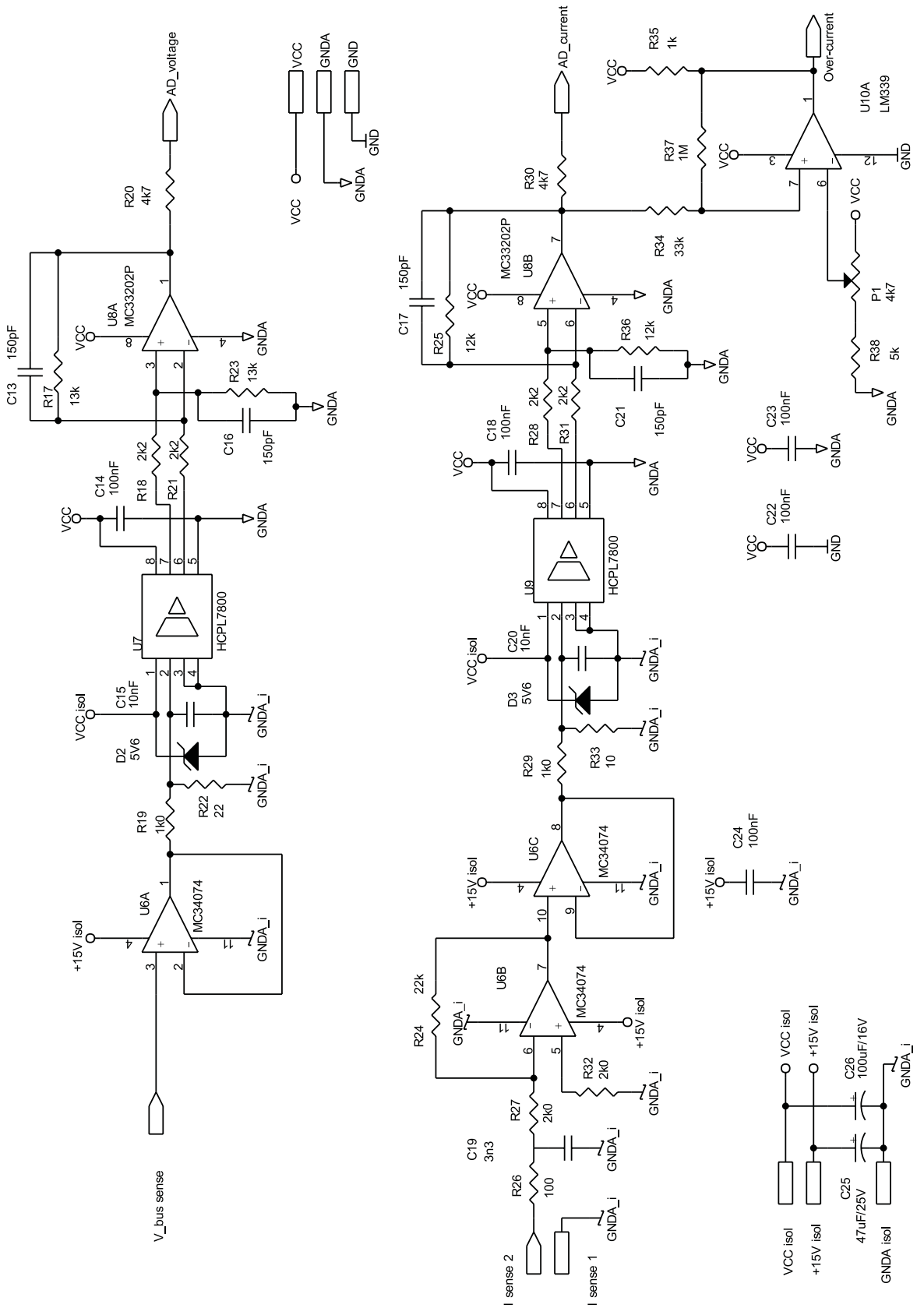

Figure A-4. Sensor Board 3 of 3 (Voltage and Current Sensing)

