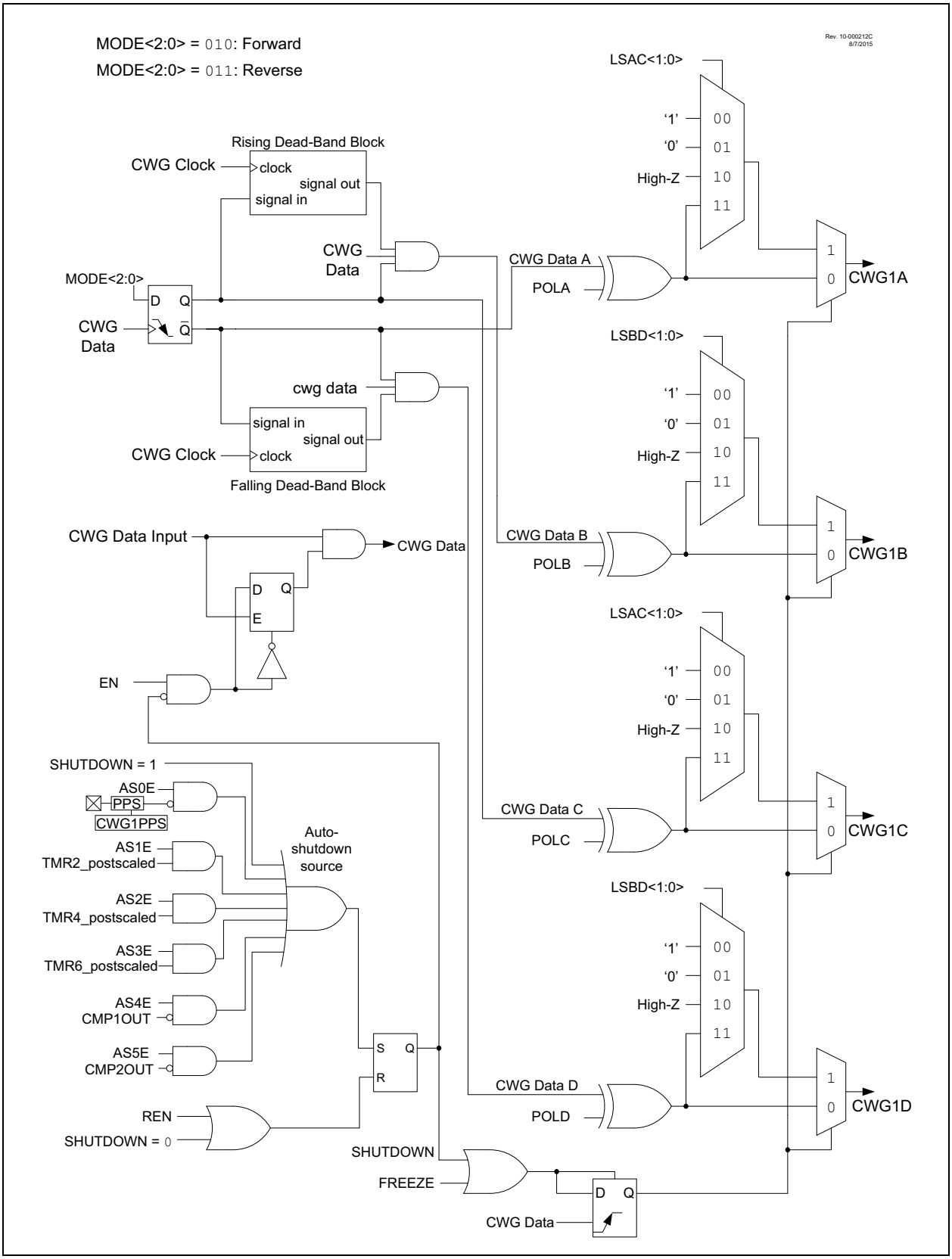


FIGURE 24-6: SIMPLIFIED CWG BLOCK DIAGRAM (FORWARD AND REVERSE FULL-BRIDGE MODES)



