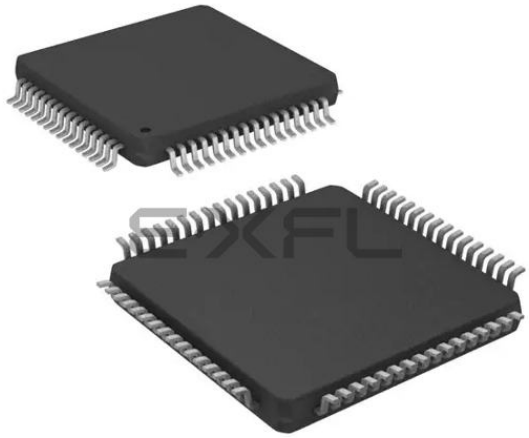
Microchip Technology - PIC18F6525T-I/PT Datasheet



Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

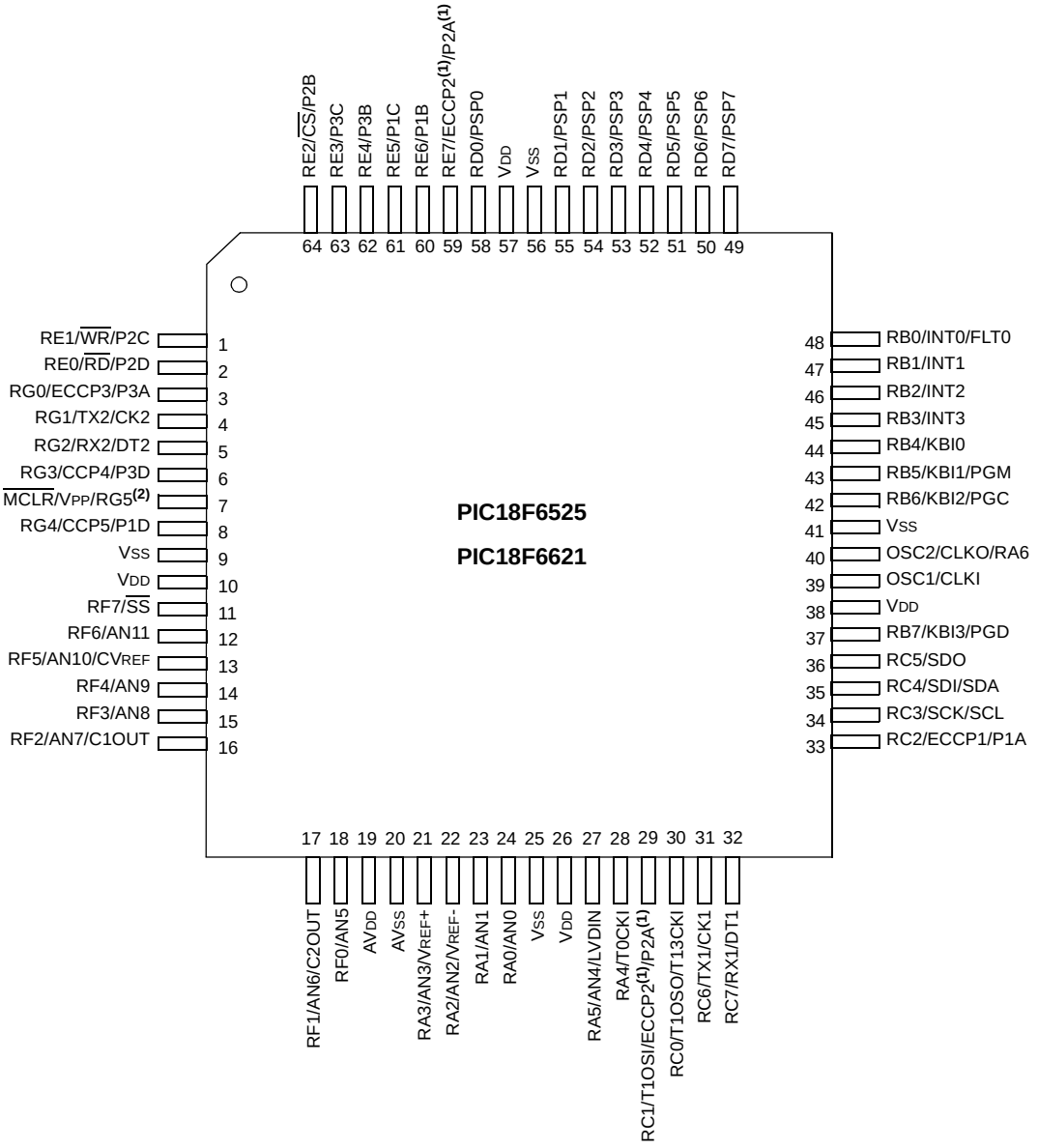
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	53
Program Memory Size	48KB (24K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.75K x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f6525t-i-pt

Pin Diagrams

64-Pin TQFP



Note 1: ECCP2/P2A are multiplexed with RC1 when CCP2MX is set, or RE7 when CCP2MX is not set.

2: RG5 is multiplexed with $\overline{\text{MCLR}}$ and is only available when the $\overline{\text{MCLR}}$ Resets are disabled.

4.7.1 TWO-WORD INSTRUCTIONS

The PIC18F6525/6621/8525/8621 devices have four two-word instructions: MOVFF, CALL, GOTO and LFSR. The second word of these instructions has the 4 MSBs set to '1's and is a special kind of NOP instruction. The lower 12 bits of the second word contain data to be used by the instruction. If the first word of the instruction is executed, the data in the second word is accessed. If the second word of the instruction is executed by itself (first word was skipped), it will execute as a NOP. This action is necessary when the two-word instruction is preceded by a conditional instruction that changes the PC. A program example that demonstrates this concept is shown in Example 4-3. Refer to **Section 25.0 "Instruction Set Summary"** for further details of the instruction set.

