An Effective Implementation of Embedded Rabbit Web Server with HTTP Protocol

[1]Mhd. Farukh Hashmi, [2]Pradeep Dhakad, [3]Baluram Nagaria
[1,2,3]Mandsaur Institute of Technology, Mandsaur, MP, India

Abstract
This paper Presented Implementation of the Embedded Rabbit Web Server, we have used Rabbit 5000 Microprocessor and RCM 5700 Embedded Module and Its Programming Software Dynamic C. The main attention is to the TCP/IP architecture and its different protocols for implementation of Web Enabled services. HTTP protocols, various configurations and its various libraries function are required for implementing Embedded Web server and their simulation results are obtained. To be accessed by any node on the internet requires assigning a URL to the target board and providing connectivity to the internet. by registering a URL and providing two DNS servers, which can even be leased or rented, anytime anywhere control of the remote device is possible RCM 5700 hardware and Dynamic C is used for finding the results as it is the simple Embedded System tool by which can simulate any system.

Keywords
TCP/IP, Ethernet, HTML, HTTP, Rabbit microprocessor,

I. Introduction
Rabbit Semiconductor was formed expressly to design a better microprocessor for use in small- and medium-scale single-board computers. The first microprocessors were the Rabbit 2000, Rabbit 3000, and the Rabbit 4000. The latest microprocessor is the Rabbit 5000. Rabbit microprocessor designers have had years of experience using Z80, Z180, and HD64180 microprocessors in small single-board computers. The Rabbit microprocessors share a similar architecture and a high degree of compatibility with these Microprocessors but represent a vast improvement. The Rabbit 5000 is a high-performance microprocessor with low electromagnetic interference (EMI), and is designed specifically for embedded control, communications, and network connectivity. Extensive integrated features and glue less architecture facilitate rapid hardware design, while a C-friendly instruction set promotes efficient development of even the most complex applications.
The Rabbit 5000 is the first Rabbit microprocessor to have full 16-bit internal bus architecture, providing significant performance improvements when used with external 16-bit memory devices. It also has the ability to support both 8-bit and 16-bit external memory devices. Now, a Digi International brand, running up to 100 MHz, with compact code and support up to 16 MB of memory [11].

II. Hardware design

A. Rabbit core module (RCM 5700)
The RCM5700 is compact module in a mini PCI Express form factor, and incorporates the powerful Rabbit® 5000 microprocessor with integrated 10/100Base-T Ethernet functionality and 128KB of on chip SRAM. The RCM5700 also includes 1MB of onboard flash memory. The Rabbit® 5000 microprocessor features include hardware DMA, I/O lines shared with up to six serial ports and four levels of alternate pin functions that include variable-phase PWM, an external I/O bus, quadrature decoder, and input capture. This equates to a core module that is fast, efficient, and the ideal solution for a wide range of embedded applications. A Development Kit is available with the essentials that you need to design your own microprocessor-based system, and includes a complete Dynamic C software development system [10]. The Development Kit also contains an Interface Board with USB and Ethernet connections that will allow us to evaluate the RCM5700, and a Prototyping Board to help you to develop your own applications. Subsystems of Rabbit 5700 is shown in fig 1

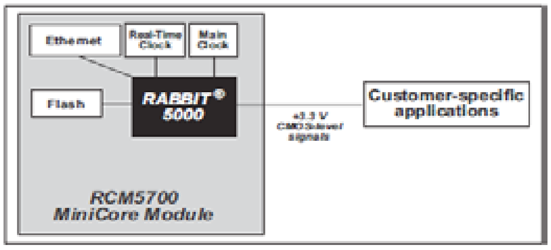


Fig. 1: RCM5700 Subsystems

1. RCM5700 Features
A. Small size: (30×51×3) mm
B. Speed: Rabbit 5000 run at 50.0 MHz
C. Up to 35 general-purpose I/O lines configurable up to four alternate functions
D. 3.3 V I/O lines
E. Six CMOS-compatible serial ports - four ports are configurable as a clocked serial port (SPI), and two ports are configurable as SDLC/HDLC serial ports.
F. Ethernet PHY interface chooses Ethernet interface automatically based on whether a crossover cable or a straight-through cable is used in a particular setup
G. External I/O bus can be conFig.d for 8 data lines, 8 address lines (shared with parallel
H. I/O lines), and I/O read/write
I. 128KB SRAM (on Rabbit 5000 chip) and 1MB flash memory
J. Battery-backable real-time clocks
K. Watchdog supervisors

