รายการอ้างอิง

Aho A;Sethi R;Ullman J. Compilers Principles, Techniques, and Tools. Canada. : Addison-Wesley Publishing Company, 1997.

Bakken S; and others. PHP Manual. PHP Documentation Group, 1999.

Barenstein, N; Freed, N. MIME (Multipurpose internet Mail Extensions). RFC1521, 1993.

Berners T; and others. Hypertext Transfer Protocol – HTTP/1.0. INTERNET DRAFT, 1996.

Berners T; and others. Hypertext Transfer Protocol – HTTP/1.1. INTERNET DRAFT, 1999.

Franks J. HTTP Authentication Basic and Digest Access Authentication. RFC2617 June, 1999.

Intel Corporation. MCS 51 MICROCONTROLLER FAMILY USER'S MANUAL. February 1994.

Lawrence S. Web Interface Development for Embedded Systems. Agranat Systems Inc. : Embedded Systems Conference, 1999.

National Semiconductor. DP83902A ST-NIC. PRELIMINARY November 1995.

Netscape Communications Corporation. JavaScript Reference. 1997.

Netscape Communications Corporation. HTML Tag Reference. 1998.

O'Brien M. Open Source Embedded Web Servers. GoAhead Software. : Embedded Systems Conference, 1999.

Plummer, D. An Ethernet Address Resolution Protocol. RFC826 November, 1982.

Postel, J. Internet Protocol. RFC791 September, 1981a.

Postel, J. Internet Control Message Protocol. RFC792 September, 1981b.

Postel, J. Transmission Control Protocol. RFC793 September, 1981c.

Snell R. Web-Based Device Monitoring and Control. Intelligent instrumentation Inc. : Embedded Systems Conference, 1999.

Stevens W. TCP/IP Illustrated. Volume1. Canada. : Addison-Wesley Publishing Company, 1994.

Stevens W; Wright G. TCP/IP Illustrated. Volume2. Canada. : Addison-Wesley Publishing Company, 1995.

Wingard S. Embedding HTTP Functionality for Web-Based Configuration and Management of Devices. Spyglass, Inc. Embedded Systems Conference, 1999.

Pratt T;Zelkowitz M. PROGRAMMING LANGUAGES Design and Implementation. USA. : PRENTICE-HALL International Inc., 1996.

Witchey N. Designing and Embedded Web Server. Institute of Electrical and Electronics Engineers, Inc.: IConline, 1998.

ภาคผนวก

ภาคผนวก ก

Development of a reconfigurable Embedded Web Server.

Krerk Piromsopa, Boonchai Sowanwanichakul.

Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University,
254 Phayathai Road, Patumwan,
Bangkok, Thailand 10330.
Tel. (662)-218-6956, Fax. (662)-218-6955
g41kpr@cp.eng.chula.ac.th

Abstract

As the Internet continue to grow, the number of devices and appliances connect to Internet are increasing. Consequently, the web server is developed to embedded with these devices for access, monitor and control. It's essential that this web server be Reconfigurable to make it function with any devices or appliances. Scripting Language such as javascript is used to customize, manage and reconfigure the system in order to integrate with different environment. This paper provides the issue of how to create an embedded web server box, the design requirements by using Intel MCS-51 the 8 bit microcontroller as a main processor with the power of sever-side script and Password Authentication.

1. Introduction

To obtain data from an embedded devices or appliances can be difficult. Traditionally, the data has been transferred through dedicate serial port. That means the numbers of devices connected is the more serial I/O interfaces require. If the terminal supported graphics, it might also be necessary to write a graphical interface: otherwise the data would dump out as straight text. By using an embedded Web server, developers can format and display the same data with HTML though any standard browser. Moreover, communication can be use Ethernet and HTTP can handle the transfer of larger amounts of data to any device on the network.

The design of embedded systems is the state of art computing system. It's the meeting of Qualities of service and Pricing. In the other word, the computer can control the cooking program in your microwave oven but it's not reasonable to use the high performance mainframe computer as the small and easy using microcontroller one. As a result this work will use the MCS-51 family from Intel which is widely use in the industry and appliances. In contract, the small 8bit microcontroller may be powerful enough for control simple device but it might be too slow to directly connect to the Ethernet network. This way, a network-interfacing controller is required. (As every systems need a dedicate NIC Adapter to connect to the network)

While each vendor try to develop their own web server for embedded into their device, the idea of how to create an Open Source Embedded Web servers is presented by GoAhead Software. Once the Open Source Embedded Web server is used, the developer have to modify and recompile the source code to make it function with their desire hardware. This is a strong drive for create the Reconfigurable embedded web server that is working as a component. This way, if the developer want to connect their device to the web, they can place this component as plug-in module and write some server-side script to satisfy their work.

In order to let the embedded web server gain an ability of reconfigured for functioning in any environment, the scripting language is the must. Scripting languages have proven particularly adept at integration applications, where new functionality is layered on top of existing components and resources rather than built from scratch. Like this, if the simple embedded web server is being built, it can be configured to working with any device as it'll connected through the I/O port of the microcontroller. Moreover, once useful resources were made available, Thus a strong security system was needed. That's the means of security authentication that should be also implemented to the systems. The very simple but powerful is the using of what you're known. (the password) as an authentication system.

This paper will take a look at the role and implementation for create and use a Reconfigurable embedded web server. The standard and protocol that must be meet and how to keep the system more secure with basic web password authentication.

2. Hardware Design Considerations.

The embedded web server is named as a thin server. With all the function that should be support in order to services, the server requires a communication channels with the client, and must provide enough memory for storing the web pages. Consequently, the network interface and Memory will be considerate in this section.

Network the embedded devices, cannot be easily done as the Personal Computer which require only a network adapter to plug in to the mother's board. Since the network interfaces adapter for PC usually comes in the standard bus such as ISA or PCI bus. Which is not support in the small microcontroller. The Alternative way is to build the LAN adapter myself. First step in creating a LAN adapter is selecting the Network Interface Controller. There are several Network Interface Controllers (NIC) from various vendors. Some of them support 100Mbs Ethernet. After take a look at each NIC, they can be classified by the bus width. In order to functioning with MCS-51 microcontroller, the 8bit bus width is chosen. The NSC DP83902A is selected since it's the very common compatible with NE2000 and the packet driver is easily ported to the MCS-51.

RAM is not only functioning as a data memory for program, but also must be acting as File Systems for the web server. As the common web server usually store information on the base of file and directory structure, It should be easy if the thin server handle it in the same manner. However, it's not quite a good idea to add a disk controller to the systems. The better solution is emulating the file systems using memory. (As if using in Window CE) The battery packed RAM is seeing more and use as an alternative to the more traditional non-volatile memory devices (As describe in Memory Interfacing and Architectures for Embedded Systems. [1]). Anyway, the MCS-51 address is limited to 64Kbyte. Most of them are using as variable and buffer. Consequently, there is not enough address for storing the big file system. This problem is also found on the old Apple computer that is limitation in memory address but still function with CPM by sliding the memory in to small bank. Each time using the memory, the preferred memory bank must be selected first.

Due to the only support of memory mapped I/O, both Memory and I/O must be connected to the same system bus with the different address mapped. The overview of how each component is connected is show in figure 1.

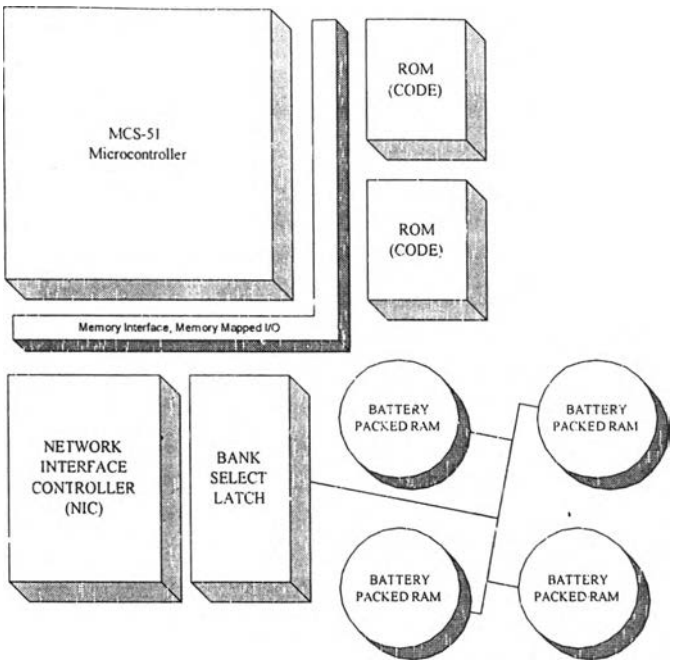


Figure 1 Hardware Block Diagram

3. Protocol and Standard Related

To construct web server, There are many protocol and standard that must be meet to make it functioning. Firstly the network standard will be considerate. Next, we will take a look at how to make web server functioning on the top of network protocol.

IP

Internet Protocol is the heart on the Internet. The current version of IP is 4.0. How to make the IP function is very simple, Since the IP is require only to identifying the network address. Which require only a little checksum function.

ARP

Address Resolution Protocol is the mechanism that make the IP protocol function on the top of Data link Layer, especially on Ethernet network. The goal of ARP protocol is to map the IP Address with the Ethernet MAC Address. ARP helps the client to find the true MAC address of the server. For example, when the client wants to send a Datagram to the server. They will first check their own ARP table if the MAC address of destination IP is known. If do so the out going Datagram is sent. In the other hand, the ARP request packet will be broadcast to ask for the MAC address of the destination IP. The server has to send the ARP response if the incoming ARP request's IP is match with the server IP address.

ICMP

Internet Control Message Protocol usually comes with the kernel of the Operating System. There are many functions that served by.ICMP. The Purpose is for the host to exchange the error messages. This function is required for the embedded web server as a result of the way that each node use to check if the other node address is alive (called PING). Also, a number of ICMP message type is shown in the standard. But the only one that must be implemented is the PING response. The others can be ignored due to the space limitation of embedded systems.

UDP

User Datagram Protocol is a simple, Datagram-oriented, transport layer protocol. UDP provides no reliability. It sends the Datagram that is requested application to the IP layer, but no guarantee for reaching the destination. The UDP is useful to send the emergency from the web server to the other host when there are any error on the embedded system. The implementation is not necessary but since you try to build TCP, the UDP is on the way ready to service for you.

TCP

Transmission Control Protocol is an connection-oriented protocol. Before either end can exchange data, a connection must be established between them. Like the UDP and IP, TCP also have a checksum property. The different between the checksum from UDP and TCP is that the one from TCP is mandatory. TCP is used to handle the data between client and server. The point that TCP is being used by various applications is the abilities to guarantee the correct and successful data transfer. Moreover, TCP can be fragmented with the powerful sliding window. But to implement the fully support for the TCP Datagram. It's might be too complex and some function is useless for embedded system. In this prototype some part of TCP is being simplified to gain a benefit of easily implementation.

HTTP

HyperText Transfer protocol is the engine of the Web server. The main function of HTTP protocol is to provide the data transfer between the client (Browser) and the server. The HTTP can be divided into 2 parts. First is the Header that will carry the request and the response that description the content of the data in the following part. The Second part carry the data, which can be any type of documents. Ex. Images, plain text or binary stream that can be seen in MIME. But most of the document type is the HTML. However, HTTP is running as the application service on the top of TCP/IP network. On the server based web server HTTP will work as an application that transfer the data located in the file system to the network. In the embedded web is view is different. The requirement of the embedded web server is to acquire data from devices as well. We will describe later on how to acquire and setup the device using the scripting language.

4. Web Authentication

Authentication is one from the three methods in create security systems. The others are the Authorization and Accounting, which are differences in each application and should be binding at the runtime (Defined in Project Athena). As describe, the only thing that web server can be done to give the security is the Authentication.

To authenticate, the HTTP is being extended using methods called basic and digest. The basic authentication provides the simple authentication with base-64encode. While the digest authentication is useful for a complex security system. But so far, difficult to be implement on the small embedded system. As a result the basic authentication system is being implemented. And give the scripting language ability to handle the Authorization and Accounting by passing the parameter as a variable to the script process.

5. Embedded server-side scripting language.

Scripting language have proven particularly adept at integration applications. where new functionality is layered on top of exist components and resources rather that built from scratch (John Ousterhout says in the Embedded System Conference on "How Scripting Adds Value to Embedded Systems"). In the server based web server. the server side script is using to bind WebPages to the application in order to reduce the overhead of CGI method. Embedded Web server requires the same theme. Various scripting languages are being reviewed to meet the requirement of web server.

Active Server Pages (ASP) is a Microsoft developed approach to allow the easy creation of dynamic Web pages. The script language can be selected at runtime by specifying the desired language. However, multiple-scripting languages would be rare suitable for embedded web servers. (Michael O'Brien, 1999)

JavaScript has gained widespread attention as a leading scripting language, but its large memory footprint (200-400K) prevented its use in embedded applications. The idea of creating a strict subset of JavaScript called "Embedded JavaScript" is introduce by Michael O'Brien from GoAhead Software. The resulting implementation is a 15K embedded JavaScript interpreter that is enough for embedded application

The Embedded JavaScript consist of global function, global variable. conditional control, loop, simple operation and comments. This feature provides a powerful to create the dynamic web application for embedded system.

6. Software Implementation

The system software part of the Embedded web server can be section into 2 parts. First is the, Network Connection. The other is the Device and I/O Handling. Both of them can be called as "Embedded Kernel". The applications that service on the top of embedded will ask the kernel to create, connect, store and retrieved information through the HTTP protocol. (As shown in figure 2)

The network function is consisting of ARP that is working in the same level as IP. The incoming packet from the Media can be only IP or ARP. Other packets that the encapsulation data is not ARP or IP Datagram will be reject. In case of IP, the MAC address of incoming packet must match with one of the Network Interface Adapter. Otherwise it'll be ignored too.

After passing the IP checking, packet can be identified as ICMP (Ping request), TCP, and UDP by checking at the IP Header. If the incoming packet is not in any of these three types, It's recommend to invoke no process, as they are not necessary to be implemented. In the case that incoming packet is classified as ICMP Ping request, the process only just answer by sending the ping reply immediately. Unlike ICMP, UDP packet will be pass to the application with out checking anything. And the answering Datagram is depending on the application. If the packet is identified as TCP, there must be method of managing and maintaining each session. Since the connection must be synchronization and acknowledge.

The File system management driver must manage the file systems that are based on the Battery packed RAM. As the memory is being sliding to small bank, It's necessary to have a file allocation table as if there are in the floppy disk. The FAT style file system is being implemented. When the Application request for file access the driver then look at the allocation table to checking for the filename and where they are exactly store. Then provide a cursor pointing at a file for read and write.

The web server is working on the top of file system handler and the TCP network. They receive data from network then checking and involve a data from file system that must be process through the Server Side Script Interpreter then send back to the Network. As there are an Interpreter, this means the small lexical analyzer and parser is implemented as a module of web server function.

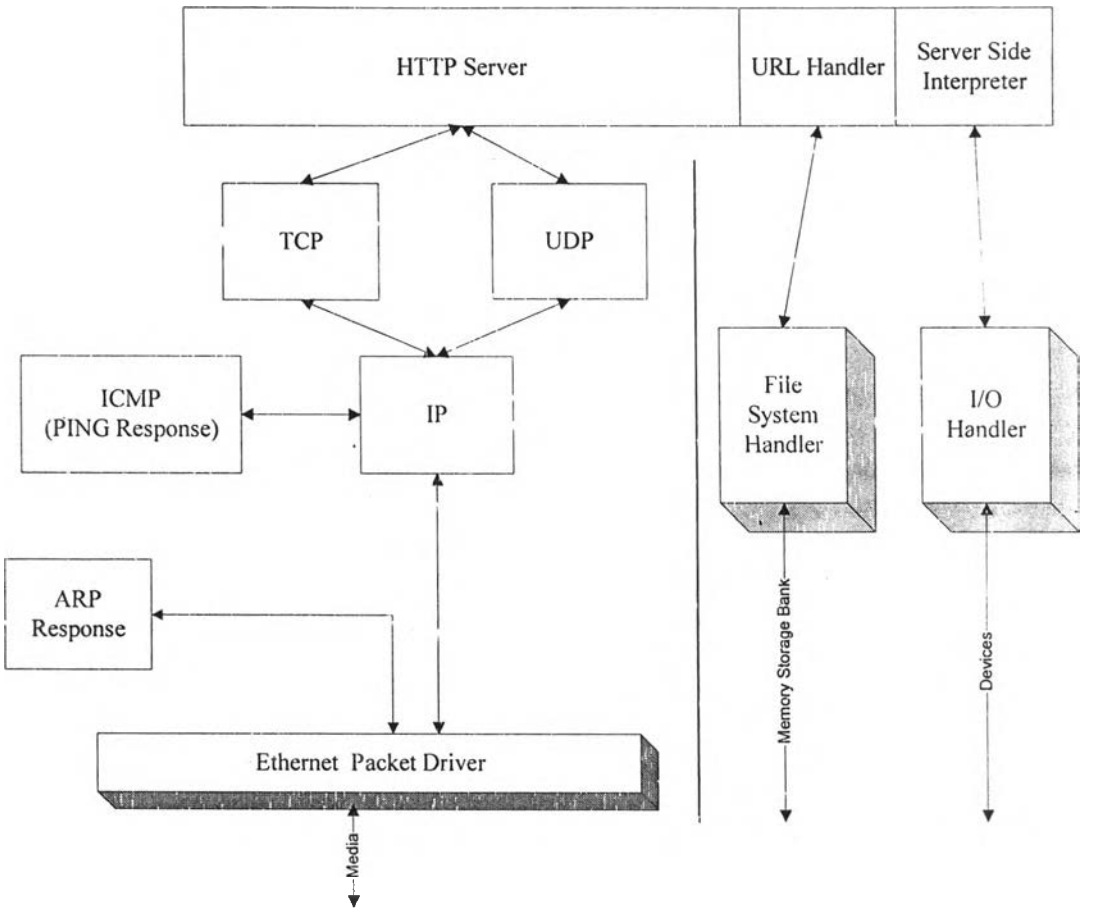


Figure 2 The software component in embedded web server.

7. Web Programming.

To program or reconfigure the web pages to dynamically function with any application, the server-side script is involved. As the server-side script usually content the standard HTML mixed with scripting language. There must be a tag to identify as if the preferred text is script or HTML tag. The ASP style of script is implemented. When the user want to write the script they must put it in the "<%" and "%>" tags.

The server will provide the standard function of JavaScript function such as read or write. With the special function to interface to the device (I/O). More over the user may create their own variable, which will always be the global variable. Or even create their own defined function with a few parameters.

Example. The Server side javascript on how to loop and output the value of I/O port.

```
<B> The Port Value of Server is  </B>
<%
        for (I=0;I<10;I++)
                write("The value of port[" + I + "] is" + getport(I) + "<BR>");
%>
```

8. Application, Future Vision and Development

Reconfigurable embedded web server can be useful in many situations. Since the power of web protocol is an easy way to create the user interface and remote data retrieval. Every devices and appliances should be embedded with the web server. As it'll provide the Open system and information sharing. For example, if the fax machine has a web capability. The whole office can use only one fax machine. The users in the difference part of the office can see the incoming document through the web. The other instance is the telephone answering machine. Suppose that you are away from you home and want to check if there is anybody making a call to you. Then you just dial your notebook to your home and see the answering machine status or even listen to the message.

It's been predicted that every device and appliance should be running over IP protocol. To archive this goal, an embedded web server and TCP/IP should have an open standard as they were in the Server based web server.

9. Conclusions

As the number of devices grow up everyday. It strongly forces to network the devices as the benefit of distributes the information and management. To add the web interfacing to device or appliance, people can utilize the information easier.

As the embedded web server can be reconfigure to function in any environment, The developer can easily connect their exist devices to the web with a little modification of server side scripting language without creating the whole system of their own.

References

[1] Joseph L.Long, "Memory Interfacing and Architectures for Embedded Systems." Embedded System Conference, Class 405

[2] D. C. Plummer, 1992. "An Ethernet Address Resolution Protocol," RFC 826. 10 pages

[3] H. Frystyk, R. Fielding, and T. Berners-Lee, 1993. "Hypertext Transfer Protocol – HTTP/1.0," RFC 1945, 60 pages

[4] J.B Postel, 1980. "User Datagram Protocol," RFC 768, 3 pages

[5] J.B Postel, 1981a. "Internet Protocol," RFC 791, 45 pages

[6] J.B Postel, 1981b. "Internet Control Message Protocol," RFC 792, 21 pages

[7] J.B Postel, 1981c. "Transmission Control Protocol," RFC 793, 85 pages

[8] H. Frystyk, R. Fielding, and T. Berners-Lee, 1993. "Hypertext Transfer Protocol – HTTP/1.0," RFC 1945, 60 pages

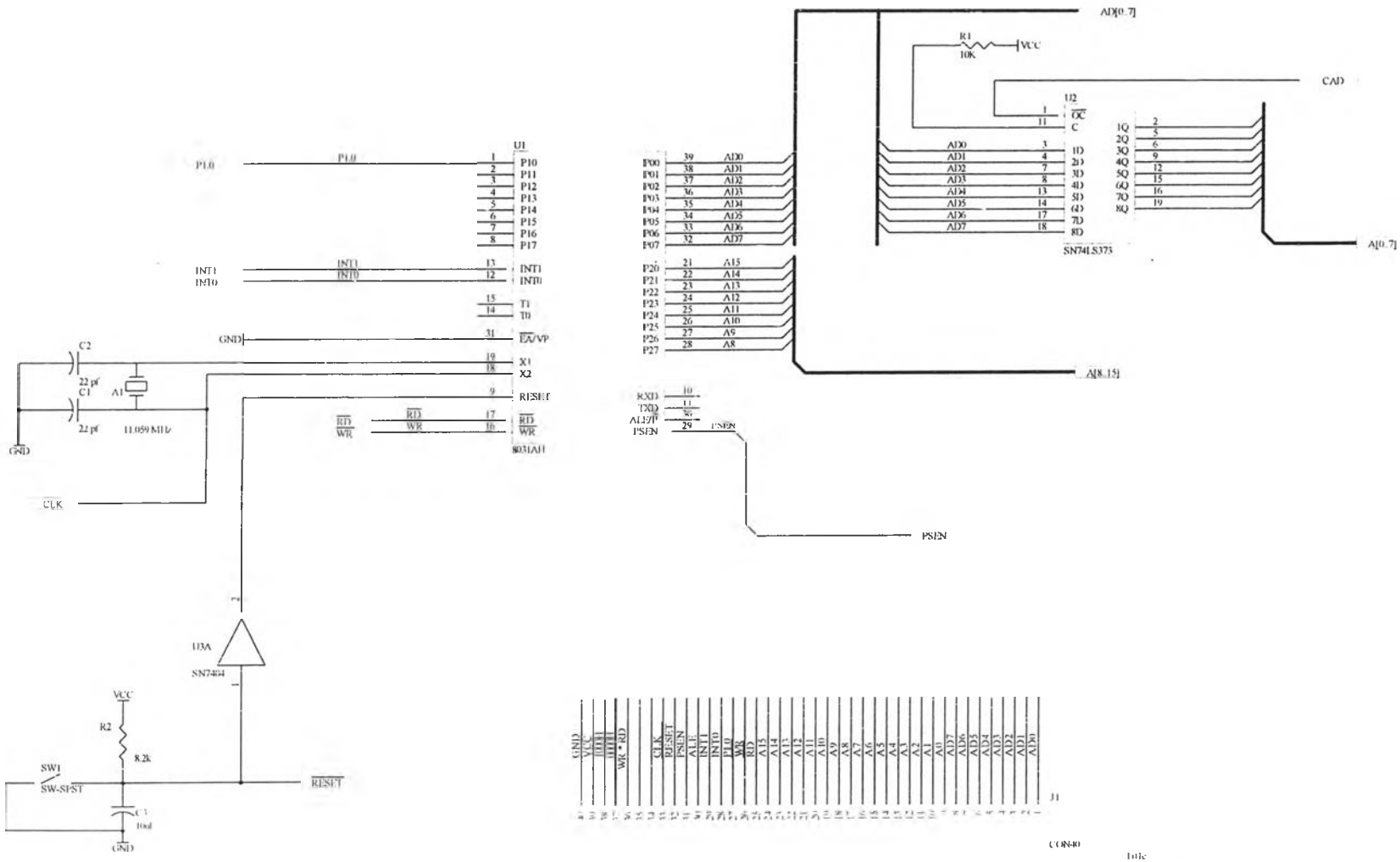
[9] H.Frystyk, J. Gettys, J. Mogul, R. Fielding, and T. Berners-Lee, 1997. "Hypertext Transfer Protocol – HTTP/1.1," RFC 2068,162 pages

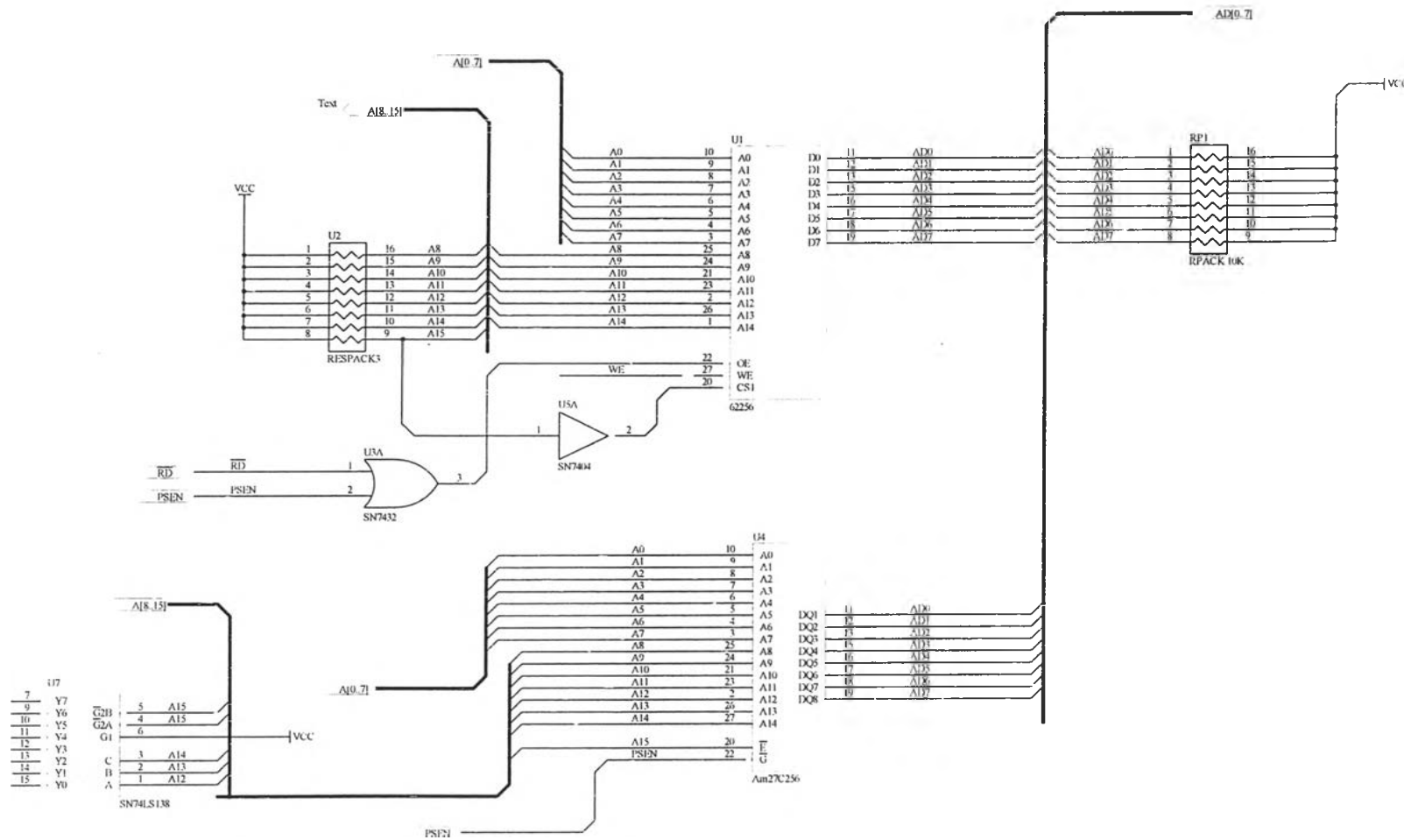
[10] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A.Luotonen, and L.Stewart, 1999. "HTTP Authentication: Basic and Digest Access Authentication," RFC 2069, 34 pages

