

Redução da Bradicinesia em Pacientes com Mal de Parkinson em Estado Inicial

Davi Antônio da Silva Santos
Graduando em Engenharia Eletrônica
Universidade de Brasília
Gama, Brasil
Email: wokep.ma.wavid@gmail.com

Victor Aguiar Coutinho
Graduando em Engenharia Eletrônica
Universidade de Brasília
Gama, Brasil
Email: victor.a.coutinho@gmail.com

Resumo—O uso repetitivo das mãos com auxílio audiovisual tem resultados positivos em pessoas diagnosticadas com Mal de Parkinson em estado inicial. Tem-se como objetivo apresentar um sistema a base de MSP430 fácil de utilizar que execute jogos interativos. Quem utilizá-lo terá redução da bradicinesia, principal sintoma da doença. O software é baseado em linguagem C e Assembly para uma melhor eficiência do dispositivo. O hardware possui quatro botões para os dedos das mãos(exceto para o polegar), dois LEDs e displays para dar informações para o usuário.

Keywords—*Mal de Parkinson; tratamento; exercício das mãos; MSP430.*

I. INTRODUÇÃO

A. Justificativa

O uso de exercícios simples com as mãos, repetição de sequências acompanhados de estímulos audiovisuais pode ser usado em conjunto com os tratamentos tradicionais para auxiliar na reabilitação dos pacientes acometidos do mal de Parkinson em estado inicial, atenuando a bradicinesia, um dos principais sintomas da doença [2], caracterizado pela extrema lentidão dos movimentos [1].

Assim, baseado no equipamento utilizado no artigo de Elisa Pelosin, propõe-se uma solução embarcada de baixo custo baseada no microcontrolador MSP430, utilizando-se a respectiva placa de desenvolvimento, a LaunchPad.

B. Objetivos

Exercício tem um forte impacto na recuperação de pacientes diagnosticados com Mal de Parkinson, porém um dos maiores problemas é a falta de interesses dos pacientes em exercícios físicos [3]. Como Diane Playford explicita em seu artigo Exercise and Parkinson's disease, o desafio não é definir programas de atividades saudáveis, mas incentivar pessoas a encontrarem atividades em que elas achem proveitosas.

Tem-se como objetivo apresentar um projeto de dispositivo que incentive aos pacientes diagnosticados com o Mal de Parkinson em estado inicial a exercitar as mãos como parte de tratamento para não perder o controle dos movimentos.

Com isso, almeja-se colocar jogos interativos de forma a interessar os pacientes a manterem o tratamento. Jogos que serão feitos a base de pesquisa em tratamentos de pacientes de Mal de Parkinson.

C. Requisitos

Com base no público alvo, pacientes diagnosticados com Parkinson em estado inicial, e o objetivo do sistema, melhorar a coordenação motora ao atenuar a bradicinesia, foram elencados os seguintes requisitos:

- O sistema proposto deve ser de fácil construção e baixo custo;
- O sistema deve ser fácil de operar;
- O sistema deve fornecer estímulos audiovisuais;
- A interface deve ser de fácil compreensão;
- Sequências geradas devem ser aleatórias, evitando que o paciente decore-as.

D. Benefícios

O sistema embarcado proposto, conforme exposto anteriormente, reduz a bradicinesia através da repetição de sequências reforçadas por estímulos audiovisuais, em conjunto com os tratamentos tradicionais, em pacientes diagnosticados com Parkinson em estado inicial.

II. DESCRIÇÃO DO SISTEMA

A proposta é apresentar um dispositivo que auxilia na redução da bradicinesia e que, de alguma forma, incentive quem a possui a exercitar as mãos. Para isso, o dispositivo possui um jogo que soa sinais com frequências diferentes e o usuário deva apertar os botões correspondentes. Importante ressaltar que o usuário deve seguir o protocolo de uso para melhores resultados.

Na inicialização do sistema, são emitidos quatro estímulos sonoros, correspondentes às frequências que serão utilizadas durante o jogo. Em seguida, é tocada sequência, a qual inicia em um elemento, e acende-se o LED verde enquanto o número de elementos da sequência não for digitado nos botões, indicando que o sistema está esperando uma entrada do usuário. Ao terminar a sequência, o LED verde desliga.

Caso o usuário acerte a sequência, o LED verde pisca, indicando o acerto, e o nível é incrementado, isto é, a próxima sequência aumentará em um elemento, colocado ao fim da sequência anterior. Também, em caso de acerto da sequência, o sistema esperará uma entrada do usuário indicando se ele quer

continuar o jogo, função atribuída ao botão S1, ou se ele quer desistir do jogo, função do botão S2. Esse momento é expresso ao usuário com ambos os LEDs ligados. Caso o usuário queira continuar, o sistema mostrará a próxima sequência, e caso ele queira desistir, o jogo desligará e exigirá um *reset* para reiniciar.

Caso o usuário erre, o LED vermelho piscará indicando um erro na digitação e, em seguida, os dois LEDs acenderão, esperando que o usuário informe que quer continuar jogando, intenção atribuída ao botão S1, ou que quer desistir do jogo, atribuída a S2. Caso o usuário deseje continuar, o jogo reini-ciaria, mas caso desista, o jogo desligará e será necessário um *reset* para reiniciar.

III. DESCRIÇÃO DO HARDWARE

O projeto é composto pela placa de desenvolvimento Launchpad, a qual contém um microcontrolador MSP430G2553; quatro *pushbuttons* para as entradas do usuário, S1, S2, S3, S4, conectados às entradas P1.3, P1.4, P1.5 e P1.7, respecti-vamente; dois LEDs, sendo um verde, conectado à P2.1(para o código no Energia a porta utilizada havia sido o P1.6 e outro vermelho, ligado à P2.0; um *buzzer* piezoelétrico; um transistor NPN BC337-25, conectado à P2.4; e resistores de 10 kΩ, 15 kΩ, 1 kΩ conectados ao transistor para limitar a corrente fornecida pelas portas do MSP e um resistor de 100Ω para reduzir a intensidade do som do *buzzer*, conforme o esquemático abaixo 1.

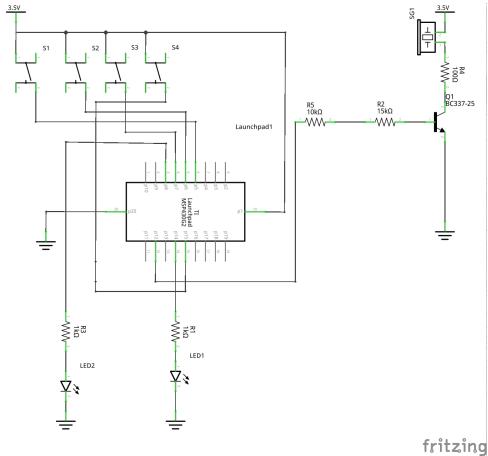


Figura 1. Esquemático do dispositivo proposto

Notou-se que no esquemático original do projeto os LEDs ficavam muito distantes no mapeamento dos pinos, então o LED verde foi movido para o pino perto do LED vermelho, resultando em LED verde em P2.1 e LED vermelho em P2.0, de acordo com o novo esquemático 2. A inserção da matriz de LEDs tornou necessário o remapeamento dos botões, pois os pinos P1.2 e P1.3 precisam ser usados para a comunicação SPI. Como consequência disso, o mapeamento das portas ficou: para a matriz, o SIMO foi ligado ao pino P1.2, o sinal de *clock* era obtida do P1.4 e o pino de LOAD foi conectado ao P1.0; e as chaves S1, S2, S3 e S4 foram conectadas respectivamente a P1.3, P1.1, P1.5 e P1.7. Não houve a necessidade de modificar os LEDs e o *buzzer* conectados nos pinos da porta P2. A lista de materiais encontra-se na tabela I.

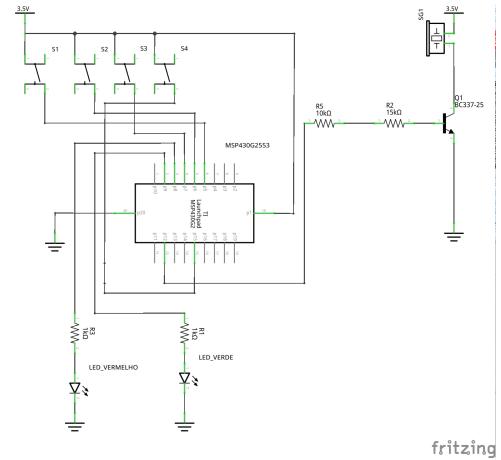


Figura 2. Novo esquemático do dispositivo proposto

Tabela I. LISTA DE MATERIAIS

Quantidade	Componente
1	Buzzer piezoelétrico
-	Jumpers
1	LaunchPad MSP430G2
1	LED verde
1	LED vermelho
1	Módulo matriz de LEDs 8x8 com MAX7219
1	MSP430G2553
1	Protoboard/Breadboard 830 pinos
4	Pushbutton
1	Resistor 10 kΩ
1	Resistor 15 kΩ
2	Resistor 1kΩ
1	Transistor NPN BC337-25

Os quatro *pushbuttons*, estão conectados diretamente ao Vcc = 3,5 V . Utilizou-se esta configuração pois no código foram habilitados os resistores internos de *pull-down* no mi-crocontrolador.

Os resistores colocados sobre os LEDs limitam a corrente entregue pelas portas digitais do MSP, que não deve ser maior que 6 mA em módulo, segundo [6]. Já os resistores colocados antes do transistor limitam a corrente entregue pela porta digital do MSP e da fonte de 3,5 V ao *buzzer*.

Os resistores conectados ao transistor foram dimensionados segundo as especificações do MSP, do *buzzer* e do próprio transistor. Usando um multímetro digital, determinou-se que o *hfe* do transistor era de 308, o que estava dentro dos limites estabelecidos no *datasheet* do mesmo, entre 160 e 400 [4].

O *datasheet* do *buzzer* indicava uma corrente máxima de 40 mA [5], que no transistor corresponderia à corrente no coletor, e este dado foi usado para se calcular a corrente na base do transistor, 129,870 μA. A resistência na base foi dimensionada através da equação $R_b = (V_{cc} - V_{be})/I_b$. Considerou-se $V_{be} = 0,7$ V e $V_{cc} = 3,5$ V, e obteve-se uma resistência de cerca de 21 kΩ, a qual foi substituída por uma de 25 kΩ.

IV. DESCRIÇÃO DO SOFTWARE

A. Software no Energia

A plataforma de prototipagem utilizada para codificar, compilar e enviar para o MSP430 foi o Energia 1.6.10E18. No código primeiramente, definiu variáveis para valores constantes e portas. A *TSOMMS* é uma constante que representa o tempo que o *buzzer* fica ligado ou desligado pois a taxa entre sons que apresenta melhores resultados é de 3 Hz [1]. A *LEDverde* representa o pino em que o LED vermelho está conectado, no caso o P1.6 do MSP430. Já a constante *LEDvermelho* representa o pino P2.0 na qual o LED vermelho está conectado. A *PPWM* é o pino onde o *buzzer* está conectado.

O software foi dividido em funções: *ler_portas*, *toca_buzzer*, *pisca2x*, *setup* e *loop*. A primeira função no código é a *ler_portas*. Essa tem como saída o valor do botão no qual foi apertado, sendo o botão S1 representando o sinal com mais baixa frequência e o S4 o sinal com mais alta frequência. A função começa com um *while(1)* pois ele só deve sair do laço quando algum botão for apertado e os demais não. Considerou os botões configurados como *emphpull down*. Para o usuário saber que que o botão foi apertado o LED verde, como pode ser visto na função principal *loop*, que anteriormente estava ligado desliga. O *delay* de 100 milissegundos para dar tempo do usuário soltar o botão.

A função *toca_buzzer* tem como entrada *nLED* e dentro da função tem o vetor com frequências já definidas. É tocado o *buzzer* na frequência que está na posição *nLED - 1* do vetor *frequencias*. Após tocado por *TSOMMS* ms o *buzzer* desliga com a função *noTone* por *TSOMMS* ms.

Para indicar que uma pessoa acertou ou errou um LED pisca duas vezes, sendo que se o usuário acertou o LED verde que pisca, caso contrário o LED vermelho que pisca. Para isso, desenvolveu a função *pisca2x* tendo como entrada o LED desejado *LED_que_pisca*. Os *delays* de 100 ms foi definido para que o usuário perceba que piscou e não confunda com, no caso do LED verde, o momento para digitar. O *delay* de 1000 ms é para o usuário perceber a parte de piscar indicando acerto ou erro e a parte que liga os LEDs para o usuário decidir se quer continuar ou não, essas duas condições serão explicadas posteriormente.

