MICROCHIP

AN701

Switch Mode Battery Eliminator Based on a PIC16C72A

Author: Brett Duane
Microchip Technology

OVERVIEW

The PIC16C72A is a member of the PICmicro® Mid-Range Family of 8-bit, high-speed microcontrollers. The PIC16C72 provides the following features:

- 5 Channel, 8-bit Analog-to-Digital Converter (A/D)
- CCP Module to generate a PWM output
- I²C™/SPI™ Module
- 3 Timers
- 8 Interrupt sources

This application note shows how to combine the A/D and CCP modules with suitable software to produce a Switch Mode Battery Eliminator (SMBE) providing 3.0, 4.5, 5.0, 6.0, 7.5 and 9.0 volt output voltages at up to 1 Amp with an AC or DC input between 12.6V and 30V peak.

HARDWARE

The system makes use of the A/D to read the input and output voltages, the Capture/Compare/Pulse module to generate a PWM output, and Timer2 to regulate how fast the program runs. External hardware includes a switching power converter and a suitable output filter. Six LEDs on PORTB indicate the output voltage as set by two push buttons on PORTA.

Optional components not installed in this project include a serial EEPROM to store the last voltage setting and a level translator to convert TTL to RS-232 for communications with a PC.

In-Circuit Serial Programming™ (ICSP) support has also been provided. LEDs D7 and D8 share clock and data lines required for ICSP. These LEDs indicate error conditions and are optional.

Analog-to-Digital Converter Module

The A/D converts an input voltage between ground and VDD to an 8-bit value presented in ADRES. In this application, the switching converter input and output voltages are sampled. Provisions have been included to read the setting of a potentiometer.

Capture/Compare/Pulse Width Modulation Module

The CCP module produces the PWM signal that controls the series pass switching transistor. Depending on the PWM period and FOSC, any number of bits between 2 and 10 bits may be used to specify the PWM on-time. The CCP module requires the use of Timer2, the Timer2 prescaler, and the PR2 register to produce a PWM output.

Timer2 Postscaler

Timer2 drives the CCP module to control the PWM period and also drives the Timer2 postscaler. The postscaler is incremented when Timer2 is reset at the end of each PWM cycle and will generate an interrupt when the postscaler overflows.

External Hardware

The switching buck converter relies on three components to function:

- Series pass switch (Q1)
- Inductor (L1)
- Commutating diode (D10).

These devices form the core of all switch mode buck converters.

AN701

Power Input Circuit

This circuit is a conventional linear power supply that accepts AC or DC power with a peak voltage of 30V (limited by the 78L05). The converter operates off the unregulated bulk power, while the regulator supplies power and a voltage reference to the controller.

Figure 1 shows the Power Input Circuit. Bridge rectifier BR1 rectifies the raw power input that may be AC or DC. Capacitor C6 provides rough filtering to reduce ripple in the input voltage. Capacitor C8 provides the short current pulses drawn by transistor Q1 when it is turned on. Capacitor C7 provides filtering of regulator U2 output.

FIGURE 1:   POWER INPUT CIRCUIT
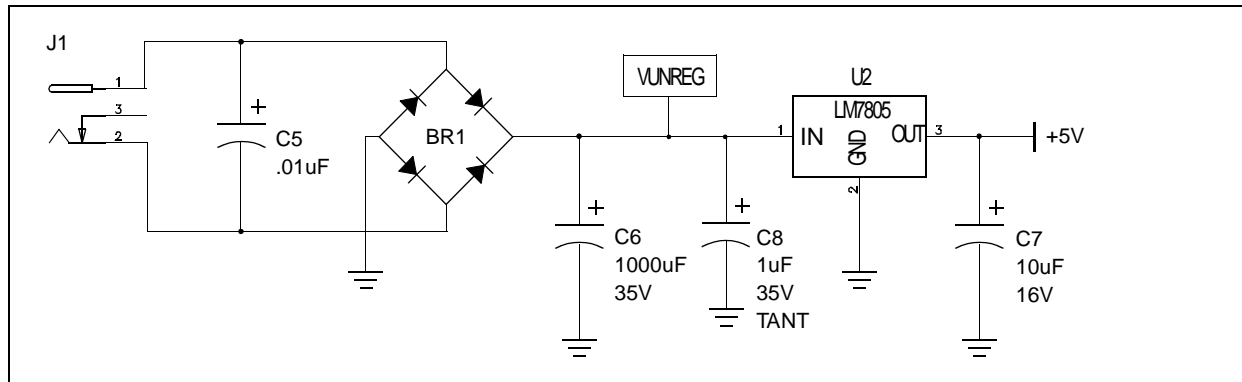
Power Converter

Figure 2 shows the switching buck converter with drive circuits. Unregulated DC is provided at the emitter of transistor Q1. Q1, inductor L1 and diode D10 form the basic buck switching converter. The output appears at connector J4.

When RC2 (PWM output) is high, transistor Q2 is turned on, pulling Q2's collector to ground. This draws current from Q1's base, turning Q1 on. When Q1 is on, current from capacitors C6 and C8 charge L1 through the load. Resistor R19 limits the current drawn from the base of Q1. Resistor R17 ensures that Q1 switches off quickly. Resistor R20 ensures that transistor Q2 switches off quickly. Resistor R18 limits Q2's base current.

When RC2 is low, R20 turns off Q2 and R17 turns off Q1. When Q1 is off, the current through L1 continues to flow through L1, D10 and the load, discharging L1. When the current through L1 becomes less than the current drawn by the load, C10 provides the additional current and reduces output voltage ripple.

To ensure that Q1 remains cool during operation, it must be driven well into saturation. When driving transistors as digital switches, divide their h_{FE} (small signal gain) by 5 and use the resulting gain for your calculations. This ensures that the transistor switches through its linear region quickly to prevent significant heat generation in the transistor.

As the unregulated input voltage decreases, the drive applied to Q1 decreases to the point where Q1 starts operating in its linear region, producing heat. Continued operation in the linear region will cause Q1 to overheat and fail. Q1 usually shorts, causing the input voltage to appear at the converter output. R19 was selected to allow Q1 to operate safely as long as V_{UNREG} is above 10V. The controller software will shut down the converter if V_{UNREG} falls below 10V.

The converter output contains a considerable amount of noise. Capacitors C10 and C11 provide filtering to reduce that noise. F1 is a PTC resistor acting as a 1 Amp, self-resetting fuse.