EXAMPLE 4-3: TWO-WORD INSTRUCTIONS

CASE 1:

Object Code	Source Code		
0110 0110 0000 0000	TSTFSZ	REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	; No, execute 2-word instruction
1111 0100 0101 0110			; 2nd operand holds address of REG2
0010 0100 0000 0000	ADDWF	REG3	; continue code

CASE 2:

Object Code	Source Code		
0110 0110 0000 0000	TSTFSZ	REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	; Yes
1111 0100 0101 0110			; 2nd operand becomes NOP
0010 0100 0000 0000	ADDWF	REG3	; continue code

4.8 Look-up Tables

Look-up tables are implemented two ways. These are:

- Computed GOTO
- Table Reads

4.8.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL).

A look-up table can be formed with an ADDWF PCL instruction and a group of RETLW 0xnn instructions. WREG 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 0xnn instructions that returns the value 0xnn to the calling function.

The offset value (value in WREG) specifies the number of bytes that the program counter should advance.

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

Note:	The ADDWF PCL instruction does not update PCLATH and PCLATU. A read operation on PCL must be performed to update PCLATH and PCLATU.

EXAMPLE 4-4: COMPUTED GOTO USING AN OFFSET VALUE

```
MAIN:  ORG    0x0000
       MOVLW  0x00
       CALL   TABLE
…
       ORG    0x8000
TABLE  MOVF   PCL, F    ; A simple read of PCL will update PCLATH, PCLATU
       RLNCF  W, W      ; Multiply by 2 to get correct offset in table
       ADDWF  PCL       ; Add the modified offset to force jump into table
       RETLW  'A'
       RETLW  'B'
       RETLW  'C'
       RETLW  'D'
       RETLW  'E'
       END
```

TABLE 4-3: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
PREINC2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register)								N/A	56
PLUSW2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – value of FSR2 offset by value in WREG								N/A	56
FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2 High Byte				---- 0000	33, 56
FSR2L	Indirect Data Memory Address Pointer 2 Low Byte								xxxx xxxx	33, 56
STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx	33, 58
TMR0H	Timer0 Register High Byte								0000 0000	33, 133
TMR0L	Timer0 Register Low Byte								xxxx xxxx	33, 133
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	33, 131
OSCCON	—	—	—	—	LOCK	PLLEN	SCS1	SCS0	---- 0000	25, 33
LVDCON	—	—	IRVST	LVDEN	LVDL3	LVDL2	LVDL1	LVDL0	--00 0101	33, 255
WDTCON	—	—	—	—	—	—	—	SWDTEN	---- ---0	33, 267
RCON	IPEN	—	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	0--1 11qq	33, 59, 101
TMR1H	Timer1 Register High Byte								xxxx xxxx	33, 139
TMR1L	Timer1 Register Low Byte								xxxx xxxx	33, 139
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	0-00 0000	33, 139
TMR2	Timer2 Register								0000 0000	33, 142
PR2	Timer2 Period Register								1111 1111	33, 142
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	33, 142
SSPBUF	MSSP Receive Buffer/Transmit Register								xxxx xxxx	33, 181
SSPADD	MSSP Address Register in I^2C Slave mode. MSSP Baud Rate Reload Register in I^2C Master mode.								0000 0000	33, 181
SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	33, 174
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	33, 175
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	33, 185
ADRESH	A/D Result Register High Byte								xxxx xxxx	33, 241
ADRESL	A/D Result Register Low Byte								xxxx xxxx	33, 241
ADCON0	—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	ADON	--00 0000	34, 233
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	--00 0000	34, 234
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	0-00 0000	34, 235
CCPR1H	Enhanced Capture/Compare/PWM Register 1 High Byte								xxxx xxxx	34, 172
CCPR1L	Enhanced Capture/Compare/PWM Register 1 Low Byte								xxxx xxxx	34, 172
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	0000 0000	34, 157
CCPR2H	Enhanced Capture/Compare/PWM Register 2 High Byte								xxxx xxxx	34, 172
CCPR2L	Enhanced Capture/Compare/PWM Register 2 Low Byte								xxxx xxxx	34, 172
CCP2CON	P2M1	P2M0	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	0000 0000	34, 157
CCPR3H	Enhanced Capture/Compare/PWM Register 3 High Byte								xxxx xxxx	34, 172
CCPR3L	Enhanced Capture/Compare/PWM Register 3 Low Byte								xxxx xxxx	34, 172
CCP3CON	P3M1	P3M0	DC3B1	DC2B0	CCP3M3	CCP3M2	CCP3M1	CCP3M0	0000 0000	34, 157
ECCP1AS	ECCP1ASE	ECCP1AS2	ECCP1AS1	ECCP1AS0	PSS1AC1	PSS1AC0	PSS1BD1	PSS1BD0	0000 0000	34, 169
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	0000 0000	34, 249

Legend: x = unknown, u = unchanged, – = unimplemented, q = value depends on condition

Note 1: RA6 and associated bits are configured as a port pin in RCIO and ECIO Oscillator modes only and read '0' in all other oscillator modes.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: These registers are unused on PIC18F6525/6621 devices and read as '0'.

4: RG5 is available only if $\overline{MCLR}$ function is disabled in configuration.

5: Enabled only in Microcontroller mode for PIC18F8525/8621 devices.

7.0 DATA EEPROM MEMORY

The data EEPROM is readable and writable during normal operation over the entire VDD range. The data memory is not directly mapped in the register file space. Instead, it is indirectly addressed through the Special Function Registers (SFR).

