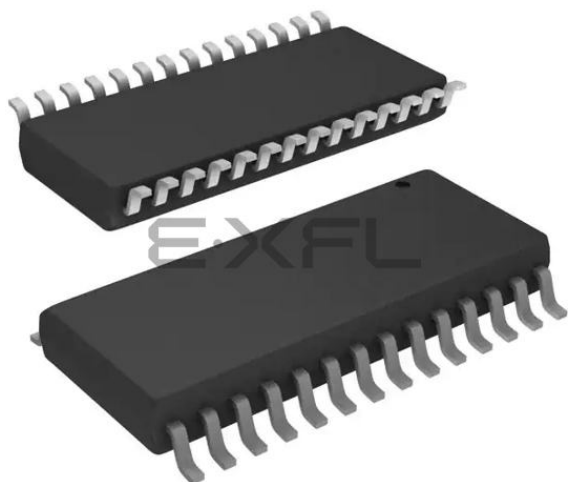




Welcome to [E-XFL.COM](https://www.e-xfl.com)

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](#)"

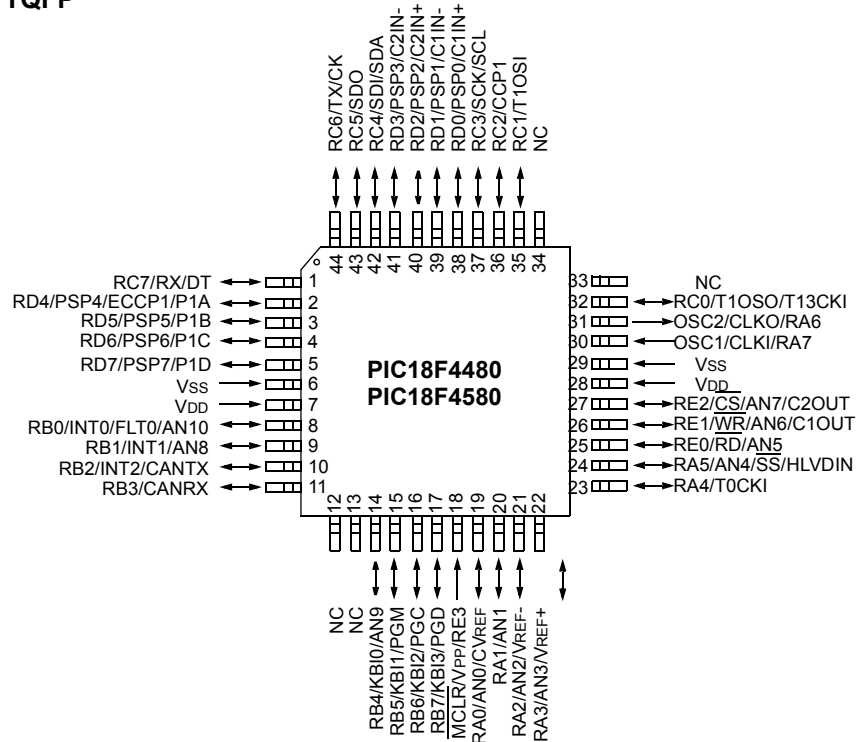
Details

Product Status	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	CANbus, I ² C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1.5K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf2580t-i-so

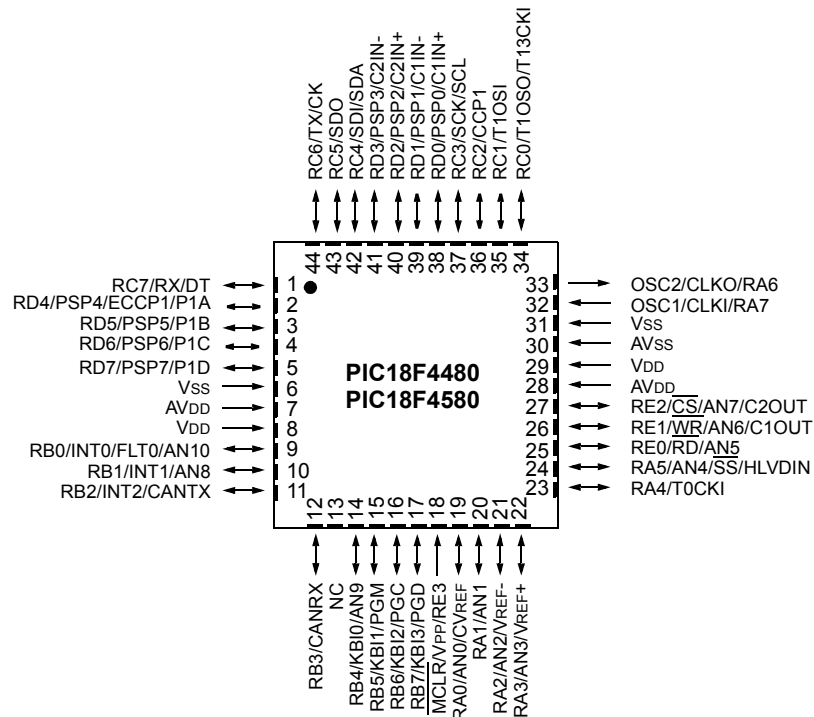
PIC18F2480/2580/4480/4580

Pin Diagrams (Continued)

44-Pin TQFP



44-Pin QFN⁽¹⁾



Note 1: For the QFN package, it is recommended that the bottom pad be connected to Vss.

PIC18F2480/2580/4480/4580

3.7 Clock Sources and Oscillator Switching

Like previous PIC18 devices, the PIC18F2480/2580/4480/4580 family includes a feature that allows the device clock source to be switched from the main oscillator to an alternate low-frequency clock source. PIC18F2480/2580/4480/4580 devices offer two alternate clock sources. When an alternate clock source is enabled, the various power-managed operating modes are available.

Essentially, there are three clock sources for these devices:

- Primary oscillators
- Secondary oscillators
- Internal oscillator block

The **primary oscillators** include the external crystal and resonator modes, the external RC modes, the external clock modes and the internal oscillator block. The particular mode is defined by the FOSC<3:0> Configuration bits. The details of these modes are covered earlier in this chapter.

The **secondary oscillators** are those external sources not connected to the OSC1 or OSC2 pins. These sources may continue to operate even after the controller is placed in a power-managed mode.

PIC18F2480/2580/4480/4580 devices offer the Timer1 oscillator as a secondary oscillator. This oscillator, in all power-managed modes, is often the time base for functions such as a Real-Time Clock (RTC).

