CM-3G 周辺モジュール拡張技術文書
MS5607 センサ(温度、気圧)

(第 1 版)

目 次

1. はじめに

本書は、CM-3G 開発キットで MS5607 センサを使用するための補足マニュアルです。
CM-3G の拡張コネクタに接続可能なセンサを使用するための手順を記述します。

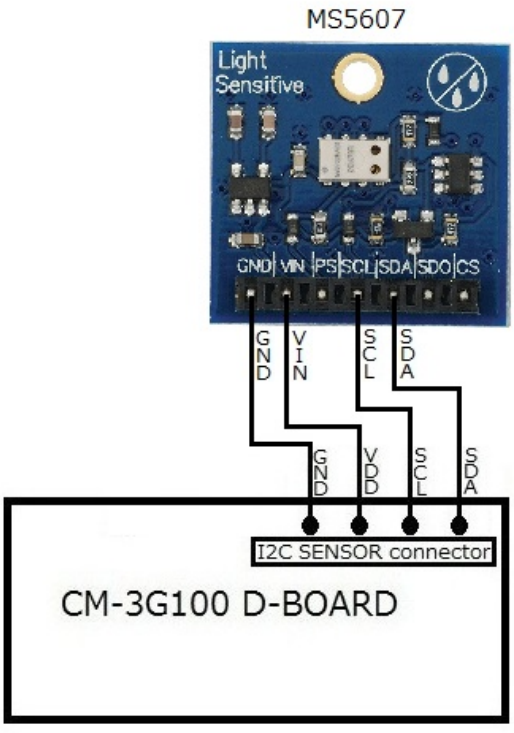
2. MS5607 について

MS5607 は、I2C で接続可能な温度、気圧センサです。

3. 接続図

■ **MS5607 と CM-3G100 D-BOARD の接続信号表**

MS5607	CM-3G100 D-BOARD SENSOR コネクタ
GND	GND
VIN	VDD
SCL	SCL
SDA	SDA



4. アプリケーション・ソース

センサ情報を表示するアプリケーション・ソースは以下の通りです。

```
/*
 *  ms5607.c
 *
 *  Copyright (C) 2012 Computex Co.,Ltd.
 *
 *  Sample to test MS5607 using CM-3G CHECK BOARD
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <linux/i2c-dev.h>

#define I2C_DEVICE          "/dev/i2c-3"
#define DEV_ADDRESS                 0x76

#define CMD_RESET           0x1E    // ADC reset command
#define CMD_ADC_READ        0x00    // ADC read command
#define CMD_ADC_CONV        0x40    // ADC conversion command
#define CMD_ADC_D1          0x00    // ADC D1 conversion
#define CMD_ADC_D2          0x10    // ADC D2 conversion
#define CMD_ADC_256                 0x00    // ADC OSR=256
#define CMD_ADC_512                 0x02    // ADC OSR=512
#define CMD_ADC_1024        0x04    // ADC OSR=1024
#define CMD_ADC_2048        0x06    // ADC OSR=2048
#define CMD_ADC_4096        0x08    // ADC OSR=4096
#define CMD_PROM_RD                 0xA0    // Prom read command

int file;
void usage(char *app_name)
{
        printf("%s - Displays Temperature and Pressure values using MS5607 sensor¥n", app_name);
        printf("Usage: %s -d [device path]¥n", app_name);
        printf("¥t¥t-d - i2c device path (Default: "I2C_DEVICE")¥n");
        printf("Eg: %s¥n", app_name);
        exit(0);
}

int i2c_send(unsigned char cmd)
{
        unsigned char buf[1];
        buf[0] = cmd;
```

```
        if (write(file, buf, 1) != 1) {
                printf("Failed to write to i2c bus. Check if MS5607 is connected¥n");
                return 1;
        }
        return 0;
}

int i2c_recv(unsigned char *buf, int count)
{
        if (read(file, buf, 2) != 2) {
                printf("Failed to read from to i2c bus. Check if MS5607 is connected¥n");
                return 1;
        }
        return 0;
}

/*************************************************
        Performs ADC conversion
*************************************************/
int read_adc(char cmd, unsigned int *val)
{
        unsigned char buf[3];
        /* send conversion command */
        if(i2c_send(CMD_ADC_CONV | cmd)){
                return 1;
        }
        /* wait for conversion */
        switch (cmd & 0x0f)
        {
        case CMD_ADC_256:
                usleep(900000);
                break;
        case CMD_ADC_512:
                usleep(3000);
                break;
        case CMD_ADC_1024:
                usleep(4000);
                break;
        case CMD_ADC_2048:
                usleep(6000);
                break;
        case CMD_ADC_4096:
                usleep(10000);
                break;
        }

        /* send read comand and read */
        if(i2c_send(CMD_ADC_READ)){
                return 1;
        }
        if(i2c_recv(buf, 3)){
```

```
                return 1;
        }
        *val = (buf[0]<<16) | (buf[1]<<8) | buf[2];
        return 0;
}

/*********************************************************
        Reads calibration coefficients
*********************************************************/
int get_coeff(unsigned int coeff_num, double *val)
{
        unsigned char buf[2];
        /* send PROM READ command and read */
        if(i2c_send(CMD_PROM_RD + coeff_num * 2))
                return 1;
        if(i2c_recv(buf, 2)){
                return 1;
        }
        *val = (buf[0]<<8) | buf[1];
        return 0;
}

/*********************************************************************
 * Main
 *********************************************************************/
int main(int argc, char* argv[])
{
        unsigned char buf[10];
        int d;
        int i;
        char shortoptions[] = "d:h";
        char input_device[16];
        float hres, hres2, lres;
        double coeff[8];
        unsigned int D1, D2;
        double dT, T, P, OFF, SENS;
        input_device[0] = 0;
        for (;;) {
                d = getopt_long(argc, argv, shortoptions, (void *)NULL, &index);
                if (-1 == d)
                {
                        break;
                }
                switch (d) {
                case 'd':
                        strcpy(input_device, optarg);
                        break;
                default:
                        usage(argv[0]);
                }
        }
```

```
        if(input_device[0] == 0)
                strcpy(input_device, I2C_DEVICE);

        file = open(input_device, O_RDWR);
        if(file < 0) {
                printf("Error: "I2C_DEVICE " open failed¥n");
                exit(1);
        }

        if(ioctl(file, I2C_SLAVE, DEV_ADDRESS) < 0) {
                printf("Error: %s setting device address to 0x%xfailed¥n", input_device,
DEV_ADDRESS);
                goto _out;
        }

        /* send reset sequence */
        if (i2c_send(CMD_RESET)){
                goto _out;
        }

        /* wait for the reset sequence timing */
        usleep(3000);

        /* read coefficients */
        for(i=1; i<7; i++)
                if(get_coeff(i, &coeff[i]))
                        goto _out;

        printf("Temperature     Pressure¥n");
        while(1)
        {
                if(read_adc(CMD_ADC_D1 | CMD_ADC_4096, &D1)
                        || read_adc(CMD_ADC_D2 | CMD_ADC_4096, &D2))
                        goto _out;
                dT = D2 - coeff[5] * 256;
                OFF = coeff[2] * 131072 + ((coeff[4] * dT) / 64);
                SENS = coeff[1] * 65536 + (coeff[3] * dT) / 128;
                T = (2000 + (dT * coeff[6]) / 8388608) / 100;
                P = (((D1 * SENS) / 2097152 - OFF) / 32768) /100;
                printf("%0.3f° ?C     %0.3f mbar¥n", T, P);
                sleep(1);
        }

_out:
        close(file);
        return 0;
}
```