There are five SFRs used to read and write the program and data EEPROM memory. These registers are:

- EECON1
- EECON2
- EEDATA
- EEADRH
- EEADR

The EEPROM data memory allows byte read and write. When interfacing to the data memory block, EEDATA holds the 8-bit data for read/write. EEADR and EEADRH hold the address of the EEPROM location being accessed. These devices have 1024 bytes of data EEPROM with an address range from 00h to 3FFh.

The EEPROM data memory is rated for high erase/write cycles. A byte write automatically erases the location and writes the new data (erase-before-write). The write time is controlled by an on-chip timer. The write time will vary with voltage and temperature, as well as from chip-to-chip. Please refer to parameter D122 (**Section 27.0 "Electrical Characteristics"**) for exact limits.

7.1 EEADR and EEADRH

The address register pair can address up to a maximum of 1024 bytes of data EEPROM. The two Most Significant bits of the address are stored in EEADRH, while the remaining eight Least Significant bits are stored in EEADR. The six Most Significant bits of EEADRH are unused and are read as '0'.

7.2 EECON1 and EECON2 Registers

EECON1 is the control register for EEPROM memory accesses.

EECON2 is not a physical register. Reading EECON2 will read all '0's. The EECON2 register is used exclusively in the EEPROM write sequence.

Control bits RD and WR initiate read and write operations, respectively. These bits cannot be cleared, only set in software. They are cleared in hardware at the completion of the read or write operation. The inability to clear the WR bit in software prevents the accidental or premature termination of a write operation.

Note:	During normal operation, the WRERR bit is read as '1'. This can indicate that a write operation was prematurely terminated by a Reset, or a write operation was attempted improperly.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a $\overline{\text{MCLR}}$ Reset or a WDT Time-out Reset during normal operation. In these situations, the user can check the WRERR bit and rewrite the location. It is necessary to reload the data and address registers (EEDATA and EEADR) due to the Reset condition forcing the contents of the registers to zero.

Note:	Interrupt flag bit, EEIF in the PIR2 register, is set when write is complete. It must be cleared in software.

9.3 PIE Registers

The PIE registers contain the individual enable bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Enable registers (PIE1, PIE2 and PIE3). When the IPEN bit (RCON<7>) is '0', the PEIE bit must be set to enable any of these peripheral interrupts.

REGISTER 9-7: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit(1)
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

Note: Enabled only in Microcontroller mode for PIC18F8525/8621 devices.

bit 6 **ADIE**: A/D Converter Interrupt Enable bit
1 = Enables the A/D interrupt
0 = Disables the A/D interrupt

bit 5 **RC1IE**: USART1 Receive Interrupt Enable bit
1 = Enables the USART1 receive interrupt
0 = Disables the USART1 receive interrupt

bit 4 **TX1IE**: USART1 Transmit Interrupt Enable bit
1 = Enables the USART1 transmit interrupt
0 = Disables the USART1 transmit interrupt

bit 3 **SSPIE**: Master Synchronous Serial Port Interrupt Enable bit
1 = Enables the MSSP interrupt
0 = Disables the MSSP interrupt

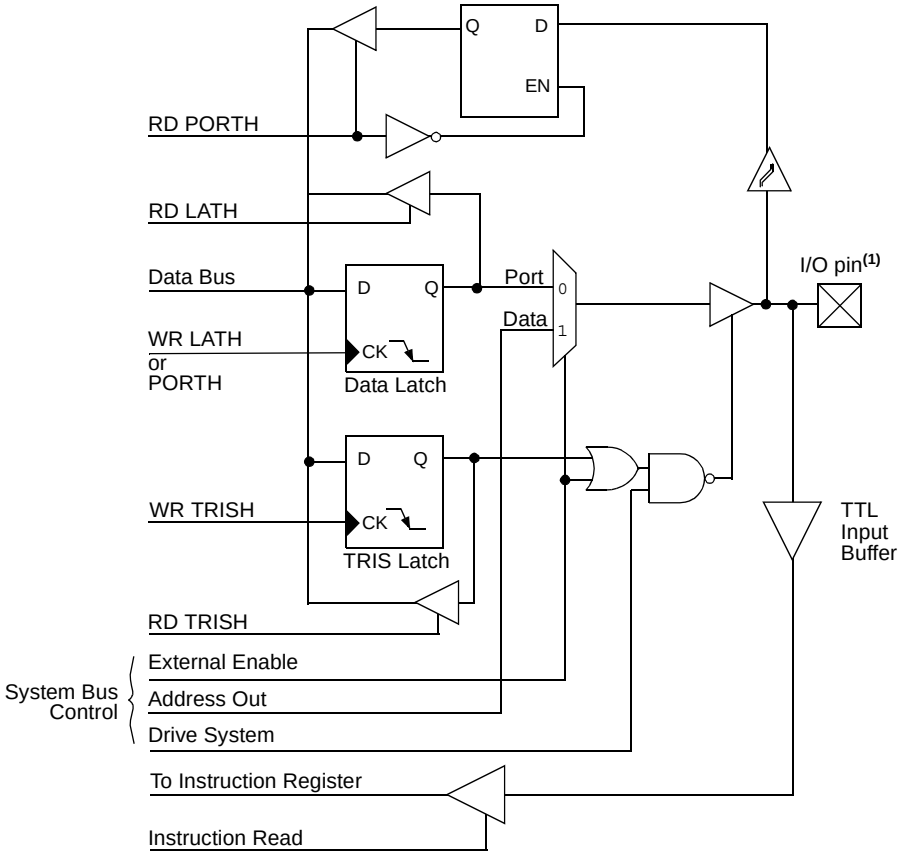
bit 2 **CCP1IE**: ECCP1 Interrupt Enable bit
1 = Enables the ECCP1 interrupt
0 = Disables the ECCP1 interrupt

bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

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

FIGURE 10-20: RH3:RH0 PINS BLOCK DIAGRAM IN SYSTEM BUS MODE



Note 1: I/O pins have diode protection to VDD and VSS.

