PICSimLab 0.8.9

Luis Claudio Gambôa Lopes <lcgamboa@yahoo.com>

Download: github

HTML version of documentation

PICSimLab on Twitter

PICSimLab on Discord

July 25, 2021

Contents

Chapter 1

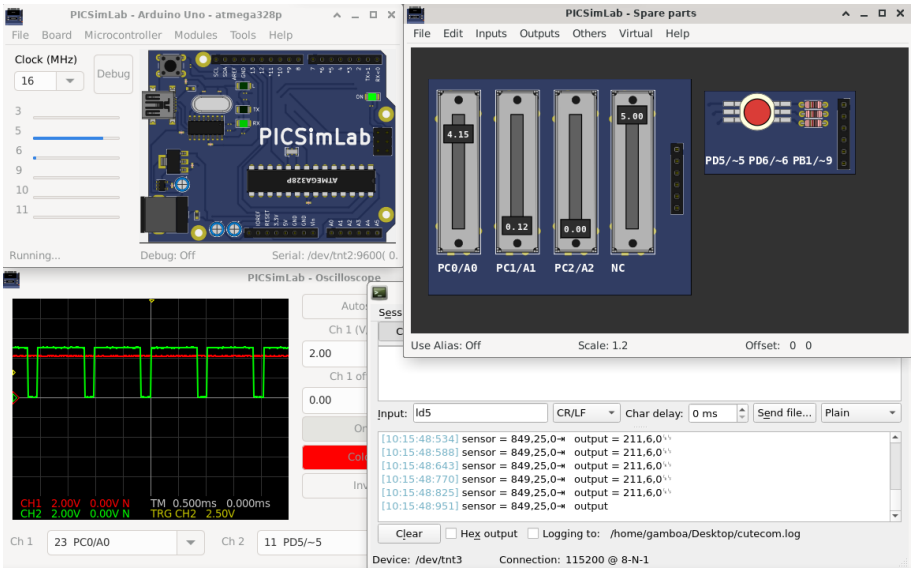
Introduction

PICSimLab means Programmable IC Simulator Laboratory

PICSimLab is a realtime emulator of development boards with integrated MPLABX/avr-gdb debugger. PICSimLab supports some picsim microcontrollers and some simavr microcontrollers. PICSimLab have integration with MPLABX/Arduino IDE for programming the boards microcontrollers. As the purpose of PICSimLab is to emulate real hardware it does not have any source code editing support. For code editing and debugging the same tools used for a real board should be used with PICSimLab, such as MPLABX or Arduino IDE.

PICSimLab supports several devices (spare parts) that can be connected to the boards for simulation. As for example LEDs and push buttons for simple outputs and inputs and some more complex ones like the ethernet shield w5500 for internet connection or the color graphic display ili9340 with touchscreen. The the complete list of parts can be accessed in the chapter Spare Parts.

The experimental version boards supports uCsim, gpsim and qemu-stm32 simulators in addition to the stable ones.


PICSimLab - Arduino Uno - atmega328p
File Board Microcontroller Modules Tools Help
Clock (MHz)
16
Debug
PICSimLab
Running...
Debug: Off
PICSimLab - Spare parts
File Edit Inputs Outputs Others Virtual Help
4.15
0.12
0.00
5.00
PC0/A0 PC1/A1 PC2/A2 NC
PD5/~5 PD6/~6 PB1/~9
Use Alias: Off
Scale: 1.2
Offset: 0 0
PICSimLab - Oscilloscope
CH1 2.00V 0.00V N
CH2 2.00V 0.00V N
TM 0.500ms 0.000ms
TRG CH2 2.50V
Ch 1
23 PC0/A0
Ch 2
11 PD5/~5
Input:
ld5
CR/LF
Char delay:
0 ms
Send file...
Plain
[10:15:48:534] sensor = 849,25,0 output = 211,6,0
[10:15:48:588] sensor = 849,25,0 output = 211,6,0
[10:15:48:643] sensor = 849,25,0 output = 211,6,0
[10:15:48:770] sensor = 849,25,0 output = 211,6,0
[10:15:48:825] sensor = 849,25,0 output = 211,6,0
[10:15:48:951] sensor = 849,25,0 output
Clear
Hex output
Logging to: /home/gamboa/Desktop/cutecom.log
Device: /dev/tnt3
Connection: 115200 @ 8-N-1

Chapter 2

Install

2.1 Stable version executables download

If you are on Linux or Windows you can download the last version at:
https://github.com/lcgamboa/picsimlab/releases
If you are on macOS you can run PICSimLab using Wine:

1. Download and install [‘xquartz‘](https://www.xquartz.org)
2. Download and install [Wine](https://dl.winehq.org/wine-builds/macosx/download.html)
3. Download the executable and double-click it to run the installer

2.2 Unstable version executables download

The binaries of last code available on github can be downloaded at: Sourceforge.net
The unstable test version have the unreleased features of Changelog_auto.md
If you need a specific binary that is not available please contact me.

2.3 Install from source

2.3.1 Linux

In Debian Linux and derivatives Linux native:
Using a user with permission to run the sudo command:
In first time build:

```
git clone --depth=1 https://github.com/lcgamboa/picsimlab.git
cd picsimlab
./picsimlab_build_all_and_install.sh
```

To recompile use:

```
make -j4
```

2.3.2 Windows

Cross-compiling for Windows (from Linux or WSL on win10)

In first time build in Debian Linux and derivatives target Windows 64 bits:

```
git clone https://github.com/lcgamboa/picsimlab.git
cd picsimlab
./picsimlab_build_w64.sh
```

To recompile use:

```
make FILE=Makefile.cross -j4
```

For target Windows 32 bits:

```
git clone https://github.com/lcgamboa/picsimlab.git
cd picsimlab
./picsimlab_build_w32.sh
```

To recompile use:

```
make FILE=Makefile.cross_32 -j4
```

2.3.3 macOS

Theoretically it is possible to compile PICSimLab natively on macOS. But I do not have access to any computer with macOS to try to compile and until today nobody has communicated that they managed to do it. (help wanted)

2.3.4 Experimental version

Experimental version can be built using the parameter "exp" on scripts:

```
./picsimlab_build_all_and_install.sh exp
./picsimlab_build_w64.sh exp
./picsimlab_build_w32.sh exp
```

And recompiled using the parameter "exp" on Makefiles:

```
make -j4 exp
make FILE=Makefile.cross -j4  exp
make FILE=Makefile.cross_32 -j4 exp
```

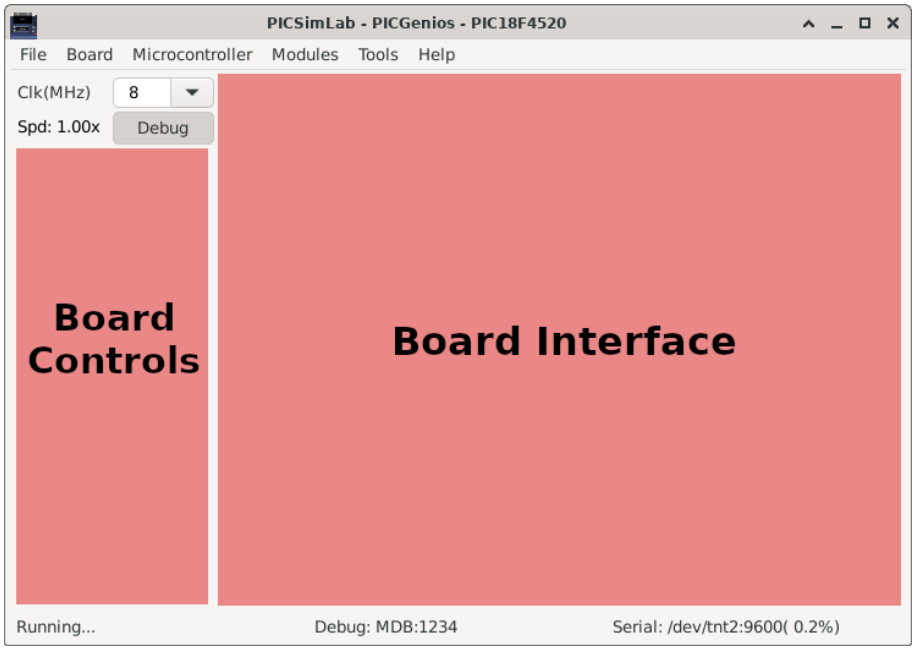
Chapter 3

Simulator Interface

3.1 Main Window

The main window consists of a menu, a status bar, a frequency selection combobox, an on/off button to trigger debugging, some board-specific controls and the part of the board interface itself.

In the title of the window is shown the name of the simulator PICSimLab, followed by the board and the microcontroller in use.



The frequency selection combobox directly changes the working speed of the mi-

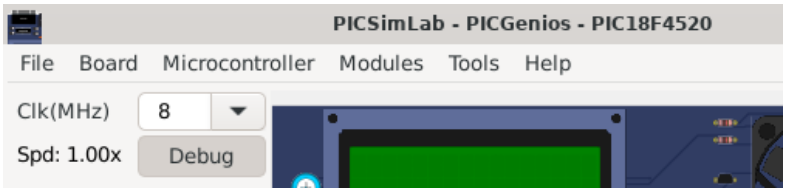
crocontroller. The “Spd” label show the ratio between simulation speed and real time. when the “Spd” label goes red indicates that the computer is not being able to run the program in real time for the selected clock. In this case the simulation may present some difference than expected and the CPU load will be increased.

The on/off button to enable debugging is used to enable debugging support, when active simulation load is increased.

The menus and their functions are listed below:

- File
  - Load Hex - Load .hex files
  - Reload Last - Reload the last used .hex file
  - Save Hex - Save memory in a .hex file
  - Configure - Open the configuration windows
  - Save Workspace - Saves all current workspace settings to a .pzw file
  - Load Workspace - Loads saved settings from a .pzw file
  - Exit
- Board
  - Arduino Uno - Choose board Arduino Uno
  - Breadboard - Choose board Breadboard
  - Franzininho - Choose board Franzininho
  - K16F - Choose board K16F
  - McLab1 - Choose board McLab1
  - McLab2 - Choose board McLab2
  - PICGenios - Choose board PICGenios
  - PQDB - Choose board PQDB
- Microcontroller
  - xxxxx - Selects the microcontroller to be used (depends on the selected board)
- Modules
  - Oscilloscope - Open the oscilloscope window
  - Spare parts - Open the spare parts window
- Tools
  - Serial Terminal - Open the serial terminal Cutecom
  - Serial Remote Tank - Open the remote tank simulator
  - Esp8266 Modem Simulator - Open the Esp8266 Modem Simulator

  - Arduino Bootloader - Load microcontroller with Arduino serial bootloader
  - MPLABX Debugger Plugin - Open the web page to download the MPLABX Debugger Plugin
  - Pin Viewer - Open the Pin Viewer
- Help
  - Contents - Open the Help window
  - Board - Open the Board Help window
  - Examples - Load the examples
  - About Board - Show message about author and version of board
  - About PICSimLab - Show message about author and version of PICSimLab



The first part of the status bar shows the state of the simulation, in the middle part the status of the debug support and in the last part the name of the serial port used, its default speed and the error in relation to the real speed configured in the microcontroller.

3.2 Interaction with the Board

On the interface area of the board it is possible to interact in some ways:

- Click in ICSP connector to load an .hex file.
- Click in PWR button to ON/OFF the emulator..
- The buttons can be activated through mouse or keys 1, 2, 3 e 4.
- Click and drag in potentiometers to change their values.
- Click on EEPROM memory to view its contents.

3.3 Command Line

PICSimLab supports two command lines format:

One for load a PICSimLab Workspace file (.pzw)

```
picsimlab file.pzw
```

And other for load .hex files

```
picsimlab boardname microcontroller [file.hex] [file.pcf]
```

3.4 Remote Control Interface

The remote control interface allows other programs to control the PICSimLab simulation through a TCP/IP socket using text formatted commands.

The PICSimLab remote control interface supports TCP connections using telnet or nc (netcat).

The default port is 5000 and can be changed in configuration windows.

The 'rlwrap' command can be used for best command edition support in telnet or nc:

```
rlwrap nc 127.0.0.1 5000
```

The supported commands can be shown using the "help" command:

```
help
List of supported commands:
  dumpe [a] [s]- dump internal EEPROM memory
  dumpf [a] [s]- dump Flash memory
  dumpr [a] [s]- dump RAM memory
  exit         - shutdown PICSimLab
  get ob       - get object value
  help         - show this message
  info         - show actual setup info and objects
  loadhex file - load hex file (use full path)
  pins         - show pins directions and values
  pinsl        - show pins formated info
  quit         - exit remote control interface
  reset        - reset the board
  set ob vl    - set object with value
  sim [cmd]    - show simulation status or execute cmd start/stop
  sync         - wait to syncronize with timer event
  version      - show PICSimLab version
Ok
```

The "info" command show all available "objects" and values:

```
info
Board:     Arduino Uno
Processor: atmega328p
Frequency:   16000000 Hz
Use Spare: 1
    board.out[00] LD_L= 254
  part[00]: LEDs
    part[00].out[08] LD_1= 254
    part[00].out[09] LD_2= 30
    part[00].out[10] LD_3= 254
    part[00].out[11] LD_4= 254
    part[00].out[12] LD_5 254
    part[00].out[13] LD_6= 254
    part[00].out[14] LD_7= 254
  part[01]: Buzzer
    part[01].out[02] LD_1= 140
  part[02]: Push buttons
    part[02].in[00] PB_1= 1
    part[02].in[01] PB_2= 0
    part[02].in[02] PB_3= 1
    part[02].in[03] PB_4= 1
    part[02].in[04] PB_5= 1
    part[02].in[05] PB_6= 1
    part[02].in[06] PB_7= 1
    part[02].in[07] PB_8= 1
Ok
```

The “pins” command show all pins directions and digital values:

```
pins
  pin[01] ( PC6/RST) < 0                pin[15] (  PB1/~9) > 0
  pin[02] (   PD0/0) < 1                pin[16] ( PB2/~10) > 0
  pin[03] (   PD1/1) < 1                pin[17] ( PB3/~11) > 0
  pin[04] (   PD2/2) < 1                pin[18] (  PB4/12) < 0
  pin[05] (  PD3/~3) > 0                pin[19] (  PB5/13) > 0
  pin[06] (   PD4/4) < 1                pin[20] (     +5V) < 1
  pin[07] (     +5V) < 1                pin[21] (    AREF) < 0
  pin[08] (     GND) < 0                pin[22] (     GND) < 0
  pin[09] (  PB6/X1) < 0                pin[23] (  PC0/A0) < 0
  pin[10] (  PB7/X2) < 0                pin[24] (  PC1/A1) < 0
  pin[11] (  PD5/~5) < 1                pin[25] (  PC2/A2) < 0
  pin[12] (  PD6/~6) < 1                pin[26] (  PC3/A3) < 0
  pin[13] (   PD7/7) < 1                pin[27] (  PC4/A4) > 0
  pin[14] (   PB0/8) > 0                pin[28] (  PC5/A5) > 0
Ok
```

The “pinsl” command show all pins info in text formatted output:

```
pins1
28 pins [atmega328p]:
  pin[01] D I 0 000 0.000 "PC6/RST "
  pin[02] D I 1 200 0.000 "PD0/0   "
  pin[03] D I 1 200 0.000 "PD1/1   "
  pin[04] D I 1 200 0.000 "PD2/2   "
  pin[05] D O 0 007 0.000 "PD3/~3  "
  pin[06] D I 1 200 0.000 "PD4/4   "
  pin[07] P I 1 200 0.000 "+5V     "
  pin[08] P I 0 000 0.000 "GND     "
  pin[09] D I 0 000 0.000 "PB6/X1  "
  pin[10] D I 0 000 0.000 "PB7/X2  "
  pin[11] D I 1 200 0.000 "PD5/~5  "
  pin[12] D I 1 200 0.000 "PD6/~6  "
  pin[13] D I 1 200 0.000 "PD7/7   "
  pin[14] D O 0 000 0.000 "PB0/8   "
  pin[15] D O 0 000 0.000 "PB1/~9  "
  pin[16] D O 0 000 0.000 "PB2/~10 "
  pin[17] D O 0 006 0.000 "PB3/~11 "
  pin[18] D I 0 000 0.000 "PB4/12  "
  pin[19] D O 0 000 0.000 "PB5/13  "
  pin[20] P I 1 200 0.000 "+5V     "
  pin[21] R I 0 000 0.000 "AREF    "
  pin[22] P I 0 000 0.000 "GND     "
  pin[23] A I 0 000 0.875 "PC0/A0  "
  pin[24] A I 0 000 1.925 "PC1/A1  "
  pin[25] A I 0 000 2.700 "PC2/A2  "
  pin[26] A I 0 000 4.275 "PC3/A3  "
  pin[27] D O 1 179 0.000 "PC4/A4  "
  pin[28] D O 1 186 0.000 "PC5/A5  "
Ok
```

You can view one input/output state using the “get” command:

```
get board.out[00]

get part[02].in[01]
```

Its possible use the “get” command to view individual pins state:

```
#digital state
get pin[19]
pin[19]= 0
Ok

#digital mean value (0-200)
get pinm[19]
pin[18]= 100
```

```
Ok

#analog state
get apin[25]
apin[25]= 2.700
Ok

#all info
get pinl[13]
pin[13] D I 1 200 0.000 "PD7/7   "
Ok
```

And set value of one input using the “set” command:

```
set part[02].in[01]  0
set part[02].in[01]  1
```

Or set value of one pin using the “set” command:

```
#digital
set pin[10]  2

#analog
set apin[20] 2.345
```

For windows users putty telnet client is a good option to access the remote control interface.

3.5 Picture Map Reference

Names used in .map files for boards and parts are standardized and used by the remote control interface.

The names must start with **I_** if it is an input, **O_** if it is an output or **B_** if it is bidirectional. And be followed by one of the two-letter types in the table below before the area name.

Function	Type	Description	RControl In	RControl Out
I	MD	Memory Dump	-	-
I	KB	Keyboard Key	0 or 1	0 or 1
I	PG	Program	-	-
I	CN	Connector	-	-
B	PO	Potentiometer	0 to 200	0 to 200
B	JP	Jumper	0 or 1	0 or 1
B	VS	Value short	-32768 to 32767	-32768 to 32767
B	PB	Push button	0 or 1	0 or 1
B	DP	Dip switch	0 or 1	0 or 1
B	SW	Switch	0 or 1	0 or 1
B	RT	Rotary encoder	0 to 200	0 to 200
B	AJ	Dip switch	-	-
O	MC	Motor Cooler	-	0 to 200
O	PN	Pin name	-	-
O	ST	Status	-	-
O	IC	IC name	-	-
O	LR	LED RGB	-	-
O	LM	LED Matrix	-	-
O	DI	Display Info	-	-
O	LD	LED	-	0 to 200
O	DS	Display	-	if alphanumeric show text
O	MT	DC motor	-	dir 0 or 1, spd. 0 to 200, pos. 0 to 200
O	DG	Degree	-	float angle
O	SS	seven segment	-	decoded number

For example area named **B_PB_Start**, which describes the position of a push button named "Start". The **B_** bidirectional indicates that the mapped area serves as user action input and drawing output.

Chapter 4

Boards

PICSimLab currently supports five backend simulators. The stable version supports picsim and simavr. The experimental version supports uCsim, gpsim and qemu-stm32 in addition to the stable ones.

The Figure 4.1 shows which cards are based on which backend simulator:

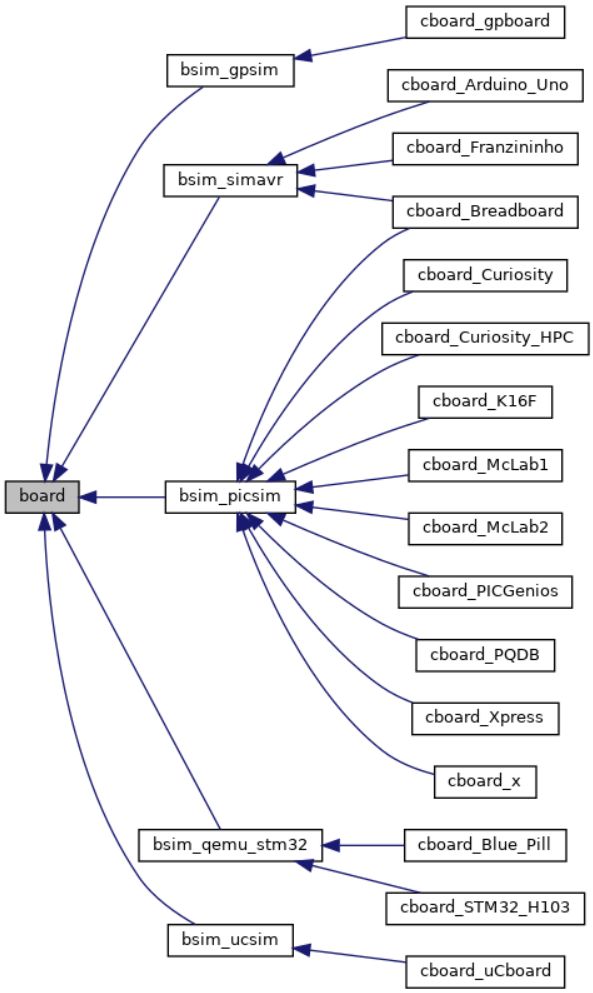


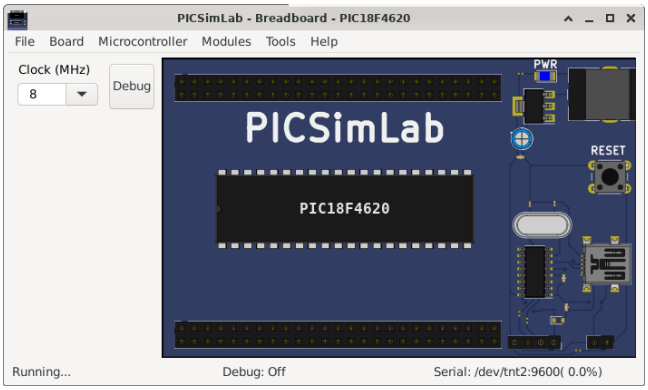
Figure 4.1: Boards backend simulators

The below table show the supported debug interface of each simulator:

Backend	Debug Support
picsim	MPLABX Integrated Debug (see section 7.1)
simavr	MPLABX Integrated Debug (see section 7.1) and remote avr-gdb (see section 7.3)
qemu-stm32	remote arm-gdb (see Chapter 7.4)
uCsim	uCsim remote console (telnet) (see section 7.5)
gpsim	none yet

4.1 Breadboard

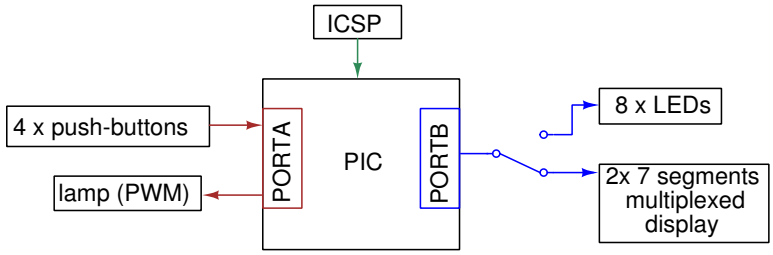
It is a generic board only with reset, serial and crystal circuits and support to multiple microcontrollers of picsim and simavr.

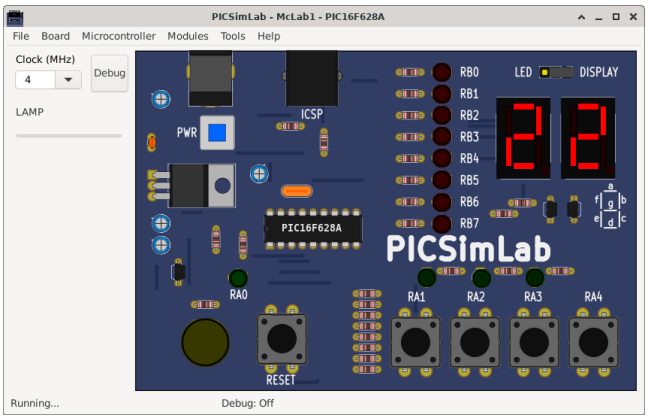


Examples

4.2 McLab1

It emulates the Labtools development board McLab1 that uses one PIC16F84, PIC16F628 or PIC16F648 of picsim.





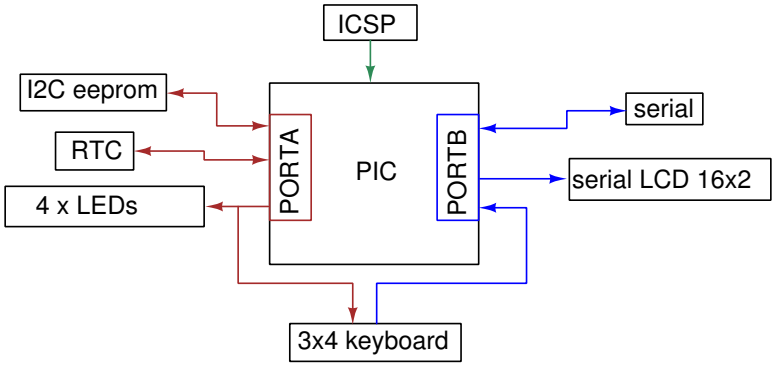
Board McLab1 schematics.

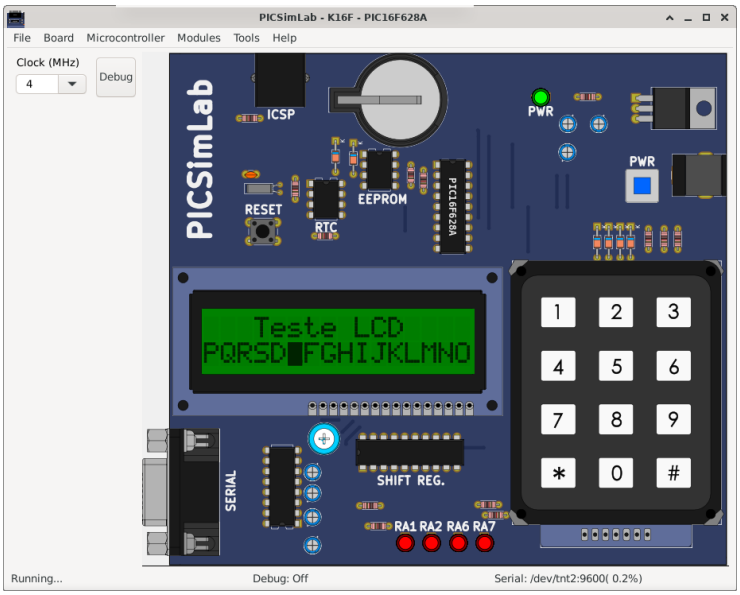
The code examples can be loaded in PICSimLab menu **Help->Examples**.

The source code of board McLab1 examples using MPLABX and XC8 compiler are in the link: board_McLab1.

4.3 K16F

It emulates an didactic board developed by author that uses one PIC16F84, PIC16F628 or PIC16F648 of picsim.





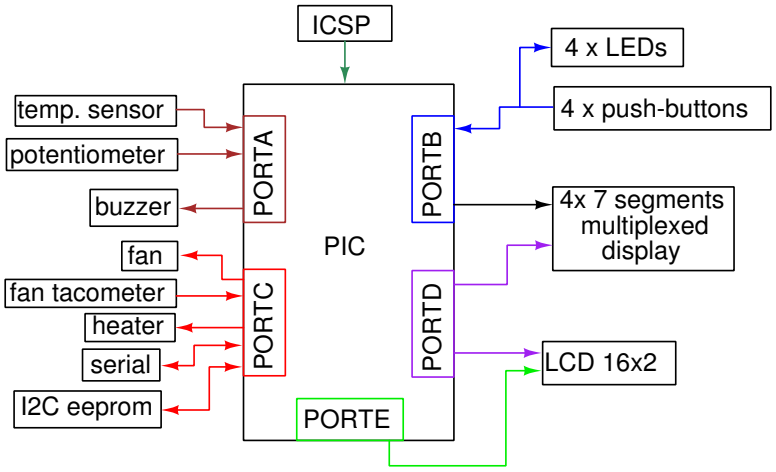
Board K16F schematics.

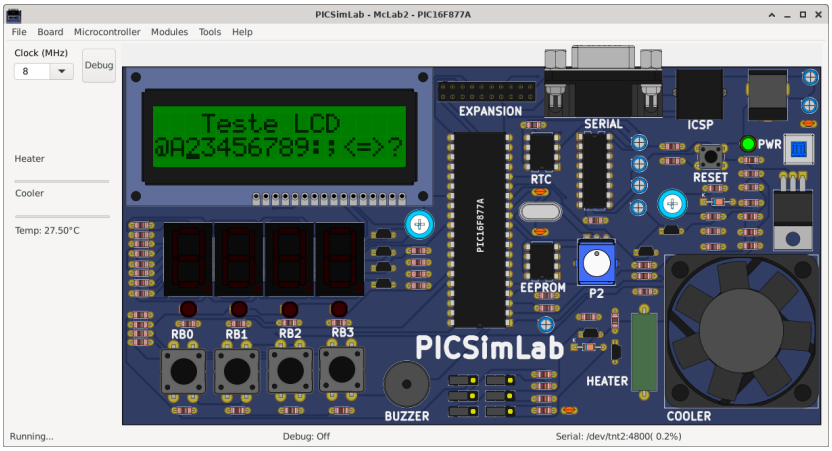
The code examples can be loaded in PICSimLab menu **Help->Examples**.

