Relatório Final

Grupo 10 - Music Player

Bruno Carvalho Faria dos Santos

Matrícula: 14/0132767
Email: bruno-fga1@hotmail.com

Elias Queiroga Vieira

Matrícula: 16/0118719
Email: eliasq_@hotmail.com

***Resumo*—Este é o relatório para a apresentação final do projeto da dupla: o music player, da disciplina eletrônica embarcada, será discuto o desenvolvimento do projeto na placa MSP430G2553, com base na proposta inicial e as dificuldades, soluções e a apresentação do protótipo funcional final .**

Palavras-Chave— Projeto, Music Player, MSP430G2553, Eletrônica Embarcada, Apresentação.

***Abstract*—This is the article for the final submition of the duo project : the music player, of Embedded Eletronics subject, it is going to be discussed the project development on MSP430G2553 board based on the initial proposition and the difficulties, the solution and functional final prototype .**

Keywords— Project, Music Player, MSP430G2553, Embedded Eletronics, Submition.

I. INTRODUÇÃO

Desde a criação dos primeiros componentes eletrônicos: os transistores, a humanidade passa por uma rápida evolução tecnológica associada a Era Informacional Tecnológica, e torna-se mais dependente do uso da tecnologia, o mundo ”*automatizável”* nada mais passa do que extenso uso de tecnologia embarcada associada aos componentes cotidianos, porém expandindo a atuação e capacidade dos mesmos.

Na matéria Eletrônica Embarcada somos apresentados a um microcontrolador específico (MSP430 da TI – Texas Instruments), sua IDE, formato de desenvolvimento, aplicações das mais diversas, e como resultado do conhecimento adquirido durante o semestre, somos desafiados a desenvolver um projeto que faça uso deste conhecimento.

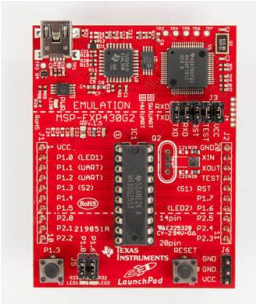

Fig 1. Placa MSP430G2553 da Texas Instruments.

O projeto desenvolvido pela dupla é um tocador de músicas, possui as funções, Play, Pause, Select Song 1, Select Song 2, Reset, Acelerar e Desacelerar, é constituído por essas funções, um LED e um speaker, difere um pouco de um music player comercial por ter restrições quantitativas e qualitativas das músicas devido a característica de implementação na placa.

O projeto foi realizado baseado em um encontrado no GitHub: cjduffett, que possui uma série de projetos aplicados em MSP430G2553, de onde foram tiradas as especificações do projeto.

O oscilador é o principal gerador do som, os VCO ou DCO geram um sinal básico que formam ondas ricas em harmônicas, porém para alguns tipos de sons, o fluxo de sinal não segue o caminho fundamental: oscilador, filtro e amplificador respectivamente, tendo a inclusão de um modulador para modular o oscilador principal, e para essa aplicação seria necessário um circuito externo ao MSP430G2553, sendo que na aplicação proposta no projeto da dupla, toda a lógica das notas das músicas será realizado no próprio microcontrolador.

Basicamente no projeto proposto é usado os conhecimentos adquiridos sobre TimerA – modo contagem e captura, WatchDog Timer - modo Interval Timer, rotinas de interrupção ISR, misturar as linguagens Assembly e C, e comunicação com periféricos.

II. DESENVOLVIMENTO

O projeto do music player consiste do uso de todas os pinos da porta P1, 2 pinos para o LED e o SPEAKER (Saídas), 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.

LISTA DE MATERIAIS:

- Placa Texas Instruments MSP430G2553;
- Protoboard;
- Speaker de 32 ohms;

- 6 botões;
- LED;
- Resistor: 1.2kΩ;
- LCD 16x2.

O tocador de música é implementado armazenando cada música em um *array*, e cada nota é armazenada em 8 bits divididos em 2 caractereres, os 3 bits mais significativos (*MSB*), **durations[]** armazenam a duração da nota (o que gera 8 durações), nos 5 bits menos significativos (*LSB*), **notes[]** armazenam a frequência das notas (o que gera 32 notas), das quais 26 são habilitadas para uso, e representam duas Oitavas completas (C4 a C6). As frequências armazenadas no *array* **notes[]** são valores de meio período. Declarado como vetor *const unsigned char[]* as duas músicas ocupam exatamente 824 caracteres, ou 824 bytes de memória.

- **durations[] =**

0. Break - Semibreve
1. 1/16th - Semifusa
2. 1/8th - Fusa
3. Dotted 1/8th - Fusa pontuada
4. 1/4th - Semicolcheia
5. Dotted 1/4th pontuada – Semicolcheia Pontuada
6. 1/2th - Colcheia
7. Dotted 1/2th - Colcheia pontuada;

Em que a Semibreve é inserida na partitura entre notas de mesma frequência, é equivalente a 1/64th de silêncio, uma nota deve ser representada por 2 ½ notas de mesma frequência sem um pausa entre elas.

- **notes[] =**

```
R,C4,C4s,D4,E4b,E4,F4,F4s,G4,G4s,A4,B4b
,B4,C5,C5s,D5,E5b,E5,F5,F5s,G5,G5s,A5,B
5b,B5,C6;
```

R de *rest* que é inaudível ao usuário, representa a frequência 0.

Quando uma nova música é selecionada, dois contadores **duration_counter** e **score_counter** são resetados, *score_counter* rastreia continuamente qual nota na música está sendo tocada, enquanto a música toca, *duration_counter* é incrementado para contar o número de interrupções do WDT por nota, depois da duração total de uma nota, o *score_counter* é incrementado e e o duration_counter é resetado. Por Exemplo, a Semicolcheia é setada no vetor **durations[NUM_DURATIONS]** caractere do tipo *const unsigned char* como valendo 32, e tendo a variável **DEFAULT_TEMPO** 2.25, temos que:

1/4th = 32 * 2.25 = 72 interrupções de WDT

TEMPO é uma variável *float* que incremente/decrementa a 0.125 s , todas as durações no vetor de durações são múltiplos de 8, possibilitando esse escalonamento linear, menos o semibreve, que ocupa 1 interrupção WDT, e tem uma duração de ~8ms.

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.

