APPENDIX C2: PWM-Inverter Program (open loop)

```
.module inv;

#include <def2101.h>;
#define   MAX_LENGTH  0x0eb8;  { 0.92 => 4.12 integer format }
                         { If this constant is changed, table file has }
                         { to be changed too !! }

#define   TRANSITION_LENGTH   0x04  { UNIT IS DOUBLE CLOCK = 125 ns}
#define   CHARGING_LENGTH     0x05
#define   DISCHARGING_LENGTH  0x03

#define   AMPLITUDE           0x6000

.const        CHARGE    = TRANSITION_LENGTH + CHARGING_LENGTH;
.const        DISCHARGE = TRANSITION_LENGTH + DISCHARGING_LENGTH;

#define   MINIMUM_LENGTH      0x06

#define   CURRENT_LIMIT       0x7200  { 1.1/1.25 * 32767 }

#define adjust  SR = lshift MR0 by 4 (lo); \
            SR = SR or ashift MR1 by 4 (hi)

.var/dm/ram/abs=0x0800   data;
.var/dm/ram/abs=0x0801   type;
.var/dm/ram/abs=0x0802   reset_altera;
.var/dm/ram/abs=0x0803   reserved;
.var/dm/ram/abs=0x0804   time1;
.var/dm/ram/abs=0x0805   time2;
.var/dm/ram/abs=0x0806   time3;
.var/dm/ram/abs=0x0807   time4;

.var/dm/ram/abs=0x0400   dac0;
.var/dm/ram/abs=0x0401   dac1;
.var/dm/ram/abs=0x0402   dac2;
.var/dm/ram/abs=0x0403   dac3;

.var/dm/ram/abs=0x0000   adc0;
.var/dm/ram/abs=0x0001   adc1;
.var/dm/ram/abs=0x0002   adc2;
.var/dm/ram/abs=0x0003   adc3;
.var/dm/ram/abs=0x0004   adc4;
.var/dm/ram/abs=0x0005   adc5;
.var/dm/ram/abs=0x0006   adc6;
.var/dm/ram/abs=0x0007   adc7;

{ Measured values }

.var/dm/ram        CosTheta;     { Channel 0 = Ea = CosTheta }
.var/dm/ram        Eb;         { Channel 1 = Eb          }
.var/dm/ram        Idref;        { Channel 2 = Ia = Ialpha   }
.var/dm/ram        Ialpha;       { Channel 3 = Ib        }
```

```
.var/dm/ram        Ib;         { Channel 4 = Idref       }

{ Calculated values }

.var/dm/ram        SinTheta;
.var/dm/ram        Ic;
.var/dm/ram        Ibeta;
.var/dm/ram             Id;
.var/dm/ram        Iq;
.var/dm/ram        dd;
.var/dm/ram        dq;
.var/dm/ram        dalpha;
.var/dm/ram        dbeta;
.var/dm/ram        d1;
.var/dm/ram        d2;
.var/dm/ram        new_cycle_flag;
.var/dm/ram        sector;
.var/dm/ram        offset;
.var/dm/ram        cos_counter;
.var/dm/ram        skip_counter;
.var/dm/ram        amp;
.var/dm/ram        old_sector;
.var/dm/ram        invert_flag;
.init              invert_flag: 0;

{ Constant variables }

.var/dm/ram    OneOverSqrTwo;
.init          OneOverSqrTwo:      0x5a82; { 32757 * 0.707106781 }
.var/dm/ram    OneOverTwoSqrThree;
.init          OneOverTwoSqrThree: 0x24f3; { 32767 * 0.288675134 }
.var/dm/ram    OneOverSqrThree;
.init          OneOverSqrThree:    0x49e6; { 32767 * 0.577350269 }
.var/dm/ram    OneOverSqrTwoSqrThree;
.init          OneOverSqrTwoSqrThree: 0x3441; { 32767 * 0.40824829 }
.var/dm/ram    OneOverTwo;
.init          OneOverTwo:      0x3FFF; { 32767 * 0.5 }
.var/dm/ram    SqrThreeOverTwo;
.init          SqrThreeOverTwo: 0x6ED9; { 32767 * 0.866025403 }
.var/dm/ram    Iqref;
.init          Iqref: 0;         { Iqref = 0 ; 4.12 Iq error      }
.var/dm/ram    Kp;
.init          Kp: 0x0800;       { Kp = 0.5 ; Kpmax = 1 ; Format 4.12}
.var/dm/ram    wL;
.init          wL: 0;{0x0c10;} { wL = 2 * Pi * 60 * 2e-3 = 0.75 ; 4.12 }

{ Circular buffers and tables }

.var/dm/ram/circ   table[200];
.init          table: <cossin.dat>;

.var/dm/ram/circ   IaIn[3];
.init          IaIn: 0,0,0;
.var/dm/ram/circ   IaOut[2];
.init          IaOut: 0,0;
.var/dm/ram/circ   IbIn[3];
.init          IbIn: 0,0,0;
.var/dm/ram/circ   IbOut[2];
.init          IbOut: 0,0;
.var/dm/ram/circ   VaIn[3];
.init          VaIn: 0,0,0;
```

```
.var/dm/ram/circ   VaOut[2];
.init          VaOut: 0,0;
.var/dm/ram/circ   VbIn[3];
.init          VbIn: 0,0,0;
.var/dm/ram/circ   VbOut[2];
.init          VbOut: 0,0;

.var/dm/ram    case_table[36];
.init          case_table: b#110, { b#XYZ  }
                           b#001,
                           b#101, { X  = 1  d1 and d2 are swaped }
                           b#000, { X  = 0  d1 and d2 are not swaped }
                           b#100, { YZ = 00 output waveform type 0 }
                           b#010, { YZ = 01 output waveform type 1 }
                           b#101, { YZ = 10 output waveform type 2 }
                           b#010,
                           b#001,
                           b#100,
                           b#110,
                           b#000,
                           b#001,
                           b#000,
                           b#010,
                           b#110,
                           b#101,
                           b#100,
                           b#010,
                           b#100,
                           b#000,
                           b#101,
                           b#001,
                           b#110,
                           b#000,
                           b#110,
                           b#100,
                           b#001,
                           b#010,
                           b#101,
                           b#100,