É definido as portas de entrada e saídas na função *setup*, padrão no Energia. Definiu os pinos P1.3, P1.4, P1.5 e P1.7 como entrada e *pull down* para leitura dos botões S1, S2, S3 e S4, respectivamente. Definiu os pinos P2.4 (representando o *buzzer*), *LEDvermelho* e *emphLEDverde* como saída. Após essas definições colocou todas as saídas em nível lógico alto e gera um número aleatório a partir da leitura da temperatura do ambiente no *A10*. Por fim habilita a entrada e saída serial da placa.

A função principal é *loop*. Começa zerando todas as variáveis, exceto o *nível* já que o primeiro nível considerado foi o zero. Ele apresenta todas as frequências no vetor na ordem dos botões que o representam. Dentro do *while(1)* está o jogo em si. A variável *continuar_jogo* igual a zero significa que o jogo pode continuar, essa condição é avaliada no final do jogo. Considerou o nível máximo de 32, caso o nível seja esse então o jogo finaliza.

A posição do vetor de frequências é gerada por meio da função *random* e colocada na posição *i* da sequência de posições. Logo após é tocado as frequências desde primeira até a atual gerada com a função *toca_buzzer*. É feito a leitura dos botões com a função *ler_portas()* e decide se o usuário acertou a sequência comparando os vetores *seq_gen* (sequência gerada) e *seq_read* (sequência dos botões).

Se a sequência digitada não for igual ao apresentada então a variável *fail* fica igual a um. Com o fail igual a um, o LED vermelho pisca duas vezes indicando o erro e o nível retorna para um. Caso contrário, ou seja acertou, o LED verde pisca duas vez para indicar o acerto o nível aumenta e a variável *i* também aumenta para quando retornar o *while* a nova posição do vetor das frequências ser guardada na próxima posição do vetor *seq_gen*.

Como a decisão de se o usuário acertou ou não, chega na etapa de decidir se o mesmo deseja continuar a jogar. Para isso, ambos os LEDs ficam ligados e é feito a leitura dos botões, essa leitura é armazenada na variável *continuar_jogo*. Se apertou S1 significa que deseja continuar e o S2 para finalizar o jogo. Se for continuar, os LEDs desligam e o LED verde pisca duas vezes indicando o desejo do usuário. Se não deseja continuar, o LED vermelho pisca duas vezes indicando o desejo de não continuar e o programa entra em um laço infinito (a linha do *while(1);*).

B. Portando o Código do Energia para C e Assembly

Para converter o código usado no Energia para a linguagem C teve-se que substituir algumas funções e criar outras. Posteriormente, para a inclusão da matriz de LEDs foi necessária a criação de uma biblioteca para a comunicação com o CI e para a implementação dos caracteres da tabela ASCII extendida. Um exemplo das modificações: no lugar de *digitalRead()* usou *P1IN* ou *P2IN* para a leituras das portas P1 e P2, respectivamente. Uma função criada foi a função *delay*, pois no energia já tinha ela na biblioteca, mas em C houve a necessidade de criar uma utilizando os *timers*.

Uma das maiores dificuldades foi a elaboração de um código em C para o MSP430 que substituisse a função *Tone* e *noTone*. Para elas, usou o canal de comparação 1. Ao fazer no modo comum, ou seja, para ativar o som colocar o TA1CCR2 igual a metade de TA1CCR0 e para desativar colocar TA1CCR0 e TA1CCR1 igual a 0. Portar a parte do código que utilizava as funções do Energia trouxe vários problemas: um sinal com frequência alta no momento em que deveria estar desligado, frequências diferentes das desejadas e algumas distorções no sinal de som.

Tais inconvenientes foram solucionados com a limpeza e reconfiguração dos registradores que controlam o *Timer A1* sempre que uma nova frequência precisa ser tocada. Para resolver esse problema, colocou no início do código da função *toca_buzzer* comandos para ativar o canal de comparação e no final dessa função colocou comandos para desativar o canal de comparação. No *switch* ele altera o período do sinal no *buzzer*. No fim são zerados os registradores TA1CTL, TA1CCR0 e TA1CCR2.

Foi adicionada uma interrupção na parte em que o usuário deve decidir se vai continuar o jogo. O comportamento padrão

do jogo foi alterado para que o usuário não precisasse confirmar se gostaria de continuar jogando a partir do acerto da primeira sequência.

A função que gerava números aleatórios disponibilizada pelo Energia teve de ser substituída por um equivalente em C. Para isso, foram elaboradas duas funções que trabalham para gerar os números aleatórios, a *configura_lcg*, que inicializa a semente do gerador a partir da leitura do sensor de temperatura interno do MSP430G2553 feita através do ADC10, o conversor analógico digital de 10 bits presente no mesmo; e *lcg*, o próprio gerador, que retorna um número aleatório baseado na semente toda vez que é chamada.

Para que o código não ficasse poluído, foi criada uma função responsável por piscar os LEDs duas vezes, *piscar2x*, ação que é repetida diversas vezes durante o código. Esta função apenas utiliza a função *delay* implementada para controlar o tempo de nível alto e nível baixo do LED que for passado para a função.

O código convertido para a linguagem C acrescenta interrupções na porta P1 e modos de baixo consumo para substituir os momentos em que o *polling*, momentos em que a CPU fica ocupada checando repetidamente se houve uma mudança em uma variável, era desnecessário; e os laços infinitos que encerravam o programa caso o nível fosse ultrapassado ou caso o usuário desistisse de continuar jogando.

O processador é desligado, isto é, entra em modo de baixo consumo zero (LPM0), sempre que o programa checa se o usuário gostaria de continuar jogando. O processador é acordado por uma interrupção nos botões responsáveis por confirmar ou negar a opção de continuar o jogo. Já nas situações em que o jogador desiste de continuar o jogo ou o nível máximo (32) é atingido, todos os sinais de *clock* do microcontrolador são desligados, pois este é colocado no modo de baixo consumo 4 (LPM4) e as interrupções são desabilitadas, fazendo com que a única maneira de reiniciar o sistema seja um *reset*.

Para evitar interrupções espúrias, a interrupção é desativada e ativada nos momentos oportunos, isto é, desativada no momento em que o novo número aleatório é gerado e é ativada somente após o sistema validar a entrada do usuário e solicitar se ele gostaria de continuar ou não o jogo, entrando em modo de baixo consumo, na função que substitui o laço principal do jogo, *jogo01*.

C. Bibliotecas utilizadas pela matriz de LEDs

A escrita da biblioteca para comunicação entre o MSP430 e a matriz de LEDs controlada pelo MAX7219 foi outro desafio. Através da leitura do *datasheet* [7] decidiu-se que o modo mais eficiente de implementar a comunicação SPI seria utilizando o módulo USCI do MSP430, pois o código já trabalhava com interrupções e os dois *timers* já estavam sendo usados, tornando difícil a sincronização da comunicação caso esta fosse implementada através do *bit-banging*.

A primeira função a ser criada foi a responsável por inicializar a comunicação SPI. Para isso, é necessário parar o módulo USCI e configurar o registrador UCA0CTL0 para adquirir os dados na borda de subida do *clock* e atualizá-los na borda seguinte, enviar o dígito mais significativo primeiro,

colocar o MSP em modo mestre e configurar a operação em modo SPI. Em seguida configura-se o módulo para usar o sinal de 1 MHz sem divisões, habilita-se os pinos de *clock* e SIMO e liga-se a comunicação.

Em seguida, a próxima função a ser criada foi a que envia os dados para o CI. Ela trabalha enviando o endereço dos 16 registradores do MAX7219 e em seguida os dados que serão escritos neste registrador. Após o envio destas informações, o pino de LOAD é levado a nível alto por pouco tempo, sinalizando para o CI que os dados escritos nos registradores podem ser interpretados.

A outra função básica da biblioteca é a responsável por configurar o CI para que este possa utilizar os LEDs como uma matriz. Esta função utiliza a anterior, para envio de dados, e limpa o registrador de entrada do CI, informa que oito fileiras serão utilizadas, configura uma das 16 intensidades de brilho disponíveis e desabilita a conversão de BCD.

A função para limpeza da matriz foi implementada primeiramente em C, mas verificou-se através da ferramenta de *debug msp430-gdb* que o código gerado pelo compilador utilizado, o msp430-gcc podia ser otimizado, pois ele limpava o registrador R14 diversas vezes antes de chamar a função de envio, sendo que seria necessário que este fosse limpo somente uma vez. Devido a isso, a função foi reescrita em *assembly* para economizar sete instruções.

As demais funções da biblioteca são responsáveis pela escrita de *strings* e caracteres na matriz e pelos efeitos de deslocamento para esquerda, direita, *fade in* e *emphfade out* e utilizam as mesmas três funções básicas em conjunto com uma biblioteca responsável pela implementação da tabela ASCII estendida, sendo esta biblioteca modificada a partir da existente no repositório do *kernel Linux*.

V. RESULTADOS

Os desenvolvedores jogaram por diversas vezes o jogo proposto. Os problemas notados foram que nem sempre era feito a leitura botão pressionado, ao passar o código do Energia para linguagem C o buzzer não funcionava corretamente sendo que a codificação foi seguindo as aulas da disciplina de Microprocessadores e Microcontroladores do curso de Engenharia Eletrônica da Universidade de Brasília. Para o primeiro problema citado foi feito alterações nos *delays* da função de leitura dos botões, isso foi feito na metade do desenvolvimento do dispositivo. Para o segundo, foi feito diversas restrições nos contadores e nos canais de comparação para que o *buzzer* tocasse somente quando fosse pedido, pois antes, mesmo ser chamado a função para tocar o *buzzer* ele ficava soando em alta frequência.

Além dos problemas citados, não houve problemas mais discrepantes. Os resultados obtidos foram satisfatórios e como desejados. Não foi possível testá-lo com algum paciente diagnosticado com Mal de Parkinson em estado inicial para validar a redução da bradicinesia devido ao tempo de entrega do projeto.

O jogo foi validado com um dos desenvolvedores esticavam a mão direita (sendo que esse era uma pessoa sestra para dificultar a jogabilidade). Por diversas vezes tocava um som com uma frequência aleatória dentre as já definidas. O usuário

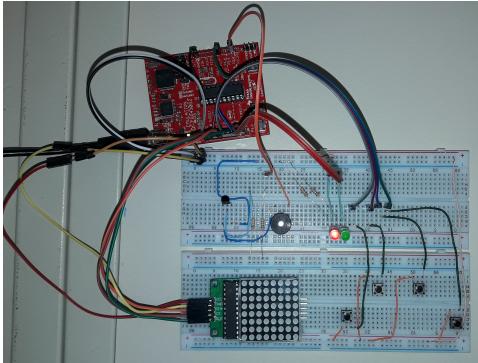


Figura 3. Protótipo montado para apresentação

apertava o botão equivalente a frequência do som tocado sendo que o botão do indicador representa o sinal com menor frequência e do dedo mínimo o de maior frequência.

VI. CONCLUSÃO

Um dos problemas que atrapalham o tratamento dos sintomas em pacientes diagnosticados em pacientes diagnosticados com Mal de Parkinson é a indisposição desses para diversos exercícios [3]. Notou-se que estímulos audiovisuais auxilia na redução da bradicinesia[1], um dos principais sintomas em pacientes diagnosticados com mal de Parkinson em estado inicial. Estímulos com sons a uma taxa de 3 Hz apresentou melhora nos participantes de uma pesquisa publicada na revista *Neurorehabilitation and Neural Repair*.

Montou um dispositivo para tratamento da bradicinesia em pacientes diagnosticados com Mal de Parkinson em estado inicial. Trata-se de um jogo na qual o usuário escuta sequências de sons com diferentes frequências e espaçadas a 3Hz, aperta então os botões equivalentes aos diferentes sons na ordem em que escutou. De acordo com que o usuário acerta, o jogo coloca mais um som na sequência. São quatro sons com diferentes frequências representado cada dedo, o de menor frequência é equivalente ao dedo indicador e o de maior frequência o dedo mínimo.