FIGURE 2:   SWITCHING BUCK CONVERTER WITH DRIVE AND OUTPUT CIRCUITS

AN701
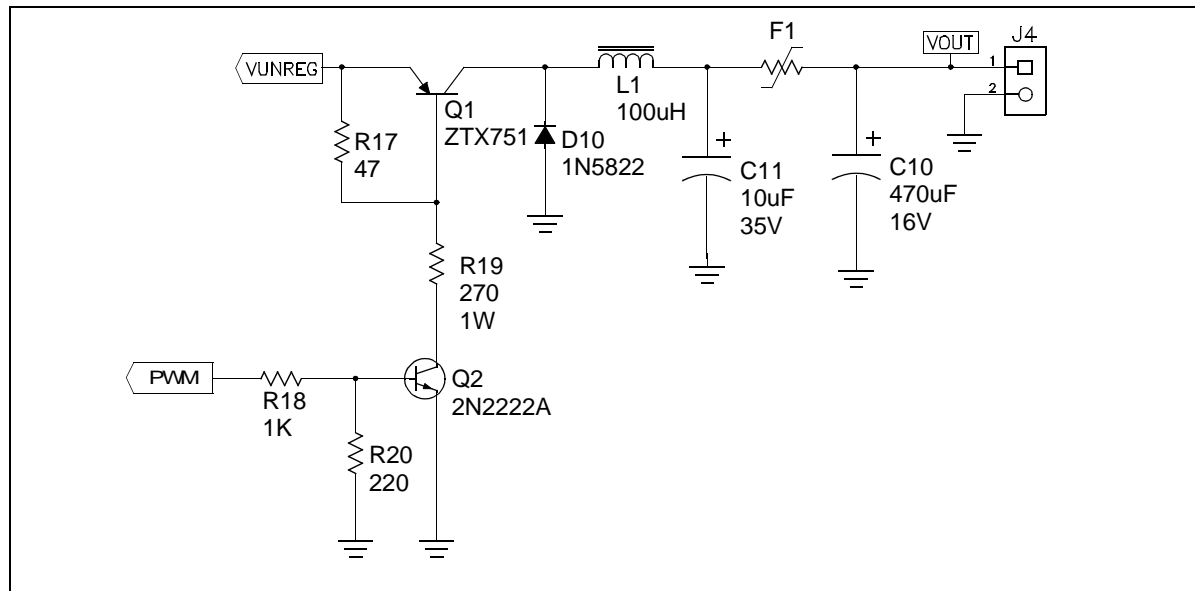
Microcontroller Circuits

The microcontroller, analog inputs and digital outputs are shown in Figure 3.

The LEDs indicate the output voltage (D1-6) and converter faults (D7-8). Switches S1 and S2 allow the user to select the desired output voltage. Resistors R3, R4 and capacitor C4 form the voltage feedback circuit. Resistors R15, R16 and capacitor C13 form the voltage source sense circuit. PWM is output at pin RC2.

Register packs RN5 and RN6 limit current through LEDs D1-D8. LEDs D7 and D8 share clock and data lines required for ICSP. Jumper J1 disables all LEDs.

When programming the controller using ICSP, J1 should be removed. If ICSP does not function properly, D7 and D8 or RN6 should be installed after programming.

S1 is the decrease voltage button and S2 is the increase voltage button. R13 and R14 are pull down resistors.

Resonator Y1, and capacitors C2 and C3 set Fosc for the controller. If Y1 is a ceramic resonator with internal capacitors, C2 and C3 are not required. Resistor R1 pulls $\overline{MCLR}$ to +5V while allowing the controller to be programmed using ICSP. In addition, connecting pins 1 and 3 of connector J2 causes a remote reset of the controller. (See figure 5.)

FIGURE 3:   MICROCONTROLLER, ANALOG INPUTS AND DIGITAL INPUTS AND OUTPUTS
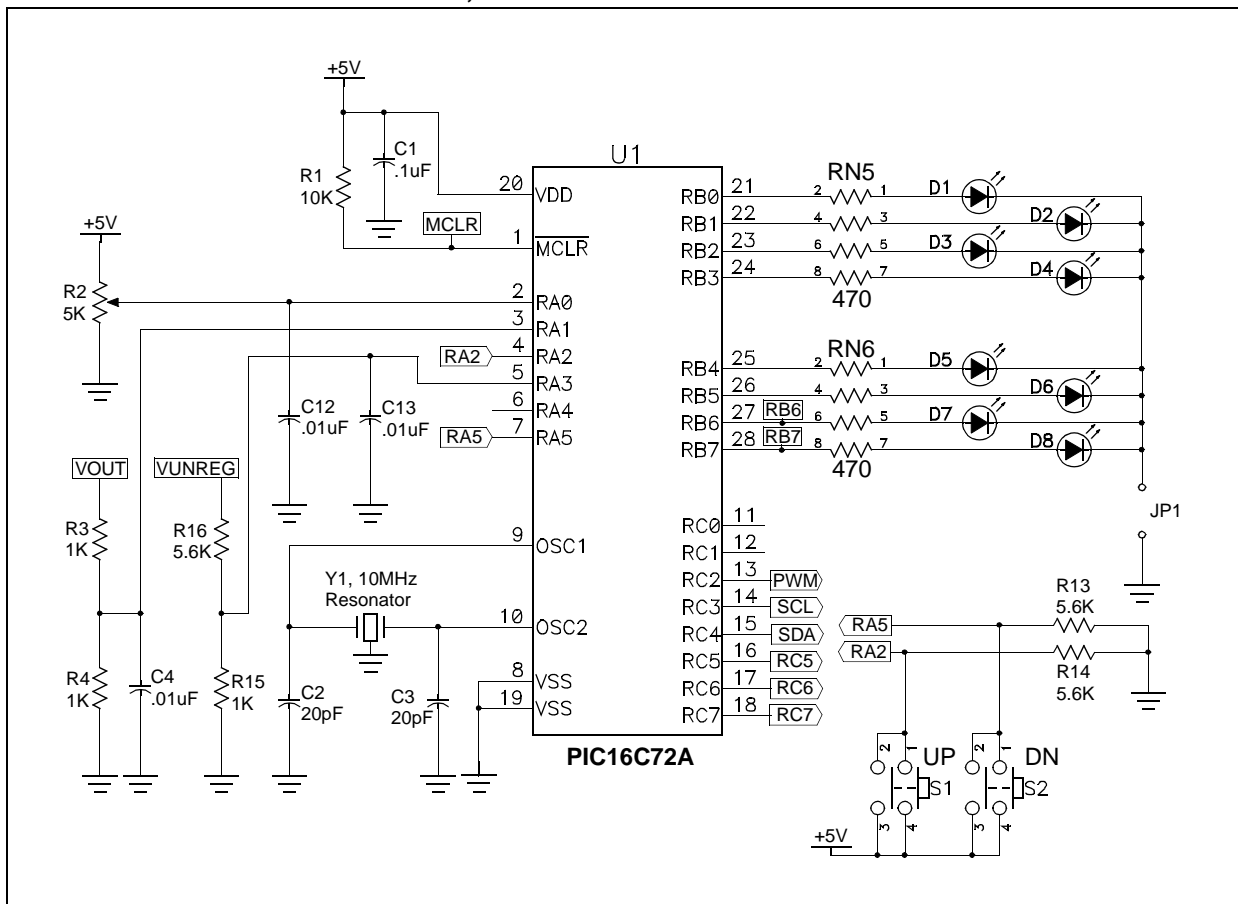
FIRMWARE

Initialization

The controller is first initialized by configuring the A/D, CCP and Timer2 peripherals, followed by clearing the RAM required for variables and initializing some variables. Controller pins are configured as required for each of the modules, or as digital outputs if they are not being used.

A/D

Pins RA0, RA1 and RA3 are configured as analog inputs. Pins RA2 and RA5 are used as digital inputs. The controller V_{DD} is used as V_{REF} for the A/D.

The conversion clock source T_{AD} is selected to be between 1.6usec and 6.4usec. Since Tosc = 0.1usec (Fosc = 10MHz), 32Tosc = 3.2usec. The A/D module is turned on and pin RA1 (V_{OUT}) is sampled for conversion later in the loop.

TRISA configures pin RA4 as an output and all others as inputs.

CCP (PWM)

The CCP module is set to PWM mode. Timer2 is enabled with a 1:1 prescaler. PR2 is set to 63 (0x3F). The resulting PWM frequency is 39.063KHz. (TPWM=25.6msec). The CCP module uses 8-bit data to control the PWM duty cycle. The PWM duty cycle is set to 0, ensuring that the PWM output is turned off.

All pins on PORTC are configured as outputs, including RC2 which is the controller PWM output.

Timer2 postscaler

Since Timer2 will also control the frequency that the main loop will execute, the Timer2 postscaler is set to a 1:1 ratio and the Timer2 postscaler interrupt is enabled. This will cause one interrupt for each PWM cycle.