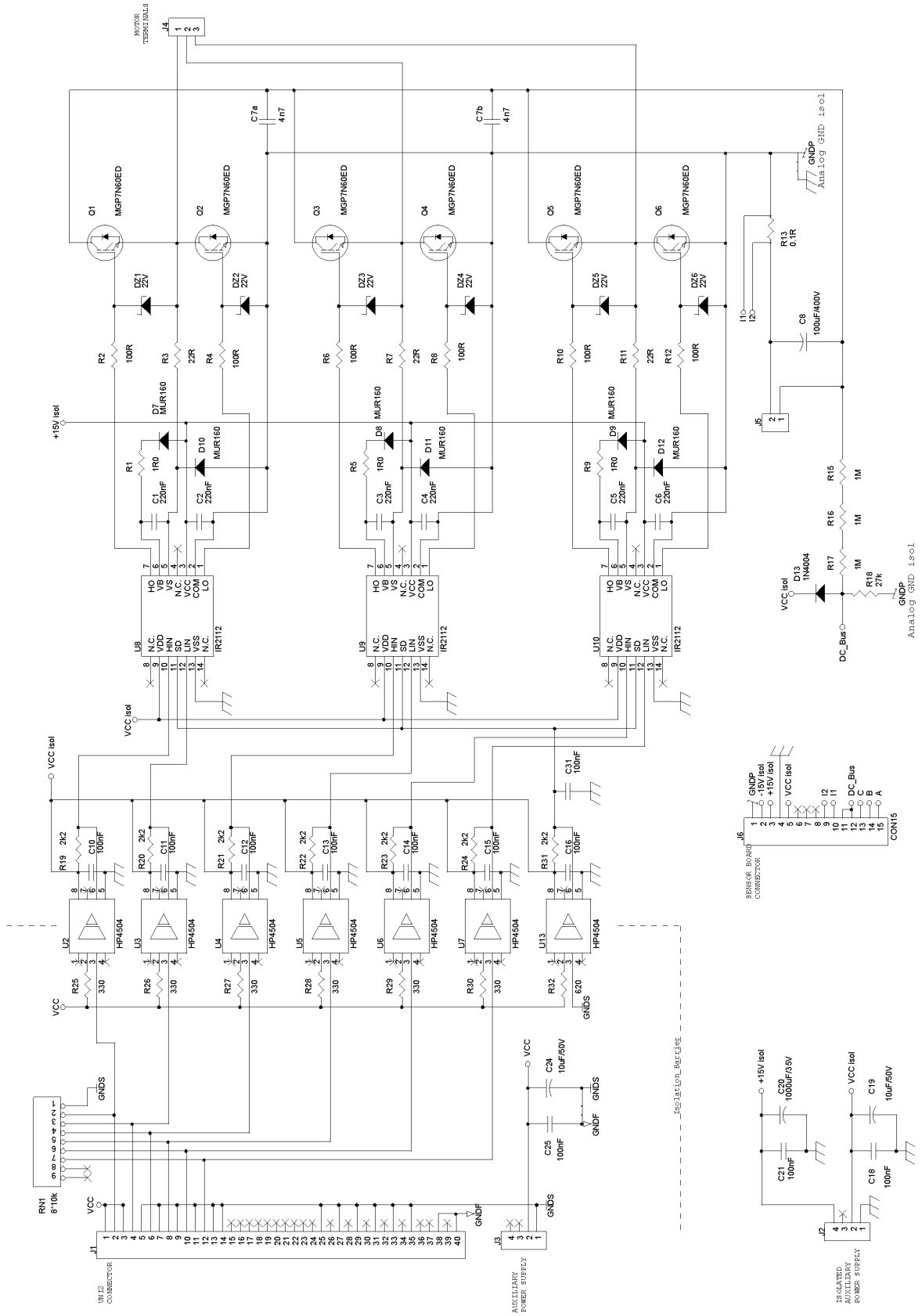

Figure A-5. Power Stage Board

A.2 Part List of Components

Table A-1 Part List - Microcontroller Board

Component	Value/Rating	Description	Quantity
U1	MC68HC908MR24FU	Microcontroller	1
D1 - D3	LED	LED	3
X1	4.9152 MHz	Crystal	1
R1 - R3	1k	Resistor	3
R4	10k	Resistor	1
R5	10M	Resistor	1
P1	10k	Potentiometer	1
C1 - C5, C8 - C10	100nF	Capacitor	8
C6, C7	20pF	Capacitor	2
C11	10µF/16V	Electrolytic Capacitor	1
J1	Con40	Connector	1
J2	Con3	Connector	1
SW1, SW2	-	Switch	2
SW3	-	2 bit DIP Switch	1

All tolerances ±10% for capacitors and ±1% for resistors, unless otherwise specified.