Os desenvolvedores testaram diversas vezes o jogo. Ele funcionou como o desejado, seguindo os passos colocados no código. Não foi possível testá-lo com pacientes diagnosticados devido ao tempo de entrega do projeto a ser entregue ao orientador da disciplina Microcontroladores e Microprocessadores ministrado pelo Prof. Dr. Diogo Caetano Garcia.

REFERÊNCIAS

- [1] PELOSIN, E. et al., Reduction of Bradykinesia of Finger Movements by a Single Session of Action Observation in Parkinson Disease. *Neurorehabilitation and Neural Repair*. p.552-560, 2013.
- [2] Medtronic Brasil. *Sobre a Doença de Parkinson*. Disponível em: <http://www.medtronicbrasil.com.br/your-health/parkinsons-disease/index.htm>. Acesso em 03 de setembro de 2017.
- [3] PLAYFORD, Diane. Exercise and Parkinson's disease. *Neurol Neurosurg Psychiatry*, 2011;82:1185.
- [4] FAIRCHILD SEMICONDUCTOR CORPORATION. BC337-25 NPN General Purpose Amplifier. 1997.
- [5] KX-1200 Series Magnetic Transducer
- [6] TEXAS INSTRUMENTS. MSP430G2x53, MSP430G2x13 Mixed Signal Microcontroller. p.70, 2011.

- [7] MAXIM INTEGRATED. *MAX7219/MAX7221 Serially Interfaced, 8-digit LED Display Drivers*. p.17, 2003.

APÊNDICE

A. Código Principal

```
*****
* Autores:
* Davi Antônio da Silva Santos
* Victor Aguiar Coutinho
*****
```

```
#include <msp430.h>
#include <legacymsp430.h>
#include "matrix8x8_MAX7219.h"

/*
AVISO:
Para o código funcionar, _retirar_ os jumpers RXD e TXD da LaunchPad.
-----
```

T	R	R	T	V
X	X	S	E	C
D	D	T	S	C
T				

```

A falha em executar essas ações resultará em um código preso na leitura da
porta P1.1 devido ao protocolo utilizado para debug.
*/
/* ***** Para a biblioteca da matriz ***** */
/* Configurar os pinos aos quais o CI será conectado */
/* Para o MSP430G2553:
* SPLSIMO = UCA0SIMO = P1.2      (PISEL |= BIT2, PISEL2 |= BIT2)
* SPI_CLK = UCA0CLK = P1.4        (PISEL |= BIT4, PISEL2 |= BIT4)
* P1.0 é o LOAD da matriz, é uma entrada e saída comum */
/* ***** Para o código principal ***** */
/* O buzzer está em P2.4 */
#define PINO_PWM BIT4
#define TEMPO_SOM_MS 333
/* LED verde está em P2.1 */
#define LED_Verde BIT1
/* LED vermelho está em P2.0 */
#define LED_Vermelho BIT0

/* Definindo os botões P1.3, P1.1, P1.5 e P1.7 */
#define BOTAO_01 BIT3
#define BOTAO_02 BIT1
#define BOTAO_03 BIT5
#define BOTAO_04 BIT7
#define BOTOES (BOTAO_01 | BOTAO_02 | BOTAO_03 | BOTAO_04)
#define BOTOES_INTRUPCAO (BOTAO_01 | BOTAO_02)

/* Definição das constantes de frequência */
/* Frequência do botão 01 é 1,04652 kHz, contar 955 us */
/* Frequência do botão 02 é 1,31852 kHz, contar 758 us */
/* Frequência do botão 03 é 1,56800 kHz, contar 638 us */
/* Frequência do botão 04 é 2,09300 kHz, contar 478 us */
#define FREQUENCIA_01_US 955
#define FREQUENCIA_02_US 758
#define FREQUENCIA_03_US 638
#define FREQUENCIA_04_US 478

/* ***** Protótipos das Funções ***** */
unsigned int configura_lcg(void);
void delay(volatile unsigned int x);
unsigned long int lcg(volatile long int x0, volatile unsigned int modo);
int ler_portas(void);
void piscar2x(char LED_que_piscar);
void toca_buzzer(volatile unsigned int nLED);
void jogo01(void);

/*
Variáveis Globais
***** */
/* Frases utilizadas */
unsigned char começo_jogo[] = {'V', 'a', 'm', 'o', 's', 'u', 'c', 'o', 'm', 'e', 'r', 'i', 'l', '0'};
unsigned char resp_errada[] = "X";
unsigned char resp_certa[] = "O";
unsigned char ordem_apertar[] = "_";
unsigned char desligando[] = "Adeus";
```

```

unsigned char nivel_max[] = "Ganhou !!!";
unsigned char continuando[] = ">>";

/* Variável de controle para determinar se o jogo irá continuar ou não */
unsigned char continuar_jogo = 0;

/********************* Início do Código *********************/
***** Início do Código *****
int main(void)
{
    unsigned int cont = 0;

    /* Parar o watchdog timer */
    WDTCTL = WDTPW + WDTHOLD;

    /* Configurar o pino do PWM como saída da comparação do Timer A1
     * no canal de comparação 2 */
    P2DIR |= PIN0_PWM;
    P2SEL |= PIN0_PWM;
    P2SEL &= ~PIN0_PWM;

    /* Configurar os LEDs de saída */
    P2DIR |= LED_Verde;
    P2DIR |= LED_Vermelho;
    P2OUT &= ~(LED_Verde + LED_Vermelho);

    /* Configurar botões como entrada e pull-down*/
    P1DIR &= ~BOToes;
    P1REN |= BOToes;
    P1OUT &= ~BOToes;

    /* Congigurar interrupção nos pinos */
    /* Borda de subida pois é pull-down */
    P1IES &= ~BOToes_INTRUPCAO;
    P1IE |= BOToes_INTRUPCAO;
    _BIS_SR(GIE);

    /* Configurar os pinos, configurar e ligar a matriz */
    liga_1a vez();
    INTENSIDADE_MAXIMA;

    /* Mostrar as instruções */
    mostra_string_dir_esq_v2(comeco_jogo);
    mostra_char_dir_esq('!');

    /*Apresenta as frequencias de sons utilizadas*/
    for (cont = 1; cont <= 4; cont++)
        toca_buzzer(cont);

    /* Colocar o LCG em modo de configuração */
    lcg(0, 0);

    /* loop que correrá enquanto o sistema estiver ligado */
    jogo01();
    mostra_char('?' );
    _BIS_SR(LPM0_bits);

    return 0;
}

***** Funções *****
/* Gerador de números aleatórios (LCG) se modo = 0, todo xn sairá da
 * conversão do ADC10. Assim que modo != 0 todo xn sairá do LCG */

void jogo01(void)
{
    static int seq_gen[32] = {0};
    static int seq_read[32] = {0};
    static unsigned char level = 1;

    static unsigned int i = 0, j = 0;
    static char fail = 0;

    /* Desativar interrupção nos pinos */
    P1IE &= ~BOToes_INTRUPCAO;

    /* Checar se nível máximo */
    if (level >= 32){
        mostra_string_fade_inout(nivel_max, 0x0F);
        _BIS_SR(LPM4_bits);
    }

    delay(500);

    /* Gerar o número aleatório*/
    seq_gen[i] = (lcg(0, 1) % 4) + 1;

    mostra_char_dir_esq('!' );

    /* Mostrar a sequência */
    for(j = 0; j < level; j++)
        toca_buzzer(seq_gen[j]);

    /* Ler os botões */
    delay(10);
    mostra_string_dir_esq_v2(ordem_apertar);

    for(j = 0; j < level; j++){
        delay(100);
        seq_read[j] = ler_portas();
    }

    /* Comparar as sequências */
    for(j = 0; j < level; j++){
        if(seq_gen[j] != seq_read[j])
            fail = 1;
    }
}

}

/* Checar se passou */
if(fail != 0){
    pista2x(LED_Vermelho);
    level = 1;
    fail = 0;

    mostra_string_fade_inout(resp_errada, 0x0F);
    continuar_jogo = 0;
} else{
    pista2x(LED_Verde);
    level++;
    i++;
    mostra_string_fade_inout(resp_certa, 0x0F);
    continuar_jogo = 1;
}
delay(500);

/* Deseja continuar?*/
P2OUT |= LED_Vermelho;
P2OUT |= LED_Verde;

/* Ativar interrupção nos pinos */
P1IE |= BOToes_INTRUPCAO;
}

unsigned long int lcg(volatile unsigned long int x0, volatile unsigned int modo)
{
    static long unsigned int xn;
    long unsigned int j;
    unsigned char i;

    if(modo){
        for(i = 0; i < (configura_lcg() % 16); i++)
            xn = (((1103515245*xn + 12345) & 0xFFFFFFFF));
    }

    j = (xn >> 28);
    xn ^= xn >> (4 + j);
    xn *= 277803737u;
    xn = xn ^ (xn >> 22);
    xn &= 0x0000FFFF;
} else{
    xn = configura_lcg();
}
return xn;
}

unsigned int configura_lcg(void)
{
    /* Usar Vcc e Vss como referência;
     * Usar como tempo de sample e hold 4 clocks;
     * Habilitar o ADC10 */
    ADC10CTL0 = SREF_0 + ADC10SHT_0 + ADC10ON;

    /* Selecionar o sensor de temperatura interno como entrada;
     * A conversão será requisitada quando receber ENC e ADC10SC;
     * O clock não será dividido;
     * O clock utilizado é o SMCLK;
     * Apens um canal de conversão AD será usado */
    ADC10CTL0 = INCH_10 + SHS_0 + ADC10DIV_0 + ADC10SEL_3 + CONSEQ_0;

    /* Começar uma conversao */
    ADC10CTL0 |= ENC + ADC10SC;

    /* Espera a conversao ficar pronta */
    while((ADC10CTL0 & ADC10FG) == 0);

    return ADC10MEM;
}

/*FUNÇÃO PARA LER BOTÕES*/
int ler_portas(void)
{
    mostra_char('?' );

    while(1){
        /* Aviso visual de modo leitura */
        P2OUT |= LED_Verde;

        int v1 = PIIN & BOTAO_01;
        int v2 = PIIN & BOTAO_02;
        int v3 = PIIN & BOTAO_03;
        int v4 = PIIN & BOTAO_04;

        delay(10);

        /* Considerando PULL DOWN => botão ligado no Vcc */
        if(v1 && !v2 && !v3 && !(v4)){
            P2OUT &= ~LED_Verde;
            while(PIIN & BOTAO_01);
            return 1;
        }
        else if(!v1 && v2 && !(v3) && !(v4)){
            P2OUT |= ~LED_Verde;
            while(PIIN & BOTAO_02);
            return 2;
        }
        else if(!(v1) && v2 && v3 && !(v4)){
            P2OUT |= ~LED_Verde;
            while(PIIN & BOTAO_03);
            return 3;
        }
        else if(!(v1) && !(v2) && !(v3) && v4){
            P2OUT &= ~LED_Verde;
            while(PIIN & BOTAO_04);
            return 4;
        }
        delay(10);
    }
}

