How to Compile PICSimLab and Create New Boards

Luis Claudio Gambôa Lopes <lcgamboa@yahoo.com>

http://sourceforge.net/projects/picsim/

August 17, 2020

Contents

Chapter 1

How to Compile PICsimLab

1.1 In Debian Linux and derivatives

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

To build experimental version use the argument "exp" with the *picsimlab_build_all_and_deps.sh* script

1.2 Cross-compiling for windows

For Windows 64 bits version from Debian Linux and derivatives or WSL (Windows Subsystem for Linux) on win10

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

For 32 bits version:

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

To build experimental version use the argument "exp" with the scripts.

Chapter 2

Creating a New Board

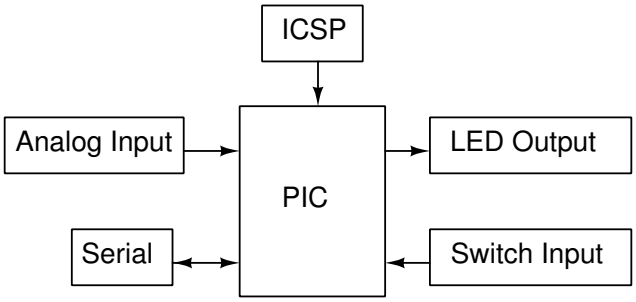
The first step is get the schematic and all information about the board hardware. The second step is the creation of five files in PICSimLab dir (consider replace the 'x' of board_x for a number or name in your case):

- Board Picture (share/boards/x/board.png);
- Board input map (share/boards/x/input.map);
- Board output map (share/boards/x/output.map);
- Board header (src/boards/board_x.h);
- Board C++ code (src/boards/board_x.cc);

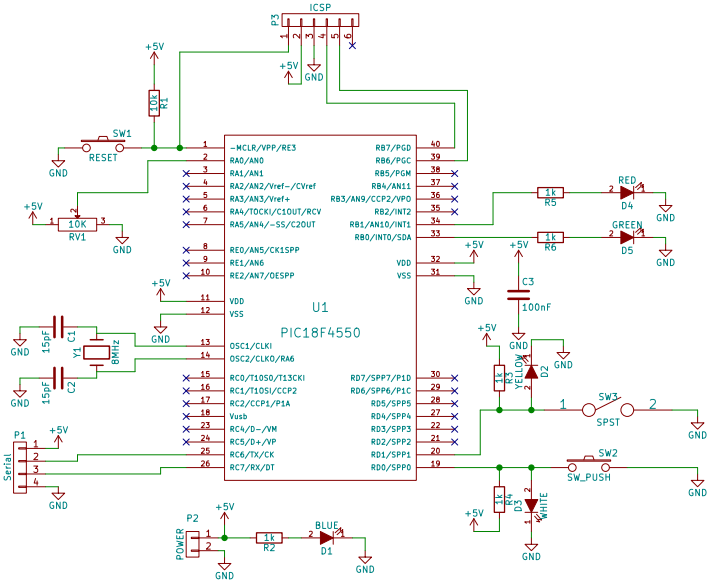
The third and last step is recompiling PICSimLab with new board support.

2.1 Board Hardware and Schematic

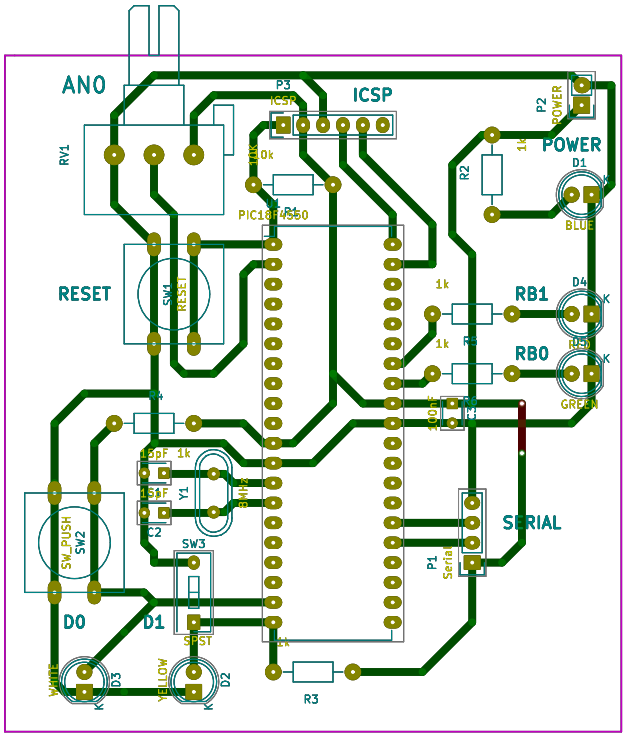
For this tutorial, the board created have the hardware shown in diagram below:



The schematic for the tutorial board made in Kicad.

