การพัฒนาระบบเว็บเซิร์ฟเวอร์แบบฝังตัวที่สามารถจัดรูปลักษณ์ใหม่ได้

นายเกริก ภิรมย์โสภา

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2543
ISBN 974-346-285-6

DEVELOPMENT OF A RECONFIGURABLE EMBEDDED WEB SERVER

Mr Krerk Piromsopa

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering in Computer Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2000

ISBN 974-346-285-6

หัวข้อวิทยานิพนธ์	การพัฒนาระบบเว็บเซิร์ฟเวอร์แบบฝังตัวที่สามารถจัดรูปลักษณ์ใหม่ได้
โดย	นายเกริก ภิรมย์โสภา
ภาควิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	ผศ.บุญชัย โสวรรณวณิชกุล

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

…… คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สมศักดิ์ ปัญญาแก้ว)

คณะกรรมการสอบวิทยานิพนธ์

…… ประธานกรรมการ
(อาจารย์ ดร.ฐิต ศิริบูรณ์)

…… อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์บุญชัย โสวรรณวณิชกุล)

…… กรรมการ
(อาจารย์ ดร.อาทิตย์ ทองทักษ์)

…… กรรมการ
(อาจารย์สุเมธ อังคะศิริกุล)

เกริก ภิรมย์โสภา : การพัฒนาระบบเว็บเซิร์ฟเวอร์แบบฝังตัวที่สามารถจัดรูปลักษณ์ใหม่ได้. (DEVELOPMENT OF A RECONFIGURABLE EMBEDDED WEB SERVER) อ. ที่ปรึกษา : ผศ.บุญชัย โสวรรณวณิชกุล, 142 หน้า. ISBN 974-346-285-6.

ระบบเว็บเซิร์ฟเวอร์แบบฝังตัว (Embedded web server) โดยทั่วไปสามารถปรับเปลี่ยน หรือแก้ไขซอฟต์แวร์ภายในได้ยากลำบาก ทั้งนี้เนื่องจากผู้พัฒนามักพัฒนาระบบที่เฉพาะเจาะจงและใช้วิธีการของตนเองในการควบคุมการทำงานของระบบ จากความไม่เป็นมาตรฐานดังกล่าวทำให้ผู้พัฒนาไม่สามารถสร้างระบบเว็บเซิร์ฟเวอร์ที่ทำงานได้กับอุปกรณ์หลากหลายรูปแบบโดยอาศัยเว็บเซิร์ฟเวอร์ตัวเดียวกัน ด้วยเหตุนี้จึงเกิดแนวทางในการวิจัยเพื่อพัฒนาระบบเว็บเซิร์ฟเวอร์ที่สามารถปรับเปลี่ยนแก้ไขรูปลักษณ์หรือซอฟต์แวร์ภายในให้เข้ากับสภาวะแวดล้อมและอุปกรณ์ต่างๆ ที่ระบบต่อพ่วงอยู่ได้โดยง่าย

งานวิจัยนี้ครอบคลุมการออกแบบและพัฒนาระบบเว็บเซิร์ฟเวอร์แบบฝังตัวด้วยไมโครคอนโทรลเลอร์ขนาดเล็กโดยเลือกใช้ไมโครคอนโทรลเลอร์ขนาด 8 บิต MCS-51 ร่วมกับ DP83902 ซึ่งเป็นหน่วยควบคุมการเชื่อมต่อระบบเครือข่ายตามมาตรฐานอีเทอร์เน็ต และระบบเชื่อมต่ออุปกรณ์ต่อพ่วงทั่วไป พร้อมทั้งพัฒนาซอฟต์แวร์เพื่อการเชื่อมต่อระบบเครือข่ายบนโพรโทคอล TCP/IP และทำหน้าที่เป็นเว็บเซิร์ฟเวอร์ รวมถึงการสร้างระบบรักษาความปลอดภัยโดยใช้การพิสูจน์ตัวจริงแบบเบื้องต้น (Basic Authentication) และพัฒนาภาษาสคริปต์อย่างง่ายเพื่อใช้ควบคุมอุปกรณ์ต่อพ่วงต่างๆ พร้อมกับตัวแปลภาษาซึ่งมีโครงสร้างคล้ายกับภาษา PHP ตามลำดับ

ระบบสามารถเชื่อมต่อกับระบบเครือข่ายตามมาตรฐานอีเทอร์เน็ตบนโพรโทคอล TCP/IP ได้อย่างมีประสิทธิภาพ นอกจากนี้ระบบยังสามารถควบคุมและจัดรูปลักษณ์ใหม่ โดยอาศัยการปรับเปลี่ยนสคริปต์ผ่านเว็บและควบคุมสิทธิการเรียกใช้งานภาษาสคริปต์ต่างๆ ด้วยชื่อและรหัสผ่านที่ตั้งค่าไว้ก่อนได้อย่างเป็นที่หน้าพอใจ

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....
สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา .....
ปีการศึกษา 2543 ลายมือชื่ออาจารย์ที่ปรึกษาร่วม .....

4170680421 : MAJOR COMPUTER ENGINEERING

KEY WORD: EMBEDDED WEB SERVER / TCP/IP / INTERNET / WEB DEVICE / NETWORK

KRERK PIROMSOPA : DEVELOPMENT OF A RECONFIGURABLE EMBEDDED WEB SERVER. THESIS ADVISOR : ASSOCIATIVE PROFESSOR BOONCHAI SOWANNAWANICHAKUL, 142 pp. ISBN 974-376-285-6.

Embedded web server is difficult to update or reconfigure the internal software. As each developer or vendor usually creates his own proprietary systems and methods. As a result, the same embedded web server cannot be reprogrammed to meet various styles of functions. To reduce this complexity, this thesis mainly focuses on how to develop an embedded web server that can be reconfigure or update the control software to easily meet the control environment.

The thesis covers design and implementation of the embedded web server with a small microcontroller, the 8-bit microcontroller MCS-51 with DP83902 as the Network Interface Controller of the Ethernet LAN and generic device interfacing subsystems. The software developments comprise a subset of TCP/IP network protocol, web server, the basic authentication system and the extra-ordinary PHP style scripting language interpreter for inquiry and control of the system.

The system in this thesis can be connected to the Ethernet Network and can work with the TCP/IP protocol. The system can also control and monitor any device with an ability to reconfigure and update the software online using the scripting language. Moreover, the system is able to control the right of user to run any script by using predefined user and password.

Department COMPUTER ENGINEERING
Field of study COMPUTER ENGINEERING
Academic year 2000

Student's signature .....
Advisor's signature .....
Co-advisor's signature .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ ด้วยความช่วยเหลืออย่างดียิ่งของผู้ช่วยศาสตราจารย์บุญชัย โสวรรณวณิชกุล อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งได้ให้คำแนะนำและข้อคิดเห็นต่างๆ ในการวิจัยมาด้วยดีตลอด

ท้ายนี้ ผู้วิจัยใคร่ขอกรอบขอบพระคุณบิดามารดา ซึ่งให้กำเนิดและสนับผู้วิจัยมาโดยเสมอมา และขอขอบคุณคณาจารย์ในภาควิชาวิศวกรรมคอมพิวเตอร์ทุกท่าน รุ่นพี่ รุ่นน้อง และเพื่อนสมาชิกผู้ร่วมศึกษาในระดับบัณฑิตศึกษาของภาควิชาทุกท่าน ที่ช่วยให้บรรยากาศในการเรียนมีความรื่นรมย์สนุกสนาน

สารบัญ

สารบัญตาราง

สารบัญภาพ

บทที่ 1
บทนำ

ความเป็นมาและคำจำกัดความต่างๆ ที่กล่าวถึงในบทนี้ เพื่อเป็นข้อแนะนำประกอบการอ่านวิทยานิพนธ์ ทั้งนี้มุ่งหวังเพื่อให้ผู้อ่านสามารถเข้าใจถึงส่วนประกอบ และแนวทางในการดำเนินการวิจัยได้โดยง่าย โดยจะกล่าวถึง คำจำกัดความและข้อควรทราบต่างๆ ที่เกี่ยวข้องกับการพัฒนาระบบเว็บเซิร์ฟเวอร์แบบฝังตัว

1.1. ความเป็นมาและความสำคัญของปัญหา

จากอัตราการเจริญเติบโตของระบบเครือข่ายอินเทอร์เน็ตในปัจจุบัน พบว่าจำนวนของอุปกรณ์และเครื่องใช้ต่างๆ ที่ต่อพ่วงเข้ากับระบบเครือข่ายข่ายนั้น มีการเจริญเติบโตอย่างรวดเร็ว มีการคาดการว่าจำนวนบุคคลที่เชื่อมต่อเข้าสู่ระบบเครือข่ายอินเทอร์เน็ตผ่านทางโทรศัพท์ จะถูกแทนที่ด้วยอุปกรณ์และเครื่องใช้ต่างๆในไม่ช้า (Michael O'Brien, 1999) ผู้ผลิตอุปกรณ์สื่อสารและอุปกรณ์ควบคุมต่างๆ ได้หันมาใช้ระบบควบคุมแบบอัตโนมัติมากขึ้น อุปกรณ์เหล่านี้มักอยู่ในรูปของระบบควบคุมแบบฝังตัว (Embedded Systems)

เนื่องจากระบบควบคุมแบบฝัง เป็นระบบที่เหมาะสมในการควบคุมงานเฉพาะทาง แต่ระบบเหล่านี้มักเป็นระบบปิด ทำให้การควบคุมและใช้งานต้องกระทำผ่านเครื่องมือหรืออุปกรณ์เฉพาะของผู้ออกแบบหรือผู้ผลิตแต่ละรายเท่านั้น ซึ่งในอดีตผู้ผลิตระบบควบคุม หรืออุปกรณ์ต่างๆ มักมีวิธีการสร้างระบบของตนเองให้เป็นระบบควบคุมแบบเปิด หลากหลายแนวทาง เช่น การสร้างระบบควบคุมแบบกระจาย (Distributed Control System) โดยสื่อสารข้อมูลผ่านระบบเครือข่ายในรูปแบบต่างๆ หรือระบบควบคุมตามลักษณะมัลติโพรเซสเซอร์ (Multi-processor System) ซึ่งมีการส่งข้อมูล สื่อสาร ควบคุม ผ่านทางช่องสัญญาณเฉพาะ แต่ไม่ว่าจะเป็นแบบใดก็ตาม การควบคุมจำเป็นจะต้องมีซอฟต์แวร์พิเศษ ซึ่งมักยึดติดกับระบบนั้น หรือ ระบบปฏิบัติการของเครื่องที่ใช้ควบคุม

จากปัญหาดังกล่าว จึงนำมาสู่แนวความคิดในการติดต่ออุปกรณ์แบบฝังตัวต่างๆ ผ่านทางระบบเครือข่ายซึ่งช่วยลดปัญหาในด้านการพัฒนาซอฟต์แวร์หรือช่องสัญญาณพิเศษ เพื่อใช้ในการติดต่อควบคุมระบบควบคุมแบบฝังตัวเหล่านั้น ซึ่งประโยชน์ของการเชื่อมต่อผ่านเครือข่ายแบบเว็บที่เด่นชัดคือ การไม่ยึดติดกับแพลตฟอร์มของผู้ใช้ ทั้งนี้ขอเพียงให้ผู้ใช้สามารถติดต่อผ่านโปรแกรมค้นผ่านเว็บได้เท่านั้น และเป็นประโยชน์สำหรับผู้พัฒนาในการที่ไม่จำเป็นจะต้องสร้างซอฟต์แวร์เชื่อมต่อพิเศษให้กับอุปกรณ์ควบคุมแบบเฉพาะทางเหล่านั้น

อย่างไรก็ตาม การเชื่อมต่อระบบควบคุมแบบฝังตัวเข้าสู่ระบบเครือข่ายอินเทอร์เน็ต ผ่านทางเว็บเซิร์ฟเวอร์แบบฝังตัวนั้น มีปัจจัยพื้นฐานที่เป็นปัญหาต้องพิจารณามากมาย อาทิ

- การสร้างระบบ TCP/IP, HTTP เว็บเซิร์ฟเวอร์, ระบบปฏิบัติการ รวมถึงส่วนต่อประสานกับผู้ใช้ (User Interface) ที่สวยงาม ลงบนไมโครคอนโทรลเลอร์ หรือ ไมโครโพรเซสเซอร์ ซึ่งมีหน่วยความจำที่จำกัด และ ความเร็วของการประมวลผลต่ำ
- ความเร็วในการควบคุม และ จำนวนผู้ใช้ ที่สามารถ ตรวจสอบ หรือ ควบคุมได้ ณ เวลาหนึ่งๆ
- การจัดระบบเชื่อมต่อภายใน ระหว่าง HTTP เว็บเซิร์ฟเวอร์ และ อุปกรณ์ควบคุมเฉพาะต่างๆ ผ่านระบบ CGI อันเป็นมาตรฐานของระบบเว็บเซิร์ฟเวอร์ทั่วไป ซึ่งทำงานช้า และพัฒนาได้ลำบาก
- ความปลอดภัยของการควบคุม โดย ควรจะมีการกำหนดสิทธิ ในการควบคุมแบบต่างๆ ให้กับผู้ใช้งานแต่ละคน ตลอดจนการเข้ารหัสข้อมูล เพื่อป้องกันอันตรายจากผู้ประสงค์ร้าย
- ปัญหาเรื่องศูนย์กลางการควบคุม หาก มีอุปกรณ์ต่อพ่วง ผ่านระบบเครือข่าย ที่ต้องการความสอดคล้อง ในการทำงาน ร่วมกัน
- ความยุ่งยากในการสร้างสคริปต์ใหม่เมื่อต้องการเปลี่ยนแปลงค่า Configuration ต่างๆ

จากปัญหาและความสำคัญดังกล่าว จึงได้เกิดงานวิจัยนี้ขึ้น เพื่อพัฒนาเทคนิคและ แนวทางของระบบเว็บเซิร์ฟเวอร์แบบฝังตัว อันจะช่วยอำนวยความสะดวกในระบบควบคุมต่างๆ ต่อไป

1.2. วัตถุประสงค์ของงานวิจัย

เพื่อพัฒนาระบบเว็บเซิร์ฟเวอร์แบบฝังตัว ที่เชื่อมต่อกับอุปกรณ์ไฟฟ้าต่างๆ และสามารถควบคุมโดยใช้เว็บ ผ่านเครือข่ายอินเทอร์เน็ตได้

1.3. คำจำกัดความที่ใช้ในงานวิจัย

คำเหล่านี้คือคำจำกัดความที่จะกล่าวถึงภายในงานวิจัยนี้ โดยบางคำจะมีความหมายครอบคลุมถึงระบบที่ใกล้เคียงหรือระบบที่เฉพาะเจาะจง

อินเทอร์เน็ต (Internet)

ระบบเครือข่ายที่ทำงานบนมาตรฐาน TCP/IP ซึ่งในที่นี้รวมความถึง กลุ่มย่อยของอินเทอร์เน็ตภายในองค์กรที่เรียกว่าอินทราเน็ต (Intranets) ซึ่งนิยมใช้งานในองค์กรขนาดใหญ่ และ ระบบเครือข่ายระหว่างองค์กรแบบเอ็กซ์ทราเน็ต (Extranets) ซึ่งนิยมใช้ระหว่างผู้ซื้อและผู้ขายระหว่างองค์กร

ระบบเครือข่ายเว็บ (Web network)

ระบบเครือข่ายเว็บ (Web network) คือการใช้งานระบบอินเทอร์เน็ตผ่านโปรแกรมค้นผ่านเว็บ (Web Browser) เช่น Netscape Navigator หรือ Internet Explorer เป็นไคลเอ็นในการขอข้อมูลจากเว็บเซิร์ฟเวอร์ ซึ่งการทำงานด้งกล่าวอ้างอิงจากโพรโทคอล HTTP เป็นหลัก โดยในที่นี้มุ่งเน้นความสนใจไปยังการติดต่อผ่าน HTTP ของผู้ใช้และผู้ให้บริการข้อมูล

อีเทอร์เน็ต (Ethernet)

มาตรฐานในการเชื่อมต่อระบบเครือข่ายท้องถิ่นที่ได้รับความนิยมมากที่สุด ซึ่งในงานวิจัยนี้จะถือว่าการเชื่อมต่อระบบเครือข่าย คือการเชื่อมต่อผ่านผ่านระบบสายสัญญาณตามมาตรฐาน Ethernet หรือ IEEE802.3 ลักษณะโดยทั่วไปของอีเทอร์เน็ตคือ การทำงานในลักษณะ Broadcast ที่จุดเชื่อมต่อต่างๆ สามารถรับส่งข้อมูลได้ตลอดเวลา

TCP/IP

Transmission Control Protocol/Internet Protocol เป็นโพรโทคอลที่กำหนดการรับส่งข้อมูลระหว่างจุดบนระบบเครือข่าย ซึ่งคุณลักษณะเฉพาะของ TCP/IP คือความสามารถในการยืนยันความถูกต้องของการรับส่ง และการได้รับข้อมูลบนระบบเครือข่าย

HTTP

Hypertext Transport Protocol หรือเกณฑ์วิธีขนส่งข้อความหลายมิติ เป็นการทำงานในระดับโปรแกรมประยุกต์ (Application) เพื่อใช้ในการแลกเปลี่ยนข้อมูล ซึ่งมุ่งเน้นที่การแลกเปลี่ยนข้อมูลระหว่างเว็บเซิร์ฟเวอร์ และ โปรแกรมค้นผ่านเว็บ HTTP จะทำการส่งข้อมูลตามมาตรฐานของ TCP/IP

URL

Uniform Resource Locator หรือมาตรฐานโปรแกรมชี้แหล่งทรัพยากรสากล เป็นข้อความที่โปรแกรมค้นผ่านเว็บใช้เพื่ออ้างอิงถึงข้อมูลที่ต้องการจากเว็บเซิร์ฟเวอร์ โดยผ่านเซิร์ฟเวอร์จะเป็นผู้จัดการด้วยตัวเอง ทั้งนี้ URL อาจอ้างถึงแฟ้มข้อมูลธรรมดาหรืออาจอ้างถึงข้อมูลที่ได้รับการสร้างแบบพลวัต (Dynamic) จากเซิร์ฟเวอร์ก็ได้

HTML

HyperText Markup Language เป็นภาษาที่ใช้ในการอธิบายรูปร่างลักษณะของหน้าเว็บ เพื่อให้โปรแกรมค้นผ่านเว็บสามารถสร้างหน้าจอและรูปภาพแสดงให้ผู้ใช้เห็นได้

Web device

อุปกรณ์ที่สามารถสื่อสารหรือควบคุมผ่านระบบเครือข่ายคอมพิวเตอร์ได้ โดยอาศัยโปรแกรมค้นผ่านเว็บเป็นตัวควบคุม หรืออาจกล่าวได้อีกนัยหนึ่งว่า เป็นอุปกรณ์ที่ฉลาดเนื่องจากสามารถพูดคุยผ่าน TCP/IP และ HTTP ได้

1.4. ขอบเขตงานวิจัย

- พัฒนาอุปกรณ์สื่อสาร โดยในที่นี้อาศัยมาตรฐาน IEEE802.3 อีเทอร์เน็ต 10BaseT ในการเชื่อมต่อ ไมโครคอนโทรลเลอร์ เข้าสู่ระบบเครือข่าย
- ทำการพัฒนา TCP/IP Stack และระบบบริการข้อมูลระบบเครือข่ายเว็บตามมาตรฐาน HTTP/1.0 พร้อมสร้างระบบความปลอดภัยอย่างง่าย
- ติดตั้งเครื่องคอมพิวเตอร์แม่ข่ายและซอฟต์แวร์ที่จำเป็น เพื่อทำหน้าที่ช่วยในการสร้างหน้าเว็บที่สวยงามและช่วยในการสร้างระบบควบคุมที่ซับซ้อนยิ่งขึ้น
- ศึกษาความเป็นไปได้ในการเข้ารหัสและกำหนดระบบรักษาความปลอดภัยของการบริการข้อมูลสำหรับเว็บเซิร์ฟเวอร์แบบฝังตัว

1.5. ประโยชน์ที่คาดว่าจะได้รับ

- สามารถพัฒนาระบบควบคุมเฉพาะทางที่เชื่อมต่อเข้ากับระบบเครือข่ายอินเทอร์เน็ต ที่มีความสามารถในการควบคุม ตั้งค่า ตรวจสอบ และ ดูแลผ่านระบบเครือข่ายเว็บ และมีความปลอดภัยในการควบคุม
- สามารถพัฒนา TCP/IP ที่มีขนาดเล็ก และเพียงพอต่อการใช้งานพื้นฐาน
- เข้าใจการทำงานของระบบเครือข่ายอินเทอร์เน็ตดีขึ้น

- เพื่อเป็นแนวทางในการพัฒนาการประมวลผลบนระบบเว็บเซิร์ฟเวอร์แบบฝังตัวต่อไป

1.6. วิธีดำเนินงานวิจัย

- ศึกษารวบรวมข้อมูลการทำงานของอีเทอร์เน็ต และ วงจรรวมที่เกี่ยวข้อง
- ออกแบบฮาร์ดแวร์ เพื่อใช้สื่อสารกับอีเทอร์เน็ตตามมาตรฐาน IEEE802.3 อีเทอร์เน็ต 10BaseT เข้ากับไมโครคอนโทรลเลอร์
- ทดสอบการสื่อสาร และอุปกรณ์ต่างๆ
- พัฒนาซอฟต์แวร์ เพื่อสร้าง TCP/IP และองค์ประกอบที่จำเป็นลงสู่ ไมโครคอนโทรลเลอร์
- ทำการพัฒนาบริการต่างๆ ให้กับ ไมโครคอนโทรลเลอร์
- ติดตั้งระบบเครื่องคอมพิวเตอร์แม่ข่ายเพื่อทำหน้าที่เป็นพี่เลี้ยงช่วยในการสร้างเว็บที่มีขนาดใหญ่และซับซ้อน
- ทดสอบต่อเชื่อมไมโครคอนโทรลเลอร์เข้ากับระบบควบคุมอย่างง่าย และ ทดลองโปรแกรมเพิ่มเติม
- วิเคราะห์และสรุปผล

บทที่ 2
มาตรฐาน โพรโทคอลและทฤษฎีที่เกี่ยวข้อง

การทำงานของระบบเครือข่ายอินเทอร์เน็ตนั้น อาศัยการทำงานของมาตรฐานการสื่อสารแบบ TCP/IP ซึ่งสามารถสรุปโดยย่อ เปรียบเทียบกับมาตรฐานลักษณะการสื่อสารบนระบบเครือข่าย ตาม ISO/OSI Model ได้ดังนี้

Layer	ISO/OSI	TCP/IP (Internet)
7	Application	Telnet, FTP, SMTP, HTTP, DNS, BOOTP,DHCP,SNMP
6	Representation	
5	Session	
4	Transport	TCP , UDP
3	Network	IP , ICMP , IGMP
2	Data Link	Device Driver and Interface
1	Physical	Media

ภาพที่ 1 เปรียบเทียบมาตรฐาน ISO/OSI Model กับ มาตรฐาน TCP/IP

การสื่อสารบนระบบเครือข่ายอินเทอร์เน็ตนั้น จะอาศัย IP เป็นหลักโดยเครื่องคอมพิวเตอร์ และ อุปกรณ์ต่างๆ บนระบบเครือข่าย จะต้องมีตำแหน่ง IP เฉพาะของตัวเองที่ไม่ซ้ำกับใคร และบริการต่างๆ นั้นจะส่งข้อมูลถึงกันระหว่างเครื่องคอมพิวเตอร์ ผ่านทาง TCP หรือ UDP เป็นหลัก ทั้งนี้ขึ้นกับประเภทของการใช้งาน

ในที่นี้ Physical Layer และ Data Link Layer ใช้มาตรฐานของอีเทอร์เน็ต (IEEE802.3) ในขณะที่ TCP และ IP จะครอบคลุมการทำงานในระดับ Network และ Transport Layer ส่วนการทำงานของ HTTP หรือ เว็บเซิร์ฟเวอร์นั้นจะอยู่ในระดับ Session, Representation และ Application

2.1. แนวทางการพัฒนา Web Device

การทำให้อุปกรณ์ที่ต้องการควบคุม สามารถควบคุมได้ผ่านทางเครือข่ายเว็บนั้น คือ การทำให้อุปกรณ์ดังกล่าวฉลาด (มีโพรเซสเซอร์ ระบบปฏิบัติการ และ TCP) ซึ่ง Rodney

Snell (1999) ได้กล่าวสรุปใน Web-Based Device Monitoring and Control ว่าวิธีการเพื่อทำให้อุปกรณ์ฉลาด สามารถควบคุมได้ผ่านระบบเครือข่ายเว็บนั้น มีด้วยกัน 4 วิธี ดังนี้