REGISTER 26-7: SSPxCON1: MSSPx CONTROL REGISTER 1 (I^2C MASTER MODE)

R/W-0	R/W/HC-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN(1)	CKP	SSPM3(2)	SSPM2(2)	SSPM1(2)	SSPM0(2)
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	HC = Bit is cleared by hardware	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **WCOL:** Write Collision Detect bit

In Master Transmit mode:
1 = A write to the SSPxBUF register was attempted while the I^2C conditions were not valid for a transmission to be started (must be cleared in software)
0 = No collision

In Slave Transmit mode:
1 = The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

In Receive mode (Master or Slave modes):
This is a "don't care" bit.

bit 6 **SSPOV:** Receive Overflow Indicator bit

In Receive mode:
1 = A byte is received while the SSPxBUF register is still holding the previous byte (must be cleared in software)
0 = No overflow

In Transmit mode:
This is a "don't care" bit in Transmit mode.

bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit(1)

1 = Enables the serial port and configures the SDAx and SCLx pins as the serial port pins
0 = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** SCKx Release Control bit

In Slave mode:
1 = Releases clock
0 = Holds clock low (clock stretch), used to ensure data setup time

In Master mode:
Unused in this mode.

bit 3-0 **SSPM<3:0>:** Master Synchronous Serial Port Mode Select bits(2)

1111 = I^2C Slave mode: 10-bit address with Start and Stop bit interrupts enabled
1110 = I^2C Slave mode: 7-bit address with Start and Stop bit interrupts enabled
1011 = I^2C Firmware Controlled Master mode (slave Idle)
1000 = I^2C Master mode: Clock = FOSC/(4 * (SSPxADD + 1))
0111 = I^2C Slave mode: 10-bit address(3,4)
0110 = I^2C Slave mode: 7-bit address

Note 1: When enabled, the SDAx and SCLx pins must be configured as inputs.
2: Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.

FIGURE 26-34: BUS COLLISION DURING START CONDITION (SCL = 0)

