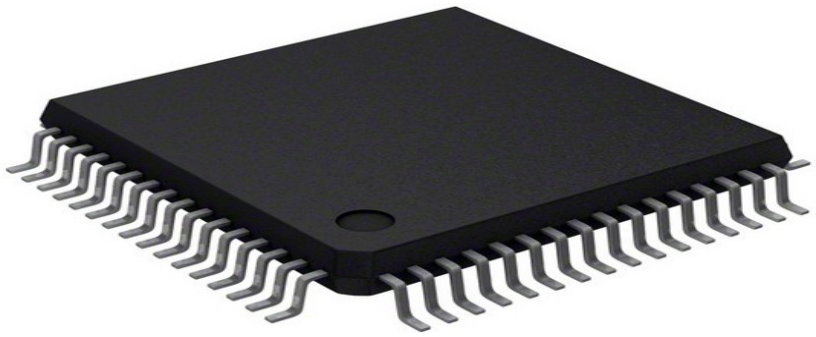

# Microprocessors Project

Lesson : Microprocessors
Teacher : V.Tenentes
Team : 5V ( X.Ketoglou , V.Kontotoli )
University of Ioannina
2019

# Table of Contents

# 1.0 Introduction



The basic idea behind this project is to supply other devices with voltage and adjust it, until it reaches a certain level that doesn’t allow the device to function properly.

In this project, for the Microprocessors lesson, we built a mini power supply in which the voltage is adjustable.

To be more specific we implement a circuit that can handle an input voltage from 6.5V-35V and can output a maximum of 5V with 5mV step. The core of the project is the microcontroller, we decided to use the PIC18F27K42 microcontroller from Microchip. The MCU can handle the output voltage of a voltage regulator and monitor the output voltage and current in real time.

In order to do all this, the MCU must be programmed with the proper firmaware. Additionally we built a PC program, which is the interface between the user and the microcontroller.

With this program the user can set the output voltage from 0V-5V with 5mV step and monitor the voltage and current in real time. The user has also the option to put a limit to the current. When the current reaches this limit, a buzzer will ring and the voltage will be set automatically to 0V.

In order to implement the project we used Altium Designer as a CAD tool to design the circuit. The hardware components that were used, are explained in later chapters. After the hardware, we implemented the firmware part of the microcontroller. In order to accomplish this, we used the XC8 compiler along with MPLAB X IDE from Microchip and of course embedded C as programming

language. To load the firmware to the microcontroller we use PICKit 3 programmer. And finally we built the software part of the project for the PC. To accomplish this, we decided to use Python as programming language due to the flexibility and easiness it provides.

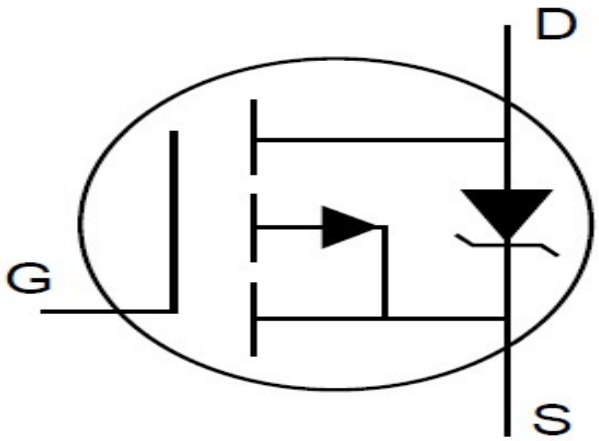
# 2.0 Hardware

The hardware of the project includes the following components:

- PIC18F277K42 as a microcontroller
- SUD50P06 P-Channel MOSFET
- L7805 voltage regulator
- CH340E USB to serial chip
- AD5272 variable resistor chip
- MCP6281 amplifier
- LM7705 negative voltage regulator
- The main LT3081 variable voltage regulator
- A variety of capacitors and resistors.

In the following sections, we will introduce every single chip and its operations in the project circuit.

