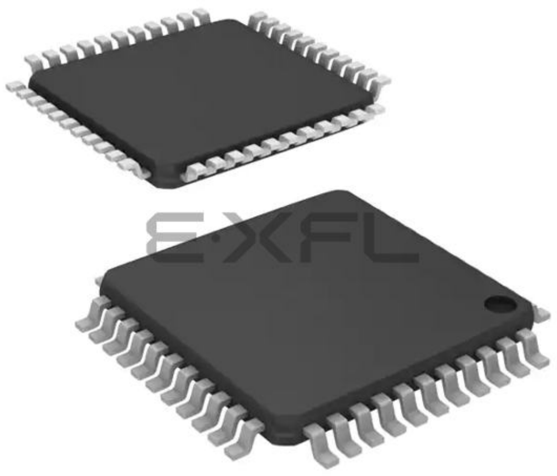
Microchip Technology - PIC18F46K42-I/PT Datasheet



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	Active
Core Processor	PIC
Core Size	8-Bit
Speed	64MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, 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	4K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 35x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f46k42-i-pt

3.1 System Arbitration

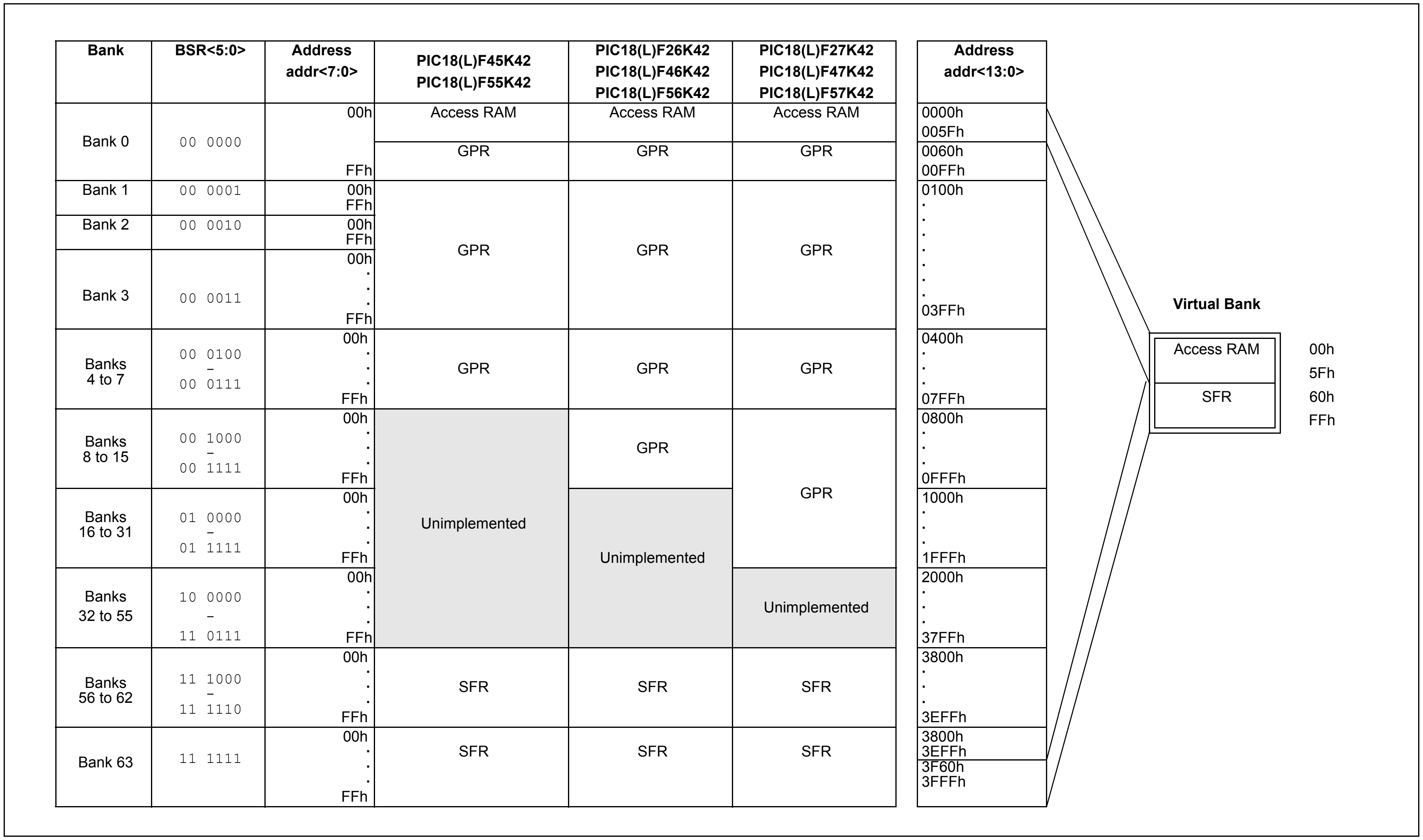
The System Arbiter resolves memory access between the System Level Selections (i.e., Main, Interrupt Service Routine) and Peripheral Selection (i.e., DMA and Scanner) based on user-assigned priorities. Each of the system level and peripheral selections has its own priority selection registers. Memory access priority is resolved using the number written to the corresponding Priority registers, 0 being the highest priority and 4 the lowest. The default priorities are listed in Table 3-1.

In case the user wants to change priorities, ensure each Priority register is written with a unique value from 0 to 4.

