# Tiva C Code

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/pin_map.h"
#include "driverlib/debug.h"
#include "driverlib/sysctl.h"
#include "driverlib/adc.h"
#include "driverlib/gpio.h"
#include "driverlib/uart.h"

#include "inc/hw_ints.h"
#include "inc/hw_timer.h"
#include "driverlib/timer.h"
#include "driverlib/interrupt.h"

#define HH_BUF_SIZE 5 // record hh pos'n for ~4ms*25 = ~100ms
#define M_BUF_SIZE     16


#define HH_THRESH0    6
#define HH_THRESH1    40

char hhEn = 1;


char mBuf[M_BUF_SIZE];                    //
char mTail = 0;
char mHead = 0;
char mCnt = 0;

uint32_t hhBuf[HH_BUF_SIZE];              //
char hhHead = 0;
char hhTail = 1;

char difCnt = 0;
signed char difference = 0;
char absDif;
char hhDif[HH_BUF_SIZE - 1];
char i = 0;

//Arrays for storing ADC FIFO values
uint32_t ui32ADC0Value[2] = {};
uint32_t ui32ADC1Value[2] = {};
int temp = 0;
int temp2 = 0;

char digit = '0';
```

# Tiva C Code

```
char digit2 = '0';

unsigned long int ulbase=0;
unsigned long int ulbase2=0;

//---------------------------------------------------------------------------

void inc_hhHead(void){

        if(hhHead == (HH_BUF_SIZE - 1)){
                hhHead = 0;
        }
        else {
                hhHead++;
        }
}


void inc_hhTail(void){

        if(hhTail == (HH_BUF_SIZE - 1)){
                hhTail = 0;
        }
        else {
                hhTail++;
        }

}

//---------------------------------------------------------------------------

// ISR for Timer0
void Timer0IntHandler(void){


        // Clear the timer interrupt
   TimerIntClear(WTIMER1_BASE, TIMER_TIMA_TIMEOUT);

        //Clear interrupt flag
        ADCIntClear(ADC1_BASE, 2);
        //Trigger ADC
        ADCProcessorTrigger(ADC1_BASE, 2);
        //wait for conversion
        while(!ADCIntStatus(ADC1_BASE, 2, false)){}
        //retrieve data
        ADCSequenceDataGet(ADC1_BASE, 2, ui32ADC1Value);

        hhBuf[hhHead] = (*ui32ADC1Value-1450)/20;    // scale to 0- 127 value -1500)/19
```

# Tiva C Code

```
        difference = (hhBuf[hhHead] - hhBuf[hhTail]);
        absDif = abs(difference);

        //UARTCharPut(UART0_BASE, hhBuf[hhHead]);

        if(absDif > HH_THRESH0){


                UARTCharPut(UART0_BASE, hhBuf[hhHead]);
                //UARTCharPut(UART0_BASE, ' ');

                if(difference<0   &&  hhEn==1 && absDif>HH_THRESH1 && hhBuf[hhHead]<0x18) {
        // Splash
                        hhEn = 0;
                        TimerEnable(WTIMER0_BASE, TIMER_A);
                        UARTCharPut(UART0_BASE, 0xFF);
                        //UARTCharPut(UART0_BASE, ' ');

                }

                else if (difference>0   &&  hhEn==1 && absDif>HH_THRESH1 && hhBuf[hhHead]>0x68) {
        // Chick
                        hhEn = 0;
                        TimerEnable(WTIMER0_BASE, TIMER_A);
                        UARTCharPut(UART0_BASE, 0x00);
                        //UARTCharPut(UART0_BASE, );

                }
        }

        inc_hhHead();
        inc_hhTail();

}

void Timer1IntHandler(void){

        // Clear the timer interrupt
   TimerIntClear(WTIMER0_BASE, TIMER_TIMA_TIMEOUT);

   hhEn = 1;                // enable the hh to make another chick soundwhen the timer runs out
}
```

# Tiva C Code

```
//-------------------------------------------------------------------

int main(void){


        //Set clock to 40Mhz
        SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);



