INTRODUCTION TO THE LABORATORY. DEVELOPMENT OF APPLICATIONS WITH A WWW INTERFACE

1. Goals of the laboratory session

The basic goal of this laboratory session will consist in the development of an application able to be managed through a WWW interface. The application will be constructed as a program written in the C programming language. This application will be later compiled for the SC12 device.

The first step before the development of the application will consist in getting used to the SC12 device, as well as to its operating system. This will be accomplished with an application already developed from which it will be possible to understand the way dynamic WWW interfaces can be generated using C programs.

In the next section the most salient features of the SC12 device and its operating environment will be introduced. Then the procedure to be followed in order to compile an application developed in the C language and that permits to generate a WWW interface will be reviewed. After indicating the steps to be followed in order to transfer the resulting code to the WWW server the specifications corresponding to the application to be developed will be explained.

The tools that will be used to develop the application are the following:

- AnyEdit: Language sensitive editor. This tool will be used to edit C programs and to compile them so as to obtain executable code for the SC12 device.
- WinFTP: FTP client that will be used to transfer files to the SC12 device.
- WWW browser (Internet Explorer, Mozilla Firefox, …): This tool will be used to evaluate the results produced by the developed application.

2. The SC12 device

The SC12 device belongs to the IPC@CHIP microcontroller family from Beck, a “System on Chip” microprocessor family based on the x86 technology. This family represents an efficient solution for the design of electronic systems requiring Ethernet TPC/IP communications.

The SC12 device contains an internal microprocessor core from AMD compatible with the 80186 microprocessor, and also 512 KB of SRAM and 512 KB of Flash memory. The internal clock (no external oscillator is required) of the system has a frequency of 20 MHz.

The main hardware peripherals included in the chip are:

- 16-bit 80186 CPU with 20 MHz clock frequency.
- 512 KB of SRAM, 512 KB of Flash memory.
- 14 programmable input/output pins.

- 2 full-duplex serial asynchronous ports.
- Ethernet IEEE 802.3 controller, 10Base-T.
- Integrated 10Base-T transceiver.
- Watchdog timer.
- PWM options.
- 2 independent DMA channels.
- Interrupt controller with 6 external interrupt sources.
- 3 16-bit programmable timers.
- Chip select logic.

Figure 1 represents the block diagram of the SC12 device.