TABLE 3-1: DEFAULT PRIORITIES

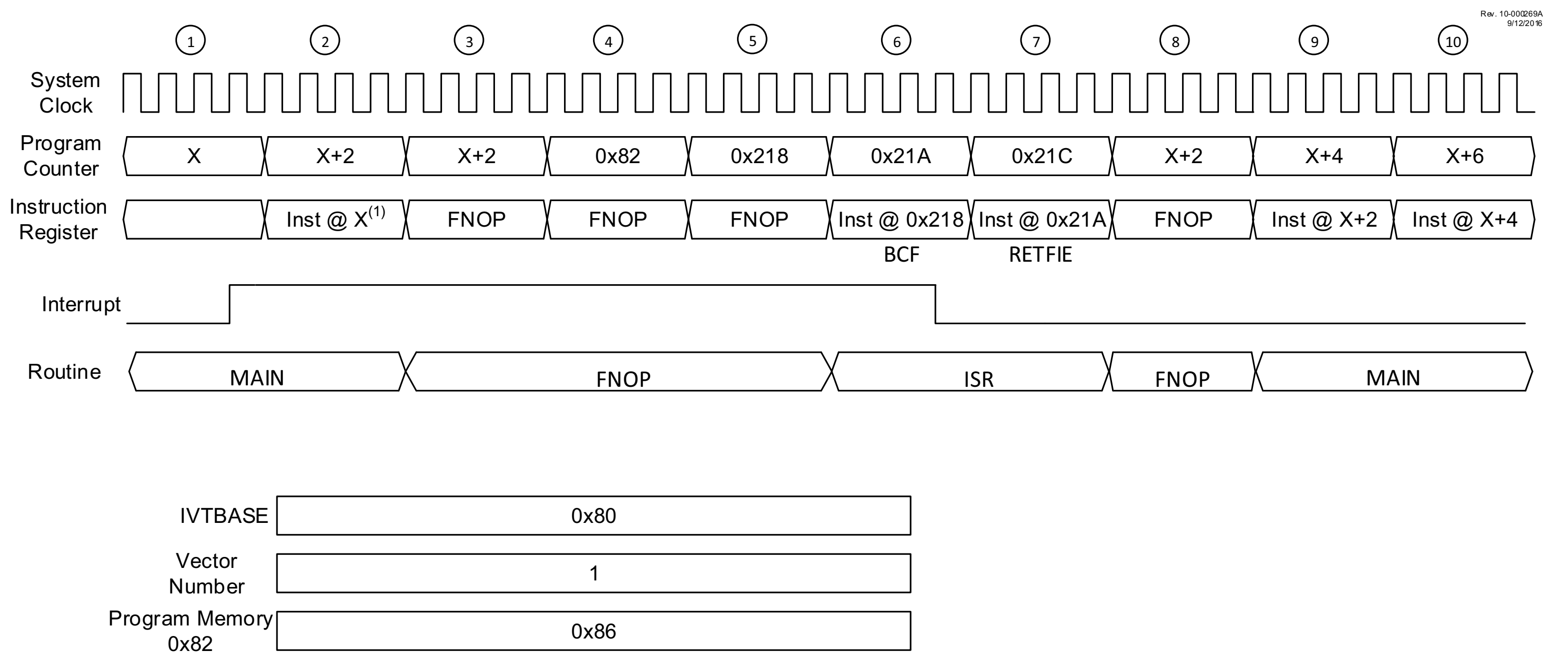
Selection		Priority register Reset value
System Level	ISR	0
	MAIN	1
Peripheral	DMA1	2
	DMA2	3
	SCANNER	4

FIGURE 4-4: DATA MEMORY MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES



Bank	BSR<5:0>	Address addr<7:0>	PIC18(L)F45K42 PIC18(L)F55K42	PIC18(L)F26K42 PIC18(L)F46K42 PIC18(L)F56K42	PIC18(L)F27K42 PIC18(L)F47K42 PIC18(L)F57K42	Address addr<13:0>
Bank 0	00 0000	00h	Access RAM	Access RAM	Access RAM	0000h 005Fh
		FFh	GPR	GPR	GPR	0060h 00FFh
Bank 1	00 0001	00h FFh	GPR	GPR	GPR	0100h 03FFh
Bank 2	00 0010	00h FFh				
Bank 3	00 0011	00h . . . FFh				
Banks 4 to 7	00 0100 – 00 0111	00h . . . FFh	GPR	GPR	GPR	0400h . . . 07FFh
Banks 8 to 15	00 1000 – 00 1111	00h . . . FFh	Unimplemented	GPR	GPR	0800h . . . 0FFFh
Banks 16 to 31	01 0000 – 01 1111	00h . . . FFh		Unimplemented		1000h . . . 1FFFh
Banks 32 to 55	10 0000 – 11 0111	00h . . . FFh			Unimplemented	2000h . . . 37FFh
Banks 56 to 62	11 1000 – 11 1110	00h . . . FFh	SFR	SFR	SFR	3800h . . . 3EFFh
Bank 63	11 1111	00h . . . FFh	SFR	SFR	SFR	3800h 3EFFh 3F60h 3FFFh

FIGURE 9-7: INTERRUPT TIMING DIAGRAM - ONE CYCLE INSTRUCTION



Interrupt Location = Interrupt vector table entry << 2
= 0x86 << 2 = 0x218

Note 1: Instruction @ X is a One-cycle Instruction

REGISTER 9-17: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
TMR0IE	U1IE	U1EIE	U1TXIE	U1RXIE	I2C1EIE	I2C1IE	I2C1TXIE
bit 7							bit 0

