

Grove 4-digit display guide

Contents

Introduction	2
4-digit display basics.....	2
GPIO setup	3
C Code.....	4
Start signal.....	4
Stop signal.....	5
Data write.....	6
Controlling the 4-digit display.....	7
Troubleshooting	9

Introduction

This guide will provide a walkthrough for using the Grove 4-digit display¹ with the BeagleBone Green. The Grove system uses a standardized 4 pin connector to simplify the use of some hardware². This guide will require that you already have some knowledge and libraries to work with GPIO using C (Go through Brian Fraser's GPIO guide).

The 4-digit display does not follow any standard protocols so we will be bit-banging³ through the Grove UART port to control the display. We will use two pins, one for the clock and the other for the data.

4-digit display basics

Before we begin, we will take a look at the 4-digit display and figure out what we need to do to use it. The display that the Grove module uses is the TM1637. The guide will refer to the TM1637 [Datasheet](#) for information. It is recommended that you take some time now to look over the datasheet.

For this guide, we will use the auto increment mode as shown on page 4 of the datasheet. These are the basic steps:

1. Send start signal
2. Send command for automatic address incrementing
3. Send stop signal
4. Send start signal
5. Send the starting address
6. Send the data for the digits
7. Send stop signal
8. Send start signal
9. Send display data
10. Send stop signal

Note

Data should only be sent when the clock signal is low⁴. The pulse width of the clock should also be at least 400ns⁵.

¹ http://wiki.seeed.cc/Grove-4-Digit_Display/

² <http://www.seeedstudio.com/blog/2016/03/09/tutorial-intro-to-grove-connectors-for-arduinoraspberry-pi-projects/>

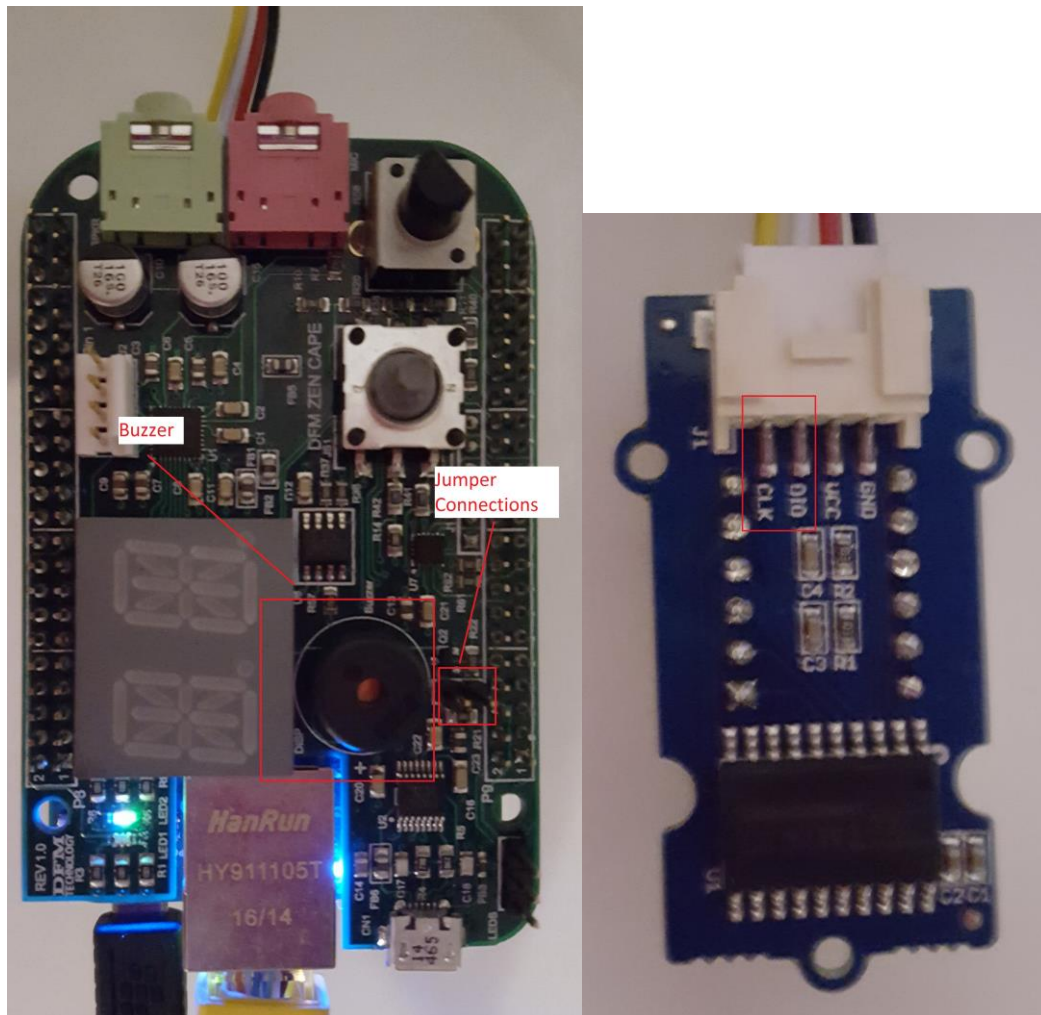
³ https://en.wikipedia.org/wiki/Bit_banging