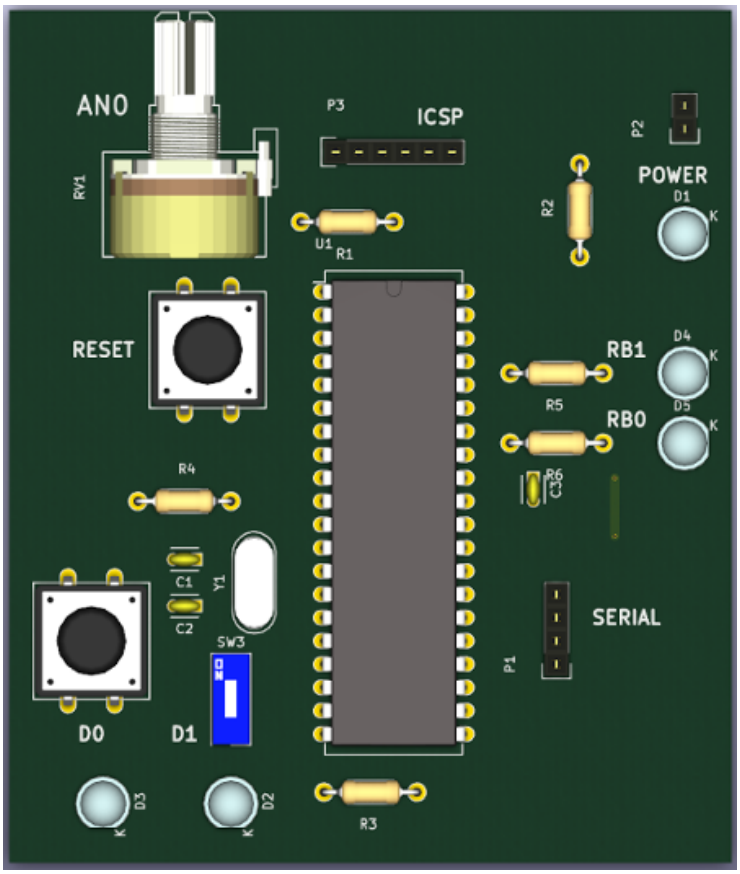

ICSP
P3
+5V
GND
10K
R1
SW1
RESET
U1
PIC18F4550
-MCLR/VPP/RE3
RA0/AN0
RA1/AN1
RA2/AN2/Vref-/CVref
RA3/AN3/Vref+
RA4/T0CKI/C1OUT/RCV
RA5/AN4/-SS/C2OUT
RE0/AN5/CK1SPP
RE1/AN6
RE2/AN7/OESPP
VDD
VSS
OSC1/CLKI
OSC2/CLKO/RA6
RC0/T1OSO/T13CKI
RC1/T1OSI/CCP2
RC2/CCP1/P1A
Vusb
RC4/D-/VM
RC5/D+/VP
RC6/TX/CK
RC7/RX/DT
RB7/PGD
RB6/PGC
RB5/PGM
RB4/AN11
RB3/AN9/CCP2/VPO
RB2/INT2
RB1/AN10/INT1
RB0/INT0/SDA
VDD
VSS
10K
RV1
15pF
C1
C2
8Mhz
Y1
P1
Serial
P2
POWER
1k
R2
BLUE
D1
1k
R5
RED
D4
1k
R6
GREEN
D5
C3
100nF
1k
R3
YELLOW
D2
SW3
SPST
SW2
SW_PUSH
1k
R4
D3
WHITE

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



2.2 Board Picture

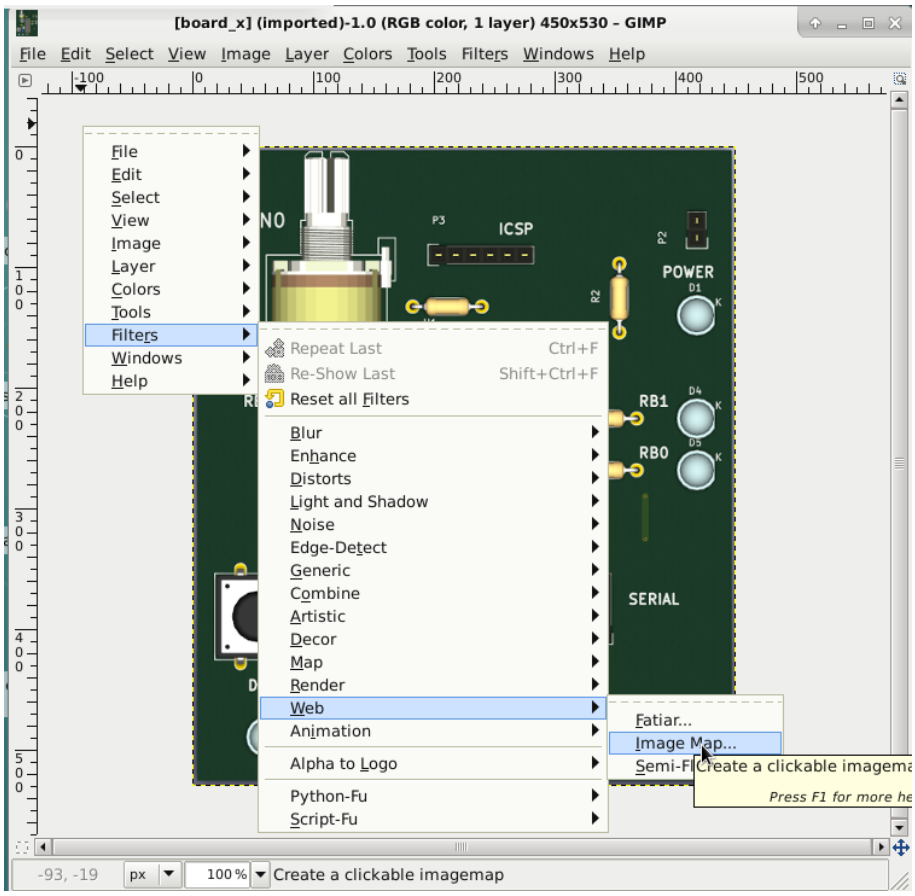
Because the real board of this tutorial never has been built, the board picture was taken from Kicad 3D viewer. The picture image is saved as “share/board/x/board.png”.