RAM

RAM required for variables is cleared. The 3.0V LED on PORTB is lit. The variable set_pt is initialized to produce the 3.0V output. Button debounce counters are initialized.

Main loop

For a digital control loop to function as well as an analog controller, the digital control loop should repeat at least 30 times faster than the fastest expected transient.

The ripple frequency at capacitor C7 is 120Hz, or twice the AC power line frequency. In this application, the loop must be executed at least 120 x 30, or 3600 times a second to adequately respond to the bulk power ripple. Transients at the converter output are also handled, but less effectively with increasing frequency.

The Timer2 postscaler generates interrupts that are counted by the interrupt service routine. When 8 interrupts have occurred, the main loop is allowed to execute once. This causes the main loop to execute 4883 times a second.

The A/D starts a conversion on RA1/AN1 (V_{OUT}). The program loops wait for the conversion to complete. The 8-bit result is placed in V_{OUT}. The A/D is then set to sample the input voltage (V_{UNREG}).

PID Controller

The control algorithm is a software implementation of a Proportional-Integral-Differential (PID) controller. The only input to this controller is the difference between the desired output voltage and the actual output voltage, and is known as the error signal.

The first module produces the error signal by finding the difference between set_pt and V_{OUT}. The result is saved in the low byte of a 2 byte signed variable e0h:e0. The high byte is set to reflect a negative value if needed. The difference is selected to produce a positive result if set_pt is greater than V_{OUT}.

    e0 = set_pt - vout
    if e0<7>=0 , e0h = 0x00, else e0h = 0xFF

A proportional term is generated from the present error signal. This is simply the signed error multiplied by some factor Kp. The 2 byte signed result is saved as proh:pro.

    proh:pro = Kp * e0h:e0
    Kp = proportional factor

The difference between the present and previous error is found and saved as the 2 byte signed result difh:dif. The previous error e1h:e1 is simply the error result found in the execution of the previous loop. The difference between errors is multiplied by some factor (Kd) and saved as difh:dif.

    difh:dif = Kd * (e1h:e1 – e0h:e0)
    Kd=difference factor

The integral component is nothing more than a total of all the errors produced since the last reset. The present error is multiplied by some factor (Ki) before being added to the running 2 byte unsigned total inth:int.

    inth:int = inth:int + Ki * e0h:e0
    Ki = integral factor

The proportional, integral and difference terms are summed together. The result of the sum is saved as the 2 byte signed result pwmh:pwm.

    pwmh:pwm = proh:pro + difh:dif
                + inth:int

The result is never greatly positive, but can sometimes cause the result to underflow. When `pwmh:pwm` underflows (as it does when the load is disconnected), the PWM drive signal must be forced to zero or the converter output becomes unpredictable. The overload LED is also lit.

> If `pwmh<7>` = 1, then `pwmh:pwm` = `0x0000` and turn on the OVLD LED, else turn off OVLD LED.

PWM Generator

The PWM generator module (software, not the peripheral) discards the 3 least and 5 most significant bits, and uses the remaining 8 bits to generate PWM with a desired on time. Of the remaining 8 bits, the 2 least significant bits are loaded into `CCP1CON<5:4>`. The last 6 bits (with 2 leading zeros to form a byte) are loaded into `CCP1RL`.

Conversion of the input voltage V_{UNREG} is started.

Reading Buttons

The LED data at PORTB is copied to a temporary variable.

The down button is read. If it is closed, its debounce counter is incremented. If no overflow occurs, execution proceeds to reading the up button. If an overflow does occur, the LED data is shifted right one bit, unless bit 0 is already set. Overflows occur approximately every half second.

The up button is read and processed similar to the down button. If it is closed, its debounce counter is incremented. If no overflow occurs, execution proceeds to convert V_{IN}. If an overflow does occur, the LED data is shifted left one bit, unless bit 5 is already set. Overflows occur approximately every half second.

The LED data is copied back to PORTB to light the corresponding LED. The LED data is also used to find the proper index to use prior to calling a look-up table.

A call to the lookup table is performed. The lookup table routine adds the index in the `W` register to the program counter. The next instruction performed is a "`retlw`" instruction that places the new `set_pt` in the `W` register and returns to the next instruction after the call to the look-up table. The value returned is saved as the new `set_pt`.

Both button debounce counters are reset.

The conversion result of V_{IN} is retrieved from the A/D and `vin` is subtracted from `0xC1` (10V set point). If the result is positive, V_{IN} is less than 10V and program execution is directed to a safety shutdown module. Otherwise, program execution continues normally.

> If (`0xC1` – `vin`) is positive, go to shutdown.

The present error, `e0h:e0`, replaces the previous error, `e1h:e1`.

Program execution then returns to the top of the loop to wait for Timer2 postscaler interrupts.

Interrupts

The Interrupt Service Routine saves the state of the `W` and `STATUS` registers, increments the interrupt counter and restores the `STATUS` and `W` registers.

Safety Shutdown

The safety shutdown module has been included to turn off the converter and to light the trip LED. Once entered, there is no exit from this module except by resetting the controller.

Gain constants Kp, Kd, and Ki

Constants Kp, Ki, and Kd were determined experimentally. The goal was to maintain V_{OUT} between 4.75V and 5.25V when the 5V output was selected. V_{OUT} should remain within this band, regardless of changes in the load current. This specifically includes 10% changes in current, unloaded to full-load, and full-load to unloaded step changes. Other step changes in loading were also examined.

Constants determined for a 5V output were then used for other voltage outputs.

Only resistive loads were considered. Some degradation of output performance may be expected with inductive or capacitive loads.

APPENDIX A: SWITCH MODE BUCK CONVERTER

A switch mode buck converter performs a voltage reduction by periodically charging an inductor from a high voltage source, then allowing the charged inductor to transfer its stored energy to the load at a lower voltage. This energy transfer occurs with little loss.

FIGURE 4: SWITCH MODE BUCK CONVERTER TOPOLOGY
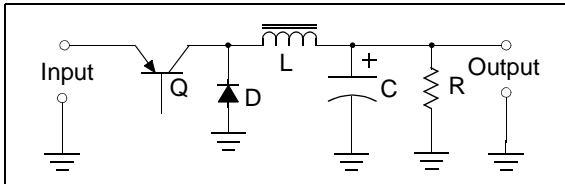
In the ideal buck converter, the average output power equals the average input power. The switch is operated at a constant frequency, but the duty cycle (on time / cycle time) controls the output voltage.

When the switch Q (a transistor or MOSFET) is closed, current from the source flows through the inductor L, through the load R and back to the source. Energy is being stored in the inductor by increasing inductor current and building up a magnetic field.

When the switch is open, the source no longer supplies energy. The energy stored in the magnetic field is transferred to the load as the magnetic field collapses. Inductor current is decreasing as current flows from the inductor through the load R and diode D and back to the inductor L.

No energy is lost in the inductor during this conversion. The majority of losses occur in the switching device as it switches though its linear region, and the commutation diode when it conducts due to its forward voltage drop. Most electrolytic filter capacitors also have high resistance to the high frequency ripple current.

If current flows through the inductor at all times, the converter is operating in the continuous mode. The average inductor current is the same as the load current. If the load current is constant, variations in the inductor current average to zero. The peak inductor current will be no greater than twice the average current, and can be reduced by raising the switching frequency. This is normally the desired operating mode.

If the current through the inductor stops, the converter is operating in the discontinuous mode. All the current that flows through the load continuously must flow through the switch and charge the inductor during the time the switch is turned on. This can result in very high currents in both the switch and inductor and risks saturating the inductor. When the inductor is saturated, its inductance decreases drastically. Considerable noise is also produced as large currents are switched, requiring a greater amount of filtering. This mode is normally used only when there is very light loading of the converter, but it is easier to stabilize than the continuous mode converter.