                           b#101,
                           b#110,
                           b#010,
                           b#000,
                           b#001;

{************************* Interrupt vectors
*****************************}

      AR = 1;               { Reset Vector }
      DM(reset_altera) = AR;
      jump start;
      rti;

interrupt_two:
      DM(new_cycle_flag) = AY1;      { irq2 }
      rti;
      rti;
      rti;

      rti; rti; rti; rti;         { sport0 TX }
      rti; rti; rti; rti;         { sport0 RX }
```

```
      rti; rti; rti; rti;         { irq0 }
      rti; rti; rti; rti;         { irq1 }

timer_interrupt:
      AR = 1;
      DM(reset_altera) = AR;
      idle;
      jump timer_interrupt;       { timer is used as a watch dog}


start:

{************************ T I M E R   S T U F F
**************************}

      AX0 = 0;    DM(Tscale_Reg)  = AX0;   { 540 x 62.5ns = 33.75us  }
      AX0 = 540;  DM(Tcount_Reg)  = AX0;   { Watchdog for 31.4 us    }
      AX0 = 540;  DM(Tperiod_Reg) = AX0;

{************** S Y S T E M   A N D   M E M O R Y   S T U F F
***************}

      AX0=h#0052; DM(Dm_Wait_Reg)=AX0;   {1-2-2 DM wait states         }

      AX0=h#0028; DM(Sys_Ctrl_Reg)=AX0;  {SPORT0 disab, SPORT1 disab,
}
                                  {use FI,FO,IRQ0,IRQ1,SCLK instead }
                                  {BOOT_PAGE=0, BMWAIT=5, PMWAIT=0  }

      IFC = 0x03f;          { Clear all interrupts just for any case }
      IFC = 0x000;

      ICNTL=b#00111;        { Nesting disable, all ints edge triggered}

      IMASK=b#100001;     { Interrupts IRQ2, SPORT0 TX, SPORT0 RX,
                              SPORT1 TX, SPORT0 RX, Timer }

      MSTAT=b#1001000; {Go mode, timer dis, frac, Sat mode, Ovr ltch
dis,}
                  {Bit rev dis, prim regs}


{*************************** Initialization
****************************}

      AR = DM(adc2);    { Start Channel 2 = Ia }

      AY1 = 0x0080;

      I0 = ^IaIn;
      M0 = 1;
      L0 = %IaIn;

      I1 = ^IaOut;
      M1 = 1;
      L1 = %IaOut;

      I2 = ^IbIn;
      M2 = 1;
      L2 = %IbIn;
```

```
      I3 = ^IbOut;
      M3 = 1;
      L3 = %IbOut;

      I4 = ^table;
      M4 = 1;
      L4 = %table;

      I5 = ^b1000;
      M5 = 1;
      L5 = 3;

      I6 = ^a1000;
      M6 = 1;
      L6 = 2;

      I7 = 0;              { Misc. pointer                 }
      M7 = 1;              { It is loaded before each access }
      L7 = 0;

      AR = 0;
      DM(new_cycle_flag) = AR;
      DM(reset_altera) = AR;

sync:
      AR = DM(new_cycle_flag);
      AR = pass AR;
      if eq jump sync;

      AR = 0;
      DM(new_cycle_flag) = AR;
      DM(cos_counter) = AR;

      AR = -1;
      DM(skip_counter) = AR;

      AR = AMPLITUDE;
      DM(amp) = AR;

      reset flag_out;
      ena timer;

{ ********************** End of Initialization
************************ }

begin:

{----------------------- Read and start conversion --------------------
-}

      AR = DM(adc3);       { Read Channel 2 = Ia, Start Channel 3 = Ib }

      AR = AR xor AY1;
      SR = lshift AR by 8 (lo);
                                  { Digital filter }
      DM(I0,M0) = SR0;
      MX0 = DM(I0,M0), MY0 = PM(I5,M5);
      MR = MX0 * MY0 (ss), MX0 = DM(I0,M0), MY0 = PM(I5,M5);
      MR = MR + MX0 * MY0 (ss), MX0 = DM(I0,M0), MY0 = PM(I5,M5);
      MR = MR + MX0 * MY0 (ss), MX0 = DM(I1,M1), MY0 = PM(I6,M6);
      MR = MR + MX0 * MY0 (ss), MX0 = DM(I1,M1), MY0 = PM(I6,M6);
```

```
      MR = MR + MX0 * MY0 (rnd);
      MY0 = 0x7fff;
      MR = MR + MX0 * MY0 (rnd);
      DM(I1,M1) = MR1;

      DM(Ialpha) = MR1;

      AY0 = CURRENT_LIMIT;
      AR = abs MR1;
      AR = AR - AY0;
      IF gt jump shutdown;

{      SR = lshift MR1 by -8 (lo);
      AR = SR0 xor AY1;
      DM(dac3) = AR;
}
{-------------------------------------------------------------------------
-}

{ ************************* Watchdog  ***************************** }

      AX0 = 540;
      DM(Tcount_Reg)    = AX0;

      AR = 0;
      DM(new_cycle_flag) = AR; { reset new_cycle_flag }
      nop;
        nop;
        nop;
        nop;
        nop;
        nop;
        nop;

{----------------------- Read and start conversion -------------------
-}

      AR = DM(adc2);       { Read Channel 3 = Ib, Start Channel 2 = Ia }

      AR = AR xor AY1;   { AR = AR xor 0x0080 }
      SR = lshift AR by 8 (lo);
                                  { Digital filter }
      DM(I2,M2) = SR0;
      MX0 = DM(I2,M2), MY0 = PM(I5,M5);
      MR = MX0 * MY0 (ss), MX0 = DM(I2,M2), MY0 = PM(I5,M5);
      MR = MR + MX0 * MY0 (ss), MX0 = DM(I2,M2), MY0 = PM(I5,M5);
      MR = MR + MX0 * MY0 (ss), MX0 = DM(I3,M3), MY0 = PM(I6,M6);
      MR = MR + MX0 * MY0 (ss), MX0 = DM(I3,M3), MY0 = PM(I6,M6);
      MR = MR + MX0 * MY0 (rnd);
      MY0 = 0x7fff;
      MR = MR + MX0 * MY0 (rnd);
      DM(I3,M3) = MR1;

      DM(Ib) = MR1;

      AY0 = CURRENT_LIMIT;
      AR = abs MR1;
      AR = AR - AY0;
      IF gt jump shutdown;

{      SR = lshift MR1 by -8 (lo);
```

