Accelerometer System

The ADXL202E accelerometer has a duty cycle modulated digital output for each of its two axes. As shown in Figure 4, a 0.1 µF bypass capacitor is used to limit the effect of noise in the power supply. The 125 kΩ resistor is used to set the output frequency to 1kHz.. The 0.1µF capacitors on the Xfilt and Yfilt pins set the bandwidth of each axis of the accelerometer to 50Hz.

The HC12 timing system is used to calculate the acceleration from the duty cycle of the two axes using the facts that a 50% duty cycle indicates 0g and the sensitivity is 12.5% change in duty cycle per g.

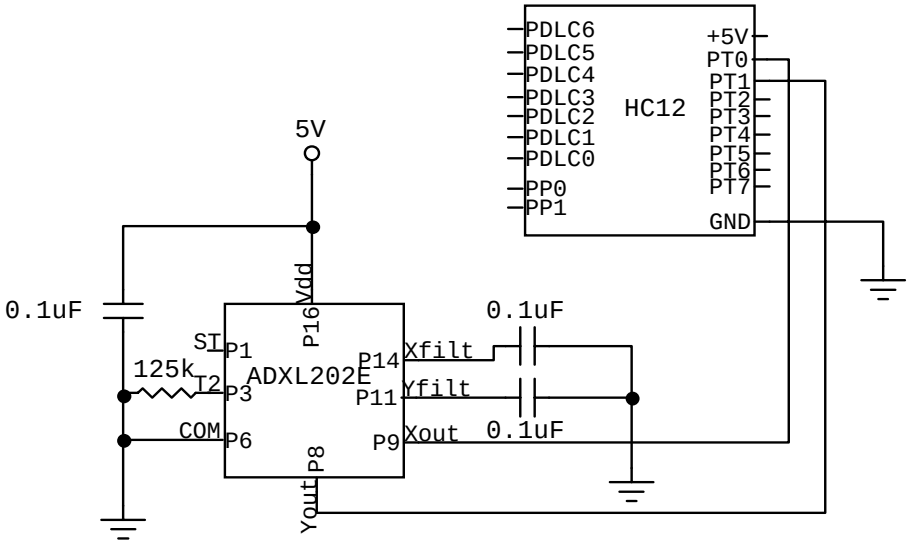

Figure 4 - Accelerometer Circuit

ADXL202E Accelerometer mounted in 16 pin chip carrier with hot glue.

Figure 5 - Accelerometer Electronics

Accelerometer Module
Interface with accelerometer
Data structure - accel:

Ta, Tb, Tc, Td = Detected edge times
calcount = number of times that the calculation has been performed
T1x, T1y, T2 = arrays containing high times and period for x and y channels
T1x_avg, T1y_avg, T2_avg = current average values of the above arrays
T1x_cal, T1y_cal, T2_cal = calibration values for high times and duty cycles
curr_accel_x, curr_accel_y = current acceleration values
K = calibration gain
AccelState = State of calculation (1 = on T0 LE, 2 = on T0 TE, 3 = on T1 LE, 4 = on T1 TE)
old_accel_x, old_accel_y = past acceleration values.

Void InitAccel(void)

Clear out accel data:
Everything in accel data structure = 0 except for AccelState = 1
Set T0 and T1 to input capture
Enable Timer SubSystem
Set to capture rising edge of T0
Turn on interrupts for input compares on T0 and T1
CalibrateAccel()

__interrupt HandleAccel(void)

Based on AccelState:
State 1:
Get Ta (Time of T0 Leading Edge)
Set next interrupt to trip on T0 TE
Set state to 2
Clear T0 and T1 interrupt flags
State 2:
Get Tb (Time of T0 Trailing Edge)
Set next interrupt to trip on T1 LE
Set state to 3
Clear T0 and T1 interrupt flags
State 3:
Get Tc (Time of T1 Leading Edge)
Set next interrupt to trip on T1 TE
Set state to 4
Clear T0 and T1 interrupt flags
State 4:
Get Td (Time of T1 Trailing Edge)
Calculate Acceleration
Set next interrupt to trip on T0 LE
Set state to 1
Increment Calibration Counter
Clear T0 and T1 interrupt flags