```

```

}

/*FUNÇÃO PARA PISCAR LEDS DUAS VEZES*/
void pisca2x( char LED_que_pisca){
P2OUT &= ~LED_que_pisca;
delay(100);
P2OUT |= LED_que_pisca;
delay(100);
P2OUT &= ~LED_que_pisca;
delay(100);
P2OUT |= LED_que_pisca;
delay(100);
P2OUT &= ~LED_que_pisca;
delay(1000);
}

void delay(volatile unsigned int x)
{
int i;

for(i = 0; i < 1000; i++){
/* 1000 contagens de 1 us = 1 ms, mas conta o zero (up) */
TA0CCR0 = x - 1;

/* Usar SMCLK, não dividi-lo e contar em modo up */
TA0CTL = TASSEL_2 + ID_0 + MC_1;

/* Espera terminar contagem */
while((TA0CTL & TAIFG) == 0);

/* Parar e zerar o timer */
TA0CTL = MC_0 + TACLR;
TA0CTL &= ~TAIFG;
}

/* Configurar timer do PWM */
void toca_buzzer(volatile unsigned int nLED)
{
/* Configurar o pino do PWM como saída da comparação do Timer A1
* no canal de comparação 2*/
/* Frequência do botão 01 é 1,04652 kHz, contar 955 us */
/* Frequência do botão 02 é 1,31852 kHz, contar 758 us */
/* Frequência do botão 03 é 1,56800 kHz, contar 638 us */
/* Frequência do botão 04 é 2,09300 kHz, contar 478 us */

/* Reiniciar todos os registradores que serão usados na
* comparação e parar o timer */
TA1CTL = 0;
TA1CCR0 = 0;
TA1CCR2 = 0;

/* Configurar o modo de comparação do pino antes de cada
* definição de frequência */
P2SEL |= PINO_PWM;
P2SEL2 &= ~PINO_PWM;

switch(nLED){
case 1:
TA1CCR0 |= FREQUENCIA_01_US - 1;
TA1CCR2 |= ((FREQUENCIA_01_US - 1) >> 1);
/* Colocar no modo de contagem up, para facilitar o
* trabalho com PWM, pois é só subir o valor do reg de
* comparação */
TA1CCTL2 = OUTMOD_7;
TA1CTL = TASSEL_2 + ID_0 + MC_1;
break;

case 2:
TA1CCR0 |= FREQUENCIA_02_US - 1;
TA1CCR2 |= ((FREQUENCIA_02_US - 1) >> 1);
/* Colocar no modo de contagem up, para facilitar o
* trabalho com PWM, pois é só subir o valor do reg de
* comparação */
TA1CCTL2 = OUTMOD_7;
TA1CTL = TASSEL_2 + ID_0 + MC_1;
break;

case 3:
TA1CCR0 |= FREQUENCIA_03_US - 1;
TA1CCR2 |= ((FREQUENCIA_03_US - 1) >> 1);
/* Colocar no modo de contagem up, para facilitar o
* trabalho com PWM, pois é só subir o valor do reg de
* comparação */
TA1CCTL2 = OUTMOD_7;
TA1CTL = TASSEL_2 + ID_0 + MC_1;
break;

case 4:
TA1CCR0 |= FREQUENCIA_04_US - 1;
TA1CCR2 |= ((FREQUENCIA_04_US - 1) >> 1);
/* Colocar no modo de contagem up, para facilitar o
* trabalho com PWM, pois é só subir o valor do reg de
* comparação */
TA1CCTL2 = OUTMOD_7;
TA1CTL = TASSEL_2 + ID_0 + MC_1;
break;

default:
TA1CTL = 0;
}

/* Pequeno delay para estabilizar */
delay(TEMPO_SOM_MS);

/* Parar o timer */
TA1CTL = 0;

/* Configurar como saída e colocar em nível baixo para
* desligar o buzzer */
P2SEL &= ~PINO_PWM;
P2SEL2 &= ~PINO_PWM;
P2OUT &= ~PINO_PWM;
P2OUT2 &= ~PINO_PWM;

/* Limpar os registradores da comparação*/
TA1CCR0 = 0;
TA1CCR2 = 0;

delay(TEMPO_SOM_MS);
}

/********************* Rotinas de Interrupção
* A interrupção faz o debounce e religa o MSP */
interrupt(PORT1_VECTOR) Interrupcao_botoes(void)
{
PIIFG = 0;

/* Desativar interrupção nos pinos */
P1IE &= ~BOTÕES_INTEERRUPCAO;

/*char continuar_jogo = 0;*/

/* Zerar o vetor de interrupções para evitar que a interrupção
* seja chamada continuamente */
if(continuar_jogo == 0)
continuar_jogo = ler_portas();

if(continuar_jogo == 1){
P2OUT &= ~LED_VERMELHO;
P2OUT &= ~LED_VERDE;
delay(100);
P2OUT |= LED_Verde;
delay(100);
P2OUT &= ~LED_Verde;
delay(100);
}

mostra_string_dir_esq_v2(continuando);

jogo01();
}else if(continuar_jogo == 2){
P2OUT &= ~LED_Verde;
P2OUT &= ~LED_Vermelho;
delay(100);
P2OUT |= LED_Vermelho;
delay(100);
P2OUT &= ~LED_Vermelho;
delay(100);

/* Não quis continuar, desligar */
mostra_string_fade_inout(desligando, 0x0F);
_BIS_SR(LPM4_bits);
}else{
continuar_jogo = 0;
}
}

B. Biblioteca da Matriz 8x8 para o CI

/*
Controlar uma matriz de LEDs 8x8 em um MAX7219, usando a USCI/USI de um
MSP430G2553 em modo SPI, sendo que o SPI está em modo 'master'.

Antes de iniciar a SPI, colocar um delay para o CI acompanhar.

* SPI_SIMO = UCA0SIMO = PI.2      (PISEL |= BIT2, PISEL2 |= BIT2)
* SPI_CLK = UCA0CLK = PI.4        (PISEL |= BIT4, PISEL2 |= BIT4)

* Procedimento para ligar a matriz:
* Configurar clock do sistema em 1 MHz
DCOCTL = CALDCO_IMHZ;
BCSCTL1 = CALBC1_IMHZ;

* Configurar as portas usadas pela matriz como saída por segurança,
pois SPI_SIMO e SPI_CLK já tem a direção escolhida pelas configurações da
USCI.
PIDIR |= SPI_SIMO + SPI_CLK + PINO_LOAD;

* Para o MAX7219, os dados entram nas bordas de subida do clock,
independente do estado do LOAD.
PIO1OUT &= ~(PINO_LOAD);

* Inicializar a comunicação SPI através da USCI. Sempre colocar um delay
antes
e depois para evitar problemas ao conectar o MSP na alimentação e para dar
tempo
do MAX7219 entender que a comunicação iniciou.
__delay_cycles(1000);
inicializar_SPI();
__delay_cycles(1000);

* Definir a intensidade que quer iniciar a matriz e limpá-la.
configura_matriz(0x00);
limpa_matriz();

* A macro LIGA_MATRIZ torna a matriz disponível para uso.

#include <msp430.h>
#include "graficos_matriz_8x8.h"

* Configurar os pinos aos quais o CI será conectado */
* Para o MSP430G2553:
* SPI_SIMO = UCA0SIMO = PI.2      (PISEL |= BIT2, PISEL2 |= BIT2)
* SPI_CLK = UCA0CLK = PI.4        (PISEL |= BIT4, PISEL2 |= BIT4) */
#define PISEL_SIMO           BIT2
#define SPI_CLK               BIT4

```

```

/* Este é um pino comum, que pode ser conectado a qualquer porta em modo IO.
 */
#define PINO_LOAD      BIT0 /* P1.0 */

/* Configurar os endereços dos registradores do CI utilizado, tabela 2 do
datasheet*/
#define MAX_NOOP        0x00
#define MAX_DIGITO      0x01
#define MAX_DIGIT1       0x02
#define MAX_DIGIT2       0x03
#define MAX_DIGIT3       0x04
#define MAX_DIGIT4       0x05
#define MAX_DIGIT5       0x06
#define MAX_DIGIT6       0x07
#define MAX_DIGIT7       0x08
#define MAX_DECODEMODE   0x09
#define MAX_INTENSITY     0x0A
#define MAX_SCANLIMIT     0x0B
#define MAX_SHUTDOWN      0x0C
#define MAX_DISPLAYTEST    0x0F

/* Outras constantes */
#define DELAY_STRING_DIR_ESQ 30000
#define DELAY_FADE_IN      20000
#define DELAY_FADE_OUT      20000
#define DELAY_STRING        100000
#define DELAY_CHAR_ESQ      30000
#define DELAY_CHAR_DIR      30000
#define DELAY_STRING_ESQ    500000

/* Outras macros */
#define LIGA_MATRIZ envia_dados_max_SPI(MAX_SHUTDOWN, 1);
#define DESLIGA_MATRIZ envia_dados_max_SPI(MAX_SHUTDOWN, 0);
#define INTENSIDADE_MAXIMA envia_dados_max_SPI(MAX_INTENSITY, 0x0F);
#define INTENSIDADE_MINIMA envia_dados_max_SPI(MAX_INTENSITY, 0x00);

/*
***** Protótipos das funções *****
***** Funções básicas *****
void inicializar_SPI(void);
void envia_dados_max_SPI(unsigned char endereco, unsigned char dados);
void configura_matriz(unsigned char intensidade);
void limpa_matriz(void);
void mostra_char(volatile unsigned char simbolo);
void mostra_string(unsigned char * string);

/* Efeitos especiais */
void mostra_char_esq(unsigned char simbolo);
void mostra_char_dir(unsigned char simbolo);
void mostra_char_dir_esq(unsigned char simbolo);
void mostra_string_esq(unsigned char * string);
void mostra_string_dir_esq(unsigned char * string);
void mostra_string_dir_esq_v2(unsigned char * string);
void mostra_char_fade_in(unsigned char simbolo, unsigned int intensidade_max );
void limpa_matriz_fade_out(unsigned char intensidade_atual);
void mostra_string_fade_inout(unsigned char * string, unsigned char
intensidade);

/* Ligando pela primeira vez, essa função será suficiente */
void liga_1a vez(void);

/*
***** Funções *****
***** Os 16 bits serão enviados para o MAX7219 no formato:
* XXXX AAAA DDDD DDDD ('don't care', endereço, dados).
* Em seguida, o MAX7219 irá executar o comando quando houver borda de subida
do
pino de LOAD. */
void envia_dados_max_SPI(unsigned char endereco, unsigned char dados)
{
/* Enviar os 4 bits do endereço como um byte e esperar terminar o
envio. */
UCA0TxBUF = endereco & 0x0F;
while ((UCA0STAT & UCBUSY);

/* Enviar os dados e esperar terminar o envio */
UCA0TxBUF = dados;
while ((UCA0STAT & UCBUSY);

/* O MAX7219 guarda os dados recebidos na borda de subida do pino LOAD.
Deixa-lo em nível baixo para a próxima chamada ocorrer corretamente. */
PIO1OUT |= PINO_LOAD;
PIO1OUT &= ~(PINO_LOAD);

/* Habilitar a comunicação SPI */
void inicializar_SPI()
{
/* Parar a comunicação colocando a UCSI em modo reset, para que possa
ser configurada. */
UCA0CTL1 |= UCSWRST;

/* Para comunicar conforme descrito no datasheet do MAX7219:
* Os dados são adquiridos na primeira borda de clock do sinal utilizado
pelo módulo e são atualizados na borda seguinte;
* O MSB é enviado primeiro;
* O MSP é o mestre (gera o clock), e o MAX7219, o escravo;
* Operação do módulo em modo síncrono, comunicação por SPI.*/
UCA0CTL0 = UCKPH + UCMSB + UCMSL + UCSYNC;

/* O clock utilizado pelo módulo será o SMCLK */
UCA0CTL1 |= UCSSEL_2;

/* A comunicação ocorrerá com o mesmo clock de SMCLK. Divisão poderia
ser até 2^16. */
UCA0BRO |= 0x01;
UCA0BRI = 0;

/* Configurar os pinos SIMO (saída de dados do MSP) e CLK. */
PISEL1 |= SPI_SIMO + SPI_CLK;
PISEL2 |= SPI_SIMO + SPI_CLK;

/* O módulo pode comunicar-se, sair do estado reset. */
UCA0CTL1 &= ~UCSWRST;
}

void configura_matriz(unsigned char intensidade)
{
/* if(intensidade > 0x0F) intensidade = 0x0F; */

/* O CI deve ser configurado para trabalhar com uma matriz de 8x8:
* limpava o registrador de deslocamento, ordenado que não se execute
nenhumha operação;
* Habilitar todas as 8 colunas (dígitos caso display de 7 segmentos como
no datasheet);
* Configurar a intensidade do display, 16 disponíveis (0x00a 0x0F).
* Desativar a decodificação de BCD, pois trabalharemos com uma matriz
e enviaremos em binário os LEDs de cada coluna que devem ser acesos.
*/
envia_dados_max_SPI(MAX_NOOP, 0x00);
envia_dados_max_SPI(MAX_NOOP, 0x00);

envia_dados_max_SPI(MAX_SCANLIMIT, 0x07);
envia_dados_max_SPI(MAX_INTENSITY, intensidade);
envia_dados_max_SPI(MAX_DECODEMODE, 0);
}

void limpa_matriz(void)
{
/*envia_dados_max_SPI(MAX_DIGITO, 0);
envia_dados_max_SPI(MAX_DIGIT1, 0);
envia_dados_max_SPI(MAX_DIGIT2, 0);
envia_dados_max_SPI(MAX_DIGIT3, 0);
envia_dados_max_SPI(MAX_DIGIT4, 0);
envia_dados_max_SPI(MAX_DIGIT5, 0);
envia_dados_max_SPI(MAX_DIGIT6, 0);
envia_dados_max_SPI(MAX_DIGIT7, 0);*/

asm(
"clr.w R14\n"
"mov.b #1,_R15\n"
"call ___.envia_dados_max_SPI_\n"
"mov.b #2,_R15\n"
"call ___.envia_dados_max_SPI_\n"
"mov.b #3,_R15\n"
"call ___.envia_dados_max_SPI_\n"
"mov.b #4,_R15\n"
"call ___.envia_dados_max_SPI_\n"
"mov.b #5,_R15\n"
"call ___.envia_dados_max_SPI_\n"
"mov.b #6,_R15\n"
"call ___.envia_dados_max_SPI_\n"
"mov.b #7,_R15\n"
"call ___.envia_dados_max_SPI_\n"
"mov.b #8,_R15\n"
"call ___.envia_dados_max_SPI_\n"
"pop.w R4\n"
"ret"
);
}

void mostra_char(volatile unsigned char simbolo)
{
unsigned int i;

for(i = 0; i < 8; i++)
envia_dados_max_SPI(MAX_DIGITO + i, fonte_8x8[(unsigned int) \
(simbolo*8) + i]);

void mostra_string(unsigned char * string)
{
unsigned int i = 0;
while(string[i] != '\0'){
mostra_char(string[i]);
i++;
__delay_cycles(DELAY_STRING);
}

/* Efeitos especiais */
void mostra_char_esq(unsigned char simbolo)
{
unsigned int i, j;

for(j = 0; j <= 8; j++){
for(i = 0; i < 8; i++)
envia_dados_max_SPI(MAX_DIGITO + i, fonte_8x8[(unsigned
int)(simbolo*8) + i] << j);
__delay_cycles(DELAY_CHAR_ESQ);
}

void mostra_char_dir(unsigned char simbolo)
{
unsigned int i, j;

for(j = 0; j <= 8; j++){
for(i = 0; i < 8; i++)
envia_dados_max_SPI(MAX_DIGITO + i, fonte_8x8[(unsigned
int)(simbolo*8) + i] >> (8-j));
__delay_cycles(DELAY_CHAR_DIR);
}
}

