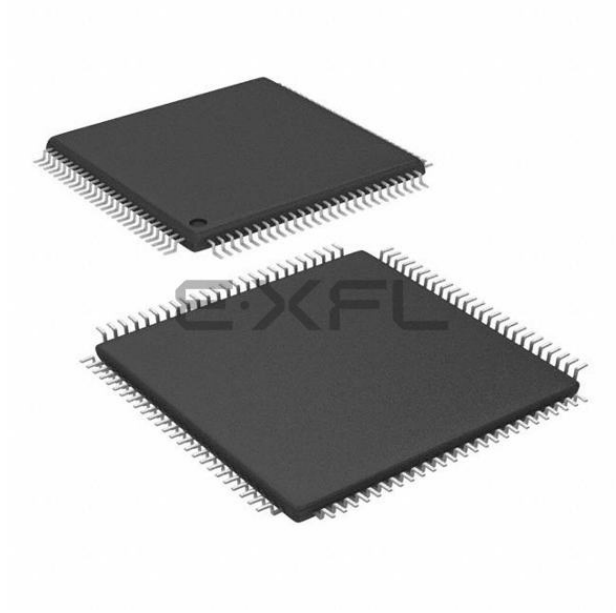
Microchip Technology - PIC18F96J94T-I/PF Datasheet



Welcome to E-XFL.COM

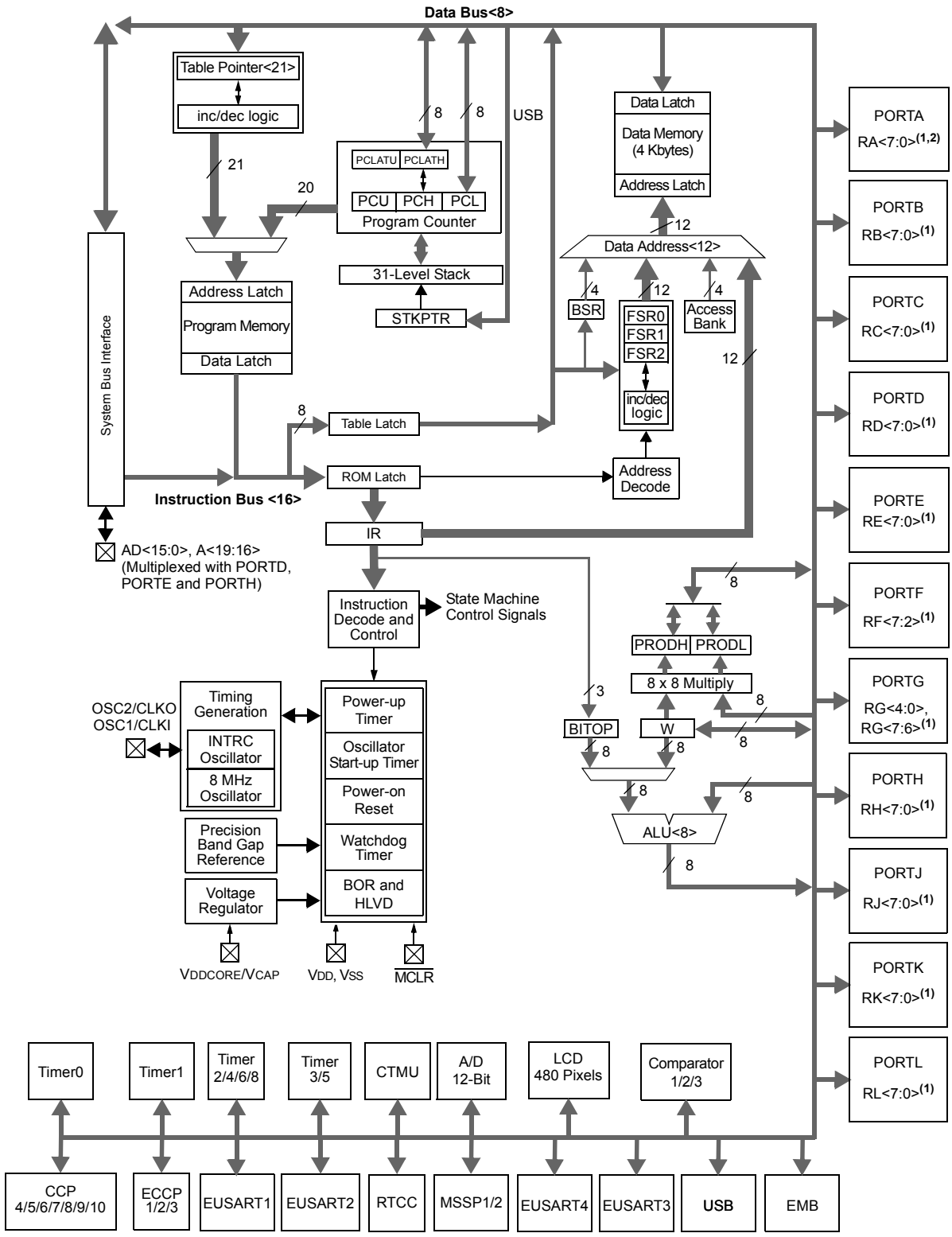
What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	64MHz
Connectivity	I²C, IrDA, LINbus, SPI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, HLVD, LCD, POR, PWM, WDT
Number of I/O	86
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 24x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-TQFP
Supplier Device Package	100-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f96j94t-i-pf

FIGURE 1-3: 100-PIN DEVICE BLOCK DIAGRAM



Note 1: See Table 1-4 for I/O port pin descriptions.

2: RA6 and RA7 are only available as digital I/O in select oscillator modes. See **Section 3.0 "Oscillator Configurations"** for more information.

4.2.4.1 Retention Sleep Mode

Retention Sleep mode allows for additional power savings over Sleep mode by maintaining key systems from the lower power retention regulator. When the retention regulator is used, the normal on-chip voltage regulator (operating at 1.8V nominal) is turned off and will enable a low-power (1.2V typical) regulator. By using a lower voltage, a lower total power consumption is achieved.

Retention Sleep also offers the advantage of maintaining the contents of the data RAM. As a trade-off, the wake-up time is longer than that for Sleep mode.

Retention Sleep mode is controlled by the SRETEN bit (RCON4<4>) and the $\overline{\text{RETEN}}$ Configuration bit, as described in Section 4.2.2, Retention Regulator.

4.3 Clock Source Considerations

When the device wakes up from either of the Sleep modes, it will restart the same clock source that was active when Sleep mode was entered. Wake-up delays for the different oscillator modes are shown in Table 4-3 and Table 4-4, respectively.

If the system clock source is derived from a crystal oscillator and/or the PLL, the Oscillator Start-up Timer (OST) and/or PLL lock times must be applied before the system clock source is made available to the device. As an exception to this rule, no oscillator delays are necessary if the system clock source is the Secondary Oscillator and it was running while in Sleep mode.

4.3.1 SLOW OSCILLATOR START-UP

The OST and PLL lock times may not have expired when the power-up delays have expired.

To avoid this condition, one can enable Two-Speed Start-up by the device that will run on FRC until the clock source is stable. Once the clock source is stable, the device will switch to the selected clock source.

4.3.2 WAKE-UP FROM SLEEP ON INTERRUPT

Any source of interrupt that is individually enabled, using its corresponding control bit in the PIEx registers, can wake-up the processor from Sleep mode. When the device wakes from Sleep mode, one of two following actions may occur:

- If the GIE bit is set, the processor will wake and the Program Counter will begin execution at the interrupt vector.
- If the GIE bit is not set, the processor will wake and the Program Counter will continue execution following the SLEEP instruction that initiated Sleep mode.

4.3.3 WAKE-UP FROM SLEEP ON RESET

All sources of device Reset will wake-up the processor from Sleep mode.