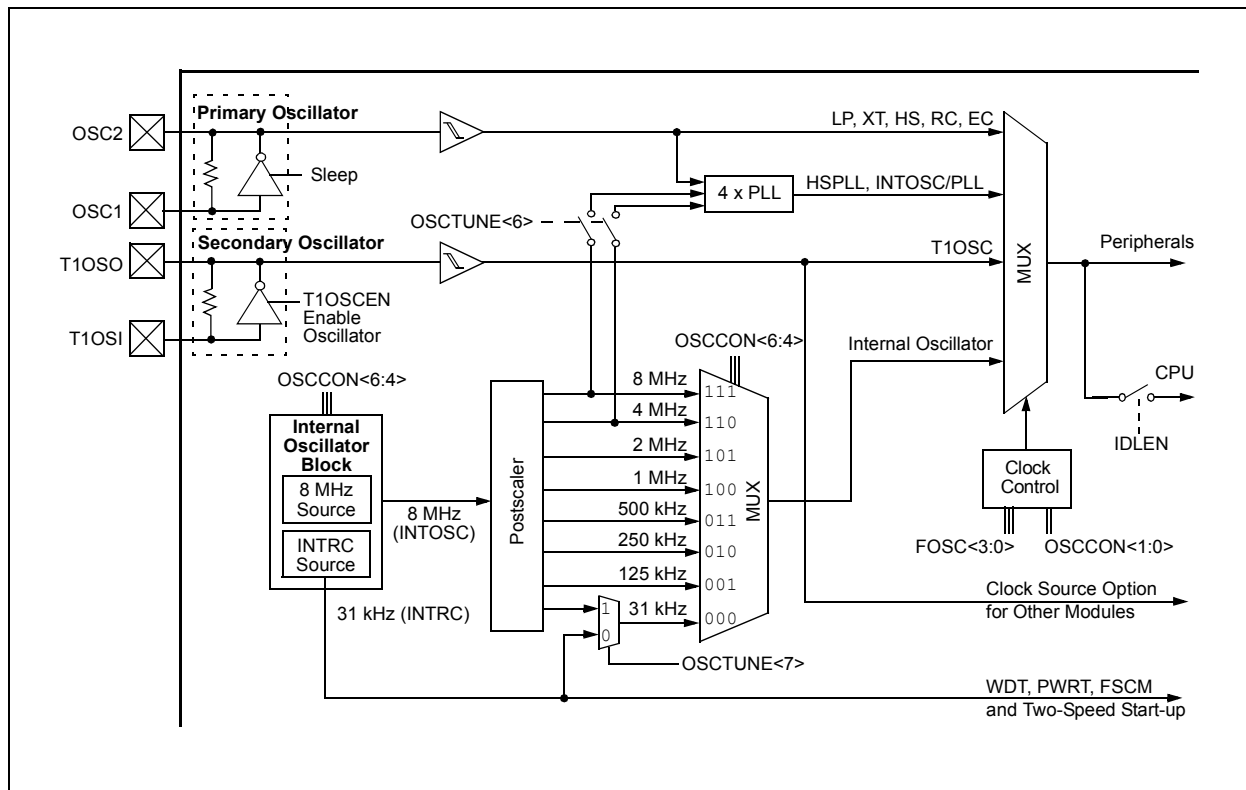
Most often, a 32.768 kHz watch crystal is connected between the RC0/T1OSO/T13CKI and RC1/T1OSI pins. Like the LP Oscillator mode circuit, loading capacitors are also connected from each pin to ground.

The Timer1 oscillator is discussed in greater detail in **Section 13.3 “Timer1 Oscillator”**.

In addition to being a primary clock source, the **internal oscillator block** is available as a power-managed mode clock source. The INTRC source is also used as the clock source for several special features, such as the WDT and Fail-Safe Clock Monitor.

The clock sources for the PIC18F2480/2580/4480/4580 devices are shown in Figure 3-8. See **Section 25.0 “Special Features of the CPU”** for Configuration register details.

FIGURE 3-8: PIC18F2480/2580/4480/4580 CLOCK DIAGRAM



PIC18F2480/2580/4480/4580

7.4 Erasing Flash Program Memory

The minimum erase block is 32 words or 64 bytes. Only through the use of an external programmer, or through ICSP control, can larger blocks of program memory be bulk erased. Word erase in the Flash array is not supported.

When initiating an erase sequence from the microcontroller itself, a block of 64 bytes of program memory is erased. The Most Significant 16 bits of the TBLPTR<21:5> point to the block being erased. TBLPTR<4:0> are ignored.

The EECON1 register commands the erase operation. The EEPGD bit must be set to point to the Flash program memory. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation.

For protection, the write initiate sequence for EECON2 must be used.

A long write is necessary for erasing the internal Flash. Instruction execution is halted while in a long write cycle. The long write will be terminated by the internal programming timer.

7.4.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

The sequence of events for erasing a block of internal program memory location is:

1. Load Table Pointer register with address of row being erased.
2. Set the EECON1 register for the erase operation:
 - set EEPGD bit to point to program memory;
 - clear the CFGS bit to access program memory;
 - set WREN bit to enable writes;
 - set FREE bit to enable the erase.
3. Disable interrupts.
4. Write 55h to EECON2.
5. Write 0AAh to EECON2.
6. Set the WR bit. This will begin the row erase cycle.
7. The CPU will stall for duration of the erase (about 2 ms using internal timer).
8. Re-enable interrupts.

EXAMPLE 7-2: ERASING A FLASH PROGRAM MEMORY ROW

	MOVLW	CODE_ADDR_UPPER	; load TBLPTR with the base
	MOVWF	TBLPTRU	; address of the memory block
	MOVLW	CODE_ADDR_HIGH	
	MOVWF	TBLPTRH	
	MOVLW	CODE_ADDR_LOW	
	MOVWF	TBLPTRL	
ERASE_ROW			
	BSF	EECON1, EEPGD	; point to Flash program memory
	BCF	EECON1, CFGS	; access Flash program memory
	BSF	EECON1, WREN	; enable write to memory
	BSF	EECON1, FREE	; enable Row Erase operation
	BCF	INTCON, GIE	; disable interrupts
Required Sequence	MOVLW	55h	
	MOVWF	EECON2	; write 55h
	MOVLW	0AAh	
	MOVWF	EECON2	; write 0AAh
	BSF	EECON1, WR	; start erase (CPU stall)
	BSF	INTCON, GIE	; re-enable interrupts

PIC18F2480/2580/4480/4580

8.6 Operation During Code-Protect

Data EEPROM memory has its own code-protect bits in Configuration Words. External read and write operations are disabled if code protection is enabled.

The microcontroller itself can both read and write to the internal data EEPROM, regardless of the state of the code-protect Configuration bit. Refer to **Section 25.0 “Special Features of the CPU”** for additional information.

8.7 Protection Against Spurious Write

There are conditions when the device may not want to write to the data EEPROM memory. To protect against spurious EEPROM writes, various mechanisms have been implemented. On power-up, the WREN bit is cleared. In addition, writes to the EEPROM are blocked during the Power-up Timer period (TPWRT, parameter 33).

The write initiate sequence and the WREN bit together help prevent an accidental write during brown-out, power glitch or software malfunction.

8.8 Using the Data EEPROM

The data EEPROM is a high-endurance, byte addressable array that has been optimized for the storage of frequently changing information (e.g., program variables or other data that are updated often). Frequently changing values will typically be updated more often than specification D124. If this is not the case, an array refresh must be performed. For this reason, variables that change infrequently (such as constants, IDs, calibration, etc.) should be stored in Flash program memory.

A simple data EEPROM refresh routine is shown in Example 8-3.

Note: If data EEPROM is only used to store constants and/or data that changes rarely, an array refresh is likely not required. See specification D124.

EXAMPLE 8-3: DATA EEPROM REFRESH ROUTINE

```
CLRF    EEADR          ; Start at address 0
BCF     EECON1, CFGS    ; Set for memory
BCF     EECON1, EEPGD    ; Set for Data EEPROM
BCF     INTCON, GIE      ; Disable interrupts
BSF     EECON1, WREN     ; Enable writes
LOOP:   BSF     EECON1, RD ; Loop to refresh array
        MOVLW   55h      ; Read current address
        MOVWF   EECON2    ;
        MOVLW   0AAh     ;
        MOVWF   EECON2    ; Write 0AAh
        BSF     EECON1, WR ; Set WR bit to begin write
        BTFSC   EECON1, WR ; Wait for write to complete
        BRA     $-2
        INCFSZ  EEADR, F  ; Increment address
        BRA     LOOP      ; Not zero, do it again

        BCF     EECON1, WREN ; Disable writes
        BSF     INTCON, GIE  ; Enable interrupts
```

PIC18F2480/2580/4480/4580

9.0 8 x 8 HARDWARE MULTIPLIER

9.1 Introduction

All PIC18 devices include an 8 x 8 hardware multiplier as part of the ALU. The multiplier performs an unsigned operation and yields a 16-bit result that is stored in the product register pair, PRODH:PRODL. The multiplier's operation does not affect any flags in the STATUS register.

Making multiplication a hardware operation allows it to be completed in a single instruction cycle. This has the advantages of higher computational throughput and reduced code size for multiplication algorithms and allows the PIC18 devices to be used in many applications previously reserved for digital signal processors. A comparison of various hardware and software multiply operations, along with the savings in memory and execution time, is shown in Table 9-1.