15.0 TIMER4 MODULE

The Timer4 module timer has the following features:

- 8-bit timer (TMR4 register)
- 8-bit period register (PR4)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4, 1:16)
- Software programmable postscaler (1:1 to 1:16)
- Interrupt on TMR4 match of PR4

Timer4 has a control register shown in Register 15-1. Timer4 can be shut off by clearing control bit, TMR4ON (T4CON<2>), to minimize power consumption. The prescaler and postscaler selection of Timer4 are also controlled by this register. Figure 15-1 is a simplified block diagram of the Timer4 module.

15.1 Timer4 Operation

Timer4 can be used as the PWM time base for the PWM mode of the CCP module. The TMR4 register is readable and writable and is cleared on any device Reset. The input clock (FOSC/4) has a prescale option of 1:1, 1:4 or 1:16, selected by control bits T4CKPS1:T4CKPS0 (T4CON<1:0>). The match output of TMR4 goes through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling inclusive) to generate a TMR4 interrupt, latched in flag bit TMR4IF (PIR3<3>).

The prescaler and postscaler counters are cleared when any of the following occurs:

- a write to the TMR4 register
- a write to the T4CON register
- any device Reset (Power-on Reset, $\overline{\text{MCLR}}$ Reset, Watchdog Timer Reset, or Brown-out Reset)

TMR4 is not cleared when T4CON is written.

REGISTER 15-1: T4CON: TIMER4 CONTROL REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	T4OUTPS3	T4OUTPS2	T4OUTPS1	T4OUTPS0	TMR4ON	T4CKPS1	T4CKPS0
bit 7							bit 0

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

bit 6-3 **T4OUTPS3:T4OUTPS0**: Timer4 Output Postscale Select bits
0000 = 1:1 Postscale
0001 = 1:2 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR4ON**: Timer4 On bit
1 = Timer4 is on
0 = Timer4 is off

bit 1-0 **T4CKPS1:T4CKPS0**: Timer4 Clock Prescale Select bits
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

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

REGISTER 18-2: 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	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

bit 7 **WCOL:** Write Collision Detect bit (Transmit mode only)

1 = The SSPBUF 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

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

Note: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.

bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit

1 = Enables serial port and configures SCK, SDO, SDI and $\overline{SS}$ as serial port pins

0 = Disables serial port and configures these pins as I/O port pins

Note: When enabled, these pins must be properly configured as input or output.

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

0101 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control disabled, $\overline{SS}$ can be used as I/O pin

0100 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ 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: Bit combinations not specifically listed here are either reserved or implemented in I^2C mode only.

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

18.4.17.2 Bus Collision During a Repeated Start Condition

During a Repeated Start condition, a bus collision occurs if:

a) A low level is sampled on SDA when SCL goes from low level to high level.

b) SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1'.

When the user deasserts SDA and the pin is allowed to float high, the BRG is loaded with SSPADD<6:0> and counts down to '0'. The SCL pin is then deasserted and when sampled high, the SDA pin is sampled.

If SDA is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0', Figure 18-29). If SDA is sampled high, the BRG is reloaded and begins counting. If SDA goes from high-to-low before the BRG times out, no bus collision occurs because no two masters can assert SDA at exactly the same time.

If SCL goes from high-to-low before the BRG times out and SDA has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition, see Figure 18-30.

If, at the end of the BRG time-out, both SCL and SDA are still high, the SDA pin is driven low and the BRG is reloaded and begins counting. At the end of the count regardless of the status of the SCL pin, the SCL pin is driven low and the Repeated Start condition is complete.

FIGURE 18-29: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)

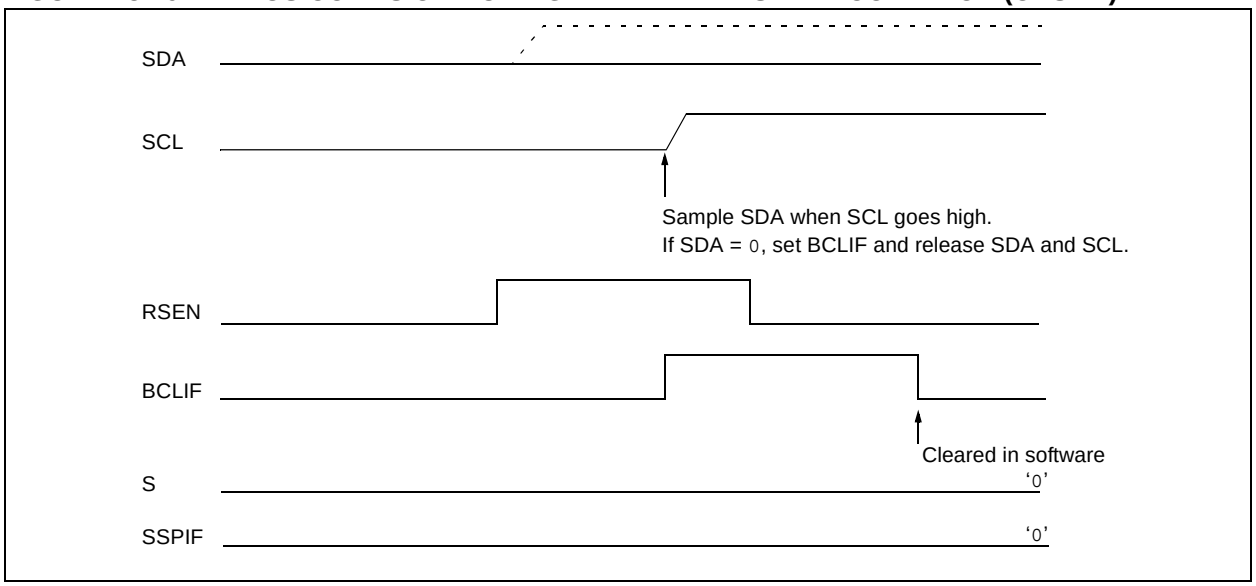


FIGURE 18-30: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)

