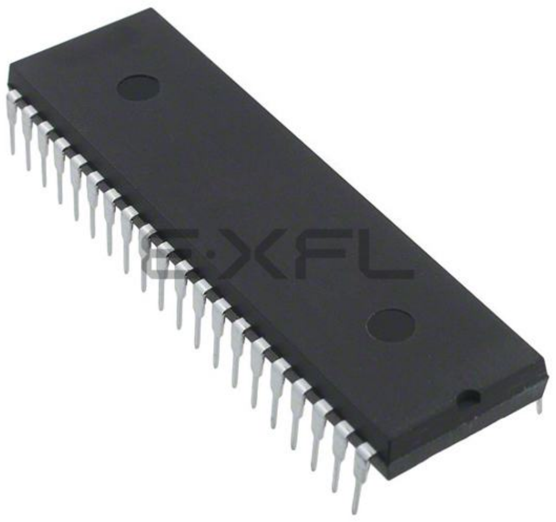
Microchip Technology - PIC18F4620-E/P Datasheet



Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	25MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.8K x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 13x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Through Hole
Package / Case	40-DIP (0.600", 15.24mm)
Supplier Device Package	40-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f4620-e-p

TABLE 1-3: PIC18F4525/4620 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
$\overline{\text{MCLR}}$/VPP/RE3	1	18	18			Master Clear (input) or programming voltage (input).
$\overline{\text{MCLR}}$				I	ST	Master Clear (Reset) input. This pin is an active-low Reset to the device.
VPP				P		Programming voltage input.
RE3				I	ST	Digital input.
OSC1/CLKI/RA7	13	32	30			Oscillator crystal or external clock input.
OSC1				I	ST	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; analog otherwise.
CLKI				I	CMOS	External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)
RA7				I/O	TTL	General purpose I/O pin.
OSC2/CLKO/RA6	14	33	31			Oscillator crystal or clock output.
OSC2				O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO				O	—	In RC mode, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6				I/O	TTL	General purpose I/O pin.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels I = Input
O = Output P = Power

Note 1: Default assignment for CCP2 when the CCP2MX Configuration bit is set.
2: Alternate assignment for CCP2 when the CCP2MX Configuration bit is cleared.
3: For the QFN package, it is recommended that the bottom pad be connected to Vss.

TABLE 4-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS

Register	Applicable Devices				Power-on Reset, Brown-out Reset	$\overline{\text{MCLR}}$ Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
TOSU	2525	2620	4525	4620	`---0 0000`	`---0 0000`	`---0 uuuu`(3)
TOSH	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`(3)
TOSL	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`(3)
STKPTR	2525	2620	4525	4620	`00-0 0000`	`uu-0 0000`	`uu-u uuuu`(3)
PCLATU	2525	2620	4525	4620	`---0 0000`	`---0 0000`	`---u uuuu`
PCLATH	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
PCL	2525	2620	4525	4620	`0000 0000`	`0000 0000`	PC + 2(2)
TBLPTRU	2525	2620	4525	4620	`--00 0000`	`--00 0000`	`--uu uuuu`
TBLPTRH	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
TBLPTRL	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
TABLAT	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
PRODH	2525	2620	4525	4620	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
PRODL	2525	2620	4525	4620	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
INTCON	2525	2620	4525	4620	`0000 000x`	`0000 000u`	`uuuu uuuu`(1)
INTCON2	2525	2620	4525	4620	`1111 -1-1`	`1111 -1-1`	`uuuu -u-u`(1)
INTCON3	2525	2620	4525	4620	`11-0 0-00`	`11-0 0-00`	`uu-u u-uu`(1)
INDF0	2525	2620	4525	4620	N/A	N/A	N/A
POSTINC0	2525	2620	4525	4620	N/A	N/A	N/A
POSTDEC0	2525	2620	4525	4620	N/A	N/A	N/A
PREINC0	2525	2620	4525	4620	N/A	N/A	N/A
PLUSW0	2525	2620	4525	4620	N/A	N/A	N/A
FSR0H	2525	2620	4525	4620	`---- 0000`	`---- 0000`	`---- uuuu`
FSR0L	2525	2620	4525	4620	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
WREG	2525	2620	4525	4620	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
INDF1	2525	2620	4525	4620	N/A	N/A	N/A
POSTINC1	2525	2620	4525	4620	N/A	N/A	N/A
POSTDEC1	2525	2620	4525	4620	N/A	N/A	N/A
PREINC1	2525	2620	4525	4620	N/A	N/A	N/A
PLUSW1	2525	2620	4525	4620	N/A	N/A	N/A

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 4-3 for Reset value for specific condition.

5: Bits 6 and 7 of PORTA, LATA and TRISA are enabled, depending on the oscillator mode selected. When not enabled as PORTA pins, they are disabled and read '0'.

TABLE 4-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices				Power-on Reset, Brown-out Reset	$\overline{\text{MCLR}}$ Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
ADRESH	2525	2620	4525	4620	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
ADRESL	2525	2620	4525	4620	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
ADCON0	2525	2620	4525	4620	`--00 0000`	`--00 0000`	`--uu uuuu`
ADCON1	2525	2620	4525	4620	`--00 0qqq`	`--00 0qqq`	`--uu uuuu`
ADCON2	2525	2620	4525	4620	`0-00 0000`	`0-00 0000`	`u-uu uuuu`
CCPR1H	2525	2620	4525	4620	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCPR1L	2525	2620	4525	4620	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCP1CON	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
	2525	2620	4525	4620	`--00 0000`	`--00 0000`	`--uu uuuu`
CCPR2H	2525	2620	4525	4620	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCPR2L	2525	2620	4525	4620	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCP2CON	2525	2620	4525	4620	`--00 0000`	`--00 0000`	`--uu uuuu`
BAUDCON	2525	2620	4525	4620	`0100 0-00`	`0100 0-00`	`uuuu u-uu`
PWM1CON	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
ECCP1AS	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
	2525	2620	4525	4620	`0000 00--`	`0000 00--`	`uuuu uu--`
CVRCON	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
CMCON	2525	2620	4525	4620	`0000 0111`	`0000 0111`	`uuuu uuuu`
TMR3H	2525	2620	4525	4620	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
TMR3L	2525	2620	4525	4620	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
T3CON	2525	2620	4525	4620	`0000 0000`	`uuuu uuuu`	`uuuu uuuu`
SPBRGH	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
SPBRG	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
RCREG	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
TXREG	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
TXSTA	2525	2620	4525	4620	`0000 0010`	`0000 0010`	`uuuu uuuu`
RCSTA	2525	2620	4525	4620	`0000 000x`	`0000 000x`	`uuuu uuuu`
EEADRH	2585	2680	4585	4680	`---- --00`	`---- --00`	`---- --uu`
EEADR	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
EEDATA	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
EECON2	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`0000 0000`
EECON1	2525	2620	4525	4620	`xx-0 x000`	`uu-0 u000`	`uu-0 u000`

