ภาคผนวก ข.

โปรแกรมควบคุมระบบ

****โปรแกรมส่วนรับค่า(MCS-51)**********

```
// u3.c
#pragma code
#include <reg51.h>
#include "define.h"
//---------------------------------------
#define KEYMINDEX       0x0a
#define KEYSTEP         0x0b
#define KEYPSHIFT       0x0c
#define KEYT 0x0d
#define KEYDTIME        0x0e
#define KEYENTER        0x0f
#define CURSON          0x0f
#define CURSOFF         0x0c
// Global variable----------------------
uchar           dM3,dM2,dM1,dM0,dT3,dT2,dT1,dT0;
uchar           dS2,dS1,dS0,dP2,dP1,dP0,dD2,dD1,dD0;
uint            dM,dT,dS,dP,dD;
// Constants---------------------------
// String for LCD display -------------
//              "0123456789ABCDEF";
code uchar LINE0[] = "M=     T=     ";
code uchar LINE1[] = "S=   P=   D=    ";
// M(modulation index) 10-1000 ;(1.0-100.0)%
// T(for cal. freq.) 1-1000 ;(3k-3M)Hz
// S(freq. step) 10-500 ;(1.0-50.0)Hz
// P(phase shift) 0-360 degree
// D(dead time) 1-160 ;(0.1-16.0)us
//---------------------------------------
void main(void)
{               uchar k;
                initial();
                while(1)
                {               while((k=scankey())!=0xff)
                                {               if(k==0x0a) { peep1(); fill_m(); }
                                                else if(k==0x0b) { peep1(); fill_s(); }
                                                else if(k==0x0c) { peep1(); fill_p(); }
                                                else if(k==0x0d) { peep1(); fill_t(); }
                                                else if(k==0x0e) { peep1(); fill_d(); }
                                }//end while((k=scankey()!=0xff)
                }//end while(1)
}

//---------------------------------------
void initial(void)
{               peep1(); lcdini(); lcdwi(CURSOFF); lcdlc(0x80,LINE0);
lcdlc(0xc0,LINE1);
}
//---------------------------------------
void fill_m(void)               //10-1000 (1.0-100.0)%
{               uchar k,b3,b2,b1,b0;
                uint bint;
                lcdwi(0x82); lcdwd(' '); lcdwd(' '); lcdwd(' '); lcdwd(' ');
lcdwi(0x82); lcdwi(CURSON);
                while((k=scankey())>1); b3=k; shownum(b3); peep1();
                while((k=scankey())>9); b2=k; shownum(b2); peep1();
                while((k=scankey())>9); b1=k; shownum(b1); peep1();
                while((k=scankey())>9); b0=k; shownum(b0); peep1();
                bint=(b3*1000)+(b2*100)+(b1*10)+b0;
                if((bint<=1000)&&(bint>=10)) { dM3=b3; dM2=b2; dM1=b1;
dM0=b0; txdata(0x00,bint); }
                else { lcdwi(0x82); shownum(dM3); shownum(dM2);
shownum(dM1); shownum(dM0); peep2(); }
                lcdwi(CURSOFF);
}
//---------------------------------------
void fill_t(void)               //1-1000 (3k-3M)Hz
{               uchar k,b3,b2,b1,b0;
                uint bint;
                lcdwi(0x8a); lcdwd(' '); lcdwd(' '); lcdwd(' ');
lcdwi(0x8a); lcdwi(CURSON);
                while((k=scankey())>1); b3=k; shownum(b3); peep1();
                while((k=scankey())>9); b2=k; shownum(b2); peep1();
                while((k=scankey())>9); b1=k; shownum(b1); peep1();
                while((k=scankey())>9); b0=k; shownum(b0); peep1();
                bint=(b3*1000)+(b2*100)+(b1*10)+b0;
                if((bint<=1000)&&(bint>=1)) { dT3=b3; dT2=b2; dT1=b1;
dT0=b0; txdata(0x03,bint); }
                else { lcdwi(0x8a); shownum(dT3); shownum(dT2);
shownum(dT1); shownum(dT0); peep2(); }
                lcdwi(CURSOFF);
}
//---------------------------------------
void fill_s(void)               //10-500 (1.0-50.0)Hz
{               uchar k,b2,b1,b0;
                uint bint;
                lcdwi(0xc2); lcdwd(' '); lcdwd(' '); lcdwd(' '); lcdwi(0xc2);
lcdwi(CURSON);
                while((k=scankey())>5); b2=k; shownum(b2); peep1();
                while((k=scankey())>9); b1=k; shownum(b1); peep1();
                while((k=scankey())>9); b0=k; shownum(b0); peep1();
                bint=(b2*100)+(b1*10)+b0;
                if((bint<=500)&&(bint>=10)) { dS2=b2; dS1=b1; dS0=b0;
txdata(0x01,bint); }
                else { lcdwi(0xc2); shownum(dS2); shownum(dS1);
shownum(dS0); peep2(); }
                lcdwi(CURSOFF);
}
//---------------------------------------
void fill_p(void)               //0-360 degree
{               uchar k,b2,b1,b0;
                uint bint;
                lcdwi(0xc7); lcdwd(' '); lcdwd(' '); lcdwd(' '); lcdwi(0xc7);
lcdwi(CURSON);
                while((k=scankey())>3); b2=k; shownum(b2); peep1();
                while((k=scankey())>9); b1=k; shownum(b1); peep1();
                while((k=scankey())>9); b0=k; shownum(b0); peep1();
                bint=(b2*100)+(b1*10)+b0;
                if(bint<=360) { dP2=b2; dP1=b1; dP0=b0; txdata(0x02,bint); }
                else { lcdwi(0xc7); shownum(dP2); shownum(dP1);
shownum(dP0); peep2(); }
                lcdwi(CURSOFF);
}
//---------------------------------------
void fill_d(void)               //1-160 (0.1-16.0)us
{               uchar k,b2,b1,b0;
                uint bint;
                lcdwi(0xcc); lcdwd(' '); lcdwd(' '); lcdwd(' '); lcdwi(0xcc);
lcdwi(CURSON);
                while((k=scankey())>1); b2=k; shownum(b2); peep1();
                while((k=scankey())>9); b1=k; shownum(b1); peep1();
                while((k=scankey())>9); b0=k; shownum(b0); peep1();
```

```
                bint=(b2*100)+(b1*10)+b0;
                if((bint<=160)&&(bint>=1)) { dD2=b2; dD1=b1; dD0=b0;
txdata(0x04,bint); }
                else { lcdwi(0xcc); shownum(dD2); shownum(dD1);
shownum(dD0); peep2(); }
                lcdwi(CURSOFF);
}
//---------------------------------------
void txdata(uchar a,uint d)
{               uchar x;
                intbyte dd;
                dd.intx=d; P1=dd.bytex[1];
                x=dd.bytex[0]; P3=((a<<2)|(x&0xe3));
                ENFPGA=0; delaym(1); ENFPGA=1;
}
//---------------------------------------
```

```
// define.h
//---------------------------------------
typedef unsigned char uchar;
typedef unsigned int uint;
//---------------------------------------
typedef union
{               uint intx;
                uchar bytex[2];
} intbyte;
//---------------------------------------
// port declaration
// P0.0-P0.3 = LCD data(output)
// P0.4-P0.7 Key Column Input
sbit LCDRS = P2^0;          //output
sbit LCDE = P2^1;           //output
// P2.2-P2.3 reserve
sbit ROW0 = P2^7;           //Output to key
sbit ROW1 = P2^6;           //Output to key
sbit ROW2 = P2^5;           //Output to key
sbit ROW3 = P2^4;           //Output to key
// P1.0-P1.7 data low out to FPGA
// P3.0-P3.1 data high out to FPGA
sbit ADDR0 = P3^2;          //output to FPGA select Address
sbit ADDR1 = P3^3;          //output to FPGA select Address
sbit ADDR2 = P3^4;          //output to FPGA select Address
sbit ENFPGA = P3^5;         //output to FPGA Enable
sbit RPM = P3^6;            //input
sbit BUZ = P3^7;            //output
//---------------------------------------
// delay.c
void delayu(uint count);
void delaym(uint count);
//void delays(uint count);
void peep(void);
void peep1(void);
void peep2(void);
//---------------------------------------
```

```
// key.c
uchar scankey(void);
//---------------------------------------
```

```
// lcd.c
void lcdwi(uchar ins);
void lcdwd(uchar dat);
void lcdlc(uchar addr,uchar *msptr);
void lcdini(void);
void shownum(uchar y);
//void showbcd(uchar y);
//---------------------------------------
// u3.c
void initial(void);
void fill_m(void);
void fill_t(void);
void fill_s(void);
void fill_p(void);
void fill_d(void);
void txdata(uchar a,uint d);
//---------------------------------------
```

```
//---------------------------delay-part----------------
// delay.c
#include <reg51.h>
#include "define.h"
//for 89C51 @12MHz,12clk
//---------------------------------------
void delayu(uint count)
{               uchar x,y,z;
                x=count/240; y=(count%240)/2;
