Implementing Irda On The Msp430: A Project Development Under The Undergraduate Research/Co-Op Education Model

Manuel Jimenez¹, and Melisa Nunez-Arzuaga²

Abstract - The combination of undergraduate research and co-op education has been found to be an effective way of building successful relations between academia and industry, as reported by the application of the undergraduate research/co-op educational model (UR/Co-op) developed at the University of Puerto Rico at Mayagüez. This paper illustrates the successful application of the UR/Co-op model for developing an industry-sponsored project in the implementation of the Infrared Data Association (IrDA) protocol on the Texas Instruments MSP430 microcontroller. This protocol has become the de-facto industry standard for short-range infrared data exchange between portable computing and communicating devices. We describe how the structure of the UR/Co-op educational model was used in the development of the project, while providing some technical details of the protocol, which highlight the student learning process. The discussion illustrates how the application of the model provided a framework for a symbiotic relation where both the sponsoring company and the students involved benefit in the exercise.

Index Terms – Co-op experiences, undergraduate research, industry collaboration, educational model.

INTRODUCTION

For over four years, the Electrical and Computer Engineering Department of the University of Puerto Rico at Mayagüez has held, in collaboration with renowned engineering companies, a program that combines traditional cooperative education (co-op) with undergraduate research experiences [1]. This model combines the results independently obtained through each mechanism to provide students with a rounded experience where research outcomes, either in terms of learning experience or particular projects are applied to problems and activities in the co-op practice.

The integration of these two models has produced several interesting projects, which underline the valuable learning experience it provides to participating students, while employers and involved faculty also benefit. The experience reported in this paper illustrates one of the program projects where the addressed problem was that of implementing the Infrared Data Association (IrDA) protocol as a memory and power efficient software stack for embedded applications. The target platform was the MSP430, an ultra-low-power microcontroller unit (MCU) developed by Texas Instruments [2].

The rest of this document has been organized to provide an overview of the UR/co-op model, a description of the project organization, and details of the program logistics. A brief description of the IrDA protocol is included to highlight the students' learning process. Project implementation and validation details provide insight into the clever solutions provided by the students in the hardware and software aspects. Finally, concluding remarks summarize the achievements and outcomes for the involved parties.

UNDERGRADUATE RESEARCH/CO-OP MODEL OVERVIEW

The UR/Co-op model structure consists of three stages, namely Pre-Co-op, Co-op, and Post-Co-op, organized as illustrated in Figure 1.

In the pre-co-op stage, students enter the program as undergraduate research assistants to work in a faculty assigned project. In this first stage, faculty guidance helps students to focus on the particular work to be developed and to gain insight into the research area. For most participants, usually in their sophomore year, this is their first undergraduate research experience where they begin to develop self-learning skills, ability to perform independent work, and to deepen their technical background into the problem being dealt with.

In the co-op stage, second in the model, students participate of a co-op experience. Here, most students continue to work in the projects they initiated back at school either as part of their regular co-op assignment, or as an activity aside from their co-op duties.

In the post co-op, the last stage in the model, students return to school as undergraduate research assistants, where they complete the documentation of their projects and disseminate the outcomes of their work.

Activities in the program typically span for a period of three regular terms plus summer, since the co-op stage is structured to last for one regular semester and the summer
immediately before or after it. This allows students to fully adapt to their new environment both in the corporate and social aspects, to become familiar with their projects, to apply and deepen their knowledge in the chosen field, and to fulfill most of the objectives set forth by their co-op supervisors. Moreover, the program structure takes the students through a rounded learning experience by allowing them to develop basic research skills in stage one, to refine and apply those skills in stage two, and finally to develop their communication skill through presentations and technical reports in the last stage.