- เว็บเซิร์ฟเวอร์แบบฝังตัว (Embedded Web Server) คือ การรวมอุปกรณ์ ในการเชื่อมต่อระบบเครือข่ายอีเทอร์เน็ต เข้ากับอุปกรณ์ที่ต้องการควบคุมโดยตรง ซึ่งจะทำให้อุปกรณ์เหล่านี้ สามารถสื่อสารได้โดยตรงผ่านทางโปรแกรมค้นผ่านเว็บ ประโยชน์ของการใช้วิธีนี้ คือสามารถควบคุมได้ง่าย ไม่ต้องการอุปกรณ์ใดเพิ่มเติม อย่างไรก็ดีเนื่องจากเว็บเซิร์ฟเวอร์แบบฝังตัวนั้น มักมีข้อจำกัดในเรื่องหน่วยความจำ ทำให้ไม่สามารถเก็บรูปภาพหรือข้อมูลขนาดใหญ่ได้ นอกจากนี้ยังขาดสูญกลางของการควบคุมอุปกรณ์ ในกรณีที่อุปกรณ์หลายชิ้นต่อพ่วงอยู่กับระบบเครือข่าย และต้องการควบคุมพร้อมๆ กัน
- เว็บเซิร์ฟเวอร์แบบฝังตัว ร่วมกับเว็บเซิร์ฟเวอร์ภายนอกทั่วไป (Embedded Web Server + External Web Server) มีลักษณะคล้ายกับเว็บเซิร์ฟเวอร์แบบฝังตัว แต่จะมีการเสริมความสามารถในการจัดเก็บข้อมูลขนาดใหญ่ เช่น Applet, รูปภาพ หรือข้อมูลขนาดใหญ่ต่างๆ ไว้กับเซิร์ฟเวอร์ภายนอกเพื่อช่วยให้สามารถควบคุมสิ่งที่ซับซ้อนได้มากขึ้น อย่างไรก็ตาม ข้อเสียใหญ่คือค่าใช้จ่ายที่ต้องเพิ่มขึ้นของเว็บเซิร์ฟเวอร์ภายนอก
- เว็บเซิร์ฟเวอร์ภายนอกทั่วไปร่วมกับ CGI (External Web Server + CGI) การออกแบบและการใช้งานนั้น เหมือนกับการใช้งานเว็บเซิร์ฟเวอร์ทั่วไป โดยพัฒนาโปรแกรม CGI (Common Gateway Interface) เพื่อทำการติดต่อกับอุปกรณ์ที่ต้องการควบคุม ซึ่งลักษณะดังกล่าวนี้ ไม่เหมาะกับงานควบคุมที่มีการเปลี่ยนแปลงอย่างรวดเร็ว และมีข้อดีคือสามารถพัฒนาได้ง่ายและรวดเร็ว
- เว็บเซิร์ฟเวอร์ภายนอกที่มีการปรับเปลี่ยนลักษณะบางประการให้เหมาะสม (Customize External Web Server) มีลักษณะคล้ายกับ การพัฒนาผ่านระบบ CGI แต่ เนื่องจาก ข้อเสียของระบบ CGI ดังกล่าว จึงดัดแปลงเซิร์ฟเวอร์ให้สามารถรับและทำการติดต่อกับอุปกรณ์ที่ต้องการควบคุมได้โดยตรง โดยไม่ต้องผ่าน โปรแกรมควบคุมพิเศษอื่น เช่นกรณีของ CGI

ลักษณะดังกล่าวข้างต้นนั้น มักเหมาะสมกับงานในรูปแบบที่แตกต่างกันไป ทั้งนี้ขึ้นอยู่กับปัจจัยต่างๆ เช่น ความเร็วของข้อมูลและการควบคุมที่ต้องการ ชนิดของอุปกรณ์ที่ต้องการควบคุม ความสามารถของอุปกรณ์ควบคุม และ ระบบเครือข่าย อย่างไรก็ดีทุกวิธี สามารถนำมาประยุกต์ใช้งานร่วมกันได้ ในงานวิจัยนี้มุ่งเน้นที่การพัฒนาระบบเว็บเซิร์ฟเวอร์แบบฝังตัวเท่านั้น

2.2. อีเทอร์เน็ต (IEEE 802.3)

การเชื่อมต่อเครื่องคอมพิวเตอร์เข้าสู่ระบบเครือข่ายแบบท้องถิ่น(Local Area Network) ในปัจจุบันนั้น นิยมใช้มาตรฐานอีเทอร์เน็ต ซึ่งเป็น Broadband Network ในการเชื่อมต่อ อย่างไรก็ตามในวิทยานิพนธ์ฉบับนี้มิได้มุ่งเน้นที่การทำงานของอีเทอร์เน็ต หากแต่มุ่งเน้นที่การเชื่อมต่ออีเทอร์เน็ตเข้าสู่ไมโครคอนโทรลเลอร์ การเชื่อมต่อตามมาตรฐานอีเทอร์เน็ตโดยทั่วไปนั้นจะอาศัยการทำงานของ Network Interface Controller หรืออุปกรณ์ควบคุมที่มีความสามารถในการจัดรูปแบบ เข้ารหัส และ ถอดรหัสตามมาตรฐาน Manchester ที่ความเร็ว 10Mbs ซึ่งเป็นความเร็วที่สูง เมื่อเปรียบเทียบกับความเร็วของไมโครคอนโทรลเลอร์ที่ใช้ในงานควบคุมทั่วไป

จากการเชื่อมต่อไมโครคอนโทรลเลอร์เข้าสู่ระบบเครือข่ายโดยอาศัย NIC (Network interface Controller) ดังกล่าว หน่วยประมวลผลหลัก จะได้รับข้อมูลที่ผ่านการตรวจสอบความถูกต้องจาก NIC แล้ว และข้อมูลที่ส่งออกจะได้รับการสร้าง Check sum และ เติม Header ให้โดย NIC เช่นกัน ลักษณะการทำงานดังกล่าวสามารถอธิบายได้ดังภาพที่ 2

	PRE	SFD	DES	SRC	TYPE	DATA	FCS
	62b	2b	6B	6B	2B	46B - 1500B	4B
การรับ	ควบคุมโดย NIC		ได้รับข้อมูลผ่าน DMA				
การส่ง	ควบคุมโดย NIC		ได้รับข้อมูลผ่าน DMA				คำนวณโดย NIC

"B" – Bytes

"b" – bits

ภาพที่ 2 ลักษณะเฟรมของ Ethernet

จากภาพที่ 2 พบว่า ข้อมูลบางส่วนได้รับการประมวลผลโดย NIC และ บางส่วนจะถูกประมวลผลโดยหน่วยประมวลผลหลักผ่านระบบ DMA โดย PRE (PREAMBLE) และ SFD นั้น เป็นข้อมูลที่ใช้ในการสื่อสารระหว่าง NIC เพื่อให้ทราบซึ่งการส่งและตำแหน่งเริ่มต้นของข้อมูล ซึ่ง NIC จะทำการตัดข้อมูลเหล่านี้ทิ้งก่อนดำเนินการตรวจสอบความถูกต้องของข้อมูลที่รับส่ง โดยอาศัย FCS (Frame Check Sum) นั้นเป็นตัวคำนวณที่ช่วยในการตรวจสอบความถูกต้องของข้อมูลที่ทำการรับส่ง สำหรับข้อมูลที่ไม่ได้ดูแลโดย NIC ซึ่งได้แก่ DES (ตำแหน่ง MAC ของผู้รับ)

SRC (ตำแหน่ง MAC ของผู้ส่ง) TYPE (ประเภทของข้อมูลที่ส่ง) และ Data (ข้อมูล) นั้น ผู้พัฒนาจะต้องทำการพัฒนาซอฟต์แวร์เพื่อดูแลข้อมูลในส่วนเหล่านี้เอง ก่อนส่งผ่านข้อมูลให้กับ IP หรือ ARP เพื่อทำการประมวลผลต่อไป

2.3. IP

IP (Internet Protocol) เป็นโพรโทคอลพื้นฐานของโพรโทคอลทั้งหมดใน TCP/IP การทำงานทุกอย่างบนอินเทอร์เน็ต ไม่ว่าจะเป็น TCP UDP ICMP หรือ IGMP ก็ตามล้วนแต่ต้องอาศัย IP ทั้งสิ้น อย่างไรก็ดีการทำงานของ IP เป็นลักษณะ Connectionless กล่าวคือไม่มีการยืนยันของการได้รับข้อมูลให้แก่ผู้ส่ง ดังนั้นในการใช้งาน IP จึงทำหน้าที่เพียงระบุหมายเลขเครือข่ายของผู้รับและผู้ส่งเท่านั้น

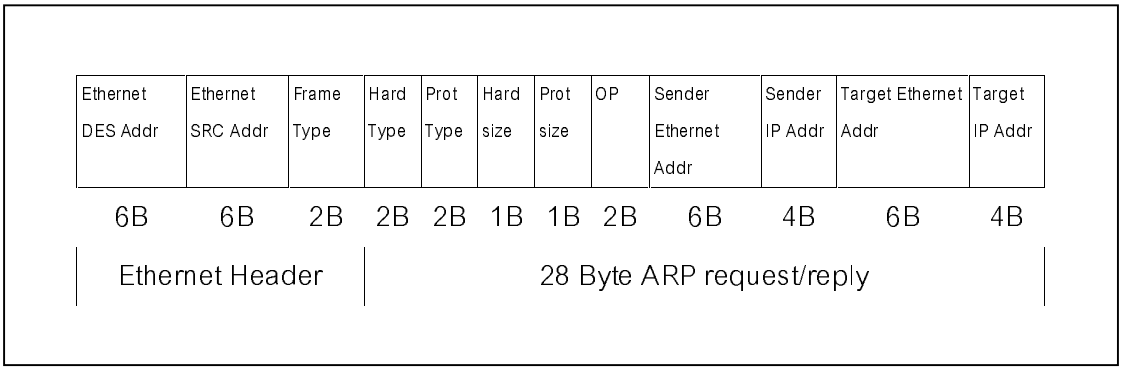
จุดมุ่งหมายของ IP คือการแสดงเครื่องและเครือข่ายที่ระบบต่อพ่วงอยู่ ซึ่งจะเป็นประโยชน์ในการหาทิศทางและการส่งผ่านข้อมูลผ่านระบบเครือข่าย โครงสร้างของ IP ประกอบด้วย Type of Service, Length, Time to Live, Protocol, Checksum และ IP Address ของผู้ส่งและผู้รับตามลำดับ จากนั้นจึงจะต่อด้วยข้อมูลส่วนอื่นๆ เช่น ICMP TCP หรือ UDP เพื่อส่งผ่านข้อมูลที่ซับซ้อนยิ่งขึ้น

2.4. ARP

ARP (Address Resolution Protocol) เป็นเกณฑ์วิธีจำแนกเลขที่อยู่ หรือมาตรฐานในการหาหมายเลขตำแหน่ง IP บนระบบเครือข่ายแบบต่างๆ ซึ่งในที่นี้มุ่งเน้นเพียงมาตรฐานของระบบเครือข่ายแบบอีเทอร์เน็ตเท่านั้น รายละเอียดเพิ่มเติมของ ARP นั้นสามารถหาอ้างอิงได้จาก RFC 826 ซึ่งสามารถกล่าวโดยสรุปได้ว่า ARP จะทำหน้าที่ในการจับคู่ระหว่างหมายเลขตำแหน่ง IP และ Physical Address ของมาตรฐานระบบเครือข่ายต่างๆ ซึ่งในที่นี้หมายเลขตำแหน่ง MAC ของเครือข่ายอีเทอร์เน็ต

หากวิเคราะห์จากลักษณะโครงสร้างของเฟรมบนมาตรฐานอีเทอร์เน็ต และ IP แล้ว จะพบว่าขนาดของหมายเลขตำแหน่งหรือ Address ซึ่งใช้บ่งบอกถึงที่อยู่บนระบบเครือข่ายของทั้งสองมาตรฐานนี้ มีความยาวแตกต่างกัน โดยหมายเลขตำแหน่ง MAC ของอีเทอร์เน็ตนั้นมีขนาด 48 บิต ในขณะที่หมายเลขตำแหน่ง IP มีขนาดเพียง 32 บิต จากลักษณะดังกล่าวจึงเป็นที่มาของโพรโทคอล ARP เพื่อใช้ในการแปลงและค้นหาหมายเลขตำแหน่ง MAC ของอีเทอร์เน็ตหรือของ Data Link Layer บนมาตรฐานเครือข่ายแบบอื่น ของหมายเลขตำแหน่ง IP ที่ระบบต้องติดต่อสื่อสารด้วย

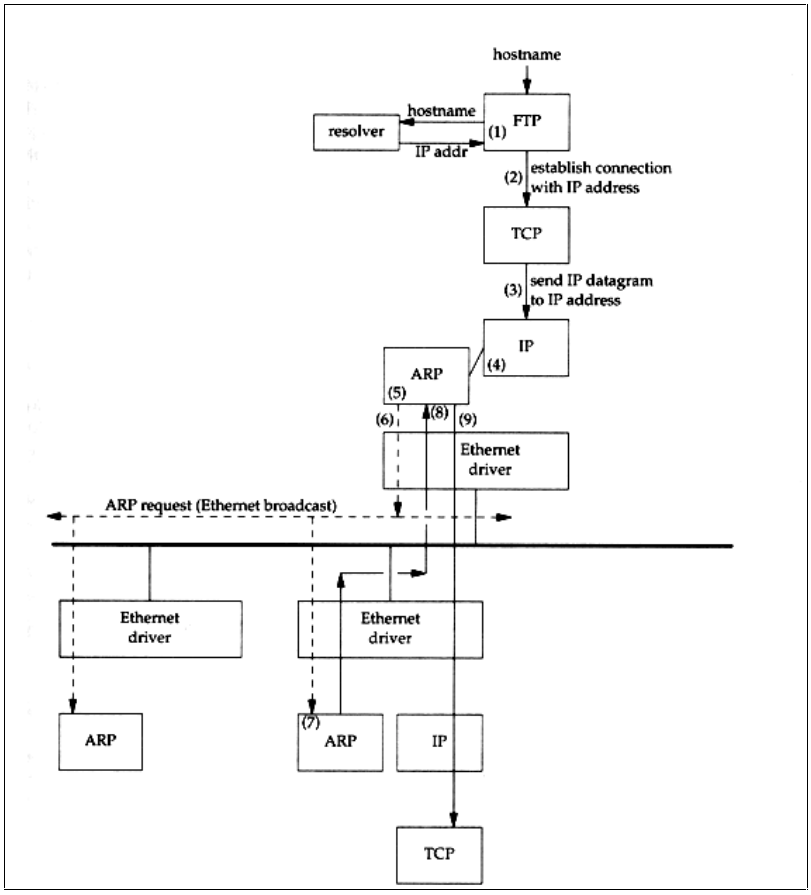
การทำงานของ ARP นั้นประกอบด้วยการขอ (Request) และการตอบ (Reply) โดยการขอ คือการขอทราบหมายเลขตำแหน่ง MAC ของหมายเลขตำแหน่ง IP ที่ระบุและการตอบคือ การตอบหมายเลขตำแหน่ง MAC ให้กับผู้ขอทราบหมายเลขตำแหน่ง IP นั้น ลักษณะของเฟรมที่ใช้ในการรับส่ง ARP Request และ Reply นั้นสามารถแสดงได้ดังภาพที่ 3



ภาพที่ 3 ลักษณะเฟรมของ ARP

จากโครงสร้างเฟรมของ ARP ดังกล่าวสามารถอธิบายความหมายของคำต่างๆ ได้ดังนี้ Ethernet DES Addr และ SRC Addr คือหมายเลขตำแหน่ง MAC ของเครื่องคอมพิวเตอร์ปลายทางและต้นทางตามลำดับ ซึ่งในกรณีของการขอนั้น หมายเลขตำแหน่ง MAC ของเครื่องคอมพิวเตอร์ปลายทางจะเป็นหมายเลขตำแหน่ง MAC แบบ Broadcast หรือเป็นหมายเลขตำแหน่ง MAC ที่ระบุถึงเครื่องคอมพิวเตอร์ทุกเครื่องบนระบบเครือข่าย ส่วน Hard Type, Prot Type, Hard Size, Prot Size นั้น จะมีค่าเป็น 0x0001, 0x0800, 6 และ 4 ตามลำดับสำหรับการหาหมายเลขตำแหน่ง IP บนระบบเครือข่ายอีเทอร์เน็ต

การทำงานของ ARPนั้นเริ่มต้นจากการที่เครื่องต้นทางตรวจสอบตาราง ARP ของตนเองก่อน หากพบหมายเลขตำแหน่ง MAC ของตำแหน่ง IP ที่จะติดต่อสื่อสารด้วย ระบบก็จะข้ามการทำงานของ ARP ไป แต่หากไม่พบเครื่องต้นทางจะทำการร้องขอหมายเลขตำแหน่ง MAC จากเครื่องคอมพิวเตอร์ทุกเครื่องที่อยู่บนระบบเครือข่ายอีเทอร์เน็ตเดียวกันด้วยการส่งข้อมูลประเภท Broadcast ไป ดังแสดงในภาพที่ 4 ซึ่งเป็นการขอทราบหมายเลขตำแหน่ง MAC จากเครื่องคอมพิวเตอร์หมายเลข 5 ไปยังเครื่องคอมพิวเตอร์ทุกเครื่อง ด้วย ARP request (Ethernet broadcast)



ภาพที่ 4 การทำงานของ ARP

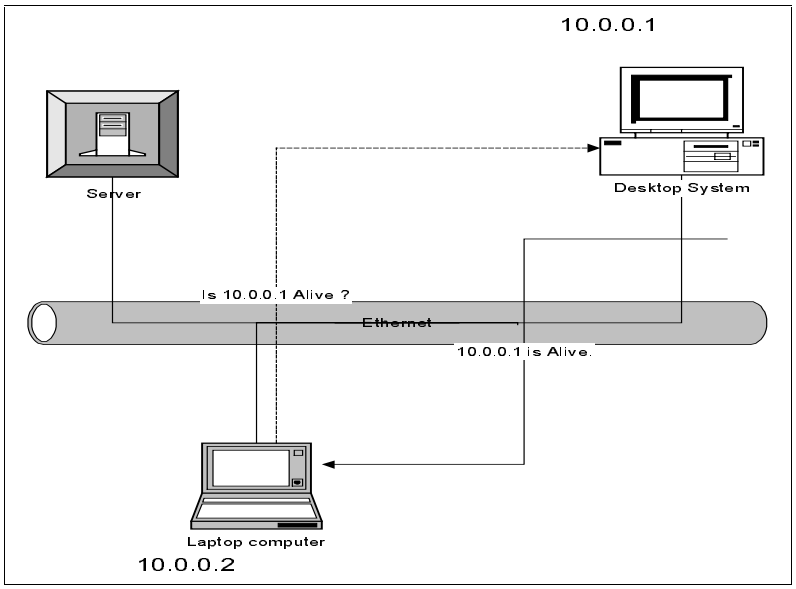
เมื่อเครื่องปลายทางได้รับ ARP เครื่องปลายทางก็จะเปรียบเทียบหมายเลขตำแหน่ง IP ที่ได้รับ ว่าตรงกับหมายเลขตำแหน่ง IP ของตนหรือไม่ หากพบว่าหมายเลขตำแหน่ง IP ดังกล่าวเป็นของตน ก็จะตอบรับ ARP Reply พร้อมทั้งระบุหมายเลขตำแหน่ง MAC ของตนเข้าไปกับ เฟรมตอบรับดังกล่าว เพื่อให้เครื่องต้นทางทราบหมายเลขตำแหน่ง Mac ของตนเอง จากลักษณะการทำงานที่กล่าวมา พบว่า ARP จะถูกเรียกใช้งานก็ต่อเมื่อ เครื่องคอมพิวเตอร์ต้นทางไม่ทราบ หมายเลขตำแหน่ง MAC ของเครื่องคอมพิวเตอร์ปลายทางเท่านั้น

ในกรณีที่ไม่มีการทำงานของการขอและการตอบรับ ARP แล้ว ผู้ใช้งานเครื่องคอมพิวเตอร์ต้นทางจะต้องปรับแต่ง ARP Table ของเครื่องคอมพิวเตอร์ต้นทางด้วยตนเองเพื่อให้เครื่องคอมพิวเตอร์ดังกล่าวทราบหมายเลขตำแหน่ง Mac โดยการใช้คำสั่ง ARP (ซึ่งจะกล่าวถึงในบทที่ 7 ต่อไป)

2.5. ICMP

ICMP (Internet Control Message Protocol) มีการทำงานในระดับเดียวกันกับ IP โดยทำหน้าที่ในการรายงานข้อผิดพลาดหรือสถานะการทำงานของ IP ซึ่งเราสามารถจัดกลุ่มของประเภท Message ใน ICMP ได้เป็นกลุ่มตามลักษณะการทำงาน แต่กลุ่มที่สำคัญคือและใช้ในการดำเนินงานวิจัยครั้งนี้คือ กลุ่ม echo request หรือที่เรียกกันโดยทั่วไปว่า ping request

Ping นั้นเป็นชื่อที่ได้มาจากการทำงานของ Sonar เพื่อใช้ในการตรวจสอบที่หมายของเครื่องคอมพิวเตอร์บนอินเทอร์เน็ตว่าสามารถติดต่อได้หรือไม่ โดยปกติหากเครื่องคอมพิวเตอร์ที่ต้องการติดต่อไม่สามารถ Ping ได้นั้น สามารถตีความหมายได้ว่าเราไม่สามารถใช้บริการอื่นๆ จากเครื่องคอมพิวเตอร์นั้นได้เช่นกัน ในงานวิจัยนี้จึงมีการพัฒนา Ping Response เพื่อให้เครื่องคอมพิวเตอร์ต้นทางสามารถตรวจสอบในส่วน การเชื่อมต่อของระบบเว็บเซิร์ฟเวอร์แบบฝังตัวเข้ากับระบบเครือข่าย ดังตัวอย่างในภาพที่ 5 แสดงการส่ง Request จากเครื่อง 10.0.0.2 ไปยัง 10.0.0.1 และการตอบรับด้วย Ping Response จากเครื่อง 10.0.0.1 ไปยัง 10.0.0.2



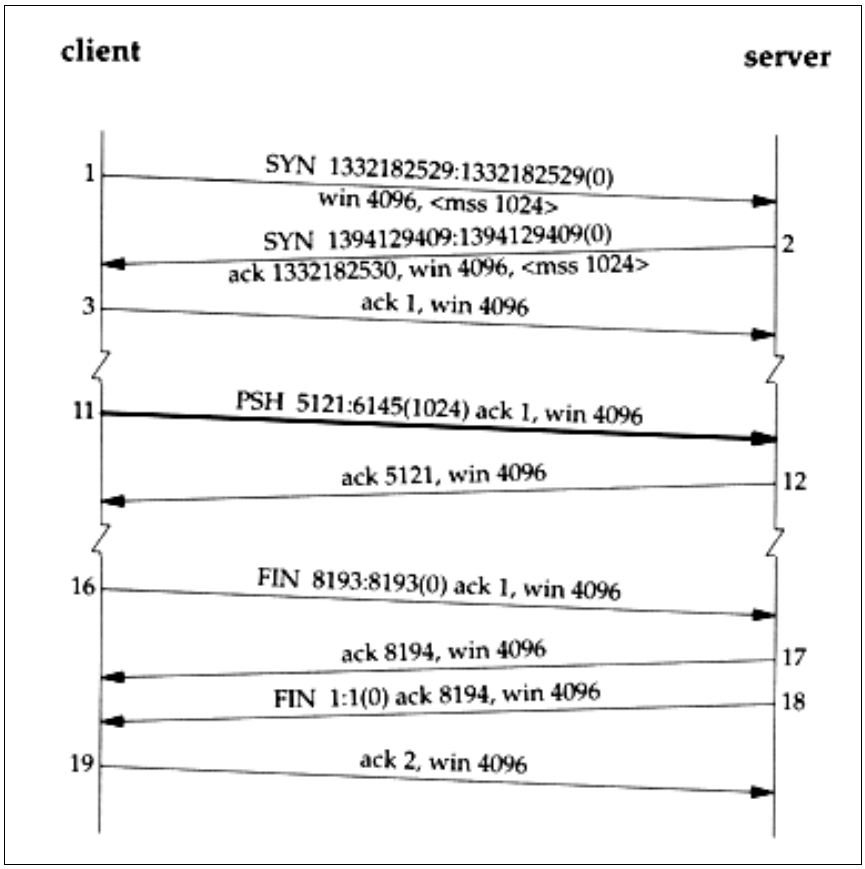
ภาพที่ 5 การทำงานของ Ping

2.6. TCP

TCP (Transmission Control Protocol) เป็นหัวใจของการทำงานบนระบบเครือข่ายอินเทอร์เน็ตทั้งนี้เนื่องจาก TCP จะทำหน้าที่ตรวจสอบความถูกต้องของข้อมูล รวมถึงยืนยันการรับและการส่งข้อมูลในลักษณะของ Sliding Window