[11] Steve Wingard, Spyglass, Inc. "Embedding HTTP Functionality for Web-Based Configutaion and Management of Devices," Embedded System Conference, Class 460

[12] Michael O'Brien, GoAhead Software, "Open Source Embedded Web Servers," Embedded System Conference, Class 407

ภาคผนวก ข


AD[0..7]
R1
10K
VCC
CAD
U2
SN74LS373
A[0..7]
A[8..15]
U1
P1.0
INT1
INT0
GND
C2
22 pf
C1
22 pf
X1
11.059 MHz
RESET
RD
WR
8031AH
CLK
U3A
SN7404
R2
8.2k
SW1
SW-SPST
C3
10uf
RESET
RXD
TXD
ALE/P
PSEN
J1
CON40

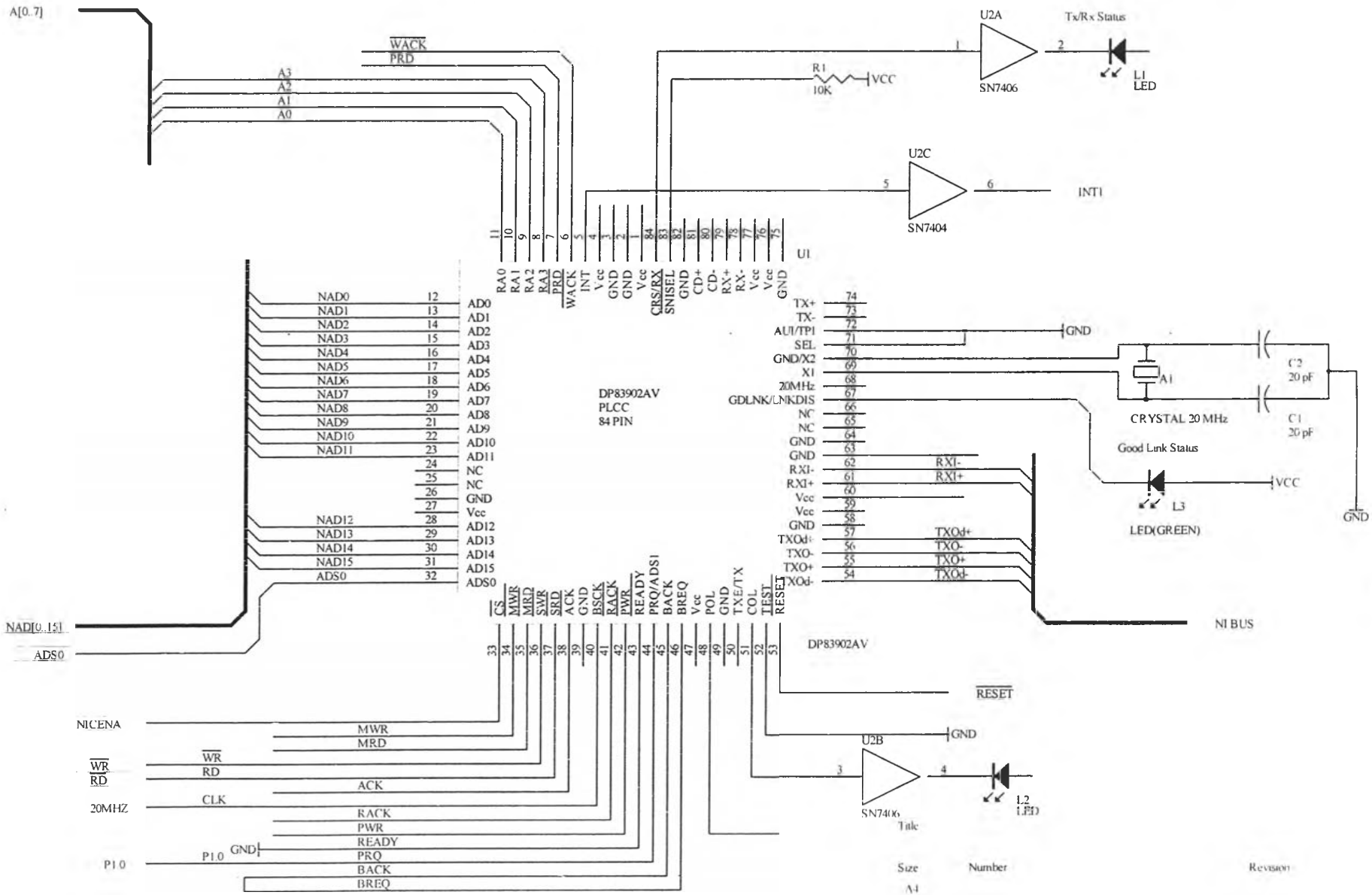


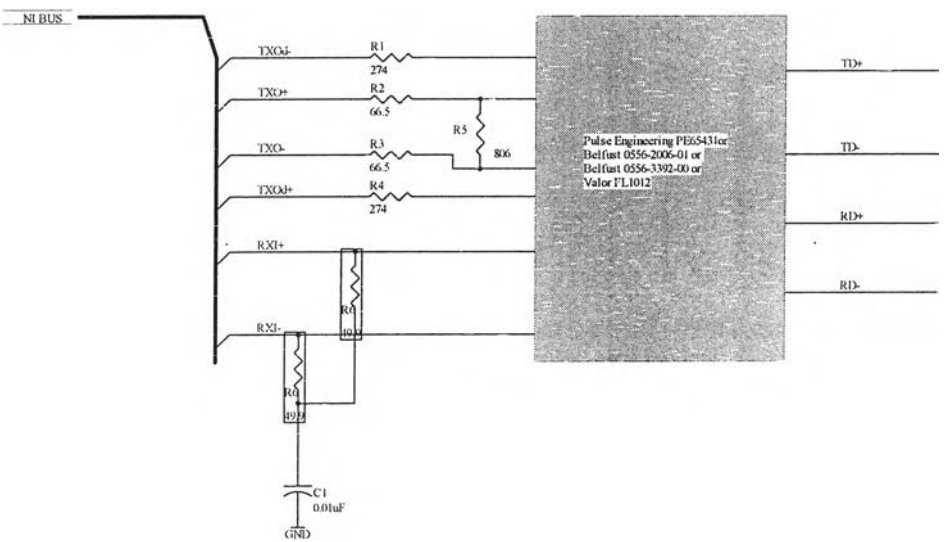
Title

Size: Number: Revision:

Date: 24-Jul-2001 Sheet of

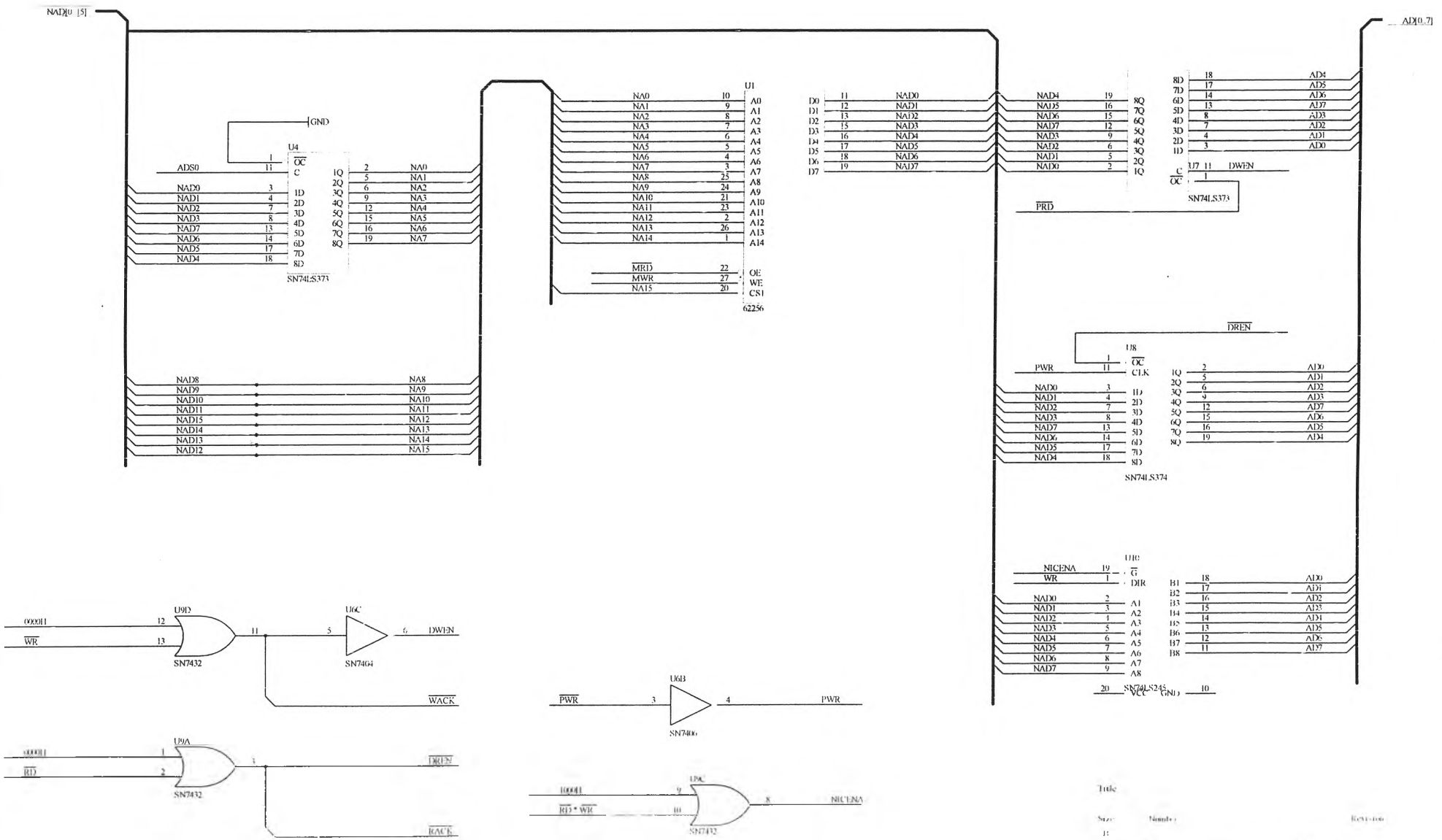
File: D:\HOME\DESIGN\MUM SC\1.ddb Drawn by:


A[0..7]
A3
A2
A1
A0
WACK
PRD
U2A
SN7406
Tx/Rx Status
R1
10K
VCC
L1
LED
U2C
SN7404
INT1
U1
DP83902AV
PLCC
84 PIN
NAD0
NAD1
NAD2
NAD3
NAD4
NAD5
NAD6
NAD7
NAD8
NAD9
NAD10
NAD11
NAD12
NAD13
NAD14
NAD15
ADS0
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
AD8
AD9
AD10
AD11
AD12
AD13
AD14
AD15
ADS0
NC
GND
Vcc
TX+
TX-
AUI/TP1
SEL
GND/X2
X1
20MHz
GDLNK/LNKDIS
RXI-
RXI+
TXOd+
TXO-
TXO+
TXOd-
GND
A1
CRYSTAL 20 MHz
C2
20 pF
C1
20 pF
Good Link Status
L3
LED(GREEN)
NI BUS
NAD[0..15]
ADS0
DP83902AV
RESET
NICENA
MWR
MRD
WR
RD
ACK
20MHZ
CLK
RACK
PWR
READY
PRQ
BACK
BREQ
P1.0
U2B
SN7406
L2
LED
Title
Size
A4
Number
Revision
Date
24-Jul-2000
File
D:\THESIS\DESIGN\NIC.SCH
Sheet of
Drawn By



Title

Size	Number	Revision
B		
Date	24-Jul-2000	Sheet of
File	D:\TIB-SIS\DOC\SIGNNI.SCH	Drawn By

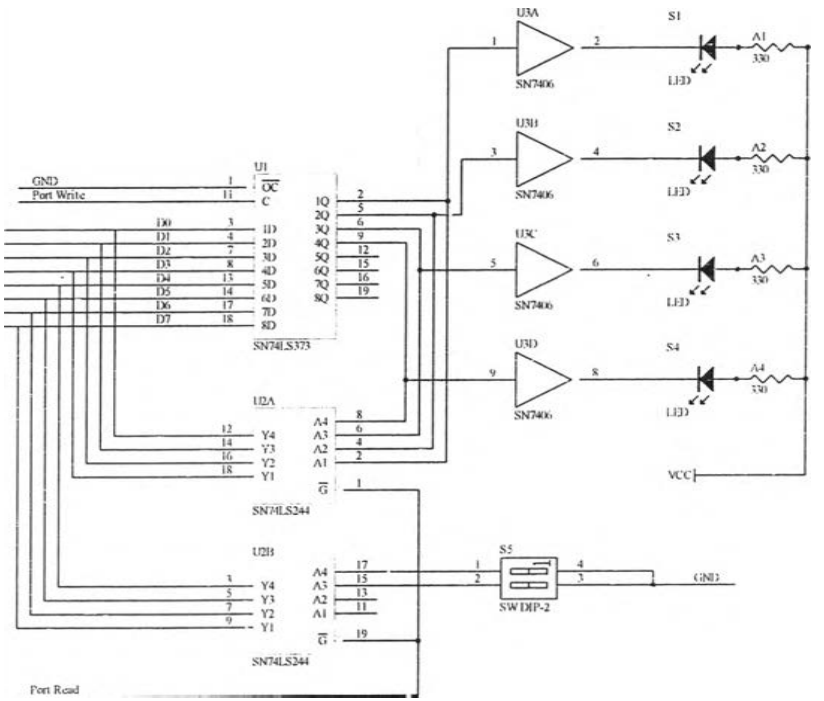


Title

Size: Number: Revision:

Date: 24-Jul-2000 Sheet of

File: D:\THESIS\SCHEMA\MUSCU1 Drawn By:



Title

Size: Number: Revision:

Date: 24-Jul-2000 Sheet of

File: D:\TIJENSIS\DESIGN\DEMO1.SCH Drawn By:

ภาคผนวก ค

```
; check sum routine Module....
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;    COMPUTER ENGINEER.
;    CHULALONGKORN UNIVERSITY

        PUBLIC CHECKSUM
        ;OUTPUT : R6 CHECKSUM H
        ;        R7 CHECKSUM L
        ;INPUT  : R0 STARTADD H
        ;        R1 STARTADD L
        ;        R2 BYTECOUNT H
        ;        R3 BYTECOUNT L

CHECKSUM:
        PUSH    DPH
        PUSH    DPL
        PUSH    A
        ; ODD NO BUG FIX
        MOV     A,R3
        JNB     ACC.0,EVEN
        MOV     DPL,R3
        MOV     DPH,R2
        INC     DPTR
        MOV     R2,DPH
        MOV     R3,DPL
        :
        ;
        MOV     R6,#0
        MOV     R7,#0
CHKLOOP:
        MOV     DPH,R0
        MOV     DPL,R1
        MOVX    A,@DPTR
        MOV     R4,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R5,A
        INC     DPTR
        MOV     R0,DPH
        MOV     R1,DPL
; BEGIN CALCULATE HERE
        CLR     C
        ;ADD AROUND CARRY
        MOV     A,R7
        ADD     A,R5
        MOV     R7,A
        MOV     A,R6
        ADDC    A,R4
        MOV     R6,A
        MOV     A,R7            ; CARRY
FIXED
        ADDC    A,#0
        MOV     R7,A
        MOV     A,R6
        ADDC    A,#0
        MOV     R6,A
        MOV     A,R7
        ADDC    A,#0
        MOV     R7,A
; END CALCULATE
        CLR     C
        MOV     A,R3
        SUBB    A,#2
        MOV     R3,A
        MOV     A,R2
        SUBB    A,#0
        MOV     R2,A
        CJNE    R3,#0,CHKLOOP
        CJNE    R2,#0,CHKLOOP
        MOV     A,R7
        CPL     A
        MOV     R7,A
        MOV     A,R6
        CPL     A
        MOV     R6,A
        POP     A
        POP     DPL
        POP     DPH
        RET
```

```
; ICMP (PING RESPONSE) Module....
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY
; Debug Serial Routine
        EXTERN  C_IN
        EXTERN  C_OUT
        EXTERN  STROUT

            EXTERN SEND_PACKET

            EXTERN CHECKSUM
        ;OUTPUT : R6 CHECKSUM M
        ;        R7 CHECKSUM L
        ;INPUT  : R0 STARTADD H
        ;        R1 STARTADD L
        ;        R2 BYTECOUNT H
        ;        R3 BYTECOUNT L

            EXTERN  MOVSTR
        ; Move Sequence of String
        ; INPUT  : R0 - Src H
        ;          R1 - Src L
        ;          R2 - Des H
        ;          R3 - Des L
        ;          R6 - Byte Count H
        ;          R7 - Byte Count L

            EXTERN  MOVBUF
        ; Move Sequence of String
        ; INPUT  : R0 - DES H
        ;          R1 - DES  L

        PUBLIC  ICMP_RSP
        PUBLIC  SWAP_MAC
        PUBLIC  SWAP_IP
        ;OUTPUT : A PROTOCOL TYPE
OBUF    EQU     E000H
IBUF    EQU     F000H
BOH             EQU 16
; MEM
IPHEADLEN EQU 32H

ICMP_RSP:
        ; CHECK IF ICMP OR NOT
        MOV     A,IPHEADLEN
        ADD     A,#BOH
        MOV     DPL,A
        MOVX    A,@DPTR
        CJNE    A,#08,NOT_ICMP      ;
IS PING REQUEST

;       LCALL   STROUT
;       DB      "PACKET IS
PING",0AH,0DH,00H
        LCALL   CREATE_ICMP
;       LCALL   STROUT
;       DB      "CREATE
ICMP",0AH,0DH,00H
        LCALL   SEND_PACKET
NOT_ICMP:
        RET

CREATE_ICMP:
        ; GET ALL IBUF TO OBUF
        MOV     DPTR,#IBUF
        MOVX    A,@DPTR
        ; GET THE LENGTH - 4B NIC FCS
        CLR     C
        SUBB    A,#4
        MOV     R7,A
        PUSH    A
        ; SAVE THE LENGTH L
        INC     DPTR
        MOVX    A,@DPTR
        SUBB    A,#0
        MOV     R6,A
        PUSH    A
        ; SAVE THE LENGTH H
        MOV     R0,#F0H             ;
GET ALL DATA (SKIP 2HEADER)
        MOV     R1,#02H
        MOV     R2,#E0H
        MOV     R3,#02H
        LCALL   MOVSTR
        LCALL   SWAP_MAC
        LCALL   SWAP_IP

      ; SET ICMP TYPE TO PING
RESPONSE (0)
        MOV     DPTR,#OBUF
        MOV     DPL,#36
        CLR     A
        MOVX    @DPTR,A
        MOV     DPL,#38         ; PLACE
THE CHECKSUM FIELD with 0000
        MOVX    @DPTR,A
        INC     DPTR
        MOVX    @DPTR,A
;       LCALL   STROUT
;       DB              "CLR
CHKSUM",0AH,0DH,00H
        ; PLACE THE NEW LENGTH (OLD -
4)
        ; & CREATE CHECKSUM
        POP     A               ; GET
THE LENGTH BACK H
        MOV     DPTR,#OBUF  ; Get Total
Length
        INC     DPTR
```

```
        MOVX    @DPTR,A
        MOV     R2,A
        POP     A
        MOV     DPTR,#OBUF
        MOVX    @DPTR,A
        CLR     C       ; CAL THE ICMP
LENGTH (Total - R1[36] + LENH[2] )
        MOV     R1,#34
        SUBB    A,R1
        MOV     R3,A
        MOV     A,R2
        SUBB    A,#0
        MOV     R2,A
        MOV     R0,#E0H
        MOV     R1,#36
        LCALL   CHECKSUM
;       LCALL   STROUT
;       DB      "PUT
CHKSUM",0AH,0DH,00H
        MOV     DPTR,#OBUF  ; PLACE
THE CHECKSUM FIELD
        MOV     DPL,#38
        MOV     A,R6
        MOVX    @DPTR,A
        MOV     A,R7
        INC     DPTR
        MOVX    @DPTR,A
        RET

SWAP_MAC:
        MOV     R0,#F0H     ; SWAP
ETH MAC ADDRESS
        MOV     R1,#02H
        MOV     R2,#E0H
        MOV     R3,#08H
        MOV     R6,#0
        MOV     R7,#6
        LCALL   MOVSTR
        MOV     R0,#F0H
        MOV     R1,#08H
        MOV     R2,#E0H
        MOV     R3,#02H
        MOV     R6,#0
        MOV     R7,#6
        LCALL   MOVSTR
        RET

SWAP_IP:
        MOV     R0,#F0H     ; SWAP IP
ADDRESS
        MOV     R1,#32
        MOV     R2,#E0H
        MOV     R3,#28
        MOV     R6,#0
        MOV     R7,#4
        LCALL   MOVSTR
        MOV     R0,#F0H
        MOV     R1,#28
        MOV     R2,#E0H
        MOV     R3,#32
        MOV     R6,#0
        MOV     R7,#4
        LCALL   MOVSTR
        RET
```

```
; IP Utility Module....
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY

        EXTERN CHECKSUM
        ;OUTPUT : R6 CHECKSUM H
        ;        R7 CHECKSUM L
        ;INPUT  : R0 STARTADD H
        ;        R1 STARTADD L
        ;        R2 BYTECOUNT H
        ;        R3 BYTECOUNT L

        PUBLIC  CAL_IP_LEN
        PUBLIC  IP_CHK
        PUBLIC  GEN_IP_CHK

OBUF    EQU     E000H
IBUF    EQU     F000H

BOH             EQU 16
; MEM
BOTCP   EQU 30H
TCPHEADLEN EQU 31H
IPHEADLEN EQU 32H
OUTLENH   EQU 39H
OUTLENL   EQU 3AH

IP0     EQU     3CH ; 3CH-3FH   ; SET
THE IP
IP1     EQU     3DH
IP2     EQU     3EH
IP3     EQU     3FH

CAL_IP_LEN:
; RETURN A FOR PROTOCOL TYPE
        MOV     DPTR,#IBUF
        MOV     DPL,#BOH        ;( skip
dma & nic header )
        MOVX    A,@DPTR
        ;GET HEADER LEN
        ANL     A,#0FH
        MOV     B,#4
        MUL     AB
        MOV     IPHEADLEN,A
        MOV     DPL,#BOH+16     ; GET
THE CURRENT IP
        MOVX    A,@DPTR
        MOV     IP0,A   ;1
        INC     DPTR
        MOVX    A,@DPTR
        MOV     IP1,A   ;2
        INC     DPTR
        MOVX    A,@DPTR
        MOV     IP2,A   ;3
        INC     DPTR
        MOVX    A,@DPTR
        MOV     IP3,A   ;4
        MOV     DPL,#BOH+9      ;
RETURN PROTOCOL TYPE
        MOVX    A,@DPTR
        RET

IP_CHK:
; A - 0 IF NO ERROR
        MOV     R0,#HIGH(IBUF)
        MOV     R1,#BOH
        MOV     R2,#0
        MOV     R3,IPHEADLEN
        LCALL   CHECKSUM
        MOV     A,R6
        ANL     A,R7
        RET

; CONST
IPCHKSUM EQU 10
GEN_IP_CHK:
        ; PUT NEW LEN TO IP & NIC
        CLR     C
        MOV     A,IPHEADLEN
        ADD     A,TCPHEADLEN
        ADD     A,OUTLENL
        MOV     R1,A
        MOV     A,OUTLENH
        ADDC    A,#0
        MOV     R0,A
        ; IP LEN (H)
        MOV     DPTR,#E000H+BOH+2
        MOVX    @DPTR,A
        MOV     A,R1
        ; IP LEN (L)
        INC     DPTR
        MOVX    @DPTR,A
        ; + 14 BYTE NIC HEAD
        MOV     A,R1
        ADD     A,#14
        MOV     R1,A
        MOV     A,R0
        ADDC    A,#0
        MOV     R0,A
        ; CHECK IF LESS THAN #60 BYTE
        MOV     A,#60
        SUBB    A,R1
        CLR     A
        SUBB    A,R0
        JB      A.7,GT
        MOV     R1,#60
        MOV     R0,#00
GT:
        ; NIC LEN (L)
        MOV     A,R1
```

```
        MOV     DPTR,#E000H
        MOVX    @DPTR,A
        MOV     A,R0
        ; NIC LEN (H)
        MOV     DPTR,#E001H
        MOVX    @DPTR,A
        ; SET TTL / CLEAR FLAGMENT /
CLEAR CHKSUM
        MOV     DPTR,#E000H+BOH+6
; CLR FLAGMENT
        CLR     A
        MOVX    @DPTR,A
        INC     DPTR
        MOVX    @DPTR,A
        INC     DPTR    ; SET TTL TO 30
        MOV     A,#30
        MOVX    @DPTR,A
        INC     DPTR
        INC     DPTR
        CLR     A       ; CLR CHKSUM
        MOVX    @DPTR,A
        INC     DPTR
        MOVX    @DPTR,A
        ;
        MOV     R0,#E0H
        MOV     R1,#BOH
        MOV     R2,#0
        MOV     R3,IPHEADLEN
        LCALL   CHECKSUM
        MOV     DPTR,#OBUF
        MOV     DPL,#BOH+IPCHKSUM
        MOV     A,R6
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,R7
        MOVX    @DPTR,A
        RET
```

```
; NETWORK INTERFACE CONTROLLER
HANDLE MODULE (NIC_RECV)
; FOR USING WITH NSC DP83902A ST-
NIC (SEE DP83902A DATASHEET FOR
MORE DETAIL
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY
; USING: MEM 21H

        EXTERN  C_IN
        EXTERN  C_OUT
        EXTERN  STROUT
        EXTERN  SHOWHEX
; UTIL FUNCTION
        EXTERN HEXSTR
        ; Convert HEX 2 ASCII
        ; INPUT  : A - Hex Value
        ; Return : R0 - First Byte
        ;         R1 - Second Byte

        PUBLIC RECV_PACKET

; Input Buffer
IBUFH   EQU     F0H
IBUFL   EQU     00H
IBUF    EQU     F000H

; CONSTANT CONFIG DATA
MASTER_ADD      EQU     0000H
SLAVE_ADD       EQU     1000H
DESMACPTR   EQU       2

; VAR
ISRBUF  EQU     21H
; CONST ISRBUF POINTER (BIT)
PRX     EQU     08H
PTX     EQU     09H
RXE     EQU     0AH
TXE     EQU     0BH
OVW     EQU     0CH
CNT     EQU     0DH
RDC     EQU     0EH
RST     EQU     0FH
; (BIT)
MACMATCH  EQU 1FH
ISRECV  EQU     1AH

;       DB      "NIC_RECV"
; RECIEVE PACKET
; MOVE PACKET FROM NIC TO BUFFER
RECV_PACKET:
        PUSH    DPH
        PUSH    DPL
        PUSH    A
        CLR     MACMATCH
        MOV     DPTR,#SLAVE_ADD ; TO
PAGE 0
        MOV     A,#22H
        MOVX    @DPTR,A
        ; BNRY -> RMSTRADR1 , 0 ->
RMSTRADR0
        MOV     DPL,#03H
        MOVX    A,@DPTR ; READ BNRY
        LCALL   SHOWHEX
        MOV     DPL,#09H
        MOVX.   @DPTR,A
        MOV     DPL,#08H
        CLR     A
        MOVX    @DPTR,A
        ; SET RBCR1 ,RBCR0
        MOV     P1,#01H
        MOV     DPL,#0BH        ; SET
RBCR1
        MOV     A,#00H
        MOVX    @DPTR,A
        MOV     DPL,#0AH        ; SET
RBCR0 (5BYTE HEADER)
        MOV     A,#05H
        MOV     R7,A
        MOVX    @DPTR,A
        MOV     DPL,#00H        ; START
        MOV     A,#0AH
        MOVX    @DPTR,A
        ; DUMMY READ HEADER 4 BYTE
        MOV     DPTR,#IBUF
DMRDLP:
        PUSH    DPH
        PUSH    DPL
        MOV     DPTR,#MASTER_ADD
        JNB     P1.0,$
        MOVX    A,@DPTR
        POP     DPL
        POP     DPH
        MOVX    @DPTR,A
        INC     DPTR
        LCALL   SHOWHEX
        DJNZ    R7,DMRDLP
DMYCHKDMA:
        MOV     DPTR,#SLAVE_ADD
        MOV     DPL,#07H
        MOVX    A,@DPTR
        MOV     ISRBUF,A
        JNB     RDC,DMYCHKDMA
        LCALL   STROUT
        DB      "END DMY
RD",0DH,0AH,00H
; CHECK FOR BOARDCAST
        MOV     DPTR,#IBUF
        MOVX    A,@DPTR
        CJNE    A,#1,MACNOTMATCH
```

```
        SETB    MACMATCH
MACNOTMATCH:
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R5,A    ; NXT PTR
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R7,A    ;REMOTE
BYTECOUNT L
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R6,A    ;REMOTE
BYTECOUNT H
        MOV     A,R7
        SUBB    A,#2
        MOV     R7,A
        MOV     A,R6
        SUBB    A,#0
        MOV     R6,A
        INC     R6
;
        MOV     DPTR,#SLAVE_ADD ; TO
PAGE 0
        MOV     A,#22H
        MOVX    @DPTR,A
        ; BNRY -> RMSTRADR1 , 2 ->
RMSTRADR0
        MOV     DPL,#03H
        MOVX    A,@DPTR ; READ BNRY
        MOV     DPL,#09H
        MOVX    @DPTR,A
        MOV     DPL,#08H
        MOV     A,#2
        MOVX    @DPTR,A
        ; SET RBCR1 ,RBCR0
        MOV     P1,#01H
        MOV     DPL,#0BH        ; SET
RBCR1
        MOV     A,R6
        MOVX    @DPTR,A
        MOV     DPL,#0AH        ; SET
RBCR0
        MOV     A,R7
        MOVX    @DPTR,A
        MOV     DPL,#00H        ; START
        MOV     A,#0AH
        MOVX    @DPTR,A
        MOV     DPTR,#IBUF
RECV_LOOP:
        PUSH    DPH
        PUSH    DPL
        JNB     P1.0,$  ; WAIT FOR PRQ
;
        MOV     DPTR,#MASTER_ADD
        MOVX    A,@DPTR
        ; WRITE TO BUFFER
        POP     DPL
        POP     DPH
        MOVX    @DPTR,A
        LCALL   SHOWHEX
        INC     DPTR
        DJNZ    R7,RECV_LOOP
        DJNZ    R6,RECV_LOOP
        MOV     A,#'-'
        LCALL   C_OUT
CHKDMA:
        MOV     DPTR,#SLAVE_ADD
        MOV     DPL,#07H
        MOVX    A,@DPTR
        MOV     ISRBUF,A
        LCALL   SHOWHEX
        JNB     RDC,CHKDMA      ; CHECK
IF REMOTE DMA IS DONE
        ; Remote Read Done
        MOV     DPTR,#SLAVE_ADD
        MOV     A,#22H
        MOVX    @DPTR,A
        ; NXTPTR -> BNRY
        MOV     DPL,#03H
        MOV     A,R5
        MOVX    @DPTR,A

            SETB    ISRECV  ; INFORM
APP THAT PACKET IS RECV
; (DEBUG SHOWOUT)
;       MOV     DPTR,#IBUF
;       MOVX    A,@DPTR
;       MOV     R6,A
;       INC     DPTR
;       MOVX    A,@DPTR
;       INC     A
;       MOV     R7,A
;DBGSHWLP:
;       INC     DPTR
;       MOVX    A,@DPTR
;       LCALL   SHOWHEX
;       DJNZ    R6,DBGSHWLP
;       DJNZ    R7,DBGSHWLP
;
;       MOV     A,#0AH
;       LCALL   C_OUT
;       MOV     A,#0DH
;       LCALL   C_OUT
;
        POP     A
        POP     DPL
        POP     DPH
        RET
```

```
; NETWORK INTERFACE CONTROLLER HANDLE MODULE (INIT_NIC)
; FOR USING WITH NSC DP83902A ST-NIC (SEE DP83902A DATASHEET FOR MORE DETAIL
; THESIS : DEVELOPMENT OF A RECONFIGURABLE EMBEDDED WEB SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY
; USING: MEM 21H
        EXTERN  C_IN
        EXTERN  C_OUT
        EXTERN  STROUT
; UTIL FUNCTION
        EXTERN HEXSTR
        ; Convert HEX 2 ASCII
        ; INPUT  : A - Hex Value
        ; Return : R0 - First Byte
        ;        R1 - Second Byte

        PUBLIC INIT_NIC

; Input Buffer
IBUFH   EQU     F0H
IBUFL   EQU     00H
; Output Buffer
OBUFH   EQU     E0H
OBUFL   EQU     00H
; INTERNAL BUFFER (IF CHANGE,MODIFY CONST ISRBUF POINTER BELOW IN MAIN &NIC_xxxx)
ISRBUF  EQU     21H
; CONST ISRBUF POINTER
PRX     EQU     08H
PTX     EQU     09H
RXE     EQU     0AH
TXE     EQU     0BH
OVW     EQU     0CH
CNT     EQU     0DH
RDC     EQU     0EH
RST     EQU     0FH

; CONSTANT CONFIG DATA
MASTER_ADD      EQU     0000H
SLAVE_ADD       EQU     1000H
; NIC MAC ADDRESS ( CHANGE HERE TO YOUR PREFER MAC ADDRESS )
PHYMAC0         EQU     00H
PHYMAC1         EQU     40H
PHYMAC2         EQU     05H
PHYMAC3         EQU     50H
PHYMAC4         EQU     4FH
PHYMAC5         EQU     4BH
; INTERNAL REGISTER
PSTART          EQU     01H             ; PAGE START (INIT <> 0)
PSTOP           EQU     03FH            ; 0000H-7F000 FOR SEND, 7F00H-FFFFH FOR RECV
DCR             EQU     00001000B       ; DATA CONFIGURATION REGISTER
TCR             EQU     00000000B       ; TRANSMIT CONFIGURATION REGISTER
LBTCR           EQU     00000010B       ; LOOPBACK MODE 1 TCR
RCR             EQU     00000000B       ; RECEIVE CONTROL REGISTER [00h] - normal
;                               +---------- Boardcast Recieve
IMR             EQU     00001011B       ; INTERRUPT MASK REGISTER (IN NIC_ISR TOO)
;
INIT_NIC:
        MOV     DPTR,#SLAVE_ADD
; PROGRAM CR FOR PAGE 0
        MOV     A,#21H
        MOVX    @DPTR,A
        MOV     DPL,#0EH                ; INIT DCR
        MOV     A,#DCR
        MOVX    @DPTR,A
        MOV     DPL,#0AH                ; CLEAR RBCR0 AND RBCR1
        CLR     A
        MOVX    @DPTR,A
        MOV     DPL,#0BH
        MOVX    @DPTR,A
        MOV     DPL,#0CH                ; INIT RCR
        MOV     A,#RCR
        MOVX    @DPTR,A
        MOV     DPL,#0DH                ; SET TO LOOPBACK 1 OR 2
        MOV     A,#LBTCR
        MOVX    @DPTR,A
        ; INIT BUFFER RING (PSTART & PSTOP & BNRY)
        MOV     DPL,#01H
        MOV     A,#PSTART
        MOVX    @DPTR,A
        MOV     DPL,#03H
; BNRY = PSTOP
        MOV     A,#PSTOP
        MOVX    @DPTR,A
        MOV     DPL,#02H
        MOV     A,#PSTOP
        MOVX    @DPTR,A
        MOV     DPL,#07H                ; CLEAR ISR
```

```
        MOV     A,#0FFH                                 RET
        MOVX    @DPTR,A
        MOV     DPL,#0FH            ;
INIT IMR
        MOV     A,#IMR
        MOVX    @DPTR,A
        MOV     DPTR,#SLAVE_ADD
; PROGRAM CR FOR PAGE 1
        MOV     A,#61H
        MOVX    @DPTR,A
        ; INIT PHYSICAL MAC ADDR ,
MCAST MAC ADDR , CUR POINTER
        MOV     DPL,#01H
        MOV     A,#PHYMAC0
        MOVX    @DPTR,A
        INC     DPL
        MOV     A,#PHYMAC1
        MOVX    @DPTR,A
        INC     DPL
        MOV     A,#PHYMAC2
        MOVX    @DPTR,A
        INC     DPL
        MOV     A,#PHYMAC3
        MOVX    @DPTR,A
        INC     DPL
        MOV     A,#PHYMAC4
        MOVX    @DPTR,A
        INC     DPL
        MOV     A,#PHYMAC5
        MOVX    @DPTR,A
        ; Set current Page Register to
PSTART
        INC     DPL
        MOV     A,#PSTART
        MOVX    @DPTR,A
;       MOV     DPL,#00H
;       MOV     A,#41
;       MOVX    @DPTR,A
;       INC     DPL
;       CLR     A
;       MOVX    @DPTR,A
        INC     DPL
        MOVX    @DPTR,A
        MOV     DPL,#00H        ; PUT
STNIC IN START MODE
        MOV     A,#22H
        MOVX    @DPTR,A
        MOV     DPL,#0DH            ;
INIT TCR
        MOV     A,#TCR
        MOVX    @DPTR,A
        ; READY
;
        LCALL   STROUT
        DB      "INIT_NIC
COMPLETE",0AH,0DH,00H
;
```

```
; TCP ACK Module
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY

        ; Debug Serial routine
        EXTERN  S_INIT
        EXTERN  C_IN
        EXTERN  C_OUT
        EXTERN  STROUT

              PUBLIC  TCP_CHK_ACK
; CONST
IBUFH   EQU F0H
OBUFH   EQU E0H
BOH             EQU 16 ; BEGIN OF IP
HEAD

BOTCP   EQU 30H
TCPHEADLEN EQU 31H
IPHEADLEN EQU 32H
;
ACK0    EQU 33H
ACK1    EQU 34H
ACK2    EQU 35H
ACK3    EQU 36H
;
DATALEN_H EQU 36H
DATALEN_L EQU 37H
; (BIT)
ACKFAIL EQU  1EH

TCP_CHK_ACK:
        CLR     ACKFAIL
        MOV     DPH,#IBUFH
        MOV     A,BOTCP
        ADD     A,#8       ; TO ACK FIELD
        MOV     DPL,A
        MOVX    A,@DPTR
        CJNE    A,ACK0,CHK_ACK_FAIL
        INC     DPTR
        MOVX    A,@DPTR
        CJNE    A,ACK1,CHK_ACK_FAIL
        INC     DPTR
        MOVX    A,@DPTR
        CJNE    A,ACK2,CHK_ACK_FAIL
        INC     DPTR
        MOVX    A,@DPTR
        CJNE    A,ACK3,CHK_ACK_FAIL
        SJMP TCP_CHK_END
CHK_ACK_FAIL:
        SETB    ACKFAIL
TCP_CHK_END:
        RET
```

```
; TCP CLOSE
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY

        ; Debug Serial routine
        EXTERN  C_IN
        EXTERN  C_OUT
        EXTERN  STROUT

        EXTERN CHECKSUM
        ;OUTPUT : R6 CHECKSUM H
        ;       R7 CHECKSUM L
        ;INPUT  : R0 STARTADD H
        ;       R1 STARTADD L
        ;       R2 BYTECOUNT H
        ;       R3 BYTECOUNT L

        EXTERN  MOVSTR
        ; Move Sequence of String
        ; INPUT  : R0 - Src H
        ;         R1 - Src L
        ;         R2 - Des H
        ;         R3 - Des L
        ;         R6 - Byte Count H
        ;         R7 - Byte Count L

        EXTERN  MOVBUF
        ; Move Sequence of String
        ; INPUT  : R0 - DES H
        ;        R1 - DES L
        EXTERN  SWAP_MAC
        EXTERN  SWAP_IP
        EXTERN  SWAP_PORT
        EXTERN  CREATE_ACK

        EXTERN  SEND_PACKET

                    ; CREATE ACK AT THE
TCP HEAD
        ; INPUT  : R6 - Byte Count H
                ;       R7 - Byte Count L
                ; USING  R2,R3,R4,R5 AS
TEMP
        EXTERN
CREATE_TCP_CHECKSUM
        ; CREATE TCP CHECKSUM
        ; INPUT : OUTLENH,OUTLENL
        EXTERN  GEN_IP_CHK
        EXTERN  CREATE_BLK_HEADER

      PUBLIC TCP_CLOSE

TCPSTAT EQU 23H ; (BYTE)
```

```
TCPINUSE EQU 18H ;(BIT)
TCP_CONN EQU 19H ; (BIT)
WFIN    EQU 1BH ; (BIT)
WACK    EQU 1CH ; (BIT)

IBUF    EQU F000H
OBUF    EQU E000H
BOH             EQU 16  ; BEGIN OF IP
HEAD
TCPFLAG EQU 22H  ;(BYTE)
FIN             EQU 10H ;(BIT)
SYN             EQU 11H
RST             EQU 12H
PSH             EQU 13H
ACK             EQU 14H
URG             EQU 15H
; MEM
BOTCP   EQU 30H
TCPHEADLEN EQU 31H
IPHEADLEN EQU 32H
ACK0    EQU 33H
ACK1    EQU 34H
ACK2    EQU 35H
ACK3    EQU 36H
DATALEN_H EQU 37H
DATALEN_L EQU 38H
OUTLENH   EQU 39H
OUTLENL   EQU 3AH
; EXPECTED NEXT INCOME SEQ
ESEQ0   EQU 2CH
ESEQ1   EQU 2DH
ESEQ2   EQU 2EH
ESEQ3   EQU 2FH
;
BLK_RSP_HEAD EQU        FFC0H

TCP_CLOSE:
        JB      TCP_CONN,ISCONN
        JNB     WFIN,ISCONN
        RET
ISCONN:
        LCALL   STROUT
        DB      "CLOSE
CONN",0AH,0DH,00H
        MOV     OUTLENH,#0
        MOV     OUTLENL,#0
        LCALL   CREATE_BLK_HEADER
        ; PUT THE OLD ESEQ TO ACK NO.
        ;MOV
DPTR,#OBUF+BOH+20+8
        ;MOV     A,ESEQ0
        ;MOVX    @DPTR,A
        ;INC     DPTR
        ;MOV     A,ESEQ1
        ;MOVX    @DPTR,A
        ;INC     DPTR
        ;MOV     A,ESEQ2
;       MOV     DPH,#HIGH(OBUF)
        ;MOVX    @DPTR,A
        ;INC     DPTR
        ;MOV     A,ESEQ3