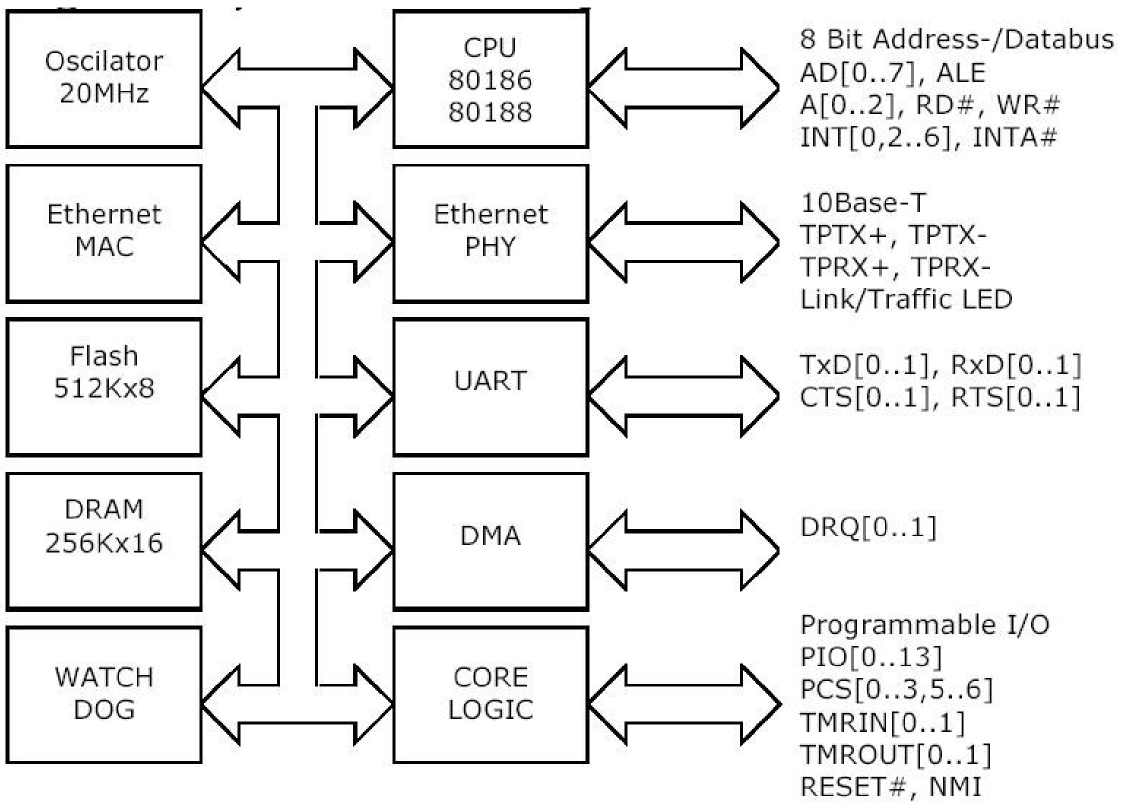

Figure 1. Block diagram of the SC12 device.

Figure 2 represents the detailed configuration of the inputs and outputs of the device. As it can be seen in this figure, the device has, among others, a data and address bus (AD bus, pin 8 to 15), typical in the 80186, that permits to access external devices through read and write processes. The waveforms corresponding to the read and write processes on this bus are presented in figure 3.

In figure 3 the ALE (Address Latch Enable) signal is used to register externally the address corresponding to the read or write process. The RD# signal (active low) indicates that the access corresponds to a read process, while the WR# signal (active low) indicates that the access corresponds to a write access. The PCSx# signals (where x may have a value from 0 to 6 in the case of the SC12 device) defines that the read or write address corresponds to a given group of devices.

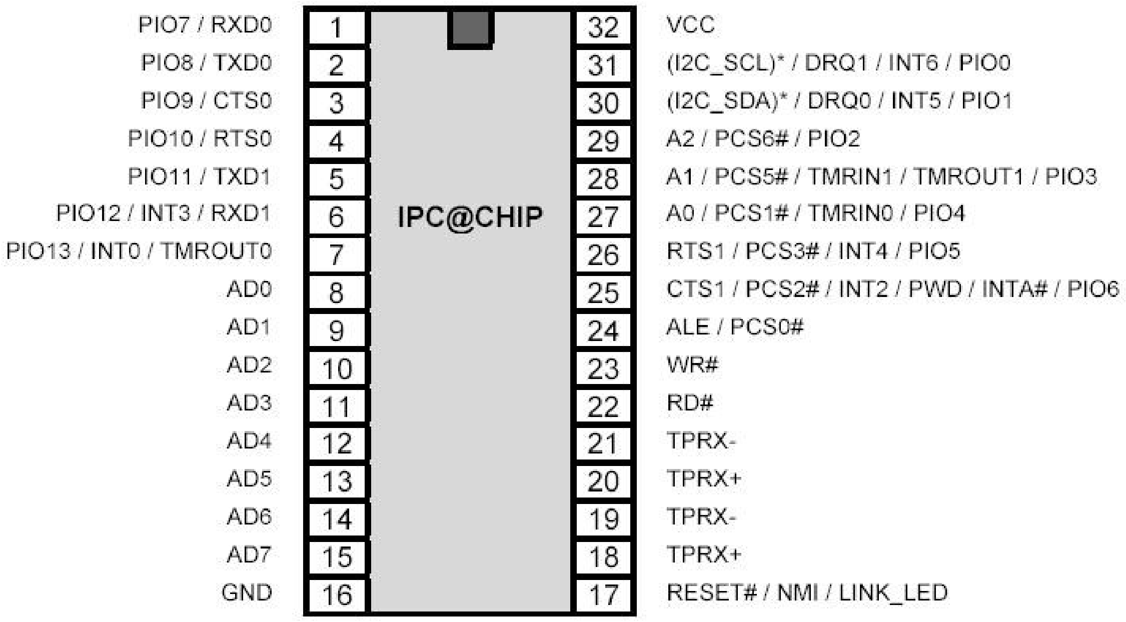

Figure 3. Pin configuration of the SC12 device.