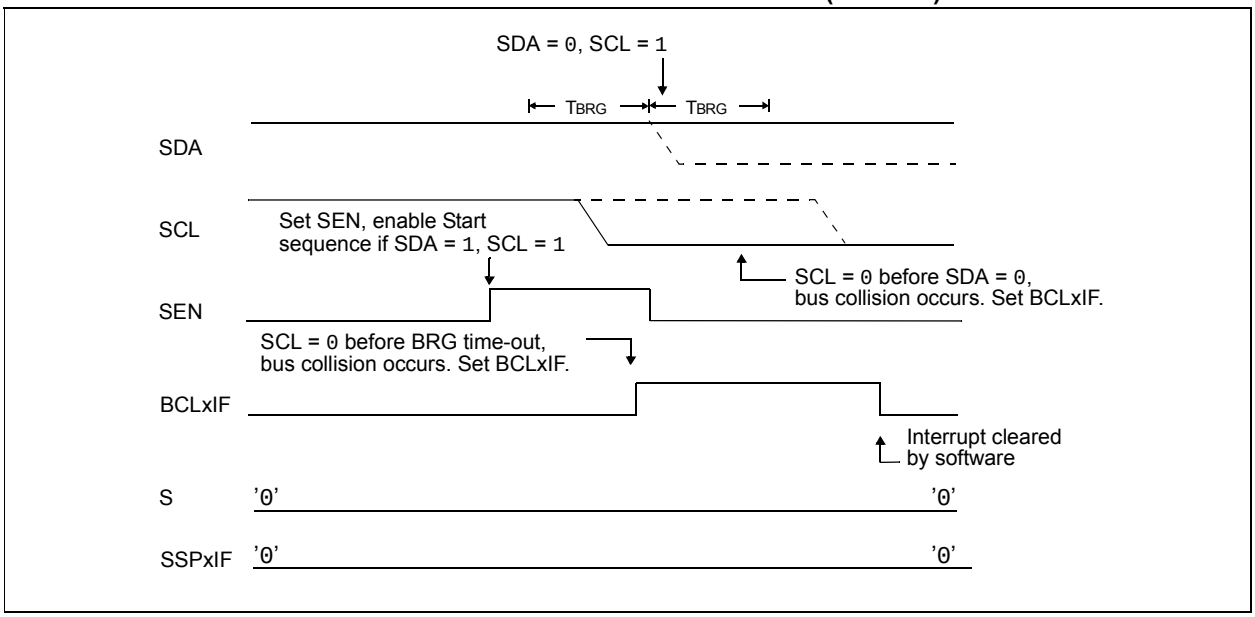
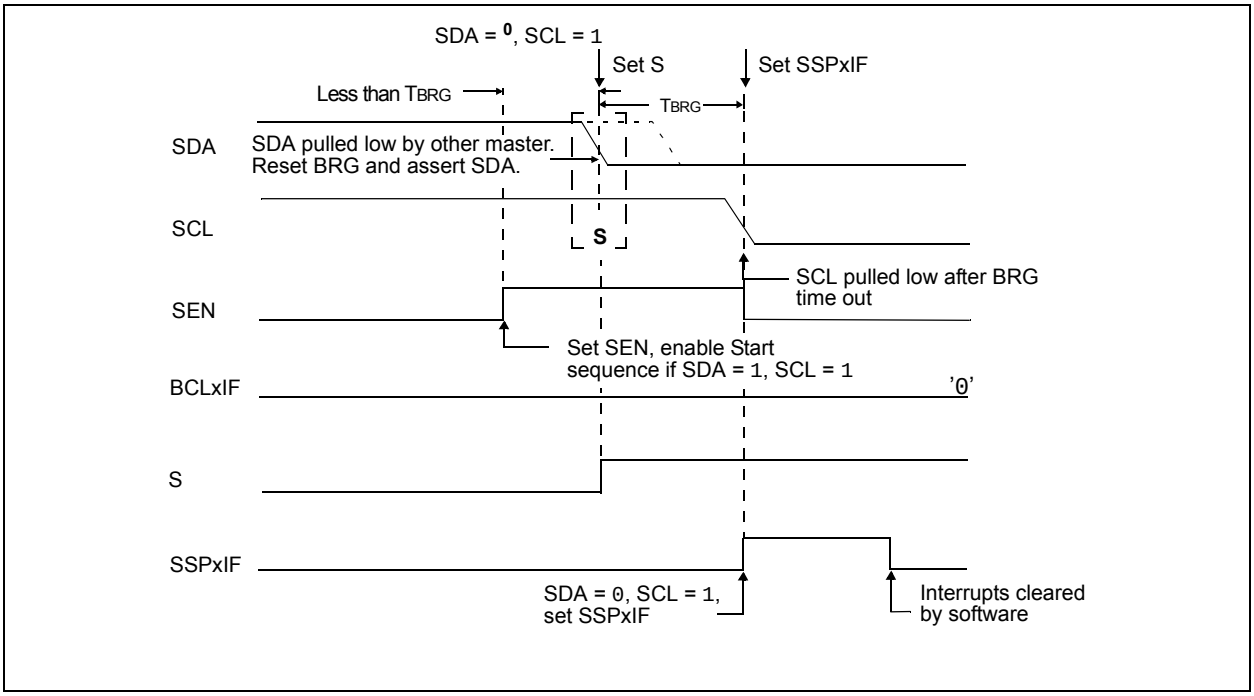


FIGURE 26-35: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION



31.3 ADC Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The Analog Input model is shown in Figure 31-4. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), refer to Figure 31-4. **The maximum recommended impedance for analog sources is 10 kΩ.** As the source impedance is decreased, the acquisition time may be decreased. After the analog input channel is selected (or changed), an ADC acquisition must be completed before the conversion can be started. To calculate the minimum acquisition time, Equation 31-1 may be used. This equation assumes that 1/2 LSb error is used (1,024 steps for the ADC). The 1/2 LSb error is the maximum error allowed for the ADC to meet its specified resolution.