In both continuous and discontinuous modes, charging or discharging a filter capacitor at the converter output makes up the difference between the inductor and load currents. This filter capacitor also reduces output ripple voltage and noise that switching converters produce.

The equations presented here assume an ideal circuit, but actual circuits have similar results.

The approximate duty cycle D.C. can be approximated by:

$$D.C. = \frac{V_{OUT}}{V_{IN}} = \frac{I_{IN}}{I_{OUT}} = \frac{T_{ON}}{T_{PWM}}$$

Where V_{OUT} = output voltage,
 I_{IN} = average input current,
 T_{ON} = switch on time
 V_{IN} = input voltage,
 I_{OUT} = output current,
 T_{PWM} = PWM period = $1/F_{PWM}$

The ripple current magnitude due to switching is approximated by:

$$I_{RIPPLE} = \frac{(V_{IN}-V_{OUT}) * D.C. * T_{PWM}}{L}$$

Where L = inductor inductance,
 I_{RIPPLE} = ripple current peak-to-peak
 F_{PWM} = PWM frequency

The ripple current is absorbed by the output filter capacitor C, but produces a small output ripple voltage that is approximated by:

$$V_{RIPPLE} = \frac{I_{RIPPLE} * D.C. * T_{PWM}}{C}$$

Where V_{RIPPLE} = output voltage ripple peak-to-peak

EXAMPLE 1: CAPACITOR AND INDUCTOR SELECTION

Given: V_{UNREG} = 20V, V_{OUT} = 6V,
 V_{RIPPLE} = 0.1V, I_{RIPPLE} = 1A,
 F_{PWM} = 39.063KHz

Solution:

$$D.C. = \frac{V_{OUT}}{V_{IN}} = \frac{6}{20} = 0.30$$

$$T_{PWM} = \frac{1}{F_{PWM}} = \frac{1}{39KHz} = 25.6uS$$

$$C = \frac{I_{RIP}*D.C.*T_{PWM}}{V_{RIPPLE}} = \frac{(1A)(.3)(25.6uS)}{0.1V}$$
$$= 76.8uF$$

$$I_{RIP} = \frac{(V_{IN}-V_{OUT})*D.C.*T_{PWM}}{L}$$

$$L = \frac{(V_{IN}-V_{OUT})*D.C.*T_{PWM}}{I_{RIP}}$$
$$= \frac{(20V-6V)(0.30)(25.6uS)}{1A}$$
$$= 107.5uH$$

AN701

APPENDIX B: ACCESSORY COMPONENTS

These accessory component are not installed on the board, but can provide additional capabilities. Connector J2 provides for In-Circuit Serial Programming (ICSP) and offers a remote $\overline{MCLR}$ reset by connecting pins 1 and 3. Integrated circuit U3 and connector J3 provide RS-232 communications with the controller. Integrated circuit U4 is a serial EEPROM that communicates with the controller using the I^{2}C™ protocol.