Static void CalculateAccel(void)

Updates curr_accel and old_accel in accel
T1x = Tb - Ta
T1y = Td - Tc
T2 = Td - Ta - T1y/2 - T1x/2
Add the new values to their respective arrays in accel
accel = K*(T1act - T1cal*T2act/T2cal) / T2_cal

Static void CalibrateAccel(void)
Wait for TSIZE iterations of accelerometer calculations
Set calibration values to the current accelerometer values.
K = 4*BSF

Static void AddToArray(int value, int *array, int *average)
Add value to the front of the array and recalculate the average.

Void GetAccel(int *x_accel, int *y_accel)
Puts the current values of x-accel and y-accel into the given addresses

Void GetOldAccel(int *x_accel, int *y_accel)
Puts the old values of x-accel and y-accel into the given addresses

---

External #defines used:
None

#defines created:
None

Internal #defines created:
TSIZE
BSF

PlayMusic
Module that runs the toy while in use.
Data structure - market_data:
session_ID
play_time
dead_time
play_count
frequency
fore_aft
left_right
data_sent
Module variable:
Unsigned char Played_Flag

Void InitMusic(void)
Init Analog to Digital Sytem
Clear out market_data
InitAccel
InitDAC (InitSPI)
Initialize timer system to 8ms rate

Void GetMarketData(unsigned char session_ID, unsigned char *play_time, unsigned char *dead_time, unsigned char *play_count, unsigned char *frequency, unsigned char fore_aft, unsigned char *left_right)
Return the values of the parameters as stored in the data structure