9.2 Operation

Example 9-1 shows the instruction sequence for an 8 x 8 unsigned multiplication. Only one instruction is required when one of the arguments is already loaded in the WREG register.

Example 9-2 shows the sequence to do an 8 x 8 signed multiplication. To account for the signed bits of the arguments, each argument's Most Significant bit (MSb) is tested and the appropriate subtractions are done.

EXAMPLE 9-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE

```
MOVF    ARG1, W    ;  
MULWF   ARG2        ; ARG1 * ARG2 ->  
                        ; PRODH:PRODL
```

EXAMPLE 9-2: 8 x 8 SIGNED MULTIPLY ROUTINE

```
MOVF    ARG1, W  
MULWF   ARG2        ; ARG1 * ARG2 ->  
                        ; PRODH:PRODL  
  
BTFSC   ARG2, SB    ; Test Sign Bit  
SUBWF   PRODH, F    ; PRODH = PRODH  
                        ;      - ARG1  
  
MOVF    ARG2, W  
BTFSC   ARG1, SB    ; Test Sign Bit  
SUBWF   PRODH, F    ; PRODH = PRODH  
                        ;      - ARG2
```

TABLE 9-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS

Routine	Multiply Method	Program Memory (Words)	Cycles (Max)	Time		
				@ 40 MHz	@ 10 MHz	@ 4 MHz
8 x 8 unsigned	Without hardware multiply	13	69	6.9 μ s	27.6 μ s	69 μ s
	Hardware multiply	1	1	100 ns	400 ns	1 μ s
8 x 8 signed	Without hardware multiply	33	91	9.1 μ s	36.4 μ s	91 μ s
	Hardware multiply	6	6	600 ns	2.4 μ s	6 μ s
16 x 16 unsigned	Without hardware multiply	21	242	24.2 μ s	96.8 μ s	242 μ s
	Hardware multiply	28	28	2.8 μ s	11.2 μ s	28 μ s
16 x 16 signed	Without hardware multiply	52	254	25.4 μ s	102.6 μ s	254 μ s
	Hardware multiply	35	40	4.0 μ s	16.0 μ s	40 μ s

11.4 PORTD, TRISD and LATD Registers

Note: PORTD is only available on PIC18F4X80 devices.

PORTD is an 8-bit wide, bidirectional port. The corresponding Data Direction register is TRISD. Setting a TRISD bit (= 1) will make the corresponding PORTD pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISD bit (= 0) will make the corresponding PORTD pin an output (i.e., put the contents of the output latch on the selected pin).

The Output Latch register (LATD) is also memory mapped. Read-modify-write operations on the LATD register read and write the latched output value for PORTD.

All pins on PORTD are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

Four of the PORTD pins are multiplexed with outputs P1A, P1B, P1C and P1D of the Enhanced CCP module. The operation of these additional PWM output pins is covered in greater detail in **Section 17.0 “Enhanced Capture/Compare/PWM (ECCP) Module”**.

Four of the PORTD pins are multiplexed with the input pins of the comparators. The operation of these input pins is covered in greater detail in **Section 21.0 “Comparator Module”**.

Note: On a Power-on Reset, these pins are configured as analog inputs.

PORTD can also be configured as an 8-bit wide micro-processor port (Parallel Slave Port) by setting control bit, PSPMODE (TRISE<4>). In this mode, the input buffers are TTL. See **Section 11.6 “Parallel Slave Port”** for additional information on the Parallel Slave Port (PSP).

EXAMPLE 11-4: INITIALIZING PORTD

```
CLRF    PORTD    ; Initialize PORTD by
                  ; clearing output
                  ; data latches
CLRF    LATD     ; Alternate method
                  ; to clear output
                  ; data latches
MOVLW   0CFh     ; Value used to
                  ; initialize data
                  ; direction
MOVWF   TRISD    ; Set RD<3:0> as inputs
                  ; RD<5:4> as outputs
                  ; RD<7:6> as inputs
```

PIC18F2480/2580/4480/4580

17.4.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the ECCPR1L register and to the ECCP1CON<5:4> bits. Up to 10-bit resolution is available. The ECCPR1L contains the eight MSBs and the ECCP1CON<5:4> bits contain the two LSBs. This 10-bit value is represented by ECCPR1L:ECCP1CON<5:4>. The PWM duty cycle is calculated by the following equation.

EQUATION 17-2:

$$\text{PWM Duty Cycle} = (\text{ECCPR1L:ECCP1CON<5:4>} \cdot \text{TOSC} \cdot (\text{TMR2 Prescale Value}))$$

ECCPR1L and ECCP1CON<5:4> can be written to at any time, but the duty cycle value is not copied into ECCPR1H until a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, ECCPR1H is a read-only register.

The ECCPR1H register and a 2-bit internal latch are used to double-buffer the PWM duty cycle. This double-buffering is essential for glitchless PWM operation. When the ECCPR1H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock or two bits of the TMR2 prescaler, the ECCP1 pin is cleared. The maximum PWM resolution (bits) for a given PWM frequency is given by the following equation.

EQUATION 17-3:

$$\text{PWM Resolution (max)} = \frac{\log\left(\frac{F_{\text{OSC}}}{F_{\text{PWM}}}\right)}{\log(2)} \text{ bits}$$

Note: If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

17.4.3 PWM OUTPUT CONFIGURATIONS

The EPWM1M<1:0> bits in the ECCP1CON register allow one of four configurations:

- Single Output
- Half-Bridge Output
- Full-Bridge Output, Forward mode
- Full-Bridge Output, Reverse mode

The Single Output mode is the standard PWM mode discussed in **Section 17.4 “Enhanced PWM Mode”**. The Half-Bridge and Full-Bridge Output modes are covered in detail in the sections that follow.

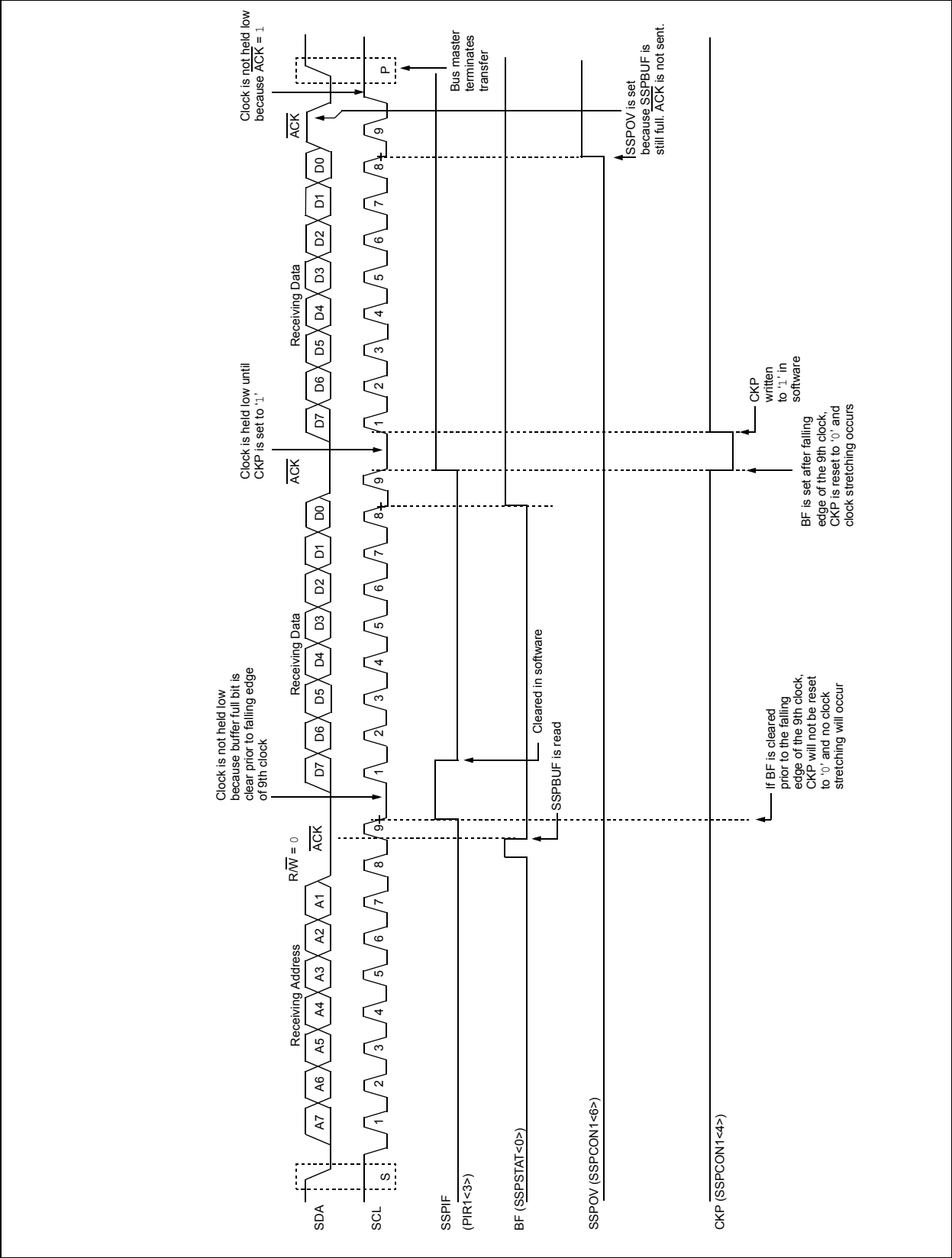
The general relationship of the outputs in all configurations is summarized in Figure 17-2.