        ;MOVX    @DPTR,A
        MOV     R6,#0
        MOV     R7,#0
        LCALL   CREATE_ACK
        ; PUT SEQ TO OBUF
        MOV     DPH,#HIGH(OBUF)
        MOV     A,BOTCP
        ADD     A,#4
        MOV     DPL,A
        MOV     A,ACK0
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ACK1
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ACK2
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ACK3
        MOVX    @DPTR,A
        ; EXPECTED RETURN ACK NO.
        CLR     C
        MOV     A,#1
        ADD     A,ACK3
        MOV     ACK3,A
        CLR     A
        ADDC    A,ACK2
        MOV     ACK2,A
        CLR     A
        ADDC    A,ACK1
        MOV     ACK1,A
        CLR     A
        ADDC    A,ACK0
        MOV     ACK0,A
        ; PUT NEW WIN SIZE
        MOV     A,BOTCP
        ADD     A,#14
        ;MOV     DPH,#E0H
        MOV     DPL,A
        MOV     A,#0EH
        MOVX    @DPTR,A
        CLR     A
        INC     DPTR
        MOVX    @DPTR,A
        ; SET TCP FLAG
        MOV     TCPFLAG,#0
        SETB    FIN
        SETB    ACK
        MOV     A,BOTCP
        ADD     A,#13
        MOV     DPL,A
        MOV     A,TCPFLAG
        MOVX    @DPTR,A
        ; CREATE NEW TCP CHECKSUM
```

```
        LCALL   CREATE_TCP_CHECKSUM
        LCALL   GEN_IP_CHK
        LCALL   SEND_PACKET
        SETB    WFIN
        SETB    WACK
;           SJMP    $
            RET
```

```
; TCP FIN
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY

        ; Debug Serial routine
        EXTERN  C_IN
        EXTERN  C_OUT
        EXTERN  STROUT
        EXTERN  HEXSTR

        EXTERN CHECKSUM
        ;OUTPUT : R6 CHECKSUM H
        ;        R7 CHECKSUM L
        ;INPUT  : R0 STARTADD H
        ;        R1 STARTADD L
        ;        R2 BYTECOUNT H
        ;        R3 BYTECOUNT L

        EXTERN  MOVSTR
        ; Move Sequence of String
        ; INPUT  : R0 - Src H
        ;          R1 - Src L
        ;          R2 - Des H
        ;          R3 - Des L
        ;          R6 - Byte Count H
        ;          R7 - Byte Count L

        EXTERN  MOVBUF
        ; Move Sequence of String
        ; INPUT  : R0 - DES H
        ;          R1 - DES L
        EXTERN  SWAP_MAC
        EXTERN  SWAP_IP
        EXTERN  SWAP_PORT
        EXTERN  SEND_PACKET
        EXTERN  CREATE_ACK
        ; CREATE ACK AT THE TCP HEAD
        ; INPUT  : R6 - Byte Count H
                ;        R7 - Byte Count L
                ; USING  R2,R3,R4,R5 AS
TEMP
        EXTERN
CREATE_TCP_CHECKSUM
        ; CREATE TCP CHECKSUM
        ; INPUT : OUTLENH,OUTLENL
        EXTERN  GEN_IP_CHK
        EXTERN  CREATE_BLK_HEADER

        PUBLIC TCP_FIN_RSP

TCPSTAT EQU 23H ; (BYTE)
TCPINUSE EQU 18H ;(BIT)
TCP_CONN EQU 19H ; (BIT)
```

```
WFIN    EQU 1BH ; (BIT)
WACK    EQU 1CH ; (BIT)

IBUF    EQU F000H
OBUF    EQU E000H
BOH             EQU 16  ; BEGIN OF IP
HEAD
TCPFLAG EQU 22H  ;(BYTE)
FIN             EQU 10H ;(BIT)
SYN             EQU 11H
RST             EQU 12H
PSH             EQU 13H
ACK             EQU 14H
URG             EQU 15H
; MEM
BOTCP   EQU 30H
TCPHEADLEN EQU 31H
IPHEADLEN EQU 32H
ACK0    EQU 33H
ACK1    EQU 34H
ACK2    EQU 35H
ACK3    EQU 36H
DATALEN_H EQU 37H
DATALEN_L EQU 38H
OUTLENH   EQU 39H
OUTLENL   EQU 3AH
; EXPECTED NEXT INCOME SEQ
ESEQ0   EQU 2CH
ESEQ1   EQU 2DH
ESEQ2   EQU 2EH
ESEQ3   EQU 2FH
;
BLK_RSP_HEAD EQU        FFC0H

TCP_FIN_RSP:
;               LCALL   STROUT
;               DB
"TCPFIN",0AH,0DH,00H
        MOV     OUTLENH,#0
        MOV     OUTLENL,#0
        LCALL   CREATE_BLK_HEADER
        MOV     R6,#0
        MOV     R7,#1
        LCALL   CREATE_ACK
        ; PUT SEQ TO OBUF
        MOV     DPH,#HIGH(OBUF)
        MOV     A,BOTCP
        ADD     A,#4
        MOV     DPL,A
        MOV     A,ACK0
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ACK1
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ACK2
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ACK3
        MOVX    @DPTR,A
        ; EXPECTED RETURN ACK NO.
        CLR     C
        MOV     A,#1
        ADD     A,ACK3
        MOV     ACK3,A
        CLR     A
        ADDC    A,ACK2
        MOV     ACK2,A
        CLR     A
        ADDC    A,ACK1
        MOV     ACK1,A
        CLR     A
        ADDC    A,ACK0
        MOV     ACK0,A
        ; PUT NEW WIN SIZE
        MOV     A,BOTCP
        ADD     A,#14
        ;MOV     DPH,#E0H
        MOV     DPL,A
        MOV     A,#0EH
        MOVX    @DPTR,A
        CLR     A
        INC     DPTR
        MOVX    @DPTR,A
        ; SET TCP FLAG
        MOV     TCPFLAG,#0
        JB      WFIN,NOWFIN
        SETB    FIN
NOWFIN:
        MOV     TCPSTAT,#0      ; CLOSE
CONN
        SETB    ACK
        MOV     A,BOTCP
        ADD     A,#13
        MOV     DPL,A
        MOV     A,TCPFLAG
        MOVX    @DPTR,A
        ; CREATE NEW TCP CHECKSUM
        LCALL   CREATE_TCP_CHECKSUM
        LCALL   GEN_IP_CHK
        LCALL   SEND_PACKET
;
        LCALL   STROUT
        DB      "TERM SESS
",0AH,0DH,00H
;
        RET
```

```
; TCP PUSH MODULE
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY

        ; Debug Serial routine
        EXTERN  C_IN
        EXTERN  C_OUT
        EXTERN  STROUT

        EXTERN CHECKSUM
        ;OUTPUT : R6 CHECKSUM H
        ;        R7 CHECKSUM L
        ;INPUT  : R0 STARTADD H
        ;        R1 STARTADD L
        ;        R2 BYTECOUNT H
        ;        R3 BYTECOUNT L

        EXTERN  MOVSTR
        ; Move Sequence of String
        ; INPUT  : R0 - Src H
        ;         R1 - Src L
        ;         R2 - Des H
        ;         R3 - Des L
                ;         R6 - Byte Count H
        ;         R7 - Byte Count L

                EXTERN  MOVBUF
        ; Move Sequence of String
        ; INPUT  : R0 - DES H
        ;         R1 - DES L
        EXTERN  SWAP_MAC
        EXTERN  SWAP_IP
        EXTERN  SWAP_PORT
        EXTERN  CREATE_ACK
        ; CREATE ACK AT THE TCP HEAD
        ; INPUT  : R6 - Byte Count H
                ;         R7 - Byte Count L
                ; USING  R2,R3,R4,R5 AS
TEMP
        EXTERN
CREATE_TCP_CHECKSUM
        ; CREATE TCP CHECKSUM
        ; INPUT : OUTLENH,OUTLENL
        EXTERN  GEN_IP_CHK
        EXTERN  CREATE_BLK_HEADER

        PUBLIC  TCP_PSH_RSP

INUSEFLAG EQU 23H
TCPINUSE EQU 18H

IBUF    EQU F000H
OBUF    EQU E000H

BOH         EQU 16  ; BEGIN OF IP
HEAD
TCPFLAG EQU 22H  ;(BYTE)
FIN         EQU 10H ;(BIT)
SYN         EQU 11H
RST         EQU 12H
PSH         EQU 13H
ACK         EQU 14H
URG         EQU 15H
; MEM
BOTCP   EQU 30H
TCPHEADLEN EQU 31H
IPHEADLEN EQU 32H
ACK0    EQU 33H
ACK1    EQU 34H
ACK2    EQU 35H
ACK3    EQU 36H
DATALEN_H EQU 37H
DATALEN_L EQU 38H
OUTLENH   EQU 39H
OUTLENL   EQU 3AH

; TCP SYN
TCP_PSH_RSP:
        LCALL   CREATE_BLK_HEADER
        ; CREATE ACK NO
        MOV     R6,DATALEN_H
        MOV     R7,DATALEN_L
        LCALL   CREATE_ACK
        ; PUT SEQ TO OBUF
        MOV     DPH,#E0H
        MOV     A,BOTCP
        ADD     A,#4
        MOV     DPL,A
        MOV     A,ACK0
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ACK1
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ACK2
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ACK3
        MOVX    @DPTR,A
        ; PUT NEW WIN SIZE
        MOV     A,BOTCP
        ADD     A,#14
        ;MOV     DPH,#E0H
        MOV     DPL,A
        MOV     A,#20H
;           MOV     A,#00H
        MOVX    @DPTR,A
        CLR     A
;           MOV     A,#100
        INC     DPTR
        MOVX    @DPTR,A
```

```
        ; SET TCP FLAG
        MOV     TCPFLAG,#0
        SETB    ACK
        MOV     A,BOTCP
        ADD     A,#13
        MOV     DPL,A
        MOV     A,TCPFLAG
        MOVX    @DPTR,A
        ; CREATE NEW TCP CHECKSUM ;
JUST ACK DO NOTHING
        MOV     OUTLENH,#0
        MOV     OUTLENL,#0
        LCALL   CREATE_TCP_CHECKSUM
        LCALL   GEN_IP_CHK
        RET
```

```
; TCP PUSH MODULE
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;    COMPUTER ENGINEER.
;    CHULALONGKORN UNIVERSITY

        ; Debug Serial routine
        EXTERN  C_IN
        EXTERN  C_OUT
        EXTERN  STROUT
        EXTERN  SHOWHEX

        EXTERN CHECKSUM
        ;OUTPUT : R6 CHECKSUM H
        ;        R7 CHECKSUM L
        ;INPUT  : R0 STARTADD H
        ;        R1 STARTADD L
        ;        R2 BYTECOUNT H
        ;        R3 BYTECOUNT L

        EXTERN  MOVSTR
        ; Move Sequence of String
        ; INPUT  : R0 - Src H
        ;          R1 - Src L
        ;          R2 - Des H
        ;          R3 - Des L
        ;          R6 - Byte Count H
        ;          R7 - Byte Count L

        EXTERN  MOVBUF
        ; Move Sequence of String
        ; INPUT  : R0 - DES H
        ;          R1 - DES L
        EXTERN  SWAP_MAC
        EXTERN  SWAP_IP
        EXTERN  SWAP_PORT
        EXTERN  CREATE_ACK

        EXTERN  SEND_PACKET

        ; CREATE ACK AT THE TCP HEAD
        ; INPUT  : R6 - Byte Count H
        ;          R7 - Byte Count L
        ; USING  R2,R3,R4,R5 AS TEMP
        EXTERN
CREATE_TCP_CHECKSUM
        ; CREATE TCP CHECKSUM
        ; INPUT : OUTLENH,OUTLENL
        EXTERN  GEN_IP_CHK
        EXTERN  CREATE_BLK_HEADER

        PUBLIC TCP_SEND
        ; INPUT : R0 - SRC DATA H
        ;          R1 - SRC DATA L
        ;          R6 - Byte Count H
```

```
;       R7 - Byte Count L

INUSEFLAG EQU 23H
TCPINUSE EQU 18H ;(BIT)
WACK    EQU 1CH ; (BIT)

IBUF    EQU F000H
OBUF    EQU E000H
BOH             EQU 16  ; BEGIN OF IP
HEAD
TCPFLAG EQU 22H  ;(BYTE)
FIN             EQU 10H ;(BIT)
SYN             EQU 11H
RST             EQU 12H
PSH             EQU 13H
ACK             EQU 14H
URG             EQU 15H
; MEM
BOTCP   EQU 30H
TCPHEADLEN EQU 31H
IPHEADLEN EQU 32H
ACK0    EQU 33H
ACK1    EQU 34H
ACK2    EQU 35H
ACK3    EQU 36H
DATALEN_H EQU 37H
DATALEN_L EQU 38H
OUTLENH   EQU 39H
OUTLENL   EQU 3AH
; EXPECTED NEXT INCOME SEQ
ESEQ0   EQU 2CH
ESEQ1   EQU 2DH
ESEQ2   EQU 2EH
ESEQ3   EQU 2FH
;
TMP     EQU 2AH
;
BLK_RSP_HEAD EQU        FFC0H
;

; SEND TCP DATA
TCP_SEND:
; INPUT : R0 - SRC DATA H
;       R1 - SRC DATA L
;       R6 - Byte Count H
;       R7 - Byte Count L
        MOV     OUTLENH,R6
        MOV     OUTLENL,R7
        MOV     R2,#HIGH(OBUF)
        MOV     R3,#LOW(OBUF)+56
        LCALL   MOVSTR
        LCALL   CREATE_BLK_HEADER
        ; PUT THE OLD ESEQ TO ACK NO.
        MOV     DPTR,#OBUF+BOH+20+8
        MOV     A,ESEQ0
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ESEQ1
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ESEQ2
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ESEQ3
        MOVX    @DPTR,A
        ; PUT SEQ TO OBUF
;           MOV     DPH,#HIGH(OBUF)
        MOV     A,BOTCP
        ADD     A,#4
        MOV     DPL,A
        MOV     A,ACK0
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ACK1
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ACK2
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ACK3
        MOVX    @DPTR,A
        ; EXPECTED RETURN ACK NO.
        CLR     C
        MOV     A,OUTLENL
        ADD     A,ACK3
        MOV     ACK3,A
        MOV     A,OUTLENH
        ADDC    A,ACK2
        MOV     ACK2,A
        CLR     A
        ADDC    A,ACK1
        MOV     ACK1,A
        CLR     A
        ADDC    A,ACK0
        MOV     ACK0,A
        ; PUT NEW WIN SIZE TO OUTLEN
        MOV     A,BOTCP
        ADD     A,#14
        ;MOV     DPH,#E0H
        MOV     DPL,A
        MOV     A,#22H
        MOVX    @DPTR,A
        MOV     A,#38H
        INC     DPTR
        MOVX    @DPTR,A
        ; SET TCP FLAG
        MOV     TCPFLAG,#0
        SETB    PSH
        SETB    ACK
        MOV     A,BOTCP
        ADD     A,#13
        MOV     DPL,A
        MOV     A,TCPFLAG
        MOVX    @DPTR,A
```

```
; CREATE NEW TCP CHECKSUM
LCALL   CREATE_TCP_CHECKSUM
LCALL   GEN_IP_CHK
LCALL   SEND_PACKET
SETB    WACK
RET
```

```
; TCP Synch MODULE
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;    COMPUTER ENGINEER.
;    CHULALONGKORN UNIVERSITY

        ; Debug Serial routine
        EXTERN  S_INIT
        EXTERN  C_IN
        EXTERN  C_OUT
        EXTERN  STROUT

        EXTERN CHECKSUM
        ;OUTPUT : R6 CHECKSUM H
        ;        R7 CHECKSUM L
        ;INPUT  : R0 STARTADD H
        ;        R1 STARTADD L
        ;        R2 BYTECOUNT H
        ;        R3 BYTECOUNT L

        EXTERN  MOVSTR
        ; Move Sequence of String
        ; INPUT  : R0 - Src H
        ;          R1 - Src L
        ;          R2 - Des H
        ;          R3 - Des L
        ;          R6 - Byte Count H
        ;          R7 - Byte Count L

                EXTERN  MOVBUF
        ; Move Sequence of String
        ; INPUT  : R0 - DES H
        ;          R1 - DES L
                EXTERN  SWAP_MAC
                EXTERN  SWAP_IP
                EXTERN  SWAP_PORT
                EXTERN  CREATE_ACK
                EXTERN
CREATE_TCP_CHECKSUM
        ; CREATE TCP CHECKSUM
        ; INPUT : OUTLENH,OUTLENL

        PUBLIC  TCP_SYN_RSP

TCPFLAG EQU 22H
FIN         EQU 10H
SYN         EQU 11H
RST         EQU 12H
PSH         EQU 13H
ACK         EQU 14H
URG     EQU 15H

INUSEFLAG EQU 23H
TCPINUSE EQU 18H
```

```
IBUF    EQU F000H
OBUF    EQU E000H
BLK_RSP_HEAD EQU        FFC0H
BOH             EQU 16  ; BEGIN OF IP
HEAD
; MEM
BOTCP   EQU 30H
TCPHEADLEN EQU 31H
IPHEADLEN EQU 32H
ACK0    EQU 33H
ACK1    EQU 34H
ACK2    EQU 35H
ACK3    EQU 36H
;
OUTLENH   EQU 39H
OUTLENL   EQU 3AH

; TCP SYN
TCP_SYN_RSP:
        ; GET ALL IBUF TO OBUF
        MOV             DPTR,#IBUF
        MOVX    A,@DPTR             ;
GET THE LENGTH - 4B NIC FCS
        CLR     C
        SUBB    A,#4
        MOV             R7,A
        PUSH    A                   ;
SAVE THE LENGTH L
        INC             DPTR
        MOVX    A,@DPTR
        SUBB    A,#0
        MOV             R6,A
        PUSH    A                   ;
SAVE THE LENGTH H
        MOV             R0,#F0H
; GET ALL DATA (SKIP 2 HEADER)
        MOV             R1,#02H
        MOV             R2,#E0H
        MOV             R3,#02H
        LCALL   MOVSTR
        LCALL   SWAP_MAC
        LCALL   SWAP_IP
        LCALL   SWAP_PORT
        ; CREATE ACK NO ( old ack +1)
        MOV     R6,#0
        MOV     R7,#1
        LCALL   CREATE_ACK
        ; EXPECTED RETURN ACK NO.
        MOV     ACK0,#00H
        MOV     ACK1,#00H
        MOV     ACK2,#00H
        MOV     ACK3,#02H
        ; PUT SEQ TO OBUF
        MOV     DPH,#E0H
        MOV     A,BOTCP
        ADD     A,#4
        MOV     DPL,A
        CLR     A
        MOVX    @DPTR,A
        INC     DPTR
        CLR     A
        MOVX    @DPTR,A
        INC     DPTR
        CLR     A
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#01H
        MOVX    @DPTR,A
        ; PUT NEW WIN SIZE
        MOV     A,BOTCP
        ADD     A,#14
        ;MOV     DPH,#E0H
        MOV     DPL,A
        MOV     A,#20H
        MOV     A,#00H
        MOVX    @DPTR,A
        CLR     A
;           MOV     A,#100
        INC     DPTR
        MOVX    @DPTR,A
        ; SET TCP & INUSED FLAG
        SETB    ACK
        MOV     A,BOTCP
        ADD     A,#13
        MOV     DPL,A
        MOV     A,TCPFLAG
        MOVX    @DPTR,A
        SETB    TCPINUSE
        ; CREATE NEW TCP CHECKSUM
        MOV     OUTLENH,#0
        MOV     OUTLENL,#0
        LCALL   CREATE_TCP_CHECKSUM
        ; PUT NEW LEN
        POP     A
        MOV     DPTR,#E001H
        MOVX    @DPTR,A
        POP     A
        MOV     DPTR,#E000H
        MOVX    @DPTR,A
        ; SAVE PACKET HEADER FOR TCP-
SEND
        ; ETH HEAD & IPHEAD
        MOV     R0,#E0H
        MOV     R1,#02H
        MOV     R2,#HIGH
(BLK_RSP_HEAD)
        MOV     R3,#LOW
(BLK_RSP_HEAD)
        MOV     R6,#0
        MOV     R7,#34
        LCALL   MOVSTR
        ; TCP HEAD
        MOV     R0,#E0H
        MOV     R1,BOTCP
```

```
        MOV     R2,#HIGH
(BLK_RSP_HEAD)
        MOV     R3,#LOW
(BLK_RSP_HEAD)+34
        MOV     R6,#0
        MOV     R7,#20
        LCALL   MOVSTR
        ; SET TCP HEAD LEN
        MOV
DPTR,#BLK_RSP_HEAD+46
        MOV     A,#50H
        MOVX    @DPTR,A
        LCALL   STROUT
        DB      "SAVE
HEAD",0AH,0DH,00H
                RET
```

```
; NETWORK INTERFACE CONTROLLER
HANDLE MODULE (NIC_RECV)
; FOR USING WITH NSC DP83902A ST-
NIC (SEE DP83902A DATASHEET FOR
MORE DETAIL)
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY
; USING: MEM 21H

        EXTERN  C_IN
        EXTERN  C_OUT
        EXTERN  STROUT
        EXTERN  SHOWHEX
; UTIL FUNCTION
        EXTERN HEXSTR
        ; Convert HEX 2 ASCII
        ; INPUT  : A - Hex Value
        ; Return : R0 - First Byte
        ;        R1 - Second Byte
        EXTERN CHK_RING

        PUBLIC RECV_PACKET

; Input Buffer
IBUFH   EQU     F0H
IBUFL   EQU     00H
IBUF    EQU     F000H

; CONSTANT CONFIG DATA
MASTER_ADD      EQU     0000H
SLAVE_ADD       EQU     1000H
DESMACPTR       EQU     2
PSTART          EQU     01H          ; PAGE
START (INIT <> 0)
PSTOP           EQU     03FH         ;
0000H-7F000 FOR SEND, 7F00H-FFFFH
FOR RECV

; VAR
ISRBUF  EQU     21H
; CONST ISRBUF POINTER (BIT)
PRX     EQU     08H
PTX     EQU     09H
RXE     EQU     0AH
TXE     EQU     0BH
OVW     EQU     0CH
CNT     EQU     0DH
RDC     EQU     0EH
RST     EQU     0FH
; (BIT)
MACMATCH  EQU 1FH
ISRECV  EQU     1AH
;
```

```
TMP     EQU 2AH

;       DB      "NIC_RECV"
; RECIEVE PACKET
; MOVE PACKET FROM NIC TO BUFFER
RECV_PACKET:
        PUSH    DPH
        PUSH    DPL
        PUSH    A
;       CLR     EX0
        MOV     DPTR,#SLAVE_ADD+7
        MOV     A,#1
        MOVX    @DPTR,A
        CLR     MACMATCH
        MOV     DPTR,#SLAVE_ADD ; TO
PAGE
        MOV     A,#62H
        MOVX    @DPTR,A
        MOV     DPL,#07H
        MOVX    A,@DPTR
        MOV     TMP,A           ; CURR
PAGE
        MOV     DPL,#00H
        MOV     A,#22H
        MOVX    @DPTR,A
        ; BNRY -> RMSTRADR1 , 0 ->
RMSTRADR0
        MOV     DPL,#03H
        MOVX    A,@DPTR ; READ BNRY
        INC     A
        CJNE
A,#PSTOP+1,DMRDEQPSTOP
        MOV     A,#PSTART
DMRDEQPSTOP:
        MOV     R4,A
        CJNE    A,TMP,CONN
;       LCALL   STROUT
;       DB      "END RX",0DH,0AH,00H
        LJMP    ENDRX
CONN:

;       LCALL   SHOWHEX
        MOV     DPL,#09H
        MOVX    @DPTR,A
        MOV     DPL,#08H
        CLR     A
        MOVX    @DPTR,A
        ; SET RBCR1 ,RBCR0
        MOV     P1,#01H
        MOV     DPL,#0BH        ; SET
RBCR1
        MOV     A,#00H
        MOVX    @DPTR,A
        MOV     DPL,#0AH        ; SET
RBCR0 (5BYTE HEADER)
        MOV     A,#05H
        MOV     R7,A
        MOVX    @DPTR,A
;
        MOV     DPTR,#MASTER_ADD
        MOVX    A,@DPTR
;
        MOV     DPTR,#SLAVE_ADD     ;
START
        MOV     A,#0AH
        MOVX    @DPTR,A
        ; DUMMY READ HEADER 4 BYTE
        MOV     DPTR,#IBUF
DMRDLP:
        PUSH    DPH
        PUSH    DPL
        MOV     DPTR,#MASTER_ADD
        JNB     P1.0,$
        MOVX    A,@DPTR
        POP     DPL
        POP     DPH
        MOVX    @DPTR,A
        INC     DPTR
;       LCALL   SHOWHEX
        DJNZ    R7,DMRDLP
DMYCHKDMA:
        MOV     DPTR,#SLAVE_ADD
        MOV     DPL,#07H
        MOVX    A,@DPTR
        MOV     ISRBUF,A
        JNB     RDC,DMYCHKDMA
;       LCALL   STROUT
;       DB      "END DMY
RD",0DH,0AH,00H
; CHECK FOR BOARDCAST
        MOV     DPTR,#IBUF
        MOVX    A,@DPTR
        CJNE    A,#1,MACNOTMATCH
        SETB    MACMATCH
MACNOTMATCH:
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R5,A    ; NXT PTR
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R7,A    ;REMOTE
BYTECOUNT L
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R6,A    ;REMOTE
BYTECOUNT H
        MOV     A,R7
        ADD     A,#2
        MOV     R7,A
        MOV     A,R6
        ADDC    A,#0
        MOV     R6,A
        INC     R6
```

```
        MOV     DPTR,#SLAVE_ADD ; TO
PAGE 0
        MOV     A,#22H
        MOVX    @DPTR,A
        ; BNRY -> RMSTRADR1 , 2 ->
RMSTRADR0
        MOV     A,R4
        MOV     DPL,#09H
        MOVX    @DPTR,A
        MOV     DPL,#08H
        MOV     A,#2
        MOVX    @DPTR,A
        ; SET RBCR1 ,RBCR0
        MOV     P1,#01H
        MOV     DPL,#0BH        ; SET
RBCR1
        MOV     A,R6
        MOVX    @DPTR,A
        MOV     DPL,#0AH        ; SET
RBCR0
        MOV     A,R7
        MOVX    @DPTR,A
        MOV     DPL,#00H        ; START
        MOV     A,#0AH
        MOVX    @DPTR,A
        MOV     DPTR,#IBUF
RECV_LOOP:
        PUSH    DPH
        PUSH    DPL
        JNB     P1.0,$  ; WAIT FOR PRQ

                MOV
DPTR,#MASTER_ADD
        MOVX    A,@DPTR
        ; WRITE TO BUFFER
        POP         DPL
        POP         DPH
        MOVX    @DPTR,A
;       LCALL   SHOWHEX
        INC     DPTR
        DJNZ    R7,RECV_LOOP
        DJNZ    R6,RECV_LOOP
        MOV     A,#'-'
        LCALL   C_OUT
CHKDMA:
        MOV     DPTR,#SLAVE_ADD
        MOV     DPL,#07H
        MOVX    A,@DPTR
        MOV     ISRBUF,A
;       LCALL   SHOWHEX
        JNB     RDC,CHKDMA      ; CHECK
IF REMOTE DMA IS DONE
        ; Remote Read Done
        MOV     DPTR,#SLAVE_ADD
        MOV     A,#22H
        MOVX    @DPTR,A
        ; NXTPTR -> BNRY
        MOV     DPL,#03H
        MOV     A,R5
        DEC     A
;       CJNE    A,TMP,NOTEQCURR
;       LCALL   SHOWHEX
;NOTEQCURR:
        MOVX    @DPTR,A

                SETB    ISRECV  ; INFORM
APP THAT PACKET IS RECV
ENDRX:
; (DEBUG SHOWOUT)
;       MOV     DPTR,#IBUF
;       MOVX    A,@DPTR
;       MOV     R6,A
;       INC     DPTR
;       MOVX    A,@DPTR
;       INC     A
;       MOV     R7,A
;DBGSHWLP:
;       INC     DPTR
;       MOVX    A,@DPTR
;       LCALL   SHOWHEX
;       DJNZ    R6,DBGSHWLP
;       DJNZ    R7,DBGSHWLP
;
;       MOV     A,#0AH
;       LCALL   C_OUT
;       MOV     A,#0DH
;       LCALL   C_OUT
;
;       SETB    EX0
        POP     A
        POP     DPL
        POP     DPH
        RET

; ARP Module....
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY

        EXTERN  STROUT
        EXTERN  SHOWHEX

        EXTERN  SEND_PACKET

        EXTERN  SWAP_MAC

        EXTERN  MOVSTR
        ; Move Sequence of String
        ; INPUT  : R0 - Src H
        ;         R1 - Src L
        ;         R2 - Des H
```

```
        ;       R3 - Des L
        ;       R6 - Byte Count H
        ;       R7 - Byte Count L

        PUBLIC  ARP_RSP

BOH     EQU 16
IBUF    EQU F000H
; NIC MAC ADDRESS ( CHANGE HERE TO
YOUR PREFER MAC ADDRESS )
PHYMAC0         EQU     00H
PHYMAC1         EQU     40H
PHYMAC2         EQU     05H
PHYMAC3         EQU     50H
PHYMAC4         EQU     4FH
PHYMAC5         EQU     4BH
; MEM
IP0     EQU 3CH
IP1     EQU 3DH
IP2     EQU 3EH
IP3     EQU 3FH

NOT_ARP:
        LCALL   STROUT
        DB      "NOT ARP",0DH,0AH,00H
        RET
NOT_IP:
        LCALL   STROUT
        DB      "NOT MYIP",0DH,0AH,00H
        RET
ARP_RSP:
        ; CHECK IF ARP OR NOT (FRAME
TYPE = 0806H)
        MOV     DPTR,#IBUF+14
        MOVX    A,@DPTR
        CJNE    A,#08H,NOT_ARP
        INC     DPTR
        MOVX    A,@DPTR
        CJNE    A,#06H,NOT_ARP
        ; CHECK IF MATCH IP THEN
RESPONSE
        MOV     DPL,#BOH+24
        MOV     A,IP0
        LCALL   SHOWHEX
        MOVX    A,@DPTR
        LCALL   SHOWHEX
        CJNE    A,IP0,NOT_IP
        INC     DPTR
        MOV     A,IP1
        LCALL   SHOWHEX
        MOVX    A,@DPTR
        LCALL   SHOWHEX
        CJNE    A,IP1,NOT_IP
        INC     DPTR
        MOV     A,IP2
        LCALL   SHOWHEX
        MOVX    A,@DPTR
        LCALL   SHOWHEX
        CJNE    A,IP2,NOT_IP
        INC     DPTR
        MOV     A,IP3
        LCALL   SHOWHEX
        MOVX    A,@DPTR
        LCALL   SHOWHEX
        CJNE    A,IP3,NOT_IP
        ;
        LCALL   STROUT
        DB      "ARP RECV",0DH,0AH,00H
        MOV     R0,#F0H
        MOV     R1,#02H
        MOV     R2,#E0H
        MOV     R3,#02H
        MOV     R6,#00
        MOV     R7,#64
        LCALL   MOVSTR
        MOV     R0,#F0H
        MOV     R1,#BOH+8
        MOV     R2,#E0H
        MOV     R3,#BOH+18
        MOV     R6,#00
        MOV     R7,#10
        LCALL   MOVSTR  ; TARGET ->
SENDERR
        MOV     R0,#F0H
        MOV     R1,#BOH+18
        MOV     R2,#E0H
        MOV     R3,#BOH+8
        MOV     R6,#00
        MOV     R7,#10
        LCALL   MOVSTR  ; SENDER ->
TARGET
        MOV     R0,#F0H
        MOV     R1,#8
        MOV     R2,#E0H
        MOV     R3,#2
        MOV     R6,#00H
        MOV     R7,#06
        LCALL   MOVSTR  ; GEN MAC
        MOV     DPH,#E0H
        MOV     DPL,#8
        MOV     A,#PHYMAC0
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#PHYMAC1
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#PHYMAC2
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#PHYMAC3
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#PHYMAC4
        MOVX    @DPTR,A
```

```
INC     DPTR
MOV     A,#PHYMAC5
;
MOVX    @DPTR,A
MOV     R0,#E0H
MOV     R1,#8
MOV     R2,#E0H
MOV     R3,#BOH+8
MOV     R6,#00H
MOV     R7,#06
LCALL   MOVSTR  ; GEN MAC
MOV     DPTR,#E000H
MOV     A,#BOH+28
MOVX    @DPTR,A
CLR     A
MOVX    @DPTR,A
MOV     DPL,#BOH+6
MOVX    @DPTR,A
MOV     A,#02
INC     DPTR
MOVX    @DPTR,A
; GEN LEN
MOV     DPTR,#E000H
MOV     A,#64
MOVX    @DPTR,A
INC     DPTR
CLR     A
MOVX    @DPTR,A
LCALL   SEND_PACKET
RET
```

```
; HTTP INPUT
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY
        EXTERNSTROUT
        EXTERN  SHOWHEX
        EXTERNSTR_CX_COMP
        EXTERNSTR_FIND_CHR

        EXTERN  MOVSTR
        ; Move Sequence of String
        ; INPUT  : R0 - Src H
        ;         R1 - Src L
        ;         R2 - Des H
        ;         R3 - Des L
        ;         R6 - Byte Count H
        ;         R7 - Byte Count L

        EXTERN  TCP_SEND
        ; INPUT : R0 - SRC DATA H
        ;       R1 - SRC DATA L
        ;       R6 - Byte Count H
        ;       R7 - Byte Count L

        EXTERN TCP_CLOSE

        EXTERN  AUTH_CHECK
        EXTERN  HTTP_HEAD_SHOW
        EXTERN  HTTP_URL

IBUF    EQU F000H
BOH     EQU 16  ; BEGIN OF IP HEAD
; MEM
BOTCP   EQU 30H
TCPHEADLEN EQU 31H
IPHEADLEN EQU 32H
DATALEN_H EQU 37H
DATALEN_L EQU 38H
;
MAXBLOCK EQU 200
;
CURRH   EQU     DFFEH
CURRL   EQU     DFFFH
CNTH    EQU     DFFCH
CNTL    EQU     DFFDH

        PUBLIC  HTTP_INIT
        PUBLIC  HTTP_IN
        PUBLIC  CHK_HTTP_STAT
        PUBLIC  HTTP_SEND

; RECV BUFF
HTTP_IBUF     EQU  D000H
; RECV BUF PTR  OFFSET
```

```
CUR_DPH EQU     00H
CUR_DPL EQU     01H
; HTTP STAT / FLAG
HTTP_STAT       EQU     24H
; BIT ADDRESSIBLE
HTTPSTART       EQU     20H
HTTPIN          EQU     21H
HTTPSEND        EQU     22H
HTTPCLOSE       EQU     23H

HTTP_INIT:
        MOV     HTTP_STAT,#0 ; CLEAR
STAT
        SETB    HTTPSTART       ; NOW
START
        MOV     A,#D0H
        MOV     DPTR,#HTTP_IBUF
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#02H
        MOVX    @DPTR,A
        RET

HTTP_IN:
        SETB    HTTPIN
        MOV     DPTR,#HTTP_IBUF         ;
MOV ALL DATA TO HTTP_IBUF
        MOVX    A,@DPTR
        MOV     R2,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R3,A
        MOV     A,BOTCP
        ADD     A,TCPHEADLEN
        MOV     R0,#F0H
        MOV     R1,A
        MOV     R6,DATALEN_H
        MOV     R7,DATALEN_L
        LCALL   MOVSTR
        MOV     A,R2
        MOV     DPH,A
        MOV     A,R3
        MOV     DPL,A
        CLR     A
        MOVX    @DPTR,A         ;
TERMINATE THE STRING
        MOV     DPTR,#HTTP_IBUF
        MOV     A,R2
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,R3
        MOVX    @DPTR,A
        RET

CHK_HTTP_STAT:
        JB      HTTPIN,ISHTTPIN
        ; JMP IF DATA ALREADY IN
;       LCALL   STROUT
;       DB      "HTTP:NOTIN",0AH,0DH,00H
        RET     ; WAIT FOR DATA TO
INPUT
ISHTTPIN:
        JNB     HTTPCLOSE,ISNCLOSE  ;
JMP IF NOT CLOSE
;       LCALL   STROUT
;       DB
"HTTP:WCLOSE",0AH,0DH,00H
        RET     ; ALREADY CLOSE WAIT
FOR NEXT SESSION
ISNCLOSE:
        JNB     HTTPSEND,ISNSEND    ;
JMP IF NOT SENDING YET
        LCALL   TCP_CLOSE       ; TIME TO
CLOSE CONN
;       LCALL   STROUT
;       DB      "HTTP:CLOSE",0AH,0DH,00H
        SETB    HTTPCLOSE
        RET
ISNSEND:
;       LCALL   STROUT
;       DB
"HTTP:ISNSEND",0AH,0DH,00H
        MOV     DPTR,#HTTP_IBUF+1
        MOVX    A,@DPTR
        CLR     C
        SUBB    A,#4    ; GO BACK 4BYTE
TO CHECK CRLFCRLF(END OF HTTP)
        MOV     R2,A
        MOV     DPL,#00H
        MOVX    A,@DPTR
        SUBB    A,#0
        MOV     R1,A
        LCALL   STR_CX_COMP
        DB      0DH,0AH,0DH,0AH,00H
        CJNE    R0,#0,ISNDONE  ; JMP IF
NOT DONE
        LCALL   STROUT
        DB      "HTTP-RSP",0AH,0DH,00H
;       LCALL   AUTH_CHECK
        LCALL   HTTP_URL
;       MOV     R0,#HIGH
(RESPONSE_TEST)
;       MOV     R1,#LOW
(RESPONSE_TEST)
;       MOV     R6,#0
;       MOV     R7,#239
;       MOV     R6,#HIGH(281)
;       MOV     R7,#LOW(281)
;       LCALL   HTTP_SEND
        SETB    HTTPSEND

ISNDONE:
;       LCALL   STROUT
;       DB      "HTTP:NEND",0AH,0DH,00H
```

```
        RET

HTTP_SEND:
; CHECK FOR ODD
        MOV     A,R7
        JB      A.0,ISODD
        SJMP    CONT
ISODD:
        ADD     A,#1
        MOV     R7,A
        MOV     A,R6
        ADDC    A,#0
        MOV     R6,A
CONT:
        MOV     DPTR,#CNTH
        MOV     A,R6
        MOVX    @DPTR,A
        MOV     A,R7
        MOV     DPTR,#CNTL
        MOVX    @DPTR,A
        MOV     DPTR,#CURRH
        MOV     A,R0
        MOVX    @DPTR,A
        MOV     DPTR,#CURRL
        MOV     A,R1
        MOVX    @DPTR,A
; START SEND LOOP
SENDLOOP:
        MOV     DPTR,#CURRH
        MOVX    A,@DPTR
        MOV     R0,A
        MOV     DPTR,#CURRL
        MOVX    A,@DPTR
        MOV     R1,A
        MOV     DPTR,#CNTL
        MOVX    A,@DPTR
        CLR     C
        SUBB    A,#MAXBLOCK
        MOV     R5,A
        MOV     DPTR,#CNTH
        MOVX    A,@DPTR
        SUBB    A,#0
        MOV     R4,A
        JNB     A.7,SENDGT
        MOV     DPTR,#CNTH
        MOVX    A,@DPTR
        MOV     R6,A
        MOV     DPTR,#CNTL
        MOVX    A,@DPTR
        MOV     R7,A
        SJMP    TOSEND
SENDGT:
        MOV     R6,#0
        MOV     R7,#MAXBLOCK
TOSEND:
; SET NEXT POINTER & COUNTER
        MOV     DPTR,#CURRL
        MOVX    A,@DPTR
        ADD     A,R7
        MOVX    @DPTR,A
        MOV     DPTR,#CURRH
        MOVX    A,@DPTR
        ADDC    A,R6
        MOVX    @DPTR,A
        MOV     DPTR,#CNTH
        MOV     A,R4
        MOVX    @DPTR,A
        MOV     DPTR,#CNTL
        MOV     A,R5
        MOVX    @DPTR,A
        LCALL   TCP_SEND
;       LCALL   STROUT
;       DB      "HTTP SEND
OUT",0DH,0AH,00H
; CHECK IF CNT >0
        MOV     DPTR,#CNTL
        MOVX    A,@DPTR
        MOV     R5,A
        MOV     DPTR,#CNTH
        MOVX    A,@DPTR
        CJNE    R5,#0,CHKEND
        CJNE    A,#0,CHKEND
        SJMP    ENDHTTPSEND
CHKEND:
        JNB     A.7,SENDLOOP
ENDHTTPSEND:
        RET
```

```
; Main Module....
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY

;       EXTERN  DEBUGMAIN
; Debug Serial routine
        EXTERN  S_INIT
        EXTERN  C_IN
        EXTERN  C_OUT
        EXTERN  STROUT
        EXTERN  Ser_isr

; NIC HANDLER ROUTINE
        EXTERN INIT_NIC
        EXTERN SEND_PACKET
        EXTERN RECV_PACKET
        EXTERN NIC_ISR

; ICMP HANDLER
        EXTERN  CAL_IP_LEN
        ;OUTPUT : A PROTOCOL TYPE
        EXTERN  ICMP_RSP
; ARP HANDLER
        EXTERN  ARP_RSP

; TCP HANDLER
        EXTERN  TCP_RSP
        EXTERN  TCP_CLOSE

; UTIL FUNCTION
        EXTERN HEXSTR
        ; Convert HEX 2 ASCII
        ; INPUT  : A - Hex Value
        ; Return : R0 - First Byte
        ;        R1 - Second Byte

; HTTP_UTIL
        EXTERN  CHK_HTTP_STAT

        EXTERN INIT_TIMER
        EXTERN ISR_TIMER

; STATUS (BYTE)
STAT    EQU     23H
HTTPSTAT EQU 24H
; (BIT)
ISRECV  EQU     1AH
MACMATCH  EQU 1FH
;
; Start of Main Program
        ORG   0000H
        LJMP  MAIN

;=======Interrupt Vector
Table==========
; External Interrupt 0 (INT0)
        ORG   0003H
        LJMP  NIC_ISR
        RETI
; Timer 0 Overflow (T0)
        ORG   000BH
        LJMP    ISR_TIMER
        RETI
; External Interrupt 1 (INT1)
        ORG   0013H
;   LJMP  NIC_ISR
        LJMP  8013H
        RETI
; Timer 1 Overflow (T1)
        ORG   001BH
        LJMP  801BH
        RETI
; Serial Interrupt
        ORG   0023H
        LJMP  Ser_isr

;=========================
==============
MAIN:
        ; SWITCH TO BLANK 0
        CLR   RS0
        CLR   RS1
        MOV   SP,#50H
        ; CLR STATUS BYTE
        MOV   STAT,#0H
        MOV     HTTPSTAT,#0H
        ;LCALL S_INIT
        LCALL   STROUT
        DB 0AH,0DH,"Thesis :
Development of A Reconfigurable
Embedded Web Server",0AH,0D
        DB    "    By : Krerk Piromsopa
(Computer Engineer Chulalongkorn
University)"
        DB      0AH,0DH,00H
        LCALL INIT_NIC
        SETB EX0
        LCALL   INIT_TIMER
        SETB  EA

MAIN_LOOP:
        JNB     ISRECV,NOPACKET     ;
CHECK IF PACKET RECV
        CLR     EX0
ISPACKET:
        CLR     ISRECV
        JB      MACMATCH,NOT_BRDCAST
        LCALL   STROUT
        DB      "BCAST
Recv.",0Ah,0Dh,00h
```

```
        LCALL   ARP_RSP
        SJMP    END_CHK
NOT_BRDCAST:
        LCALL   STROUT
        DB      "Recv.",0Ah,0Dh,00h
        LCALL   CAL_IP_LEN   ; CAL IP
HEADER LEN FOR OTHER USE
                        ; [A] CONTENT IP
PROTOCOL TYPE
        CJNE    A,#01,NOT_ICMP
        LCALL   ICMP_RSP            ; ICMP
HANDLER
        SJMP    END_CHK
NOT_ICMP:
        CJNE    A,#06,NOT_TCP
        LCALL   STROUT
        DB      "TCP",0DH,0AH,00H
;   LCALL   WATCHDOG
        LCALL   TCP_RSP             ; TCP
HANDLER
NOT_TCP:
END_CHK:
        LCALL    RECV_PACKET
        JB      ISRECV,ISPACKET     ;
RING IS NOT EMPTY
        SETB     EX0
NOPACKET:
        JNB      19H,NOCONN
        JB       1CH,ISWAIT
        JB       1BH,ISWAIT
        ; Do Server Job Here....
        CLR      EX0
        LCALL    CHK_HTTP_STAT
        SETB     EX0
NOCONN:
        ; Do Another Job Here....
ISWAIT:
        LJMP    MAIN_LOOP
        END
```

```
; NETWORK INTERFACE CONTROLLER
HANDLE MODULE (NIC_ISR)
; FOR USING WITH NSC DP83902A ST-
NIC (SEE DP83902A DATASHEET FOR
MORE DETAIL)
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY
; USING: MEM 21H
        EXTERN  C_IN
        EXTERN  C_OUT
        EXTERN  STROUT
        EXTERN  SHOWHEX
; UTIL FUNCTION
        EXTERN HEXSTR
        ; Convert HEX 2 ASCII
        ; INPUT  : A - Hex Value
        ; Return : R0 - First Byte
        ;         R1 - Second Byte

            EXTERN RECV_PACKET

            PUBLIC NIC_ISR
        PUBLIC SEND_PACKET

; Input Buffer
IBUFH   EQU     F0H
IBUFL   EQU     00H
; Output Buffer
OBUF    EQU     E000H
OBUFH   EQU     E0H
OBUFL   EQU     00H
; CONSTANT CONFIG DATA
MASTER_ADD      EQU     0000H
SLAVE_ADD       EQU     1000H
PSTOP   EQU     3FH
PSTART  EQU     01H
ISRBUF  EQU     21H
; CONST ISRBUF POINTER
PRX     EQU     08H
PTX     EQU     09H
RXE     EQU     0AH
TXE     EQU     0BH
OVW     EQU     0CH
CNT     EQU     0DH
RDC     EQU     0EH
RST     EQU     0FH

IMR             EQU     00011011B   ;
INTERRUPT MASK REGISTER (EDIT IN
NIC_INIT TOO)
TMP     EQU   2AH
```

```
TRBUF   EQU             041H    ; TRANSFER
BUFFER (NIC LOCAL) DO NOT MATCH
WITH(PSTART&PSTOP)
; SEND PACKET
; MOVE BUFFER TO NIC THEN TRANSMIT
SEND_PACKET:
        LCALL   STROUT
        DB      'TX',0AH,0DH,00H
;           MOV     DPH,#OBUFH
;           MOV     DPL,#OBUFL
        MOV     DPTR,#OBUF
        MOVX    A,@DPTR
        MOV     R1,A                    ;
GET LOW BYTE
        MOV     R7,A
        INC     DPTR ; INC COUNTER
        MOVX    A,@DPTR
        MOV     R0,A                    ;
GET HIGH BYTE
        MOV     R6,A

        MOV     DPTR,#SLAVE_ADD
CHK_SNDNOW:
        MOVX    A,@DPTR
; CHECK IF STILL SENDING OR NOT.
        CJNE    A,#26H,SEND_NOTNOW
        ; IF DO THEN WAIT(POOLLING).
                JMP
        CHK_SNDNOW
SEND_NOTNOW:
;       CLR     EX0

              MOV     DPTR,#SLAVE_ADD
        MOV     DPL,#0AH        ; SET
RBCR[0,1]  RSAR[0,1]
        MOV     A,R1
        MOVX    @DPTR,A
        MOV     DPL,#0BH
        MOV     A,R0
        MOVX    @DPTR,A
        MOV     DPL,#08H        ; SET
RSAR[0,1]
        MOV     A,00H
        MOVX    @DPTR,A
        MOV     DPL,#09H
        MOV     A,#TRBUF                ;
THE START OF TRANSFER PAGE
        MOVX    @DPTR,A
        ; SET RD[2,1,0] -> [0,1,0]
        MOV     DPL,#00H
        MOV     A,#12H          ; Remote
Write
        MOVX    @DPTR,A
;       MOV     DPH,#OBUFH
;       MOV     DPL,#OBUFL
        MOV     DPTR,#OBUF+2
;               INC             DPTR            ;
SKIP 2 BYTE LENGTH
;               INC             DPTR
        INC     R6
        PUSH    DPH
        PUSH    DPL
WDMALP2:
WDMALP1:
        POP     DPL
        POP     DPH
        MOVX    A,@DPTR
        INC     DPTR
        PUSH    DPH
        PUSH    DPL
        JNB     P1.0,$                  ; WAIT
FOR PRQ
        MOV     DPTR,#MASTER_ADD
        MOVX    @DPTR,A
;
;       LCALL   SHOWHEX
;
        DJNZ    R7,WDMALP1
        DJNZ    R6,WDMALP2

WDMA_DONE:
        MOV     DPTR,#SLAVE_ADD+07H
;       MOV     DPL,#07H                ;
READ ISR
        MOVX    A,@DPTR
        MOV     ISRBUF,A
        JNB     RDC,WDMA_DONE
;       JNB     RDC,WDMALP1
        POP     DPL
        POP     DPH
;       CLEAR RDC
        MOV     DPTR,#SLAVE_ADD+07H
        MOV             A,#40H
        MOVX    @DPTR,A
        ; Set TPSR , TBCR1 , TBCR0
        MOV     DPTR,#SLAVE_ADD+04H
;       MOV     DPL,#04H        ;TPSR
        MOV     A,#TRBUF
        MOVX    @DPTR,A
        INC     DPTR            ;TBCR0
        MOV     A,R1
        MOVX    @DPTR,A
        INC     DPTR            ;TBCR1
        MOV     A,R0
        MOVX    @DPTR,A
        MOV     DPL,#00H
        MOV     A,#26H                  ;
Transmit Packet
        MOVX    @DPTR,A
;
        LCALL   STROUT
        DB      '..Complete',0AH,0DH,00H
;
```

```
;       SETB    EX0
        RET

; INTERRUPT SERVICE ROUTINE FOR NIC
NIC_ISR:

        PUSH    DPH
        PUSH    DPL
        PUSH    A
;           CLR         EX1         ;
DISABLE INT1
        CLR     EX0

;
;       LCALL   STROUT
;       DB      0AH,0DH,"ISR :",00H
;
        MOV     DPTR,#SLAVE_ADD     ;
CHANGE TO PAGE 0
        MOVX    A,@DPTR
        ANL     A,#00111111B
        MOVX    @DPTR,A
        MOV     DPL,#07H            ;
READ ISR
        MOVX    A,@DPTR
        MOV     ISRBUF,A
;
;       PUSH    A
;       LCALL   SHOWHEX
;       LCALL   STROUT
;       DB      0AH,0DH,00H
;       POP     A

          JNB     PRX,NOT_PRX     ;
CHECK IF PRX (PACKET RECV)
IS_DAT:
;  LCALL   STROUT
;  DB      "START RDMA
READ",0Dh,0Ah,00h
        LCALL   RECV_PACKET
;  LCALL   STROUT
;  DB      "RDMA DONE",0DH,0AH,00H
NOT_PRX:
;
;       LCALL   CHK_RING
;       CJNE    A,#00,IS_DAT
;
        MOV     DPTR,#SLAVE_ADD+7
; WRITE TO ISR TO CLEAR INT
        MOV     A,#0FFH
        MOVX    @DPTR,A

        MOV     DPL,#0FH            ;
CLEAR IMR
        MOV     A,#0
        MOVX    @DPTR,A

        SETB    EX0                 ;
ENABLE INT0

        MOV     DPL,#0FH            ;
SET IMR
        MOV     A,#IMR              ; (THE
8051 will WILL NOT MISS INT)
        MOVX    @DPTR,A

        POP     A
        POP     DPL
        POP     DPH
        RETI
```

```
; String Utility...
; By Krerk Piromsopa
;
        EXTERN C_OUT

        PUBLIC SHOWHEX
        ; PRINT OUT HEX TO SERIAL
        ; INPUT : A - HEX VALUE

        PUBLIC  HEXSTR
        ; Convert HEX 2 ASCII
        ; INPUT  : A - Hex Value
        ; Return : R0 - First Byte
        ;        R1 - Second Byte
        PUBLIC  STRHEX
        ; CONVERT STR TO VALUE
        ; INPUT  : R0 - First Byte
        ;        R1 - Second Byte
        ; Return : a - Hex Value

             PUBLIC  MOVSTR
        ; Move Sequence of String
        ; INPUT  : R0 - Src H
        ;        R1 - Src L
        ;        R2 - Des H
        ;        R3 - Des L
        ;              R6 - Byte Count H
        ;        R7 - Byte Count L

             PUBLIC  MOVBUF
        ; Move Sequence of String
        ; INPUT  : R0 - DES H
        ;        R1 - DES L

STRTABLE        DB
"0123456789ABCDEF"
HEXSTR:
        PUSH    A
        PUSH    DPH
        PUSH    DPL
        MOV     DPTR,#STRTABLE
        MOV     R1,A
        SWAP    A
        ANL     A,#0FH
        MOVC    A,@A+DPTR
        MOV     R0,A
        MOV     A,R1
        ANL     A,#0FH
        MOVC    A,@A+DPTR
        MOV     R1,A
        POP     DPL
        POP     DPH
        POP     A
        RET

SHOWHEX:
        MOV     B,A
        MOV     A,R0
        PUSH    A
        MOV     A,R1
        PUSH    A
        MOV     A,B
        PUSH    A
        LCALL   HEXSTR
        MOV     A,R0
        LCALL   C_OUT
        MOV     A,R1
        LCALL   C_OUT
        MOV     A,#'x'
        LCALL   C_OUT
        POP     A
        MOV     B,A
        POP     A
        MOV     R1,A
        POP     A
        MOV     R0,A
        MOV     A,B
        RET

STRHEX:
        MOV     A,R0
        JB      A.6,ALPHA1
        SJMP    DIGIT1
ALPHA1:
        ADD     A,#9
DIGIT1:
        ANL     A,#0FH
        SWAP    A
;       LCALL   SHOWHEX
        PUSH    A
        MOV     A,R1
        JB      A.6,ALPHA2
        SJMP    DIGIT2
ALPHA2:
        ADD     A,#9
DIGIT2:
        ANL     A,#0FH
        MOV     R1,A
;       LCALL   SHOWHEX
        POP     A
        ORL     A,R1
;       LCALL   SHOWHEX
        RET

MOVSTR:
        PUSH    A
        PUSH    DPH
        PUSH    DPL
        INC     R6
MOVSTR_LOOP:
        MOV     DPH,R0
        MOV     DPL,R1
;
        CLR     A
```

```
        MOVC    A,@A+DPTR
;
;       MOVX    A,@DPTR
        INC     DPTR
        MOV     R0,DPH
        MOV     R1,DPL
        MOV     DPH,R2
        MOV     DPL,R3
        MOVX    @DPTR,A
        INC     DPTR
        MOV     R2,DPH
        MOV     R3,DPL
        DJNZ    R7,MOVSTR_LOOP
        DJNZ    R6,MOVSTR_LOOP
        POP     DPL
        POP     DPH
        POP     A
        RET

MOVBUF:
        POP     DPH             ;get in-line
string address from stack
        POP     DPL
        ;PUSH    A
MOVBUFLOOP:
        CLR     A
        MOVC    A,@A+DPTR       ;Read
next byte.
        INC             DPTR
        ;Bump pointer.
        CJNE    A,#0,MOVBUF1
        SJMP    END2BUF
MOVBUF1:
        PUSH    DPH
        PUSH    DPL
        MOV     DPH,R0
        MOV     DPL,R1
        MOVX    @DPTR,A
        INC     DPTR
        MOV     R0,DPH
        MOV     R1,DPL
        POP     DPL
        POP     DPH
        SJMP    MOVBUFLOOP
END2BUF:
        ;POP     A
        CLR     A
        JMP     @A+DPTR         ;Return to
program.
```

```
; TCP Module
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY

; Debug Serial routine
        EXTERN  S_INIT
        EXTERN  C_IN
        EXTERN  C_OUT
        EXTERN  STROUT
        EXTERN  HEXSTR
        ; Convert HEX 2 ASCII
        ; INPUT  : A - Hex Value
        ; Return : R0 - First Byte
        ;        R1 - Second Byte

