實驗九 LCD 及 DS18B20

0510532 楊上萱

1. Lab objectives 實驗目的

- 瞭解LCD的使用

- 瞭解DS18B20的使用

2. Steps 實驗步驟

9-1 跑馬燈

- ➔ Init需要使用到的GPIO。
- ➔ 瞭解LCD的Protocol並且實作WriteToLCD
- ➔ 瞭解LCD初始化過程，並且完成初始化
- ➔ 設定Systick_Handler為0.3秒。

9-2 客製化圖形顯示與按鈕切換

實作兩種模式，並且透過板子上的按鈕（PC13）在模式之間切換（放開才反應在LCD上）
模式一：自製一個兩格的圖像（使用CGR），然後讓他跑2-1的實驗（由左到右），並且一樣每0.3秒一個動畫（請使用Systick_Handler）。
模式二：讓LCD可以顯示宣告好的字串（助教會改DEMO字串），每0.3秒顯示一個字元（請使用Systick_Handler）。

- ➔ 實作WriteStrToLCD
- ➔ 實作CreateFont
- ➔ 可以使用<string.h>

9-3 跑馬燈與溫度計

承接2.2並且將第二種模式改成顯示當前的溫度，並且讓溫度計擁有0.125的

精度。在第一種模式的情況下，動畫仍然以0.3秒的週期往右移動，在第二種模式，則請以1秒為週期刷新溫度讀值，讀值不需要一個一個慢慢顯示，一次顯示完即可。

3.　Results and analysis 實驗結果與分析

9-1 跑馬燈

首先先 initialize LCD 上的 pin 腳，並加上背光(最後兩個 pin)。

然後初始化 LCD，設定雙排顯示、每次寫資料計數器遞增而畫面不動、隱藏游標、清除畫面、DD RAM 位址設為左上角第一個字元位置等。

下指令給 LCD，先讓 RS 設為 0，RW 設為 0，然後寫指令到 D[7:0]，最後將 EN 設為 1，等待 10ms，EN 再設為 0，再等待 10ms。

9-2 客製化圖形顯示與按鈕切換

Mode1:

先將 RS 設為 1，RW 設為 0，自己設計完字型並設定好 CG RAM 後，把點陣字的每一橫排依序寫入 D[7:0]，EN=1，等待 10ms，EN=0，再等待 10ms，共八排所以做八次，(前面三位 don’t care，只要設計後五個 bits)。

Mode2:

每次往後顯示一個字元，直到遇到字串結尾的 0，再清除畫面重來一次。

9-3 跑馬燈與溫度計

One wire 照著 protocol 去實作，適時切換 input 和 output 模式，寫入或是讀取 byte 時，每一個 bit 中間都要有適當的 delay，讀取時要再 15us 內將值讀出來。

4.　Conclusions and ideas 心得討論與應用聯想

LCD 的實驗很好玩，可以自己設計字型，而且如果有照著老師的投影片做就蠻順利的，溫度計的部分超級複雜，要實作 protocol，還要讓 delay 的時間恰恰好，而且 debug 很困難，所以花了許多時間。

5.　Code

9-1

```c
/*** main9_1.c ***/

#include "libtmd.h"

const GPIO_TypeDef *LCD_DATA_PORT[8] = {
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB
};
const GPIO_TypeDef *LCD_RS_PORT = GPIOA;
const GPIO_TypeDef *LCD_RW_PORT = GPIOA;
const GPIO_TypeDef *LCD_EN_PORT = GPIOA;

const uint16_t LCD_DATA_PIN[8] = {
    GPIO_PIN_0,
    GPIO_PIN_1,
    GPIO_PIN_2,
    GPIO_PIN_3,
    GPIO_PIN_4,
    GPIO_PIN_5,
    GPIO_PIN_6,
    GPIO_PIN_7
};
const uint16_t LCD_RS_PIN = GPIO_PIN_5;
const uint16_t LCD_RW_PIN = GPIO_PIN_6;
const uint16_t LCD_EN_PIN = GPIO_PIN_7;

void SysTick_UserConfig(float);
void SysTick_Handler();
void init();
void init_lcd();
void write_to_lcd(int, int);

int counter = 0;

int main() {
    fpu_enable();
    init();
    SysTick_UserConfig(0.3);
    while (1);
    return 0;
}

void SysTick_UserConfig(float n) {
    SysTick->CTRL |= 0x00000004;
    SysTick->LOAD = (uint32_t) (n * 4000000.0);
    SysTick->VAL = 0;
    SysTick->CTRL |= 0x00000003;
}