                while(x) { for(z=120;z>0;z--); }
                for(z=y;z>0;z--);
}
//---------------------------------------
void delaym(uint count)
{               uint x;
                uchar y;
                for(x=count;x>0;x--)
                {               for(y=250;y>0;y--); for(y=250;y>0;y--); }
}
//---------------------------------------
//void delays(uint count)
//{             uint x;
//              for(x=count;x>0;x--) { delaym(1000); }
//}
//---------------------------------------
void peep(void)             //@2.5kHz
{               uint x;
                for(x=0;x<400;x++) { BUZ=~BUZ; delayu(200); }
}
//---------------------------------------
void peep1(void)
{               peep(); delaym(200);
}
//---------------------------------------
void peep2(void)
{               peep(); delaym(100); peep();
}
//---------------------------------------
```

```
// lcd.c
#include <reg51.h>
#include <string.h>
#include "define.h"
//---------------------------------------
void lcdwi(uchar ins)
{       P0=(ins>>4)|(0xf0);
        LCDRS=0; LCDE=1; LCDE=0; delaym(2);  // lcd instruction
        P0=(ins)|(0xf0);
        LCDRS=0; LCDE=1; LCDE=0; delaym(2); // lcd instruction
}
//---------------------------------------
void lcdwd(uchar dat)
{       P0=(dat>>4)|(0xf0);
        LCDRS=1; LCDE=1; LCDE=0; delaym(2); // lcd data
        P0=(dat)|(0xf0);
        LCDRS=1; LCDE=1; LCDE=0; delaym(2); // lcd data
}
//---------------------------------------
void lcdlc(uchar addr,uchar *msptr)
{       uchar len;
        len=strlen(msptr); lcdwi(addr);
        while(len--) { lcdwd(*msptr); msptr++; }
}
//---------------------------------------
void lcdini(void)
{       LCDE = 0;
        lcdwi(0x33);   /* 8 bits */
        lcdwi(0x32);   /* 4 bits */
        lcdwi(0x28);   /* function set */
        lcdwi(0x06);   /* entry mode set */
        lcdwi(0x0f);   /* display on/off */
        lcdwi(0x01);   /* clear display */
        delaym(10);
}
//---------------------------------------
void shownum(uchar y)
{       uchar x;
        if(y<=0x09) x=y+'0'; else x=(y-0x0a+'A');
        lcdwd(x);
}
//---------------------------------------
//void showbcd(uchar y)
//{       uchar x;
//      x=y>>4; x=x&0x0f; shownum(x);
//      x=y&0x0f; shownum(x);
//}
//---------------------------------------
```

```
// key.c
#include <reg51.h>
#include "define.h"
//---------------------------------
// return 0xff if has no key press
uchar scankey(void)
{       uchar col;
        P2=0x7f; col=P0&0xf0; //row 0
        if(col != 0xf0)
                switch(col)
                {       case 0x70: return(0x01); break; //
                        case 0xb0: return(0x02); break; //
                        case 0xd0: return(0x03); break; //
                        case 0xe0: return(0x0a); break; //
                };
        P2=0xbf; col=P0&0xf0; //row 1
        if(col != 0xf0)
                switch(col)
                {       case 0x70: return(0x04); break; //
                        case 0xb0: return(0x05); break; //
                        case 0xd0: return(0x06); break; //
                        case 0xe0: return(0x0b); break; //
                };
        P2=0xdf; col=P0&0xf0; //row 2
        if(col != 0xf0)
                switch(col)
                {       case 0x70: return(0x07); break; //
                        case 0xb0: return(0x08); break; //
                        case 0xd0: return(0x09); break; //
                        case 0xe0: return(0x0c); break; //
                };
        P2=0xef; col=P0&0xf0; //row 3
        if(col != 0xf0)
                switch(col)
                {       case 0x70: return(0x0f); break; //
                        case 0xb0: return(0x00); break; //
                        case 0xd0: return(0x0e); break; //
                        case 0xe0: return(0x0d); break; //
                };
        return(0xff);   // no key press
}
//---------------------------------------
```

```
********โปรแกรมส่วนของ FPGA************
// Author
// Version 2
// Created 22-Jun-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================
// Top-level module without LCD user interface
// New system_vp2.v

module igbt_noLCD(clk ,reset, lag, a, b, c, x, y, z, dbg);

        input clk, lag, reset;
        //PWM output
        output [1:0] a,b,c;             //3phase inverter
        output [1:0] x,y,z;             //1phase inverter

        //debugging ports
        output [3:0] dbg;

        //system_vp interface
        wire        [7:0] dtime;
        wire        [8:0] step;
        wire        [8:0] pshift;
        wire        [9:0] T, m_index;
        wire [3:0] dbg;

        wire start;
        assign start = reset ;

        //clock divider
        wire clksrc, clkdiv2;
        wire clkdiv1024, blink;
        defparam div2.n = 1;
        div2e_n div2(.clkin(clk), .clkout(clkdiv2));
        defparam div1024.n = 10;
        div2e_n div1024(.clkin(clksrc), .clkout(clkdiv1024));
        defparam div1024x.n = 10;
        div2e_n div1024x(.clkin(clkdiv1024), .clkout(blink));

        //for probing with external logic analyser
        assign dbg[0] = a[1]; //ref pwm
        assign dbg[1] = z[1]; //shifted pwm
        assign dbg[2] = ~reset;
        assign dbg[3] = blink;


        /*
        //input and display control module
        input_ctrl InputControl(
        .clk(clkdiv1024),
        .row(row),
        .col(col),
        .rw(rw),
        .en(en),
        .rs(rs),
        .db(db),
        .enter(enter),
        .dtime(dtime),
        .step(step),
        .pshift(pshift),
        .T(T),
        .m_index(m_index)
        );
        */

        assign clksrc = clkdiv2;        //use 'clk' as sampling clk

        assign m_index = 1000;  //0-1000
        assign step = 3;                //0-500
        assign dtime = 6;               //0-100 * 1/fclk
        assign pshift = 96;   //0-360 degree
        assign T = 50;   //0-1000

        defparam System.sm1.VecIni = 3'b000;        //0
        defparam System.sm2.VecIni = 3'b000;        //pshift
        defparam System.sm3.VecIni = 3'b111;        //180

        //Useful Formula
================================================
        //Fc = Operating Frequency
        //Fs = Sampling Frequency
        //Fm = Motor Frequency (Speed)
        //
        //Fs = 1/Ts = 1/(2T*Tc) = Fc/(2T)
        //Fm = 1/Tm = 1/(512/step * 6 * Ts/2) = (step)/(1536*Ts) =
(Fs*step)/1536
        //Deadtime = dtime * 1/Fc
        //=======================================
==========================

        //core module
        system_vp2 System(
        .clk(clksrc),
        .dt_clk(clksrc),
        .cal_clk(clksrc),
        .reset(reset),
        .dtime(dtime),
        .m_index(m_index),
        .step(step),   //0..511
        .pshift(pshift),  //0..360 degree
        .lag(lag),    // phase shift lead => lag = '0' : lag => lag = '1'
        .T(T),
        .a(a),  .b(b),  .c(c),          //for 3 phase induction motor
        .x(x),  .y(y),  .z(z)           //for 1 phase induction motor
        );

        endmodule
```

```
// Author
// Version 1
// Created 09-Jun-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================


// Divided by 2^n
// n is parameterisable

module div2e_n(clkin, clkout);
   input clkin;
   output clkout;

parameter n = 8;

reg [n-1:0] divff;
always @ (posedge clkin)
                divff <= divff + 1;

assign clkout = divff[n-1];

endmodule
```

```
// Author
// Version 2
// Created 22-Jun-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================


// PWM for 1-phase and 3-phase inverter
// This system can vary phase from 0 - 360 degree with 1 degree resolution
// However, we can increase the resultion to 0.2 degree.
// Outputs are  for both 3-phase IGBT (abc) also 1-phase inverter (xyz)
// Differences:
// Change Phasc10n (36states 112slices) to Vector_gen (4 state 50 slices!)