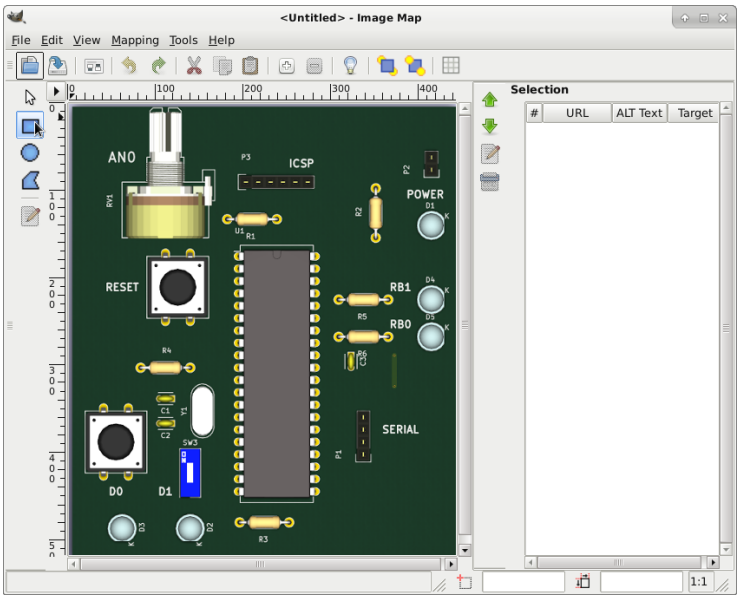
2.3 Picture maps

The PICSimLab use two type of image maps. The input map mark the areas in board picture which user can interact (by mouse click). The output map mark the areas in board picture to be redraw according simulator status. The picture maps 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.

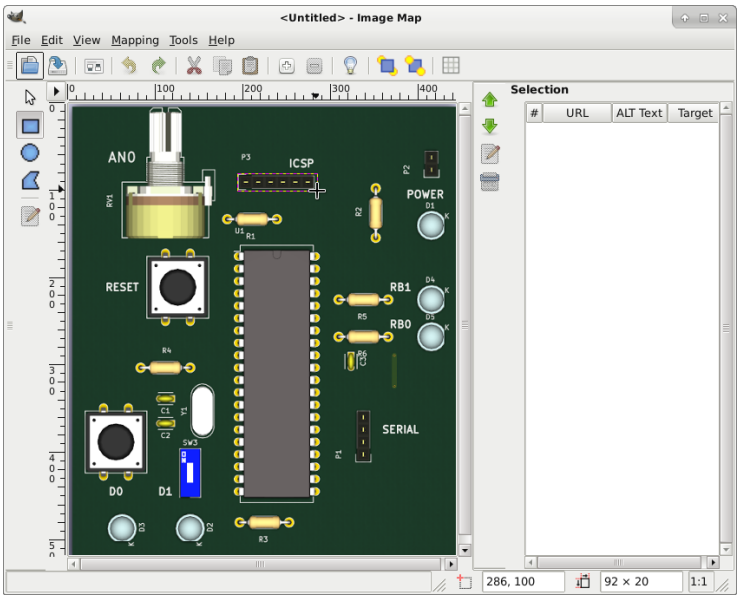
To start, in the GIMP, use the Filters->Web->Image Map to open image map editor window.



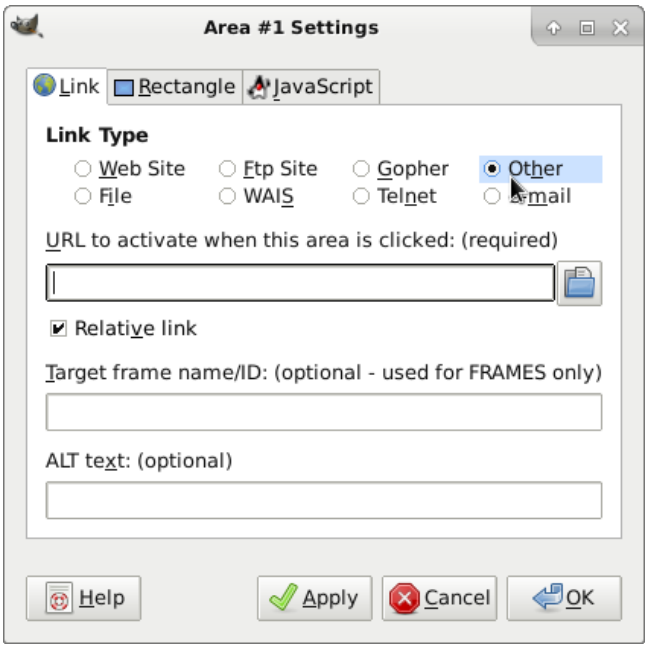
Then select rectangle or circle map on toolbar.



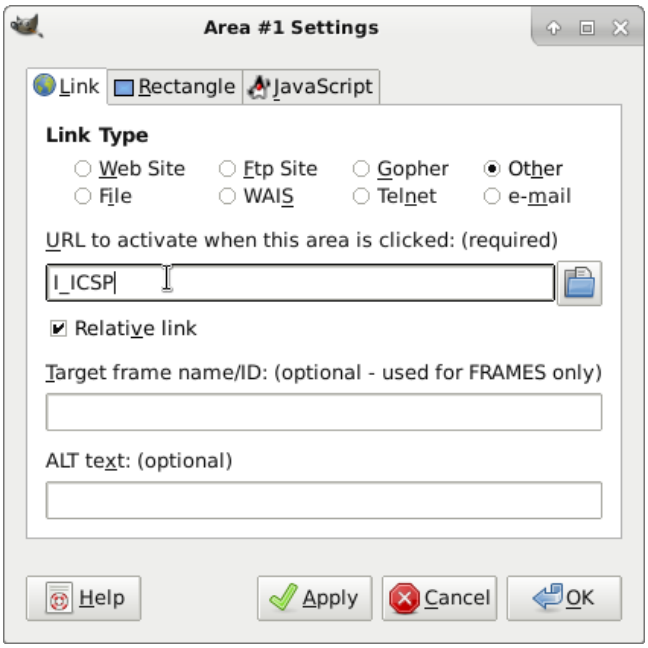
And mark the area in picture.



