# Microcontroller

## Choosing Microcontroller

### Requirements

* Rich set of peripherals such as I2C, SPI, UART, USB, etc. to be able to use various components without compromise.
* Sufficient number of GPIO pins
* Easy to extend functionality of project
* Easy debugging
* Enough clock speed to support small signal analysis (Future extension of the project)
* Easily available (in WEL off course)

TM4C123GH6PM is a high performance 32-bit ARM Cortex-M microcontroller. It provides support for rich set of peripherals along with higher clock rate. That’s why it is suitable for the easy customizable and extendable project like this. Its 32-Bit architecture provides extensive computation power which may be useful in small signal analysis.

## Peripherals

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Peripheral** | **Use** |
| 1 | USB | To achieve high speed data and instruction transfer between microcontroller and application running on computer. |
| 2 | SPI | Current version of project uses SPI to control ADC, DAC and Digital Potentiometer |
| 3 | I2C | Current and Voltage across Peltier device are received through I2C interface |
| 4 | PWM Block | Signalling LED and voltage across Peltier are controlled using Pulse Width Modulation |
| 5 | GPIO Pins | Number of GPIO pins are used for controlling on board devices such as relays and ADC, ADC, etc. |

### Serial Peripheral Interface

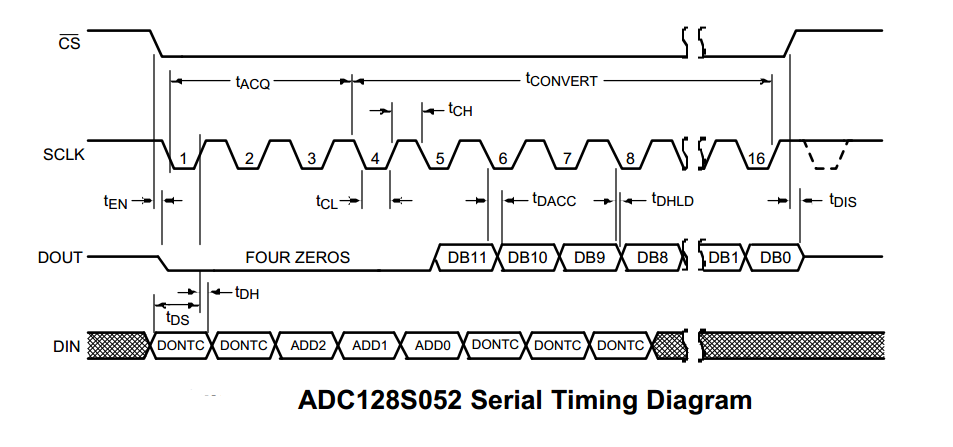
The SSI0 module on microcontroller is used as SPI to control ADC, DAC and Digital Potentiometer. These devices use 16-Bit packet format and support SPI clock speed above 1 Meg/Sec so, the configuration of SSI0 module is as follows:

|  |  |
| --- | --- |
| * Protocol: SPI Mode 0 with MSB first | * Mode: Master |
| * SPI Clock Rate: 1 Meg/Sec | * Packer Width: 16-Bits |

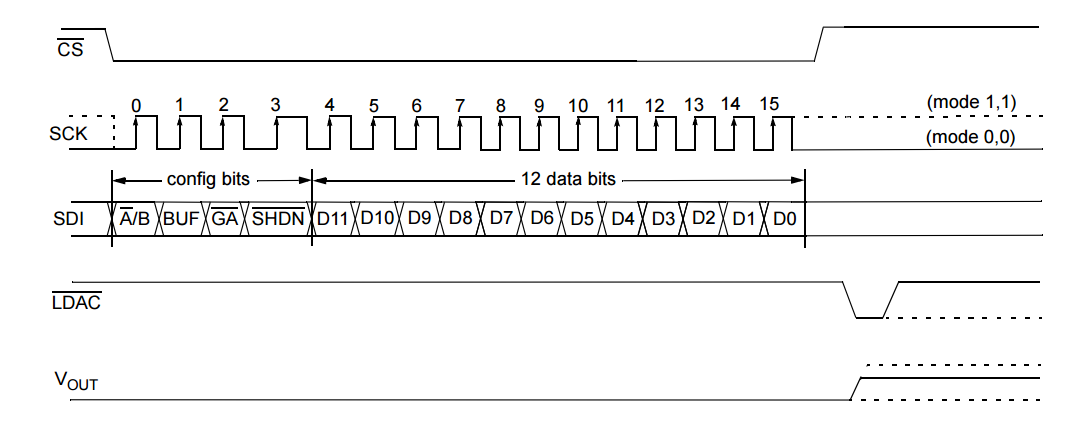
SSI0 module controls multiple devices connected in parallel pattern. So, chip selection is supported through multiple GPIO pins.