        ; NIC HANDLER ROUTINE
        EXTERN SEND_PACKET

        EXTERN CHECKSUM
        ;OUTPUT : R6 CHECKSUM H
        ;       R7 CHECKSUM L
        ;INPUT  : R0 STARTADD H
        ;       R1 STARTADD L
        ;       R2 BYTECOUNT H
        ;       R3 BYTECOUNT L

        EXTERN  MOVSTR
        ; Move Sequence of String
        ; INPUT  : R0 - Src H
        ;        R1 - Src L
        ;        R2 - Des H
        ;        R3 - Des L
        ;        R6 - Byte Count H
        ;        R7 - Byte Count L

        EXTERN  MOVBUF
        ; Move Sequence of String
        ; INPUT  : R0 - DES H
        ;        R1 - DES L

        EXTERN  TCP_SYN_RSP
        EXTERN  TCP_CHK_ACK
        EXTERN  TCP_PSH_RSP
        EXTERN  TCP_FIN_RSP
        EXTERN  TCP_CHK_ISEQ
;           EXTERN  INIT_TCP_SEND
;           EXTERN  CHECK_TCP_SEND

      ; WEB SERVER COMMAND
        EXTERN  HTTP_INIT
        EXTERN  HTTP_IN
        EXTERN  CHK_HTTP_STAT
        EXTERN  START_TIMER
```

```
        PUBLIC  TCP_RSP
HTTP_PORT EQU 80

OBUF    EQU     E000H
OBUFH   EQU 0E0H
IBUF    EQU     F000H
IBUFH   EQU 0F0H

TCPFLAG EQU 22H  ;(BYTE)
FIN             EQU 10H ;(BIT)
SYN             EQU 11H
RST             EQU 12H
PSH             EQU 13H
ACK             EQU 14H
URG             EQU 15H

STAT EQU 23H ; (BYTE)
TCPINUSE EQU 18H ; (BIT)  0 - LISTEN ,
1 - SYNRECV
TCPCON   EQU 19H ; (BIT)  0 - WAIT FOR
OPEN, 1 - CONNECTION ESTABLISH
WFIN     EQU 1BH ; (BIT)  0 - NO WAIT ,
1 - WAIT FOR FIN
WACK     EQU 1CH ; (BIT)  0 - NOWAIT ,
1 - WAIT FOR ACK PACKET
ACKFAIL  EQU 1EH ; (BIT)  0 - PASS , 1 -
FAIL
NOT_ESEQ EQU 1DH ; (BIT)  0 - PASS , 1
- NOMATCH

BOH             EQU 16
; MEM
BOTCP   EQU 30H
TCPHEADLEN EQU 31H
IPHEADLEN EQU 32H
DATALEN_H EQU 37H
DATALEN_L EQU 38H
TCPPORT EQU   3BH

TCP_RSP:
        ;CREATE PSUEDO HEADER FROM
LAST 12 BYTE OF IP
        MOV     R0,#IBUFH
        MOV     R1,#BOH+8
        MOV     R2,#OBUFH
        MOV     R3,#00H
        MOV     R6,#00H
        MOV     R7,#12
        LCALL   MOVSTR
        MOV     DPTR,#OBUF
        CLR     A               ; INSERT 0 TO
TTL FIELD
        MOVX    @DPTR,A
        INC     DPTR
        INC     DPTR    ; CLEAR IP
CHKSUM & REPLACE WITH TCP LENGTH
        MOVX    @DPTR,A
        INC     DPTR
        MOVX    @DPTR,A
        ;
        ;GET IP HEADER LEN
        MOV     DPH,#IBUFH
        MOV     DPL,#BOH+2
        MOVX    A,@DPTR
        MOV     DATALEN_H,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     DATALEN_L,A
        MOV     A,IPHEADLEN
        ADD     A,#BOH  ; %A CONTENT
THE BEGIN OF TCP HEADER
        MOV     BOTCP,A
        ADD     A,#3
        MOV     DPL,A
        MOVX    A,@DPTR
        ; GET TCP PORT
        MOV     TCPPORT,A
        MOV     A,BOTCP
        ADD     A,#12   ; ADD 12 BYTE
TO GET TCP HEADER LEN (H)
        MOV     DPL,A
        MOVX    A,@DPTR
        SWAP    A
        ANL     A,#0FH
        MOV     B,#4
        MUL     AB              ; %A
CONTENT THE LEN OF HEADER
        MOV     TCPHEADLEN,A
        MOV     DPH,#OBUFH
;               MOV     DPL,#3
;               MOVX    @DPTR,A ; PUT THE
LEN TO PSUEDO HEADER
        CLR     C
        MOV     A,DATALEN_L
        SUBB    A,IPHEADLEN
        MOV     DATALEN_L,A
        MOV     A,DATALEN_H
        SUBB    A,#0
        MOV     DATALEN_H,A
        MOV     DPL,#2          ; PUT LEN
TO PSUEDO HEADER
        MOVX    @DPTR,A
        MOV     A,DATALEN_L
        INC     DPTR
        MOVX    @DPTR,A
        ; MOVE TO OBUF (SKIP 12 BYTE
PSUEDO HEADER) FOR DO CHECKSUM
        MOV     R7,DATALEN_L
        MOV     R6,DATALEN_H
        MOV     R0,#IBUFH
        MOV     R1,BOTCP
        MOV     R2,#OBUFH
```

```
        MOV     R3,#12  ; FOR 12 BYTE
PSUEDO HEADER
        LCALL   MOVSTR
        ; PAD 1 ZERO
        MOV     DPH,R2
        MOV     DPL,R3
        MOV     A,#00H
        MOVX    @DPTR,A
        ;
        MOV     R0,#OBUFH
        MOV     R1,#00H
        MOV     A,DATALEN_L
        CLR     C
        JB      ACC.0,ODD
        ADD     A,#12   ; + PSUEDO
HEADER
        SJMP    ODD_NXT
ODD:
        ADD     A,#13
ODD_NXT:
        MOV     R3,A
        MOV     A,DATALEN_H
        ADDC    A,#0
        MOV     R2,A
        LCALL   CHECKSUM
        CJNE    R6,#0,TCP_FAIL
        CJNE    R7,#0,TCP_FAIL
        SJMP    TCP_CHKSUM_PASS
TCP_FAIL:
        LCALL   STROUT
        DB      "TCP CHKSUM
FAIL",0AH,0DH,00H
        RET
TCP_CHKSUM_PASS:
        LCALL   STROUT
        DB      "TCP CHECKSUM
PASS",0AH,0DH,00H
        ; RECALC FOR TRUE DATALEN
        CLR     C
        MOV     A,DATALEN_L
        SUBB    A,TCPHEADLEN
        MOV     DATALEN_L,A
        MOV     A,DATALEN_H
        SUBB    A,#0
        MOV     DATALEN_H,A
        ;----------------
        MOV     DPTR,#IBUF
        MOV     A,BOTCP
        ADD     A,#13   ; FIND THE FLAG
BYTE
        MOV     DPL,A
        MOVX    A,@DPTR
        MOV     TCPFLAG,A
        JNB     SYN,NOT_SYN
        JB      TCPINUSE,TCP_FAIL_INUSE
        SJMP    NOTINUSE
TCP_FAIL_INUSE:
        LCALL   STROUT
        DB      "FAIL : TCP
INUSE",0AH,0DH,00H
        RET
NOTINUSE:

        MOV     A,TCPPORT
        CJNE
A,#HTTP_PORT,NOT_HTTP_PORT  ;
CHECK IF NOT HTTP_PORT
        SJMP    PORTMATCH
NOT_HTTP_PORT:
        LCALL   STROUT
        DB      "NOT HTTP",0AH,0DH,00H
        RET
PORTMATCH:
        LCALL   HTTP_INIT
        LCALL   TCP_SYN_RSP             ;
PACKET IS SYNC
        LCALL   STROUT
        DB      "SYN
COMPLETE",0AH,0DH,00H
        SETB    WACK    ; MUST WAIT
FOR TCPACK
        LCALL   SEND_PACKET
        LCALL   START_TIMER
        RET
NOT_SYN:
        LCALL   TCP_CHK_ISEQ    ;
CHECK FOR INCOMING SEQ
        JB
NOT_ESEQ,TCP_FAIL_NOTSEQ
        JNB
TCPINUSE,TCP_FAIL_NOCONN        ;
CHECK IF CONN
        JNB     ACK,NO_ACK
        ; PACKET GOT ACK
        ; Check for EXPECTED ACK no...
        LCALL   TCP_CHK_ACK
        JB      ACKFAIL,TCP_ACK_FAIL
        SETB    TCPCON          ; TCP
NOW CONNECTING
        LCALL   STROUT
        DB      "ACK PASS",0AH,0DH,00H
        CLR     WACK
NO_ACK:
        ; IF WACK IS SET MUST WAIT ...
DO NOTHING
        JNB     WACK,NOWACK
        LCALL   STROUT
        DB      "WACK",0AH,0DH,00H
        ; CHECK IF SEND QUEUE IS
EMPTY
        ;               LCALL
CHECK_TCP_SEND
        RET
NOWACK:
```

```
        JNB     FIN,NOT_FIN
        ;
        LCALL   TCP_FIN_RSP
        RET
        ; PACKET IS FIN ; WILLING TO
MANAGE
NOT_FIN:
        JNB
TCPCON,TCP_FAIL_NOCONN     ; CHECK
IF NO CONN
        JNB     PSH,NOT_PSH
        LCALL   STROUT
        DB      "PSH",0AH,0DH,00H
        LCALL   HTTP_IN
        LCALL   TCP_PSH_RSP
        LCALL   SEND_PACKET
NOT_PSH:
        RET
TCP_FAIL_NOCONN:
        LCALL   STROUT
        DB      "FAIL : NO TCP
CONN",0AH,0DH,00H
        RET
TCP_ACK_FAIL:
                LCALL   STROUT
        DB      "ACK FAIL",0AH,0DH,00H
        RET
TCP_FAIL_NOTSEQ:
        LCALL   STROUT
        DB      "NOT EXP
SEQ",0AH,0DH,00H
                RET
```

```
; TCP Util Module (MISC...)
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY

; Debug Serial routine
        EXTERN  S_INIT
        EXTERN  C_IN
        EXTERN  C_OUT
        EXTERN  STROUT

        EXTERN CHECKSUM
        ;OUTPUT : R6 CHECKSUM H
        ;       R7 CHECKSUM L
        ;INPUT  : R0 STARTADD H
        ;       R1 STARTADD L
        ;       R2 BYTECOUNT H
        ;       R3 BYTECOUNT L

        EXTERN  MOVSTR
        ; Move Sequence of String
        ; INPUT  : R0 - Src H
        ;       R1 - Src L
        ;       R2 - Des H
        ;       R3 - Des L
        ;       R6 - Byte Count H
        ;       R7 - Byte Count L

        EXTERN  MOVBUF
        ; Move Sequence of String
        ; INPUT  : R0 - DES H
        ;       R1 - DES L
        EXTERN  SWAP_MAC
        EXTERN  SWAP_IP

        PUBLIC  CREATE_ACK
        ; CREATE ACK AT THE TCP HEAD
        ; INPUT  : R6 - Byte Count H
        ;       R7 - Byte Count L
        ; USING  R2,R3,R4,R5 AS TEMP
        PUBLIC  SWAP_PORT
        PUBLIC  CREATE_TCP_CHECKSUM
        ; CREATE TCP CHECKSUM
        ; INPUT : OUTLENH,OUTLENL

        PUBLIC  TCP_CHK_ISEQ
        ; CHECK FOR EXPECTED
INCOMING SEQ
        PUBLIC  CREATE_BLK_HEADER

TCPFLAG EQU 22H
FIN             EQU 10H
SYN             EQU 11H
RST             EQU 12H
```

```
PSH             EQU 13H
ACK             EQU 14H
URG     EQU 15H

INUSEFLAG EQU 23H
TCPINUSE EQU 18H        ;(BIT)
NOT_ESEQ EQU 1DH        ;(BIT)

IBUF    EQU F000H
OBUF    EQU E000H
BLK_RSP_HEAD EQU        FFC0H
BOH             EQU 16 ; BEGIN OF IP
HEAD
; MEM
BOTCP   EQU 30H
TCPHEADLEN EQU 31H
IPHEADLEN EQU 32H
OUTLENH   EQU 39H
OUTLENL   EQU 3AH
; EXPECTED NEXT INCOME SEQ
ESEQ0   EQU 2CH
ESEQ1   EQU 2DH
ESEQ2   EQU 2EH
ESEQ3   EQU 2FH

SWAP_PORT:
        MOV     R0,#F0H         ; SWAP
PORT ADDRESS
        MOV     R1,BOTCP
        MOV     R2,#E0H
        MOV     R3,BOTCP
        INC     R3          ;( +2 BYTE FOR
DES PORT)
        INC     R3
        MOV     R6,#0
        MOV     R7,#2
        LCALL   MOVSTR
        MOV     R0,#F0H
        MOV     R1,BOTCP
        INC     R1          ; ( +2 BYTE FOR
DES PORT)
        INC     R1
        MOV     R2,#E0H
        MOV     R3,BOTCP
        MOV     R6,#0
        MOV     R7,#2
        LCALL   MOVSTR
        RET

; CREATE ACK AT THE TCP HEAD
; INPUT  : R6 - Byte Count H
;        R7 - Byte Count L
; USING  R2,R3,R4,R5 AS TEMP
; SET ESEQ[0..4]
CREATE_ACK:
        MOV     DPH,#F0H
        MOV     A,BOTCP
        ADD     A,#4    ; FOR IBUF SEQ
NO
        MOV     DPL,A
        MOVX    A,@DPTR
        MOV     R2,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R3,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R4,A
        INC     DPTR
        MOVX    A,@DPTR
        ;MOV    R5,A
        CLR     C
        ADD     A,R7
        MOV     ESEQ3,A
        MOV     A,R4
        ADDC    A,R6
        MOV     ESEQ2,A
        MOV     A,R3
        ADDC    A,#0
        MOV     ESEQ1,A
        MOV     A,R2
        ADDC    A,#0
        MOV     ESEQ0,A
        ; OUTPUT TO OBUF
        MOV     DPH,#E0H
        MOV     A,BOTCP
        ADD     A,#8    ; FOR OBUF ACK
NO
        MOV     DPL,A
        MOV     A,ESEQ0
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ESEQ1
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ESEQ2
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,ESEQ3
        MOVX    @DPTR,A
        ; COMPLETE
        RET

; CREATE TCP CHECKSUM FIELD
CREATE_TCP_CHECKSUM:
        LCALL   STROUT
        DB      "GEN TCP
CHKSUM",0AH,0DH,00H
        ; GEN PSUEDO HEADER
        MOV     R0,#E0H
        MOV     R1,#16+8
        MOV     R2,#C0H
        MOV     R3,#00
        MOV     R6,#0
```

```
        MOV     R7,#12
        LCALL   MOVSTR
        MOV     DPTR,#C000H
        CLR     A
        MOVX    @DPTR,A
        ; PUT LEN TO PSUEDO HEAD
        MOV     DPL,#3
        MOV     A,OUTLENL
        ADD     A,TCPHEADLEN
        PUSH    A
        MOVX    @DPTR,A
        CLR     A
        ADDC    A,OUTLENH
        MOV     DPL,#2
        PUSH    A
        MOVX    @DPTR,A
        ; CLEAR OBUF CHECKSUM
        MOV     DPH,#E0H
        MOV     A,BOTCP
        ADD     A,#16
        MOV     DPL,A
        CLR     A
        MOVX    @DPTR,A
        INC     DPTR
        MOVX    @DPTR,A
        ; COPY TCP HEAD & DATA
        MOV     R0,#E0H
        MOV     R1,BOTCP
        MOV     R2,#C0H
        MOV     R3,#12
        POP     A
        MOV     R6,A
        POP     A
        MOV     R7,A
        PUSH    A       ; LEN L
        MOV     A,R6
        PUSH    A       ; LEN H
        LCALL   MOVSTR
        ; RUN CHECKSUM
        MOV     R0,#C0H
        MOV     R1,#00H
        POP     A
        MOV     R2,A
        POP     A
        MOV     R3,A
        CLR     C
        ADD     A,#12  ; ADD PSUDO
HEAD LEN
        MOV     R3,A
        MOV     A,R2
        ADDC    A,#0
        MOV     R2,A
        LCALL   CHECKSUM
        MOV     DPH,#E0H      ; PUT
BACK THE CHKSUM
        MOV     A,BOTCP
        ADD     A,#16
        MOV     DPL,A
        MOV     A,R6
        MOVX    @DPTR,A
        MOV     A,R7
        INC     DPTR
        MOVX    @DPTR,A
        LCALL   STROUT
        DB
        "CHKSUM_CMP",0DH,0AH,00H
        RET

TCP_CHK_ISEQ:
; CHECK FOR EXPECTED INCOMING SEQ
        CLR     NOT_ESEQ
        MOV     DPH,#F0H
        MOV     A,BOTCP
        ADD     A,#4    ; FOR IBUF SEQ
NO
        MOV     DPL,A
        MOVX    A,@DPTR
        CJNE    A,ESEQ0,WRONG_SEQ
        INC     DPTR
        MOVX    A,@DPTR
        CJNE    A,ESEQ1,WRONG_SEQ
        INC     DPTR
        MOVX    A,@DPTR
        CJNE    A,ESEQ2,WRONG_SEQ
        INC     DPTR
        MOVX.   A,@DPTR
        CJNE    A,ESEQ3,WRONG_SEQ
        SJMP    ETCP_ISEQ
WRONG_SEQ:
        SETB    NOT_ESEQ
ETCP_ISEQ:
        RET

CREATE_BLK_HEADER:
        ; UPDATE IDENT NO.
        MOV
DPTR,#BLK_RSP_HEAD+14+5
        MOVX    A,@DPTR
        ADD     A,#2
        MOVX    @DPTR,A
        MOV
DPTR,#BLK_RSP_HEAD+14+4
        MOVX    A,@DPTR
        ADDC.   A,#0
        MOVX    @DPTR,A
        ; MOV TO OBUF
        MOV     R0,#HIGH
(BLK_RSP_HEAD)
        MOV     R1,#LOW
(BLK_RSP_HEAD)
        MOV     R2,#HIGH(OBUF)
        MOV     R3,#LOW(OBUF)+2
        MOV     R6,#0
        MOV     R7,#53
```

```
LCALL   MOVSTR
MOV     IPHEADLEN,#20
MOV     TCPHEADLEN,#20
MOV     BOTCP,#2+14+20
RET
```

```
; PHP-LITE SCRIPT INTERPRET
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY
        EXTERN  C_IN
        EXTERN  C_OUT
        EXTERN  STROUT
        EXTERN  SHOWHEX
        EXTERN  HEXSTR
        EXTERN  STRHEX

        EXTERN  MOVSTR
        EXTERN  EXPRESSION
        EXTERN  ISCHAR
        EXTERN  ISNUM
        EXTERN  INITSYMTABLE
        EXTERN  SETVAR
        EXTERN  GETTOKEN
        EXTERN  DOINP
        EXTERN  DOOUTP
        EXTERN  DOIF
        EXTERN  DOWHILE
        EXTERN  DOSTRCONV
        EXTERN  PHPENV
        PUBLIC  GETDAT
        PUBLIC  SKPDLIM
        PUBLIC  SEMICOLONEND
        PUBLIC  PHP
        PUBLIC  DOSTATEMENTS

STOP    EQU     0
VAR     EQU     1
WHILE   EQU     2
IF      EQU     3
ECHO    EQU     4
INP     EQU     5
OUTP    EQU     6
COMMA   EQU     7
OB      EQU     8
CB      EQU     9
SCOLON  EQU     10
START   EQU     11
PLUS    EQU     12
MINUS   EQU     13
DOT     EQU     14
CONST   EQU     15
EQ      EQU     16
STRCONV EQU     17
ERROR   EQU     FFH

;       ORG     0000H
;       LJMP    PHPMAIN

;PAGE 0
```

```
;---------------------------
;R2 - curptr h
;R3 - curptr l
;R4 -
;R5 -
;R6 - TMP
;R7 - TMP
;---------------------------
;PAGE 1 FOR OTHER MODULE

VARNAME      EQU   0DFA0H ; 256
BYTE BUFFER
VARBUFF      EQU   0DFA8H
SYMBUFF      EQU   0DFC0H
EXPOUT       EQU   0DF00H ; FIRST 2
BYTE DPH,DPL
ASSBUFF      EQU   0DF90H ; ASSIGN
BUFFER

DOECHO:
; ECHO DATA -> ECHO (EXP);
        LCALL  STROUT
        DB     "ECHO",0DH,0AH,00H
        LCALL  GETTOKEN
        CJNE   A,#OB,NOTOB
        LCALL  EXPRESSION
        MOV    DPTR,#EXPOUT+2
ECHOLOOP:
        MOVX   A,@DPTR
        INC    DPTR
        CJNE   A,#00,ECHOEMIT
        SJMP   SEMICOLONEND
ECHOEMIT:
        MOV    R6,A
        LCALL  EMIT_CHAR
        SJMP   ECHOLOOP
NOTOB:
SEMICOLONEND:
        LCALL  GETTOKEN
        CJNE
A,#SCOLON,SEMICOLONEND
        RET

DOASSIGN:
;
        LCALL  STROUT
        DB     "ASSIGN",00H
        SETB   RS0
        MOV    R0,#HIGH(SYMBUFF)
        MOV    R1,#LOW(SYMBUFF)
        MOV    R2,#HIGH(ASSBUFF)
        MOV    R3,#LOW(ASSBUFF)
        MOV    R6,#0
        MOV    R7,#8
        LCALL  MOVSTR
        CLR    RS0
        LCALL  GETTOKEN
        CJNE   A,#EQ,NOTEQ
        SJMP   ASSIGNEXP
NOTEQ:
ENDASSIGN:
;       LJMP   SEMICOLONEND  ;
ALREADY END WITH SEMICOLON
        LCALL  STROUT
        DB     "END ASSIGN",00H
        RET
ASSIGNEXP:
        LCALL  EXPRESSION
        SETB   RS0
        MOV    R0,#HIGH(ASSBUFF)
        MOV    R1,#LOW(ASSBUFF)
        MOV    R2,#HIGH(VARNAME)
        MOV    R3,#LOW(VARNAME)
        MOV    R6,#0
        MOV    R7,#8
        LCALL  MOVSTR
        MOV    R0,#HIGH(EXPOUT+2)
        MOV    R1,#LOW(EXPOUT+2)
        MOV    R2,#HIGH(VARBUFF)
        MOV    R3,#LOW(VARBUFF)
        MOV    R6,#0
        MOV    R7,#24
        LCALL  MOVSTR
        CLR    RS0
        LCALL  SETVAR
        SJMP   ENDASSIGN

DOSTATEMENTS:
; CHECK FOR MATCH FUNCTION
; IF MATCH JUMP TO ROUTINE
STMLOOP:
        LCALL  GETTOKEN
        CJNE   A,#WHILE,NOTWHILE
        LCALL  DOWHILE
        SJMP   STMLOOP
NOTWHILE:
        CJNE   A,#IF,NOTIF
        LCALL  DOIF
        SJMP   STMLOOP
NOTIF:
        CJNE   A,#ECHO,NOTECHO
        LCALL  DOECHO
        SJMP   STMLOOP
NOTECHO:
        CJNE   A,#INP,NOTINP
        LCALL  DOINP
        SJMP   STMLOOP
NOTINP:
        CJNE   A,#OUTP,NOTOUTP
        LCALL  DOOUTP
        SJMP   STMLOOP
NOTOUTP:
        CJNE   A,#VAR,NOTVAR
        LCALL  DOASSIGN
```

```
        SJMP    STMLOOP
NOTVAR:
        CJNE
A,#STRCONV,NOTSTRCONV
        LCALL   DOSTRCONV
        SJMP    STMLOOP
NOTSTRCONV:
; NOT SUPPORT / END SO RETURN
        LCALL   SHOWHEX
        LCALL   STROUT
        DB      "END STATEMENTS",00H
        RET


EMIT_CHAR:
; EMIT DATA
; R6 - CURDAT
        PUSH    DPH
        PUSH    DPL
        PUSH    A
;       MOV     A,R6
;       LCALL   C_OUT
        MOV     DPTR,#D000H
        MOVX    A,@DPTR
        PUSH    A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     DPL,A
        POP     DPH
        MOV     A,R6
        MOVX    @DPTR,A
        INC     DPTR
        CLR     A       ; TERMINATE
OUTPUT
        MOVX    @DPTR,A
        PUSH    DPH
        PUSH    DPL
        MOV     DPTR,#D001H
        POP     A
        MOVX    @DPTR,A
        MOV     DPTR,#D000H
        POP     A
        MOVX    @DPTR,A
        POP     A
        POP     DPL
        POP     DPH
        RET

GETDAT:
; INPUT  R2,R3
; OUTPUT R6 - *(CURRDAT)
;       R7 - *(CURRDAT+1)  .. LOOK
AHEAD
        PUSH    DPH
        PUSH    DPL
        PUSH    A
        MOV     DPH,R2
        MOV     DPL,R3
        MOVX    A,@DPTR
        MOV     R6,A    ; CURR DAT
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R7,A    ; LOOK AHEAD
        MOV     R2,DPH
        MOV     R3,DPL
        POP     A
        POP     DPL
        POP     DPH
        RET

FINDESC:
; FIND THE ESC BLOCK <%
        LCALL   GETDAT
        CJNE    R6,#00,CHKESC
        SJMP    ENDFINDESC
CHKESC:
        CJNE    R6,#'<',TOEMIT
        CJNE    R7,#'%',TOEMIT
        LCALL   GETDAT  ; SKIP %\
        SJMP    ENDFINDESC
TOEMIT:
        LCALL   EMIT_CHAR
        SJMP    FINDESC ; LOOP
ENDFINDESC:
        RET

SKPDLIM:
; skip delimeter
        LCALL   GETDAT
        CJNE    R6,#' ',NOTBLK  ; IF ' '
THEN SKIP
        SJMP    SKPDLIM
NOTBLK:
        CJNE    R6,#0DH,NOT0D  ; IF
0DH THEN SKIP
        SJMP    SKPDLIM
NOT0D:
        CJNE    R6,#0AH,NOT0A  ; IF 0AH
THEN SKIP
        SJMP    SKPDLIM
NOT0A:
; CHECK IF END ESC BLOCK
        CJNE    R6,#'%',ENDSKPDLIM
        CJNE    R7,#'>',ENDSKPDLIM
        LCALL   GETDAT  ; SKIP >
        LCALL   FINDESC
        CJNE    R6,#0,SKPDLIM   ;
SHOULD NOT BE HERE
ENDSKPDLIM:
        RET

PHPMAIN:
PHP:
        PUSH    DPH
```

```
        PUSH    DPL
; INIT STACK & POINTER
        LCALL   INITSYMTABLE
        LCALL   PHPENV
        MOV     DPTR,#D000H
        MOV     A,#HIGH(D002H)
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#LOW(D002H)
        MOVX    @DPTR,A
        MOV     R2,#HIGH(B020H)
        MOV     R3,#LOW(B020H)
        LCALL   FINDESC
        CJNE    R6,#0,PHPDO
        SJMP    PHPEND
PHPDO:
        LCALL   DOSTATEMENTS
PHPEND:
        POP     DPL
        POP     DPH
        RET
```

```
; Computer Engineer Senior Project
; String Routine
; By Krerk Piromsopa (3817166)
; Date : 29 July 1998 (0.2a)
; Rewrite / Optimize for Thesis
; Date : 23 February 2000
; Reversion : 0.3a

        EXTERN SHOWHEX

; Usage Internal MEM : 2BH
; String must Terminate with null (00H)
TMP   EQU   2BH
NUL   EQU   00H

        PUBLIC STR_CX_COMP
; Compare CODE String with EX DATA
String
; Input
; R1 - Hi byte address pointer to STR1
; R2 - Low byte address pointer to STR1
; using
; R3 - Hi byte address pointer to CODE
STR2 ( not use )
; R4 - Low byte address pointer to CODE
STR2 ( not use )
; now change to immediate
; Output
; R0 - 0 if same
;      1 if difference
STR_CX_COMP:
        POP     DPH             ;get in-line
string address from stack
        POP     DPL
        MOV     R3,DPH
        MOV     R4,DPL
NXT_CHR:
        MOV   DPH,R1
        MOV   DPL,R2
        MOVX  A,@DPTR
        INC   DPTR
        MOV   R1,DPH
        MOV   R2,DPL
        MOV   TMP,A
        MOV   DPH,R3
        MOV   DPL,R4
;   MOV   A,#0
        CLR   A
        MOVC  A,@A+DPTR
        INC   DPTR
        MOV   R3,DPH
        MOV   R4,DPL
;   MOV   TMP,R5
        CJNE  A,#NUL,STR_NOT_NULL
        SJMP  STR_NULL
STR_NOT_NULL:
```

```
        CJNE
A,TMP,STR_CX_COMP_NOTEQ
        SJMP  NXT_CHR
;END
STR_NULL:
        MOV   R0,#0
        SJMP  STR_END_RET
STR_CX_COMP_NOTEQ:
        MOV   R0,#1
STR_END:
        CLR  A
        MOVC A,@A+DPTR
        CJNE A,#00H,STR_END_SRC
        SJMP STR_END_RET
STR_END_SRC:
        INC  DPTR
        SJMP STR_END
STR_END_RET:
        CLR     A
        JMP     @A+DPTR


        PUBLIC STR_FIND_CHR
; Find First Position of CHAR in String
; Input
; R1 - Character to find
; DPTR - Pointer to the String. ; End with
NUL
; Output
; R0 - OFFSET
;     FF if not found..
STR_FIND_CHR:
        MOV   R0,#00H
        MOV   TMP,R1
STR_FIND_LOOP:
        MOVX  A,@DPTR
        CJNE
A,TMP,STR_FIND_CHR_NOTEQ