FIGURE 5:   ACCESSORY COMPONENTS

© 1999 Microchip Technology Inc.
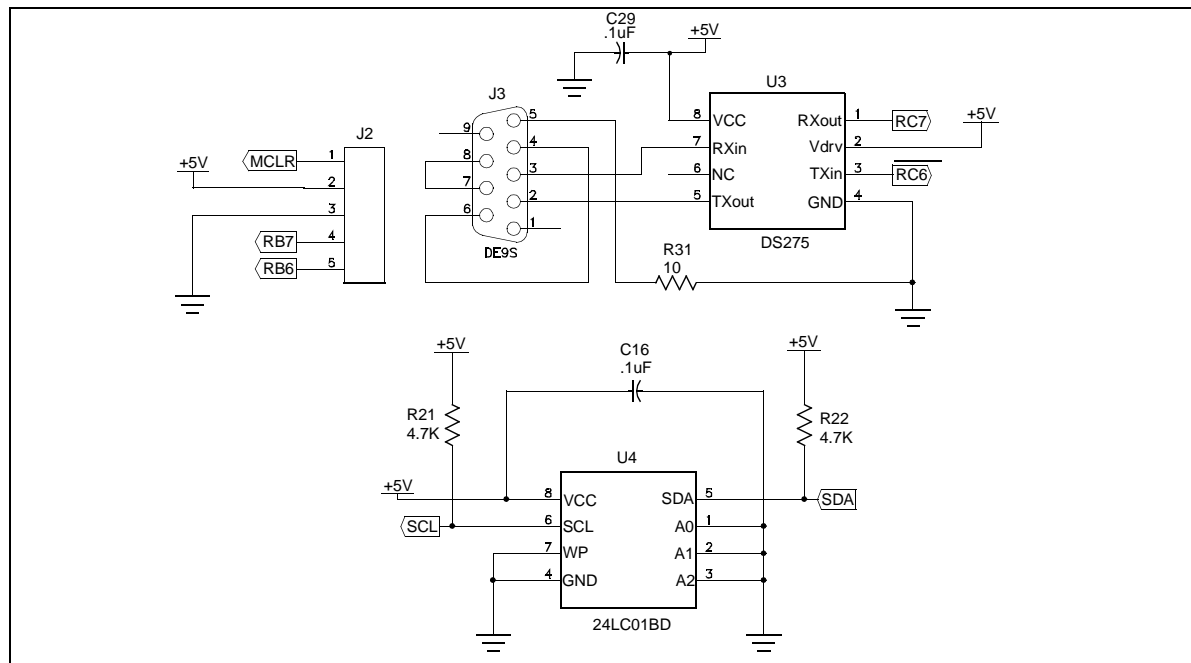
APPENDIX C: CODE

```
MPASM 02.20 Released        SW_REG1A.ASM   3-9-1999  19:28:40


LOC  OBJECT CODE    LINE SOURCE TEXT
   VALUE


              00001
;*****************************************************************************
              00002 ;*   Filename: APPNOTE.ASM
              00003
;*****************************************************************************
              00004 ;*        Author:      Brett Duane
              00005 ;*        Company:     Microchip Technology
              00006 ;*        Revision:    Rev 1.0
              00007 ;*        Date:        3-9-99
              00008 ;*        Assembled using MPASM rev 4.00.00
              00009
;*****************************************************************************
              00010 ;*        Include files: p16c72a.inc Rev 1.01
              00011
;*****************************************************************************
              00012 ;*
              00013 ;*        Switching buck regulator using 16C72A using A/D,
              00014 ;*        PWM (CCP) and timer2 modules.
              00015 ;*
              00016 ;*        PID control loop implementation.
              00017 ;*        Executes 4883 loops per second.
              00018 ;*        Spends most of its time looping waiting for timer2 overflows.
              00019 ;*
              00020 ;*        A/D inputs and PWM output are 8 bits,
              00021 ;*        with internal calculations done in 16 bits
              00022 ;*
              00023 ;*        Timer2 postscaler (1:16) generates an interrupt.
              00024 ;*
              00025 ;*        RA0 converts a 0-5V input from trimmer, but is not used.
              00026 ;*        RA1 monitors VOUT.
              00027 ;*        RA2 is the up voltage push button input.
              00028 ;*        RA3 monitors VUNREG.
              00029 ;*        RA5 is the down voltage push button input.
              00030 ;*
              00031 ;*        RB<5:0> drives LEDs to indicate the output voltage.
              00032 ;*        RB<7:6> drives LEDs to indicate overload or shutdown.
              00033 ;*
              00034 ;*        RC2 is the PWM source for the switching converter.
              00035 ;*
              00036 ;*
              00037 ;*        Configuration Bit Settings
              00038 ;*            Brown-out detect is off
              00039 ;*            Code protect is off
              00040 ;*            Power-up timer is enabled
              00041 ;*            Watchdog timer is disabled
              00042 ;*            10MHz resonator is driven in XT mode
              00043 ;*
              00044 ;*        Program Memory Words Used:   230
              00045 ;*        Program Memory Words Free:  1818
              00046 ;*
              00047 ;*        Data Memory Bytes Used:       24
              00048 ;*        Data Memory Bytes Free:      104
              00049 ;*
              00050
;*****************************************************************************
              00051 ;*        What's Changed
              00052 ;*
              00053 ;*        Date       Description of Change
```

```
                00054 ;*
                00055 ;*    3-3-99      This is the initial release.
                00056 ;*
                00057
;***********************************************************************
                00058
                00059      list p=16c72a
                00060      #include   <p16c72a.inc>
                00001      LIST
                00002 ; P16C72A.INC Standard Header File,Version 1.01 Microchip Technology, Inc.
                00249      LIST
                00061
2007 3FB1       00062      __config _BODEN_OFF & _CP_OFF & _PWRTE_ON & _WDT_OFF & _XT_OSC
                00063
                00064
                00065          cblock 0x020
                00066
  00000020      00067              UPCL     ;up button debounce
  00000021      00068              UPCH     ;up button debounce
  00000022      00069              DNCL     ;down button debounce
  00000023      00070              DNCH     ;down button debounce
  00000024      00071              SETPOINT ;voltage setpoint - RA0 result (unsigned)
  00000025      00072              VOUT     ;output voltage feedback - RA1 result
  00000026      00073              VUNREG   ;source voltage feedback - RA3 result
  00000027      00074              TEMPA    ;temp variable
  00000028      00075              TEMPB    ;temp variable
  00000029      00076              INT      ;integral component
  0000002A      00077              INTH     ;integral component
  0000002B      00078              PRO      ;proportional component
  0000002C      00079              PROH     ;proportional component
  0000002D      00080              DIF      ;difference component
  0000002E      00081              DIFH     ;difference component
  0000002F      00082              PWM      ;PWM drive
  00000030      00083              PWMH     ;PWM drive
  00000031      00084              E0       ;present error
  00000032      00085              E0H      ;present error
  00000033      00086              E1       ;past error
  00000034      00087              E1H      ;past error
  00000035      00088              T2POST   ;postscaler interrupt counter
  00000036      00089              ISRS     ;isr variable
  00000037      00090              ISRW     ;isr variable
                00091
                00092          endc
                00093
  000000F9      00094 DEL1    equ 0xf9     ;text equate - debounce delay
  00000089      00095 AVOUT   equ 0x89     ;text equate - select VOUT channel
  00000099      00096 AVUNREG equ 0x99     ;text equate - select VUNREG channel
                00097
                00098
;***********************************************************************
                00099 ;*    Reset vector
                00100 ;*        This is the reset vector
                00101 ;*
                00102
;***********************************************************************
                00103
0000            00104          org    0x00   ;reset vector
0000 280E       00105          goto   Main   ;program start
                00106
                00107
                00108
;***********************************************************************
                00109 ;*      INTERRUPT SERVICE ROUTINE
                00110 ;*          This ISR counts timer2 interrupts
                00111 ;*
                00112 ;*      Input Variables:
```

```
                00113 ;*      T2POST       Counts overflow interrupts
                00114 ;*
                00115 ;*   Output Variables:
                00116 ;*      T2POST       Counts overflow interrupts
                00117 ;*
                00118
;******************************************************************************
                00119
0004            00120       org    0x04     ;interrupt service routine
                00121
0004 00B7       00122       movwf  ISRW     ;save W
0005 0E03       00123       swapf  STATUS,W ;get status
0006 00B6       00124       movwf  ISRS     ;save status
                00125
0007 108C       00126       bcf    PIR1,TMR2IF;clear interrupt request flag
                00127
0008 0AB5       00128       incf   T2POST,F ;increment interrupt counter
                00129
0009 0E36       00130       swapf  ISRS,W   ;get status
000A 0083       00131       movwf  STATUS   ;restore status
000B 0EB7       00132       swapf  ISRW,F   ;restore W
000C 0E37       00133       swapf  ISRW,W   ;restore W
                00134
000D 0009       00135       retfie          ;return from interrupt
                00136
                00137
;******************************************************************************
                00138 ;*   Main   Beginning of main loop
                00139 ;*          First segment of code initializes controller
                00140
;******************************************************************************
                00141
000E            00142 Main
000E 1303       00143       bcf    STATUS,RP1;bank1 initialization
000F 1683       00144       bsf    STATUS,RP0;bank1
                00145 ;adc
0010 3004       00146       movlw  0x04     ;RA0,1,3 analog, RA2,5 digital, Vref=Vdd
0011 009F       00147       movwf  ADCON1
                00148
0012 302F       00149       movlw  0x2f     ;RA0,1,3 analog in; RA2,5 digital in, RA4 dig out
0013 0085       00150       movwf  TRISA
                00151 ;PWM
0014 30FF       00152       movlw  0xFF     ;set portC to all inputs
0015 0087       00153       movwf  TRISC
0016 1107       00154       bcf    TRISC,2  ;make RC2/PWM output
                00155
0017 303F       00156       movlw  .63      ;PWM period =  78.125KHz (8 bit resolution)
0018 0092       00157       movwf  PR2
                00158
                00159 ;timer2
0019 148C       00160       bsf    PIE1,TMR2IE;enable timer2 postscaler interrupts
                00161
                00162 ;display
001A 0186       00163       clrf   TRISB    ;make PORTB all outputs
                00164
001B 1283       00165       bcf    STATUS,RP0;select bank0
                00166 ;adc