Legend: u = unchanged, x = unknown, `-` = unimplemented bit, read as '`0`', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 4-3 for Reset value for specific condition.

5: Bits 6 and 7 of PORTA, LATA and TRISA are enabled, depending on the oscillator mode selected. When not enabled as PORTA pins, they are disabled and read '`0`'.

5.1.2.4 Stack Full 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 will set the appropriate STKFUL or STKUNF bit and then cause a device Reset. When STVREN is cleared, a full or underflow condition will set the appropriate STKFUL or STKUNF bit but not cause a device Reset. The STKFUL or STKUNF bits are cleared by the user software or a Power-on Reset.

5.1.3 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 in 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 5-1 shows a source code example that uses the Fast Register Stack during a subroutine call and return.

EXAMPLE 5-1: 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
```

5.1.4 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

5.1.4.1 Computed `GOTO`

A computed `GOTO` is accomplished by adding an offset to the program counter. An example is shown in Example 5-2.

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 2 (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 5-2: COMPUTED `GOTO` USING AN OFFSET VALUE

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

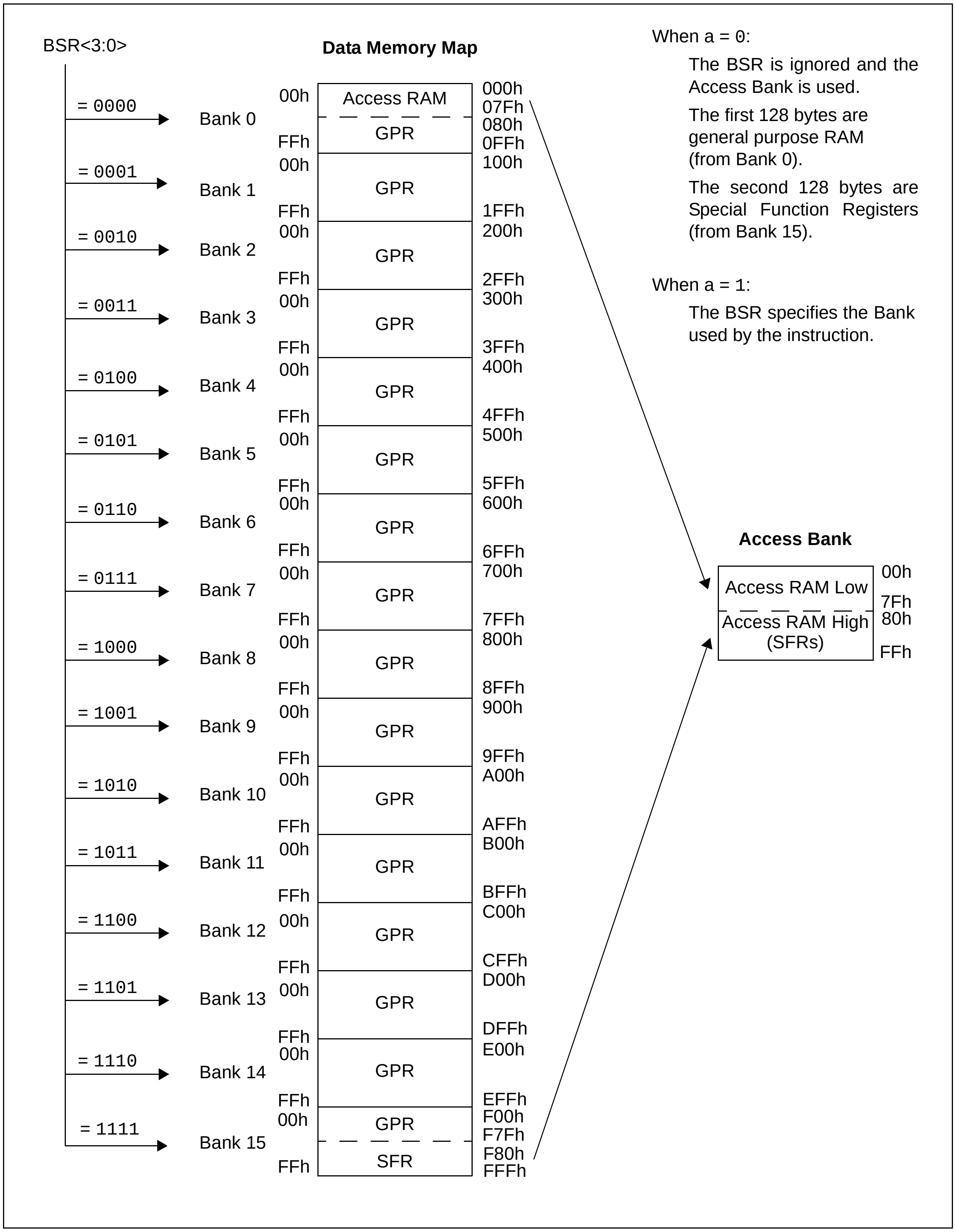
5.1.4.2 Table Reads and Table Writes

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 7.1 "Table Reads and Table Writes"**.

FIGURE 5-5: DATA MEMORY MAP FOR PIC18F2525/2620/4525/4620 DEVICES



When a = 0:

The BSR is ignored and the Access Bank is used.

The first 128 bytes are general purpose RAM (from Bank 0).

The second 128 bytes are Special Function Registers (from Bank 15).

When a = 1:

The BSR specifies the Bank used by the instruction.

7.0 FLASH PROGRAM MEMORY

The Flash program memory is readable, writable and erasable during normal operation over the entire VDD range.

A read from program memory is executed on one byte at a time. A write to program memory is executed on blocks of 64 bytes at a time. Program memory is erased in blocks of 64 bytes at a time. A bulk erase operation may not be issued from user code.

Writing or erasing program memory will cease instruction fetches until the operation is complete. The program memory cannot be accessed during the write or erase, therefore, code cannot execute. An internal programming timer terminates program memory writes and erases.

A value written to program memory does not need to be a valid instruction. Executing a program memory location that forms an invalid instruction results in a `NOP`.

7.1 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`)

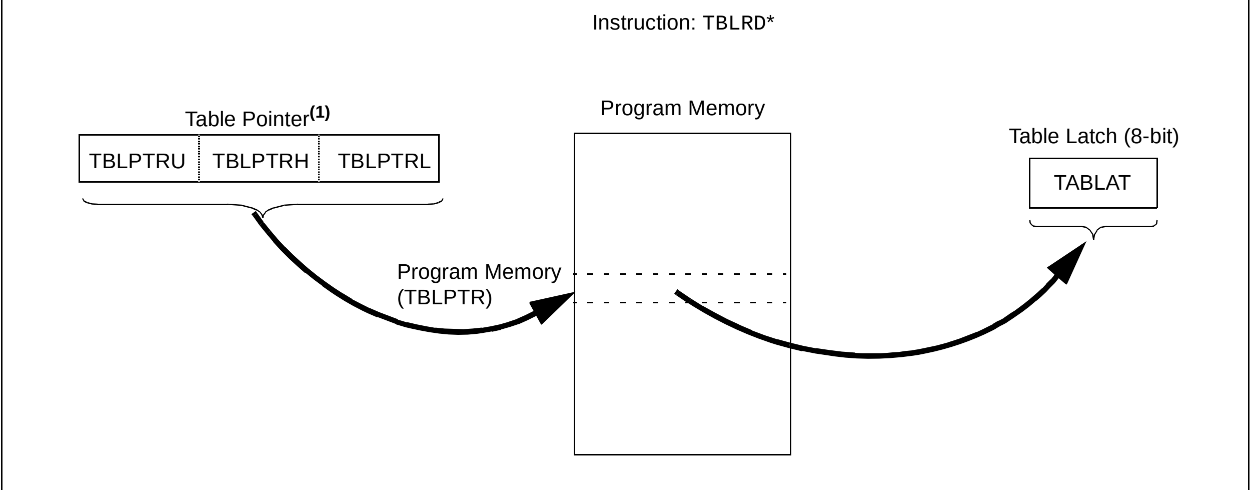
The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

Table read operations retrieve data from program memory and place it into the data RAM space. Figure 7-1 shows the operation of a table read with program memory and data RAM.

Table write operations store data from the data memory space into holding registers in program memory. The procedure to write the contents of the holding registers into program memory is detailed in **Section 7.5 "Writing to Flash Program Memory"**. Figure 7-2 shows the operation of a table write with program memory and data RAM.

Table operations work with byte entities. A table block containing data, rather than program instructions, is not required to be word aligned. Therefore, a table block 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 7-1: TABLE READ OPERATION



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

TABLE 9-5: PORTC I/O SUMMARY

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RC0/T1OSO/ T13CKI	RC0	0	O	DIG	LATC<0> data output.
		1	I	ST	PORTC<0> data input.
	T1OSO	x	O	ANA	Timer1 oscillator output; enabled when Timer1 oscillator enabled. Disables digital I/O.
	T13CKI	1	I	ST	Timer1/Timer3 counter input.
RC1/T1OSI/CCP2	RC1	0	O	DIG	LATC<1> data output.
		1	I	ST	PORTC<1> data input.
	T1OSI	x	I	ANA	Timer1 oscillator input; enabled when Timer1 oscillator enabled. Disables digital I/O.
	CCP2(1)	0	O	DIG	CCP2 compare and PWM output; takes priority over port data.
		1	I	ST	CCP2 capture input.
RC2/CCP1/P1A	RC2	0	O	DIG	LATC<2> data output.
		1	I	ST	PORTC<2> data input.
	CCP1	0	O	DIG	ECCP1 compare or PWM output; takes priority over port data.
		1	I	ST	ECCP1 capture input.
	P1A(2)	0	O	DIG	ECCP1 Enhanced PWM output, channel A. May be configured for tri-state during Enhanced PWM shutdown events. Takes priority over port data.
RC3/SCK/SCL	RC3	0	O	DIG	LATC<3> data output.
		1	I	ST	PORTC<3> data input.
	SCK	0	O	DIG	SPI clock output (MSSP module); takes priority over port data.
		1	I	ST	SPI clock input (MSSP module).
	SCL	0	O	DIG	I^2C™ clock output (MSSP module); takes priority over port data.
		1	I	I^2C/SMB	I^2C clock input (MSSP module); input type depends on module setting.