EQUATION 31-1: ACQUISITION TIME EXAMPLE

Assumptions: $Temperature = 50°C$ *and external impedance of* $10k\Omega$ *5.0V* V_{DD}

$$T_{ACQ} = \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient}$$
$$= T_{AMP} + T_C + T_{COFF}$$
$$= 2\mu s + T_C + [(\text{Temperature} - 25°C)(0.05\mu s/°C)]$$

The value for T_C *can be approximated with the following equations:*

$$V_{APPLIED}\left(1 - \frac{1}{(2^{n+1}) - 1}\right) = V_{CHOLD} \quad ;[1]\ V_{CHOLD}\ \text{charged to within 1/2 lsb}$$

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{CHOLD} \quad ;[2]\ V_{CHOLD}\ \text{charge response to}\ V_{APPLIED}$$

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{APPLIED}\left(1 - \frac{1}{(2^{n+1}) - 1}\right) \quad ;\text{combining [1] and [2]}$$

Note: Where n = number of bits of the ADC.

Solving for T_C*:*

$$T_C = -C_{HOLD}(R_{IC} + R_{SS} + R_S)\ ln(1/2047)$$
$$= -10pF(1k\Omega + 7k\Omega + 10k\Omega)\ ln(0.0004885)$$
$$= 1.37\mu s$$

Therefore:

$$T_{ACQ} = 2\mu s + 892ns + [(50°C - 25°C)(0.05\mu s/°C)]$$
$$= 4.62\mu s$$

Note 1: The reference voltage (VREF) has no effect on the equation, since it cancels itself out.

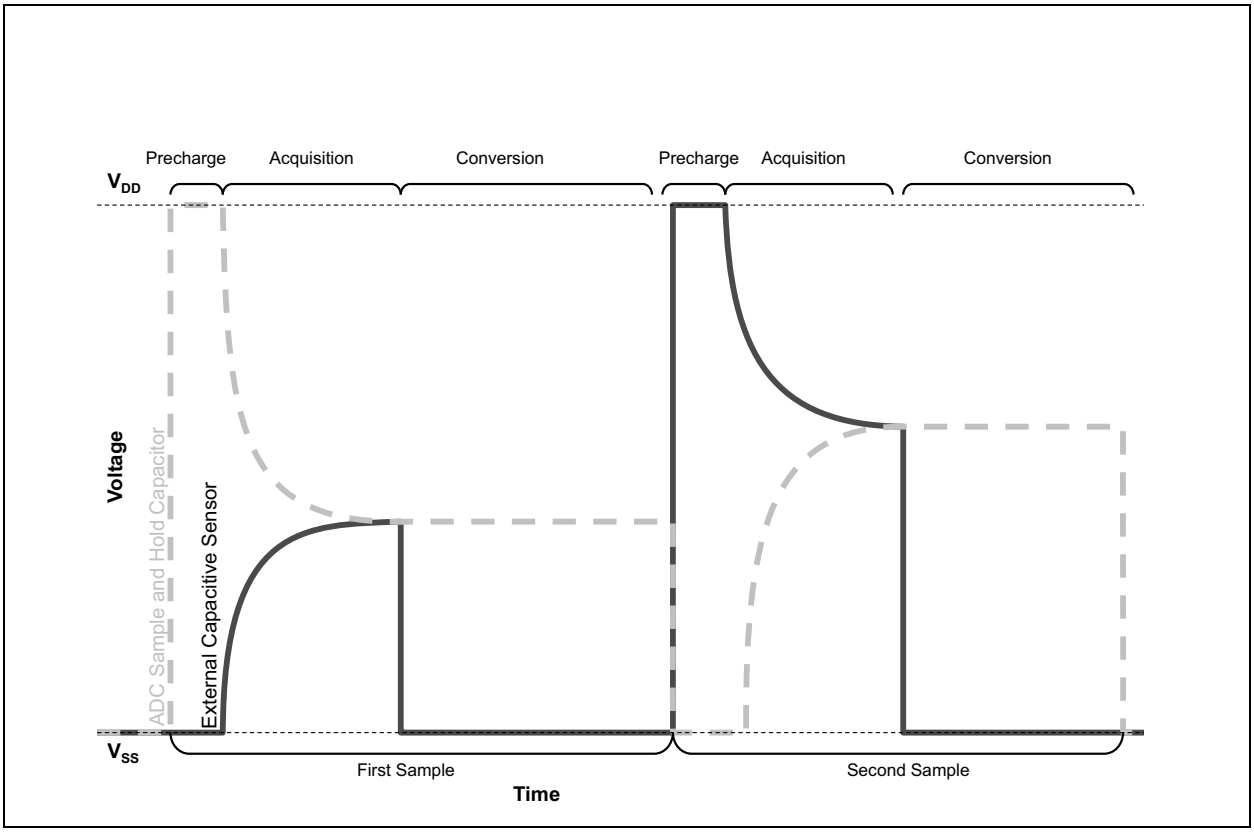
2: The charge holding capacitor (CHOLD) is not discharged after each conversion.

