

**GMICRO – GRUPO DE MICROELETRÔNICA
UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA**



MCP2210 UTILIZANDO A BIBLIOTECA LIBUSB

**Bernardo Favero Andreeti
Eduardo Luzzi
José Augusto Comiotto Rottini**

**Santa Maria, RS, Brasil
2014**

MCP2210 UTILIZANDO A BIBLIOTECA LIBUSB

**Bernardo Favero Andreeti
Eduardo Luzzi
José Augusto Comiotto Rottini**

Manual referente ao código de manipulação do kit de desenvolvimento MCP2210 utilizando a biblioteca libusb. Projeto do Grupo de Microeletrônica da Universidade Federal de Santa Maria (GMICRO - UFSM, RS).

Orientador: Dr. Everton Alceu Carara

Santa Maria, RS, Brasil

2014

LISTA DE FIGURAS

Figura 1 – Erro na instalação da biblioteca libusb.....	7
Figura 2 – Analise do código de exemplo.	8
Figura 3 – Código do Chip Settings.	10
Figura 4 – Código do Transfer Settings.....	11
Figura 5 – Código de configuração da transferência.	12
Figura 6 – Código para leitura da temperatura.	15

LISTA DE TABELAS

Tabela 1 – Descrição Do Chip Settings.....	10
Tabela 2 – Descrição Do Transfer Settings.	11
Tabela 3 – Descrição Das Configurações Para Transferência.....	12
Tabela 4 – Descrição Das Configurações Para A Leitura Da Temperatura.	14

SUMÁRIO

INTRODUÇÃO	6
1. INSTALAÇÃO E UTILIZAÇÃO DA LIBUSB	7
1.1 Instalação.....	7
1.2 Utilização	8
2. DEFINIÇÕES	9
2.1 Device Endpoint.....	9
2.2 Host Endpoint	9
2.3 Device VID	9
2.4 Device PID	9
3. CONFIGURAÇÃO	10
3.1 Chip Settings	10
3.2 Transfer Settings	11
4. TRANSFERÊNCIA	12
5. FUNÇÕES DA LIBUSB UTILIZADAS	13
6. LEITURA DA TEMPERATURA	14
REFERÊNCIAS	16

INTRODUÇÃO

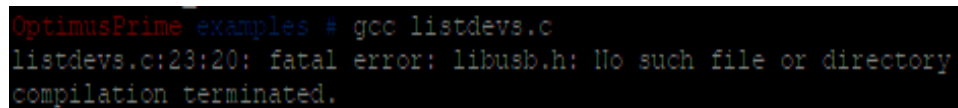
O código implementado em linguagem C realiza a transferência de dados e a leitura da temperatura utilizando o kit de desenvolvimento MCP2210. Para auxiliar a construção do mesmo, fez-se uso da biblioteca libusb.

Libusb é uma biblioteca *open source* que fornece aos aplicativos fácil acesso aos dispositivos USB, podendo ser utilizada em diversos sistemas operacionais. Nesse projeto foi utilizado o sistema operacional Linux.

1. INSTALAÇÃO E UTILIZAÇÃO DA LIBUSB

1.1 Instalação

- Faça o download do arquivo “**libusb-1.0.9.tar.bz2**”, encontrado no site <http://www.libusb.org/>.
- Extraia o arquivo e abra o terminal.
- Digite “**sudo su**” e coloque a senha.
- Pelo terminal, acesse a pasta da lib que foi extraída:
“**cd /home/nomedousuario/Downloads/libusb-1.0.9**”.
- Para compilar e instalar a biblioteca, digite os seguintes comandos:
“**./configure**”.
“**make**”
“**make install**”
- Instale o pacote dev: “**sudo apt-get install libusb-1.0-0-dev**”
- Teste o programa: “**gcc listdevs.c**”
- Erro 1:

A terminal window with a black background and green text. The prompt is 'OptimusPrime examples #'. The user has entered 'gcc listdevs.c'. The output shows a fatal error: 'listdevs.c:23:20: fatal error: libusb.h: No such file or directory compilation terminated.'

```
OptimusPrime examples # gcc listdevs.c
listdevs.c:23:20: fatal error: libusb.h: No such file or directory
compilation terminated.
```

Figura 1 – Erro na instalação da biblioteca libusb.