RC4/SDI/SDA	RC4	0	O	DIG	LATC<4> data output.
		1	I	ST	PORTC<4> data input.
	SDI	1	I	ST	SPI data input (MSSP module).
	SDA	0	O	DIG	I^2C data output (MSSP module); takes priority over port data.
		1	I	I^2C/SMB	I^2C data input (MSSP module); input type depends on module setting.
RC5/SDO	RC5	0	O	DIG	LATC<5> data output.
		1	I	ST	PORTC<5> data input.
	SDO	0	O	DIG	SPI data output (MSSP module); takes priority over port data.
RC6/TX/CK	RC6	0	O	DIG	LATC<6> data output.
		1	I	ST	PORTC<6> data input.
	TX	0	O	DIG	Asynchronous serial transmit data output (EUSART module); takes priority over port data. User must configure as output.
	CK	0	O	DIG	Synchronous serial clock output (EUSART module); takes priority over port data.
		1	I	ST	Synchronous serial clock input (EUSART module).
RC7/RX/DT	RC7	0	O	DIG	LATC<7> data output.
		1	I	ST	PORTC<7> data input.
	RX	1	I	ST	Asynchronous serial receive data input (EUSART module).
	DT	0	O	DIG	Synchronous serial data output (EUSART module); takes priority over port data.
		1	I	ST	Synchronous serial data input (EUSART module). User must configure as an input.

Legend: DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; I^2C/SMB = I^2C/SMBus input buffer; x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

Note 1: Default assignment for CCP2 when the CCP2MX Configuration bit is set. Alternate assignment is RB3.

2: Enhanced PWM output is available only on PIC18F4525/4620 devices.

12.0 TIMER1 MODULE

The Timer1 timer/counter module incorporates these features:

- Software selectable operation as a 16-bit timer or counter
- Readable and writable 8-bit registers (TMR1H and TMR1L)
- Selectable clock source (internal or external) with device clock or Timer1 oscillator internal options
- Interrupt-on-overflow
- Reset on CCP Special Event Trigger
- Device clock status flag (T1RUN)

A simplified block diagram of the Timer1 module is shown in Figure 12-1. A block diagram of the module's operation in Read/Write mode is shown in Figure 12-2.

The module incorporates its own low-power oscillator to provide an additional clocking option. The Timer1 oscillator can also be used as a low-power clock source for the microcontroller in power-managed operation.

Timer1 can also be used to provide Real-Time Clock (RTC) functionality to applications with only a minimal addition of external components and code overhead.

Timer1 is controlled through the T1CON Control register (Register 12-1). It also contains the Timer1 Oscillator Enable bit (T1OSCEN). Timer1 can be enabled or disabled by setting or clearing control bit, TMR1ON (T1CON<0>).

REGISTER 12-1: T1CON: TIMER1 CONTROL REGISTER

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
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 **RD16:** 16-Bit Read/Write Mode Enable bit
1 = Enables register read/write of TImer1 in one 16-bit operation
0 = Enables register read/write of Timer1 in two 8-bit operations

bit 6 **T1RUN:** Timer1 System Clock Status bit
1 = Device clock is derived from Timer1 oscillator
0 = Device clock is derived from another source

bit 5-4 **T1CKPS1:T1CKPS0:** Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN:** Timer1 Oscillator Enable bit
1 = Timer1 oscillator is enabled
0 = Timer1 oscillator is shut off
The oscillator inverter and feedback resistor are turned off to eliminate power drain.

bit 2 **$\overline{\text{T1SYNC}}$:** Timer1 External Clock Input Synchronization Select bit
When TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS:** Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T13CKI (on the rising edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR1ON:** Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

REGISTER 17-2: SSPCON1: MSSP CONTROL REGISTER 1 (SPI MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV[(1)]	SSPEN[(2)]	CKP	SSPM3[(3)]	SSPM2[(3)]	SSPM1[(3)]	SSPM0[(3)]
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 **WCOL:** Write Collision Detect bit

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

bit 6 **SSPOV:** Receive Overflow Indicator bit[(1)]

SPI Slave mode:
`1` = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. The user must read the SSPBUF, even if only transmitting data, to avoid setting overflow (must be cleared in software).
`0` = No overflow

bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit[(2)]

`1` = Enables serial port and configures SCK, SDO, SDI and $\overline{\text{SS}}$ as serial port pins
`0` = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** Clock Polarity Select bit

`1` = Idle state for clock is a high level
`0` = Idle state for clock is a low level

bit 3-0 **SSPM3:SSPM0:** Master Synchronous Serial Port Mode Select bits[(3)]

`0101` = SPI Slave mode, clock = SCK pin, $\overline{\text{SS}}$ pin control disabled, $\overline{\text{SS}}$ can be used as I/O pin
`0100` = SPI Slave mode, clock = SCK pin, $\overline{\text{SS}}$ pin control enabled
`0011` = SPI Master mode, clock = TMR2 output/2