⁴ Interface Interpretation - p. 2 TM1637 datasheet

⁵ Timing character – p.10 TM1637 datasheet

GPIO setup

On the Grove 4-digit display, we can see that we will need to control the CLK and DIO lines. Since we are using the Grove UART port, these map to RX and TX pins respectively. The Grove UART connector on the BeagleBone is connected to UART2. Looking at the BeagleBone GPIO reference⁶, RX maps to P9_22 (GPIO 2) and TX goes to P9_21 (GPIO 3). Make sure that the pins are not set for PWM and remove the jumper from the connection by the buzzer on the Zen cape (plastic piece below the buzzer).



⁶ http://wiki.seeed.cc/BeagleBone_Green/

To get started, export GPIO pins 2 and 3 for output either through Linux or using C.

Note

The guide will refer to GPIO pin 2 as CLK and pin 3 as DIO to match with the data sheet.

Code snippets

The guide assumes you have a way to control GPIO pins in C. The code snippets will be using the functions [setClk\(\)](#) and [setDio\(\)](#) to set the values of pin 2 and pin 3. The function [wait1\(\)](#) will sleep for 400ns. [setDirection\(\)](#) sets the GPIO direction and [getValue\(\)](#) reads the value of the pin

C Code

Start signal

From the datasheet⁷, to send the start signal, CLK will have to be high then DIO will be changed from high to low.

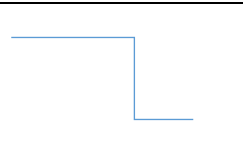
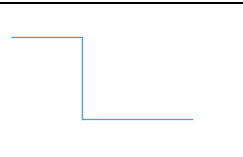
C

```
static void tm_start(void)
{
    /*
     * When CLK is high, and DIO goes from high to Low, input begins
     */
    setClk(HIGH);
    setDio(HIGH);
    wait1();

    setDio(LOW);
    wait1();

    setClk(LOW);
    wait1();
}
```

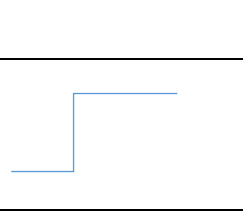
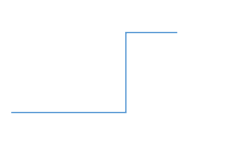
⁷ Page 3 of TM1637 datasheet

Waveform	
CLK	
DIO	

Stop signal

To send the stop signal, CLK will be high, and DIO will change from low to high

C
<pre> static void tm_stop(void) { /* * When CLK is high, and DIO goes from low to high, input ends */ setClk(LOW); setDio(LOW); wait1(); setClk(HIGH); wait1(); setDio(HIGH); wait1(); } </pre>

Waveform	
CLK	
DIO	

Data write

To send data to the 4-digit display, the bits are sent from **lower order to higher order**⁸. After a byte is sent, the 4-digit display will send an ACK on the falling edge of the 8th clock cycle and last until the falling edge of the 9th clock cycle.

C

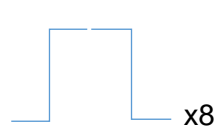
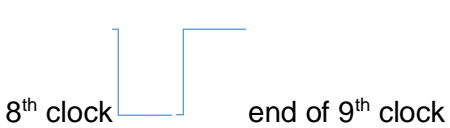


```
static void tm_write(char data) {
    /*
     *Send each bit of data
     */
    for(int i = 0; i < 8; i++) {
        //transfer data when clock is low, from low bit to high bit
        setClk(LOW);
        setDio(data & 0x01);
        data >>= 1;
        wait1();

        setClk(HIGH);
        wait1();
    }

    /*
     * End of 8th clock cycle is the start of ACK from TM1637
     */
    setClk(LOW);
    setDirection(DIO, IN);
    wait1();
    //Check that we are getting the ACK from the device
    assert(getValue(DIO) == 0);

    setClk(HIGH);
    wait1();

    setClk(LOW);
    setDirection(DIO, OUT);
}
```

Waveform	Data	ACK
CLK		
DIO		

⁸ Page 3 of TM1637 datasheet

Controlling the 4-digit display

Now we have the basic functions that we need to control the 4-digit display.