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

bit 7	**TMR0IE:** TMR0 Interrupt Enable bit 1 = Enabled 0 = Disabled
bit 6	**U1IE:** UART1 Interrupt Enable bit 1 = Enabled 0 = Disabled
bit 5	**U1EIE:** UART1 Framing Error Interrupt Enable bit 1 = Enabled 0 = Disabled
bit 4	**U1TXIE:** UART1 Transmit Interrupt Enable bit 1 = Enabled 0 = Disabled
bit 3	**U1RXIE:** UART1 Receive Interrupt Enable bit 1 = Enabled 0 = Disabled
bit 2	**I2C1EIE:** I^2C1 Error Interrupt Enable bit 1 = Enabled 0 = Disabled
bit 1	**I2C1IE:** I^2C1 Interrupt Enable bit 1 = Enabled 0 = Disabled
bit 0	**I2C1TXIE:** I^2C1 Transmit Interrupt Enable bit 1 = Enabled 0 = Disabled

10.2.3.2 Peripheral Usage in Sleep

Some peripherals that can operate in Sleep mode will not operate properly with the Low-Power Sleep mode selected. The Low-Power Sleep mode is intended for use with these peripherals:

- Brown-out Reset (BOR)
- Windowed Watchdog Timer (WWDT)
- External interrupt pin/Interrupt-On-Change pins
- Peripherals that run off external secondary clock source

It is the responsibility of the end user to determine what is acceptable for their application when setting the VREGPM settings in order to ensure operation in Sleep.

Note:	The PIC18F26/27/45/46/47/55/56/57K42 devices do not have a configurable Low-Power Sleep mode. PIC18F26/27/45/46/47/55/56/57K42 devices are unregulated and are always in the lowest power state when in Sleep, with no wake-up time penalty. These devices have a lower maximum VDD and I/O voltage than the PIC18(L)F26/27/45/46/47/55/56/57K42. See **Section 44.0 “Electrical Specifications”** for more information.

10.2.4 IDLE MODE

When IDLEN is set (IDLEN = `1`), the `SLEEP` instruction will put the device into Idle mode. In Idle mode, the CPU and memory operations are halted, but the peripheral clocks continue to run. This mode is similar to Doze mode, except that in IDLE both the CPU and PFM are shut off.

Note:	If $\overline{\text{CLKOUTEN}}$ is enabled ($\overline{\text{CLKOUTEN}}$ = `0`, Configuration Word 1H), the output will continue operating while in idle.

10.2.4.1 Idle and Interrupts

IDLE mode ends when an interrupt occurs (even if GIE = `0`), but IDLEN is not changed. The device can re-enter IDLE by executing the `SLEEP` instruction.

If Recover-On-Interrupt is enabled (ROI = `1`), the interrupt that brings the device out of idle also restores full-speed CPU execution when doze is also enabled.

10.2.4.2 Idle and WWDT

When in idle, the WWDT Reset is blocked and will instead wake the device. The WWDT wake-up is not an interrupt, therefore ROI does not apply.

Note:	The WDT can bring the device out of idle, in the same way it brings the device out of Sleep. The DOZEN bit is not affected.

10.3 Peripheral Operation in Power Saving Modes

All selected clock sources and the peripherals running off them are active in both IDLE and DOZE mode. Only in Sleep mode, both the FOSC and FOSC/4 clocks are unavailable. All the other clock sources are active, if enabled manually or through peripheral clock selection before the part enters Sleep.

EXAMPLE 13-4: WRITING TO PROGRAM FLASH MEMORY (CONTINUED)

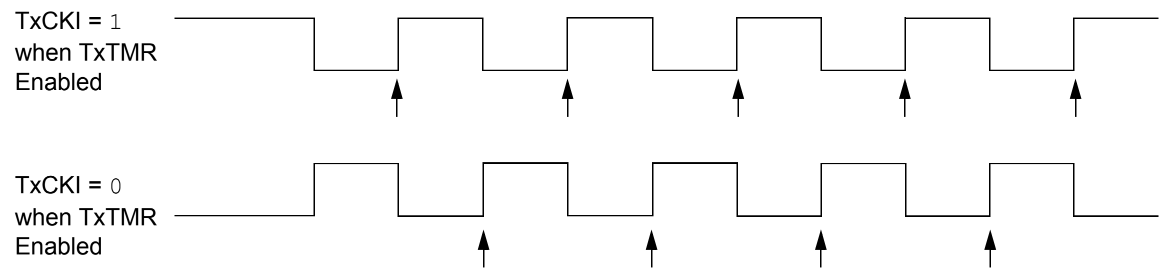
```
WRITE_BYTE_TO_HREGS
             MOVF      POSTINC0, W                ; get low byte of buffer data
             MOVWF     TABLAT                     ; present data to table latch
             TBLWT+*                              ; write data, perform a short write
                                                  ; to internal TBLWT holding register.
             DECFSZ    COUNTER                    ; loop until holding registers are full
             BRA       WRITE_WORD_TO_HREGS
PROGRAM_MEMORY
             BCF       NVMCON1, REG0              ; point to Program Flash Memory
             BSF       NVMCON1, REG1              ; point to Program Flash Memory
             BSF       NVMCON1, WREN              ; enable write to memory
             BCF       NVMCON1, FREE              ; enable write to memory
             BCF       INTCON0, GIE               ; disable interrupts
             MOVLW     55h
Required     MOVWF     NVMCON2                    ; write 55h
Sequence     MOVLW     0AAh
             MOVWF     NVMCON2                    ; write 0AAh
             BSF       NVMCON1, WR                ; start program (CPU stall)
             DCFSZ     COUNTER2                   ; repeat for remaining write blocks
             BRA       WRITE_BYTE_TO_HREGS
             BSF       INTCON0, GIE               ; re-enable interrupts
             BCF       NVMCON1, WREN              ; disable write to memory
```

