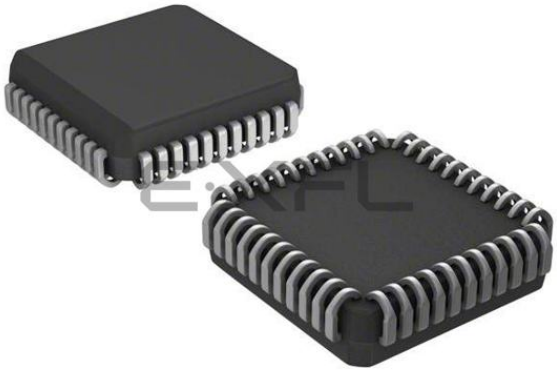


Welcome to E-XFL.COM

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	4MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	33
Program Memory Size	7KB (4K x 14)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	192 x 8
Voltage - Supply (Vcc/Vdd)	4V ~ 5.5V
Data Converters	-
Oscillator Type	External
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	44-LCC (J-Lead)
Supplier Device Package	44-PLCC (16.59x16.59)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16c65b-04-l

CHAPTER 1
8-Pin Flash PIC® Microcontrollers Tips ‘n Tricks

Table Of Contents

TIPS ‘N TRICKS INTRODUCTION

Microchip continues to provide innovative products that are smaller, faster, easier to use and more reliable. The 8-pin Flash PIC® microcontrollers (MCU) are used in an wide range of everyday products, from toothbrushes, hair dryers and rice cookers to industrial, automotive and medical products.

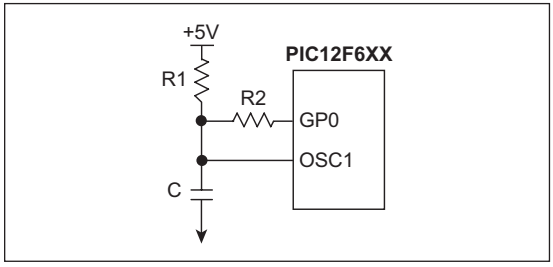
The PIC12F629/675 MCUs merge all the advantages of the PIC MCU architecture and the flexibility of Flash program memory into an 8-pin package. They provide the features and intelligence not previously available due to cost and board space limitations. Features include a 14-bit instruction set, small footprint package, a wide operating voltage of 2.0 to 5.5 volts, an internal programmable 4 MHz oscillator, on-board EEPROM data memory, on-chip voltage reference and up to 4 channels of 10-bit A/D. The flexibility of Flash and an excellent development tool suite, including a low-cost In-Circuit Debugger, In-Circuit Serial Programming™ and MPLAB® ICE 2000 emulation, make these devices ideal for just about any embedded control application.

TIPS ‘N TRICKS WITH HARDWARE

The following series of Tips ’n Tricks can be applied to a variety of applications to help make the most of the 8-pin dynamics.

TIP #1 Dual Speed RC Oscillator

Figure 1-1



1. After reset I/O pin is High-Z
2. Output ‘1’ on I/O pin
3. R1, R2 and C determine OSC frequency
4. Also works with additional capacitors

Frequency of PIC MCU in external RC oscillator mode depends on resistance and capacitance on OSC1 pin. Resistance is changed by the output voltage on GP0. GP0 output ‘1’ puts R2 in parallel with R1 reduces OSC1 resistance and increases OSC1 frequency. GP0 as an input increases the OSC1 resistance by minimizing current flow through R2, and decreases frequency and power consumption.

Summary:

GP0 = Input: Slow speed for low current

GP0 = Output high: High speed for fast processing

TIP #2 Input/Output Multiplexing

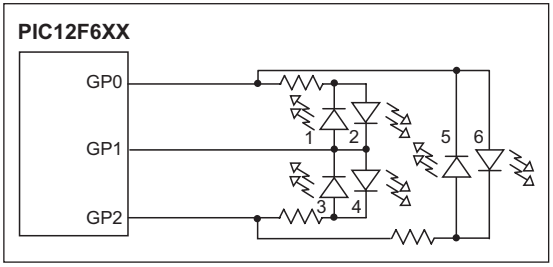
Individual diodes and some combination of diodes can be enabled by driving I/Os high and low or switching to inputs (Z). The number of diodes (D) that can be controlled depends on the number of I/Os (GP) used.

The equation is: D = GP x (GP - 1).

Example 2-1: Six LEDs on Three I/O Pins

GPx			LEDs					
0	1	2	1	2	3	4	5	6
0	0	0	0	0	0	0	0	0
0	1	Z	1	0	0	0	0	0
1	0	Z	0	1	0	0	0	0
Z	0	1	0	0	1	0	0	0
Z	1	0	0	0	0	1	0	0
0	Z	1	0	0	0	0	1	0
1	Z	0	0	0	0	0	0	1
0	0	1	0	0	1	0	1	0
0	1	0	1	0	0	1	0	0
0	1	1	1	0	0	0	1	0
1	0	0	0	1	0	0	0	1
1	0	1	0	1	1	0	0	0
1	1	0	0	0	0	1	0	1
1	1	1	0	0	0	0	0	0