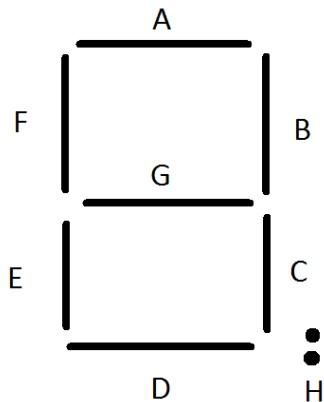
First we start by writing **0x40** which is the command to use auto address incrementing⁹. This means we can just write 4 bytes of data for the digits.

C

```
#define CMD_AUTO_ADDR 0x40

tm_start();
tm_write(CMD_AUTO_ADDR);
tm_stop();
```

Next we have to write the starting address, followed by the data for the digits. We will write all four digits at once so our starting address is **0xC0**¹⁰. We also need to get the value to display a digit. Here is the table of digits and their corresponding values¹¹. To show a colon separator we bitwise or the value of the digit with 0x80.



Digit	H	G	F	E	D	C	B	A	Value
0	0	0	1	1	1	1	1	1	0x3f
1	0	0	0	0	0	1	1	0	0x06
2	0	1	0	1	1	0	1	1	0x5b
3	0	1	0	0	1	1	1	1	0x4f
4	0	1	1	0	0	1	1	0	0x66
5	0	1	1	0	1	1	0	1	0x6d
6	0	1	1	1	1	1	0	1	0x7d
7	0	0	0	0	0	1	1	1	0x07
8	0	1	1	1	1	1	1	1	0x7f
9	0	1	1	0	0	1	1	1	0x67

⁹ Data command setting – p.4 of TM1637 datasheet

¹⁰ Address command setting – p.5 of TM1637 datasheet

¹¹ Display register address – p.2 and Hardware connection drawing – p.8

C

```
#define CMD_AUTO_ADDR 0x40
#define START_ADDR 0xc0
#define NUM_DIGITS 4

#define COLON_FLAG 0x80
#define ASCII_0 48
#define ASCII_9 57
const static char displayDigits[10] = {
    0x3f,
    0x06,
    0x5b,
    0x4f,
    0x66,
    0x6d,
    0x7d,
    0x07,
    0x7f,
    0x67,
};

static char convertChar(char ch, _Bool colon) {
    char val = 0;
    if ((ASCII_0 <= ch) && (ch <= ASCII_9)) {
        val = displayDigits[ch - ASCII_0];
    }

    if (colon) {
        return val | COLON_FLAG;
    }

    return val;
}

void fourDigit_display(char* digits, _Bool colonOn) {
    assert(strlen(digits) == NUM_DIGITS);

    tm_start();
    tm_write(CMD_AUTO_ADDR);
    tm_stop();

    tm_start();
    tm_write(START_ADDR);
    for (int i = 0; i < NUM_DIGITS; i++) {
        tm_write(convertChar(digits[i], colonOn));
    }
    tm_stop();
}
```


Next, we need to set the brightness of the display. To do this we will write **0x88**¹² which turns on the display and bitwise or it with the brightness setting from 0 – 7.

C

```
#define DISPLAY_ON 0x88

.
.
.

void fourDigit_display(char* digits, _Bool colonOn) {
    assert(strlen(digits) == NUM_DIGITS);

    tm_start();
    tm_write(CMD_AUTO_ADDR);
    tm_stop();

    tm_start();
    tm_write(START_ADDR);
    for (int i = 0; i < NUM_DIGITS; i++) {
        tm_write(convertChar(digits[i], colonOn));
    }
    tm_stop();

    tm_start();
    //This sets it to the brightest
    tm_write(DISPLAY_ON | 0x07);
    tm_stop();
}
```

Troubleshooting

- **No ACK received from the 4-digit display (the assert fails)**
Ensure that the jumper by the buzzer is removed. Also check that the GPIO pins are configured correctly for output and that it can also be properly switched to input.
- **Nothing appears on the 4-digit display**
Make sure that you are writing valid digits (the example code writes 0 for invalid characters). Also make sure that you are writing the correct display setting value (0x88).

¹² Display control – p.5 of TM1637 datasheet