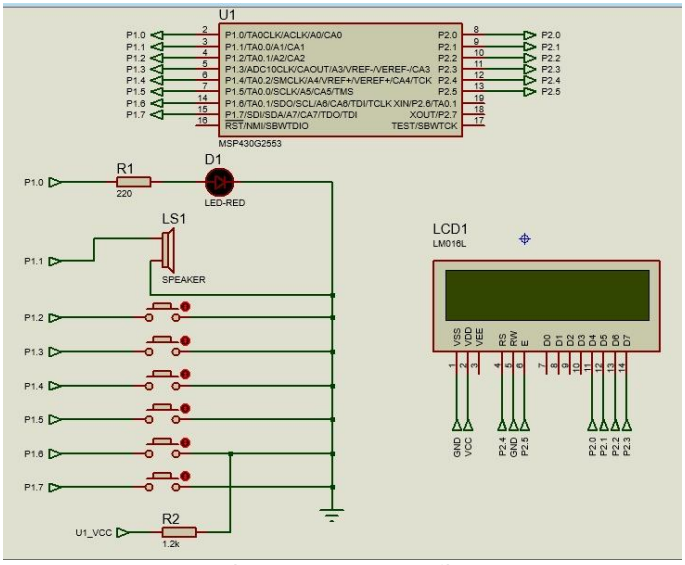

Fig 2. Esquemático da implementação física do projeto.

A main do programa é inicializado calibrando o *clk* para que o SMCLK seja do tipo *Digital Controlled Oscillator* (DCO) calibrado para funcionar a ~1MHZ.

```
BCSCTL1 = CALBC1_1MHZ;
DCOCTL  = CALDCO_1MHZ;
```

além das funções *void* ***init_timerA****(void)*, responsável pela configuração do canal A0 do TimerA (uso da SMCLK = 1.1MHz), configurado no modo UP, divisão do clk por 0, e que também é usado no programa no modo comparação, com o objetivo de gerar interrupções em intervalos de tempos específicos (a utilidade deste modo é gerar sinais PWM – *Pulse Width Modulation*),

```
TA0CTL |= TACLR;
TA0CTL = (TASSEL_2 + ID_0+ MC_1);
```

void ***init_WDT****(void)* como 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.

```
WDTCTL = (WDTPW+WDTTMSEL+WDTCNTCL+0+1);
IE1 |= WDTIE;
```

A interrupção de cada porta é realizado pela rotina *interrupt void* ***button_handler*** (*void*), e que chama as funções associadas a cada botão para execução da mesma, *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, além de possuir os comandos *CLR_DISPLAY, POS0_DISPLAY, e Send_string("")* que mostram os *states* de acordo com a função *void* ***InitLCD****()*.

Como um dos critérios para implementação do projeto, em algum trecho do Código deveria ter uma rotina implementada em linguagem assembly, de maneira a integrar as linguagens C – Apêndice A - e assembly – Apêndice B - , que permite uma melhor utilização do microcontrolador e configurações mais complexas de controle que apenas a linguagem C não seria capaz de implementar, como controle dos clks, Timers e WDT, e interação com periféricos. Para isso foi usado uma metodologia que criava um arquivo ***funcao****.asm* (arquivo de assembly) para implementar, no caso do nosso projeto, as configurações do pino SPEAKER setado na porta P1, que é chamado no arquivo com o Código em C usando *extern void* ***funcao*** *(void).*

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.

III. RESULTADOS

A implementação do projeto seguiu o esquema da fig.2., o funcionamento ocorreu perfeitamente, quando iniciado, o sistema, a função default dele é mostrar a mensagem "Aperte Play", fig.3, para que a música comece a tocar, a cada pressionamento de botão, a *string* com o nome do botão é apresentado na tela do LCD, e a função associada àquele botão era executada perfeitamente, porém o projeto possui limitações práticas, como cada nota na música é armazenada em um único byte (para indicar frequência da nota e duração), a capacidade de músicas no projeto é limitado a quantidade de bytes livres na memória, e como só 5 bits são usados para frequência, o projeto é limitado a duas oitavas, logo, músicas que necessitam de mais tons, não terão como ser implementadas sem alterações, os 3 bits para duração das notas também é um limitador, caso uma música tenha uma quartifusa, semifusa pontuada, ou terceto de fusa, a música também terá que sofrer alterações.

Estima-se que aproximadamente 10 minutos de música podem ser armazenados, o que poderia ser aumentado utilizando outra placa da família MSP430X que possua mais memória disponível do que a G2553, que possui apenas 512 Bytes de RAM e 2K de flash.

O pino 1.6 estava entregando tensão abaixo da necessária para ligamento do botão, para isso o botão foi conectado utilizando a tensão de 5V com um resistor de 1.2 kΩ para fazer um divisor de tensão e entregar a tensão certa.

Fig 3. Sistema funcionando com mensagem de default para inicialização das músicas.

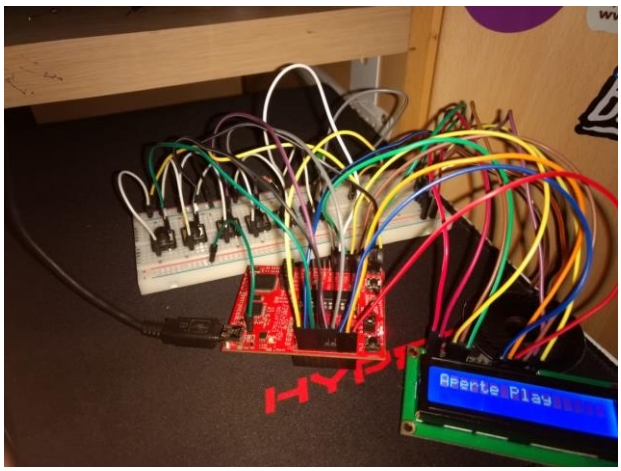

Fig 4. Projeto final montado na protoboard.

IV. CONCLUSÃO

O projeto foi bem desafiador, a lógica da implementação das notas musicais em função da frequência e tempo de duração foi baseada no projeto usado como referencia especificado no ponto de controle 1, o uso de todas as funções e conhecimentos adquiridos ao longo do semestre ocorreram concomitantemente com a apresentação dos conteúdos, os desafios da lógica de implementação exigiu um tanto quanto desenvolvimento de software e hardware, porém tanto a proposta inicial quanto a sugestão de mudança proposta durante os pontos de controle foram apresentados. Esperava-se a comunicação com o periférico, LCD, fosse feito utilizando protocolos de comunicação, porém devido a falta do módulo I²C, e notando que a ativação do LCD poderia ser feito utilizando apenas os pinos da placa, preferiu-se utilizar essa metodologia.