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



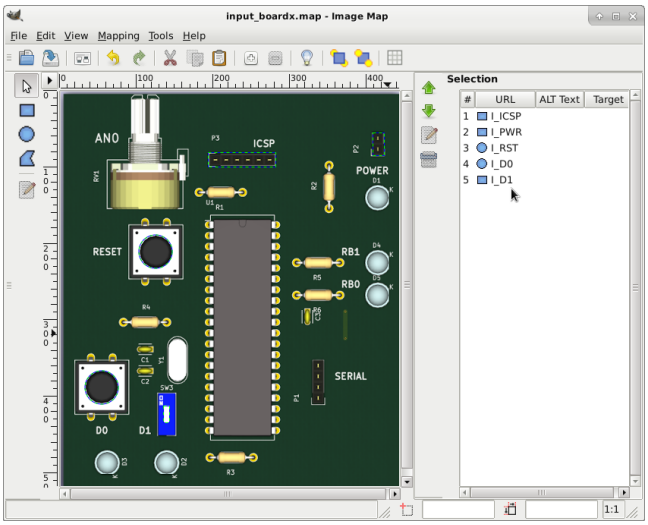
And write the name of area. The name must describe the area function on the board.



2.3.1 Input map

For this tutorial board, five input areas are marked:

- I_ICSP - where user click to load hexfile.
- I_PWR - where user click to turn on/off the board.
- I_RST - Button to reset board.
- I_D0 - Button connected in RD0.
- I_D1 - Switch connected in RD1.



Input map generated by Gimp image map editor and saved as “share/boards/x/input.map”.

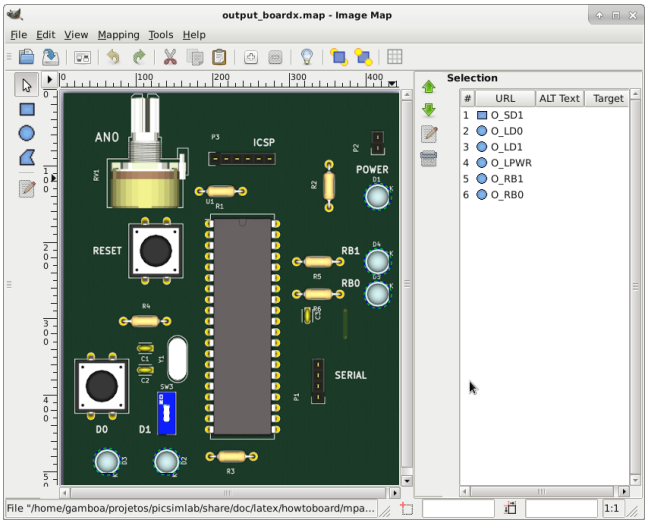
```
<img src="[board_x](imported)" width="450" height="530" border="0" usemap="#map" />

<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="194,80,283,99" href="I_ICSP" />
<area shape="rect" coords="411,54,426,84" href="I_PWR" />
<area shape="circle" coords="125,211,22" href="I_RST" />
<area shape="circle" coords="54,390,22" href="I_D0" />
<area shape="rect" coords="135,414,143,436" href="I_D1" />
</map>
```

2.3.2 Output map

For this tutorial board, six output areas are marked:

- O_SD1 - draw the switch on/off.
- O_LD0 - draw LED connected in button.
- O_LD1 - draw LED connected in switch.
- O_LPWR - draw power LED indicator.
- O_RB0 and O_RB1 - draw LEDs connected in RB0 and RB1.



Output map generated by Gimp image map editor and saved as “share/boards/x/output.map”.

```
<img src="[board_x](imported)" width="450" height="530" 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="135,414,143,436" href="O_SD1" />
<area shape="circle" coords="61,489,17" href="O_LD0" />
<area shape="circle" coords="140,489,17" href="O_LD1" />
<area shape="circle" coords="418,140,17" href="O_LPWR" />
<area shape="circle" coords="418,226,17" href="O_RB1" />
<area shape="circle" coords="418,269,17" href="O_RB0" />
</map>
```

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

2.4 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.

2.4.1 board_x.h

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

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

   PICsimLab - PIC laboratory simulator

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

   Copyright (c) : 2015-2020  Luis Claudio GambÃ´a 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 "board_picsim.h"

//new board class must be derived from board class defined in board.h
class cboard_x:public board_picsim
{
```