Table A-1 Part List - Sensor Board

Component	Value/Rating	Description	Quantity
U3, U10	LM339	Comparator	2
U6	MC34074	Operational Amplifier	1
U7, U9	HCPL7800	Isolation Amplifier	2
U8	MC33202P	Operational Amplifier	1
D1	1N5817	Diode	1
D2, D3	5V6	Zener Diode	2
R11	100k	Resistor	1
R12	15k	Resistor	1
R13	10k	Resistor	1
R14	6k8	Resistor	1
R15, R18, R21, R28, R31	2k2	Resistor	5

Table A-1 Part List - Sensor Board

Component	Value/Rating	Description	Quantity
R16	470	Resistor	1
R20, R30	4k7	Resistor	2
R17, R23	13k	Resistor	2
R19, R29	1k0	Resistor	2
R22	22	Resistor	1
R24	22k	Resistor	1
R25, R36	12k	Resistor	2
R26	100	Resistor	1
R27, R32	2k0	Resistor	2
R33	10	Resistor	1
R34	33k	Resistor	1
R35	1k	Resistor	1
R37	1M	Resistor	1
R38	5k	Resistor	1
P1	4k7	Trimmer	1
C10, C26	100µF/16V	Electrolytic Capacitor	2
C11	220nF/100V	Capacitor	1
C12, C14, C18, C22, C23, C24	100nF	Capacitor	6
C13, C16, C17, C21	150pF	Capacitor	4
C15, C20	10nF	Capacitor	2
C19	3n3	Capacitor	1
C25	47µF/25V	Electrolytic Capacitor	1
J1	Con15	Connector	1
J2	Con 40	Connector	1
J3	Con 2	Connector	1

All tolerances ±10% for capacitors and ±1% for resistors, unless otherwise specified.

Table A-1 Part List - Three Phase HV Power Board

Component	Value/Rating	Description	Quantity
U2, U3, U4, U5, U6, U7, U13	HP4504	Optocoupler	6
U8, U9, U10	IR2112	Gate Driver	3
Q1 - Q6	MGP8N60ED	Copack IGBT	6
DZ1 - DZ6	22V	Zener Diode	6
D7, D8, D9, D10, D11, D12	MUR160	Diode	6
D13	1N4004	Diode	1
R1, R5, R9	1R0	Resistor	3
R2, R4, R6, R8, R10, R12	100R	Resistor	6
R3, R7, R11	22R	Resistor	3
R13	0.1R	Current Sense Resistor	1
R15, R16, R17	1M	Resistor	3
R18	27k	Resistor	1
R19 - R24, R31	2k2	Resistor	7
R25 - R30	330	Resistor	6
R32	620	Resistor	1
RN1	8x10k	Resistor Net	1
C1 - C6	220nF/63V	Capacitor	6
C7a, C7b	4n7/630V-	Capacitor	2
C8	100µF/400V	Electrolytic Capacitor	1
C10 - C16, C18, C21, C25, C31	100nF	Capacitor	11
C19, C24	100µF/50V	Electrolytic Capacitor	2
C20	1000µF/35V	Capacitor	1
J1	Con40	Connector	1
J2, J3	Con4	Connector	2
J4	Con3	Connector	1
J5	Con2	Connector	1
J6	Con15	Connector	1

All tolerances ±10% for capacitors and ±1% for resistors, unless otherwise specified.

