APPLICATION NOTE

A DC MOTOR CONTROLLER USING THE ZILOG Z86E06 MCU

THIS DC MOTOR CONTROLLER MORE THAN MEETS THE CHALLENGES OF SMALLER PACKAGING AND HIGHER RELIABILITY.

INTRODUCTION

Many applications such as printers, ATM machines, robotics, CD players, and disk drives require DC motor control. Additionally, the demand for smaller packaging and higher reliability creates a need for more integrated motor driver solutions. In the past, a DC motor controller comprised a microcontroller (MCU) and many discrete components. Presently, advanced IC technology reduces the drive electronics to just two chips: an MCU and an integrated motor driver. This application note will present a DC motor controller, which uses the Zilog Z86E06 MCU and the National LMD18200 DC Motor Controller IC, that is both low cost and features a low parts count.

MOTOR CONTROL BASICS

Several DC motor design topologies exist, but certainly the most widely used method is the "H-bridge" configuration. This method uses four Bi-polar Junction Transistor (BJT) or Metal-Oxide Silicon Field Effect Transistor (MOSFET) devices configured in an "H" pattern (see Figure 1). In the center of the "H" is the motor itself. To drive the motor in the forward direction, current flows through Q1 and Q4. To turn the motor in the reverse direction, Q1 and Q4 are turned off, and Q2 and Q3 are energized. External logic is needed to gate the devices. Motor speed is controlled by the average current flowing in the legs. This is regulated by a Pulse-Width Modulation (PWM) drive method, which relies on the duty cycle of a digital output to control the drive voltage to the MOSFETs (see Figure 2). Conventional designs require low-pass filtration to produce a constant DC voltage. Varying the duty cycle of the output varies the DC voltage. This DC voltage would then drive the power MOSFETs. The necessary steering logic is incorporated inside the LMD18200. The PWM is also accepted without the need for an external low-pass filter.

Since the voltage drop across the collector and emitter can reach more than 1V during saturation, high heat dissipation is encountered using BJT devices. MOSFETs, with their intrinsically low Rds (drain to source turn-on resistance), are better suited for motor driver applications. Typically, power MOSFETs need a gate voltage of least 8V to turn on, which is a problem when using an MCU whose outputs swing from 0–5V. Special logic MOSFETs have been developed that have gate turn-on voltages of 5V. This works fine for the lower legs of the "H" motor drive, but what about the upper legs? Unfortunately, the upper legs need a higher gate voltage, due to the fact that the motor's winding resistance raises the MOSFETs source reference above ground potential. A separate DC-DC convertor chip can be used, but this adds more cost and complexity to the design. The LMD18200 solves this problem by having a built-in DC-DC convertor.

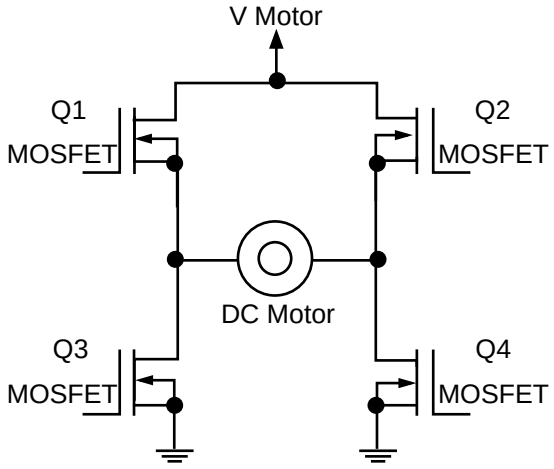

Figure 1. H-Pattern Motor Driver Configuration

HARDWARE DESCRIPTION

The heart of the motor controller is the Zilog Z86E06, an 18-pin, One-Time Programmable (OTP) MCU, which contains 1KB of ROM, 128 bytes of RAM, 14 I/O lines, two timer/counters, and two on-board comparators. A block diagram of the chip is shown in Figure 2. The Z8® MCU is clocked by a 4 MHz ceramic resonator, and the motor is controlled by the National LMD18200. This is a 3A H-Bridge driver IC with direction and braking logic, along with thermal and current sensing of the output drivers. The functional diagram of the LMD18200 is shown in Figure 3.