É notável a ampla diversidade de aplicações que o microcontrolador MSP430G2553 pode ter, a comunicação com os periféricos, configurações que permitem controlar tempos de reação do *clock*, interação do usuário com o projeto, ou simplesmente determinação de uma série de parâmetros que controlarão o sistema, além dos módulos que não foram usados no projeto, mas que tem funções variadas e exploráveis em termos práticos e teóricos.

Algumas sugestões de melhoria propostas são, o MSP430G2553 tem pouca disponibilidade de memória, que é um dos critérios que restringe a ampliação do trabalho, logo, o uso de outra placa com mais memória pode permitir que o tamanho das músicas possa ser ampliado, e a questão de implementar algum protocolo de comunicação para efeito de aumentar o uso dos tópicos da disciplina no trabalho, e também possibilitar o uso de menos pinos da placa, o que permitiria que mais periféricos fossem implementados, caso houvesse necessidade.

V. REVISÃO BIBLIOGRÁFICA

[1]<https://github.com/cjduffett/MSP430Projects/tree/master/Music%20Player> .

VI. BIBLIOGRAFIA

[1] Ünsalan, C., Gürhan, H. D., ***Programmable Microcontrollers with Applications – MSP430 Launchpad with CCS and Grace***, McGraw-Hill, 2014.
[2] Davies, J., ***MSP430 Microcontroller Basics***, Elsevier, 2008.