APPENDIX B

B.1 Subroutine “PWM Calculation”

```
/*
 * Project:              CLOSED LOOP 3-PHASE AC DRIVE
 *
 * Microcontroller:      Motorola MC68HC908MR24
 *
 * Module:               PWMCALC.C
 * Revision/Date:        1.0 / June 1998
 * Description:          This routine is 2nd level ISR responding to PWM interrupt.
 *                       Input:   New waveform parameters Incval and Amplitude
 *                       Output:  Load 3 PWM registers PVAL1, PVAL3, PVAL5
 *
 * Compiler:             C Cross Compiler - COSMIC Software Inc.
 *
 * Author:               Radim VISINKA
 * Company:              MOTOROLA SPS
 *                       Roznov System Application Laboratory
 *                       Roznov pod Radhostem, Czech Republic
 *
 * ================================================================
 *
 * Copyright (c):        MOTOROLA Inc.,1998, All rights reserved.
 *
 * ================================================================
 * THIS SOFTWARE IS PROVIDED BY MOTOROLA RSAL "AS IS" AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL MOTOROLA RSAL OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ================================================================
 */

/* DEFINITION_START */

/* Include Header Files */
#include <mr24io.h>       /* file contains input/output file */
#include "CONST.H“        /* file contains global constants and definitions */
#include <3rdhquad.h>     /* contains wave table for one quadrant*/
                          /* 3rdhquad.h for sine wave with third harmonic */
                          /* constant unsigned int wavequad[256] */
```

```
/* Global Variables (External) - 8 bit */
extern unsigned char Amplitude;           /* 0 to 255 gives 0 to 100% modulation*/

/* Global Variables (External) - 16 bit */
extern signed int Table_inc;      /* table wave increment */

/* Local Variables - 8 bit */
unsigned char Table_value;        /* Value read from wave table */

/* Local Variables - 16 bit */
unsigned int Wave_ptr_a = 0;      /* wave pointer for phase A */
unsigned int Wave_ptr_b;          /* wave pointer for phase B */
unsigned int Wave_ptr_c;          /* wave pointer for phase C */
unsigned int Quad_ptr;            /* quadrant pointer for phase A */
unsigned int Pwmmod_wave;         /* wave modulus */

/* DEFINITION_END */



void PWMcalc (void)
        {
        COPCTL=0x00;                      /* service COP */
        PCTL1 &= 0xef;                    /* clear PWMF bit */

/* PHASE A */

     Wave_ptr_a += Table_inc;             /* load new wave pointer for phase A */

     if (Wave_ptr_a < 0x4000)                     /* QUADRANT 1 */
     {
       Quad_ptr = (Wave_ptr_a)<<2;                /* calculate quadrant pointer
                                                  from wave pointer */
       Table_value = (wavequad[Quad_ptr>>8]);             /* fetch value from table */
       Pwmmod_wave = (Table_value * Amplitude);           /* scale by Amplitude */
       PVAL1 = (Pwmmod_wave>>8) + (PWM_MODULUS/2);
                                          /* update PVAL1 register   for QUADRANT 1 */
     }

     else if (Wave_ptr_a < 0x7fff)                /* QUADRANT 2 */
     {
       Quad_ptr = (0x7fff - Wave_ptr_a)<<2;               /* quadrant pointer */
       Table_value = (wavequad[Quad_ptr>>8]);             /* fetch value from table */
       Pwmmod_wave = (Table_value * Amplitude);           /* scale by Amplitude */
       PVAL1 = (Pwmmod_wave>>8) + (PWM_MODULUS/2);
                                          /* update PVAL1 register for QUADRANT 2 */
     }

     else if (Wave_ptr_a < 0xbfff)                /* QUADRANT 3 */
     {
       Quad_ptr = (Wave_ptr_a-0x7fff)<<2;                 /* quadrant pointer */
       Table_value = (wavequad[Quad_ptr>>8]);             /* fetch value from table */
       Pwmmod_wave = (Table_value * Amplitude);           /* scale by Amplitude */
```

```
      PVAL1 = (PWM_MODULUS/2) - (Pwmmod_wave>>8);
                                        /* update PVAL1 register for QUADRANT 3 */
    }

    else       /* (Wave_ptr_a < 0xffff)          QUADRANT 4 */
    {
      Quad_ptr = (0xffff - Wave_ptr_a)<<2;              /* quadrant pointer */
      Table_value = (wavequad[Quad_ptr>>8]);            /* fetch value from table */