Accelerometer Module

```
/*****************************************************************************
 Module
        u:\_Teams\Thirteen\Accelerometer\Accelerometer.h

 Revision
        1.0

 Description
        header file for Accelerometer Interface

 History
 When           Who What/Why
 -------------- --- --------
5/14/01 18:15  jkk  first pass
*****************************************************************************/
void InitAccel( void );
__interrupt void HandleAccel( void );
void GetAccel( int *x_accel, int *y_accel );
void GetOldAccel( int *x_accel, int *y_accel );

/*------------------------------ End of file ------------------------------*/
//#define TESTACCEL
//#define PWMTEST
//#define DEBUG
//#define LIVETEST
//#define DEBUG2
//#define FAKE
#ifdef PWMTEST
   #include "pwm.h"
#endif

/*****************************************************************************
 Module
     u:\_Teams\Offbeat\ToyProgram\Accelerometer.c

 Revision
        1.0

 Description

       Interface with ADXL202E Accelerometer.


 Notes

 History
 When           Who What/Why
 -------------- --- --------
 5/14/01 16:15  jkk  first pass
*****************************************************************************/
/*---------------------------- Include Files ----------------------------*/
#include <stdio.h>
#include <dbprintf.h>
#include <me218_912.h>
#include "Accelerometer.h"

/*------------------------------Definitions------------------------------*/
#define TSIZE 3     // Length of Accel Data Buffer
#define BSF 0xFF    // Factor gives 8 bit resolution in accel (FF)

/*----------------------------Module Prototypes----------------------------*/
static void CalculateAccel( void );
static void AddToArray( int value, int *array, int *average );
static void CalibrateAccel( void );
static void AddToArray( int value, int *array, int *average );

/*---------------------------- Module Variables ---------------------------*/
struct Accel {
    int Ta, Tb, Tc, Td;                 // Detected Edge Times
    int calcount;                       // Calibration Count
    int T1x[TSIZE+1],T1y[TSIZE+1],T2[TSIZE+1];
                        // Arrays containing T1x, T1y, and T2 data
    int T1x_avg, T1y_avg, T2_avg;       // Current Average Values
    int T1x_cal, T1y_cal, T2_cal;       // Calibration Values
    int curr_accel_x, curr_accel_y;     // Current Acceleration Value
    int K;                             // Calibration Gain (K=2*T2cal*BSF)
    unsigned char AccelState;           // State of Calculation
                                        // 1 = On T0 LE
                                        // 2 = On T0 TE
                                        // 3 = On T1 LE
                                        // 4 = On T1 TE
         int old_accel_x, old_accel_y;          // Past Acceleration Value
};

static struct Accel accel;

/*------------------------------ Module Code ------------------------------*/
/*****************************************************************************
 Function
        InitAccel

 Parameters

 Returns

 Description
    Initializes timer ports 0 and 1 to deal with the accelerometer.
 * Sets Ports T0 and T1 to input capture
 * Enables timer sub-system
 * Set to capture rising edge of Port T0
 * Allow hardware interrupts for T0 and T1

 Notes
        None.

 Author
        Jonathan Karpick, 5/14/01  16:15
*****************************************************************************/
void InitAccel( void )
{
    int i;                              // increment variable

    accel.calcount = 0;
    accel.T1x_avg = 0;
    accel.T1y_avg = 0;
    accel.T2_avg = 0;
    accel.AccelState = 1;
    accel.T1x_cal = 0;
    accel.T1y_cal = 0;
    accel.T2_cal = 0;
    accel.Ta = 0;
    accel.Tb = 0;
    accel.Tc = 0;
    accel.Td = 0;
    accel.curr_accel_x = 0;
    accel.curr_accel_y = 0;
    accel.K=0;

    for ( i=0; i<=TSIZE; i++)
    {
        accel.T1x[i]=0;
        accel.T1y[i]=0;
        accel.T2[i]=0;
    }