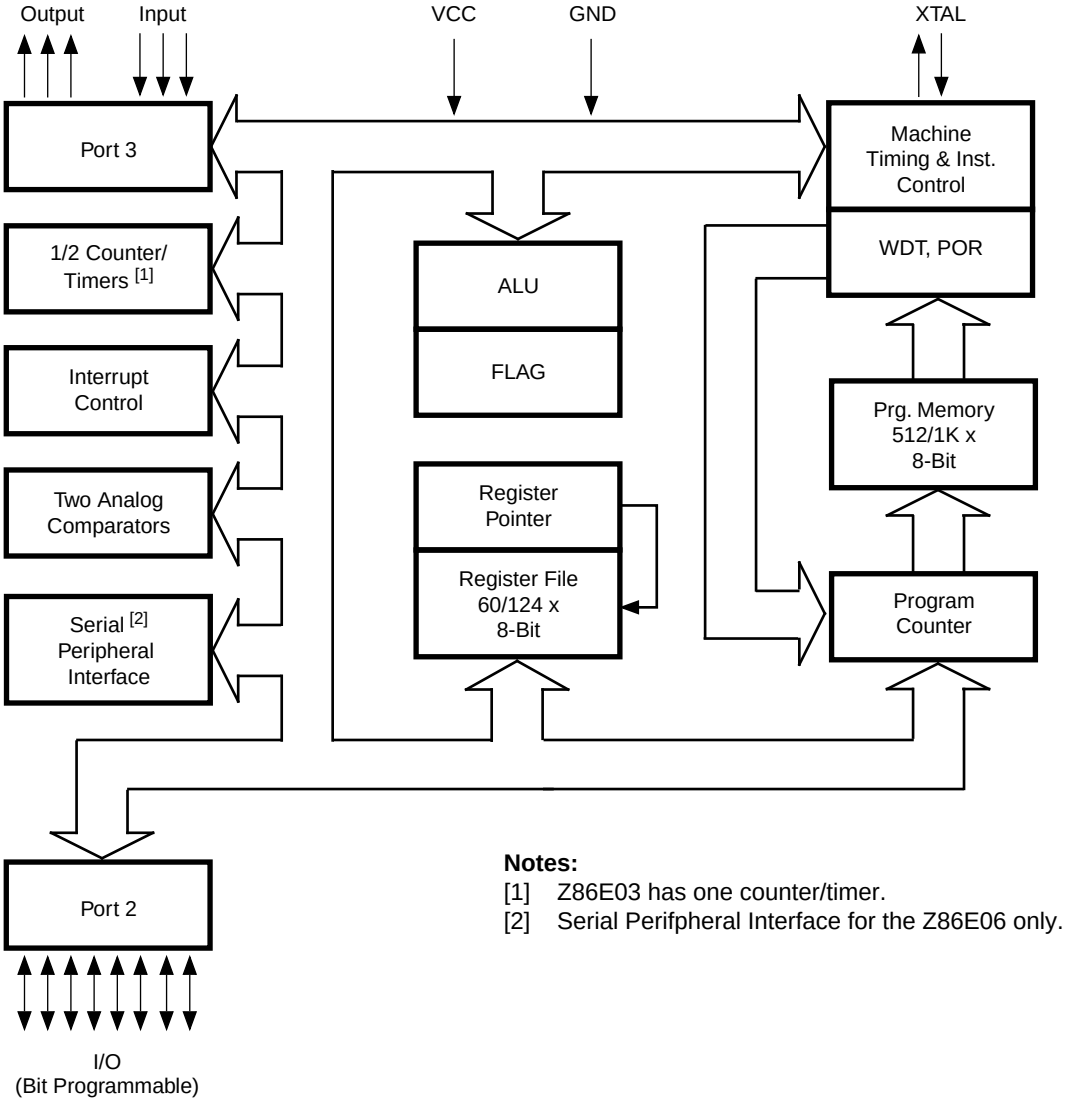

Figure 2. Z86E03/E06 Functional Block Diagram

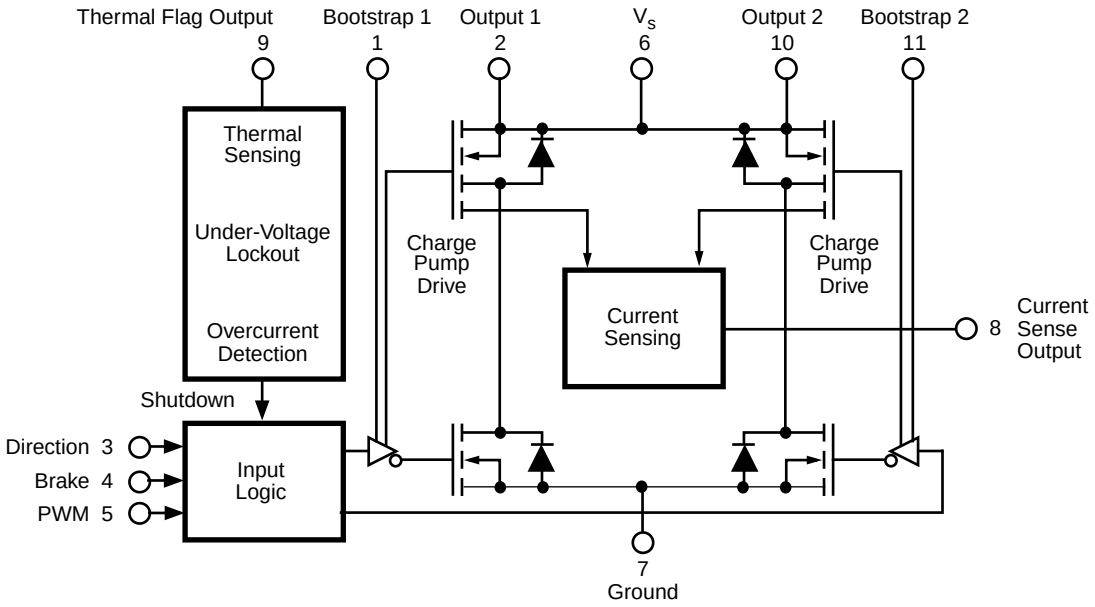

Figure 3. LMD18200 Functional Diagram

Referring to the Z86E03/E06 schematic diagram (see Figure 4), output port pins P34, P35, and P36 of the Z86E06 provide direction, brake, and PWM signals, respectively, to the LMD18200. The current sensing is done with a resistor to ground from pin 8 of the LMD18200. The current output from this pin is typically 377 µA/per A. If the motor is rated at 1A, a resistor value is chosen so that the voltage developed across the resistor does not exceed 4V, which is the maximum input voltage to the comparator on-board the Z86E06. This calculates out to be 10.6K ohms (4V/377µA). For this application, the motor was rated at 12V @ 0.6A. A resistor value of 2.7K ohms was chosen to give a maximum voltage across the resistor of approximately 2.5V at all speeds. This voltage is sensed by one of the Z8® MCU's on-board comparators. The reference pin for the comparators (P33) is biased at 3.3V through a voltage divider.

Any significant load on the motor will increase the motor current, thereby increasing the voltage across the current sense resistor. If the voltage across the sense resistor exceeds 3.3V, a comparator interrupt will be generated, and the BRAKE line to the LMD18200 will be activated, stopping the motor. When the load condition of the motor is removed, pressing the STOP push button reactivates the motor.

The temperature flag pin (9) of the LMD18200 is connected to P32 of the Z8 MCU. This is an open-collector output, therefore it is pulled up to +5V with a 10K ohm resistor. The function of this pin is to interrupt the processor when the junction temperature of the LMD18200 exceeds 145 degrees C. This is an active-low output, which will trip the Z8 MCU comparator at P32. A push button connected to P20-P22 of the Z8 MCU provides the speed and brake control.

The SPDT toggle switch connected to P23 provides the direction (high is forward, low is reverse). Direction changing can only be done in a STOP condition. Status LEDs show the direction and stop conditions. The STOP LED (red) is connected to P24. The FORWARD (yellow), and REVERSE (green) LEDs are connected to P25 and P26, respectively.

The motor terminals are connected between the OUT1 and OUT2 terminals of the LMD18200. The 0.1 µF ceramic capacitors are connected between the bootstrap pins (1,11) and motor output pins (2,10). The Vs pin is connected to +12V. This line is bypassed with a 220 µF and 0.1 µF capacitors.