```
36  private:
37     int p_BT1;           //first board switch in RD0
38     int p_BT2;           //second board switch in RD1
39
40     //controls to be added in simulator window
41     CScroll *scroll1; //scroll for analog input AN0
42     CGauge *gauge1;   //gauge to show mean value of RB0
43     CGauge *gauge2;   //gauge to show mean value of RB1
44     CLabel *label1;   //label of scroll AN0
45     CLabel *label2;   //label of gauge RB0
46     CLabel *label3;   //label of gauge RB1
47
48   public:
49      //Constructor called once on board creation
50      cboard_x(void);
51      //Destructor called once on board destruction
52      ~cboard_x(void);
53      //Return the about info of board
54      String GetAboutInfo(void){return lxT("L.C. Gamboa \n <lcgamboa@yahoo.com>");};
55      //Called ever 100ms to draw board
56      void Draw(CDraw *draw,double scale);
57      void Run_CPU(void);
58      //Return a list of board supported microcontrollers
59      String GetSupportedDevices(void){return lxT("PIC18F4550,PIC16F877A,");};
60      //Return the filename of board picture
61      String GetPictureFileName(void){return lxT("x/board.png");};
62      //Return the filename of board picture input map
63      String GetInputMapFile(void){return lxT("x/input.map");};
64      //Return the filename of board picture output map
65      String GetOutputMapFile(void){return lxT("x/output.map");};
66      //Reset board status
67      void Reset(void);
68      //Event on the board
69      void EvMouseButtonPress(uint button, uint x, uint y,uint state);
70      //Event on the board
71      void EvMouseButtonRelease(uint button, uint x, uint y,uint state);
72      //Event on the board
73      void EvKeyPress(uint key,uint mask);
74      //Event on the board
75      void EvKeyRelease(uint key,uint mask);
76      void EvOnShow(void){};
77      //Called ever 1s to refresh status
78      void RefreshStatus(void);
79      //Called to save board preferences in configuration file
80      void WritePreferences(void);
81      //Called whe configuration file load  preferences
82      void ReadPreferences(char *name,char *value);
83      //return the input ids numbers of names used in input map
```

```
84          unsigned short get_in_id(char * name);
85          //return the output ids numbers of names used in output map
86          unsigned short get_out_id(char * name);
87  };
88
89  #endif          /* BOARD_x_H */
```

2.4.2 board_x.cc

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

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

   PICsimLab - PIC laboratory simulator

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

   Copyright (c) : 2015-2020  Luis Claudio GambÃ´a 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_ICSP, //ICSP connector
 I_PWR, //Power button
 I_RST, //Reset button
 I_D0, //RD0 push button
 I_D1 //RD1 switch
};

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

```
 O_SD1, //switch position (On/Off)
 O_LD0, //LED on RD0 push button
 O_LD1, //LED on RD1 switch
 O_LPWR, //Power LED
 O_RB0, //LED on RB0 output
 O_RB1 //LED on RB1 output
};
//return the input ids numbers of names used in input map

unsigned short
cboard_x::get_in_id(char * name)
{
 if (strcmp (name, "I_ICSP") == 0)return I_ICSP;
 if (strcmp (name, "I_PWR") == 0)return I_PWR;
 if (strcmp (name, "I_RST") == 0)return I_RST;
 if (strcmp (name, "I_D0") == 0)return I_D0;
 if (strcmp (name, "I_D1") == 0)return I_D1;

 printf ("Error input '%s' don't have a valid id! \n", name);
 return -1;
}

//return the output ids numbers of names used in output map

unsigned short
cboard_x::get_out_id(char * name)
{

 if (strcmp (name, "O_SD1") == 0)return O_SD1;
 if (strcmp (name, "O_LD0") == 0)return O_LD0;
 if (strcmp (name, "O_LD1") == 0)return O_LD1;
 if (strcmp (name, "O_LPWR") == 0)return O_LPWR;
 if (strcmp (name, "O_RB1") == 0)return O_RB1;
 if (strcmp (name, "O_RB0") == 0)return O_RB0;

 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)
{
 Proc = "PIC18F4550"; //default microcontroller if none defined in preferences
 ReadMaps (); //Read input and output board maps

 //controls properties and creation
 //scroll1
```

```
93  scroll1 = new CScroll ();
94  scroll1->SetFOwner (&Window1);
95  scroll1->SetName (lxT ("scroll1_px"));
96  scroll1->SetX (12);
97  scroll1->SetY (273 - 160);
98  scroll1->SetWidth (140);
99  scroll1->SetHeight (22);
100 scroll1->SetEnable (1);
101 scroll1->SetVisible (1);
102 scroll1->SetRange (100);
103 scroll1->SetPosition (50);
104 scroll1->SetType (4);
105 Window1.CreateChild (scroll1);
106 //gauge1
107 gauge1 = new CGauge ();
108 gauge1->SetFOwner (&Window1);
109 gauge1->SetName (lxT ("gauge1_px"));
110 gauge1->SetX (13);
111 gauge1->SetY (382 - 160);
112 gauge1->SetWidth (140);
113 gauge1->SetHeight (20);
114 gauge1->SetEnable (1);
115 gauge1->SetVisible (1);
116 gauge1->SetRange (100);
117 gauge1->SetValue (0);
118 gauge1->SetType (4);
119 Window1.CreateChild (gauge1);
120 //gauge2
121 gauge2 = new CGauge ();
122 gauge2->SetFOwner (&Window1);
123 gauge2->SetName (lxT ("gauge2_px"));
124 gauge2->SetX (12);
125 gauge2->SetY (330 - 160);
126 gauge2->SetWidth (140);
127 gauge2->SetHeight (20);
128 gauge2->SetEnable (1);
129 gauge2->SetVisible (1);
130 gauge2->SetRange (100);
131 gauge2->SetValue (0);
132 gauge2->SetType (4);
133 Window1.CreateChild (gauge2);
134 //label1
135 label1 = new CLabel ();
136 label1->SetFOwner (&Window1);
137 label1->SetName (lxT ("label1_px"));
138 label1->SetX (12);
139 label1->SetY (249 - 160);
140 label1->SetWidth (60);
```