// Can be added for 3 inverters simultaneously
// NEW---------
// New cnt_address with loop_num3 for 180 degree

module system_vp2(
                clk,
                dt_clk,
                cal_clk,
                reset,
                dtime,
                m_index,
                step,   //0..511
                pshift, //0..360 degree
                lag,    // phase shift lead => lag = '0' : lag => lag = '1'
                T,
                a, b, c,            //for 3 phase induction motor
                x, y, z             //for 1 phase induction motor

);

input           clk;
input           dt_clk;
input           cal_clk;
input           reset;
input           [7:0] dtime;
input           [8:0] step;
input           [8:0] pshift;
input           lag;
input           [9:0] T, m_index;
output    [1:0] a, b, c;
output    [1:0] x, y, z;

wire            check_wire1, check_wire2;
wire cal_wire;
wire [2:0] mod_wire;
wire [8:0] shift;
wire            [2:0] pwm_ref_wire, pwm_lag_wire, pwm_180_wire;
wire            [2:0] sector1, sector2, sector3;
wire            [9:0] ta1_wire,tb1_wire,to1_wire;
wire            [9:0] ta2_wire,tb2_wire,to2_wire;
wire            [9:0] MemA1, MemB1;
wire            [9:0] MemA2, MemB2;
```

```
wire [2:0] st_pc1, st_pc2;
wire [2:0] st_cal1, st_cal2;
wire [10:1] cnt1, cnt2, tmon1, tmon2;
wire [1:0] delta1, delta2;

//find sector and shifted degree of  shifted PWM
mod60 modulus_60(.indeg(pshift), .lag(lag), .mod(mod_wire), .shift(shift));

//access to the vlaues of vector stored in ROM
cnt_adr_vp2 mem_access( .clk(clk), .reset(reset), .sector_ini(mod_wire),
.step(step), .pshift(shift),
                .start(check_wire1 | check_wire2), .A1(MemA1), .B1(MemB1),
.A2(MemA2), .B2(MemB2),
                .ready(cal_wire), .loop_num1(sector1), .loop_num2(sector2),
.loop_num3(sector3));

//calculate ta, tb, tx of reference PWM
cal_time
                cal_time1(.clk(cal_clk),.cal(cal_wire),.m(m_index),.mema(MemA
1),.memb(MemB1),
                .t(T),.ta(ta1_wire),.tb(tb1_wire),.tx(to1_wire), .state(st_cal1));

//calculate ta, tb, tx of shifted PWM
cal_time
                cal_time2(.clk(cal_clk),.cal(cal_wire),.m(m_index),.mema(MemA
2),.memb(MemB2),
                .t(T),.ta(ta2_wire),.tb(tb2_wire),.tx(to2_wire), .state(st_cal2));

//generate a new vector of reference PWM regarding to ta, tb, tx
//defparam sm1.VecIni = 3'b000;
vector_gen      sm1(.clk(clk),.reset(reset),.sector(sector1),
                .ta(ta1_wire),.tb(tb1_wire),.tx(to1_wire),
                .check(check_wire1),.abc(pwm_ref_wire), .state(st_pc1),
.cnt(cnt1), .tmon(tmon1));

//generate a new vector of shifted PWM regarding to ta, tb, tx
//defparam sm2.VecIni = 3'b000;
vector_gen      sm2(.clk(clk),.reset(reset),.sector(sector2),
                .ta(ta2_wire),.tb(tb2_wire),.tx(to2_wire),
                .check(check_wire2),.abc(pwm_lag_wire), .state(st_pc2),
.cnt(cnt2), .tmon(tmon2));

//generate a new vector of 180 PWM regarding to ta, tb, tx
//defparam sm3.VecIni = 3'b111;
vector_gen      sm3(.clk(clk),.reset(reset),.sector(sector3),
                .ta(ta1_wire),.tb(tb1_wire),.tx(to1_wire),
                .check(),.abc(pwm_180_wire), .state(), .cnt(), .tmon());

//insert deadtime for 3 phase inverter
defparam three_phase.e = 1'b0; //falling edge deferring (for active low driver)
dtime6x         three_phase(.clk(dt_clk),
.ina1(~pwm_ref_wire[2]),.ina2(pwm_ref_wire[2]),  // 0 degree
.inb1(~pwm_ref_wire[1]),.inb2(pwm_ref_wire[1]),  // 120 degree
.inc1(~pwm_ref_wire[0]),.inc2(pwm_ref_wire[0]),  // 240 degree
.dtime(dtime),
.outa1(a[1]),.outa2(a[0]),
.outb1(b[1]),.outb2(b[0]),
.outc1(c[1]),.outc2(c[0]));

//insert deadtime for 1 phase inverter              (can be added for 3 inverters
simultaneously)
defparam one_phase.e = 1'b0; //falling edge deferring (for active low driver)
dtime6x         one_phase(.clk(dt_clk),
.ina1(~pwm_ref_wire[2]),.ina2(pwm_ref_wire[2]), // 0 degree (ref PWM)
.inb1(~pwm_lag_wire[2]),.inb2(pwm_lag_wire[2]), // pshift degree
.inc1(~pwm_180_wire[2]),.inc2(pwm_180_wire[2]), // 180 degree
.dtime(dtime),
.outa1(x[1]),.outa2(x[0]),
.outb1(y[1]),.outb2(y[0]),
.outc1(z[1]),.outc2(z[0]));

assign delta1 = x[1]-z[1];
assign delta2 = y[1]-z[1];

endmodule
```

```
// Author
// Version 1
// Created 09-Jun-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================


// Calculate Ta, Tb and Tx for ref PWM and shift PWM independently
// Shared multiplier to cal Ta, Tb and Tx respectively
// Make non-zero value for ta, tb to prevent missing VecOld stored at rising edge
of cnt_rst in vector_gen.v



module cal_time(clk, cal, m, t, mema, memb, ta, tb, tx, state);
  input clk, cal;
  input [9:0] m;
  input [9:0] t;
  input [9:0] mema;
  input [9:0] memb;
  output [9:0] ta;
  output [9:0] tb;
  output [9:0] tx;
  output [2:0] state;

  wire [19:0] mult_result;
  reg [9:0] mult_dataa;
  reg [9:0] mult_datab;
  reg [9:0] aff;
  reg [9:0] bff;
  reg [9:0] mtff;
  reg [2:0] ss__d;
  reg [2:0] ss__q;
  reg [9:0] a;
  reg [9:0] b;
  reg [9:0] mt;
  reg clka, clkb, clkmt, ldx;

  always @ (posedge clkmt)
   mtff <= mt;

  always @ (posedge clka)
  if (a==0)
                aff <= 1; //to prevent missing VecOld stored at rising edge of
cnt_rst in vector_gen.v
  else
   aff <= a;

  always @ (posedge clkb)
  if (b==0)
   bff <= 1; //to prevent missing VecOld stored at rising edge of cnt_rst in
vector_gen.v
  else
   bff <= b;

  always @ (posedge clk)
   ss__q <= ss__d;

// Sub Module Section
  cal_time_lpm_mult0  mult (.dataa(mult_dataa), .datab(mult_datab),
                .result(mult_result));

  always @(ss__q or m or t or mema or mtff or memb or mult_result
  or clk or cal) begin
                {clkmt, clka, clkb} = 3'b000;
                ldx = 1'b0;
   casex (ss__q)
   3'b000: begin
                  if (cal) begin
                    ss__d = 3'b001;
                  end else begin
                    ss__d = 3'b000;
                  end
                end
   3'b001: begin
                  mult_dataa <= m;
                  mult_datab <= t;
                  mt <= mult_result[19:10];
                  ss__d = 3'b010;
                  clkmt = !clk;
                end
   3'b010: begin
                  mult_dataa <= mtff;
                  mult_datab <= mema;
                  a <= mult_result[19:10];
                  ss__d = 3'b011;
                  clka = !clk;
                end
   3'b011: begin
                  mult_dataa <= mtff;
                  mult_datab <= memb;
                  b <= mult_result[19:10];
                  ss__d = 3'b100;
                  clkb = !clk;
                end
   3'b100: begin
                  if (!cal) begin
                    ss__d = 3'b000;
                  end
                end
   endcase
  end

   assign ta = aff;
   assign tb = bff;
   assign tx= t - (aff + bff);

  assign state = ss__q;

endmodule


module cal_time(clk, cal, m, t, mema, memb, ta, tb, tx, state);
  input clk, cal;
  input [9:0] m;
  input [9:0] t;
  input [9:0] mema;
```

```
  input [9:0] memb;
  output [9:0] ta;
  output [9:0] tb;
  output [9:0] tx;
  output [2:0] state;

  wire [19:0] mult_result;
  reg [9:0] mult_dataa;
  reg [9:0] mult_datab;
  reg [9:0] aff;
  reg [9:0] bff;
  reg [9:0] mtff;
  reg [2:0] ss__d;
  reg [2:0] ss__q;
  reg [9:0] a;
  reg [9:0] b;
  reg [9:0] mt;
  reg clka, clkb, clkmt, ldx;

  always @ (posedge clkmt)
   mtff <= mt;

  always @ (posedge clka)
  if (a==0)
                aff <= 1; //to prevent missing VecOld stored at rising edge of
cnt_rst in vector_gen.v
  else
   aff <= a;

  always @ (posedge clkb)
  if (b==0)