ลักษณะโดยทั่วไปของ TCP จะประกอบด้วย Sequence Number เพื่อใช้แสดงหมายเลขของ Packet ที่ส่ง และ Acknowledgement Number เพื่อใช้ยืนยันการรับข้อมูลของ Sequence Number ที่ได้รับ เราจึงสังเกตได้ว่า Packet ทุกแบบจะต้องมีการตอบรับด้วย Packet ประเภท Acknowledgement (ACK) เสมอ นอกจากนี้ยังประกอบด้วย Checksum ซึ่งทำหน้าที่ตรวจสอบความถูกต้องของข้อมูลที่ได้รับ ข้อมูลในระดับ Application ต่างๆ ที่ทำงานบน TCP จะเป็นส่วนหนึ่งของการคำนวณ Checksum ของ TCP

ลักษณะการทำงานของ TCP นั้นสามารถอธิบายประกอบภาพที่ 6 ได้ดังนี้ เริ่มต้นด้วยการส่ง SYN Packet ไปยังฝ่ายที่ระบบต้องการติดต่อด้วยหรือเซิร์ฟเวอร์ จากนั้นฝ่ายเซิร์ฟเวอร์จะทำการตอบรับด้วย ACK และ SYN Packet มายังไคล์เอ็น (Client) ซึ่งไคล์เอ็นจะต้องตอบ ACK การ SYN ของเซิร์ฟเวอร์อีกครั้งหนึ่ง เมื่อเสร็จสมบูรณ์แล้ว ทั้ง 2 ฝ่ายจึงอยู่ในสถานะที่พร้อมทำการรับส่งข้อมูล โดยการแลกเปลี่ยนข้อมูลระหว่างกันนั้นจะกระทำด้วย PSH และ ACK Packet เพื่อใช้ส่งและยืนยันการได้รับตามลำดับ เมื่อการรับส่งข้อมูลเรียบร้อยสมบูรณ์แล้ว จึงส่ง FIN Packet และ ACK ระหว่างกันในทำนองเดียวกับการเปิด เพื่อเป็นการปิดการสื่อสาร

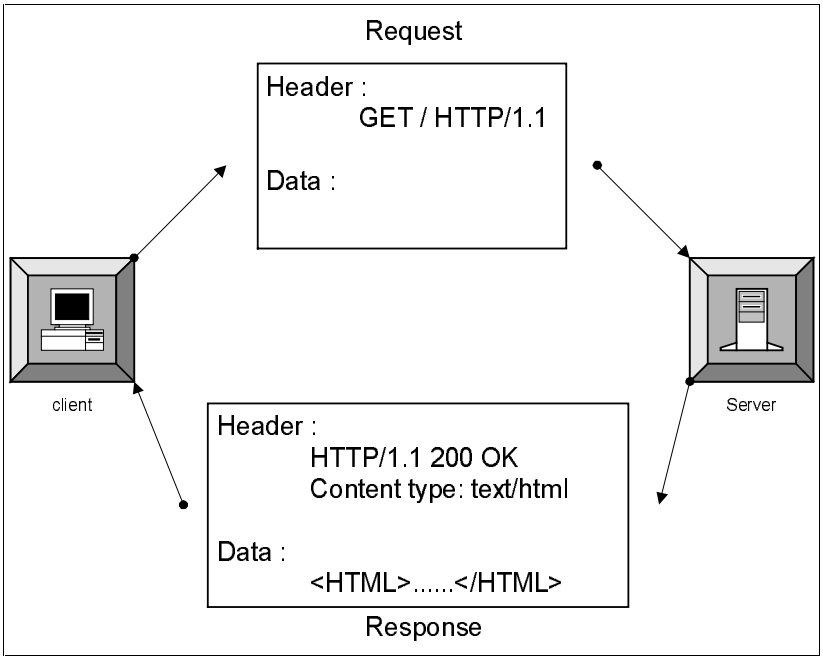


ภาพที่ 6 การทำงานของ TCP

2.7. HTTP

HTTP (HyperText Transfer Protocol) เป็นโพรโทคอลที่ทำงานในระดับ Application เพื่อใช้ในการขอ และบริการข้อมูลบนระบบเครือข่าย การทำงานของ HTTP สามารถแบ่งเป็นกลุ่มได้ 2 กลุ่ม คือ กลุ่ม Request และ กลุ่ม Response โดยในแต่ละกลุ่มจะประกอบด้วย Header และ Data

ส่วน HTTP Header นั้นใช้เพื่อแสดงรายระเอียดของประเภทข้อมูลที่ขอหรือให้บริการ ในขณะที่ส่วน Data นั้นใช้เพื่อรับส่งข้อมูล โดยการทำงานของ HTTP นั้น สามารถอธิบายได้ดังภาพที่ 7 คือ เมื่อผู้ใช้ต้องการข้อมูลโปรแกรมค้นผ่านเว็บจะทำการส่ง Request ไปยังผู้ให้บริการหรือเซิร์ฟเวอร์ โดยใน Header จะระบุถึงข้อมูลที่ต้องการ และอาจมีข้อมูลในส่วน Data ด้วย หากเป็นการ Request แบบ Post หลังจากที่เซิร์ฟเวอร์ได้รับการร้องขอแล้ว เซิร์ฟเวอร์จะทำการประมวลผลเพื่อสร้างผลลัพธ์ และ ส่งผลลัพธ์กลับด้วย Response โดยส่วน Header Response จะบุถึงประเภทของข้อมูลที่ระบบตอบรับ ขนาด ภาษา และรายเอียดอื่นๆ จากนั้นจึงตามด้วยข้อมูลผลลัพธ์ในส่วน Data



ภาพที่ 7 การทำงานของ HTTP

2.8. การเข้ารหัสแบบ Base 64

การเข้ารหัสแบบ Base 64 เป็นมาตรฐานการเข้ารหัสพื้นฐานที่นิยมใช้ในการสื่อสารผ่านระบบเครือข่ายอินเทอร์เน็ต โดยเฉพาะอย่างยิ่งการรับส่ง Email และระบบการพิสูจน์ตัวจริงแบบเบื้องต้นบนเว็บ (Basic Web Authentication) ซึ่งจะกล่าวถึงในบทที่ 5

การทำงานของ Base 64 นั้น จะทำการแปลงข้อมูล 3 ไบต์เป็น 4 ไบต์โดยอ้างอิงจากตารางแสดงผลของข้อมูล 6 บิตดังต่อไปนี้ (ตาราง 1)

ตาราง 1 การเข้ารหัสแบบ Base 64

ค่า HEX	String	ค่า HEX	String	ค่า HEX	String	ค่า HEX	String
0X00	“A”	0X10	“Q”	0X20	“g”	0X30	“w”
0X01	“B”	0X11	“R”	0X21	“h”	0X31	“x”
0X02	“C”	0X12	“S”	0X22	“I”	0X32	“y”
0X03	“D”	0X13	“T”	0X23	“j”	0X33	“z”
0X04	“E”	0X14	“U”	0X24	“k”	0X34	“0”
0X05	“F”	0X15	“V”	0X25	“l”	0X35	“1”
0X06	“G”	0X16	“W”	0X26	“m”	0X36	“2”
0X07	“H”	0X17	“X”	0X27	“n”	0X37	“3”
0X08	“I”	0X18	“Y”	0X28	“o”	0X38	“4”
0X09	“J”	0X19	“Z”	0X29	“p”	0X39	“5”
0X0A	“K”	0X1A	“a”	0X2A	“q”	0X3A	“6”
0X0B	“L”	0X1B	“b”	0X2B	“r”	0X3B	“7”
0X0C	“M”	0X1C	“c”	0X2C	“s”	0X3C	“8”
0X0D	“N”	0X1D	“d”	0X2D	“t”	0X3D	“9”
0X0E	“O”	0X1E	“e”	0X2E	“u”	0X3E	“+”
0X0F	“P”	0X1F	“f”	0X2F	“v”	0X3F	“/”

การเข้ารหัสทำได้โดยการดึงข้อมูลรหัสแอสกีที่ต้องการเข้ารหัสมาทีละ 3 ไบต์ จากนั้นทำการเขียนเป็นสายบิต (Bit Stream) ต่อกัน เมื่อได้แล้วให้ทำการแบ่งทีละ 6 บิต จะได้

ข้อมูลทั้งสิ้น 4 ชุด และทำการอ้างอิงข้อมูลทั้ง 4 ชุดจากตัวอักษรในตาราง BASE 64 ข้างต้น (ดูตัวอย่างภาพที่ 8)

ข้อมูลต้นฉบับ	“pok”
ค่า ASCII (Hex)	0x70 0x6F 0x6B
ทำการเขียนเป็น Bit Stream	01110000 01101111 01101011
ทำการตัดข้อมูลทีละ 6 Bit	011100 000110 111101 101011
ค่า Hex	0x1C 0x06 0x3D 0x2B
ผลลัพธ์ที่ได้จากการเข้ารหัส	“cG9r”

ภาพที่ 8 ตัวอย่างการเข้าและถอดรหัสแบบ Base64

หากความยาวของข้อมูลหารด้วย 3 ไม่ลงตัว ให้เติมตัวอักษรว่าง (Space) ต่อท้ายเข้าไปจนสามารถจัดกลุ่มได้ 3 ตัว เช่นกรณีข้อความตั้งต้นเป็น “pok:test” จะพบว่าข้อมีความยาว 8 ตัวอักษร และเพื่อให้ความยาวของข้อมูลหารด้วย 3 ลงตัว จึงต้องปรับข้อความตั้งต้นเป็น “pok:test ” นอกจากนี้ในการใช้งานจริงยังพบว่ามีตัวอักษรอื่นบางตัวที่มิได้กล่าวถึงในตารางแต่มีการใช้งานอยู่จริง เช่น “?” แทน 0x00 เป็นต้น

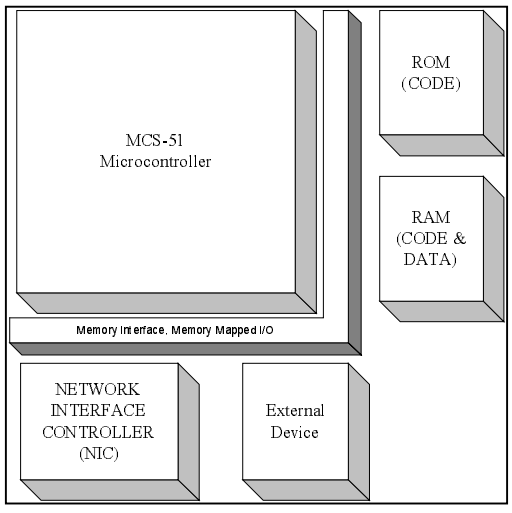
การถอดรหัสนั้นทำในทำนองเดียวกันคือ แปลงข้อมูลที่ได้เป็น Hex จากนั้นเขียนเป็นสายบิต (Bit Stream) และจัดกลุ่มที่ละ 8 บิต แล้วจึงนำผลลัพธ์ที่ได้เปลี่ยนค่ากลับเป็นรหัสแอสกี

บทที่ 3
การออกแบบฮาร์ดแวร์

ฮาร์ดแวร์ของระบบควบคุมแบบฝังตัวโดยทั่วไปนั้น ประกอบด้วยไมโครคอนโทรลเลอร์ หน่วยความจำ และหน่วยรับข้อมูลเข้าออก (I/O Port) นอกจากนี้ยังประกอบด้วยช่องสัญญาณสำหรับการสื่อสาร (BUS) เพื่อใช้ในการเชื่อมต่ออุปกรณ์ระบบควบคุมเข้ากับอุปกรณ์ต่างๆ รายละเอียดการออกแบบฮาร์ดแวร์ที่กล่าวถึงในที่นี้ จึงประกอบด้วยการออกแบบส่วนประกอบหลักของไมโครคอนโทรลเลอร์ การเชื่อมต่อไมโครคอนโทรลเลอร์เข้ากับระบบเครือข่ายโดยอาศัยอุปกรณ์ควบคุมการเชื่อมต่อระบบเครือข่าย (Network Interface Controller) รวมถึงอธิบายการจัดสรรหน่วยความจำของระบบ

3.1. ส่วนประกอบทางฮาร์ดแวร์ของระบบ

ระบบควบคุมแบบฝังตัวนั้นมีโครงสร้างทั่วไปเหมือนคอมพิวเตอร์ขนาดเล็ก กล่าวคือประกอบด้วย ไมโครคอนโทรเลอร์ หน่วยความจำโปรแกรมซึ่งมักเป็นรอม และหน่วยความจำข้อมูลหรือแรม นอกจากนี้ยังมีระบบอินพุต/เอาต์พุตเพื่อใช้สำหรับต่อเชื่อมกับอุปกรณ์ภายนอกอื่นๆ แต่ลักษณะของเว็บเซิร์ฟเวอร์แบบฝังตัวนั้น จำต้องมีความสามารถในการติดต่อสื่อสารผ่านระบบเครือข่ายด้วย (ในที่นี้คือระบบอีเทอร์เน็ต) ซึ่งการเชื่อมต่อบนระบบเครือข่ายของเครื่องคอมพิวเตอร์นั้น โดยทั่วไปอาศัยการทำงานของอุปกรณ์ควบคุมการเชื่อมต่อระบบเครือข่าย (Network Interface Controller) ทั้งนี้เนื่องจากไมโครคอนโทรเลอร์ทำงานที่ความเร็วต่ำ เมื่อเทียบกับความเร็วของระบบเครือข่ายที่มีการส่งข้อมูลที่ 10 Mbps โครงสร้างโดยรวมดังกล่าวสามารถแสดงได้ดังภาพที่ 9



ภาพที่ 9 ส่วนประกอบทางฮาร์ดแวร์ของระบบ

3.2. ส่วนประมวลผลหลัก

หน้าที่ของส่วนประมวลผลหลักคือการควบคุมการทำงานของส่วนต่างๆ ให้สอดคล้องกันรวมถึงประมวลผลโปรแกรมประยุกต์ต่างๆ รวมถึงการติดต่อและควบคุมอุปกรณ์ต่อพ่วงพิเศษอื่นๆ การเชื่อมต่อระหว่างไมโครคอนโทรเลอร์กับหน่วยประมวลผลพิเศษทางระบบเครือข่ายและอุปกรณ์ต่อพ่วงพิเศษอื่นๆ นั้น จะกล่าวถึงในหัวข้อถัดไป ในหัวข้อนี้จะกล่าวถึงการเชื่อมต่อไมโครคอนโทรเลอร์เข้ากับหน่วยความจำหลักและ ระบบอินพุต/เอาต์พุตสำหรับต่อพ่วงกับอุปกรณ์ต่างๆ เพื่อใช้รองรับการทำงานของซอฟต์แวร์ควบคุมระบบ

หน่วยความจำหลักของระบบนั้น ประกอบด้วยหน่วยความจำภายในและหน่วยความจำภายนอก ตามโครงสร้างของ MCS-51 หน่วยความจำภายนอกจะถูกจำแนกเป็น หน่วยความจำโปรแกรมและหน่วยความจำข้อมูล สำหรับระบบอินพุต/เอาต์พุตนั้น MCS-51 ประกอบด้วยพอร์ตทั้งสิ้น 3 ชุด แต่ทั้งนี้เนื่องจากมีการใช้สายสัญญาณร่วมกันเพื่ออ้างอิงหน่วยความจำภายนอก ทำให้มีพอร์ตที่สามารถใช้งานได้เพียง 1 ชุดเท่านั้น จึงต้องอาศัยการทำงานของ Memory Mapped I/O เพื่อให้ระบบสามารถเชื่อมต่อกับอุปกรณ์ได้มากยิ่งขึ้น

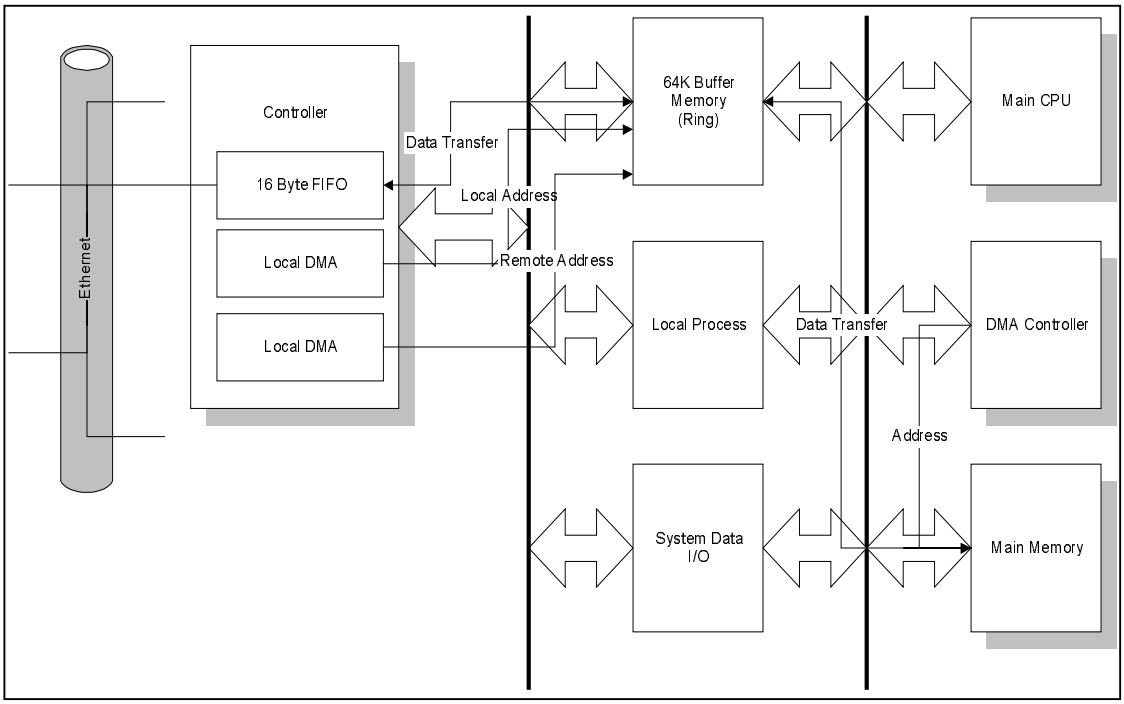
ระบบหน่วยความจำของ MCS-51 นั้น ยังแบ่งออกเป็น หน่วยความจำภายในและหน่วยความจำภายนอก (ซึ่งจะกล่าวถึงในหัวข้อที่ 3.5) โดยโปรแกรมและข้อมูลส่วนใหญ่จะเก็บอยู่ที่หน่วยความจำภายนอก สำหรับการจัดการหน่วยความจำภายนอกนั้นสามารถอธิบายได้ดังภาพที่ 10 ดังนี้ ตำแหน่ง 0000H-7FFFH ของส่วนโปรแกรมจะเป็นซอฟต์แวร์ระบบ และของส่วนข้อมูลจะใช้สำหรับติดต่อสื่อสารกับอุปกรณ์ต่างๆ เช่น หน่วยประมวลผลระบบเครือข่าย หรือ Real time clock ส่วนตำแหน่ง 8000H ถึง 0FFFFH นั้น จะใช้อ้างอิงร่วมกันทั้งโปรแกรมและข้อมูล ทั้งนี้ เพื่อประโยชน์ในการอ้างอิงและแก้ไขโปรแกรมผ่านระบบเครือข่าย

	Program	Data
0000H – 7FFFH	ROM	Memory Mapped I/O
8000H – 0FFFFH	RAM	

ภาพที่ 10 การจัดการระบบหน่วยความจำภายนอก

3.3. ส่วนเชื่อมต่อระบบเครือข่าย

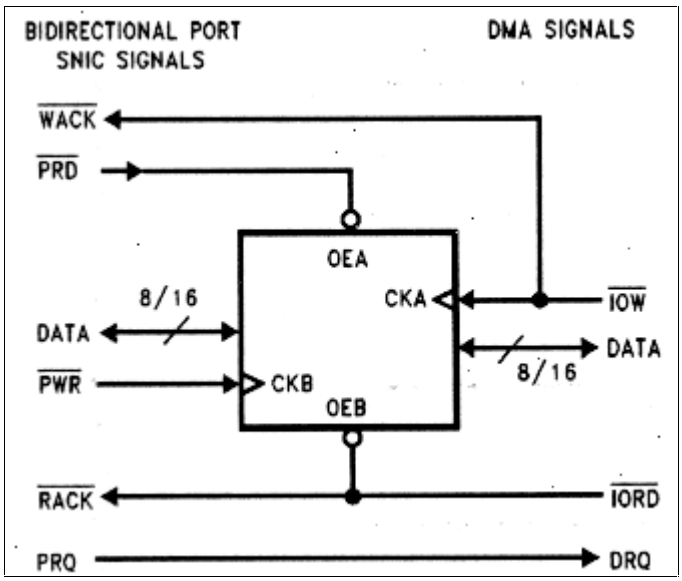
การเชื่อมต่ออุปกรณ์ควบคุมการเชื่อมต่อระบบเครือข่าย (Network Interface controller) เข้ากับระบบเครือข่ายที่มีความเร็วสูงนั้นสามารถกระทำได้โดยตรง หากแต่เป็นการยากลำบากที่จะทำการเชื่อมต่อสายสัญญาณ (Bus) ของหน่วยประมวลผลพิเศษทางระบบเครือข่ายเข้ากับไมโครคอนโทรเลอร์ซึ่งมีความเร็วต่ำเช่น MCS-51 ซึ่งใช้ในงานวิจัยนี้ การเชื่อมต่อไมโครคอนโทรลเลอร์เข้ากับอุปกรณ์ควบคุมการเชื่อมต่อเครือข่าย จึงต้องอาศัยระบบการเข้าถึงหน่วยความจำโดยตรงแบบ 2 ทาง (Dual Direct Memory Access) ทางที่ 1 คือการเชื่อมต่ออุปกรณ์ควบคุมเข้ากับหน่วยความจำชั่วคราว ซึ่งใช้ในการพักและตรวจสอบข้อมูลที่ได้รับและเตรียมข้อมูลเพื่อส่งออกบนสายสัญญาณอีเทอร์เน็ต และ ทางที่ 2 คือการย้ายข้อมูลระหว่างหน่วยความจำชั่วคราวและหน่วยความจำหลักของไมโครคอนโทรเลอร์ เพื่อใช้ในการประมวลผลหรือส่งข้อมูลต่อไป ซึ่งลักษณะดังกล่าวมีการใช้งานแพร่หลายบนเครื่องที่มีความเร็ว 4 – 33 MHz และสามารถแสดงการทำงานโดยรวมได้ดังภาพที่ 11



ภาพที่ 11 การทำงานของระบบ Dual DMA เพื่อเชื่อมต่อระบบเครือข่าย

การเชื่อมต่อแบบ DMA โดยทั่วไปนั้น มักมีหน่วยประมวลผลช่วยทำหน้าที่เป็น DMA Controller มักจะมีหน่วยประมวลผลพิเศษเพื่อช่วยในการทำงานดังกล่าว แต่บนระบบควบคุมแบบฝังตัวในงานวิจัยนี้ ไม่มีวงจรประมวลเพื่อทำหน้าที่ดังกล่าว จึงจำเป็นจะต้องสร้างมี

วงจรพิเศษบางส่วนเพื่อช่วยในการทำงานของ DMA ดังกล่าวอันประกอบด้วย Latch และสัญญาณตอบรับ (Acknowledge) ซึ่งสามารถอธิบายได้ดังภาพที่ 12