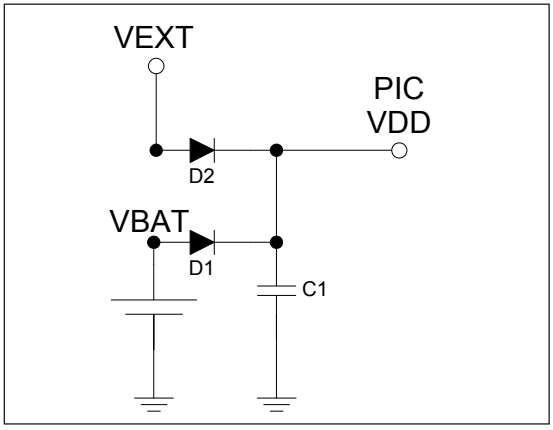
Figure 2-1



TIP #7 Battery Backup for PIC MCUs

For an application that can operate from either an external supply or a battery backup, it is necessary to be able to switch from one to the other without user intervention. This can be accomplished with battery backup ICs, but it is also possible to implement with a simple diode OR circuit, shown in Figure 7-1. Diode D1 prevents current from flowing into the battery from VEXT when the external power is supplied. D2 prevents current from flowing into any external components from the battery if VEXT is removed. As long as the external source is present and higher voltage than the battery, no current from the battery will be used. When VEXT is removed and the voltage drops below VBAT, the battery will start powering the MCU. Low forward voltage Schottky diodes can be used in order to minimize the voltage dropout from the diodes. Additionally, inputs can be referenced to VEXT and VBAT in order to monitor the voltage levels of the battery and the external supply. This allows the micro to enter lower power modes when the supply is removed or the battery is running low. In order to avoid glitches on V_{DD} caused by the diode turn-on delay when switching supplies, ensure enough decoupling capacitance is used on V_{DD} (C1).

Figure 7-1:



Dynamic Operation Tips n’ Tricks

The following tips and tricks apply to methods of improving the dynamic operating current consumption of an application. This allows an application to get processing done quicker which enables it to sleep more and will help reduce the current consumed while processing.

TIP #8 Enhanced PIC16 Mid-Range Core

The Enhanced PIC16 mid-range core has a few features to assist in low power. New instructions allow many applications to execute in less time. This allows the application to spend more time asleep and less time processing and can provide considerable power savings. It is important not to overlook these new instructions when designing with devices that contain the new core. The Timer1 oscillator and WDT have also been improved, now meeting nanoWatt XLP requirements and drawing much less current than in previous devices.

TIP #14 Use NOP and Idle Mode

When waiting on a blocking loop (e.g. waiting for an interrupt), instead put the device into Idle mode to disable the CPU. The peripheral interrupt will wake up the device. Idle mode consumes much less current than constantly reading RAM and jumping back. If the CPU cannot be disabled because the loop required some calculations, such as incrementing a counter, instead of doing a very tight loop that loops many times, add NOPs into the loop. See the code example below. A NOP requires less current to execute than reading RAM or branching operations, so current can be reduced. The overall loop count can be adjusted to account for the extra instructions for the NOPs.

Example:

Replace:

```
while(!_T1IF);
```

with Idle mode:

```
IEC0bits.T1IE = 1;
Idle();
```

and replace:

```
while(!_T1IF){
    i++;
}
```

with extra NOP instructions:

```
while(!_T1IF){
   i++;
    Nop();
    Nop();
    Nop();
    Nop();
    Nop();
}
```

TIP #15 Peripheral Module Disable (PMD) Bits

PIC24, dsPIC DSCs, and PIC32 devices have PMD bits that can be used to disable peripherals that will not be used in the application. Setting these bits disconnects all power to the module as well as SFRs for the module. Because power is completely removed, the PMD bits offer additional power savings over disabling the module by turning off the module’s enable bit. These bits can be dynamically changed so that modules which are only used periodically can be disabled for the remainder of the application. The PMD bits are most effective at high clock speeds and when operating at full speed allowing the average power consumption to be significantly reduced.

TIP #9 Generating the Time Tick for a RTOS

Real Time Operating Systems (RTOS) require a periodic interrupt to operate. This periodic interrupt, or "tick rate", is the basis for the scheduling system that RTOS's employ. For instance, if a 2 ms tick is used, the RTOS will schedule its tasks to be executed at multiples of the 2 ms. A RTOS also assigns a priority to each task, ensuring that the most critical tasks are executed first. Table 9-1 shows an example list of tasks, the priority of each task and the time interval that the tasks need to be executed.

Table 9-1: Tasks

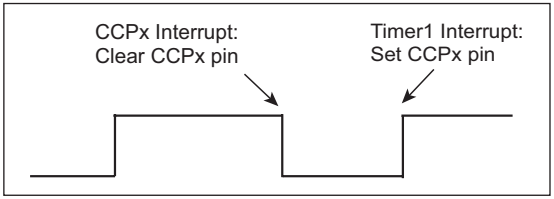
Task	Interval	Priority
Read ADC Input 1	20 ms	2
Read ADC Input 2	60 ms	1
Update LCD	24 ms	2
Update LED Array	36 ms	3
Read Switch	10 ms	1
Dump Data to Serial Port	240 ms	1

The techniques described in Tip #7 can be used to generate the 2 ms periodic interrupt using the CCP module configured in Compare mode.

Note: For more information on RTOSs and their use, see Application Note AN777 "*Multitasking on the PIC16F877 with the Salvo™ RTOS*".

TIP #10 16-Bit Resolution PWM

Figure 10-1: 16-Bit Resolution PWM



1. Configure CCPx to clear output (CCPx pin) on match in Compare mode (CCPxCON <CCPSM3:CCPxM0>).
2. Enable the Timer1 interrupt.
3. Set the period of the waveform via Timer1 prescaler (T1CON <5:4>).
4. Set the duty cycle of the waveform using CCPRxL and CCPRxH.
5. Set CCPx pin when servicing the Timer1 overflow interrupt(1).

Note 1: One hundred percent duty cycle is not achievable with this implementation due to the interrupt latency in servicing Timer1. The period is not affected because the interrupt latency will be the same from period to period as long as the Timer1 interrupt is serviced first in the ISR.

Timer1 has four configurable prescaler values. These are 1:1, 1:2, 1:4 and 1:8. The frequency possibilities of the PWM described above are determined by Equation 10-1.

Equation 10-1

$$F_{PWM} = F_{OSC}/(65536*4*prescaler)$$

For a microcontroller running on a 20 MHz oscillator (Fosc) this equates to frequencies of 76.3 Hz, 38.1 Hz, 19.1 Hz and 9.5 Hz for increasing prescaler values.