The source code of board K16F examples using MPLABX and XC8 compiler are in the link: board_K16F.

4.4 McLab2

It emulates the Labtools development board McLab2 that uses one PIC16F777, PIC16F877A, PIC18F452, PIC18F4520, PIC18F4550 or PIC18F4620 of picsim.





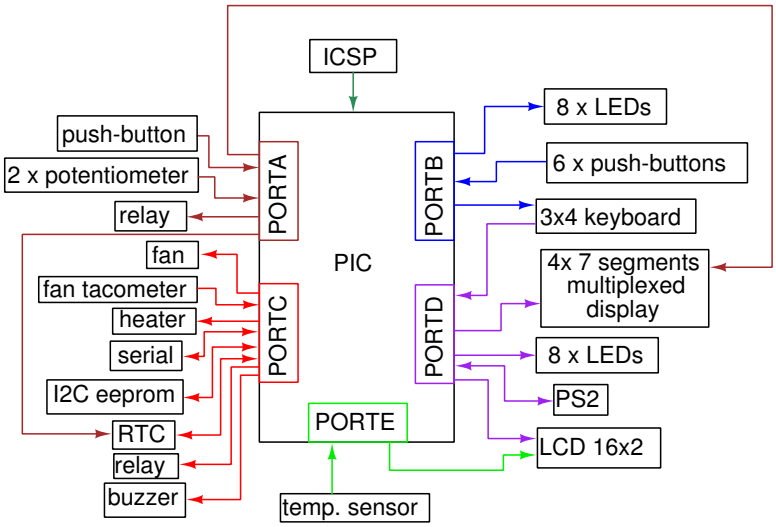
Board McLab2 schematics.

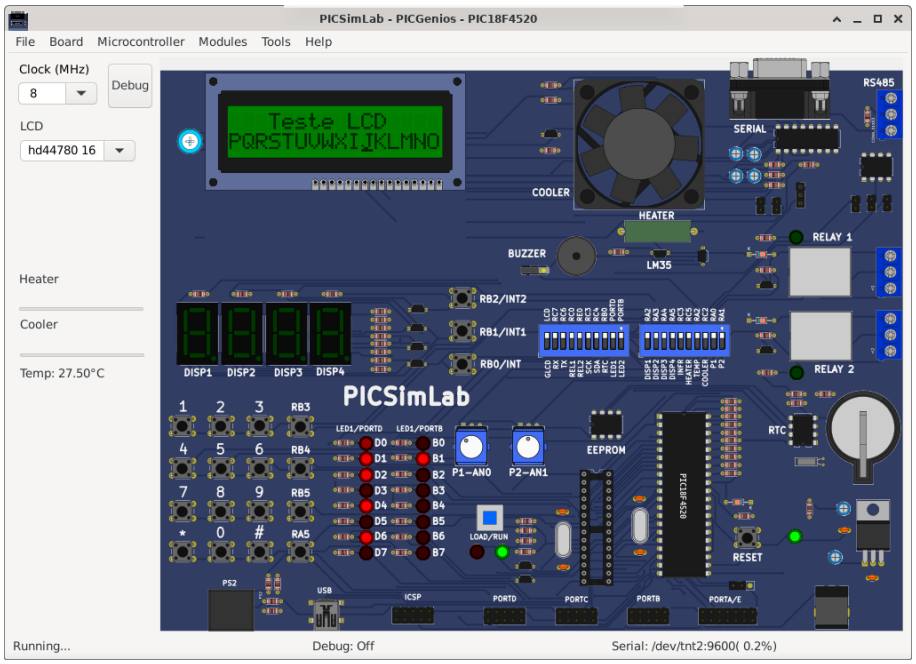
The code examples can be loaded in PICSimLab menu **Help->Examples**.

The source code of board McLab2 examples using MPLABX and XC8 compiler are in the link: board_McLab2.

4.5 PICGenios

It emulates the microgenius development board PICGenios PIC18F e PIC16F Microchip that uses one PIC16F777, PIC16F877A, PIC18F452, PIC18F4520, PIC18F4550 or PIC18F4620 of picsim.





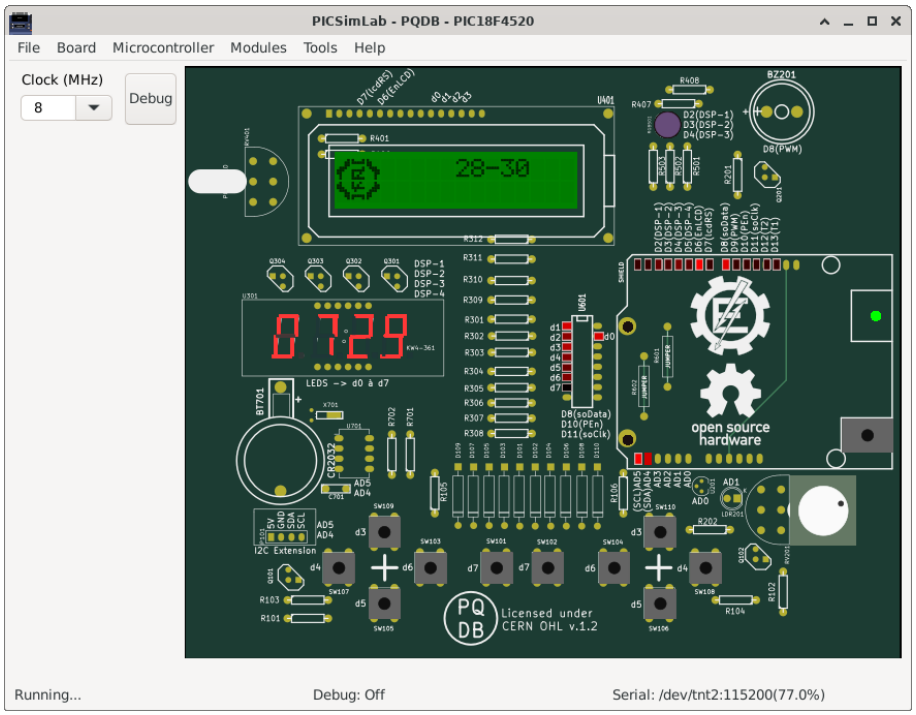
Board PICGenios schematics.

The code examples can be loaded in PICSimLab menu **Help->Examples**.

The source code of board PICGenios examples using MPLABX and XC8 compiler are in the link: board_PICGenios.

4.6 PQDB

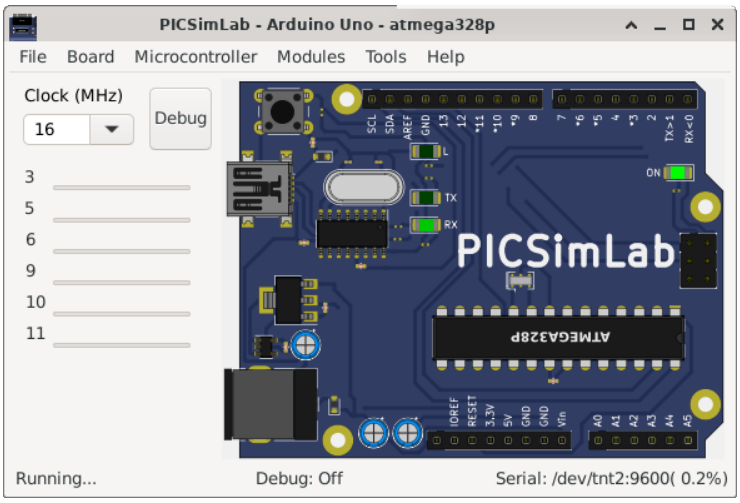
The PQDB board is an opensource/openhardware project, more info at https://github.com/projetopqdb/. It was developed to be used with arduino/freedom boards, but adapted to use the microcontroller PIC18F4520 of picsim on PICSImLab.



Examples

4.7 Arduino Uno

It emulates the Arduino Uno development board that uses one ATMEGA328P microcontroller of simavr.



Board Arduino Uno schematics.

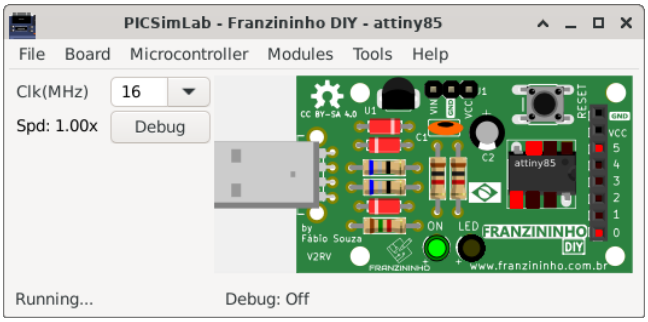
The code examples can be loaded in PICSimLab menu **Help->Examples**.

The source code of board Arduino Uno examples using the Arduino IDE with avr-gcc are in the link: board_Arduino_Uno.

More information about the Arduino in www.arduino.cc

4.8 Franzininho DIY

The Franzininho DIY board is an openhardware project, more info at https://franzininho.com.br/. It was developed to be used with the microcontroller ATtiny85 of of simavr.



Examples

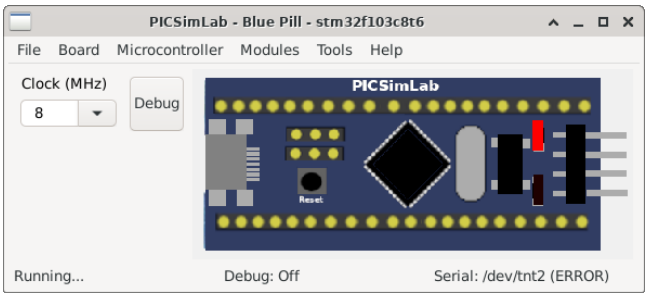
Chapter 5

Experimental Boards

Boards in the experimental phase. Probably with some bugs and missing features.

5.1 Blue Pill

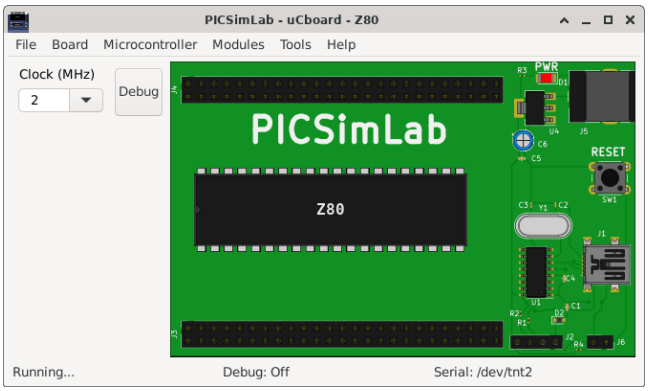
It is a generic board only with reset, serial and crystal circuits and support to stm32f103c8t6 microcontroller of qemu-stm32.



Examples

5.2 uCboard

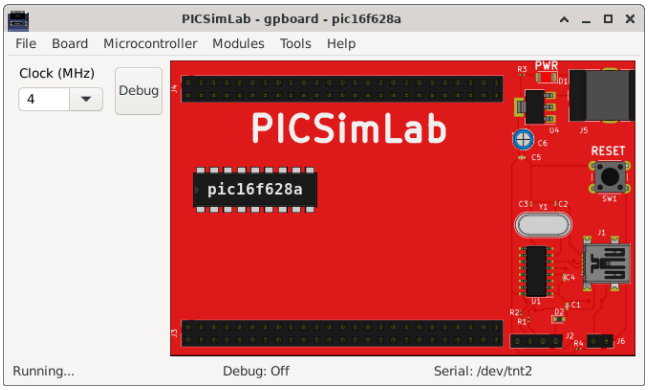
It is a generic board only with reset, serial and crystal circuits and support to multiple microcontrollers (initially C51, Z80 and STM8S103)of uCsim.



Examples

5.3 gpboard

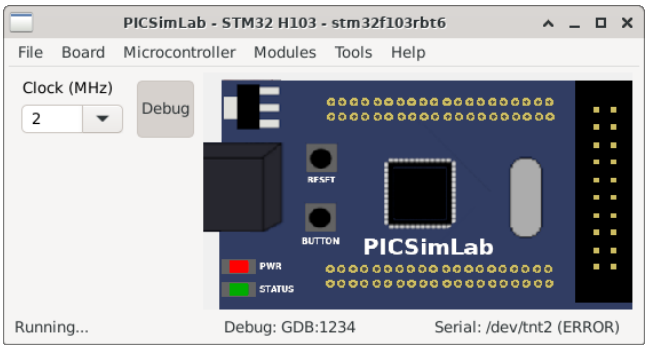
It is a generic board only with reset, serial and crystal circuits and support to multiple microcontrollers of gpsim.



Examples

5.4 STM32 H103

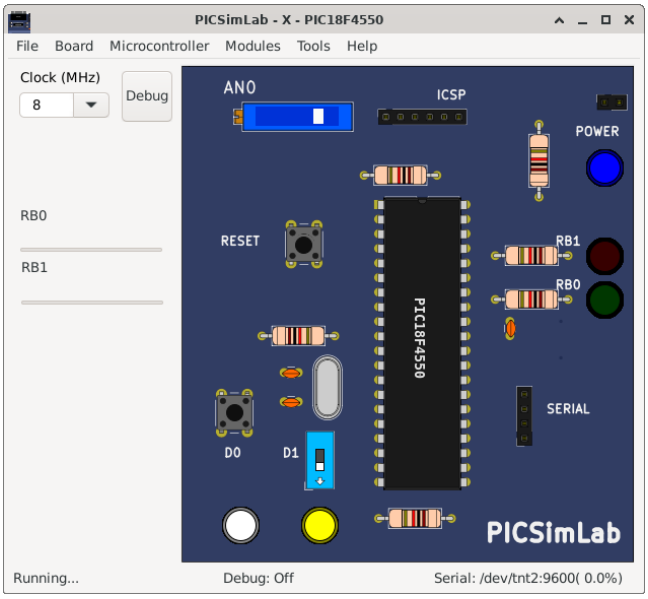
It is a generic board only with reset, one push button, serial and crystal circuits and support to stm32f103rbt6 microcontroller of qemu-stm32.



Examples

5.5 X

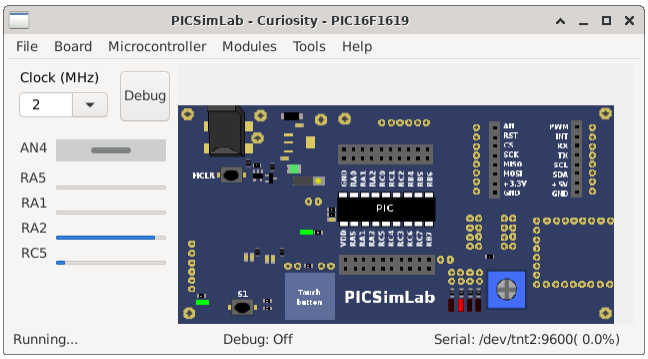
It is a generic board, used as example in How to Compile PICsimLab and Create New Boards.



Examples

5.6 Curiosity

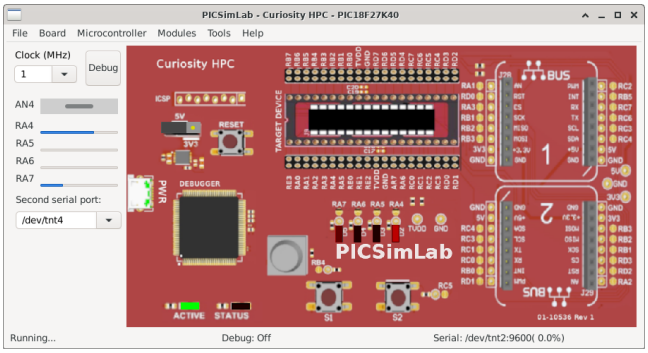
This is a simple PIC microcontroller development board that uses picsim.



Examples

5.7 Curiosity HPC

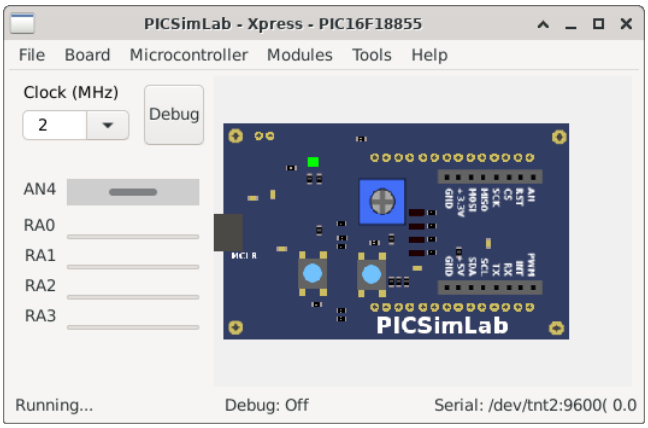
This is a simple PIC microcontroller development board that uses picsim.



Examples

5.8 Xpress

This is a simple PIC microcontroller development board that uses picsim.


PICSimLab - Xpress - PIC16F18855
File
Board
Microcontroller
Modules
Tools
Help
Clock (MHz)
2
Debug
AN4
RA0
RA1
RA2
RA3
PICSimLab
Running...
Debug: Off
Serial: /dev/tnt2:9600(0.0

Examples

Chapter 6

Serial Communication

To use the simulator serial port, install a NULL-MODEM emulator:

- Windows: com0com http://sourceforge.net/projects/com0com/
- Linux: tty0tty https://github.com/lcgamboa/tty0tty

For communication the PICSimLab should be connected in one port of the NULL-MODEM emulator and the other application connected in the other port. Configuration examples linking PICSimLab to Cutecom for serial communication:

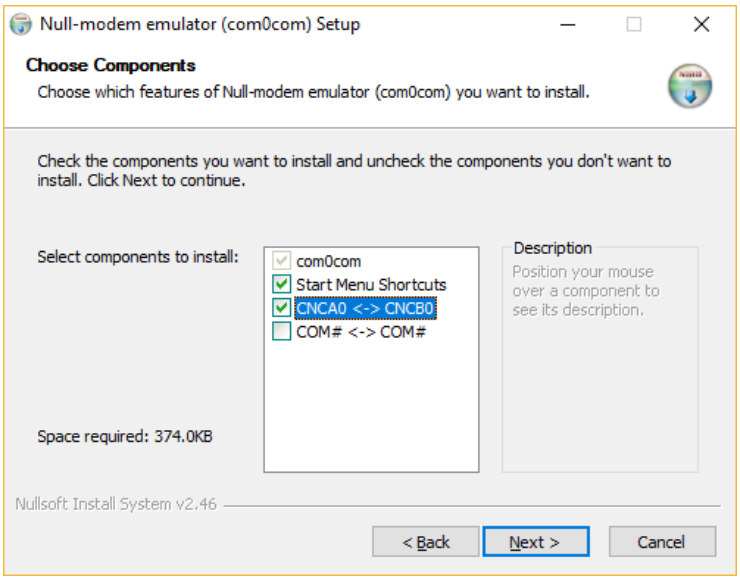
OS	PicsimLab port	Cutecom port	NULL-Modem prog.	Connection
Windows	com1	com2	com0com	com1<=>com2
Linux	/dev/tnt2	/dev/tnt3	tty0tty	/dev/tnt2<=>/dev/tnt3

6.1 Com0com Installation and Configuration(Windows)

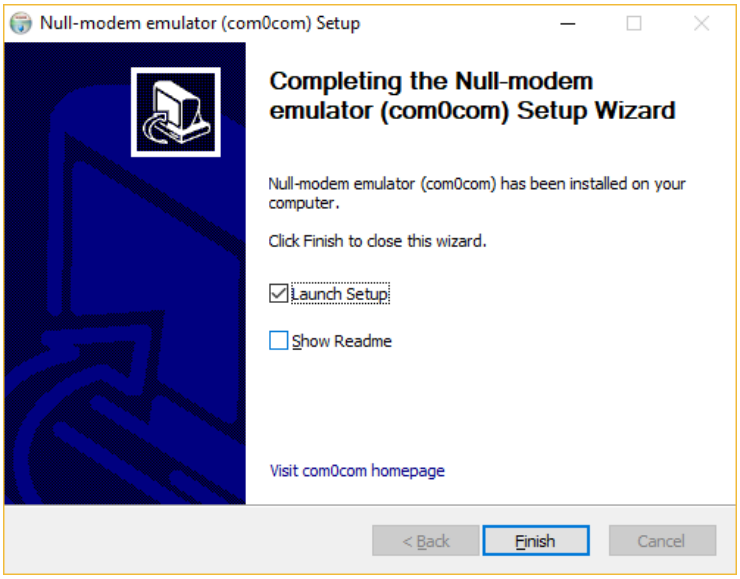
Download the signed version of com0com.

Unzip the downloaded .zip file and run the specific installer of your operating system, x86 for windows 32-bit or x64 for windows 64-bit.

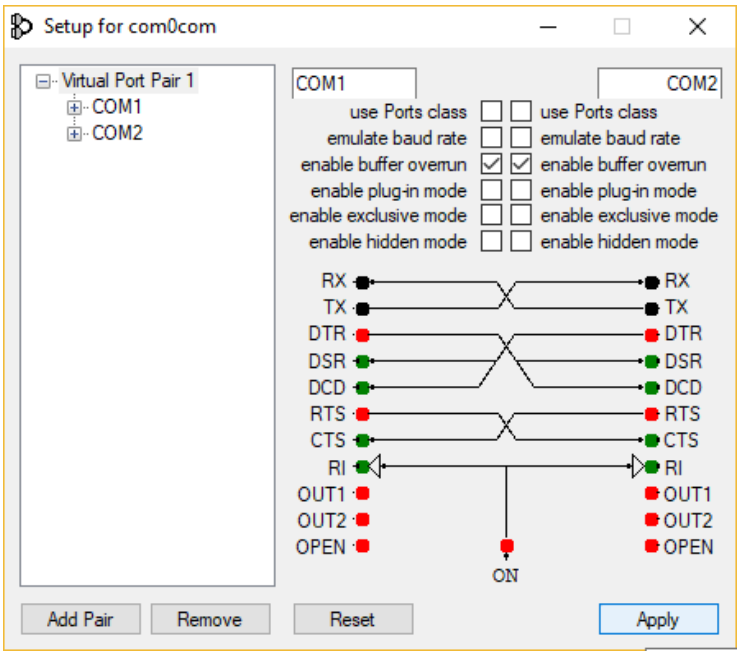
Configure the “choose components” window as the figure below:



In the last configuration window, check the "Launch setup" option:



In the setup window, change the port names to COM1, COM2, COM3 .... Just check the "enable buffer overrun" option on the two ports, click in the "Apply" button and close the setup. In the configuration shown in the figure below, the COM1 and COM2 ports form a NULL-MODEM connection, where one port must be used by the PICSimLab and another by the application with serial communication.



6.2 tty0tty Installation and Configuration (Linux)

Download the tty0tyy. Unzip the downloaded folder.

Open a terminal and enter in the tty0tty/module/ folder and enter the following commands:

```
sudo apt-get update
sudo apt-get -y upgrade
sudo apt-get -y install gcc make linux-headers-`uname -r`
sudo ./dkms-install.sh
sudo modprobe tty0tty
```

The user must be in the **dialout** group to access the ports. To add your user to **dialout** group use the command:

```
sudo usermod -a -G dialout your_user_name
```

after this is necessary logout and login to group permissions take effect.

Once installed, the module creates 8 interconnected ports as follows:

```
/dev/tnt0  <=>  /dev/tnt1
/dev/tnt2  <=>  /dev/tnt3
/dev/tnt4  <=>  /dev/tnt5
/dev/tnt6  <=>  /dev/tnt7
```

the connection between each pair is of the form:

```
TX   ->  RX
RX   <-  TX
RTS  ->  CTS
CTS  <-  RTS
DSR  <-  DTR
CD   <-  DTR
DTR  ->  DSR
DTR  ->  CD
```

Any pair of ports form a NULL-MODEM connection, where one port must be used by the PICSimLab and another by the application with serial communication.

Chapter 7

Debug Support

The type of debug interface depends on the backend simulator utilized.

7.1 MPLABX Integrated Debug (picsim and simavr)

To use the MPLABX IDE for debug and program the PicsimLab, install the plugin com-picsim-picsimlab.nbm in MPLABX.

The plugin connect to Picsimlab through a TCP socket using port 1234 (or other defined in configuration window), and you have to allow the access in the firewall.

Tutorial: how to use MPLABX to program and debug PICsimLab.

It's possible import and debug a Arduino sketch into MPLABX using the Arduino import plugin.

7.2 Arduino IDE Integration (simavr)

For integrated use with the Arduino IDE, simply configure the serial port as explained in the section 6 and load the Arduino bootloader. The bootloader can be loaded from the "Tools->Arduino bootloader" menu.

In Windows, considering com0com making a NULL-MODEM connection between COM1 and COM2, simply connect the PICSimLab on the COM1 port (defined in configuration window) and the Arduino IDE on the COM2 port or vice versa.

On Linux the operation is the same, but using for example the ports /dev/tnt2 and /dev/tnt3.

In Linux for the virtual ports to be detected in Arduino it is necessary to replace the library lib/liblistSerialsj.so of the Arduino with a version which support the detection of tty0tty ports, that can be downloaded in the link listSerialC with tty0tty support.

7.3 avr-gdb Debug (simavr)

With debug support enabled you can use avr-gdb to debug the code used in the simulator. Use the configuration window to choose between MDB (MPLABX) or GDB to debug AVR microcontrollers.

Use avr-gdb with the .elf file as the parameter:

```
avr-gdb compiled_file.elf
```

and the command below to connect (1234 is the default port):

```
target remote localhost:1234
```

Graphic debug mode can be made using eclipse IDE with Sloeber Arduino plugin.

7.4 arm-gdb Debug (qemu-stm32)

With debug support enabled you can use arm-none-eabi-gdb (or gdb-multiarch) to debug the code used in the simulator.

Use arm-none-eabi-gdb with the .elf file as the parameter:

```
arm-none-eabi-gdb compiled_file.elf
```

and the command below to connect (1234 is the default port):

```
target remote localhost:1234
```

Graphic debug mode can be made using eclipse IDE with Eclipse Embedded CDT.

7.5 uCsim Debug

The uCsim debug console can be accessed with the telnet (1234 is the default port):

```
telnet localhost 1234
```

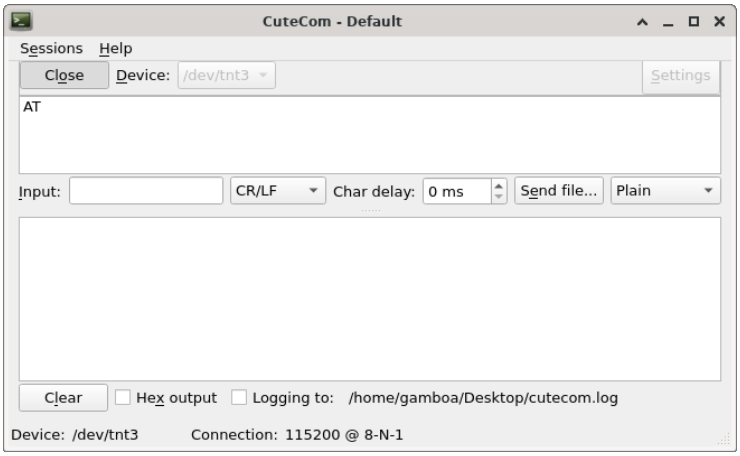
All uCsim commands are supported.
For windows users putty telnet client is a good option to access the uCsim console.

Chapter 8

Tools

8.1 Serial Terminal

Open the serial terminal Cutecom.