```
141   label1->SetHeight (20);
142   label1->SetEnable (1);
143   label1->SetVisible (1);
144   label1->SetText (lxT ("AN0"));
145   label1->SetAlign (1);
146   Window1.CreateChild (label1);
147   //label2
148   label2 = new CLabel ();
149   label2->SetFOwner (&Window1);
150   label2->SetName (lxT ("label2_px"));
151   label2->SetX (12);
152   label2->SetY (306 - 160);
153   label2->SetWidth (60);
154   label2->SetHeight (20);
155   label2->SetEnable (1);
156   label2->SetVisible (1);
157   label2->SetText (lxT ("RB0"));
158   label2->SetAlign (1);
159   Window1.CreateChild (label2);
160   //label3
161   label3 = new CLabel ();
162   label3->SetFOwner (&Window1);
163   label3->SetName (lxT ("label3_px"));
164   label3->SetX (13);
165   label3->SetY (357 - 160);
166   label3->SetWidth (60);
167   label3->SetHeight (20);
168   label3->SetEnable (1);
169   label3->SetVisible (1);
170   label3->SetText (lxT ("RB1"));
171   label3->SetAlign (1);
172   Window1.CreateChild (label3);
173  }
174
175  //Destructor called once on board destruction
176
177  cboard_x::~cboard_x(void)
178  {
179   //controls destruction
180   Window1.DestroyChild (scroll1);
181   Window1.DestroyChild (gauge1);
182   Window1.DestroyChild (gauge2);
183   Window1.DestroyChild (label1);
184   Window1.DestroyChild (label2);
185   Window1.DestroyChild (label3);
186  }
187
188  //Reset board status
```

```
189
190 void
191 cboard_x::Reset(void)
192 {
193  pic_reset (1);
194
195  p_BT1 = 1; //set push button  in default state (high)
196
197  //write button state to pic pin 19 (RD0)
198  pic_set_pin (19, p_BT1);
199  //write switch state to pic pin 20 (RD1)
200  pic_set_pin (20, p_BT2);
201
202
203  //verify serial port state and refresh status bar
204 #ifndef _WIN_
205  if (pic.serial[0].serialfd > 0)
206 #else
207  if (pic.serial[0].serialfd != INVALID_HANDLE_VALUE)
208 #endif
209    Window1.statusbar1.SetField (2, lxT ("Serial: ") +
210                                 String::FromAscii (SERIALDEVICE) + lxT (":") + itoa (pic.serial[0].se
211                                 String ().Format ("%4.1f", fabs ((100.0 * pic.serial[0].serialexbaud
212                                                                  pic.serial[0].serialbaud) / pic.ser
213  else
214    Window1.statusbar1.SetField (2, lxT ("Serial: ") +
215                                 String::FromAscii (SERIALDEVICE) + lxT (" (ERROR)"));
216
217  if (use_spare)Window5.Reset ();
218 }
219
220 //Called ever 1s to refresh status
221
222 void
223 cboard_x::RefreshStatus(void)
224 {
225  //verify serial port state and refresh status bar
226 #ifndef _WIN_
227  if (pic.serial[0].serialfd > 0)
228 #else
229  if (pic.serial[0].serialfd != INVALID_HANDLE_VALUE)
230 #endif
231    Window1.statusbar1.SetField (2, lxT ("Serial: ") +
232                                 String::FromAscii (SERIALDEVICE) + lxT (":") + itoa (pic.serial[0].se
233                                 String ().Format ("%4.1f", fabs ((100.0 * pic.serial[0].serialexbaud
234                                                                  pic.serial[0].serialbaud) / pic.ser
235  else
236    Window1.statusbar1.SetField (2, lxT ("Serial: ") +
```

```
237                                 String::FromAscii (SERIALDEVICE) + lxT (" (ERROR)"));
238
239 }
240
241 //Called to save board preferences in configuration file
242
243 void
244 cboard_x::WritePreferences(void)
245 {
246  //write selected microcontroller of board_x to preferences
247  Window1.saveprefs (lxT ("X_proc"), Proc);
248  //write switch state of board_x to preferences
249  Window1.saveprefs (lxT ("X_bt2"), String ().Format ("%i", p_BT2));
250  //write microcontroller clock to preferences
251  Window1.saveprefs (lxT ("X_clock"), String ().Format ("%2.1f", Window1.GetClock()));
252 }
253
254 //Called whe configuration file load  preferences
255
256 void
257 cboard_x::ReadPreferences(char *name, char *value)
258 {
259  //read switch state of board_x of preferences
260  if (!strcmp (name, "X_bt2"))
261   {
262    if (value[0] == '0')
263     p_BT2 = 0;
264    else
265     p_BT2 = 1;
266   }
267  //read microcontroller of preferences
268  if (!strcmp (name, "X_proc"))
269   {
270    Proc = value;
271   }
272  //read microcontroller clock
273  if (!strcmp (name, "X_clock"))
274  {
275   Window1.SetClock (atof(value));
276  }
277 }
278
279
280 //Event on the board
281
282 void
283 cboard_x::EvKeyPress(uint key, uint mask)
284 {
```

```
285   //if keyboard key 1 is pressed then activate button (state=0)
286   if (key == '1')
287    {
288     p_BT1 = 0;
289    }
290
291   //if keyboard key 2 is pressed then toggle switch state
292   if (key == '2')
293    {
294     p_BT2 ^= 1;
295    }
296
297  }
298
299  //Event on the board
300
301  void
302  cboard_x::EvKeyRelease(uint key, uint mask)
303  {
304   //if keyboard key 1 is pressed then deactivate button (state=1)
305   if (key == '1')
306    {
307     p_BT1 = 1;
308    }
309
310  }
311
312  //Event on the board
313
314  void
315  cboard_x::EvMouseButtonPress(uint button, uint x, uint y, uint state)
316  {
317
318   int i;
319
320   //search for the input area which owner the event
321   for (i = 0; i < inputc; i++)
322    {
323     if (((input[i].x1 <= x)&&(input[i].x2 >= x))&&((input[i].y1 <= y)&&
324                                                    (input[i].y2 >= y)))
325      {
326
327       switch (input[i].id)
328        {
329         //if event is over I_ISCP area then load hex file
330        case I_ICSP:
331         Window1.menu1_File_LoadHex_EvMenuActive (NULL);
332         break;
```