TABLE 17-3: SUMMARY OF REGISTERS ASSOCIATED WITH THE PPS MODULE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
PPSLOCK	—	—	—	—	—	—	—	PPSLOCKED	283
INT0PPS	—	—	—	INT0PPS<4:0>					277
INT1PPS	—	—	—	INT1PPS<4:0>					277
INT2PPS	—	—	—	INT2PPS<4:0>					277
T0CKIPPS	—	—	—	T0CKIPPS<4:0>					277
T1CKIPPS	—	—	—	T1CKIPPS<4:0>					277
T1GPPS	—	—	—	T1GPPS<4:0>					277
T3CKIPPS	—	—	—	T3CKIPPS<4:0>					277
T3GPPS	—	—	—	T3GPPS<4:0>					277
T5CKIPPS	—	—	—	T5CKIPPS<4:0>					277
T5GPPS	—	—	—	T5GPPS<4:0>					277
T2INPPS	—	—	—	T2INPPS<4:0>					277
T4INPPS	—	—	—	T4INPPS<4:0>					277
T6INPPS	—	—	—	T6INPPS<4:0>					277
CCP1PPS	—	—	—	CCP1PPS<4:0>					277
CCP2PPS	—	—	—	CCP2PPS<4:0>					277
CCP3PPS	—	—	—	CCP3PPS<4:0>					277
CCP4PPS	—	—	—	CCP4PPS<4:0>					277
SMT1WINPPS	—	—	—	SMT1WINPPS<4:0>					277
SMT1SIGPPS	—	—	—	SMT1SIGPPS<4:0>					277
CWG1PPS	—	—	—	CWG1PPS<4:0>					277
CWG2PPS	—	—	—	CWG2PPS<4:0>					277
CWG3PPS	—	—	—	CWG3PPS<4:0>					277
MD1CARLPPS	—	—	—	MDCARLPPS<4:0>					277
MD1CARHPPS	—	—	—	MDCARHPPS<4:0>					277
MD1SRCPPS	—	—	—	MDSRCPPS<4:0>					277
CLCIN0PPS	—	—	—	CLCIN0PPS<4:0>					277
CLCIN1PPS	—	—	—	CLCIN1PPS<4:0>					277
CLCIN2PPS	—	—	—	CLCIN2PPS<4:0>					277
CLCIN3PPS	—	—	—	CLCIN3PPS<4:0>					277
ADACTPPS	—	—	—	ADACTPPS<4:0>					277
SPI1SCKPPS	—	—	—	SPI1SCKPPS<4:0>					277
SPI1SDIPPS	—	—	—	SPI1SDIPPS<4:0>					277
SPI1SSPPS	—	—	—	SPI1SSPPS<4:0>					277
I2C1SCLPPS	—	—	—	I2C1SCLPPS<4:0>					277
I2C1SDAPPS	—	—	—	I2C1SDAPPS<4:0>					277
I2C2SCLPPS	—	—	—	I2C2SCLPPS<4:0>					277
I2C2SDAPPS	—	—	—	I2C2SDAPPS<4:0>					277
U1RXPPS	—	—	—	U1RXPPS<4:0>					277
U1CTSPPS	—	—	—	U1CTSPPS<4:0>					277
U2RXPPS	—	—	—	U2RXPPS<4:0>					277
U2CTSPPS	—	—	—	U2CTPPS<4:0>					277
RxyPPS	—	—	—	RxyPPS<4:0>					280

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the PPS module.

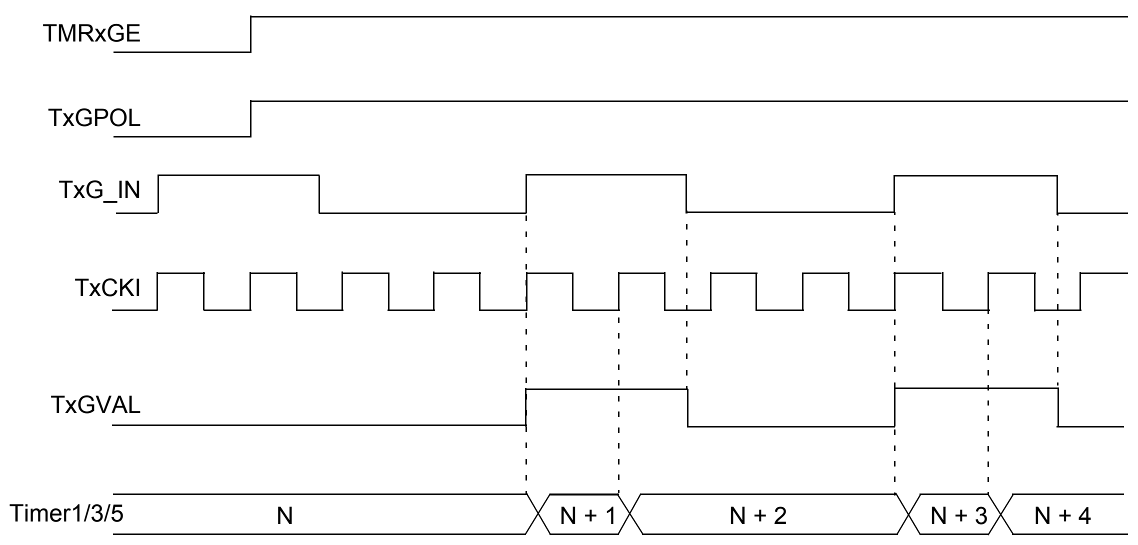
FIGURE 21-3: TIMER1/3/5 INCREMENTING EDGE



Note 1: Arrows indicate counter increments.