```
      AR = SR0 xor AY1;
      DM(dac1) = AR;
}
{-------------------------------------------------------------------------
-}

{ ************************* Calculate Ic
***************************** }

      AX0 = DM(Ialpha);
{
      SI = AX0;
      SR = lshift SI by -8 (hi);
      AR = SR1 xor AY1;
      DM(dac1) = AR;
}
      AY0 = DM(Ib);
      AR = AX0 + AY0;        { Ic = -(Ia + Ib) }
      AR = -AR, MR1 = AY0;   { MR1 = Ib }
      DM(Ic) = AR;
{
      SR = lshift AR by -8 (hi);
      AR = SR1 xor AY1;
      DM(dac2) = AR;
}


{ ****************** Generating cos(theta) and sin(theta)
***************** }

      AR = DM(skip_counter);
      AY0 = 1;
      AR = AR + AY0;

      AY0 = 0x0000;
      AR = AR and AY0;
      DM(skip_counter) = AR;
      if ne jump conversion;

      AR = DM(I4,M4);
      SR = lshift AR by 8 (hi);
      AY0 = 0xff00;
      AR = AR and AY0;
      AX1 = AR;
      MY0 = DM(amp);

      AY0 = 200;
      AR = DM(cos_counter);
      AR = AR - AY0;
      if lt jump l0to200;
      AR = AR - AY0;
      if lt jump l200to400;
      AR = AR - AY0;
      if lt jump l400to600;

l600to800:
      AR = AX1;
      MR = AR * MY0 (rnd);
      AR = -MR1;
      DM(dbeta) = AR;
```

```
      MR = SR1 * MY0 (rnd);
      DM(dalpha) = MR1;
      jump inc_cos_counter;

l400to600:
      AR = AX1;
      MR = AR * MY0 (rnd);
      AR = -MR1;
      DM(dalpha) = AR;

      MR = SR1 * MY0 (rnd);
      AR = -MR1;
      DM(dbeta) = AR;
      jump inc_cos_counter;

l200to400:
      AR = AX1;
      MR = AR * MY0 (rnd);
      DM(dbeta) = MR1;

      MR = SR1 * MY0 (rnd);
      AR = -MR1;
      DM(dalpha) = AR;
      jump inc_cos_counter;

l0to200:
      AR = AX1;
      MR = AR * MY0 (rnd);
      DM(dalpha) = MR1;

{      AY0 = 0x4000;
      AR = AX1 - AY0;
      if eq set flag_out;
}

      MR = SR1 * MY0 (rnd);
      DM(dbeta) = MR1;

inc_cos_counter:
      AR = DM(cos_counter);
      AY0 = 1;
      AR = AR + AY0;
      DM(cos_counter) = AR;
      AY0 = 800;
      AR = AR - AY0;
      if ge jump reset_cos_counter;
      jump conversion;

reset_cos_counter:
      AR = 0;
      DM(cos_counter) = AR;
      jump conversion;
                           { !!!!!!!!!!!!!!!!!!!! HERE !!!!!!!!}
{
      MY0 = DM(amp);
      AR = 0x5400;
      MR = AR * MY0 (rnd);
      DM(dalpha) = MR1;
      AR = 0x5F00;
      MR = AR * MY0 (rnd);
      DM(dbeta) = MR1;
```

```
}
{ ************* Conversion (dalpha,dbeta) to (d1, d2, sec #)
************ }

conversion:
      MY0 = DM(dalpha);
{
      SI = MY0;
      SR = lshift SI by -8 (hi);
      AR = SR1 xor AY1;
      DM(dac0) = AR;
}
      MR1 = DM(dbeta);
{
      SR = lshift MR1 by -8 (hi);
      AR = SR1 xor AY1;
      DM(dac2) = AR;
}
      AX1 = MR1;           { AX1 = dbeta }
      MR = MR1 * MY0 (rnd);
      AR = pass MR1, AX0 = MY0;  { AX0 = dalpha }
      if lt jump not_same_sign;
      same_sign:
          AF = abs AX0;       { AF = abs(dalpha) }
          AR = abs AX1;       { AR = abs(dbeta)  }
          MY0 = DM(OneOverSqrThree);
          MR = AR * MY0 (rnd);      { MR1 = 1/Sqrt(3) * abs(dbeta) }
          AR = MR1 - AF, AY0 = MR1; { AY0 = 1/Sqrt(3) * abs(dbeta) }
          SR0 = AR;              {SR0 = 1/Sqrt(3) * abs(dbeta) -
abs(dalpha)}
          if ge jump Sqr3dalpha_lt_dbeta_same_sign;
            AR = MR1 + AY0;        { AR = 2/Sqrt(3) * abs(dbeta)  }

            DM(d2) = AR;  { d2 = 2/Sqrt(3) * abs(dbeta) }
            AR = - SR0;
            DM(d1) = AR; { d1 = abs(dalpha) - 1/Sqrt(3) * abs(dbeta) }

            AF = pass AX0;
            if lt jump sector_5;
            sector_2:          { dalpha > 0, dbeta > 0 }
                MY0 = 1;             { SecNumber = 1 }
                AR = 2;
                jump continue;

            sector_5:          { dalpha < 0, dbeta < 0 }
                MY0 = 4;             { SecNumber = 4 }
                AR = 5;
                jump continue;

          Sqr3dalpha_lt_dbeta_same_sign:
            AR = MR1 + AF;      { AR = 1/Sqrt(3) * abs(dbeta) +
abs(dalpha) }

            DM(d1) = AR;        { d1 = 1/Sqrt(3) * abs(dbeta) +
abs(dalpha) }
            DM(d2) = SR0;
                       { d2 = 1/Sqrt(3) * abs(dbeta) - abs(dalpha) }
            AF = pass AX1;
            if lt jump sector_6;
            sector_3:          { dalpha > 0, dbeta > 0 }
                MY0 = 2;
```

```
                AR = 3;
                jump continue;

            sector_6:          { dalpha < 0, dbeta < 0 }
                MY0 = 5;             { SecNumber = 5 }
                AR = 4;
                jump continue;

      not_same_sign:
          AF = abs AX0;        { AF = abs(dalpha) }
          AR = abs AX1;        { AR = abs(dbeta)  }
          MY0 = DM(OneOverSqrThree);
          MR = AR * MY0 (rnd);       { MR1 = 1/Sqrt(3) * abs(dbeta) }
          AR = MR1 - AF, AY0 = MR1;  { AY0 = 1/Sqrt(3) * abs(dbeta) }
          SR0 = AR;              {SR0 = 1/Sqrt(3) * abs(dbeta) -
abs(dalpha)}
          if ge jump Sqr3dalpha_lt_dbeta_not_same_sig;