      Pwmmod_wave = (Table_value * Amplitude);          /* scale by Amplitude */
      PVAL1 = (PWM_MODULUS/2) - (Pwmmod_wave>>8);
                                        /* update PVAL1 register for QUADRANT 4 */
    }

    /* PVAL2 is updated automaticaly because of COMPLEMENTARY PWM MODE */

/* PHASE B */

    Wave_ptr_b = Wave_ptr_a + 0x5555;

    if (Wave_ptr_b < 0x4000)                    /* QUADRANT 1 */
    {
      Quad_ptr = Wave_ptr_b<<2;                         /* quadrant pointer */
      Table_value = (wavequad[Quad_ptr>>8]);            /* fetch value from table */
      Pwmmod_wave = (Table_value * Amplitude);          /* scale by Amplitude */
      PVAL3 = (Pwmmod_wave>>8) + (PWM_MODULUS/2);
                                        /* update PVAL3 register for QUADRANT 1 */
    }

    else if (Wave_ptr_b < 0x7fff)               /* QUADRANT 2 */
    {
      Quad_ptr = (0x7fff - Wave_ptr_b)<<2;              /* quadrant pointer */
      Table_value = (wavequad[Quad_ptr>>8]);            /* fetch value from table */
      Pwmmod_wave = (Table_value * Amplitude);          /* scale by Amplitude */
      PVAL3 = (Pwmmod_wave>>8) + (PWM_MODULUS/2);
                                        /* update PVAL3 register for QUADRANT 2 */
    }

    else if (Wave_ptr_b < 0xbfff)               /* QUADRANT 3 */
    {
      Quad_ptr = (Wave_ptr_b - 0x7fff)<<2;              /* quadrant pointer */
      Table_value = (wavequad[Quad_ptr>>8]);            /* fetch value from table */
      Pwmmod_wave = (Table_value * Amplitude);          /* scale by Amplitude */
      PVAL3 = (PWM_MODULUS/2) - (Pwmmod_wave>>8);
                                        /* update PVAL3 register for QUADRANT 3 */
    }

    else       /* (Wave_ptr_b < 0xffff)    QUADRANT 4 */
    {
      Quad_ptr = (0xffff - Wave_ptr_b)<<2;              /* quadrant pointer */
      Table_value = (wavequad[Quad_ptr>>8]);            /* fetch value from table */
      Pwmmod_wave = (Table_value * Amplitude);          /* scale by Amplitude */
      PVAL3 = (PWM_MODULUS/2) - (Pwmmod_wave>>8);
                                        /* update PVAL3 register for QUADRANT 4 */
```

```
    }

    /* PVAL4 is updated automaticaly because of COMPLEMENTARY PWM MODE */

/* PHASE C */

    Wave_ptr_c = Wave_ptr_a + 0xaaaa;

    if (Wave_ptr_c < 0x4000)                    /* QUADRANT 1 */
    {
      Quad_ptr = Wave_ptr_c<<2;                         /* quadrant pointer */
      Table_value = (wavequad[Quad_ptr>>8]);            /* fetch value from table */
      Pwmmod_wave = (Table_value * Amplitude);          /* scale by Amplitude */
      PVAL5 = (Pwmmod_wave>>8) + (PWM_MODULUS/2);
                              /* update PVAL5 register for QUADRANT 1 */
    }

    else if (Wave_ptr_c < 0x7fff)               /* QUADRANT 2 */
    {
      Quad_ptr = (0x7fff - Wave_ptr_c)<<2;              /* quadrant pointer */
      Table_value = (wavequad[Quad_ptr>>8]);            /* fetch value from table */
      Pwmmod_wave = (Table_value * Amplitude);          /* scale by Amplitude */
      PVAL5 = (Pwmmod_wave>>8) + (PWM_MODULUS/2);

/* update PVAL5 register

   for QUADRANT 2 */
    }

    else if (Wave_ptr_c < 0xbfff)               /* QUADRANT 3 */
    {
      Quad_ptr = (Wave_ptr_c - 0x7fff)<<2;              /* quadrant pointer */
      Table_value = (wavequad[Quad_ptr>>8]);            /* fetch value from table */
      Pwmmod_wave = (Table_value * Amplitude);          /* scale by Amplitude */
      PVAL5 = (PWM_MODULUS/2) - (Pwmmod_wave>>8);
                                      /* update PVAL5 register for QUADRANT 3 */
    }

    else       /* (Wave_ptr_c < 0xffff)          QUADRANT 4 */
    {
      Quad_ptr = (0xffff - Wave_ptr_c)<<2;              /* quadrant pointer */
      Table_value = (wavequad[Quad_ptr>>8]);            /* fetch value from table */
      Pwmmod_wave = (Table_value * Amplitude);          /* scale by Amplitude */
      PVAL5 = (PWM_MODULUS/2) - (Pwmmod_wave>>8);
                                      /* update PVAL5 register for QUADRANT 4 */
    }

    /* PVAL6 is updated automaticaly because of COMPLEMENTARY PWM MODE */

    PCTL1 |= 0x02;                      /* set LDOK bit */
    }
```

