Extrapoint_02 - Application note

Il progetto è realizzato sulla scheda LandTiger LPC1768. I moduli più importanti sono:

- `main`: contiene *sample.c* il quale gestisce tutte le inizializzazioni dei moduli necessari al funzionamento del progetto.
- `button_EINT`: gestisce gli interrupt determinati dalla pressione degli interruttori INT0, KEY1, KEY2 (con debouncing implementato con RIT).
- `time`r: gestisce i timer. In *IRQ_timer.c* sono presenti gli handler degli interrupt scatenati dai timer 0-1-2-3.
- `RIT`: gestisce il Repetitive Interruption Timer, utilizzato come polling per joystick e touchscreen e per il debouncing di tutti gli interruttori.
- `joystick`: gestisce la comunicazione con le porte del joystick
- `adc`: gestisce potenziometro
- `dac`: gestisce lo speaker
- `GLCD`: gestisce il display LCD
- `TouchPanel`: gestisce il touschscreen

All'interno di ogni file sono presenti tutti i commenti inerenti alle variabili / procedure utilizzate.

Utilizzo dei timer

TIMER 0 → usato per il moto dell'ascensore: 0.09 secondi
TIMER 1 → usato per controllare casi di allarme e attivazione dell'ascensore: 60 secondi
usato per evidenziare pulsanti sul touch screen
TIMER 2 → usato per blinking a 2Hz durante il moto,
usato per riprodurre suoni sullo speaker durante allarme/emergenza
TIMER 3 → usato per blinking a 5 Hz a fine moto,
usato per blinking a 4 Hz e attivazione speaker in caso di allarme

Tutti i timer sono stati impostati in modo tale che al termine del conteggio scatenino un' eccezione e si resettino.

Gestione del RIT - file IRQ_RIT.c

Nel file *IRQ_RIT.c* sono presenti le funzioni che implementano il moto dell'ascensore, la sua prenotazione, la gestione delle situazioni di allarme. Per identificare tutte le casistiche di utilizzo è presente una lista di flag globali in modo tale da poter essere accessibili e modificabili anche dagli altri moduli coinvolti.
All'interno del file si trovano le seguenti funzioni:

`void manageJoystick(void);` → funzione che gestisce i casi in cui l'utente debba utilizzare il joystick: salita, discesa, allarme.
`void manageFloorPanels(void);` → funzione che gestisce le prenotazioni dell'ascensore attraverso i pulsanti dei pannelli presenti ai piani
`void manageEmergencyButton(void);` → funzione che gestisce il pulsante di emergenza
`void manageTouchPanel(void);` → funzione che gestisce l'accesso al "maintenance mode" attraverso il touch screen

Il RIT è settato a 50millisecondi con una frequenza di clock di 100MHz e scatena un interrupt ogni volta che termina.

L'handler di questo interrupt viene utilizzato come meccanismo di **polling** per il joystick (necessario per poterne leggere i valori dato che i pulsanti del joystick non scatenano interrupt) e per il touchscreen e come meccanismo di **debouncing**.

- (funzione ***manageJoystick***) : in questo punto del codice si riesce a sapere quale pulsante del joystick si sta premendo e viene implementato il movimento dell'ascensore (simulato attraverso dei timer). Muovendosi su una rampa lunga 8m con una velocità di 4 km/h, ci si impiegherà 7.2 secondi a percorrerla interamente. Poiché è possibile invertire il moto o fermarsi in qualsiasi momento, si è pensato di creare un timer di 0.09 secondi, il quale scatena un interrupt che incrementa una variabile `position` che tiene conto della posizione dell'ascensore lungo la rampa (ha valori da 0 a 80); Se nell'handler del RIT si dovesse rilevare una situazione di "stop", questo timer viene disabilitato. Negli altri casi l'handler del timer decide se incrementare (in caso di salita) o decrementare (in caso di discesa) la variabile della posizione. Parallelamente agisce il timer 2 per far lampeggiare lo Status Led a 2 Hz durante il moto (e che avvia il timer 3 per farlo lampeggiare a 5 Hz alla fine del moto).

- (funzione ***manageEmergencyButton***) : nel caso in cui l'utente prema il pulsante INT0 per 2 secondi, oppure l'ascensore rimanga fermo lungo la rampa per più di 60 secondi, si attiva una procedura di Emergenza/allarme che può essere sbloccata dall'utente stesso (ripremendo INT0) o da un utente esterno che chiama l'ascensore da uno dei due piani.
  Il conteggio dei 2 secondi per la pressione di INT0 necessari a riconoscere una situazione di emergenza viene implementato attraverso un contatore (lo stesso utilizzato per il debouncing di INT0 – variabile `down3`): se ho premuto il pulsante per 40 "ripetizioni" del RIT si entra in modalità emergenza. Il conteggio dei 60 secondi di stallo lungo la rampa viene effettuato dal timer 1, il quale scatena un interrupt il cui handler gestisce la procedura di allarme. Lo Status Led lampeggia ad una frequenza di 4Hz (attraverso il timer 2) e, in modo sincrono al led, lo speaker alterna due note (le note sono rappresentate come indici di un vettore di frequenze e possono essere scelte in fase di gestione – vedi punti seguenti).

- (funzione ***manageFloorPanels) :*** attraverso il meccanismo di debouncing si riconosce la pressione di KEY1/KEY2/INT0. I pulsanti KEY1 e KEY2 sono utilizzati per prenotare o richiamare l'ascensore verso un piano. In base al piano da cui l'ascensore è stato chiamato e alla sua posizione corrente, l'ascensore può essere subito disponibile oppure si reca al piano corretto. Successivamente si avvia un timer di 60 secondi (timer 1) entro i quali è necessario attivare l'ascensore premendo il tasto select del joystick (pressione riconosciuta nella funzione `manageJoystick`).