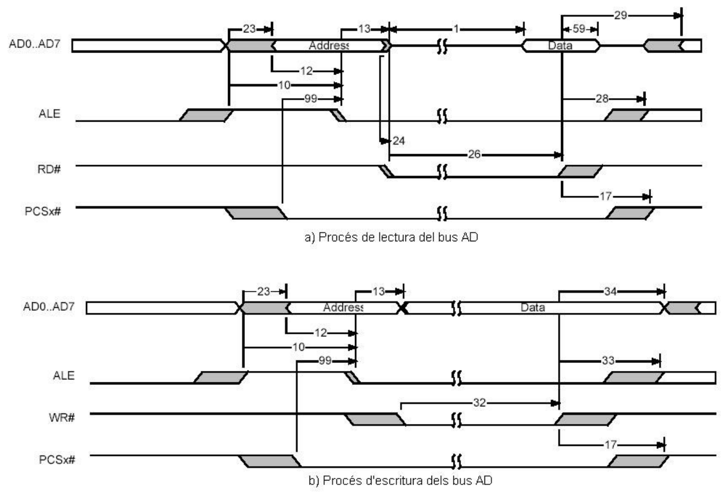

Figure 3. Read and write processes on the AD bus of the SC12 device.

In order to perform read or write operations on the AD bus it is possible to use the ‘inportb’ or ‘outportb’, respectively, from the C language. These instructions include the specification of the address relative to PCSx in order to indicate to the external device that a communication through the AD bus will be established.

For instance, let’s consider that a given device is connected to the SC12 device using the transceiver 74HC245, as depicted in figure 4.

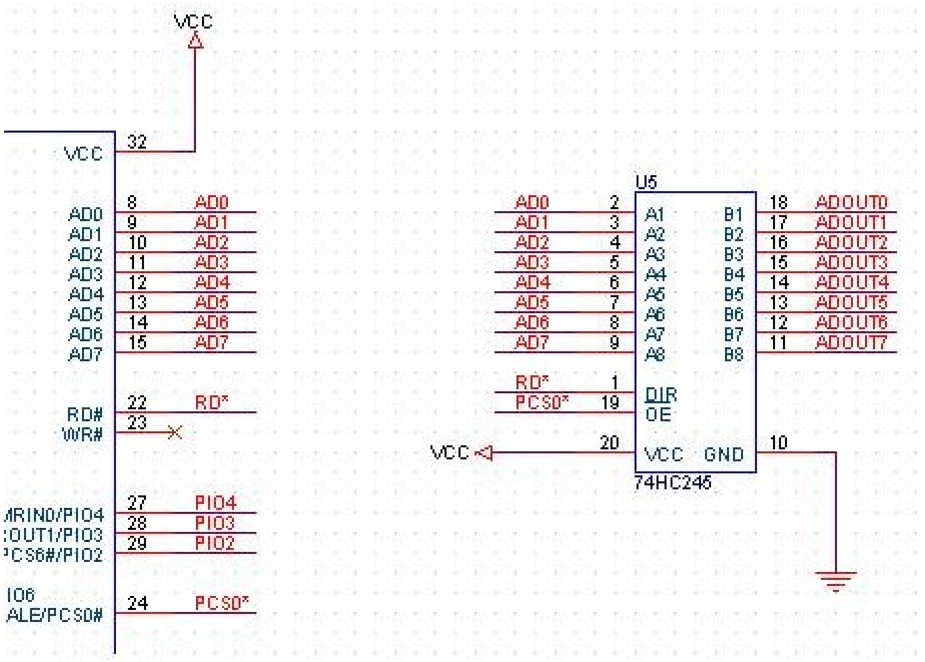

Figure 4. Use of the AD bus in the SC12 device.

To carry out the read and write processes on the external device using the transceiver 74HC245 the following C code can be used:

```
#include <dos.h>

void main( void )
{
      unsigned char read;
      unsigned char write= 0xFF;

      ActivatePCS0();

      read = inportb( 0x0000 );           // read activating PCS0*
      outportb( 0x0000, write );          // write activating PCS0*
}
```

As it has been explained previously, another interesting feature of the device is the availability of 14 programmable input/output lines. These lines, labelled as PIO0 to PIO13 in figure 2, will be used to communicate the microcontroller with external devices. As it will be explained later, the lines PIO0 and PIO1 will be used for the application proposed in this practical session.