```
333         //if event is over I_PWR area then toggle board on/off
334        case I_PWR:
335         if (Window1.Get_mcupwr ()) //if on turn off
336          {
337           Window1.Set_mcurun (0);
338           Window1.Set_mcupwr (0);
339           Reset ();
340           p_BT1 = 1;
341           Window1.statusbar1.SetField (0, lxT ("Stoped"));
342          }
343         else //if off turn on
344          {
345           Window1.Set_mcupwr (1);
346           Window1.Set_mcurun (1);
347           Reset ();
348           Window1.statusbar1.SetField (0, lxT ("Running..."));
349          }
350         break;
351         //if event is over I_RST area then turn off and reset
352        case I_RST:
353         if (Window1.Get_mcupwr () && pic_reset (-1))//if powered
354          {
355           Window1.Set_mcupwr (0);
356           Window1.Set_mcurst (1);
357          }
358         p_MCLR = 0;
359         break;
360         //if event is over I_D0 area then activate button (state=0)
361        case I_D0:
362         p_BT1 = 0;
363         break;
364         //if event is over I_D1 area then toggle switch state
365        case I_D1:
366         p_BT2 ^= 1;
367         break;
368        }
369      }
370    }
371
372  }
373
374  //Event on the board
375
376  void
377  cboard_x::EvMouseButtonRelease(uint button, uint x, uint y, uint state)
378  {
379   int i;
380
```

```
381   //search for the input area which owner the event
382   for (i = 0; i < inputc; i++)
383    {
384     if (((input[i].x1 <= x)&&(input[i].x2 >= x))&&((input[i].y1 <= y)&&
385                                                    (input[i].y2 >= y)))
386      {
387       switch (input[i].id)
388        {
389         //if event is over I_RST area then turn on
390        case I_RST:
391         if (Window1.Get_mcurst ())//if powered
392          {
393           Window1.Set_mcupwr (1);
394           Window1.Set_mcurst (0);
395
396           if (pic_reset (-1))
397            {
398             Reset ();
399            }
400          }
401         p_MCLR = 1;
402         break;
403         //if event is over I_D0 area then deactivate button (state=1)
404        case I_D0:
405         p_BT1 = 1;
406         break;
407        }
408      }
409    }
410
411  }
412
413
414  //Called ever 100ms to draw board
415  //This is the critical code for simulator running speed
416
417  void
418  cboard_x::Draw(CDraw *draw, double scale)
419  {
420   int i;
421
422   draw->Canvas.Init (scale, scale); //initialize draw context
423
424   //board_x draw
425   for (i = 0; i < outputc; i++) //run over all outputs
426    {
427     if (!output[i].r)//if output shape is a rectangle
428      {
```

```
429       if (output[i].id == O_SD1)//if output is switch
430        {
431         //draw a background white rectangle
432         draw->Canvas.SetBgColor (255, 255, 255);
433         draw->Canvas.Rectangle (1, output[i].x1, output[i].y1,
434                                 output[i].x2 - output[i].x1, output[i].y2 - output[i].y1);
435
436         if (!p_BT2) //draw switch off
437          {
438           //draw a grey rectangle
439           draw->Canvas.SetBgColor (70, 70, 70);
440           draw->Canvas.Rectangle (1, output[i].x1, output[i].y1 +
441                                   ((int) ((output[i].y2 - output[i].y1)*0.35)), output[i].x2 - output
442                                   (int) ((output[i].y2 - output[i].y1)*0.65));
443          }
444         else //draw switch on
445          {
446           //draw a grey rectangle
447           draw->Canvas.SetBgColor (70, 70, 70);
448           draw->Canvas.Rectangle (1, output[i].x1,
449                                   output[i].y1, output[i].x2 - output[i].x1,
450                                   (int) ((output[i].y2 - output[i].y1)*0.65));
451          }
452        }
453      }
454     else //if output shape is a circle
455      {
456
457       draw->Canvas.SetFgColor (0, 0, 0); //black
458
459       switch (output[i].id)//search for color of output
460        {
461        case O_LD0: //White using pin 19 mean value (RD0)
462         draw->Canvas.SetColor (pic.pins[18].oavalue, pic.pins[18].oavalue, pic.pins[18].oavalue);
463         break;
464        case O_LD1: //Yelllow using pin 20 mean value (RD1)
465         draw->Canvas.SetColor (pic.pins[19].oavalue, pic.pins[19].oavalue, 0);
466         break;
467        case O_LPWR: //Blue using mcupwr value
468         draw->Canvas.SetColor (0, 0, 225 * Window1.Get_mcupwr () + 30);
469         break;
470        case O_RB0: //Green using pin 33 mean value (RB0)
471         draw->Canvas.SetColor (0, pic.pins[32].oavalue, 0);
472         break;
473        case O_RB1: //Red using pin 34 mean value (RB1)
474         draw->Canvas.SetColor (pic.pins[33].oavalue, 0, 0);
475         break;
476        }
```