ภาพที่ 12 วงจร DMA Controller

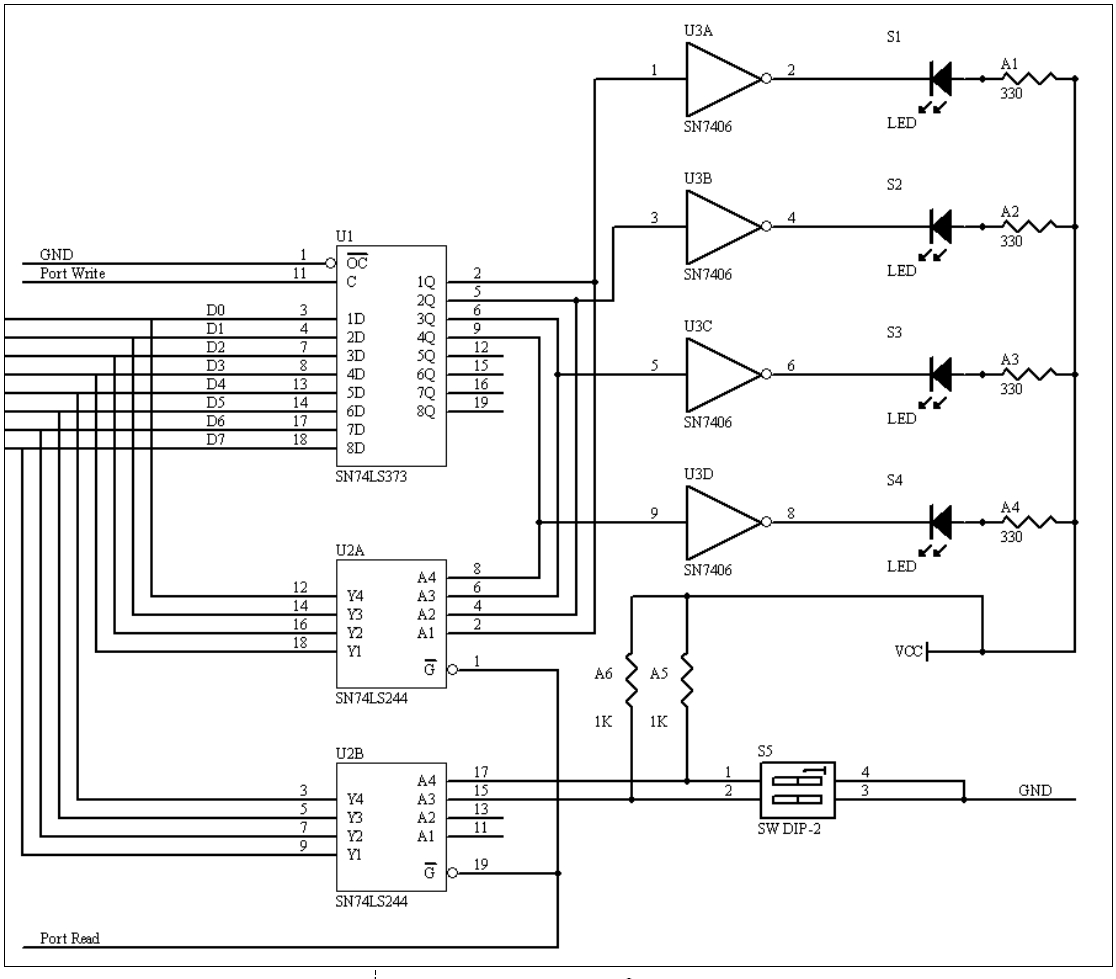
จากภาพที่ 12 พบว่าสัญญาณ $\overline{IOW}$ ใช้กำหนดการเขียนข้อมูลจากหน่วยประมวลผลหลักไปยังหน่วยประมวลผลการเชื่อมต่อระบบเครือข่าย พร้อมทั้ง Acknowledge ($\overline{WACK}$) ให้หน่วยประมวลผลเครือข่ายทราบ และ สัญญาณ $\overline{PWR}$ จะกำหนดการเขียนข้อมูลจากหน่วยประมวลผลระบบเครือข่าย โดยมีสัญญาณ PRQ แจ้งให้หน่วยประมวลผลหลักทราบถึงการเขียนข้อมูล และเมื่อหน่วยประมวลผลหลักอ่านข้อมูลด้วยสัญญาณ $\overline{IORD}$ มันจะทำการ Acknowledge ($\overline{RACK}$) ให้หน่วยประมวลระบบเครือข่ายทราบเพื่อทำการอ่านข้อมูลเข้าสู่หน่วยความจำชั่วคราวด้วยสัญญาณ $\overline{PRD}$ ต่อไป

3.4. การติดต่อกับอุปกรณ์ต่าง ๆ

การติดต่อกับอุปกรณ์ต่างๆ ที่พ่วงต่อเข้ากับส่วนประมวลผลหลักนั้นสามารถทำได้ 2 ลักษณะคือ การติดต่อผ่านพอร์ตของไมโครคอนโทรลเลอร์ และ การติดต่อผ่าน Memory Mapped I/O ซึ่งในที่นี้อุปกรณ์ส่วนใหญ่อาศัยการติดต่อผ่านระบบ Memory Mapped I/O ทั้งนี้เนื่องจากพอร์ตของไมโครคอนโทรลเลอร์นั้นมีจำนวนจำกัด และไม่เพียงพอต่อการใช้งานของอุปกรณ์ที่มาต่อพ่วง

ในงานวิจัยต้นแบบ ได้มีการพัฒนาอุปกรณ์ต่อพ่วงทดสอบซึ่งประกอบด้วยสวิตช์สำหรับอ่านค่าขนาด 2 บิต และ LED เพื่อใช้ในการแสดงผลลัพธ์ขนาด 4 บิตโดยใช้จำลองแบบ

การทำงานของสวิตช์อุปกรณ์ไฟฟ้า และ ทดสอบการทำงานของระบบ พัฒนาด้วย Latch และ Input Buffer ซึ่งสามารถแสดงได้ดังภาพที่ 13



ภาพที่ 13 วงจรทดสอบการทำงานของระบบ

3.5. การจัดสรรหน่วยความจำของระบบ

ไมโครคอนโทรลเลอร์ MCS-51 นั้น ประกอบด้วยหน่วยความจำภายใน และหน่วยความจำภายนอก หน่วยความจำภายนอกนี้สามารถจำแนกได้เป็น หน่วยความจำข้อมูลและหน่วยความจำโปรแกรม และพื้นที่บางส่วนของหน่วยความจำภายนอกประเภทข้อมูล (ตำแหน่ง 0000H-7FFFH) ถูกใช้สำหรับเป็น Memory Mapped I/O โดยการจัดสรรหน่วยความจำทั้งหมดที่ใช้ในงานวิจัยนี้สามารถแสดงได้ดังนี้

หน่วยความจำประเภทเรจิสเตอร์เป็นหน่วยความจำภายใน บางส่วนจะถูกใช้งานโดยซอฟต์แวร์ระบบ ซึ่งในที่นี้ Bank 0 จะถูกใช้งานโดยซอฟต์แวร์ระบบ ส่วน Bank 1 นั้นถูกใช้งานโดยรูทีนพิเศษที่ระบบจัดให้สำหรับการเรียกใช้งานของโปรแกรมประยุกต์ ดังนั้นหากผู้พัฒนา

ต้องการจะพัฒนาโปรแกรมประยุกต์เพื่อเพิ่มเติมความสามารถของระบบควบคุม ผู้พัฒนาสามารถเลือกใช้งานเรจิสเตอร์ใน Bank 2 และ Bank 3 ได้

ตารางที่ 2 การจัดสรรหน่วยความจำเรจิสเตอร์

Register / Bank

00H – 07H (Bank 0)	STATIC SYSTEM,ISR
08H – 0FH (Bank 1)	Dynamic for each routine
10H – 17H (Bank 2)	
18H – 1FH (Bank 3)	

หน่วยความจำภายในที่มีการอ้างอิงแบบ Bit Addressable และ Byte Addressable นั้น บางส่วนจะถูกใช้เพื่อช่วยในการคำนวณค่าต่างๆ ในขณะที่บางส่วนจะถูกใช้งานเพื่อทำการเก็บสถานะของระบบ ทั้งนี้ สถานะของระบบนั้นมีความหมายครอบคลุมตั้งแต่สถานะของหน่วยประมวลผลต่าง ค่าที่ได้จากการตรวจสอบความถูกต้องของข้อมูลต่างๆ ประเภทของข้อมูลที่กำลังประมวลผลอยู่ และ สถานะของการประมวลผล

ตารางที่ 3 การจัดสรรหน่วยความจำ Bit Addressable

Bit Addressable

Byte Address	Bit							
	7	6	5	4	3	2	1	0
20H (SERIO)								
21H (NIC-ISR)	RST	RDC	CNT	OVW	TXE	RXE	PTX	PRX
22H (TCP STATUS)			URG	ACK	PSH	RST	SYN	FIN
23H (SESSION STATUS)			NOT ESEQ	ACK FAIL	WACK	WFIN	TCP CON	TCP INUSE
24H (HTTP FLAG)					HTTP CLOSE	HTTP SEND	HTTP IN	HTTP START
25H – 27H	Not Used							
28H (CNT0)								
29H (CNT1)								
2AH – 2BH (TMP)								
2CH-2F (SEQ0-3)								
2DH (SEQ1)								
2EH (SEQ2)								
2FH (SEQ3)								

ตารางที่ 4 การจัดสรรหน่วยความจำ Byte Addressable

Byte Addressable

	+0	+1	+2	+3	+4	+5	+6	+7	
00H	Register (Bank 0)								07H
08H	Register (Bank 1)								0FH
10H	Register (Bank 2)								17H
18H	Register (Bank 3)								1FH
20H	SER I/O	ISR	TCP STAT	SESS STAT	HTTP STAT				27H
28H	CNT0	CNT1	TMP	STR TMP	SEQ0	SEQ1	SEQ2	SEQ3	2FH
30H	BTCP	TCP LEN	IP H LEN	ACK0	ACK1	ACK2	ACK3	Data H LEN	37H
38H	DataL LEN	OutH LEN	OutL LEN	TCP PORT	IP0	IP1	IP2	IP3	3FH
40H									47H
48H									4FH
50H	STACK								57H
58H	STACK								5FH
60H	STACK								67H
68H	STACK								6FH
70H	STACK								77H
78H	STACK								7FH

หน่วยความจำภายนอกนั้นบางส่วนถูกใช้งานเป็น ที่พักข้อมูลชั่วคราวสำหรับการรับส่งข้อมูลผ่านระบบเครือข่าย และบางส่วนถูกใช้งานสำหรับการประมวลผลสคริปต์รวมถึงเป็นที่เก็บสคริปต์หรือค่าที่กำหนดโดยผู้ใช้งาน นอกจากนี้ในระหว่างการพัฒนาหน่วยความจำภายนอกยังถูกใช้เป็นที่เก็บโปรแกรม เพื่อความสะดวกในการแก้ไขและปรับปรุงค่าต่างๆ ด้วย

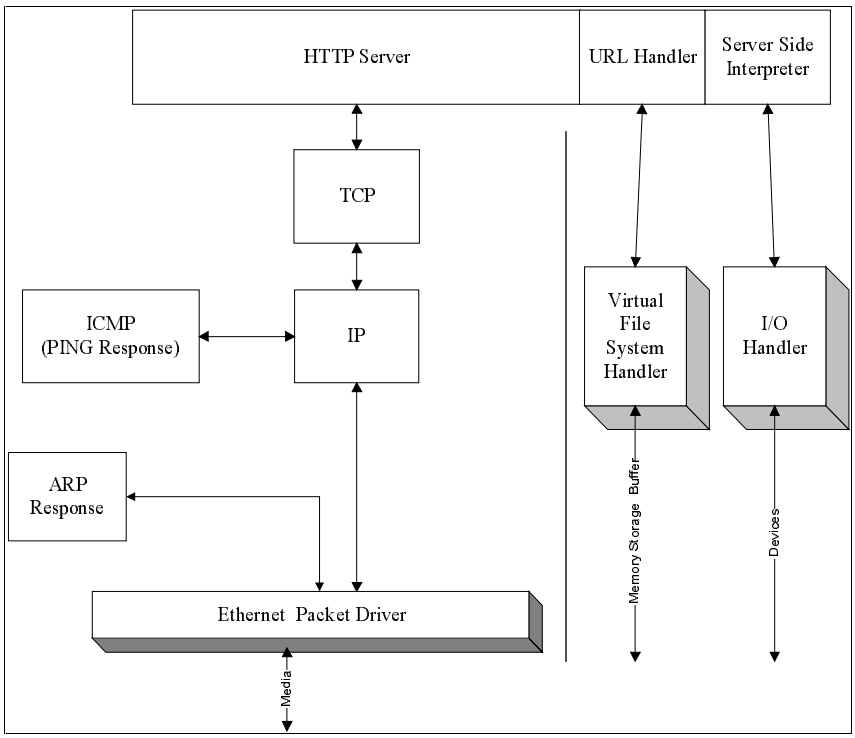
ตารางที่ 5 การจัดสรรหน่วยความจำภายนอก

External Memory

0000H - 7FFFH	MEMORY MAPPED I/O
8000H - 8FFFH	SYSTEM SOFTWARE
9000H - 9FFFH	SYSTEM SOFTWARE
A000H – AFFFH	SYSTEM SOFTWARE
B000H – BFFFH	USER SCRIPT
C000H – CFFFH	PHP SCRIPT / TCP CHECKSUM TEMP
D000H – DFFFH	HTTP INPUT / OUTPUT BUFFER
E000H – EFFFH	NIC OUTPUT BUFFER
F000H – FFFFH	NIC INPUT BUFFER / OUTPUT HEADER TEMPLATE

บทที่ 4
การพัฒนาซอฟต์แวร์เชื่อมต่อระบบเครือข่าย

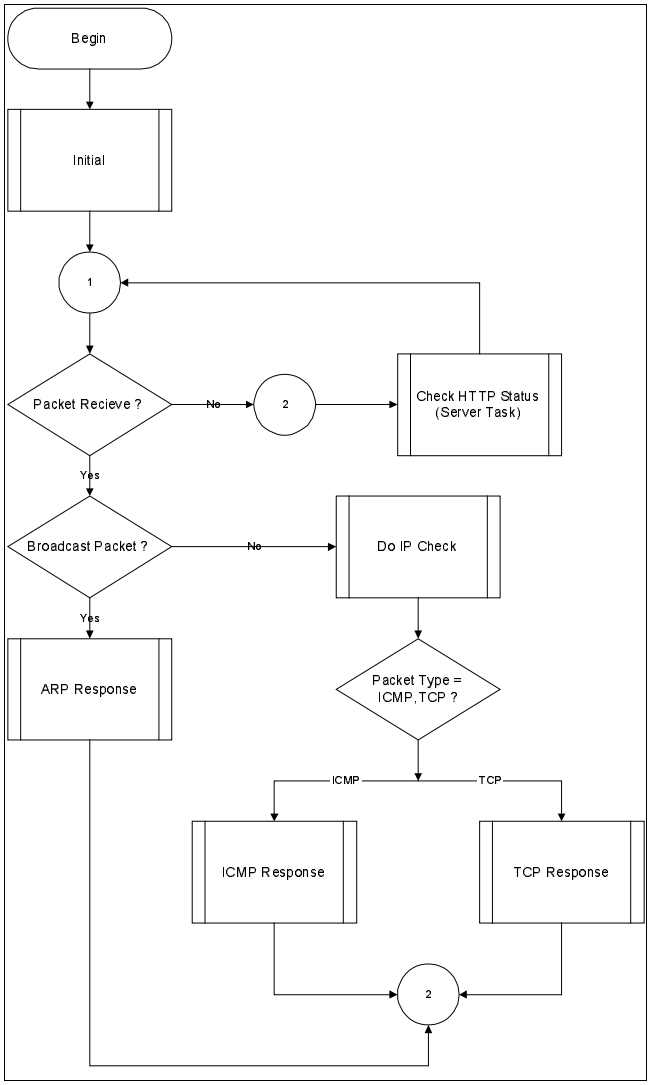
ซอฟต์แวร์เพื่อใช้ในการควบคุมและเชื่อมต่อระบบเครือข่ายนั้น บนระบบงานทั่วไปมักถูกจัดการโดยซอฟต์แวร์ระบบชนิดพิเศษ หรือระบบปฏิบัติการ (Operating System) ทั้งนี้เนื่องจากในงานวิจัยดังกล่าว มิได้ใช้ระบบปฏิบัติการบนระบบควบคุมแบบฝังตัวเข้าช่วยแต่อย่างใด การควบคุมงานทุกอย่างจึงถูกจัดการโดยซอฟต์แวร์ที่พัฒนาขึ้นโดยเฉพาะ ซึ่งประกอบไปด้วยซอฟต์แวร์ติดต่ออุปกรณ์การควบคุมการเชื่อมต่อระบบเครือข่าย (NIC Driver) และซอฟต์แวร์เพื่อช่วยในการจัดการการเชื่อมต่อระบบเครือข่ายตามมาตรฐาน TCP/IP เฉพาะส่วนที่จำเป็นสำหรับการสร้างระบบเว็บเซิร์ฟเวอร์แบบฝังตัว ซึ่งโครงสร้างของซอฟต์แวร์ส่วนที่เกี่ยวข้องทั้งหมดสามารถอธิบายได้ดังภาพที่ 14



ภาพที่ 14 ส่วนประกอบทางซอฟต์แวร์ของระบบ

การทำงานทางซอฟต์แวร์ของระบบแบ่งออกเป็นระดับชั้น ตามการทำงานได้โดยในระดับล่างจะเป็นซอฟต์แวร์ติดต่ออุปกรณ์ควบคุมการเชื่อมต่อระบบเครือข่ายหรือ Driver ซึ่งจะทำหน้าที่ควบคุมการถ่ายโอนข้อมูล ระหว่างหน่วยความจำหลักและหน่วยความจำชั่วคราวของอุปกรณ์ควบคุมการเชื่อมต่อระบบเครือข่าย เมื่อข้อมูลหรือ Packet ผ่านเข้าสู่ระบบ ซอฟต์แวร์ Driver ดังกล่าว จะทำการตรวจสอบว่าข้อมูลใน Packet นั้นเป็นข้อมูลประเภท Broadcast หรือไม่

หากข้อมูลที่ได้รับเป็นข้อมูลประเภท Broadcast ระบบจะส่งต่อให้กับโปรแกรม ARP Response เพื่อตอบรับต่อไป ส่วนกรณีที่ข้อมูลที่ได้รับมิได้เป็น Packet ประเภท Broadcast ระบบจะส่งต่อให้กับโปรแกรม IP เพื่อตรวจสอบและจำแนกข้อมูลซึ่งหากข้อมูลที่ได้รับถูกจำแนกเป็น PING โปรแกรม ICMP จะถูกเรียกเพื่อสร้าง Ping Response และ กรณีที่ข้อมูลที่ได้รับถูกจำแนกเป็น TCP โปรแกรม TCP จะตรวจสอบและแยกข้อมูลเพื่อส่งต่อให้กับซอฟต์แวร์ระบบเว็บเซิร์ฟเวอร์ต่อไป ในทำนองเดียวกันหากเป็นการส่งข้อมูล ข้อมูลจะถูกส่งผ่านขั้นตอนต่างๆ จากบนลงล่างเพื่อประมวล ตรวจสอบความถูกต้องและส่งข้อมูลออกสู่ระบบเครือข่ายในที่สุด ซึ่งจากลักษณะการทำงานดังกล่าว ทำให้การพัฒนาซอฟต์แวร์ ถูกแบ่งเป็นแต่ละส่วน และการทำงานระหว่างโปรแกรมย่อยต่างๆ ถูกควบคุมหรือเรียกใช้งานจากโปรแกรมหลัก ซึ่งรายละเอียดของซอฟต์แวร์ส่วนโปรแกรมหลักสามารถแสดงได้ดังภาพที่ 15



ภาพที่ 15 การทำงานของโปรแกรมหลัก

เมื่อเริ่มต้นการทำงาน โปรแกรมหลักจะทำการเตรียมข้อมูล (Initialize) และตั้งค่าสถานะต่างๆ ของระบบ ซึ่งในที่นี้รวมความถึงการตั้งค่า Stack และการตั้งค่าของหน่วยควบคุมการเชื่อมต่อระบบเครือข่ายด้วย จากนั้นโปรแกรมหลักจะทำการตรวจสอบว่าระบบได้รับ Packet จากหน่วยควบคุมการเชื่อมต่อเครือข่ายหรือไม่ หากไม่ได้รับก็จะทำการประมวลผลงานในระดับ Application อันได้แก่ซอฟต์แวร์เว็บเซิร์ฟเวอร์ ทั้งนี้หากต้องการให้ระบบทำงานอื่นๆ สามารถแทรกการทำงานเพิ่มเติมได้ที่ตำแหน่งนี้ ในกรณีที่มี Packet รออยู่ในหน่วยความจำ ซอฟต์แวร์จะประมวลผล Packet ที่รอบอยู่นั้น ตามแนวการทำงานดังอธิบายข้างต้น ซึ่งซอฟต์แวร์หลักจะประมวลผลเช่นนี้เรื่อยไป

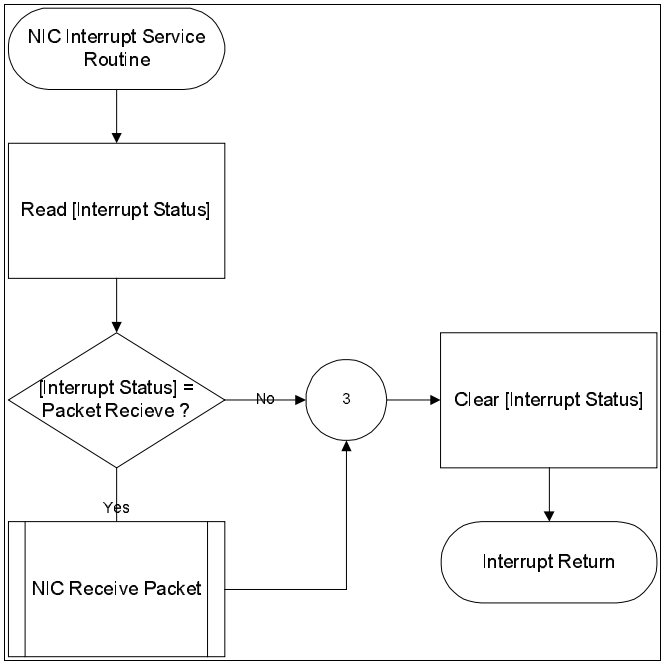
4.1. ซอฟต์แวร์ติดต่อหน่วยควบคุมการเชื่อมต่อระบบเครือข่าย

การเชื่อมต่อระบบเครือข่ายตามมาตรฐานอีเทอร์เน็ตนั้น ในที่นี้ได้อาศัยคุณสมบัติของหน่วยควบคุมการเชื่อมต่อระบบเครือข่าย ดังนั้นหน้าที่ของซอฟต์แวร์คือการรับและส่งข้อมูลระหว่างหน่วยความจำชั่วคราวของหน่วยควบคุมดังกล่าว และหน่วยความจำของหน่วยประมวลผลหลักเพื่อให้หน่วยประมวลผลหลักทำการประมวลผลต่างๆ

ตามมาตรฐานการทำงานของหน่วยควบคุมการเชื่อมต่อระบบเครือข่ายนั้น การโอนถ่ายข้อมูลเข้าและออกจะต้องการทำผ่านระบบการเข้าถึงหน่วยความจำโดยตรง (Direct Memory Access) เท่านั้น การทำงานของซอฟต์แวร์ในส่วนนี้จึงเป็นเพียงการสื่อสารเพื่อโอนย้ายข้อมูล ที่ได้ทำการถอดระหัสจากหน่วยประมวลการเชื่อมต่อระบบเครือข่ายแล้ว และส่งข้อมูลที่ต้องการออกสู่ระบบเครือข่ายให้กับหน่วยควบคุมผลการเชื่อมต่อระบบเครือข่ายเท่านั้น โดยอาศัยคุณสมบัติของวงจรเข้าถึงหน่วยความจำโดยตรงดังแสดงในบทที่ 3 หน้าที่ของซอฟต์แวร์ในส่วนนี้รวมถึงการจัดการสัญญาณขัดจังหวะ (Interrupt) ของหน่วยควบคุมผลพิเศษทางระบบเครือข่ายด้วย จากลักษณะการทำงานดังกล่าว ซอฟต์แวร์เพื่อช่วยประมวลผลในระดับนี้ จึงถูกจำแนกออกเป็นโปรแกรมย่อย 3 ส่วน ซึ่งได้แก่ โปรแกรมย่อยเพื่อให้บริการสัญญาณขัดจังหวะ (Interrupt Service Routine) โปรแกรมย่อยเพื่อรับข้อมูลจากระบบเครือข่าย และ โปรแกรมย่อยเพื่อการส่งข้อมูลเข้าสู่ระบบเครือข่าย ซึ่งการทำงานของโปรแกรมย่อยแต่ละส่วนนั้น สามารถแสดงได้ดังภาพที่ 16, 17 และ 18 ตามลำดับ