2: In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge of the clock.

FIGURE 21-4: TIMER1/3/5 GATE ENABLE MODE



REGISTER 30-2: MD1CON1: MODULATION CONTROL REGISTER 1

U-0	U-0	R/W-0/0	R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0
—	—	CHPOL	CHSYNC	—	—	CLPOL	CLSYNC
bit 7							bit 0

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

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

bit 5 **CHPOL:** Modulator High Carrier Polarity Select bit
1 = Selected high carrier signal is inverted
0 = Selected high carrier signal is not inverted

bit 4 **CHSYNC:** Modulator High Carrier Synchronization Enable bit
1 = Modulator waits for a falling edge on the high time carrier signal before allowing a switch to the low time carrier
0 = Modulator output is not synchronized to the high time carrier signal(1)

bit 3-2 **Unimplemented:** Read as ‘0’

bit 1 **CLPOL:** Modulator Low Carrier Polarity Select bit
1 = Selected low carrier signal is inverted
0 = Selected low carrier signal is not inverted

bit 0 **CLSYNC:** Modulator Low Carrier Synchronization Enable bit
1 = Modulator waits for a falling edge on the low time carrier signal before allowing a switch to the high time carrier
0 = Modulator output is not synchronized to the low time carrier signal(1)

Note 1: Narrowed carrier pulse widths or spurs may occur in the signal stream if the carrier is not synchronized.

TABLE 31-3: SUMMARY OF REGISTERS ASSOCIATED WITH THE UART

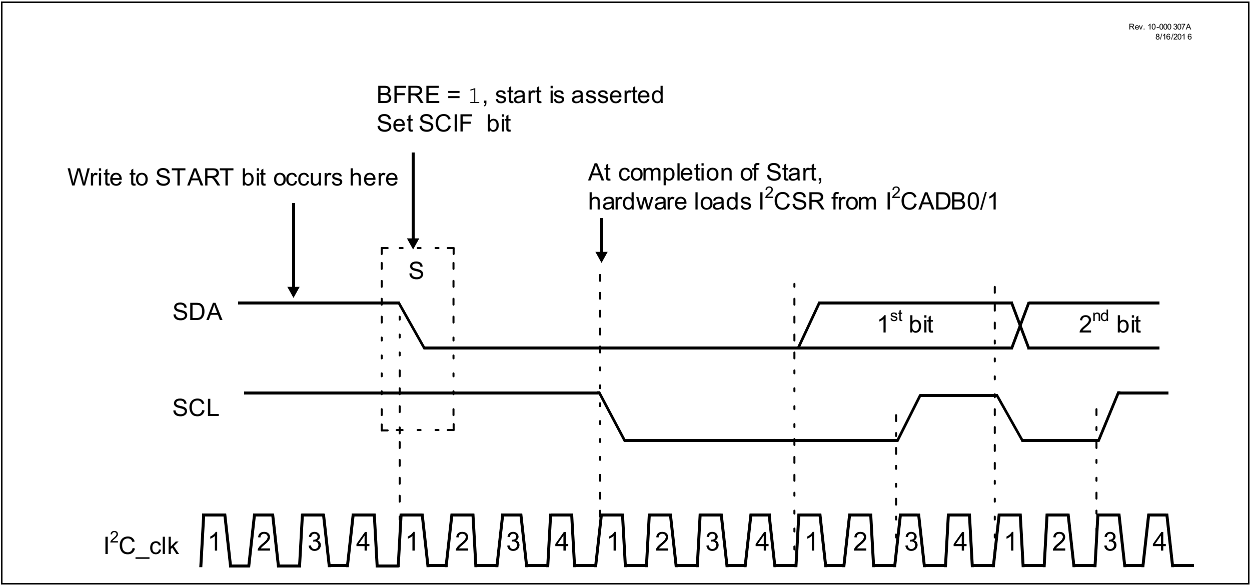
Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
UxCON0	BRGS	ABDEN	TXEN	RXEN	MODE<3:0>				498
UxCON1	ON	—	—	WUE	RXBIMD	—	BRKOVR	SENDB	499
UxCON2	RUNOVF	RXPOL	STP<1:0>		C0EN	TXPOL	FLO<1:0>		500
UxERRIR	TXMTIF	PERIF	ABDOVF	CERIF	FERIF	RXBKIF	RXFOIF	TXCIF	501
UxERRIE	TXMTIE	PERIE	ABDOVE	CERIE	FERIE;	RXBKIE	RXFOIE	TXCIE	502
UxUIR	WUIF	ABDIF	—	—	—	ABDIE	—	—	503
UxFIFO	TXWRE	STPMD	TXBE	TXBF	RXIDL	XON	RXBE	RXBF	504
UxBRGL	BRG<7:0>								505
UxBRGH	BRG<15:8>								505
UxRXB	RXB<7:0>								506
UxTXB	TXB<7:0>								506
UxP1H	—	—	—	—	—	—	—	P1<8>	507
UxP1L	P1<7:0>								507
UxP2H	—	—	—	—	—	—	—	P2<8>	508
UxP2L	P2<7:0>								508
UxP3H	—	—	—	—	—	—	—	P3<8>	509
UxP3L	P3<7:0>								509
UxTXCHK	TXCHK<7:0>								510
UxRXCHK	RXCHK<7:0>								510

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the UART module.