TABLE 17-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 40 MHz

PWM Frequency	2.44 kHz	9.77 kHz	39.06 kHz	156.25 kHz	312.50 kHz	416.67 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	FFh	FFh	FFh	3Fh	1Fh	17h
Maximum Resolution (bits)	10	10	10	8	7	6.58

PIC18F2480/2580/4480/4580

FIGURE 18-13: I²C™ SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 7-BIT ADDRESS)



18.4.17.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

- After the SDA pin has been deasserted and allowed to float high, SDA is sampled low after the BRG has timed out.
- After the SCL pin is deasserted, SCL is sampled low before SDA goes high.

The Stop condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPADD<6:0> and counts down to 0. After the BRG times out, SDA is sampled. If SDA is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data '0' (Figure 18-31). If the SCL pin is sampled low before SDA is allowed to float high, a bus collision occurs. This is another case of another master attempting to drive a data '0' (Figure 18-32).

FIGURE 18-31: BUS COLLISION DURING A STOP CONDITION (CASE 1)

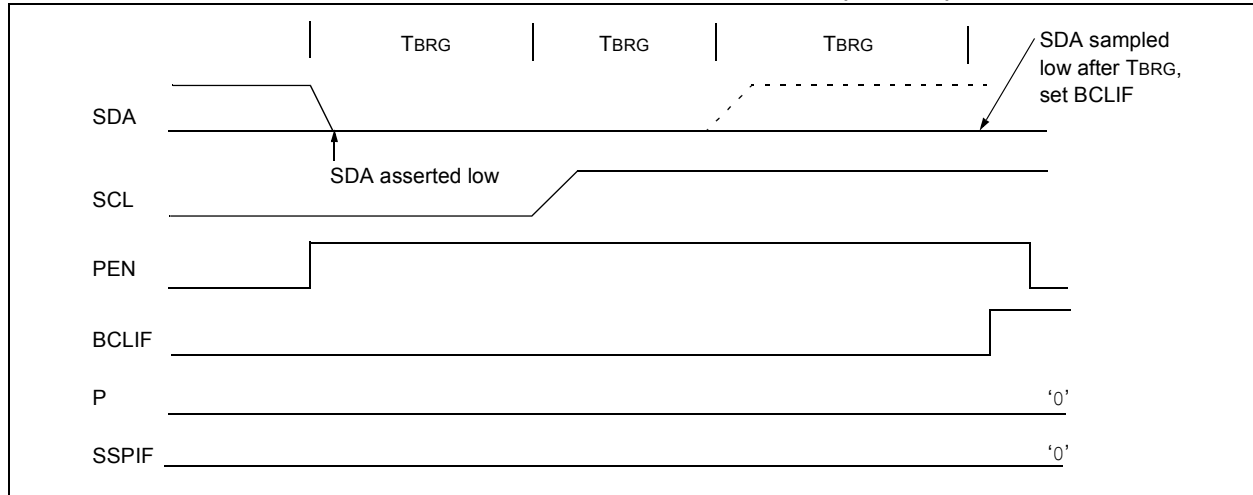
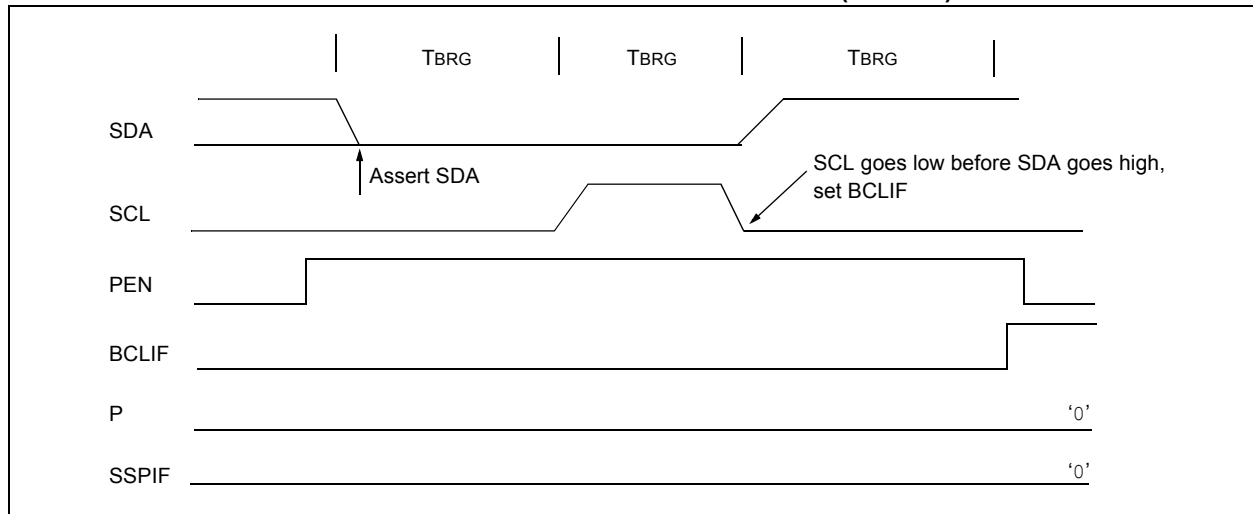


FIGURE 18-32: BUS COLLISION DURING A STOP CONDITION (CASE 2)



19.2 EUSART Asynchronous Mode

The Asynchronous mode of operation is selected by clearing the SYNC bit (TXSTA<4>). In this mode, the EUSART uses standard Non-Return-to-Zero (NRZ) format (one Start bit, eight or nine data bits and one Stop bit). The most common data format is 8 bits. An on-chip dedicated 8-bit/16-bit Baud Rate Generator can be used to derive standard baud rate frequencies from the oscillator.

The EUSART transmits and receives the LSb first. The EUSART's transmitter and receiver are functionally independent but use the same data format and baud rate. The Baud Rate Generator produces a clock, either x16 or x64 of the bit shift rate depending on the BRGH and BRG16 bits (TXSTA<2> and BAUDCON<3>). Parity is not supported by the hardware, but can be implemented in software and stored as the 9th data bit.