# 2.1 SUD50P06




The SUD50P06 is a P-Channel MOSFET and it is used at the beginning of the circuit. Its purpose is to protect the circuit from inverse polarity currents. It is frequent phenomenon for the user to connect the energy source to the device improperly. To prevent damage to the circuit, from such situation, we use a P-Channel MOSFET which can handle a maximum Vds of -60V which is far more from the recommended value we suggest. In normal operation, when the connection of energy source is properly plugged, the Drain of the MOSFET is connected to the positive terminal of the energy source, the Gate is connected to the negative terminal of the energy source and because of the P-Channel, we will have a current flow from Drain to Source. Otherwise the Drain is connected to the negative terminal and the Gate to the positive terminal and the current doesn’t flow through the circuit.

Mathematically the Vds must be >= Vgs-Vt for the transistor to be fully on. We also added a led next to the MOSFET so we can see when we have power in the circuit.

# 2.2 L7805

The L7805 is a stable voltage regulator. It can handle up to 35V as an input and produce a stable 5V of output. We use this regulator so the MCU can operate independently from the voltage level of the energy source (35V max input). This circuit need two capacitors to work properly, one in the input and one in the output of the circuit.

# 2.3 PIC18F27K42


The core of this project is the microcontroller. For this project we used an MCU from advanced 8-bit family (PIC18) of Microchip. The PIC18F27K42 is an 8-bit microcontroller with 128KB program flash memory, 8KB of SRAM, it operates from 2.3V-5.5V and it can run up to 64MHz. It also includes a lot of peripheral circuits and for this project we will use the following:

- Interrupt controller
- PPS module
- Watchdog timer
- Timer 0

- UART
- I2C
- ADC

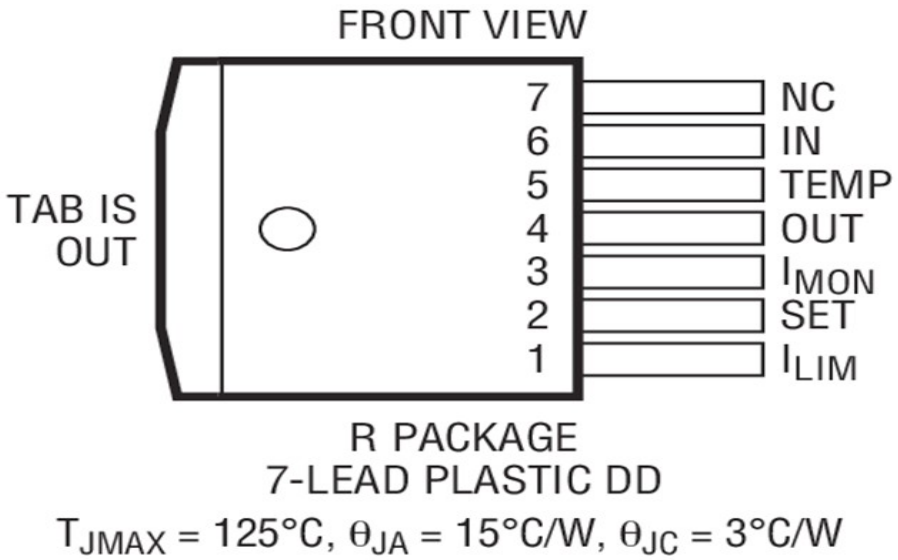
This microcontroller needs pull-up resistors and a capacitor at MCLR pin as recommended by the manufacturer, in order to function properly. Additionally we will use two pull-up resistors at RBO, RB1 pins which will be our communicate pins for I2C protocol. RB0 is the SDA so data can flow through this pin and RB1 is the SCL for clock syncronization. The RB3 pin it is used to reset the AD5272 in case something goes wrong and the chip does not respond. The RB4 pin read the voltage from LT3081 output and the RB5 pin read the differential amplifier MCP6281 amplified voltage so we can calculate the current. The RC6 and RC7 pins are the RX and TX respectively and are used to communicate with the PC. A buzzer is connected at RA1 pin and can be set to ring from user when the current exceeds the limit. Finally a led is flashing from RA0 pin to indicate that the microcontroller is fully functional and working. To flash the firmware, we use the PICKit 3 programmer and the RB6 and RB7 pins needed for this job as clock and data pins.

# 2.4 CH340E

This IC is a simple converter. It converts the differential USB signals from PC to TTL serial signals for the microcontroller. The microcontroller can communicate only via UART because it does not include a USB peripheral so it can understand the USB signals

directly. The MCU can communicate only at TTL levels and this converter makes this conversion for the MCU.