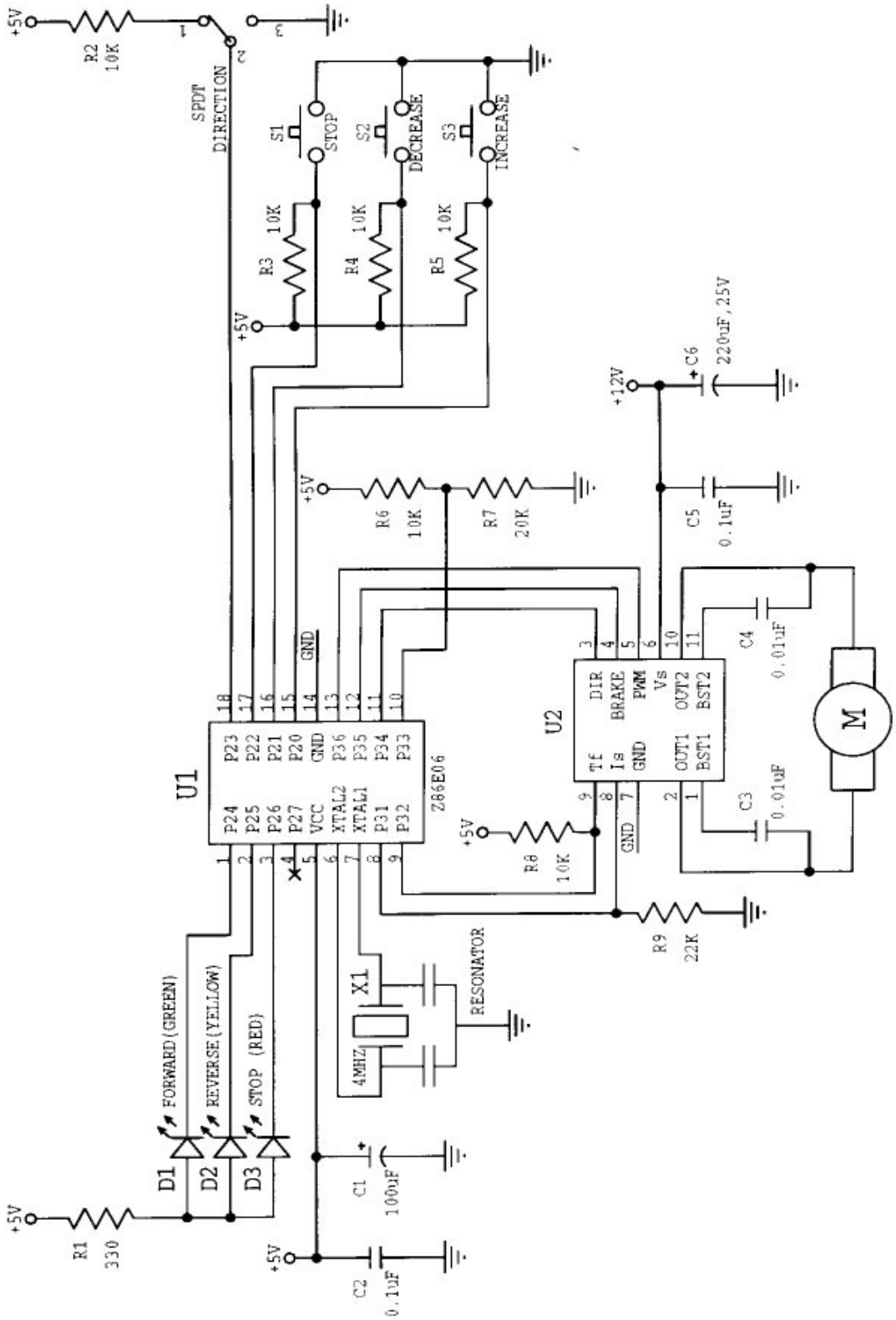

Figure 4. DC Motor Controller Schematic

SOFTWARE DESCRIPTION

The motor controller software flowchart is shown in Figure 5. After initialization, the controller waits for a timer T0 interrupt. The timer interval is set for about 500 µs. This was chosen to provide a 2000 Hz. sample rate for the PWM.

The SAMPLE interrupt service routine essentially handles two functions:

- The PWM Output
- Sensing and Debouncing of the Push Button

The comparator interrupts sense overload and over-temperature conditions. (Refer to Code Listing at the conclusion of this application note.)

