A System to Assist the Learning of Embedded Microprocessor Programming

Yu-fai Fung, Shuyun Ren
Department of Electrical Engineering
The Hong Kong Polytechnic University
Hong Kong SAR
eeyffung@polyu.edu.hk

M. Fikret Ercan
School of Electrical and Electronic Engineering
Singapore Polytechnic
Singapore
mfercan@sp.edu.sg

***Abstract*—When implementing final year projects, students may need to program a microprocessor in order to achieve the project objectives. Therefore, a system is designed in order to assist students who have limited background knowledge in computer engineering to learn the basics of microprocessor programming primarily using the C programming language. In addition to the software, various hardware components are provided so that students can develop simple systems so the learning outcomes can be enhanced by practice. From feedback collected, users found that they are able to learn simple C language programming for a microprocessor with the help of the system.**

***Index Terms*—embedded systems; microprocessor; C programming language; forms programming**

I. INTRODUCTION

There are two undergraduate programs which are Electrical Engineering and Transportations System Engineering offered by the Department of Electrical Engineering of The Hong Kong Polytechnic University. Due to the design of the curricula, students acquire only a basic training in computing. For Electrical Engineering (EE) students, C++ programming, basic assembly language programming and fundamental computer engineering are included in the first and second-year curriculum. But in the case of Transportation Systems Engineering (TSE), students only learn C++ programming in their first year of studies. Many students find programming irrelevant to their major studies and are not interested in learning the subject. Therefore, when students are required to apply their computing knowledge to implement their final-year projects, they find it difficult partly because their knowledge is built by memorizing facts which is now fading and also due to their training being not comprehensive. In order to help the students, we would like to implement a system through which students can learn both software and hardware techniques relevant to implementing an embedded system themselves. We targeted embedded system because many final-year projects developed by students can be classified as embedded systems, projects such as auto-pilot system for an electrical vehicle, regenerative braking system and hybrid energy source management system are some examples. For TSE students, without the training in computer engineering, they need to put in extra efforts in order to achieve similar tasks; hence the need for a self-learning tool is more obvious.

Our objective is to develop a system that can supplement our teaching for which the 8051-type microprocessor (the Analog Device ADuC832 [1]) is used, therefore, our system is also targeted for the same microprocessor instead of the Arduino or the LEGO system. With the help of the system, our students are able to learn the basic syntax of programming the processor, using C language, as well as the design of certain hardware modules such as a motor driver, keypad etc.

In this paper, we will first discuss the design of the system in the next section which is followed by details regarding the architecture of the system. In Section IV, the results of evaluation of the system will be presented and this is followed by the conclusions.

II. SYSTEM DESIGN

A. *System Model*

A typical embedded system involves both hardware and software components. The hardware may include sensors and actuators, while the software is used to control these hardware modules. For example, to implement an auto-pilot system, we can apply camera to locate the road markings and motors to drive the robot. The software written should process signals received from the camera and then produce a proper output to control the actuators so that the robot can follow a path. The basic operating mechanism of the system is to process input signal coming from input devices and then output a proper control signal to various output devices according to the program written by the user. From our observations, such model maps well to most projects implemented by our final-year students. So the operation flow of the system, as depicted in Fig. 1, is based on the input/output interaction between the different components.

The initial stages of the operation mechanism are defining the input and output modules included in the current design. The system will provide a list of available I/O components for the user to select and the user only need to input information regarding the port and pin number that the component is connected to; all these are through a graphical user interface (GUI). Once the I/O components have been defined, the next step which is optional, user can also define the initial operation to be performed when powered is on. The next step is most important; it allows the user to relate the action to be performed

This work was supported by a small-scale learning and teaching development project grant from the The Hong Kong Polytechnic University (Project code: LTG09-12/SS/EE1).

by the output device according to the status of the input condition. The GUI for defining the relationships between input signal to output operation is shown in Fig. 2. For example, if the input device is an IR sensor and the output device is a 7-segment display then the user can configure the system to execute the desired action such as when the IR sensor is ON then the display will show the value 1. Once again, all these are completed by using the corresponding GUI. The final step is to create the C program according to the information given by the user.

Once the source program is created, the student can examine the program in order to learn its syntax. To test the program, the student should connect the components as defined and then compile the program using the Keil compiler [2]. As a learning process, student can modify the source program directly in order to test his/her understanding of the programming language. Furthermore, the student can create another program with different I/O combinations to compare the various programming approaches. Therefore, our system provides students an environment to learn by doing [3].