```

```

}

void mostra_char_dir_esq(unsigned char simbolo)
{
    mostra_char_dir(simbolo);
    mostra_char_esq(simbolo);
}

void mostra_string_dir_esq(unsigned char * string)
{
    unsigned int i=0, j, k;

/*Enquanto não achar a última posição*/
while(string[i] != '\0'){
for(j = 0; j <= 8; j++){
/* Para o primeiro caractere, fazer apenas aparecer da direita para a esquerda*/
if(i == 0){
for(k = 0; k < 8; k++){
envia_dados_max_SPI(MAX_DIGITO + k,
fonte_8x8[(unsigned int)(string[i]*8)
+k] >> (8-j));
}
/* Para as demais posições, será enviado o caractere concatenado com o próximo, de modo que quando o lo esteja no meio da tela o 2º comece a aparecer na mesma proporção.*/
}else if (i > 0) {
for(k = 0; k < 8; k++){
envia_dados_max_SPI(MAX_DIGITO + k,
(fonte_8x8[(unsigned int)(string[i-1]*8)
+k] << (j+1)) | (fonte_8x8[(unsigned
int)(string[i]*8) + k] >> (8 - j)));
}
}
__delay_cycles(50000);
}
i++;
}
mostra_char_esq(string[i-1]);
}

void mostra_string_dir_esq_v2(unsigned char * string)
{
    unsigned int i=0, j, k;

/*Enquanto não achar a última posição*/
while(string[i] != '\0'){
/* Para o primeiro caractere, fazer apenas aparecer da direita para a esquerda*/
if(i == 0){
mostra_char_dir(string[0]);
}else if(i > 0){
for(j = 0; j < 8; j++){
/* Para as demais posições, será enviado o caractere concatenado com o próximo, de modo que quando o lo esteja no meio da tela o 2º comece a aparecer na mesma proporção.*/
for(k = 0; k < 8; k++){
envia_dados_max_SPI(MAX_DIGITO
+k, (fonte_8x8[(unsigned int)
(string[i-1]*8) + k] << (j+1)) |
(fonte_8x8[(unsigned int)
(string[i]*8) + k] >> (8 - j)));
}
}
__delay_cycles(DELAY_STRING_DIR_ESQ);
}
}
i++;
}
mostra_char_esq(string[i-1]);
}

void mostra_string_esq(unsigned char * string)
{
    unsigned int i = 0;

while(string[i] != '\0'){
mostra_char_esq(string[i]);
i++;
__delay_cycles(DELAY_STRING_ESQ);
}
}

void mostra_char_fade_in(unsigned char simbolo, unsigned int intensidade_max )
{
    unsigned char i;

limpa_matriz();
envia_dados_max_SPI(MAX_INTENSITY, 0x00);
mostra_char(simbolo);

for(i = 0x00; i < intensidade_max; i++){
envia_dados_max_SPI(MAX_INTENSITY, i);
__delay_cycles(DELAY_FADE_IN);
}

void limpa_matriz_fade_out(unsigned char intensidade_atual )
{
while((intensidade_atual >= 0x00) && (intensidade_atual <= 0x0F)){
envia_dados_max_SPI(MAX_INTENSITY, intensidade_atual);
intensidade_atual--;
__delay_cycles(DELAY_FADE_OUT);
}

limpa_matriz();
}

void mostra_string_fade_inout(unsigned char * string , unsigned char
intensidade)
{
    unsigned int i = 0;

while(string[i] != '\0'){
mostra_char_fade_in(string[i], intensidade);
limpa_matriz_fade_out(intensidade);
i++;
}

limpa_matriz();
envia_dados_max_SPI(MAX_INTENSITY, intensidade);
}

void liga_lig_a vez(void)
{
/* Configurar as portas usadas pela matriz como saída por segurança,
pois SPI_SIMO e SPI_CLK já tem a direção escolhida pelas configurações
da USCI. */
PDIR |= SPI_SIMO + SPI_CLK + PINO_LOAD;

/* Para o MAX7219, os dados entram nas bordas de subida do clock,
independente do estado do LOAD*/
PIO |= ~PINO_LOAD;

/* Inicializar a comunicação SPI através da USCI */
__delay_cycles(1000);
inicializar_SPI();
__delay_cycles(1000);

configura_matriz(0x00);
limpa_matriz();

/* Pode ligar o display */
LIGA_MATRIZ;
}
}

C. Biblioteca com os gráficos da Matriz

/*
Inspirada na fonte 8x8 do kernel Linux
https://raw.githubusercontent.com/torvalds/linux/master/lib/fonts/font_8x8.c
*/
#define TAM_FONTE 2048
/* São 256 caracteres*8 linhas*/
const unsigned char fonte_8x8[TAM_FONTE] = {

/* 0x00 '^@' */
0x00, /* 00000000 */

/* 1 0x01 '^A' */
0x7e, /* 01111110 */
0x81, /* 10000001 */
0xa5, /* 10100101 */
0x81, /* 10000001 */
0xbd, /* 10111101 */
0x99, /* 10011001 */
0x81, /* 10000001 */
0x7e, /* 01111110 */

/* 2 0x02 '^B' */
0x7e, /* 01111110 */
0xff, /* 11111111 */
0xdb, /* 11010101 */
0xff, /* 11111111 */
0xc3, /* 11000011 */
0x7e, /* 11000011 */
0xff, /* 11111111 */
0x7e, /* 01111110 */

/* 3 0x03 '^C' */
0x6c, /* 01101100 */
0xfe, /* 11111110 */
0xfe, /* 11111110 */
0xfe, /* 11111110 */
0x7e, /* 01111100 */
0x38, /* 00111000 */
0x10, /* 00010000 */
0x00, /* 00000000 */

/* 4 0x04 '^D' */
0x10, /* 00010000 */
0x38, /* 00111000 */
0x7c, /* 01111100 */
0xfe, /* 11111110 */
0x7c, /* 01111100 */
0x38, /* 00111000 */
0x10, /* 00010000 */
0x00, /* 00000000 */

/* 5 0x05 '^E' */
0x38, /* 00111000 */
0x7c, /* 01111100 */
0x38, /* 00111000 */
0xfe, /* 11111110 */
0xfe, /* 11111110 */
0xd6, /* 11010110 */
0x10, /* 00010000 */
0x38, /* 00111000 */
}

```

```

/* 6 0x06 '^F' */
0x10, /* 00010000 */
0x38, /* 00111000 */
0x7c, /* 01111100 */
0xfe, /* 11111110 */
0xfe, /* 11111110 */
0x7c, /* 01111100 */
0x10, /* 00010000 */
0x38, /* 00111000 */

/* 7 0x07 '^G' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x18, /* 00011000 */
0x3c, /* 00111100 */
0x3c, /* 00111100 */
0x18, /* 00011000 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 8 0x08 '^H' */
0xff, /* 11111111 */
0xff, /* 11111111 */
0xc7, /* 11000111 */
0xc3, /* 11000011 */
0xc3, /* 11000011 */
0xc7, /* 11000111 */
0xff, /* 11111111 */
0xff, /* 11111111 */

/* 9 0x09 '^I' */
0x00, /* 00000000 */
0x3c, /* 00111100 */
0x66, /* 01000110 */
0x42, /* 01000010 */
0x42, /* 01000010 */
0x66, /* 01000110 */
0x3c, /* 00111100 */
0x00, /* 00000000 */

/* 10 0x0a '^J' */
0xff, /* 11111111 */
0xc3, /* 11000011 */
0x99, /* 10011001 */
0xbd, /* 10111101 */
0xbd, /* 10111101 */
0x99, /* 10011001 */
0xc3, /* 11000011 */
0xff, /* 11111111 */

/* 11 0x0b '^K' */
0x0f, /* 00001111 */
0x07, /* 00000111 */
0x0f, /* 00001111 */
0x7d, /* 01111101 */
0xcc, /* 11001100 */
0xcc, /* 11001100 */
0xcc, /* 11001100 */
0x78, /* 01111000 */

/* 12 0x0c '^L' */
0x3c, /* 00111100 */
0x66, /* 01000110 */
0x66, /* 01000110 */
0x66, /* 01000110 */
0x3c, /* 00111100 */
0x18, /* 00011000 */
0x7e, /* 01111110 */
0x18, /* 00011000 */

/* 13 0x0d '^M' */
0x3f, /* 00111111 */
0x33, /* 00110011 */
0x3f, /* 00111111 */
0x30, /* 00110000 */
0x30, /* 00110000 */
0x70, /* 01110000 */
0xf0, /* 11100000 */
0xe0, /* 11100000 */

/* 14 0x0e '^N' */
0x7f, /* 01111111 */
0x63, /* 01100011 */
0x7f, /* 01111111 */
0x63, /* 01100011 */
0x63, /* 01100011 */
0x67, /* 01100111 */
0xe6, /* 11100110 */
0xc0, /* 11000000 */

/* 15 0x0f '^O' */
0x18, /* 00011000 */
0xdb, /* 11011011 */
0x3c, /* 00111100 */
0xc7, /* 11100111 */
0xc7, /* 11100111 */
0x3c, /* 00111100 */
0xdb, /* 11011011 */
0x18, /* 00011000 */

/* 16 0x10 '^P' */
0x80, /* 10000000 */
0xe0, /* 11100000 */
0xf8, /* 11111000 */
0xfe, /* 11111100 */
0xf8, /* 11111000 */
0xe0, /* 11100000 */
0x80, /* 10000000 */
0x00, /* 00000000 */

/* 17 0x11 '^Q' */
0x02, /* 00000010 */