4.3.4 WAKE-UP FROM SLEEP ON WATCHDOG TIME-OUT

If the Watchdog Timer (WDT) is enabled and expires while the device is in Sleep mode, the processor will wake-up. The SWDTEN Status bit (RCON2<5>) is set to indicate that the device resumed operation due to the WDT expiration. Note that this event does not reset the device. Operation continues from the instruction following the SLEEP instruction that initiated Sleep mode.

4.3.5 CONTROL BIT SUMMARY FOR SLEEP MODES

Table 4-2 shows the settings for the bits relevant to Sleep modes.

TABLE 4-2: BIT SETTINGS FOR ALL SLEEP MODES

Mode	DSEN DSCONH<7>	Retention Regulator		
		$\overline{\text{RETEN}}$ CONFIG7L<0>	SRETEN RCON4<4>	State
Sleep	x	1	x	Disabled
	x	0	0	Disabled
Retention Sleep	x	0	1	Enabled

4.3.6 WAKE-UP DELAYS

The restart delay, associated with waking up from Sleep and Retention Sleep modes, parallel each other in terms of clock start-up times. They differ in the time it takes to switch over from their respective regulators. The delays for the different oscillator modes are shown in Table 4-3 and Table 4-4, respectively.

REGISTER 5-3: RCON3: RESET CONTROL REGISTER 3

U-0	U-0	U-0	U-0	R/C-0	R/C-0	R/C-0	R/W-0
—	—	—	—	VDDBOR(1)	VDDPOR(1,2)	VBPOR(1,3)	VBAT
bit 7							bit 0

Legend:	C = Clearable bit		
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 7-4 **Unimplemented**: Read as '0'

bit 3 **VDDBOR:** VDD Brown-out Reset Flag bit(1)
- 1 = A VDD Brown-out Reset has occurred
- 0 = A VDD Brown-out Reset has not occurred

bit 2 **VDDPOR:** VDD Power-On Reset Flag bit(1,2)
- 1 = A VDD Power-up Reset has occurred
- 0 = A VDD Power-up Reset has not occurred

bit 1 **VBPOR:** VBPOR Flag bit(1,3)
- 1 = A VBAT POR has occurred
- 0 = A VBAT POR has not occurred

bit 0 **VBAT**: VBAT Flag bit(1)
- 1 = A POR exit has occurred while power was applied to VBAT pin
- 0 = A POR exit from VBAT has not occurred

Note 1: This bit is set in hardware only; it can only be cleared in software.

2: Indicates a VDD POR. Setting the $\overline{\text{POR}}$ bit (RCON<0>) indicates a VCORE POR.

3: This bit is set when the device is originally powered up, even if power is present on VBAT.

TABLE 5-3: 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	64-pin	80-pin	100-pin	---0 0000	---0 0000	---0 uuuu(1)
TOSH	64-pin	80-pin	100-pin	0000 0000	0000 0000	uuuu uuuu(1)
TOSL	64-pin	80-pin	100-pin	0000 0000	0000 0000	uuuu uuuu(1)
STKPTR	64-pin	80-pin	100-pin	00-0 0000	uu-0 0000	uu-u uuuu(1)
PCLATU	64-pin	80-pin	100-pin	---0 0000	---0 0000	---u uuuu
PCLATH	64-pin	80-pin	100-pin	0000 0000	0000 0000	uuuu uuuu
PCL	64-pin	80-pin	100-pin	0000 0000	0000 0000	PC + 2(2)
TBLPTRU	64-pin	80-pin	100-pin	--00 0000	--00 0000	--uu uuuu
TBLPTRH	64-pin	80-pin	100-pin	0000 0000	0000 0000	uuuu uuuu
TBLPTRL	64-pin	80-pin	100-pin	0000 0000	0000 0000	uuuu uuuu
TABLAT	64-pin	80-pin	100-pin	0000 0000	0000 0000	uuuu uuuu
PRODH	64-pin	80-pin	100-pin	xxxx xxxx	uuuu uuuu	uuuu uuuu
PRODL	64-pin	80-pin	100-pin	xxxx xxxx	uuuu uuuu	uuuu uuuu
INTCON	64-pin	80-pin	100-pin	0000 000x	0000 000x	uuuu uuuu(3)
INTCON2	64-pin	80-pin	100-pin	1111 1111	1111 1111	uuuu uuuu(3)
INTCON3	64-pin	80-pin	100-pin	1100 0000	1100 0000	uuuu uuuu(3)
INDF0	64-pin	80-pin	100-pin	N/A	N/A	N/A
POSTINC0	64-pin	80-pin	100-pin	N/A	N/A	N/A
POSTDEC0	64-pin	80-pin	100-pin	N/A	N/A	N/A
PREINC0	64-pin	80-pin	100-pin	N/A	N/A	N/A
PLUSW0	64-pin	80-pin	100-pin	N/A	N/A	N/A
FSR0H	64-pin	80-pin	100-pin	---- xxxx	---- uuuu	---- uuuu
FSR0L	64-pin	80-pin	100-pin	xxxx xxxx	uuuu uuuu	uuuu uuuu
WREG	64-pin	80-pin	100-pin	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF1	64-pin	80-pin	100-pin	N/A	N/A	N/A
POSTINC1	64-pin	80-pin	100-pin	N/A	N/A	N/A
POSTDEC1	64-pin	80-pin	100-pin	N/A	N/A	N/A
PREINC1	64-pin	80-pin	100-pin	N/A	N/A	N/A
PLUSW1	64-pin	80-pin	100-pin	N/A	N/A	N/A
FSR1H	64-pin	80-pin	100-pin	---- xxxx	---- uuuu	---- uuuu
FSR1L	64-pin	80-pin	100-pin	xxxx xxxx	uuuu uuuu	uuuu uuuu
BSR	64-pin	80-pin	100-pin	---- 0000	---- 0000	---- uuuu
INDF2	64-pin	80-pin	100-pin	N/A	N/A	N/A

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

Note 1: 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.

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

4: See Table 5-2 for Reset value for specific condition.

5: Bits 7,6 are unimplemented on 64 and 80-pin devices.

6: If the V_{BAT} is always powered, the DSGPx register values will remain unchanged after the first POR.

6.1.3 PROGRAM COUNTER

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21 bits wide and contained in three separate 8-bit registers.

The low byte, known as the PCL register, is both readable and writable. The high byte, or PCH register, contains the PC<15:8> bits and is not directly readable or writable. Updates to the PCH register are performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits; it is also not directly readable or writable. Updates to the PCU register are performed through the PCLATU register.

The contents of PCLATH and PCLATU are transferred to the Program Counter by any operation that writes PCL. Similarly, the upper two bytes of the Program Counter are transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (see **Section 6.1.6.1 "Computed GOTO"**).

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the Least Significant bit of PCL is fixed to a value of '0'. The PC increments by two to address sequential instructions in the program memory.

The CALL, RCALL, GOTO and program branch instructions write to the Program Counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the Program Counter.

6.1.4 RETURN ADDRESS STACK

The return address stack enables execution of any combination of up to 31 program calls and interrupts. The PC is pushed onto the stack when a CALL or RCALL instruction is executed or an interrupt is Acknowledged. The PC value is pulled off the stack on a RETURN, RETLW or a RETFIE instruction. The value also is pulled off the stack on ADDULNK and SUBULNK instructions, if the extended instruction set is enabled. PCLATU and PCLATH are not affected by any of the RETURN or CALL instructions.

The stack operates as a 31-word by 21-bit RAM and a 5-bit Stack Pointer, STKPTR. The stack space is not part of either program or data space. The Stack Pointer is readable and writable and the address on the top of the stack is readable and writable through the Top-of-Stack Special Function Registers. Data can also be pushed to, or popped from, the stack using these registers.

A CALL type instruction causes a push onto the stack. The Stack Pointer is first incremented and the location pointed to by the Stack Pointer is written with the contents of the PC (already pointing to the instruction following the CALL). A RETURN type instruction causes a pop from the stack. The contents of the location pointed to by the STKPTR are transferred to the PC and then the Stack Pointer is decremented.