```

```
void SysTick_Handler() {
    counter = counter + 1;
    if (counter == 18) {
        write_to_lcd(0x80 + 0x0F, 1);
        write_to_lcd(0x20, 0); // print ' '
        write_to_lcd(0x20, 0); // print ' '
        write_to_lcd(0x80 + 0x41, 1);
    }
    if (counter == 34) {
        write_to_lcd(0x80 + 0x4F, 1);
        write_to_lcd(0x20, 0); // print ' '
        write_to_lcd(0x20, 0); // print ' '
        write_to_lcd(0x80 + 0x1, 1);
        counter = 2;
    }
    write_to_lcd(0x10, 1); // shift cursor
    write_to_lcd(0x10, 1); // shift cursor
    write_to_lcd(0x20, 0); // print ' '
    write_to_lcd(0x31, 0); // print '1'
    write_to_lcd(0x32, 0); // print '2'
    if (counter == 17) {
        write_to_lcd(0x80 + 0x40, 1);
        write_to_lcd(0x32, 0); // print '2'
        write_to_lcd(0x80 + 0x0F, 1);
    }
    if (counter == 33) {
        write_to_lcd(0x80 + 0x0, 1);
        write_to_lcd(0x32, 0); // print '2'
        write_to_lcd(0x80 + 0x4F, 1);
    }
}

void init() {
    TMD_GPIO_Init();
    init_lcd();
}

void init_lcd() {
    write_to_lcd(0x38, 1); // function setting
    write_to_lcd(0x06, 1); // entry mode
    write_to_lcd(0x0C, 1); // display on
    write_to_lcd(0x01, 1); // clear screen
    write_to_lcd(0x80, 1); // move to top left
}

void write_to_lcd(int input, int is_cmd) {
    if (is_cmd)
        TMD_GPIO_SetPinLow(LCD_RS_PORT, LCD_RS_PIN);
```

```c
        else
            TMD_GPIO_SetPinHigh(LCD_RS_PORT, LCD_RS_PIN);

        TMD_GPIO_SetPinLow(LCD_RW_PORT, LCD_RW_PIN);

        for (int i = 0; i < 8; ++i) {
            if (input & (1 << i))
                TMD_GPIO_SetPinHigh(LCD_DATA_PORT[i], LCD_DATA_PIN[i]);
            else
                TMD_GPIO_SetPinLow(LCD_DATA_PORT[i],  LCD_DATA_PIN[i]);
        }

        TMD_GPIO_SetPinHigh(LCD_EN_PORT, LCD_EN_PIN);
        delay_ms(10);
        TMD_GPIO_SetPinLow(LCD_EN_PORT, LCD_EN_PIN);
        delay_ms(10);
    }