```

```

0xc0, /* 11000000 */
0xc0, /* 11000000 */
0xfe, /* 11111110 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 29 0x1d '^' */
0x00, /* 00000000 */
0x24, /* 0100100 */
0x66, /* 01100110 */
0xff, /* 11111111 */
0x66, /* 01100110 */
0x24, /* 0100100 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 30 0x1e ''' */
0x00, /* 00000000 */
0x18, /* 00011000 */
0x3c, /* 00111100 */
0x7e, /* 01111110 */
0xff, /* 11111111 */
0xff, /* 11111111 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 31 0x1f '^-' */
0x00, /* 00000000 */
0xff, /* 11111111 */
0xff, /* 11111111 */
0x7e, /* 01111110 */
0x3c, /* 00111100 */
0x18, /* 00011000 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 32 0x20 ' ' */
0x00, /* 00000000 */

/* 33 0x21 '!' */
0x18, /* 00011000 */
0x3c, /* 00111100 */
0x3c, /* 00111100 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x00, /* 00000000 */
0x18, /* 00011000 */
0x00, /* 00000000 */

/* 34 0x22 ''' */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x24, /* 0100100 */
0x00, /* 00000000 */

/* 35 0x23 '#' */
0x6c, /* 01101100 */
0x6c, /* 01101100 */
0xfe, /* 11111110 */
0x6c, /* 01101100 */
0xfe, /* 11111110 */
0x6c, /* 01101100 */
0x6c, /* 01101100 */
0x00, /* 00000000 */

/* 36 0x24 '$' */
0x18, /* 00011000 */
0x3c, /* 00111110 */
0x60, /* 01100000 */
0x3c, /* 00111100 */
0x06, /* 00000110 */
0x7c, /* 01111100 */
0x18, /* 00011000 */
0x00, /* 00000000 */

/* 37 0x25 '%' */
0x00, /* 00000000 */
0x6c, /* 11000110 */
0xcc, /* 11001100 */
0x18, /* 00011000 */
0x30, /* 00110000 */
0x66, /* 01100110 */
0x6c, /* 11000110 */
0x76, /* 01110110 */
0xcc, /* 11001100 */
0x76, /* 01110110 */
0x00, /* 00000000 */

/* 38 0x26 '&' */
0x38, /* 00111000 */
0x6c, /* 01101100 */
0x38, /* 00111000 */
0x76, /* 01110110 */
0xdc, /* 11011100 */
0xcc, /* 11001100 */
0x76, /* 01110110 */
0x00, /* 00000000 */

/* 39 0x27 ''' */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x30, /* 00110000 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 40 0x28 '(`' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x0c, /* 00001100 */
0x18, /* 00011000 */
0x30, /* 00110000 */
0x30, /* 00110000 */
0x18, /* 00011000 */
0x0c, /* 00001100 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 41 0x29 ')`' */
0x30, /* 00110000 */
0x18, /* 00011000 */
0x0c, /* 00001100 */
0x0c, /* 00001100 */
0x18, /* 00011000 */
0x0c, /* 00001100 */
0x18, /* 00011000 */
0x30, /* 00110000 */
0x00, /* 00000000 */

/* 42 0x2a '*)*' */
0x00, /* 00000000 */
0x66, /* 01100110 */
0x3c, /* 00111100 */
0xff, /* 11111111 */
0x3c, /* 00111100 */
0x66, /* 01100110 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 43 0x2b '+-' */
0x00, /* 00000000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x7e, /* 01111110 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 44 0x2c '.,' */
0x00, /* 00000000 */
0x30, /* 00110000 */

/* 45 0x2d '-.' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x7e, /* 01111110 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 46 0x2e '..' */
0x00, /* 00000000 */
0x30, /* 00110000 */

/* 47 0x2f '/.' */
0x06, /* 00001110 */
0x0c, /* 00001100 */
0x18, /* 00011000 */
0x30, /* 00110000 */
0x60, /* 01100000 */
0x30, /* 01100000 */
0x80, /* 10000000 */
0x00, /* 00000000 */

/* 48 0x30 '0' */
0x38, /* 00111000 */
0x6c, /* 01101100 */
0x6c, /* 11000110 */
0xd6, /* 11010110 */
0x6c, /* 11000110 */
0x6c, /* 01101100 */
0x38, /* 00111000 */
0x00, /* 00000000 */

/* 49 0x31 '1' */
0x18, /* 00011000 */
0x38, /* 00111000 */
0x78, /* 01110000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x7e, /* 01111110 */
0x00, /* 00000000 */

/* 50 0x32 '2' */
0x7c, /* 01111100 */
0x6c, /* 11000110 */
0x06, /* 00001100 */
0x1c, /* 00011100 */
0x30, /* 00110000 */
0x66, /* 01100110 */
0xfe, /* 11111110 */

```

```

0x00, /* 00000000 */
/* 51 0x33 '3' */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0x06, /* 00000110 */
0x3c, /* 00111100 */
0x06, /* 00000110 */
0xc6, /* 11000110 */
0x7c, /* 01111100 */
0x00, /* 00000000 */

/* 52 0x34 '4' */
0x1c, /* 00011100 */
0x3c, /* 00111100 */
0x6c, /* 01101100 */
0xcc, /* 11001100 */
0x0c, /* 00001100 */
0xfe, /* 11111110 */
0x0c, /* 00000000 */
0xfc, /* 11000000 */
0xfc, /* 11111100 */
0x06, /* 00000110 */
0xc6, /* 11000110 */
0x7c, /* 01111100 */
0x00, /* 00000000 */

/* 53 0x35 '5' */
0xfe, /* 11111110 */
0xc0, /* 11000000 */
0xc0, /* 11000000 */
0xfc, /* 11111100 */
0x06, /* 00000110 */
0xc6, /* 11000110 */
0x7c, /* 01111100 */
0x00, /* 00000000 */

/* 54 0x36 '6' */
0x38, /* 00111000 */
0x60, /* 01100000 */
0xc0, /* 11000000 */
0xfc, /* 11111100 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x7c, /* 01111100 */
0x00, /* 00000000 */

/* 55 0x37 '7' */
0xfe, /* 11111110 */
0xc6, /* 11000110 */
0x0c, /* 00001100 */
0x18, /* 00011000 */
0x30, /* 00110000 */
0x30, /* 00110000 */
0x30, /* 00110000 */
0x00, /* 00000000 */

/* 56 0x38 '8' */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x7c, /* 01111100 */
0x00, /* 00000000 */

/* 57 0x39 '9' */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x7e, /* 01111110 */
0x06, /* 00000110 */
0x0c, /* 00001100 */
0x78, /* 01111000 */
0x00, /* 00000000 */

/* 58 0x3a ':' */
0x00, /* 00000000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x00, /* 00000000 */

/* 59 0x3b ';' */
0x00, /* 00000000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x30, /* 00110000 */

/* 60 0x3c '<' */
0x06, /* 00000110 */
0x0c, /* 00000110 */
0x18, /* 00011000 */
0x30, /* 00110000 */
0x18, /* 00011000 */
0x0c, /* 00001100 */
0x06, /* 00000110 */
0x00, /* 00000000 */

/* 61 0x3d '=' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x7e, /* 01111110 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x7e, /* 01111110 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 62 0x3e '>' */
0x60, /* 01100000 */
0x30, /* 00110000 */
0x18, /* 00011000 */
0x0c, /* 00001100 */
0x18, /* 00011000 */
0x30, /* 00110000 */
0x60, /* 01100000 */
0x00, /* 00000000 */

/* 63 0x3f '?' */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0x0c, /* 00001100 */
0x18, /* 00011000 */
0x00, /* 00000000 */
0x18, /* 00011000 */
0x00, /* 00000000 */

/* 64 0x40 '@' */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xde, /* 11011110 */
0xde, /* 11011110 */
0xde, /* 11011110 */
0x0c, /* 00000000 */
0x78, /* 01111000 */
0x00, /* 00000000 */

/* 65 0x41 'A' */
0x38, /* 00111100 */
0x6c, /* 01101100 */
0xc6, /* 11000110 */
0xfe, /* 11111110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x00, /* 00000000 */

/* 66 0x42 'B' */
0xfc, /* 11111100 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x7c, /* 01111100 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0xfc, /* 11111100 */
0x00, /* 00000000 */

/* 67 0x43 'C' */
0x3c, /* 00111100 */
0x66, /* 01100110 */
0xc0, /* 11000000 */
0xc0, /* 11000000 */
0x66, /* 01100110 */
0x3c, /* 00111100 */
0x00, /* 00000000 */

/* 68 0x44 'D' */
0xf8, /* 11111100 */
0x6c, /* 01101100 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x6c, /* 01101100 */
0x8f, /* 11111100 */
0x00, /* 00000000 */

/* 69 0x45 'E' */
0xfe, /* 11111110 */
0x62, /* 01100010 */
0x68, /* 01101000 */
0x78, /* 01111000 */
0x68, /* 01101000 */
0x62, /* 01100010 */
0xfe, /* 11111110 */
0x00, /* 00000000 */

/* 70 0x46 'F' */
0xfe, /* 11111110 */
0x62, /* 01100010 */
0x68, /* 01101000 */
0x78, /* 01111000 */
0x68, /* 01101000 */
0x60, /* 01100000 */
0xf0, /* 11110000 */
0x00, /* 00000000 */

/* 71 0x47 'G' */
0x3c, /* 00111100 */
0x66, /* 01100110 */
0xc0, /* 11000000 */
0xc0, /* 11000000 */
0xce, /* 11001110 */
0x66, /* 01100110 */
0x3a, /* 00111101 */
0x00, /* 00000000 */

/* 72 0x48 'H' */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xfe, /* 11111110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x00, /* 00000000 */

/* 73 0x49 'I' */
0x3c, /* 00111100 */

```

```

0x18, /* 00011000 */
0x3c, /* 00111100 */
0x00, /* 00000000 */

/* 74 0x4a 'J' */
0x1e, /* 00011110 */
0x0c, /* 00001100 */
0x0c, /* 00001100 */
0x0c, /* 00001100 */
0x0c, /* 00001100 */
0xcc, /* 11001100 */
0xcc, /* 11001100 */
0x78, /* 01111000 */
0x00, /* 00000000 */

/* 75 0x4b 'K' */
0xc6, /* 11100110 */
0x66, /* 01100110 */
0x6c, /* 01101100 */
0x78, /* 01111000 */
0x6c, /* 01101100 */
0x66, /* 01100110 */
0x6e, /* 11100110 */
0x00, /* 00000000 */

/* 76 0x4c 'L' */
0xf0, /* 11110000 */
0x60, /* 01100000 */
0x60, /* 01100000 */
0x60, /* 01100000 */
0x62, /* 01100010 */
0x66, /* 01100110 */
0xfe, /* 11111110 */
0x00, /* 00000000 */

/* 77 0x4d 'M' */
0xc6, /* 11000110 */
0xee, /* 11101110 */
0xfe, /* 11111110 */
0xfe, /* 11111110 */
0xd6, /* 11010110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x00, /* 00000000 */

/* 78 0x4e 'N' */
0xc6, /* 11000110 */
0xc6, /* 11100110 */
0xf6, /* 11110110 */
0xde, /* 11011110 */
0xce, /* 11001110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x00, /* 00000000 */

/* 79 0x4f 'O' */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x7c, /* 01111100 */
0x00, /* 00000000 */

/* 80 0x50 'P' */
0xfc, /* 11111100 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x7c, /* 01111100 */
0x60, /* 01100000 */
0x60, /* 01100000 */
0x00, /* 00000000 */

/* 81 0x51 'Q' */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x7c, /* 01111100 */
0x0e, /* 00001110 */

/* 82 0x52 'R' */
0xfc, /* 11111100 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x7c, /* 01111100 */
0x6c, /* 01101100 */
0x66, /* 01100110 */
0x66, /* 11000110 */
0x00, /* 00000000 */

/* 83 0x53 'S' */
0x3c, /* 00111100 */
0x66, /* 01100110 */
0x30, /* 00110000 */
0x18, /* 00011000 */
0x0c, /* 00001100 */
0x66, /* 01100110 */
0x3c, /* 01111100 */
0x00, /* 00000000 */

/* 84 0x54 'T' */
0x7e, /* 01111110 */
0x7e, /* 01111110 */
0x5a, /* 01011010 */
0x18, /* 00001100 */
0x66, /* 01100110 */
0x3c, /* 01111100 */
0x00, /* 00000000 */

/* 85 0x55 'U' */
0xc6, /* 11000110 */
0xc6, /* 01111100 */
0x00, /* 00000000 */

/* 86 0x56 'V' */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xc6, /* 01101100 */
0x38, /* 00111000 */
0x00, /* 00000000 */

/* 87 0x57 'W' */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xd6, /* 11010110 */
0xd6, /* 11010110 */
0xe, /* 11111110 */
0xc6, /* 01101100 */
0x00, /* 00000000 */

/* 88 0x58 'X' */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xc6, /* 01101100 */
0x38, /* 00111000 */
0xc6, /* 01101100 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x00, /* 00000000 */

/* 89 0x59 'Y' */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x3c, /* 00111100 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x3c, /* 00111100 */
0x00, /* 00000000 */

/* 90 0x5a 'Z' */
0xfe, /* 11111110 */
0xc6, /* 11000110 */
0x8c, /* 100001100 */
0x18, /* 00011000 */
0x32, /* 00110010 */
0x66, /* 01100110 */
0xfe, /* 11111110 */
0x00, /* 00000000 */

/* 91 0x5b '[' */
0x3c, /* 00111100 */
0x30, /* 00110000 */
0x30, /* 00110000 */
0x30, /* 00110000 */
0x30, /* 00110000 */
0x3c, /* 00111100 */
0x00, /* 00000000 */

/* 92 0x5c '\\' */
0xc0, /* 11000000 */
0x60, /* 01100000 */
0x30, /* 00110000 */
0x18, /* 00011000 */
0x0c, /* 00001100 */
0x06, /* 00000110 */
0x02, /* 00000010 */
0x00, /* 00000000 */

/* 93 0x5d ']' */
0x3c, /* 00111100 */
0x0c, /* 00001100 */
0x0c, /* 00001100 */
0x0c, /* 00001100 */
0x0c, /* 00001100 */
0x3c, /* 00111100 */
0x00, /* 00000000 */

/* 94 0x5e '^' */
0x10, /* 00010000 */
0x38, /* 00111000 */
0x6c, /* 01101100 */
0xc6, /* 11000110 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 95 0x5f '_' */
0x00, /* 00000000 */