```
477
478       //draw a circle
479       draw->Canvas.Circle (1, output[i].x1, output[i].y1, output[i].r);
480      }
481
482    }
483
484   //end draw
485   draw->Canvas.End ();
486   draw->Update ();
487
488   //RB0 mean value to gauge1
489   gauge1->SetValue (0.4444 * (pic.pins[33].oavalue - 30));
490   //RB1 mean value to gauge2
491   gauge2->SetValue (0.44444 * (pic.pins[32].oavalue - 30));
492
493  }
494
495  void
496  cboard_x::Run_CPU(void)
497  {
498   int i;
499   int j;
500   unsigned char pi;
501   const picpin * pins;
502   unsigned int alm[40];
503
504   int JUMPSTEPS = Window1.GetJUMPSTEPS (); //number of steps skipped
505   long int NSTEPJ = Window1.GetNSTEPJ (); //number of steps in 100ms
506
507
508   //reset pins mean value
509   memset (alm, 0, 40 * sizeof (unsigned int));
510
511   //read pic.pins to a local variable to speed up
512   pins = pic.pins;
513
514   //Spare parts window pre process
515   if (use_spare)Window5.PreProcess ();
516
517   j = JUMPSTEPS; //step counter
518   if (Window1.Get_mcupwr ()) //if powered
519    for (i = 0; i < Window1.GetNSTEP (); i++) //repeat for number of steps in 100ms
520     {
521
522      if (j >= JUMPSTEPS)//if number of step is bigger than steps to skip
523       {
524        pic_set_pin (pic.mclr, p_MCLR);
```

```
525         pic_set_pin (19, p_BT1); //Set pin 19 (RD0) with button state
526         pic_set_pin (20, p_BT2); //Set pin 20 (RD1) with switch state
527        }
528
529      //verify if a breakpoint is reached if not run one instruction
530      if (!mplabxd_testbp ())pic_step ();
531      //Oscilloscope window process
532      if (use_oscope)Window4.SetSample ();
533      //Spare parts window process
534      if (use_spare)Window5.Process ();
535
536      //increment mean value counter if pin is high
537      if (j < pic.PINCOUNT)
538       alm[j] += pins[j].value;
539
540      if (j >= JUMPSTEPS)//if number of step is bigger than steps to skip
541       {
542
543        //set analog pin 2 (AN0) with value from scroll
544        pic_set_apin (2, ((5.0 * (scroll1->GetPosition ())) /
545                          (scroll1->GetRange () - 1)));
546
547        j = -1; //reset counter
548       }
549      j++; //counter increment
550     }
551
552   //calculate mean value
553   for (pi = 0; pi < pic.PINCOUNT; pi++)
554    {
555     pic.pins[pi].oavalue = (int) (((225.0 * alm[pi]) / NSTEPJ) + 30);
556    }
557
558   //Spare parts window pre post process
559   if (use_spare)Window5.PostProcess ();
560
561  }
562
563  //Register the board in PICSimLab
564  board_init("X", cboard_x);
```

2.5 Integration with PICsimLab

To integration of the new board in PICSimLab, are necessary edit one file.

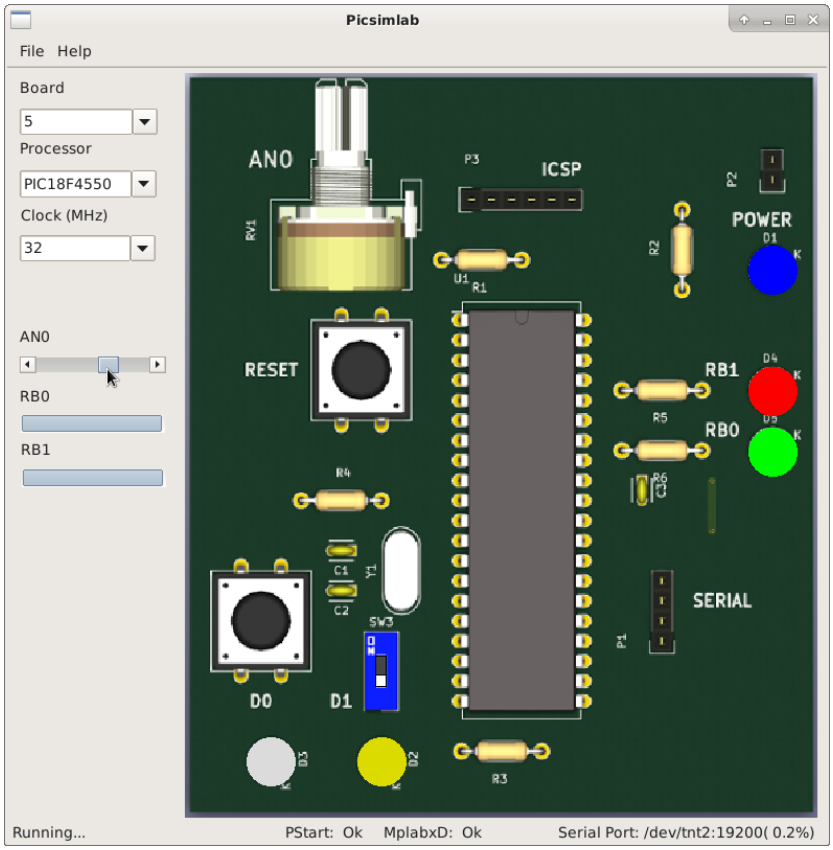
The file is Makefile.Common. The only change to be made is include object **boards/board_board_x.o** in objects list. They can be added in variable OBJS or

OBJS_EXP (used only in experimental version).

After change the Makefile.Common and include the five files created for new board, the PICSimLab can be recompiled, as described in first chapter.

2.6 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:

```
1  #include <xc.h>;
2
3  #include "config_4550.h"
4  #include "adc.h"
5  #include "serial.h"
6  #include "itoa.h"
7
```

```
8  void main()
9  {
10   unsigned int  val;
11   char buffer[10];
12
13   ADCON1=0x02;
14   TRISA=0xFF;
15   TRISB=0xFC;
16   TRISC=0xBF;
17   TRISD=0xFF;
18   TRISE=0x0F;
19
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  }
```

Chapter 3

License