#### Devices:

1. ADC: ADC sends quantize value of selected signal in 16 bits out of which first 4 bits are zero. Microcontroller must send channel to be selected for next conversion while receiving current value. For detailed information please refer data sheet of the ADC. GPIO pin PA3 is used as chip select pin for the ADC.



1. DAC: DAC (MCP 4921) is controlled using 16-Bit message out of which first four bits are control bits which are followed by 12-Bit wide voltage value. GPIO pin PA6 is used as chip select pin for the DAC



### I2C Interface

Current and Voltage across Peltier device are measured using INA 260 which implements I2C interface in I2C standard speed mode (100 Kb/s). It sends measured values in the form of 16-bit signed integer. Microcontroller is configured to use I2C0 module in standard mode to implement I2C connection with INA 260 which has the bus address of 0x44 and sets it in averaging mode to get more reliable measurements. For more information please refer the data sheet of the same.

### USB Interface

Microcontroller uses USB0 module to exchange data and information with the controlling computer. It uses USB 2.0 interface in bulk transfer mode to exchange packages.

Configuration of the USB device:

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Parameter** | **Value** |
| 1 | Input FIFO depth | 256 Bytes (in RAM) |
| 2 | Output FIFO depth | 256 bytes (in RAM) |
| 3 | Device Vender ID | 0x1CBE |
| 4 | Product String | Peltier and CHIL |
| 5 | Serial Number String | 12345678 |
| 6 | Data Interface String | Bulk Data Interface |

### PWM Block

PWM block on the controller controls the onboard RGB LEDs and voltage across peltier device by varying duty cycle of PWM signal. PWM1 module is configured with clock rate of 50 MHz and pre-set value of 4000 which results into PWM frequency of 12.5 KHz.

## Pin Mapping

Some of the pins are kept reserved for future extension of the project

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr. No. | Device | Peripheral | Pins | | Configurations |
| Function | Name |
| 1 | SPI | SSI0 | CLK | PA2 | SSI0 as SPI Master in mode0.  Bit rate = 8MHz, Data width = 16bits  *(read two channels one by one*  *To get differential voltage)*  All CS pins are initially pulled up |
| MISO | PA4 |
| MOSI | PA5 |
| CS(ADC) | PA3 |
| CS(DAC) | PA6 |
| CS(DigiPot) | PA7 |
| CS(Reserved) | PB[4-7] |
| 2 | INA | I2C0 | SCL | PB2 | I2C0 as I2C Master in fast mode (1 MHz) |
| SDA | PB3 | Data width = 16 bits |
| 3 | PWM | M1\_PWM5 | Red LED | PF1 | PWM frequency 50 MHz  Max duty width 1000 |
| M1\_PWM7 | Green LED | PF3 |
| M1\_PWM6 | Blue LED | PF2 |
| M1\_PWM4 | Peltier Cooler | PF0 |
| 4 | USB | USB0 | VBUS | PB1 | USB 2.0 |
| ID | PB0 |
| D+ | PD5 |
| D- | PD4 |
| 5 | GPIO |  | IV Relay | PC4 | Default Low |
| Peltier Relay | PC5 | Default Low |
|  | Reserve | PC[6,7] |  |
|  | Reserve | PE[1-5] |  |

## Control Flow



Overall Picture of the Controller

The microcontroller is guided by the application running on the PC using various instructions. Application sends messages corresponding to each operation and waits until controller finishes the given task. At boot microcontroller starts USB peripheral only waits for application to send future command.

Some important aspects of the controller are as follows:

* Microcontroller initializes only USB peripheral after booting. Application later sends it command to initialize other peripherals. This helps to avoid unnecessary operations until user has not commanded to start.
* Temperature controlling function and IV acquiring function are called frequently after specific time using a timer interrupt. Current application sets timer value to get an interrupt after every 10ms. Interrupt serves as temperature controlling function for once and IV acquiring function for next three times.
* IV acquisition function acquires data only when temperature is stable.
* Microcontroller acquires IV data for specified voltage and current interval for given temperature and sends completion signal to application and waits until called to do so again.
* Microcontroller maintains the temperature even after completion until application sends stop instruction.

