Ayden Dauenhauer and Alan Rieger

Prof. Wolfe

ELEN 121L Tuesday 2:15 p.m.

4 June 2024

Lab 9 – USB Music Player

Part 1:

Timer Interrupt Code:

```
47  int i=0;

214 void TIM6_DAC_IRQHandler(void)
215 {
216   /* USER CODE BEGIN TIM6_DAC_IRQn 0 */
217
218   /* USER CODE END TIM6_DAC_IRQn 0 */
219   HAL_TIM_IRQHandler(&htim6);
220   HAL_DAC_IRQHandler(&hdac1);
221   /* USER CODE BEGIN TIM6_DAC_IRQn 1 */
222   write_DAC1Ch2(audiobuffer[i], DFLT_VOLUME);
223   i=(i+1)%ABUF_SIZE;
224   /*****************************************************/
225   /* Put your code here to produce a 400Hz Sine Wave */
226   /*****************************************************/
227
228
229   /* USER CODE END TIM6_DAC_IRQn 1 */
230 }
231
232 /* USER CODE BEGIN 1 */
233
234 /* USER CODE END 1 */
235
```

Part 2:

Timer 6 Code:

```
/* Private variables ----
/* USER CODE BEGIN PV */
  int i=0;
  int j=0;
  int flag=0;
  int counter=0;
  int k=0;
    int a=0;

void TIM6_DAC_IRQHandler(void)
{
  /* USER CODE BEGIN TIM6_DAC_IRQn 0 */

  /* USER CODE END TIM6_DAC_IRQn 0 */
  HAL_TIM_IRQHandler(&htim6);
  HAL_DAC_IRQHandler(&hdac1);
  /* USER CODE BEGIN TIM6_DAC_IRQn 1 */
    if(lastbuffer == 0){
    if (flag == 0) {
            if(abuf_full[0] == true){
                write_DAC1Ch2(audiobuffer[a][i], DFLT_VOLUME);
                i=(i+1)%ABUF_SIZE;
            }
            if(i==0){
                abuf_full[0]=false;
                flag = 1;
                a++;
            }
        }
        if (flag == 1) {
            if(abuf_full[1] == true){
                write_DAC1Ch2(audiobuffer[a][j], DFLT_VOLUME);
                j=(j+1)%ABUF_SIZE;
            }
            if(j==0){
                abuf_full[1]=false;
                flag = 0;
                a--;
            }
        }
    }
  else if(flag!=2){
        write_DAC1Ch2(audiobuffer[a][counter], DFLT_VOLUME);
        counter=(counter+1)%ABUF_SIZE;
        if(counter/2>(lastbuffer-1)){
            flag=2;
        }
    }
  /*****************************************************/
  /* Put your code here to play a song                 */
  /*****************************************************/


  /* USER CODE END TIM6_DAC_IRQn 1 */
}
```

Part 3:

Code to Initialize the LCD with 0s:

```
HAL_DAC_Start(&hdac1, DAC1_CHANNEL_2);
HAL_TIM_Base_Start_IT(&htim6);      /* Start Timer 6 at 16KHz to run DAC */
  HAL_TIM_Base_Start_IT(&htim16);
BSP_LCD_GLASS_Init();
  start_LCD();
```

```
#include "main.h"
#include "stm32l476g_discovery.h"
#include "stm32l476g_discovery_glass_lcd.h"
volatile int seccount=0;
volatile int mincount=0;
volatile int seccount2=0;
volatile int mincount2=0;
volatile int tenth=0;
uint8_t bob=48;
void start_LCD(void){
   BSP_LCD_GLASS_DisplayChar(&bob, POINT_OFF, DOUBLEPOINT_OFF, 2);
   BSP_LCD_GLASS_DisplayChar(&bob, POINT_OFF, DOUBLEPOINT_ON, 3);
   BSP_LCD_GLASS_DisplayChar(&bob, POINT_OFF, DOUBLEPOINT_OFF, 4);
   BSP_LCD_GLASS_DisplayChar(&bob, POINT_OFF, DOUBLEPOINT_OFF, 5);
}
```

Code in Timer 16 to create the Timer:

```
void TIM1_UP_TIM16_IRQHandler(void)
{
  /* USER CODE BEGIN TIM1_UP_TIM16_IRQn 0 */

  /* USER CODE END TIM1_UP_TIM16_IRQn 0 */
  HAL_TIM_IRQHandler(&htim16);
  /* USER CODE BEGIN TIM1_UP_TIM16_IRQn 1 */
  if(lastbuffer==0){
    if(stop%2==0){
        tenth++;
        if(tenth>9){
            tenth=0;
            seccount++;
            display=seccount+48;
            BSP_LCD_GLASS_DisplayChar(&display, POINT_OFF, DOUBLEPOINT_OFF, 5);
        }
        if(seccount>9){
            seccount=0;
            seccount2++;
            display=48;
            BSP_LCD_GLASS_DisplayChar(&display, POINT_OFF, DOUBLEPOINT_OFF, 5);
            display=seccount2+48;
            BSP_LCD_GLASS_DisplayChar(&display, POINT_OFF, DOUBLEPOINT_OFF, 4);
        }
        if(seccount2>5){
            seccount2=0;
            mincount++;
            display=48;
            BSP_LCD_GLASS_DisplayChar(&display, POINT_OFF, DOUBLEPOINT_OFF, 4);
            display=mincount+48;
            BSP_LCD_GLASS_DisplayChar(&display, POINT_OFF, DOUBLEPOINT_ON, 3);
        }
        if(mincount>9){
            mincount=0;
            mincount2++;
            BSP_LCD_GLASS_DisplayChar(&display, POINT_OFF, DOUBLEPOINT_ON, 3);
            display=mincount2+48;
            BSP_LCD_GLASS_DisplayChar(&display, POINT_OFF, DOUBLEPOINT_OFF, 2);
        }
        if(mincount2>5){
            mincount2=0;
        }
    }
  }
  /* USER CODE END TIM1_UP_TIM16_IRQn 1 */
}
```