   bff <= 1; //to prevent missing VecOld stored at rising edge of cnt_rst in
vector_gen.v
  else
   bff <= b;

  always @ (posedge clk)
   ss__q <= ss__d;

// Sub Module Section
  cal_time_lpm_mult0  mult (.dataa(mult_dataa), .datab(mult_datab),
                .result(mult_result));

  always @(ss__q or m or t or mema or mtff or memb or mult_result
  or clk or cal) begin
                {clkmt, clka, clkb} = 3'b000;
                ldx = 1'b0;
    casex (ss__q)
    3'b000: begin
                  if (cal) begin
                    ss__d = 3'b001;
                  end else begin
                    ss__d = 3'b000;
                  end
                end
    3'b001: begin
                  mult_dataa <= m;
                  mult_datab <= t;
                  mt <= mult_result[19:10];
                  ss__d = 3'b010;
                  clkmt = !clk;
                end
    3'b010: begin
                  mult_dataa <= mtff;
                  mult_datab <= mema;
                  a <= mult_result[19:10];
                  ss__d = 3'b011;
                  clka = !clk;
                end
    3'b011: begin
                  mult_dataa <= mtff;
                  mult_datab <= memb;
                  b <= mult_result[19:10];
                  ss__d = 3'b100;
                  clkb = !clk;
                end
    3'b100: begin
                  if (!cal) begin
                    ss__d = 3'b000;
                  end
                end
    endcase
  end

  assign ta = aff;
  assign tb = bff;
  assign tx= t - (aff + bff);

  assign state = ss__q;

endmodule
```

```
// Author
// Version 1
// Created 28-Apr-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================
// Calculate Time Ta, Tb, Tx
// Referenced from module: cal_time



module cal_time_lpm_mult0(dataa, datab, result);
  input [9:0] dataa;
  input [9:0] datab;
  output [19:0] result;

  wire [19:0] res;



// Start of original equations
  assign res = dataa * datab;
  assign result = res;
endmodule
```

```
// Author
// Version 2
// Created 22-Jun-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================


// Acess data in ROMs for ref PWM and shift PWM independently
// Use 2 ROMS synchronise with posedge of clk signal
// Multiplex adrr to access memory
// Modified to be used in system_vp1.v
// that has initial values of "sector" and "phase shift" loaded when 'reset'
// NEW----------
// Access mem for 0, pshift, 180 (loop_num3)

module cnt_adr_vp2( clk, reset, sector_ini, step, pshift, start, A1, B1, A2, B2,
ready, loop_num1, loop_num2, loop_num3);
  input clk, reset;
  input [2:0] sector_ini;
  input [8:0] step, pshift;
  input start;
  output ready;
  output [9:0] A1, B1, A2, B2;
  output [2:0] loop_num1, loop_num2, loop_num3;

  reg ld1, ld2; //for loading rom outputs of two diff addresses (ref signal and
shifted signal)
  reg ready;
  wire lp_cnt1, lp_cnt2;
  reg [2:0] loop_num1, loop_num2, loop_num3;
  reg [1:0] ss__d;
  reg [1:0] ss__q;
  reg [8:0] adr;
  reg [8:0] adr1;
  reg [8:0] adr2;
  reg [9:0] A1, B1, A2, B2;
  wire [9:0] A, B;

  assign state =  ss__q;

                always @(posedge clk)
                if (reset)
                ss__q <= 0;
                else
                ss__q <= ss__d;

  always @(ss__q or start or adr or adr1 or adr2 or step or pshift)
                begin
                ld1 <= 1'b0;
                ld2 <= 1'b0;
                ready <= 1'b0;
   case (ss__q)
   2'b00: begin
                    ready <= 1'b0;
                    if (start) begin
```

```
            ss__d <= 2'b01;
          end else begin
            ss__d <= 2'b00;
          end
        end
        2'b01: begin
            ld1 <= 1'b1;
                        ready <= 1'b0;
                        ss__d <= 2'b10;
        end
        2'b10: begin
                        ld2 <= 1'b1;
                        ready <= 1'b0;
                        ss__d <= 2'b11;
        end
        2'b11: begin
                        ready <= 1'b1;
                        if (!start)
            ss__d <= 2'b00;
                        else
                        ss__d <= 2'b11;
        end
        endcase
        end


        always @ (posedge start  or posedge reset)
        begin
          if (reset)  //load ini value
          begin adr1 <= 0;
                adr2 <= pshift;
          end
          else              //regular count
          begin
                        adr1 <= adr1 + step;
                        adr2 <= adr2 + step;
          end
        end


//mux adr1 and adr2 to roms' address
        always @ ( ld1 or  ld2)
        begin
          if (ld1) adr <= adr1;
          else if (ld2) adr <= adr2;
        end

        //read data from rom at falling edge of clk
        syn_rom_0       rom0(.clk(~clk), .address(adr),.q(A));
        syn_rom_1       rom1(.clk(~clk), .address(adr),.q(B));
        //rom0          rom0(.addr(adr), .clk(clk), .dout(A));
        //rom1          rom1(.addr(adr), .clk(clk), .dout(B));

        //load read data at falling edge of ld1 and ld2
        always @ (negedge ld1)
        begin
                        A1 <= A;
                        B1 <= B;          end
        always @ (negedge ld2)
        begin
                        A2 <= A;
                        B2 <= B;          end

        assign lp_cnt1 = ~adr1[8];
        assign lp_cnt2 = ~adr2[8];

        always @ (posedge lp_cnt1 or posedge reset )
        begin
                        if ( reset) loop_num1 <= 0;     //reference PWM
starts with sector_ini 0
                        else
                         if (loop_num1 < 5)
                                loop_num1 <= loop_num1 + 1;
                         else
                                loop_num1 <= 0;
        end

        always @ (posedge lp_cnt2 or posedge reset )
        begin
                        if (reset) loop_num2 <= sector_ini;  //shifted
PWM starts with 'sector_ini'
                        else
                         if (loop_num2 < 5)
                                loop_num2 <= loop_num2 + 1;
                         else
                                loop_num2 <= 0;
        end

        always @ (posedge lp_cnt1 or posedge reset )
        begin
                        if ( reset) loop_num3 <= 3;     //180 PWM starts
with sector_ini 3
                        else
                         if (loop_num3 < 5)
                                loop_num3 <= loop_num3 + 1;
                         else
                                loop_num3 <= 0;
        end

endmodule
```

```
// Author
// Version 1
// Created 28-Apr-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================
// Attention! use negedge of clk

module syn_rom_0(clk,address,q);
input clk;
input [8:0] address;
output [9:0] q;
reg [9:0] q;


always @(posedge clk)
begin
   case (address)
0   :   q   =   766   ;
1   :   q   =   765   ;
2   :   q   =   764   ;
3   :   q   =   763   ;
4   :   q   =   762   ;
5   :   q   =   761   ;
6   :   q   =   760   ;
7   :   q   =   759   ;
8   :   q   =   758   ;
9   :   q   =   758   ;
10  :   q   =   757   ;
11  :   q   =   756   ;
12  :   q   =   755   ;
13  :   q   =   754   ;
14  :   q   =   753   ;
15  :   q   =   752   ;
16  :   q   =   751   ;
17  :   q   =   750   ;
18  :   q   =   749   ;
19  :   q   =   748   ;
20  :   q   =   747   ;
21  :   q   =   746   ;
22  :   q   =   745   ;
23  :   q   =   744   ;
24  :   q   =   743   ;
25  :   q   =   742   ;
26  :   q   =   741   ;
27  :   q   =   740   ;
28  :   q   =   739   ;
29  :   q   =   738   ;
30  :   q   =   737   ;
31  :   q   =   736   ;
32  :   q   =   735   ;
33  :   q   =   734   ;