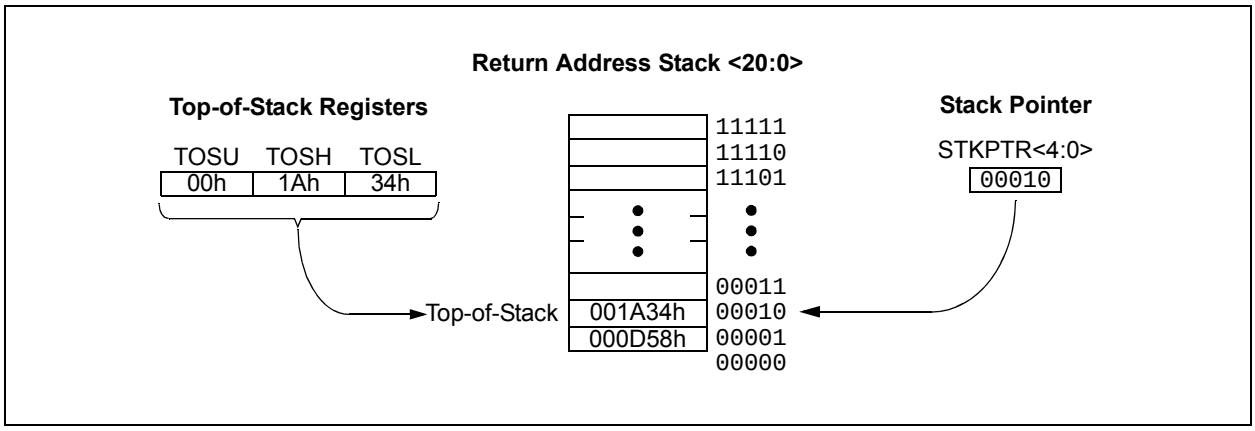
The Stack Pointer is initialized to '00000' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '00000'; this is only a Reset value. Status bits indicate if the stack is full, has overflowed or has underflowed.

6.1.4.1 Top-of-Stack Access

Only the top of the return address stack (TOS) is readable and writable. A set of three registers, TOSU:TOSH:TOSL, holds the contents of the stack location pointed to by the STKPTR register (Figure 6-3). This allows users to implement a software stack, if necessary. After a CALL, RCALL or interrupt (or ADDULNK and SUBULNK instructions, if the extended instruction set is enabled), the software can read the pushed value by reading the TOSU:TOSH:TOSL registers. These values can be placed on a user-defined software stack. At return time, the software can return these values to TOSU:TOSH:TOSL and do a return.

While accessing the stack, users must disable the Global Interrupt Enable bits to prevent inadvertent stack corruption.

FIGURE 6-3: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS



6.4 Data Addressing Modes

Note: The execution of some instructions in the core PIC18 instruction set are changed when the PIC18 extended instruction set is enabled. For more information, see **Section 6.6 "Data Memory and the Extended Instruction Set"**.

While the program memory can be addressed in only one way, through the Program Counter, information in the data memory space can be addressed in several ways. For most instructions, the addressing mode is fixed. Other instructions may use up to three modes, depending on which operands are used and whether or not the extended instruction set is enabled.

The addressing modes are:

- Inherent
- Literal
- Direct
- Indirect

An additional addressing mode, Indexed Literal Offset, is available when the extended instruction set is enabled (XINST Configuration bit = 1). For details on this mode's operation, see **Section 6.6.1 "Indexed Addressing with Literal Offset"**.

6.4.1 INHERENT AND LITERAL ADDRESSING

Many PIC18 control instructions do not need any argument at all. They either perform an operation that globally affects the device or they operate implicitly on one register. This addressing mode is known as Inherent Addressing. Examples of this mode include SLEEP, RESET and DAW.

Other instructions work in a similar way, but require an additional explicit argument in the opcode. This method is known as the Literal Addressing mode because the instructions require some literal value as an argument. Examples of this include ADDLW and MOVLW which, respectively, add or move a literal value to the W register. Other examples include CALL and GOTO, which include a 20-bit program memory address.

6.4.2 DIRECT ADDRESSING

Direct Addressing specifies all or part of the source and/or destination address of the operation within the opcode itself. The options are specified by the arguments accompanying the instruction.

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of Direct Addressing by default. All of these instructions include some 8-bit literal address as their Least Significant Byte. This address specifies the instruction's data source as either a register address in one of the banks of data RAM (see **Section 6.3.3 "General Purpose Register File"**) or a location in the Access Bank (see **Section 6.3.2 "Access Bank"**).

The Access RAM bit, 'a', determines how the address is interpreted. When 'a' is '1', the contents of the BSR (**Section 6.3.1 "Bank Select Register"**) are used with the address to determine the complete 12-bit address of the register. When 'a' is '0', the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

A few instructions, such as MOVFF, include the entire 12-bit address (either source or destination) in their opcodes. In these cases, the BSR is ignored entirely.

The destination of the operation's results is determined by the destination bit, 'd'. When 'd' is '1', the results are stored back in the source register, overwriting its original contents. When 'd' is '0', the results are stored in the W register. Instructions without the 'd' argument have a destination that is implicit in the instruction, either the target register being operated on or the W register.

6.4.3 INDIRECT ADDRESSING

Indirect Addressing allows the user to access a location in data memory without giving a fixed address in the instruction. This is done by using File Select Registers (FSRs) as pointers to the locations to be read or written to. Since the FSRs are themselves located in RAM as Special Function Registers, they can also be directly manipulated under program control. This makes FSRs very useful in implementing data structures such as tables and arrays in data memory.

The registers for Indirect Addressing are also implemented with Indirect File Operands (INDFs) that permit automatic manipulation of the pointer value with auto-incrementing, auto-decrementing or offsetting with another value. This allows for efficient code using loops, such as the example of clearing an entire RAM bank in Example 6-5. It also enables users to perform Indexed Addressing and other Stack Pointer operations for program memory in data memory.

EXAMPLE 6-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

```
        LFSR    FSR0, 100h ;
NEXT    CLRF    POSTINC0   ; Clear INDF
                           ; register then
                           ; inc pointer
        BTFSS   FSR0H, 1   ; All done with
                           ; Bank1?
        BRA     NEXT       ; NO, clear next
CONTINUE                   ; YES, continue
```

6.4.3.1 FSR Registers and the INDF Operand

At the core of Indirect Addressing are three sets of registers: FSR0, FSR1 and FSR2. Each represents a pair of 8-bit registers: FSRnH and FSRnL. The four upper bits of the FSRnH register are not used, so each FSR pair holds a 12-bit value. This represents a value that can address the entire range of the data memory in a linear fashion. The FSR register pairs, then, serve as pointers to data memory locations.

Indirect Addressing is accomplished with a set of Indirect File Operands, INDF0 through INDF2. These can be thought of as "virtual" registers. The operands are mapped in the SFR space, but are not physically implemented. Reading or writing to a particular INDF register actually accesses its corresponding FSR register pair. A read from INDF1, for example, reads the data at the address indicated by FSR1H:FSR1L.

Instructions that use the INDF registers as operands actually use the contents of their corresponding FSR as a pointer to the instruction's target. The INDF operand is just a convenient way of using the pointer.

Because Indirect Addressing uses a full 12-bit address, data RAM banking is not necessary. Thus, the current contents of the BSR and the Access RAM bit have no effect on determining the target address.