            AR = MR1 + AY0;        { AR  = 2/Sqrt(3) * abs(dbeta) }

            DM(d1) = AR;        { d2 = 2/Sqrt(3) * abs(dbeta) }
            AR = - SR0;
            DM(d2) = AR; { d1 = abs(dalpha) - 1/Sqrt(3) * abs(dbeta) }

            AF = pass AX0;
            if lt jump sector_4;
            sector_1:             { dalpha > 0, dbeta < 0 }
                MY0 = 0;                { SecNumber = 0 }
                AR = 6;
                jump continue;

            sector_4:             { dalpha < 0, dbeta > 0 }
                MY0 = 3;                { SecNumber = 3 }
                AR = 1;
                jump continue;

          Sqr3dalpha_lt_dbeta_not_same_sig:
            AR = MR1 + AF;      { AR = 1/Sqrt(3) * abs(dbeta) +
abs(dalpha) }

            DM(d2) = AR;
                      { d2 = 1/Sqrt(3) * abs(dbeta) + abs(dalpha) }
            DM(d1) = SR0;       { d1 = 1/Sqrt(3) * abs(dbeta) -
abs(dalpha) }

            AF = pass AX1;
            if ge jump sector_3a;
            sector_6a:             { dalpha > 0, dbeta < 0 }
                MY0 = 5;                { SecNumber = 5 }
                AR = 4;
                jump continue;

            sector_3a:         { dalpha < 0, dbeta > 0 }
                MY0 = 2;             { SecNumber = 2 }
                AR = 3;
continue:

      SR0 = DM(sector);
      DM(old_sector) = SR0;
      DM(sector) = MY0;
      AF = pass AR;
      AF = AF - 1;
```

```
                                        { Duty cycle output }
{      SI = DM(d1);
       SR = lshift SI by -8 (hi);
       AR = SR1 xor AY1;
       DM(dac1) = AR;

       SI = DM(d2);
       SR = lshift SI by -8 (hi);
       AR = SR1 xor AY1;
       DM(dac2) = AR;
}
                                { Sector number output }
{      MX0 = 25;
       MR = MX0 * MY0 (rnd);
       DM(dac1) = MR0;
}

{      AF = pass 0;
       AX0 = 0x02;
       AR = DM(Ic);
       AR = pass AR;
       if ge AF = AF + 1;

       AR = DM(Ib);
       AR = pass AR;
       if ge AF = AX0 + AF;

       AX0 = 4;
       AR = DM(Ialpha);
       AR = pass AR;
       if ge AF = AX0 + AF;

       AF = AF - 1;
}
                                { Case output }
{      AR = pass AF;
       MY1 = 25;
       MR = AR * MY1 (rnd);
       DM(dac2) = MR0;
}

sequence:
       MX0 = 6;
       MR = MX0 * MY0 (ss);        { MY0 = SecNumber }
       SR = lshift MR0 by -1 (lo); { Fractional adjust is not needed }
       AR = SR0 + AF;              { entry_point = 6 * SecNumber + offset
}
       MR0 = AR;             { MR0 stores offset }

       DM(offset) = AR;


       AY0 = ^case_table;
       AR = AR + AY0;
       I7 = AR;
       AX1 = DM(I7,M7);      { AX1 stores value from case_table }
       AY0 = 0x04;
       AR = AX1 and AY0;
       if eq jump check_waveform_type;
```

```
swap: AR = DM(d1);
      AX0 = DM(d2);
      DM(d1) = AX0;
      DM(d2) = AR;

check_waveform_type:
      AY0 = 0x03;
      AR = AX1 and AY0;    { AX1 = entry from the case table }
      DM(type) = AR;         { AR = waveform_type }

                          { Waveform type output}
{      MY1 = 25;
      MR = AR * MY1 (rnd);
      DM(dac3) = MR0;
}
      AY0 = 0x01;
      AR = AR - AY0;
      if eq jump waveform_one;
      AR = AR - AY0;
      if eq jump waveform_two;

{ ********** Load time registers *********}

waveform_zero:
      AY0 = MINIMUM_LENGTH;

      SI = DM(d2);
      SR = lshift SI by -7 (hi); { d2out = d2 >> 7 }
      AR = SR1 - AY0;
      if ge jump zero1;
          SR1 = AY0;
zero1:
      DM(time1) = SR1;
      AF = pass SR1;

      AR = CHARGE;
      DM(time2) = AR;
      AF = AR + AF;

      AR = DISCHARGE;
      DM(time3) = AR;
      AF = AR + AF;

      SI = DM(d1);
      SR = lshift SI by -7 (hi); { d1out = d1 >> 7 }
      AR = SR1 - AY0;
      if ge jump zero2;
          SR1 = AY0;
zero2:
      AF = SR1 + AF;
      AR = 255;
      AR = AR - AF;
      DM(time4) = AR;

      jump label1;

waveform_one:
      AY0 = MINIMUM_LENGTH;

      AR = CHARGE;
      DM(time1) = AR;
```

```
      AR = DISCHARGE;
      DM(time2) = AR;

      SI = DM(d1);
      SR = lshift SI by -7 (hi); { d1out = d1 >> 7 }
      AR = SR1 - AY0;
      if ge jump one1;
          SR1 = AY0;
one1:
      DM(time3) = SR1;
      SI = DM(d2);
      SR = lshift SI by -7 (hi); { d2out = d2 >> 7 }
      AR = SR1 - AY0;
      if ge jump one2;
          SR1 = AY0;
one2:
      DM(time4) = SR1;
      jump label1;

waveform_two:
      AY0 = MINIMUM_LENGTH;

      SI = DM(d1);
      SR = lshift SI by -7 (hi); { d1out = d1 >> 7 }
      AR = SR1 - AY0;
      if ge jump two1;
         SR1 = AY0;
two1:
      DM(time4) = SR1;
      AF = pass SR1;

      AR = CHARGE;
      DM(time2) = AR;
      AF = AR + AF;

      AR = DISCHARGE;
      DM(time3) = AR;
      AF = AR + AF;

      SI = DM(d2);
      SR = lshift SI by -7 (hi); { d2out = d2 >> 7 }
      AR = SR1 - AY0;
      if ge jump two2;
          SR1 = AY0;
two2:
      AF = SR1 + AF;
      AR = 255;
      AR = AR - AF;

      SR = lshift AR by -1 (hi);  { Devide by 2 }
      DM(time1) = SR1;

      jump label1;