34  :   q   =   733   ;
35  :   q   =   732   ;
36  :   q   =   731   ;
37  :   q   =   730   ;
38  :   q   =   729   ;
39  :   q   =   728   ;
40  :   q   =   727   ;
41  :   q   =   726   ;
42  :   q   =   725   ;
43  :   q   =   724   ;
44  :   q   =   723   ;
45  :   q   =   722   ;
46  :   q   =   721   ;
47  :   q   =   720   ;
48  :   q   =   719   ;
49  :   q   =   718   ;
50  :   q   =   716   ;
51  :   q   =   715   ;
52  :   q   =   714   ;
53  :   q   =   713   ;
54  :   q   =   712   ;
55  :   q   =   711   ;
56  :   q   =   710   ;
57  :   q   =   709   ;
58  :   q   =   708   ;
59  :   q   =   707   ;
60  :   q   =   706   ;
61  :   q   =   705   ;
62  :   q   =   703   ;
63  :   q   =   702   ;
64  :   q   =   701   ;
65  :   q   =   700   ;
66  :   q   =   699   ;
67  :   q   =   698   ;
68  :   q   =   697   ;
69  :   q   =   696   ;
70  :   q   =   695   ;
71  :   q   =   693   ;
72  :   q   =   692   ;
73  :   q   =   691   ;
74  :   q   =   690   ;
75  :   q   =   689   ;
76  :   q   =   688   ;
77  :   q   =   687   ;
78  :   q   =   685   ;
79  :   q   =   684   ;
80  :   q   =   683   ;
81  :   q   =   682   ;
82  :   q   =   681   ;
83  :   q   =   680   ;
84  :   q   =   679   ;
85  :   q   =   677   ;
86  :   q   =   676   ;
87  :   q   =   675   ;
88  :   q   =   674   ;
89  :   q   =   673   ;
90  :   q   =   672   ;
91  :   q   =   670   ;
92  :   q   =   669   ;
93  :   q   =   668   ;
94  :   q   =   667   ;
95  :   q   =   666   ;
96  :   q   =   664   ;
97  :   q   =   663   ;
```

98 : q = 662 ; 159 : q = 584 ;
99 : q = 661 ; 160 : q = 582 ;
100 : q = 660 ; 161 : q = 581 ;
101 : q = 658 ; 162 : q = 580 ;
102 : q = 657 ; 163 : q = 578 ;
103 : q = 656 ; 164 : q = 577 ;
104 : q = 655 ; 165 : q = 575 ;
105 : q = 654 ; 166 : q = 574 ;
106 : q = 652 ; 167 : q = 573 ;
107 : q = 651 ; 168 : q = 571 ;
108 : q = 650 ; 169 : q = 570 ;
109 : q = 649 ; 170 : q = 569 ;
110 : q = 647 ; 171 : q = 567 ;
111 : q = 646 ; 172 : q = 566 ;
112 : q = 645 ; 173 : q = 564 ;
113 : q = 644 ; 174 : q = 563 ;
114 : q = 642 ; 175 : q = 562 ;
115 : q = 641 ; 176 : q = 560 ;
116 : q = 640 ; 177 : q = 559 ;
117 : q = 639 ; 178 : q = 557 ;
118 : q = 637 ; 179 : q = 556 ;
119 : q = 636 ; 180 : q = 555 ;
120 : q = 635 ; 181 : q = 553 ;
121 : q = 634 ; 182 : q = 552 ;
122 : q = 632 ; 183 : q = 550 ;
123 : q = 631 ; 184 : q = 549 ;
124 : q = 630 ; 185 : q = 547 ;
125 : q = 629 ; 186 : q = 546 ;
126 : q = 627 ; 187 : q = 545 ;
127 : q = 626 ; 188 : q = 543 ;
128 : q = 625 ; 189 : q = 542 ;
129 : q = 623 ; 190 : q = 540 ;
130 : q = 622 ; 191 : q = 539 ;
131 : q = 621 ; 192 : q = 537 ;
132 : q = 620 ; 193 : q = 536 ;
133 : q = 618 ; 194 : q = 535 ;
134 : q = 617 ; 195 : q = 533 ;
135 : q = 616 ; 196 : q = 532 ;
136 : q = 614 ; 197 : q = 530 ;
137 : q = 613 ; 198 : q = 529 ;
138 : q = 612 ; 199 : q = 527 ;
139 : q = 610 ; 200 : q = 526 ;
140 : q = 609 ; 201 : q = 524 ;
141 : q = 608 ; 202 : q = 523 ;
142 : q = 606 ; 203 : q = 521 ;
143 : q = 605 ; 204 : q = 520 ;
144 : q = 604 ; 205 : q = 518 ;
145 : q = 602 ; 206 : q = 517 ;
146 : q = 601 ; 207 : q = 516 ;
147 : q = 600 ; 208 : q = 514 ;
148 : q = 598 ; 209 : q = 513 ;
149 : q = 597 ; 210 : q = 511 ;
150 : q = 596 ; 211 : q = 510 ;
151 : q = 594 ; 212 : q = 508 ;
152 : q = 593 ; 213 : q = 507 ;
153 : q = 592 ; 214 : q = 505 ;
154 : q = 590 ; 215 : q = 504 ;
155 : q = 589 ; 216 : q = 502 ;
156 : q = 588 ; 217 : q = 501 ;
157 : q = 586 ; 218 : q = 499 ;
158 : q = 585 ; 219 : q = 498 ;

220 : q = 496 ; 281 : q = 401 ;
221 : q = 495 ; 282 : q = 399 ;
222 : q = 493 ; 283 : q = 398 ;
223 : q = 492 ; 284 : q = 396 ;
224 : q = 490 ; 285 : q = 395 ;
225 : q = 489 ; 286 : q = 393 ;
226 : q = 487 ; 287 : q = 391 ;
227 : q = 486 ; 288 : q = 390 ;
228 : q = 484 ; 289 : q = 388 ;
229 : q = 483 ; 290 : q = 386 ;
230 : q = 481 ; 291 : q = 385 ;
231 : q = 480 ; 292 : q = 383 ;
232 : q = 478 ; 293 : q = 382 ;
233 : q = 477 ; 294 : q = 380 ;
234 : q = 475 ; 295 : q = 378 ;
235 : q = 473 ; 296 : q = 377 ;
236 : q = 472 ; 297 : q = 375 ;
237 : q = 470 ; 298 : q = 373 ;
238 : q = 469 ; 299 : q = 372 ;
239 : q = 467 ; 300 : q = 370 ;
240 : q = 466 ; 301 : q = 368 ;
241 : q = 464 ; 302 : q = 367 ;
242 : q = 463 ; 303 : q = 365 ;
243 : q = 461 ; 304 : q = 363 ;
244 : q = 460 ; 305 : q = 362 ;
245 : q = 458 ; 306 : q = 360 ;
246 : q = 457 ; 307 : q = 359 ;
247 : q = 455 ; 308 : q = 357 ;
248 : q = 453 ; 309 : q = 355 ;
249 : q = 452 ; 310 : q = 354 ;
250 : q = 450 ; 311 : q = 352 ;
251 : q = 449 ; 312 : q = 350 ;
252 : q = 447 ; 313 : q = 349 ;
253 : q = 446 ; 314 : q = 347 ;
254 : q = 444 ; 315 : q = 345 ;
255 : q = 442 ; 316 : q = 344 ;
256 : q = 441 ; 317 : q = 342 ;
257 : q = 439 ; 318 : q = 340 ;
258 : q = 438 ; 319 : q = 339 ;
259 : q = 436 ; 320 : q = 337 ;
260 : q = 435 ; 321 : q = 335 ;
261 : q = 433 ; 322 : q = 334 ;
262 : q = 431 ; 323 : q = 332 ;
263 : q = 430 ; 324 : q = 330 ;
264 : q = 428 ; 325 : q = 328 ;
265 : q = 427 ; 326 : q = 327 ;
266 : q = 425 ; 327 : q = 325 ;
267 : q = 424 ; 328 : q = 323 ;
268 : q = 422 ; 329 : q = 322 ;
269 : q = 420 ; 330 : q = 320 ;
270 : q = 419 ; 331 : q = 318 ;
271 : q = 417 ; 332 : q = 317 ;
272 : q = 416 ; 333 : q = 315 ;
273 : q = 414 ; 334 : q = 313 ;
274 : q = 412 ; 335 : q = 312 ;
275 : q = 411 ; 336 : q = 310 ;
276 : q = 409 ; 337 : q = 308 ;
277 : q = 408 ; 338 : q = 306 ;
278 : q = 406 ; 339 : q = 305 ;
279 : q = 404 ; 340 : q = 303 ;
280 : q = 403 ; 341 : q = 301 ;

342 : q = 300 ; 403 : q = 194 ;
343 : q = 298 ; 404 : q = 192 ;
344 : q = 296 ; 405 : q = 190 ;
345 : q = 295 ; 406 : q = 188 ;
346 : q = 293 ; 407 : q = 187 ;
347 : q = 291 ; 408 : q = 185 ;
348 : q = 289 ; 409 : q = 183 ;
349 : q = 288 ; 410 : q = 181 ;
350 : q = 286 ; 411 : q = 179 ;
351 : q = 284 ; 412 : q = 178 ;
352 : q = 283 ; 413 : q = 176 ;
353 : q = 281 ; 414 : q = 174 ;
354 : q = 279 ; 415 : q = 172 ;
355 : q = 277 ; 416 : q = 171 ;
356 : q = 276 ; 417 : q = 169 ;
357 : q = 274 ; 418 : q = 167 ;
358 : q = 272 ; 419 : q = 165 ;
359 : q = 271 ; 420 : q = 163 ;
360 : q = 269 ; 421 : q = 162 ;
361 : q = 267 ; 422 : q = 160 ;
362 : q = 265 ; 423 : q = 158 ;
363 : q = 264 ; 424 : q = 156 ;
364 : q = 262 ; 425 : q = 155 ;
365 : q = 260 ; 426 : q = 153 ;
366 : q = 258 ; 427 : q = 151 ;
367 : q = 257 ; 428 : q = 149 ;
368 : q = 255 ; 429 : q = 147 ;
369 : q = 253 ; 430 : q = 146 ;
370 : q = 251 ; 431 : q = 144 ;
371 : q = 250 ; 432 : q = 142 ;
372 : q = 248 ; 433 : q = 140 ;
373 : q = 246 ; 434 : q = 138 ;
374 : q = 245 ; 435 : q = 137 ;
375 : q = 243 ; 436 : q = 135 ;
376 : q = 241 ; 437 : q = 133 ;
377 : q = 239 ; 438 : q = 131 ;
378 : q = 238 ; 439 : q = 129 ;
379 : q = 236 ; 440 : q = 128 ;
380 : q = 234 ; 441 : q = 126 ;
381 : q = 232 ; 442 : q = 124 ;
382 : q = 231 ; 443 : q = 122 ;
383 : q = 229 ; 444 : q = 121 ;
384 : q = 227 ; 445 : q = 119 ;
385 : q = 225 ; 446 : q = 117 ;
386 : q = 224 ; 447 : q = 115 ;
387 : q = 222 ; 448 : q = 113 ;
388 : q = 220 ; 449 : q = 112 ;
389 : q = 218 ; 450 : q = 110 ;
390 : q = 217 ; 451 : q = 108 ;
391 : q = 215 ; 452 : q = 106 ;
392 : q = 213 ; 453 : q = 104 ;
393 : q = 211 ; 454 : q = 103 ;
394 : q = 209 ; 455 : q = 101 ;
395 : q = 208 ; 456 : q = 99 ;
396 : q = 206 ; 457 : q = 97 ;
397 : q = 204 ; 458 : q = 95 ;
398 : q = 202 ; 459 : q = 94 ;
399 : q = 201 ; 460 : q = 92 ;
400 : q = 199 ; 461 : q = 90 ;
401 : q = 197 ; 462 : q = 88 ;
402 : q = 195 ; 463 : q = 86 ;

```
464     :     q     =     85     ;
465     :     q     =     83     ;
466     :     q     =     81     ;
467     :     q     =     79     ;
468     :     q     =     77     ;
469     :     q     =     76     ;
470     :     q     =     74     ;