Makefile は以下の通りです。

```
# Makefile

BUILDROOT_DIR:=../../../../../../
OUTPUT_DIR:=output
CROSS_COMPILE:=$(BUILDROOT_DIR)/$(OUTPUT_DIR)/host/usr/bin/arm-linux-
#APP_DEBUG:=-ggdb

CC = $(CROSS_COMPILE)gcc $(APP_DEBUG) -I$(TSLIB_DIR) -I$(KERNEL_HEADERS) -O0 -lm
APP=ms5607
LIB= -ldl -lpthread

SRCS=ms5607.c

all: $(SRCS)
	$(CC) -o $(APP) $(SRCS) $(LIB)

clean:
	rm -f *.o *~ $(APP)
```

5. アプリケーションのコンパイル方法

buildroot の以下のディレクトリでアプリケーションをコンパイルする手順を説明します。

${buildroot}/board/ckb/cm-3g/samples/service-patch/MS5607

まず、アプリケーションをコンパイルする作業ディレクトリを作成します。

```
LINUXPC$ mkdir -p board/ckb/cm-3g/samples/service-patch/MS5607
```

アプリケーション・ソース(ms5607.c)と Makefile を作業ディレクトリにコピーし、カレント・ディレクトリを移動します。

```
LINUXPC$ cd board/ckb/cm-3g/samples/service-patch/MS5607
```

make を実行し、ソース・ファイルをコンパイルします。

```
LINUXPC$ make
```

make が成功すると、実行ファイル(ms5607)が作成されますので microSD カードの/usr/bin に実行ファイルをコピーします。

```
LINUXPC$ sudo cp ms5607 /media/rootfs/usr/bin
```

6. アプリケーションの実行

以下のようにアプリケーションを実行すると、センサの値を取得して表示します。

```
# ms5607
```

アプリケーションの実行結果は以下のようになり、温度、気圧の情報を一定時間ごとに表示します。
アプリケーションを終了する場合は、**[Ctrl]** + **[C]**を入力してください。

```
# ms5607

Temperature     Pressure
23.712℃         1020.483 mbar
23.712℃         1020.483 mbar
23.712℃         1020.483 mbar
23.721℃         1020.501 mbar
```

変更履歴

日付	版	内容
2016-1	1	初版

株式会社コンピューテックス

本 社
〒605-0846 京都市東山区五条橋東 4-432-13 對嵐坊ビル
TEL:075-551-0528(代) FAX:075-551-2585
東京営業所
TEL:03-5753-9911(代) FAX:03-5753-9917
テクニカルセンタ
TEL:075-551-0373 FAX:075-551-2585

CM-3G 周辺モジュール拡張技術文書
MS5607 センサ(温度、気圧)
2016 年 1 月 第 1 版
CX534(A)1601