2. RCM5700 pin configuration
Fig. 2 shows the RCM5700 pin outs.

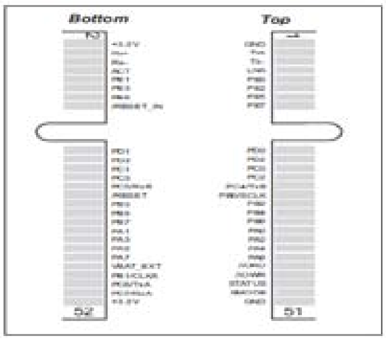


Fig. 2: RCM5700 pin outs diagram

B. Dynamic C
Dynamic C has been in use worldwide since 1989. It is specially

designed for programming embedded systems, and features quick compile and interactive debugging. You should do your software development in the flash memory on the RCM5700. The flash memory and options are selected with the Options > Program Options > Compiler menu. The RCM5700 and Dynamic C were designed to accommodate flash devices with various sector sizes in response to the volatility of the flash-memory market. Developing software with Dynamic C is simple. Users can write, compile, and test C and assembly code without leaving the Dynamic C development environment. Debugging occurs while the application runs on the target [8]. Alternatively, users can compile a program to an image file for later loading. Dynamic C runs on PCs under Windows NT. for additional information if you are using a Dynamic C under Windows Vista. Programs can be downloaded at baud rates of up to 460,800 bps after the program compiles.

Dynamic C has a number of standard features.

A. Full-feature source and/or assembly-level debugger, no in-circuit emulator required.
B. Royalty-free TCP/IP stacks with source code and most common protocols.
C. Powerful language extensions for cooperative or preemptive multitasking
D. Loader utility program to load binary images into Rabbit targets in the absence of Dynamic C.
E. Provision for customers to create their own source code libraries and augment on-line help by creating "function description" block comments using a special format for library functions.

III. TCP/IP based embedded internet working

The TCP/IP layers are located in the middle of the entire communication stack. They ensure proper point-to-point communication and take care of the routing and delivery of packets. To get any device hooked-up to the Internet successfully, at least one path through the entire communication stack is required. If the device supports multiple applications and/or multiple physical interfaces, several paths through the stack are possible [2, 4].

The Dynamic C TCP/IP is intended for embedded system designers and support professionals who are using a Rabbit-based controller board. Most of the information contained here is meant for use with Ethernet- or Wi-Fi-enabled boards, but using only serial communication is also an option. Knowledge of networks and TCP/IP (Transmission Control Protocol/Internet Protocol) is assumed. The Dynamic C implementation of TCP/IP comprises several libraries. The main library is DCRTCP.LIB. As of Dynamic C 10.54, this library is a light wrapper around DNS.LIB, IP.LIB, NET.LIB, TCP.LIB and UDP.LIB. These libraries implement DNS (Domain Name Server), IP, TCP, and UDP (User Datagram Protocol). This, along with the libraries ARP.LIB, ICMP.LIB, IGMP.LIB and PPP.LIB are the transport and network layers of the TCP/IP protocol stack [9].