471     :     q     =     72     ;
472     :     q     =     70     ;
473     :     q     =     68     ;
474     :     q     =     66     ;
475     :     q     =     65     ;
476     :     q     =     63     ;
477     :     q     =     61     ;
478     :     q     =     59     ;
479     :     q     =     57     ;
480     :     q     =     56     ;
481     :     q     =     54     ;
482     :     q     =     52     ;
483     :     q     =     50     ;
484     :     q     =     48     ;
485     :     q     =     47     ;
486     :     q     =     45     ;
487     :     q     =     43     ;
488     :     q     =     41     ;
489     :     q     =     39     ;
490     :     q     =     38     ;
491     :     q     =     36     ;
492     :     q     =     34     ;
493     :     q     =     32     ;
494     :     q     =     30     ;
495     :     q     =     28     ;
496     :     q     =     27     ;
497     :     q     =     25     ;
498     :     q     =     23     ;
499     :     q     =     21     ;
500     :     q     =     19     ;
501     :     q     =     18     ;
502     :     q     =     16     ;
503     :     q     =     14     ;
504     :     q     =     12     ;
505     :     q     =     10     ;
506     :     q     =     9      ;
507     :     q     =     7      ;
508     :     q     =     5      ;
509     :     q     =     3      ;
510     :     q     =     1      ;
511     :     q     =     0      ;

  endcase
end


endmodule
```

```
// Author
// Version 1
// Created 28-Apr-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================
// Attention! use negedge of clk

module syn_rom_1(clk,address,q);
input clk;
input [8:0] address;
output [9:0] q;
reg [9:0] q;

always @(posedge clk)
begin
   case (address)
0   :   q   =   1   ;
1   :   q   =   3   ;
2   :   q   =   5   ;
3   :   q   =   7   ;
4   :   q   =   9   ;
5   :   q   =   10  ;
6   :   q   =   12  ;
7   :   q   =   14  ;
8   :   q   =   16  ;
9   :   q   =   18  ;
10  :   q   =   19  ;
11  :   q   =   21  ;
12  :   q   =   23  ;
13  :   q   =   25  ;
14  :   q   =   27  ;
15  :   q   =   28  ;
16  :   q   =   30  ;
17  :   q   =   32  ;
18  :   q   =   34  ;
19  :   q   =   36  ;
20  :   q   =   38  ;
21  :   q   =   39  ;
22  :   q   =   41  ;
23  :   q   =   43  ;
24  :   q   =   45  ;
25  :   q   =   47  ;
26  :   q   =   48  ;
27  :   q   =   50  ;
28  :   q   =   52  ;
29  :   q   =   54  ;
30  :   q   =   56  ;
31  :   q   =   57  ;
32  :   q   =   59  ;
33  :   q   =   61  ;
34  :   q   =   63  ;
35  :   q   =   65  ;
36  :   q   =   66  ;
37  :   q   =   68  ;
38  :   q   =   70  ;
39  :   q   =   72  ;
40  :   q   =   74  ;
41  :   q   =   76  ;
42  :   q   =   77  ;
43  :   q   =   79  ;
44  :   q   =   81  ;
45  :   q   =   83  ;
46  :   q   =   85  ;
47  :   q   =   86  ;
48  :   q   =   88  ;
49  :   q   =   90  ;
50  :   q   =   92  ;
51  :   q   =   94  ;
52  :   q   =   95  ;
53  :   q   =   97  ;
54  :   q   =   99  ;
55  :   q   =   101 ;
56  :   q   =   103 ;
57  :   q   =   104 ;
58  :   q   =   106 ;
59  :   q   =   108 ;
60  :   q   =   110 ;
61  :   q   =   112 ;
62  :   q   =   113 ;
63  :   q   =   115 ;
64  :   q   =   117 ;
65  :   q   =   119 ;
66  :   q   =   121 ;
67  :   q   =   122 ;
68  :   q   =   124 ;
69  :   q   =   126 ;
70  :   q   =   128 ;
71  :   q   =   129 ;
72  :   q   =   131 ;
73  :   q   =   133 ;
74  :   q   =   135 ;
75  :   q   =   137 ;
76  :   q   =   138 ;
77  :   q   =   140 ;
78  :   q   =   142 ;
79  :   q   =   144 ;
80  :   q   =   146 ;
81  :   q   =   147 ;
82  :   q   =   149 ;
83  :   q   =   151 ;
84  :   q   =   153 ;
85  :   q   =   155 ;
86  :   q   =   156 ;
87  :   q   =   158 ;
88  :   q   =   160 ;
89  :   q   =   162 ;
90  :   q   =   163 ;
91  :   q   =   165 ;
92  :   q   =   167 ;
93  :   q   =   169 ;
94  :   q   =   171 ;
95  :   q   =   172 ;
96  :   q   =   174 ;
97  :   q   =   176 ;
98  :   q   =   178 ;
```

99	:	q	=	179	;	160	:	q	=	286	;								
100	:	q	=	181	;	161	:	q	=	288	;								
101	:	q	=	183	;	162	:	q	=	289	;								
102	:	q	=	185	;	163	:	q	=	291	;								
103	:	q	=	187	;	164	:	q	=	293	;								
104	:	q	=	188	;	165	:	q	=	295	;								
105	:	q	=	190	;	166	:	q	=	296	;								
106	:	q	=	192	;	167	:	q	=	298	;								
107	:	q	=	194	;	168	:	q	=	300	;								
108	:	q	=	195	;	169	:	q	=	301	;								
109	:	q	=	197	;	170	:	q	=	303	;								
110	:	q	=	199	;	171	:	q	=	305	;								
111	:	q	=	201	;	172	:	q	=	306	;								
112	:	q	=	202	;	173	:	q	=	308	;								
113	:	q	=	204	;	174	:	q	=	310	;								
114	:	q	=	206	;	175	:	q	=	312	;								
115	:	q	=	208	;	176	:	q	=	313	;								
116	:	q	=	209	;	177	:	q	=	315	;								
117	:	q	=	211	;	178	:	q	=	317	;								
118	:	q	=	213	;	179	:	q	=	318	;								
119	:	q	=	215	;	180	:	q	=	320	;								
120	:	q	=	217	;	181	:	q	=	322	;								
121	:	q	=	218	;	182	:	q	=	323	;								
122	:	q	=	220	;	183	:	q	=	325	;								
123	:	q	=	222	;	184	:	q	=	327	;								
124	:	q	=	224	;	185	:	q	=	328	;								
125	:	q	=	225	;	186	:	q	=	330	;								
126	:	q	=	227	;	187	:	q	=	332	;								
127	:	q	=	229	;	188	:	q	=	334	;								
128	:	q	=	231	;	189	:	q	=	335	;								
129	:	q	=	232	;	190	:	q	=	337	;								
130	:	q	=	234	;	191	:	q	=	339	;								
131	:	q	=	236	;	192	:	q	=	340	;								
132	:	q	=	238	;	193	:	q	=	342	;								
133	:	q	=	239	;	194	:	q	=	344	;								
134	:	q	=	241	;	195	:	q	=	345	;								
135	:	q	=	243	;	196	:	q	=	347	;								
136	:	q	=	245	;	197	:	q	=	349	;								
137	:	q	=	246	;	198	:	q	=	350	;								
138	:	q	=	248	;	199	:	q	=	352	;								
139	:	q	=	250	;	200	:	q	=	354	;								
140	:	q	=	251	;	201	:	q	=	355	;								
141	:	q	=	253	;	202	:	q	=	357	;								
142	:	q	=	255	;	203	:	q	=	359	;								
143	:	q	=	257	;	204	:	q	=	360	;								
144	:	q	=	258	;	205	:	q	=	362	;								
145	:	q	=	260	;	206	:	q	=	363	;								
146	:	q	=	262	;	207	:	q	=	365	;								
147	:	q	=	264	;	208	:	q	=	367	;								
148	:	q	=	265;		209	:	q	=	368	;								
149	:	q	=	267	;	210	:	q	=	370	;								
150	:	q	=	269	;	211	:	q	=	372	;								
151	:	q	=	271	;	212	:	q	=	373	;								
152	:	q	=	272	;	213	:	q	=	375	;								
153	:	q	=	274	;	214	:	q	=	377	;								
154	:	q	=	276	;	215	:	q	=	378	;								
155	:	q	=	277	;	216	:	q	=	380	;								
156	:	q	=	279	;	217	:	q	=	382	;								
157	:	q	=	281	;	218	:	q	=	383	;								
158	:	q	=	283	;	219	:	q	=	385	;								
159	:	q	=	284	;	220	:	q	=	386	;								

221 : q = 388 ; 282 : q = 484 ;
222 : q = 390 ; 283 : q = 486 ;
223 : q = 391 ; 284 : q = 487 ;
224 : q = 393 ; 285 : q = 489 ;
225 : q = 395 ; 286 : q = 490 ;
226 : q = 396 ; 287 : q = 492 ;
227 : q = 398 ; 288 : q = 493 ;
228 : q = 399 ; 289 : q = 495 ;
229 : q = 401 ; 290 : q = 496 ;
230 : q = 403 ; 291 : q = 498 ;
231 : q = 404 ; 292 : q = 499 ;
232 : q = 406 ; 293 : q = 501 ;
233 : q = 408 ; 294 : q = 502 ;
234 : q = 409 ; 295 : q = 504 ;
235 : q = 411 ; 296 : q = 505 ;
236 : q = 412 ; 297 : q = 507 ;
237 : q = 414 ; 298 : q = 508 ;
238 : q = 416 ; 299 : q = 510 ;
239 : q = 417 ; 300 : q = 511 ;
240 : q = 419 ; 301 : q = 513 ;
241 : q = 420 ; 302 : q = 514 ;
242 : q = 422 ; 303 : q = 516 ;
243 : q = 424 ; 304 : q = 517 ;
244 : q = 425 ; 305 : q = 518 ;
245 : q = 427 ; 306 : q = 520 ;
246 : q = 428 ; 307 : q = 521 ;
247 : q = 430 ; 308 : q = 523 ;
248 : q = 431 ; 309 : q = 524 ;
249 : q = 433 ; 310 : q = 526 ;
250 : q = 435 ; 311 : q = 527 ;
251 : q = 436 ; 312 : q = 529 ;
252 : q = 438 ; 313 : q = 530 ;
253 : q = 439 ; 314 : q = 532 ;
254 : q = 441 ; 315 : q = 533 ;
255 : q = 442 ; 316 : q = 535 ;
256 : q = 444 ; 317 : q = 536 ;
257 : q = 446 ; 318 : q = 537 ;
258 : q = 447 ; 319 : q = 539 ;
259 : q = 449 ; 320 : q = 540 ;