VII. APÊNDICE A

```
#include <msp430g2553.h>
extern void funcao (void);

//---------------------------------
#define LED            0x01
#define SPEAKER        0x02
#define RESET_BUTTON   0x04
#define PLAY_BUTTON    0x08
#define SLOW_BUTTON    0x10
#define FAST_BUTTON    0x20
#define SONG1_BUTTON   0X40
#define SONG2_BUTTON   0x80
#define LCD_OUT P2OUT
#define LCD_DIR P2DIR
#define D4_1 BIT0
#define D5_1 BIT1
#define D6_1 BIT2
#define D7 BIT3
#define RS BIT4
#define E  BIT5
#define DADOS 1
#define COMANDO 0
#define CMND_DLY 1000
#define DATA_DLY 1000
#define BIG_DLY  20000
#define CLR_DISPLAY Send_Byte(1, COMANDO, BIG_DLY)
#define POS0_DISPLAY Send_Byte(2, COMANDO, BIG_DLY)

//-----------------------------------------
#define NUM_NOTES   32

#define _R      1

#define _C4     1911
#define _C4s    1803
#define _D4     1703
#define _E4b    1607
#define _E4     1517
#define _F4     1432
#define _F4s    1351
#define _G4     1276
#define _G4s    1204
#define _A4     1136
#define _B4b    1073
#define _B4     1012
#define _C5     956
#define _C5s    902
#define _D5     851
#define _E5b    803
#define _E5     758
#define _F5     716
#define _F5s    676
#define _G5     638
#define _G5s    602
#define _A5     568
#define _B5b    536
#define _B5     506
#define _C6     478
#define _C6s    451
#define _D6     426
#define _E6b    401
#define _E6     379
#define _F6     358
#define _F6s    338
#define _G6     319
#define _G6s    301
#define _A6     285
#define _B6b    268
#define _B6     253
#define _C7     239

const unsigned int notes[NUM_NOTES] = {
        _R,

_C4,_C4s,_D4,_E4b,_E4,_F4,_F4s,_G4,_G4s,_A4,_B4b,_B4,

_C5,_C5s,_D5,_E5b,_E5,_F5,_F5s,_G5,_G5s,_A5,_B5b,_B5,
        _C6,_C6s,_D6,_E6b,_E6,_F6,_F6s
};

#define R       0x00
#define C4      0x01
#define C4s     0x02
#define D4      0x03
#define E4b     0x04
#define E4      0x05
#define F4      0x06
#define F4s     0x07
#define G4      0x08
#define G4s     0x09
#define A4      0x0A
#define B4b     0x0B
#define B4      0x0C
#define C5      0x0D
#define C5s     0x0E
#define D5      0x0F
```

```
#define E5b     0x10
#define E5      0x11
#define F5      0x12
#define F5s     0x13
#define G5      0x14
#define G5s     0x15
#define A5      0x16
#define B5b     0x17
#define B5      0x18
#define C6      0x19
#define C6s     0x1A
#define D6      0x1B
#define E6b     0x1C
#define E6      0x1D
#define F6      0x1E
#define F6s     0x1F

#define NOTE_MASK   0x1F
//----------------------------------
#define NUM_DURATIONS   8

const unsigned char durations[NUM_DURATIONS] = {
      1,
      8,
      16,
      24,
      32,
      48,
      64,
      96
};

#define BREAK       0x00
#define SIXTEENTH   0x20
#define EIGTH       0x40
#define D_EIGTH     0x60
#define QUARTER     0x80
#define D_QUARTER   0xA0
#define HALF        0xC0
#define D_HALF      0xE0

// Tetris
//----------------------------------
#define SONG1_LENGTH    406

unsigned const char song1[SONG1_LENGTH] = {
                        (EIGTH + E6), (EIGTH + E4), (EIGTH + B5), (EIGTH + C6), (EIGTH + D6), (SIXTEENTH + E6), (SIXTEENTH + D6), (EIGTH + C6),
                        (EIGTH + B5), (EIGTH + A5), (BREAK + R),  (EIGTH + A5), (BREAK + R),  (EIGTH + A5), (EIGTH + C6), (EIGTH + E6), (EIGTH + A5), (EIGTH + D6),
                        (EIGTH + C6), (EIGTH + B5), (EIGTH + E5), (EIGTH + G5), (EIGTH + C6), (EIGTH + D6), (EIGTH + E4), (EIGTH + E6),
                        (EIGTH + E4), (EIGTH + C6), (EIGTH + A5), (BREAK + R),  (EIGTH + A5), (BREAK + R),  (EIGTH + A5), (BREAK + R),  (EIGTH + A5), (EIGTH + B5),
                        (EIGTH + C4), (EIGTH + D4), (QUARTER +D6), (EIGTH + F6), (EIGTH + A5), (SIXTEENTH + C6), (SIXTEENTH + C6), (EIGTH + G5),
                        (EIGTH + F6), (EIGTH + E6), (EIGTH + C4), (EIGTH + R), (EIGTH + C6), (EIGTH + E6), (SIXTEENTH + A5), (SIXTEENTH + G5), (EIGTH + D6),
                        (EIGTH + C6), (EIGTH + B5), (EIGTH + E5), (EIGTH + B5), (EIGTH + C6), (EIGTH + D6), (EIGTH + G5), (EIGTH + E6),
                        (EIGTH + G5), (EIGTH + C6), (EIGTH + E5), (EIGTH + A5), (EIGTH + E4), (QUARTER +A5), (QUARTER +R),
                        (EIGTH + E6), (EIGTH + E4), (EIGTH + B5), (EIGTH + C6), (EIGTH + D6), (SIXTEENTH + E6), (SIXTEENTH + D6), (EIGTH + C6),
                        (EIGTH + B5), (EIGTH + A5), (BREAK + R),  (EIGTH + A5), (BREAK + R),  (EIGTH + A5), (EIGTH + C6), (EIGTH + E6), (EIGTH + A5), (EIGTH + D6),
                        (EIGTH + C6), (EIGTH + B5), (EIGTH + E5), (EIGTH + G5), (EIGTH + C6), (EIGTH + D6), (EIGTH + E4), (EIGTH + E6),
                        (EIGTH + E4), (EIGTH + C6), (EIGTH + A5), (BREAK + R),  (EIGTH + A5), (BREAK + R),  (EIGTH + A5), (BREAK + R),  (EIGTH + A5), (EIGTH + B5),
                        (EIGTH + C4), (EIGTH + D4), (QUARTER +D6), (EIGTH + F6), (EIGTH + A5), (SIXTEENTH + C6), (SIXTEENTH + C6), (EIGTH + G5),
                        (EIGTH + F6), (EIGTH + E6), (EIGTH + C4), (EIGTH + R), (EIGTH + C6), (EIGTH + E6), (SIXTEENTH + A5), (SIXTEENTH + G5), (EIGTH + D6),
                        (EIGTH + C6), (EIGTH + B5), (EIGTH + E5), (EIGTH + B5), (EIGTH + C6), (EIGTH + D6), (EIGTH + G5), (EIGTH + E6),
                        (EIGTH + G5), (EIGTH + C6), (EIGTH + E5), (EIGTH + A5), (EIGTH + E4), (QUARTER +A5), (QUARTER +R),
                        (EIGTH + E5), (EIGTH + E4), (EIGTH + A4), (EIGTH + E4), (EIGTH + C5), (EIGTH + E4), (EIGTH + A4), (EIGTH + E4),
                        (EIGTH + D5), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4), (EIGTH + B5), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4),
                        (EIGTH + C5), (EIGTH + E4), (EIGTH + A4), (EIGTH + E4), (EIGTH + A5), (EIGTH + E4), (EIGTH + A4), (EIGTH + E4),
                        (EIGTH + G5s), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4), (EIGTH + B5), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4),
                        (EIGTH + E5), (EIGTH + E4), (EIGTH + A4), (EIGTH + E4), (EIGTH + C5), (EIGTH + E4), (EIGTH + A4), (EIGTH + E4),
                        (EIGTH + D5), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4), (EIGTH + B5), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4),
                        (EIGTH + C5), (EIGTH + E4), (EIGTH + E5), (EIGTH + E4), (EIGTH + A5), (EIGTH + E4), (EIGTH + A4), (EIGTH + E4),
                        (EIGTH + G5s), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4),
```

```
                (EIGTH + E6), (EIGTH + E4), (EIGTH + B5), (EIGTH + C6), (EIGTH + D6), (SIXTEENTH + E6), (SIXTEENTH + D6), (EIGTH + C6),
                (EIGTH + B5), (EIGTH + A5), (BREAK + R), (EIGTH + A5), (BREAK + R), (EIGTH + A5), (EIGTH + C6), (EIGTH + E6), (EIGTH + A5), (EIGTH + D6),
                (EIGTH + C6), (EIGTH + B5), (EIGTH + E5), (EIGTH + G5), (EIGTH + C6), (EIGTH + D6), (EIGTH + E4), (EIGTH + E6),
                (EIGTH + E4), (EIGTH + C6), (EIGTH + A5), (BREAK + R), (EIGTH + A5), (BREAK + R), (EIGTH + A5), (BREAK + R), (EIGTH + A5), (EIGTH + B5),
                (EIGTH + C4), (EIGTH + D4), (QUARTER +D6), (EIGTH + F6), (EIGTH + A5), (SIXTEENTH + C6), (SIXTEENTH + C6), (EIGTH + G5),
                (EIGTH + F6), (EIGTH + E6), (EIGTH + C4), (EIGTH + R), (EIGTH + C6), (EIGTH + E6), (SIXTEENTH + A5), (SIXTEENTH + G5), (EIGTH + D6),
                (EIGTH + C6), (EIGTH + B5), (EIGTH + E5), (EIGTH + B5), (EIGTH + C6), (EIGTH + D6), (EIGTH + G5), (EIGTH + E6),
                (EIGTH + G5), (EIGTH + C6), (EIGTH + E5), (EIGTH + A5), (EIGTH + E4), (QUARTER +A5), (QUARTER +R),
                (EIGTH + E6), (EIGTH + E4), (EIGTH + B5), (EIGTH + C6), (EIGTH + D6), (SIXTEENTH + E6), (SIXTEENTH + D6), (EIGTH + C6),
                (EIGTH + B5), (EIGTH + A5), (BREAK + R), (EIGTH + A5), (BREAK + R), (EIGTH + A5), (EIGTH + C6), (EIGTH + E6), (EIGTH + A5), (EIGTH + D6),
                (EIGTH + C6), (EIGTH + B5), (EIGTH + E5), (EIGTH + G5), (EIGTH + C6), (EIGTH + D6), (EIGTH + E4), (EIGTH + E6),
                (EIGTH + E4), (EIGTH + C6), (EIGTH + A5), (BREAK + R), (EIGTH + A5), (BREAK + R), (EIGTH + A5), (BREAK + R), (EIGTH + A5), (BREAK + R), (EIGTH + A5), (EIGTH + B5),
                (EIGTH + C4), (EIGTH + D4), (QUARTER +D6), (EIGTH + F6), (EIGTH + A5), (SIXTEENTH + C6), (SIXTEENTH + C6), (EIGTH + G5),
                (EIGTH + F6), (EIGTH + E6), (EIGTH + C4), (EIGTH + R), (EIGTH + C6), (EIGTH + E6), (SIXTEENTH + A5), (SIXTEENTH + G5), (EIGTH + D6),
                (EIGTH + C6), (EIGTH + B5), (EIGTH + E5), (EIGTH + B5), (EIGTH + C6), (EIGTH + D6), (EIGTH + G5), (EIGTH + E6),
                (EIGTH + G5), (EIGTH + C6), (EIGTH + E5), (EIGTH + A5), (EIGTH + E4), (QUARTER +A5), (QUARTER +R),
                (EIGTH + E5), (EIGTH + E4), (EIGTH + A4), (EIGTH + E4), (EIGTH + C5), (EIGTH + E4), (EIGTH + A4), (EIGTH + E4),
                (EIGTH + D5), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4), (EIGTH + B5), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4),
                (EIGTH + C5), (EIGTH + E4), (EIGTH + A4), (EIGTH + E4), (EIGTH + A5), (EIGTH + E4), (EIGTH + A4), (EIGTH + E4),
                (EIGTH + G5s), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4), (EIGTH + B5), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4),
                (EIGTH + E5), (EIGTH + E4), (EIGTH + A4), (EIGTH + E4), (EIGTH + C5), (EIGTH + E4), (EIGTH + A4), (EIGTH + E4),
                (EIGTH + D5), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4), (EIGTH + B5), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4),
                (EIGTH + C5), (EIGTH + E4), (EIGTH + E5), (EIGTH + E4), (EIGTH + A5), (EIGTH + E4), (EIGTH + A4), (EIGTH + E4),
                (EIGTH + G5s), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4), (EIGTH + G4s), (EIGTH + E4)
};

// "Super Mario Theme"
//---------------------------------
#define SONG2_LENGTH    418

unsigned const char song2[SONG2_LENGTH] = {
    (SIXTEENTH + E5),
    (BREAK + R),
    (SIXTEENTH + E5),
    (SIXTEENTH + R),
    (SIXTEENTH + E5),
    (SIXTEENTH + R),
    (SIXTEENTH + C5),
    (EIGTH + E5),
    (EIGTH + G5),
    (EIGTH + R),
    (EIGTH + G4),
    (EIGTH + R),
    (D_EIGTH + C5),
    (SIXTEENTH + G4),
    (EIGTH + R),
    (EIGTH + E4),
    (SIXTEENTH + E4),
    (EIGTH + A4),
    (EIGTH + B4),
    (SIXTEENTH + B4b),
    (EIGTH + A4),
    (SIXTEENTH + G4),
    (EIGTH + E5),
    (SIXTEENTH + G5),
    (EIGTH + A5),
    (SIXTEENTH + F5),
    (SIXTEENTH + G5),
    (SIXTEENTH + R),
    (EIGTH + E5),
    (SIXTEENTH + C5),
    (SIXTEENTH + D5),
    (EIGTH + B4),
    (SIXTEENTH + R),
    (D_EIGTH + C5),
    (SIXTEENTH + G4),
    (EIGTH + R),
    (EIGTH + E4),
    (SIXTEENTH + E4),
    (EIGTH + A4),
    (EIGTH + B4),
```