## Code Organization

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **File Name** | **Description** |
| 1 | Peltier\_and\_CHIL.c | Main function and instruction decoding functions |
| 2 | startup\_ccs.c | Map of all the interrupts to corresponding handlers |
| 3 | temp\_loop.h | Temperature Controlling Functions |
| 4 | temp\_loop.c |
| 5 | IC.h | IV Acquisition function |
| 6 | IV.c |
| 7 | TivaC\_USB.h | USB interrupt handler and data transfer functions |
| 8 | TivaC\_USB.c |
| 9 | usb\_bulk\_struct.h | Defines USB structures such as FIFOs, Device String, etc. |
| 10 | usb\_bulk\_struct.c |
| 11 | basic\_includes.h | All libraries and global variables |

## Message Packets

Message packets are divided into two groups, command and data. All the data packets start with byte 0x22 while command packets start with 0x21. The second last byte of all the packets is 0x00 and last byte indicates the size of the packet. This two bytes can help to ensure coherency of the packet.

### Data Packets

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IV Data | | | | | | | | | |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Value | 0x22 | 0x53 | Sign | Device Voltage | | Device Current | | 0x00 | 0x09 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Temp Loop Data | | | | | | | | | | |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Value | 0x22 | 0x59 | Temperature | | Peltier Current | | PWM Duty | | 0x00 | 0x0A |

### Command Packets

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Start IV | | | | | | |
| Index | 0 | 1 | 2 | 3 | 4 | 5 |
| Value | 0x21 | 0x81 | Temperature | | 0x00 | 0x06 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Start Devices | | | | |
| Index | 0 | 1 | 2 | 3 |
| Value | 0x21 | 0x99 | 0x00 | 0x04 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set Device Parameters | | | | | | | | | | | | | |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Value | 0x21 | 0x66 | Sign | Voltage Max | | Voltage Min | | Current Max | | Current Min | | 0x00 | 0x0D |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stop Devices | | | | |
| Index | 0 | 1 | 2 | 3 |
| Value | 0x21 | 0x55 | 0x00 | 0x04 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Command Acknowledgment | | | | |
| Index | 0 | 1 | 2 | 3 |
| Value | 0x21 | 0xFF | 0x00 | 0x04 |

## Installing Softwares for TM4C Launchpad Board

### Code Composer Studio 7

Code Composer Studio (CCStudio or CCS) is an integrated development environment (IDE) to develop applications for Texas Instruments (TI) embedded processors.

To install CCS run the downloaded executable and follow the instructions.

Download Link: <http://processors.wiki.ti.com/index.php/Download_CCS>

### TivaWare for C Series

Includes royalty-free libraries (Peripheral, USB, Graphics, Sensor) and kit-/peripheral-specific code examples for all TM4C devices.

Download Link: <http://www.ti.com/tool/sw-tm4c>

Note: Download complete package

### StellarisWare embedded USB drivers

The Stellaris USB library provides a Windows™-based INF for supported USB classes in a precompiled DLL that saves development time. This library is helpful for debugging USB peripheral with Windows PC. However, this library is not necessary for this project.

Download Link: <http://www.ti.com/tool/SW-USB-WINDRIVERS>

Note: This is not in circuit debugger driver which helps to program and debug device from debug unit

### Libusb-win32

libusb is a C library that gives applications easy access to USB devices on many different operating systems. This library is useful for interfacing Tiva-C Launchpad with GUI using USB 2 interface.

Installing Lib-USB for specific device on Windows PC

1. Connect the USB device (Device USB not Programmer) and open Zadig.
2. Go to options and select device vendor ID (0x1CBE in this case) and install libUSB-win32 latest version
3. Verify installation using test program from Libusb filter package.

Download Links:

Zadig: <http://zadig.akeo.ie/downloads/zadig_2.2.exe>

Libusb filter package:

<https://excellmedia.dl.sourceforge.net/project/libusb-win32/libusb-win32-releases/1.2.6.0/libusb-win32-devel-filter-1.2.6.0.exe>

# User Interface Application

The GUI application is developed using Python 3.5 and PyQt5 with other supporting packages such as Matplotlib. Important operations of the application are:

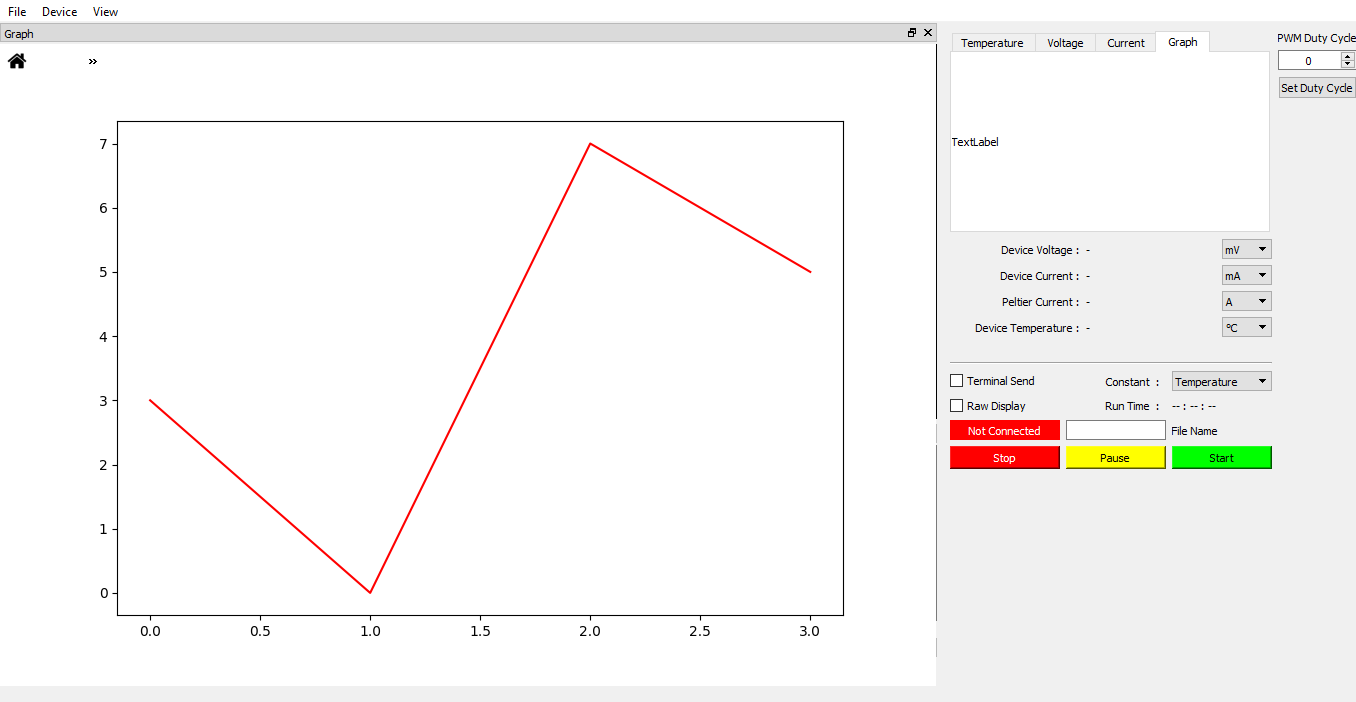
* Interact with the user to get device parameters
* Control tasks performed by microcontroller
* Display output to the user

Some important aspects of the application

* Separate thread for receiving data from USB. This allows unblocked interaction with the user and USB device simultaneously
* Dock widgets for plotting graphs which helps to user to focus on data
* Live display of device parameters
* Platform independent application (Compatible with Linux, Windows, Mac)

Note: GUI is under development.

Complete GUI will be close to the one shown in the image



## Installing Required Python-3 Packages for GUI

We have used Python 3.5 for this project so, all the libraries are for Python 3.

### PyQt5

PyQt brings together the Qt C++ cross-platform application framework and the cross-platform interpreted language Python. Qt is more than a GUI toolkit. It includes abstractions of network sockets, threads, Unicode, regular expressions, SQL databases, SVG, OpenGL, XML, a fully functional web browser, a help system, a multimedia framework, as well as a rich collection of GUI widgets. GUI of this project uses PyQt extensively for threading and user interface.

This project is developed using PyQt5.

Installing PyQt5 using PIP3

pip3 install pyqt5

Installing PyQt5 designer using PIP3

pip3 install pyqt5-dev-tools

Note: You need root access for installing these packages. On Windows PC, you can use Command Prompt(Admin)

### Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. This project uses Matplotlib for plotting graphs.

Installing Matplotlib

pip install matplotlib

### PyUSB

PyUSB aims to be an easy to use Python module to access USB devices. PyUSB relies on a native system library for USB access.

Installing PyUSB

pip install pyusb

Note for Windows user: You must have libusb-win32 installed for using PyUSB