We have provided a number of programs demonstrating various uses of TCP/IP for networking embedded systems. These programs require you to connect your PC and the RCM5700 module together on the same network. This network can be a local private network (preferred for initial experimentation and debugging), or a connection via the Internet Obtaining IP addresses to interact over an existing, operating, network can involve a number of complications, and must usually be done with cooperation from your ISP and/or network systems administrator. Socket Architecture is shown in Fig. 3

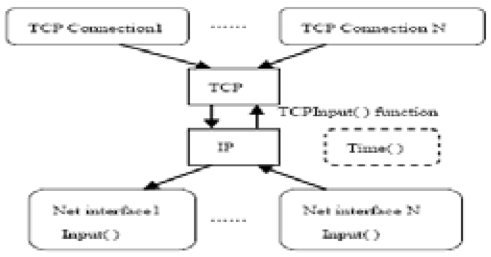


Fig. 3: Socket Architecture of TCP/IP

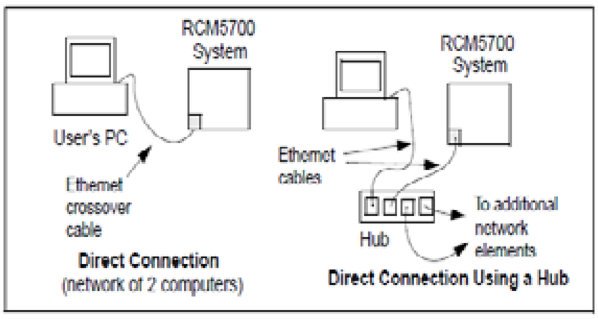


Fig. 4.Network Connection

IV. Problem formulation

Dynamic C provides libraries that implement most of the functions required to implement a web server, more formally known as an HTTP (Hypertext Transfer Protocol) server. (The browser is formally called an HTTP client). You only need to write code specific to your application, such as dealing with I/Os and the Rabbit peripheral devices, and possibly some code to help the HTTP server generate the appropriate responses back to the user's web browser. In addition, there is a small amount of "boilerplate" that needs to be written to include and conFig. the HTTP server and any ancillary libraries such as the TCP/IP suite and file systems [3].

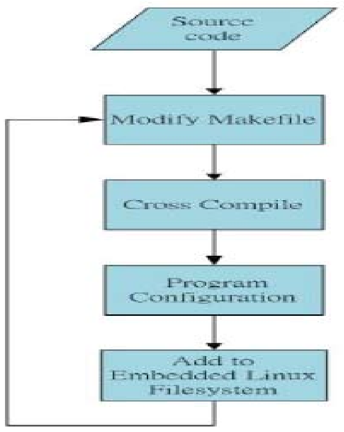


Fig. 4: The development processes

V. Design methodology

We probably need to lay out some forms. Forms, in web parlance, allow the browser's user to fill in some information then submit it to the server [9]. The server then performs the requested actions and sends a confirmation back to the browser. This is the most common means for implementing control of the server as opposed to merely querying it.

Relate to the HTTP upload facility [3]. The last two questions concern the overall design of your application; in particular, a large application may necessitate more storage than is usually available for a given Rabbit product, and may require a sophisticated file system to manage the large number of resources. Since the terms small, medium and large are rather vague, we shall define them by example. A small application would be limited to less than 10 different web pages, and up to about 30 different "controls" (buttons to press, dials to twiddle, options to select etc.). A large application may have upwards of 100 pages, and more than 10KB of configurable data. A medium application sits, as you might expect, near the middle of these [5].