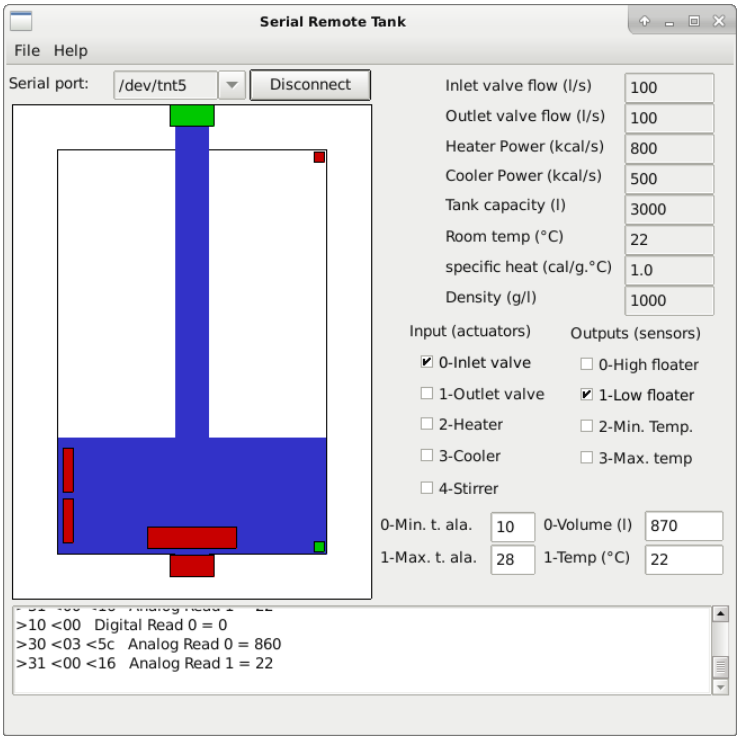
A serial terminal is used to send and receive data over a serial communication channel. The use of this terminal can be replaced by others like the Arduino IDE serial monitor.

To use this tool with PICSimLab you first need to configure a virtual serial port as described in Chapter: Serial Communication. It is possible to use this tool with a real serial port connected to a real device.

8.2 Serial Remote Tank

The serial remote tank is a tank simulator controlled by a serial communication protocol. The tank has several sensors and actuators that can be read and controlled using

the communication protocol. The parameters of the serial communication port must be 19200 8N1.



To use this tool with PICSimLab you first need to configure a virtual serial port as described in Chapter: Serial Communication. It is possible to use this tool with a real serial port connected to a real device.

8.2.1 Actuators

Digital inputs:

1. Inlet valve
2. Outlet valve
3. Heater
4. Cooler
5. Stirrer

Analog inputs:

1. Minimal temperature alarm trigger level
2. Maximal temperature alarm trigger level

8.2.2 Sensors

Digital outputs:

1. High floater
2. Low floater
3. Minimal temperature
4. Maximal temperature

Analog outputs:

1. Volume
2. Temperature

8.2.3 Communication Protocol

Writing on Digital Input

Sent one byte in 0x0N hexadecimal format where N is the number of input followed by a second byte with value 0x00 for disable or 0x01 for enable.

Example to turn on the input 2:

```
Serial_write(0x02);
Serial_write(0x01);
```

Reading Digital Output

Sent one byte in 0x1N hexadecimal format where N is the number of output and read one byte. The byte readed have value 0x00 for disable or 0x01 for enable.

Example to read output 3:

```
Serial_write(0x13);
valor=Serial_read(0);
```

Writing on Analog Input

Sent one byte in 0x2N hexadecimal format where N is the number of input followed by two bytes with the 16 bits value.

Example to write the value 230 on analog input 1:

```
Serial_write(0x21);
valor=230;
Serial_write((valor&0xFF00)>>8);
Serial_write(valor&0x00FF);
```

Reading Analog Output

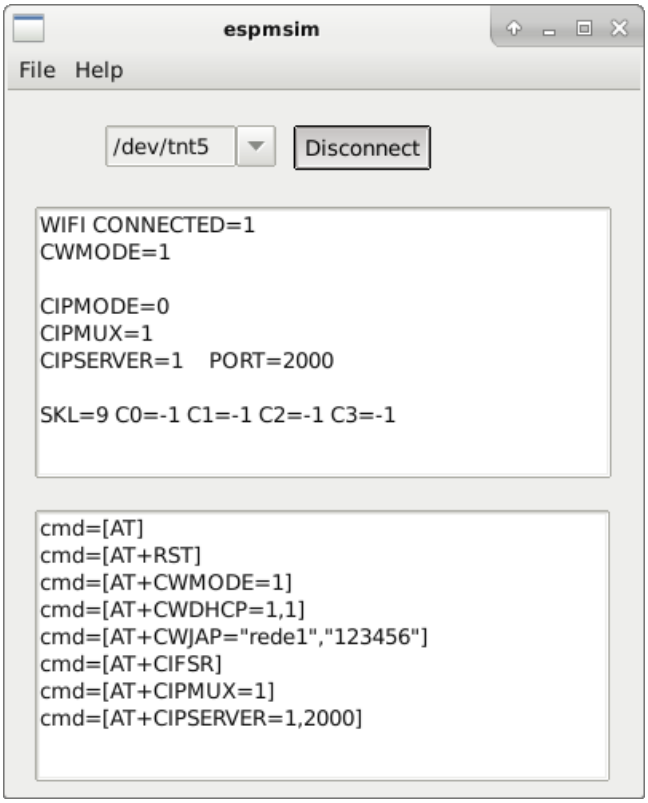
Sent one byte in 0x3N hexadecimal format where N is the number of output and read two bytes to form the 16 bits value.

Example to read analog output 2:

```
Serial_write(0x32);
valorh=Serial_read(0);
valorl=Serial_read(0);
valor=(valorh<<8)|valorl;
```

8.3 Esp8266 Modem Simulator

The ESP8266 modem simulator emulates the operation of an esp8266 with wifi modem firmware. Communication is done using a serial channel via AT modem commands. The parameters of the serial communication port must be 115200 8N1.



To use this tool with PICSimLab you first need to configure a virtual serial port as described in Chapter: Serial Communication. It is possible to use this tool with a real

serial port connected to a real device.

8.3.1 Supported Commands

- AT
- AT+RST
- AT+GMR
- AT+CWMODE=1
- AT+CWDHCP=1,1
- AT+CWLAP
- AT+CWJAP="rede1","123456"
- AT+CIFSR
- AT+CIPMUX=1
- AT+CIPSERVER=1,2000
- AT+CIPSEND=0,10
- AT+CIPCLOSE=0

8.4 Arduino Bootloader

This menu option load PICSimLab microcontroller with Arduino serial bootloader. The microcontroller with the bootloader loaded can be programmed directly by the Arduino IDE or using the avrdude program.

To use this tool with PICSimLab you first need to configure a virtual serial port as described in Chapter: Serial Communication.

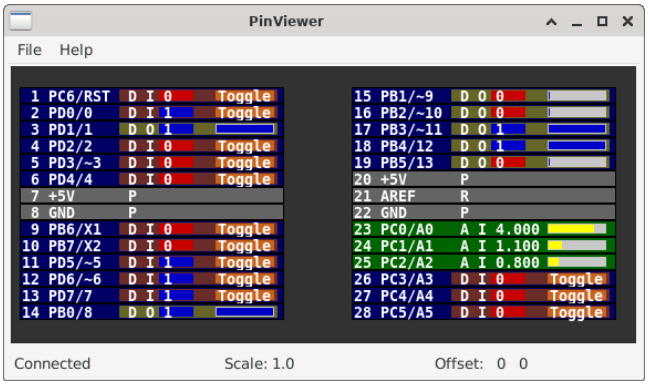
8.5 MPLABX Debugger Plugin

This menu option open the web page to download the MPLABX Debugger Plugin.

The plugin must be installed on MPLABX to allow debugging and programming PICSimLab (PICs and AVRs) from the IDE, like a real tool for debugging and programming.

8.6 Pin Viewer

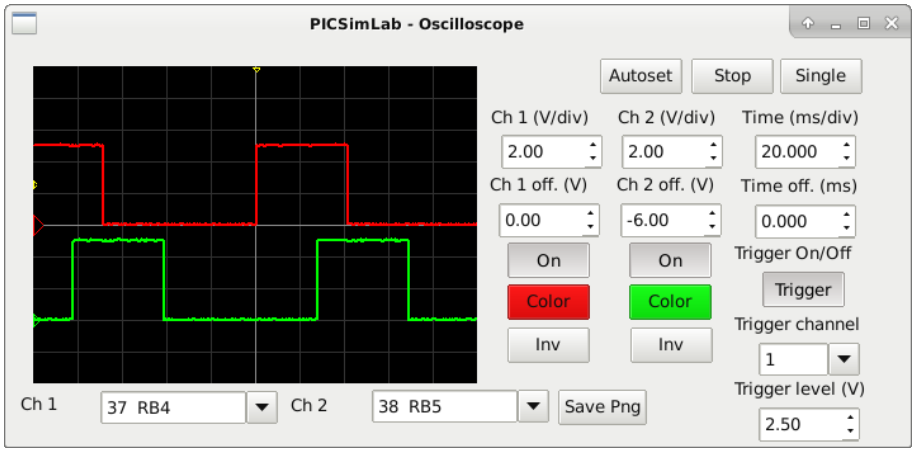
The PinViewer connects to PICSimLab through the rcontrol interface and allows viewing the status and direction of all microcontroller pins. It is also possible to change the state of the digital pins and adjust the voltage value on the analog pins configured as input. Pins configured as outputs also show the average value, useful for evaluating the functioning of PWM outputs.



Chapter 9

Oscilloscope

The PICSimLab has a basic two-channel oscilloscope that can be used to view the signal on any pin of the microcontroller. The oscilloscope can be accessed through the "Modules->Oscilloscope" menu.



Chapter 10

Spare Parts

The PICSimLab has a window that allows the connection of spare parts to the microcontroller, it can be accessed through the menu " Modules-> Spare parts ".

The main window has the menu with the following functions:

- File
  - New configuration - Clear the spare parts window
  - Save configuration - Saves the current settings of the spare parts into .pcf file
  - Load configuration - Loads the settings from .pcf file
  - Save pin alias - Saves the current pin alias to .ppa text file
  - Load pin alias - Loads the pin alias from .ppa file
- Edit
  - Clear pin alias - Clear the pin alias
  - Toggle pin alias - Enable/Disable pin alias use
  - Edit pin alias - Open current pin alias .ppa file in text editor
  - Reload pin alias - Reload the current .ppa pin alias file (need after edit .ppa file)
  - Zoom in - Increase draw scale
  - Zoom out - Decrease draw scale
- Inputs
  - Encoder - Adds a rotary quadrature encoder with push button
  - FM50 (Temperature) - Adds a analog temperature sensor
  - Fixed Voltage - Adds a analog fixed voltage reference
  - Gamepad - Adds a gamepad

  - Gamepad (Analogic) - Adds a gamepad with one analogic output
  - Keypad - Adds one matrix keypad
  - LM35 (Temperature) - Adds a analog temperature sensor
  - MPU6050 - Adds a accelerometer and gyroscope (only raw values)
  - Potentiometers - Adds 4 potentiometers
  - Potentiometers (Rotary) - Adds 4 rotary potentiometers
  - Push Buttons - Adds 8 push buttons
  - Push Buttons (Analogic) - Adds 8 push buttons with analog output
  - SHT3X - Adds a analog temperature and humidity sensor
  - Switchs - Adds eight switchs
  - Ultrasonic HC-SR04 - Adds a ultrasonic range sensor
- Outputs
  - 7 Segments Display - Adds four multiplexed 7 segments displays
  - 7 Segments Display (w/dec) - Adds four multiplexed 7 segments displays with decoder
  - Buzzer - Adds a active/passive buzzer
  - DC Motor - Adds a DC motor with H-bridge and quadrature encoder
  - LCD hd44780 - Adds a text display hd44780
  - LCD ili9340 - Adds a color graphic display ili9340 with touchscreen
  - LCD pcd8544 - Adds a monochrome graphic display pcd8544 (Nokia 5110)
  - LCD pcf8833 - Adds a color graphic display pcf8833
  - LCD ssd1306 - Adds a monochrome graphic display ssd1306
  - LED Matrix - Adds a 8x8 LED matrix with MAX72xx controller
  - LEDs - Adds 8 red LEDs
  - RGB LED - Adds one RGB LED
  - Servo Motor - Adds a servo motor
  - Step Motor - Adds a step motor
- Others
  - ETH w5500 - Adds a ethernet shield w5500
  - IO 74xx595 - Adds a 74xx595 SIPO 8 bit shift register
  - IO MCP23S17 - Adds a MCP23S17 serial SPI IO expander
  - IO PCF8574 - Adds a PCF8574 serial I2C IO expander
  - IO UART - Adds a UART serial port
  - Jumper Wires - Adds sixteen jumper wires

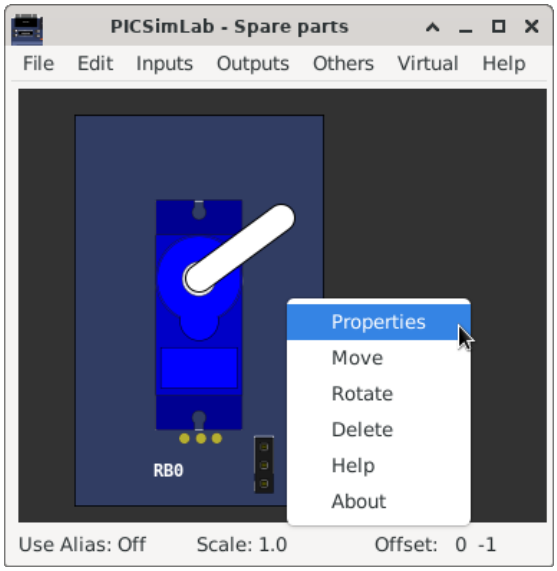
  - MEM 24CXXX - Adds a 24CXXX serial I2C EEPROM memory
  - RTC ds1307 - Adds a ds1307 real time clock
  - RTC pfc8563 - Adds a pfc8563 real time clock
  - SD Card - Adds a SD card shield
  - Temperature System - Adds a temperature control system

- Virtual
  - D. Transfer Function - Adds a discrete transfer function mathematical model
  - IO Virtual term - Adds a virtual serial terminal
  - Signal Generator - Adds a virtual signal generator
  - VCD Dump - Adds a digital value file dump recorder
  - VCD Dump (Analogic) - Adds a analog value file dump recorder
  - VCD Play - Adds a digital value file dump player

- Help
  - Contents - Open Help window
  - About - Show message about author and version

After adding the part, with a right click of the mouse you can access the options menu of the part with the options:

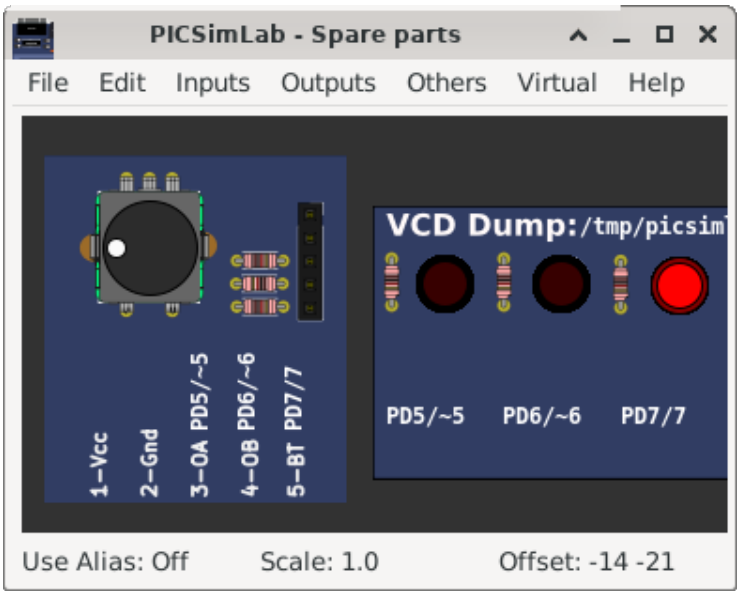
- Properties - Opens the connection settings window
- Move - Unlocks the part to move
- Rotate - Change the orientation of part
- Delete - Remove part
- Help - Open Help window of part
- About - Show message about author and version of part



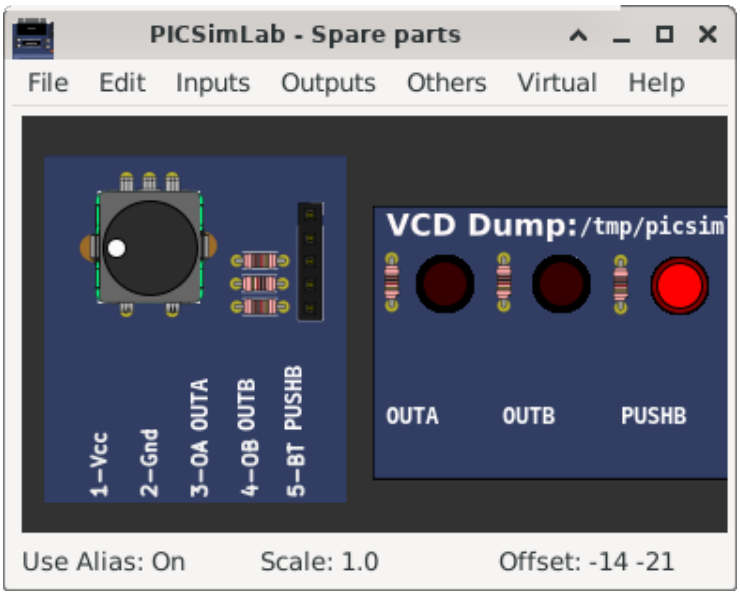
10.1 Pin Alias

The pin alias support allows the user to place custom names on the pins making it easy to identify according to the project.

When off the normal names are shown:



When on the alias names are shown:



To use:

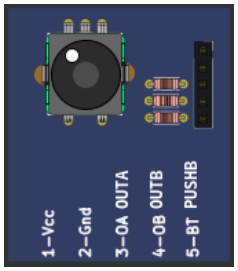
1. active the menu “Edit->Clear pin alias” to reset the pin alias file
2. active the menu “Edit->Edit pin alias” to open pin alias file, change the names, save and close.

3. active the menu “Edit->Reload pin alias” to load new alias
4. active the menu “Edit->Toggle pin alias” to show new alias

10.2 Inputs

10.2.1 Encoder

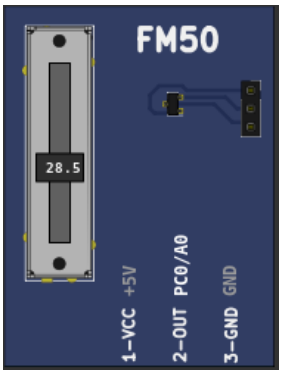
This part is a rotary quadrature encoder with push button. The output is twenty pulses per revolution.



Examples

10.2.2 FM50 (Temperature)

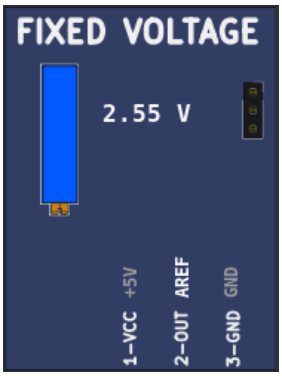
This part is FM50 analog temperature sensor. The measurement range is -40 to 125 °C and voltage output is 10mV/°C + 500mV.



Examples

10.2.3 Fixed Voltage

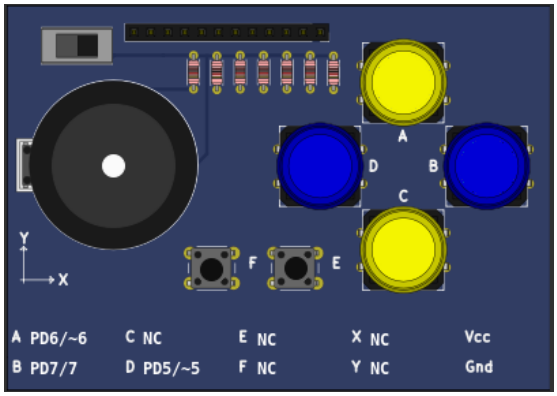
This part is analog fixed voltage reference. The value range is 0 to 5V.

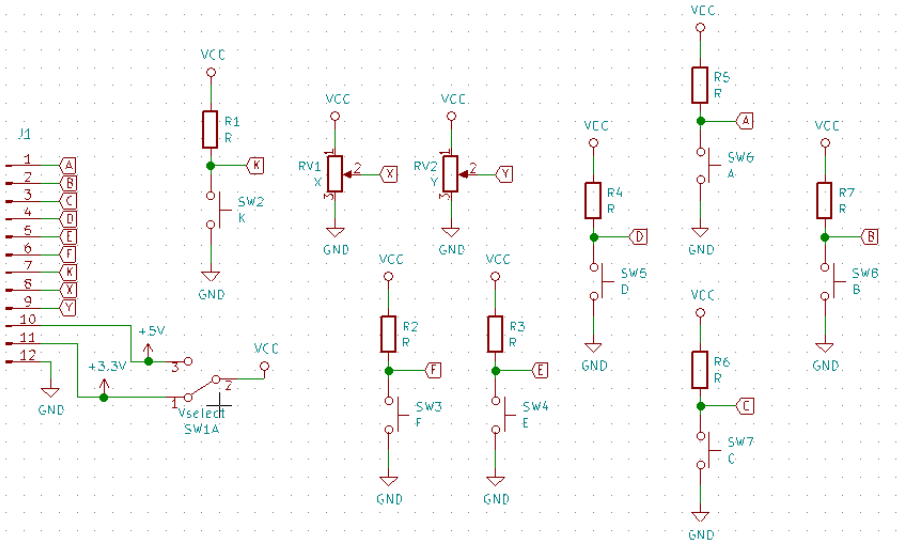


Examples

10.2.4 Gamepad

This part is a gamepad with two analog axis and 7 push buttons.





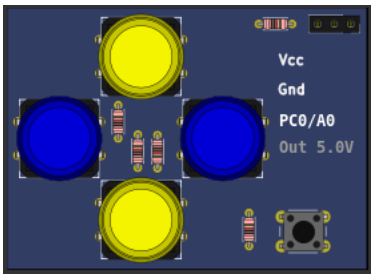
The gamepad can be controlled by keyboards keys:

- X axis - keys 'A' and 'D'
- Y axis - keys 'W' and 'S'
- Button A - key 'I'
- Button B - key 'L'
- Button C - key 'K'
- Button D - key 'J'
- Button E - key 'E'
- Button F - key 'O'
- Button K - key 'R'

Examples

10.2.5 Gamepad (Analogic)

This part is a gamepad with 5 push buttons and one analogic output.



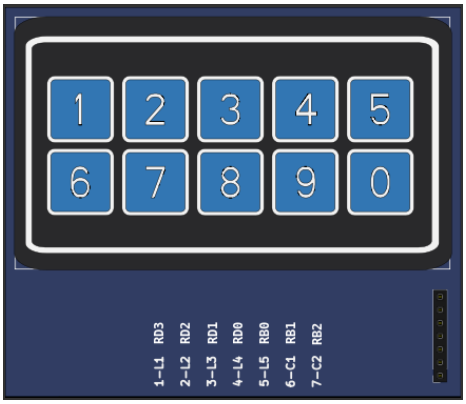
The gamepad can be controlled by keyboards keys:

- Button A - key 'L'
- Button B - key 'I'
- Button C - key 'K'
- Button D - key 'J'
- Button E - key 'O'

Examples

10.2.6 Keypad

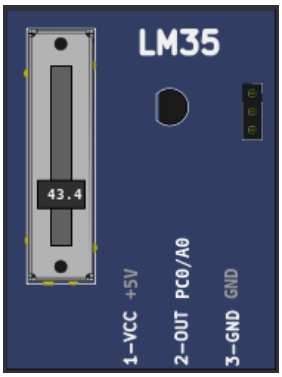
It is a matrix keyboard configurable to 4x3 , 4x4 or 2x5 rows/columns.



Examples

10.2.7 LM35 (Temperature)

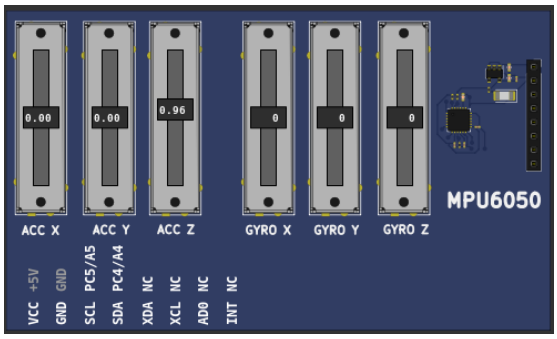
This part is LM35 analog temperature sensor. The measurement range is 2 to 150 °C and voltage output is 10mV/°C.



Examples

10.2.8 MPU6050

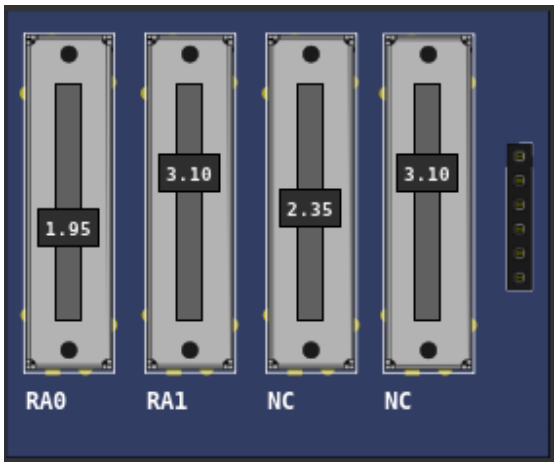
This part is MPU6050 accelerometer and gyroscope with I2C interface. Only raw values are available, DMP is not supported.

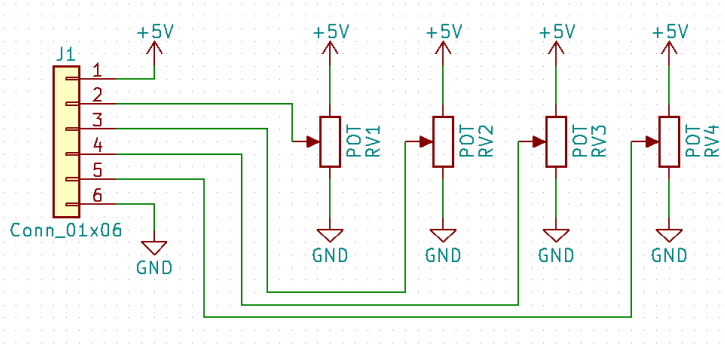


Examples

10.2.9 Potentiometers

This part is formed by 4 potentiometers connected between 0 and 5 volts, the output is connected to the cursor and varies within this voltage range.

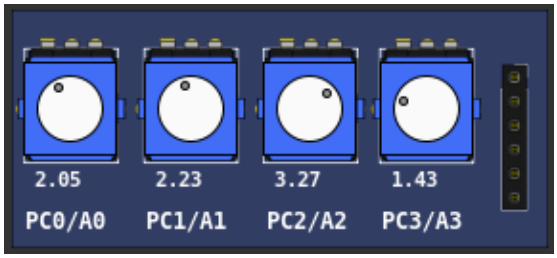


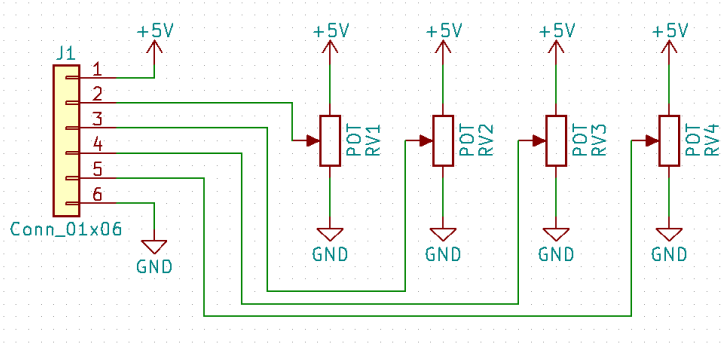


Examples

10.2.10 Potentiometers (Rotary)

This part is formed by 4 rotary potentiometers connected between 0 and 5 volts, the output is connected to the cursor and varies within this voltage range.

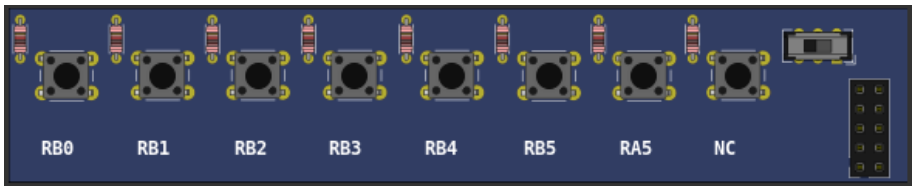


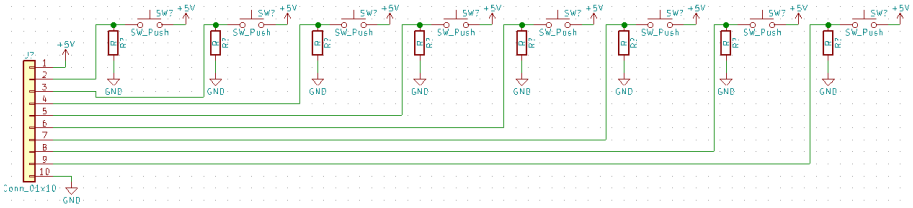


Examples

10.2.11 Push Buttons

This part consists of 8 push buttons. The output active state can be configurable.