{ ****** Calculating the entry point to the sector's switching table
****** }
{ ********************** and the output waveform type
******************** }
```

```
label1:
      AY0 = ^entry_points;
      MR0 = DM(offset);
      AR = MR0 + AY0;     { MR0 = offset }
      I7 = AR;     { I7 = take entry point from table of entry points }
      AR = PM(I7,M7);
      I7 = AR;     { I7 = entry point to switching sequence }

{ *************** Coping data into the altera buffer  ************* }

      AX0 = DM(old_sector);
      AY0 = DM(sector);
      AR = AX0 - AY0;
      if eq jump no_sector_change;

sector_change:
      AX0 = DM(invert_flag);
      AR = pass AX0;
      if eq jump do_not_invert_it;
    invert_it:
          AR = PM(I7,M7);
          AR = not AR;
          DM(data) = AR;
          AR = PM(I7,M7);
          AR = not AR;
          DM(data) = AR;
          AR = PM(I7,M7);
          AR = not AR;
          DM(data) = AR;
          AR = PM(I7,M7);
          AR = not AR;
          DM(data) = AR;
          AX0 = PM(I7,M7);
          AX1 = PM(I7,M7);
          DM(data) = AX1;
          nop;
          DM(data) = AX0;

          SI = DM(d2);
          SR = lshift SI by -7 (hi); { d2out = d1 >> 7 }
          AR = SR1 - AY0;
          if ge jump two1a;
             SR1 = AY0;
    two1a:
          DM(time4) = SR1;

          jump old_cycle;

    do_not_invert_it:
          AR = PM(I7,M7);
          DM(data) = AR;
          AR = PM(I7,M7);
          DM(data) = AR;
          AR = PM(I7,M7);
          DM(data) = AR;
          AR = PM(I7,M7);
          DM(data) = AR;
          AR = PM(I7,M7);
          DM(data) = AR;
          AR = PM(I7,M7);
          DM(data) = AR;
          AR = PM(I7,M7);
          DM(data) = AR;
```

```
          jump old_cycle;

no_sector_change:
      AX0 = DM(invert_flag);
      AR = not AX0;
      DM(invert_flag) = AR;
      if eq jump do_not_invert_it;
      jump invert_it;

{ ********************** wait new switching cycle ****************** }

old_cycle:
      AR = DM(new_cycle_flag);
      AR = pass AR;
      if eq jump old_cycle;

      jump begin;

shutdown:
      AR = 1;              { turn off drivers }
      DM(reset_altera) = AR;
      set flag_out;
      idle;
      jump shutdown;


{ ************************************************************** }
{ ****************** The end of the program ******************** }
{ ************************************************************** }

{ Coeficients for digital filters }

{ Buterrworth, fc = 600 Hz, sampling 16 kHz, order 2 }

.var/pm/ram/circ/abs=0x02f0 b600[3];
.init        b600:
             0x006800, 0x00d000, 0x006800;
                              { b3=0.0032, b2=0.0068, b1=0.0032}

.var/pm/ram/circ/abs=0x02f4 a600[2];
.init        a600:
             0x93a700, 0x6ab500; { a3=-0.8645, a1=1+0.8337}

{ Buterrworth, fc = 1000 Hz, sampling 16 kHz, order 2 }

.var/pm/ram/circ/abs=0x02f8 b1000[3];
.init        b1000:
             0x011300, 0x022600, 0x011300;
                              { b3=0.0084, b2=0.0169, b1=0.0084}

.var/pm/ram/circ/abs=0x02fc a1000[2];
.init        a1000:
             0x9f0b00, 0x5ca400; { a3=-0.7575, a1=1+0.7238}


{************************ Program RAM tables
*****************************}

.var/pm/ram/abs=0x02fe  entry_points[36];
.init entry_points:     ^sector_one_case_one,
                  ^sector_one_case_two,
```

```
                    ^sector_one_case_three,
                    ^sector_one_case_four,
                    ^sector_one_case_five,
                    ^sector_one_case_six,

                    ^sector_two_case_one,
                    ^sector_two_case_two,
                    ^sector_two_case_three,
                    ^sector_two_case_four,
                    ^sector_two_case_five,
                    ^sector_two_case_six,

                    ^sector_three_case_one,
                    ^sector_three_case_two,
                    ^sector_three_case_three,
                    ^sector_three_case_four,
                    ^sector_three_case_five,
                    ^sector_three_case_six,

                    ^sector_four_case_one,
                    ^sector_four_case_two,
                    ^sector_four_case_three,
                    ^sector_four_case_four,
                    ^sector_four_case_five,
                    ^sector_four_case_six,

                    ^sector_five_case_one,
                    ^sector_five_case_two,
                    ^sector_five_case_three,
                    ^sector_five_case_four,
                    ^sector_five_case_five,
                    ^sector_five_case_six,

                    ^sector_six_case_one,
                    ^sector_six_case_two,
                    ^sector_six_case_three,
                    ^sector_six_case_four,
                    ^sector_six_case_five,
                    ^sector_six_case_six;


.var/pm/ram/abs=0x0328  sector_one_case_one[6]; { Type 2 }
.init             sector_one_case_one:
                                  b#001101000000000,