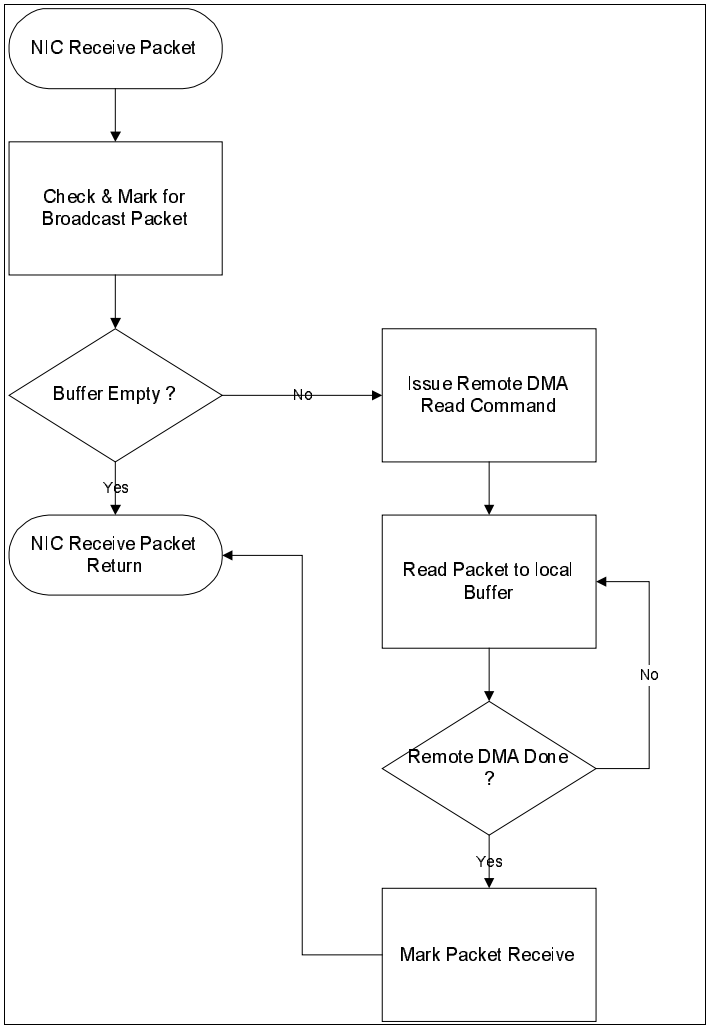
เมื่อได้รับสัญญาณขัดจังหวะ (Interrupt) จากหน่วยประมวลผลการเชื่อมต่อระบบเครือข่าย ระบบจะทำการเรียกใช้งานโปรแกรมย่อยเพื่อให้บริการสัญญาณขัดจังหวะ (Interrupt Service Routine) ซึ่งในบริการดังกล่าวจะเริ่มต้นด้วยการ จำแนกประเภทของสัญญาณขัดจังหวะ ซึ่งหากเป็นสัญญาณขัดจังหลวะของการรับข้อมูล โปรแกรมจะตรวจสอบว่ามี

Packet ค้างอยู่ในหน่วยความจำชั่วคราวของหน่วยควบคุมการเชื่อมต่อระบบเครือข่ายหรือไม่ หากยังมีข้อมูลตกค้างอยู่จึงย้ายข้อมูลผ่านระบบเข้าหน่วยความจำโดยตรง มายังหน่วยความจำหลักของระบบโดยอาศัยการทำงานของโปรแกรมย่อยเพื่อรับข้อมูล จากนั้นจึงลบค่าสถานะของสัญญาณขัดจังหวะ เพื่อรอรับสัญญาณขัดจังหวะครั้งต่อไป (ดูการทำงานประกอบในภาพที่ 16)



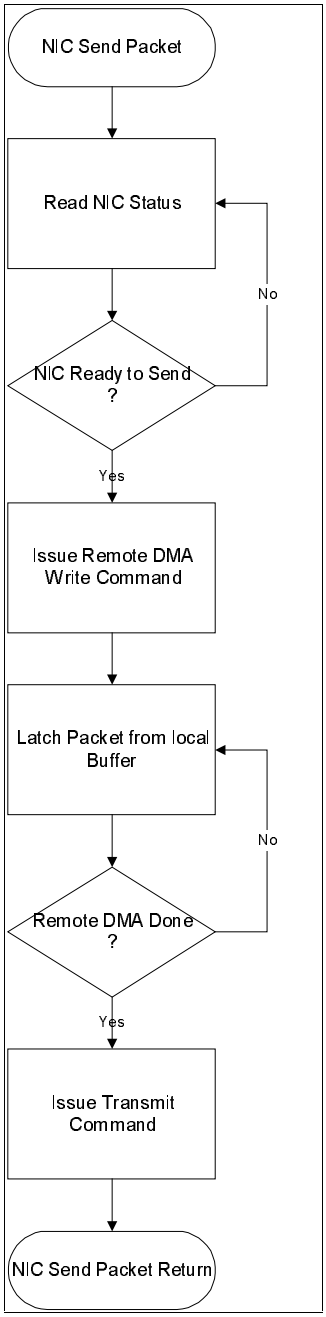
ภาพที่ 16 โปรแกรมย่อยเพื่อให้บริการสัญญาณขัดจังหวะ

การทำงานในโปรแกรมย่อยเพื่อรับข้อมูลจากระบบเครือข่ายนั้น สามารถอธิบายได้ตามภาพที่ 17 ดังนี้ เริ่มต้นด้วยการเก็บสถานะของข้อมูลที่ได้รับว่าเป็นข้อมูลประเภท Broadcast หรือไม่ ทั้งนี้เพื่อการใช้งานในโปรแกรมหลัก จากนั้นโปรแกรมจะตรวจสอบว่า มีข้อมูลอยู่ภายในหน่วยความจำชั่วคราวของหน่วยควบคุมการเชื่อมต่อระบบเครือข่ายหรือไม่ หากยังคงมีอยู่โปรแกรมจะร้องขอหน่วยควบคุมการเชื่อมต่อระบบเครือข่าย ให้มีการย้ายข้อมูลเข้าสู่หน่วยประมวลผลหลัก โดยการส่งคำสั่ง Remote DMA Read และรอรับข้อมูลจนเสร็จสิ้น เมื่อเสร็จสมบูรณ์ จึงทำการแจ้งให้ซอฟต์แวร์ระดับที่สูงกว่าหรือโปรแกรมหลักทราบโดยตั้งค่าตัวแปร Packet Receive



ภาพที่ 17 โปรแกรมย่อยเพื่อรับข้อมูลจากระบบเครือข่าย

ในกรณีของการส่งข้อมูลนั้น เมื่อมีการเรียกขอจากซอฟต์แวร์ระดับที่สูงกว่าให้ส่งข้อมูล โปรแกรมย่อยเพื่อส่งข้อมูลเข้าสู่ระบบเครือข่ายจะตรวจสอบความพร้อมในการส่งข้อมูลของหน่วยควบคุมการเชื่อมต่อระบบเครือข่าย และรอจนกว่าจะพร้อมส่งข้อมูล จากนั้นเมื่อหน่วยควบคุมพร้อมที่จะส่งข้อมูล โปรแกรมจะย้ายข้อมูลที่ต้องการจะส่งจากหน่วยความจำหลัก ไปยังหน่วยความจำชั่วคราวภายของหน่วยควบคุมการเชื่อมต่อระบบเครือข่าย โดยการส่งคำสั่ง Remote DMA Write ให้กับหน่วยควบคุมการเชื่อมต่อระบบเครือข่าย และส่งค่าจนเสร็จสมบูรณ์ จากนั้นจึงสั่งให้หน่วยควบคุมการเชื่อมต่อระบบเครือข่ายส่งข้อมูลให้ ซึ่งหน่วยควบคุมการเชื่อมต่อระบบเครือข่ายจะสร้าง Frame Check sum และ ส่งข้อมูลเข้าสู่ระบบเครือข่ายดังอธิบายแล้วในบทที่3 เมื่อเสร็จทุกขั้นตอนดังกล่าว ถือเป็นการเสร็จสิ้นการทำงานส่งข้อมูล

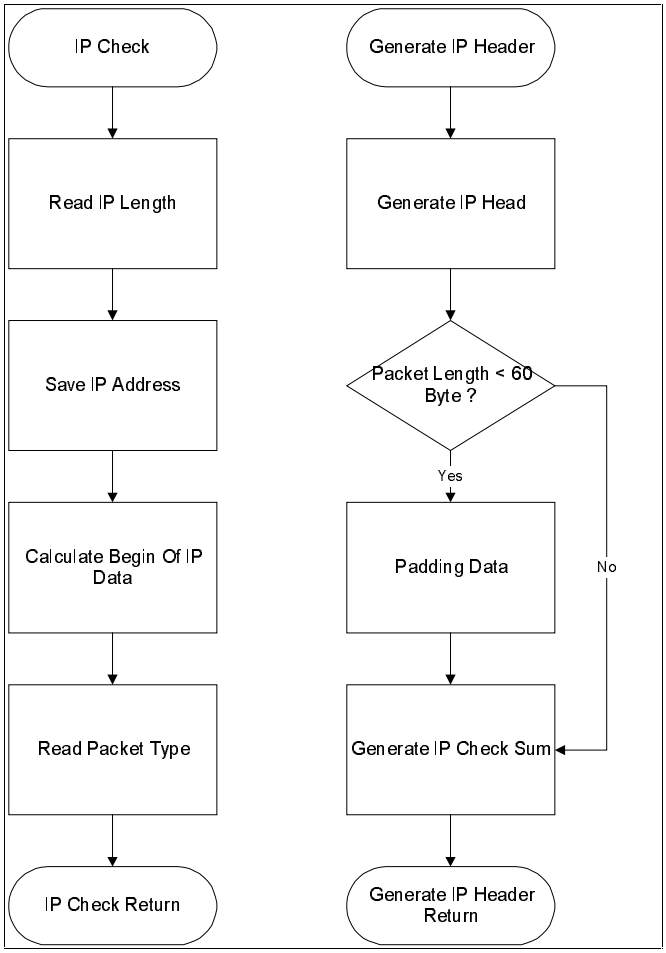


ภาพที่ 18 โปรแกรมย่อยเพื่อการส่งข้อมูลเข้าสู่ระบบเครือข่าย

4.2. IP

หน้าที่การทำงานของระบบซอฟต์แวร์ตามมาตรฐาน IP นั้น ประกอบด้วยการสร้าง IP Packet เพื่อระบุต้นทางและปลายทาง การวัดความยาวของข้อมูลที่ส่ง และการตรวจสอบความถูกต้องของหัว IP โดยสามารถแบ่งการทำงานตามทิศทางของการไหลของข้อมูลได้เป็นการรับและการส่งข้อมูล ซึ่งโปรแกรมย่อยที่เกี่ยวกับการทำงานดังกล่าว สามารถอธิบายได้ดังภาพที่ 19 ดังนี้ ด้านซ้ายแสดงการทำงานของโปรแกรมย่อยเพื่อตรวจสอบความถูกต้องของข้อมูลที่ได้

รับ และ ด้านขวาแสดงการทำงานของโปรแกรมย่อยเพื่อการสร้าง IP Header สำหรับใช้ในการส่งข้อมูล



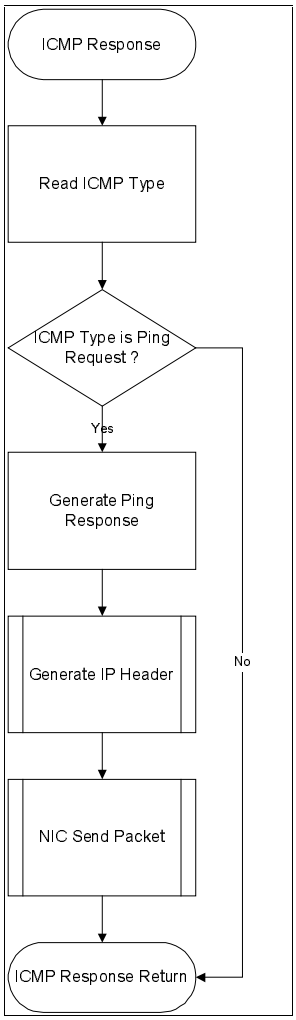
ภาพที่ 19 โปรแกรมย่อยในการประมวลผล IP

กรณีของการรับข้อมูล เมื่อระบบได้รับข้อมูลจากหน่วยประมวลผลระบบเครือข่าย ระบบจะคำนวณหาความยาวและตำแหน่งเริ่มต้นของข้อมูล และจำแนกประเภท Packet ว่าเป็น TCP หรือ ICMP เพื่อให้โปรแกรมหลักประมวลผลต่อไป

กรณีการส่งข้อมูล เมื่อ ICMP หรือ TCP ต้องการส่งข้อมูล IP จะทำการสร้างหัว IP เพื่อระบุต้นทางและปลายทางของข้อมูล ทั้งนี้หากข้อมูลที่ต้องการส่งมีขนาดสั้นเกินไป โปรแกรมจะทำ Padding หรือ เติมข้อมูลให้มีความยาวเพียงพอ จากนั้นจะทำการคำนวณหาความยาวทั้งหมดของข้อมูลที่ต้องการจะส่ง คำนวณ และบันทึกค่า IP Check sum ในส่วนหัวของ IP บน Packet ที่จะส่งออก

4.3. ICMP

ICMP นั้นโดยทั่วไปจะใช้ตรวจสอบสถานะของระบบเครือข่าย IP แต่ที่นี้จะทำหน้าที่เพียง Ping Response เพื่อตอบรับข้อมูลประเภท Ping ของ ICMP เท่านั้นการตอบรับข้อมูลประเภท ICMP เป็นเพียงการสลับผู้รับและผู้ส่งในส่วนหัวของ Packet เท่านั้น ดังนั้นการทำงานของซอฟต์แวร์ในส่วนนี้จึงไม่มีความซับซ้อนใดๆ เป็นเพียงการสลับที่ข้อมูลตามปกติ และสามารถอธิบายได้ดังภาพที่ 20

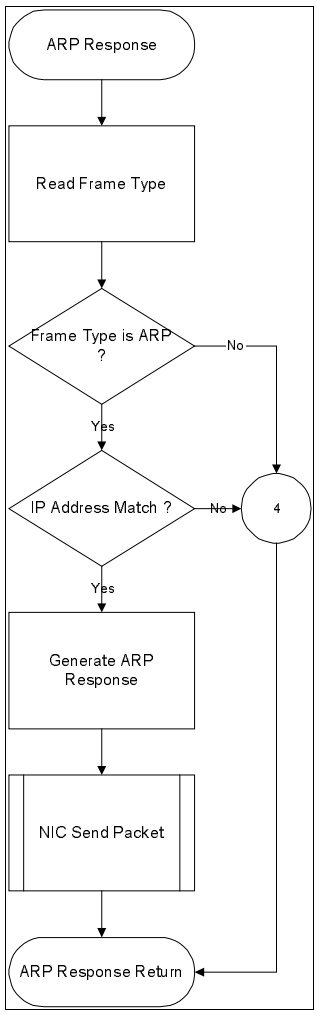


ภาพที่ 20 โปรแกรมย่อมเพื่อประมวลผล ICMP

การทำงานของโปรแกรมย่อยเพื่อสร้าง Ping Response ของระบบเว็บเซิร์ฟเวอร์แบบฝังตัวในที่นี้ เริ่มจากการตรวจสอบว่าข้อมูลที่ได้รับนั้น เป็นข้อมูลประเภท Ping Request หรือไม่ หากเป็นระบบจึงจะสร้าง Ping Response และ ร้องขอให้หน่วยควบคุมการเชื่อมต่อระบบเครือข่ายทำการส่งข้อมูลเข้าสู่ระบบเครือข่าย

4.4. ARP

ในที่นี้หมายถึงการตอบรับข้อมูลประเภท ARP อันเป็นการถามหาหมายเลข MAC Address ของหมายเลข IP ที่กำหนด โดยโปรแกรมในส่วนนี้จะตรวจสอบว่าข้อมูลที่ได้รับนั้นมีหมายเลข IP ตรงกับหมายเลข IP ของระบบหรือไม่ หากตรงกันระบบจะตอบหมายเลข IP ดังกล่าวด้วย MAC Address ของตน หากไม่ตรงกัน ระบบจะทำการละทิ้งข้อมูลชุดนั้นไป ซึ่งการทำงานดังกล่าวสามารถอธิบายเป็นแผนภาพและลำดับการทำงานได้ภาพที่ 21

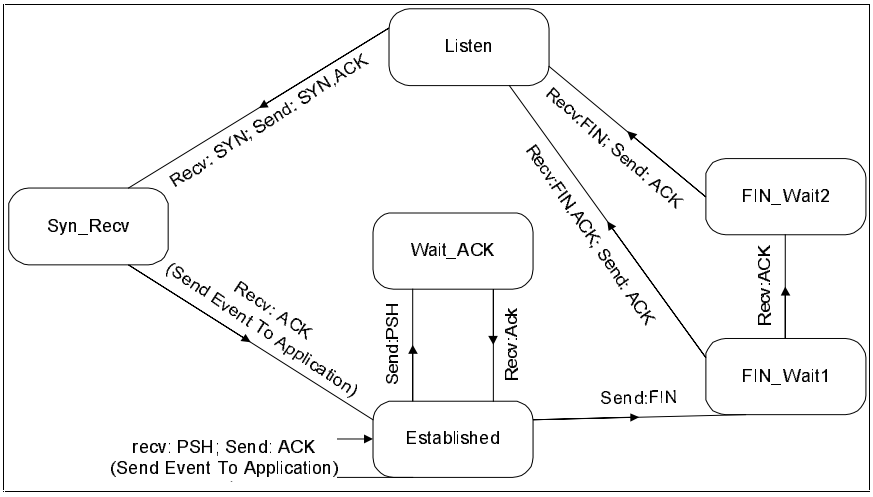


ภาพที่ 21. โปรแกรมย่อยเพื่อประมวลผล ARP

4.5. TCP

ระบบ TCP นั้นมีการทำงานที่ค่อนข้างซับซ้อนทั้งนี้เนื่องจากต้องมีการตรวจสอบสถานะและความถูกต้องของข้อมูลที่ทำการสื่อสารตลอดเวลา ในที่นี้การทำงานของ TCP จะเป็น

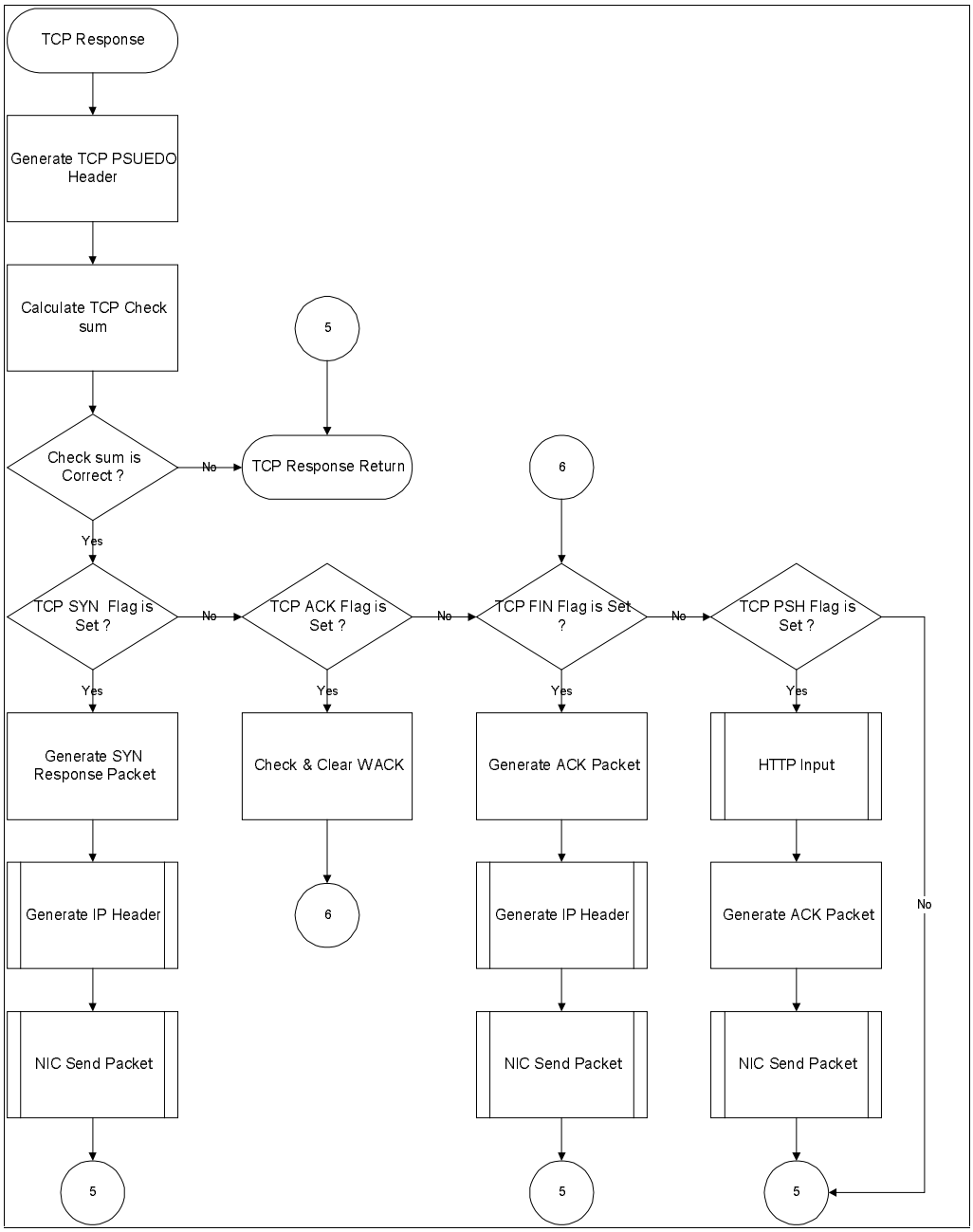
การทำงานในลักษณะของเซิร์ฟเวอร์ซึ่งจะรอฟังสัญญาณการร้องขอจากผู้ใช้งาน ลักษณะการทำงานดังกล่าวสามารถอธิบายโดยย่อได้ดังภาพที่ 22



ภาพที่ 22 สถานะการทำงานของ TCP บนระบบเว็บเซิร์ฟเวอร์แบบฝังตัว

จากภาพที่ 22 จะพบว่าเมื่อมีการรับข้อมูลทุกครั้ง โปรแกรมจะตรวจสอบความถูกต้องของข้อมูลที่ได้รับและตอบรับด้วยการส่ง ACK เสมอ ซึ่งการส่ง ACK อาจส่งเพียง ACK เดี่ยวๆ หรือ ส่ง ACK ไปพร้อมกับการส่งข้อมูล (SYN, FIN หรือ PSH) ก็ได้ เมื่อเริ่มต้นระบบ TCP จะอยู่ในสถานะ Listen หากข้อมูลที่ได้รับเป็น SYN โปรแกรมจะตอบรับด้วย SYN และ ACK การได้รับข้อมูล จากนั้นรอจนกว่าจะได้รับข้อมูล ACK เพื่อยืนยันการ SYN ของเซิร์ฟเวอร์เมื่อเสร็จสิ้นตามขั้นตอนดังกล่าวจึงจะถือว่าการเปิด Session สมบูรณ์ ซึ่งที่สถานะนี้ระบบพร้อมสำหรับการรับและส่งข้อมูล โดยกรณีที่ได้รับข้อมูลโปรแกรมจะเก็บข้อมูลลงสู่หน่วยความจำและส่ง ACK เพื่อยืนยันการได้รับข้อมูล ส่วนกรณีการส่งนั้น ระบบจะส่งข้อมูลและรอคอยจนกว่าจะได้รับ ACK จึงจะทำงานต่อไป เมื่อเสร็จสิ้นการสื่อสาร โปรแกรมจะส่ง FIN และรอคอยจนกว่าจะได้รับสัญญาณ ACK และ FIN ตอบรับ เป็นการเสร็จสิ้นการสื่อสารข้อมูล นอกจากนี้เพื่อป้องกันความผิดพลาดที่เกิดขึ้นจากการสื่อสาร จึงมีการกำหนดเวลาในการสื่อสารแต่ละครั้ง ไม่เกิน 15 วินาทีโดยเริ่มจับเวลาตั้งแต่มีการเริ่ม Established หากระบบไม่มีการเปลี่ยนสถานะเป็น Listen ภายใน 15 วินาที โปรแกรมจะยกเลิกการสื่อสารในครั้งนั้นทั้งหมด ทั้งนี้อาศัยการทำงานของ Timer Interrupt ช่วยในการประมวลผลดังกล่าว (รายละเอียดเพิ่มเติมค้นคว้าได้จากคู่มือ MCS-51 ทั่วไป) การทำงานของซอฟต์แวร์ส่วนที่เกี่ยวข้องกับการทำงานดังกล่าวข้างต้นสามารถอธิบายได้ดังภาพที่ 23