PWM TIPS ‘N TRICKS

The ECCP and CCP modules produce a 10-bit resolution Pulse-Width Modulated (PWM) waveform on the CCPx pin. The ECCP module is capable of transmitting a PWM signal on one of four pins, designated P1A through P1D. The PWM modes available on the ECCP module are:

- Single output (P1A only)
- Half-bridge output (P1A and P1B only)
- Full-bridge output forward
- Full-bridge output reverse

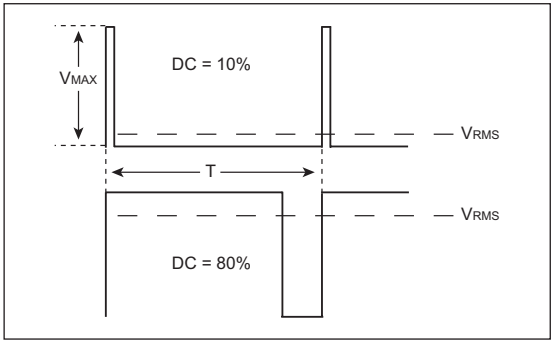
One of the following configurations must be chosen when using the ECCP module in PWM Full-bridge mode:

- P1A, P1C active-high; P1B, P1D active-high
- P1A, P1C active-high; P1B, P1D active-low
- P1A, P1C active-low; P1B, P1D active-high
- P1A, P1C active-low; P1B, P1D active-low

“Why Would I Use PWM Mode?”

As the next set of Tips ‘n Tricks demonstrate, Pulse-Width Modulation (PWM) can be used to accomplish a variety of tasks from dimming LEDs to controlling the speed of a brushed DC electric motor. All these applications are based on one basic principle of PWM signals – as the duty cycle of a PWM signal increases, the average voltage and power provided by the PWM increases. Not only does it increase with duty cycle, but it increases linearly. The following figure illustrates this point more clearly. Notice that the RMS and maximum voltage are functions of the duty cycle (DC) in the following Figure 12-3.

Figure 12-3: Duty Cycle Relation to V_{RMS}



Equation 12-1 shows the relation between V_{RMS} and V_{MAX}.

Equation 12-1: Relation Between V_{RMS} and V_{MAX}

$$V_{RMS} = DC \times V_{MAX}$$

TIP #13 Deciding on PWM Frequency

In general, PWM frequency is application dependent although two general rules-of-thumb hold regarding frequency in all applications. They are:

1. As frequency increases, so does current requirement due to switching losses.
2. Capacitance and inductance of the load tend to limit the frequency response of a circuit.

In low-power applications, it is a good idea to use the minimum frequency possible to accomplish a task in order to limit switching losses. In circuits where capacitance and/or inductance are a factor, the PWM frequency should be chosen based on an analysis of the circuit.

Motor Control

PWM is used extensively in motor control due to the efficiency of switched drive systems as opposed to linear drives. An important consideration when choosing PWM frequency for a motor control application is the responsiveness of the motor to changes in PWM duty cycle. A motor will have a faster response to changes in duty cycle at higher frequencies. Another important consideration is the sound generated by the motor. Brushed DC motors will make an annoying whine when driven at frequencies within the audible frequency range (20 Hz-4 kHz.) In order to eliminate this whine, drive brushed DC motors at frequencies greater than 4 kHz. (Humans can hear frequencies at upwards of 20 kHz, however, the mechanics of the motor winding will typically attenuate motor whine above 4 kHz).

LED and Light Bulbs

PWM is also used in LED and light dimmer applications. Flicker may be noticeable with rates below 50 Hz. Therefore, it is generally a good rule to pulse-width modulate LEDs and light bulbs at 100 Hz or higher.

TIP #14 Unidirectional Brushed DC Motor Control Using CCP

Figure 14-1: Brushed DC (BDC) Motor Control Circuit

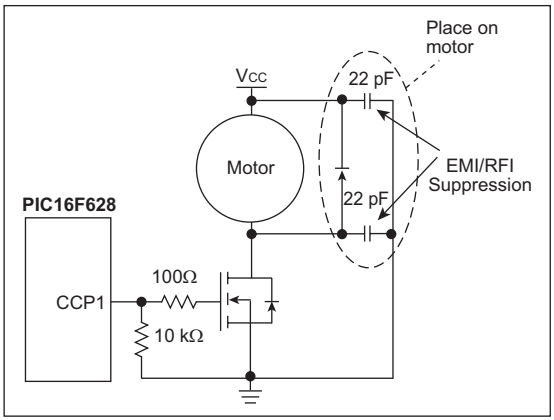


Figure 14-1 shows a unidirectional speed controller circuit for a brushed DC motor. Motor speed is proportional to the duty cycle of the PWM output on the CCP1 pin. The following steps show how to configure the PIC16F628 to generate a 20 kHz PWM with 50% duty cycle. The microcontroller is running on a 20 MHz crystal.

Step #1: Choose Timer2 Prescaler

a) F_{PWM} = Fosc/((PR2+1)*4*prescaler) = 19531 Hz for PR2 = 255 and prescaler of 1

b) This frequency is lower than 20 kHz, therefore a prescaler of 1 is adequate.