# 2.5 LT3081

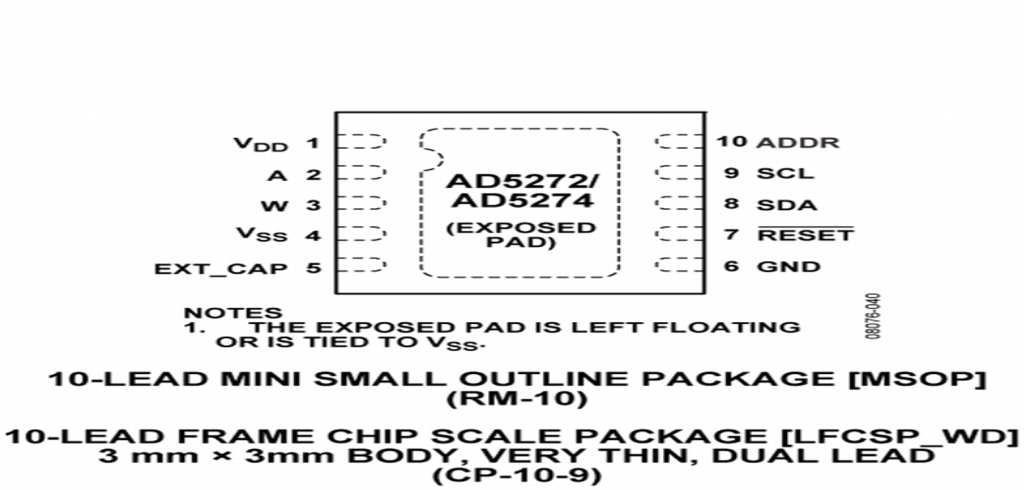



The LT3081 is a variable regulator with a lot of features. It has six active pins and one inactive. This pins are for input, output, voltage set, current set, current monitor and internal temperature. For this project we use only the voltage set pin, the current monitor pin and off course the input and output pins of the regulator. The minimum voltage for the regulator at this project is 6.5V and the maximum 35V. The output varies from 0V-5V. In order to set the output voltage, a resistor must be connected from voltage set pin to the ground. The value of the resistor affects the output voltage according to the mathematical type from data sheet of the regulator :

$$Vout = 50\mu A * Rset$$

The resistor in this project is set from AD5272 variable resistor chip which can be commanded from the MCU.
Finally the current monitor pin produces a voltage drop across a resistance which is measured and amplified by MCP6281 amplifier and sent to MCU for reading.

# 2.6 AD5272

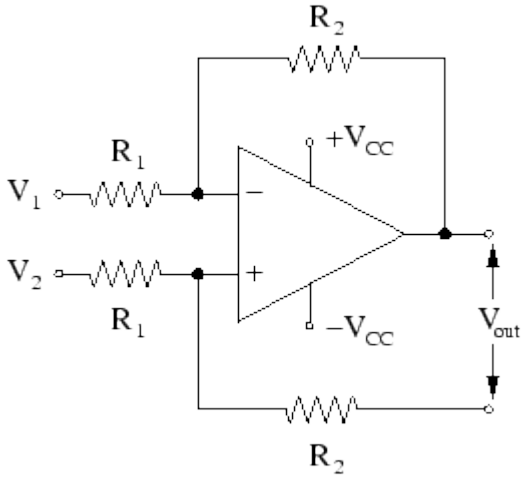



10-LEAD MINI SMALL OUTLINE PACKAGE [MSOP]
(RM-10)

10-LEAD FRAME CHIP SCALE PACKAGE [LFCSP_WD]
3 mm × 3mm BODY, VERY THIN, DUAL LEAD
(CP-10-9)

The AD5272 is a variable resistor chip, that can communicate via I2C with the MCU, and set the appropriate resistor according to the communication. It is 100KΩ and 1023 steps, which means that the resistor has to be set from 0Ω to 100KΩ with a step of 97,75Ω. So, if we need an output of 3.3V from the LT3081, we need to set a 3.3V/50μA = 66KΩ resistor. Though, the closest value that can be set is 675*97.75 = 65981,25Ω (66KΩ/97.75 = 675,19). Errors like this do not affect the result of this project.