33.5.5 I^{2}C MASTER MODE START CONDITION TIMING

The user can initiate a Start condition by either writing to the Start bit (S) of the I2CxCON0 register or by writing to the I2CxTXB register based on the ABD bit setting. Master hardware waits for BFRE = `1`, before asserting the Start condition. The action of the SDA being driven low while SCL is high is the Start condition, causing the SCIF bit to be set. One TSCL later the SCL is asserted low, ending the start sequence. Figure 33-15 shows the Start condition timing.

FIGURE 33-15: START CONDITION TIMING



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

A Repeated Start condition occurs when the Start bit of the I2CxCON0 register is set and the master module is waiting from a Restart clock stretch event (RSEN = `1` and I2CxCNT = `0`).

When the Start bit is set, the SDA pin is released high for TSCL/2. Then the SCL pin is released floated high) for TSCL/2. If the SDA pin is detected low, bus collision flag (BCLIF) is set and the master goes idle. If SDA is detected high, the SDA pin will be pulled low (Start condition) for TSCL. Last, SCL is asserted low and I2CxADB0/1 is loaded into the shift register. As soon as a Restart condition is detected on the SDA and SCL pins, the RSCIF bit is set. Figure 33-16 shows the timings for repeated Start Condition.

REGISTER 33-9: I2CxCNT: I^{2}C BYTE COUNT REGISTER

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
CNT<7:0>							
bit 7							bit 0

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

bit 7-0 **CNT<7:0>:** I^{2}C Byte Count Register bits

If receiving data,

decremented 8th SCL edge, when a new data byte is loaded into I2CxRXB

If transmitting data,

decremented 9th SCL edge, when a new data byte is moved from I2CxTXB

CNTIF flag is set on 9th falling SCL edge, when I2CxCNT = 0. (Byte count cannot decrement past ‘0’)

Note 1: It is recommended to write this register only when the module is IDLE (MMA = 0, SMA = 0) or when clock stretching (CSTR = 1 || MDR = 1).

ADDWFC — ADD W and CARRY bit to f

Syntax:	ADDWFC f {,d {,a}}
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(W) + (f) + (C) $\rightarrow$ dest
Status Affected:	N,OV, C, DC, Z
Encoding:	`0010` `00da` `ffff` `ffff`
Description:	Add W, the CARRY flag and data memory location 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in data memory location 'f'. 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 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `ADDWFC    REG, 0, 1`

Before Instruction

CARRY bit	=	1
REG	=	02h
W	=	4Dh

After Instruction

CARRY bit	=	0
REG	=	02h
W	=	50h

ANDLW — AND literal with W

Syntax:	ANDLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .AND. k $\rightarrow$ W
Status Affected:	N, Z
Encoding:	`0000` `1011` `kkkk` `kkkk`
Description:	The contents of W are AND'ed with the 8-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: `ANDLW    05Fh`

Before Instruction

W	=	A3h

After Instruction

W	=	03h

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 2-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:
If Jump:

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

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE 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 2-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:
If Jump:

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

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BNN Jump`

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

BRA Unconditional Branch

Syntax:	BRA n
Operands:	-1024 ≤ n ≤ 1023
Operation:	(PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	`1101` `0nnn` `nnnn` `nnnn`
Description:	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 2-cycle instruction.
Words:	1
Cycles:	2

Q Cycle Activity:

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

Example: `HERE BRA Jump`

Before Instruction
PC = address `(HERE)`
After Instruction
PC = address `(Jump)`

BSF Bit Set f

Syntax:	BSF f, b {,a}
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]
Operation:	1 → f<b>
Status Affected:	None
Encoding:	`1000` `bbba` `ffff` `ffff`
Description:	Bit 'b' in register 'f' is set. 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 41.2.3 "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: `BSF FLAG_REG, 7, 1`

Before Instruction
FLAG_REG = 0Ah
After Instruction
FLAG_REG = 8Ah

INFSNZ **Increment f, skip if not 0**

Syntax:	INFSNZ f {,d {,a}}
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	$(f) + 1 \rightarrow$ dest, skip if result $\neq$ `0`
Status Affected:	None

Encoding:

`0100`	`10da`	`ffff`	`ffff`

Description: The contents of register 'f' are incremented. If 'd' is '`0`', the result is placed in W. If 'd' is '`1`', the result is placed back in register 'f' (default).
If the result is not '`0`', the next instruction, which is already fetched, is discarded and a `NOP` is executed instead, making it a 2-cycle instruction.
If 'a' is '`0`', the Access Bank is selected. If 'a' is '`1`', the BSR is used to select the GPR bank.
If 'a' is '`0`' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)

Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

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

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE    INFSNZ  REG, 1, 0
ZERO
NZERO
```

Before Instruction
PC = Address `(HERE)`
After Instruction
REG = REG + 1
If REG ≠ 0;
PC = Address `(NZERO)`
If REG = 0;
PC = Address `(ZERO)`

IORLW **Inclusive OR literal with W**

Syntax:	IORLW k
Operands:	$0 \leq k \leq 255$
Operation:	(W) .OR. $k \rightarrow$ W
Status Affected:	N, Z

Encoding:

`0000`	`1001`	`kkkk`	`kkkk`

Description: The contents of W are ORed with the 8-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: `IORLW 35h`

Before Instruction
W = 9Ah
After Instruction
W = BFh

SUBWF — Subtract W from f

Syntax:	SUBWF f {,d {,a}}
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f) – (W) → 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 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 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

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