Esse erro informa que não encontrou o arquivo “**libusb.h**”. Se analisarmos o código do programa de exemplo, temos na figura 2:

```
OptimusPrime examples # cat listdevs.c
/*
 * libusb example program to list devices on the bus
 * Copyright (C) 2007 Daniel Drake <dsd@gentoo.org>
 *
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with this library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

#include <stdio.h>
#include <sys/types.h>

#include <libusb.h>

static void print_devs(libusb_device **devs)
{
    libusb_device *dev;
```

Figura 2 – Análise do código de exemplo.

O “`#include <libusb.h>`” não está encontrando a lib, portanto ela está instalada em outro local.

- Para encontrar a lib, execute o seguinte comando no terminal:
“`ls /usr/include/libusb-1.0/`”
- Verifique se o retorno do comando será “`libusb.h`”.

1.2 Utilização

A inclusão da biblioteca libusb no código deve ser da forma “`#include <libusb-1.0/libusb.h>`”, ao invés de apenas “`#include <libusb.h>`”.

Ao se fazer a compilação do código é utilizado o seguinte comando: “`gcc prog.c -o prog.exe -lusb-1.0`” no terminal do Linux.

Para manipular o dispositivo USB, antes é necessário verificar se o Sistema operacional está com o controle do mesmo. Caso esteja, deve ser solicitado a liberação para depois prosseguir com a comunicação. Devido a isso, os códigos devem ser compilados sob privilégios *root*.

2. DEFINIÇÕES

Os valores de endereçamento do dispositivo foram previamente configurados e são fixos para o dispositivo em questão.

2.1 Device Endpoint

Trata-se do endereço que o protocolo USB utiliza para o dispositivo MCP2210:

```
#define DEV_ENDPOINT 0x01
```

2.2 Host Endpoint

Trata-se do endereço do host:

```
#define HOST_ENDPOINT 0x81
```

2.3 Device VID

É a identificação do fornecedor do dispositivo:

```
#define DEV_VID 1240
```

2.4 Device PID

É a identificação do produto:

```
#define DEV_PID 222
```

3. CONFIGURAÇÃO

A configuração do dispositivo se baseia no envio de três vetores de 64 posições. Cada posição contém os parâmetros descritos conforme as tabelas abaixo, juntamente com o segmento do código que os representa. Alguns desses parâmetros são formados por duas posições, sendo a primeira a parte baixa e a segunda a parte alta do mesmo.

Tais valores foram escolhidos conforme o manual do kit disponível na seção de bibliografia, onde podem ser encontradas informações mais detalhadas acerca do assunto.

3.1 Chip Settings

Posição	Função	Valor
0	Parâmetros NVRAM (padrão).	0x60
1	Configurações Iniciais (padrão).	0x20
2,3	(Reservado).	0x0000
4-12	Chip Selects: Todos ativados.	0xFF
13,14	Valor da Saída GPIO (low byte e high byte, respectivamente).	0xFFFF
15,16	Direção de GPIO (low e high): Definido como saída.	0xFFFF
17	Wake-up desabilitado, Sem Interrupt Counting e o SPI Bus não é liberado entre as transferências.	0x01
18	Define as configurações do chip como não protegidas.	0x00
19-26	Sem Password.	0x00
27-63	(Reservado).	0x00

Tabela 1 – Descrição do Chip Settings.

```

/* SET CHIP SETTINGS POWER-UP DEFAULT */
SetChipSettings[0] = 0x60; // Set NVRAM Parameters Command Code
SetChipSettings[1] = 0x20; // Set Chip Settings
SetChipSettings[2] = 0x00;
SetChipSettings[3] = 0x00;
for(n=4;n<13;n++)
{
    SetChipSettings[n] = 0x01; // All GP's as Chip Select
}
SetChipSettings[13] = 0xFF; // GPIO Value
SetChipSettings[14] = 0xFF;
SetChipSettings[15] = 0xFF; // GPIO Direction
SetChipSettings[16] = 0xFF;
SetChipSettings[17] = 0x01; // Wake-up Disabled, No Interrupt Counting, SPI Bus is not Released Between Transfer
SetChipSettings[18] = 0x00; // Chip Settings not protected
for(n=19;n<64;n++)
{
    SetChipSettings[n] = 0x00; // Reserved
}

```

Figura 3 – Código do Chip Settings.