3: The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

31.4.1 CVD OPERATION

A CVD operation begins with the ADC's internal sample and hold capacitor (C_{HOLD}) being disconnected from the path which connects it to the external capacitive sensor node. While disconnected, C_{HOLD} is precharged to VDD or VSS, while the path to the sensor node is also discharged to VDD or VSS. Typically, this node is discharged to the level opposite that of CHOLD. When the precharge phase is complete, the VDD/VSS bias paths for the two nodes are shut off and CHOLD and the path to the external sensor node are re-connected, at which time the acquisition phase of the CVD operation begins. During acquisition, a capacitive voltage divider is formed between the precharged CHOLD and sensor nodes, which results in a final voltage level setting on CHOLD, which is determined by the capacitances and precharge levels of the two nodes. After acquisition, the ADC converts the voltage level on CHOLD. This process is then repeated with the selected precharge levels for both the CHOLD and the inverted sensor nodes. Figure 31-7 shows the waveform for two inverted CVD measurements, which is known as differential CVD measurement.

FIGURE 31-7: DIFFERENTIAL CVD MEASUREMENT WAVEFORM



REGISTER 31-19: ADRESL: ADC RESULT REGISTER LOW, ADFM = 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
ADRES<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 **ADRES<7:0>**: ADC Result Register bits. Lower eight bits of 10-bit conversion result.

REGISTER 31-20: ADPREVH: ADC PREVIOUS RESULT REGISTER

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
ADPREV<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 **ADPREV<15:8>**: Previous ADC Results bits
If ADPSIS = 1:
Upper byte of ADFLTR at the start of current ADC conversion
If ADPSIS = 0:
Upper bits of ADRES at the start of current ADC conversion(1)

Note 1: If ADPSIS = 0, ADPREVH and ADPREVL are formatted the same way as ADRES is, depending on the ADFM bit.

32.2 Register Definitions: Comparator Control

Long bit name prefixes for the Comparators are shown in Table 32-1. Refer to **Section 1.4.2.2 "Long Bit Names"** for more information.