```

```

0x00, /* 00000000 */
0x00, /* 00000000 */
0xff, /* 11111111 */

/* 96 0x60 'c' */
0x30, /* 00110000 */
0x18, /* 00011000 */
0x0c, /* 00001100 */
0x00, /* 00000000 */

/* 97 0x61 'd' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x78, /* 01111000 */
0x0c, /* 00001100 */
0x7c, /* 01111100 */
0xcc, /* 11001100 */
0x76, /* 01101110 */
0x00, /* 00000000 */

/* 98 0x62 'e' */
0xe0, /* 11100000 */
0x60, /* 01000000 */
0x7c, /* 01111100 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0xdc, /* 11011100 */
0x00, /* 00000000 */

/* 99 0x63 'f' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xc0, /* 11000000 */
0xc6, /* 11000110 */
0x7c, /* 01111100 */
0x00, /* 00000000 */

/* 100 0x64 'g' */
0x1c, /* 00011100 */
0x0c, /* 00001100 */
0x7c, /* 01111100 */
0xcc, /* 11001100 */
0xcc, /* 11001100 */
0x76, /* 01101110 */
0x00, /* 00000000 */

/* 101 0x65 'h' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xfe, /* 11111110 */
0xc0, /* 11000000 */
0x7c, /* 01111100 */
0x00, /* 00000000 */

/* 102 0x66 'i' */
0x3c, /* 00111100 */
0x66, /* 01100110 */
0x60, /* 01100000 */
0x18, /* 11111000 */
0x60, /* 01100000 */
0x60, /* 01100000 */
0xf0, /* 11110000 */
0x00, /* 00000000 */

/* 103 0x67 'j' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x76, /* 01101110 */
0xcc, /* 11001100 */
0x7c, /* 01111100 */
0x0c, /* 00001100 */
0x18, /* 11111000 */

/* 104 0x68 'k' */
0xe0, /* 11100000 */
0x60, /* 01100000 */
0x6c, /* 01101100 */
0x76, /* 01101110 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x00, /* 00000000 */

/* 105 0x69 'l' */
0x18, /* 00011000 */
0x00, /* 00000000 */
0x38, /* 00111000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x3c, /* 00111100 */
0x00, /* 00000000 */

/* 106 0x6a 'm' */
0x06, /* 00000110 */
0x00, /* 00000000 */
0x06, /* 00000110 */
0x06, /* 00000110 */
0x06, /* 00000110 */
0x66, /* 01100110 */
0x66, /* 01100110 */

/* 107 0x6b 'n' */
0xe0, /* 11100000 */
0x60, /* 01100000 */
0x66, /* 01101110 */
0x6c, /* 01101100 */
0x78, /* 01111000 */
0x6c, /* 01101100 */
0x6e, /* 11100110 */
0x00, /* 00000000 */

/* 108 0x6c 'o' */
0x38, /* 00111000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x3c, /* 00111100 */
0x00, /* 00000000 */

/* 109 0x6d 'p' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0xec, /* 11101100 */
0xfe, /* 11111110 */
0xd6, /* 11010110 */
0xd6, /* 11010110 */
0xd6, /* 11010110 */
0x00, /* 00000000 */

/* 110 0x6e 'q' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0xd0, /* 11011100 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x00, /* 00000000 */

/* 111 0x6f 'r' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x7c, /* 01111100 */
0x00, /* 00000000 */

/* 112 0x70 's' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0xd0, /* 11011100 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x7c, /* 01111100 */
0x60, /* 01100000 */
0xf0, /* 11110000 */
0x00, /* 00000000 */

/* 113 0x71 't' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x76, /* 01110110 */
0xcc, /* 11001100 */
0xcc, /* 11001100 */
0x7c, /* 01111100 */
0xc0, /* 00001100 */
0x1e, /* 00111110 */

/* 114 0x72 'u' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0xdc, /* 11011100 */
0x76, /* 01110110 */
0x60, /* 01100000 */
0x60, /* 01100000 */
0x60, /* 01100000 */
0x00, /* 00000000 */

/* 115 0x73 'v' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x7e, /* 01111110 */
0xc0, /* 11000000 */
0x7c, /* 01111100 */
0x06, /* 00000110 */
0xfc, /* 11111100 */
0x00, /* 00000000 */

/* 116 0x74 'w' */
0x30, /* 00110000 */
0x30, /* 00110000 */
0xfc, /* 11111100 */
0x30, /* 00110000 */
0x30, /* 00110000 */
0x36, /* 00110110 */
0x1c, /* 00111100 */
0x00, /* 00000000 */

/* 117 0x75 'x' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0xcc, /* 11001100 */
0x3c, /* 00111100 */
0x3c, /* 00111100 */
0x76, /* 01110110 */
0x00, /* 00000000 */

```

```

/* 118 0x76 'v' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x6c, /* 01101100 */
0x38, /* 00111000 */
0x00, /* 00000000 */

/* 119 0x77 'w' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0xc6, /* 11000110 */
0xd6, /* 11010110 */
0xd6, /* 11010110 */
0xfe, /* 11111110 */
0x6c, /* 01101100 */
0x00, /* 00000000 */

/* 120 0x78 'x' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0xc6, /* 11000110 */
0x6c, /* 01101100 */
0x38, /* 00111000 */
0x6c, /* 01101100 */
0xc6, /* 11000110 */
0x00, /* 00000000 */

/* 121 0x79 'y' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x7e, /* 01111110 */
0x06, /* 00011000 */
0xfc, /* 11111100 */

/* 122 0x7a 'z' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x7e, /* 01111110 */
0x4c, /* 01001100 */
0x18, /* 00011000 */
0x32, /* 00110010 */
0x7e, /* 01111110 */
0x00, /* 00000000 */

/* 123 0x7b '{' */
0xe0, /* 00001110 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x70, /* 01101000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0xe0, /* 00001110 */
0x00, /* 00000000 */

/* 124 0x7c '|' */
0x18, /* 00011000 */
0x00, /* 00000000 */

/* 125 0x7d '}' */
0x70, /* 01110000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0xe0, /* 00001110 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x70, /* 01101000 */
0x00, /* 00000000 */

/* 126 0x7e '^' */
0x76, /* 01101100 */
0xdc, /* 11011100 */
0x00, /* 00000000 */

/* 127 0x7f '-' */
0x00, /* 00000000 */
0x10, /* 00010000 */
0x38, /* 00111000 */
0x6c, /* 01101100 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xfe, /* 11111110 */
0x00, /* 00000000 */

/* 128 0x80 ',' */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xc0, /* 11000000 */
0xc0, /* 11000000 */
0xc6, /* 11000110 */
0x7c, /* 01111100 */
0x0c, /* 00001100 */
0x78, /* 01111100 */

/* 129 0x81 ',' */
0xcc, /* 11001100 */
0x00, /* 00000000 */

/* 130 0x82 ',' */
0x00, /* 00000000 */
0xcc, /* 11000110 */
0xcc, /* 11000110 */
0xcc, /* 11000110 */
0x76, /* 01101100 */
0x00, /* 00000000 */

/* 131 0x83 ',' */
0x7c, /* 01111100 */
0x82, /* 10000010 */
0x78, /* 01111100 */
0x0c, /* 00001100 */
0x7c, /* 01111100 */
0xcc, /* 11000110 */
0x76, /* 01101100 */
0x00, /* 00000000 */

/* 132 0x84 ',' */
0xc6, /* 11000110 */
0x00, /* 00000000 */
0x78, /* 01111100 */
0x0c, /* 00001100 */
0x7c, /* 01111100 */
0xcc, /* 11000110 */
0x76, /* 01101100 */
0x00, /* 00000000 */

/* 133 0x85 ',' */
0x30, /* 00110000 */
0x18, /* 00011000 */
0x78, /* 01111100 */
0x0c, /* 00001100 */
0x7c, /* 01111100 */
0xcc, /* 11000110 */
0x76, /* 01101100 */
0x00, /* 00000000 */

/* 134 0x86 ',' */
0x30, /* 00110000 */
0x30, /* 00110000 */
0x78, /* 01111100 */
0x0c, /* 00001100 */
0x7c, /* 01111100 */
0xcc, /* 11000110 */
0x76, /* 01101100 */
0x00, /* 00000000 */

/* 135 0x87 ',' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x7c, /* 01111110 */
0xc0, /* 11000000 */
0xc0, /* 11000000 */
0x7e, /* 01111110 */
0x0c, /* 00001100 */
0x38, /* 00111000 */

/* 136 0x88 ',' */
0x7c, /* 01111100 */
0x82, /* 10000010 */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xfe, /* 11111110 */
0xc0, /* 11000000 */
0x7c, /* 01111100 */
0x00, /* 00000000 */

/* 137 0x89 ',' */
0xc6, /* 11000110 */
0x00, /* 00000000 */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xfe, /* 11111110 */
0xc0, /* 11000000 */
0x7c, /* 01111100 */
0x00, /* 00000000 */

/* 138 0x8a ',' */
0x30, /* 00110000 */
0x18, /* 00011000 */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xfe, /* 11111110 */
0xc0, /* 11000000 */
0x7c, /* 01111100 */
0x00, /* 00000000 */

/* 139 0x8b ',' */
0x66, /* 01100110 */
0x00, /* 00000000 */
0x38, /* 00111000 */
0x18, /* 00011000 */
0x00, /* 00001100 */
0x18, /* 00011000 */
0x3c, /* 00111100 */
0x00, /* 00000000 */

/* 140 0x8c ',' */
0x7c, /* 01111100 */
0x82, /* 10000010 */
0x38, /* 00111000 */