(SIXTEENTH + B4b),
(EIGTH + A4),
(SIXTEENTH + G4),
(EIGTH + E5),
(SIXTEENTH + G5),
(EIGTH + A5),
(SIXTEENTH + F5),
(SIXTEENTH + G5),
(SIXTEENTH + R),
(EIGTH + E5),
(SIXTEENTH + C5),
(SIXTEENTH + D5),
(EIGTH + B4),
(SIXTEENTH + R),
(EIGTH + R),
(SIXTEENTH + G5),
(SIXTEENTH + F5s),
(SIXTEENTH + F5),
(EIGTH + E5b),
(SIXTEENTH + E5),
(SIXTEENTH + R),
(SIXTEENTH + G4s),
(SIXTEENTH + A4),
(SIXTEENTH + C5),
(SIXTEENTH + R),
(SIXTEENTH + A4),
(SIXTEENTH + C5),
(SIXTEENTH + D5),
(EIGTH + R),
(SIXTEENTH + G5),
(SIXTEENTH + F5s),
(SIXTEENTH + F5),
(EIGTH + E5b),
(SIXTEENTH + E5),
(SIXTEENTH + R),
(SIXTEENTH + C6),
(SIXTEENTH + R),
(SIXTEENTH + C6),
(BREAK + R),
(QUARTER + C6),
(EIGTH + R),
(SIXTEENTH + G5),
(SIXTEENTH + F5s),
(SIXTEENTH + F5),
(EIGTH + E5b),
(SIXTEENTH + E5),
(SIXTEENTH + R),
(SIXTEENTH + G4s),
(SIXTEENTH + A4),
(SIXTEENTH + C5),
(SIXTEENTH + R),
(SIXTEENTH + A4),
(SIXTEENTH + C5),
(SIXTEENTH + D5),
(EIGTH + R),
(EIGTH + E5b),
(SIXTEENTH + R),
(EIGTH + D5),
(SIXTEENTH + R),
(QUARTER + C5),
(QUARTER + R),
(EIGTH + R),
(SIXTEENTH + G5),
(SIXTEENTH + F5s),
(SIXTEENTH + F5),
(EIGTH + E5b),
(SIXTEENTH + E5),
(SIXTEENTH + R),
(SIXTEENTH + G4s),
(SIXTEENTH + A4),
(SIXTEENTH + C5),
(SIXTEENTH + R),
(SIXTEENTH + A4),
(SIXTEENTH + C5),
(SIXTEENTH + D5),
(EIGTH + R),
(EIGTH + E5b),
(SIXTEENTH + R),
(EIGTH + D5),
(SIXTEENTH + R),
(QUARTER + C5),
(QUARTER + R),
(SIXTEENTH + C5),
(BREAK + R),
(EIGTH + C5),
(BREAK + R),
(SIXTEENTH + C5),
(SIXTEENTH + R),
(SIXTEENTH + C5),
(EIGTH + D5),
(SIXTEENTH + E5),
(EIGTH + C5),
(SIXTEENTH + A4),
(QUARTER + G4),
(SIXTEENTH + C5),
(BREAK + R),
(EIGTH + C5),
(BREAK + R),