Step #2: Calculate PR2

PR2 = Fosc/(F_{PWM}*4*prescaler) – 1 = 249

Step #3: Determine CCPR1L and CCP1CON<5:4>

a) CCPR1L:CCP1CON<5:4> = DutyCycle*0x3FF = 0x1FF

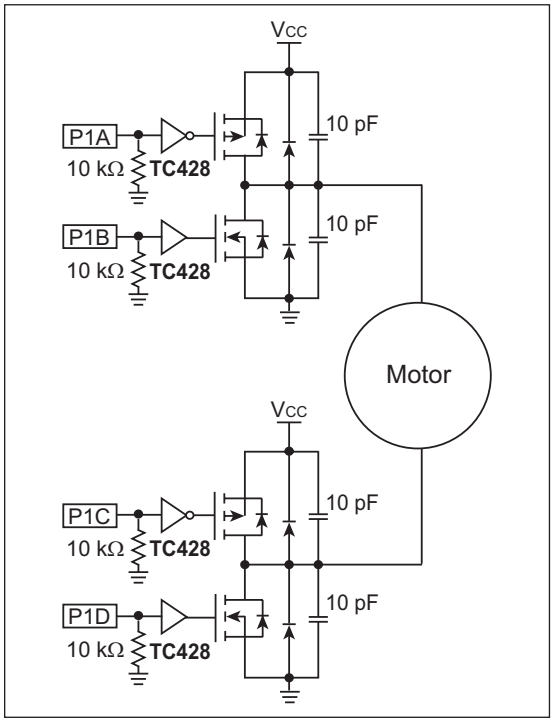
b) CCPR1L = 0x1FF >> 2 = 0x7F, CCP1CON<5:4> = 3

Step #4: Configure CCP1CON

The CCP module is configured in PWM mode with the Least Significant bits of the duty cycle set, therefore, CCP1CON = '`b001111000`'.

TIP #15 Bidirectional Brushed DC Motor Control Using ECCP

Figure 15-1: Full-Bridge BDC Drive Circuit



The ECCP module has brushed DC motor control options built into it. Figure 15-1 shows how a full-bridge drive circuit is connected to a BDC motor. The connections P1A, P1B, P1C and P1D are all ECCP outputs when the module in configured in "Full-bridge Output Forward" or "Full-bridge Output Reverse" modes (CCP1CON<7:6>). For the circuit shown in Figure 15-1, the ECCP module should be configured in PWM mode: P1A, P1C active high; P1B, P1D active high (CCP1CON<3:1>). The reason for this is the MOSFET drivers (TC428) are configured so a high input will turn on the respective MOSFET.

The following table shows the relation between the states of operation, the states of the ECCP pins and the ECCP Configuration register.

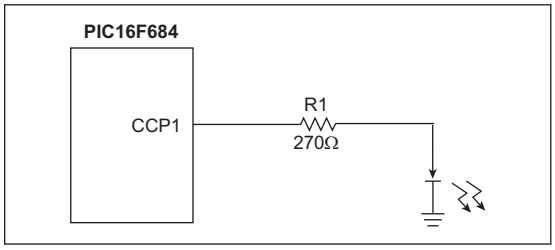
State	P1A	P1B	P1C	P1D	CCP1CON
Forward	1	tri-state	tri-state	mod	'b01xx1100'
Reverse	tri-state	mod	1	tri-state	'b11xx1100'
Coast	tri-state	tri-state	tri-state	tri-state	N/A
Brake	tri-state	1	1	tri-state	N/A

Legend: '1' = high, '0' = low, mod = modulated, tri-state = pin configured as input

TIP #18 Varying LED Intensity

The intensity of an LED can be varied by pulse-width modulating the voltage across the LED. A microcontroller typically drives an LED with the circuit shown in Figure 18-1. The purpose of R1 is to limit the LED current so that the LED runs in its specified current and voltage range, typically around 1.4 volts at 20 mA. Modulating the LED drive pin on the microcontroller will vary the average current seen by the LED and thus its intensity. As mentioned in Tip #13, LEDs and other light sources should be modulated at no less than 100 Hz in order to prevent noticeable flicker.

Figure 18-1: LED Drive

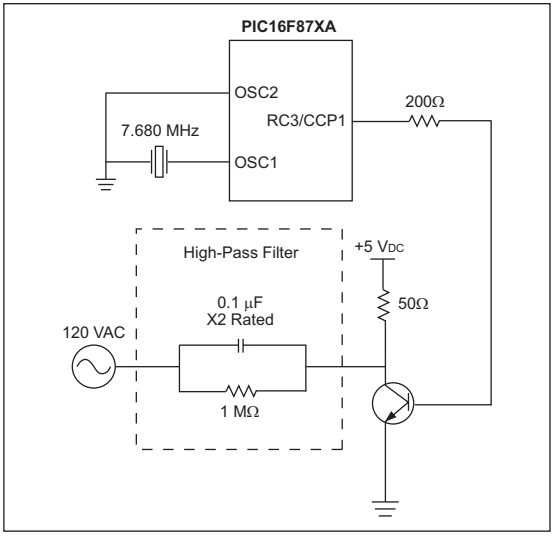


The CCP module, configured in PWM mode, is ideal for varying the intensity of an LED. Adjustments to the intensity of the LED are made by simply varying the duty cycle of the PWM signal driving the LED. This is accomplished by varying the CCPRxL register between 0 and 0xFF.

TIP #19 Generating X-10® Carrier Frequency

X-10 uses a piggybacked 120 kHz square wave (at 50% duty cycle) to transmit information over 60 Hz power lines. The CCP module, running in PWM mode, can accurately create the 120 kHz square wave, referred to as the carrier frequency. Figure 19-1 shows how the 120 kHz carrier frequency is piggybacked onto the sinusoidal 60 Hz power waveform.

Figure 19-1: Carrier Frequency With Sinusoidal Waveform



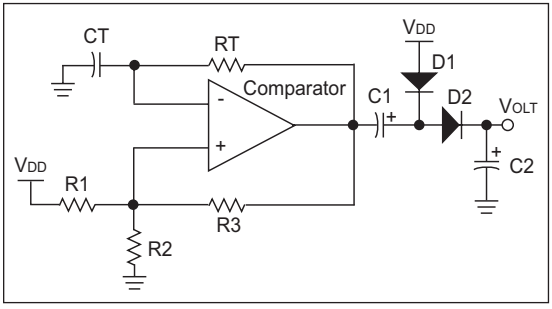
X-10 specifies the carrier frequency at 120 kHz (± 2 kHz). The system oscillator in Figure 18-1 is chosen to be 7.680 MHz, so that the CCP module can generate precisely 120 kHz. X-10 requires that the carrier frequency be turned on and off at different points on the 60 Hz power waveform. This is accomplished by configuring the TRIS register for the CCP1 pin as either an input (carrier frequency off) or an output (carrier frequency on). Refer to Application Note AN236 "*X-10 Home Automation Using the PIC16F877A*" for more details on X-10 and for source code for setting up the CCP module appropriately.

