Universidade de Brasília - UnB
Faculdade do Gama - FGA
Eletrônica Embarcada 2/2019
Prof. Drº Gilmar Beserra
Alunos :
Bruno Carvalho Faria dos Santos – 14/0132767
Elias Queiroga Vieira – 16/0118719
GRUPO 10

Ponto de controle 3:

Projeto: Music Player

Brasília – Df, 13 de Novembro de 2019.

O projeto do music player consiste do uso de todas os pinos da porta P1, 2 pinos para o LED e o SPEAKER (Saídas) e 6 pinos para os botões (RESET_BUTTON, PLAY_BUTTON, SLOW_BUTTON, FAST_BUTTON, SONG1_BUTTON, SONG2_BUTTON) (entradas), no qual o funcionamento dos buttons é baseado em rotinas de interrupção. Além de ter sido implementado o uso de LCD como periférico afim apresentar o nome do *state* do music player, para isto, foram usados todos os pinos da porta P2, a função de saída do pino P2OUT e P2DIR estão associados a entrada de dados no LCD.

Além disso foram criadas funções (e protótipos de funções) para implementar o funcionamento de cada um dos botões, além da configuração do Timer_A e do Watchdog Timer (WDT), incluindo a habilitação do bit de interrupção das portas e indicação de Low Power Mode quando as ISR forem chamadas (_bis_SR_register(GIE+LPM0_bits)). Após o acréscimo do LCD foi necessário implementar outra configuração para que houvesse um controle de apresentação dos caracteres da *string* com o nome dos states, de forma que a velocidade de escrita das palavras no LCD possam ser visíveis.

As funções *void* ***init_timerA****(void), void* ***init_WDT****(void),* são responsáveis pela configuração do canal A0 do TimerA (uso da SMCLK = 1.1MHz) configurado no modo UP, divisão do clk por 0, e configuração do Watchdog Timer (diferente do usual da aula em que costuma-se desativar o WDT, no projeto ele é configurado para gerar interrupção em períodos específicos) , *void* ***init_P1****(void)* , onde todos os pinos da porta P1, que estão relacionados aos botões externos a placa são configurados como saída, e tendo setados o resistor de pull up e bit de interrupção de cada porta , *void* ***toggle_pause****(void), void* ***restart_song****(void), void* ***increase_song****(void), void* ***deacrease_song****(void), void* ***song_song1****(void), void* ***song_song2****(void)* são as funções que são específicas ao funcionamento do music player, cada um é executado em função da Subrotina *ISR_VECTOR(button_handler,"int02")* que é a subrotina de interrupção acionada pelo pressionamento dos botões de controle do *music player* que faz o programa entrar na rotina *interrupt void* ***button_handler*** (*void*), e que chama as funções associadas a cada botão para execução da mesma, além de possuir os comandos *CLR_DISPLAY, POS0_DISPLAY, e Send_string("")* que mostram os *states*.

Dentro das funções implementadas na rotina de interrupção são usadas os ponteiros e variáveis definidos no *ISR_VECTOR(WDT_interval_handler,".int10")* que faz o programa entrar na rotina *interrupt void* ***WDT_interval_handler****(void)* a cada 8.2 *ms* (1MHz/8k), que realiza a determinação dos endereços onde estão contidos as declarações dos valores das notas das músicas, controlando assim o que ocorre quando os botões são pressionados, por exemplo, quando o botão *song_1* for pressionado, entrará na rotina de interrupção do botão que usará a rotina de interrupção do WDT para controlar os ponteiros de acordo com a lógica da programação, ou seja, pegará o endereço onde contém o primeiro dado de nota musical da música 1 e iniciará o percorrimento de todo o vetor onde estão as notas e continuará a execução até que outro botão seja pressionado.

A cada pressionamento de botão, ambas *ISR_VECTOR(button_handler,"int02")* e *ISR_VECTOR(WDT_interval_handler,".int10")*, são chamadas e implementadas, e também é implementado o funcionamento do LCD, este deve apresentar o *state* do music player, se está tocando, pausado, quando a música 1, ou a 2 for selecionada, quando ocorre o reset para música recomeçar, ou o estado de acelerar ou desacelerar a música, para que isso ocorra, as funções *void* ***initLCD****(void)*, *void* ***Atraso_us****(volatile unsigned int us)*, *void* ***Send_Nibble***, *void* ***Send_Byte***, *void* ***Send_Data,*** *void* ***Send_String***, *void* ***Send_int*** são implementadas, todas essas funções estão associadas ao funcionamento apenas do LCD, em que basicamente o nome de cada um dos *states* é mostrado na tela, para o funcionamento do LCD foi necessário utilizar outro canal do timerA, o A1 devido a configuração de tempo diferir da configurada para o gerador de frequências, que utilizou o canal A0, isso ocorre devido a necessidade de controle do tempo de apresentação do texto escrito com uma velocidade que o olho possa ver sem problemas.