3. Introduction to the @Chip operating system

The SC12 device has a real-time operating system (RTOS) able to manage up to 35 tasks, 15 timers, 60 semaphores, etc. This operating system establishes the communication between the hardware system and the software applications, and it contains all the modules needed to handle the different functions available in the chip. It is also possible to load or execute instructions or applications during the system initialisation by properly configuring the 'autoexec.bat' file. The RTOS is placed in the Flash memory of the device. Its architecture is depicted in figure 5.

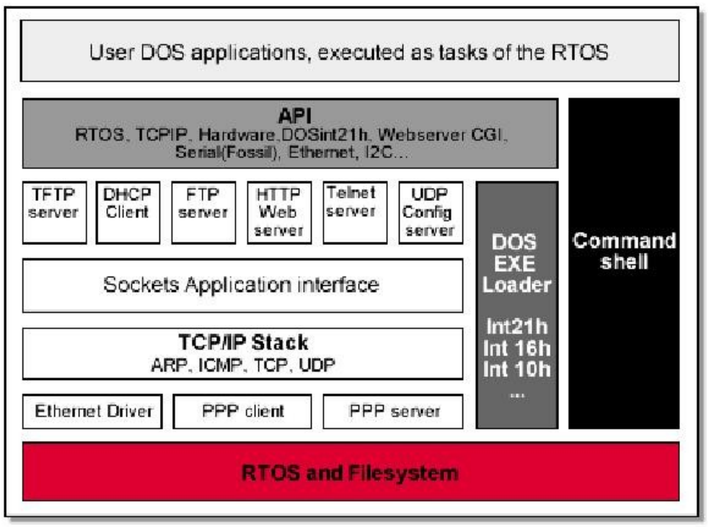

Figure 5. Architecture of the operating system of the SC12 device.

As it can be observed in this figure, the operating system permits to access, configure and use all the applications integrated in the device: FTP server, WEB server, telnet server, command interpreter, file and input/output signal access, etc.

One of the most important parts in the operating system is the API (Application Programmers Interface). As its name indicates, it constitutes the interface between the operating system and the applications created by the user of the SC12 device. This interface contains programs that can be used to configure the device applications without a deep knowledge of the hardware of the system.

The operating system uses software interrupts to call the API's procedures. Two values are required to call an interrupt: the interrupt number and the function number. The interrupt number specifies the API and the function number indicates the procedure of the API.

Some examples of interrupts and system functions are:

Interrupt	Function	Description
0xAB	0x01	Installs a function in the WEB server
0xA1	0x8D	Specifies the logic levels for an I/O pin
0xA2	0x80	Enables the AD bus of the device
0xAC	0x09	Makes the system to enter in sleep mode for a given amount of time

If the SC12 device is programmed using the C language the following structures are used in order to call the system software interrupts:

```
#include <dos.h>

int int86( int int_num, union REGS *in_regs, union REGS *out_regs );
int int86x( int int_num, union REGS *in_regs, union REGS *out_regs,
            struct SREGS *sregs );
```

The function int86( ) executes software interrupts specified by int_num. The contents of the union in_regs is first copied into the processor register and then the interrupt is executed. The union out_regs contains the values of the CPU registers returned by the interrupt. This permits to verify the correct execution of the function.

The unions REGS and struct SREGS are defined in the header file DOS.H

The function intx86( ) is equivalent to the previous one, but in this case it is possible to specify values for the segment registers ES and DS of the processor using the values contained in sregs.

An example of what has been explained previously will be explained thereafter. As it was introduced previously the function 0x09 of the interrupt 0xAC makes the SC12 to enter in sleep mode during an amount of time specified by the programmer. The implementation of this function would be therefore as follows:

```
#include <dos.h>
union REGS inregs, outregs;

void main( void )
{
      inregs.h.ah = 0x09;      // reg. AH contains the function to execute
      inregs.x.bx = 1000;      // reg. BX contains the time in ms

      int86( 0xAC, &inregs, &outregs );   // interrupt is executed
}
```

In this way it is quite easy and intuitive to use and configure the devices integrated into the SC12.