(SIXTEENTH + C5),
(SIXTEENTH + R),
(SIXTEENTH + C5),
(SIXTEENTH + D5),
(SIXTEENTH + E5),
(HALF + R),
(SIXTEENTH + C5),
(BREAK + R),
(EIGTH + C5),
(BREAK + R),
(SIXTEENTH + C5),
(SIXTEENTH + R),
(SIXTEENTH + C5),
(EIGTH + D5),
(SIXTEENTH + E5),
(EIGTH + C5),
(SIXTEENTH + A4),
(QUARTER + G4),
(SIXTEENTH + E5),
(BREAK + R),
(SIXTEENTH + E5),
(SIXTEENTH + R),
(SIXTEENTH + E5),
(SIXTEENTH + R),
(SIXTEENTH + C5),
(EIGTH + E5),
(EIGTH + G5),
(EIGTH + R),
(EIGTH + G4),
(EIGTH + R),
(D_EIGTH + C5),
(SIXTEENTH + G4),
(EIGTH + R),
(EIGTH + E4),
(SIXTEENTH + E4),
(EIGTH + A4),
(EIGTH + B4),
(SIXTEENTH + B4b),
(EIGTH + A4),
(SIXTEENTH + G4),
(EIGTH + E5),
(SIXTEENTH + G5),
(EIGTH + A5),
(SIXTEENTH + F5),
(SIXTEENTH + G5),
(SIXTEENTH + R),
(EIGTH + E5),
(SIXTEENTH + C5),
(SIXTEENTH + D5),
(EIGTH + B4),
(SIXTEENTH + R),
(D_EIGTH + C5),
(SIXTEENTH + G4),
(EIGTH + R),
(EIGTH + E4),
(SIXTEENTH + E4),
(EIGTH + A4),
(EIGTH + B4),
(SIXTEENTH + B4b),
(EIGTH + A4),
(SIXTEENTH + G4),
(EIGTH + E5),
(SIXTEENTH + G5),
(EIGTH + A5),
(SIXTEENTH + F5),
(SIXTEENTH + G5),
(SIXTEENTH + R),
(EIGTH + E5),
(SIXTEENTH + C5),
(SIXTEENTH + D5),
(EIGTH + B4),
(SIXTEENTH + R),
(SIXTEENTH + E5),
(EIGTH + C5),
(SIXTEENTH + G4),
(EIGTH + R),
(EIGTH + G4s),
(SIXTEENTH + A4),
(EIGTH + F5),
(BREAK + R),
(SIXTEENTH + F5),
(QUARTER + A4),
(SIXTEENTH + B4),
(EIGTH + A5),
(BREAK + R),
(SIXTEENTH + A5),
(BREAK + R),
(SIXTEENTH + A5),
(EIGTH + G5),
(SIXTEENTH + F5),
(SIXTEENTH + E5),
(EIGTH + C5),
(SIXTEENTH + A4),
(QUARTER + G4),
(SIXTEENTH + E5),
(EIGTH + C5),
(SIXTEENTH + G4),
(EIGTH + R),
(EIGTH + G4s),
(SIXTEENTH + A4),
(EIGTH + F5),
(BREAK + R),
(SIXTEENTH + F5),
(QUARTER + A4),
(SIXTEENTH + B4),
(EIGTH + F5),
(BREAK + R),
(SIXTEENTH + F5),
(BREAK + R),
(SIXTEENTH + F5),
(EIGTH + E5),
(SIXTEENTH + D5),
(QUARTER + C5),
(QUARTER + C4),
(SIXTEENTH + E5),
(EIGTH + C5),
(SIXTEENTH + G4),
(EIGTH + R),
(EIGTH + G4s),
(SIXTEENTH + A4),
(EIGTH + F5),
(BREAK + R),
(SIXTEENTH + F5),
(QUARTER + A4),