- (funzione ***manageTouchPanel***) : L'handler dei RIT verifica, inoltre, se si vuole accedere al pannello di gestione (solo se l'ascensore non è in uso) attraverso un touch screen. Una volta accedutovi permette di modificare le note del suono d'allarme (si noti l'utilizzo delle funzioni di selezione dei pulsanti (***ButtonSelection*** e ***ButtonDeselection*** definite in GLCD.c: alla pressione dei pulsanti questi cambiano colore - implementati utilizzando il timer 1 per poter cambiare temporaneamente il colore dei pulsanti e del testo visualizzati sullo schermo).

  Premendo sul nome di una delle due note è possibile modificarla utilizzando il potenziometro.

  Selezione delle note e utilizzo del potenziometro

  Le possibili note sono 13, ovvero 13 frequenze che indicano quando mandare allo speaker una sinusoide di 45 campioni (sinusoide salvata in ***SinTab***, frequenze salvate in *vettK*). L'handler dell'interrupt generato dal potenziometro (***IRQ_adc.c***) confronta il valore analogico appena letto con quello letto in precedenza: se si è in modalità di modifica delle nota (flag `modTone`) e se il valore letto è diverso dal precedente lo si utilizza per calcolare il corrispettivo valore intero in un intervallo da 0 a 12. Questo valore intero viene salvato nelle variabili intere note1 e note2 (in base al fatto se sul touch screen è stato selezionato un'opzione piuttosto che un'altra) le quali sono utilizzate all'interno dell'handler dell'interrupt del timer 3 che si occupa dell'emergency mode. Le stesse variabili vengono anche utilizzate per accedere ad un vettore di stringhe per poter stampare a schermo i corrispettivi nomi delle note (vettore `NameNote`).

  Handler dell'interrupt generato dal potenziometro:

```
void ADC_IRQHandler(void) {

  AD_current = ((LPC_ADC->ADGDR>>4) & 0xFFF);/* Read Conversion Result             */
  if(modTone==1){
    //se sono nella modifica di un tono --> devo stampare sul display e devo modificare la nota
  if(AD_current != AD_last){

    if(nTone==3){
        note1=AD_current*12/0xFFF; //scelta della prima nota attraverso scelta dell'indice "note1"
        GUI_Text(77,100, (uint8_t *) NameNote[13],Blue,White);
        GUI_Text(77,100, (uint8_t *) NameNote[note1],Blue,White);
    }
    if(nTone==4){
        note2=AD_current*12/0xFFF; //scelta della seconda nota attraverso scelta dell'indice "note2"
        GUI_Text(75,200, (uint8_t *) NameNote[13],Blue,White);
        GUI_Text(75,200, (uint8_t *) NameNote[note2],Blue,White);
    }
    AD_last = AD_current;
  }
}
}
```

  La frequenza letta dal vettore `vettK` viene utilizzata dal timer 3 per definire ed abilitare un ulteriore timer (timer 2) il quale, al termine, scatena un interrupt e invia allo speaker l'intero insieme dei valori della sinusoide ***SinTab***.

  Le frequenze sono state calcolate come nella formula presente nella seguente porzione di codice.

  Handler dell'interrupt generato dal timer 3 (modalità di emergenza/allarme) e dal timer 2

```
void TIMER3_IRQHandler (void)
{
  int vettK[13]={1062,1125,1192,1263,1339,1417,1502,1592,1684,1786,1890,2006,2120};
//vettore delle frequenze
  /* k=1/f'*f/n  k=f/(f'*n) k=25MHz/(f'*45) */
        /*C: *523 Hz --> k=25M/(523*45)= 1062
        B: 494Hz --> k=25M/(494*45)= 1125
        Bb: 466 Hz --> k=25M/(466*45)= 1192
        A: 440 Hz --> k=25M/(440*45)= 1263
        Ab: 415 Hz --> k=25M/(415*45)= 1339
        G: 392 Hz --> k=25M/(392*45)= 1417
        Gb: 370 Hz --> k=25M/(370*45)= 1502
        F: 349 Hz --> k=25M/(349*45)= 1592
        E: 330 Hz --> k=25M/(330*45)= 1684
        Eb: 311 Hz --> k=25M/(311*45)= 1786
        D: 294 Hz --> k=25M/(294*45)= 1890
        Db: 277 Hz --> k=25M/(277*45)= 2006
        C: 262 Hz --> k=25M/(262*45)= 2120
   */
  static int i3=0;

  if(i3%2==0){
    LED_On(7);
    if(elevator_emergency==1){
        init_timer(2,vettK[note1]);
        reset_timer(2);
        enable_timer(2);
    }

  }
  else{
    LED_Off(7);
    if(elevator_emergency==1){
        init_timer(2,vettK[note2]);
        reset_timer(2);
        enable_timer(2);
    }
  }
  i3++;
```

```
void TIMER2_IRQHandler (void)
{
  static int i=0;
  static int ticks=0;
  if(elevator_emergency==1){
    //speaker sounds in emergency mode
    /* DAC management */
  DAC_convert (SinTable[ticks]<<6);
  ticks++;
    if(ticks==45)
      ticks=0;
  }else{
    //blinking 2Hz, moving of elevator
  if(i%2==0){
    LED_On(7);
  }
  else{
    LED_Off(7);
  }
  i++;
  if(i==10000){
    i=0;
  }
}
  LPC_TIM2->IR = 1;     /* clear interrupt flag */
  return;
}
```