NOTES:

TIP #10 Capacitive Voltage Doubler

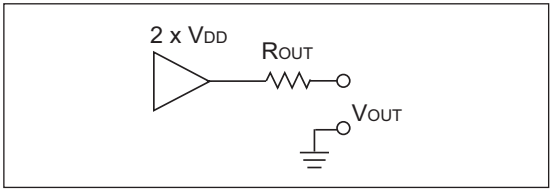
This tip takes the multi-vibrator described in Tip #8 and builds a capacitive voltage doubler around it (see Figure 10-1). The circuit works by alternately charging capacitor C1 through diode D1, and then charge balancing the energy in C1 with C2 through diode D2. At the start of the cycle, the output of the multi-vibrator is low and charge current flows from V_{DD} through D1 and into C1. When the output of the multi-vibrator goes high, D1 is reverse biased and the charge current stops. The voltage across C1 is added to the output voltage of the multi-vibrator, creating a voltage at the positive terminal of C1 which is 2 x V_{DD}. This voltage forward biases D2 and the charge in C1 is shared with C2. When the output of the multi-vibrator goes low again, the cycle starts over.

Figure 10-1: Capacitive Voltage Doubler



Note: The output voltage of a capacitive double is unregulated and will sag with increasing load current. Typically, the output is modeled as a voltage source with a series resistance (see Figure 10-2).

Figure 10-2: Equivalent Output Model



To design a voltage doubler, first determine the maximum tolerable output resistance based on the required output current and the minimum tolerable output voltage. Remember that the output current will be limited to one half of the output capability of the comparator. Then choose a transfer capacitance and switching frequency using Equation 10-1.

Equation 10-1

$$R_{OUT} = \frac{1}{F_{SWITCH} * C1}$$

Note: R_{OUT} will be slightly higher due to the dynamic resistance of the diodes. The equivalent series resistance or ESR, of the capacitors and the output resistance of the comparator. See the TC7660 data sheet for a more complete description.

Once the switching frequency is determined, design a square-wave multi-vibrator as described in Tip #8.

Finally, select diodes D1 and D2 for their current rating and set C2 equal to C1.

Example:

From Tip #8, the values are modified for a F_{OSC} of 4.8 kHz.

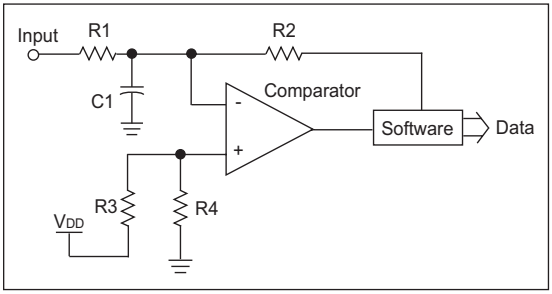
- C1 and C2 = 10 μF
- R_{OUT} = 21

TIP #14 Delta-Sigma ADC

This tip describes the creation of a hardware/software-based Delta-Sigma ADC. A Delta-Sigma ADC is based on a Delta-Sigma modulator composed of an integrator, a comparator, a clock sampler and a 1-bit DAC output. In this example, the integrator is formed by R1 and C1. The comparator is an on-chip voltage comparator. The clock sampler is implemented in software and the 1-bit DAC output is a single I/O pin. The DAC output feeds back into the integrator through R2.

Resistors R3 and R4 form a $V_{DD}/2$ reference for the circuit (see Figure 14-1).

Figure 14-1: Delta-Sigma Modulator



In operation, the feedback output from the software is a time sampled copy of the comparator output. In normal operation, the modulator output generates a PWM signal which is inversely proportional to the input voltage. As the input voltage increases, the PWM signal will drop in duty cycle to compensate. As the input decreases, the duty cycle rises.

To perform an A-to-D conversion, the duty cycle must be integrated over time, digitally, to integrate the duty cycle to a binary value. The software starts two counters. The first counts the total number of samples in the conversion and the second counts the number of samples that were low. The ratio of the two counts is equal to the ratio of the input voltage over V_{DD}.

> **Note:** This assumes that R1 and R2 are equal and R3 is equal to R4. If R1 and R2 are not equal, then the input voltage is also scaled by the ratio of R2 over R1, and R3 must still be equal to R4.

For a more complete description of the operation of a Delta-Sigma ADC and example firmware, see Application Note AN700 “*Make A Delta-Sigma Converter Using a Microcontroller’s Analog Comparator Module*.”

Example:

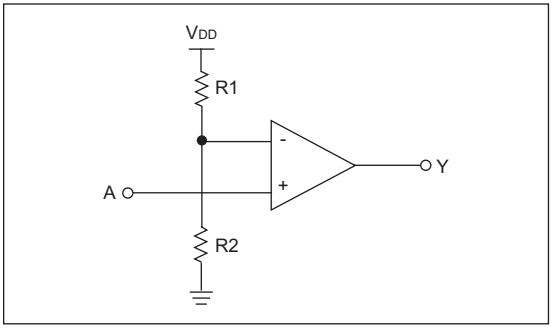
- R3 = R4 = 10 kHz
- R1 = R2 = 5.1k
- C1 = 1000 pF

TIP #15 Level Shifter

This tip shows the use of the comparator as a digital logic level shifter. The inverting input is biased to the center of the input voltage range ($V_{IN}/2$). The non-inverting input is then used for the circuit input. When the input is below the $V_{IN}/2$ threshold, the output is low. When the input is above $V_{IN}/2$, then the output is high. Values for R1 and R2 are not critical, though their ratio should result in a threshold voltage $V_{IN}/2$ at the mid-point of the input signal voltage range. Some microcontrollers have the option to connect the inverting input to an internal voltage reference. To use the reference in place of R1 and R2, simply select the internal reference and configure it for one half the input voltage range.