(SIXTEENTH + B4),
(EIGTH + A5),
(BREAK + R),
(SIXTEENTH + A5),
(BREAK + R),
(SIXTEENTH + A5),
(EIGTH + G5),
(SIXTEENTH + F5),
(SIXTEENTH + E5),
(EIGTH + C5),
(SIXTEENTH + A4),
(QUARTER + G4),
(SIXTEENTH + E5),
(EIGTH + C5),
(SIXTEENTH + G4),
(EIGTH + R),
(EIGTH + G4s),
(SIXTEENTH + A4),
(EIGTH + F5),
(BREAK + R),
(SIXTEENTH + F5),
(QUARTER + A4),
(SIXTEENTH + B4),
(EIGTH + F5),
(BREAK + R),
(SIXTEENTH + F5),
(BREAK + R),
(SIXTEENTH + F5),
(EIGTH + E5),
(SIXTEENTH + D5),
(QUARTER + C5),
(QUARTER + C4),
(SIXTEENTH + C5),
(BREAK + R),
(EIGTH + C5),
(BREAK + R),
(SIXTEENTH + C5),
(SIXTEENTH + R),
(SIXTEENTH + C5),
(EIGTH + D5),
(SIXTEENTH + E5),
(EIGTH + C5),
(SIXTEENTH + A4),
(QUARTER + G4),
(SIXTEENTH + C5),
(BREAK + R),
(EIGTH + C5),
(BREAK + R),
(SIXTEENTH + C5),
(SIXTEENTH + R),
(SIXTEENTH + C5),
(SIXTEENTH + D5),
(SIXTEENTH + E5),
(HALF + R),
(SIXTEENTH + C5),
(BREAK + R),
(EIGTH + C5),
(BREAK + R),
(SIXTEENTH + C5),
(SIXTEENTH + R),
(SIXTEENTH + C5),
(EIGTH + D5),
(SIXTEENTH + E5),
(EIGTH + C5),
(SIXTEENTH + A4),
(QUARTER + G4),
(SIXTEENTH + E5),
(BREAK + R),
(SIXTEENTH + E5),
(SIXTEENTH + R),
(SIXTEENTH + E5),
(SIXTEENTH + R),
(SIXTEENTH + C5),
(EIGTH + E5),
(EIGTH + G5),
(EIGTH + R),
(EIGTH + G4),
(EIGTH + R),
(SIXTEENTH + E5),
(EIGTH + C5),
(SIXTEENTH + G4),
(EIGTH + R),
(EIGTH + G4s),
(SIXTEENTH + A4),
(EIGTH + F5),
(BREAK + R),
(SIXTEENTH + F5),
(QUARTER + A4),
(SIXTEENTH + B4),
(EIGTH + A5),
(BREAK + R),
(SIXTEENTH + A5),
(BREAK + R),
(SIXTEENTH + A5),
(EIGTH + G5),
(SIXTEENTH + F5),
(SIXTEENTH + E5),
(EIGTH + C5),
(SIXTEENTH + A4),
(QUARTER + G4),
(SIXTEENTH + E5),
(EIGTH + C5),
(SIXTEENTH + G4),
(EIGTH + R),
(SIXTEENTH + G4s),
(SIXTEENTH + A4),
(EIGTH + F5),
(BREAK + R),
(SIXTEENTH + F5),
(QUARTER + A4),
(SIXTEENTH + B4),
(EIGTH + F5),
(BREAK + R),
(SIXTEENTH + F5),
(BREAK + R),
(SIXTEENTH + F5),
(EIGTH + E5),
(SIXTEENTH + D5),
(QUARTER + C5),
(QUARTER + C4),
(D_EIGTH + C5),
(D_EIGTH + G4),
(EIGTH + E4),
(SIXTEENTH + A4),

```
    (EIGTH + B4),
    (SIXTEENTH + A4),
    (SIXTEENTH + G4s),
    (EIGTH + B4b),
    (SIXTEENTH + A4),
    (HALF + G4)
};

//-----------------------------------
#define DEFAULT_TEMPO   1.45
#define FLASH_INTERVAL  30

unsigned char sys_mod = 2;
unsigned int curr_song_len = SONG1_LENGTH;
const unsigned char *curr_song = song1;
unsigned int duration_counter = 0;
unsigned int score_counter = 0;
unsigned char flash_counter = 0;
float tempo = DEFAULT_TEMPO;
unsigned char isbreak = 0;


//-----------------------------------
void Atraso_us(volatile unsigned int us);
void Send_Nibble(volatile unsigned char nibble, volatile
unsigned char dados, volatile unsigned int microsegs);
void Send_Byte(volatile unsigned char byte, volatile unsigned
char dados, volatile unsigned int microsegs);
void Send_Data(volatile unsigned char byte);
void Send_String(char str[]);
void Send_Int(int n);
void InitLCD(void);
void init_timerA(void);
void init_WDT(void);
void init_P1(void);
void toggle_pause(void);
void restart_song(void);
void increase_tempo(void);
void decrease_tempo(void);
void select_song1(void);
void select_song2(void);

// main
//-----------------------------------
void main() {
    BCSCTL1 = CALBC1_1MHZ;
    DCOCTL  = CALDCO_1MHZ;

    init_P1();
    init_WDT();
    init_timerA();
    restart_song();
    InitLCD();
    Send_String("Aperte Play");
    tempo = DEFAULT_TEMPO;

    _bis_SR_register(GIE+LPM0_bits);
}

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

void init_timerA(void) {
    TA0CTL |= TACLR;
    TA0CTL = (TASSEL_2 +    // clock source = SMCLK
            ID_0 +      // clock divider = 1
            MC_1);      // UP mode

    TA0CCTL0=0;
}

void init_WDT(void) {

     WDTCTL =  (WDTPW +     // password
            WDTTMSEL + // select interval timer mode
            WDTCNTCL + // clear watchdog timer counter
            0 +        // SMCLK is the source
            1);        // source/8k

    IE1 |= WDTIE;
}

void init_P1(void) {

    funcao();
    P1DIR |= LED;
    P1OUT |= LED;

    //P1SEL |= SPEAKER;
    //P1DIR |= SPEAKER;

    P1OUT |= RESET_BUTTON;  // pullup
    P1REN |= RESET_BUTTON;  // enable pullup resistor
    P1IES |= RESET_BUTTON;  // set for 1->0 transition
    P1IFG &= ~RESET_BUTTON; // clear interrupt flag
    P1IE  |= RESET_BUTTON;  // enable interrupt

    P1OUT |= PLAY_BUTTON;   // pullup
    P1REN |= PLAY_BUTTON;   // enable pullup resistor
    P1IES |= PLAY_BUTTON;   // set for 1->0 transition
    P1IFG &= ~PLAY_BUTTON;  // clear interrupt flag
    P1IE  |= PLAY_BUTTON;   // enable interrupt

    P1OUT |= SLOW_BUTTON;   // pullup
    P1REN |= SLOW_BUTTON;   // enable pullup resistor
    P1IES |= SLOW_BUTTON;   // set for 1->0 transition
    P1IFG &= ~SLOW_BUTTON;  // clear interrupt flag
    P1IE  |= SLOW_BUTTON;   // enable interrupt

    P1OUT |= FAST_BUTTON;   // pullup
    P1REN |= FAST_BUTTON;   // enable pullup resistor
    P1IES |= FAST_BUTTON;   // set for 1->0 transition
    P1IFG &= ~FAST_BUTTON;  // clear interrupt flag
    P1IE  |= FAST_BUTTON;   // enable interrupt

    P1OUT |= SONG1_BUTTON;  // pullup
    P1REN |= SONG1_BUTTON;  // enable pullup resistor
    P1IES |= SONG1_BUTTON;  // set for 1->0 transition
    P1IFG &= ~SONG1_BUTTON; // clear interrupt flag