001C 3089       00167       movlw  AVOUT    ;(10MHz osc)set A/D conv clock(Fosc/32),
001D 009F       00168       movwf  ADCON0   ;select RA1(AN1), turn on A/D
                00169 ;PWM
001E 1217       00170       bcf    CCP1CON,4;clear ls bits of PWM duty cycle
001F 1297       00171       bcf    CCP1CON,5
0020 0195       00172       clrf   CCPR1L   ;set PWM to 0 (turn off PWM)
                00173
0021 3004       00174       movlw  0x04     ;enable timer2, set prescale=1:1,
                                                      postscale=1:1
```

```
0022 0092     00175            movwf   T2CON
0023 300C     00176            movlw   0x0C       ;set CCP1 to PWM mode
0024 0097     00177            movwf   CCP1CON
              00178
              00179 ;****************************************************************************
              00180 ;*        Restart  Clears memory
              00181 ;*
              00182 ;*               Initializes display LEDs, desired output voltage,
              00183 ;*               debounce counters
              00184 ;*
              00185 ;*               Enables interrupts
              00186 ;*
              00187 ;*        Output Variables:
              00188 ;*            SETPOINT    desired output voltage, set to 3.0V
              00189 ;*            DNCL, DNCH  Down voltage button debounce counter
              00190 ;*            UPCL, UPCH  Up voltage button debounce counter
              00191 ;*
              00192 ;****************************************************************************
              00193
0025 3020     00194 Restart  movlw   0x20       ;clear memory from 0x20 to 0x3f
0026 0084     00195            movwf   FSR
0027 0180     00196 ClrMem   clrf    INDF
0028 0A84     00197            incf    FSR,F
0029 1F04     00198            btfss   FSR,6
002A 2827     00199            goto    ClrMem
              00200
002B 3001     00201            movlw   0x01       ;light 3.0V LED
002C 0086     00202            movwf   PORTB
              00203
002D 304D     00204            movlw   0x4d       ;initial 3.0V setting
002E 00A4     00205            movwf   SETPOINT
              00206
002F 01A2     00207            clrf    DNCL       ;clear down button debounce counter
0030 01A0     00208            clrf    UPCL       ;clear up button debounce counter
              00209
0031 30F9     00210            movlw   DEL1
0032 00A3     00211            movwf   DNCH       ;preset down button debounce counter byte
0033 00A1     00212            movwf   UPCH       ;preset up button debounce counter high byte
              00213
0034 170B     00214            bsf     INTCON,PEIE ;enable peripheral interrupt sources
0035 178B     00215            bsf     INTCON,GIE  ;enable all interrupts
              00216
              00217 ;****************************************************************************
              00218 ;*
              00219 ;*        Again  Top of main loop
              00220 ;*
              00221 ;*               Waits for 8 timer2 interrupts
              00222 ;*
              00223 ;*               A/D converts VOUT
              00224 ;*
              00225 ;*               Acquires VUNREG channel
              00226 ;*
              00227 ;*        Input Variables:
              00228 ;*               T2POST      Timer2 interrupt counter
              00229 ;*
              00230 ;*        Output Variables:
              00231 ;*               VOUT        Conversion result
              00232 ;*
              00233 ;****************************************************************************
              00234
0036          00235 Again
0036 1DB5     00236            btfss   T2POST,3   ;long delay
0037 2836     00237            goto    Again      ;try again
              00238
0038 01B5     00239            clrf    T2POST     ;clear counter
              00240
```

```
                 00241 ;--- start conversion - feedback
0039 151F        00242      bsf     ADCON0,GO_DONE  ;start conversion
003A 0000        00243      nop
003B 191F        00244 Wc2  btfsc   ADCON0,GO_DONE  ;test if done
003C 283B        00245      goto    Wc2             ;no, wait some more
003D 081E        00246      movf    ADRES,W         ;get conversion result
003E 00A5        00247      movwf   VOUT            ;save result
                 00248 ;--- end conversion
003F 3099        00249      movlw   AVUNREG         ;select VUNREG channel
0040 009F        00250      movwf   ADCON0
                 00251
                 00252
;*****************************************************************************
                 00253 ;*   FErr   Finds difference (error) between SETPOINT and VOUT
                 00254 ;*   E0H:E0 = SETPOINT - VOUT
                 00255 ;*
                 00256 ;*   Input Variables:
                 00257 ;*           SETPOINT    Desired output voltage
                 00258 ;*
                 00259 ;*   Output Variables:
                 00260 ;*           E0H:E0      Signed error
                 00261
;*****************************************************************************
                 00262
0041             00263 FErr
0041 01B2        00264      clrf    E0H             ;clear error high byte
                 00265
0042 0825        00266      movf    VOUT,W
0043 0224        00267      subwf   SETPOINT,W      ;f-w=d
0044 00B1        00268      movwf   E0              ;save new error
                 00269
0045 1C03        00270      btfss   STATUS,C        ;was there a borrow?
0046 09B2        00271      comf    E0H,F           ;yes
                 00272
                 00273
;*****************************************************************************
                 00274 ;*   Ppp    Produces proportional term by multiplying E0H:E0 by Kp
                 00275 ;*
                 00276 ;*          PROH:PRO = E0H:E0 * Kp
                 00277 ;*          Kp = 2^3 - Produced by 3 left shifts
                 00278 ;*
                 00279 ;*          This term forces the output close to the desired output
                 00280 ;*          quickly, but will never completely eliminate the error.
                 00281 ;*
                 00282 ;*   Input Variables:
                 00283 ;*           E0H:E0      Error found at top of loop
                 00284 ;*           Kp          Proportional gain factor (constant)
                 00285 ;*
                 00286 ;*   Output Variables:
                 00287 ;*           PROH:PRO    Proportional component
                 00288 ;*
                 00289
;*****************************************************************************
                 00290
0047             00291 Ppp
0047 0831        00292      movf    E0,W            ;move E0 to temp space
0048 00A7        00293      movwf   TEMPA
0049 0832        00294      movf    E0H,W
004A 00A8        00295      movwf   TEMPB
                 00296
004B 1003        00297      bcf     STATUS,C        ;mult E0 by 2
004C 0DA7        00298      rlf     TEMPA,F
004D 0DA8        00299      rlf     TEMPB,F
                 00300
004E 1003        00301      bcf     STATUS,C        ;mult E0 by 2
004F 0DA7        00302      rlf     TEMPA,F
```

```
0050 0DA8      00303      rlf     TEMPB,F
              00304
0051 1003      00305      bcf     STATUS,C      ;mult E0 by 2
0052 0DA7      00306      rlf     TEMPA,F
0053 0DA8      00307      rlf     TEMPB,F
              00308
0054 0827      00309 PppD movf    TEMPA,W       ;move result in temp space to pro
0055 00AB      00310      movwf   PRO
0056 0828      00311      movf    TEMPB,W
0057 00AC      00312      movwf   PROH
              00313
              00314
;****************************************************************************
              00315 ;*    DifCom Computes differential component
              00316 ;*
              00317 ;*       Finds difference between this loop error and previous
                                loop error
              00318 ;*    DIFH:DIF = (E1H:E1 - E0H:E0) * Kd
              00319 ;*
              00320 ;*       Kd = 2^3 - Produced by 3 left shifts
              00322 ;*       This term tends to slow controller response.
              00323 ;*
              00324 ;*    Input Variables:
              00325 ;*       E1H:E1      Previous loop error
              00326 ;*       E0H:E0      Present loop error
              00327 ;*       Kd          Differential gain factor (constant)
              00328 ;*
              00329 ;*    Output Variables:
              00330 ;*       DIFH:DIF    differential component
              00331 ;*
              00332
;****************************************************************************
              00333
0058          00334 DifCom
0058 0834     00335      movf    E1H,W        ;get prev error high byte
0059 0232     00336      subwf   E0H,W        ;f-w=d   E0-E1=w
005A 00AE     00337      movwf   DIFH         ;save difference high byte
              00338
005B 0833     00339      movf    E1,W         ;get prev error low byte
005C 0231     00340      subwf   E0,W         ;f-w=d   E0-E1=w
005D 00AD     00341      movwf   DIF          ;save difference low byte
              00342
005E 1C03     00343      btfss   STATUS,C     ;was there a borrow?
005F 03AE     00344      decf    DIFH,F       ;yes
              00345