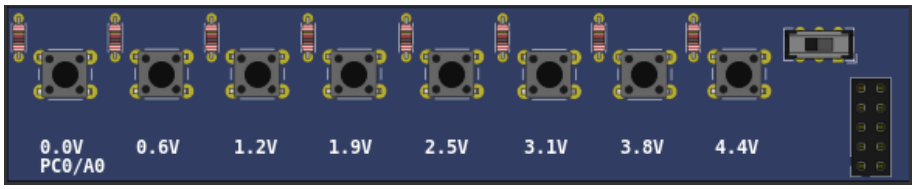




Examples

10.2.12 Push Buttons (Analogic)

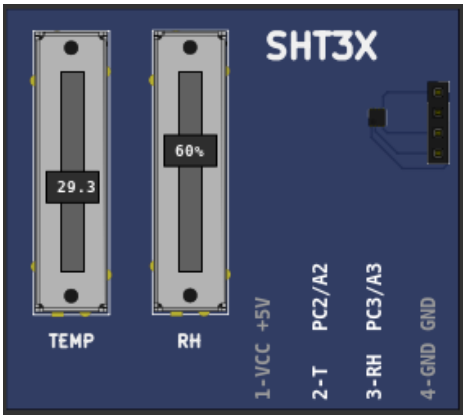
This part consists of 8 push buttons connected in a resistive ladder.



Examples

10.2.13 SHT3X (Temp. Hum.)

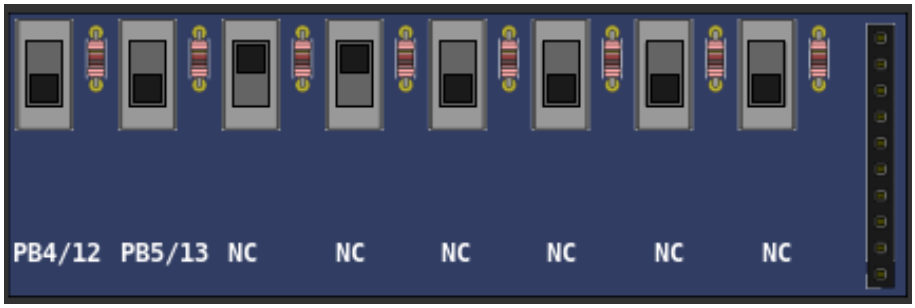
This part is SHT3X analog temperature and humidity sensor. The temperature range is -40 to 125 °C and voltage output is 22.85mV/°C + 1.53V . The relative humidity range is 0 to 100 % and voltage output is 40mV/% + 500mV.

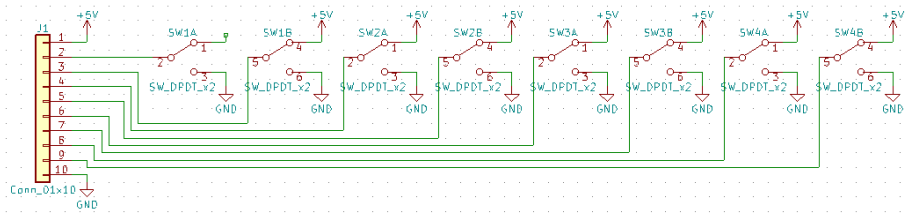


Examples

10.2.14 Switchs

This part consists of 8 keys with on or off position (0 or 1).





Examples

10.2.15 Ultrasonic HC-SR04

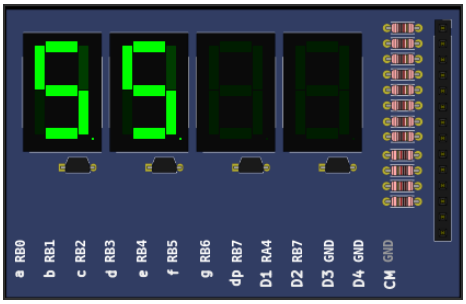
This part is ultrasonic range meter sensor.

Examples

10.3 Outputs

10.3.1 7 Segments Display

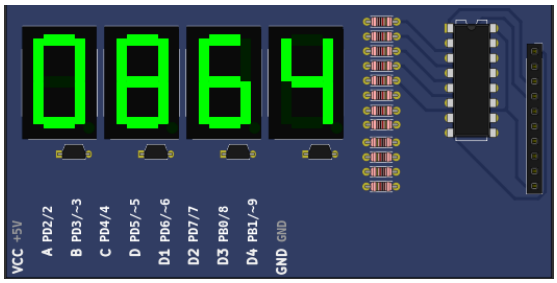
This is a four multiplexed 7 segments displays.

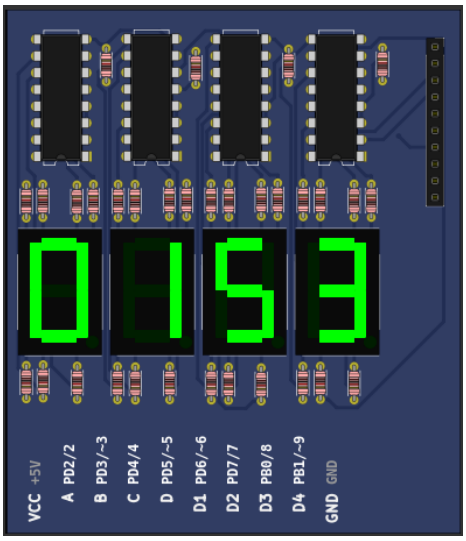


Examples

10.3.2 7 Segments Display (Decoder)

This is a four multiplexed 7 segments displays with BCD to 7 segments decoder (CD4511).





Examples

10.3.3 Buzzer

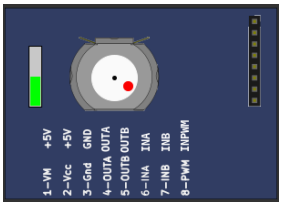
This is a active/passive buzzer. The buzzer has 3 operating modes:

- **Active**: When powered, the buzzer emits a frequency of 440Hz.
- **Passive**: In this mode the values read from the input pin are sent directly to the sound card, this mode only works well if the simulation is in real time.
- **Tone**: This second passive mode measures the frequency on the input and updates the frequency of the buzzer every 100ms, its accuracy is less than the normal passive mode but it works better when the simulation is not in real time.

Examples

10.3.4 DC Motor

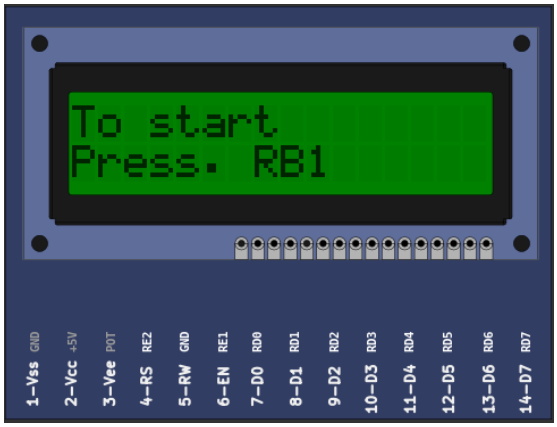
This part is DC motor with H-bridge driver and quadrature encoder.

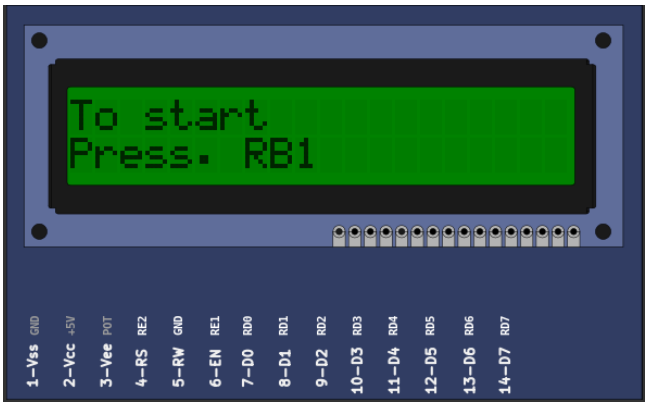


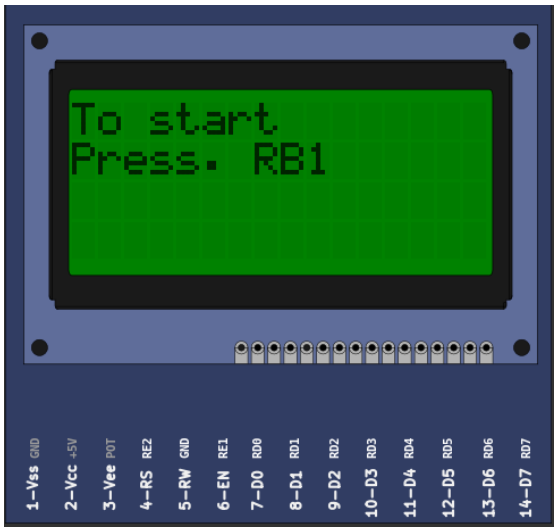
Examples

10.3.5 LCD hd44780

This part is a text display with 2 (or 4) lines by 16 (or 20) columns.



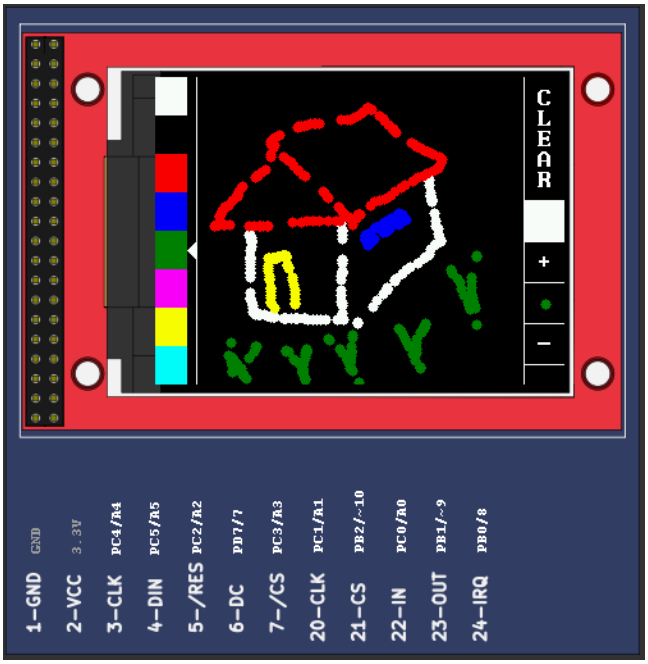

To start
Press. RB1
1-Vss GND
2-Vcc +5V
3-Vee POT
4-RS RE2
5-RW GND
6-EN RE1
7-D0 RD0
8-D1 RD1
9-D2 RD2
10-D3 RD3
11-D4 RD4
12-D5 RD5
13-D6 RD6
14-D7 RD7


To start
Press. RB1
1-Vss GND
2-Vcc +5V
3-Vee POT
4-RS RE2
5-RW GND
6-EN RE1
7-D0 RD0
8-D1 RD1
9-D2 RD2
10-D3 RD3
11-D4 RD4
12-D5 RD5
13-D6 RD6
14-D7 RD7

Examples

10.3.6 LCD ili9341

This part is a color graphic display with 240x320 pixels with touchscreen (xpt2046 controller). Only 4 SPI mode and 8 bits parallel mode is avaliable.



Examples

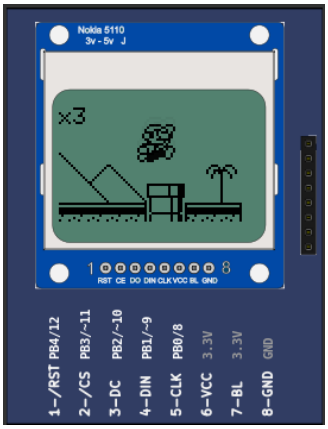
10.3.7 LCD pcf8833

This part is a color graphic display with 132x132 pixels.

Examples

10.3.8 LCD pcd8544

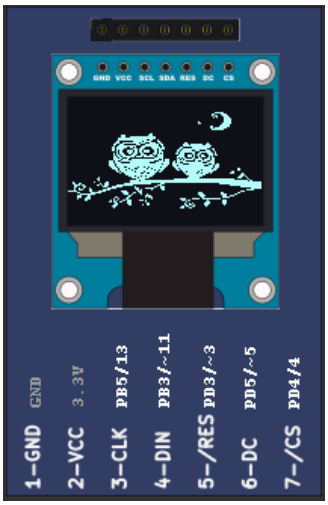
This part is a monochrome graphic display with 48x84 pixels. (Nokia 5110)



Examples

10.3.9 LCD ssd1306

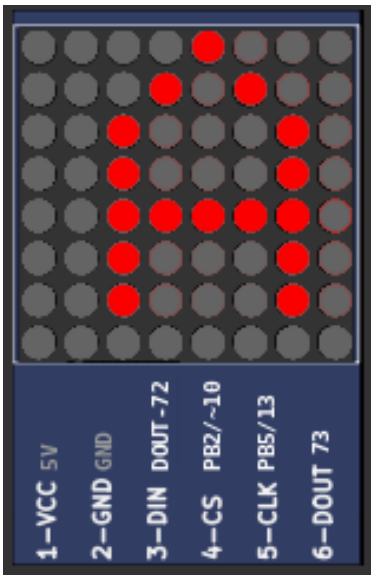
This part is a monochrome oled graphic display with 128x64 pixels. The part suport I2C and 4 SPI serial mode.



Examples

10.3.10 LED Matrix

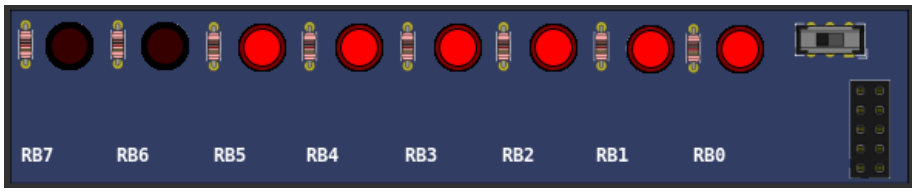
It is a 8x8 LED matrix with MAX72xx controller.

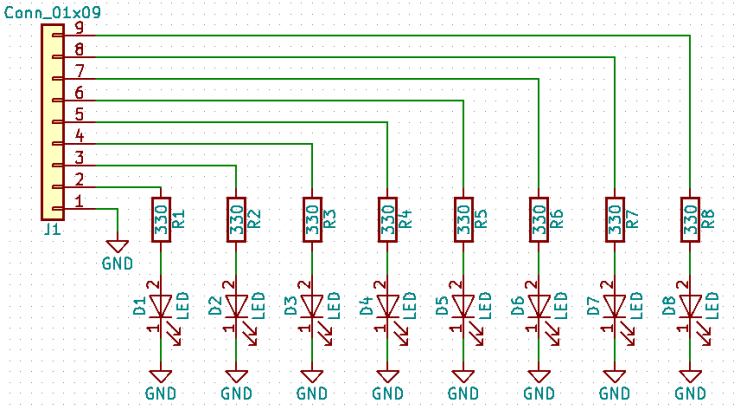


Examples

10.3.11 LEDs

This part is a bar of 8 independent red LEDs.

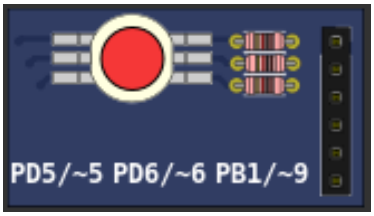


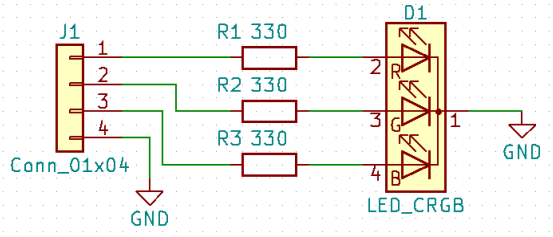


Examples

10.3.12 RGB LED

This part consists of a 4-pin RGB LED. Each color can be triggered independently. Using PWM it is possible to generate several colors by combining the 3 primary colors.

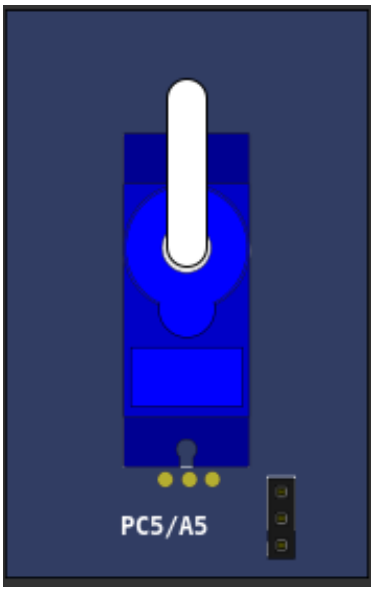


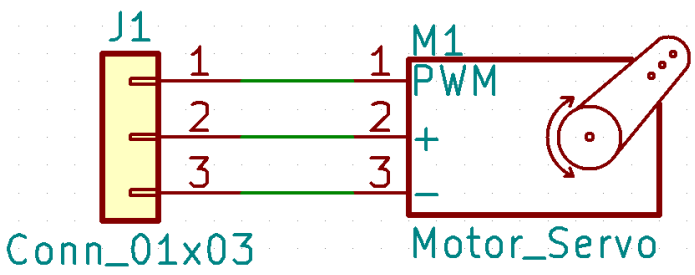


Examples

10.3.13 Servo Motor

The servo motor is a component that must be activated with a pulse of variable width from 1ms to 2ms every 20 ms. A pulse of 1ms positions the servo at -90º, one from 1.5ms to 0º and one from 2ms to 90º.

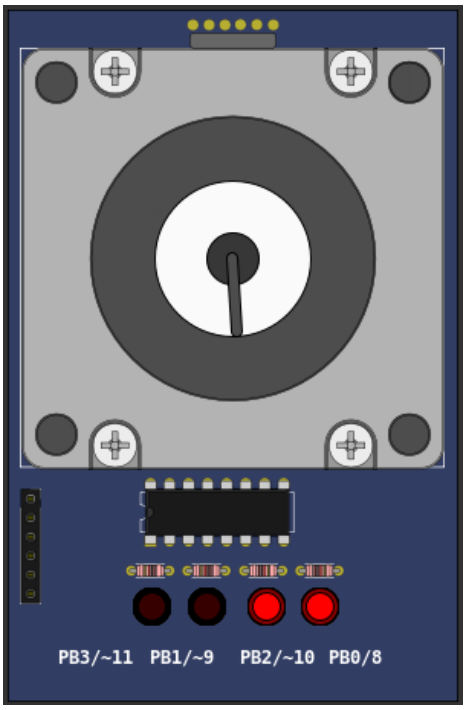


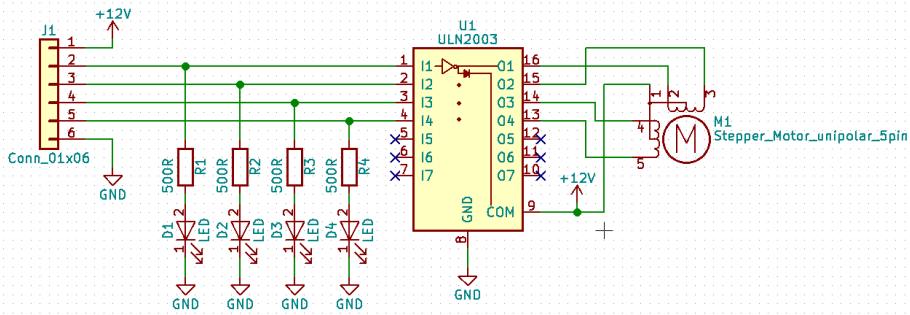


Examples

10.3.14 Step Motor

The stepper motor is a component with 4 coils that must be driven in the correct order to rotate the rotor. Each step of the motor is 1.8º.





Examples

10.4 Others

10.4.1 ETH w5500

This part is a ethernet shield w5500 with support to 8 sockets simultaneously.

Only TCP/UDP unicast address sockets is supported. DHCP is emulated and return a fake ipv4 address.

All listening ports below 2000 are increased by 2000 to avoid operational system services ports. For example listening on port 80 becomes 2080.

w5500 Status Legend:

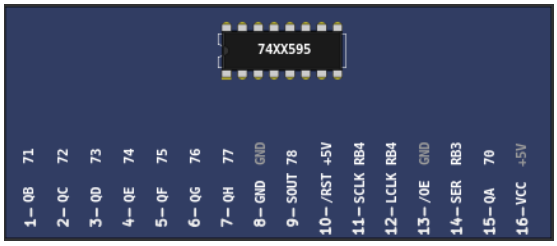
1º Letter - Type	2º Letter - Status	3º Letter - Error
C - Closed	C - Closed	B - Bind
T - TCP	I - Initialized	S - Send
U - UDP	L - Listen	R - Receive
M - MACRAW (don't supported)	S - Syn sent	L - Listen
	E - Established	U - Reuse
	W - Close wait	C - Connecting
	U - UDP	D - Shutdown
	M - MACRAW (don't supported)	

Click on connector to toggle link status.

Examples

10.4.2 IO 74xx595

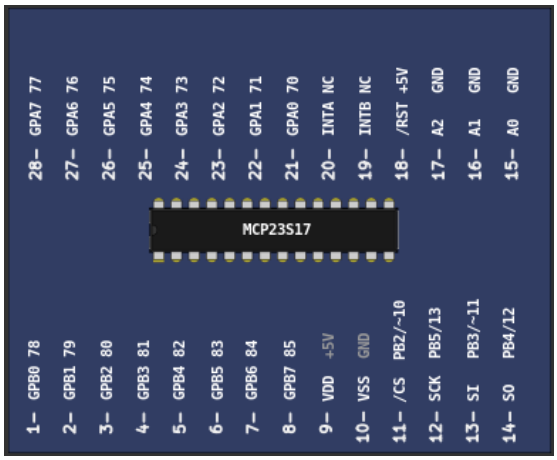
This is one 74xx595 serial input and parallel output 8 bit shift register.



Examples

10.4.3 IO MCP23S17

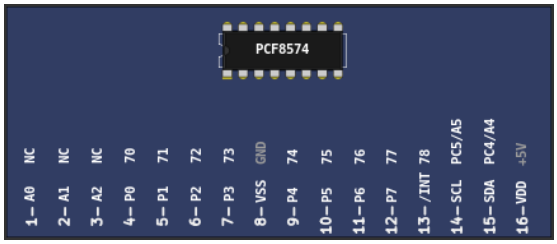
It is a MCP23S17 serial SPI IO expander part.



Examples

10.4.4 IO PCF8574

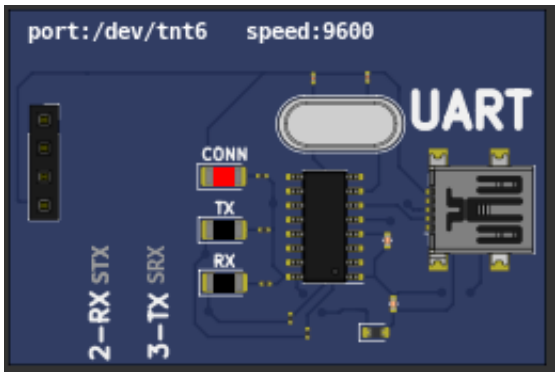
It is a PCF8574 serial I2C IO expander.



Examples

10.4.5 IO UART

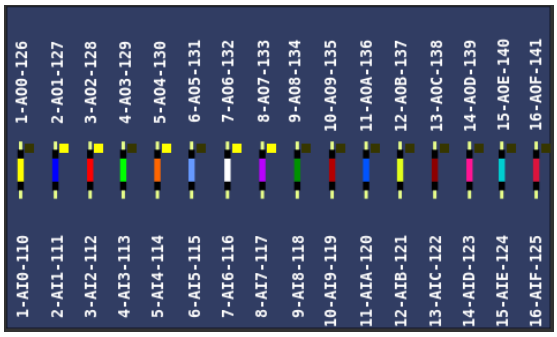
This part is a UART serial port. This part connects the hardware/software UART IO pins of microcontroller to one real/virtual PC serial port. To use virtual port is need to install a virtual port software, as described in 6.



Examples

10.4.6 Jumper Wires

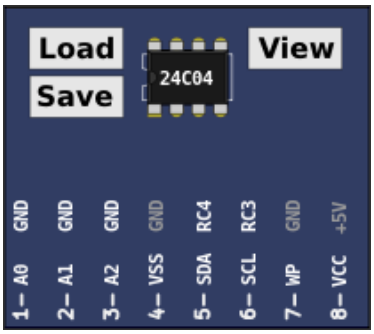
This part are formed by sixteen jumper wires. Each jumper has one input and one output. The jumper input must be connected to one pin output, the jumper output can be connected to multiple pin inputs. The jumper can be used to connect microcontroller pins or make connection between spare parts pins.



Examples

10.4.7 MEM 24CXXX

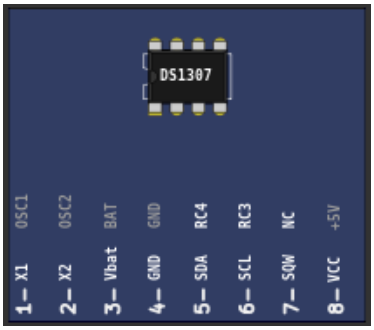
It is a 24CXXX serial I2C EEPROM part. There are support to the models 24C04 and 24C512.



Examples

10.4.8 RTC ds1307

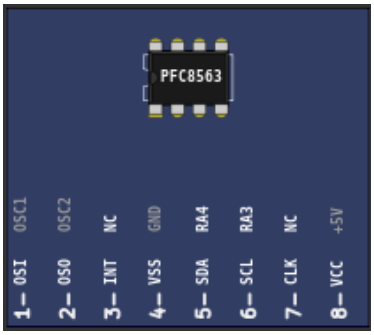
This part is a ds1307 real time clock with serial I2C interface.



Examples

10.4.9 RTC pfc8563

This part is a pfc8563 real time clock with serial I2C interface.



Examples

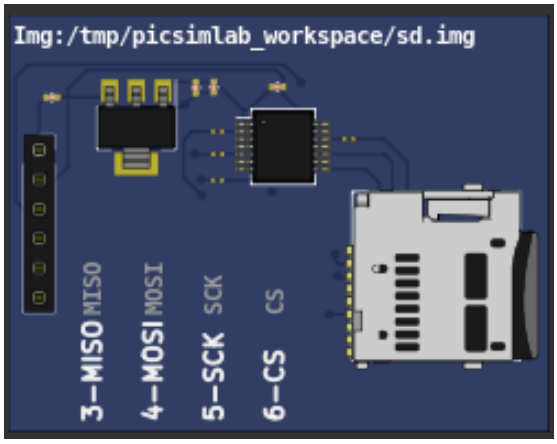
10.4.10 SD Card

This part is a SD Card shield. It's necessary set one sd card file image before use it. (Click on SD card connector to open file dialog)

On Linux one empty image can be created with this command:

```
dd if=/dev/zero of=sd.img bs=1M count=32
```

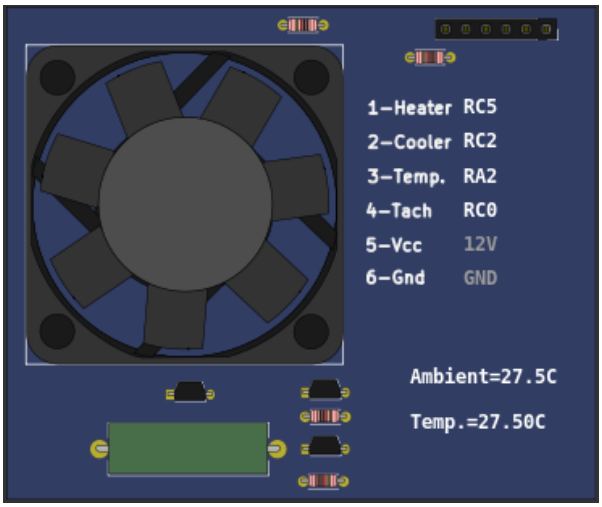
This empty image can be used with raw sd card access, to work with FAT file system the image need to be formatted before the use. (using SdFormatter.ino for example)



Examples

10.4.11 Temperature System

This part is a temperature control system. The temperature control system consists of a heating resistor, an LM35 temperature sensor, a cooler and an infrared tachometer.