`0010` = SPI Master mode, clock = FOSC/64
`0001` = SPI Master mode, clock = FOSC/16
`0000` = SPI Master mode, clock = FOSC/4

Note 1: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.
2: When enabled, these pins must be properly configured as input or output.
3: Bit combinations not specifically listed here are either reserved or implemented in I^{2}C™ mode only.

REGISTER 17-4: SSPCON1: MSSP CONTROL REGISTER 1 (I^2C™ MODE)

R/W-0	R/W-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	SSPM2	SSPM1	SSPM0
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 **WCOL:** Write Collision Detect bit

In Master Transmit mode:
`1` = A write to the SSPBUF 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 SSPBUF 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 SSPBUF 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 SDA and SCL pins as the serial port pins
`0` = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** SCK 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 **SSPM3:SSPM0:** 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 * (SSPADD + 1))
`0111` = I^2C Slave mode, 10-bit address
`0110` = I^2C Slave mode, 7-bit address
Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.

Note 1: When enabled, the SDA and SCL pins must be properly configured as inputs or outputs.

17.4.9 I^{2}C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the RSEN bit (SSPCON2<1>) is programmed high and the I^{2}C logic module is in the Idle state. When the RSEN bit is set, the SCL pin is asserted low. When the SCL pin is sampled low, the Baud Rate Generator is loaded with the contents of SSPADD<5:0> and begins counting. The SDA pin is released (brought high) for one Baud Rate Generator count (TBRG). When the Baud Rate Generator times out, if SDA is sampled high, the SCL pin will be deasserted (brought high). When SCL is sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and begins counting. SDA and SCL must be sampled high for one TBRG. This action is then followed by assertion of the SDA pin (SDA = 0) for one TBRG while SCL is high. Following this, the RSEN bit (SSPCON2<1>) will be automatically cleared and the Baud Rate Generator will not be reloaded, leaving the SDA pin held low. As soon as a Start condition is detected on the SDA and SCL pins, the S bit (SSPSTAT<3>) will be set. The SSPIF bit will not be set until the Baud Rate Generator has timed out.

Note 1: If RSEN is programmed while any other event is in progress, it will not take effect.

2: A bus collision during the Repeated Start condition occurs if:

- SDA is sampled low when SCL goes from low-to-high.
- SCL goes low before SDA is asserted low. This may indicate that another master is attempting to transmit a data '1'.

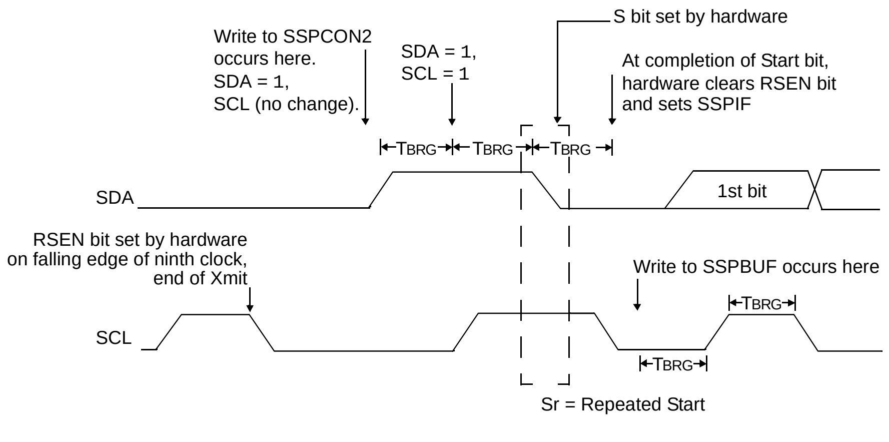
Immediately following the SSPIF bit getting set, the user may write the SSPBUF with the 7-bit address in 7-bit mode, or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional eight bits of address (10-bit mode) or eight bits of data (7-bit mode).

17.4.9.1 WCOL Status Flag

If the user writes the SSPBUF when a Repeated Start sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

Note: Because queueing of events is not allowed, writing of the lower 5 bits of SSPCON2 is disabled until the Repeated Start condition is complete.

FIGURE 17-20: REPEAT START CONDITION WAVEFORM


S bit set by hardware
Write to SSPCON2 occurs here. SDA = 1, SCL (no change).
SDA = 1, SCL = 1
At completion of Start bit, hardware clears RSEN bit and sets SSPIF
TBRG
TBRG
TBRG
SDA
1st bit
RSEN bit set by hardware on falling edge of ninth clock, end of Xmit
Write to SSPBUF occurs here
TBRG
SCL
TBRG
Sr = Repeated Start

REGISTER 18-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN[(1)]	SYNC	SENDB	BRGH	TRMT	TX9D
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 **CSRC:** Clock Source Select bit

Asynchronous mode:
Don’t care.

Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9:** 9-Bit Transmit Enable bit

1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN:** Transmit Enable bit[(1)]

1 = Transmit enabled
0 = Transmit disabled

bit 4 **SYNC:** EUSART Mode Select bit

1 = Synchronous mode
0 = Asynchronous mode

bit 3 **SENDB:** Send Break Character bit

Asynchronous mode:
1 = Send Sync Break on next transmission (cleared by hardware upon completion)
0 = Sync Break transmission completed

Synchronous mode:
Don’t care.

bit 2 **BRGH:** High Baud Rate Select bit

Asynchronous mode:
1 = High speed
0 = Low speed

Synchronous mode:
Unused in this mode.

bit 1 **TRMT:** Transmit Shift Register Status bit

1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th Bit of Transmit Data

Can be address/data bit or a parity bit.

Note 1: SREN/CREN overrides TXEN in Sync mode.

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 two-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 two-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)
```