```
Example 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
```

SUBWFB — Subtract W from f with Borrow

Syntax:	SUBWFB f {,d {,a}}
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f) – (W) – ($\overline{C}$) → dest
Status Affected:	N, OV, C, DC, Z
Encoding:	`0101` `10da` `ffff` `ffff`
Description:	Subtract W and the CARRY flag (borrow) 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 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 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

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

```
Example 1:        SUBWFB  REG, 1, 0
    Before Instruction
        REG     =   19h     (0001 1001)
        W       =   0Dh     (0000 1101)
        C       =   1
    After Instruction
        REG     =   0Ch     (0000 1100)
        W       =   0Dh     (0000 1101)
        C       =   1
        Z       =   0
        N       =   0       ; result is positive
Example 2:        SUBWFB  REG, 0, 0
    Before Instruction
        REG     =   1Bh     (0001 1011)
        W       =   1Ah     (0001 1010)
        C       =   0
    After Instruction
        REG     =   1Bh     (0001 1011)
        W       =   00h
        C       =   1
        Z       =   1       ; result is zero
        N       =   0
Example 3:        SUBWFB  REG, 1, 0
    Before Instruction
        REG     =   03h     (0000 0011)
        W       =   0Eh     (0000 1110)
        C       =   1
    After Instruction
        REG     =   F5h     (1111 0101)
                            ; [2's comp]
        W       =   0Eh     (0000 1110)
        C       =   0
        Z       =   0
        N       =   1       ; result is negative
```

41.2.5 SPECIAL CONSIDERATIONS WITH MICROCHIP MPLAB® IDE TOOLS

The latest versions of Microchip's software tools have been designed to fully support the extended instruction set of the PIC18(L)F2x/4x/5xK42 family of devices. This includes the MPLAB C18 C compiler, MPASM assembly language and MPLAB Integrated Development Environment (IDE).