FIGURE 6-8: INDIRECT ADDRESSING

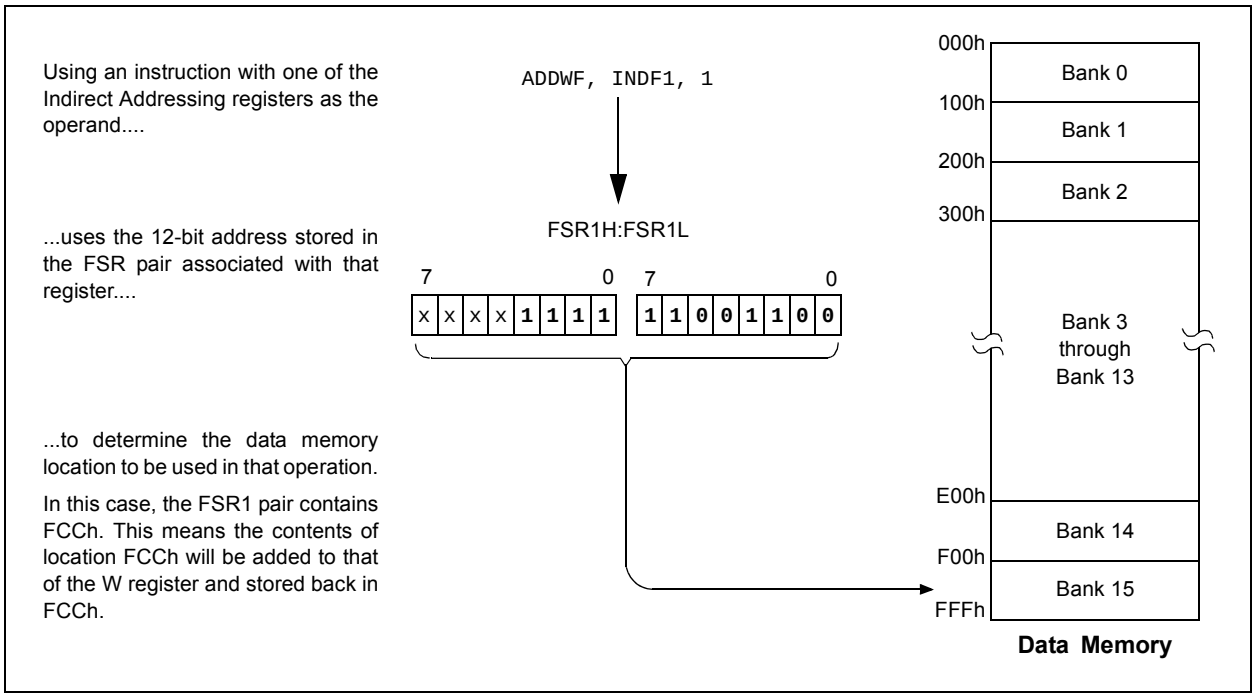


TABLE 11-13: RPIN REGISTERS AND AVAILABLE FUNCTIONS

PPS-Lite Input Peripheral Group 4n	
(1) To Map this signal	(4) to the Associated RPIN Register
SDI1	RPINR8_9<7:4>
FLT0	RPINR14_15<3:0>
IOC0	RPINR18_19<3:0>
IOC4	RPINR22_23<3:0>
MDCIN1	RPINR30_31<3:0>
T0CKI	RPINR38_39<7:4>
T5G	RPINR44_45<3:0>
U3RX	RPINR4_5<3:0>
U4RX	RPINR6_7<3:0>
CCP5	RPINR32_33<7:4>
CCP8	RPINR36_37<3:0>
RVP0	RPINR46_47<3:0>
RVP4	RPINR50_51<3:0>
(2) with this RPn Pin	(3) Write this Corresponding Value
RP0	h'0
RP4	h'1
RP8	h'2
RP12	h'3
RP16	h'4
RP20	h'5
RP24	h'6
RP28	h'7
RP32	h'8
RP36	h'9
RP40	h'A
RP44	h'B
—	h'C
—	h'D
—	h'E
Vss	h'F

PPS-Lite Input Peripheral Group 4n + 1	
(1) To Map this Signal	(4) to the Associated RPIN Register
SDI2	RPINR12_13<3:0>
INT1	RPINR26_27<3:0>
IOC1	RPINR18_19<7:4>
IOC5	RPINR22_23<7:4>
MDCIN2	RPINR30_31<7:4>
T1CKI	RPINR40_41<7:4>
T1G	RPINR40_41<3:0>
T3CKI	RPINR42_43<7:4>
T3G	RPINR42_43<3:0>
T5CKI	RPINR44_45<7:4>
U3TX	RPINR4_5<7:4>
U4TX	RPINR6_7<7:4>
CCP7	RPINR34_35<7:4>
CCP9	RPINR36_37<7:4>
RVP1	RPINR46_47<7:4>
RVP5	RPINR50_51<7:4>
(2) with this RPn Pin	(3) Write this Corresponding Value
RP1	h'0
RP5	h'1
RP9	h'2
RP13	h'3
RP17	h'4
RP21	h'5
RP25	h'6
RP29	h'7
RP33	h'8
RP37	h'9
RP41	h'A
RP45	h'B
—	h'C
—	h'D
—	h'E
Vss	h'F

FIGURE 12-2: ON-OFF KEYING (OOK) SYNCHRONIZATION

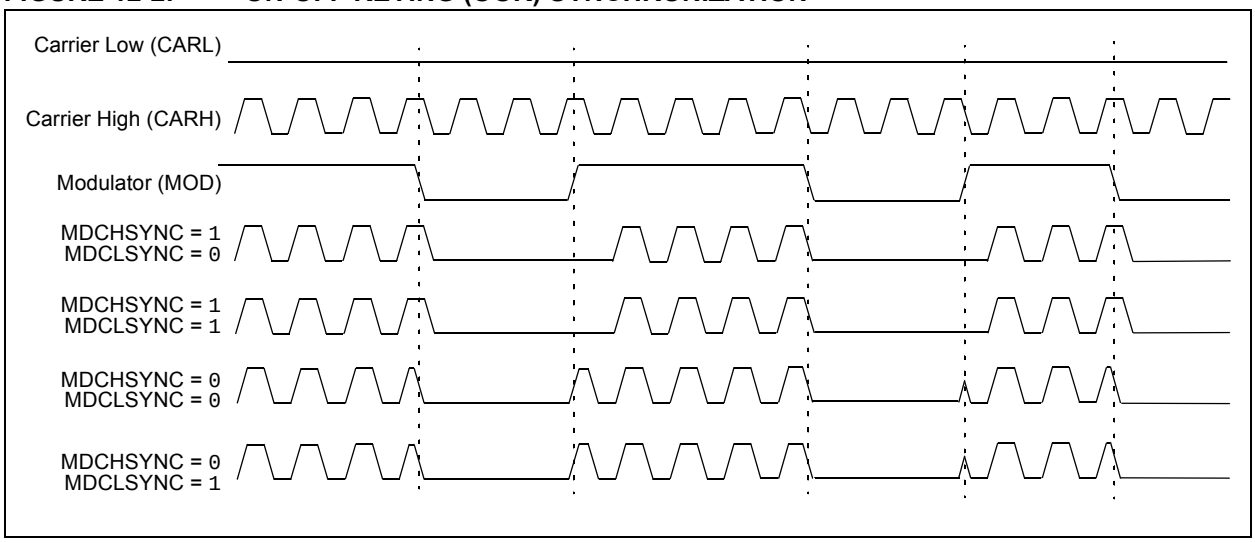


FIGURE 12-3: NO SYNCHRONIZATION (MDCHSYNC = 0, MDCLSYNC = 0)