                                  b#101101000000000,
                                  b#110010100000000,
                                  b#010010100000000,
                                  b#010100100000000,
                                  b#001100100000000;

.var/pm/ram/abs=0x032e  sector_one_case_two[6]; { Type 1 }
.init             sector_one_case_two:
                                  b#110011000000000,
                                  b#101100100000000,
                                  b#001100100000000,
                                  b#010100100000000,
                                  b#010101000000000,
                                  b#000000000000000;

.var/pm/ram/abs=0x0334  sector_one_case_three[6]; { Type 1 }
.init             sector_one_case_three:
```

```
                                    b#101011000000000,
                                    b#110100100000000,
                                    b#010100100000000,
                                    b#001100100000000,
                                    b#001010100000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x033a  sector_one_case_four[6]; { Type 0 }
.init             sector_one_case_four:
                                    b#010100100000000,
                                    b#110100100000000,
                                    b#101011000000000,
                                    b#001010100000000,
                                    b#001100100000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x0340  sector_one_case_five[6]; { Type 0 }
.init             sector_one_case_five:
                                    b#001100100000000,
                                    b#101100100000000,
                                    b#110011000000000,
                                    b#010101000000000,
                                    b#010100100000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x0346  sector_one_case_six[6]; { Type 2 * }
.init             sector_one_case_six:
                                    b#001010100000000,
                                              b#001010100000000,
                                              b#001010100000000,
                                              b#001010100000000,
                                    b#001100100000000,
                                    b#010100100000000;

.var/pm/ram/abs=0x034c  sector_two_case_one[6]; { Type 1 }
.init             sector_two_case_one:
                                    b#101101000000000,
                                    b#110010100000000,
                                    b#010010100000000,
                                    b#010100100000000,
                                    b#010101000000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x0352  sector_two_case_two[6];  { Type 2 * }
.init             sector_two_case_two:
                                    b#010101000000000,
                                              b#010101000000000,
                                              b#010101000000000,
                                              b#010101000000000,
                                    b#010100100000000,
                                    b#010010100000000;

.var/pm/ram/abs=0x0358  sector_two_case_three[6]; { Type 1 }
.init             sector_two_case_three:
                                    b#101011000000000,
                                    b#110100100000000,
                                    b#010100100000000,
                                    b#010010100000000,
                                    b#001010100000000,
                                    b#000000000000000;
```

```
.var/pm/ram/abs=0x035e  sector_two_case_four[6];  { Type 0 }
.init             sector_two_case_four:
                                    b#010100100000000,
                                    b#110100100000000,
                                    b#101011000000000,
                                    b#001010100000000,
                                    b#010010100000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x0364  sector_two_case_five[6];  { Type 2 }
.init             sector_two_case_five:
                                    b#001100100000000,
                                    b#101100100000000,
                                    b#110011000000000,
                                    b#010011000000000,
                                    b#010010100000000,
                                    b#010100100000000;

.var/pm/ram/abs=0x036a  sector_two_case_six[6]; { Type 0 }
.init             sector_two_case_six:
                                    b#010010100000000,
                                    b#110010100000000,
                                    b#101101000000000,
                                    b#010101000000000,
                                    b#010100100000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x0370 sector_three_case_one[6];  { Type 1 }
.init              sector_three_case_one:
                                    b#101101000000000,