**Co-op experience**

The problem addressed in this project was that of developing a power and memory efficient, IrDA compliant software stack, portable to any member of the MSP430 family. Although several levels of implementation have been reported for the IrDA protocol, either as independent ASICs or as soft modules for different microcontroller families [3][4], porting to the MSP430 has been hardly addressed. Besides an early attempt to implement the physical layer of the protocol [5], no complete implementation has been found for the MSP430, that includes discovery, negotiation, and exchange between IrDA compliant devices. Moreover, the requirements of a short code capable of co-existing with other applications in the MSP430 memory, of being portable to all members of the family, and able to take advantage of the low power features available in the host MCU, made the project particularly challenging. These requirements made it necessary for the students to deepen their knowledge not only in embedded systems programming and interfacing, but also on data communications protocols, the technical specifications of the IrDA protocol (over 200 pages of details), the MSP430 architectural and programming specifications, and the criteria for power management within the MCU.

**Project Logistics**

In the first stage of this project, a team of three students was assembled to work on the problem. However, shortly after, one of them withdrew from the project, leaving two students to complete the work. These students made an outstanding work in understanding the IrDA protocol, the MCU specifications, and prototyping the stack in C language. At this point, the preliminary results were presented in a local workshop for the Industrial Affiliates Program [6]. In this initial phase, the collaboration with industry provided technical information about both the IrDA standard and the MSP430.

In the second stage, one student went to the sponsoring company for continuing to the Co-op phase. This student continued to work in the IrDA stack project for her co-op, allowing her to take advantage of the knowledge gained in the first stage. The C prototype of the stack, although functional, did not meet the memory requirements neither fully exploit the low-power modes of the MSP430. Therefore, an implementation in assembly language was required. In accomplishing this task, mentoring by an industry advisor was a key factor for success of the project. The next sections provide an overview of the protocol requirements and the details of its hardware and software implementations.

**Project Implementation**

The structure of an IrDA stack is shown in Figure 2. The first three layers bottom-up above the IR Adapter are mandatory, while those in those on top of IrLMP –except IAS– are optional protocols required only for specific applications. Details of the structure and functions of each layer are available in the IrDA protocol specification [7].

Due to the memory limitations imposed by the requirement of co-existence with other applications, the implemented version of the protocol was the IrDA Lite [8]. This specification is a sub-set of the full IrDA protocol, which provides the minimum requirements for any device to be IrDA compliant. Some of the strategies used in the
development of this specification limit the performance of the stack. For example, speed is restricted to 9600 bps and LAP packet size is limited to 64 bytes.

The following subsections describe the hardware and software design aspects of the protocol. The hardware components are designed to comply with the IrPHY specifications and regulations, while the software components span over all the remaining stack layers.

**Hardware Design**

The hardware design for this application focuses on the interfacing of the TI MSP430F149 and SHARP GP2W004YP IrDA Transceiver. Figure 3 shows a connection diagram of the hardware components. The stack hardware was prototyped around an MSP430F149, which provides 60KB of flash memory and 2KB of RAM, enough for program additions, buffering space, and for storing and manipulating the frames sent back and forth during dialog. It also has six general-purpose, 8-bit I/O ports and provides with two UARTs that enable communications with a compatible serial port in order to develop applications for personal computers.

The SHARP’s GP2W004YP infrared transceiver provided adherence to the IrDA v1.0 ISO specifications [9]. This part can be powered from a 3.0V supply, allowing direct an interface to the MSP430 without additional external circuitry. Moreover, it takes only three MCU I/O pins for transmit, receive, and shutdown functions.

**Software Design**

All software components were written in MSP430 assembly language. Service primitives for IrPHY, IrLAP and IrLMP were implemented according to the IrDA Lite specifications. TinyTP and IrCOMM 3-Wire were implemented in order to provide with a demonstration of the working stack. An eighty-byte FIFO buffer was implemented to store incoming and outgoing bytes in transit through the IR channel, residing in address 0x250h through 0x29Fh. Table 1 shows the RAM memory-map for the data area in this application.

<table>
<thead>
<tr>
<th>Byte variables</th>
<th>0232h-0200h</th>
<th>50 bytes for byte variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word variables</td>
<td>0248h-0234h</td>
<td>11 words for word variables</td>
</tr>
<tr>
<td>Buffer pointer</td>
<td>IR_BUFFER</td>
<td>029Fh-0250h</td>
</tr>
</tbody>
</table>

The implementation of the physical layer was performed using Timer_A. The approach was to design a unit that would behave as a UART but would process data received from and sent to the IR transceiver. The duty cycle is 3/16th of a pulse as specified in the IrPHY documentation. The transmission and reception schemes are very similar.