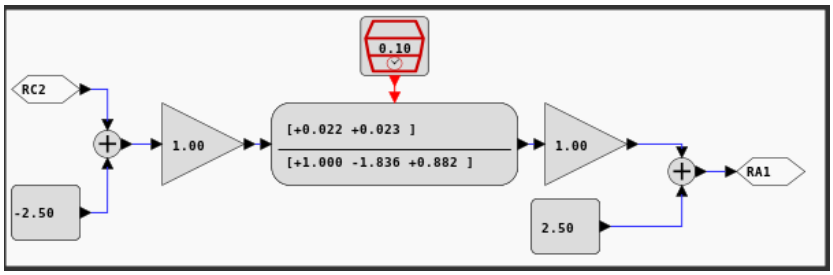


Examples

10.5 Virtual

10.5.1 D. Transfer Function

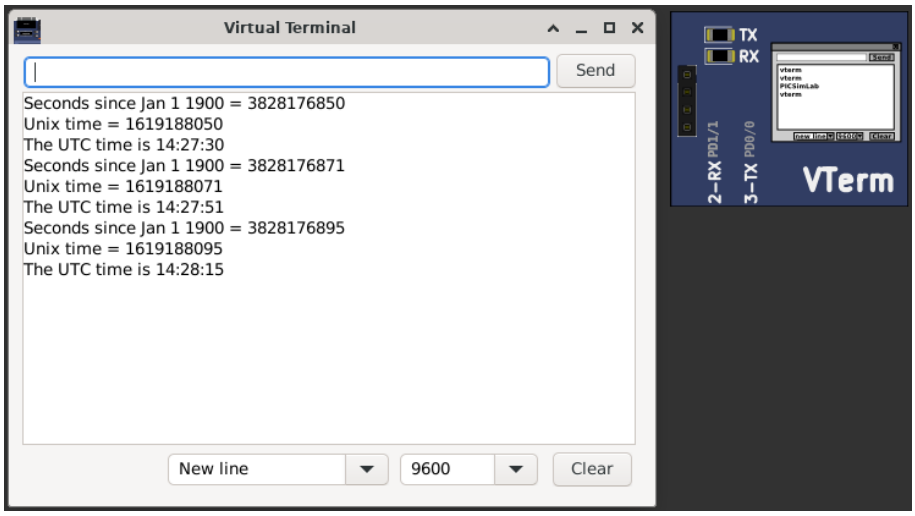
This is a discrete transfer function mathematical model.



Examples

10.5.2 IO Virtual Term

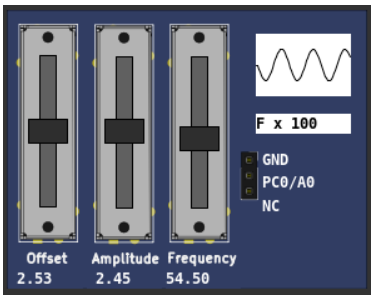
This part is a virtual serial terminal. This part can be used to read and write RX/TX pins UART signals. This part don't need the use or install of virtual serial ports on computer. Clik on terminal picture to open the terminal window.



Examples

10.5.3 Signal Generator

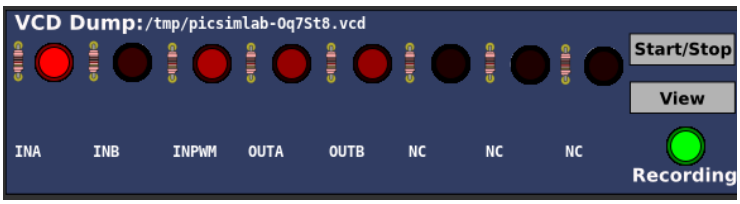
This part is a virtual signal generator with support for sine, square and triangular waves generation with amplitude and frequency adjustment.

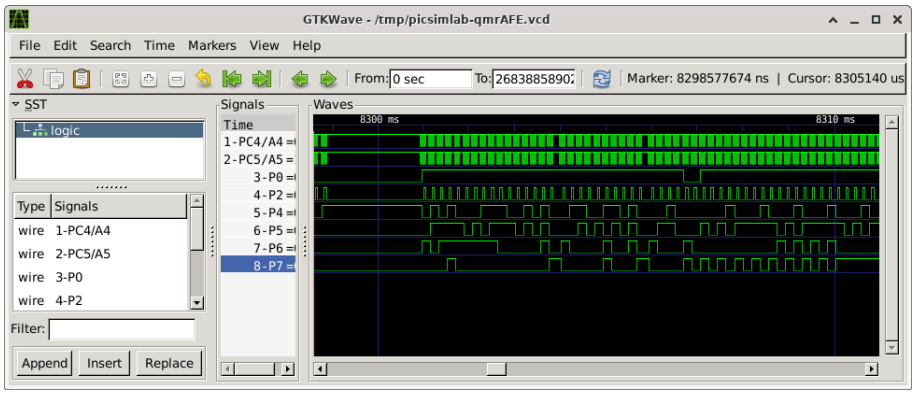


Examples

10.5.4 VCD Dump

This part is a digital value file dump recorder. The file can be visualized with gtkwave.

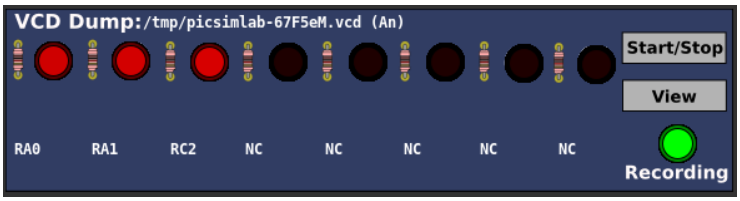


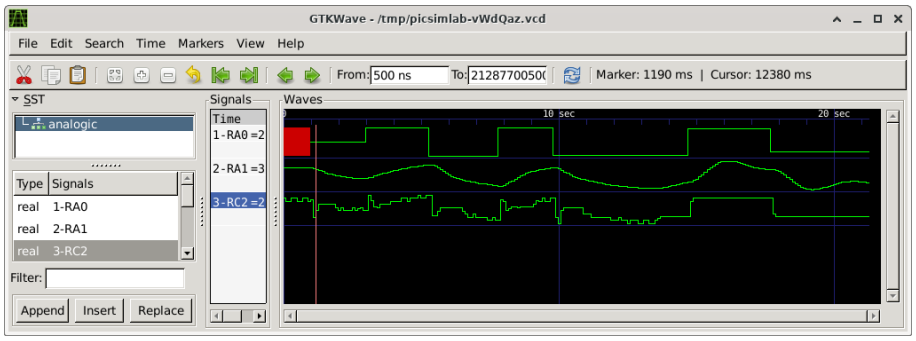


Examples

10.5.5 VCD Dump (Analogic)

This part is a analog value file dump recorder. The file can be visualized with gtkwave.

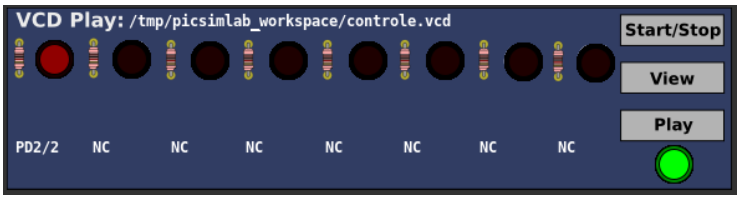




Examples

10.5.6 VCD Play

This part play a VCD file saved from VCD Dump part.


VCD Play: /tmp/picsimlab_workspace/controle.vcd
Start/Stop
View
Play
PD2/2
NC
NC
NC
NC
NC
NC
NC

Examples

Chapter 11

Troubleshooting

The simulation in PICSimLab consists of 3 parts:

- The microcontroller program
- Microcontroller simulation (made by picsim and simavr)
- Simulation of boards and parts

When a problem occurs it is important to detect where it is occurring.

One of the most common problems is the error in the microcontroller program. Before creating an issue, test your code on a real circuit (even partially) to make sure the problem is not there.

Errors in the microcontroller simulation can be detected using code debugging. Any instruction execution or peripheral behavior outside the expected should be reported in the project of simulator used (picsim or simavr).

If the problem is not in either of the previous two options, the problem is probably in PICSimLab. A good practice is to send a source code together with a PICSimLab workspace (.pzw file) to open the issue about the problem.

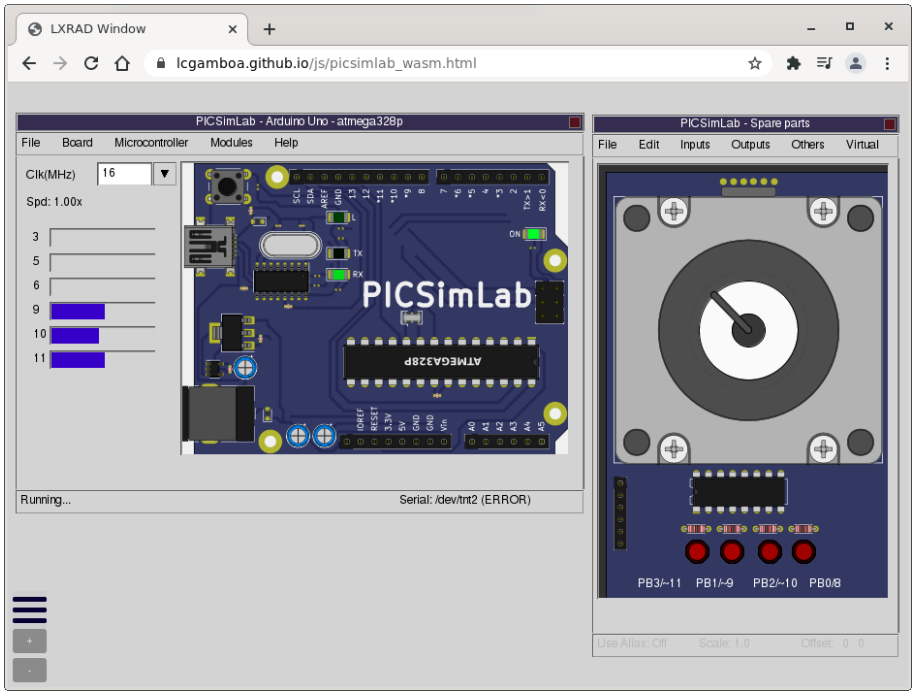
Chapter 12

License

Appendix A

Online Simulator

The online version of PICSimLab has the same source code as the desktop version compiled using Emscripten. The online version does not have the Tools menu, support for debugging and serial communication (only for IO Vterm).

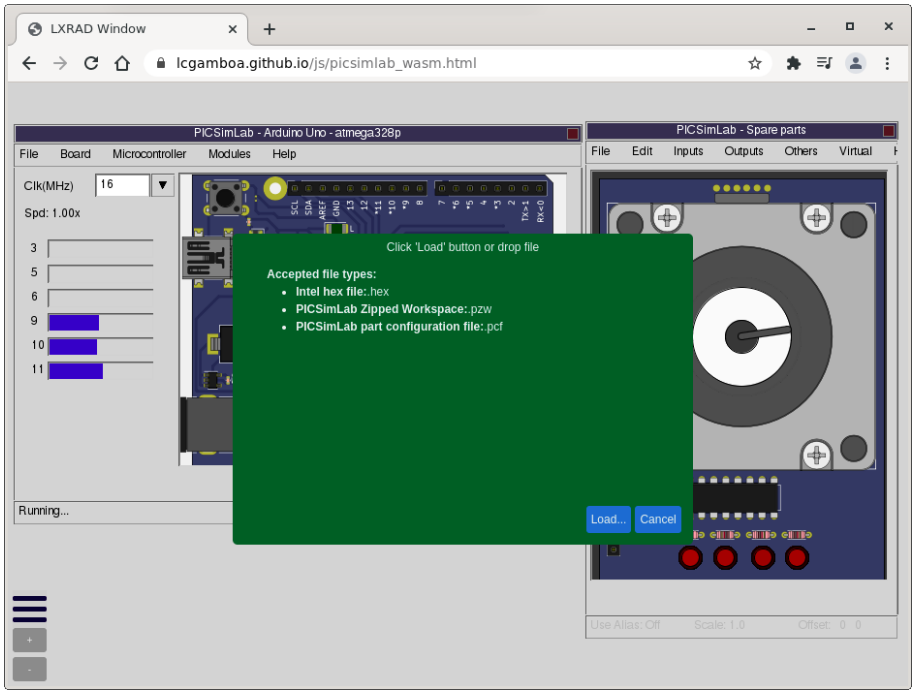


There are three versions generated with Emscripten:

- WASM with multithread - Fast speed (50% of desktop version) and worse browser compatibility (currently only work in Chrome desktop and Firefox nightly desktop because javascript SharedArrayBuffer security requirements)

- WASM - Good speed and good browser compatibility
- ASM.JS - Slow speed and better browser compatibility

Clicking on the three-bar menu (botton left) or any loading option in the menus to open the file loading window. Files can be loaded by drag and drop or by the load button.



The simulator can also be accessed from the examples page for online viewing of most examples (View Online link).

Due to the limitations of the online version, it is advisable to use the desktop version which has more resources and higher simulation speed, especially above 8Mhz clocks.

Appendix B

Use with MPLABX

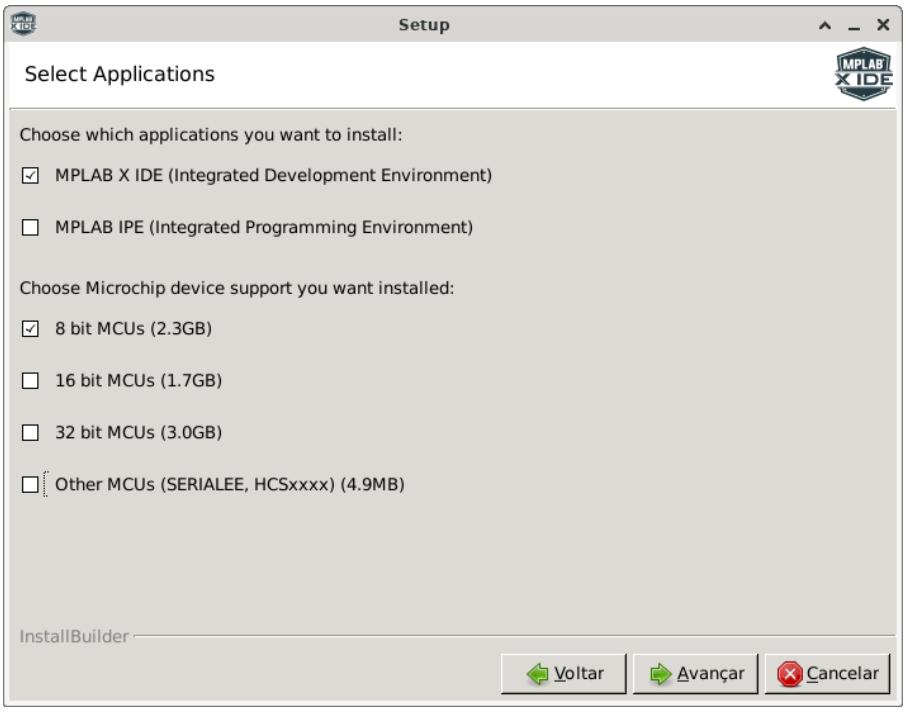
Use with MPLABX to program and Debug

B.1 Installing the Necessary Tools

B.1.1 Install MPLABX IDE and XC8 Compiler

Links for download MPLABX IDE and XC8 Compiler installers. Download and install.

For PICSimLab only MPLAB X IDE and 8 bit MCU support needs to be installed.

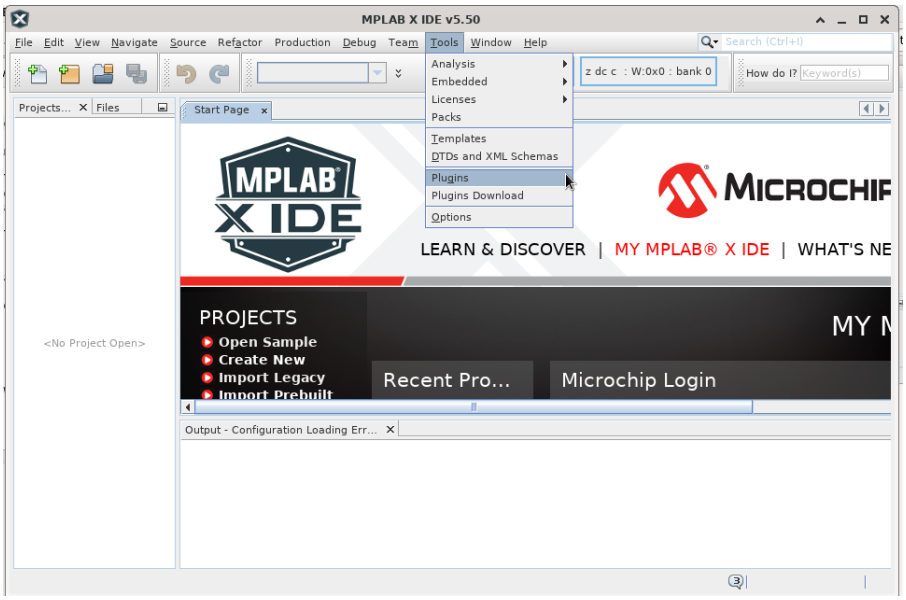


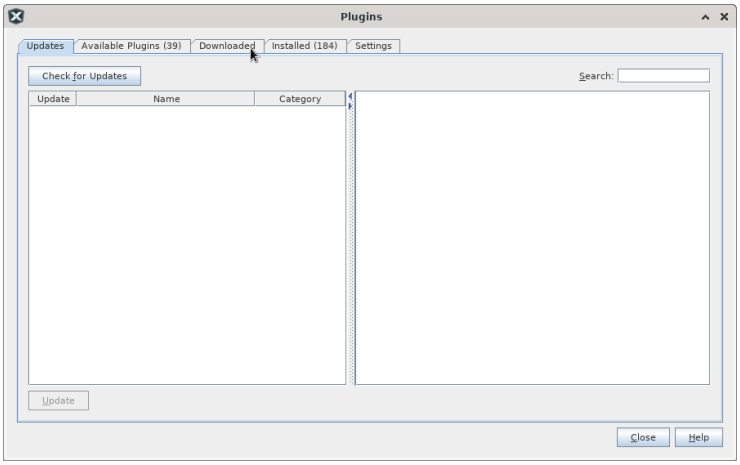
B.1.2 Install PICsimLab

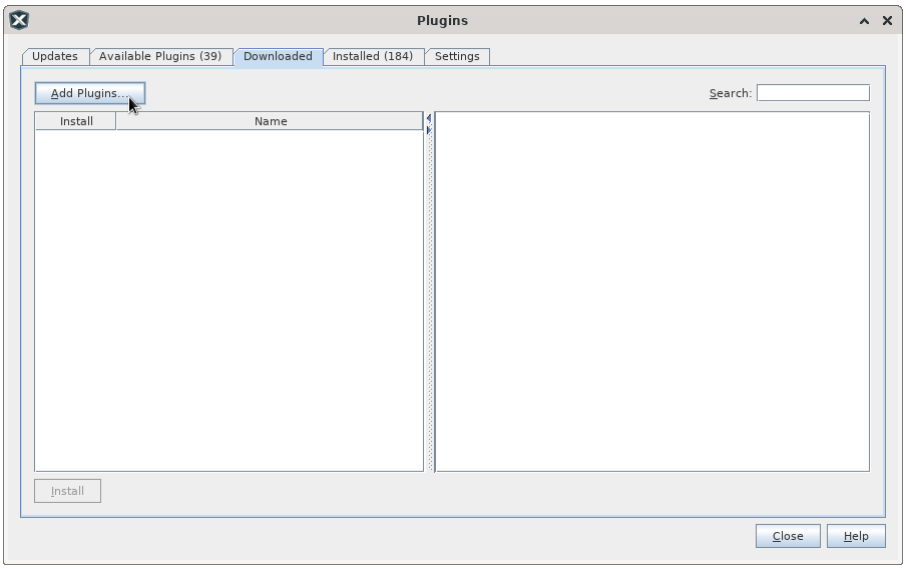
Link for download PICSimLab installer. Download and install

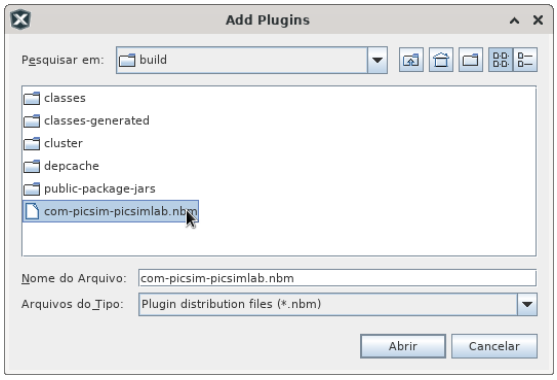
B.1.3 How to Install PICSimLab MPLABX Debugger plugin

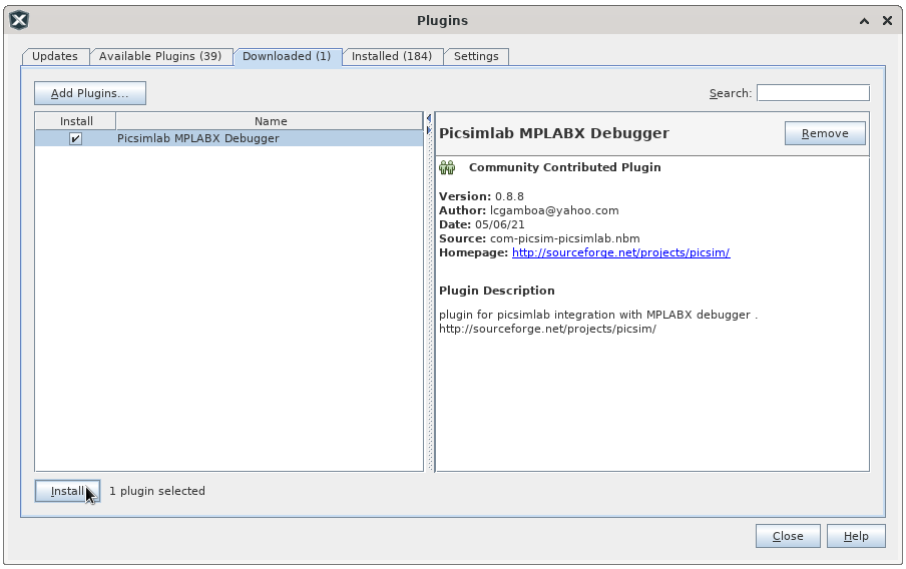
Link for download PicsimLab MPLABX Debugger plugin (com-picsim-picsimlab.nbm)

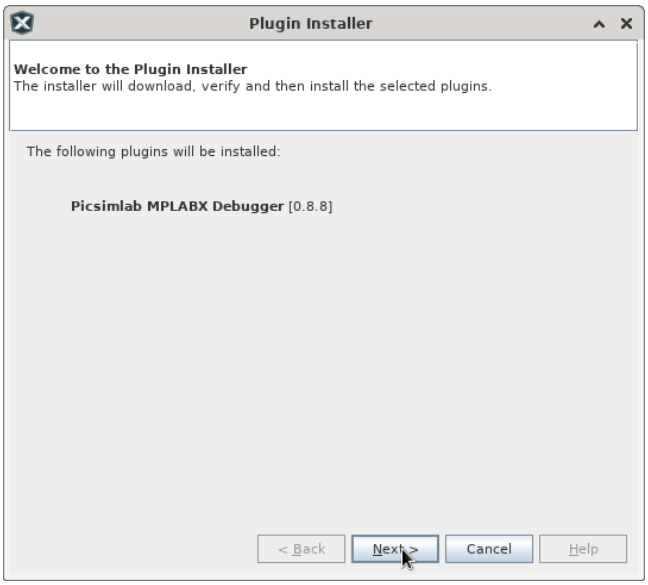

MPLAB X IDE v5.50
File Edit View Navigate Source Refactor Production Debug Team Tools Window Help
Search (Ctrl+I)
Analysis
Embedded
Licenses
Packs
Templates
DTDs and XML Schemas
Plugins
Plugins Download
Options
z dc c : W:0x0 : bank 0
How do I? Keyword(s)
Projects... Files
Start Page
MPLAB X IDE
MICROCHIP
LEARN & DISCOVER | MY MPLAB® X IDE | WHAT'S NE
<No Project Open>
PROJECTS
Open Sample
Create New
Import Legacy
Import Prebuilt
Recent Pro...
Microchip Login
Output - Configuration Loading Err...

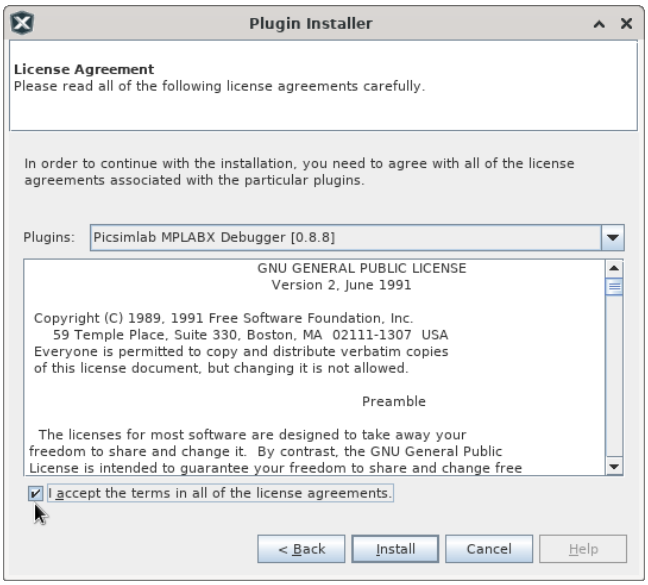

Plugins
Updates
Available Plugins (39)
Downloaded
Installed (184)
Settings
Check for Updates
Search:
Update
Name
Category
Update
Close
Help

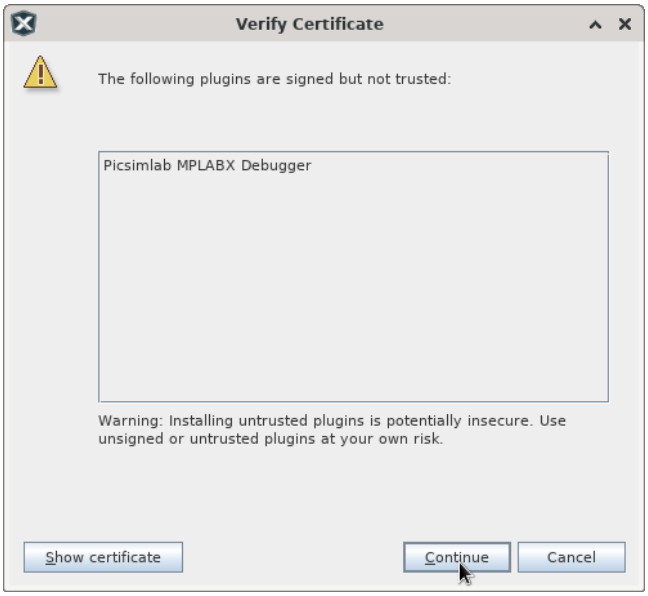

Plugins
Updates
Available Plugins (39)
Downloaded
Installed (184)
Settings
Add Plugins...
Search:
Install
Name
Install
Close
Help

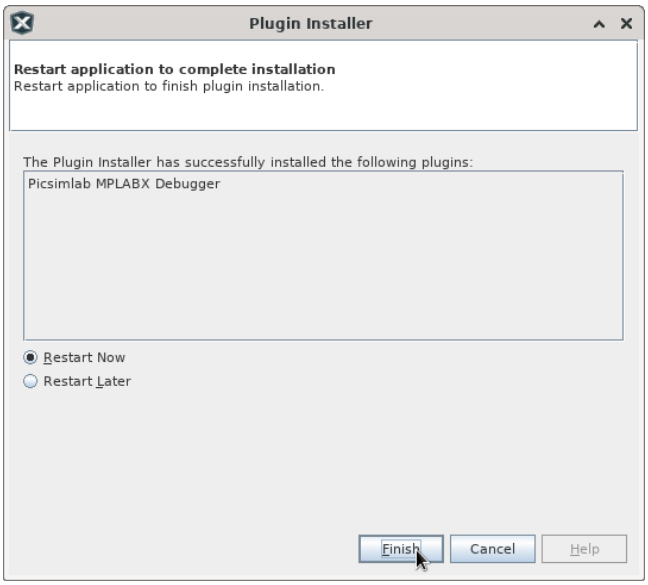

Add Plugins
Pesquisar em:
build
classes
classes-generated
cluster
depcache
public-package-jars
com-picsim-picsimlab.nbm
Nome do Arquivo:
com-picsim-picsimlab.nbm
Arquivos do Tipo:
Plugin distribution files (*.nbm)
Abrir
Cancelar


Plugins
Updates
Available Plugins (39)
Downloaded (1)
Installed (184)
Settings
Add Plugins...
Search:
Install
Name
Picsimlab MPLABX Debugger
Picsimlab MPLABX Debugger
Remove
Community Contributed Plugin
Version: 0.8.8
Author: lcgamboa@yahoo.com
Date: 05/06/21
Source: com-picsim-picsimlab.nbm
Homepage: http://sourceforge.net/projects/picsim/
Plugin Description
plugin for picsimlab integration with MPLABX debugger .
http://sourceforge.net/projects/picsim/
Install
1 plugin selected
Close
Help


Plugin Installer
Welcome to the Plugin Installer
The installer will download, verify and then install the selected plugins.
The following plugins will be installed:
Picsimlab MPLABX Debugger [0.8.8]
< Back
Next >
Cancel
Help