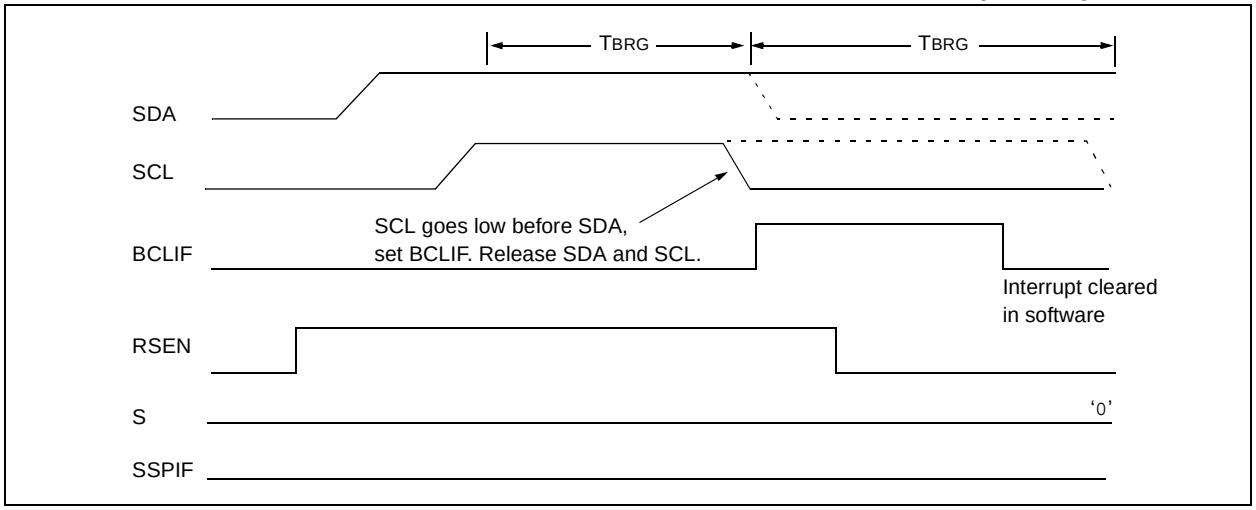


FIGURE 19-3: ASYNCHRONOUS TRANSMISSION

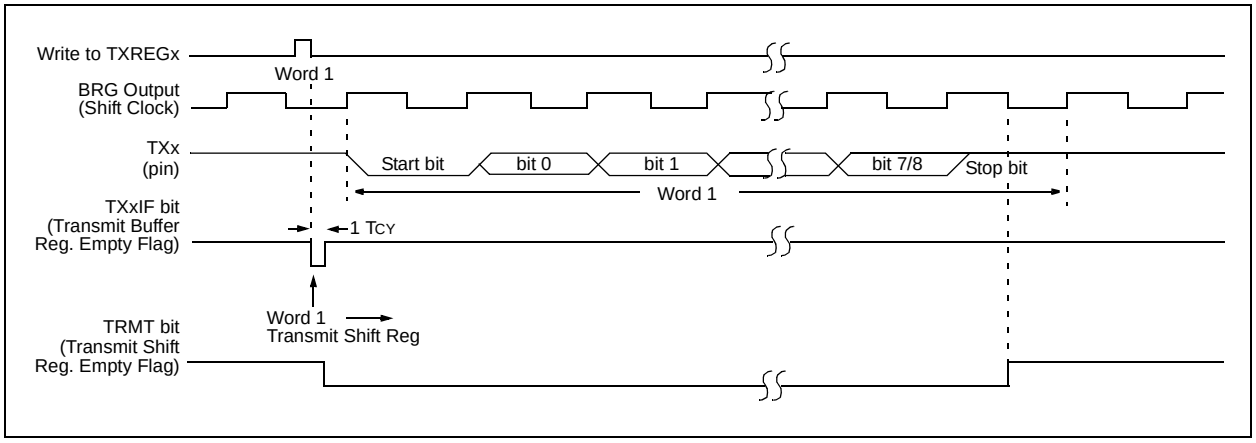
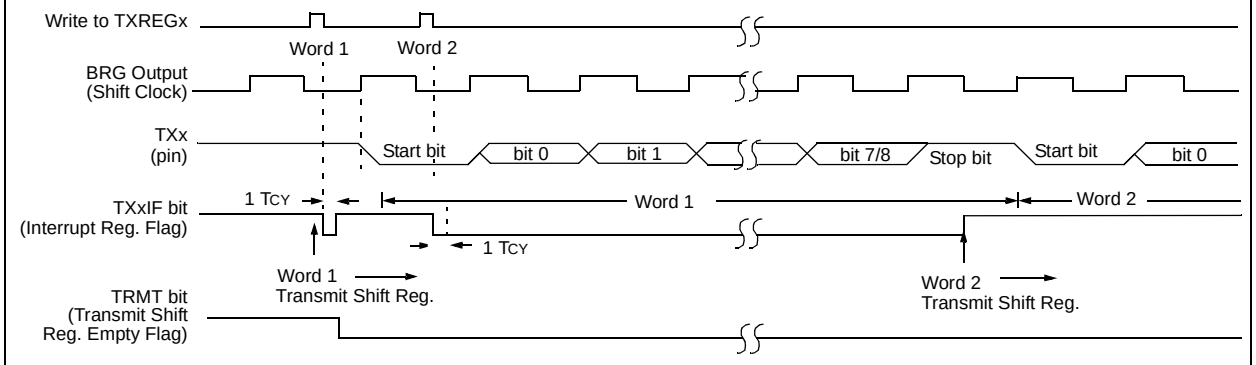


FIGURE 19-4: ASYNCHRONOUS TRANSMISSION (BACK TO BACK)



Note: This timing diagram shows two consecutive transmissions.