    _H12TIOS = _H12TIOS & ~_H12_IOS0 & ~_H12_IOS1;  // Ports 0 and 1 to input capture
```

```
    _H12TSCR = _H12TSCR | _H12_TEN;   // Enable timer sub-system
    _H12TCTL4 = _H12TCTL4 & ~_H12_EDG0B & ~_H12_EDG1A & ~_H12_EDG1B;
    _H12TCTL4 = _H12TCTL4 | _H12_EDG0A;  // Capture on rising edge of Port 0
    _H12TMSK1 = _H12TMSK1 | _H12_C1I | _H12_C0I;  // Hardware interrupts for 0 and 1
    CalibrateAccel();
}

/*************************************************************************
 Function
        HandleAccel

 Parameters

 Returns

 Description
     Increment Calibration Count
     On T0 leading edge:
        Ta = Time
        Set input capture for T0 trailing edge
    On T0 trailing edge:
        Tb = Time
        Set input capture for T1 leading edge
    On T1 leading edge:
        Tc = Time
        Set input capture for T1 trailing edge
    On T1 trailing edge:
        Td = Time
        CalculateAccel
        Set input capture for T0 leading edge


 Notes
         None.

 Author
         Jonathan Karpick, 5/14/01  17:15
*************************************************************************/
__interrupt void HandleAccel( void )
{
  #ifdef DEBUG
    DB_printf("Handling Trigger, Accel State = %d, Count=%d/%d\n",
              accel.AccelState, accel.calcount, TSIZE);
  #endif

    switch ( accel.AccelState)
    {
        case 1 :                        // On T0 Leading Edge
            accel.Ta = _H12TC0;
           // _H12PORTDLC = _H12PORTDLC ^ BIT0HI;
            _H12TCTL4 = _H12TCTL4 & ~_H12_EDG0A & ~_H12_EDG1A & ~_H12_EDG1B;
            _H12TCTL4 = _H12TCTL4 | _H12_EDG0B;  // Capture on trailing edge of Port 0
            accel.AccelState = 2;
            _H12TFLG1 = _H12_C0F | _H12_C1F;
            break;
        case 2 :
            accel.Tb = _H12TC0;
          //  _H12PORTDLC = _H12PORTDLC ^ BIT0HI;
            _H12TCTL4 = _H12TCTL4 & ~_H12_EDG0A & ~_H12_EDG0B & ~_H12_EDG1B;
            _H12TCTL4 = _H12TCTL4 | _H12_EDG1A;  // Capture on leading edge of Port 1
            accel.AccelState = 3;
            _H12TFLG1 = _H12_C0F | _H12_C1F;
            break;
        case 3 :
            accel.Tc = _H12TC1;
         //   _H12PORTDLC = _H12PORTDLC ^ BIT0HI;
            _H12TCTL4 = _H12TCTL4 & ~_H12_EDG0A & ~_H12_EDG0B & ~_H12_EDG1A;
            _H12TCTL4 = _H12TCTL4 | _H12_EDG1B;  // Capture on trailing edge of Port 1
            accel.AccelState = 4;
            _H12TFLG1 = _H12_C0F | _H12_C1F;
            break;
        case 4 :
            accel.Td = _H12TC1;
          //   _H12PORTDLC = _H12PORTDLC ^ BIT0HI;
            CalculateAccel();
            _H12TCTL4 = _H12TCTL4 & ~_H12_EDG0B & ~_H12_EDG1A & ~_H12_EDG1B;
            _H12TCTL4 = _H12TCTL4 | _H12_EDG0A;  // Capture on trailing edge of Port 0
            accel.AccelState = 1;
            accel.calcount++;
            _H12TFLG1 = _H12_C0F | _H12_C1F;
            break;
    }
}

/*************************************************************************
 Function
        CalculateAccel

 Parameters

 Returns

 Description
 * Updates curr_accel in accel
 *  Saves curr_accel to old_accel
 *  T1x = Tb - Ta
 *  T1y = Td - Tc
 *  T2 = Td - Ta - T1y/2 - T1x/2
 *  Add the new values to their respective arrays in accel
 *  accel = KFACT*(T1act - T1cal*T2act/T2cal) / T2_cal

 Notes
        None.

 Author
        Jonathan Karpick, 5/14/01  17:30
*************************************************************************/
static void CalculateAccel( void )
{
    int T1x, T1y, T2;                   // Current values of T1x, T1y, and T2
    long temp;

         accel.old_accel_x = accel.curr_accel_x;
         accel.old_accel_y = accel.curr_accel_y;

    T1x = accel.Tb - accel.Ta;
    T1y = accel.Td - accel.Tc;
    T2  = accel.Td -accel.Ta - T1y/2 - T1x/2;
    AddToArray(T1x, accel.T1x, &(accel.T1x_avg));
    AddToArray(T1y, accel.T1y, &(accel.T1y_avg));
    AddToArray(T2, accel.T2, &(accel.T2_avg));

    temp = ((long)accel.T1x_cal*accel.T2_avg) / accel.T2_cal;