260 : q = 450 ; 321 : q = 542 ;
261 : q = 452 ; 322 : q = 543 ;
262 : q = 453 ; 323 : q = 545 ;
263 : q = 455 ; 324 : q = 546 ;
264 : q = 457 ; 325 : q = 547 ;
265 : q = 458 ; 326 : q = 549 ;
266 : q = 460 ; 327 : q = 550 ;
267 : q = 461 ; 328 : q = 552 ;
268 : q = 463 ; 329 : q = 553 ;
269 : q = 464 ; 330 : q = 555 ;
270 : q = 466 ; 331 : q = 556 ;
271 : q = 467 ; 332 : q = 557 ;
272 : q = 469 ; 333 : q = 559 ;
273 : q = 470 ; 334 : q = 560 ;
274 : q = 472 ; 335 : q = 562 ;
275 : q = 473 ; 336 : q = 563 ;
276 : q = 475 ; 337 : q = 564 ;
277 : q = 477 ; 338 : q = 566 ;
278 : q = 478 ; 339 : q = 567 ;
279 : q = 480 ; 340 : q = 569 ;
280 : q = 481 ; 341 : q = 570 ;
281 : q = 483 ; 342 : q = 571 ;

343 : q = 573 ; 404 : q = 652 ;
344 : q = 574 ; 405 : q = 654 ;
345 : q = 575 ; 406 : q = 655 ;
346 : q = 577 ; 407 : q = 656 ;
347 : q = 578 ; 408 : q = 657 ;
348 : q = 580 ; 409 : q = 658 ;
349 : q = 581 ; 410 : q = 660 ;
350 : q = 582 ; 411 : q = 661 ;
351 : q = 584 ; 412 : q = 662 ;
352 : q = 585 ; 413 : q = 663 ;
353 : q = 586 ; 414 : q = 664 ;
354 : q = 588 ; 415 : q = 666 ;
355 : q = 589 ; 416 : q = 667 ;
356 : q = 590 ; 417 : q = 668 ;
357 : q = 592 ; 418 : q = 669 ;
358 : q = 593 ; 419 : q = 670 ;
359 : q = 594 ; 420 : q = 672 ;
360 : q = 596 ; 421 : q = 673 ;
361 : q = 597 ; 422 : q = 674 ;
362 : q = 598 ; 423 : q = 675 ;
363 : q = 600 ; 424 : q = 676 ;
364 : q = 601 ; 425 : q = 677 ;
365 : q = 602; 426 : q = 679 ;
366 : q = 604 ; 427 : q = 680 ;
367 : q = 605 ; 428 : q = 681 ;
368 : q = 606 ; 429 : q = 682 ;
369 : q = 608 ; 430 : q = 683 ;
370 : q = 609 ; 431 : q = 684 ;
371 : q = 610 ; 432 : q = 685 ;
372 : q = 612 ; 433 : q = 687 ;
373 : q = 613 ; 434 : q = 688 ;
374 : q = 614 ; 435 : q = 689 ;
375 : q = 616 ; 436 : q = 690 ;
376 : q = 617 ; 437 : q = 691 ;
377 : q = 618 ; 438 : q = 692 ;
378 : q = 620 ; 439 : q = 693 ;
379 : q = 621 ; 440 : q = 695 ;
380 : q = 622 ; 441 : q = 696 ;
381 : q = 623 ; 442 : q = 697 ;
382 : q = 625 ; 443 : q = 698 ;
383 : q = 626 ; 444 : q = 699 ;
384 : q = 627 ; 445 : q = 700 ;
385 : q = 629 ; 446 : q = 701 ;
386 : q = 630 ; 447 : q = 702 ;
387 : q = 631 ; 448 : q = 703 ;
388 : q = 632 ; 449 : q = 705 ;
389 : q = 634 ; 450 : q = 706 ;
390 : q = 635 ; 451 : q = 707 ;
391 : q = 636 ; 452 : q = 708 ;
392 : q = 637 ; 453 : q = 709 ;
393 : q = 639 ; 454 : q = 710 ;
394 : q = 640 ; 455 : q = 711 ;
395 : q = 641 ; 456 : q = 712 ;
396 : q = 642 ; 457 : q = 713 ;
397 : q = 644 ; 458 : q = 714 ;
398 : q = 645 ; 459 : q = 715 ;
399 : q = 646 ; 460 : q = 716 ;
400 : q = 647 ; 461 : q = 718 ;
401 : q = 649 ; 462 : q = 719 ;
402 : q = 650 ; 463 : q = 720 ;
403 : q = 651 ; 464 : q = 721 ;

```
465 : q = 722 ;
466 : q = 723 ;
467 : q = 724 ;
468 : q = 725 ;
469 : q = 726 ;
470 : q = 727 ;
471 : q = 728 ;
472 : q = 729 ;
473 : q = 730 ;
474 : q = 731 ;
475 : q = 732 ;
476 : q = 733 ;
477 : q = 734 ;
478 : q = 735 ;
479 : q = 736 ;
480 : q = 737 ;
481 : q = 738 ;
482 : q = 739 ;
483 : q = 740 ;
484 : q = 741 ;
485 : q = 742 ;
486 : q = 743 ;
487 : q = 744 ;
488 : q = 745 ;
489 : q = 746 ;
490 : q = 747 ;
491 : q = 748 ;
492 : q = 749 ;
493 : q = 750 ;
494 : q = 751 ;
495 : q = 752 ;
496 : q = 753 ;
497 : q = 754 ;
498 : q = 755 ;
499 : q = 756 ;
500 : q = 757 ;
501 : q = 758 ;
502 : q = 758 ;
503 : q = 759 ;
504 : q = 760 ;
505 : q = 761 ;
506 : q = 762 ;
507 : q = 763 ;
508 : q = 764 ;
509 : q = 765 ;
510 : q = 766 ;
511 : q = 767 ;
  endcase
end


endmodule
```

```
// Author
// Version 1
// Created 28-Apr-2005
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================
// top level of dead time inserter contains 6 'deadtime's

module dtime6x(
                ina1,
                clk,
                inc1,
                inc2,
                inb1,
                inb2,
                ina2,
                dtime,
                outa1,
                outc1,
                outc2,
                outb1,
                outb2,
                outa2
);

parameter e = 1'b1;

input           ina1;
input           clk;
input           inc1;
input           inc2;
input           inb1;
input           inb2;
input           ina2;
input           [8:1] dtime;
output          outa1;
output          outc1;
output          outc2;
output          outb1;
output          outb2;
output          outa2;

defparam b2v_1.f = e;
deadtime        b2v_1(.clk(clk),.in(ina1),.dtime(dtime),.out(outa1));
defparam b2v_2.f = e;
deadtime        b2v_2(.clk(clk),.in(ina2),.dtime(dtime),.out(outa2));
defparam b2v_3.f = e;
deadtime        b2v_3(.clk(clk),.in(inb1),.dtime(dtime),.out(outb1));
defparam b2v_4.f = e;
deadtime        b2v_4(.clk(clk),.in(inb2),.dtime(dtime),.out(outb2));
defparam b2v_5.f = e;
deadtime        b2v_5(.clk(clk),.in(inc1),.dtime(dtime),.out(outc1));
defparam b2v_6.f = e;
deadtime        b2v_6(.clk(clk),.in(inc2),.dtime(dtime),.out(outc2));
endmodule
```

```
// Author
// Version 1
// Created 28-Apr-2005

// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================


// insert deadtime by delaying rising edges of all signals
// determine values of dtime and has independent clk to count the delay

module deadtime(clk, dtime, in, out);
  parameter f = 1'b1;
  input clk;
  input [7:0] dtime;
  input in;
  output out;

  wire out_clk;
  wire [7:0] cnt;
  wire cnt1_clk_ctrl;
  reg out_d, out_q;
  reg [7:0] cnt_d;
  reg [7:0] cnt_q;


  assign out = out_q;
  always @(posedge out_clk)
    out_q <= out_d;

  always @(posedge cnt1_clk_ctrl)
    cnt_q <= cnt_d;

// Start of original equations
  assign out_clk = clk;
  assign cnt1_clk_ctrl = clk;


  always @(out_q or cnt_q or dtime or in) begin
    out_d = out_q;
    cnt_d = 8'b0000_0000;
    if (dtime == 8'b0000_0000) begin
                out_d = in;
    end else begin
                if (f)
                        casex (out_q)
                        1'b0: begin
                            if (in == 1'b1) begin
                                    cnt_d = cnt_q + 8'b0000_0001;
                            end
                            if (cnt_q == dtime) begin
                                    out_d = 1'b1;
                                    cnt_d = 8'b0000_0000;
                            end else begin
                                    out_d = 1'b0;
                            end
                        end
                        1'b1: begin
                            if (in == 1'b0) begin
                                    out_d = 1'b0;
                            end
                        end
                        endcase
                else
                        casex (out_q)
                        1'b1: begin
                            if (in == 1'b0) begin
                                    cnt_d = cnt_q + 8'b0000_0001;
                            end
                            if (cnt_q == dtime) begin
                                    out_d = 1'b0;
                                    cnt_d = 8'b0000_0000;
                            end else begin
                                    out_d = 1'b1;
                            end
                        end
                        1'b0: begin
                            if (in == 1'b1) begin
                                    out_d = 1'b1;
                            end
                        end
                        endcase
        end
    end
endmodule
```

```
// Author
// Version 1
// Created 28-Apr-2005

// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================


// find sector and shift degree which shifted PWN will be locate at the beginning
// sector = indeg mod 60
// phase shift = remainder