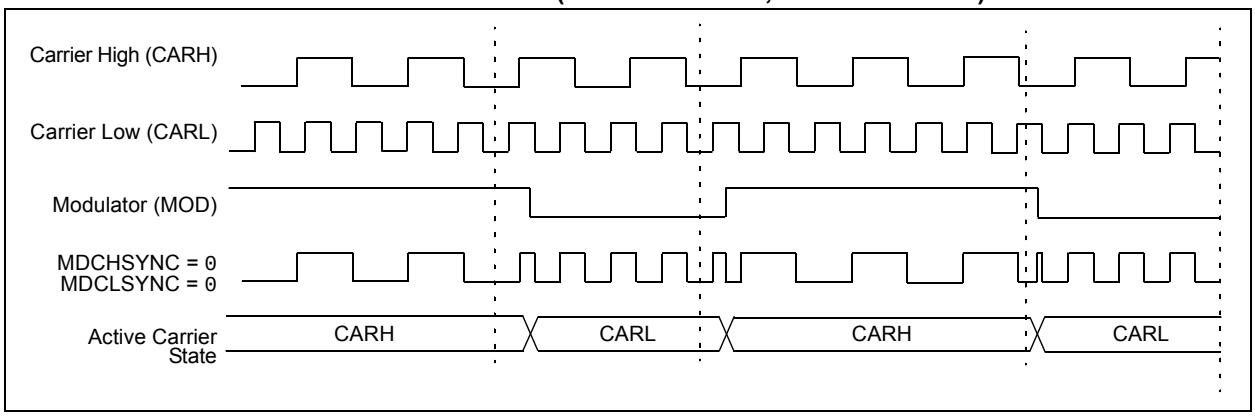
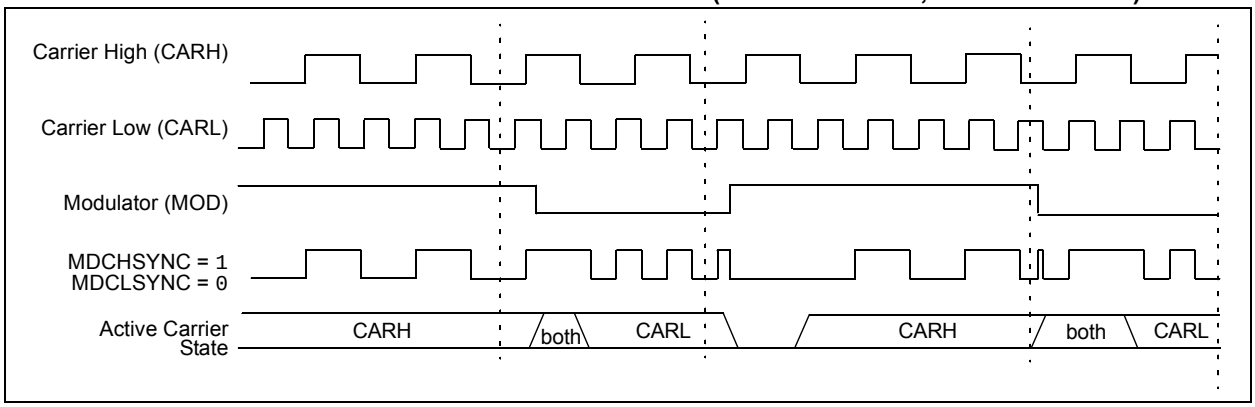


FIGURE 12-4: CARRIER HIGH SYNCHRONIZATION (MDCHSYNC = 1, MDCLSYNC = 0)



REGISTER 20-8: SSPxCON2: MSSPx CONTROL REGISTER 2 (I^2C MASTER 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
GCEN	ACKSTAT	ACKDT(1)	ACKEN(2)	RCEN(2)	PEN(2)	RSEN(2)	SEN(2)
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 **GCEN:** General Call Enable bit
Unused in Master mode.

bit 6 **ACKSTAT:** Acknowledge Status bit (Master Transmit mode only)
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave

bit 5 **ACKDT:** Acknowledge Data bit (Master Receive mode only)(1)
1 = Not Acknowledge
0 = Acknowledge

bit 4 **ACKEN:** Acknowledge Sequence Enable bit(2)
1 = Initiates Acknowledge sequence on SDAx and SCLx pins and transmits ACKDT data bit; automatically cleared by hardware
0 = Acknowledge sequence is Idle

bit 3 **RCEN:** Receive Enable bit (Master Receive mode only)(2)
1 = Enables Receive mode for I^2C
0 = Receive is Idle

bit 2 **PEN:** Stop Condition Enable bit(2)
1 = Initiates Stop condition on SDAx and SCLx pins; automatically cleared by hardware
0 = Stop condition is Idle

bit 1 **RSEN:** Repeated Start Condition Enable bit(2)
1 = Initiates Repeated Start condition on SDAx and SCLx pins; automatically cleared by hardware
0 = Repeated Start condition is Idle

bit 0 **SEN:** Start Condition Enable bit(2)
1 = Initiates Start condition on SDAx and SCLx pins; automatically cleared by hardware
0 = Start condition is Idle

Note 1: The value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.

2: If the I^2C module is active, these bits may not be set (no spooling) and the SSPxBUF may not be written (or writes to the SSPxBUF are disabled).

20.5.10 BAUD RATE

In I^2C Master mode, the Baud Rate Generator (BRG) reload value is placed in the lower 7 bits of the SSPxADD register (Figure 20-19). When a write occurs to SSPxBUF, the Baud Rate Generator will automatically begin counting. The BRG counts down to 0 and stops until another reload has taken place. The BRG count is decremented, twice per instruction cycle (TCY), on the Q2 and Q4 clocks. In I^2C Master mode, the BRG is reloaded automatically.

Once the given operation is complete (i.e., transmission of the last data bit is followed by $\overline{ACK}$), the internal clock will automatically stop counting and the SCLx pin will remain in its last state.

Table 20-2 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPxADD. The SSPxADD BRG value of '0x00' is not supported.

20.5.10.1 Baud Rate and Module Interdependence

Because MSSP1 and MSSP2 are independent, they can operate simultaneously in I^2C Master mode at different baud rates. This is done by using different BRG reload values for each module.

Because this mode derives its basic clock source from the system clock, any changes to the clock will affect both modules in the same proportion. It may be possible to change one or both baud rates back to a previous value by changing the BRG reload value.

FIGURE 20-19: BAUD RATE GENERATOR BLOCK DIAGRAM

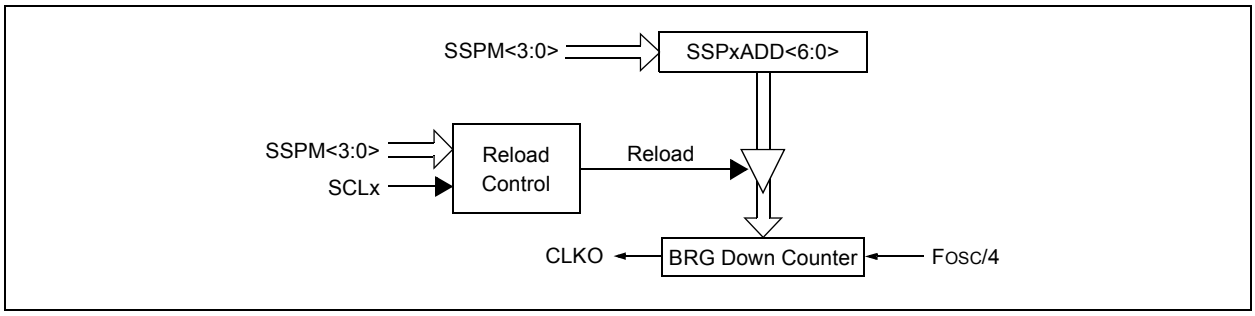


TABLE 20-2: I^2C CLOCK RATE w/BRG

FOSC	FCY	FCY * 2	BRG Value	FSCL (2 Rollovers of BRG)
64 MHz	16 MHz	32 MHz	27h	400 kHz(1)
64 MHz	16 MHz	32 MHz	32h	313.72 kHz
64 MHz	16 MHz	32 MHz	9Fh	100 kHz
16 MHz	4 MHz	8 MHz	09h	400 kHz(1)
16 MHz	4 MHz	8 MHz	0Ch	308 kHz
16 MHz	4 MHz	8 MHz	27h	100 kHz
4 MHz	1 MHz	2 MHz	02h	333 kHz(1)
4 MHz	1 MHz	2 MHz	09h	100 kHz
16 MHz	4 MHz	8 MHz	03h	1 MHz(1,1)

Note 1: A minimum of 16 MHz FOSC is required to get 1 MHz I^2C.

21.2.2 EUSARTx ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 21-6. The data is received on the RXx pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at x16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at FOSC. This mode would typically be used in RS-232 systems.

