Xavier Rouby

Student in physics
UCL
Belgium

CERN Summer Student

2001

Project Report

Introduction

This is my work report, as a Summer Student. As a student in physics, at the *Universite Catholique de Louvain* in Belgium, I was looking forward from being here during the summer. I worked here at CERN in the EP-CMT division with Mr. Piero Giorgio Verdini as my supervisor. My stay here was as wonderful as I could expect. Lots of visits, lots of things to learn, lots of friends; nice surroundings, many things to do. CERN looks sometimes like a magic place… but not for the architecture.

I really enjoyed my stay as a 2001 Summer Student.

Acknowledgements

There are many people who helped me before and during my stay at CERN. First of all, I would like to thanks Piero Giorgio Verdini, Patrice Siegrist and Alessandro Marchioro, who helped me in my work. Moreover, the meetings with others summer students gave help and interest to our work. In particular, I would like to thanks them for letting me go to take part to the ESA parabolic flight campaign for four days; I was really happy to be able to do this, even if I had to work later and on rainy Sunday afternoons…

I want also to thanks the Summer Student who were with me, Thomas, Rene and Katharina. I had a lot of fun with them, in particular with Rene during these extreme sport adventures.

Thank you also to Seamus Hegarty, Ingrid Schmid, Karine Joasset and to all the team that organize the Summer Student activities. In particular, the lectures were very interesting and also he visits and the workshop.

Eventually, I want to thanks those people who helped me to come here and to make my life here easier: Mr. Gregoire, Govaerts, Lemaitre, Van der Aa, Delaere and Graulich

My stay at CERN

As a summer student, I followed almost each lecture. I also took part to two workshops (*JAS and WIRED tutorials* and *Mirror and Light*). This learning program was big but very interesting.

I also made a lot of visits: Microcosm, CMS, SPS Control Center, Computer Center, ISOLDE and HARP. I am sure not to have seen everything here, but these ones were particularly interesting. Both visits and lectures were the learning part of my stay.

My work consisted in writing code to make life easy when using the EZ-USB AN2131QC chip. For this, I received an EZ-USB Xcelerator Development Kit, consisting in a Development Board (with the chip, LED's and buttons on it), which is only a demonstration board. After having read a lot on USB bus and having refreshed my C programming, I started using the provided EZ-USB specific software and understanding main examples. So, I was able to do basic operations with the chip on the I^2C bus where the LED's and buttons are plugged on. After such activities, my new aim was to build an interface, using *Visual Basic*, to allow remote control on the board (and in particular on the I^2C bus), directly with the computer. So, I had to learn to program with *Visual Basic*. Having some issues, I could not finish my work. The visual part of the interface was build, but no events have been written.

Job Completed

Books

During the first three weeks at CERN, I read the main chapters of books [b1], [b2] and [b3].

First of all, the *Technical Reference manual* [b3] gives an introduction to EZ-USB and also some technical information (CPU, Memory, I/O, Enumeration and ReNumeration, Transfers, Endpoints,...). The book also gives block diagrams (such as the blue one on the printed version of [dk1]) that explain the chip architecture, pin description, memory map and so on.

USB Design by Example [b2] explains the way of working of USB connections (hub, devices), the enumeration process. It gives some examples (also on the CDROM; not always working) but not directly linked with the Development Board.

USB Complete [b1] gives many explanations with many details. It really helps to understand what happens, for instance, during the Enumeration process.

Software

After reading these books and some online documentation, I learned how to use the tools included within the package. Main software are the **EzMr USB Control Panel** and **Keil Micro Vision 2** (µVision).

The latter is mainly a user-friendly compiler that allows debugging written programs. It is a *C* compiler, specific to the enhanced EZ-USB *8051* processor. Among other files, it creates executable *.hex* files that can be loaded on the chip via the *Control Panel.* Sometimes, unexpected errors prevent me from compiling anymore source code. To save time for any user who will uses this later, I collected the compilation options needed to compile correctly source code. See below the section called "*Compilation options in µVision 2*".

The *Control Panel* allows fast downloading of firmwares into the chip, and fast analysis via the set of basic instructions and requests the chip has to respond to. Moreover, this software is written in *C++* and its source code is available.