              00346      ;allow difference to be modified here
              00347
0060 1003     00348      bcf     STATUS,C     ;mult dif by 2
0061 0DAD     00349      rlf     DIF,F
0062 0DAE     00350      rlf     DIFH,F
              00351
0063 1003     00352      bcf     STATUS,C     ;mult dif by 2
0064 0DAD     00353      rlf     DIF,F
0065 0DAE     00354      rlf     DIFH,F
              00355
0066 1003     00356      bcf     STATUS,C     ;mult dif by 2
0067 0DAD     00357      rlf     DIF,F
0068 0DAE     00358      rlf     DIFH,F
              00359
              00360
              00361
              00362
;****************************************************************************
              00363 ;*    IntCom Computes integral component
              00364 ;*
              00365 ;*    Multiplies present error by Ki,
```

```
              00366 ;*   adds result to INTH:INT
              00367 ;*
              00368 ;*   INTH:INT = INTH:INT + E0H:E0 * Ki
              00369 ;*
              00370 ;*   Ki = 2^3 -- Produced by 3 left shifts
              00371 ;*
              00372 ;*   This term will eliminate all error,
              00373 ;*   but not quickly
              00374 ;*
              00375 ;*   Input Variables:
              00376 ;*   E0H:E0      Present loop error
              00377 ;*   INTH:INT    Running total of errors
              00378 ;*   Ki          Integral gain factor (constant)
              00379 ;*
              00380 ;*   Output Variables:
              00381 ;*   DIFH:DIF    differential component
              00382
;****************************************************************************
              00383
0069          00384 IntCom
0069 0832     00385      movf   E0H,W        ;move E0 to temp space
006A 00A8     00386      movwf  TEMPB
006B 0831     00387      movf   E0,W
006C 00A7     00388      movwf  TEMPA
              00389
              00390      ;allow error to be modified here before adding to integral
              00391
006D 1003     00392      bcf    STATUS,C
006E 0DA7     00393      rlf    TEMPA,F      ;E0
006F 0DA8     00394      rlf    TEMPB,F      ;E0H
              00395
0070 1003     00396      bcf    STATUS,C
0071 0DA7     00397      rlf    TEMPA,F      ;E0
0072 0DA8     00398      rlf    TEMPB,F      ;E0H
              00399
0073 1003     00400      bcf    STATUS,C
0074 0DA7     00401      rlf    TEMPA,F      ;E0
0075 0DA8     00402      rlf    TEMPB,F      ;E0H
              00403
0076 0827     00404 IntD movf   TEMPA,W      ;get current error, E0
0077 07A9     00405      addwf  INT,F        ;add to INT, store in INT
0078 1803     00406      btfsc  STATUS,C     ;was there a carry?
0079 0AAA     00407      incf   INTH,F       ;yes, inc INT high byte
              00408
007A 0828     00409      movf   TEMPB,W      ;get E0 high byte, E0H
007B 07AA     00410      addwf  INTH,F
              00411
              00412
;****************************************************************************
              00413 ;*   Total  Sums proportional, integral, and differential terms
              00414 ;*
              00415 ;*       PWMH:PWM = INTH:INT + PROH:PRO + DIFH:DIF
              00416 ;*
              00417 ;*       If the result should ever go negative,
              00418 ;*       the result is forced to 0,and the overload LED is set.
              00419 ;*       (This is an error condition.Can't have a negative PWM.)
              00420 ;*
              00421 ;*   Input Variables:
              00422 ;*       INTH:INT    Integral term
              00423 ;*       PROH:PRO    Proportional term
              00424 ;*       DIFH:DIF    Differential term
              00425 ;*
              00426 ;*   Output Variables:
              00427 ;*       PWMH:PWM    Sum of terms
              00428
;****************************************************************************
```

```
                00429
007C            00430 Total
007C 082C       00431 PCom    movf   PROH,W      ;add in proportional term
007D 00B0       00432         movwf  PWMH
                00433
007E 082B       00434         movf   PRO,W
007F 00AF       00435         movwf  PWM
                00436
0080 082A       00437 ICom    movf   INTH,W      ;add in integral term
0081 07B0       00438         addwf  PWMH,F
                00439
0082 0829       00440         movf   INT,W
0083 07AF       00441         addwf  PWM,F
0084 1803       00442         btfsc  STATUS,C
0085 0AB0       00443         incf   PWMH,F
                00444
0086 082E       00445 DCom    movf   DIFH,W      ;add in differential term
0087 07B0       00446         addwf  PWMH,F
                00447
0088 082D       00448         movf   DIF,W
0089 07AF       00449         addwf  PWM,F
008A 1803       00450         btfsc  STATUS,C
008B 0AB0       00451         incf   PWMH,F
                00452
008C 1BB0       00453 Ovrld   btfsc  PWMH,7      ;did PWM go negative?
008D 2890       00454         goto   NegPwm      ;yes
008E 1306       00455         bcf    PORTB,6     ;no – turn off overload LED
008F 2893       00456         goto   PwmGen
                00457
0090 1706       00458 NegPwm  bsf    PORTB,6     ;turn on overload LED
0091 01B0       00459         clrf   PWMH        ;set PWM to 0
0092 01AF       00460         clrf   PWM
                00461
                00462
;****************************************************************************
                00463 ;*       PwmGen  Divides PWHM:PWM by 8 (3 right shifts)
                00464 ;*              2 LSbits of PWM sent to 2 LSbits of duty cycle register
                00465 ;*              remaining 6 bits sent to CCPR1L (duty cycle register)
                00466 ;*
                00467 ;*              A/D has been acquiring VUNREG, start conversion.
                00468 ;*
                00469 ;*       Input Variables:
                00470 ;*              PWMH:PWM   PWM drive
                00471
;****************************************************************************
                00472
009             00473 PwmGen
0093 0CB0       00474         rrf    PWMH,F      ;PWMH
0094 0CAF       00475         rrf    PWM,F       ;PWM
                00476
0095 0CB0       00477         rrf    PWMH,F      ;PWMH
0096 0CAF       00478         rrf    PWM,F       ;PWM
                00479
0097 0CB0       00480         rrf    PWMH,F      ;PWMH – can ignore contents of PWMH now
0098 0CAF       00481         rrf    PWM,F       ;PWM
                00482
0099 1217       00483         bcf    CCP1CON,4   ;clear ls bits of PWM duty cycle
009A 1297       00484         bcf    CCP1CON,5
                00485
009B 0CAF       00486         rrf    PWM,F       ;shift carry INTo PWM, lsbit INTo carry
009C 1803       00487         btfsc  STATUS,C    ;is carry set?
009D 1617       00488         bsf    CCP1CON,4   ;set PWM duty cycle lsb
                00489
009E 0CAF       00490         rrf    PWM,F       ;shift carry INTo PWM, lsbit INTo carry
009F 1803       00491         btfsc  STATUS,C    ;is carry set?
00A0 1697       00492         bsf    CCP1CON,5   ;set PWM duty cycle lsb
```

```
               00493
00A1 082F      00494        movf   PWM,W        ;get PWM
00A2 393F      00495        andlw  0x3f         ;mask off 2 ms bits (78.125KHz)
00A3 0095      00496        movwf  CCPR1L       ;put in PWM duty cycle
               00497
00A4 151F      00498        bsf    ADCON0,2     ;start conversion VUNREG
               00499
               00500
;*****************************************************************************
               00501 ;*   up/dn buttons
               00502 ;*          Debounces UP and DOWN buttons
               00503 ;*              Increments counters once for each pass
               00504 ;*              through the loop when button pressed.
               00505 ;*
               00506 ;*              If button not pressed, counter is reset.
               00507 ;*
               00508 ;*              If a button is successfully debounced,
               00509 ;*              both counters are reset.
               00510 ;*
               00511 ;*              Debounce delay is about 0.5 seconds
               00512 ;*
               00513 ;*          Moves voltage indicator bit as required.
               00514 ;*
               00515 ;*          Finds index into lookup table, and calls table.
               00516 ;*
               00517 ;*          Saves result from lookup table as new SETPOINT
               00518 ;*
               00519 ;*   Input Variables:
               00520 ;*          PORTB         Current indicator data
               00521 ;*          DNCH:DNCL     Down button debounce counter
               00522 ;*          UPCH:UPCL     Up button debounce counter
               00523 ;*
               00524 ;*   Output Variables:
               00525 ;*          PORTB         Current indicator data
               00526 ;*          DNCH:DNCL     Down button debounce counter
               00527 ;*          UPCH:UPCL     Up button debounce counter
               00528 ;*          SETPOINT      New voltage setpoint
               00529
;*****************************************************************************
               00530
00A5 0806      00531        movf   PORTB,W      ;move LED data to temp space
00A6 393F      00532        andlw  0x3f         ;mask off non-voltage LEDs