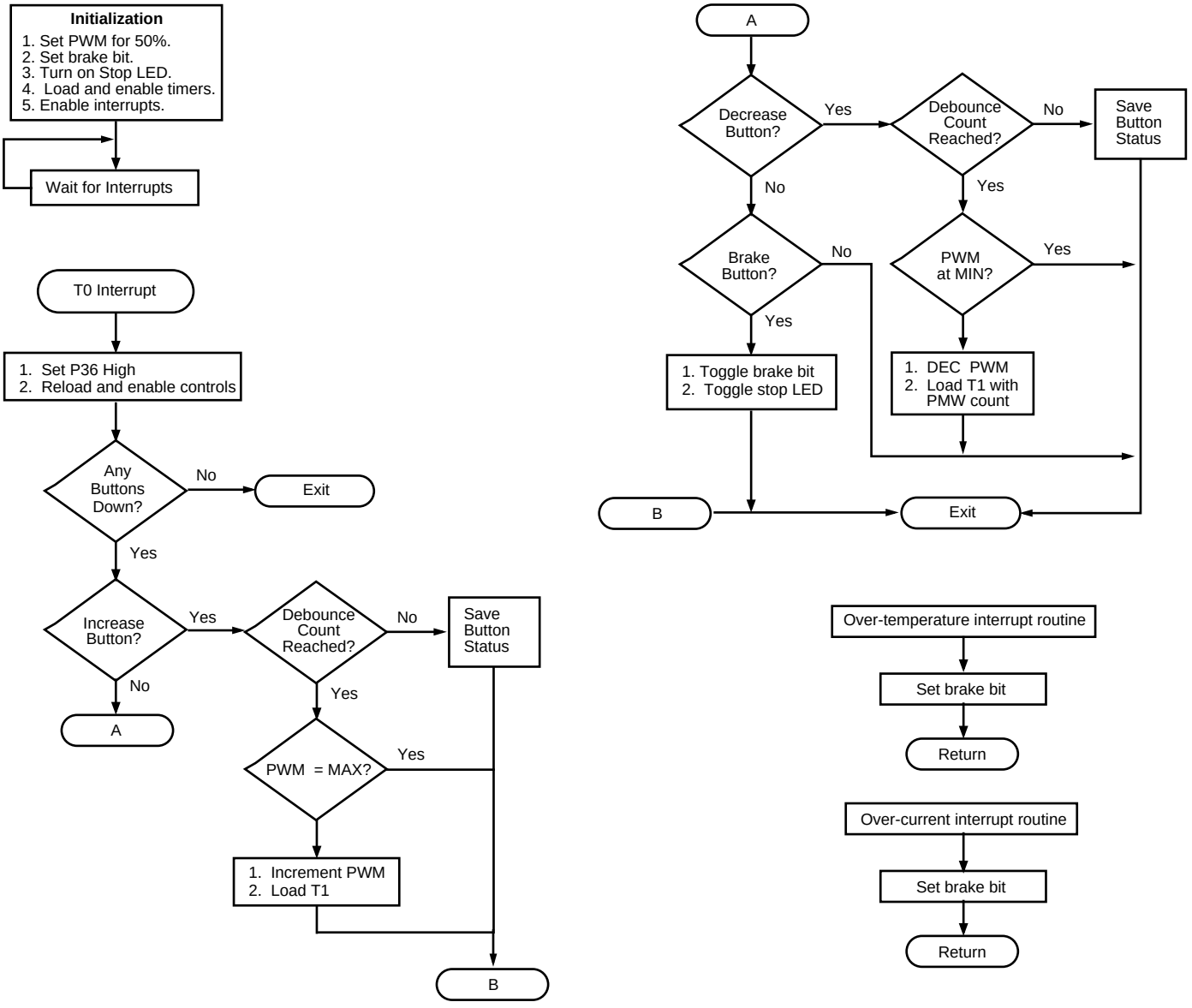

Figure 5. DC Motor Flowchart

Refer to Figure 6 for PWM Waveforms. The initial PWM is set for a 50-percent duty cycle. The initial state is STOP, as indicated by the red LED. The motor may be taken out of STOP by momentarily pressing the STOP button. The motor will then turn at a speed determined by the PWM value and either clockwise or counterclockwise, as determined by the DIRECTION switch. The duty cycle of the PWM is controlled by timer T1. This is loaded with a value contained in register PWM. The timer is configured to count down, then stop when it reaches zero. When it hits terminal count, it toggles port pin P36 from high to low. The timer is loaded with its initial value, and P36 is taken high on the next pass of the T0 interrupt interval. The timer T0 sets the sampling rate, in this case 2000 Hz.

Pressing the INCREASE push button increases the duty cycle of the PWM, while the DECREASE push button decreases the duty cycle of the PWM. The maximum limits are 240/256 (93 percent) and 32/256 (12.5 percent), respectively. Pressing the STOP button toggles the BRAKE output to the LMD18200, and either brakes or enables the motor, depending on the present state.

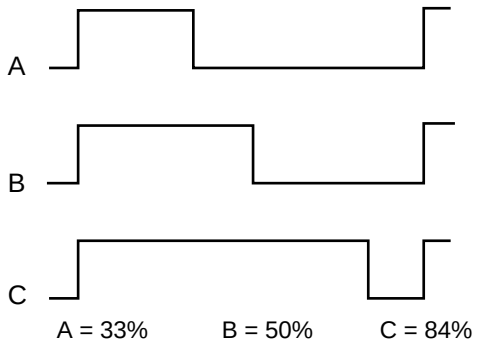

Figure 6. PWM Waveforms Duty Cycle

The temperature flag input to the Z8® MCU senses the condition of this signal from the LMD18200. This pin goes active low when the junction temperature of the motor controller reaches 145 degrees C. The assertion of this pin (active low) causes the compartor to trip at P32, interrupting the processor. The OVER_TEMP interrupt service routine sets the BRAKE output to high, disabling the motor. This condition will exist until the THERMAL FLAG bit goes high again.

REMOTE CONTROL

Instead of push button switches to control the functions and speed of the motor, an I^2C or asynchronous communications protocol can be added for remote control.

REFERENCES

1. Chuck McManis, “A PIC-based Motor Speed Controller,” *Circuit Cellar Ink* , July, 1995.
2. Jeff Bachiacchi, “Creating The Smart-MD,” *Circuit Cellar Ink*, September–October,1995.
3. National Semiconductor, LMD18200 datasheet.
4. Zilog, *Zilog Discrete Z8 Microcontrollers Product Specifications Databook*, DC 8318-02.
5. Zilog, Zilog Z8 Microcontroller User’s Manual, UM95Z800103.

.