เมื่อเริ่มต้นการประมวลผลเพื่อการตอบรับข้อมูลประเภท TCP ระบบจะตรวจสอบความถูกต้องของข้อมูล TCP โดยการสร้าง PSUEDO HEADER และคำนวณ Check sum ของข้อมูลที่ได้รับ จากนั้นจึงประมวลผลตาม Flag ของข้อมูลที่ได้รับ ได้แก่ SYN, ACK, FIN และ PSH

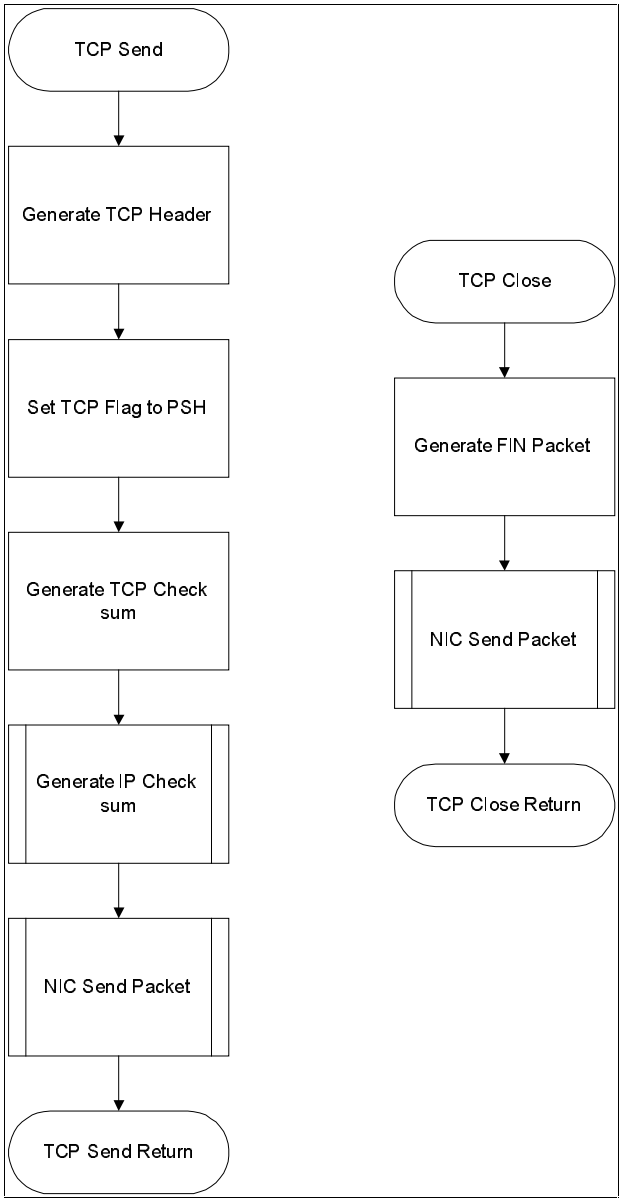


ภาพที่ 23 โปรแกรมย่อยเพื่อประมวลผล TCP

เมื่อซอฟต์แวร์ในระดับที่สูงกว่า ต้องการส่งข้อมูลบน TCP จะร้องขอให้ TCP ส่ง ซึ่งในขั้นตอนนี้ โปรแกรม TCP Send จะทำการสร้าง TCP Header โดยกำหนด Flag เป็น PSH สำหรับการส่งข้อมูล สร้าง TCP Check sum จากนั้นจึงร้องขอให้ IP สร้าง Check sum เมื่อเสร็จ

สิ้นการทำงานดังกล่าว จึงขอให้ซอฟต์แวร์ Driver ทำการส่งข้อมูลออกสู่ระบบเครือข่าย (ดังแสดงในภาพที่ 24 ซ้าย)

หลักจากเสร็จสิ้นการทำงานทั้งหมด ซอฟต์แวร์เซิร์ฟเวอร์จะต้องร้องขอให้ TCP ปิดการสื่อสาร โดยการเรียกโปรแกรมย่อย TCP Close ซึ่งการทำงานของโปรแกรมดังกล่าวประกอบด้วยการสร้าง TCP FIN Packet และส่งออกสู่ระบบเครือข่าย ดังแสดงในทางขวาของภาพที่ 24



ภาพที่ 24 โปรแกรมย่อยเพื่อส่งข้อมูลบน TCP และปิดการสื่อสารบน TCP

อย่างไรก็ตาม การสร้างซอฟต์แวร์ทั้งหมด เป็นเพียงต้นแบบ มิได้มีการสร้างหรือพัฒนา TCP อย่างสมบูรณ์แบบ หากแต่เป็นเพียงการพัฒนาเพียงซอฟต์แวร์ส่วนที่จำเป็นสำหรับการใช้งานเพื่อเป็นเซิร์ฟเวอร์แบบฝังตัวออกมาเท่านั้น ซอฟต์แวร์การสื่อสาร TCP/IP ในที่นี้จึงไม่สามารถทำงานซับซ้อน เช่น IP fragment หรือ TCP Sliding Window ได้

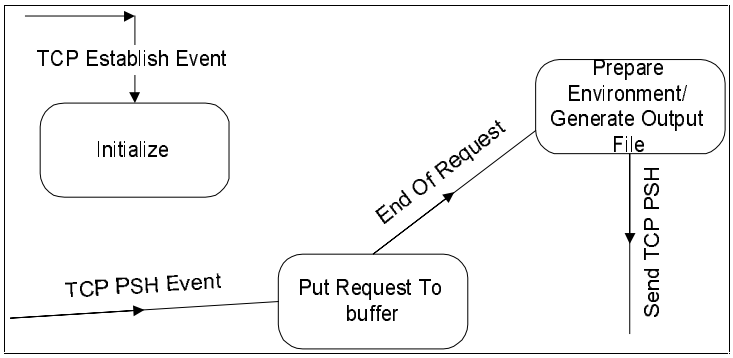
บทที่ 5
การพัฒนาระบบเว็บเซิร์ฟเวอร์ และระบบการจำแนกผู้ใช้

ระบบซอฟต์แวร์เว็บเซิร์ฟเวอร์นั้นจะทำงานโดยอาศัยมาตรฐาน HTTP/1.1 เป็นหลัก การทำงานของโปรแกรมประกอบด้วยการขอข้อมูล (Request) การตอบรับ (Response) ทั้งนี้รวมถึงระบบรักษาความปลอดภัยแบบซึ่งอาศัยวิธีการจำแนกผู้ใช้งานแบบ ชื่อและรหัสผ่าน ตามมาตรฐานการพิสูจน์ตัวจริงแบบเบื้องต้น (Basic Authentication)

5.1. สถานะและขั้นตอนการทำงาน

การทำงานของระบบเว็บเซิร์ฟเวอร์ ประกอบด้วยการเตรียมสภาวะแวดล้อมต่างๆ การถอดข้อความ Request เพื่อให้ได้รายละเอียดต่างๆ ของผลลัพธ์ที่ต้องการ จากนั้นจึงจะเป็นการสร้างผลลัพธ์ที่ได้ โดยการทำงานนั้นสามารถอธิบายได้ดังนี้

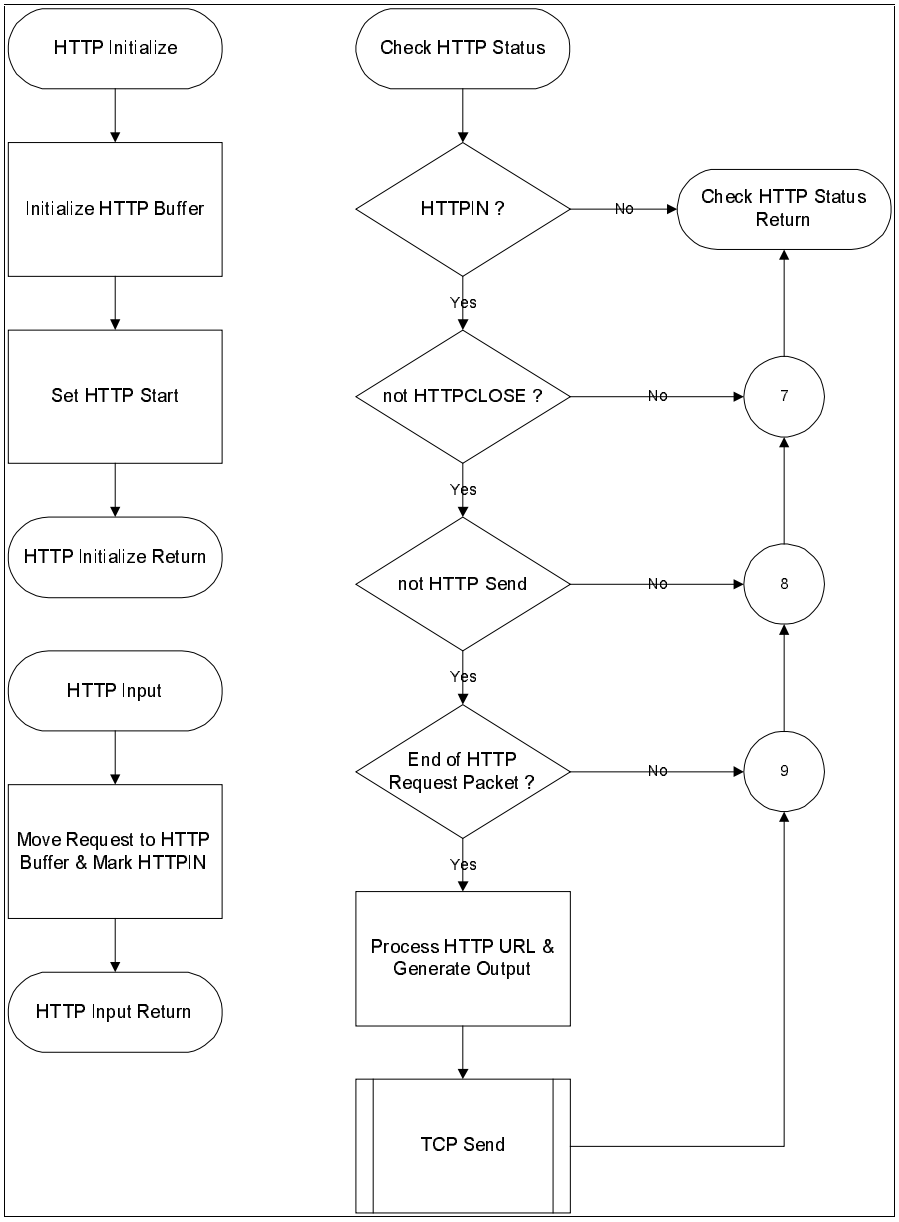
เมื่อ TCP เริ่มต้นการสื่อสารโปรแกรมระบบแม่ข่ายจะทำการเตรียมสภาวะแวดล้อมต่างๆ เพื่อเริ่มต้นการประมวลผลข้อมูล และเมื่อ TCP ได้รับข้อมูลโปรแกรมจะเก็บการ Request ไว้ในหน่วยความจำของเว็บเซิร์ฟเวอร์และคอยตรวจสอบจนกว่าจะพบว่าการ Request จบสิ้นแล้ว ซึ่งเมื่อการ Request เสร็จสมบูรณ์ โปรแกรมจึงตรวจสอบข้อมูลที่ต้องการ และเตรียมสภาวะแวดล้อมต่างๆ จากนั้นจึงสร้างผลลัพธ์ซึ่งอาจเกี่ยวข้องกับการประมวลผลภาษา PHP Lite Script ซึ่งจะกล่าวถึงในบทถัดไป การทำงานดังกล่าว อาจสรุปได้ว่าเว็บเซิร์ฟเวอร์มีการทำงานเป็นสถานะต่างๆ ดังแสดงได้รับภาพที่ 25



ภาพที่ 25 สถานะการทำงานของระบบเว็บเซิร์ฟเวอร์

จากลักษณะการทำงานของเว็บเซิร์ฟเวอร์ดังกล่าว สามารถพัฒนาโปรแกรมย่อยที่เกี่ยวข้องซึ่งประกอบด้วย โปรแกรมย่อยสำหรับการเตรียมค่าเว็บเซิร์ฟเวอร์ โปรแกรมย่อยเพื่อรับค่าการร้องของเข้าสู่หน่วยความจำชั่วคราวของเว็บเซิร์ฟเวอร์ และ โปรแกรมย่อยเพื่อตรวจสอบการ

ทำงานหลักของเว็บเซิร์ฟเวอร์ การทำงานของโปรแกรมย่อยในแต่ละส่วนสามารถแสดงได้ดังภาพที่ 26



ภาพที่ 26 โปรแกรมย่อยสำหรับการทำงานของเว็บเซิร์ฟเวอร์

โปรแกรมย่อย HTTP Initialize จะตั้งค่าเริ่มต้นต่างๆ และกำหนดให้เว็บเซิร์ฟเวอร์อยู่ในสถานะพร้อมทำงาน (Start) จากนั้นเมื่อมีการรับข้อมูลด้วยโปรแกรมย่อย HTTP Input โปรแกรมจะบันทึกข้อมูลการร้องขอเข้าสู่หน่วยความจำชั่วคราว พร้อมทั้งเปลี่ยนสถานะของเว็บเซิร์ฟเวอร์ให้อยู่ในสถานะเตรียมตรวจสอบการตอบรับ (HTTPIN) และ โปรแกรมย่อย Check HTTP Status จะถูกเรียกใช้งานโดยโปรแกรมหลักจะตรวจสอบว่า ข้อมูลการร้องขอที่ได้รับ จบ

สมบูรณ์หรือไม่ หากสมบูรณ์จะตรวจสอบ URL และสร้างผลลัพธ์ตามประเภทของ URL และส่งผลลัพธ์ออกสู่ระบบเครือข่ายผ่านโปรแกรมย่อย TCP Send

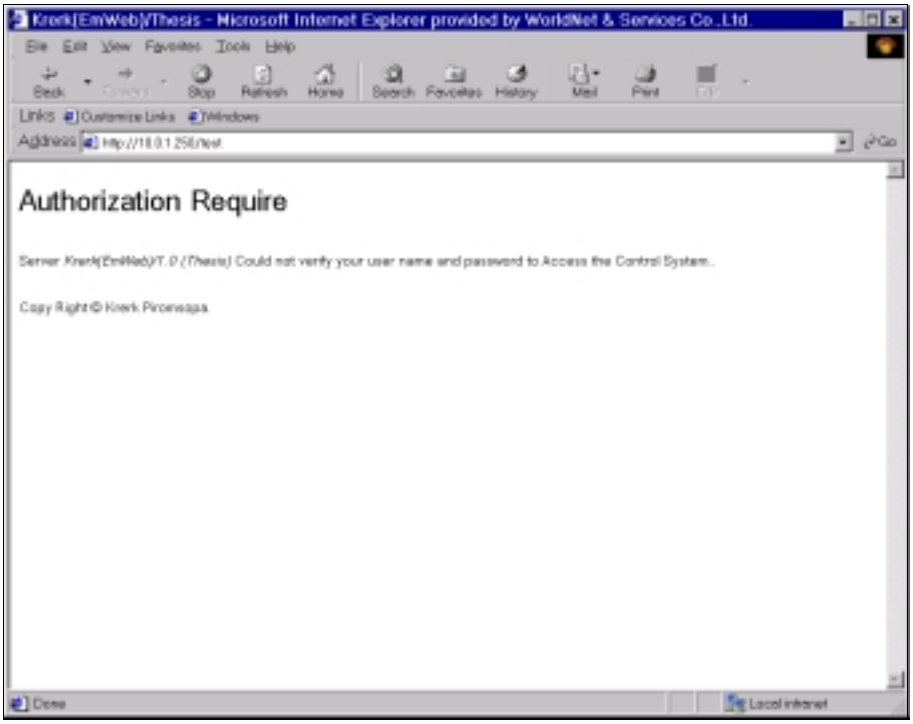
5.2. การร้องขอข้อมูลจากผู้ใช้บริการ

การร้องขอข้อมูลจากผู้ใช้บริการนั้น ในที่นี้ระบบสามารถรองรับได้เพียง Get Method เท่านั้น ทั้งนี้เนื่องจากการทำงานใน Get Method จะผนวกเอาค่าต่างๆ มาเป็นส่วนหนึ่งของ URL ทำให้ง่ายต่อการตรวจรับ โดยข้อมูลที่ถูกส่งมาพร้อมกับการร้องขอคือ Directory ระบุถึงข้อมูลที่ต้องการ หมายเลขตำแหน่ง IP รวมถึงชื่อและรหัสผ่านกรณีที่มีการใช้งานรวมกับระบบรักษาความปลอดภัย การทำงานของโปรแกรมในส่วนนี้จะทำหน้าที่รวบรวมข้อมูลทั้งหมดที่ได้รับจาก TCP เก็บไว้ในหน่วยความจำชั่วคราวและตรวจสอบว่าข้อมูลการร้องขอดังกล่าวจบสิ้นหรือไม่ ซึ่งข้อมูลในการร้องขอจะจบด้วย CR-LF CR-LF หากยังไม่จบโปรแกรมจะคอยรับข้อมูลต่อไปจนกว่าจะจบ

เมื่อได้รับข้อมูลการร้องขอทั้งหมดแล้ว โปรแกรมจะตรวจสอบ Directory หรือ URL ที่ได้เปรียบเทียบกับระบบไฟล์เสมือนภายใน (Virtual File System) หากตรงกับชื่อไฟล์ที่ภายในระบบ โปรแกรมจะทำการตอบรับข้อมูลที่ตรงกับชื่อไฟล์นั้นๆ หากไม่ตรงกัน ระบบจะถือว่าข้อมูลที่ร้องขอนั้นเป็น ภาษาสคริปต์และจะทำการเรียกตัวแปลภาษาเพื่อตอบผลลัพธ์ต่อไป

5.3. การตอบรับข้อมูล

การตอบรับข้อมูลของระบบนั้น ประกอบด้วยองค์ประกอบ 2 ส่วนคือ ส่วนหัวของข้อมูล แสดงสถานะและประเภทของข้อมูลที่ตอบรับ ซึ่งในที่นี้โปรแกรมจะตอบรับข้อมูลในส่วนหัวด้วยกันทั้งสิ้น 3 ประเภทคือ เมื่อข้อมูลที่ได้รับถูกต้องจะตอบรับว่าข้อมูลถูกต้องสถานะ 200 ตามมาตรฐาน HTTP และตามด้วยผลลัพธ์ ที่ได้จากระบบไฟล์เสมือนหรือตัวแปลภาษาสคริปต์ (ทั้งนี้ขึ้นกับ ชื่อไฟล์ที่ทำการร้องขอ) กรณีที่ข้อมูลการร้องขอเป็นแบบสคริปต์ ระบบจะตรวจสอบสิทธิการใช้งานและหากไม่พบชื่อผู้ใช้โปรแกรมจะตอบรับด้วยสถานะ 401 ตามมาตรฐาน HTTP เพื่อให้โปรแกรมค้นผ่านเว็บถามพร้อมทั้งระบุชื่อและรหัสผ่านสำหรับผู้ใช้ ส่วนกรณีสุดท้ายคือเมื่อระบบไม่เข้าใจการร้องขอข้อมูลส่วนหัวจะตอบรับด้วยสถานะ 400 ซึ่งหมายถึงระบบไม่รองรับรูปแบบการร้องขอข้อมูลในรูปแบบดังกล่าว (ดังแสดงในภาพที่ 27.)



ภาพที่ 27 ผลลัพธ์ที่ได้เมื่อผู้ใช้ระบุชื่อและรหัสผ่านผิดพลาด

5.4. ระบบรักษาความปลอดภัย

กรณีของระบบรักษาความปลอดภัยนั้นในที่นี้หมายถึงระบบการพิสูจน์ตัวจริงแบบเบื้องต้นบนเว็บ (Basic Web Authentication) ซึ่งเมื่อระบบได้รับการร้องขอข้อมูล โปรแกรมจะตรวจสอบชื่อและรหัสผู้ใช้ดังกล่าวข้างต้น โดยหากไม่พบระบบจะตอบรับด้วยสถานะ 401 ซึ่งจะส่งผลให้โปรแกรมค้นผ่านเว็บสอบถามชื่อและรหัสผ่านจากผู้ใช้ (ดังภาพที่ 28.)

ภาพที่ 28 การสอบถามชื่อผู้ใช้และรหัสผ่านของโปรแกรมค้นผ่านเว็บ

เมื่อผู้ใช้ใส่ชื่อและรหัสผ่านแล้ว โปรแกรมค้นผ่านเว็บจะทำการเข้ารหัสชื่อผู้ใช้และรหัสผ่านตามมาตรฐาน Base 64 จากนั้นจึงส่งเข้ามาในส่วนหัวการร้องขอ ซึ่งหลังจากได้รับแล้ว ระบบจะทำการถอดรหัสดังกล่าว แล้วเปรียบเทียบกับชื่อและรหัสผ่านที่มีสิทธิในการดูข้อมูลหน้านั้นๆ ซึ่งหากถูกต้องก็จะประมวลผลต่อไปตามขั้นตอน หากไม่ถูกต้องระบบจะตอบรับด้วยสถานะ 401 เรื่อยไปจนกว่าชื่อและรหัสผ่านจะถูกต้อง

บทที่ 6
การประมวลผลภาษาสคริปต์ที่ใช้ในการควบคุม

PHP Lite Script เป็นภาษาฝังตัว (Embedded Script) ที่ถูกพัฒนาขึ้น เพื่อใช้ในการควบคุมการทำงานของ เว็บเซิร์ฟเวอร์แบบฝังตัว โดยบางส่วนของโครงสร้างคำสั่งถูกดัดแปลงมาจาก PHP (Personal Home Page) อันเป็นภาษาสคริปต์ที่นิยมใช้งานแพร่หลาย ในการสร้างโปรแกรมประยุกต์ทางด้านเว็บ

6.1. ลักษณะทั่วไปของ PHP Lite Script

PHP Lite Script ประกอบด้วย ตัวแปร (Variables) ชุดคำสั่ง (Statements) และประโยคควบคุม (Control Statements) ทั้งนี้จุดมุ่งหมายหลักในการพัฒนา PHP Lite Script คือ เพื่อให้ผู้ใช้สามารถควบคุมการทำงานของ ระบบเว็บเซิร์ฟเวอร์แบบฝังตัวได้ในขณะทำงาน (Runtime Control) ซึ่งวิธีการใช้งานนั้น สคริปต์จะถูกฝังเป็นส่วนหนึ่งของ HTML Code เพื่อให้สามารถสร้างผลลัพธ์ได้อย่างมีประสิทธิภาพ ดังนั้นจึงต้องมีการ Escape หรือใส่รหัสหลีกเพื่อแยกระหว่าง Code ส่วนที่เป็น HTML Code และ PHP Code โดยทำการแทรกสคริปต์ลงระหว่างกรอบ Escape เพื่อช่วยในการสร้างผลลัพธ์แบบพลวัต (Dynamic)

6.2. โครงสร้างของภาษา PHP Lite Script

การ Escape ระหว่าง HTML Code และ PHP Code ตามมาตรฐานของ PHP ในปัจจุบันนั้น สามารถกระทำได้ทั้งสิ้น 4 วิธี คือการใช้ เครื่องหมาย “<? ?>”, “<?php ?>”, “<% %>” และ การกำหนดด้วย tag SCRIPT ของ HTML แต่บน PHP Lite Script ในที่นี้ เลือกใช้เพียงวิธีเดียวคือ การใช้เครื่องหมาย “<%” และ “%>” ทั้งนี้เนื่องจากเป็นวิธีการที่ใช้กันบนหลายระบบ เช่น ASP (Active Server Page ของ Microsoft) และ PHP (ดังตัวอย่างในภาพที่ 29)

```
<HTML><HEAD><TITLE>
<% echo ("TEST OF PHP LITE SCRIPT"); %>
</TITLE></HEAD>
<BODY>
<% echo ("PHP Lite Script Body"); %>
</BODY></HTML>
```

ภาพที่ 29 ตัวอย่างการแทรก Code ของ PHP Lite Script ลงใน HTML Code

6.2.1. ประเภทของค่าตัวแปรและค่าคงที่ (Variable and Constant)

ค่าตัวแปร (Variable) ภายใน PHP Lite Script นั้น ประกอบด้วยประเภทข้อมูล (Type) 2 ประเภท คือ จำนวนเต็ม (Integer) และสายอักษร (String)การอ้างอิงชื่อของตัวแปลทำได้โดยการใช้ชื่อซึ่งเป็นตัวอักษรความยาวไม่เกิน 8 ตัวอักษร และขึ้นต้นด้วย $ เท่านั้น ส่วนค่าคงที่นั้น ในกรณีของจำนวนเต็ม สามารถอ้างอิงได้ดังนี้