To set up an Asynchronous Reception:

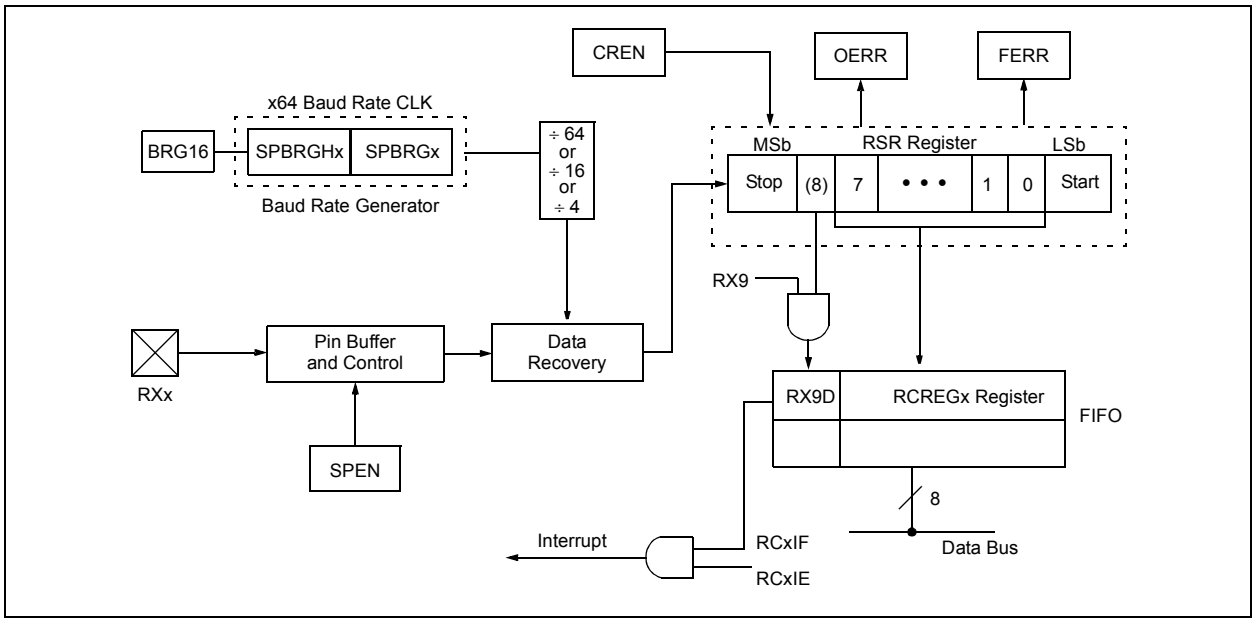
1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit, SYNC, and setting bit, SPEN.
3. If interrupts are desired, set enable bit, RCxIE.
4. If 9-bit reception is desired, set bit, RX9.
5. Enable the reception by setting bit, CREN.
6. Flag bit, RCxIF, will be set when reception is complete and an interrupt will be generated if enable bit, RCxIE, was set.
7. Read the RCSTAx register to get the 9th bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREGx register.
9. If any error occurred, clear the error by clearing enable bit, CREN.
10. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

21.2.3 SETTING UP 9-BIT MODE WITH ADDRESS DETECT

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If interrupts are required, set the RCEN bit and select the desired priority level with the RCxIP bit.
4. Set the RX9 bit to enable 9-bit reception.
5. Set the ADDEN bit to enable address detect.
6. Enable reception by setting the CREN bit.
7. The RCxIF bit will be set when reception is complete. The interrupt will be Acknowledged if the RCxIE and GIE bits are set.
8. Read the RCSTAx register to determine if any error occurred during reception, as well as read bit 9 of data (if applicable).
9. Read RCREGx to determine if the device is being addressed.
10. If any error occurred, clear the CREN bit.
11. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and interrupt the CPU.

FIGURE 21-6: EUSARTx RECEIVE BLOCK DIAGRAM



REGISTER 22-6: ADCON2H: A/D CONTROL REGISTER 2 HIGH

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0
PVCFG1	PVCFG0	NVCFG0	OFFCAL	BUFREGEN	CSCNA	—	—
bit 7							bit 0

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

bit 7-6 **PVCFG<1:0>:** Converter Positive Voltage Reference Configuration bits
1x =Unimplemented, do not use
01 =External V_{REF+}
00 =AV_{DD}

bit 5 **NVCFG0:** Converter Negative Voltage Reference Configuration bit
1 = External V_{REF-}
0 = AV_{SS}

bit 4 **OFFCAL:** Offset Calibration Mode Select bit
1 = Inverting and non-inverting inputs of channel Sample-and-Hold are connected to AV_{SS}
0 = Inverting and non-inverting inputs of channel Sample-and-Hold are connected to normal inputs

bit 3 **BUFREGEN:** A/D Buffer Register Enable bit
1 = Conversion result is loaded into the buffer location determined by the converted channel
0 = A/D result buffer is treated as a FIFO

bit 2 **CSCNA:** Scan Input Selections for CH0+ During Sample A bit
1 = Scans inputs
0 = Does not scan inputs

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

The module is enabled by setting the HLVDEN bit (HLVDCON<4>). Each time the HLVD module is enabled, the circuitry requires some time to stabilize. The IRVST bit (HLVDCON<5>) is a read-only bit used to indicate when the circuit is stable. The module can only generate an interrupt after the circuit is stable and IRVST is set.

The VDIRMAG bit (HLVDCON<7>) determines the overall operation of the module. When VDIRMAG is cleared, the module monitors for drops in VDD below a predetermined set point. When the bit is set, the module monitors for rises in VDD above the set point.

25.1 Operation

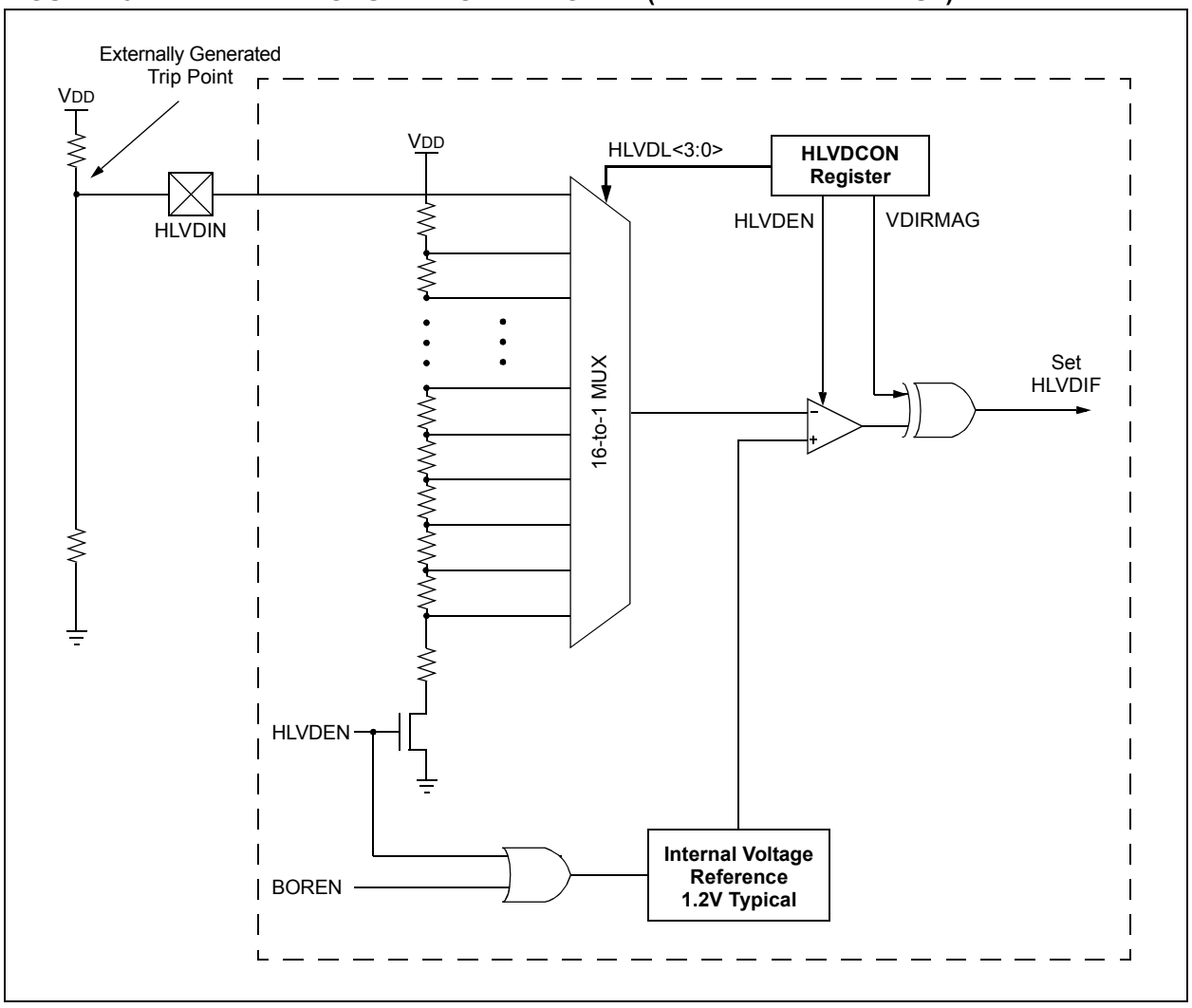
When the HLVD module is enabled, a comparator uses an internally generated reference voltage as the set point. The set point is compared with the trip point, where each node in the resistor divider represents a trip point voltage. The "trip point" voltage is the voltage level at which the device detects a high or low-voltage event, depending on the configuration of the module.