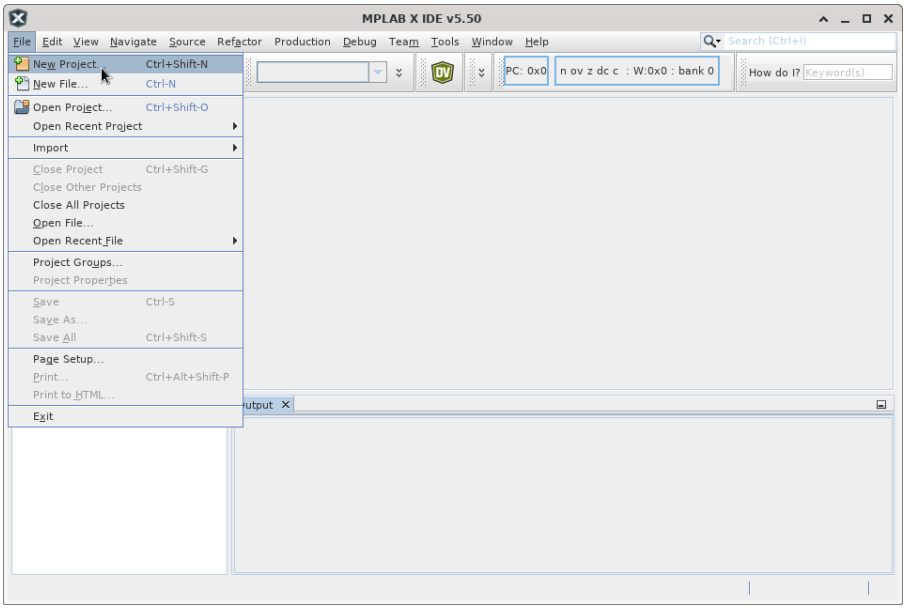

Plugin Installer
License Agreement
Please read all of the following license agreements carefully.
In order to continue with the installation, you need to agree with all of the license agreements associated with the particular plugins.
Plugins:
Picsimlab MPLABX Debugger [0.8.8]
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
Preamble
The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
I accept the terms in all of the license agreements.
< Back
Install
Cancel
Help

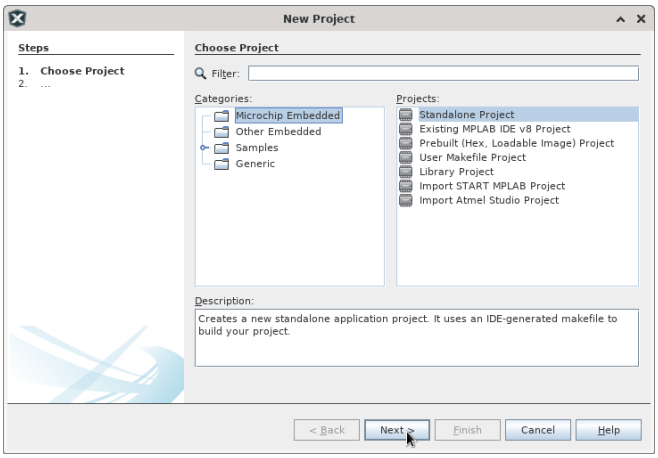

Verify Certificate
The following plugins are signed but not trusted:
Picsimlab MPLABX Debugger
Warning: Installing untrusted plugins is potentially insecure. Use unsigned or untrusted plugins at your own risk.
Show certificate
Continue
Cancel

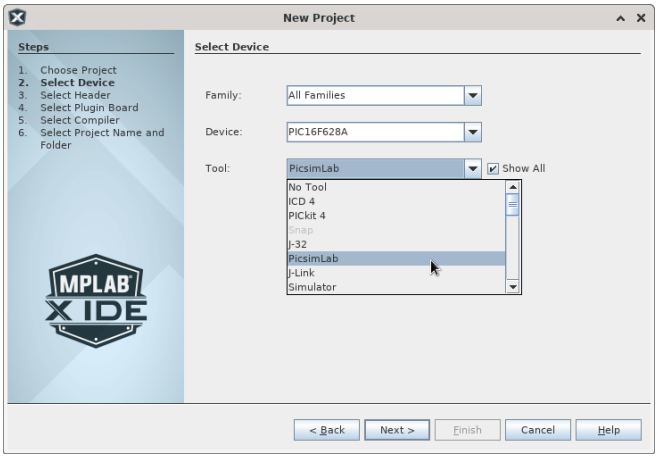


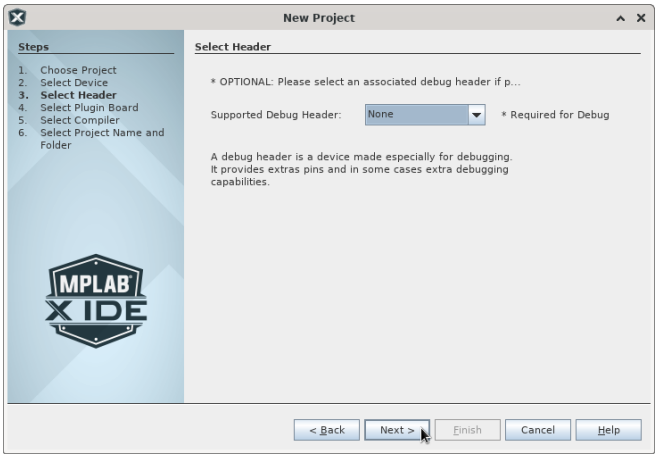
B.2 Configuring a New Project in MPLABX

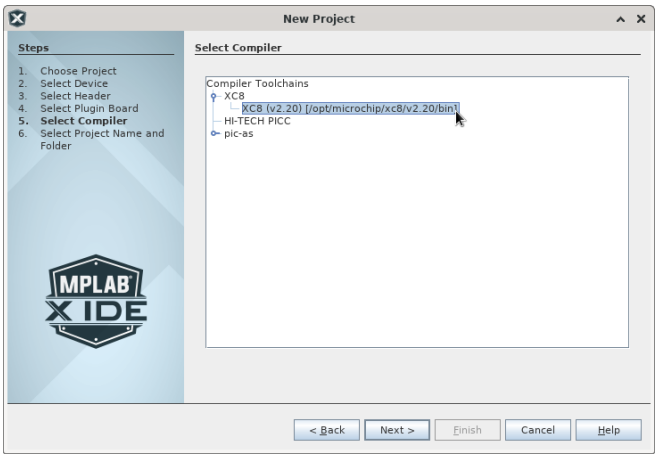
B.2.1 Project Creation

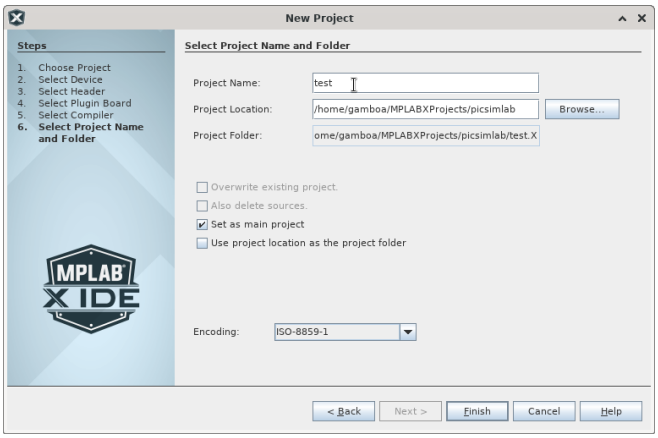



New Project
Steps
1. Choose Project
2. ...
Choose Project
Filter:
Categories:
Microchip Embedded
Other Embedded
Samples
Generic
Projects:
Standalone Project
Existing MPLAB IDE v8 Project
Prebuilt (Hex, Loadable Image) Project
User Makefile Project
Library Project
Import START MPLAB Project
Import Atmel Studio Project
Description:
Creates a new standalone application project. It uses an IDE-generated makefile to build your project.
< Back
Next >
Finish
Cancel
Help

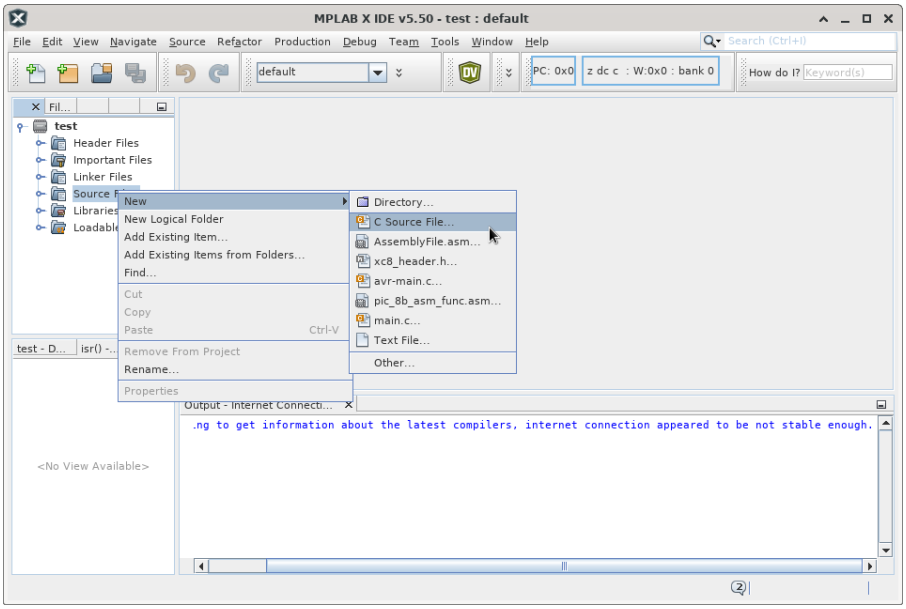

New Project
Steps
1. Choose Project
2. Select Device
3. Select Header
4. Select Plugin Board
5. Select Compiler
6. Select Project Name and Folder
Select Device
Family:
All Families
Device:
PIC16F628A
Tool:
PicsimLab
Show All
No Tool
ICD 4
PICkit 4
SNAP
J-32
PicsimLab
J-Link
Simulator
MPLAB X IDE
< Back
Next >
Finish
Cancel
Help

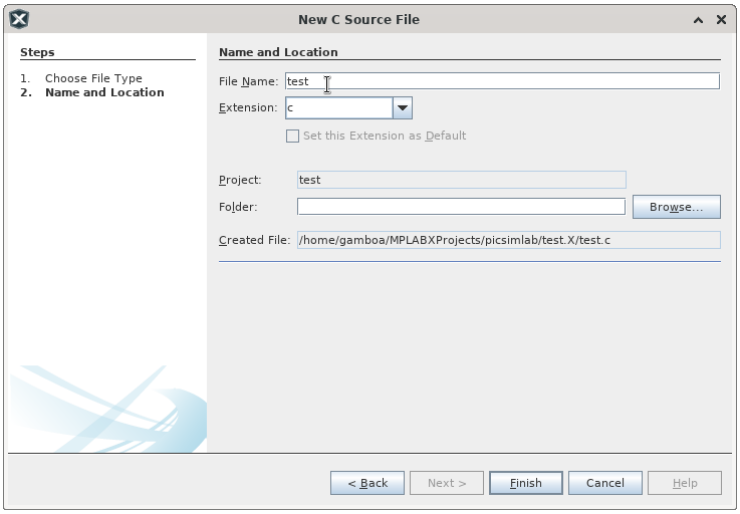

New Project
Steps
1. Choose Project
2. Select Device
3. Select Header
4. Select Plugin Board
5. Select Compiler
6. Select Project Name and Folder
Select Header
* OPTIONAL: Please select an associated debug header if p...
Supported Debug Header:
None
* Required for Debug
A debug header is a device made especially for debugging.
It provides extras pins and in some cases extra debugging capabilities.
MPLAB X IDE
< Back
Next >
Finish
Cancel
Help


New Project
Steps
1. Choose Project
2. Select Device
3. Select Header
4. Select Plugin Board
5. Select Compiler
6. Select Project Name and Folder
Select Compiler
Compiler Toolchains
XC8
XC8 (v2.20) [/opt/microchip/xc8/v2.20/bin]
HI-TECH PICC
pic-as
MPLAB X IDE
< Back
Next >
Finish
Cancel
Help

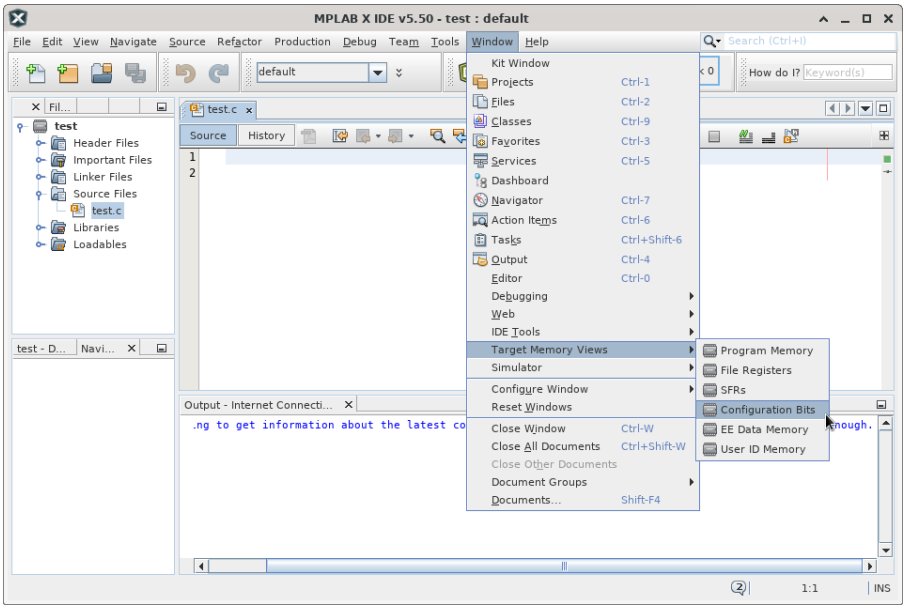

New Project
Steps
1. Choose Project
2. Select Device
3. Select Header
4. Select Plugin Board
5. Select Compiler
6. Select Project Name and Folder
Select Project Name and Folder
Project Name:
test
Project Location:
/home/gamboa/MPLABXProjects/picsimlab
Browse...
Project Folder:
ome/gamboa/MPLABXProjects/picsimlab/test.X
Overwrite existing project.
Also delete sources.
Set as main project
Use project location as the project folder
Encoding:
ISO-8859-1
MPLAB X IDE
< Back
Next >
Finish
Cancel
Help

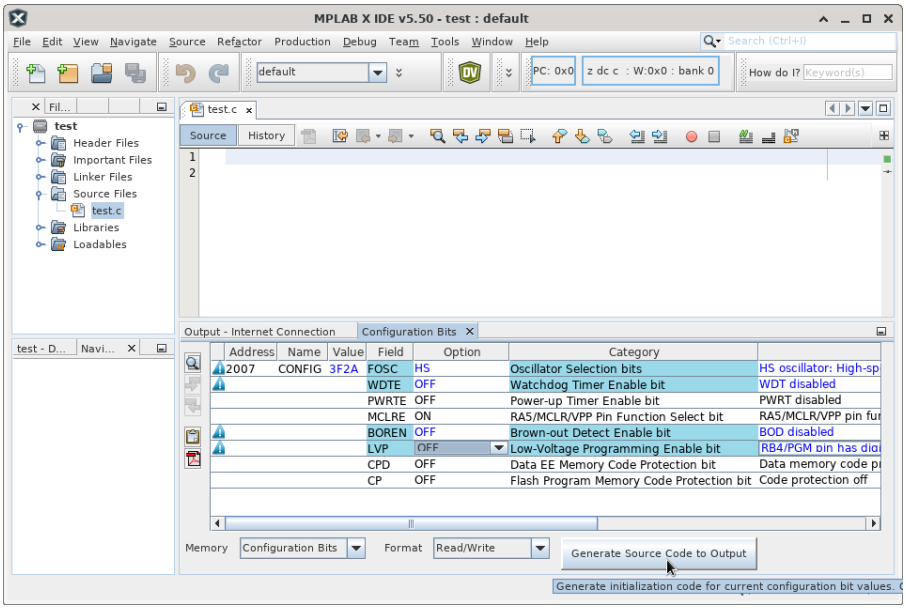
B.2.2 File Creation

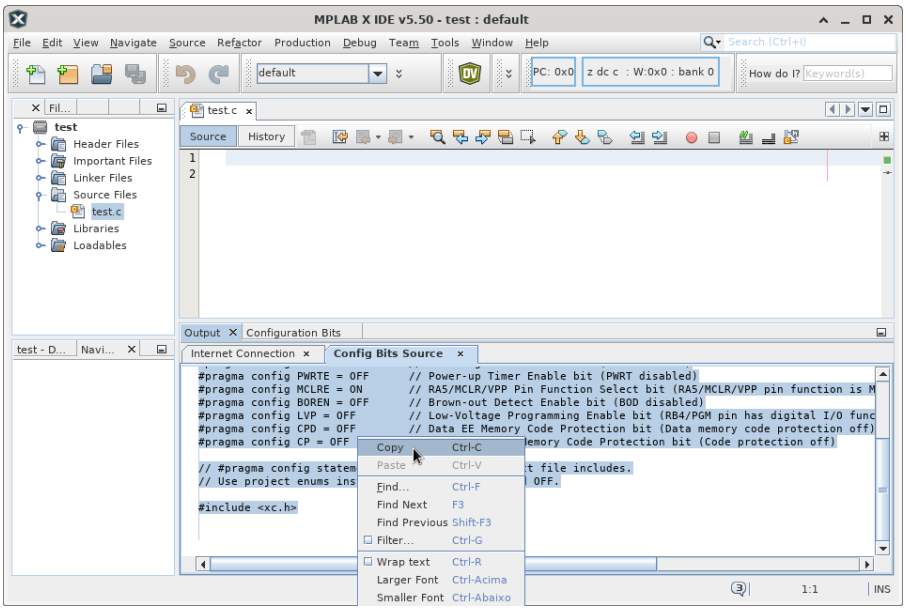




B.2.3 PIC Configuration Bits



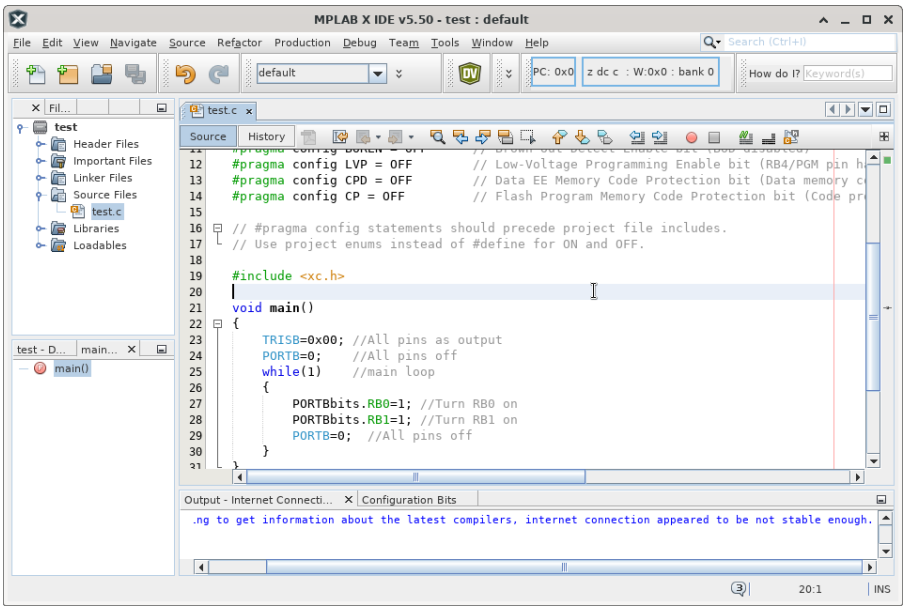




B.2.4 Code Example

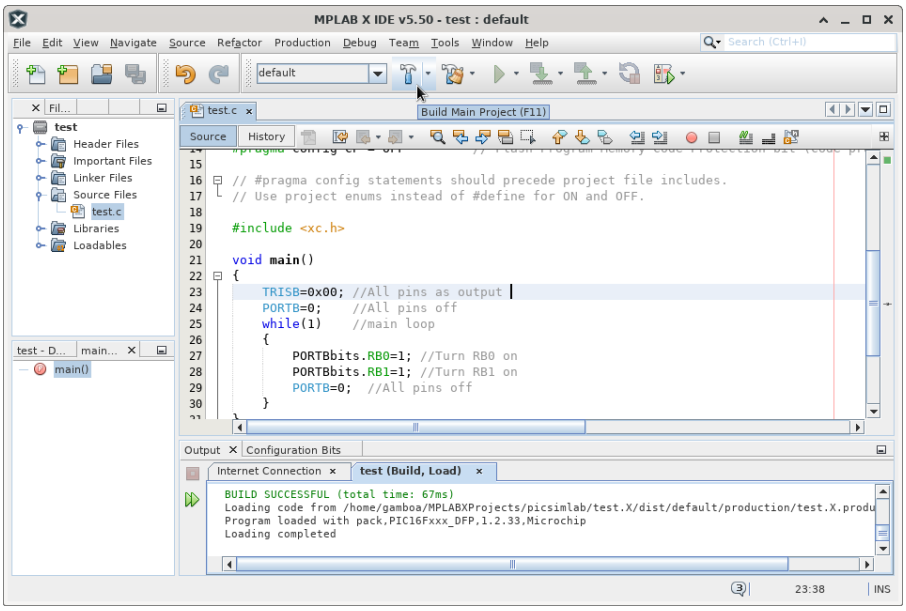
Paste the configuration and this simple code example in test.c:

```
void main()
{
    TRISB=0x00; //All pins as output
    PORTB=0;    //All pins off
    while(1)    //main loop
    {
        PORTBbits.RB0=1; //Turn RB0 on
        PORTBbits.RB1=1; //Turn RB1 on
        PORTB=0;  //All pins off
    }
}
```



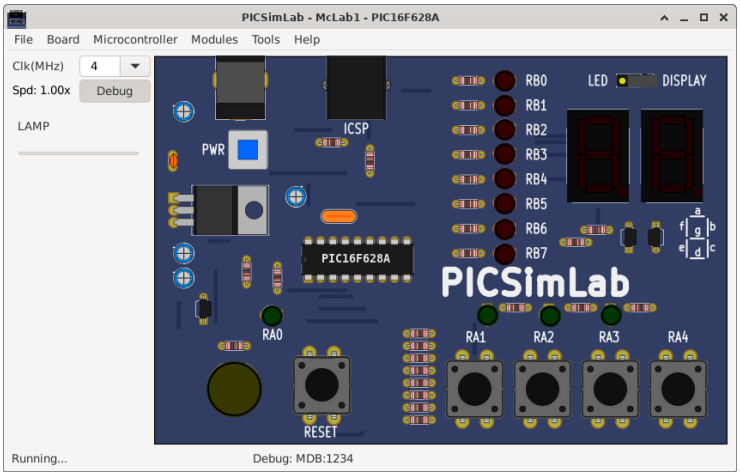
B.2.5 Building the Project

Use the **Build** button and wait for the message "BUILD SUCCESSFUL".



B.3 Program and Debug PICsimLab With MPLABX

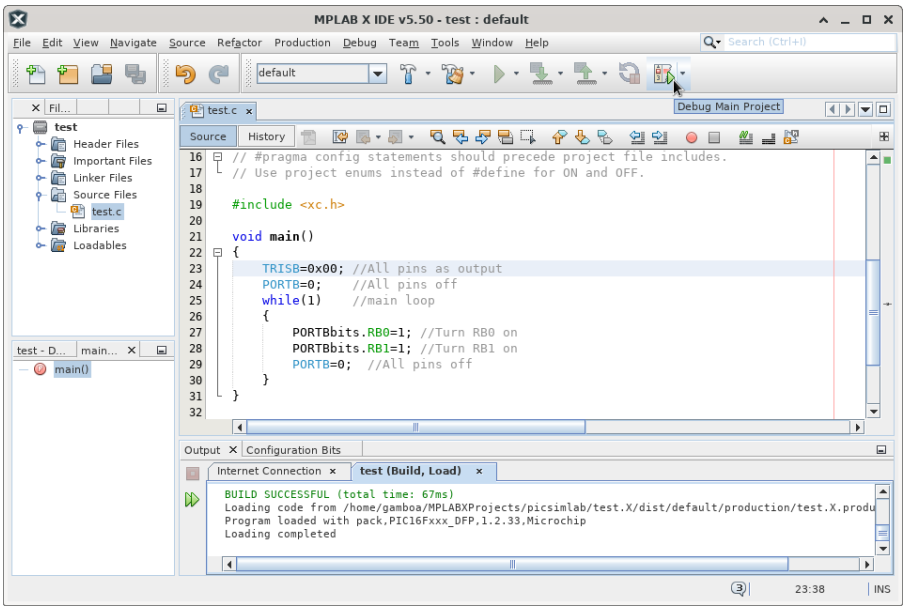
B.3.1 Starting PICsimLab



The plugin connect to Picsimlab through a TCP socket using port 1234, and you have to allow the access in the firewall. Verify in the PICsimLab statusbar the message "MplabxD: Ok". It's show debugger server state.

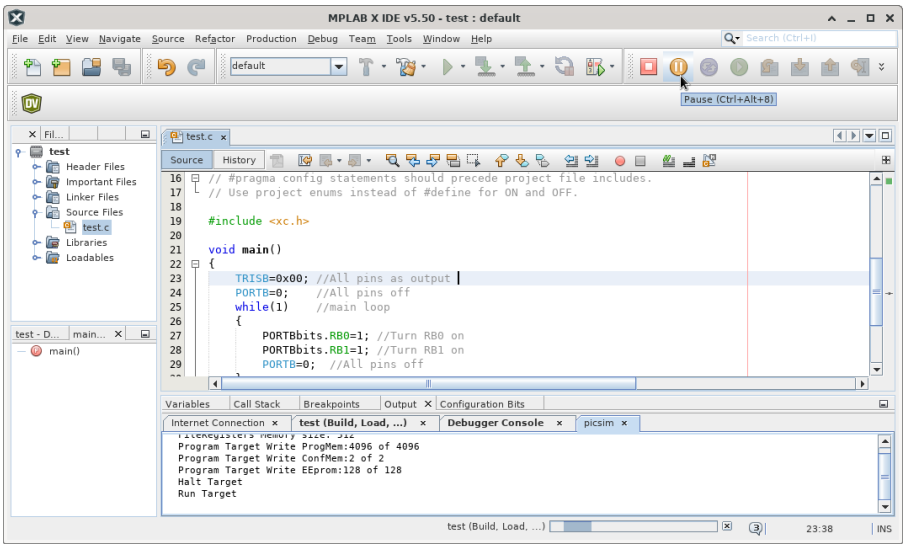
B.3.2 Programming PICsimLab

Use the **Debug** button to programming PICsimLab.



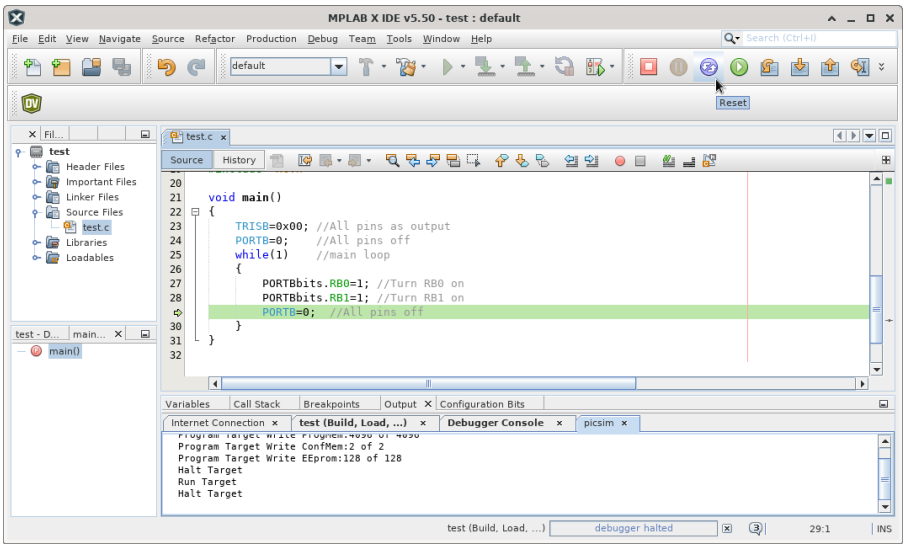
B.3.3 Pausing the Program

Use the **Pause** button to stop the program and inspect the code and memory.