As it was explained previously the SC12 RTOS also includes the functionality required to implement a web server. This means that HTML pages can be registered in the device and they can be accessed using a web browser. The web server is started as soon as the SC12 is connected to the network. Therefore, the user has just to specify the IP address of the SC12 device in the browser in order to get connected to it.

Furthermore the web server includes a CGI (Common Gateway Interface) interface. By using this interface it is possible to construct dynamic web pages.

If a web page is static this means that its content is predefined in an HTML file located in a server. Using a CGI interface it is possible to modify the content of a page as required by the user interaction. Additionally, the CGI interface is also able to work in reverse mode. This means that a given web page may send information to the server that can be processed by an application. An example of this communication between a client and a server is explained thereafter.

It is assumed that the IP address of the web server is 147.83.49.24. Furthermore, let's assume that an application is executed that loads a web page when the following address is entered in the browser: http://147.83.49.24/test. The method for the communication is based on using in the web page a link as: http://147.83.49.24/test?value1. This method is called 'GET' and a server with a CGI interface is able to read 'value1' and return a new web page whose content depends on the value send after the character '?'.

Additionally, if this web page is used in applications programmed in C, the application may evaluate the parameter and use its value to carry out a given function within the SC12 device, like setting a given I/O pin. Therefore, the use of the CGI interface permits the SC12 device to establish an actual connection between the physical world and internet.

4. Example of dynamic HTML programming

In the folder that will be indicated in the laboratory session the following files may be found:

- cgi.h: Header file with the definitions required to manage the CGI interface of the web server included in the operating system.
- hard.c: This file contains a set of basic functions that permit to manage several features of the SC12 device (like the AD bus or the input/output lines).
- web.c: This file will contain the code required to generate dynamic HTML pages.
- proc.c: This file will contain the program code that will be executed by the SC12 device.

These files have to be copied into an appropriate folder within the user area that will be used to perform the laboratory session (Note: Due to limitations of the C compiler that will be used it is important that the name of this folder is not longer than 8 characters, and it should not contain the space character neither).

The first step to verify the correct functionality of the application consists in opening the AnyEdit application (present in the user Windows desktop). Once the application is opened the first thing to do is to create a new project. To do it first select the option ***File -> New …*** in the main menu. A dialog window with several options will appear. In this window the ***New Project*** option has to be chosen, and within it the ***Blank Project*** option has to be selected. After that it will be necessary to specify the name and the location for the project (options ***Project Name*** and ***Project Location*** in this dialog window). Once these options are set it can be seen that on the left section of the user interface of the tool, named ***Workspace***, a new group with the name assigned to the project has been created.

The next step will consist in adding to the project the files required to carry out the application. Therefore, by clicking with the right mouse button on the last element in the list (the element that has just been created) the option ***Add Files*** should be chosen. In the next dialog window it is necessary to select all the files that were indicated previously.

After opening these files it is possible to observe that the ***Workspace*** section now contains the files containing the application code. By clicking twice on a give file it is possible to edit its contents in the right section of the user interface.

As it was previously explained the file ***cgi.h*** contains definitions required for managing the CGI interface of the web server. This definition file is referenced in the file ***web.c*** using the directive ***#include "cgi.h"***.

The file ***hard.c*** contains the definition of the functions that permit to handle specific hardware resources (like the AD bus or the programmable input/output lines) of the SC12 device. This file is referenced, using the directive ***#include "hard.c"***, in the file ***proc.c***. The functions defined in this file will be later used to carry out the application proposed in section 5.

In order to understand the structure included in the file ***web.c*** it is necessary to know the structure of the web pages to be developed in the device. The entrance page to the device is called ***Index***, so that in order to access this page it is necessary to enter in the browser http://sdii0.upc.es/Index. This page just shows some basic information about the laboratory and contains a link to the control page that is called ***Main***. This page is structured around two frames. The upper frame, called ***Banner***, shows a simple figure, while the bottom frame, called ***COM***, will show the command that can be executed from the application, as well as their results. The parameter that contains the value triggering the type of action to be executed is called ***parameter_value***, and is defined in the initial section of the code.