1. จำนวนเต็ม แทนด้วยตัวเลขฐานสิบ เช่น 100

2. อ้างอิงด้วยเลขฐาน 16 เช่น 0x4000,0x2000 เป็นต้น

ทั้งนี้ค่าต่างๆ จะอ้างอิงได้ไม่เกิน 16 bit

ส่วนกรณีของค่าคงที่ประเภทสายอักษรนั้น จะอ้างอิงด้วยการใช้เครื่องหมาย " (Double Quote) เช่น "Name" เป็นต้น

6.2.2. ตัวปฏิบัติการเกี่ยวกับตัวแปร (Operator)

ตัวปฏิบัติการนั้น จะทำหน้าที่คำนวณและประมวลค่าเกี่ยวกับตัวแปรต่างๆ เช่นการบวกลบ หรือต่อสายอักษรซึ่งสามารถจำแนกเป็นกลุ่มตามตัวแปรได้ดังนี้

6.2.2.1. ตัวปฏิบัติการบนตัวแปรประเภทจำนวนเต็ม (Number Operator) ประกอบด้วย “+” เพื่อทำการบวกค่าจำนวนต่าง และ “-“ เพื่อทำการลบจำนวนเต็ม

6.2.2.2. ตัวปฏิบัติการบนตัวแปรประเภทสายอักษร (String Operator) ประกอบด้วย “.” เพื่อใช้ทำการต่อสายอักษร (Concatenate) เช่น $A = "My ". "Name"; จะมีความหมายเหมือนกับ $A = "My Name"; เป็นต้น

6.2.3. นิพจน์ (Expression)

เป็นการกระทำเพื่อ ประมวลผลลัพธ์ที่เกิดการการกระทำของตัวปฏิบัติการ บนค่าคงที่และตัวแปลต่างๆ เช่น “1 + 2 + 0x2000” โดยผลลัพธ์ที่ได้จากนิพจน์ จะให้ค่าเป็นคำตอบของผลลัพธ์ ทั้งนี้ขึ้นอยู่กับว่านิพจน์ดังกล่าวอาศัยตัวปฏิบัติการใด ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 1

1+2-3+0x2000

ผลลัพธ์ที่ได้จะเป็น เลขจำนวนเต็มมีค่าเป็น 0x2000 ใน เลขฐาน 16 หรือ 8192 ในเลขฐาน 10 นั้นเอง

ตัวอย่างที่ 2

“User [“ . $Name . “] is Authenticated”

หากค่าที่กำหนดในตัวแปร $Name เป็น “Krerk” ผลลัพธ์ที่ได้จะเป็น “User [Krerk] is Authenticated” เป็นต้น

ไวยากรณ์ของนิพจน์

Expression -> Term

Term -> Term Operator Expression

Term -> Variable

Term -> Constant

Term -> Null

6.2.4. ประโยค (Statements)

ประโยคหรือ Statements คือสิ่งที่กำหนดการทำงานของ PHP Lite Script เช่น การประมวลผล หรือ การกำหนดค่า เป็นต้น Statement สามารถจำแนกเป็นกลุ่มตามลักษณะการทำงานได้ดังนี้

1. ประโยคกำหนดค่า (Assignment Statement)
2. ประโยคควบคุม (Control Statement)
3. ประโยคประยุกต์ของระบบ (System Statement)

ไวยากรณ์ของ Statements

Statements -> Statements

Statements -> Statement

Statement -> Assignment Statement

Statement -> Control Statement

Statement -> System Statement

6.2.5. ประโยคกำหนดค่า (Assignment Statement)

ใช้เพื่อการกำหนดค่าให้กับตัวแปรต่างๆ โดยการใช้เครื่องหมาย = เช่น ต้องการให้ตัวแปร $a มีค่าเป็น 5 จะกระทำโดย $a=5; เป็นต้น โครงสร้างของ Assignment จะประกอบด้วย ตัวแปรตามด้วยเครื่องหมาย “=” และ นิพจน์ตามลำดับ

ไวยากรณ์ของประโยคกำหนดค่า

Assignment -> Variable = Expression;

6.2.6. ประโยคควบคุม (Control Statement)

ประโยคควบคุมจะทำหน้าที่ในการกำหนดทิศทางของการทำงานของระบบ เช่น การทำงานวนซ้ำเป็นต้น ในต้นแบบของงานวิจัยนี้ ได้กำหนดประโยคควบคุมขึ้นมา 1 ประเภทคือ While loop เพื่อใช้ในการทำงานที่ต้องวนซ้ำเป็นรอบ โดย While Loop จะทำงานซ้ำไปเรื่อยตราบเท่าที่เงื่อนไขของ Expression ยังเป็นจริง (True) หรือ ไม่เท่ากับศูนย์

ไวยากรณ์ของประโยคควบคุม

Control Statement -> While Loop

While Loop -> while (Expression) {Statements}

6.2.7. ประโยคประยุกต์ของระบบ (System Statement)

ประโยคประยุกต์ของระบบ เป็นประโยคที่ระบบกำหนดให้เพื่อให้ผู้ใช้สามารถทำการควบคุมการทำงานที่เกี่ยวข้องกับระบบได้ เช่น echo เพื่อแสดงค่าของ Expression เป็นต้น ในงานวิจัยนี้มีการพัฒนาประโยคประยุกต์ขึ้นมาทั้งสิ้น 4 ประโยคดังนี้

1. Echo Statement
2. Outport Statement
3. Inport Statement
4. String Convert Statement

ไวยากรณ์ของประโยคประยุกต์ของระบบ

System Statement -> Echo Statement

System Statement -> Outport Statement

System Statement -> Inport Statement

System Statement -> String Convert Statement

6.2.7.1. Echo Statement

เป็น Statement เพื่อใช้สำหรับแสดงค่าของ Expression โดยโครงสร้างของประโยคสามารถอธิบายได้ดังนี้

โครงสร้างของประโยค echo

echo (String Expression);

ไวยากรณ์ของประโยค echo

Echo Statement -> echo (Expression);

ประโยคดังกล่าวจะทำการประมวลนิพจน์จากนั้น จะแสดงค่าผลลัพธ์ที่ได้ของมายังหน้าเว็บ ทั้งนี้ ค่าของนิพจน์ที่ได้ควรมีผลลัพธ์เป็น ข้อมูลประเภทสายอักษร เพื่อให้สามารถแสดงผลได้ถูกต้อง

6.2.7.2. Outport Statement

เป็นประโยคเพื่อใช้สำหรับแสดงค่าที่ต้องการออกไปยัง Memory Mapped I/O ที่ระบุ โดยผู้ใช้สามารถระบุค่าของพอร์ตและข้อมูลที่ต้องการผ่านทางนิพจน์ จุดประสงค์ของประโยคดังกล่าวเพื่อใช้ในการกำหนดค่าให้กับอุปกรณ์ (Device) ที่ต้องการ

โครงสร้างของประโยค Outport

outp (Port , Data);

ไวยากรณ์ของประโยค Outport

Outport Statement -> outp (Expression , Expression);

ประโยค outp จะนำผลลัพธ์ที่ได้จากนิพจน์ที่ 1 มากำหนดค่าพอร์ตหรือตำแหน่งของหน่วยความจำ จากนั้นจะนำผลลัพธ์ที่จากการประมวลนิพจน์ที่ 2 แสดงเป็นผลลัพธ์ไปเก็บยังหน่วยความจำในตำแหน่งของพอร์ตที่กำหนดข้างต้น

6.2.7.3. Inport Statement

ประโยค inp จะทำหน้าที่อ่านค่าข้อมูลในหน่วยความจำ หรือ Memory Mapped I/O ที่ระบุโดยนิพจน์มาเก็บยังตัวแปรที่ต้องการ วัตถุประสงค์ของประโยคคือการอ่านค่าและสถานะจากอุปกรณ์ที่ต่อพ่วงอยู่กับระบบ

โครงสร้างของประโยค Inport

inp (Port , Variable);

ไวยากรณ์ของประโยค Inport

Inport Statement -> inp (Expression , Variable);

ผลลัพธ์ที่ได้จากนิพจน์ตัวแรกจะใช้กำหนดพอร์ตหรือตำแหน่งของหน่วยความจะที่ต้องการอ่านข้อมูล โดยประโยคดังกล่าวจะทำการอ่านค่าจากพอร์ต มาเก็บยังตัวแปรที่ระบุ

6.2.7.4. String Convert Statement

เป็นประโยคประยุกต์เพื่อทำหน้าที่แปลข้อมูลประเภทจำนวนเต็มที่เก็บอยู่ในตัวแปร ให้เป็นข้อมูลประเภทสายอักษร

โครงสร้างของประโยค String Convert

str (Variable);

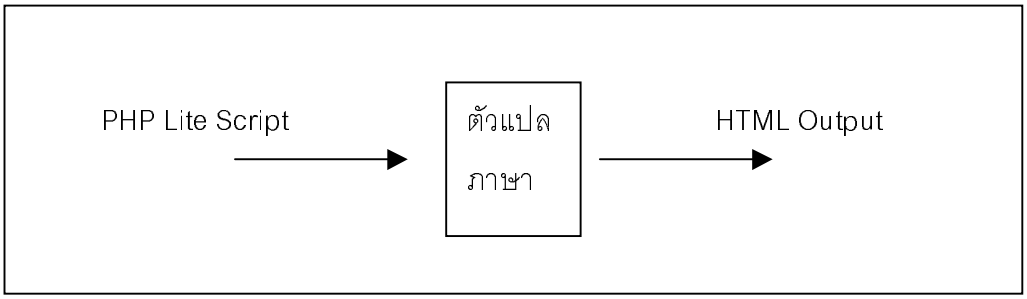
ไวยากรณ์ของประโยค String Convert

String Convert Statement -> str (Variable);

ผลลัพธ์ที่ได้จากประโยคนี้คือ การเปลี่ยนค่าคงที่ประเภทจำนวนเต็มในตัวแปรที่ระบุ มาเป็นค่าคงที่ประเภทสายอักษร เช่น $A มีค่าเป็น 0x0001 เมื่อผ่านการทำงานของประโยค str($A); จะได้ผลลัพธ์ค่า $A เป็น “0001” เป็นต้น

6.3. การสร้างตัวแปลภาษา PHP Lite Script

การแปลภาษานั้น คือการสร้างผลลัพธ์ที่ได้จากการประมวล PHP Lite Script รวมถึง การจัดการสภาพแวดล้อม (Environment) ในการทำงานเพื่อสนับสนุนการทำงาน ของภาษาด้วย ทั้งนี้การแปลจะใช้การ Interpret เพื่อเป็นการสร้างผลลัพธ์แบบขณะทำงาน และสามารถปรับเปลี่ยนตัว Script ได้ง่าย ลักษณะการทำงานของ PHP Lite Script นั้น เป็นการแปลภาษาแบบ Interpret ซึ่งสามารถอธิบายโครงสร้างการทำงานได้ดังภาพที่ 30



ภาพที่ 30 การทำงานของตัวแปลภาษา PHP Lite Script

การทำงานภายในของตัวแปลภาษานั้น ยังแบ่งออกได้อีก 2 ขั้นตอนย่อย คือ การวิเคราะห์คำศัพท์ (Lexical Analysis) ซึ่งจะทำหน้าที่ในการตรวจสอบและรู้จำคำต่างๆ จากนั้นผลลัพธ์ที่ได้จากการวิเคราะห์คำศัพท์ หรือ TOKEN จะถูกส่งผ่านไปทำการวิเคราะห์ความหมาย (Semantic Analysis) เพื่อทำการตีความและสร้างผลลัพธ์ต่อไป

6.3.1. การเตรียมสภาวะแวดล้อม และ การรับค่าผ่านฟอร์มใน HTML

สภาวะแวดล้อมในการทำงานของ PHP Lite Script นั้น ประกอบด้วยค่าต่างๆ ที่ส่งผ่านให้ตัวแปลภาษาเรียกใช้งาน เช่น ค่าที่ได้รับผ่านเว็บในรูปแบบต่างๆ สภาวะของระบบในขณะนั้น ผู้ใช้งาน Script สามารถอ้างถึงตัวแปรต่างๆ ที่ได้จากการ Submit ผ่านฟอร์มบน HTML ได้โดยตรงภายใน PHP Lite Script

จากตัวอย่างในภาพประกอบ 31 จะพบว่า ตัวแปร username จะอ้างอิงด้วย $username เมื่ออยู่ภายใน PHP Lite Script อย่างไรก็ตาม เบื้องต้นนั้นระบบจะสามารถรับค่าได้จาก METHOD GET เท่านั้น

```
<FORM METHOD=GET>
<INPUT TYPE="text" name="username">
<INPUT TYPE="SUBMIT">
</FORM>
<%
echo ($username);

%>
```

ภาพที่ 31 การอ้างอิงค่าจากฟอร์มบน HTML

6.3.2. การวิเคราะห์คำศัพท์ (Lexical Analysis)

ในการแปลภาษาโดยทั่วไปนั้น ขั้นตอนแรกของการแปลคือการหาค่า TOKEN ของภาษา ซึ่ง TOKEN เหล่านี้คือข้อความที่เป็น KEYWORD ภายในภาษานั้น สำหรับกรณีของ PHP Lite Script นี้ TOKEN ของภาษาประกอบด้วย 18 TOKEN ดังนี้

1. STOP หรือเครื่องหมาย “}”
2. VAR คือตัวแปรทุกตัว ขึ้นต้นด้วย “$”
3. WHILE เป็นข้อความคำว่า “while” ซึ่งเป็น Control Statement
4. IF ข้อความ “if” เป็น Control Statement (มิได้ทำการทดสอบในงานวิจัยนี้) ซึ่งจะพัฒนาต่อไปในอนาคต
5. ECHO ข้อความ “echo”
6. INP ข้อความ “inp”
7. OUTP ข้อความ “outp”
8. COMMA หรือเครื่องหมาย “,”
9. OB เครื่องหมาย “(“
10. CB เครื่องหมาย “)“
11. SCOLON เครื่องหมาย “;“ (Semicolon)
12. START หรือเครื่องหมาย “{“
13. PLUS ในที่นี้คือ Operator “+”
14. MINUS ในที่นี้คือ Operator “-”
15. DOT ในที่นี้คือ Operator “.”

16. CONST ค่าคงที่ ซึ่งอาจเป็นสายอักษร จำนวนเต็ม หรือเลขฐาน 16 ก็ได้
17. EQ เครื่องหมาย “=”
18. STRCONV ข้อความ “str”

การทำงานของการวิเคราะห์คำศัพท์นั้น เบื้องต้นจะเปรียบเทียบเครื่องหมายหรือสัญลักษณ์ที่เป็นอักษรเพียงตัวเดียวก่อน จากนั้นจึงจะเปรียบเทียบกับสัญลักษณ์อื่นๆ หากไม่ตรงกับสัญลักษณ์ใดๆ จะคืนค่าเป็น Error กรณีที่ TOKEN ที่ได้เป็น VAR จะคืนค่าเป็น VAR และ บอก VALUE เป็นชื่อตัวแปร ในทำนองเดียวกัน กรณีที่ TOKEN ที่ได้เป็น CONST ผลลัพธ์ที่ได้จะบอก VALUE เป็นค่าของCONST ที่อ่านได้

6.3.3. การวิเคราะห์ความหมาย (Semantic Analysis)

การทำงานของการวิเคราะห์ความหมาย จะตรวจสอบความถูกต้องของภาษา ประโยค และนิพจน์ พร้อมทั้งแสดงผลลัพธ์ที่ได้ออกมา โดยเราสามารถจัดกลุ่ม Semantic Analysis เป็นหมวดหมู่ตามหน้าที่การทำงานได้ตามไวยากรณ์ของภาษาดังกล่าวข้างต้น ดังนี้

1. นิพจน์ หรือ Expression จะทำหน้าที่ประมวลผล Expression
2. กลุ่มประโยคควบคุม Control Statement
3. กลุ่มประโยคประยุกต์ของระบบ
4. กลุ่มประโยคกำหนดค่า

กลุ่มนิพจน์จะประมวลค่าต่างๆ ตามตัวปฏิบัติการหรือ Operator ที่ได้รับ จากนั้นจะคืนค่าเป็นผลลัพธ์เพื่อให้กลุ่มอื่นใช้งานต่อไป

กลุ่มประโยคควบคุม จะตรวจสอบผลลัพธ์ที่ได้จากนิพจน์ให้เป็นไปตามเงื่อนไขที่กำหนด หากเป็นไปตามเงื่อนไขที่กำหนดจะทำงานในช่วง START และ STOP หากไม่เป็น ประโยคจะข้ามไปจนกว่าจะเจอ STOP

กลุ่มประโยคประยุกต์ของระบบจะทำงานตามแต่ความหมายของ TOKEN ที่ได้ เช่น echo จะทำการแสดงผลนิพจน์ เป็นต้น

กลุ่มประโยคกำหนดค่า จะนำค่าที่ได้จากนิพจน์มาเก็บยังค่าของชื่อตัวแปรที่กำหนด

บทที่ 7
การใช้งานระบบเว็บเซิร์ฟเวอร์แบบฝังตัว

การใช้งานระบบเว็บเซิร์ฟเวอร์แบบฝังตัวนั้น เริ่มต้นด้วยการการตั้งค่าต่างๆ ของระบบเครือข่ายและการควบคุมการใช้งานในระดับโปรแกรมประยุกต์ ซึ่งวัตถุประสงค์ของการตั้งค่าดังกล่าว เพื่อให้ระบบเว็บเซิร์ฟเวอร์แบบฝังตัวสามารถติดตั้งเข้าได้กับระบบเครือข่ายทั่วไปซึ่งมีการกำหนดหมายเลขระบบเครือข่ายที่แตกต่างกัน พร้อมทั้งตรวจสอบการทำงานของเว็บเซิร์ฟเวอร์แบบฝังตัวในการสื่อสารข้อมูลผ่านระบบเครือข่ายไปพร้อมกันด้วย และการตั้งค่าในระดับโปรแกรมประยุกต์คือการเตรียมระบบเว็บเซิร์ฟเวอร์แบบฝังตัว เพื่อให้ปรับเปลี่ยนและควบคุมอุปกรณ์ต่อพ่วงต่างๆ โดยการกำหนดภาษาสคริปต์และระบบรักษาความปลอดภัย

7.1. การกำหนดหมายเลข IP ให้กับระบบเว็บเซิร์ฟเวอร์แบบฝังตัว

เนื่องจากระบบเครือข่ายโดยทั่วไปนั้น แต่ระบบเครือข่ายจะมีหมายเลข IP ที่แตกต่างกัน ดังนั้นเพื่อให้ระบบเว็บเซิร์ฟเวอร์แบบฝังตัวสามารถทำงานได้บนระบบเครือข่ายทุกแห่ง ในที่นี้จึงให้ผู้ใช้ทำการกำหนดหมายเลข IP ให้กับระบบเว็บเซิร์ฟเวอร์แบบฝังตัวด้วยตัวเอง การกำหนดหมายเลข IP นั้นทำได้โดยการแก้ไขตาราง ARP ให้กับเครื่องไคลเอ็นต์ที่ต้องการเรียกใช้ เช่น กรณีต้องการให้เว็บเซิร์ฟเวอร์แบบฝังตัวมีหมายเลข IP เป็น 10.0.1.250 สามารถทำได้ดังนี้

บนระบบ Unix

```
arp –s 10.0.1.250 00:40:05:50:4f:4b
```

บนระบบ Window

```
arp –s 10.0.1.250 00-40-05-50-4f-4b
```

ผลลัพธ์ที่ได้จากการทำงานของคำสั่งดังกล่าวทำให้เครื่องไคลเอ็นต์ กำหนดตาราง ARP เพื่อชี้หมายเลขตำแหน่ง IP 10.0.1.250 ไปยัง หมายเลขตำแหน่ง MAC ของระบบเว็บเว็บเซิร์ฟเวอร์แบบฝังตัว โดยในที่นี้หมายเลขดังกล่าวคือ 00:40:05:50:4f:4b หลังจากนั้นเมื่อมีการส่ง IP Packet เข้าสู่ระบบเว็บเซิร์ฟเวอร์แบบฝังตัว ระบบจะทำการจดจำหมายเลขตำแหน่ง IP ไว้เพื่อกำหนดเป็นหมายเลขตำแหน่ง IP ของเว็บเซิร์ฟเวอร์แบบฝังตัวต่อไป ซึ่ง IP Packet ในที่นี้อาจเป็น Ping หรือ TCP Packet ก็ได้

7.2. การทดสอบการทำงานในระดับเครือข่าย

การทดสอบการทำงานของระบบเครือข่ายนั้น วิธีการที่ง่ายที่สุดคือการทดสอบด้วยโปรแกรม PING ซึ่งพบทั้งบน Unix และ Window โดยทำการ Ping หมายเลข IP ของระบบเว็บเซิร์ฟเวอร์แบบฝังตัว เช่นหากต้องการตรวจสอบว่าระบบเว็บเซิร์ฟเวอร์แบบฝังตัวยังคงทำงานติดต่อกับระบบเครือข่ายหรือไม่ ให้ทดสอบโดยการใช้คำสั่ง ping ตามด้วยหมายเลข IP ที่ต้องการทดสอบ

หากระบบยังคงเชื่อมต่ออยู่ผลลัพธ์ที่ได้ (ดูในภาพที่ 32) จะแสดงความเร็วในการตอบรับของระบบเว็บเซิร์ฟเวอร์แบบฝังตัว ซึ่งการทดสอบพบว่า ระบบเว็บเซิร์ฟเวอร์จะตอบรับข้อมูลโดยใช้เวลาเฉลี่ย 20.7ms เมื่อทำงานปกติ ในกรณีที่ระบบไม่ตอบรับนั้น ให้ทำการตรวจสอบสถานะของการเชื่อมต่อและสายสัญญาณต่างๆ ของระบบเครือข่าย

```
$ ping 10.0.1.250
PING 10.0.1.250 (10.0.1.250) from 10.0.1.10 : 56 data
bytes
64 bytes from 10.0.1.250: icmp_seq=0 ttl=64 time=20.7 ms
64 bytes from 10.0.1.250: icmp_seq=1 ttl=64 time=20.7 ms
64 bytes from 10.0.1.250: icmp_seq=2 ttl=64 time=20.7 ms
```

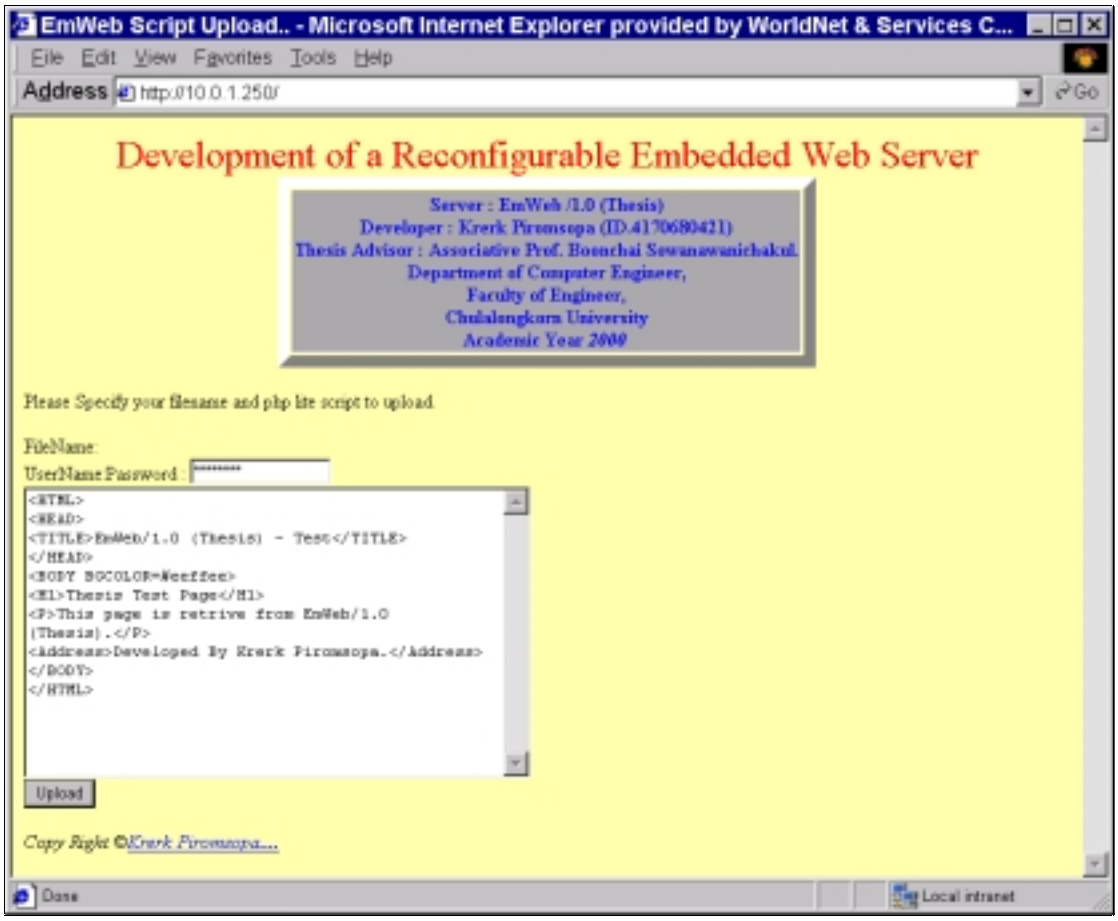
ภาพที่ 32 ผลลัพธ์จากการทดสอบสถานะทางเครือข่ายด้วยคำสั่ง ping