        /* Set the clock for the GPIO Port F and WTIMER-1*/
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_WTIMER1);
   //Set timer
   TimerDisable(WTIMER1_BASE, TIMER_A);
        TimerConfigure(WTIMER1_BASE, TIMER_CFG_SPLIT_PAIR|TIMER_CFG_A_PERIODIC);
        ulbase=40000;              //
        TimerLoadSet(WTIMER1_BASE, TIMER_A,ulbase);
        TimerPrescaleSet(WTIMER1_BASE,TIMER_A,0x0003);
        //IntMasterEnable();
        TimerIntRegister(WTIMER1_BASE, TIMER_A, Timer0IntHandler);
        TimerIntEnable(WTIMER1_BASE, TIMER_TIMA_TIMEOUT);
        IntEnable(INT_WTIMER1A);
        TimerEnable(WTIMER1_BASE,TIMER_A);

                ///* Set the clock for the GPIO Port F and WTIMER-1
          //SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
          SysCtlPeripheralEnable(SYSCTL_PERIPH_WTIMER0);
          //Set timer
          TimerDisable(WTIMER0_BASE, TIMER_A);
                TimerConfigure(WTIMER0_BASE, TIMER_CFG_SPLIT_PAIR|TIMER_CFG_A_ONE_SHOT);
                ulbase2=1000000;
                                                        //  Set amount of time to disable hh pedal after
a chick
                TimerLoadSet(WTIMER0_BASE, TIMER_A, ulbase2);
                TimerPrescaleSet(WTIMER0_BASE,TIMER_A,0x0003);
                IntMasterEnable();
                TimerIntRegister(WTIMER0_BASE, TIMER_A, Timer1IntHandler);
                TimerIntEnable(WTIMER0_BASE, TIMER_TIMA_TIMEOUT);
                IntEnable(INT_WTIMER0A);
                TimerEnable(WTIMER0_BASE, TIMER_A);

                // */
```

# Tiva C Code

```
        //enable GPIO Port A to access UART0
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
        SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
        //configure pins for uart
        GPIOPinConfigure(GPIO_PA1_U0TX);
        GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0|GPIO_PIN_1);
        //configure UART for 115,200 baud rate, 8 data bits, 1 stop bit, and 0 parity
        UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(), 115200,
(UART_CONFIG_WLEN_8|UART_CONFIG_STOP_ONE|UART_CONFIG_PAR_NONE));
        //enable UART
        UARTEnable(UART0_BASE);



        //enagle GPIO Port F to access LEDs
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
        //configuer LED pins as output
        GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);



        //enable GPIO Port E to access pins AIN0 and AIN1
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);

        //enable ADC0
        SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
        //configure Pin PE3 (AIN0) for ADC usage
        GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_3);
        //Before Configuring ADC Sequencer 3, it should be OFF
        ADCSequenceDisable(ADC0_BASE, 3);
        //configure ADC sequencer
        ADCSequenceConfigure(ADC0_BASE, 3, ADC_TRIGGER_PROCESSOR, 0);
        //configure ADC sequencer steps
        ADCSequenceStepConfigure(ADC0_BASE, 3, 0, ADC_CTL_CH0|ADC_CTL_IE|ADC_CTL_END);
        //enable ADC sequencer
        ADCSequenceEnable(ADC0_BASE, 3);



        //enable ADC1
        SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC1);
        //configure Pin PE2 (AIN1) for ADC usage
        GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_2);
        //Before Configuring ADC Sequencer 2, it should be OFF
```

# Tiva C Code

```
        ADCSequenceDisable(ADC1_BASE, 2);
        //configure ADC sequencer
        ADCSequenceConfigure(ADC1_BASE, 2, ADC_TRIGGER_PROCESSOR, 0);
        //configure ADC sequencer steps
        ADCSequenceStepConfigure(ADC1_BASE, 2, 0, ADC_CTL_CH1|ADC_CTL_IE|ADC_CTL_END);
        //enable ADC sequencer
        ADCSequenceEnable(ADC1_BASE, 2);



        while(1){


        }


}
```