Examples and source codes

Several examples are available, in terms of *C* files and *Keil µVision* projects. The most interesting one, in my job, is the example called **dev_io**. It lies inside the folder *C:\Cypress\USB\Examples\EzUsb\dev_io* (where the installation folder was supposed to be *C:\Cypress)* This example shows how to control the I^2C bus via the Ez-USB chip. Indeed, the four buttons on the right side of the Development Board and also the LED's just above (see the picture of the board) are connected to the I^2C bus. When the program is running (for instance, by downloading the file *dev_io.hex* into the chip via the *Control Panel*), the first button set the LED screen to "0", the second button do the "minus one" operation, the third button "plus one" and the fourth set the counter to its maximum value which is "F". In fact, the chip is continuously reading button status on the I^2C bus. When a button is pressed, the index of a table of hexadecimal numbers changes. These hexadecimal numbers correspond to the code that will be sent to the LED's, to show the desired character (The solution to the code is given below). So, as the index has

changed, the new hexadecimal number, corresponding to the new index, is sent to the LED's via a "writing" onto the I^2C bus and the final display changes. The C source code of the *dev_io* example is given in appendix.

To be sure to have understood everything in this important example, I wrote another program (still in *C*), called **xav_hello**. The previous program inspirited the code of *xav_hello*. When running, this program also controls buttons and LED's via the I^2C bus, with the Ez-USB chip. The LED's constantly display the letters of the word "Hello." The buttons controls the scrolling speed: the first one sets maximum speed, the last one minimum speed and the second one increases the speed while the third one decreases it. Once more, the *C* source code is given in appendix, with some self-explanatory comments.

A basic program with read and write operations on the I^2C bus has its source code in appendix.
Two tables and one variable are defined:

- Alpha contains the hexadecimal codes for every letter of the alphabet.
- Digit is the same with the numbers.
- Blank is the code corresponding to a blank LED screen.

Before the very first *read* or *write* operation, an initialization must occur. It is done by calling the EZUSB_InitI2C () function. Reading is easy (EZUSB_ReadI2C(address, length, where_to_put_it)) and so is writing (EZUSB_WriteI2C(address, length, what_to_write)). The explanation of every I^2C standard operation is available in the printed files *Anchor EZ-USB Frameworks.pdf* and *Anchor EZ-USB Firmware Library.pdf*. They are in the folder *C:\Cypress\USB\Doc\EZ-USB General*.

User interface

After having understood how to access devices connected to the I^2C bus, another part of my job was to build a user interface, in *Visual Basic* (*VB*). This interface must replace the real buttons and LED's, by displaying virtual buttons and LED's. After having transformed previous C code into Dynamic Link Library (*.DLL*), the *C* programs can be used by *VB*. The major problem is that the *Keil C* compiler, needed to create executable files for the Ez-USB chip, cannot create such standard *.DLL*. My work stops with this issue.

One way to solve this is to write a small *C* or *C++* code and to copy the part of the *EzMr USB Control Panel C++* source code, where a function deals with downloading *.hex* file into the chip. Then one would be able to download directly the desired *.hex* file into the chip, and to do standard read and write operations on the I^2c bus. If this code is compiled to a *.DLL* (this can be done via *Visual C++,* I think), then it can be inserted in the *VB* interface. Then, an event related to each "button press event" can be easily written (each time using the same EZUSB_ReadI2C and EZUSB_WriteI2C functions, differing only by the arguments) and everything can be done under remote control, only by using the *Visual Basic* interface.

However, the user interface is already done (shown below).

The rest of the work consists in providing the buttons and the LED's with events. The source code of this *VB* application lies in
C:\Program Files\Microsoft Visual Studio\VB98\EZUSB Development Board.vbp.

Technical notes: Tips and information

Cypress Semiconductor EZ- USB

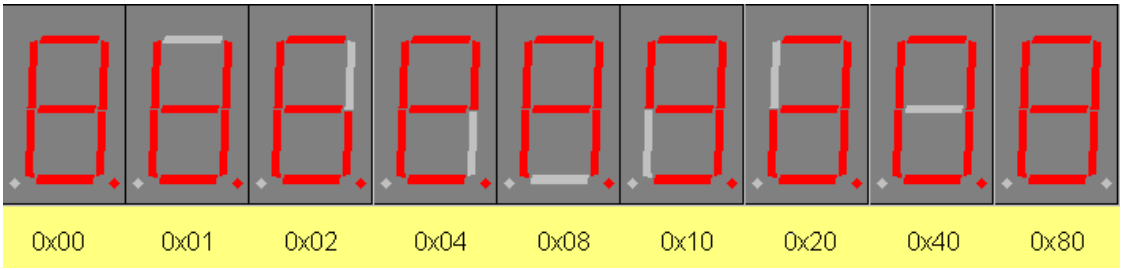
Here is a small summary about the Ez-USB chip. {pp 132-135, chap 7 (Chip choices), USB Complete [b1]}