    P1IE  |= SONG1_BUTTON;  // enable interrupt

    P1OUT |= SONG2_BUTTON;  // pullup
    P1REN |= SONG2_BUTTON;  // enable pullup resistor
    P1IES |= SONG2_BUTTON;  // set for 1->0 transition
```

```
   P1IFG &= ~SONG2_BUTTON; // clear interrupt flag
   P1IE  |= SONG2_BUTTON;  // enable interrupt
}

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

void toggle_pause(void) {

   if (sys_mod != 3) {

      TACCTL0 ^= OUTMOD_4;
      sys_mod = !sys_mod;
      CLR_DISPLAY;
           POS0_DISPLAY;
           Send_String("Pausado");
      if (!sys_mod) {
         P1OUT &= ~LED;

            CLR_DISPLAY;
            POS0_DISPLAY;
            Send_String("Tocando");

      }
   }
}

void decrease_tempo(void) {
   if (tempo < 10.0) {
      tempo += 0.125;
   }
}

void increase_tempo(void) {
   if(tempo > 0.125) {
      tempo -= 0.125;
   }
}

void restart_song(void) {
   TA0CTL |= TACLR;
   TACCTL0 &= ~OUTMOD_4;
   duration_counter = 0;
   score_counter = 0;
   isbreak = 0;
   sys_mod = 2;
   TA0CCR0 = notes[(curr_song[0] & NOTE_MASK)]-1;
   P1OUT |= LED;
}

void select_song1(void) {
   sys_mod = 2;
   curr_song = song1;
   curr_song_len = SONG1_LENGTH;
   tempo = DEFAULT_TEMPO;
   restart_song();
}

void select_song2(void) {
   sys_mod = 2;
   curr_song = song2;
   curr_song_len = SONG2_LENGTH;
   tempo = 2.25;
   restart_song();
}

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

interrupt void button_handler(void) {

   if (P1IFG & RESET_BUTTON) {
      P1IFG &= ~RESET_BUTTON;
      restart_song();
      CLR_DISPLAY;
      POS0_DISPLAY;
      Send_String("Reset");
   }
   else if (P1IFG & PLAY_BUTTON) {
      P1IFG &= ~PLAY_BUTTON;
      toggle_pause();
   }
   else if (P1IFG & SLOW_BUTTON) {
      P1IFG &= ~SLOW_BUTTON;
      decrease_tempo();
      CLR_DISPLAY;
      POS0_DISPLAY;
      Send_String("Desacelerar");
   }
   else if (P1IFG & FAST_BUTTON) {
      P1IFG &= ~FAST_BUTTON;
      increase_tempo();
      CLR_DISPLAY;
      POS0_DISPLAY;
      Send_String("Acelerar");
   }
   else if (P1IFG & SONG1_BUTTON) {
      P1IFG &= ~SONG1_BUTTON;
      select_song1();
      CLR_DISPLAY;
      POS0_DISPLAY;
      Send_String("Tetris");
   }
   else if (P1IFG & SONG2_BUTTON) {
      P1IFG &= ~SONG2_BUTTON;
      select_song2();
      CLR_DISPLAY;
      POS0_DISPLAY;
      Send_String("Super Mario");
   }
}

interrupt void WDT_interval_handler(void) {

   if (sys_mod == 0) {

      if (score_counter < curr_song_len) {

         if (!isbreak) {
            if (duration_counter >=
(durations[(curr_song[score_counter] >> 5)]) * tempo) {

               duration_counter = 0;
               score_counter++;
```

```
            TA0CCR0 = notes[(curr_song[score_counter] &
NOTE_MASK)];

            if (durations[(curr_song[score_counter] >> 5)] ==
1) {
               isbreak = 1;
            }
            else {
               isbreak = 0;
            }

         }
         else {
            duration_counter++;
         }
      }
      else {
         duration_counter = 0;
         score_counter++;
         TA0CCR0 = notes[(curr_song[score_counter] &
NOTE_MASK)];
         isbreak = 0;
      }

   }
   else {
      sys_mod = 3;
      TACCTL0 &= ~OUTMOD_4;
      P1OUT |= LED;
   }
 }
 else if (sys_mod == 1) {

   if (flash_counter == FLASH_INTERVAL) {
      flash_counter = 0;
      P1OUT ^= LED;
   }
   else {
      flash_counter++;
   }
 }
}
void Atraso_us(volatile unsigned int us)
{
  TA1CCR0 = us-1;
  TA1CTL = TASSEL_2 + ID_0 + MC_1 + TAIE;
  while((TA1CTL & TAIFG)==0);
  TA1CTL = TACLR;
  TA1CTL = 0;
}

void Send_Nibble(volatile unsigned char nibble, volatile
unsigned char dados, volatile unsigned int microsegs)
{
  LCD_OUT |= E;
  LCD_OUT &= ~(RS + D4_1 + D5_1 + D6_1 + D7);
  LCD_OUT |= RS*(dados==DADOS) +
     D4_1*((nibble & BIT0)>0) +
     D5_1*((nibble & BIT1)>0) +
     D6_1*((nibble & BIT2)>0) +
     D7*((nibble & BIT3)>0);
  LCD_OUT &= ~E;
  Atraso_us(microsegs);
}

void Send_Byte(volatile unsigned char byte, volatile unsigned
char dados, volatile unsigned int microsegs)
{
  Send_Nibble(byte >> 4, dados, microsegs/2);
  Send_Nibble(byte & 0xF, dados, microsegs/2);
}

void Send_Data(volatile unsigned char byte)
{
  Send_Byte(byte, DADOS, DATA_DLY);
}

void Send_String(char str[])
{
  while((*str)!='\0')
  {
     Send_Data(*(str++));
  }
}

void InitLCD(void)
{
  unsigned char CMNDS[] = {0x20, 0x14, 0xC, 0x6};
  unsigned int i;
  // Atraso de 10ms para o LCD fazer o boot
  Atraso_us(10000);
  LCD_DIR |= D4_1+D5_1+D6_1+D7+RS+E;
  Send_Nibble(0x2, COMANDO, CMND_DLY);
  for(i=0; i<4; i++)
     Send_Byte(CMNDS[i], COMANDO, CMND_DLY);
  CLR_DISPLAY;
  POS0_DISPLAY;
}

//-----------------------------------
ISR_VECTOR(button_handler,".int02")          // P1 interrupt
handler
ISR_VECTOR(WDT_interval_handler,".int10")      // WDT
interrupt handler
```

VIII. APÊNDICE B

```
     .cdecls C,NOLIST,"msp430g2553.h"
     .global   funcao
     .sect ".text"
funcao:   .asmfunc
          mov.b   #02h,&P1SEL
          mov.b   #02h,&P1DIR

  .if    ($defined(__MSP430_HAS_MSP430XV2_CPU__)    |
$defined(__MSP430_HAS_MSP430X_CPU__))
       reta
  .else
       ret
  .endif
        .endasmfunc

       .end
```