# 2.7 MCP6281 & LM7705




The MCP6281 plays the role of a simple differential amplifier. The current monitor pin of the LT3081 sources a small current across a resistance, which voltage drop is measured by the amplifier and amplified by G=R2/R1= 4.9. The MCU measure that voltage and sends it to the PC. From there, the software of the PC, which know that the voltage is amplified by 4.9 factor and the resistor which is measured is 1KΩ, can easily calculate the real voltage drop across the resistance via Ohm law and finally the real current flow through this resistor. Last but not least, this current, according to the manual of LT3081, is equal to the 1/5000 of the current flow through the load. So, in this way the real current can be calculated.

A common problem in amplifiers is that if the V- pin is not tied to a negative voltage source, it always returns something as output because the ground is never at 0V in reality. This problem is known as “Zero Cross” problem and the solution here is given from LM7705 negative voltage regulator which sources a voltage of -0.232V. This negative voltage is enough for the amplifier to produce a real 0V, when no voltage drop exists at the measured resistor.

# 3.0 Firmware

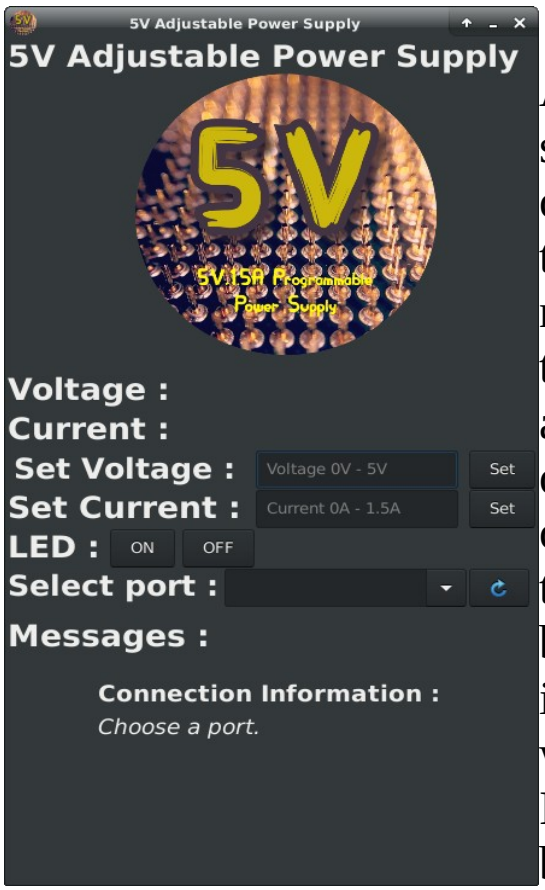
In this section we will refer to the functions that the firmware use. We don’t explain what we do in any function or what register we set etc. We explain only what the function does and what result provides. Additional information and explanation can be found at the comments in source code files.

- Void main(void) : The main function it just makes some initializations at the beginning and after that it goes in an endless loop where it waits to service the commands the MCU receives.
- void timer0_init(void) : Thins function initialize the Timer 0 module,so we can have an interrupt at a specific periodic time.
- void USART_handler(void) : This function receives the command and process it. Finally it send a respond back to PC.
- void ADC_Init(void) : This function initialize the Analog to Digital Converter.
- void ADC_Start(byte pin) : This function starts a analog to digital measure at a specific pin that we give as argument.
- int GetStringSize(void) : A helping function that returns the size of a specific string variable.
- void memset(char *st,char x,int size) : The classic C function,with our own impementation.
- void UART1_Init(unsigned char baud_rate) : This function initialize the UART module at a specific baud rate that we give as argument.Only 3 baud rates are available 19200,57600 and 115200.
- unsigned char UART1_SendByte(unsigned char byte) : With this function we can send a single byte of data to the PC.This byte is the argument.