```

9-2

```c
/*** main9_2.c ***/

#include "libtmd.h"

const GPIO_TypeDef *LCD_DATA_PORT[8] = {
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB
};
const GPIO_TypeDef *LCD_RS_PORT = GPIOA;
const GPIO_TypeDef *LCD_RW_PORT = GPIOA;
const GPIO_TypeDef *LCD_EN_PORT = GPIOA;

const uint16_t LCD_DATA_PIN[8] = {
    GPIO_PIN_0,
    GPIO_PIN_1,
    GPIO_PIN_2,
    GPIO_PIN_3,
    GPIO_PIN_4,
    GPIO_PIN_5,
    GPIO_PIN_6,
    GPIO_PIN_7
};
const uint16_t LCD_RS_PIN = GPIO_PIN_5;
const uint16_t LCD_RW_PIN = GPIO_PIN_6;
const uint16_t LCD_EN_PIN = GPIO_PIN_7;

const int map_one[8] = {
    0x1C,
    0x4,
    0x4,
    0x4,
    0x4,
    0x4,
    0x4,
    0x1F
};

const int map_two[8] = {
    0x1F,
    0x1,
    0x1,
    0x1F,
    0x10,
    0x10,
    0x10,
    0x1F
};

const char *test_string = "Hey! claclalc";

void SysTick_UserConfig(float);
void SysTick_Handler();
void init();
void init_lcd();
void write_to_lcd(int, int);
void create_font(int, const int *);
void write_str_to_lcd(char *);

int counter = 0, mode = 0, position = 0;

int main() {
    int prev_btn = 1, curr_btn = 1;
    fpu_enable();
    init();
    SysTick_UserConfig(0.3);
    while (1) {
        if (!prev_btn && curr_btn) {
            mode ^= 1;
            position = 0;
            counter = 0;
            SysTick->CTRL &= 0xFFFFFFFE;
            init();
            SysTick->CTRL |= 0x00000001;
        }
        prev_btn = curr_btn;
        curr_btn = GPIOC->IDR & GPIO_PIN_13;
    }
    return 0;
}

void SysTick_UserConfig(float n) {
    SysTick->CTRL |= 0x00000004;
    SysTick->LOAD = (uint32_t) (n * 4000000.0);
    SysTick->VAL = 0;
    SysTick->CTRL |= 0x00000003;
}

void SysTick_Handler() {
    if (mode == 0) {
        counter = counter + 1;
        if (counter == 18) {
            write_to_lcd(0x80 + 0x0F, 1);
            write_to_lcd(0x20, 0); // print ' '
            write_to_lcd(0x20, 0); // print ' '
            write_to_lcd(0x80 + 0x41, 1);
        }
        if (counter == 34) {
            write_to_lcd(0x80 + 0x4F, 1);
            write_to_lcd(0x20, 0); // print ' '
            write_to_lcd(0x20, 0); // print ' '
            write_to_lcd(0x80 + 0x1, 1);
            counter = 2;
        }
        write_to_lcd(0x10, 1); // shift cursor
        write_to_lcd(0x10, 1); // shift cursor
        write_to_lcd(0x20, 0); // print ' '
        write_to_lcd(0x00, 0); // print '4'
        write_to_lcd(0x01, 0); // print '5'
        if (counter == 17) {
            write_to_lcd(0x80 + 0x40, 1);
            write_to_lcd(0x01, 0); // print '5'
            write_to_lcd(0x80 + 0x0F, 1);
        }
        if (counter == 33) {
            write_to_lcd(0x80 + 0x0, 1);
            write_to_lcd(0x01, 0); // print '5'
            write_to_lcd(0x80 + 0x4F, 1);
        }
    }
    else
        write_str_to_lcd(test_string);
}

void init() {
    TMD_GPIO_Init();
    init_lcd();
    create_font(0, map_one);
    create_font(8, map_two);
    write_to_lcd(0x80, 1); // move to top left
}

void init_lcd() {
    write_to_lcd(0x38, 1); // function setting
    write_to_lcd(0x06, 1); // entry mode
    write_to_lcd(0x0C, 1); // display on
    write_to_lcd(0x01, 1); // clear screen
    write_to_lcd(0x80, 1); // move to top left
}

void write_to_lcd(int input, int is_cmd) {
    if (is_cmd)
        TMD_GPIO_SetPinLow(LCD_RS_PORT, LCD_RS_PIN);
    else
        TMD_GPIO_SetPinHigh(LCD_RS_PORT, LCD_RS_PIN);

    TMD_GPIO_SetPinLow(LCD_RW_PORT, LCD_RW_PIN);

    for (int i = 0; i < 8; ++i) {
        if (input & (1 << i))
            TMD_GPIO_SetPinHigh(LCD_DATA_PORT[i], LCD_DATA_PIN[i]);
        else
            TMD_GPIO_SetPinLow(LCD_DATA_PORT[i],  LCD_DATA_PIN[i]);
    }

    TMD_GPIO_SetPinHigh(LCD_EN_PORT, LCD_EN_PIN);
    delay_ms(10);
    TMD_GPIO_SetPinLow(LCD_EN_PORT, LCD_EN_PIN);
    delay_ms(10);
}

void create_font(int location, const int *font_array) {
    write_to_lcd(location & 0x3F | 0x40, 1);
    for (int i = 0; i < 8; ++i)
```

```
            write_to_lcd(font_array[i] & 0x1F, 0);
    }

    void write_str_to_lcd(char *str) {
        if (str[position] == 0) {
            position = 0;
            counter = 0;
            SysTick->CTRL &= 0xFFFFFFFE;
            init();
            SysTick->CTRL |= 0x00000001;
        }
        write_to_lcd(str[position], 0);
        position++;
    }