B. Hardware Modules

In order to provide students a working system to test the concept of an embedded system, there must be some hardware modules. As mentioned in the Introduction, the microprocessor adopted in the system is the ADuC832 from Analog Devices. The processor has the 8052 core and therefore it is compatible with the 8051 instruction set. Besides the processor module, there are some basic I/O modules including keypad, display, different types of motors and IR sensors. Based on the available modules, it is possible to implement a basic speed control of a DC motor which represents a typical robotic system.

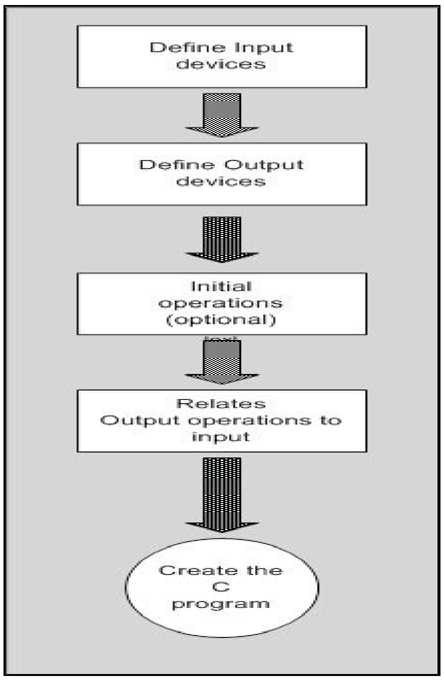

Define Input devices
Define Output devices
Initial operations (optional)
Relates Output operations to input
Create the C program

Figure 1. Operation flow of the system.

The hardware components do not require special design consideration provided that they can be interfaced with the I/O ports of the processor. On the other hand, the software of the system is more significant and is presented in the next section.

III. The Software System

The major role of the software component is to assist students to develop a proper C language program for the ADuC832 processor so that control of the input/output devices can be achieved. Based on the design of a similar system [4], which was tailored for assembly language programming, we had identified the following design criteria:

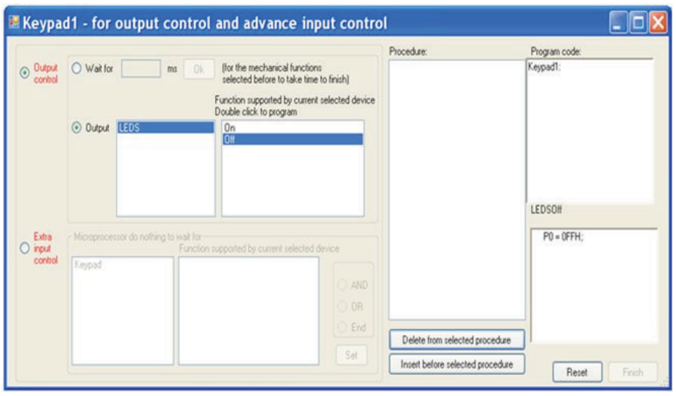
1) user-friendliness;

2) proper support of hardware modules; and

3) able to produce the correct C program.

In order to make the system user-friendly, a graphical user interface (GUI) based on the Windows Forms programming [5] is adopted as it is easy to implement and maintain. Most importantly, it is executable in most computers available in our campus.

There are many different types of I/O modules that can be included in our system, therefore, it must be expandable to incorporated new modules easily. The approach used is based on definition files which are used to identify features of a device as well as the required program codes to achieve those features. The definition files are simple text files and can be created by any text editor. There are two master files, one for storing all the available input modules and the other is for the output modules. If the name of the module is not included in the master files then users will not able to utilize it in the rest of the process.


Keypad1 - for output control and advance input control

[✓ Output control] Wait for ms (for the mechanical functions selected before to take time to finish)

Function supported by current selected device Double click to program

[✓ Output] LEDS — On, [✓ Off]

Extra input control — Microprocessor do nothing to wait for — Function supported by current selected device — Keypad

Procedure:

Program code:
Keypad1:

LEDSOff
P0 = 0FFH;

Ok
AND
OR
End
Set
Delete from selected procedure
Insert before selected procedure
Reset
Finish

Figure 2. The GUI to determine operation performed by an output device.