"*Cypress' EZ-USB is a 8051-compatible family of chips with USB capability. This chip uses a different approach to storing firmware. Rather than storing the firmware on-chip, an EZ-USB can store its firmware on the host, which loads it into the chip on each power-up or attachment. The obvious advantage is easy updates for firmware. To update the firmware, just store it on the host and the driver will send the firmware to the device on the next power-up or attachment. The downsides are increased driver complexity, the need to have the firmware available on the host, and longer enumeration time.*

The EZ-USB is an 8051 with a redesigned core. The chip uses four clock cycles per instruction cycle, compared to the 8051's twelve. The CPU is clocked at 24MHz. All of the chip's 8-kilobytes of combined code and data memory is RAM; there is no non-volatile memory. However, the chip does support non-volatile storage in its I^2C *serial interface that can read and write to serial EEPROM. There are 32 general-purpose I/O pins.*"

Hexadecimal Code for LED's

Any character may be display on the LED screen, by lighting desired LED, if the correct hexadecimal code is sent as an I^2C writing. Each segment within the LED screen can be separately lit off, and to light off many ones at once, the corresponding hexadecimal code is simply the sum of the code of each segment alone. Here is the hexadecimal number for each segment:



So, to display the character “E”, without any point on the right, we have to light off two segments and the dot. The code is then 0x02 + 0x08 + 0x80 = 0x86.

Compilation options in µVision 2:

To make things working correctly, after encountering some strange errors, I found that the following compilation options work perfectly. They can be changed via the Menu Project, choosing the command called “*Option for Target 'Target1'* “. The more frequent error, after a right compilation (no error, no warning), occurred during the automatic creation of the executable (*.hex*) file. The error message was “*Unexpected End of File Example.HEX*” (where *Example* is the project name, here). To solve such a problem, just set the main options as following:

Target

```
Xtal                        48.0 MHz
Memory Model:                     Small variables in DATA
Code Rom Size:                    Large 64K program
Operating System:           none
Off-chip
 Code memory: Eprom               Start 0x0000, Size 0x10000
 XData memory: Ram                Start 0x0000, Size 0x10000
```

Output

```
Name of Executable xav_hello (for instance) and NOT xav_hello.hex!
If you put a name with ".hex", it wouldn't work ("UNEXPECTED END OF
FILE ... .HEX")
```

L51 Locate

```
Use memory Layout from Target dialog: NO.
Code Range    --
Xdata Range   --
Code          0x4000
Xdata         0x5000
```

Debug

```
Use: Keil monitor-51 driver ==> Settings COM1, 19200 bauds + all
options selected
```

Folder architecture:

```
C:\Cypress
++ C:\Cypress\USB
++ C:\Cypress\CYDB
```

C:\Cypress\USB\Bin

This folder contains all useful executable files.
The most important is the **EzMr** file, which is the *USB control Panel*. **Bin2c**, **Hex2bix**, **hex2c** are programs used, for instance, by *Keil µVision2*. **RegClear** is a software which help to clear registers.

C:\Cypress\USB\Doc

This folder gather all documentation files, split into several subfolders.

Main files:

++ **\EZ-USB Series 2100\Series 2100 TechRefManual v1.9.pdf**
This is the *.pdf* version of the printed Technical Reference Manual, provided within the Xcelerator Development Kit. All information about technical settings is found in this file.

++ **\EZ-USB Series 2100\Series 2100 Dev Kit v1.9.pdf**
This file gives some information about the softwares included into the Kit. Also provided in a paper file inside the Kit

++ **\EzMrDevK\EzMrDevK.html**
Ez-USB Developer's Kit Content and Tutorial

++ \EzMrUser\EzMrUser.html
User guide for EZ-USB Control Panel (*EzMr.EXE*)

++ **\EZ-USB General\Anchor Library**
This file gives a description of some C functions encountered in examples.

C:\Cypress\USB\Drivers

This folder contains the drivers needed for the installation of the board.