```

9-3

main.c

```

#include "libtmd.h"
#include "ds18b20.h"

const GPIO_TypeDef *LCD_DATA_PORT[8] = {
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB
};
const GPIO_TypeDef *LCD_RS_PORT = GPIOA;
const GPIO_TypeDef *LCD_RW_PORT = GPIOA;
const GPIO_TypeDef *LCD_EN_PORT = GPIOA;

const uint16_t LCD_DATA_PIN[8] = {
    GPIO_PIN_0,
    GPIO_PIN_1,
    GPIO_PIN_2,
    GPIO_PIN_3,
    GPIO_PIN_4,
    GPIO_PIN_5,
    GPIO_PIN_6,
    GPIO_PIN_7
};
const uint16_t LCD_RS_PIN = GPIO_PIN_5;
const uint16_t LCD_RW_PIN = GPIO_PIN_6;
const uint16_t LCD_EN_PIN = GPIO_PIN_7;

const int map_one[8] = {
    0x1C,
    0x4,
    0x4,
    0x4,
    0x4,
    0x4,
    0x4,
    0x1F
};

const int map_two[8] = {
    0x1F,
    0x1,
    0x1,
    0x1F,
    0x10,
    0x10,
    0x10,
    0x1F
};
// 9-bits, 10-bits, 11-bits, 12-bits
// 0.5, 0.25, 0.125, 0.0625
const char *test_string = "Hey! Blabla...";
const unsigned resolution = 11;

void SysTick_UserConfig(float);
void SysTick_Handler();
void init();
void init_lcd();
void write_to_lcd(int, int);
void create_font(int, const int *);
void write_str_to_lcd(char *);
void write_int_to_lcd(int16_t);

int counter = 0, mode = 0, position = 0;
int16_t now_temp;

int main() {
    int prev_btn = 1, curr_btn = 1;
    fpu_enable();
    init();
    set_resolution(resolution);
    SysTick_UserConfig(0.3);
    while (1) {
```

```c
        if (!prev_btn && curr_btn) {
            mode ^= 1;
            position = 0;
            counter = 0;
            SysTick->CTRL &= 0xFFFFFFF8;
            init();
            if (mode == 0)
                SysTick_UserConfig(0.3);
            else
                SysTick_UserConfig(1);
        }
        prev_btn = curr_btn;
        curr_btn = GPIOC->IDR & GPIO_PIN_13;
    }
    return 0;
}

void SysTick_UserConfig(float n) {
    SysTick->CTRL |= 0x00000004;
    SysTick->LOAD = (uint32_t) (n * 4000000.0);
    SysTick->VAL = 0;
    SysTick->CTRL |= 0x00000003;
}

void SysTick_Handler() {
    if (mode == 0) {
        counter = counter + 1;
        if (counter == 18) {
            write_to_lcd(0x80 + 0x0F, 1);
            write_to_lcd(0x20, 0); // print ' '
            write_to_lcd(0x20, 0); // print ' '
            write_to_lcd(0x80 + 0x41, 1);
        }
        if (counter == 34) {
            write_to_lcd(0x80 + 0x4F, 1);
            write_to_lcd(0x20, 0); // print ' '
            write_to_lcd(0x20, 0); // print ' '
            write_to_lcd(0x80 + 0x1, 1);
            counter = 2;
        }
        write_to_lcd(0x10, 1); // shift cursor
        write_to_lcd(0x10, 1); // shift cursor
        write_to_lcd(0x20, 0); // print ' '
        write_to_lcd(0x00, 0); // print '4'
        write_to_lcd(0x01, 0); // print '5'
        if (counter == 17) {
            write_to_lcd(0x80 + 0x40, 1);
            write_to_lcd(0x01, 0); // print '5'
```

```
                write_to_lcd(0x80 + 0x0F, 1);
            }
            if (counter == 33) {
                write_to_lcd(0x80 + 0x0, 1);
                write_to_lcd(0x01, 0); // print '5'
                write_to_lcd(0x80 + 0x4F, 1);
            }
        }