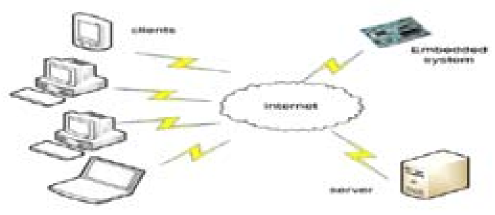


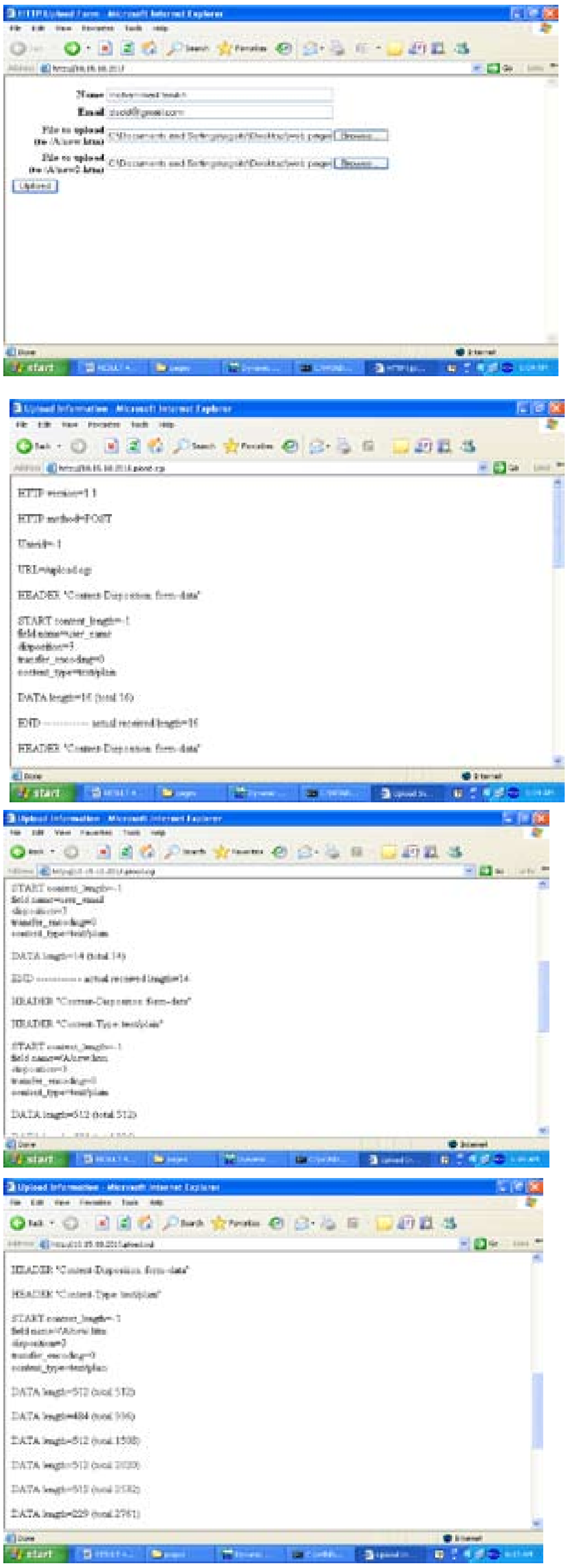
Fig. 5: General Diagram of Client Server Embedded Design using Rabbit Processor

With the introduction of Dynamic C 10.54we have taken steps to make it easier to run many of our sample programs. You will see a TCPCONFIG macro. This macro tells Dynamic C to select your configuration from a list of default configurations. You will have three choices when you encounter a sample program with the TCPCONFIG macro [1].

Development of HTML and dynamic code for the web server: This Section is intended to be a detailed description of the HTTP server, and how it interfaces to other libraries, such as Zserver and TCP/IP. HTTP is implemented by HTTP.LIB, thus you need to write #use "http.lib" near the top of your program. HTTP depends on the Dynamic C networking suite, which is included in your program by writing #use "dcrtcp.lib".Setting up the network subsystem is a necessary pre-requisite for use of HTTP. However, it can be quite simple for test applications and samples to initialize the network subsystem. In the file tcp_config.lib are predefined configurations that may be accessed by#define of the macro TCPCONFIG. For instructions on how to set up different configurations, look in the file \LIB\TCPIP\TCP_CONFIG.LIB. HTTP makes use of the Zserver library to manage resources and access control. The previous chapter discusses Zserver. When reading this chapter on the HTTP server, it will help if you are familiar with Zserver, its interfaces and capabilities [6].

VI. Simulation results

This program demonstrates the HTTP file upload facility. The CGI merely dumps the action codes and information which is presented by the server. After the upload is completed, the CGI switches back to the initial form (index.html). This Sample used for details on using the default upload handler CGI, and adding security. When we browsed the URL 10.15.10.201.It Demonstrate the HTTP file upload facility. The CGI merely dumps the action codes and information which is presented by the server. After the upload is completed, the Data length, total data length and received data length is shown on Web page.