These definitions can be seen in the first part of the code included in the file ***web.c***:

```
#include <DOS.H>
#include <STDIO.H>
#include <STRING.H>
#include <STDLIB.H>
#include "cgi.h"

#define TCPIPVECT       0xAC
#define CGIVECT         0xAB
#define PFE_INT         0xA2
#define HAL_INT         0xA1
#define BIOS_INT        0xA0
#define I2C_INT         0xAA

int parameter_value = 0;

void ConfigWebServer( void );

/*************************************************************/
/************** FEATURES OF THE WEB PAGE ************************/
/*************************************************************/

char *NameBanner          = "Banner";
char *NameFrame           = "Main";
char *NameInitpage        = "Index";
char *NamePage           = "COM";
```

Now it can be seen that the code corresponding to the different pages that have to be generated have been included in the file web.c using several strings:

```
char *Frame =  /* Main web page that contains the remaining ones */
"<HTML>"
    "<HEAD>"
        "<TITLE> DS. Laboratory </TITLE>"
    "</HEAD>"

    "<FRAMESET NORESIZE FRAMEBORDER=NO ROWS=24%,*>"

        "<FRAME SCROLLING=NO SRC=\"Banner\">"
        "<FRAME SCROLLING=NO SRC=\"COM\">"
        "</FRAMESET>"
    "</FRAMESET>";

/*******************************************************************/

char *Logo =      /* Upper page. Contains just the logo */
"<HTML>"
    "<BODY bgcolor=\"#A0A0A0\">"
        "<TABLE border = 0 width = 100% >"
        "<TH align = center><IMG src=\"imatges/logo.jpg\">"
        "</BODY>"
"</HTML>";

/*******************************************************************/

char *InitPage = /* Presentation page. Contains the link to the
                  main page */
"<HTML>"
    "<HEAD>"
        "<TITLE> DS. Laboratory </TITLE>"
    "</HEAD>"
        "<BODY bgcolor=\"#A0A0A0\">"
        "<BR><BR><BR>"
        "<H2><P Align = center >Acquisition and control system via web
for a digital camera </P></H2>"
        "<BR><BR>"
        "<H1><P Align = center ><A href = \http://sdii0.upc.es/Main\ >
         ENTRAR </A></P></H1>"
        "<BR>"
        "<TABLE>"
        "<TD>"
        "<I><H2>"
        "<P Align = left >Digital Systems. Laboratory</P>"
        "<H3>"
        "<PRE>"
        "<P Align = left   >ETSETB</P>"
        "<TD width = 70% align = CENTER>"
        "<IMG src = \"imatges/sc12.jpg\">"
        "</BODY>"
"</HTML>";

/***** Bottom left page. Contains the control menu *******/

char *MainPage =             /* Initial menu */

    "<HTML>"
        "<BODY bgcolor=\"#A0A0A0\">"
        "<TABLE border = 1 cellspacing = 1 cellpadding = 1 width
=100%>"
```

```
        "<TH align = center width = 20%>MENU"
        "<TH align = center width = 20%>State"
        "<TR>"
        "<TD align = center>"
        "<BR>"
        "<H5><A href=\"COM?0\>Activate LED<P></A></H5>"
        "<H5><A href=\"COM?1>Deactivate LED<P></A></H5>"
        "<H5><Ahref=\"COM?2>Periodic activation/deactivation
        <P></A></H5>"
        "<TD align = center>";

char *Ledon =
    "<IMG SRC=\"imatges/ledon.jpg\"></P></H2>"
    "</TABLE>"
    "</BODY>"
    "</HTML>";

char *Ledoff =
    "<IMG SRC=\"imatges/ledoff.jpg\"></P></H2>"
    "</TABLE>"
    "</BODY>"
    "</HTML>";

char *Periodic =
    "<P><B>The LED is being activated/deactivated periodically
    </B></P>"
    "</TABLE>"
    "</BODY>"
    "</HTML>";
```

The control page is contained in the strings called “MainPage”, “Ledon”, “Ledoff” and “Periodic”. This separation is due to the fact that, depending on the action to be taken the state of the system should be modified, and so the visualisation of the state should also be changed. In this way, as it will be explained later, the string corresponding to the control page is generated by concatenating the string “MainPage” and one of the strings “Ledon”, “Ledoff” or “Periodic”. This concatenation will be done depending on the parameter returned by the links present in the page. As it can be noted in the previous code, the value of the parameter returned by a given link is specified as `<A href=\"COM?1\"...` Within this reference, “COM” is the name of the page, as it was explained previously, while “?1” indicates that the value of the parameter that is returned when the link is selected will be 1.