When selecting a target device for software development, MPLAB IDE will automatically set default Configuration bits for that device. The default setting for the XINST Configuration bit is '0', disabling the extended instruction set and Indexed Literal Offset Addressing mode. For proper execution of applications developed to take advantage of the extended instruction set, XINST must be set during programming.

To develop software for the extended instruction set, the user must enable support for the instructions and the Indexed Addressing mode in their language tool(s). Depending on the environment being used, this may be done in several ways:

- A menu option, or dialog box within the environment, that allows the user to configure the language tool and its settings for the project
- A command line option
- A directive in the source code

These options vary between different compilers, assemblers and development environments. Users are encouraged to review the documentation accompanying their development systems for the appropriate information.

FIGURE 44-12: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS

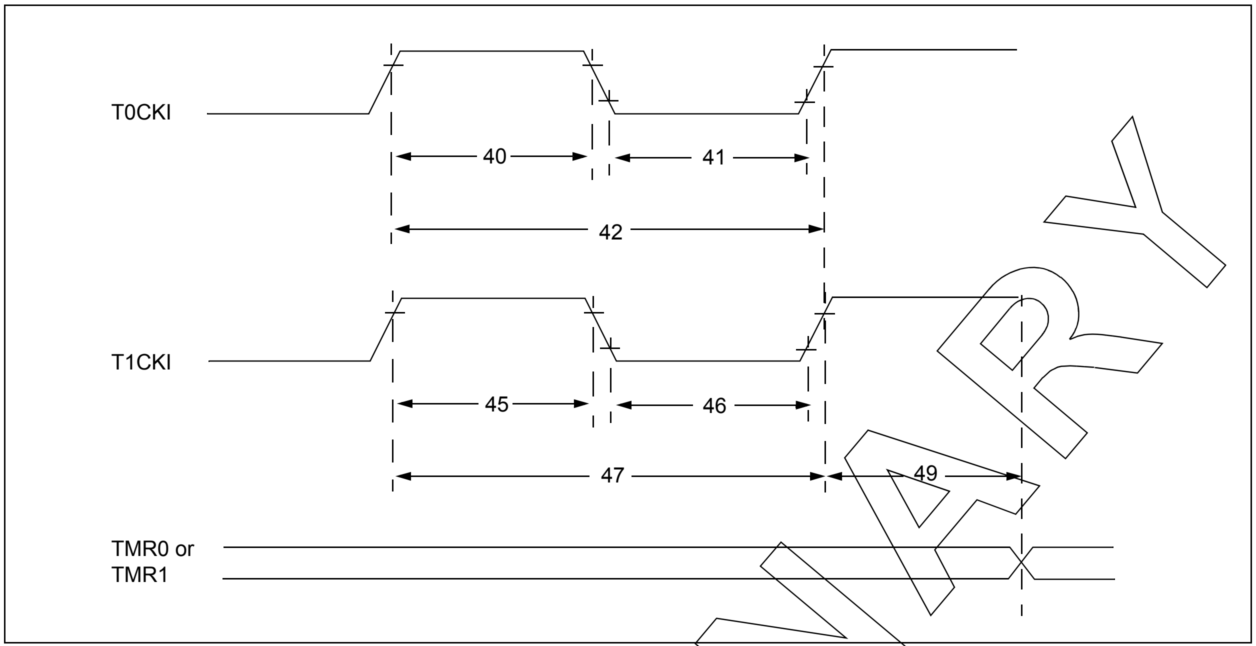


TABLE 44-21: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

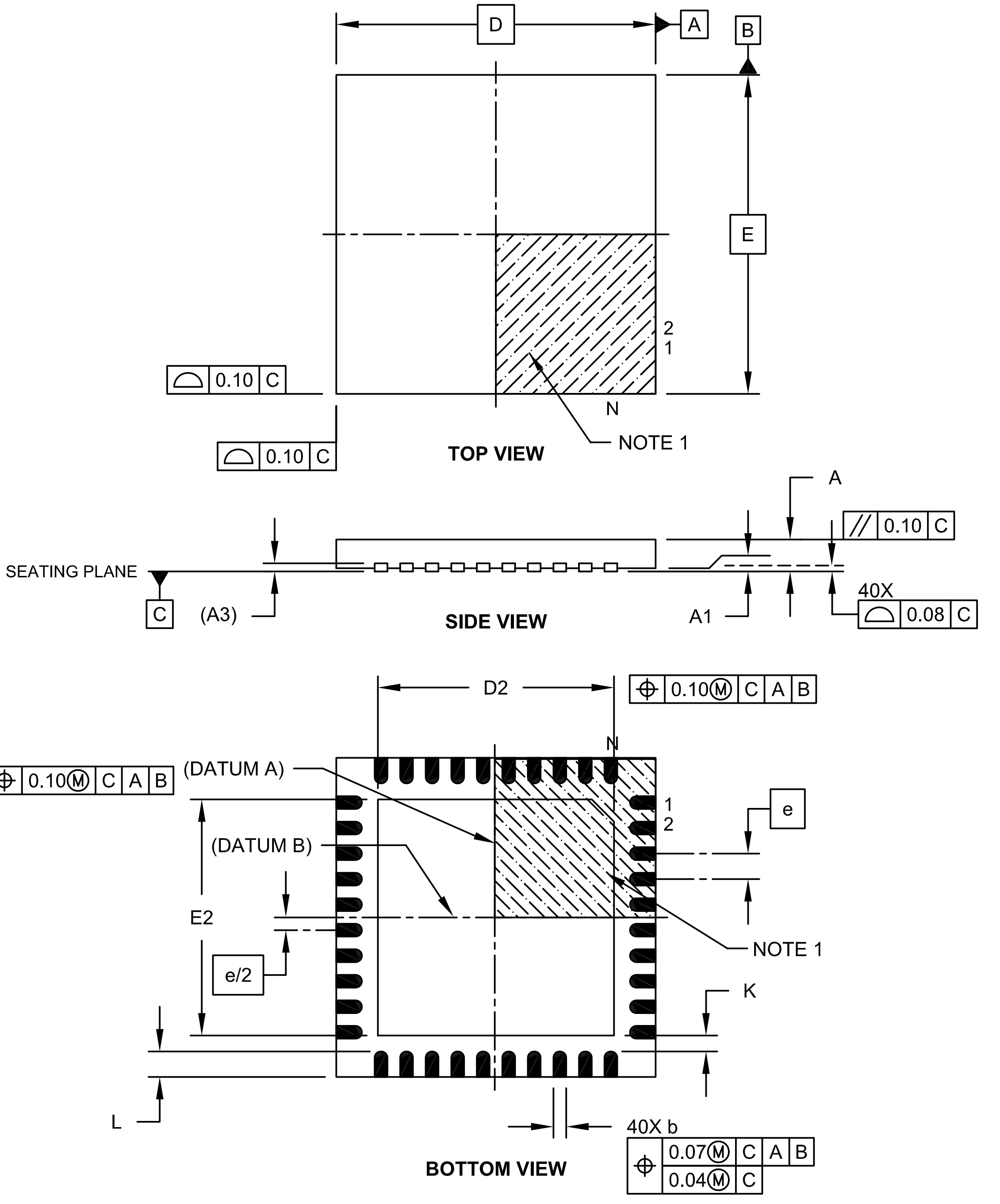
Standard Operating Conditions (unless otherwise stated)
Operating Temperature -40°C ≤ TA ≤ +125°C

Param No.	Sym.	Characteristic		Min.	Typ†	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		Greater of: 20 or $\frac{T_{CY} + 40}{N}$	—	—	ns	N = prescale value
45*	TT1H	T1CKI High Time	Synchronous, No Prescaler	0.5 TCY + 20	—	—	ns	
			Synchronous, with Prescaler	15	—	—	ns	
			Asynchronous	30	—	—	ns	
46*	TT1L	T1CKI Low Time	Synchronous, No Prescaler	0.5 TCY + 20	—	—	ns	
			Synchronous, with Prescaler	15	—	—	ns	
			Asynchronous	30	—	—	ns	
47*	TT1P	T1CKI Input Period	Synchronous	Greater of: 30 or $\frac{T_{CY} + 40}{N}$	—	—	ns	N = prescale value
			Asynchronous	60	—	—	ns	
49*	TCKEZTMR1	Delay from External Clock Edge to Timer Increment		2 TOSC	—	7 TOSC	—	Timers in Sync mode

* These parameters are characterized but not tested.
† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

40-Lead Ultra Thin Plastic Quad Flat, No Lead Package (MV) – 5x5x0.5 mm Body [UQFN]

Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-156A Sheet 1 of 2