When operating in Asynchronous mode, the EUSART module consists of the following important elements:

In Asynchronous mode, clock polarity is selected with the TXCKP bit (BAUDCON<4>). Setting TXCKP sets the Idle state on CK as high, while clearing the bit sets the Idle state as low. Data polarity is selected with the RXDTP bit (BAUDCON<5>).

Setting RXDTP inverts data on RX, while clearing the bit has no affect on received data.

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver
- Auto-Wake-up on Sync Break Character
- 12-Bit Break Character Transmit
- Auto-Baud Rate Detection

19.2.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in Figure 19-3. The heart of the transmitter is the Transmit (Serial) Shift Register (TSR). The Shift register obtains its data from the Read/Write Transmit Buffer register, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the Stop bit has been transmitted from the previous load. As soon as the Stop bit is transmitted, the TSR is loaded with new data from the TXREG register (if available).

Once the TXREG register transfers the data to the TSR register (occurs in one Tcy), the TXREG register is empty and the TXIF flag bit (PIR1<4>) is set. This interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXIE (PIE1<4>). TXIF will be set regardless of the state of TXIE; it cannot be cleared in software. TXIF is also not cleared immediately upon loading TXREG, but becomes valid in the second instruction cycle following the load instruction. Polling TXIF immediately following a load of TXREG will return invalid results.

While TXIF indicates the status of the TXREG register, another bit, TRMT (TXSTA<1>), shows the status of the TSR register. TRMT is a read-only bit which is set when the TSR register is empty. No interrupt logic is tied to this bit so the user has to poll this bit in order to determine if the TSR register is empty.

Note 1: The TSR register is not mapped in data memory so it is not available to the user.

2: Flag bit, TXIF, is set when enable bit, TXEN, is set.

To set up an Asynchronous Transmission:

1. Initialize the SPBRGH:SPBRG 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, TXIE.
4. If 9-bit transmission is desired, set transmit bit, TX9. Can be used as address/data bit.
5. Enable the transmission by setting bit, TXEN, which will also set bit, TXIF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
7. Load data to the TXREG register (starts transmission).
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

19.2.5 BREAK CHARACTER SEQUENCE

The Enhanced EUSART module has the capability of sending the special Break character sequences that are required by the LIN/J2602 bus standard. The Break character transmit consists of a Start bit, followed by twelve '0' bits and a Stop bit. The Frame Break character is sent whenever the SENDB and TXEN bits (TXSTA<3> and TXSTA<5>) are set while the Transmit Shift register is loaded with data. Note that the value of data written to TXREG will be ignored and all '0's will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN/J2602 specification).

Note that the data value written to the TXREG for the Break character is ignored. The write simply serves the purpose of initiating the proper sequence.

The TRMT bit indicates when the transmit operation is active or Idle, just as it does during normal transmission. See Figure 19-10 for the timing of the Break character sequence.

19.2.5.1 Break and Sync Transmit Sequence

The following sequence will send a message frame header made up of a Break, followed by an Auto-Baud Sync byte. This sequence is typical of a LIN/J2602 bus master.

1. Configure the EUSART for the desired mode.
2. Set the TXEN and SENDB bits to set up the Break character.
3. Load the TXREG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREG to load the Sync character into the transmit FIFO buffer.
5. After the Break has been sent, the SENDB bit is reset by hardware. The Sync character now transmits in the preconfigured mode.

When the TXREG becomes empty, as indicated by the TXIF, the next data byte can be written to TXREG.

19.2.6 RECEIVING A BREAK CHARACTER

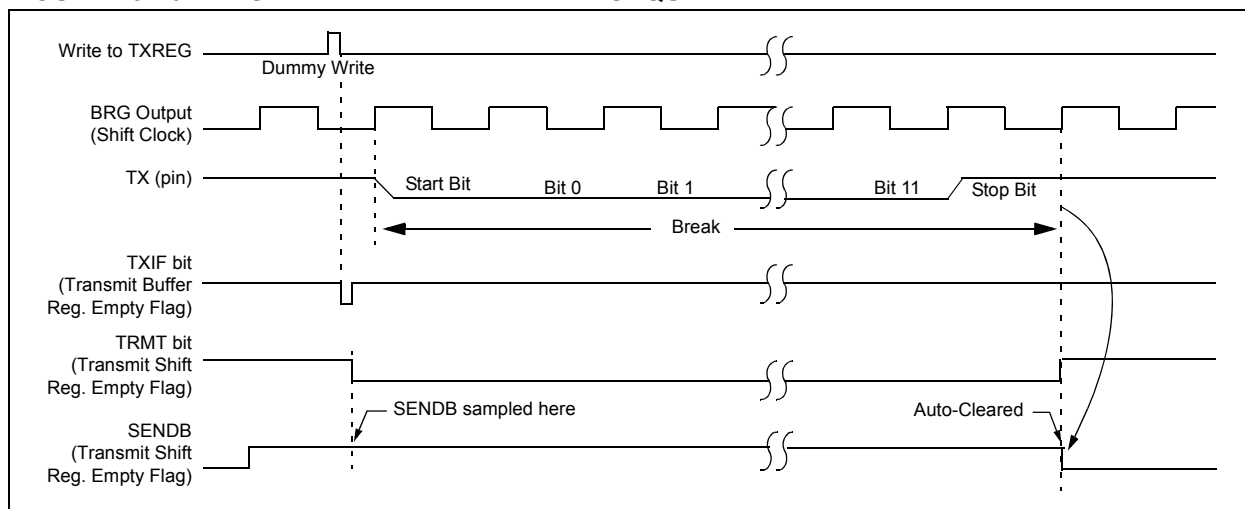
The Enhanced USART module can receive a Break character in two ways.

The first method forces configuration of the baud rate at a frequency of 9/13 the typical speed. This allows for the Stop bit transition to be at the correct sampling location (13 bits for Break versus Start bit and 8 data bits for typical data).

The second method uses the auto-wake-up feature described in **Section 19.2.4 "Auto-Wake-up on Sync Break Character"**. By enabling this feature, the EUSART will sample the next two transitions on RX/DT, cause an RCIF interrupt and receive the next data byte followed by another interrupt.

Note that following a Break character, the user will typically want to enable the Auto-Baud Rate Detect feature. For both methods, the user can set the ABD bit once the TXIF interrupt is observed.

FIGURE 19-10: SEND BREAK CHARACTER SEQUENCE



PIC18F2480/2580/4480/4580

20.1 A/D Acquisition Requirements

For the A/D Converter to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 20-2. The source impedance (Rs) and the internal sampling switch (Rss) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (Rss) impedance varies over the device voltage (VDD). The source impedance affects the offset voltage at the analog input (due to pin leakage current). **The maximum recommended impedance for analog sources is 2.5 kΩ.** After the analog input channel is selected (changed), the channel must be sampled for at least the minimum acquisition time before starting a conversion.

Note: When the conversion is started, the holding capacitor is disconnected from the input pin.