CODE LISTING

```
asmS8 version 2.1
Tue Feb 27 02:59:25 1996        b:\testdnm
LOC      OBJ              LINE# --- SOURCE ---
                              1 ;--------------------------------------------------------------------------
                              2 ;      This program is intended to control a DC motor with a Zilog
                              3 ;      microcontroller. The Z8 produces a PWM (Pulse-Width Modulation) signal
                              4 ;      that drives a National LMD18200 H-bridge motor driver. This allows
                              5 ;      full directional and speed control of the motor. The speed,
                              6 ;      direction, and stop functions are controlled by pushbutton switches.
                              7 ;      A status LED indicates Forward (Green), Reverse (Yellow), and Stop
                              8 ;      (Red) conditions. A voltage feedback from the H-bridge allows the
                              9 ;      microcontroller to monitor the current through the motor. This is
                             10 ;      implemented with a series resistor from the Is pin of the H-bridge,
                             11 ;      and the voltage developed across the resistor is proportional to the
                             12 ;      current through the motor. This voltage is fed back to one of the Z8's
                             13 ;      on-chip analog comparators.
                             14 ;
                             15 ;
                             16 ;
                             17 ;      Z86E06 pin assignments
                             18 ;      ---------------------
                             19 ;                1             18
                             20 ;                 ---------------
                             21 ;FORWARD LED  - | P24        P23 | -  DIR SWITCH
                             22 ;               |                |
                             23 ;REVERSE LED  - | P25        P22 | -  STOP PUSHBUTTON
                             24 ;               |                |
                             25 ;   STOP LED  - | P26        P21 | -  SPEED DECREASE PUSHBUTTON
                             26 ;               |                |
                             27 ;        N/C  - | P27        P20 | -  SPEED INCREASE PUSHBUTTON
                             28 ;               |                |
                             29 ;      +5V   -  | Vcc        GND | -  GND
                             30 ;               |                |
                             31 ;      OSC   -  | Xtal1      P36 | -  PWM OUT
                             32 ;               |                |
                             33 ;      OSC   -  | Xtal2      P35 | -  STOP
                             34 ;               |                |
                             35 ;      Vref  -  | P33        P34 | -  DIRECTION
                             36 ;               |                |
                             37 ;      Tf    -  | P32        P31 | -  Is
                             38 ;                 ---------------
                             39 ;                9               10
                             40 ;
                             41 ;--------------------------------------------------------------------------
                             42
       abs 00000004          43 bounce           .equ   r4
       abs 00000005          44 count            .equ   r5
       abs 00000006          45 key_cnt          .equ   r6
       abs 00000007          46 key_temp         .equ   r7
       abs 00000008          47 temp_1           .equ   r8
       abs 00000009          48 pwm              .equ   r9
       abs 0000000a          49 make             .equ   r10
       abs 0000000b          50 STATE            .EQU   R11
       abs 0000000c          51 temp_led         .equ   r12
       abs 0000000e          52 delay_hi         .equ   r14
       abs 0000000f          53 delay_lo         .equ   r15
       abs 0000000e          54 delay            .equ   rr14
                             55
     0000000000000000001     56 increase         .equ   01h
     0000000000000000002     57 decrease         .equ   02h
     0000000000000000003     58 brake            .equ   03h
     0000000000000000008     59 dir_sw           .equ   08h
     0000000000000000007     60 switches         .equ   07h
     0000000000000000020     61 min              .equ   20h
     00000000000000000e0     62 max              .equ   0e0h
     0000000000000000001     63 irq0             .equ   01h
     0000000000000000004     64 irq2             .equ   04h
     0000000000000000010     65 irq4             .equ   10h
     0000000000000000020     66 brakes_on        .equ   20h
     0000000000000000040     67 stop_led         .equ   40h
```

```
     0000000000000000020    68 reverse_led     .equ   20h
     0000000000000000010    69 forward_led     .equ   10h
     0000000000000000000    70 stopped         .equ   00h
     0000000000000000001    71 start_up        .equ   01h
     0000000000000000002    72 running         .equ   02h
                            73 ;---------------------------------------------------------------------------
                            74 ;               INITIALIZATION
                            75 ;---------------------------------------------------------------------------
00000000                    76                 .org   0000h
                            77
00000000 Wwww               78                 .word  over_temp
00000002 Wwww               79                 .word  no_irq
00000004 Wwww               80                 .word  over_current
00000006 Wwww               81                 .word  no_irq
00000008 Wwww               82                 .word  sample
0000000a Wwww               83                 .word  no_irq
                            84
0000000c                    85                 .org   000ch
                            86
0000000c 8f                 87                 di                          ; disable int