```

```

0x18, /* 00011000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x3c, /* 00111100 */
0x00, /* 00000000 */

/* 141 0x8d ' ' */
0x30, /* 00110000 */
0x18, /* 00011000 */
0x00, /* 00000000 */
0x38, /* 00111000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x3c, /* 00111100 */
0x00, /* 00000000 */

/* 142 0x8e ' ' */
0xc6, /* 11000110 */
0x38, /* 00111000 */
0x6c, /* 01101100 */
0xc6, /* 11000110 */
0xfe, /* 11111110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x00, /* 00000000 */

/* 143 0x8f ' ' */
0x38, /* 00111000 */
0x6c, /* 01101100 */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xfe, /* 11111110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x00, /* 00000000 */

/* 144 0x90 ' ' */
0x18, /* 00011000 */
0x30, /* 00110000 */
0xfe, /* 11111110 */
0xc0, /* 11000000 */
0xf8, /* 11111000 */
0xc0, /* 11000000 */
0xfe, /* 11111110 */
0x00, /* 00000000 */

/* 145 0x91 ' ' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x7c, /* 01111110 */
0x18, /* 00011000 */
0x7c, /* 01111110 */
0xd8, /* 11011000 */
0x7c, /* 01111110 */
0x00, /* 00000000 */

/* 146 0x92 ' ' */
0x3e, /* 00111110 */
0x6c, /* 01101100 */
0xcc, /* 11001100 */
0xfe, /* 11111110 */
0xcc, /* 11001100 */
0xcc, /* 11001100 */
0xcc, /* 11001110 */
0x00, /* 00000000 */

/* 147 0x93 ' ' */
0x7c, /* 01111100 */
0x82, /* 10000100 */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x7c, /* 01111100 */
0x00, /* 00000000 */

/* 148 0x94 ' ' */
0xc6, /* 11000110 */
0x00, /* 00000000 */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x7c, /* 01111100 */
0x00, /* 00000000 */

/* 149 0x95 ' ' */
0x30, /* 00110000 */
0x18, /* 00011000 */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x7c, /* 01111100 */
0x00, /* 00000000 */

/* 150 0x96 ' ' */
0x78, /* 01111000 */
0x84, /* 10000100 */
0x00, /* 00000000 */
0xcc, /* 11001100 */
0xcc, /* 11001100 */
0x76, /* 01110110 */
0x00, /* 00000000 */

/* 151 0x97 ' ' */
0x60, /* 01100000 */
0x30, /* 00110000 */
0xcc, /* 11001100 */
0xcc, /* 11001100 */
0xcc, /* 11001100 */

/* 152 0x98 ' ' */
0xcc, /* 11001100 */
0x76, /* 01110110 */
0x00, /* 00000000 */

/* 153 0x99 ' ' */
0xc6, /* 11000110 */
0x38, /* 00111000 */
0x6c, /* 01101100 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x7e, /* 01111110 */
0x06, /* 00000110 */
0xfc, /* 11111100 */

/* 154 0x9a ' ' */
0xc6, /* 11000110 */
0x00, /* 00000000 */
0x6c, /* 11000110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x7c, /* 01111100 */
0x00, /* 00000000 */

/* 155 0x9b ' ' */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x7e, /* 01111110 */
0xc0, /* 11000000 */
0xe0, /* 11000000 */
0x7e, /* 01111110 */
0x18, /* 00011000 */
0x18, /* 00011000 */

/* 156 0x9c ' ' */
0x38, /* 00111000 */
0x6c, /* 01101100 */
0x64, /* 01100100 */
0xf0, /* 11110000 */
0x60, /* 01100000 */
0x66, /* 01100110 */
0xfc, /* 11111100 */
0x00, /* 00000000 */

/* 157 0x9d ' ' */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x3c, /* 00111100 */
0x7e, /* 01111110 */
0x18, /* 00011000 */
0x7e, /* 01111110 */
0x18, /* 00011000 */
0x18, /* 00011000 */

/* 158 0x9e ' ' */
0x18, /* 00011000 */
0xcc, /* 11001100 */
0xfa, /* 11111010 */
0xc6, /* 11000110 */
0xcf, /* 11001111 */
0xc6, /* 11000110 */
0xc7, /* 11000111 */

/* 159 0x9f ' ' */
0x0e, /* 00001110 */
0x1b, /* 00011011 */
0x18, /* 00011000 */
0x3c, /* 00111100 */
0x18, /* 00011000 */
0x8d, /* 11011000 */
0x70, /* 01110000 */
0x00, /* 00000000 */

/* 160 0xa0 ' ' */
0x18, /* 00011000 */
0x30, /* 00110000 */
0x78, /* 01111000 */
0x0c, /* 00001100 */
0x7c, /* 01111100 */
0xcc, /* 11001100 */
0x76, /* 01110110 */
0x00, /* 00000000 */

/* 161 0xa1 ' ' */
0x0c, /* 00001100 */
0x18, /* 00011000 */
0x00, /* 00000000 */
0x38, /* 00111000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x3c, /* 00111100 */
0x00, /* 00000000 */

/* 162 0xa2 ' ' */
0x0c, /* 00001100 */
0x18, /* 00011000 */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x7c, /* 01111100 */

```

```

0x00, /* 00000000 */
/* 163 0xa3 ' ' */
0x18, /* 00011000 */
0x30, /* 00110000 */
0xcc, /* 11001100 */
0xcc, /* 11001100 */
0xcc, /* 11001100 */
0xcc, /* 11001100 */
0x76, /* 01110110 */
0x00, /* 00000000 */

/* 164 0xa4 ' ' */
0x76, /* 01110110 */
0xdc, /* 11011100 */
0x00, /* 00000000 */
0xdc, /* 11011100 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x00, /* 00000000 */

/* 165 0xa5 ' ' */
0x76, /* 01110110 */
0xdc, /* 11011100 */
0x00, /* 00000000 */
0x66, /* 11100110 */
0xf6, /* 11110110 */
0xde, /* 11011110 */
0xce, /* 11001110 */
0x00, /* 00000000 */

/* 166 0xa6 ' ' */
0x3c, /* 00111100 */
0x6c, /* 01101100 */
0x6c, /* 01101100 */
0x3e, /* 00111110 */
0x00, /* 00000000 */
0x7e, /* 01111110 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 167 0xa7 ' ' */
0x38, /* 00111000 */
0x6c, /* 01101100 */
0x6c, /* 01101100 */
0x38, /* 00111000 */
0x00, /* 00000000 */
0x7c, /* 01111100 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 168 0xa8 ' ' */
0x18, /* 00011000 */
0x00, /* 00000000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x30, /* 00110000 */
0x63, /* 01100011 */
0x3e, /* 01111110 */
0x00, /* 00000000 */

/* 169 0xa9 ' ' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0xfe, /* 11111110 */
0xc0, /* 11000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 170 0xaa ' ' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0xfc, /* 11111110 */
0x06, /* 00000110 */
0x06, /* 00000110 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 171 0xab ' ' */
0x63, /* 01100011 */
0x6e, /* 11100110 */
0x6c, /* 01101100 */
0x7e, /* 01111110 */
0x33, /* 00110011 */
0x66, /* 01100110 */
0xcc, /* 11001100 */
0x0f, /* 00001111 */

/* 172 0xac ' ' */
0x63, /* 01100011 */
0x6e, /* 11100110 */
0x6c, /* 01101100 */
0x7a, /* 01111010 */
0x36, /* 00110110 */
0x6a, /* 01101010 */
0xdf, /* 11011111 */
0x06, /* 00000110 */

/* 173 0xad ' ' */
0x18, /* 00011000 */
0x00, /* 00000000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x3c, /* 00111100 */
0x3c, /* 00111100 */
0x18, /* 00011000 */
0x00, /* 00000000 */

/* 174 0xae ' ' */
0x00, /* 00000000 */
0x33, /* 00110011 */
0x66, /* 01100110 */
0xcc, /* 11001100 */
0x66, /* 01100110 */
0x33, /* 00110011 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 175 0xaf ' ' */
0x00, /* 00000000 */
0xcc, /* 11001100 */
0x66, /* 01100110 */
0x33, /* 00110011 */
0x66, /* 01100110 */
0xcc, /* 11001100 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 176 0xb0 ' ' */
0x22, /* 00100010 */
0x88, /* 10001000 */

/* 177 0xb1 ' ' */
0x55, /* 01010101 */
0xaa, /* 10101010 */

/* 178 0xb2 ' ' */
0x77, /* 01110111 */
0xdd, /* 11011101 */

/* 179 0xb3 ' ' */
0x18, /* 00011000 */

/* 180 0xb4 ' ' */
0x18, /* 00011000 */

/* 181 0xb5 ' ' */
0x18, /* 00011000 */

/* 182 0xb6 ' ' */
0x36, /* 00110110 */
0x36, /* 00110110 */
0x36, /* 00110110 */
0x16, /* 11110110 */
0x36, /* 00110110 */
0x36, /* 00110110 */
0x36, /* 00110110 */
0x36, /* 00110110 */

/* 183 0xb7 ' ' */
0x00, /* 00000000 */

/* 184 0xb8 ' ' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x18, /* 00011000 */

/* 185 0xb9 ' ' */
0x36, /* 00110110 */

```



```

0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 208 0xd0 ' ' */
0x36, /* 00110110 */
0x36, /* 00110110 */
0x36, /* 00110110 */
0x36, /* 00110110 */
0xff, /* 11111111 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 209 0xd1 ' ' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0xff, /* 11111111 */
0x00, /* 00000000 */
0xff, /* 11111111 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x18, /* 00011000 */

/* 210 0xd2 ' ' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0xff, /* 11111111 */
0x36, /* 00110110 */
0x36, /* 00110110 */
0x36, /* 00110110 */

/* 211 0xd3 ' ' */
0x36, /* 00110110 */
0x36, /* 00110110 */
0x36, /* 00110110 */
0x36, /* 00110110 */
0x3f, /* 00111111 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 212 0xd4 ' ' */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x1f, /* 00011111 */
0x18, /* 00011000 */
0x1f, /* 00011111 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 213 0xd5 ' ' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x1f, /* 00011111 */
0x18, /* 00011000 */
0x1f, /* 00011111 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x18, /* 00011000 */

/* 214 0xd6 ' ' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x3f, /* 00111111 */
0x36, /* 00110110 */
0x36, /* 00110110 */
0x36, /* 00110110 */

/* 215 0xd7 ' ' */
0x36, /* 00110110 */
0x36, /* 00110110 */
0x36, /* 00110110 */
0x36, /* 00110110 */
0xff, /* 11111111 */
0x36, /* 00110110 */
0x36, /* 00110110 */
0x36, /* 00110110 */

/* 216 0xd8 ' ' */
0x18, /* 00011000 */
0x18, /* 00011000 */
0xff, /* 11111111 */
0x18, /* 00011000 */
0xff, /* 11111111 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x18, /* 00011000 */

/* 217 0xd9 ' ' */
0x18, /* 00011000 */
0x18, /* 00011000 */
0x18, /* 00011000 */
0xf8, /* 11111000 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 218 0xda ' ' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x1f, /* 00011111 */
0x18, /* 00011000 */
0x18, /* 00011000 */

```

```

/* 230 0xe6 ' ' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x7c, /* 01111100 */
0xc0, /* 11000000 */

/* 231 0xe7 ' ' */
0x00, /* 00000000 */
0x76, /* 01110110 */
0xdc, /* 11011100 */
0x18, /* 00011000 */
0x00, /* 00000000 */

/* 232 0xe8 ' ' */
0x7e, /* 01111100 */
0x18, /* 00011000 */
0x3c, /* 00111100 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x3c, /* 00111100 */
0x18, /* 00011000 */
0x7e, /* 01111110 */

/* 233 0xe9 ' ' */
0x38, /* 00111000 */
0x6c, /* 01101100 */
0xc6, /* 11000110 */
0xfe, /* 11111110 */
0xc6, /* 11000110 */
0x6c, /* 01101100 */
0x38, /* 00111000 */
0x00, /* 00000000 */

/* 234 0xea ' ' */
0x38, /* 00111100 */
0x6c, /* 01101100 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x6c, /* 01101100 */
0x6c, /* 01101100 */
0xee, /* 11011110 */
0x00, /* 00000000 */

/* 235 0xeb ' ' */
0xe0, /* 00001110 */
0x18, /* 00011000 */
0x0c, /* 00001100 */
0x3e, /* 00111110 */
0x66, /* 01100110 */
0x66, /* 01100110 */
0x3c, /* 00111110 */
0x00, /* 00000000 */

/* 236 0xec ' ' */
0x00, /* 00000000 */
0x00, /* 00000000 */
0x7e, /* 01111110 */
0xdb, /* 11011011 */
0xdb, /* 11011011 */
0x7e, /* 01111110 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 237 0xed ' ' */
0x06, /* 00000110 */
0x0c, /* 00001100 */
0x7e, /* 01111110 */
0xdb, /* 11011011 */
0xdb, /* 11011011 */
0x7e, /* 01111110 */
0x60, /* 01100000 */
0xc0, /* 11000000 */

/* 238 0xee ' ' */
0x1e, /* 00011110 */
0x30, /* 00110000 */
0x60, /* 01100000 */
0x7e, /* 01111110 */
0x60, /* 01100000 */
0x30, /* 00110000 */
0x1e, /* 00011110 */
0x00, /* 00000000 */

/* 239 0xef ' ' */
0x00, /* 00000000 */
0x7c, /* 01111100 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0xc6, /* 11000110 */
0x00, /* 00000000 */

/* 240 0xf0 ' ' */
0x00, /* 00000000 */
0xfe, /* 11111110 */
0x00, /* 00000000 */
0xfe, /* 11111110 */
0x00, /* 00000000 */
0xfe, /* 11111110 */
0x00, /* 00000000 */
0x00, /* 00000000 */

/* 241 0xf1 ' ' */
0x18, /* 00011000 */

```