The IrLAP layer was implemented as a set of functions performing all the tasks described by the primitives and state tables of the IrDA Lite specification. The handling of frames begins by identifying the frame type. Three types of frames are supported: supervisory (S), information (I) or unnumbered (U). Identification is performed by parsing. Once the frame is identified, program control is transferred to the corresponding handling routines.

The implementation of the IrLMP layer adheres to the IrDA Lite specifications as well. This layer verifies the correctness of all fields and ensures that services requested by the peer are correctly supported. It also provides the frame format that carries the data to be exchanged by the two communicating devices.

The IAS services provide a type of yellow pages that store information about other devices. It also provides information about the services supported by the implementation.

The operation of TinyTP involves the exchange of Protocol Data Units (TTP-PDUs) for negotiation and flow control. This effectively adds a single octet of header to the IrLMP-MUX data used to convey increments (credits) to the number of TTP-PDUs frames that may be exchanged in each direction using the underlying LM_Data service. Connect TTP-PDUs exchanged during connection establishment are not regarded as requiring or consuming credit. Segmentation and reassembly are not implemented. Since the IrLAP window size is equal to one, a single TinyTP connection can take full advantage of the underlying IrLAP window.

For the purpose of this particular implementation, when the initial TinyTP connection frame is identified, a credit is issued so that the peer entity can transfer its data. After the MSP430 responds and has received an RR command from the primary, it then issues more credit for the peer entity to continue transmitting data.
IrCOMM’s services are the same as those provided by the IrLMP layer. IrCOMM calls the services provided by the service layer, and this call propagates down the stack. IrCOMM uses the data PDUs from IrLMP to transmit all data and control channel information.

Testing the Stack

Given the nature of the application, the IrDA stack needed to be tested for compatibility with the operating systems that make the most use of IrDA, namely Linux, Windows, and PalmOS.

In order to assess the compatibility with the Linux platform the driver IrCOMM2k was used. This driver, developed by Jan Kiszka, ports the Linux IrDA stack to Windows [10]. Another important element was the network analyzer Ethereal. Ethereal counts with plug-ins with the capability of parsing IrDA frames in order to simplify the debugging process. The combination of these two elements was crucial for the test and debug phase of the project, allowing to verify the compatibility through the use of the IrCOMM2k driver.

An application was developed using the IrCOMM sockets provided by Windows. This application did not require the use of IrCOMM2k and functions exclusively with the services provided by Windows. In order to test this application, a buffer in RAM stored the frames exchanged by the devices. This allowed the debugger to see the frames and check for errors. Some problems were found but solutions were implemented which allowed simultaneous functionality for the Linux and Windows IrDA stacks.

The PalmOS stack was tested in the same way as the Windows stack. The tests were successful corroborating compatibility of the IrDA stack.

CONCLUSION AND FUTURE WORK

A project development in the implementation of the IrDA protocol on the MSP430 has been presented, illustrating the types of projects that can be carried under a combined co-op undergraduate research educational model. The particular application of the model allowed the participating students to have a rounded experience which included study and background development, identifying opportunities to improve previous problem approaches, implementing, fine tuning, and testing their solutions, applying them in a corporate environment, and later disseminating their knowledge.

The sponsoring industry partner was able to document through the project an application report on the task of implementing a complete working version of the IrDA protocol on the target MCU family, and produced a potential in-job trained recruit. The students working in the project surely gained an experience that will live with them forever. Currently they are still at school completing their bachelor’s degrees in EE and CE, considering the development of the testing set-up to obtain an official IrDA certification for the protocol implementation, and enjoying the satisfaction of having completed a job well done.

ACKNOWLEDGMENTS

The authors would like to thank Mark Buccini and Mike Mitchell at the ASP Division in Texas Instruments, Dallas, TX for their valuable help in the development of this project, and UPRM student Pedro Zayas for his contributions in the pre-co-op stage of the project.

REFERENCES