00A7 00A7      00533        movwf  TEMPA
               00534
00A8 1E85      00535 Dnb    btfss  PORTA,5      ;down
00A9 28B3      00536        goto   Upb          ;down not pushed
00AA 0FA2      00537        incfsz DNCL,f       ;down is pushed, inc debounce
00AB 28CE      00538        goto   Wc3          ;no carry - go to next module
00AC 0FA3      00539        incfsz DNCH,f       ;inc debounce counter high byte
00AD 28CE      00540        goto   Wc3          ;no carry - go to next module
               00541                            ;--------------------- select next lower LED
00AE 1003      00542        bcf    STATUS,C     ;select next lower LED
00AF 0CA7      00543        rrf    TEMPA,F      ;shift LED data down 1 voltage
00B0 1803      00544        btfsc  STATUS,C     ;3V LED was set before rotate, so
00B1 1427      00545        bsf    TEMPA,0      ;set it again
               00546
00B2 28BE      00547        goto   Dunb
               00548
00B3 1D05      00549 Upb    btfss  PORTA,2      ;up button
00B4 28C9      00550        goto   Nob          ;up not pushed - no buttons pushed
00B5 0FA0      00551        incfsz UPCL,f       ;down is pushed, inc debounce
00B6 28CE      00552        goto   Wc3          ;no carry - go to next module
00B7 0FA1      00553        incfsz UPCH,F       ;inc debounce counter high byte
00B8 28CE      00554        goto   Wc3          ;no carry - go to next module
               00555                            ;------------------- select next higher LED
00B9 1003      00556        bcf    STATUS,C     ;select next higher voltage LED
```

```
00BA 0DA7      00557       rlf     TEMPA,F       ;shift LED data up 1 voltage
00BB 1B27      00558       btfsc   TEMPA,6       ;if 9V LED was set before,
00BC 16A7      00559       bsf     TEMPA,5       ;set it again, and
00BD 1327      00560       bcf     TEMPA,6       ;clear the overload LED
               00561
00BE 0827      00562 Dunb  movf    TEMPA,W       ;move LED data back to PORTB
00BF 0086      00563       movwf   PORTB
               00564
00C0 01A8      00565       clrf    TEMPB
00C1 03A8      00566       decf    TEMPB,F       ;set TEMPB to -1
               00567
00C2 0AA8      00568 NewSetincf  TEMPB,F       ;count up
00C3 0CA7      00569       rrf     TEMPA,F       ;rotate least sig bit INTo carry
00C4 1C03      00570       btfss   STATUS,C      ;is carry set now?
00C5 28C2      00571       goto    NewSet        ;no, try again
               00572
00C6 0828      00573       movf    TEMPB,W       ;yes, put count in w
00C7 20DD      00574       call    Tbl           ;get corresponding value for PWM
00C8 00A4      00575       movwf   SETPOINT      ;put value in SETPOINT
               00576
00C9 01A2      00577 Nob   clrf    DNCL          ;clear down button debounce counter
00CA 01A0      00578       clrf    UPCL          ;clear up button debounce counter
00CB 30F9      00579       movlw   DEL1
00CC 00A3      00580       movwf   DNCH          ;preset down button debounce counter
00CD 00A1      00581       movwf   UPCH          ;preset up button debounce counter high byte
               00582
               00583
;****************************************************************************
               00584 ;*   Wc3     VUNREG has been converted by now,
               00585 ;*           Get result, save in VUNREG.
               00586 ;*
               00587 ;*           Select VOUT to aquire.
               00588 ;*
               00589 ;*           Test VUNREG to see if it has dropped too low.
               00590 ;*           If too low, call protective shut-down.
               00591 ;*
               00592 ;*   Input Variables:
               00593 ;*           VUNREG       Input voltage.
               00594
;****************************************************************************
               00595
00CE           00596 Wc3
00CE 191F      00597       btfsc   ADCON0,GO_DONE ;test if done
00CF 28CE      00598       goto    Wc3            ;no, wait some more
00D0 081E      00599       movf    ADRES,W        ;get conversion result
00D1 00A6      00600       movwf   VUNREG         ;save result
               00601
00D2 3089      00602       movlw   AVOUT          ;select feedback channel to aquire
00D3 009F      00603       movwf   ADCON0
               00604
00D4 0826      00605       movf    VUNREG,W       ;get UNREG
00D5 3C50      00606       sublw   0x50           ;10V-VUNREG=?  C=1 if VUNREG<10V
00D6 1803      00607       btfsc   STATUS,C
00D7 28E4      00608       goto    ShutDn         ;turn off regulator (dies here)
               00609
               00610
               00611
;****************************************************************************
               00612 ;*   Shift   shift errors - save current error as old
               00613 ;*           E1H:E1 = E0H:E0
               00614 ;*
               00615 ;*           Go back to top of loop.
               00616 ;*
               00617 ;*   Input Variables:
               00618 ;*           E0H:E0        Present error
               00619 ;*
```

```
          00620 ;*   Output Variables:
          00621 ;*           E1H:E1          Previous error
          00622
;*****************************************************************************
          00623
00D8      00624 Shift
00D8 0831 00625       movf    E0,W
00D9 00B3 00626       movwf   E1
          00627
00DA 0832 00628       movf    E0H,W
00DB 00B4 00629       movwf   E1H
          00630
00DC 2836 00631       goto    Again
          00632
          00633
;*****************************************************************************
          00634 ;*   Tbl     Look up table.
          00635 ;*
          00636 ;*           Called with an index in W.
          00637 ;*
          00638 ;*           Index is added to program counter.
          00639 ;*
          00640 ;*           Execution jumps to "retlw" which will return
          00641 ;*           to the calling program with table value in w.
          00642 ;*
          00643 ;*   Input Variables:
          00644 ;*           w               Offset index into table
          00645 ;*
          00646 ;*   Output Variables:
          00647 ;*           w               Value from table
          00648
;*****************************************************************************
          00649
00DD      00650 Tbl                          ;call with index in w
00DD 0782 00651       addwf   PCL,F          ;add index to PC
          00652
00DE 344D 3474 3482 00653         dt        0x4d, 0x74, 0x82, 0x9b, 0xc2, 0xea
     349B 34C2 34EA
          00654 ;output voltage              3.0   4.5   5.0   6.0   7.5   9.0
          00655
          00656
;*****************************************************************************
          00657 ;*   ShutDn  Protective shutdown. Entry into this routine is a result
          00658 ;*   of a low input voltage. This turns off the converter
          00659 ;*   before the series pass transistor can overheat.
          00660 ;*
          00661 ;*           PWM on-time is set to 0, turning off the converter.
          00662 ;*
          00663 ;*           The converter trip LED is lit, indicating shutdown.
          00664 ;*
          00665 ;*           Execution then loops without exit. The only exit is to
          00666 ;*           reset the controller.
          00667
;*****************************************************************************
          00668
00E4      00669 ShutDn
00E4 1217 00670       bcf     CCP1CON,4      ;turn off PWM
00E5 1297 00671       bcf     CCP1CON,5
00E6 0195 00672       clrf    CCPR1L
          00673
00E7 1786 00674       bsf     PORTB,7        ;light trip LED
          00675
00E8      00676 Dead
00E8 28E8 00677       goto    Dead           ;stays here
          00678
          00679
          00680             END
```

```
MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : X---XXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXX------- ----------------
2000 : -------X-------- ---------------- ---------------- ----------------

All other memory blocks unused.

Program Memory Words Used:   230
Program Memory Words Free:  1818


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     7 suppressed
```

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable".
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Rocky Mountain
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

Atlanta
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo
2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia
Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing
Microchip Technology Consulting (Shanghai)
Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu
Microchip Technology Consulting (Shanghai)
Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

China - Fuzhou
Microchip Technology Consulting (Shanghai)
Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Shanghai
Microchip Technology Consulting (Shanghai)
Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen
Microchip Technology Consulting (Shanghai)
Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

Hong Kong
Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan
Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark
Microchip Technology Nordic ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

01/18/02