module mod60(indeg,lag,mod,shift);

input [8:0] indeg;
input lag;
output [2:0] mod;
output  [8:0] shift;

wire [8:0] degree;
reg [2:0] mod;
reg [5:0] rem;



assign degree = lag ? 360-indeg: indeg ;

always @ (degree)
begin
                if (degree > 360)                               begin
                                mod <= 3'bxxx;
                                rem <= 6'bxxxxxx;               end
                else if (degree >= 300)         begin
                                mod <= 5;
                                rem <= degree-300;              end
                else if (degree >= 240)         begin
                                mod <= 4;
                                rem <= degree-240;              end
                else if (degree >= 180)         begin
                                mod <= 3;
                                rem <= degree-180;              end
                else if (degree >= 120)         begin
                                mod <= 2;
                                rem <= degree-120;              end
                else if (degree >= 60)          begin
                                mod <= 1;
                                rem <= degree-60;               end
                else                                                    begin
                                mod <= 0;
                                rem <= degree;                  end
end

assign shift = (rem * 17) >> 1; // scaled by 512/60 = 8.5 = 17/2

endmodule
// Author P
// Version 2
// Created 09-Jun-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================


// new version of vector generator
// reduced from 36 state machine to only 5 state machines (2states at initial)
// capable to start at any sector and any phase shift
// make it own decision of the next vector to minimize switching activity

module vector_gen(cnt, ta, tb, tx, clk, sector, reset, abc, check, state, tmon);
parameter VecZ0 =               3'b000 ;
parameter VecZ1 =   3'b111 ;
parameter VecIni =  3'b000 ;
  input [10:1] ta;
  input [10:1] tb;
  input [10:1] tx;
  input clk;
  input [3:1] sector;
  input reset;
  output [3:1] abc;
  output check;
  output [3:1] state;
  output [10:1] cnt;
  output [10:1] tmon;

  reg check;
  reg [10:1] cnt;
  reg cnt_rst;
  reg [3:1] st_q, st_d;
  reg [3:1] abc, abc_st;
  reg [10:1] tZero, tOdd, tEven;
  reg [10:1] Tab, Tz;   //store period of ta, tb, tx for comparison in state
machine
  reg [3:1]  VecA, VecB, VecOdd, VecEven, VecAB, VecZ; //classify vector
to be odd or even (no of '1')
  reg [3:1]  VecOld;              //store previous value to select the next vector
with mim switching

  //debugging port
  assign state = st_q;
  assign tmon = Tab;

  //select vector A and B to be combined
  always @(sector)
  begin
                case (sector)
                                0 : begin VecA <= 3'b100; VecB <= 3'b110; end
                                1 : begin VecA <= 3'b110; VecB <= 3'b010; end
                                2 : begin VecA <= 3'b010; VecB <= 3'b011; end
                                3 : begin VecA <= 3'b011; VecB <= 3'b001; end
                                4 : begin VecA <= 3'b001; VecB <= 3'b101; end
                                5 : begin VecA <= 3'b101; VecB <= 3'b100; end
                                default : begin VecA <= 3'b000; VecB <=
3'b000; end
```

```
                endcase
end

//Categorise Odd or Even type when 'check' signal, for the next vector
always @ (check or VecA or VecB or ta or tb or tx)
begin
if (check) begin
                tZero <= tx;
                if (^VecA)
                 begin
                                VecOdd <= VecA;
                                VecEven <= VecB;
                                tOdd <= ta;
                                tEven <= tb;
                 end
                else
                 begin
                                VecOdd <= VecB;
                                VecEven <= VecA;
                                tOdd <= tb;
                                tEven <= ta;
                end
end
end

//store previous vector
always @ (posedge cnt_rst)
                VecOld <= abc;

//select the right vector to minimize switching
wire [3:1] NextVecAB, NextVecZ;
wire [10:1] NextTab;
assign NextVecAB =  ^VecOld ? VecEven : VecOdd;
assign NextTab = ^VecOld ? tEven : tOdd;
assign NextVecZ              =  ^VecOld ? VecZ0 : VecZ1;

//register the right vector
always @ (negedge cnt_rst)
begin
                VecAB <= NextVecAB;
                Tab <= NextTab;
                VecZ <= NextVecZ;
                Tz <= tZero;
end

//counter ta, tb or tx with reset
always @ (posedge clk)
if (cnt_rst) cnt <= 0;
else cnt <= cnt+1;

//state change with reset
=====================================================
always @(posedge clk )
  if (reset)
                st_q <= 0;
  else
                st_q <= st_d;

//state combinational functions
always @(st_q  or Tab or Tz or VecAB or VecZ or cnt )
begin
                cnt_rst <= 0;
                case (st_q)
                                0: begin          //initial state
                                  check <= 1'b0;
                                  abc_st <= 0;
                                  cnt_rst <= 1;
                                  st_d <= 7;
                                end
                                7: begin          //start with gen 'check' pulse
                                  check <= 1'b1;
                                  abc_st <= 0; //initial value
                                  if (cnt == 15) //waiting for calculating ta, tb, tx
                                  begin
                                   cnt_rst <= 1;
                                                st_d <= 1;       //change state
                                  end
                                  else
                                   st_d <= 7;    //same state
                                end
                1: begin       //period 1
                                  check <= 1'b0;
                                  abc_st <= VecAB;
                                  if (cnt == Tab)   //change state
                                  begin
                                    cnt_rst <= 1'b1;
                                    st_d <= 2;
                                  end
                                  else
                                    st_d <= 1; //same state
                                end
                                2: begin          //period 2
                                  check <= 1'b0;
                                  abc_st <= VecAB;
                                  if (cnt == Tab)   //change state
                                  begin
                                  cnt_rst <= 1'b1;
                                  st_d <= 3;
                                  end
                                  else
                                    st_d <= 2; //same state
                                end
                3: begin       //period 3
                                  check <= 1'b1;
                                  abc_st <= VecZ;
                                  if (cnt == Tz)  //change state
                                  begin
                                  cnt_rst <= 1'b1;
                                  st_d <= 1;
                                  end
                                  else
                                    st_d <= 3; //same state
                                end
                endcase

end

//sync output
always @ (posedge clk)
  abc <= abc_st;

endmodule
```