                                    b#110010100000000,
                                    b#010010100000000,
                                    b#010011000000000,
                                    b#010101000000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x0376 sector_three_case_two[6];  { Type 0 }
.init              sector_three_case_two:
                                    b#010011000000000,
                                    b#110011000000000,
                                    b#101100100000000,
                                    b#001010100000000,
                                    b#010010100000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x037c sector_three_case_three[6]; { Type 2 * }
.init              sector_three_case_three:
                                    b#001010100000000,
                                              b#001010100000000,
                                              b#001010100000000,
                                              b#001010100000000,
                                    b#010010100000000,
                                    b#010011000000000;

.var/pm/ram/abs=0x0382 sector_three_case_four[6];   { Type 2 }
.init              sector_three_case_four:
                                    b#010100100000000,
                                    b#110100100000000,
                                    b#101011000000000,
                                    b#001011000000000,
                                    b#010011000000000,
```

```
                                    b#010010100000000;

.var/pm/ram/abs=0x0388 sector_three_case_five[6]; { Type 1 }
.init               sector_three_case_five:
                                    b#101100100000000,
                                    b#110011000000000,
                                    b#010011000000000,
                                    b#010010100000000,
                                    b#001010100000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x038e sector_three_case_six[6];  { Type 0 }
.init               sector_three_case_six:
                                    b#010010100000000,
                                    b#110010100000000,
                                    b#101101000000000,
                                    b#010101000000000,
                                    b#010011000000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x0394 sector_four_case_one[6]; { Type 2 * }
.init               sector_four_case_one:
                                    b#010101000000000,
                                              b#010101000000000,
                                              b#010101000000000,
                                              b#010101000000000,
                                    b#010011000000000,
                                    b#001011000000000;

.var/pm/ram/abs=0x039a sector_four_case_two[6]; { Type 0 }
.init               sector_four_case_two:
                                    b#010011000000000,
                                    b#110011000000000,
                                    b#101100100000000,
                                    b#001010100000000,
                                    b#001011000000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x03a0 sector_four_case_three[6]; { Type 0 }
.init               sector_four_case_three:
                                    b#001011000000000,
                                    b#101011000000000,
                                    b#110100100000000,
                                    b#010101000000000,
                                    b#010011000000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x03a6 sector_four_case_four[6]; { Type 1 }
.init               sector_four_case_four:
                                    b#110100100000000,
                                    b#101011000000000,
                                    b#001011000000000,
                                    b#010011000000000,
                                    b#010101000000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x03ac sector_four_case_five[6]; { Type 1 }
.init               sector_four_case_five:
                                    b#101100100000000,
                                    b#110011000000000,
                                    b#010011000000000,
```

```
                                    b#001011000000000,
                                    b#001010100000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x03b2 sector_four_case_six[6]; { Type 2 }
.init               sector_four_case_six:
                                    b#010010100000000,
                                    b#110010100000000,
                                    b#101101000000000,
                                    b#001101000000000,
                                    b#001011000000000,
                                    b#010011000000000;

.var/pm/ram/abs=0x03b8 sector_five_case_one[6]; { Type 0 }
.init               sector_five_case_one:
                                    b#001101000000000,
                                    b#101101000000000,
                                    b#110010100000000,
                                    b#001010100000000,
                                    b#001011000000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x03be sector_five_case_two[6]; { Type 2 }
.init               sector_five_case_two:
                                    b#010011000000000,
                                    b#110011000000000,
                                    b#101100100000000,
                                    b#001100100000000,
                                    b#001101000000000,
                                    b#001011000000000;

.var/pm/ram/abs=0x03c4 sector_five_case_three[6]; { Type 0 }
.init               sector_five_case_three:
                                    b#001011000000000,
                                    b#101011000000000,
                                    b#110100100000000,
                                    b#010101000000000,
                                    b#001101000000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x03ca sector_five_case_four[6]; { Type 1 }
.init               sector_five_case_four:
                                    b#110100100000000,
                                    b#101011000000000,
                                    b#001011000000000,
                                    b#001101000000000,
                                    b#010101000000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x03d0 sector_five_case_five[6];  { Type 2 * }
.init               sector_five_case_five:
                                    b#001010100000000,
                                              b#001010100000000,
                                              b#001010100000000,
                                              b#001010100000000,
                                    b#001011000000000,
                                    b#001101000000000;

.var/pm/ram/abs=0x03d6 sector_five_case_six[6];  { Type 1 }
.init               sector_five_case_six:
                                    b#110010100000000,
```

```
                                    b#101101000000000,
                                    b#001101000000000,
                                    b#001011000000000,
                                    b#001010100000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x03dc  sector_six_case_one[6]; { Type 0 }
.init             sector_six_case_one:
                                    b#001101000000000,
                                    b#101101000000000,
                                    b#110010100000000,
                                    b#001010100000000,
                                    b#001100100000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x03e2  sector_six_case_two[6]; { Type 1 }
.init             sector_six_case_two:
                                    b#110011000000000,
                                    b#101100100000000,
                                    b#001100100000000,
                                    b#001101000000000,
                                    b#010101000000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x03e8  sector_six_case_three[6]; { Type 2 }
.init             sector_six_case_three:
                                    b#001011000000000,
                                    b#101011000000000,
                                    b#110100100000000,
                                    b#010100100000000,
                                    b#001100100000000,
                                    b#001101000000000;

.var/pm/ram/abs=0x03ee  sector_six_case_four[6];  { Type 2 * }
.init             sector_six_case_four:
                                    b#010101000000000,
                                              b#010101000000000,
                                              b#010101000000000,
                                              b#010101000000000,
                                    b#001101000000000,
                                    b#001100100000000;

.var/pm/ram/abs=0x03f4  sector_six_case_five[6]; { Type 0 }
.init             sector_six_case_five:
                                    b#001100100000000,
                                    b#101100100000000,
                                    b#110011000000000,
                                    b#010101000000000,
                                    b#001101000000000,
                                    b#000000000000000;

.var/pm/ram/abs=0x03fa  sector_six_case_six[6]; { Type 1 }
.init             sector_six_case_six:
                                    b#110010100000000,
                                    b#101101000000000,
                                    b#001101000000000,
                                    b#001100100000000,
                                    b#001010100000000,
                                    b#000000000000000;
.endmod;
```