Once the code corresponding to the different pages that constitute the application has been defined, three functions are defined in the file ***web.c***: “FuncioInici”, “FuncioFrame” and “FuncioLogo”. These functions install these pages inside the system. As it can be noticed in the code, these functions just copy (strcpy() function) a given string over the global string called “PaginaHTML”, and then they call the functions required to initialise the pages.

The next function in the file ***web.c*** is called “FuncioCGI”. This function evaluates the parameter that has been returned when a given link has been selected. As it can be seen in the code, initially (i.e., before selecting any link) the page is generated by concatenating the strings “MainPage” and “Ledoff”. When the user selects a specific link the corresponding parameter value is determined and then the page to be shown will be generated by concatenating the corresponding strings.

The next function, called "Process_Func", is in charge of determining if the pages have been correctly installed in the server. This function must be always present in the code to be developed.

Finally the function "ConfigWebServer" is in charge of configuring the web server present in the device from the contents of the file ***web.c***. This function must also be always present in the code to be developed.

The file ***proc.c*** will contain for the moment just a call to the function "ConfigWebServer", so that the SC12 device will not make any action regardless of the options selected by the user in the web pages.

The sequence of steps to be followed in order to compile the code and transfer it to the SC12 device is the following:

1. In the editing area of the AnyEdit tool the file proc.c has to be selected.
2. Choose the option ***Tools -> Compilador C*** from the main menu (the same effect can be obtained by clicking on the button labelled as in the action buttons section of the tool). As it can be observed in the window called ***Output*** in the bottom of the tool, this action calls the compiler that will translate the C code into executable code for the SC12 device. If the compilation has completed without errors a new file called proc.exe can be found in the folder were the code files were placed. This file contains the binary code to be executed by the SC12 device.
3. Once the executable file is generated it has to be transferred to the device using the FTP protocol. In the laboratory it will be explained how this transfer can be performed.

5. Development of a hardware control application using a web interface

The goal of this laboratory session is to modify the file ***proc.c*** in order to permit a physical realisation or the actions explained in the previous section. For this purpose the board that contains the SC12 device will be connected to a DIO4 board. This board contains, among other elements, 8 LEDs that can be managed independently. The voltage applied to the anode of each LED comes from the output of a latch whose enable signal is called LEDG (active high). This enable signal is the same for all the latches.

The file ***proc.c*** has to be modified using the functions defined in the file ***hard.c***, so that the first LED in the DIO4 board can be:

- Activated
- Deactivated
- Activated/deactivated periodically. The activation/deactivation interval will be 1 second.

To complete the development the following considerations have to be taken into account:

- The signal LEDG is connected to the pin PIO0 of the SC12 device.

- The input data of the latch actuating on the first LED of the DIO4 board is connected to the pin PIO1 of the SC12 device.
- The functions to be used from those defined in the file ***hard.c*** are:
    - void EnablePIO(unsigned char PIN, unsigned char mode): This function initialises a given input/output pin of the SC12 device. The parameter PIN indicates the number of the pin to be initialised (a value from 0 to 13), while the parameter mode determines how the pin will be used (a value 4 corresponds to an input pin, while a value 5 corresponds to an output pin with an initial value of 0).
    - void WritePIO (unsigned char PIN, unsigned char value): This function permits to write a value on a given input/output pin of the SC12 device. The parameter PIN indicates the number of the pin to be written (a value from 0 to 13), while the parameter value has to specify the value to be written.
    - void api_sleep (int ms): This function hold the system in sleep mode for a number of milliseconds equal to the value specified for the parameter ms.
- The file ***web.c*** contains a global variable called parameter_value. This variable indicates the action to be takes. When its value is 0 the LED has to be deactivated, when its value is 1 the LED has to be activated, and when its value is 2 the LED has to be activated and deactivated periodically.
- If the program is structured as an infinite loop, it will be necessary to call within this loop the function api_sleep at least once (for instance, with a value for the parameter ms equal to 10), so as to avoid the system to become blocked by the program execution.