0000000d 3100               88                 srp    #0                   ; lowest bank
0000000f e6f60f             89                 ld     p2m,#0fh             ; inputs on p20-p23, outputs on p24-p27
00000012 e602bf             90                 ld     p2,^C #stop_led      ; load with initial values
00000015 e6f702             91                 ld     p3m,#2               ; open drain on P2, comparators on
00000018 e6f804             92                 ld     p01m,#04h            ; int stack
0000001b e6ff80             93                 ld     spl,#80h             ; stack at highest ram location
0000001e b0fe               94                 clr    sph                  ; clear stack pointer high byte
00000020 b0fa               95                 clr    irq                  ; clear  int request reg
00000022 e6f504             96                 ld     pre0,#04h            ; load prescaler 0 with /1, one-shot
00000025 b0f4               97                 clr    t0                   ; set timer TC for period of 0.5 mS
00000027 e6fb10             98                 ld     imr,#irq4            ; set interrupt levels
0000002a e6f306             99                 ld     pre1,#06h            ; one shot
0000002d e60360            100                 ld     p3,#60h              ; brakes on, pwm high
00000030 bc00              101                 ld     STATE,#stopped       ;
00000032 b0f9              102                 clr    ipr                  ;
00000034 b0ee              103                 clr    delay_hi             ;
00000036 b0ef              104                 clr    delay_lo             ;
00000038 9c80              105                 ld     pwm,#80h             ; start with pwm = 50%
0000003a 99f2              106                 ld     t1,pwm               ; load timer with pwm value
0000003c e6f18f            107                 ld     tmr,#8fh             ; load and enable t0,t1
0000003f e6fa80            108                 ld     irq,#80h             ; rising on irq2, falling on irq0
00000042 9f                109 delay_loop:     ei                          ; enable interrupts
00000043 8bfd              110                 jr     delay_loop           ; wait for interrupts
                           111 ;---------------------------------------------------------------------------
                           112 ;               OVER-TEMPERATURE INTERRUPT ROUTINE
                           113 ;---------------------------------------------------------------------------
00000045 460320            114 over_temp:      or     p3,#brakes_on        ; set BRAKE line high
00000048 bc00              115                 ld     STATE,#stopped       ;
0000004a e602bf            116                 ld     p2,^C #stop_led      ;
0000004d e6fb10            117                 ld     imr,#irq4            ;
00000050 bf                118                 iret                        ; return from interrupt
                           119 ;---------------------------------------------------------------------------
                           120 ;               OVER-CURRENT INTERRUPT ROUTINE
                           121 ;---------------------------------------------------------------------------
00000051 460320            122 over_current:   or     p3,#brakes_on        ; set BRAKE line high
00000054 bc00              123                 ld     STATE,#stopped       ;
00000056 e602bf            124                 ld     p2,^C #stop_led      ;
00000059 e6fb10            125                 ld     imr,#irq4            ;
0000005c bf                126                 iret                        ;
                           127 ;---------------------------------------------------------------------------
                           128 ;               TIMER 0 INTERRUPT ROUTINE
                           129 ;
                           130 ;               This routine performs the following functions:
                           131 ;
                           132 ;               1) Sets sample rate for PWM at 2000 Hz
                           133 ;               2) Tests for key closures and checks direction switch
                           134 ;---------------------------------------------------------------------------
0000005d a6eb00            135 sample:         cp     STATE,#stopped       ; check if stopped
00000060 eb**              136                 jr     ne,test_start_up;
00000062 760208            137                 tm     p2,#dir_sw           ; test direction switch
00000065 6b**              138                 jr     z,ccw                ; if low, then reverse
00000067 460310            139                 or     p3,#10h              ; if high, then forward
```

```
0000006a ccef              140                 ld     temp_led,^C #forward_led
0000006c 8b**              141                 jr     continue                ;
0000006e 5603ef            142 ccw:            and    p3,^C #10h              ; take direction bit low
00000071 ccdf              143                 ld     temp_led,^C #reverse_led
00000073 a6eb01            144 test_start_up:  cp     STATE,#start_up         ;
00000076 eb**              145                 jr     ne,continue             ;
00000078 80ee              146                 decw   delay                   ;
0000007a eb**              147                 jr     nz,continue             ;
0000007c e6fb15            148                 ld     imr,#irq4+irq2+irq0;
0000007f bc02              149                 ld     STATE,#running          ;
00000081 460340            150 continue:       or     p3,#40h                 ; take pwm high
00000084 e6f18f            151                 ld     tmr,#8fh                ; load and enable timer
00000087 660207            152 key_scan:       tcm    p2,#switches            ; any switches pressed?
0000008a 6b**              153                 jr     z,exit                  ; no, then exit
0000008c 8802              154                 ld     temp_1,p2               ; get switch data
0000008e 5c03              155                 ld     count,#3                ; load counter
00000090 6e                156 key_loop:       inc    key_cnt                 ; inc key count