BTG Bit Toggle f

Syntax:	BTG f, b {,a}
Operands:	0 ≤ f ≤ 255 0 ≤ b < 7 a ∈ [0,1]
Operation:	$(\overline{f<b>}) \rightarrow f<b>$
Status Affected:	None
Encoding:	`0111` `bbba` `ffff` `ffff`
Description:	Bit ‘b’ in data memory location ‘f’ is inverted. 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 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

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

<u>Example:</u> `BTG PORTC, 4, 0`

Before Instruction:
PORTC = `0111 0101` [75h]
After Instruction:
PORTC = `0110 0101` [65h]

BOV Branch if Overflow

Syntax:	BOV n
Operands:	-128 ≤ n ≤ 127
Operation:	if Overflow bit is ‘`1`’, (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	`1110` `0100` `nnnn` `nnnn`
Description:	If the Overflow 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 two-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

<u>Example:</u> `HERE BOV Jump`

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

BZ Branch if Zero

Syntax:	BZ n
Operands:	$-128 \le n \le 127$
Operation:	if Zero bit is '1', (PC) + 2 + 2n → PC
Status Affected:	None

Encoding:

1110	0000	nnnn	nnnn

Description: If the Zero 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 two-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 BZ Jump`

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

CALL Subroutine Call

Syntax:	CALL k {,s}
Operands:	$0 \le k \le 1048575$ $s \in [0,1]$
Operation:	(PC) + 4 → TOS, k → PC<20:1>; if s = 1, (W) → WS, (STATUS) → STATUSS, (BSR) → BSRS
Status Affected:	None

Encoding:

1st word (k<7:0>)	1110	110s	$k_7$kkk	kkkk$_0$
2nd word(k<19:8>)	1111	k_{19}kkk	kkkk	kkkk$_8$

Description: Subroutine call of entire 2-Mbyte memory range. First, return address (PC + 4) is pushed onto the return stack. If 's' = 1, the W, STATUS and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = 0, no update occurs (default). Then, the 20-bit value 'k' is loaded into PC<20:1>. CALL is a two-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	PUSH PC to stack	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example: `HERE CALL THERE, 1`

```
Before Instruction
    PC       =   address (HERE)
After Instruction
    PC       =   address (THERE)
    TOS      =   address (HERE + 4)
    WS       =   W
    BSRS     =   BSR
    STATUSS  =   STATUS
```

CLRF Clear f

Syntax:	CLRF f {,a}
Operands:	$0 \le f \le 255$ $a \in [0,1]$
Operation:	$000h \rightarrow f$, $1 \rightarrow Z$
Status Affected:	Z
Encoding:	`0110` `101a` `ffff` `ffff`
Description:	Clears the contents of the specified register. 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 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `CLRF FLAG_REG, 1`

```
Before Instruction
    FLAG_REG  =  5Ah
After Instruction
    FLAG_REG  =  00h
```