C:\Cypress\USB\Examples\EzUsb

In this folder are collected a few examples. The one using the led and the buttons on the board is inside the subfolder **dev_io**.
The easiest way to download the example firmware into the chip is to click on the "D*ownload..*" button in the Control Panel (*EzMr*), and then to browse and select the **.hex* corresponding file inside the interesting folder. (for instance, download the **dev_io.hex** file))
The **.iic* files can be directly downloaded into the EEPROM, using the "*EEPROM..*" button.

In most of these example subfolders, on will find

++ **.c**, **.h** files
C source codes
++ **.hex**, **.bix** or **.iic** files
To be downloaded into the chip or the EEPROM
++ **.uv2** file
Keil Uvision2 project
++ **readme.txt** file
Example aims and explanation

C:\Cypress\USB\Hardware\EZ-USB Series 2100

Contains some information about the chip itself.

C:\Cypress\USB\Target

++ **\Inc**
This directory contains the include files used by the frameworks and example files for the Anchor Chips EZ-USB chip.

++ **\Lib\Ezusb**
This directory contains the source code for the 8051 library provided with the Anchor Chips EZ-USB chip.

C:\Cypress\USB\Util

++ **\EzMr**
This directory contains the C++ Source Code for the Control Panel Application.

Printed documents

The following documents provide fast and easy documentation:

- Ez-USB Developer's Kit Content and Tutorials (29 pages)
- ezusb.h (4 pages)+ EZRegs.h (6 pages) source codes
- fw.c (4 pages) + EP_PAIR.c (3 pages) + PERIPH.C (6 pages) + MEMTEST.C (2 pages) source codes
- Anchor EZ-USB Frameworks (11 pages) + Anchor EZ-USB Firmware Library (3 pages)

How to start quickly with the Development Board

To be read first:

Before doing anything with the Development Board, the best thing is to read the following file: *EZ-USB Developer's Kit Content and Tutorials* (c:\Cypress\USB\Doc\EzMrDevK\EzMrDevK.html).
This file contains all the information to start with the Kit. It was printed. The mainrt of it is a tutorial. This explains how to use simultaneously the *Keil* compiler and debugger and the *Control Panel*, by trying some of the given examples.

The Cypress web page about AN2131-DK001 Development Kit (http://www.cypress.com/design/progprods/usb/devtools/an2131-dk001.html) was also printed. However there is no important news there, but the possibility to have inline Technical Reference Manual.

Important software to be installed...

… to test the EZ-USB Developer's Kit:

- EzMr USB Control Panel (*c:\cypress\usb\bin\EzMr.exe*)
- Keil µVision 2 (*c:\Keil\uv2\uv2.exe*)

The Technical Reference Manual is the primary hardware reference manual for the EZ USB chip. It contains pinouts, register definitions, a description of system architecture, and interface definitions (*c:\cypress\usb\doc\EZ-USB Series 2100\EZ-USB Series 2100 TechRefManual.pdf*)

Web Pages

About the Development Kit and the Chip

[dk1] http://www.cypress.com/design/progprods/usb/devtools/an2131-dk001.html (printed)
[dk2] http://www.cypress.com/cypress/prodgate/usb/ezusb1pg.html (printed)
[dk3] http://www.beyondlogic.org/usb/cypress.htm
[dk4] http://www.usb-by-example.com/Examples/DevicePC.pdf

Visual basic tutorial

[vb1] http://vkliew.tripod.com/vbtutor.html: A quick tutorial for very early VB programmer.
[vb2] http://cuinl.tripod.com/tutorials.htm: Basic tutorial + Active X + Resource Files
[vb3] http://www.geocities.com/SiliconValley/Bay/5707/vbasic.html: Very good tutorial with description of every standard controller
[vb4] http://www.free-ed.net/fr03/lfc/030202/120/: Very big tutorial, covering a lot of material.
[vb5] http://www.boondog.com/..%5Ctutorials%5Cdlltutor%5Cdlltutor.htm: about DLL.

Bibliography

- [b1] **USB COMPLETE**, **Everything You Need to Develop Custom USB Peripherals**, *Jan AXELSON,* Lakeview Research, 1999 (http://www.lvr.com/usb.htm/)

- [b2] **USB Design by Example** [2], **A practical Guide to Building I/O Devices**, *John Hyde*, Intel University Press, Wiley, 1999 (http://www.usb-by-example.com/)

- [b3] **The EZ-USB Integrated Circuit** [3]**, Technical Reference Manual**, CYPRESS, 2000 (http://www.cypress.com/usb/ezusb_trm1-9.pdf)

- [b4] **c:\Keil\C51\HLP\Gs51.pdf**