Note: Typical propagation delay for the circuit is 250-350 ns using the typical on-chip comparator peripheral of a microcontroller.

Figure 15-1: Level Shifter



Example:

- V_{IN} = 0 - 2V, $V_{IN}/2$ = 1V, V_{DD} = 5V
- R2 = 10k, R3 = 3.9k

TIP #16 Logic: Inverter

When designing embedded control applications, there is often the need for an external gate. Using the comparator, several simple gates can be implemented. This tip shows the use of the comparator as an inverter.

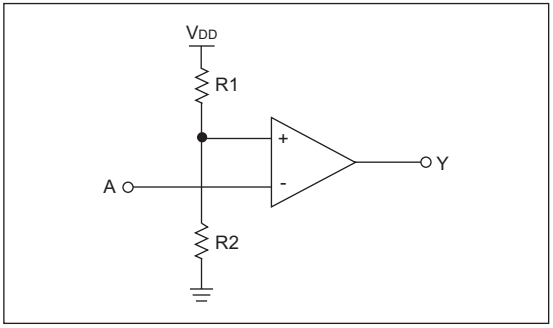
The non-inverting input is biased to the center of the input voltage range, typically $V_{DD}/2$. The inverting input is then used for the circuit input. When the input is below $V_{DD}/2$, the output is high. When the input is above $V_{DD}/2$, then the output is low.

Values for R1 and R2 are not critical, though they must be equal to set the threshold at $V_{DD}/2$.

Some microcontrollers have the option to connect the inverting input to an internal voltage reference. To use the reference in place of R1 and R2, move the input to the non-inverting input and set the output polarity bit in the comparator control register to invert the comparator output.

Note: Typical propagation delay for the circuit is 250-350 ns using the typical on-chip comparator peripheral of a microcontroller.

Figure 16-1: Inverter



NOTES:

TIP #1 Typical Ordering Considerations and Procedures for Custom Liquid Displays

1. Consider what useful information needs to be displayed on the custom LCD and the combination of alphanumeric and custom icons that will be necessary.
2. Understand the environment in which the LCD will be required to operate. Operating voltage and temperature can heavily influence the contrast of the LCD and potentially limit the type of LCD that can be used.
3. Determine the number of segments necessary to achieve the desired display on the LCD and reference the PIC Microcontroller LCD matrix for the appropriate LCD PIC microcontroller.
4. Create a sketch/mechanical print and written description of the custom LCD and understand the pinout of the LCD. (Pinout definition is best left to the glass manufacturer due to the constraints of routing the common and segment electrodes in two dimensions.)
5. Send the proposed LCD sketch and description for a written quotation to at least 3 vendors to determine pricing, scheduling and quality concerns.
   a) Take into account total NRE cost, price per unit, as well as any setup fees.
   b) Allow a minimum of two weeks for formal mechanical drawings and pin assignments and revised counter drawings.
6. Request a minimal initial prototype LCD build to ensure proper LCD development and ensure proper functionality within the target application.
   a) Allow typically 4-6 weeks for initial LCD prototype delivery upon final approval of mechanical drawings and pin assignments.
7. Upon receipt of prototype LCD, confirm functionality before giving final approval and beginning production of LCD.

Note: Be sure to maintain good records by keeping copies of all materials transferred between both parties, such as initial sketches, drawings, pinouts, etc.

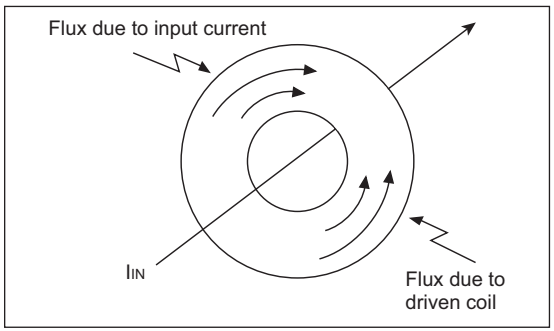
TIP #2 LCD PIC® MCU Segment/ Pixel Table

Table 2-1: Segment Matrix Table

Multiplex Commons	Maximum Number of Segments/Pixels					Bias
	PIC16F913/ 916	PIC16F914/ 917	PIC16F946	PIC18F6X90 (PIC18F6XJ90)	PIC18F8X90 (PIC18F8XJ90)	
Static (COM0)	15	24	42	32/ (33)	48	Static
1/2 (COM1: COM0)	30	48	84	64/ (66)	96	1/2 or 1/3
1/3 (COM2: COM0)	45	72	126	96/ (99)	144	1/2 or 1/3
1/4 (COM3: COM0)	60	96	168	128/ (132)	192	1/3

This Segment Matrix table shows that Microchip's 80-pin LCD devices can drive up to 4 commons and 48 segments (192 pixels), 64-pin devices can drive up to 33 segments (132 pixels), 40/44 pin devices can drive up to 24 segments (96 pixels) and 28-pin devices can drive 15 segments (60 segments).

Figure 15-2: Flux Directions



The net flux in the core should be approximately zero. Because the flux will always be very near zero, the core will be very linear over the small operating range.

When I_{IN} = 0, the output of the comparator will have an approximate 50% duty cycle. As the current moves one direction, the duty cycle will increase. As the current moves the other direction, the duty cycle will decrease. By measuring the duty cycle of the resulting comparator output, we can determine the value of I_{IN}.