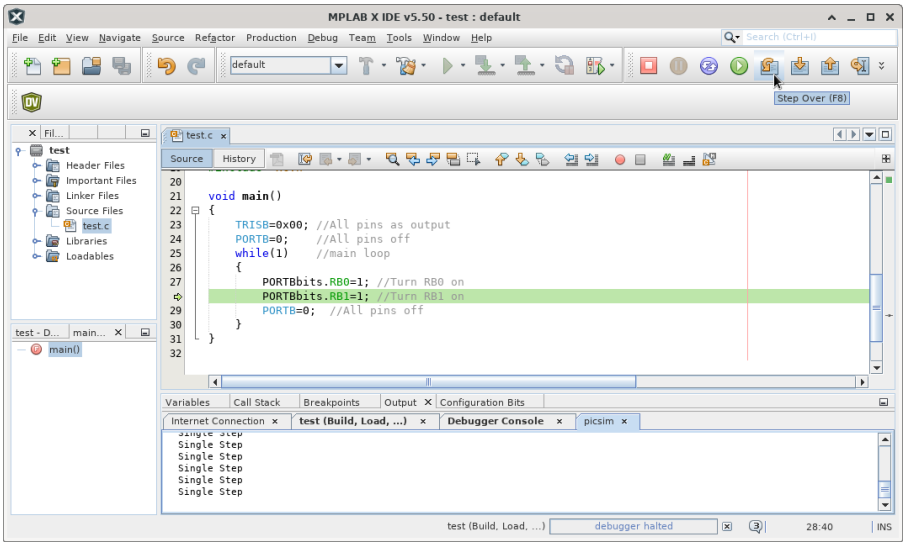
B.3.4 Restarting the Program

Use the **Restart** button to restart the program.

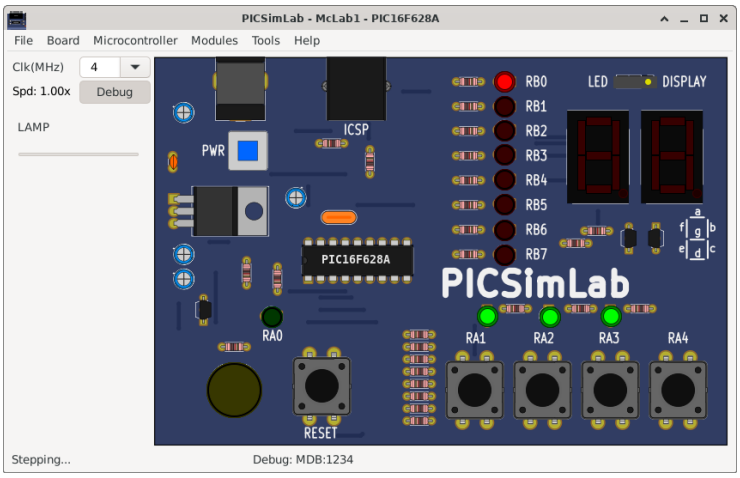


B.3.5 Running Step by Step

Use the **Step** or **Step Over** button to run the program step by step.

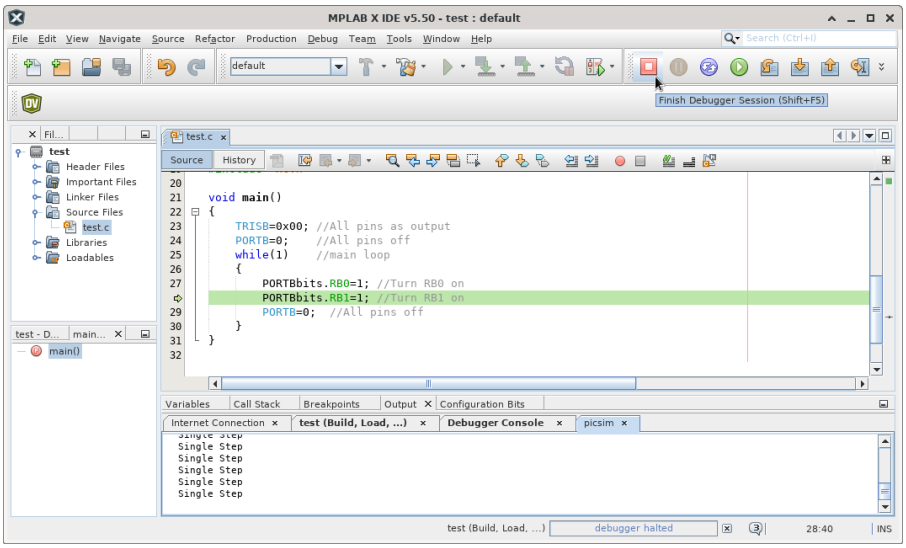


See in the PICsimLab the changes of each step.



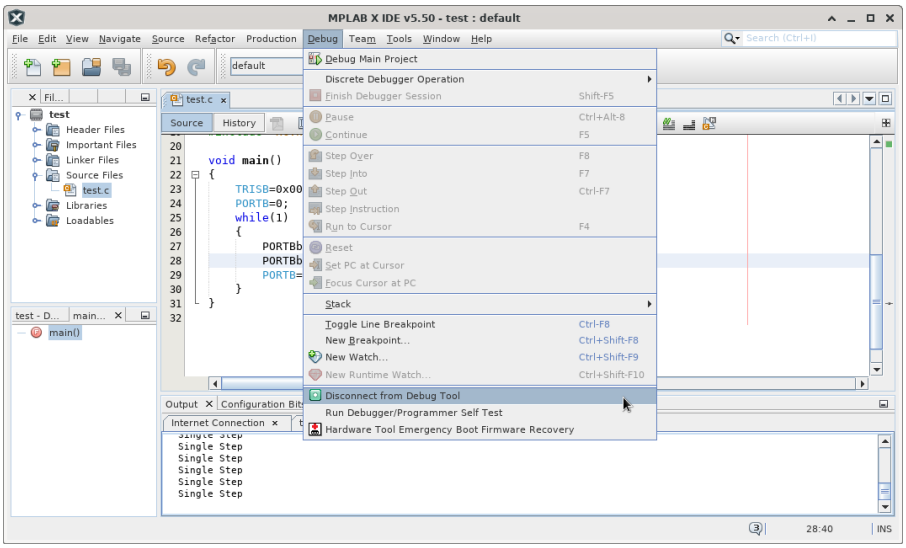
B.3.6 Stopping Debugger

Use the **Stop** button to turn off the MPLABX debugger.



B.3.7 Disconnect Debugger

Use the menu **Debug->Disconnect From Debug Tool** to disconnect the MPLABX debugger. The program continues running in PICsimLab after MPLABX debugger is disconnected.



B.4 This Tutorial in Video

Link for Youtube video version of this tutorial: How to use MPLABX to program and debug PicsimLab 0.6

Appendix C

Creating New Boards

First get the source code and compile as described in Install from source.

C.1 Creating a New Board

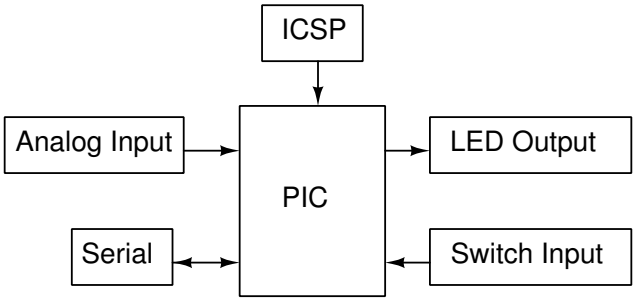
The first step is get the schematic and all information about the board hardware. The second step is the creation of four files in PICSimLab dir (consider replace the 'x' of board_x for a name of your board in your case):

- Board Picture (share/boards/X/board.svg) or (share/boards/X/board.png);
- Board map (share/boards/X/board.map);
- Board header (src/boards/board_x.h);
- Board C++ code (src/boards/board_x.cc);

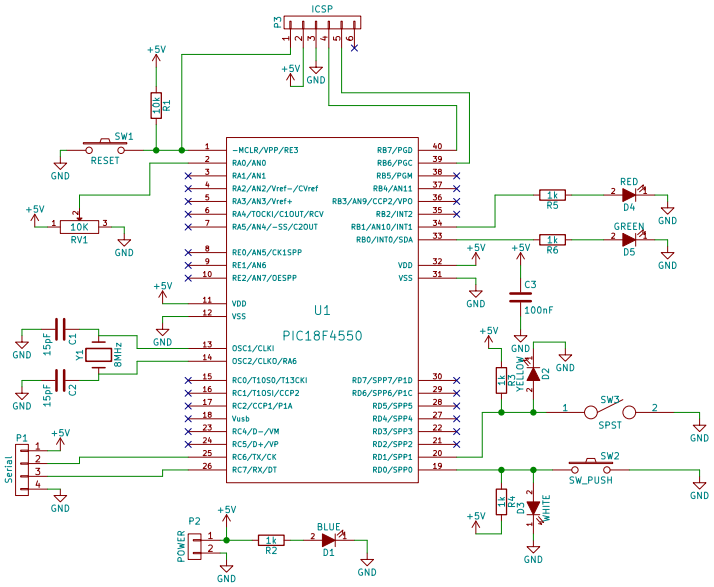
The third and last step is recompiling PICSimLab with new board support.

C.1.1 Board Hardware and Schematic

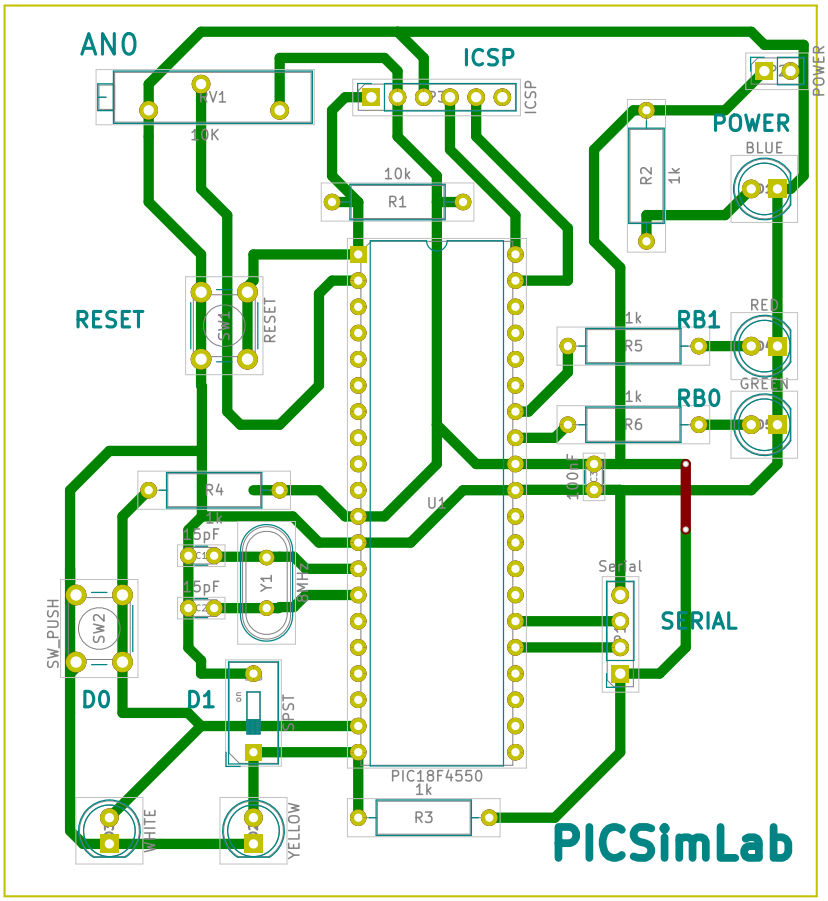
For this tutorial, the board created have the hardware shown in diagram below:



The schematic for the tutorial board made in Kicad.

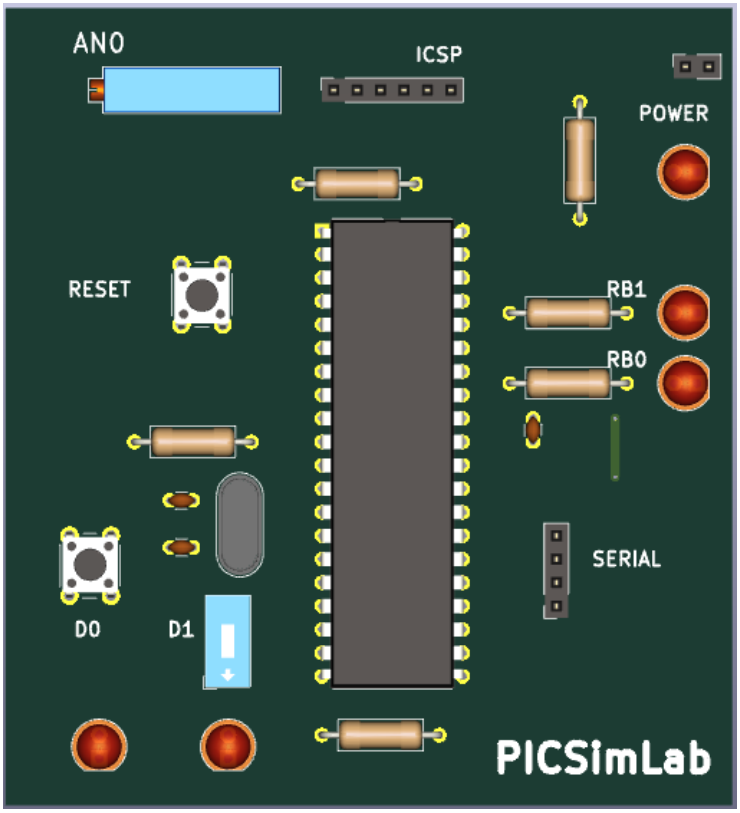


And the PCB layout was made in Kicad too. The PCB is not necessary if you have a real board.

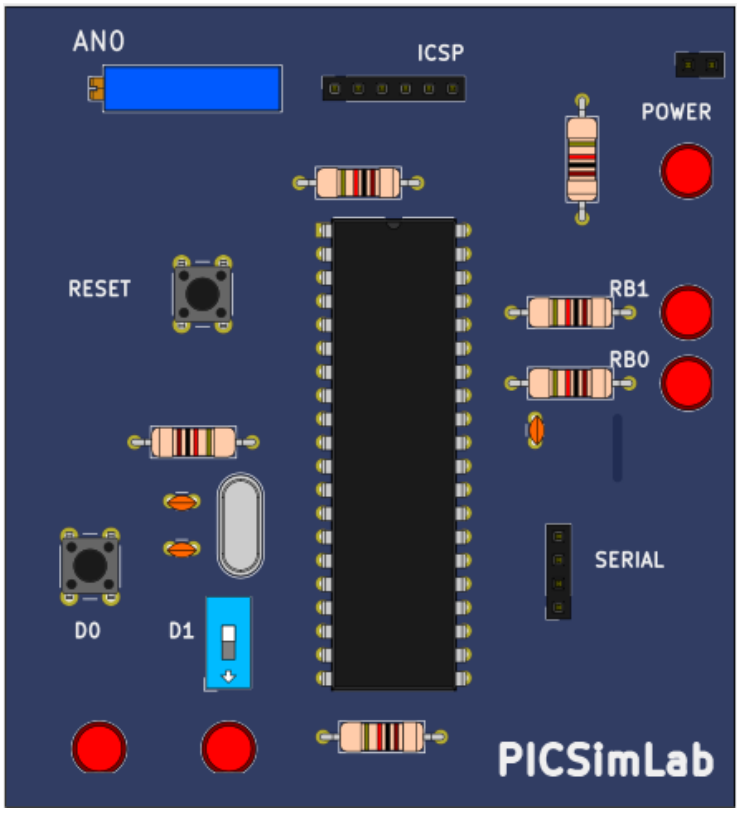


C.1.2 Board Picture

The PNG board picture was taken from Kicad 3D viewer. The picture image is saved as "share/board/X/board.png".



It is also possible to use images in SVG format for better viewing quality. PCBDraw can be used to convert a Kicad PCB project to an SVG image. The picture image is saved as "share/board/X/board.svg".


AN0
ICSP
POWER
RESET
RB1
RB0
SERIAL
D0
D1
PICSimLab

C.1.3 Picture map

The PICSimLab use one picture image map for inputs and outputs.

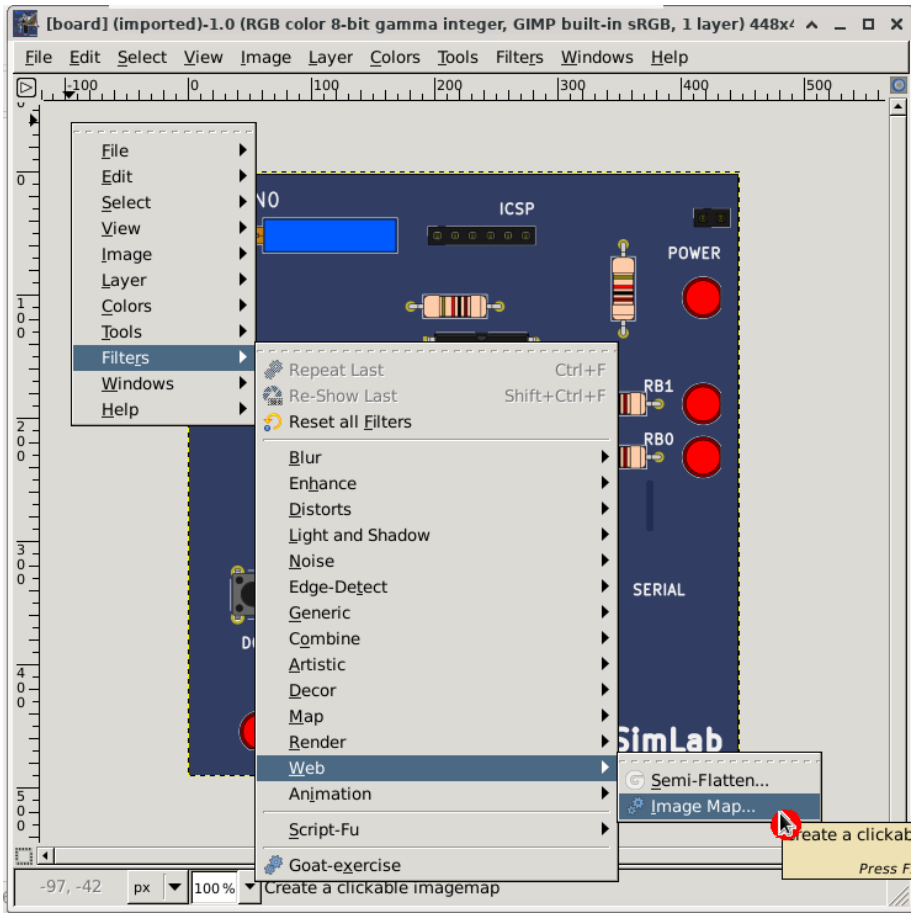
The inputs are the areas in board picture which user can interact (by mouse click) and start with letters “I_”.

The output are the areas in board picture to be redraw according simulator status and start with letters “O_”.

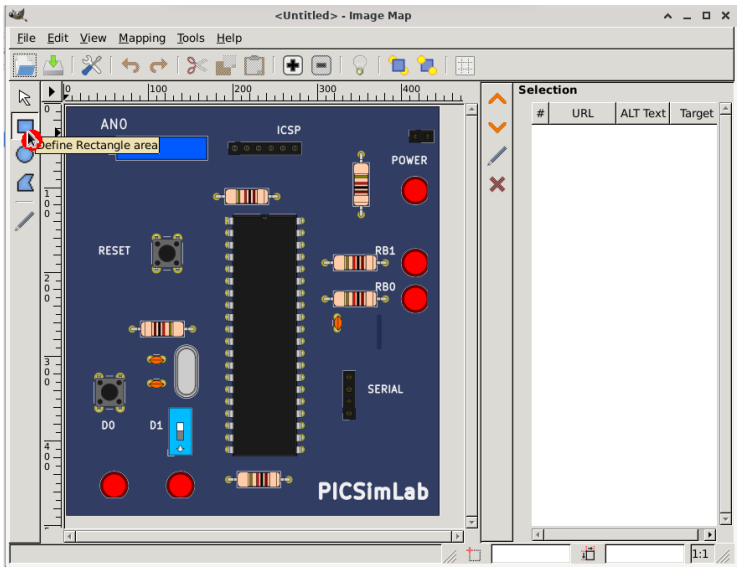
The bidirectional areas in board picture which user can interact and need to be redraw according simulator status are started with letter “B_”.

The picture map used for PICSimLab are normal HTML image-map. They can be made by hand or using any software which can handle image maps. The original PICSimLab maps are made using Gimp image editor.

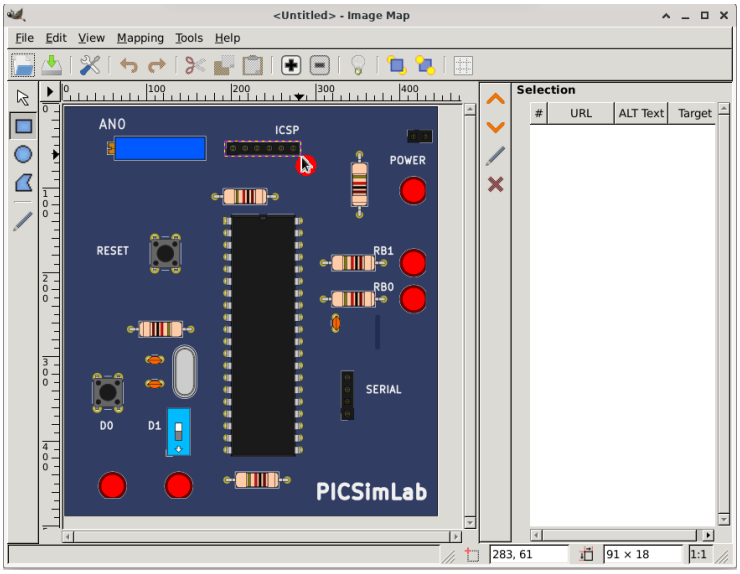
To start, in the GIMP, use the Filters->Web->Image Map to open image map editor window.



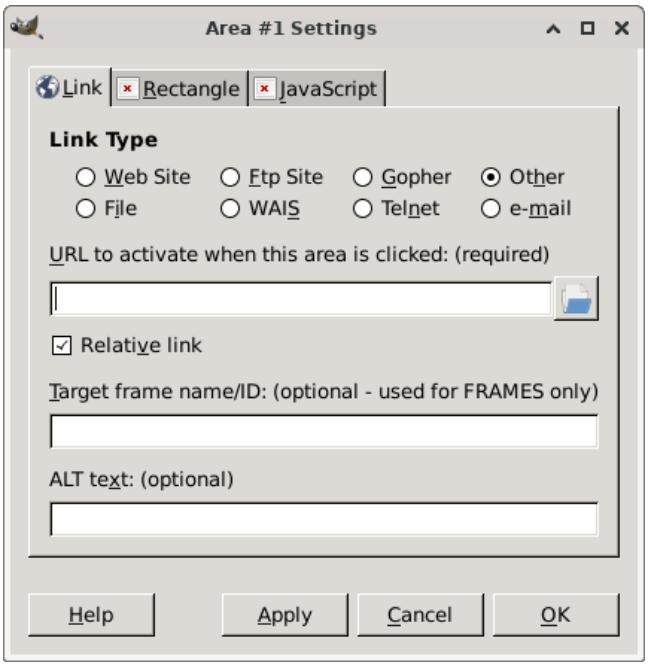
Then select rectangle or circle map on toolbar.



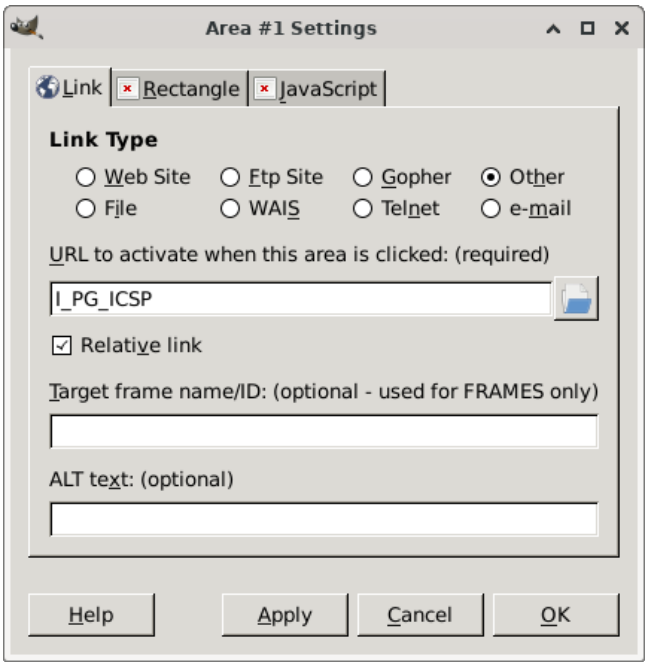
And mark the area in picture.



After area is select, in the settings windows select the link type for “Other”.



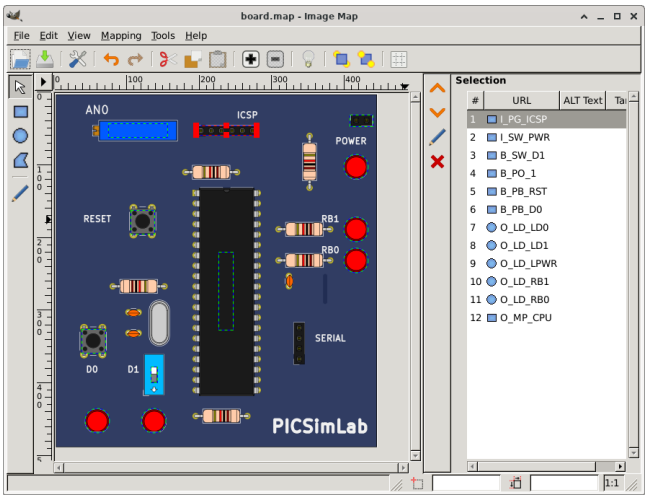
And write the name of area. The name must describe the area function on the board and follow the Picture Map Reference.



Board map

For this tutorial board, twelve areas are marked:

- I_PG_ICSP - where user click to load hexfile.
- I_SW_PWR - where user click to turn on/off the board.
- B_SW_D1 - Switch connected in RD1.
- B_PO_1 - Potentiometer connected to RA0.
- B_PB_RST - Button to reset board.
- B_PB_D0 - Button connected in RD0.
- O_LD_LD0 - draw LED connected in push button D0.
- O_LD_LD1 - draw LED connected in switch D1.
- O_LD_LPWR - draw power LED indicator.
- O_LD_RB1 - draw LED connected in RB1.
- O_LD_RB0 - draw LED connected in RB0.
- O_IC_CPU - draw microcontroller name.



Board map generated by Gimp image map editor and saved as “share/boards/X/board.map”.

```
<img src="[board_x] (imported)" width="448" height="491" border="0" usemap="#map" />

<map name="map">
<!-- #$-:Image map file created by GIMP Image Map plug-in -->
<!-- #$-:GIMP Image Map plug-in by Maurits Rijk -->
<!-- #$-:Please do not edit lines starting with "#$" -->
<!-- #$VERSION:2.3 -->
<!-- #$AUTHOR:lcgamboa@yahoo.com          -->
<area shape="rect" coords="196,45,280,58" href="I_PG_ICSP" />
<area shape="rect" coords="409,30,441,46" href="I_SW_PWR" />
<area shape="rect" coords="133,379,142,401" href="B_SW_D1" />
<area shape="rect" coords="74,42,156,61" href="B_PO_1" />
<area shape="rect" coords="105,162,138,195" href="B_PB_RST" />
<area shape="rect" coords="37,327,70,360" href="B_PB_D0" />
<area shape="circle" coords="59,454,17" href="O_LD_LD0" />
<area shape="circle" coords="137,454,17" href="O_LD_LD1" />
<area shape="circle" coords="418,102,17" href="O_LD_LPWR" />
<area shape="circle" coords="418,189,17" href="O_LD_RB1" />
<area shape="circle" coords="418,232,17" href="O_LD_RB0" />
<area shape="rect" coords="227,220,247,328" href="O_IC_CPU" />
</map>
```

The kicad project files can be download from github PICSimLab repository.

