Alan Rieger and Ayden Dauenhauer

5/14/2024

ELEN 121L Lab 6

Part 1:

Main Code:

```
124  int main(void)
125  {
126    /* USER CODE BEGIN 1 */
127
128    /* USER CODE END 1 */
129
130    /* MCU Configuration--------------------------------------------------------*/
131
132    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
133    HAL_Init();
134
135    /* USER CODE BEGIN Init */
136
137    /* USER CODE END Init */
138
139    /* Configure the system clock */
140    SystemClock_Config();
141
142    /* USER CODE BEGIN SysInit */
143
144    /* USER CODE END SysInit */
145
146    /* Initialize all configured peripherals */
147    MX_GPIO_Init();
148    MX_LCD_Init();
149    MX_RTC_Init();
150    MX_TIM7_Init();
151    /* USER CODE BEGIN 2 */
152    HAL_TIM_Base_Start_IT(&htim7);
153    BSP_LCD_GLASS_Init();
154    start_LCD();
155    //display_test();
156    int hex=0;
157    /* USER CODE END 2 */
158
159    /* Infinite loop */
160    /* USER CODE BEGIN WHILE */
161    while (1)
162    {
163      if(empty==1){
164        hex=parseIRCode();
165        if(hex==IR_B){
166          HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_2);
167        }
168        if(hex==IR_CIRCLE){
169          HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_8);
170        }
171        empty=0;
172        value=0;
173      }
174      /* USER CODE END WHILE */
175
176      /* USER CODE BEGIN 3 */
177    }
178    /* USER CODE END 3 */
179  }
```

parseIRCode:

```
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
volatile int irdat[SAMPLE_COUNT];
volatile int empty;
int value=0;
int binary[32];
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--;
  }
  decimal++;
  return decimal;
}
/* USER CODE END 0 */
```

Timer 7 Interrupt Handler:

```
243  void TIM7_IRQHandler(void)
244  {
245    /* USER CODE BEGIN TIM7_IRQn 0 */
246
247    /* USER CODE END TIM7_IRQn 0 */
248    HAL_TIM_IRQHandler(&htim7);
249    /* USER CODE BEGIN TIM7_IRQn 1 */
250    int in;
251    if(empty==0){
252      in=HAL_GPIO_ReadPin(IR_IN_GPIO_Port, IR_IN_Pin);
253      if(in==0){
254        start=1;
255      }
256      if(start==1){
257        irdat[value]=in;
258        value++;
259      }
260      if(value>700){
261        start=0;
262        empty=1;
263      }
264    }
265    /* USER CODE END TIM7_IRQn 1 */
266  }
267
268  /* USER CODE BEGIN 1 */
269
270  /* USER CODE END 1 */
```

Part 2:

Main Code:

```
124  int main(void)
125  {
126    /* USER CODE BEGIN 1 */
127
128    /* USER CODE END 1 */
129
130    /* MCU Configuration--------------------------------------------------------*/
131
132    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
133    HAL_Init();
134
135    /* USER CODE BEGIN Init */
136
137    /* USER CODE END Init */
138
139    /* Configure the system clock */
140    SystemClock_Config();
141
142    /* USER CODE BEGIN SysInit */
143
144    /* USER CODE END SysInit */
145
146    /* Initialize all configured peripherals */
147    MX_GPIO_Init();
148    MX_LCD_Init();
149    MX_RTC_Init();
150    MX_TIM7_Init();
151    /* USER CODE BEGIN 2 */
152    HAL_TIM_Base_Start_IT(&htim7);
153    BSP_LCD_GLASS_Init();
154    int hex=0;
155    int a=0;
156    uint8_t string;
157    /* USER CODE END 2 */
158
159    /* Infinite loop */
160    /* USER CODE BEGIN WHILE */
161    while (1)
162    {
163      if(empty==1){
164        hex=parseIRCode();
165        if(hex==IR_POWER){
166          string= 'P';
167          BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 1);
168          string= 'O';
169          BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 2);
170          string= 'W';
171          BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 3);
172          string= 'E';
173          BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 4);
174          string= 'R';
175          BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 5);
```

```
176         }
177         if(hex==IR_A){
178           string= 'A';
179           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 5);
180         }
181         if(hex==IR_B){
182           string= 'B';
183           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 5);
184         }
185         if(hex==IR_C){
186           string= 'C';
187           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 5);
188         }
189         if(hex==IR_UP){
190           string= 'U';
191           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 4);
192           string= 'P';
193           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 5);
194         }
195         if(hex==IR_DOWN){
196           string= 'D';
197           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 2);
198           string= 'O';
199           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 3);
200           string= 'W';
201           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 4);
202           string= 'N';
203           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 5);
204         }
205         if(hex==IR_LEFT){
206           string= 'L';
207           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 2);
208           string= 'E';
209           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 3);
210           string= 'F';
211           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 4);
212           string= 'T';
213           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 5);
214         }
215         if(hex==IR_RIGHT){
216           string= 'R';
217           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 1);
218           string= 'I';
219           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 2);
220           string= 'G';
221           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 3);
222           string= 'H';
223           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 4);
224           string= 'T';
225           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 5);
226         }
227         if(hex==IR_CIRCLE){
228           string= 'C';
229           BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 0);
```

```
230         string= 'I';
231         BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 1);
232         string= 'R';
233         BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 2);
234         string= 'C';
235         BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 3);
236         string= 'L';
237         BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 4);
238         string= 'E';
239         BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 5);
240       }
241       HAL_Delay(1000);
242       string= '\0';
243       BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 0);
244       BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 1);
245       BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 2);
246       BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 3);
247       BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 4);
248       BSP_LCD_GLASS_DisplayChar(&string, POINT_OFF, DOUBLEPOINT_OFF, 5);
249       empty=0;
250       value=0;
251     }
252     /* USER CODE END WHILE */
253
254     /* USER CODE BEGIN 3 */
255   }
256   /* USER CODE END 3 */
257 }
```

parseIR Code:

```
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
volatile int irdat[SAMPLE_COUNT];
volatile int empty;
int value=0;
int binary[32];
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--;
  }
  decimal++;
  return decimal;
}
/* USER CODE END 0 */
```

Timer 7 Interrupt Handler:

```
243  void TIM7_IRQHandler(void)
244  {
245    /* USER CODE BEGIN TIM7_IRQn 0 */
246
247    /* USER CODE END TIM7_IRQn 0 */
248    HAL_TIM_IRQHandler(&htim7);
249    /* USER CODE BEGIN TIM7_IRQn 1 */
250    int in;
251    if(empty==0){
252      in=HAL_GPIO_ReadPin(IR_IN_GPIO_Port, IR_IN_Pin);
253      if(in==0){
254        start=1;
255      }
256      if(start==1){
257        irdat[value]=in;
258        value++;
259      }
260      if(value>700){
261        start=0;
262        empty=1;
263      }
264    }
265    /* USER CODE END TIM7_IRQn 1 */
266  }
267
268  /* USER CODE BEGIN 1 */
269
270  /* USER CODE END 1 */
```