        else {
            SysTick->CTRL &= 0xFFFFFFFE;
            now_temp = get_temperature();
            int16_t aaa = 0x100000;
            write_int_to_lcd(aaa);
            SysTick->CTRL |= 0x00000001;
        }
    }

    void init() {
        TMD_GPIO_Init();
        init_lcd();
        create_font(0, map_one);
        create_font(8, map_two);
        write_to_lcd(0x80, 1); // move to top left
    }

    void init_lcd() {
        write_to_lcd(0x38, 1); // function setting
        write_to_lcd(0x06, 1); // entry mode
        write_to_lcd(0x0C, 1); // display on
        write_to_lcd(0x01, 1); // clear screen
        write_to_lcd(0x80, 1); // move to top left
    }

    void write_to_lcd(int input, int is_cmd) {
        if (is_cmd)
            TMD_GPIO_SetPinLow(LCD_RS_PORT, LCD_RS_PIN);
        else
            TMD_GPIO_SetPinHigh(LCD_RS_PORT, LCD_RS_PIN);

        TMD_GPIO_SetPinLow(LCD_RW_PORT, LCD_RW_PIN);

        for (int i = 0; i < 8; ++i) {
            if (input & (1 << i))
                TMD_GPIO_SetPinHigh(LCD_DATA_PORT[i],
 LCD_DATA_PIN[i]);
            else
                TMD_GPIO_SetPinLow(LCD_DATA_PORT[i],  LCD_DATA_PIN[i]);
        }
```

```c

    TMD_GPIO_SetPinHigh(LCD_EN_PORT, LCD_EN_PIN);
    delay_ms(10);
    TMD_GPIO_SetPinLow(LCD_EN_PORT, LCD_EN_PIN);
    delay_ms(10);
}

void create_font(int location, const int *font_array) {
    write_to_lcd((location & 0x3F) | 0x40, 1);
    for (int i = 0; i < 8; ++i)
        write_to_lcd(font_array[i] & 0x1F, 0);
}

void write_str_to_lcd(char *str) {
    if (str[position] == 0) {
        position = 0;
        counter = 0;
        SysTick->CTRL &= 0xFFFFFFFE;
        init();
        SysTick->CTRL |= 0x00000001;
    }
    write_to_lcd(str[position], 0);
    position++;
}

void write_int_to_lcd(int16_t in) {
    switch (resolution) {
        case 12:
            in &= 0xFFFF;
            break;
        case 11:
            in &= 0xFFFE;
            break;
        case 10:
            in &= 0xFFFC;
            break;
        case 9:
            in &= 0xFFF8;
            break;
        default:
            break;
    }
    int16_t in1 = in >> 4;
    int16_t in2 = ((in & 0x0001) * 0.0625 + (in & 0x0002) * 0.125 + \
                   (in & 0x0004) * 0.25 + (in & 0x0008) * 0.5) * 1000;
    init();
```

```
        write_to_lcd(0x30, 0);
        write_to_lcd(0x30, 0);
        write_to_lcd(0x2E, 0);
        write_to_lcd(0x30, 0);
        write_to_lcd(0x30, 0);
        write_to_lcd(0x30, 0);
        write_to_lcd(0x30, 0);
        write_to_lcd(0x10, 1);
        write_to_lcd(0x30 + in2 % 10, 0);
        in2 /= 10;
        write_to_lcd(0x10, 1);
        write_to_lcd(0x30 + in2 % 10, 0);
        in2 /= 10;
        write_to_lcd(0x10, 1);
        write_to_lcd(0x30 + in2 % 10, 0);
        in2 /= 10;
        write_to_lcd(0x10, 1);
        write_to_lcd(0x30 + in2 % 10, 0);
        write_to_lcd(0x10, 1);
        write_to_lcd(0x10, 1);
        write_to_lcd(0x30 + in1 % 10, 0);
        in1 /= 10;
        write_to_lcd(0x10, 1);
        write_to_lcd(0x30 + in1 % 10, 0);
        //delay_ms(10);
    }