When the supply voltage is equal to the trip point, the voltage tapped off of the resistor array is equal to the internal reference voltage generated by the voltage reference module. The comparator then generates an interrupt signal by setting the HLVDIF bit.

The trip point voltage is software programmable to any of 16 values. The trip point is selected by programming the HLVDL<3:0> bits (HLVDCON<3:0>).

The HLVD module has an additional feature that allows the user to supply the trip voltage to the module from an external source. This mode is enabled when bits, HLVDL<3:0>, are set to '1111'. In this state, the comparator input is multiplexed from the external input pin, HLVDIN. This gives users the flexibility of configuring the High/Low-Voltage Detect interrupt to occur at any voltage in the valid operating range.

FIGURE 25-1: HLVD MODULE BLOCK DIAGRAM (WITH EXTERNAL INPUT)



REGISTER 27-2: UCFG: USB CONFIGURATION REGISTER

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
UTEYE	UOEMON	—	UPUEN(1,2)	UTRDIS(1,3)	FSEN(1)	PPB1	PPB0
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 **UTEYE:** USB Eye Pattern Test Enable bit
1 = Eye pattern test is enabled
0 = Eye pattern test is disabled

bit 6 **UOEMON:** USB OE Monitor Enable bit
1 = $\overline{\text{UOE}}$ signal is active, indicating intervals during which the D+/D- lines are driving
0 = $\overline{\text{UOE}}$ signal is inactive

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

bit 4 **UPUEN:** USB On-Chip Pull-up Enable bit(1,2)
1 = On-chip pull-up is enabled (pull-up on D+ with FSEN = 1 or D- with FSEN = 0)
0 = On-chip pull-up is disabled

bit 3 **UTRDIS:** On-Chip Transceiver Disable bit(1,3)
1 = On-chip transceiver is disabled
0 = On-chip transceiver is active

bit 2 **FSEN:** Full-Speed Enable bit(1)
1 = Full-speed device: Controls transceiver edge rates; requires input clock at 48 MHz
0 = Low-speed device: Controls transceiver edge rates; requires input clock at 6 MHz

bit 1-0 **PPB<1:0>:** Ping-Pong Buffers Configuration bits
11 = Even/Odd ping-pong buffers are enabled for Endpoints 1 to 15
10 = Even/Odd ping-pong buffers are enabled for all endpoints
01 = Even/Odd ping-pong buffer are enabled for OUT Endpoint 0
00 = Even/Odd ping-pong buffers are disabled

Note 1: The UPUEN, UTRDIS and FSEN bits should never be changed while the USB module is enabled. These values must be preconfigured prior to enabling the module.
2: This bit is only valid when the on-chip transceiver is active (UTRDIS = 0); otherwise, it is ignored.
3: If UTRDIS is set, the $\overline{\text{UOE}}$ signal will be active, independent of the UOEMON bit setting.

27.2.4 USB ENDPOINT CONTROL

Each of the 16 possible bidirectional endpoints has its own independent control register, UEPn (where 'n' represents the endpoint number). Each register has an identical complement of control bits.

Register 27-4 provides the prototype.

The EPHSHK bit (UEPn<4>) controls handshaking for the endpoint; setting this bit enables USB handshaking. Typically, this bit is always set except when using isochronous endpoints.

The EPCONDIS bit (UEPn<3>) is used to enable or disable USB control operations (SETUP) through the endpoint. Clearing this bit enables SETUP transactions. Note that the corresponding EPINEN and EPOUTEN bits must be set to enable IN and OUT transactions. For Endpoint 0, this bit should always be cleared since the USB specifications identify Endpoint 0 as the default control endpoint.

The EPOUTEN bit (UEPn<2>) is used to enable or disable USB OUT transactions from the host. Setting this bit enables OUT transactions. Similarly, the EPINEN bit (UEPn<1>) enables or disables USB IN transactions from the host.

The EPSTALL bit (UEPn<0>) is used to indicate a STALL condition for the endpoint. If a STALL is issued on a particular endpoint, the EPSTALL bit for that endpoint pair will be set by the SIE. This bit remains set until it is cleared through firmware or until the SIE is reset.

REGISTER 27-4: UEPn: USB ENDPOINT n CONTROL REGISTER (UEP0 THROUGH UEP15)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL
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 7-5 **Unimplemented:** Read as '0'

bit 4 **EPHSHK:** Endpoint Handshake Enable bit
1 = Endpoint handshake is enabled
0 = Endpoint handshake is disabled (typically used for isochronous endpoints)

bit 3 **EPCONDIS:** Bidirectional Endpoint Control bit
If EPOUTEN = 1 and EPINEN = 1:
1 = Disables Endpoint n from control transfers; only IN and OUT transfers are allowed
0 = Enables Endpoint n for control (SETUP) transfers; IN and OUT transfers are also allowed

bit 2 **EPOUTEN:** Endpoint Output Enable bit
1 = Endpoint n output is enabled
0 = Endpoint n output is disabled

bit 1 **EPINEN:** Endpoint Input Enable bit
1 = Endpoint n input is enabled
0 = Endpoint n input is disabled

bit 0 **EPSTALL:** Endpoint Stall Indicator bit
1 = Endpoint n has issued one or more STALL packets
0 = Endpoint n has not issued any STALL packets

27.4.1.3 BDnSTAT Register (SIE Mode)

When the BD and its buffer are owned by the SIE, most of the bits in BDnSTAT take on a different meaning. The configuration is shown in Register 27-6. Once UOWN is set, any data or control settings previously written there by the user will be overwritten with data from the SIE.

The BDnSTAT register is updated by the SIE with the token Packet Identifier (PID) which is stored in BDnSTAT<5:2>. The transfer count in the corresponding BDnCNT register is updated. Values that overflow the 8-bit register carry over to the two most significant digits of the count, stored in BDnSTAT<1:0>.

27.4.2 BD BYTE COUNT

The byte count represents the total number of bytes that will be transmitted during an IN transfer. After an IN transfer, the SIE will return the number of bytes sent to the host.

For an OUT transfer, the byte count represents the maximum number of bytes that can be received and stored in USB RAM. After an OUT transfer, the SIE will return the actual number of bytes received. If the number of bytes received exceeds the corresponding byte count, the data packet will be rejected and a NAK handshake will be generated. When this happens, the byte count will not be updated.