    accel.curr_accel_x = (int) ( (long) accel.K*(accel.T1x_avg-temp) / accel.T2_cal );

   #ifdef DEBUG
    DB_printf("  xaccel = %d\n",accel.curr_accel_x);
   #endif

   #ifdef DEBUG2
    DB_printf("  temp = %d, x_accel = %d\n",(int)temp,accel.curr_accel_x);
   #endif


    temp = ((long)accel.T1y_cal*accel.T2_avg) / accel.T2_cal;
    accel.curr_accel_y = (int) ( (long) accel.K*(accel.T1y_avg-temp) / accel.T2_avg );

   #ifdef DEBUG
    DB_printf("  yaccel = %d\n",accel.curr_accel_y);
   #endif
```

```
   #ifdef DEBUG2
    DB_printf("  temp = %d, y_accel = %d\n",(int)temp,accel.curr_accel_y);
   #endif

}

/*************************************************************************
 Function
         CalibrateAccel

 Parameters

 Returns

 Description
    Waits for TSIZE steps to be completed then sets calibration values

 Notes
         None.

 Author
         Jonathan Karpick, 5/14/01  18:50
*************************************************************************/
static void CalibrateAccel( void )
{
  #ifndef FAKE
    while ( accel.calcount <= (TSIZE+1)) {     // Wait for TSIZE steps
       #ifdef DEBUG2
         DB_printf("Calibrating...calcount=%d/%d\n",accel.calcount,TSIZE+1);
       #endif
    }

    accel.T1x_cal = accel.T1x_avg;
    accel.T1y_cal = accel.T1y_avg;
    accel.T2_cal = accel.T2_avg;

  #else
    DB_printf("Enter T1x Cal: \n");
    scanf("%d",&(accel.T1x_cal));
    DB_printf("Enter T1y Cal: \n");
    scanf("%d",&(accel.T1y_cal));
    DB_printf("Enter T2 Cal: \n");
    scanf("%d",&(accel.T2_cal));

  #endif

   // accel.K = 4 * accel.T2_cal * BSF;
    accel.K = 4*BSF;
}
```

```
/*************************************************************************
 Function
         AddToArray

 Parameters
         Value - The Value to Be Added
         Array - The array to which the value is added
         Average - The average value of the array.

 Returns

 Description

 Notes
 * Assumes all arrays have length TSIZE


 Author
         Jonathan Karpick, 5/14/01  17:45
*************************************************************************/
static void AddToArray( int value, int *array, int *average )
{
    int i;                              // increment variable
    int hold;                           // holder of the value being booted

    hold = array[TSIZE];

    for ( i=TSIZE; i>0; i--)
    {
        array[i]=array[i-1];
    }
    array[0]=value;

    *average = *average - hold/(TSIZE+1) + value/(TSIZE+1);

}

/*************************************************************************
 Function
         GetAccel

 Parameters
         x_accel - Pointer to place to put x-axis acceleration
                   y_accel - Pointer to place to put y-axis acceleration

 Returns

 Description
          Puts the values of the acceleration from the accel data structure in the
          respective places

 Notes


 Author
         Jonathan Karpick, 5/22/01  19:45
*************************************************************************/
void GetAccel( int *x_accel, int *y_accel )
{
          *x_accel = accel.curr_accel_x;
          *y_accel = accel.curr_accel_y;
}
```

```
/*************************************************************************
 Function
        GetOldAccel

 Parameters
        x_accel - Pointer to place to put the old x-axis acceleration
                  y_accel - Pointer to place to put the old y-axis acceleration

 Returns

 Description
         Puts the values of the old acceleration from the accel data structure
         in the respective places

 Notes


 Author
        Jonathan Karpick, 5/26/01  17:00
*************************************************************************/
void GetOldAccel( int *x_accel, int *y_accel )
{
         *x_accel = accel.old_accel_x;
         *y_accel = accel.old_accel_y;
}


/****************** TEST HARNESS ****************************/
#ifdef TESTACCEL
void main (void)
{

  #ifdef FAKE
     int i;
     disable();
  #endif

  #ifdef PWMTEST
    int i;
    int selection;
    unsigned short duty;