```

onewire.h

```
#include "libtmd.h"

void OneWire_Reset()
{
    ONEWIRE_INPUT();
    GPIOA->BRR = GPIO_PIN_8; // high -> low
    ONEWIRE_OUTPUT();
    //ONEWIRE_DELAY(480);
    delay_us(480);
    ONEWIRE_INPUT();
    //ONEWIRE_DELAY(70);
    delay_us(70);
    //ONEWIRE_DELAY(410);
    delay_us(410);
}

```

```c
void OneWire_WriteBit(uint8_t bit)
{
    //ONEWIRE_DELAY(4);
    delay_us(4);
    ONEWIRE_INPUT();
    if (bit) // 1
    {
        // Set line low
        GPIOA->BRR = GPIO_PIN_8;
        ONEWIRE_OUTPUT();
        // Bit high
        ONEWIRE_INPUT();
    }
    else // 0
    {
        // Set line low
        GPIOA->BRR = GPIO_PIN_8;
        ONEWIRE_OUTPUT();
        //ONEWIRE_DELAY(70);
        delay_us(70);
    }
    ONEWIRE_INPUT();
}

void OneWire_WriteByte(int data)
{
    int mask = 0x80;
    for (int i = 0; i < 8; i++)
    {
        OneWire_WriteBit(mask & data);
        mask = mask >> 1;
    }
}

uint8_t OneWire_ReadBit()
{
    //ONEWIRE_DELAY(4);
    delay_us(4);
    uint8_t data = 0;
    ONEWIRE_INPUT();
    GPIOA->BRR = GPIO_PIN_8; // high -> low
    ONEWIRE_OUTPUT();
    //ONEWIRE_DELAY(1);
    delay_us(1);
    ONEWIRE_INPUT();
    data = GPIOA->IDR & 0x1;
    return data;
}
```

```c

int OneWire_ReadByte()
{
    int mask = 1, ans = 0;
    for (int i = 0; i < 8; i++)
    {
        ans = ans | (mask & OneWire_ReadBit());
        mask = mask << 1;
    }
}

void ONEWIRE_INPUT()
{
    GPIOA->MODER   &= 0b11111111111111001111111111111111;
    GPIOA->PUPDR   &= 0b11111111111111001111111111111111;
    GPIOA->PUPDR   |= 0b00000000000000010000000000000000;
    GPIOA->OSPEEDR &= 0b11111111111111001111111111111111;
    GPIOA->OSPEEDR |= 0b00000000000000010000000000000000;
    GPIOA->OTYPER  |= 0b00000000000000000000000100000000;
}

void ONEWIRE_OUTPUT()
{
    GPIOA->MODER   &= 0b11111111111111001111111111111111;
    GPIOA->MODER   |= 0b00000000000000010000000000000000;
    GPIOA->PUPDR   &= 0b11111111111111001111111111111111;
    GPIOA->PUPDR   |= 0b00000000000000010000000000000000;
    GPIOA->OSPEEDR &= 0b11111111111111001111111111111111;
    GPIOA->OSPEEDR |= 0b00000000000000010000000000000000;
    GPIOA->OTYPER  |= 0b00000000000000000000000100000000;
}

void ONEWIRE_DELAY(unsigned microseconds)
{
    usleep(microseconds);
  }

```

ds18b20.h

```c
#include "libtmd.h"
#include "onewire.h"

void set_resolution(unsigned resolution) {
    // Initialization
    OneWire_Reset();
    // ROM Command: Skip ROM [CCh]
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    // DS18B20 Function Command: Write Scratchpad [4Eh]
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    // Data Exchange: TH Register [40h]
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    // Data Exchange: TL Register [08h]
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    // Data Exchange: Configuration Register
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    switch (resolution) {
        case 9:
            OneWire_WriteBit(0);
            OneWire_WriteBit(0);
            break;
        case 10:
            OneWire_WriteBit(1);
            OneWire_WriteBit(0);
            break;
        case 11:
            OneWire_WriteBit(0);
            OneWire_WriteBit(1);
            break;
        case 12:
        default:
            OneWire_WriteBit(1);
            OneWire_WriteBit(1);
            break;
    }
    OneWire_WriteBit(0);

    // Initialization
    OneWire_Reset();
    // ROM Command: Skip ROM [CCh]
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    // DS18B20 Function Command: Copy Scratchpad [48h]
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
}

int16_t get_temperature() {
    // Initialization
    OneWire_Reset();
    // ROM Command: Skip ROM [CCh]
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    // DS18B20 Function Command: Convert T [44h]
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);

    // Wait!!!!!!!!!
    //usleep(750000);
    delay_us(100000);

    // Initialization
    OneWire_Reset();
    // ROM Command: Skip ROM [CCh]
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    // DS18B20 Function Command: Read Scratchpad [BEh]
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    // Data Exchange: Temperature LSB Register
    int16_t r = 0;
    r |= OneWire_ReadBit() << 0;
    r |= OneWire_ReadBit() << 1;
    r |= OneWire_ReadBit() << 2;
    r |= OneWire_ReadBit() << 3;
    r |= OneWire_ReadBit() << 4;
    r |= OneWire_ReadBit() << 5;
    r |= OneWire_ReadBit() << 6;
    r |= OneWire_ReadBit() << 7;
    // Data Exchange: Temperature MSB Register
    r |= OneWire_ReadBit() << 8;
    r |= OneWire_ReadBit() << 9;
    r |= OneWire_ReadBit() << 10;
    r |= OneWire_ReadBit() << 11;
    r |= OneWire_ReadBit() << 12;
    r |= OneWire_ReadBit() << 13;
    r |= OneWire_ReadBit() << 14;
    r |= OneWire_ReadBit() << 15;

    // Initialization
    OneWire_Reset();

    return r;
}

```