The 10-bit byte count is distributed over two registers. The lower 8 bits of the count reside in the BDnCNT register; the upper two bits reside in BDnSTAT<1:0>. This represents a valid byte range of 0 to 1023.

27.4.3 BD ADDRESS VALIDATION

The BD Address register pair contains the starting RAM address location for the corresponding endpoint buffer. No mechanism is available in hardware to validate the BD address.

If the value of the BD address does not point to an address in the USB RAM, or if it points to an address within another endpoint's buffer, data is likely to be lost or overwritten. Similarly, overlapping a receive buffer (OUT endpoint) with a BD location in use can yield unexpected results. When developing USB applications, the user may want to consider the inclusion of software-based address validation in their code.

REGISTER 27-6: BDnSTAT: BUFFER DESCRIPTOR n STATUS REGISTER (BD0STAT THROUGH BD63STAT), SIE MODE (DATA RETURNED BY THE SIE TO THE MCU)

R/W-x	r-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
UOWN	r	PID3	PID2	PID1	PID0	BC9	BC8
bit 7							bit 0

Legend:	r = Reserved bit		
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 **UOWN:** USB Own bit
1 = The SIE owns the BD and its corresponding buffer

bit 6 **Reserved:** Not written by the SIE

bit 5-2 **PID<3:0>:** Packet Identifier bits
The received token PID value of the last transfer (IN, OUT or SETUP transactions only).

bit 1-0 **BC<9:8>:** Byte Count 9 and 8 bits
These bits are updated by the SIE to reflect the actual number of bytes received on an OUT transfer and the actual number of bytes transmitted on an IN transfer.

REGISTER 28-15: DEVID2: DEVICE ID REGISTER 2 FOR THE PIC18FXXJ94

R	R	R	R	R	R	R	R
DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3
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 7-0 **DEV<10:3>:** Device ID bits
These bits are used with the DEV<2:0> bits in the Device ID Register 1 to identify the part number.

REGISTER 28-16: DEVID1: DEVICE ID REGISTER 1 FOR THE PIC18FXXJ94

R	R	R	R	R	R	R	R
DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0
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 7-5 **DEV<2:0>:** Device ID bits
These bits are used with the DEV<10:3> bits in the Device ID Register 2 to identify the part number:

```
0110 0010 101 = PIC18F97J94
0110 0010 111 = PIC18F96J94
0110 0011 000 = PIC18F95J94
0110 0011 001 = PIC18F87J94
0110 0011 011 = PIC18F86J94
0110 0011 100 = PIC18F85J94
0110 0011 101 = PIC18F67J94
0110 0011 111 = PIC18F66J94
0110 0100 000 = PIC18F65J94
```

bit 4-0 **REV<4:0>:** Revision ID bits
These bits are used to indicate the device revision.

RETFIE **Return from Interrupt**

Syntax: RETFIE {s}

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
1 → GIE/GIEH or PEIE/GIEL;
if s = 1,
(WS) → W,
(STATUSS) → STATUS,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged

Status Affected: GIE/GIEH, PEIE/GIEL.

Encoding:

0000	0000	0001	000s

Description: Return from interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into the PC. Interrupts are enabled by setting either the high or low-priority Global Interrupt Enable bit. If 's' = 1, the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers W, STATUS and BSR. If 's' = 0, no update of these registers occurs.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	POP PC from stack Set GIEH or GIEL
No operation	No operation	No operation	No operation

Example: RETFIE 1

After Interrupt
PC = TOS
W = WS
BSR = BSRS
STATUS = STATUSS
GIE/GIEH, PEIE/GIEL = 1

RETLW **Return Literal to W**

Syntax: RETLW k

Operands: 0 ≤ k ≤ 255

Operation: k → W,
(TOS) → PC,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

0000	1100	kkkk	kkkk

Description: W is loaded with the 8-bit literal 'k'. The Program Counter is loaded from the top of the stack (the return address). The high address latch (PCLATH) remains unchanged.

Words: 1

Cycles: 2

Q Cycle Activity:

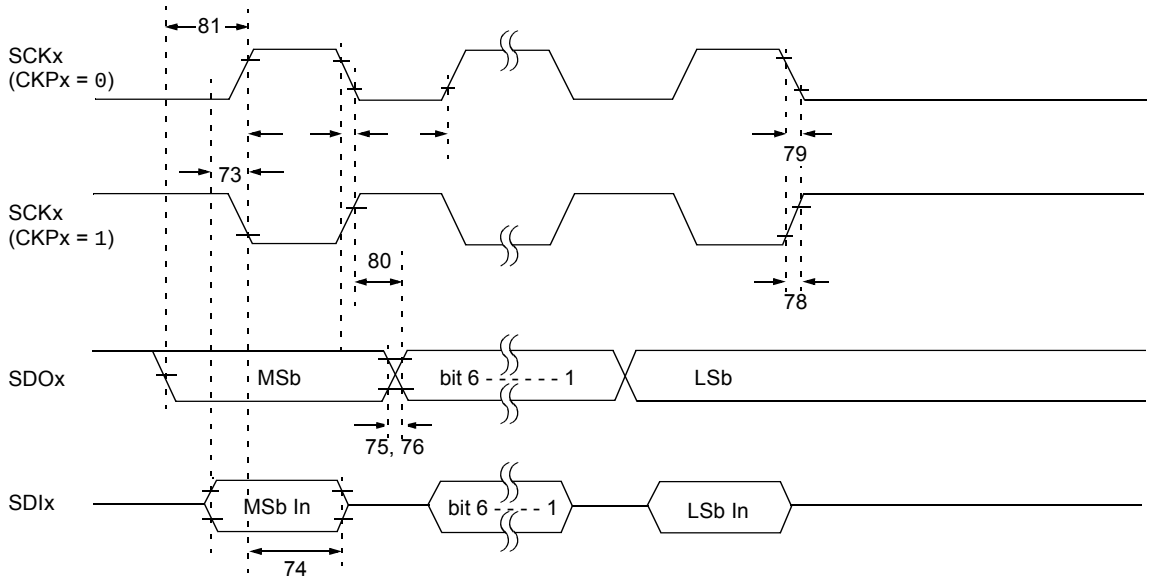
Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	POP PC from stack, write to W
No operation	No operation	No operation	No operation

Example:

```
    CALL TABLE ; W contains table
               ; offset value
               ; W now has
               ; table value
    :
TABLE
    ADDWF PCL  ; W = offset
    RETLW k0   ; Begin table
    RETLW k1   ;
    :
    :
    RETLW kn   ; End of table
```

Before Instruction
W = 07h
After Instruction
W = value of kn

FIGURE 30-13: EXAMPLE SPI MASTER MODE TIMING (CKE = 1)



Note: Refer to Figure 30-2 for load conditions.

TABLE 30-31: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 1)

Param. No.	Symbol	Characteristic	Min.	Max.	Units	Conditions
73	TDIV2SCH, TDIV2SCL	Setup Time of SDIx Data Input to SCKx Edge	20	—	ns	
73A	TB2B	Last Clock Edge of Byte 1 to the 1st Clock Edge of Byte 2	1.5 TCY + 40	—	ns	
74	TSCH2DIL, TSCL2DIL	Hold Time of SDIx Data Input to SCKx Edge	40	—	ns	
75	TDOR	SDOx Data Output Rise Time	—	25	ns	
76	TDOF	SDOx Data Output Fall Time	—	25	ns	
78	TSCR	SCKx Output Rise Time (Master mode)	—	25	ns	
79	TSCF	SCKx Output Fall Time (Master mode)	—	25	ns	
80	TSCH2DOV, TSCL2DOV	SDOx Data Output Valid after SCKx Edge	—	50	ns	
81	TDOV2SCH, TDOV2SCL	SDOx Data Output Setup to SCKx Edge	TCY	—	ns	