CLRWDT Clear Watchdog Timer

Syntax:	CLRWDT
Operands:	None
Operation:	$000h \rightarrow$ WDT, $000h \rightarrow$ WDT postscaler, $1 \rightarrow \overline{TO}$, $1 \rightarrow \overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Encoding:	`0000` `0000` `0000` `0100`
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits, $\overline{TO}$ and $\overline{PD}$, are set.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	No operation

Example: `CLRWDT`

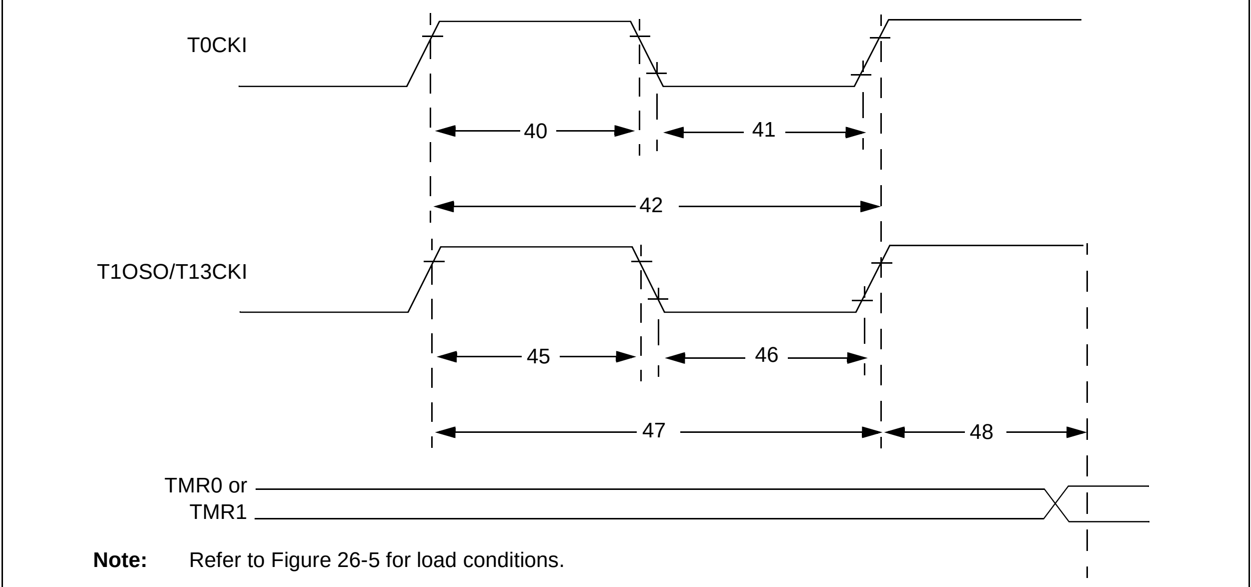
Before Instruction

WDT Counter	=	?

After Instruction

WDT Counter	=	00h
WDT Postscaler	=	0
$\overline{TO}$	=	1
$\overline{PD}$	=	1

FIGURE 26-10: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS



Note: Refer to Figure 26-5 for load conditions.

TABLE 26-11: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

Param No.	Symbol	Characteristic			Min	Max	Units	Conditions
40	Tt0H	T0CKI High Pulse Width	No prescaler		0.5 TCY + 20	—	ns	
			With prescaler		10	—	ns	
41	Tt0L	T0CKI Low Pulse Width	No prescaler		0.5 TCY + 20	—	ns	
			With prescaler		10	—	ns	
42	Tt0P	T0CKI Period	No prescaler		TCY + 10	—	ns	
			With prescaler		Greater of: 20 ns or (TCY + 40)/N	—	ns	N = prescale value (1, 2, 4,..., 256)
45	Tt1H	T13CKI High Time	Synchronous, no prescaler		0.5 TCY + 20	—	ns	
			Synchronous, with prescaler	PIC18**F**XXXX	10	—	ns	
				PIC18**LF**XXXX	25	—	ns	VDD = 2.0V
			Asynchronous	PIC18**F**XXXX	30	—	ns	
				PIC18**LF**XXXX	50	—	ns	VDD = 2.0V
46	Tt1L	T13CKI Low Time	Synchronous, no prescaler		0.5 TCY + 5	—	ns	
			Synchronous, with prescaler	PIC18**F**XXXX	10	—	ns	
				PIC18**LF**XXXX	25	—	ns	VDD = 2.0V
			Asynchronous	PIC18**F**XXXX	30	—	ns	
				PIC18**LF**XXXX	50	—	ns	VDD = 2.0V
47	Tt1P	T13CKI Input Period	Synchronous		Greater of: 20 ns or (TCY + 40)/N	—	ns	N = prescale value (1, 2, 4, 8)
			Asynchronous		60	—	ns	
	Ft1	T13CKI Oscillator Input Frequency Range			DC	50	kHz	
48	Tcke2tmrI	Delay from External T13CKI Clock Edge to Timer Increment			2 TOSC	7 TOSC	—	

FIGURE 27-2: TYPICAL IPD vs. VDD ACROSS TEMPERATURE (SLEEP MODE)