Da implementação física do projeto , exemplificado na *figura 1*, todas os pinos disponíveis da MSP430G2553 foram utilizadas, a porta P1 foi usada pelos botões, e a porta P2 para o LCD, foi pedido a implementação do LCD por meio de comunicação, de forma a evidenciar o conteúdo de comunicação serial por meio de algum dos protocolos aprendidos na aula, porém para o uso de comunicação I²C seria necessário utilizar os pinos BIT6 e BIT7 da porta P1, que são os pinos que possuem a configuração para implementação deste tipo de comunicação, então não foi possível implementar pelo uso destes pinos pelo módulo principal do projeto.

O pino P1.6 estava entregando tensão abaixo da necessária para ligamento do botão, para isso, o botão foi conectado utilizando a tensão de 5V e com um resistor de 1.2$k\Omega$ para fazer um divisor de tensão e entregar a tensão certa.

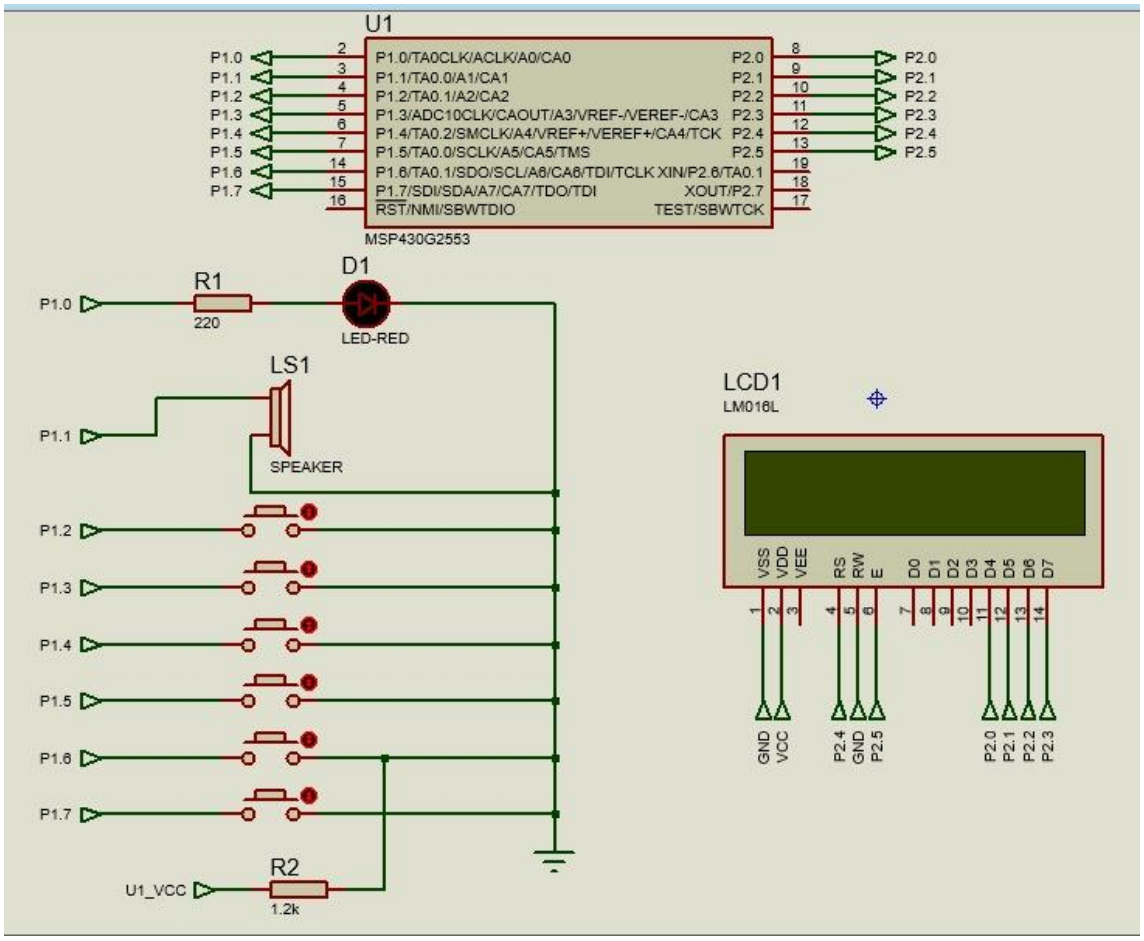

Figura 1. Esquemático da implementação física do projeto.

Todos os requisitos apresentados para o desenvolvimento do projeto foram executados, utilizando os princípios aprendidos em sala uso de WDT, Timer, ISR, comunicação serial com periférico (melhoria proposta no Ponto de controle 2), implementado na placa MSP430G2553, o programa apresentou o resultado esperado, e possui como melhoria proposta para o último ponto de controle a mudança das músicas para diferir das presentes no projeto, e a implementação do trecho de assembly que ainda não foi implementado por problemas de compilação, visto que foram utilizadas com base no material usado como referencial indicado no Ponto de controle 1.