TABLE 19-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RC1IF	TX1IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RC1IP	TX1IP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
PIR3	—	—	RC2IF	TX2IF	TMR4IF	CCP5IF	CCP4IF	CCP3IF	--00 0000	--00 0000
PIE3	—	—	RC2IE	TX2IE	TMR4IE	CCP5IE	CCP4IE	CCP3IE	--00 0000	--00 0000
IPR3	—	—	RC2IP	TX2IP	TMR4IP	CCP5IP	CCP4IP	CCP3IP	--11 1111	--11 1111
RCSTAx	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
TXREGx	Enhanced USARTx Transmit Register								0000 0000	0000 0000
TXSTAx	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	0000 0010
BAUDCONx	—	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	-1-0 0-00	-1-0 0-00
SPBRGHx	Enhanced USARTx Baud Rate Generator Register High Byte								0000 0000	0000 0000
SPBRGx	Enhanced USARTx Baud Rate Generator Register Low Byte								0000 0000	0000 0000

Legend: x = unknown, – = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.

Note 1: Enabled only in Microcontroller mode for PIC18F8525/8621 devices.

20.4 Configuring Analog Port Pins

The ADCON1, TRISA, TRISF and TRISH registers control the operation of the A/D port pins. The port pins needed as analog inputs must have their corresponding TRIS bits set (input). If the TRIS bit is cleared (output), the digital output level (V_{OH} or V_{OL}) will be converted.

The A/D operation is independent of the state of the CHS3:CHS0 bits and the TRIS bits.

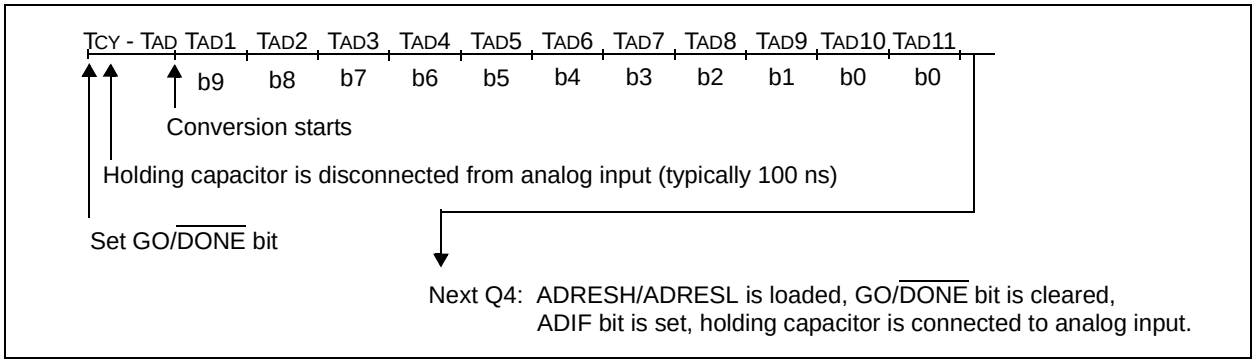
Note 1:	When reading the port register, all pins configured as analog input channels will read as cleared (a low level). Pins configured as a digital input will convert as an analog input. Analog levels on a digitally configured input will not affect the conversion accuracy.
2:	Analog levels on any pin defined as a digital input may cause the input buffer to consume current out of the device's specification limits.

20.5 A/D Conversions

Figure 20-3 shows the operation of the A/D converter after the GO/$\overline{\text{DONE}}$ bit has been set. Clearing the GO/$\overline{\text{DONE}}$ bit during a conversion will abort the current conversion. The A/D Result register pair will NOT be updated with the partially completed A/D conversion sample. That is, the ADRESH:ADRESL registers will continue to contain the value of the last completed conversion (or the last value written to the ADRESH:ADRESL registers). After the A/D conversion is aborted, a 2 T_{AD} wait is required before the next acquisition is started. After this 2 T_{AD} wait, acquisition on the selected channel is automatically started.

Note:	The GO/$\overline{\text{DONE}}$ bit should **NOT** be set in the same instruction that turns on the A/D.

FIGURE 20-3: A/D CONVERSION T_{AD} CYCLES



BNOV **Branch if Not Overflow**

Syntax: [*label*] BNOV n

Operands: -128 ≤ n ≤ 127

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

Status Affected: None

Encoding:

1110	0101	nnnn	nnnn

Description: If the Overflow 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 BNOV Jump`

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

BNZ **Branch if Not Zero**

Syntax: [*label*] BNZ n

Operands: -128 ≤ n ≤ 127

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

Status Affected: None

Encoding:

1110	0001	nnnn	nnnn

Description: If the Zero 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 BNZ Jump`

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

RCALL **Relative Call**

Syntax: [*label*] RCALL n

Operands: -1024 ≤ n ≤ 1023

Operation: (PC) + 2 → TOS;
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1101	1nnn	nnnn	nnnn

Description: Subroutine call with a jump up to 1K from the current location. First, return address (PC + 2) is pushed onto the stack. Then, add the 2's complement number '2n' 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 a two-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

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

Example: `HERE RCALL Jump`

Before Instruction
PC = Address `(HERE)`
After Instruction
PC = Address `(Jump)`
TOS = Address `(HERE + 2)`

RESET **Reset**

Syntax: [*label*] RESET

Operands: None

Operation: Reset all registers and flags that are affected by a $\overline{\text{MCLR}}$ Reset.