        RET
STR_FIND_CHR_NOTEQ:
        INC   R0
        INC   DPTR
        CJNE  A,#NUL,STR_FIND_LOOP
STR_FIND_CHR_NOTFOUND:
        MOV   R0,#FFH
        RET

              PUBLIC  STR_NCX_COMP
; COMPARE CHAR UNTIL NULL with CODE
DATA
; input
; R1 - Hi byte address pointer to STR1
; R2 - Low byte address pointer to STR1
; R3 - Hi byte address pointer to CODE
STR2
; R4 - Low byte address pointer to CODE
STR2
; Output
; R0 - 0 if same
;      1 if not found..
STR_NCX_COMP:
        MOV   DPH,R1
        MOV   DPL,R2
        MOVX  A,@DPTR
        INC   DPTR
        MOV   R1,DPH
        MOV   R2,DPL
        MOV   R6,A
        MOV   DPH,R3
        MOV   DPL,R4
        MOV   A,#0
        MOVC  A,@A+DPTR
        INC   DPTR
        MOV   R3,DPH
        MOV   R4,DPL
        MOV   TMP,R6
        CJNE
A,TMP,STR_CX_COMP_N_CHAR_NOTEQ
        CJNE  A,#NUL,STR_NCX_COMP
; END
        MOV   R0,#0
        RET
STR_CX_COMP_N_CHAR_NOTEQ:
        MOV   R0,#1
        RET
```

```
; HTTP AUTHENTICATION
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY
        EXTERN  C_OUT
        EXTERN  STROUT
        EXTERN  SHOWHEX
        EXTERNSTR_CX_COMP
        EXTERN  STR_NCX_COMP
; COMPARE CHAR UNTIL NULL with CODE
DATA
; input
; R1 - Hi byte address pointer to STR1
; R2 - Low byte address pointer to STR1
; R3 - Hi byte address pointer to CODE
STR2
; R4 - Low byte address pointer to CODE
STR2
; Output
; R0 - 0 if same
;     1 if not found..
        EXTERN STR_FIND_CHR

        EXTERN  MOVSTR
        ; Move Sequence of String
        ; INPUT  : R0 - Src H
        ;          R1 - Src L
        ;          R2 - Des H
        ;          R3 - Des L
        ;          R6 - Byte Count H
        ;          R7 - Byte Count L

        EXTERN  HTTP_SEND
        PUBLIC  HTTP_HEAD_SHOW
        PUBLIC  AUTH_CHECK
        PUBLIC  AUTH_SHOW
        PUBLIC  BASE64DECODE
        PUBLIC  DECODE1

TMP     EQU     2BH
;
HTTP_IBUF       EQU     D000H
AUTHDAT         EQU     DFE0H
FILEBUFF        EQU     B000H

AUTH_CHECK:
; RETURN 0 IF FOUND AUTHDAT
; SEND AUTH AND RETURN FF IF NOT
FOUND
        LCALL   STROUT
        DB      "CHECK
AUTH",0DH,0AH,00H
        MOV     DPTR,#(HTTP_IBUF+2)
AUTH_CHECKFINDA:
        MOV     R1,#'A'
        LCALL   STR_FIND_CHR
;       MOV     A,DPH
;       LCALL   SHOWHEX
;       MOV     A,DPL
;       LCALL   SHOWHEX
        INC     DPTR
        CJNE    R0,#FFH,FOUND
        SJMP    AUTH_NOTFOUND
FOUND:
        PUSH    DPH
        PUSH    DPL
        MOV     R1,DPH
        MOV     R2,DPL
        LCALL   STR_CX_COMP
        DB      "uthorization: Basic ",00H
;       MOV     A,R0
;       LCALL   SHOWHEX
        POP     DPL
        POP     DPH
        CJNE
R0,#0,AUTH_CHECKFINDA
        MOV     A,DPL
        ADD     A,#20
        MOV     DPL,A
        MOV     A,DPH
        ADDC    A,#0
        MOV     DPH,A
        SJMP    AUTH_CHECKEND
AUTH_NOTFOUND:
        LCALL   STROUT
        DB      "NO AUTH",0DH,0AH,00H
        SJMP    AUTH_SHOW
AUTH_CHECKEND:
        LCALL   BASE64DECODE
        LCALL   STROUT
        DB      "AUTH
PASS",0DH,0AH,00H
        MOV     R1,#HIGH(FILEBUFF+15)
        MOV     R2,#LOW(FILEBUFF+15)
        MOV     R3,#HIGH(AUTHDAT)
        MOV     R4,#LOW(AUTHDAT)
        LCALL   STR_NCX_COMP
        CJNE    R0,#0,AUTH_NOTFOUND
; AUTH FAIL
        MOV     A,#0
        RET

HTTP_HEAD_SHOW:
        MOV     R0,#HIGH(RSP_HEAD)
        MOV     R1,#LOW(RSP_HEAD)
        MOV     R6,#HIGH(101)
        MOV     R7,#LOW(101)
        LCALL   HTTP_SEND
        LCALL   STROUT
        DB      "SEND HTTP
HEAD",0DH,0AH,00H
```

```
        RET

AUTH_SHOW:
        MOV     R0,#HIGH(AUTH_TEXT)
        MOV     R1,#LOW(AUTH_TEXT)
        MOV     R6,#HIGH(426)
        MOV     R7,#LOW(426)
        LCALL   HTTP_SEND
        LCALL   STROUT
        DB      "SEND
AUTH",0DH,0AH,00H
        MOV     A,#FFH
        RET

BTAB    DB      "ABCDEFGHIJKLMNOP"
        DB      "QRSTUVWXYZabcdef"
        DB      "ghijklmnopqrstuv"
        DB      "wxyz0123456789+/",00H
DECODE1:
        PUSH    DPH
        PUSH    DPL
        MOV     TMP,A
        CLR     A
        MOV     R0,A
        MOV     DPTR,#BTAB
DECODE1LOOP:
        MOV     A,R0
        MOVC    A,@A+DPTR
        CJNE    A,#00,CHK1
        SJMP    ENDDECODE1
CHK1:
        CJNE    A,TMP,DECODENXT
        SJMP    ENDDECODE1
DECODENXT:
        INC     R0
        SJMP    DECODE1LOOP
ENDDECODE1:
        MOV     A,R0
        ANL     A,#3FH
        POP     DPL
        POP     DPH
        RET

BASE64DECODE:
;       MOV     A,DPH
;       LCALL   SHOWHEX
;       MOV     A,DPL
;       LCALL   SHOWHEX
        MOV     R2,#HIGH(AUTHDAT)
        MOV     R3,#LOW(AUTHDAT)
B64LOOP:
        MOVX    A,@DPTR
        INC     DPTR
        LCALL   DECODE1
        RL      A
        RL      A
        MOV     R1,A
        MOVX    A,@DPTR
        INC     DPTR
        LCALL   DECODE1
        PUSH    A
        SWAP    A

        ANL     A,#03H
        ORL     A,R1
;       HERE COME DIGIT 1
        PUSH    DPH
        PUSH    DPL
        MOV     DPH,R2
        MOV     DPL,R3
        MOVX    @DPTR,A
;       LCALL   SHOWHEX
        INC     DPTR
        MOV     R3,DPL
        MOV     R2,DPH
        POP     DPL
        POP     DPH
        POP     A
        SWAP    A
        ANL     A,#F0H
        MOV     R1,A
        MOVX    A,@DPTR
        INC     DPTR
        LCALL   DECODE1
        PUSH    A
        RR      A
        RR      A
        ANL     A,#0FH
        ORL     A,R1
;       HERE COME DIGIT 2
        PUSH    DPH
        PUSH    DPL
        MOV     DPH,R2
        MOV     DPL,R3
        MOVX    @DPTR,A
;       LCALL   SHOWHEX
        INC     DPTR
        MOV     R3,DPL
        MOV     R2,DPH
        POP     DPL
        POP     DPH
        POP     A
        SWAP    A
        ANL     A,#F0H
        RL      A
        RL      A
        MOV     R1,A
        MOVX    A,@DPTR
        INC     DPTR
        LCALL   DECODE1
        ORL     A,R1
;       HERE COME DIGIT 3
        PUSH    DPH
        PUSH    DPL
```

```
        MOV     DPH,R2
        MOV     DPL,R3
        MOVX    @DPTR,A
;       LCALL   SHOWHEX
        INC     DPTR
        MOV     R3,DPL
        MOV     R2,DPH
        POP     DPL
        POP     DPH
        MOVX    A,@DPTR
        CJNE    A,#0DH,B64LOOP
; TERMINATE STRING
        PUSH    DPH
        PUSH    DPL
        MOV     DPH,R2
        MOV     DPL,R3
        CLR     A
        MOVX    @DPTR,A
        POP     DPL
        POP     DPH
        RET

AUTH_TEXT:
        ;
        00000000001111111111222222222233333333334444444444
        ;
01234567890123456789012345678901234567890123456789
        DB      "HTTP/1.1 401 Authorization Required",0DH,0AH      ;38 Byte
        DB      "Server: Krerk(EmWeb)/1.0 (Thesis)",0DH,0AH    ; 35 Byte
        DB      "WWW-Authenticate: Basic realm=",22h,"EmWeb",22h,0DH,0AH ; 39 Byte
        DB      "Connection: close",0DH,0AH     ; 19 Byte
        DB      "Content-Type: text/html",0DH,0AH,0DH,0AH,0DH,0AH  ; 29 Byte
        DB      "<HTML>"      ;6
        DB      "<HEAD><TITLE>Krerk (EmWeb)/Thesis</TITLE></HEAD>"  ; 47
        DB      "<BODY>"      ;6
        DB      "<H1>Authorization Require</H1>"      ; 30
        DB      "<P>Server <I>Krerk (EmWeb)/1.0 (Thesis)</I> "  ; 43
        DB      "Could not verify your user name and password " ; 45
        DB      "to Access the Control System..</P>" ; 34
        DB      "<A>Copy Right &copy; Krerk Piromsopa.</A>" ; 41
        DB      "</BODY>"      ;7
        DB      "</HTML> ",00H   ;7

RSP_HEAD:
        DB      "HTTP/1.1 200 OK",0DH,0AH    ;17 Byte
        DB      "Server: Krerk(EmWeb)/1.0 (Thesis)",0DH,0AH   ; 36 Byte
        DB      "Connection: close",0DH,0AH ; 19 Byte
        DB      "Content-Type: text/html",0DH,0AH,0DH,0AH,0DH,0AH  ; 29 Byte
        DB      00H   ;0
```

```
;
;       CONSOLE I/O ROUTINES AND
DRIVERS:
;       ======= === ========
=== =======
; S_INIT - Initializes Serial port.
;
; C_IN   - waits for character from serial
port. Returns it in A.
; C_OUT  - sends character in A.
; C_STS  - console status. if char RXD,
C=1, A=char, else C=0.
; STROUT - Sends in-line character string
to console. String is terminated
;       by last character's MSB set.
Hence, can only handle 7 bit ASCII.
;
;**********************************
**********
;
; Baud rate manually set to desired rate
; Using TIMER 1
; This version uses 1x baud rate
;
        public  S_INIT
        public  C_IN
        public  C_OUT
        public  STROUT
        public  Ser_isr
;
;
;Baud_Rate = Fosc/12 * (2^smod/32) /
(256-TH1)
;       where 2^smod = 2 for smod=1,
and 1 for smod=0
;       Fosc is oscillator frequency; and
TH1 is timer 1 reload value.
;TH1 =  256 - (Fosc/12 * (2^smod/32) /
Baud_Rate)
;
;Fosc = 12MHz  16MHz  20MHz  11.0592
        14.745618.4320smod
;Rate
;150    030H    --      --      040H
        000H    --      0
;300    098H    075H    052H    0A0H
        080H    060H    0
;600    0CCH    0BBH    0A9H    0D0H
        0C0H    0B0H    0
;1200   0E6H    0DEH    0D5H    0E8H
        0E0H    0D8H    0
;2400   0F3H    0EFH    0EAH    0F4H
        0F0H    0ECH    0
;4800   *       *       0F5H    0FAH
        0F8H    0F6H    0
;9600   --      --      *       0FDH
        0FCH    0FBH    0
;19200  --      --      --      0FDH
        0FCH    0FBH    1
;38400  --      --      --      --
        0FEH    --      1
;76800  --      --      --      --
        0FFH    --      1
;* These baud rates available by using the
previous value, and setting SMOD=1
BaudLoad        equ     0FDh    ;9600 baud
@ 11.059
;
Ser_isr:
        reti
S_INIT:
        CLR     TR1
        MOV     SCON,#01011010B
        ;TI set indicates transmitter ready.
                                ; mode
1,REN
        MOV     TMOD,#00100001B
        ;Timer 1 is set to 8 bit auto reload
mode
;       orl     PCON,#SMOD      ; Set to
double rate.
        mov     th1,#BaudLoad ; Set
reload value
        setb    tr1             ; start
timer.
        ret
;
;=======
;
C_IN:
;       Console character input routine.
;       Waits for next input from console
device and returns with character
;       code in accumulator.
;
        JNB     RI,$            ;Wait until
character received.
        MOV     A,SBUF          ;Read input
character.
        CLR     RI              ;Clear
reception flag.
        ret                     ;
;
;=======
;
C_OUT:
;       Console character output routine.
;       Outputs character received in
accumulator to console output device.
;
        JNB     TI,$            ;Wait until
transmission completed.
        CLR     TI              ;Clear interrupt
flag.
```

```
        MOV     SBUF,A          ;Write out
character.
        RET
;
;=======
;
;Console Status
;
; Returns C=0 if no character is ready
; if character ready, returns C=1 and
character in A
; Note: Serial input status can be checked
by RI bit, also.
;
C_STS: MOV     C,RI
        JNC     CNTRET
        ;Poll whether character has been
typed.
        CALL    C_IN
CNTRET:         RET
;
;=======
;
;STROUT
;       Copy in-line character string to
console output device.
;       uses:   DPTR,ACC
;
STROUT:         POP     DPH
        ;get in-line string address from
stack
        POP     DPL
STRO_1:         CLR     A
        MOVC    A,@A+DPTR       ;Read
next byte.
        INC     DPTR            ;Bump
pointer.
;       JBC     ACC.7,STRO_2    ;Escape
after last character.
        CJNE    A,#0,STRO_3
        SJMP    STRO_2
STRO_3:
        CALL    C_OUT           ;Output
character.
        SJMP    STRO_1          ;Loop
until done.
;
STRO_2:
;       CALL    C_OUT           ;Output
character.
;       CLR     A
        JMP     @A+DPTR
        ;Return to program.
;
;       end
```

```
; HTTP URL / FILE HANDLER
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY
        EXTERN  C_OUT
        EXTERN  STROUT
        EXTERN  SHOWHEX
        EXTERN  STRHEX
        ; R0,R1 -> A
        EXTERNSTR_CX_COMP
        EXTERNSTR_FIND_CHR

            EXTERN  MOVSTR
        ; Move Sequence of String
        ; INPUT  : R0 - Src H
        ;         R1 - Src L
        ;         R2 - Des H
        ;         R3 - Des L
        ;         R6 - Byte Count H
        ;         R7 - Byte Count L

            EXTERN  HTTP_SEND
        EXTERN  HTTP_HEAD_SHOW
        EXTERN  AUTH_CHECK
        EXTERN  PHP

            PUBLIC  HTTP_URL
        PUBLIC  READCHAR

TMP     EQU     2BH
;
HTTP_IBUF       EQU     D000H
FILEBUFF        EQU     B000H
URL     EQU     DFD0H

; DEFAULT PAGE FOR UPLOAD FILE
HTTP_URL:
        MOV     DPTR,#HTTP_IBUF+2
;       PUSH    DPH
;       PUSH    DPL
        MOV     R1,DPH
        MOV     R2,DPL
        LCALL   STR_CX_COMP
        DB      "GET /",00H
;       POP     DPL
;       POP     DPH
        CJNE    R0,#0,NOTSUPP
        MOV     DPTR,#HTTP_IBUF+7
        MOV     R1,#HIGH(URL)
        MOV     R2,#LOW(URL)
URLLOOP:
        MOVX    A,@DPTR
        LCALL   C_OUT
        CJNE    A,#' ',CHK1
```

```
        SJMP    ENDURLNAME
CHK1:
        CJNE    A,#'?',TOURLBUF
        SJMP    ENDURLNAME
TOURLBUF:
        PUSH    DPH
        PUSH    DPL
        MOV     DPH,R1
        MOV     DPL,R2
        MOVX    @DPTR,A
        INC     DPTR
        MOV     R1,DPH
        MOV     R2,DPL
        POP     DPL
        POP     DPH
        INC     DPTR
        SJMP    URLLOOP
ENDURLNAME:
        PUSH    DPH
        PUSH    DPL
        ; TERMINATE STRING
        MOV     DPH,R1
        MOV     DPL,R2
        CLR     A
        MOVX    @DPTR,A
        MOV     DPTR,#URL
        MOVX    A,@DPTR
        CJNE    A,#0,NOTHOME
        POP     DPL
        POP     DPH
        SJMP    HOMEPAGE
NOTHOME:
        MOV     R1,#HIGH(URL)
        MOV     R2,#LOW(URL)
        LCALL   STR_CX_COMP
        DB      "upload",00h
        POP     DPL
        POP     DPH
        CJNE    R0,#0,OTHER
        SJMP    UPLOAD_FILE
NOTSUPP:
HOMEPAGE:
        LCALL   HTTP_HEAD_SHOW
        MOV     R0,#HIGH(DEFAULT)
        MOV     R1,#LOW(DEFAULT)
        MOV     R6,#HIGH(1278)
        MOV     R7,#LOW(1278)
        LCALL   HTTP_SEND
        RET
OTHER:  ; OTHER URL ; ASSUME TO BE
THE SCRIPT
        ; CHECK AUTH
        LCALL   AUTH_CHECK
        CJNE    A,#0,NOAUTH
        ; SEARCH END OF STRING
        LCALL   PHP
        LCALL   HTTP_HEAD_SHOW
        MOV     R1,#0
        MOV     DPTR,#D000H     ; START
OF OUTPUTFILE
        LCALL   STR_FIND_CHR
        ; CAL LENGTH THEN SENDOUT
        MOV     A,DPL
        CLR     C
        SUBB    A,#LOW(D002H)
        MOV     R7,A
        MOV     A,DPH
        SUBB    A,#HIGH(D002H)
        MOV     R6,A
        MOV     R0,#HIGH(D002H)
        MOV     R1,#LOW(D002H)
        LCALL   HTTP_SEND
NOAUTH:
        RET

UPLOAD_FILE:
; SAVE THE UPLOAD FILE TO FILEBUFF
; E5H, 14 BYTE FILENAME , 17 BYTE
USERNAME:PASSWORD
        MOV     DPTR,#HTTP_IBUF+14
        PUSH    DPH
        PUSH    DPL
        MOV     DPTR,#FILEBUFF
        MOV     A,#E5H
        MOVX    @DPTR,A
        POP     DPL
        POP     DPH
        MOVX    A,@DPTR
        INC     DPTR
        CJNE    A,#20H,NOTEND
        LJMP    ENDUPLOAD
NOTEND:
        CJNE    A,#'f',GETUNAME
GETFILENAME:
        INC     DPTR    ; SKIP =
        MOV     R6,#HIGH(FILEBUFF+1)
        MOV     R7,#LOW(FILEBUFF+1)
GFNAMELOOP:
        LCALL   READCHAR
        CJNE    A,#00H,NOTENDGFNAME
        SJMP    ENDUPLOAD
NOTENDGFNAME:
        CJNE    A,#FFH,GFNAME
        LCALL   READCHAR
        SJMP    NOTEND
GFNAME:
        PUSH    DPH
        PUSH    DPL
        MOV     DPH,R6
        MOV     DPL,R7
        MOVX    @DPTR,A
        INC     DPTR
        CLR     A
        MOVX    @DPTR,A
```

```
        MOV     R6,DPH
        MOV     R7,DPL
        POP     DPL
        POP     DPH
        SJMP    GFNAMELOOP
GETUNAME:
        CJNE    A,#'p',GETCODE
        INC     DPTR    ; SKIP =
        MOV     R6,#HIGH(FILEBUFF+15)
        MOV     R7,#LOW(FILEBUFF+15)
GUNAMELOOP:
        LCALL   READCHAR
        CJNE    A,#00H,NOTENDGUNAME
        SJMP    ENDUPLOAD
NOTENDGUNAME:
        CJNE    A,#FFH,GUNAME
        LCALL   READCHAR

              SJMP    NOTEND
GUNAME:
        PUSH    DPH
        PUSH    DPL
        MOV     DPH,R6
        MOV     DPL,R7
        MOVX    @DPTR,A
        INC     DPTR
        CLR     A
        MOVX    @DPTR,A
        MOV     R6,DPH
        MOV     R7,DPL
        POP     DPL
        POP     DPH
        SJMP    GUNAMELOOP
GETCODE:
        INC     DPTR    ; SKIP =
        MOV     R6,#HIGH(FILEBUFF+32)
        MOV     R7,#LOW(FILEBUFF+32)
GCODELOOP:
        LCALL   READCHAR
        CJNE    A,#00H,NGCODE
        SJMP    ENDUPLOAD
NGCODE:
        PUSH    DPH
        PUSH    DPL
        MOV     DPH,R6
        MOV     DPL,R7
        MOVX    @DPTR,A
        INC     DPTR
        CLR     A
        MOVX    @DPTR,A
        MOV     R6,DPH
        MOV     R7,DPL
        POP     DPL
        POP     DPH
        SJMP    GCODELOOP
ENDUPLOAD:
        LCALL   HTTP_HEAD_SHOW
        MOV     R0,#HIGH(UPLOAD)
        MOV     R1,#LOW(UPLOAD)
        MOV     R6,#HIGH(29)
        MOV     R7,#LOW(29)
        LCALL   HTTP_SEND
        RET

READCHAR:
; RETURN A - THE CHAR (DECODE IF
NEEDED)
        MOVX    A,@DPTR
        INC     DPTR
        CJNE    A,#'%',NOTPCT
        MOVX    A,@DPTR
;       LCALL   C_OUT
        MOV     R0,A
        INC     DPTR
        MOVX    A,@DPTR
;       LCALL   C_OUT
        MOV     R1,A
        INC     DPTR
        LCALL   STRHEX
;       LCALL   C_OUT
        SJMP    ENDREADCHAR
NOTPCT:
        CJNE    A,#20H,NOTSPC
        MOV     A,#00H  ; END OF
STRING
NOTSPC:
        CJNE    A,#'+',NOTPLUS
        MOV     A,#20H
NOTPLUS:
        CJNE    A,#'&',ENDREADCHAR
        MOV     A,#FFH
ENDREADCHAR:
        LCALL   C_OUT
        RET

DEFAULT:
        ;
00000000001111111111222222222333333333344444444455555555556
        ;
01234567890123456789012345678901234567890123456789012345678901234567890
        DB  "<HTML><HEAD><TITLE>
EmWeb Script Upload..</TITLE></HEAD>
<BODY"
        DB   " BGCOLOR=#ffffaa>
<SCRIPT LANGUAGE=JavaScript>function
bgfadei"
        DB   "n(){var doc = document;var
i,j;var tstr = new String(",22H,"012345"
        DB   "6789ABCDEF",22H,");var
oldcolor = doc.bgColor;for (i=16;i>0;i--){
"
```

```
        DB   "for (j=0;j<65535;j++) ;c0 =
tstr.charAt(parseInt(i));doc.bgC"
        DB   "olor=
c0+c0+",22H,"0000",22H,";}
doc.bgColor=oldcolor;}bgfadein();
</SCRIP"
        DB   "T><CENTER><BLINK>
<FONT COLOR=RED SIZE=+3>
Development of a Re"
        DB   "configurable Embedded Web
Server</FONT></BLINK><TABLE
BORDER"
        DB   "=10><TR><TD
ALIGN=CENTER BGCOLOR=#aaaaaa>
<STRONG><FONT COLOR"
        DB   "=blue>Server : EmWeb /1.0
(Thesis)<BR>Developer : Krerk Piro"
        DB   "msopa (ID.4170680421)
<BR>Thesis Advisor : Associative Prof. "
        DB   "Boonchai
Sowanawanichakul.<BR>Department of
Computer Enginee"
        DB   "r,<BR>Faculty of Engineer,
<BR>Chulalongkorn University<BR>Ac"
        DB   "ademic Year <I>2000</I>
</FONT></STRONG></TD></TR>
</TABLE></C"
        DB   "ENTER><P>Please Specify
your filename and php lite script to"
        DB   " upload.</P><FORM
ACTION=upload METHOD=GET>
FileName: <INPUT "
        DB   "TYPE=HIDDEN NAME=f
SIZE=15 VALUE=test><BR>
UserName:Password : <INPUT
TYPE=password "
        DB   "NAME=p SIZE=15><BR>
<TEXTAREA NAME=c ROWS=15 COLS=5"
        DB   "0>Put your Code Here
</TEXTAREA><BR><INPUT
TYPE=SUBMIT VALUE="
        DB   "Upload></FORM>
<ADDRESS>Copy Right &copy;<A
HREF=mailto:pok@u"
        DB   "nforgettable.com>Krerk
Piromsopa....</A></ADDRESS></BODY>
</H"
        DB   "TML>",00H

UPLOAD:
        DB     "<H1>UPLOAD
Complete.....</H1>",00H
```

```
; PHP-LITE SCRIPT EXPRESSION
INTERPRET
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY
;
        EXTERN  C_OUT
        EXTERN  STROUT
        EXTERN  SHOWHEX
        EXTERN  HEXSTR
        EXTERN  MOVSTR
        ; Move Sequence of String
        ; INPUT  : R0 - Src H
        ;         R1 - Src L
        ;         R2 - Des H
        ;         R3 - Des L
        ;         R6 - Byte Count H
        ;         R7 - Byte Count L
        EXTERN  STR_NCX_COMP

;PAGE 0
;-------------------------
;R2 - curptr h
;R3 - curptr l
;R4 -
;R5 -
;R6 - TMP
;R7 - TMP
;-------------------------
;PAGE 1 FOR OTHER MODULE

SYMTABLE      EQU    0C000H
EXPOUT        EQU    0DF00H ; FIRST 2
BYTE DPH,DPL
EXPTMP        EQU    0DF50H
VARNAME       EQU    0DFA0H
VARBUFF       EQU    0DFA8H ;
SYMBUFF       EQU    0DFC0H ; 16 BYTE
BUFFER

        EXTERN  GETTOKEN

        PUBLIC  EXPRESSION
        PUBLIC  SYM2VARNAME
        EXTERN  GETVAR
        EXTERN  SETVAR

STOP   EQU    0
VAR    EQU    1
WHILE  EQU    2
IF     EQU    3
ECHO   EQU    4
INP    EQU    5
OUTP   EQU    6
```

```
COMMA   EQU     7
OB      EQU     8
CB      EQU     9
SCOLON  EQU     10
START   EQU     11
PLUS    EQU     12
MINUS   EQU     13
DOT     EQU     14
CONST   EQU     15
ERROR   EQU     FFH

EXPRESSION:
        PUSH    DPH
        PUSH    DPL
        LCALL   STROUT
        DB      "EXPRESSION",00H
        MOV     DPTR,#EXPOUT
        MOV     A,#HIGH(EXPOUT+2)
        MOVX    @DPTR,A
        MOV     A,#LOW(EXPOUT+2)
        INC     DPTR
        MOVX    @DPTR,A
        LCALL   GETTOKEN
        LCALL   LOADVAL
        LCALL   EXPTMP2EXPOUT
EXPLOOP:
        LCALL   GETTOKEN        ; get
operator
        ; CHECK END OF PROCESS
        CJNE    A,#CB,NOTCB
        SJMP    ENDEXP
NOTCB:
        CJNE    A,#COMMA,NOTCOMMA
        SJMP    ENDEXP
NOTCOMMA:
        CJNE    A,#SCOLON,NOTSCOLON
        SJMP    ENDEXP
NOTSCOLON:
        PUSH    A