B.2 Subroutine “V/Hz Ramp”

```
/* DEFINITION START */

/* Constant Definitions */
#define VOLTS_BOOST         10            /* min. voltage for boost = 10% from 255 */
#define FREQ_BOOST          0x0f00        /* boost frequency = 15Hz = 0x0f00 */
#define FREQ_BASE           0x3200        /* frequency base point 50Hz=0x3200 */

/* Global Variables (External) - 8 bit */
extern unsigned char Amplitude;           /* 0 to 255 gives 0 to 100% modulation */

/* Global Variables (External) - 16 bit */
extern signed int Table_inc;              /* table wave increment */
extern signed int V_out;                  /* actual generated frequency */

/* Local Variables - 16 bit */
signed int V_out_abs;                     /* ABS(V_out) */
unsigned int Boost_slope;                 /* Boost slope pre-calculation */
unsigned int Temp_var_16;                 /* temporary 16-bit variable */
unsigned int Amplitude_16;                /* temporary 16-bit amplitude */

/* Local Variables - 32 bit */
unsigned long Temp_var_32;                /* temporary 32-bit variable */

/* DEFINITION END */



          /* SUBROUTINE VHZ_RAMP */

/*
Based on output from PI controller (V_out = required motor frequency) the routine calculates
the wave increment Incval and voltage scale Amplitude.
These two parameters are inputs to the PWM Calculation routine PWMcalc()
*/

void vhz_ramp (void)
     {
          /* Calculate wave increment Incval for rolling thru wavetable */

     Table_inc = V_out >> 4;        /* divide by 16 to get proper wave increment
                                    in 8.8 format for PWM reload=PWM/4 */

          /* Calculate Amplitude according V/Hz ramp */

     V_out_abs = abs(V_out);

     if (V_out_abs < FREQ_BOOST)
     {                      /* if ABS(V_out) < FREQ_BOOST */

          /* Initialise boost of V/Hz ramp (can be implemented during
                  program initialisation)*/
```

```
Boost_slope = (FREQ_BOOST<<16)/(FREQ_BASE) - (VOLTS_BOOST * 0x028f);
              /* 0x028f scales the range of VOLTS_BOOST from 100% to 0xffff */

Temp_var_32 = (long)Boost_slope * (long)V_out_abs;
Temp_var_16 = Temp_var_32 / FREQ_BOOST;
Amplitude_16 = Temp_var_16 + (VOLTS_BOOST * 0x028f);
Amplitude = Amplitude_16>>8;           /* 16 to 8 bit */
}
else
{
  if (V_out_abs < FREQ_BASE)
  {                    /* if FREQ_BOOST < ABS(V_out) < FREQ_BASE */
      Amplitude = V_out_abs/(FREQ_BASE>>8);
  }
  else                 /* if ABS(V_out) > BASE_FREQ*/
  {
    Amplitude = 0xff;
  }
}
}
```

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™
semiconductor