To calculate the minimum acquisition time, Equation 20-1 may be used. This equation assumes that 1/2 LSb error is used (1024 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

Example 20-3 shows the calculation of the minimum required acquisition time TACQ. This calculation is based on the following application system assumptions:

CHOLD	=	120 pF
Rs	=	2.5 kΩ
Conversion Error	≤	1/2 LSb
VDD	=	5V → Rss = 7 kΩ
Temperature	=	50°C (system max.)
VHOLD	=	0V @ time = 0

EQUATION 20-1: ACQUISITION TIME

$$\begin{aligned} \text{TACQ} &= \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= \text{TAMP} + \text{TC} + \text{Tcoff} \end{aligned}$$

EQUATION 20-2: A/D MINIMUM CHARGING TIME

$$\begin{aligned} \text{VHOLD} &= (\text{VREF} - (\text{VREF}/2048)) \cdot (1 - e^{(-\text{Tc}/\text{CHOLD}(\text{RIC} + \text{RSS} + \text{RS})))} \\ \text{or} \\ \text{TC} &= -(\text{CHOLD})(\text{RIC} + \text{RSS} + \text{RS}) \ln(1/2048) \end{aligned}$$

EQUATION 20-3: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME

$$\begin{aligned} \text{TACQ} &= \text{TAMP} + \text{TC} + \text{Tcoff} \\ \text{TAMP} &= 5 \mu\text{s} \\ \text{Tcoff} &= (\text{Temp} - 25^\circ\text{C})(0.05 \mu\text{s}/^\circ\text{C}) \\ &\quad (50^\circ\text{C} - 25^\circ\text{C})(0.05 \mu\text{s}/^\circ\text{C}) \\ &\quad 1.25 \mu\text{s} \end{aligned}$$

Temperature coefficient is only required for temperatures > 25°C. Below 25°C, Tcoff = 0 ms.

$$\begin{aligned} \text{TC} &= -(\text{CHOLD})(\text{RIC} + \text{RSS} + \text{RS}) \ln(1/2047) \mu\text{s} \\ &\quad -(120 \text{ pF})(1 \text{ k}\Omega + 7 \text{ k}\Omega + 2.5 \text{ k}\Omega) \ln(0.0004883) \mu\text{s} \\ &\quad 9.61 \mu\text{s} \\ \text{TACQ} &= 5 \mu\text{s} + 1.25 \mu\text{s} + 9.61 \mu\text{s} \\ &\quad 12.86 \mu\text{s} \end{aligned}$$

FIGURE 22-2: COMPARATOR VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE

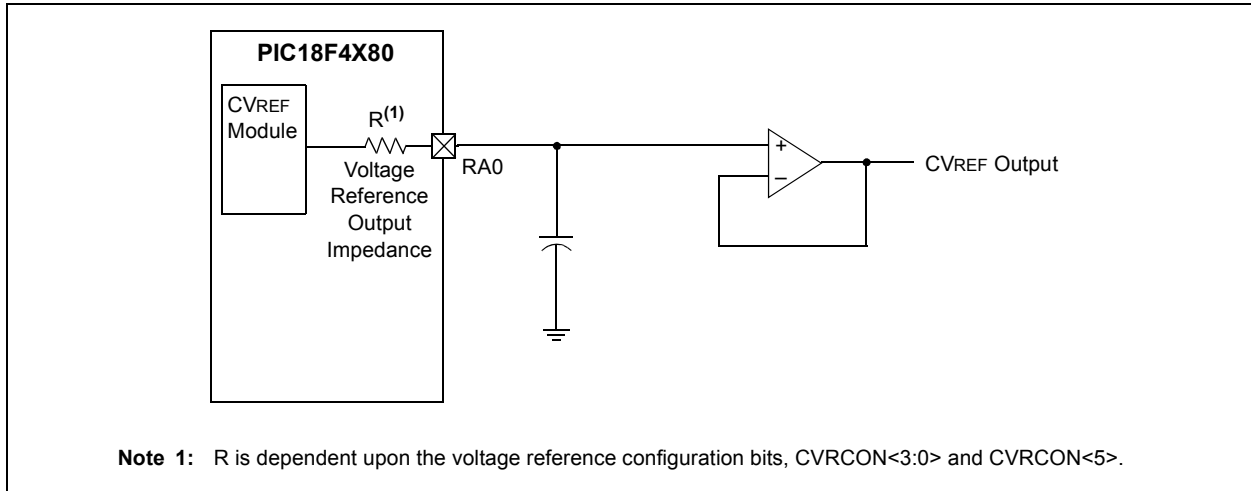


TABLE 22-1: REGISTERS ASSOCIATED WITH COMPARATOR VOLTAGE REFERENCE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
CVRCON ⁽²⁾	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	57
CMCON ⁽²⁾	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	57
TRISA	TRISA7 ⁽¹⁾	TRISA6 ⁽¹⁾	PORTA Data Direction Register						58

Legend: Shaded cells are not used with the comparator voltage reference.

Note 1: PORTA pins are enabled based on oscillator configuration.

2: These registers are unimplemented on PIC18F2X80 devices.

24.13 Bit Timing Configuration Registers

The Baud Rate Control registers (BRGCON1, BRGCON2, BRGCON3) control the bit timing for the CAN bus interface. These registers can only be modified when the PIC18F2480/2580/4480/4580 devices are in Configuration mode.

24.13.1 BRGCON1

The BRP bits control the baud rate prescaler. The SJW<1:0> bits select the synchronization jump width in terms of multiples of T_Q.

24.13.2 BRGCON2

The PRSEG bits set the length of the propagation segment in terms of T_Q. The SEG1PH bits set the length of Phase Segment 1 in T_Q. The SAM bit controls how many times the RXCAN pin is sampled. Setting this bit to a '1' causes the bus to be sampled three times: twice at T_Q/2 before the sample point and once at the normal sample point (which is at the end of Phase Segment 1). The value of the bus is determined to be the value read during at least two of the samples. If the SAM bit is set to a '0', then the RXCAN pin is sampled only once at the sample point. The SEG2PHTS bit controls how the length of Phase Segment 2 is determined. If this bit is set to a '1', then the length of Phase Segment 2 is determined by the SEG2PH bits of BRGCON3. If the SEG2PHTS bit is set to a '0', then the length of Phase Segment 2 is the greater of Phase Segment 1 and the information processing time (which is fixed at 2 T_Q for the PIC18F2480/2580/4480/4580).

24.13.3 BRGCON3

The PHSEG2<2:0> bits set the length (in T_Q) of Phase Segment 2 if the SEG2PHTS bit is set to a '1'. If the SEG2PHTS bit is set to a '0', then the PHSEG2<2:0> bits have no effect.

24.14 Error Detection

The CAN protocol provides sophisticated error detection mechanisms. The following errors can be detected.

24.14.1 CRC ERROR

With the Cyclic Redundancy Check (CRC), the transmitter calculates special check bits for the bit sequence, from the start of a frame until the end of the data field. This CRC sequence is transmitted in the CRC field. The receiving node also calculates the CRC sequence using the same formula and performs a comparison to the received sequence. If a mismatch is detected, a CRC error has occurred and an error frame is generated. The message is repeated.

24.14.2 ACKNOWLEDGE ERROR

In the Acknowledge field of a message, the transmitter checks if the Acknowledge slot (which was sent out as a recessive bit) contains a dominant bit. If not, no other node has received the frame correctly. An Acknowledge error has occurred, an error frame is generated and the message will have to be repeated.

24.14.3 FORM ERROR

If a node detects a dominant bit in one of the four segments, including End-Of-Frame (EOF), interframe space, Acknowledge delimiter or CRC delimiter, then a form error has occurred and an error frame is generated. The message is repeated.

24.14.4 BIT ERROR

A bit error occurs if a transmitter sends a dominant bit and detects a recessive bit, or if it sends a recessive bit and detects a dominant bit, when monitoring the actual bus level and comparing it to the just transmitted bit. In the case where the transmitter sends a recessive bit and a dominant bit is detected during the arbitration field and the Acknowledge slot, no bit error is generated because normal arbitration is occurring.

24.14.5 STUFF BIT ERROR

If, between the Start-Of-Frame (SOF) and the CRC delimiter, six consecutive bits with the same polarity are detected, the bit stuffing rule has been violated. A stuff bit error occurs and an error frame is generated. The message is repeated.

24.14.6 ERROR STATES

Detected errors are made public to all other nodes via error frames. The transmission of the erroneous message is aborted and the frame is repeated as soon as possible. Furthermore, each CAN node is in one of the three error states; "error-active", "error-passive" or "bus-off", according to the value of the internal error counters. The error-active state is the usual state where the bus node can transmit messages and activate error frames (made of dominant bits) without any restrictions. In the error-passive state, messages and passive error frames (made of recessive bits) may be transmitted. The bus-off state makes it temporarily impossible for the node to participate in the bus communication. During this state, messages can neither be received nor transmitted.