Code to Play the Song in Timer 6:

```
void TIM6_DAC_IRQHandler(void)
{
  /* USER CODE BEGIN TIM6_DAC_IRQn 0 */

  /* USER CODE END TIM6_DAC_IRQn 0 */
  HAL_TIM_IRQHandler(&htim6);
  HAL_DAC_IRQHandler(&hdac1);
  /* USER CODE BEGIN TIM6_DAC_IRQn 1 */
  if(lastbuffer == 0){
    if (flag == 0) {
        if(abuf_full[0] == true){
            write_DAC1Ch2(audiobuffer[a][i], DFLT_VOLUME);
            i=(i+1)%ABUF_SIZE;
        }
        if(i==0){
            abuf_full[0]=false;
            flag = 1;
            a++;
        }
    }
    if (flag == 1) {
          if(abuf_full[1] == true){
              write_DAC1Ch2(audiobuffer[a][j], DFLT_VOLUME);
              j=(j+1)%ABUF_SIZE;
          }
          if(j==0){
              abuf_full[1]=false;
              flag = 0;
              a--;
          }
    }
  }
  else if(flag!=2){
      write_DAC1Ch2(audiobuffer[a][counter], DFLT_VOLUME);
      counter=(counter+1)%ABUF_SIZE;
      if(counter/2>(lastbuffer-1)){
          flag=2;
      }
  }
  /*****************************************************/
  /* Put your code here to play a song                 */
  /*****************************************************/


  /* USER CODE END TIM6_DAC_IRQn 1 */
}
```

Global Variables for the Timer Codes:

```
int i=0;
int j=0;
int flag=0;
int counter=0;
int k=0;
  int a=0;
  int stop=0;
  uint8_t reset=48;
  uint8_t display=0;
```

Part 4:

Code to check what button was pressed and do what it is supposed to do:

```
303       if(empty==1){
304         hex=parseIRCode();
305         if(hex==IR_UP){
306           if(sound != 250){
307             sound=sound+VOLUME_INCREMENT;
308           }
309         }
310         if(hex==IR_DOWN){
311           if(sound != 10){
312             sound=sound-VOLUME_INCREMENT;
313           }
314         }
315         if(hex==IR_LEFT){
316           FS_FileClose();
317           string= 48;
318           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 2);
319           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_ON, 3);
320           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 4);
321           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 5);
322           tenth=0;
323           seccount=0;
324           seccount2=0;
325           mincount=0;
326           mincount2=0;
327           abuf_full[0]=false;
328           abuf_full[1]=false;
329           Appli_state=APPLICATION_READY;
330         }
331         if(hex==IR_CIRCLE){
332           if (flag == 2) {
333             flag = temp;
334             temp = 2;
335           }
336           else {
337             temp = flag;
338             flag = 2;
339           }
340           stall = (stall + 1) % 2;
341         }
342         empty=0;
343         value=0;
344       }
```

Code to Parse the IR Data:

```
volatile int irdat[SAMPLE_COUNT];
volatile int empty;
int value=0;
int binary[32];
extern int tenth;
extern int seccount;
extern int seccount2;
extern int mincount;
extern int mincount2;
uint32_t parseIRCode(){
  int i=0;
  int counter=0;
  int j=0;
  // Get through starting zeros and ones
  while(irdat[i]!=1){
      i++;
  }
  while(irdat[i]!=0){
      i++;
  }
  // Determine the binary values of the signal
  for(j=0;j<32;j++){
    while(irdat[i]==0){
      i++;
    }
    while(irdat[i]==1){
      i++;
      counter++;
    }
    if(counter>=12){
      binary[j]=1;
    }
    else if(counter<12){
      binary[j]=0;
    }
    counter=0;
  }
  // Convert binary to hex
  int k=0;
  int multiple=31;
  int decimal=0;
  for(k=0;k<32;k++){
    decimal+=binary[k]*pow(2, multiple);
    multiple--;
  }
  return decimal;
}
```

Timer Interrupt Code to Receive Message from IR:

```
void TIM7_IRQHandler(void)
{
  /* USER CODE BEGIN TIM7_IRQn 0 */

  /* USER CODE END TIM7_IRQn 0 */
  HAL_TIM_IRQHandler(&htim7);
  /* USER CODE BEGIN TIM7_IRQn 1 */
  int in;
  if(empty==0){
    in=HAL_GPIO_ReadPin(IR_IN_GPIO_Port, IR_IN_Pin);
    if(in==0){
      start=1;
    }
    if(start==1){
      irdat[value]=in;
      value++;
    }
    if(value>700){
      start=0;
      empty=1;
    }
  }
  /* USER CODE END TIM7_IRQn 1 */
}
```