HTTP Upload Form - Microsoft Internet Explorer
Name
Email
File to upload (to /A/new.html)
File to upload (to /A/new2.html)
Browse...
Upload
Upload Information - Microsoft Internet Explorer
start

```
HTTP version=1.1

HTTP method=POST

Userid=-1

URL=/upload.cgi

HEADER "Content-Disposition: form-data"

START content_length=-1
field name=user_name
disposition=3
transfer_encoding=0
content_type=text/plain

DATA length=16 (total 16)

END ---------- actual received length=16

HEADER "Content-Disposition: form-data"
```

```
START content_length=-1
field name=user_email
disposition=3
transfer_encoding=0
content_type=text/plain

DATA length=14 (total 14)

END ---------- actual received length=14

HEADER "Content-Disposition: form-data"

HEADER "Content-Type: text/plain"

START content_length=-1
field name=/A/new.htm
disposition=3
transfer_encoding=0
content_type=text/plain

DATA length=512 (total 512)
```

```
HEADER "Content-Disposition: form-data"

HEADER "Content-Type: text/plain"

START content_length=-1
field name=/A/new2.htm
disposition=3
transfer_encoding=0
content_type=text/plain

DATA length=512 (total 512)

DATA length=484 (total 996)

DATA length=512 (total 1508)

DATA length=512 (total 2020)

DATA length=512 (total 2532)

DATA length=229 (total 2761)
```

Fig. 6 : Simulation Results

VII. Conclusion

The full featured TCP/IP stack from Rabbit along with other more proprietary tools such as Rabbit Web, allow a system designer to web enable a design with a minimum investment in time and money.

As with any Technology, tools used to mold the technology into products are as important as the technology. Rabbit's highly integrated embedded processor is remarkable technology

References

[1] Rawashdeh, Zaydoun, Mahmud, Syed Masud, "AC 2009-2441: Teaching real-time embedded systems networking and assessment of student learning", ASEE Annual Conference and Exposition, Conference Proceedings, 2009.

[2] Fisher, Joseph A., "moving from embedded systems to embedded computing", CASES 2003: International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, p 1, 2003.

[3] "TCP/IP Lean: Web Servers for Embedded Systems", Jeremyy Bentham CMP Books, 2002.

[4] Jan Axelson, "Embedded Ethernet and Internet Complete", Lakview Research, 2003

[5] Addison Wesley, "An embedded Software Primer", David Simon, 1999.

[6] "TCP/IP Application Layer Protocols for Embedded systems", M.Timm Jones, Charles River Media, June 2002.

[7] Danny Goodman, "Dynamic HTML: The Definitive reference", O'Reilly & Associates 2002.

[8] Dynamic C Functional Reference Manual.

[9] Dynamic C TCP/IP User's Manual.

[10] Rabbit Core RCM5700 getting Started Manual.

[11] Rabbit 5000 Microprocessor user's manual.

[12] Schreiner, Dietmar, Schordan, Markus, "Source code based component recognition in software stacks for embedded systems", 2008 IEEE/ASME International Conference on Mechatronics and Embedded Systems and Applications, MESA 2008, pp. 463-468

Mohammad Farukh Hashmi received his B.E. degree in Electronics & Communication from Mandsaur Institute of Technology, Mandsaur, India, in 2007, the M.E. degree in Digital Techniques and Instrumentation from SGSITS, Indore, India, in 2010. He was a, lecturer, with Department of Electronics & communication in MIT Mandsaur, Rajiv Gandhi Prodhogiki University, in 2007-08, 2010 till date respectively. Presently He is a Lecturer of College MIT Mandsaur, RGPV Bhopal, in Sept 2010 respectively. His research interests include Digital signal processing, Embedded System, VLSI Design and Digital Image Processing. At present, He is engaged in Image Compression Techniques using DCT and DWT and Circuit implementations in VLSI application