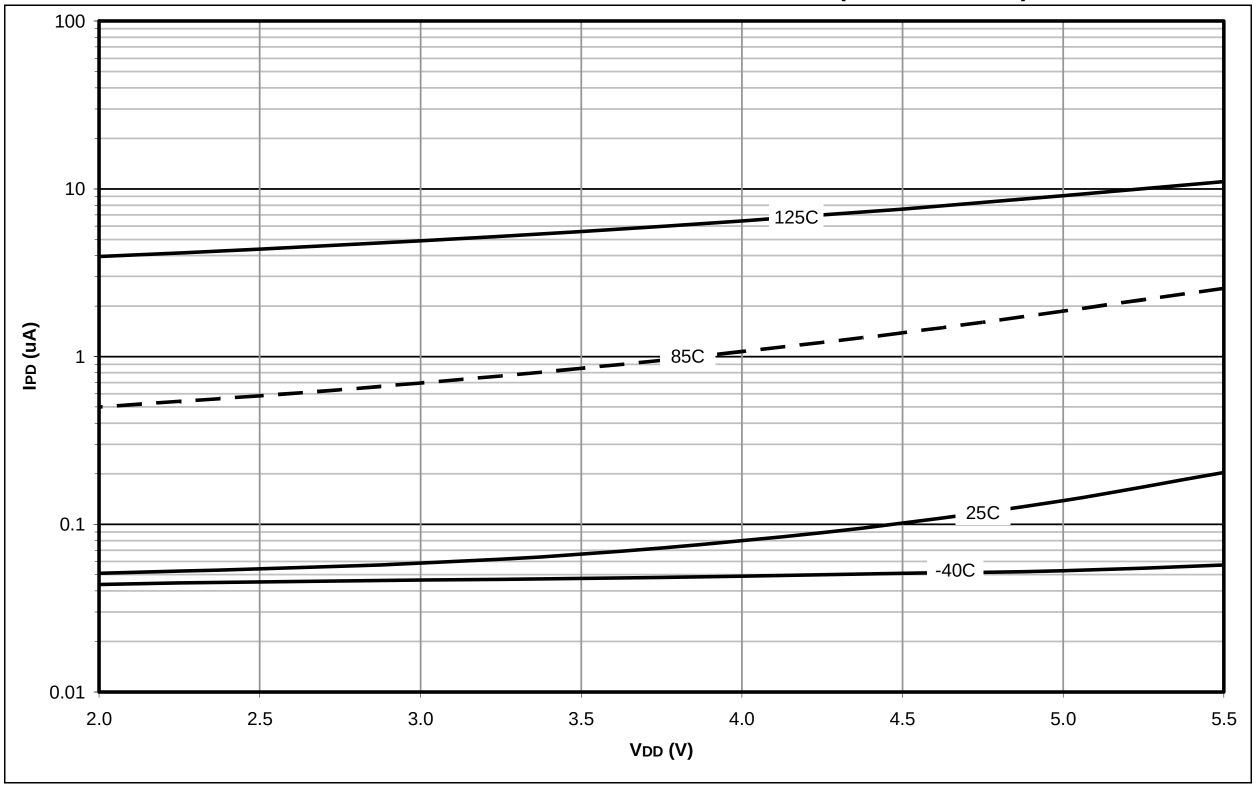


FIGURE 27-3: MAXIMUM IPD vs. VDD ACROSS TEMPERATURE (SLEEP MODE)

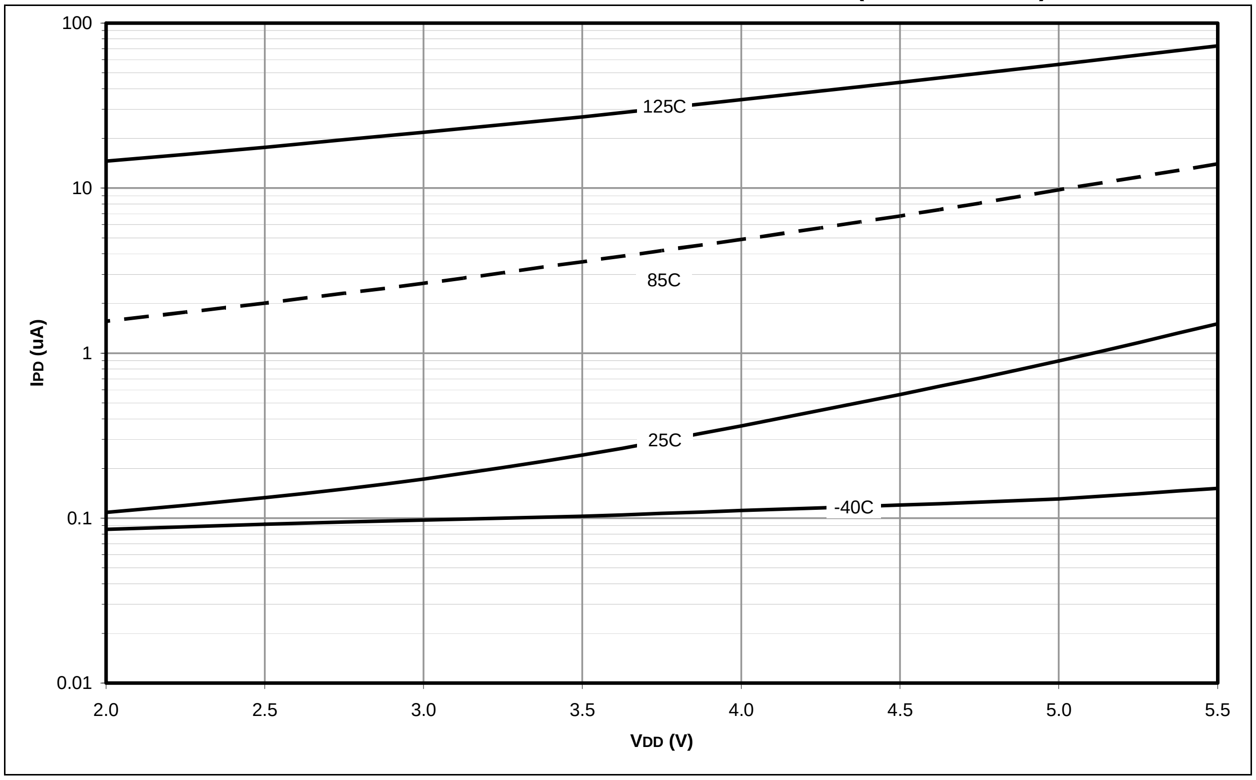


FIGURE 27-17: TYPICAL AND MAXIMUM SEC_RUN CURRENT vs. VDD ACROSS TEMPERATURE (T1OSC IN LOW-POWER MODE)