TABLE 32-1:

Peripheral	Bit Name Prefix
C1	C1
C2	C2

REGISTER 32-1: CMxCON0: COMPARATOR x CONTROL REGISTER 0

R/W-0/0	R-0/0	U-0	R/W-0/0	U-0	U-1	R/W-0/0	R/W-0/0
EN	OUT	—	POL	—	—	HYS	SYNC
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 **EN:** Comparator Enable bit
- 1 = Comparator is enabled
- 0 = Comparator is disabled and consumes no active power

bit 6 **OUT:** Comparator Output bit

If POL = 0 (non-inverted polarity):
- 1 = CxVP > CxVN
- 0 = CxVP < CxVN

If POL = 1 (inverted polarity):
- 1 = CxVP < CxVN
- 0 = CxVP > CxVN

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

bit 4 **POL:** Comparator Output Polarity Select bit
- 1 = Comparator output is inverted
- 0 = Comparator output is not inverted

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

bit 2 **Unimplemented:** Read as '1'

bit 1 **HYS:** Comparator Hysteresis Enable bit
- 1 = Comparator hysteresis enabled
- 0 = Comparator hysteresis disabled

bit 0 **SYNC:** Comparator Output Synchronous Mode bit
- 1 = Comparator output to Timer1/3/5 and I/O pin is synchronous to changes on Timer1 clock source.
- 0 = Comparator output to Timer1/3/5 and I/O pin is asynchronous
  Output updated on the falling edge of Timer1/3/5 clock source.

TABLE 35-2: INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
BIT-ORIENTED OPERATIONS									
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4
BTG	f, b, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2
CONTROL OPERATIONS									
BC	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None	
BN	n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None	
BNC	n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None	
BNN	n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None	
BNOV	n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None	
BNZ	n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None	
BRA	n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	
CALL	k, s	Call subroutine 1st word	2	1110	110s	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
CLRWDT	—	Clear Watchdog Timer	1	0000	0000	0000	0100	$\overline{TO}$, $\overline{PD}$	
DAW	—	Decimal Adjust WREG	1	0000	0000	0000	0111	C	
GOTO	k	Go to address 1st word	2	1110	1111	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
NOP	—	No Operation	1	0000	0000	0000	0000	None	
NOP	—	No Operation	1	1111	xxxx	xxxx	xxxx	None	4
POP	—	Pop top of return stack (TOS)	1	0000	0000	0000	0110	None	
PUSH	—	Push top of return stack (TOS)	1	0000	0000	0000	0101	None	
RCALL	n	Relative Call	2	1101	1nnn	nnnn	nnnn	None	
RESET		Software device Reset	1	0000	0000	1111	1111	All	
RETFIE	s	Return from interrupt enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
RETURN	s	Return from Subroutine	2	0000	0000	0001	001s	None	
SLEEP	—	Go into Standby mode	1	0000	0000	0000	0011	$\overline{TO}$, $\overline{PD}$	

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

2: If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

4: Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

ANDWF **AND W with f**

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

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

Operation: (W) .AND. (f) → dest

Status Affected: N, Z

Encoding:

0001	01da	ffff	ffff

Description: The contents of W are AND'ed with 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 ≤ 95 (5Fh). See **Section 35.2.3 "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: `ANDWF REG, 0, 0`

Before Instruction
W = 17h
REG = C2h
After Instruction
W = 02h
REG = C2h

BC **Branch if Carry**

Syntax: BC n

Operands: $-128 \le n \le 127$

Operation: if CARRY bit is '1'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0010	nnnn	nnnn

Description: If the CARRY bit is '1', 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    BC   5
```
Before Instruction
PC = address (HERE)
After Instruction
If CARRY = 1;
PC = address (HERE + 12)
If CARRY = 0;
PC = address (HERE + 2)

DAW **Decimal Adjust W Register**

Syntax: DAW

Operands: None

Operation: If [W<3:0> > 9] or [DC = 1] then
(W<3:0>) + 6 → W<3:0>;
else
(W<3:0>) → W<3:0>;

If [W<7:4> + DC > 9] or [C = 1] then
(W<7:4>) + 6 + DC → W<7:4> ;
else
(W<7:4>) + DC → W<7:4>

Status Affected: C

Encoding:

0000	0000	0000	0111

Description: DAW adjusts the 8-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register W	Process Data	Write W

Example1:

```
DAW
```

Before Instruction
W = A5h
C = 0
DC = 0
After Instruction
W = 05h
C = 1
DC = 0

Example 2:

Before Instruction
W = CEh
C = 0
DC = 0
After Instruction
W = 34h
C = 1
DC = 0

DECF **Decrement f**

Syntax: DECF 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:

0000	01da	ffff	ffff

Description: Decrement 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 ≤ 95 (5Fh). See **Section 35.2.3 "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: DECF CNT, 1, 0

Before Instruction
CNT = 01h
Z = 0
After Instruction
CNT = 00h
Z = 1

TABLE 37-2: SUPPLY CURRENT (I_{DD})(1,2,4)

PIC18LF27/47K40			Standard Operating Conditions (unless otherwise stated)					
PIC18F27/47K40								
Param. No.	Symbol	Device Characteristics	Min.	Typ.†	Max.	Units	Conditions	
							V_{DD}	Note
D100	$I_{DD_{XT4}}$	XT = 4 MHz	—	450	650	µA	3.0V	
D100	$I_{DD_{XT4}}$	XT = 4 MHz	—	550	750	µA	3.0V	
D100A	$I_{DD_{XT4}}$	XT = 4 MHz	—	310	—	µA	3.0V	PMD's all 1's
D100A	$I_{DD_{XT4}}$	XT = 4 MHz	—	410	—	µA	3.0V	PMD's all 1's
D101	$I_{DD_{HFO16}}$	HFINTOSC = 16 MHz	—	1.9	2.6	mA	3.0V	
D101	$I_{DD_{HFO16}}$	HFINTOSC = 16 MHz	—	2.0	2.7	mA	3.0V	
D101A	$I_{DD_{HFO16}}$	HFINTOSC = 16 MHz	—	1.4	—	mA	3.0V	PMD's all 1's
D101A	$I_{DD_{HFO16}}$	HFINTOSC = 16 MHz	—	1.5	—	mA	3.0V	PMD's all 1's
D102	$I_{DD_{HFOPLL}}$	HFINTOSC = 64 MHz	—	7.4	9.4	mA	3.0V	
D102	$I_{DD_{HFOPLL}}$	HFINTOSC = 64 MHz	—	7.5	9.5	mA	3.0V	
D102A	$I_{DD_{HFOPLL}}$	HFINTOSC = 64 MHz	—	5.2	—	mA	3.0V	PMD's all 1's
D102A	$I_{DD_{HFOPLL}}$	HFINTOSC = 64 MHz	—	5.3	—	mA	3.0V	PMD's all 1's
D103	$I_{DD_{HSPLL32}}$	HS+PLL = 64 MHz	—	6.9	8.9	mA	3.0V	
D103	$I_{DD_{HSPLL32}}$	HS+PLL = 64 MHz	—	7.0	9.0	mA	3.0V	
D103A	$I_{DD_{HSPLL32}}$	HS+PLL = 64 MHz	—	4.9	—	mA	3.0V	PMD's all 1's
D103A	$I_{DD_{HSPLL32}}$	HS+PLL = 64 MHz	—	5.0	—	mA	3.0V	PMD's all 1's
D104	$I_{DD_{IDLE}}$	IDLE mode, HFINTOSC = 16 MHz	—	1.05	—	mA	3.0V	
D104	$I_{DD_{IDLE}}$	IDLE mode, HFINTOSC = 16 MHz	—	1.15	—	mA	3.0V	
D105	$I_{DD_{DOZE}}$(3)	DOZE mode, HFINTOSC = 16 MHz, Doze Ratio = 16	—	1.1	—	mA	3.0V	
D105	$I_{DD_{DOZE}}$(3)	DOZE mode, HFINTOSC = 16 MHz, Doze Ratio = 16	—	1.2	—	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 6-2).

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