00000091 df                157                 scf                            ; set carry flag
00000092 c0e8              158                 rrc    temp_1                  ; rotate right
00000094 7b**              159                 jr     c,no_keys               ; any carry?
00000096 a276              160                 cp     key_temp,key_cnt        ; same key?
00000098 eb**              161                 jr     ne,load_keys            ; not the same
0000009a 4e                162                 inc    bounce                  ; increment bounce counter
0000009b a6e480            163                 cp     bounce,#80h             ; bounce = 127 ?
0000009e 7b**              164                 jr     ult,load_keys           ; no action if less
000000a0 a6e601            165                 cp     key_cnt,#increase       ; increase key?
000000a3 eb**              166                 jr     ne,try_decrease         ; no, try decrease key
000000a5 a6e9e0            167                 cp     pwm,#max                ; are we at maximum pwm?
000000a8 6b**              168                 jr     eq,exit                 ; yes, don't increment
000000aa 9e                169                 inc    pwm                     ; increment pwm (increase speed)
000000ab 99f2              170                 ld     t1,pwm                  ; load timer with pwm value
000000ad 8b**              171                 jr     exit                    ; exit routine
000000af a6e602            172 try_decrease:   cp     key_cnt,#decrease       ; decrease key?
000000b2 eb**              173                 jr     ne,try_brake            ; try brake key
000000b4 a6e920            174                 cp     pwm,#min                ; is pwm at minimum?
000000b7 3b**              175                 jr     ule,exit                ; if yes, don't decrement
000000b9 00e9              176                 dec    pwm                     ; decrement pwm (decrease speed)
000000bb 99f2              177                 ld     t1,pwm                  ; load new pwm value
000000bd 8b**              178                 jr     exit                    ; exit
000000bf a6e603            179 try_brake:      cp     key_cnt,#brake          ; brake button down?
000000c2 eb**              180                 jr     ne,exit                 ; exit if not
000000c4 a6eaff            181                 cp     make,#0ffh              ; button still down?
000000c7 6b**              182                 jr     eq,exit_1               ; yes - take no action
000000c9 b60320            183                 xor    p3,#20h                 ; toggle brake bit
000000cc acff              184                 ld     make,#0ffh              ; set make flag
000000ce 760320            185                 tm     p3,#20h                 ; brake high?
000000d1 eb**              186                 jr     nz,test_dir             ; test dir sw if brakes are on only
000000d3 c902              187                 ld     p2,temp_led             ; load direction led
000000d5 bc01              188                 ld     STATE,#start_up         ; motor turning
000000d7 8b**              189                 jr     exit_1                  ; exit
000000d9 bc00              190 test_dir:       ld     STATE,#stopped          ;
000000db e6fb10            191                 ld     imr,#irq4               ;
000000de c802              192                 ld     temp_led,p2             ; get led data
000000e0 e602bf            193                 ld     p2,^C #stop_led         ; turn on stop led
000000e3 8b**              194                 jr     exit_1                  ;
000000e5 78e6              195 load_keys:      ld     key_temp,key_cnt        ; transfer key data
000000e7 b0e6              196                 clr    key_cnt                 ; clear key register
000000e9 bf                197                 iret                           ; return to interrupt
000000ea 5aa4              198 no_keys:        djnz   count,key_loop          ; tested all keys?
000000ec b0ea              199 exit:           clr    make                    ;
000000ee b0e4              200 exit_1:         clr    bounce                  ; clear key registers
000000f0 b0e6              201                 clr    key_cnt                 ;
000000f2 b0e7              202                 clr    key_temp                ;
000000f4 b0e5              203                 clr    count                   ;
000000f6 bf                204 no_irq:         iret                           ; return from interrupt
                           205
                           206                 .end
```

© 1997 by Zilog, Inc. All rights reserved. No part of this document may be copied or reproduced in any form or by any means without the prior written consent of Zilog, Inc. The information in this document is subject to change without notice. Devices sold by Zilog, Inc. are covered by warranty and patent indemnification provisions appearing in Zilog, Inc. Terms and Conditions of Sale only. Zilog, Inc. makes no warranty, express, statutory, implied or by description, regarding the information set forth herein or regarding the freedom of the described devices from intellectual property infringement. Zilog, Inc. makes no warranty of merchantability or fitness for any purpose. Zilog, Inc. shall not be responsible for any errors that may appear in this document. Zilog, Inc. makes no commitment to update or keep current the information contained in this document.

Zilog's products are not authorized for use as critical components in life support devices or systems unless a specific written agreement pertaining to such intended use is executed between the customer and Zilog prior to use. Life support devices or systems are those which are intended for surgical implantation into the body, or which sustains life whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in significant injury to the user.

Zilog, Inc., 210 East Hacienda Ave.
Campbell, CA 95008-6600
Telephone (408) 370-8000
FAX (408) 370-8056
BBS (408) 370-8024
Internet: http://www.zilog.com