Finally, a Delta-Sigma ADC can be used to perform the actual measurement. Features such as comparator sync and Timer1 gate allow the Delta-Sigma conversion to be taken care of entirely in hardware. By taking 65,536 (2^16) samples and counting the number of samples that the comparator output is low or high, we can obtain a 16-bit A/D result.

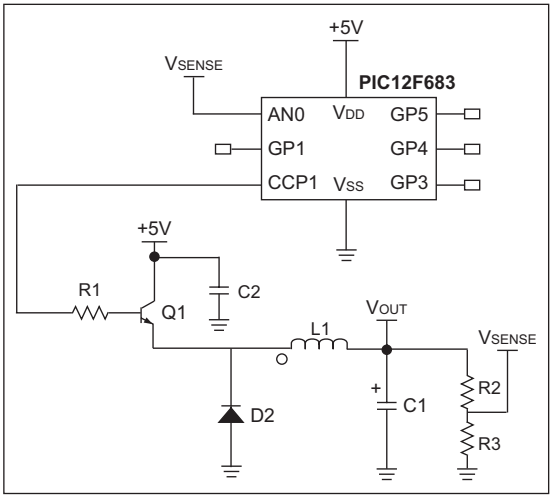
Example schematic and software are provided for the PIC12F683 in both C and Assembly.

For more information on using a PIC MCU to implement a Delta-Sigma converter, please refer to AN700, “*Make a Delta-Sigma Converter Using a Microcontroller’s Analog Comparator Module*” (DS00700), which includes example software.

TIP #16 Implementing a PID Feedback Control in a PIC12F683-Based SMPS Design

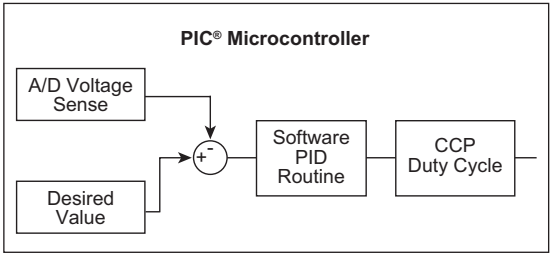
Simple switching power supplies can be controlled digitally using a Proportional Integral Derivative (PID) algorithm in place of an analog error amplifier and sensing the voltage using the Analog-to-Digital Converter (ADC).

Figure 16-1: Simple PID Power Supply



The design in Figure 16-1 utilizes an 8-pin PIC12F683 PIC MCU in a buck topology. The PIC12F683 has the basic building blocks needed to implement this type of power supply: an A/D converter and a CCP module.

Figure 16-2: PID Block Diagram



TIP #19 Execution-Indexed Software State Machine

Another common type of state machine is the execution-indexed state machine. This type of state machine uses a state variable in order to determine what is executed. In C, this can be thought of as the switch statement structure as shown in Example 19-1.

Example 19-1: Example Using Switch Statement

```
SWITCH (State)
{
  CASE 0: IF (in_key()==5)  THEN state = 1;
          Break;
  CASE 1: IF (in_key()==8)  THEN State = 2;
                            Else State = 0;
          Break;
  CASE 2: IF (in_key()==3)  THEN State = 3;
                            Else State = 0;
          Break;
  CASE 3: IF (in_key()==2)  THEN UNLOCK();
                            Else State = 0;
          Break;
}
```

Each time the software runs through the loop, the action taken by the state machine changes with the value in the state variable. By allowing the state machine to control its own state variable, it adds memory, or history, because the current state will be based on previous states. The microcontroller is able to make current decisions based on previous inputs and data.

In assembly, an execution-indexed state machine can be implemented using a jump table.

Example 19-2: Example Using a Jump Table

```
  MOVFW   state      ;load state into w
  ADDWF   PCL,f      ;jump to state
                     ;number
  GOTO    state0     ;state 0
  GOTO    state1     ;state 1
  GOTO    state2     ;state 2
  GOTO    state3     ;state 3
  GOTO    state4     ;state 4
  GOTO    state5     ;state 5
```

In Example 19-2, the program will jump to a GOTO statement based on the state variable. The GOTO statement will send the program to the proper branch. Caution must be taken to ensure that the variable will never be larger than intended. For example, six states (000 to 101) require a three-bit state variable. Should the state variable be set to an undefined state (110 to 111), program behavior would become unpredictable.

Means for safeguarding this problem include:

- Mask off any unused bits of the variable. In the above example, ANDLW b’00000111’ will ensure that only the lower 3 bits of the number contain a value.
- Add extra cases to ensure that there will always be a known jump. For example in this case, two extra states must be added and used as error or Reset states.

Neglecting the affects of R_S and R_L, the formula for determining the values for R1 and R2 is given by Equation 12-1.

Equation 12-1: Divider Values

$$\frac{V_S}{R_1 + R_2} = \frac{V_L}{R_2} \quad \text{; General relationship}$$

$$R_1 = \frac{(V_S - V_L) \cdot R_2}{V_L} \quad \text{; Solving for R1}$$

$$R_1 = 0.515 \cdot R_2 \quad \text{; Substituting voltages}$$

The formula for determining the rise and fall times is given in Equation 12-2. For circuit analysis, the Thevenin equivalent is used to determine the applied voltage, V_A, and the series resistance, R. The Thevenin equivalent is defined as the open circuit voltage divided by the short circuit current. The Thevenin equivalent, R, is determined to be 0.66*R1 and the Thevenin equivalent, V_A, is determined to be 0.66*V_S for the circuit shown in Figure 12-2 according to the limitations imposed by Equation 12-2.

Equation 12-2: Rise/Fall Time

$$t = -\left[R \cdot C \cdot \ln\left(\frac{V_F - V_A}{V_I - V_A}\right)\right]$$

Where:

- t = Rise or Fall time
- R = 0.66*R_1
- C = C_S+C_L
- V_I = Initial voltage on C (V_L)
- V_F = Final voltage on C (V_L)
- V_A = Applied voltage (0.66*V_S)

As an example, suppose the following conditions exist:

- Stray capacitance = 30 pF
- Load capacitance = 5 pF
- Maximum rise time from 0.3V to 3V ≤ 1 µS
- Applied source voltage Vs = 5V

The calculation to determine the *maximum* resistances is shown in Equation 12-3.