24.14.7 ERROR MODES AND ERROR COUNTERS

The PIC18F2480/2580/4480/4580 devices contain two error counters: the Receive Error Counter (RXERRCNT) and the Transmit Error Counter (TXERRCNT). The values of both counters can be read by the MCU. These counters are incremented or decremented in accordance with the CAN bus specification.

PIC18F2480/2580/4480/4580

REGISTER 25-6: CONFIG5L: CONFIGURATION REGISTER 5 LOW (BYTE ADDRESS 300008h)

U-0	U-0	U-0	U-0	R/C-1	R/C-1	R/C-1	R/C-1
—	—	—	—	CP3 ⁽¹⁾	CP2 ⁽¹⁾	CP1	CP0
bit 7							bit 0

Legend:

R = Readable bit

C = Clearable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

u = Unchanged from programmed state

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

bit 3 **CP3:** Code Protection bit⁽¹⁾

1 = Block 3 (006000-007FFFh) not code-protected

0 = Block 3 (006000-007FFFh) code-protected

bit 2 **CP2:** Code Protection bit⁽¹⁾

1 = Block 2 (004000-005FFFh) not code-protected

0 = Block 2 (004000-005FFFh) code-protected

bit 1 **CP1:** Code Protection bit

1 = Block 1 (002000-003FFFh) not code-protected

0 = Block 1 (002000-003FFFh) code-protected

bit 0 **CP0:** Code Protection bit

1 = Block 0 (000800-001FFFh) not code-protected

0 = Block 0 (000800-001FFFh) code-protected

Note 1: Unimplemented in PIC18FX480 devices; maintain this bit set.

REGISTER 25-7: CONFIG5H: CONFIGURATION REGISTER 5 HIGH (BYTE ADDRESS 300009h)

R/C-1	R/C-1	U-0	U-0	U-0	U-0	U-0	U-0
CPD	CPB	—	—	—	—	—	—
bit 7							bit 0

Legend:

R = Readable bit

C = Clearable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

u = Unchanged from programmed state

bit 7 **CPD:** Data EEPROM Code Protection bit

1 = Data EEPROM not code-protected

0 = Data EEPROM code-protected

bit 6 **CPB:** Boot Block Code Protection bit

1 = Boot Block (000000-0007FFFh) not code-protected

0 = Boot Block (000000-0007FFFh) code-protected

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

PIC18F2480/2580/4480/4580

REGISTER 25-12: DEVID1: DEVICE ID REGISTER 1 FOR PIC18F2480/2580/4480/4580

R	R	R	R	R	R	R	R
DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0
bit 7							bit 0

Legend:

R = Read-only bit

P = Programmable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

u = Unchanged from programmed state

bit 7-5 **DEV<2:0>**: Device ID bits

111 = PIC18F2480

110 = PIC18F2580

101 = PIC18F4480

100 = PIC18F4580

bit 4-0 **REV<3:0>**: Revision ID bits

These bits are used to indicate the device revision.

REGISTER 25-13: DEVID2: DEVICE ID REGISTER 2 FOR PIC18F2480/2580/4480/4580

R	R	R	R	R	R	R	R
DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3
bit 7							bit 0

Legend:

R = Read-only bit

P = Programmable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

u = Unchanged from programmed state

bit 7-0 **DEV<10:3>**: Device ID bits

These bits are used with the DEV<2:0> bits in Device ID Register 1 to identify the part number.

0001 1010 = PIC18F2480/2580/4480/4580 devices

Note 1: These values for DEV<10:3> may be shared with other devices. The specific device is always identified by using the entire DEV<10:0> bit sequence.

PIC18F2480/2580/4480/4580

BTG Bit Toggle f

Syntax: BTG f, b {,a}

Operands: $0 \leq f \leq 255$
 $0 \leq b < 7$
 $a \in [0,1]$

Operation: $\overline{(f \ll b)} \rightarrow f \ll b$

Status Affected: None

Encoding:

0111	bbba	ffff	ffff
------	------	------	------

Description: Bit 'b' in data memory location 'f' is inverted.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See **Section 26.2.3 "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: BTG PORTC, 4, 0

Before Instruction:

PORTC = 0111 0101 [75h]

After Instruction:

PORTC = 0110 0101 [65h]

BOV Branch if Overflow

Syntax: BOV n

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

Operation: if Overflow bit is '1',
 $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1110	0100	nnnn	nnnn
------	------	------	------

Description: If the Overflow bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be $PC + 2 + 2n$. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

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

If No Jump:

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

Example: HERE BOV Jump

Before Instruction

PC = address (HERE)

After Instruction

If Overflow = 1;

PC = address (Jump)

If Overflow = 0;

PC = address (HERE + 2)

PIC18F2480/2580/4480/4580

MOVFF Move f to f

Syntax: MOVFF f_s, f_d

Operands: $0 \leq f_s \leq 4095$
 $0 \leq f_d \leq 4095$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

1100	ffff	ffff	ffff _s
1111	ffff	ffff	ffff _d

1st word (source)

2nd word (destin.)

Description: The contents of source register ' f_s ' are moved to destination register ' f_d '. Location of source ' f_s ' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination ' f_d ' can also be anywhere from 000h to FFFh. Either source or destination can be W (a useful special situation). MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port). The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

Example: MOVFF REG1, REG2

Before Instruction

REG1 = 33h
 REG2 = 11h

After Instruction

REG1 = 33h
 REG2 = 33h

MOVLB Move Literal to Low Nibble in BSR

Syntax: MOVLW k

Operands: $0 \leq k \leq 255$

Operation: $k \rightarrow \text{BSR}$

Status Affected: None

Encoding:

0000	0001	kkkk	kkkk
------	------	------	------

Description: The eight-bit literal 'k' is loaded into the Bank Select Register (BSR). The value of BSR<7:4> always remains '0', regardless of the value of $k_7:k_4$.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: MOVLB 5

Before Instruction

BSR Register = 02h

After Instruction

BSR Register = 05h

PIC18F2480/2580/4480/4580

FIGURE 28-1: PIC18F2480/2580/4480/4580 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL, EXTENDED)