3.2 Transfer Settings

Posição	Função	Valor
0	Parâmetros NVRAM (padrão).	0x60
1	Configurações de Transferência SPI (padrão).	0x10
2,3	(Reservado).	0x0000
4-7	Bit Rate: 6.000.000 bps.	0x005B8D80
8,9	Idle Chip Select (low e high byte, respectivamente).	0xFFFF
10,11	Active Chip Select (low e high byte).	0xFFEF
12,13	Chip Select to Data Delay (low e high): Sem Delay.	0x0000
14,15	Delay entre último dado e mudança no CS (low e high): Sem Delay.	0x0000
16,17	Delay entre bytes subsequentes (low e high): Sem Delay.	0x0000
18,19	Bytes por Transferência (low e high):3.	0x0003
20	Modo SPI:0.	0x00
21-63	(Reservado).	0x00

Tabela 2 – Descrição do Transfer Settings.

```

/* SET SPI POWER-UP TRANSFER SETTINGS */
SetSpiSettings[0] = 0x60; // Set NVRAM Parameters Command Code
SetSpiSettings[1] = 0x10; // Set SPI Transfer Settings
SetSpiSettings[2] = 0x00;
SetSpiSettings[3] = 0x00;
SetSpiSettings[4] = 0x80; // 4 Bytes to configure Bit Rate
SetSpiSettings[5] = 0x8D;
SetSpiSettings[6] = 0x5B;
SetSpiSettings[7] = 0x00; // 6.000.000 bps = 005B8D80 hex
SetSpiSettings[8] = 0xFF; // Idle Chip Select Value
SetSpiSettings[9] = 0xFF;
SetSpiSettings[10] = 0xEF; // Active Chip Select Value
SetSpiSettings[11] = 0xFF;
SetSpiSettings[12] = 0x00; // Chip Select to Data Delay (low byte)
SetSpiSettings[13] = 0x00; // Chip Select to Data Delay (high byte)
SetSpiSettings[14] = 0x00; // Last Data Byte to CS (low byte)
SetSpiSettings[15] = 0x00; // Last Data Byte to CS (high byte)
SetSpiSettings[16] = 0x00; // Delay Between Subsequent Data Bytes (low byte)
SetSpiSettings[17] = 0x00; // Delay Between Subsequent Data Bytes (high byte)
SetSpiSettings[18] = 0x03; // Bytes to Transfer per SPI Transaction (low byte)
SetSpiSettings[19] = 0x00; // Bytes to Transfer per SPI Transaction (high byte)
SetSpiSettings[20] = 0x00; // SPI mode 0
for(n=21;n<64;n++)
{
    SetSpiSettings[n] = 0x00; // Reserved
}

```

Figura 4 – Código do Transfer Settings.

4. TRANSFERÊNCIA

A transferência, assim como no item anterior, consiste no envio de um vetor de 64 posições. O significado de cada posição e o valor utilizado para a mesma é descrito na tabela abaixo:

Posição	Função	Valor
0	Comando de Transferência (padrão).	0x42
1	Bytes transmitidos por vez: 3.	0x03
2,3	(Reservado).	0x0000
4	Dado para envio: Primeiro envio define o Modo (padrão para envio, somente).	0x40
5-63	Dados para envio.	-

Tabela 3 – Descrição das configurações para transferência.

Onde os dados para envio consistem nos dados que serão transmitidos, propriamente dito. Após esses dados, as demais posições do vetor devem ser preenchidas com 0xFF.

```

/* TRANSFER SPI DATA */
TransferSpiData[0] = 0x42; // Transfer SPI Data Command Code
TransferSpiData[1] = 0x03; // Number of bytes to be transferred
TransferSpiData[2] = 0x00;
TransferSpiData[3] = 0x00; // Reserved
TransferSpiData[4] = 0x40; // SPI data to be sent
TransferSpiData[5] = 0x00;
TransferSpiData[6] = 0x00;
for(n=7;n<64;n++)
{
    TransferSpiData[n] = 0xFF;
}

```

Figura 5 – Código de configuração da transferência.