C.1.4 Board code

The header file and c++ code file with comments are listed in the next two subsections. This files control the behavior of board in simulator.

board_x.h

board_x.h online file.
board_x.h online doxygen version.

```
/* ########################################################################

   PICsimLab - PIC laboratory simulator

   ########################################################################

   Copyright (c) : 2015-2021  Luis Claudio Gambôa Lopes

   This program is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation; either version 2, or (at your option)
   any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program; if not, write to the Free Software
   Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

   For e-mail suggestions :  lcgamboa@yahoo.com
   ######################################################################## */

#ifndef BOARD_x_H
#define        BOARD_x_H

#include<lxrad.h>

#include "bsim_picsim.h"

#define        BOARD_x_Name "X"

//new board class must be derived from board class defined in board.h
class cboard_x:public bsim_picsim
{
  private:
     unsigned char p_BT1;           //first board push button in RD0
     unsigned char p_BT2;           //second board switch in RD1

     //value of potentiometer
     unsigned char pot1;
```

```
44      //flag to control if potentiometer is active
45      unsigned char active;
46
47      //controls to be added in simulator window
48      CGauge *gauge1;   //gauge to show mean value of RB0
49      CGauge *gauge2;   //gauge to show mean value of RB1
50      CLabel *label2;   //label of gauge RB0
51      CLabel *label3;   //label of gauge RB1
52
53      //Register controls for remote interface called once on board creation
54      void RegisterRemoteControl(void);
55
56      lxColor color1;//LEDs color 1
57      lxColor color2;//LEDs color 2
58      lxFont font;
59    public:
60       //Constructor called once on board creation
61       cboard_x(void);
62       //Destructor called once on board destruction
63       ~cboard_x(void);
64       //Return the board name
65       lxString GetName(void) {return lxT(BOARD_x_Name); };
66       //Return the about info of board
67       lxString GetAboutInfo(void){return lxT("L.C. Gamboa \n <lcgamboa@yahoo.com>");};
68       //Called ever 100ms to draw board
69       void Draw(CDraw *draw);
70       void Run_CPU(void);
71       //Return a list of board supported microcontrollers
72       lxString GetSupportedDevices(void){return lxT("PIC16F877A,PIC18F4550,PIC18F4620,");};
73       //Reset board status
74       void Reset(void);
75       //Event on the board
76       void EvMouseButtonPress(uint button, uint x, uint y,uint state);
77       //Event on the board
78       void EvMouseButtonRelease(uint button, uint x, uint y,uint state);
79       //Event on the board
80       void EvMouseMove(uint button, uint x, uint y, uint state);
81       //Event on the board
82       void EvKeyPress(uint key,uint mask);
83       //Event on the board
84       void EvKeyRelease(uint key,uint mask);
85       //Called ever 1s to refresh status
86       void RefreshStatus(void);
87       //Called to save board preferences in configuration file
88       void WritePreferences(void);
89       //Called whe configuration file load  preferences
90       void ReadPreferences(char *name,char *value);
```

```
91        //return the input ids numbers of names used in input map
92        unsigned short get_in_id(char * name);
93        //return the output ids numbers of names used in output map
94        unsigned short get_out_id(char * name);
95  };
96
97  #endif        /* BOARD_x_H */
```

board_x.cc

board_x.cc online file.
board_x.cc online doxygen version.

```
/* ########################################################################

   PICsimLab - PIC laboratory simulator

   ########################################################################

   Copyright (c) : 2015-2021  Luis Claudio Gambôa Lopes

   This program is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation; either version 2, or (at your option)
   any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program; if not, write to the Free Software
   Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

   For e-mail suggestions :  lcgamboa@yahoo.com
   ######################################################################## */

//include files
#include"../picsimlab1.h"
#include"../picsimlab4.h" //Oscilloscope
#include"../picsimlab5.h" //Spare Parts
#include"board_x.h"

/* ids of inputs of input map*/
enum
{
 I_POT1, //potentiometer
 I_ICSP, //ICSP connector
 I_PWR, //Power button
 I_RST, //Reset button
 I_BD0, //RD0 push button
 I_SD1 //RD1 switch
};

/* ids of outputs of output map*/
```

```
44  enum
45  {
46   O_POT1, //potentiometer
47   O_RST, //Reset button
48   O_SD1, //switch position (On/Off)
49   O_LD0, //LED on RD0 push button
50   O_LD1, //LED on RD1 switch
51   O_LPWR, //Power LED
52   O_RB0, //LED on RB0 output
53   O_RB1, //LED on RB1 output
54   O_BD0, //RD1 switch
55   O_CPU //CPU name
56  };
57
58  //return the input ids numbers of names used in input map
59
60  unsigned short
61  cboard_x::get_in_id(char * name)
62  {
63   if (strcmp (name, "PG_ICSP") == 0)return I_ICSP;
64   if (strcmp (name, "SW_PWR") == 0)return I_PWR;
65   if (strcmp (name, "PB_RST") == 0)return I_RST;
66   if (strcmp (name, "PB_D0") == 0)return I_BD0;
67   if (strcmp (name, "SW_D1") == 0)return I_SD1;
68   if (strcmp (name, "PO_1") == 0)return I_POT1;
69
70   printf ("Error input '%s' don't have a valid id! \n", name);
71   return -1;
72  }
73
74  //return the output ids numbers of names used in output map
75
76  unsigned short
77  cboard_x::get_out_id(char * name)
78  {
79
80   if (strcmp (name, "SW_D1") == 0)return O_SD1;
81   if (strcmp (name, "LD_LD0") == 0)return O_LD0;
82   if (strcmp (name, "LD_LD1") == 0)return O_LD1;
83   if (strcmp (name, "LD_LPWR") == 0)return O_LPWR;
84   if (strcmp (name, "LD_RB1") == 0)return O_RB1;
85   if (strcmp (name, "LD_RB0") == 0)return O_RB0;
86   if (strcmp (name, "PB_D0") == 0)return O_BD0;
87   if (strcmp (name, "PO_1") == 0)return O_POT1;
88   if (strcmp (name, "PB_RST") == 0)return O_RST;
89   if (strcmp (name, "IC_CPU") == 0)return O_CPU;
90
91   printf ("Error output '%s' don't have a valid id! \n", name);
```

```
 return 1;
}

//Constructor called once on board creation

cboard_x::cboard_x(void) :
font(10, lxFONTFAMILY_TELETYPE, lxFONTSTYLE_NORMAL, lxFONTWEIGHT_BOLD)
{
 Proc = "PIC18F4550"; //default microcontroller if none defined in preferences
 ReadMaps (); //Read input and output board maps

 pot1 = 100;

 active = 0;

 //controls properties and creation
 //gauge1
 gauge1 = new CGauge ();
 gauge1->SetFOwner (&Window1);
 gauge1->SetName (lxT ("gauge1_px"));
 gauge1->SetX (13);
 gauge1->SetY (382 - 160);
 gauge1->SetWidth (140);
 gauge1->SetHeight (20);
 gauge1->SetEnable (1);
 gauge1->SetVisible (1);
 gauge1->SetRange (100);
 gauge1->SetValue (0);
 gauge1->SetType (4);
 Window1.CreateChild (gauge1);
 //gauge2
 gauge2 = new CGauge ();
 gauge2->SetFOwner (&Window1);
 gauge2->SetName (lxT ("gauge2_px"));
 gauge2->SetX (12);
 gauge2->SetY (330 - 160);
 gauge2->SetWidth (140);
 gauge2->SetHeight (20);
 gauge2->SetEnable (1);
 gauge2->SetVisible (1);
 gauge2->SetRange (100);
 gauge2->SetValue (0);
 gauge2->SetType (4);
 Window1.CreateChild (gauge2);
 //label2
 label2 = new CLabel ();
 label2->SetFOwner (&Window1);
```

```
 label2->SetName (lxT ("label2_px"));
 label2->SetX (12);
 label2->SetY (306 - 160);
 label2->SetWidth (60);
 label2->SetHeight (20);
 label2->SetEnable (1);
 label2->SetVisible (1);
 label2->SetText (lxT ("RB0"));
 label2->SetAlign (1);
 Window1.CreateChild (label2);
 //label3
 label3 = new CLabel ();
 label3->SetFOwner (&Window1);
 label3->SetName (lxT ("label3_px"));
 label3->SetX (13);
 label3->SetY (357 - 160);
 label3->SetWidth (60);
 label3->SetHeight (20);
 label3->SetEnable (1);
 label3->SetVisible (1);
 label3->SetText (lxT ("RB1"));
 label3->SetAlign (1);
 Window1.CreateChild (label3);
}

//Destructor called once on board destruction

cboard_x::~cboard_x(void)
{
 //controls destruction
 Window1.DestroyChild (gauge1);
 Window1.DestroyChild (gauge2);
 Window1.DestroyChild (label2);
 Window1.DestroyChild (label3);
}

//Reset board status

void
cboard_x::Reset(void)
{
 pic_reset (1);

 p_BT1 = 1; //set push button  in default state (high)

 //write button state to pic pin 19 (RD0)
 pic_set_pin (19, p_BT1);
```

```
186  //write switch state to pic pin 20 (RD1)
187  pic_set_pin (20, p_BT2);
188
189
190  //verify serial port state and refresh status bar
191 #ifndef _WIN_
192  if (pic.serial[0].serialfd > 0)
193 #else
194  if (pic.serial[0].serialfd != INVALID_HANDLE_VALUE)
195 #endif
196   Window1.statusbar1.SetField (2, lxT ("Serial: ") +
197                                lxString::FromAscii (SERIALDEVICE) + lxT (":") + itoa (pic.serial[0]
198                                lxString ().Format ("%4.1f", fabs ((100.0 * pic.serial[0].serialexbau
199                                                                   pic.serial[0].serialbaud) / pic.s
200  else
201   Window1.statusbar1.SetField (2, lxT ("Serial: ") +
202                                lxString::FromAscii (SERIALDEVICE) + lxT (" (ERROR)"));
203
204  if (use_spare)Window5.Reset ();
205
206  RegisterRemoteControl ();
207 }
208
209 //Register variables to be controled by remote control
210
211 void
212 cboard_x::RegisterRemoteControl(void)
213 {
214  //register inputa
215  input_ids[I_BD0]->status = &p_BT1;
216  input_ids[I_SD1]->status = &p_BT2;
217  input_ids[I_POT1]->status = &pot1;
218
219  //register output to be updated on input change
220  input_ids[I_BD0]->update = &output_ids[O_BD0]->update;
221  input_ids[I_SD1]->update = &output_ids[O_SD1]->update;
222  input_ids[I_POT1]->update = &output_ids[O_POT1]->update;
223
224  //register outputa
225  output_ids[O_RB0]->status = &pic.pins[32].oavalue;
226  output_ids[O_RB1]->status = &pic.pins[33].oavalue;
227  output_ids[O_LD0]->status = &pic.pins[18].oavalue;
228  output_ids[O_LD1]->status = &pic.pins[19].oavalue;
229 }
230
231 //Called ever 1s to refresh status
232
```

```
233 void
234 cboard_x::RefreshStatus(void)
235 {
236  //verify serial port state and refresh status bar
237 #ifndef _WIN_
238  if (pic.serial[0].serialfd > 0)
239 #else
240  if (pic.serial[0].serialfd != INVALID_HANDLE_VALUE)
241 #endif
242   Window1.statusbar1.SetField (2, lxT ("Serial: ") +
243                                lxString::FromAscii (SERIALDEVICE) + lxT (":") + itoa (pic.serial[0]
244                                lxString ().Format ("%4.1f", fabs ((100.0 * pic.serial[0].serialexbau
245                                                                   pic.serial[0].serialbaud) / pic.s
246  else
247   Window1.statusbar1.SetField (2, lxT ("Serial: ") +
248                                lxString::FromAscii (SERIALDEVICE) + lxT (" (ERROR)"));
249
250 }
251
252 //Called to save board preferences in configuration file
253
254 void
255 cboard_x::WritePreferences(void)
256 {
257  //write selected microcontroller of board_x to preferences
258  Window1.saveprefs (lxT ("X_proc"), Proc);
259  //write switch state of board_x to preferences
260  Window1.saveprefs (lxT ("X_bt2"), lxString ().Format ("%i", p_BT2));
261  //write microcontroller clock to preferences
262  Window1.saveprefs (lxT ("X_clock"), lxString ().Format ("%2.1f", Window1.GetClock ()));
263  //write potentiometer position to preferences
264  Window1.saveprefs (lxT ("X_pot1"), lxString ().Format ("%i", pot1));
265 }
266
267 //Called whe configuration file load  preferences
268
269 void
270 cboard_x::ReadPreferences(char *name, char *value)
271 {
272  //read switch state of board_x of preferences
273  if (!strcmp (name, "X_bt2"))
274   {
275    if (value[0] == '0')
276     p_BT2 = 0;
277    else
278     p_BT2 = 1;
279   }
```

```
280  //read microcontroller of preferences
281  if (!strcmp (name, "X_proc"))
282   {
283    Proc = value;
284   }
285  //read microcontroller clock
286  if (!strcmp (name, "X_clock"))
287   {
288    Window1.SetClock (atof (value));
289   }
290
291  //read potentiometer position
292  if (!strcmp (name, "X_pot1"))
293   {
294    pot1 = atoi (value);
295   }
296 }
297
298
299 //Event on the board
300
301 void
302 cboard_x::EvKeyPress(uint key, uint mask)
303 {
304  //if keyboard key 1 is pressed then activate button (state=0)
305  if (key == '1')
306   {
307    p_BT1 = 0;
308    output_ids[O_BD0]->update = 1;
309   }
310
311  //if keyboard key 2 is pressed then toggle switch state
312  if (key == '2')
313   {
314    p_BT2 ^= 1;
315    output_ids[O_SD1]->update = 1;
316   }
317
318 }
319
320 //Event on the board
321
322 void
323 cboard_x::EvKeyRelease(uint key, uint mask)
324 {
325  //if keyboard key 1 is pressed then deactivate button (state=1)
326  if (key == '1')
```

```
  {
   p_BT1 = 1;
   output_ids[O_BD0]->update = 1;
  }

}

//Event on the board

void
cboard_x::EvMouseButtonPress(uint button, uint x, uint y, uint state)
{

 int i;

 //search for the input area which owner the event
 for (i = 0; i < inputc; i++)
  {
   if (((input[i].x1 <= x)&&(input[i].x2 >= x))&&((input[i].y1 <= y)&&
                                                  (input[i].y2 >= y)))
    {

     switch (input[i].id)
      {
       //if event is over I_ISCP area then load hex file
      case I_ICSP:
       Window1.menu1_File_LoadHex_EvMenuActive (NULL);
       break;
       //if event is over I_PWR area then toggle board on/off
      case I_PWR:
       if (Window1.Get_mcupwr ()) //if on turn off
        {
         Window1.Set_mcurun (0);
         Window1.Set_mcupwr (0);
         Reset ();
         p_BT1 = 1;
         Window1.statusbar1.SetField (0, lxT ("Stoped"));
        }
       else //if off turn on
        {
         Window1.Set_mcupwr (1);
         Window1.Set_mcurun (1);
         Reset ();
         Window1.statusbar1.SetField (0, lxT ("Running..."));
        }
       output_ids[O_LPWR]->update = 1;
       break;
```

```
374         //if event is over I_RST area then turn off and reset
375       case I_RST:
376        if (Window1.Get_mcupwr () && pic_reset (-1))//if powered
377         {
378          Window1.Set_mcupwr (0);
379          Window1.Set_mcurst (1);
380         }
381        p_RST = 0;
382        output_ids[O_RST]->update = 1;
383        break;
384        //if event is over I_D0 area then activate button (state=0)
385       case I_BD0:
386        p_BT1 = 0;
387        output_ids[O_BD0]->update = 1;
388        break;
389        //if event is over I_D1 area then toggle switch state
390       case I_SD1:
391        p_BT2 ^= 1;
392        output_ids[O_SD1]->update = 1;
393        break;
394       case I_POT1:
395        {
396         active = 1;
397         pot1 = (x - input[i].x1)*2.77;
398         if (pot1 > 199)pot1 = 199;
399         output_ids[O_POT1]->update = 1;
400        }
401        break;
402       }
403     }
404   }
405
406 }
407
408 //Event on the board
409
410 void
411 cboard_x::EvMouseMove(uint button, uint x, uint y, uint state)
412 {
413  int i;
414
415  for (i = 0; i < inputc; i++)
416   {
417    switch (input[i].id)
418     {
419     case I_POT1:
420      if (((input[i].x1 <= x)&&(input[i].x2 >= x))&&((input[i].y1 <= y)&&(input[i].y2 >= y)))
```

```
      {
       if (active)
        {
         pot1 = (x - input[i].x1)*2.77;
         if (pot1 > 199)pot1 = 199;
         output_ids[O_POT1]->update = 1;
        }
      }
     break;
    }
  }
}

//Event on the board

void
cboard_x::EvMouseButtonRelease(uint button, uint x, uint y, uint state)
{
 int i;

 //search for the input area which owner the event
 for (i = 0; i < inputc; i++)
  {
   if (((input[i].x1 <= x)&&(input[i].x2 >= x))&&((input[i].y1 <= y)&&
                                                  (input[i].y2 >= y)))
    {
     switch (input[i].id)
      {
       //if event is over I_RST area then turn on
      case I_RST:
       if (Window1.Get_mcurst ())//if powered
        {
         Window1.Set_mcupwr (1);
         Window1.Set_mcurst (0);

         if (pic_reset (-1))
          {
           Reset ();
          }
        }
       p_RST = 1;
       output_ids[O_RST]->update = 1;
       break;
       //if event is over I_D0 area then deactivate button (state=1)
      case I_BD0:
       p_BT1 = 1;
       output_ids[O_BD0]->update = 1;
```

```
468         break;
469       case I_POT1:
470        {
471         active = 0;
472         output_ids[O_POT1]->update = 1;
473        }
474        break;
475       }
476     }
477   }
478
479 }
480
481
482 //Called ever 100ms to draw board
483 //This is the critical code for simulator running speed
484
485 void
486 cboard_x::Draw(CDraw *draw)
487 {
488  int update = 0; //verifiy if updated is needed
489  int i;
490
491
492  //board_x draw
493  for (i = 0; i < outputc; i++) //run over all outputs
494   {
495    if (output[i].update)//only if need update
496     {
497      output[i].update = 0;
498
499      if (!update)
500       {
501        draw->Canvas.Init (Scale, Scale);
502       }
503      update++; //set to update buffer
504
505      if (!output[i].r)//if output shape is a rectangle
506       {
507        if (output[i].id == O_SD1)//if output is switch
508         {
509          //draw a background white rectangle
510          draw->Canvas.SetBgColor (255, 255, 255);
511          draw->Canvas.Rectangle (1, output[i].x1, output[i].y1,
512                                  output[i].x2 - output[i].x1, output[i].y2 - output[i].y1);
513
514          if (!p_BT2) //draw switch off
```

```
515           {
516            //draw a grey rectangle
517            draw->Canvas.SetBgColor (70, 70, 70);
518            draw->Canvas.Rectangle (1, output[i].x1, output[i].y1 +
519                                    ((int) ((output[i].y2 - output[i].y1)*0.35)), output[i].x2 - outp
520                                    (int) ((output[i].y2 - output[i].y1)*0.65));
521           }
522          else //draw switch on
523           {
524            //draw a grey rectangle
525            draw->Canvas.SetBgColor (70, 70, 70);
526            draw->Canvas.Rectangle (1, output[i].x1,
527                                    output[i].y1, output[i].x2 - output[i].x1,
528                                    (int) ((output[i].y2 - output[i].y1)*0.65));
529           }
530         }
531        else if (output[i].id == O_BD0)
532         {
533          draw->Canvas.SetColor (102, 102, 102);
534          draw->Canvas.Circle (1, output[i].cx, output[i].cy, 10);
535          if (p_BT1)
536           {
537            draw->Canvas.SetColor (15, 15, 15);
538           }
539          else
540           {
541            draw->Canvas.SetColor (55, 55, 55);
542           }
543          draw->Canvas.Circle (1, output[i].cx, output[i].cy, 8);
544         }
545        else if (output[i].id == O_RST)
546         {
547          draw->Canvas.SetColor (102, 102, 102);
548          draw->Canvas.Circle (1, output[i].cx, output[i].cy, 10);
549
550          if (p_RST)
551           {
552            draw->Canvas.SetColor (15, 15, 15);
553           }
554          else
555           {
556            draw->Canvas.SetColor (55, 55, 55);
557           }
558          draw->Canvas.Circle (1, output[i].cx, output[i].cy, 8);
559         }
560        else if (output[i].id == O_POT1)
561         {
```

```
562           draw->Canvas.SetColor (0, 50, 215);
563           draw->Canvas.Rectangle (1, output[i].x1, output[i].y1, output[i].x2 - output[i].x1, output
564           draw->Canvas.SetColor (250, 250, 250);
565           draw->Canvas.Rectangle (1, output[i].x1 + pot1 / 2.77, output[i].y1 + 2, 10, 15);
566          }
567        else if (output[i].id == O_CPU)
568          {
569           draw->Canvas.SetFont (font);
570           int x, y, w, h;
571           draw->Canvas.SetColor (26, 26, 26);
572           draw->Canvas.Rectangle (1, output[i].x1, output[i].y1, output[i].x2 - output[i].x1, output
573
574           draw->Canvas.SetColor (230, 230, 230);
575           w = output[i].x2 - output[i].x1;
576           h = output[i].y2 - output[i].y2;
577           x = output[i].x1 + (w / 2) + 7;
578           y = output[i].y1 + (h / 2) + (Proc.length ());
579           draw->Canvas.RotatedText (Proc, x, y, 270);
580          }
581       }
582      else //if output shape is a circle
583       {
584        draw->Canvas.SetFgColor (0, 0, 0); //black
585
586        switch (output[i].id)//search for color of output
587         {
588         case O_LD0: //White using pin 19 mean value (RD0)
589          draw->Canvas.SetBgColor (pic.pins[18].oavalue, pic.pins[18].oavalue, pic.pins[18].oavalue);
590          break;
591         case O_LD1: //Yelllow using pin 20 mean value (RD1)
592          draw->Canvas.SetBgColor (pic.pins[19].oavalue, pic.pins[19].oavalue, 0);
593          break;
594         case O_LPWR: //Blue using mcupwr value
595          draw->Canvas.SetBgColor (0, 0, 200 * Window1.Get_mcupwr () + 55);
596          break;
597         case O_RB0: //Green using pin 33 mean value (RB0)
598          draw->Canvas.SetBgColor (0, pic.pins[32].oavalue, 0);
599          break;
600         case O_RB1: //Red using pin 34 mean value (RB1)
601          draw->Canvas.SetBgColor (pic.pins[33].oavalue, 0, 0);
602          break;
603         }
604
605        //draw a LED
606        color1 = draw->Canvas.GetBgColor ();
607        int r = color1.Red () - 120;
608        int g = color1.Green () - 120;
```

```
609         int b = color1.Blue () - 120;
610         if (r < 0)r = 0;
611         if (g < 0)g = 0;
612         if (b < 0)b = 0;
613         color2.Set (r, g, b);
614         draw->Canvas.SetBgColor (color2);
615         draw->Canvas.Circle (1, output[i].x1, output[i].y1, output[i].r + 1);
616         draw->Canvas.SetBgColor (color1);
617         draw->Canvas.Circle (1, output[i].x1, output[i].y1, output[i].r - 2);
618        }
619      }
620    }
621   //end draw
622
623   if (update)
624    {
625     draw->Canvas.End ();
626     draw->Update ();
627    }
628
629   //RB0 mean value to gauge1
630   gauge1->SetValue ((pic.pins[33].oavalue - 55) / 2);
631   //RB1 mean value to gauge2
632   gauge2->SetValue ((pic.pins[32].oavalue - 55) / 2);
633
634 }
635
636 void
637 cboard_x::Run_CPU(void)
638 {
639   int i;
640   int j;
641   unsigned char pi;
642   const picpin * pins;
643   unsigned int alm[40];
644
645   int JUMPSTEPS = Window1.GetJUMPSTEPS (); //number of steps skipped
646   long int NSTEP = Window1.GetNSTEP () / MGetPinCount (); //number of steps in 100ms
647
648
649   //reset pins mean value
650   memset (alm, 0, 40 * sizeof (unsigned int));
651
652   //read pic.pins to a local variable to speed up
653   pins = pic.pins;
654
655   //Spare parts window pre process
```

```
656  if (use_spare)Window5.PreProcess ();
657
658  j = JUMPSTEPS; //step counter
659  if (Window1.Get_mcupwr ()) //if powered
660   for (i = 0; i < Window1.GetNSTEP (); i++) //repeat for number of steps in 100ms
661    {
662
663     if (j >= JUMPSTEPS)//if number of step is bigger than steps to skip
664      {
665       pic_set_pin (pic.mclr, p_RST);
666       pic_set_pin (19, p_BT1); //Set pin 19 (RD0) with button state
667       pic_set_pin (20, p_BT2); //Set pin 20 (RD1) with switch state
668      }
669
670     //verify if a breakpoint is reached if not run one instruction
671     if (!mplabxd_testbp ())pic_step ();
672     //Oscilloscope window process
673     if (use_oscope)Window4.SetSample ();
674     //Spare parts window process
675     if (use_spare)Window5.Process ();
676
677     //increment mean value counter if pin is high
678     alm[i % pic.PINCOUNT] += pins[i % pic.PINCOUNT].value;
679
680     if (j >= JUMPSTEPS)//if number of step is bigger than steps to skip
681      {
682
683       //set analog pin 2 (AN0) with value from scroll
684       pic_set_apin (2, (5.0 * pot1 / 199));
685
686       j = -1; //reset counter
687      }
688     j++; //counter increment
689    }
690
691  //calculate mean value
692  for (pi = 0; pi < pic.PINCOUNT; pi++)
693   {
694    pic.pins[pi].oavalue = (int) (((200.0 * alm[pi]) / NSTEP) + 55);
695   }
696
697  //Spare parts window pre post process
698  if (use_spare)Window5.PostProcess ();
699
700  //verifiy if LEDS need update
701  if (output_ids[O_LD0]->value != pic.pins[18].oavalue)
702   {
```

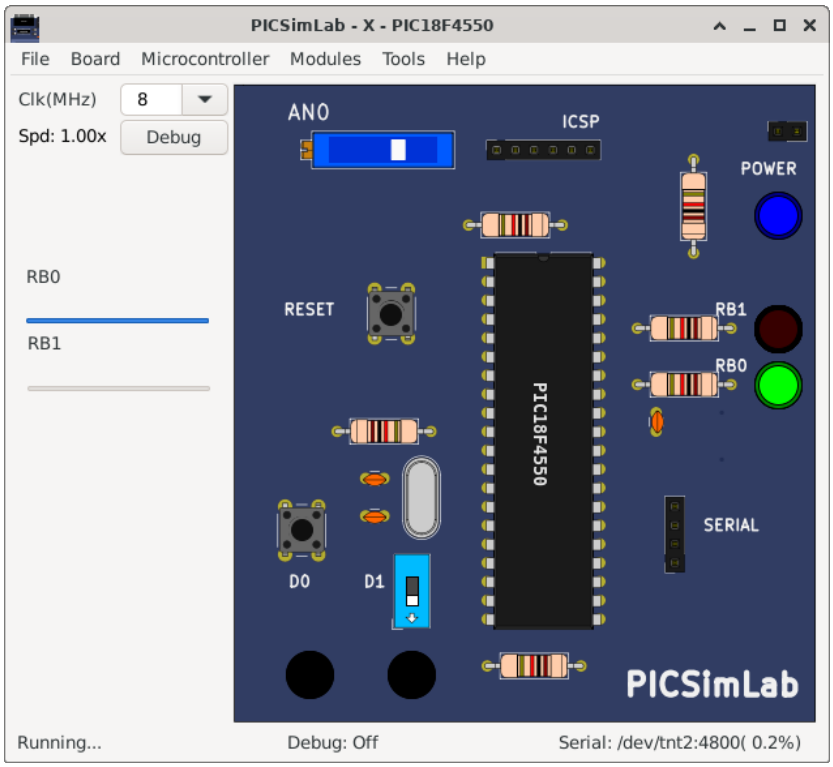
```
703     output_ids[O_LD0]->value = pic.pins[18].oavalue;
704     output_ids[O_LD0]->update = 1;
705    }
706   if (output_ids[O_LD1]->value != pic.pins[19].oavalue)
707    {
708     output_ids[O_LD1]->value = pic.pins[19].oavalue;
709     output_ids[O_LD1]->update = 1;
710    }
711   if (output_ids[O_RB0]->value != pic.pins[32].oavalue)
712    {
713     output_ids[O_RB0]->value = pic.pins[32].oavalue;
714     output_ids[O_RB0]->update = 1;
715    }
716   if (output_ids[O_RB1]->value != pic.pins[33].oavalue)
717    {
718     output_ids[O_RB1]->value = pic.pins[33].oavalue;
719     output_ids[O_RB1]->update = 1;
720    }
721
722 }
723
724
725 //Register the board in PICSimLab
726 board_init(BOARD_x_Name, cboard_x);
```

C.1.5 Integration with PICsimLab

After include the four files created for new board, the PICSimLab can be recompiled, as described in Install from source.

C.1.6 Final Result

The PICSimLab board created for this tutorial are shown in the figure below.



The sample program below can be used to test new board, this code is write for XC8 compiler:

```
#include <xc.h>;

#include "config_4550.h"
#include "adc.h"
#include "serial.h"
#include "itoa.h"

void main()
{
  unsigned int  val;
  char buffer[10];

  ADCON1=0x02;
  TRISA=0xFF;
  TRISB=0xFC;
  TRISC=0xBF;
  TRISD=0xFF;
  TRISE=0x0F;

```

```
20    adc_init();
21    serial_init();
22
23
24    while(1)
25    {
26        val=adc_amostra(0);
27
28        if(PORTDbits.RD1)
29        {
30          if(val > 340)
31             PORTBbits.RB0=1;
32          else
33            PORTBbits.RB0=0;
34
35          if(val > 680)
36            PORTBbits.RB1=1;
37          else
38            PORTBbits.RB1=0;
39        }
40        else
41        {
42            if(PORTDbits.RD0)
43            {
44                PORTBbits.RB0=1;
45                PORTBbits.RB1=0;
46            }
47            else
48            {
49                PORTBbits.RB0=0;
50                PORTBbits.RB1=1;
51            }
52
53        }
54
55        serial_tx_str(itoa(val,buffer));
56        serial_tx_str("\r\n");
57    }
58
59  }
```