Status Affected: All

Encoding:

0000	0000	1111	1111

Description: This instruction provides a way to execute a $\overline{\text{MCLR}}$ Reset in software.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Start Reset	No operation	No operation

Example: `RESET`

After Instruction
Registers = Reset Value
Flags* = Reset Value

SUBLW **Subtract W from Literal**

Syntax: [*label*] SUBLW k

Operands: $0 \le k \le 255$

Operation: $k - (W) \rightarrow W$

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

Encoding:

0000	1000	kkkk	kkkk

Description: W is subtracted from the eight-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1:
```
SUBLW   0x02
```
Before Instruction
W = 1
C = ?
After Instruction
W = 1
C = 1 ; result is positive
Z = 0
N = 0

Example 2:
```
SUBLW   0x02
```
Before Instruction
W = 2
C = ?
After Instruction
W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3:
```
SUBLW   0x02
```
Before Instruction
W = 3
C = ?
After Instruction
W = FF ; (2's complement)
C = 0 ; result is negative
Z = 0
N = 1

SUBWF **Subtract W from f**

Syntax: [*label*] SUBWF f [,d [,a]

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

Operation: $(f) - (W) \rightarrow$ dest

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

Encoding:

0101	11da	ffff	ffff

Description: Subtract W from register 'f' (2's complement method). 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 will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1:
```
SUBWF   REG, 1, 0
```
Before Instruction
REG = 3
W = 2
C = ?
After Instruction
REG = 1
W = 2
C = 1 ; result is positive
Z = 0
N = 0

Example 2:
```
SUBWF   REG, 0, 0
```
Before Instruction
REG = 2
W = 2
C = ?
After Instruction
REG = 2
W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3:
```
SUBWF   REG, 1, 0
```
Before Instruction
REG = 1
W = 2
C = ?
After Instruction
REG = FFh ;(2's complement)
W = 2
C = 0 ; result is negative
Z = 0
N = 1

TBLWT Table Write

Syntax: [*label*] TBLWT (*; *+; *-; +*)

Operands: None

Operation:
if TBLWT*
(TABLAT) → Holding Register;
TBLPTR – No Change
if TBLWT*+
(TABLAT) → Holding Register;
(TBLPTR) + 1 → TBLPTR
if TBLWT*-
(TABLAT) → Holding Register;
(TBLPTR) – 1 → TBLPTR
if TBLWT+*
(TBLPTR) + 1 → TBLPTR;
(TABLAT) → Holding Register

Status Affected: None

Encoding:

0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*

Description: This instruction uses the 3 LSBs of TBLPTR to determine which of the 8 holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to **Section 5.0 “Flash Program Memory”** for additional details on programming Flash memory.)
The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-MByte address range. The LSB of the TBLPTR selects which byte of the program memory location to access.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word

TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLWT instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

TBLWT Table Write (Continued)

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read TABLAT)	No operation	No operation (Write to Holding Register)

Example 1:
```
TBLWT  *+;
```
Before Instruction
TABLAT = 0x55
TBLPTR = 0x00A356
HOLDING REGISTER (0x00A356) = 0xFF
After Instructions (table write completion)
TABLAT = 0x55
TBLPTR = 0x00A357
HOLDING REGISTER (0x00A356) = 0x55

Example 2:
```
TBLWT  +*;
```
Before Instruction
TABLAT = 0x34
TBLPTR = 0x01389A
HOLDING REGISTER (0x01389A) = 0xFF
HOLDING REGISTER (0x01389B) = 0xFF
After Instruction (table write completion)
TABLAT = 0x34
TBLPTR = 0x01389B
HOLDING REGISTER (0x01389A) = 0xFF
HOLDING REGISTER (0x01389B) = 0x34

NOTES:

27.4.2 TIMING CONDITIONS

The temperature and voltages specified in Table 27-5 apply to all timing specifications, unless otherwise noted. Figure 27-4 specifies the load conditions for the timing specifications.

TABLE 27-5: TEMPERATURE AND VOLTAGE SPECIFICATIONS – AC

AC CHARACTERISTICS	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended Operating voltage VDD range as described in DC spec **Section 27.1** and **Section 27.3**. LF parts operate for industrial temperatures only.