5. FUNÇÕES DA LIBUSB UTILIZADAS

Abaixo segue a lista das funções da biblioteca utilizadas, juntamente com o seu significado:

libusb_init() Inicializa a biblioteca.

libusb_set_debug() Registro de mensagens.

libusb_get_device_list() Obtém a lista de dispositivos conectados.

libusb_get_device_descriptor() Obtém o descritor do dispositivo.

libusb_open_device_with_vid_pid() Tenta manipular a MCP2210 utilizando o VID e o PID.

libusb_kernel_driver_active() Verifica se o Sistema operacional está com o controle do dispositivo.

libusb_detach_kernel_driver() Solicita a liberação do dispositivo pelo sistema operacional .

libusb_claim_interface() Reivindica a interface com a MCP2210.

libusb_bulk_transfer () Executa uma transferencia USB em massa.

libusb_free_device_list() Libera o dispositivo.

libusb_release_interface() Libera a interface.

libusb_close()Fecha a biblioteca libusb.

libusb_exit() Fechamento de contexto

6. LEITURA DA TEMPERATURA

O TC77, chip responsável pela medição da temperatura, consiste em um sensor de temperatura de *band-gap*, um conversor analógico digital de 13 bits *left aligned*, um oscilador de conversão interno de 30kHz e uma porta serial de entrada e saída. A precisão do sensor é de 0.0625°C.

A temperatura é representada por uma palavra de 13 bits (bits 15:3) em complemento de dois. Os últimos 3 bits (2:0) são 1.

Após a primeira conversão de temperatura o terceiro bit (bit 2) vai para 1, garantindo que o resultado é válido. Para conversão contínua, como no caso, esse passo não é necessário. A primeira conversão demora cerca de 300ms para se completar.

A configuração inicial do dispositivo segue o explicado no item 3. No primeiro dado a ser enviado no campo de Dado para Envio (posição 4 do vetor) envia-se 0. Desta forma, configura-se a conversão contínua. Então, retornará uma resposta (*ReceivedData*). O significado das primeiras posições pode ser verificado na tabela abaixo:

Posição	Função	Valor
0	Comando de Transferência: Echos	0x42
1	SPI Data Accept: Comando Completado com Sucesso	0x00
2	Quantidade de bytes por transferência (no caso, foi configurado 2)	0x02
3	Transferência Finalizada	0x10
4-63	Dados a receber	-

Tabela 4 – Descrição das configurações para a leitura da temperatura.

Ao se confirmar as primeiras 4 posições, tem-se uma leitura de temperatura a ser feita.

Para interpretar o valor recebido utiliza-se o *ReceivedData*[4] como parte alta e, devido ao valor ser em complemento de 2, desloca-se o *ReceivedData*[5] em três posições a direita, formando a temperatura. Para completar o processo, basta então converter para graus Celsius.

```

rslt = libusb_bulk_transfer(handle, DEV_ENDPOINT, data, 64, &byte_count, 0);
if(rslt == 0 && byte_count == 64)
{
    /**
    @brief libusb_bulk_transfer()
    | Receives device response.
    */
    rslt = libusb_bulk_transfer(handle, HOST_ENDPOINT, Response, 64, &byte_count, 0);
    if(rslt == 0 && byte_count == 64) // successfully received all bytes
    {
        if(ReceivedData[0]==0x42 && ReceivedData[1]==0x00 && ReceivedData[2] == 0x02 && ReceivedData[3]==0x10)
        {
            sign = ReceivedData[4] & 0x80;

            if(sign == 0)
                temp = (ReceivedData[4] << 8 | ReceivedData[5]) >> 3;
            else
                temp = (((ReceivedData[4] & 0x7F) << 8 | ReceivedData[5]) >> 3) - 4096;
            tempC = temp;
            tempC = tempC * 0.0625; // conversion to celsius

            printf("-> Temperature = %.2f Celsius\n", tempC);
        }
    }
}

```

Figura 6 – Código para leitura da temperatura.

REFERÊNCIAS

LIBUSB. A Cross-Platform User Library to Access USB Devices. Disponível em: <

<http://libusb.sourceforge.net/api-1.0/> > Acessado em Agosto/2014

MICROCHIP. USB-to-SPI Protocol Converter with GPIO (Master Mode). Disponível em: <

<http://ww1.microchip.com/downloads/en/DeviceDoc/22288A.pdf> > Acessado em

Agosto/2014.

MICROCHIP. Thermal Sensor with SPI Interface. Disponível em: <

<http://ww1.microchip.com/downloads/en/DeviceDoc/20092a.pdf> > Acessado em

Agosto/2014

WONG, Kerry D. MCP2210 Library – SPI Example Using TC77. Disponível em: <

<http://www.kerrywong.com/2012/10/10/mcp2210-library-spi-example-using-tc77/>

> Acessado em Agosto/2014.