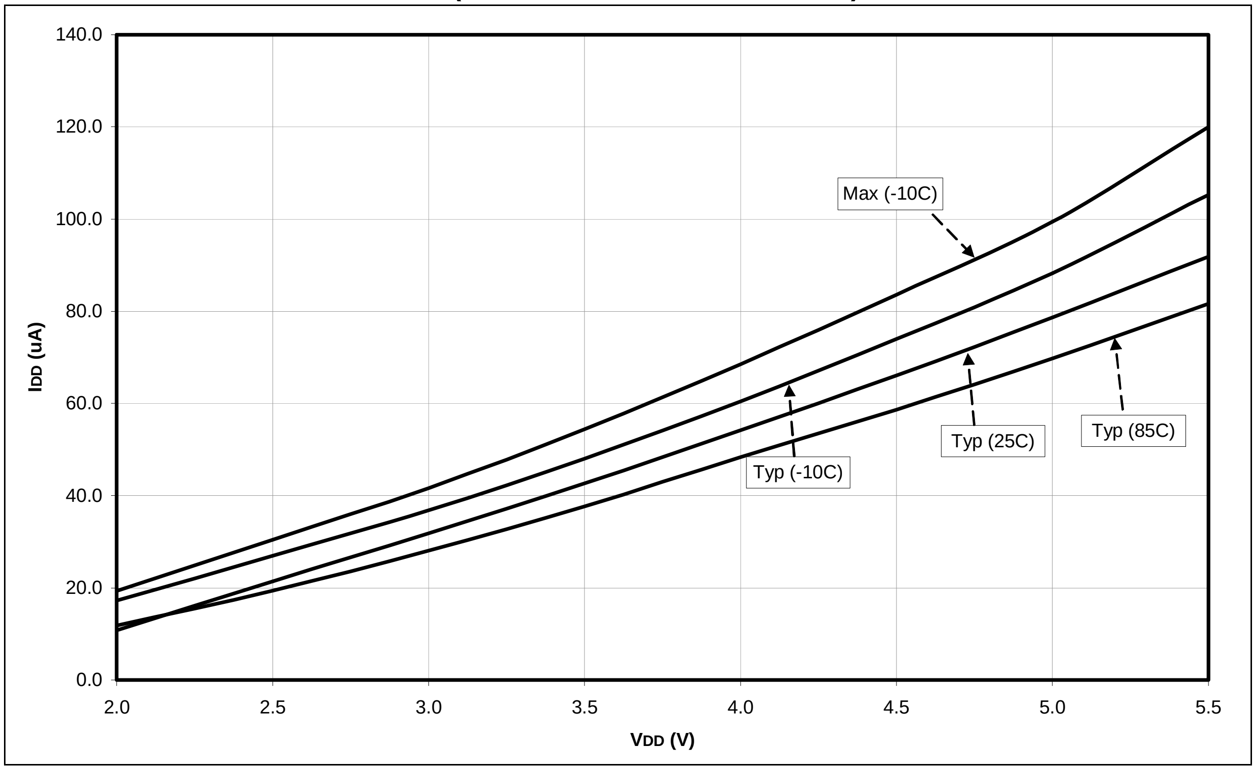
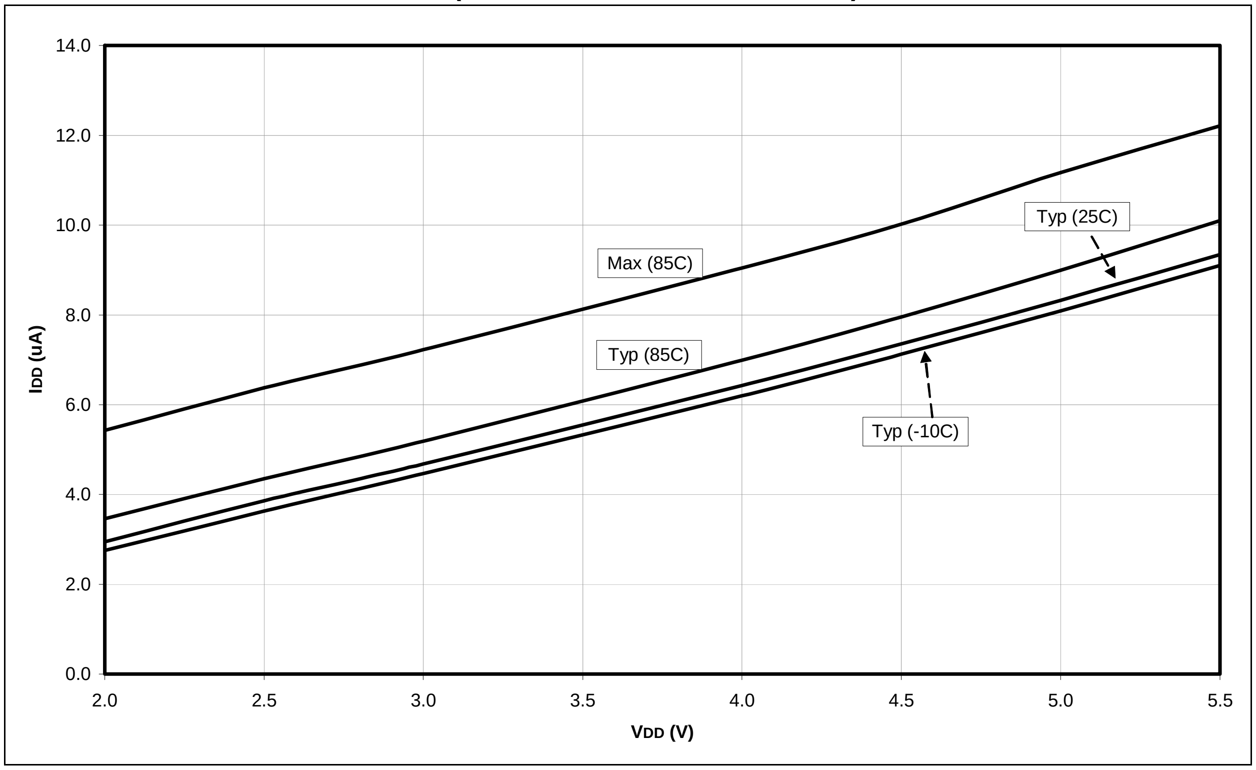


FIGURE 27-18: TYPICAL AND MAXIMUM SEC_IDLE CURRENT vs. VDD ACROSS TEMPERATURE (T1OSC IN LOW-POWER MODE)


IDD (uA)
VDD (V)
Max (85C)
Typ (25C)
Typ (85C)
Typ (-10C)