Equation 12-3: Example Calculation

Solve Equation 12-2 for *R*:

$$R = -\left[\frac{t}{C \cdot \ln\left(\frac{V_F - V_A}{V_I - V_A}\right)}\right]$$

Substitute values:

$$R = -\left[\frac{10 \cdot 10^{-7}}{35 \cdot 10^{-12} \cdot \ln\left(\frac{3 - (0.66 \cdot 5)}{0.3 - (0.66 \cdot 5)}\right)}\right]$$

Thevenin equivalent maximum R:

$R = 12408$

Solve for maximum R1 and R2:

$R1 = 0.66 \cdot R$ $\qquad$ $R2 = \frac{R1}{0.515}$

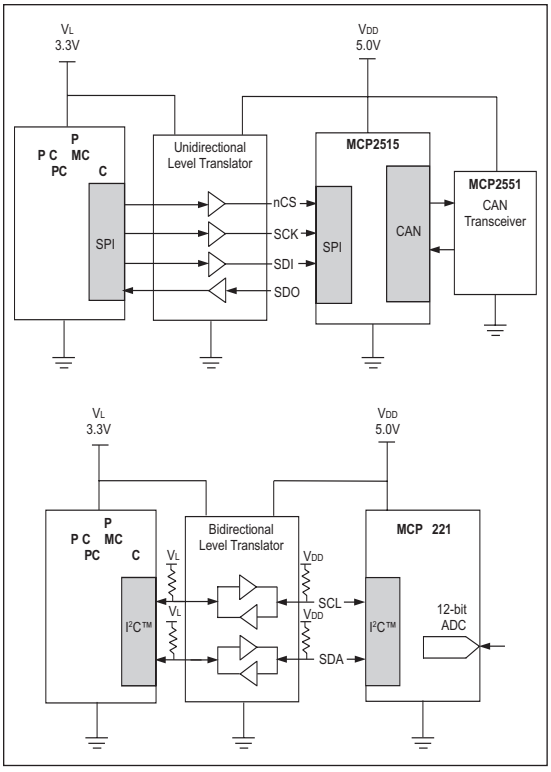
$R1 = 8190$ $\qquad$ $R2 = 15902$

TIP #13 3.3V → 5V Level Translators

While level translation can be done discretely, it is often preferred to use an integrated solution. Level translators are available in a wide range of capabilities. There are unidirectional and bidirectional configurations, different voltage translations and different speeds, all giving the user the ability to select the best solution.

Board-level communication between devices (e.g., MCU to peripheral) is most often done by either SPI or I²C™. For SPI, it may be appropriate to use a unidirectional level translator and for I²C, it is necessary to use a bidirectional solution. Figure 13-1 illustrates both solutions.

Figure 13-1: Level Translator



Analog

The final 3.3V to 5V interface challenge is the translation of analog signals across the power supply barrier. While low level signals will probably not require external circuitry, signals moving between 3.3V and 5V systems will be affected by the change in supply. For example, a 1V peak analog signal converted by an ADC in a 3.3V system will have greater resolution than an ADC in a 5V system, simply because more of the ADCs range is used to convert the signal in the 3.3V ADC. Alternately, the relatively higher signal amplitude in a 3.3V system may have problems with the system's lower common mode voltage limitations.

Therefore, some interface circuitry, to compensate for the differences, may be needed. This section will discuss interface circuitry to help alleviate these problems when the signal makes the transition between the different supply voltages.

Table 18-1: Bipolar Transistor DC Specifications

Characteristic	Sym	Min	Max	Unit	Test Condition
OFF CHARACTERISTICS					
Collector-Base Breakdown Voltage	$V_{(BR)CBO}$	60	–	V	I_C = 50 µA, I_E = 0
Collector-Emitter Breakdown Voltage	$V_{(BR)CEO}$	50	–	V	I_C = 1.0 mA, I_B = 0
Emitter-Base Breakdown Voltage	$V_{(BR)EBO}$	7.0	–	V	I_E = 50 µA, I_C = 0
Collector Cutoff Current	I_{CBO}	–	100	nA	V_{CB} = 60V
Emitter Cutoff Current	I_{EBO}	–	100	nA	V_{EB} = 7.0V
ON CHARACTERISTICS					
DC Current Gain	h_{FE}	120 180 270	270 390 560	–	V_{CE} = 6.0V, I_C = 1.0 mA
Collector-Emitter Saturation Voltage	$V_{CE(SAT)}$	–	0.4	V	I_C = 50 mA, I_B = 5.0 mA

When using bipolar transistors as switches to turn on and off loads controlled by the microcontroller I/O port pin, use the minimum h_{FE} specification and margin to ensure complete device saturation.

Equation 18-1: Calculating the Base Resistor Value

$$R_{BASE} = \frac{(V_{DD} - V_{BE}) \times h_{FE} \times R_{LOAD}}{V_{LOAD}}$$

3V Technology Example

V_{DD} = +3V, V_{LOAD} = +40V, R_{LOAD} = 400Ω, h_{FE} min. = 180, V_{BE} = 0.7V

R_{BASE} = 4.14 kΩ, I/O port current = 556 µA

5V Technology Example

V_{DD} = +5V, V_{LOAD} = +40V, R_{LOAD} = 400Ω, h_{FE} min. = 180, V_{BE} = 0.7V

R_{BASE} = 7.74 kΩ, I/O port current = 556 µA

For both examples, it is good practice to increase base current for margin. Driving the base with 1 mA to 2 mA would ensure saturation at the expense of increasing the input power consumption.