    _H12DDRDLC = _H12DDRP | BIT0HI;
    _H12PORTDLC = _H12PORTP & BIT0LO;
  #endif

     InitAccel();

  #ifdef FAKE
    CalibrateAccel();

       DB_printf("T1x = ");
       for ( i=0; i<=TSIZE; i++)
       {
            DB_printf("%d ",accel.T1x[i]);
       }
       DB_printf("\n");

       DB_printf("T1y = ");
       for ( i=0; i<=TSIZE; i++)
       {
            DB_printf("%d ",accel.T1y[i]);
       }
       DB_printf("\n");

       DB_printf("T2 = ");
       for ( i=0; i<=TSIZE; i++)
       {
            DB_printf("%d ",accel.T2[i]);
       }
       DB_printf("\n");

          DB_printf("\n");
          while ( 1 ) {
       DB_printf("Enter Ta: \n");
       scanf("%d", &accel.Ta);
       DB_printf("Enter Tb: \n");
       scanf("%d", &accel.Tb);
       DB_printf("Enter Tc: \n");
       scanf("%d", &accel.Tc);
       DB_printf("Enter Td: \n");
       scanf("%d", &accel.Td);

       CalculateAccel();
       DB_printf("T1x = ");
       for ( i=0; i<=TSIZE; i++)
       {
            DB_printf("%d ",accel.T1x[i]);
       }
       DB_printf("\n");

       DB_printf("T1y = ");
       for ( i=0; i<=TSIZE; i++)
       {
            DB_printf("%d ",accel.T1y[i]);
       }
       DB_printf("\n");

       DB_printf("T2 = ");
       for ( i=0; i<=TSIZE; i++)
       {
            DB_printf("%d ",accel.T2[i]);
       }
       DB_printf("\n");


       DB_printf("X-Acceleration = %d\n",accel.curr_accel_x);
       DB_printf("Y-Acceleration = %d\n",accel.curr_accel_y);
 }
#endif

#ifdef PWMTEST
// 100 counts, 50% duty cycle, no prescale, polarity = 0, scale of 80,
// center = 1, clock A is source for channel 0 and 1
InitPWM(99, 49, 0x00, 0x00, 19, _H12_CENTR, _H12_PCLK0 | _H12_PCLK1);
// SetDuty(1, 77);
DB_printf("\nPWM Initialized\n\n");
CalibrateAccel();
DB_printf("Calibrated, T1x_cal=%x, T1y_cal=%x, T2_cal=%x, K=%x\n"
                  ,accel.T1x_cal,accel.T1y_cal,accel.T2_cal,accel.K);
while ( 1 )
{
  while ( accel.AccelState != 1) // Wait for start of process
     ;
  disable();
  _H12TMSK1 = 0x00;

  DB_printf("State = %x\n",accel.AccelState);
  DB_printf("  X-Accel = %x, Y-Accel = %x\n",
             accel.curr_accel_x,accel.curr_accel_y);
  DB_printf("  Ta = %x, Tb = %x,  Tc = %x, Td = %x\n",accel.Ta,accel.Tb,
          accel.Tc,accel.Td);

  DB_printf("T1x = ");
     for ( i=0; i<=TSIZE; i++)
     {
          DB_printf("%x ",accel.T1x[i]);
     }
     DB_printf("\n");
```

```
       DB_printf("T1y = ");
       for ( i=0; i<=TSIZE; i++)
       {
            DB_printf("%x ",accel.T1y[i]);
       }
       DB_printf("\n");

       DB_printf("T2 = ");
       for ( i=0; i<=TSIZE; i++)
       {
            DB_printf("%x ",accel.T2[i]);
       }
       DB_printf("\n");



    DB_printf("Menu:\n  1: Set X Duty Cycle\n  2: Set Y Duty Cycle\n");
    scanf("%d",&selection);
    //selection = 666;
    DB_printf("Sel=%d\n",selection);
    switch ( selection )
    {
        case 1 :
            DB_printf("Enter X Duty Cycle (0-99):\n");
            scanf("%d",&duty);
            SetDuty(0, duty);
            DB_printf("   Duty Cycle = %d%%",(99-duty)*100/99);
            break;
        case 2 :
            DB_printf("Enter Y Duty Cycle (0-99):\n");
            scanf("%d",&duty);
            SetDuty(1, duty);
            DB_printf("   Duty Cycle = %d%%",(99-duty)*100/99);
            break;
    }
    _H12TMSK1 = _H12_C1I | _H12_C0I;
    _H12TCTL4 = _H12_EDG0A;
    accel.AccelState = 1;
    _H12TFLG1 = _H12_C0F | _H12_C1F;

    enable();
    while ( accel.AccelState == 1) // Wait for at least 1 measurement
       ;
    while ( accel.AccelState == 2) // Wait for at least 1 measurement
       ;


  }

  #endif

  #ifdef LIVETEST
    //CalibrateAccel();
    while ( 1 )  {
        DB_printf("Hit a key to get next reading:\n");
        getchar();
        DB_printf("X-accel = %d, Y-accel = %d\n",accel.curr_accel_x,
                accel.curr_accel_y);
    }
   #endif


}

#endif
/*------------------------------ End of file ------------------------------*/
```