Besides the master files, the files pertained to the device are the most important. For example, if the device is a LED module that can perform two functions – ON and OFF then three files will be created, namely LEDS.txt, LEDSON.txt and LEDSOFF.txt. The file, LEDS.txt, will define functions supported by the device and in this case – ON and OFF. The LEDSON.txt file contains programming information regarding how to turn-on the LED and in this example, a single line of code (P* = 0FFH) is needed. The symbol 'P*' in the statement represents the port that the LED is connected to. Since which port is being used cannot be determined in advance so the symbol '*' will be replaced by the actual port number based on information provided by the user. There is one advantage of using such an approach. As there are various methods to implement a function so the designer, or the instructor, can highlight different programming techniques with different modules. For example, a LED can be used to emphasize how to output data from a port, whilst the keypad can be used to illustrate the IF-ELSE construct. Alternatively, different versions of the same device can be provided so that the user can learn different approaches to achieve the same goal.

When a device is being selected the information related to its functions will be extracted from corresponding files. The process of converting user's input into a correct C program is relied on the source codes provided in the function specific files, such as LEDON.txt, described in the previous paragraphs. The core operations performed by the system are:

1) substituting the port number and pin number used by the device; and
2) combining all necessary components into a valid program.

IV. System Evaluation

The design and architecture of the system discussed in the previous sections are evaluated for teaching microprocessor programming. In this section, we will describe how the system was evaluated. Electrical Engineering students were presented with the system and required to achieve the task of writing a program to control the speed of a DC motor based on the PWM mechanism with a keypad acting as the input device. Our objective is to gauge the effectiveness of the system as a self-learning tool, therefore, some features of the system were disabled. Hence, students must modify the program generated by the system in order to complete the task. In this case, the system only supports a single speed setting which is 50% duty-cycle. Therefore, students must study the program which can only output a 50% duty-cycle and then extending its functions to include two more speed levels namely slow (30% duty-cycle) and fast (70% duty-cycle).

A simple manual was given to the students so that by following the steps, students can learn how to make use of the system. Students were asked first to go through a simple exercise in order to learn the basic operating mechanisms of the system. At the end of the exercise, a survey was conducted and 32 students participated in the survey.

Regarding functionalities of the system, majority of the students agree that the system is easy to use (over 80%) and the system can produce a proper C program based on information provided by users. It is interesting that only about 63% of the students would like to use the system to implement an actual system and only 50% of the students think that other subjects should also provide similar tools. Certainly, the most important question would be related to the learning outcome. Half of the students, 56% and 44% respectively, agree that the system can help them to learn as well as they have learnt some basic C language programming syntax for the microprocessor. The other half is mainly neutral and only a small percentage (6% and 3%) is holding a negative attitude towards the matter.

In addition to the survey, students were also given a quiz to evaluate their understanding of the programming syntax of the C language by comparing to those of an assembly language. Results from the quiz indicate that over 90% of the group were able to answer the questions correctly. This could be another indicator of their learning outcome.

V. Conclusion

In this paper, we discussed the design of a system which can be used as a self-learning tool for C language programming for an 8052 core microprocessor. The system allows user to implement an embedded system very rapidly provided that the proper hardware modules are available. The C source program can be accessed by the user. Therefore, by testing different settings and combinations, it enables the student to understand the different programming syntax and achieving the different programming tasks. With such exercise, self-learning can be accomplished.

The system was evaluated by a group of students who have background training in assembly language programming. From the survey collected from the students, we can conclude that students are in general satisfied with the functionalities of the system. As a self-learning tool, it is too soon to draw a final conclusion regarding its effectiveness as from the survey only half of the students answered positively to questions related to the learning aspect. More studies should be conducted especially in cases where users have limited knowledge in microprocessor systems, for example, students doing TSE program in our Department will be a suitable user group to test the system.

References

[1] Analog Devices, *ADuC832 Data Sheet*. Norwood, MA: Author, 2009.

[2] *Keil Compiler User Manual, ARM* [Online]. Available: http://www.keil.com/.

[3] D. H. Jonassen, "Designing constructivist learning environments," in *Instructional-Design Theories and Models, A New Paradigm of Instructional Theory,* vol. 2, C. M. Reigeluth, Ed. Mahwah, NJ: Lawrence Erlbaum, 1999, pp. 215–239.

[4] C. F. Chau and Y. F. Fung, "A tool for self-learning assembly language programming and computer architecture: design and evaluation," *Comput. Applic. Eng. Educ.*, vol. 19, no. 2, pp. 286–293, 2011.

[5] H. M. Deitel et al., *Visual C++.NET: How to Program*. Upper Saddle River, NJ: Prentice Hall, 2004.