        LCALL   GETTOKEN
        LCALL   LOADVAL
        POP     A
        CJNE    A,#PLUS,NOTPLUS
        ; ADD EXPTMP TO EXPOUT
        MOV     DPTR,#EXPTMP+1
        MOVX    A,@DPTR
        MOV     R0,A
        MOV     DPTR,#EXPOUT+3
        MOVX    A,@DPTR
        ADD     A,R0
        MOVX    @DPTR,A
        MOV     DPTR,#EXPTMP
        MOVX    A,@DPTR
        MOV     R0,A
        MOV     DPTR,#EXPOUT+2
        MOVX    A,@DPTR
        ADDC    A,R0
        MOVX    @DPTR,A
        SJMP    EXPLOOP
NOTPLUS:
        CJNE    A,#MINUS,NOTMINUS
        ; SUBB EXPOUT WITH EXPTMP
        MOV     DPTR,#EXPTMP+1
        MOVX    A,@DPTR
        MOV     R0,A
        MOV     DPTR,#EXPOUT+3
        MOVX    A,@DPTR
        CLR     C
        SUBB    A,R0
        MOVX    @DPTR,A
        MOV     DPTR,#EXPTMP
        MOVX    A,@DPTR
        MOV     R0,A
        MOV     DPTR,#EXPOUT+2
        MOVX    A,@DPTR
        SUBB    A,R0
        MOVX    @DPTR,A
        SJMP    EXPLOOP
NOTMINUS:
        CJNE    A,#DOT,NOTDOT
        ; CONCAT EXPOUT2EXPTMP
        LCALL   EXPTMP2EXPOUT
        SJMP    EXPLOOP
NOTDOT:
        MOV     A,#ERROR
;       SJMP    ENDEXP
ENDEXP:
        PUSH    A
        MOV     DPTR,#EXPOUT+3
        MOVX    A,@DPTR
        LCALL   SHOWHEX
        POP     A
        POP     DPL
        POP     DPH
        RET

LOADVAL:
        CJNE    A,#VAR,NOTVAR
        LCALL   SYM2VARNAME
        LCALL   GETVAR
        LCALL   VAR2EXPTMP
        SJMP    ENDLOADVAL
NOTVAR:
        LCALL   SYM2EXPTMP
ENDLOADVAL:
        RET

SYM2EXPTMP:
        PUSH    DPH
        PUSH    DPL
        SETB    RS0
        MOV     R0,#HIGH(SYMBUFF)
        MOV     R1,#LOW(SYMBUFF)
        MOV     R2,#HIGH(EXPTMP)
```

```
        MOV     R3,#LOW(EXPTMP)
        MOV     R6,#0
        MOV     R7,#20H
        LCALL   MOVSTR
        CLR     RS0
        POP     DPL
        POP     DPH
        RET

EXPTMP2EXPOUT:
        PUSH    DPH
        PUSH    DPL
        SETB    RS0
        MOV     DPTR,#EXPOUT
        MOVX    A,@DPTR
        MOV     R0,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R1,A
        MOV     DPTR,#EXPTMP
; DUMMY READ 1 BYTE IN CASE OF NUM
        MOVX    A,@DPTR
        INC     DPTR
        MOV     R2,DPH
        MOV     R3,DPL
        MOV     DPH,R0
        MOV     DPL,R1
        MOVX    @DPTR,A
EXPT2OLOOP:
        INC     DPTR
        PUSH    DPH
        PUSH    DPL
        MOV     DPH,R2
        MOV     DPL,R3
        MOVX    A,@DPTR
        INC     DPTR
        MOV     R2,DPH
        MOV     R3,DPL
        POP     DPL
        POP     DPH
        MOVX    @DPTR,A
        CJNE    A,#0,EXPT2OLOOP
        MOV     R0,DPH
        MOV     R1,DPL
        MOV     DPTR,#EXPOUT
        MOV     A,R0
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,R1
        MOVX    @DPTR,A

        CLR     RS0
        POP     DPL
        POP     DPH
        RET

VAR2EXPTMP:
        PUSH    DPH
        PUSH    DPL
        SETB    RS0
        MOV     R0,#HIGH(VARBUFF)
        MOV     R1,#LOW(VARBUFF)
        MOV     R2,#HIGH(EXPTMP)
        MOV     R3,#LOW(EXPTMP)
        MOV     R6,#0
        MOV     R7,#24
        LCALL   MOVSTR
        CLR     RS0
        POP     DPL
        POP     DPH
        RET

SYM2VARNAME:
        PUSH    DPH
        PUSH    DPL
        SETB    RS0
        MOV     R0,#HIGH(SYMBUFF)
        MOV     R1,#LOW(SYMBUFF)
        MOV     R2,#HIGH(VARNAME)
        MOV     R3,#LOW(VARNAME)
        MOV     R6,#0
        MOV     R7,#8
        LCALL   MOVSTR
        CLR     RS0
        POP     DPL
        POP     DPH
        RET
```

```
; PHP-LITE SCRIPT EXPRESSION
INTERPRET
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY
;
        EXTERN  C_OUT
        EXTERN  STROUT

             PUBLIC  ISNUM
        PUBLIC  ISCHAR

ISNUM:
; CHECK IF R6 IS CHAR
; RETURN A = 0 IF FALSE
;       A = 1 IF [R6] IS NUM
          MOV     A,R6
          SUBB    A,#'0'
          JB      ACC.7,ISNOTNUM
          SUBB    A,#10
          JNB     ACC.7,ISNOTNUM
          MOV     A,#1
          SJMP    ENDISNUM
ISNOTNUM:
          CLR     A
ENDISNUM:
          RET

ISCHAR:
; CHECK IF R6 IS CHAR
; RETURN A = 0 IF FALSE
;       A = 1 IF [R6] IS CHAR
          MOV     A,R6
          SUBB    A,#41H
          JNB     ACC.7,CHKISCHAR
          CLR     A
          SJMP    ENDISCHAR
CHKISCHAR:
          MOV     A,#1
ENDISCHAR:
          RET
; PHP-LITE SCRIPT INTERPRET
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY
          EXTERN  STROUT
          EXTERN  HEXSTR
          EXTERN  SHOWHEX
          EXTERN  SETVAR
          EXTERN  GETVAR
          EXTERN  SYM2VARNAME
          EXTERN  EXPRESSION
          EXTERN  GETTOKEN
          EXTERN  SEMICOLONEND
          EXTERN  DOSTATEMENTS

STOP    EQU     0
VAR     EQU     1
WHILE   EQU     2
IF      EQU     3
ECHO    EQU     4
INP     EQU     5
OUTP    EQU     6
COMMA   EQU     7
OB      EQU     8
CB      EQU     9
SCOLON  EQU     10
START   EQU     11
PLUS    EQU     12
MINUS   EQU     13
DOT     EQU     14
CONST   EQU     15
EQ      EQU     16
STRCONV EQU     17
ERROR   EQU     FFH


;PAGE 0
;---------------------------
;R2 - curptr h
;R3 - curptr l
;R4 -
;R5 -
;R6 - TMP
;R7 - TMP
;---------------------------
;PAGE 1 FOR OTHER MODULE

VARNAME         EQU     0DFA0H ; 256
BYTE BUFFER
VARBUFF         EQU     0DFA8H
SYMBUFF         EQU     0DFC0H
EXPOUT          EQU     0DF00H ; FIRST 2
BYTE DPH,DPL
ASSBUFF         EQU     0DF90H ; ASSIGN
BUFFER

             PUBLIC  DOOUTP
          PUBLIC  DOINP
          PUBLIC  DOWHILE
          PUBLIC  DOIF
          PUBLIC  DOSTRCONV

DOOUTP:
; OUTPUT DATA TO PORT  -> OUTP
(PORTEXP,VALEXP);
          LCALL   STROUT
          DB      "OUTP",00H
```

```
        LCALL   GETTOKEN
        CJNE    A,#OB,OUTPNOTOB
        LCALL   EXPRESSION
        MOV     DPTR,#EXPOUT+2
        MOVX    A,@DPTR
        PUSH    A
        INC     DPTR
        MOVX    A,@DPTR
        PUSH    A
;       LCALL   GETTOKEN
;       CJNE
A,#COMMA,OUTPNOTCOMMA
        LCALL   EXPRESSION
        MOV     DPTR,#EXPOUT+3
        MOVX    A,@DPTR
        POP     DPL
        POP     DPH
        MOVX    @DPTR,A
        SJMP    OUTPNOTOB
;OUTPNOTCOMMA:
;       POP     A
;       POP     A
OUTPNOTOB:
        LJMP    SEMICOLONEND

DOINP:
; INPUT DATA FROM PORT  -> INP
(PORTEXP,VAR);
        LCALL   STROUT
        DB      "INP",00H
        LCALL   GETTOKEN
        CJNE    A,#OB,INPNOTOB
        LCALL   EXPRESSION
        MOV     DPTR,#EXPOUT+2
        MOVX    A,@DPTR
        PUSH    A
        INC     DPTR
        MOVX    A,@DPTR
        PUSH    A
        LCALL   GETTOKEN
        CJNE    A,#VAR,INPNOTVAR
        LCALL   SYM2VARNAME
        POP     DPL
        POP     DPH
        MOVX    A,@DPTR
        PUSH    A
        MOV     DPTR,#VARBUFF
        CLR     A
        MOVX    @DPTR,A
        INC     DPTR
        POP     A
        MOVX    @DPTR,A
;       LCALL   SHOWHEX
        CLR     A
        INC     DPTR
        MOVX    @DPTR,A
        LCALL   SETVAR
        SJMP    INPNOTOB
INPNOTVAR:
        POP     A
        POP     A
INPNOTOB:
        LJMP    SEMICOLONEND

DOSTRCONV:
; STR(VAR) CONVERT NUM TO STR
        LCALL   GETTOKEN
        CJNE    A,#OB,ENDSTRCONV
        LCALL   GETTOKEN
        CJNE    A,#VAR,ENDSTRCONV
        LCALL   SYM2VARNAME
        LCALL   GETVAR
        SETB    RS0
        MOV     DPTR,#VARBUFF+1
        MOVX    A,@DPTR
        LCALL   HEXSTR
        MOV     DPTR,#VARBUFF+2
        MOV     A,R0
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,R1
        MOVX    @DPTR,A
        INC     DPTR
        CLR     A
        MOVX    @DPTR,A
        MOV     DPTR,#VARBUFF
        MOVX    A,@DPTR
        LCALL   HEXSTR
        MOV     A,R0
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,R1
        MOVX    @DPTR,A
        CLR     RS0
        LCALL   SETVAR
ENDSTRCONV:
        LJMP    SEMICOLONEND

DOWHILE:
;
        LCALL   STROUT
        DB      "WHILE",00H
        LCALL   GETTOKEN        ; SKIP '('
        MOV     A,R2            ; SAVE CURR
POINTER
        PUSH    A
        MOV     A,R3
        PUSH    A
WHILELOOP:
        LCALL   EXPRESSION
        MOV     DPTR,#EXPOUT+3
        MOVX    A,@DPTR
        CJNE    A,#0,WHILENOTEND
        POP     A
```

```
        POP     A
STOPEND:
        LCALL   GETTOKEN
        CJNE    A,#STOP,STOPEND
        RET
WHILENOTEND:
        LCALL   GETTOKEN
        CJNE    A,#START,STOPEND
        LCALL   DOSTATEMENTS
        POP     A
        MOV     R3,A
        POP     A
        MOV     R2,A
        PUSH    A
        MOV     A,R3
        PUSH    A
        SJMP    WHILELOOP

DOIF:
;
        LCALL   STROUT
        DB      "IF",00H

            RET
```

```
; PHP-LITE SCRIPT INTERPRET GET
TOKEN (LEXICAL ANALYZER)
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY
        EXTERN  SHOWHEX
        EXTERN  STROUT
        EXTERN  C_IN
        EXTERN  C_OUT

            EXTERN  SKPDLIM
        EXTERN  GETDAT
        EXTERN  ISCHAR
        EXTERN  ISNUM
        EXTERN  STRHEX
        EXTERN  STR_CX_COMP

            PUBLIC  GETTOKEN

STOP    EQU     0
VAR     EQU     1
WHILE   EQU     2
IF      EQU     3
ECHO    EQU     4
INP     EQU     5
OUTP    EQU     6
COMMA   EQU     7
OB      EQU     8
CB      EQU     9
SCOLON  EQU     10
START   EQU     11
PLUS    EQU     12
MINUS   EQU     13
DOT     EQU     14
CONST   EQU     15
EQ      EQU     16
STRCONV EQU     17
ERROR   EQU     FFH


SYMBUFF         EQU     0DFC0H

GETTOKEN:
; (LEXICAL ANALYZER)
; GET THE SYMBOL TO STR BUFFER
; AND DETERMINE THE TOKEN TYPE to A
        PUSH    DPH
        PUSH    DPL
;       LCALL   STROUT
;       DB      0DH,0AH,"TOKEN:",00H
        MOV     DPTR,#SYMBUFF
        LCALL   SKPDLIM ; SKIP
DELIMETER
;       MOV     A,R6
```

```
;       LCALL  C_OUT
        CJNE   R6,#00,NOTEOF
        SJMP   TOSTOP
NOTEOF:
        CJNE   R6,#'}',NOTSTOP
TOSTOP:
        MOV    A,#STOP
        SJMP   ENDTOKEN
NOTSTOP:
        CJNE   R6,#',',NOTCOMMA
        MOV    A,#COMMA
        SJMP   ENDTOKEN
NOTCOMMA:
        CJNE   R6,#'(',NOTOB
        MOV    A,#OB
        SJMP   ENDTOKEN
NOTOB:
        CJNE   R6,#')',NOTCB
        MOV    A,#CB
        SJMP   ENDTOKEN
NOTCB:
        CJNE   R6,#';',NOTSCOLON
        MOV    A,#SCOLON
        SJMP   ENDTOKEN
NOTSCOLON:
        CJNE   R6,#'{',NOTSTART
        MOV    A,#START
        SJMP   ENDTOKEN
NOTSTART:
        CJNE   R6,#'+',NOTPLUS
        MOV    A,#PLUS
        SJMP   ENDTOKEN
NOTPLUS:
        CJNE   R6,#'-',NOTMINUS
        MOV    A,#MINUS
        SJMP   ENDTOKEN
NOTMINUS:
        CJNE   R6,#'.',NOTDOT
        MOV    A,#DOT
        SJMP   ENDTOKEN
NOTDOT:
        CJNE   R6,#'=',NOTEQ
        MOV    A,#EQ
        SJMP   ENDTOKEN
NOTEQ:
        CJNE   R6,#22H,NOTSTRCONST
STRCONSTLOOP:
        LCALL  GETDAT
        CJNE
R6,#22H,NOTENDSTRCONST
        SJMP   ENDSTRCONST
NOTENDSTRCONST:
        MOV    A,R6
        MOVX   @DPTR,A
        INC    DPTR
        CLR    A
        MOVX   @DPTR,A ; TERMINATE
RESULT
        SJMP   STRCONSTLOOP
ENDSTRCONST:
        MOV    A,#CONST
        SJMP   ENDTOKEN
ENDTOKEN:
        LCALL  SHOWHEX
;       PUSH   A
;       LCALL  C_IN
;       POP    A
        POP    DPL
        POP    DPH
        RET
NOTSTRCONST:
        CJNE   R6,#'0',NOTHEXCONST
        CJNE   R7,#'x',NOTHEXCONST
        LCALL  GETDAT  ; SKIP x
        LCALL  GETDAT
        MOV    A,R6
        MOV    R0,A
        LCALL  GETDAT
        MOV    A,R6
        MOV    R1,A
        LCALL  STRHEX
        MOVX   @DPTR,A
        INC    DPTR
        LCALL  GETDAT
        MOV    A,R6
        MOV    R0,A
        LCALL  GETDAT
        MOV    A,R6
        MOV    R1,A
        LCALL  STRHEX
        MOVX   @DPTR,A
        INC    DPTR
        CLR    A
        MOVX   @DPTR,A
        MOV    A,#CONST
ENDTOKEN1:
        SJMP   ENDTOKEN
NOTHEXCONST:
        LCALL  ISNUM
        CJNE   A,#1,NOTNUMCONST
        CLR    A
        MOV    R0,A
        MOV    R1,A
NUMCONSTLOOP:
        XCH    A,R1
        MOV    B,A
        MOV    A,#10
        MUL    AB
        XCH    A,R1
        MOV    A,B
        XCH    A,R0
        MOV    B,A
        MOV    A,#10
```

```
        MUL     AB
        ADD     A,R0
        MOV     R0,A
;
        MOV     A,R6
        ANL     A,#0FH
        ADD     A,R1
;       LCALL   SHOWHEX
        MOV     R1,A
        MOV     A,R0
        ADDC    A,#0
;       LCALL   SHOWHEX
        MOV     R0,A
; CHECK IF END
        MOV     A,R6
        PUSH    A
        MOV     A,R7
        MOV     R6,A
        LCALL   ISNUM
        MOV     R0,A
        POP     A
        MOV     R6,A
        MOV     A,R0
        CJNE    A,#1,ENDNUMCONST
        LCALL   GETDAT
        SJMP    NUMCONSTLOOP
ENDNUMCONST:
        MOV     A,R0
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,R1
        MOVX    @DPTR,A
        INC     DPTR
        CLR     A
        MOVX    @DPTR,A
        MOV     A,#CONST
        SJMP    ENDTOKEN1
NOTNUMCONST:
        CJNE    R6,#'$',NOTVAR
        LCALL   GETDAT
        LCALL   DATISCHAR
        MOV     A,#VAR
        SJMP    ENDTOKEN1
NOTVAR:
        LCALL   DATISCHAR
        SETB    RS0
        MOV     R1,#HIGH(SYMBUFF)
        MOV     R2,#LOW(SYMBUFF)
        LCALL   STR_CX_COMP
        DB      "while",00H
        MOV     A,R0
        CLR     RS0
        CJNE    A,#0,NOTWHILE
        MOV     A,#WHILE
ENDTOKEN2:
        SJMP    ENDTOKEN1
NOTWHILE:
        SETB    RS0
        MOV     R1,#HIGH(SYMBUFF)
        MOV     R2,#LOW(SYMBUFF)
        LCALL   STR_CX_COMP
        DB      "if",00H
        MOV     A,R0
        CLR     RS0
        CJNE    A,#0,NOTIF
        MOV     A,#IF
        SJMP    ENDTOKEN2
NOTIF:
        SETB    RS0
        MOV     R1,#HIGH(SYMBUFF)
        MOV     R2,#LOW(SYMBUFF)
        LCALL   STR_CX_COMP
        DB      "echo",00H
        MOV     A,R0
        CLR     RS0
        CJNE    A,#0,NOTECHO
        MOV     A,#ECHO
        SJMP    ENDTOKEN2
NOTECHO:
        SETB    RS0
        MOV     R1,#HIGH(SYMBUFF)
        MOV     R2,#LOW(SYMBUFF)
        LCALL   STR_CX_COMP
        DB      "inp",00H
        MOV     A,R0
        CLR     RS0
        CJNE    A,#0,NOTINP
        MOV     A,#INP
        SJMP    ENDTOKEN2
NOTINP:
        SETB    RS0
        MOV     R1,#HIGH(SYMBUFF)
        MOV     R2,#LOW(SYMBUFF)
        LCALL   STR_CX_COMP

        DB      "outp",00H
        MOV     A,R0
        CLR     RS0
        CJNE    A,#0,NOTOUTP
        MOV     A,#OUTP
        SJMP    ENDTOKEN2
NOTOUTP:
        SETB    RS0
        MOV     R1,#HIGH(SYMBUFF)
        MOV     R2,#LOW(SYMBUFF)
        LCALL   STR_CX_COMP
        DB      "str",00H
        MOV     A,R0
        CLR     RS0
        CJNE    A,#0,NOTSTRCONV
        MOV     A,#STRCONV
        SJMP    ENDTOKEN2
NOTSTRCONV:
        MOV     A,#ERROR
```

```
        SJMP    ENDTOKEN2

CHARLOOP:
        LCALL   GETDAT
DATISCHAR:
        MOV     A,R6
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,R6
        PUSH    A
        MOV     A,R7
        MOV     R6,A
        LCALL   ISCHAR
        MOV     R0,A
        POP     A
        MOV     R6,A
        MOV     A,R0
        CJNE    A,#0,CHARLOOP
        CLR     A       ; TERMINATE
STRING
        MOVX    @DPTR,A
        RET
```

```
; PHP-LITE SCRIPT VARIABLE HANDLER
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY
; VARTABLE FORMAT
; 8 BYTES - SYMNAME , 24 BYTES - VALUE
        EXTERN  C_OUT
        EXTERN  STROUT
        EXTERN  SHOWHEX
        EXTERN  HEXSTR
        EXTERN  MOVSTR
        ; Move Sequence of String
        ; INPUT  : R0 - Src H
        ;         R1 - Src L
        ;         R2 - Des H
        ;         R3 - Des L
        ;         R6 - Byte Count H
        ;         R7 - Byte Count L
        EXTERN  STR_NCX_COMP

        PUBLIC  GETVAR
        PUBLIC  SETVAR
        PUBLIC  INITSYMTABLE

SYMTABLE        EQU     0C000H
VARNAME         EQU     0DFA0H
VARBUFF         EQU     0DFA8H ;

GETVAR:
; GET DATA FROM SYMBOL TABLE
; Return 0 in case not found
        PUSH    DPH
        PUSH    DPL
        PUSH    A
        SETB    RS0
; CLEAR VALUE
        MOV     R6,#24
        MOV     DPTR,#VARBUFF
        CLR     A
CLRVALLOOP:
        MOVX    @DPTR,A
        INC     DPTR
        DJNZ    R6,CLRVALLOOP
        MOV     DPTR,#SYMTABLE-20H
GETVARFINDLOOP:
        MOV     A,DPL
        ADD     A,#20H
        MOV     DPL,A
        MOV     A,DPH
        ADDC    A,#0
        MOV     DPH,A
        MOVX    A,@DPTR
        CJNE    A,#00,GETFINDNEXT
; ADD NEW
```

```
        SJMP    GETVALUE
GETFINDNEXT:
        PUSH    DPH
        PUSH    DPL
        MOV     R1,DPH
        MOV     R2,DPL
        MOV     R3,#HIGH(VARNAME)
        MOV     R4,#LOW(VARNAME)
        LCALL   STR_NCX_COMP
        POP     DPL
        POP     DPH
        CJNE    R0,#0,GETVARFINDLOOP
GETVALUE:
        MOV     R2,#HIGH(VARNAME)
        MOV     R3,#LOW(VARNAME)
        MOV     R0,DPH
        MOV     R1,DPL
        MOV     R6,#0
        MOV     R7,#1FH
        LCALL   MOVSTR
        CLR     RS0
        POP     A
        POP     DPL
        POP     DPH
        RET

INITSYMTABLE:
        PUSH    DPH
        PUSH    DPL
        PUSH    A
        MOV     A,R6
        PUSH    A
        CLR     A
        MOV     R6,#255
        MOV     DPTR,#SYMTABLE
INITSYMTLOOP:
        MOVX    @DPTR,A
        INC     DPTR
        DJNZ    R6,INITSYMTLOOP
        POP     A
        MOV     R6,A
        POP     A
        POP     DPL
        POP     DPH
        RET

SETVAR:
; SET DATA IN SYMBOL TABLE
; ADD new in case not found
        PUSH    DPH
        PUSH    DPL
        PUSH    A
;       LCALL   STROUT
;       DB      "SETVAR",00H
        SETB    RS0
        MOV     DPTR,#SYMTABLE-20H
SETVARFINDLOOP:
        MOV     A,DPL
        ADD     A,#20H
        MOV     DPL,A
        MOV     A,DPH
        ADDC    A,#0
        MOV     DPH,A
        MOVX    A,@DPTR
        CJNE    A,#00,FINDNEXT
; ADD NEW
        SJMP    SETVALUE
FINDNEXT:
        PUSH    DPH
        PUSH    DPL
        MOV     R1,DPH
        MOV     R2,DPL
        MOV     R3,#HIGH(VARNAME)
        MOV     R4,#LOW(VARNAME)
        LCALL   STR_NCX_COMP
        POP     DPL
        POP     DPH
        CJNE    R0,#0,SETVARFINDLOOP
SETVALUE:
        MOV     R0,#HIGH(VARNAME)
        MOV     R1,#LOW(VARNAME)
        MOV     R2,DPH
        MOV     R3,DPL
        MOV     R6,#0
        MOV     R7,#1FH
        LCALL   MOVSTR
        CLR     RS0
        POP     A
        POP     DPL
        POP     DPH
        RET
```

```
; PHP-LITE INTERPRET ENVIRONMENT
INIT
; THESIS : DEVELOPMENT OF A
RECONFIGURABLE EMBEDDED WEB
SERVER
; BY KRERK PIROMSOPA
;   COMPUTER ENGINEER.
;   CHULALONGKORN UNIVERSITY
        EXTERN  STR_FIND_CHR
        EXTERN  SETVAR
        EXTERN  STRHEX
        PUBLIC  PHPENV

VARNAME       EQU    0DFA0H ; 256
BYTE BUFFER
VARBUFF       EQU    0DFA8H

HTTPIBUF      EQU    0D000H

PHPENV:
        MOV    DPTR,#HTTPIBUF+6
FINDQ:
        INC    DPTR
        MOVX   A,@DPTR
; FIND '?' OR IF FOUND ' ' THEN END
        CJNE   A,#' ',NOTEND
ENDPHPENV:

              RET
NOTEND:
        CJNE   A,#'?',FINDQ
GETVARNAME:
        MOV    R2,#HIGH(VARNAME)
        MOV    R3,#LOW(VARNAME)

VARNAMELOOP:
        INC    DPTR
        MOVX   A,@DPTR
        CJNE   A,#'=',NOTEQ
        SJMP   GETVALUE
NOTEQ:
        PUSH   DPH
        PUSH   DPL
        MOV    DPH,R2
        MOV    DPL,R3
        MOVX   @DPTR,A
        INC    DPTR
        CLR    A
        MOVX   @DPTR,A
        MOV    R2,DPH
        MOV    R3,DPL
        POP    DPL
        POP    DPH
        SJMP   VARNAMELOOP
GETVALUE:
        MOV    R2,#HIGH(VARBUFF)
        MOV    R3,#LOW(VARBUFF)

GETVALUELOOP:
        INC    DPTR
        MOVX   A,@DPTR
        CJNE   A,#' ',NOTENDVAL
        LCALL  SETVAR
        SJMP   ENDPHPENV
NOTENDVAL:
        CJNE   A,#'&',NOTAMP
        LCALL  SETVAR
        SJMP   GETVARNAME
NOTAMP:
        CJNE   A,#'%',NOTENCODE
        INC    DPTR
        MOVX   A,@DPTR
        MOV    R0,A
        INC    DPTR
        MOVX   A,@DPTR
        MOV    R1,A
        LCALL  STRHEX
NOTENCODE:
        PUSH   DPH
        PUSH   DPL
        MOV    DPH,R2
        MOV    DPL,R3
        MOVX   @DPTR,A
        INC    DPTR
        CLR    A
        MOVX   @DPTR,A
        MOV    R2,DPH
        MOV    R3,DPL
        POP    DPL
        POP    DPH
        SJMP   GETVALUELOOP
```

```
; TCP TIME OUT
; TIMER ROUTINE

     PUBLIC  INIT_TIMER
     PUBLIC  ISR_TIMER
     PUBLIC  START_TIMER
     PUBLIC  STOP_TIMER
     EXTERN  C_OUT
     EXTERN  TCP_CLOSE

; TCP STAT
STAT   EQU    23H
;
CNT0   EQU    28H
CNT1   EQU    29H

INIT_TIMER:
     ANL    TMOD,#F0H
     ORL    TMOD,#01H
     SETB   ET0
     MOV    A,#'>'
     LCALL  C_OUT
     RET

ISR_TIMER:
     PUSH   A
     INC    CNT0
     MOV    A,CNT0
     CJNE   A,#15,NISR1
     MOV    CNT0,#0
     INC    CNT1
     MOV    A,CNT1
     CJNE   A,#15,NISR1
     CLR    TR0
     MOV    A,#'X'
     LCALL  C_OUT
     LCALL  TCP_CLOSE
     MOV    STAT,#0
NISR1:
     POP    A
     RETI

START_TIMER:
     MOV    CNT0,#0
     MOV    CNT1,#0
     MOV    TH0,#00H
     MOV    TL0,#00H
     SETB   TR0
     RET

STOP_TIMER:
     PUSH   A
     CLR    TR0
     MOV    A,#'>'
     LCALL  C_OUT
     POP    A
     RET
```

ประวัติผู้เขียนวิทยานิพนธ์

นายเกริก ภิรมย์โสภา เกิดเมื่อวันที่ 17 กุมภาพันธ์ พ.ศ. 2521 ที่อำเภอปทุมวัน จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาปริญญาตรีวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2541 ด้วยระยะเวลาสามปีครึ่ง และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตร์มหาบัณฑิต ที่จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ. 2541