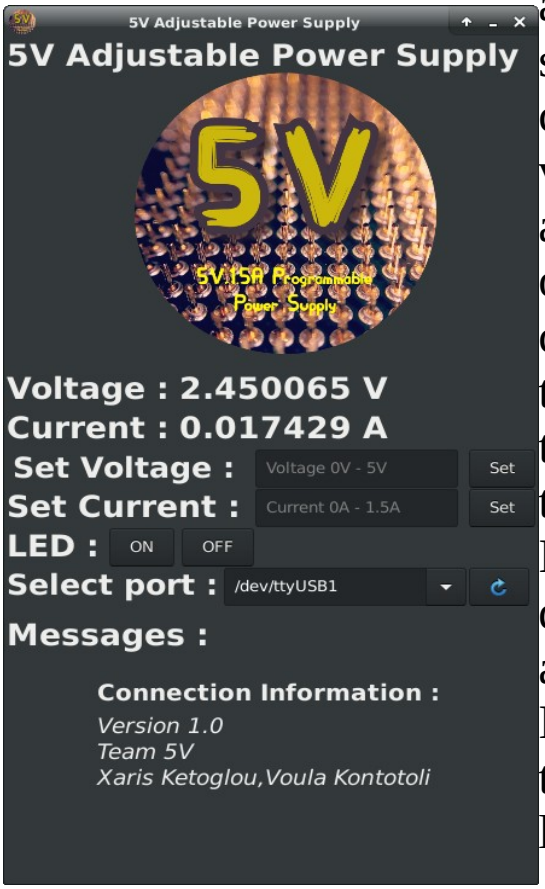
- unsigned char UART1_SendString(char *str,int size) : With this function we can send a string of data to the PC.The string and its size are given as arguments. It has to be said that the size can not be up to a specific limit that we define at UART1.h as TRANSMIT_BUFFER_SIZE.
- unsigned char UART1_ReceiveCommand(void) : This function is responsible for commands that the MCU receive. It receives and decode the data and if a proper command it is received this function set some variables that the firmware use after to understand what must be done.
- void I2C_Init(void) : This function initialize the I2C module.
- unsigned char I2C_Transmit(unsigned char *buffer,unsigned char buffer_size,unsigned char address) : This function send via I2C protocol some data to a specific device which defined by the address argument. The data is in the buffer argument and their number is defined by the size argument.
- void I2C_handler(int value) : This function just send the value we give as argument at the AD5272 chip.

# 4.0 Software

We make a PC software so we can communicate with the MCU. The software have developed and tested in Linux operating system (Ubuntu). This software is pretty simple,we use Python 3 to develop it and the source code is easy to understand. Glade was used to implement a graphical interface for the user. To run the software the user need to install pyserial library for Python 3.In this section we will introduce how the software works,with images. After the user make sure that he has Python 3 and pyserial library for Python 3 installed, he can run the software with the ***command : python3 ps_app.py***. The software will start and something like the image below will appear.




After the window open, the user must specify the port that the software must connect. This can be done by pressing the white arrow that face downward next to the “Select port” label. After that all the available ports will appear and the user must select the proper one. If the port the user want to connect does not appear he can refresh the list by pressing the blue refresh button. If the port that the user chooses is the correct,the version information will appear under the “Connection Information label” and the voltage will be read from the device and will appear next to the “Voltage” label,the same thing will happen with the current. After that the user can type the voltage he want the device to output and the current limit he wants. Another option that the software have is the control of the on-board LED. Next to the “Messages” label will appear all the useful messages after an action and the user need to pay attention to them. If the software cannot connect with the device, an appropriate message will appear under the “Connection Information” label. The user then must try to reconnect by refreshing the port list and choose the right one.



# 5.0 References


- https://www.analog.com/media/en/technical-documentation/data-sheets/3081fc.pdf
- http://ww1.microchip.com/downloads/en/DeviceDoc/Microchip%208bit%20mcu%20%20PIC18%20L_F26_27_45_46_47_55_56_57K42%20low-power%20high-performance%20mcu%20with%20xlp%20tech%2040001919B.pdf
- https://www.analog.com/media/en/technical-documentation/data-sheets/AD5272_5274.pdf
- http://ww1.microchip.com/downloads/en/devicedoc/21811d.pdf
- http://www.ti.com/lit/ds/symlink/lm7705.pdf
- https://www.mpja.com/download/35227cpdata.pdf
- https://datasheet.octopart.com/L7805CV-STMicroelectronics-datasheet-7264666.pdf
- http://www.vishay.com/docs/68940/sud50p06-15.pdf