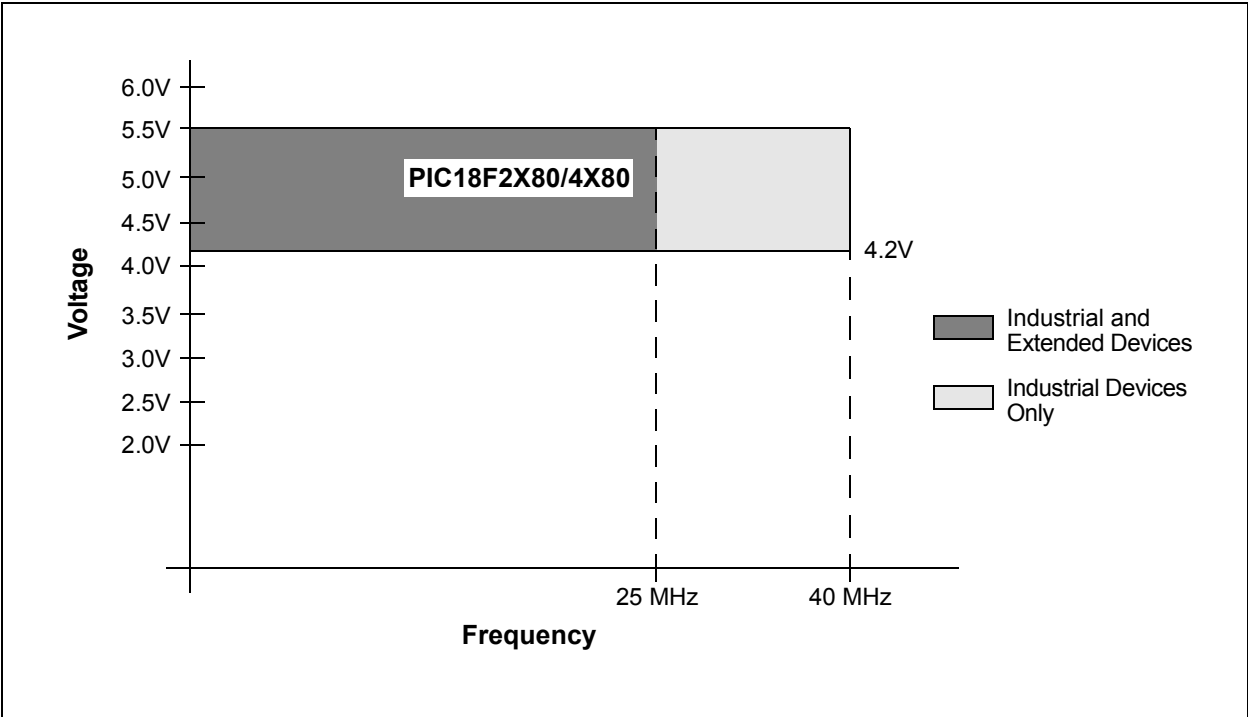
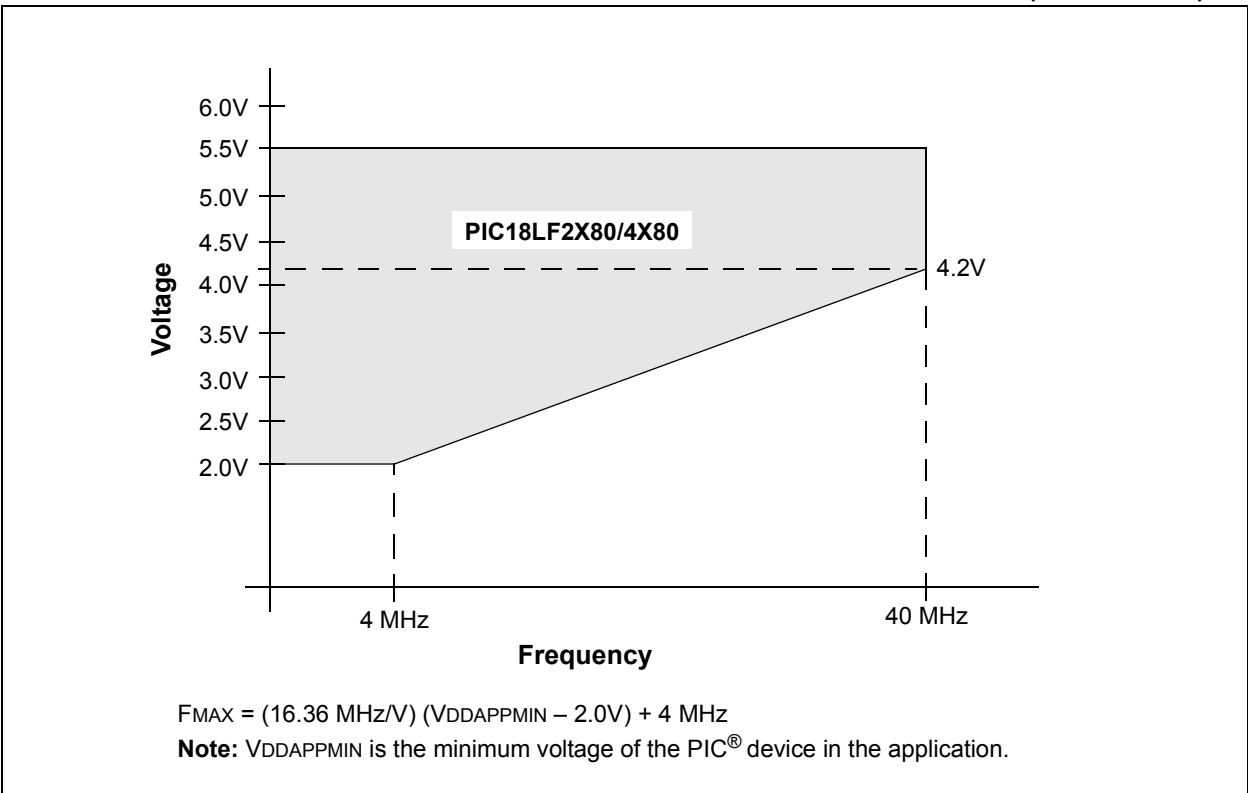


FIGURE 28-2: PIC18LF2480/2580/4480/4580 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)



PIC18F2480/2580/4480/4580

FIGURE 28-3: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS

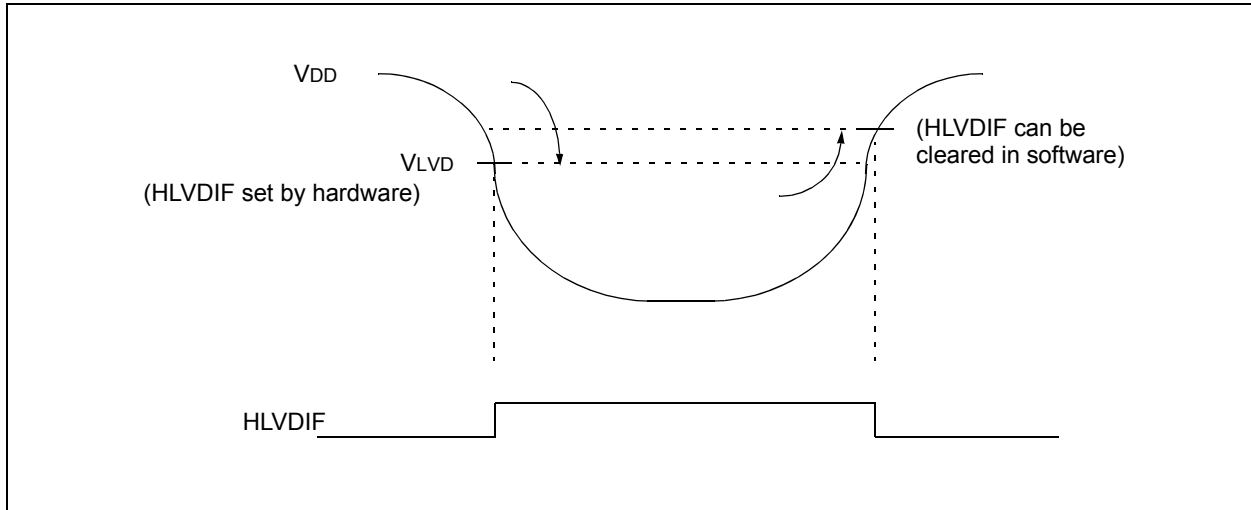


TABLE 28-4: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS

				Standard Operating Conditions (unless otherwise stated)				
				Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} < T_A < +125^{\circ}\text{C}$ for extended				
Param No.	Symbol	Characteristic		Min	Typ	Max	Units	Conditions
D420		HLVD Voltage on V_{DD} Transition High-to-Low	LVV = 0000	2.12	2.17	2.22	V	
			LVV = 0001	2.18	2.23	2.28	V	
			LVV = 0010	2.31	2.36	2.42	V	
			LVV = 0011	2.38	2.44	2.49	V	
			LVV = 0100	2.54	2.60	2.66	V	
			LVV = 0101	2.72	2.79	2.85	V	
			LVV = 0110	2.82	2.89	2.95	V	
			LVV = 0111	3.05	3.12	3.19	V	
			LVV = 1000	3.31	3.39	3.47	V	
			LVV = 1001	3.46	3.55	3.63	V	
			LVV = 1010	3.63	3.71	3.80	V	
			LVV = 1011	3.81	3.90	3.99	V	
			LVV = 1100	4.01	4.11	4.20	V	
			LVV = 1101	4.23	4.33	4.43	V	
			LVV = 1110	4.48	4.59	4.69	V	
			LVV = 1111	1.14	1.2	1.26	V	