FIGURE 27-4: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS

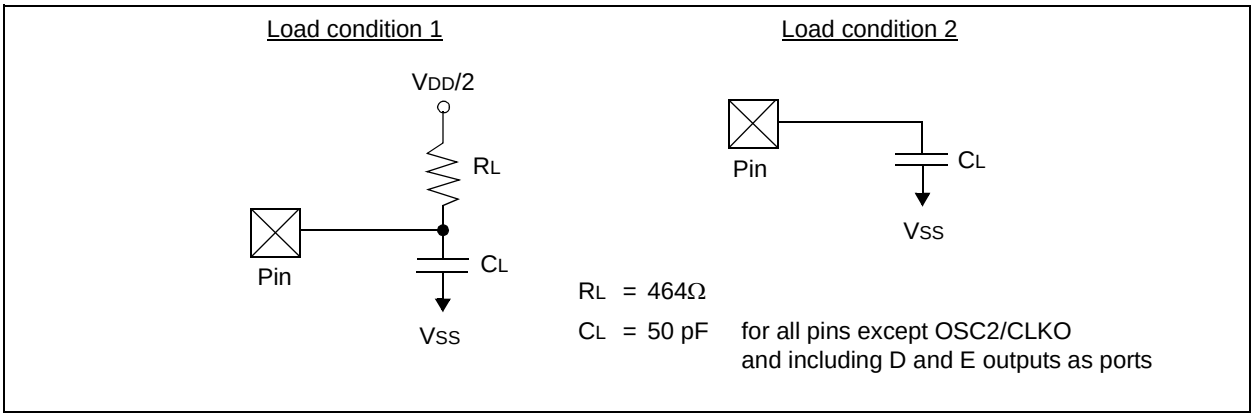
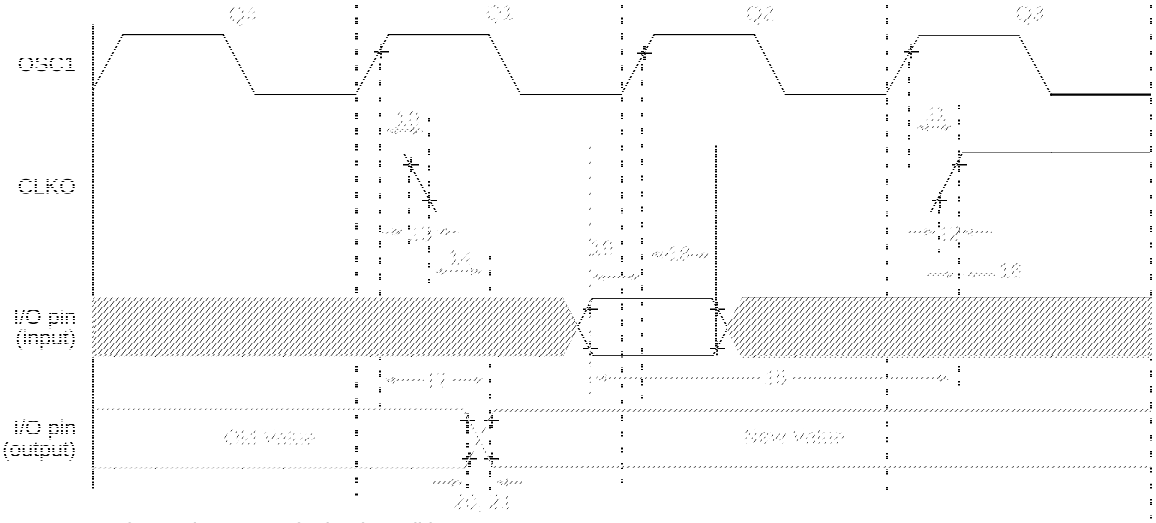


TABLE 27-7: PLL CLOCK TIMING SPECIFICATIONS (V_{DD} = 4.2 TO 5.5V)

Param. No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
	FOSC	Oscillator Frequency Range	4	—	10	MHz	HS mode
	FSYS	On-Chip VCO System Frequency	16	—	40	MHz	HS mode
	t_{rc}	PLL Start-up Time (Lock Time)	—	—	2	ms	
	ΔCLK	CLKO Stability (Jitter)	-2	—	+2	%	

† Data in "Typ" column is at 5V, 25°C, unless otherwise stated. These parameters are for design guidance only and are not tested.

FIGURE 27-6: CLKO AND I/O TIMING



Note: Refer to Figure 27-4 for load conditions.

TABLE 27-8: CLKO AND I/O TIMING REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Typ	Max	Units	Conditions
10	TosH2ckL	OSC1 ↑ to CLKO ↓		—	75	200	ns	**(Note 1)**
11	TosH2ckH	OSC1 ↑ to CLKO ↑		—	75	200	ns	**(Note 1)**
12	TckR	CLKO Rise Time		—	35	100	ns	**(Note 1)**
13	TckF	CLKO Fall Time		—	35	100	ns	**(Note 1)**
14	TckL2ioV	CLKO ↓ to Port Out Valid		—	—	0.5 TCY + 20	ns	**(Note 1)**
15	TioV2ckH	Port In Valid before CLKO ↑		0.25 TCY + 25	—	—	ns	**(Note 1)**
16	TckH2ioI	Port In Hold after CLKO ↑		0	—	—	ns	**(Note 1)**
17	TosH2ioV	OSC1 ↑ (Q1 cycle) to Port Out Valid		—	50	150	ns	
18	TosH2ioI	OSC1 ↑ (Q2 cycle) to Port Input Invalid (I/O in hold time)	PIC18F6525/6621/8525/8621	100	—	—	ns	
18A			PIC18LF6X2X/8X2X	200	—	—	ns	
19	TioV2osH	Port Input Valid to OSC1 ↑ (I/O in setup time)		0	—	—	ns	
20	TioR	Port Output Rise Time	PIC18F6525/6621/8525/8621	—	10	25	ns	
20A			PIC18LF6X2X/8X2X	—	—	60	ns	
21	TioF	Port Output Fall Time	PIC18F6525/6621/8525/8621	—	10	25	ns	
21A			PIC18LF6X2X/8X2X	—	—	60	ns	

† These parameters are asynchronous events not related to any internal clock edges.

Note 1: Measurements are taken in RC mode, where CLKO output is 4 x TOSC.

APPENDIX E: MIGRATION FROM HIGH-END TO ENHANCED DEVICES

A detailed discussion of the migration pathway and differences between the high-end MCU devices (i.e., PIC17CXXX) and the enhanced devices (i.e., PIC18FXXXX) is provided in *AN726, "PIC17CXXX to PIC18CXXX Migration*."

This Application Note is available as Literature Number DS00726.

P