7.3. การตั้งค่าสคริปต์เข้าสู่ระบบเว็บเซิร์ฟเวอร์แบบฝังตัว

ในการตั้งค่าภาษาสคริปต์นั้น ผู้ใช้งานจะพัฒนาภาษาสคริปต์และหน้าเว็บโดยใช้โปรแกรมแก้ไขข้อมูลทั่วไป เช่น NOTEPAD หรือ PICO จากนั้นให้ส่งค่าที่ได้ดังกล่าวเข้าสู่ระบบโดยการกำหนด URL บนโปรแกรมค้นผ่านเว็บ ให้ชี้ไปยังหมายเลข IP ของระบบเว็บเซิร์ฟเวอร์แบบฝังตัว เช่น http://10.0.1.250/ เป็นต้น

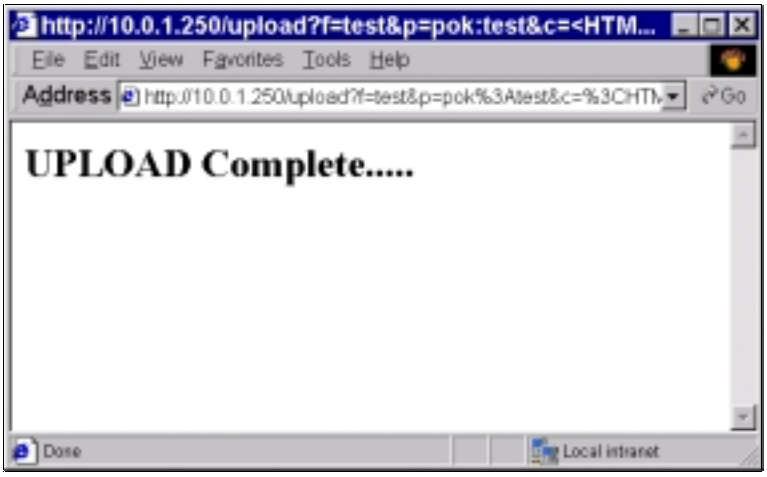
ระบบเว็บเซิร์ฟเวอร์แบบฝังตัว จะตอบรับกลับมาเป็นฟอร์ม (ดูตัวอย่างภาพที่ 32) ให้ผู้ใช้ตั้งค่า Username, Password และสคริปต์ลงในช่องว่างที่กำหนด โดยชื่อผู้ใช้และรหัสผ่านนั้น จะกำหนดในช่อง Username:Password โดยใช้เครื่องหมาย “:” (Colon) เป็นตัวขั้น เช่นกรณีต้องการกำหนดชื่อผู้ใช้เป็น pok และ รหัสผ่านเป็น test ให้กำหนด Username:Password เป็น “pok:test” และใส่ข้อมูลภาษาสคริปต์ที่ต้องการลงในช่องว่างข้างล่าง (ทั้งนี้ผู้ใช้อาจอาศัย

โปรแกรมประยุกต์อื่นๆ เพื่อช่วยในการสร้างหน้าเว็บก็ได้ เช่น โปรแกรม Microsoft Front Page หรือ โปรแกรม Dreamweaver) เมื่อเสร็จสิ้นสมบูรณ์แล้วจึงกดปุ่ม Upload เพื่อส่งข้อมูลทั้งหมดเข้าสู่ระบบ



ภาพที่ 33 การส่งสคริปต์เข้าสู่ระบบเว็บเซิร์ฟเวอร์แบบฝังตัว

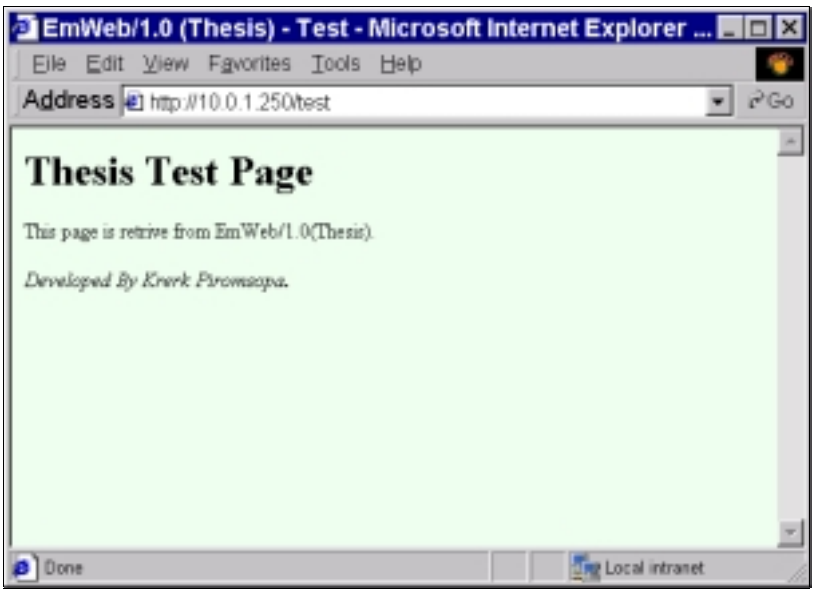
เมื่อระบบได้รับข้อมูลแล้วจะตอบรับการรับข้อมูลด้วยข้อความ Upload Complete ดังแสดงในภาพที่ 34 อย่างไรก็ตามเนื่องจากระบบดังกล่าว สามารถรองรับได้เพียงการส่งข้อมูลแบบ Method Get เท่านั้น ทำให้มีข้อจำกัดในเรื่องของความยาวของข้อมูลที่ส่งเข้าสู่ระบบ ซึ่งหากระบบไม่ทำการตอบรับใดๆ หมายความว่าสคริปต์ที่ส่งเข้าสู่ระบบนั้นอาจมีความยาวมากเกินไป การแก้ไขสามารถกระทำได้โดยการปรับสคริปต์ให้สั้นลง และส่งเข้าสู่ระบบใหม่อีกครั้งหนึ่ง



ภาพที่ 34 ผลลัพธ์จากการส่งข้อมูลเข้าสู่ระบบเว็บเซิร์ฟเวอร์แบบฝังตัว

7.4. การเรียกใช้งานสคริปต์บนระบบเว็บเซิร์ฟเวอร์แบบฝังตัว

เมื่อผู้ใช้งานได้ทำการส่งสคริปต์เข้าสู่ระบบเว็บเซิร์ฟเวอร์แบบฝังตัวแล้ว ผู้ใช้สามารถเรียกใช้งานสคริปต์ดังกล่าวได้โดยการกำหนด URL ให้ชี้ไปยังไฟล์ชื่อ test ของระบบเช่น http://10.0.1.250/test (ดูตัวอย่างผลลัพธ์ในภาพที่ 35) ซึ่งระบบจะทำการประมวลผลผลลัพธ์ที่ได้จากสคริปต์ดังกล่าวและแสดงผลออกมาทางหน้าเว็บ ทั้งนี้โปรแกรมค้นผ่านเว็บจะถาม Username และ Password เพื่อเข้าถึงข้อมูลดังกล่าว (ตามตัวอย่างในบทที่ 5) ให้ผู้ใช้ใส่ชื่อผู้ใช้และรหัสผ่านตามที่กำหนดไว้ระหว่างการส่งสคริปต์เข้าสู่ระบบ ซึ่งหากระบบเว็บว่าชื่อผู้ใช้และรหัสผ่านผิดพลาด ระบบเว็บเซิร์ฟเวอร์แบบฝังตัวจะไม่อนุญาตให้ผู้ใช้เรียกดูข้อมูลดังกล่าว



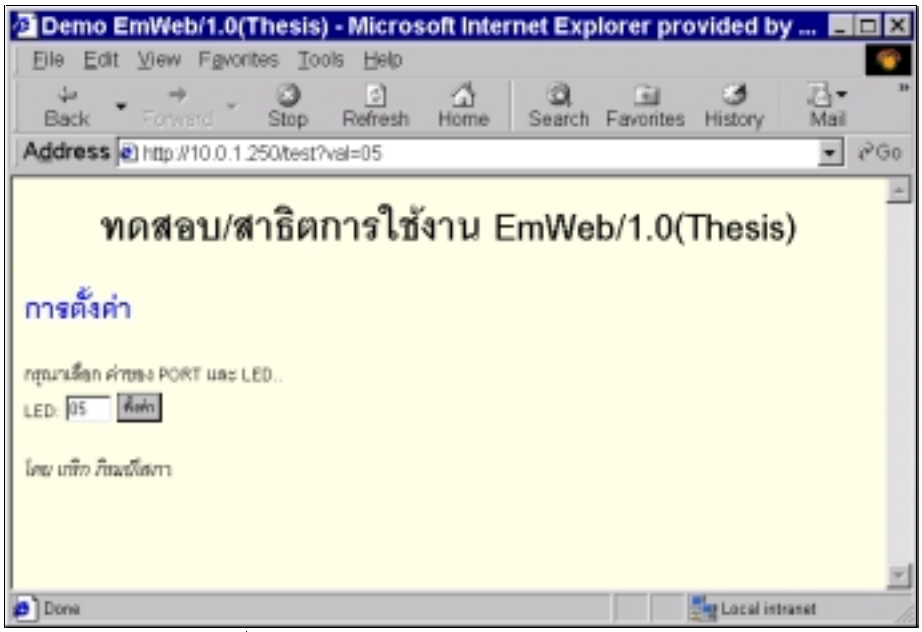
ภาพที่ 35 ตัวอย่างผลลัพธ์จากการเรียกใช้งานสคริปต์

ทั้งนี้เนื่องจากระบบเว็บเซิร์ฟเวอร์แบบฝังตัวถูกกำหนดให้มีการตั้งชื่อผู้ใช้และรหัสผ่านทุกครั้ง เมื่อมีการตั้งค่าสคริปต์เข้าสู่ระบบ ดังนั้นในการเรียกใช้งานสคริปต์เพื่อแสดงผลลัพธ์นี้หากมีการเปลี่ยนแปลงรหัสผ่าน หรือเป็นการเรียกใช้งานครั้งแรกเมื่อมีการเปิดโปรแกรมค้นผ่านเว็บนั้น ผู้ใช้จะต้องระบุชื่อผู้ใช้และรหัสผ่านเสมอ แต่หากเคยมีการระบุชื่อผู้ใช้และรหัสผ่านที่ถูกต้องไว้ก่อนแล้ว โปรแกรมค้นผ่านเว็บจะส่งค่ารหัสผ่านนั้นในการเรียกขอข้อมูลทุกครั้งจนกว่าจะปิดโปรแกรม (ดูตัวอย่างการใส่ชื่อผู้ใช้และรหัสผ่านได้ในภาพที่ 28 บทที่ 5)

7.5. การรับค่าผ่านฟอร์มเพื่อใช้ประมวลผลในภาษาสคริปต์

การรับค่าผ่านฟอร์มเพื่อใช้สำหรับการประมวลผลในภาษาสคริปต์นั้น จัดทำขึ้นเพื่อให้ผู้ใช้งานสามารถโต้ตอบกับระบบควบคุมแบบฝังตัวได้โดยอาศัยการทำงานของโปรแกรมค้นผ่านเว็บ โดยผู้ใช้สามารถอ้างถึงตัวแปรต่างๆ บนหน้าเว็บเพื่อใช้งานในสคริปต์ได้ทันที (ดังตัวอย่างในบทที่ 6) อย่างไรก็ตามข้อควรระวังในการอ้างอิงตัวแปรบนฟอร์มไปใช้งานในภาษาสคริปต์คือข้อมูลทุกตัวจะถูกจัดเก็บเป็นตัวแปรประเภทสายอักษรมิใช้ตัวแปรประเภทจำนวน ซึ่งมิควรใช้ค่าดังกล่าวเพื่อการคำนวณใดทางคณิตศาสตร์โดยไม่ทำการแปลงค่าก่อน

ตัวอย่างเช่น เราต้องการรับค่าจาก TEXT บนฟอร์ม HTML ชื่อ val เพื่อนำค่าที่ได้แสดงออกทางพอร์ตของระบบควบคุมแบบฝังตัว (ดังตัวอย่างในภาพที่ 36) ผู้ใช้สามารถพัฒนาสคริปต์ โดยการอ้างอิงตัวแปร val ด้วย $val และแสดงผลลัพธ์ออกยังพอร์ต 0x2000 ด้วยประโยค outp ซึ่งตัวอย่างสคริปต์เพื่อสามารถผลลัพธ์ดังกล่าว สามารถแสดงดังภาพที่ 36 ทั้งนี้รายละเอียดเพิ่มเติมเกี่ยวกับคำสั่งต่างๆ ในภาษา PHP Lite นั้น สามารถดูได้ในบทที่ 6



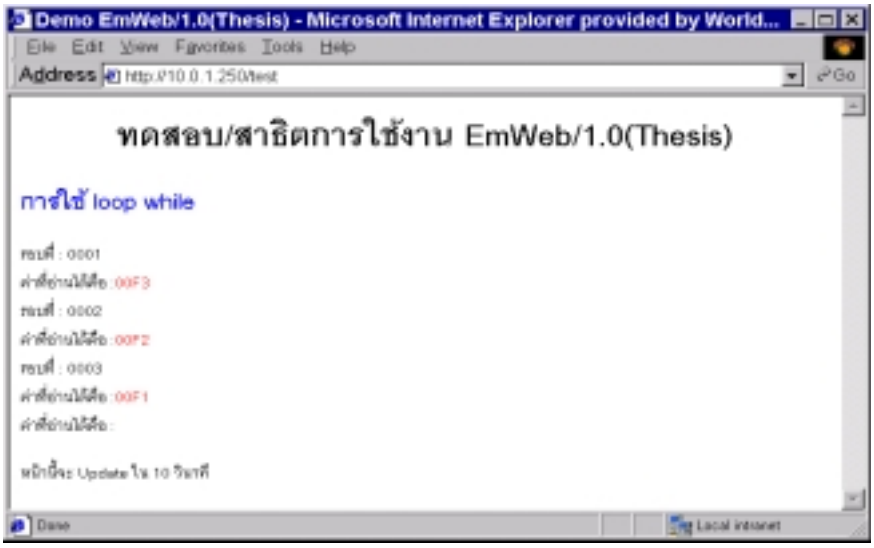
ภาพที่ 36 ตัวอย่างหน้าเว็บในการรับค่าผ่านฟอร์ม

```
<HTML>
<HEAD><TITLE>Demo EmWeb/1.0(Thesis)</TITLE></HEAD>
<BODY BGCOLOR=lightyellow>
<H1 ALIGN=CENTER>ทดสอบ/สาธิตการใช้งาน EmWeb/1.0(Thesis)</H1>
<H2><FONT COLOR=BLUE>การตั้งค่า</FONT></H2>
<FORM METHOD=GET>
กรุณาเลือก ค่าของ PORT และ LED..<BR>
LED: <INPUT TYPE=TEXT NAME=val VALUE="00" SIZE=4>
<INPUT TYPE=SUBMIT VALUE="ตั้งค่า">
</FORM>
<%outp(0x2000,$val);%>
<Address>โดย เกริก ภิรมย์โสภา.</Address>
</BODY>
</HTML>
```

ภาพที่ 37 ตัวอย่างสคริปต์ในการรับค่าผ่านฟอร์ม

7.6. การทำงานของภาษาสคริปต์ที่ซับซ้อนยิ่งขึ้น

ในการทำงานที่ซับซ้อนยิ่งขึ้น ผู้ใช้อาจอาศัยการ Refresh ของโปรแกรมค้นผ่านเว็บเพื่อดึงข้อมูลสถานะของระบบตามช่วงเวลาที่กำหนด หรืออาศัยการทำงานแบบวนรอบเพื่อตั้งค่าหรืออ่านค่าจากอุปกรณ์ต่อพ่วงหลายตัวพร้อมกันดังตัวอย่างในภาพที่ 38 แสดงการวนรอบเพื่ออ่านข้อมูล ซึ่งโปรแกรมค้นผ่านเว็บจะทำการร้องขอข้อมูลทุกๆ 10 วินาที



ภาพที่ 38 ตัวอย่างผลลัพธ์จากการ Refresh และการทำงานแบบวนรอบ

การ Refresh นั้น สามารถกระทำได้หลายแนวทางเช่นการใช้ Meta Tag หรือการใช้ JavaScript ทั้งนี้การ Refresh ที่ง่ายที่สุดคือการใช้ Meta Tag สำหรับการวนรอบนั้น ในภาษาสคริปต์ที่กำหนดขึ้นนี้มีการวนรอบแบบ while ให้ผู้พัฒนาเลือกใช้ โดยตัวอย่างการพัฒนาสคริปต์เพื่อการทำงานแบบวนรอบและการใช้ Meta Tag ในการ Refresh นั้น สามารถแสดงได้ดังภาพที่ 39

```
<HEAD>
<TITLE>Demo EmWeb/1.0(Thesis)</TITLE>
<META HTTP-EQUIV="Refresh" CONTENT="10;URL=/test">
</HEAD>
<H1 ALIGN=CENTER>ทดสอบ/สาธิตการใช้งาน EmWeb/1.0(Thesis)</H1>
<H2><FONT COLOR=BLUE>การใช้ loop while</FONT></H2>
<%
$I=3;
while ($i) {
$s=4-$i; str($s);
echo("รอบที่ : ".$s."<BR>");
outp(0x2000,$i);inp(0x2000,$val); str ($val);
$I=$i-1;
%>
ค่าที่อ่านได้คือ :<FONT COLOR=RED>
<% echo($val);%>
<BR></FONT>
<% } %>

<P>หน้านี้จะ Update ใน 10 วินาที</P>
```

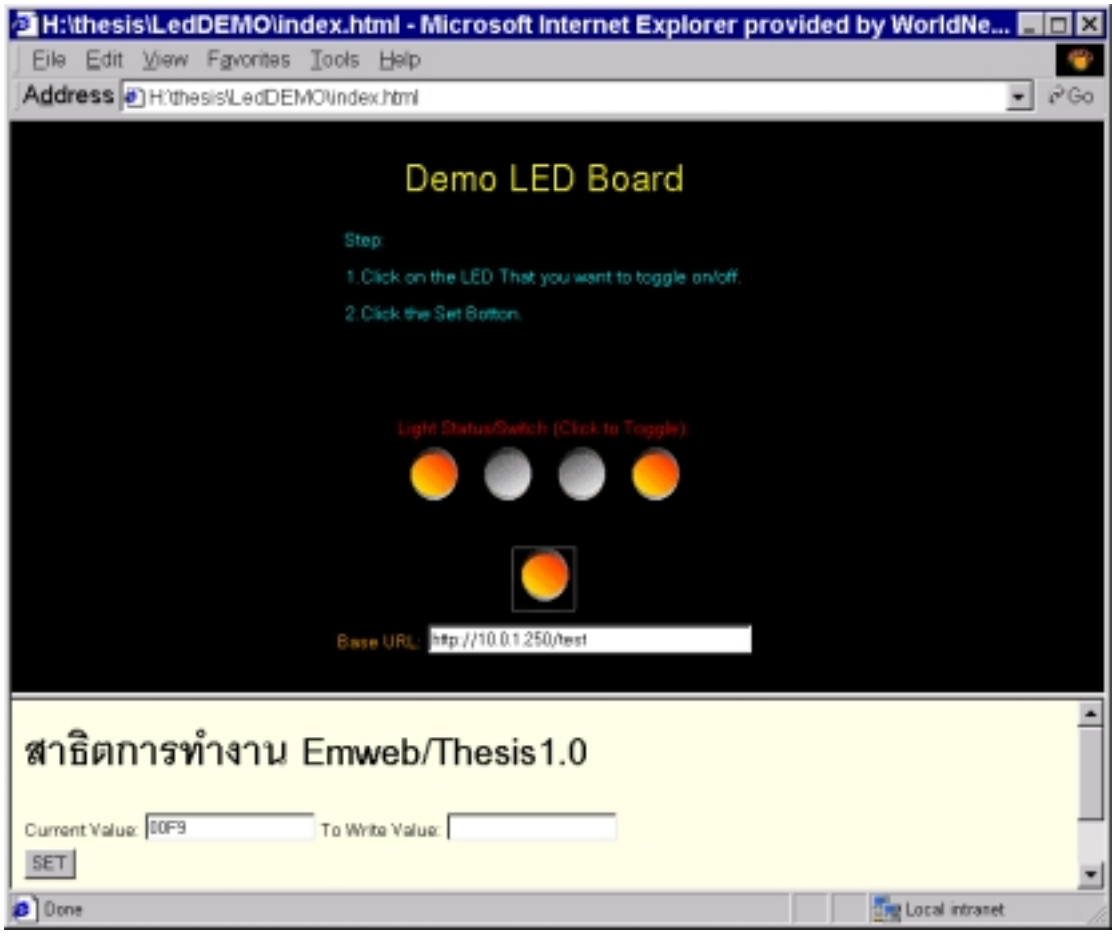
ภาพที่ 39 ตัวอย่างสคริปต์การทำงานแบบมีการ Refresh และการทำงานแบบวนรอบ

7.7. การทำงานร่วมกับภาษาอื่น ๆ เพื่อการควบคุมที่ซับซ้อนยิ่งขึ้น

ในการทำงานที่ซับซ้อนยิ่งขึ้น ภาษาสคริปต์ที่ฝังอยู่บนระบบควบคุมแบบฝังตัวอาจไม่เพียงพอต่อการควบคุมทั้งนี้เนื่องจากข้อจำกัดต่างๆ ของภาษาสคริปต์ และระบบควบคุมแบบฝังตัวที่มีหน่วยความจำจำกัด ไม่สามารถเก็บข้อมูลที่มีขนาดใหญ่ ดังนั้นจึงอาจมีการเขียนโปรแกรมหรือสคริปต์ด้วยภาษาอื่น เพื่อเป็น Component ที่สนับสนุนการทำงานของระบบเว็บ

เซิร์ฟเวอร์แบบฝังตัวโดยฝากไว้ที่เซิร์ฟเวอร์อื่นๆ ซึ่งทำหน้าที่เป็นพี่เลี้ยงช่วยในการเก็บข้อมูลขนาดใหญ่

ตัวอย่างของ Component ในที่นี้พัฒนาด้วย JavaScript เพื่อใช้ในการทำงานร่วมกับอุปกรณ์ทดสอบ เพื่อทำการควบคุมการเปิดปิดสวิตซ์ไฟ สคริปต์ประกอบด้วย 2 ส่วนคือ JavaScript ซึ่งเก็บอยู่ที่เว็บเซิร์ฟเวอร์ภายนอกหรือเครื่องคอมพิวเตอร์ของผู้ใช้งาน และ สคริปต์เพื่อฝังในเว็บเซิร์ฟเวอร์แบบฝังตัวเพื่อทำงานร่วมกับ Component นั้นๆ ทั้งนี้ผู้ใช้งานจะต้องทำการส่งสคริปต์ส่วนที่เป็น PHP Lite เข้าสู่ระบบตามขั้นตอนปกติก่อน แล้วจึงเรียก URL ไปยังหน้าเว็บที่เก็บส่ง Component นั้นๆ ซึ่งลักษณะการควบคุมด้วย Component ดังกล่าว สามารถอธิบายได้ดังภาพที่ 40 โดยผู้ใช้งานสามารถเปิด-ปิด LED ได้โดยการกดที่รูป LED ทั้ง 4 ดวง จากนั้นจึงทำกดที่รูป LED ดวงกลางเพื่อปรับเปลี่ยนสถานะของอุปกรณ์ทดสอบ



ภาพที่ 40 ตัวอย่างผลลัพธ์การประยุกต์ใช้ JavaScript บนเว็บเซิร์ฟเวอร์แบบฝังตัว

เพื่อให้สามารถควบคุมได้ดังภาพที่ 40 จะต้องมีการพัฒนา Component เป็น JavaScript เพื่อช่วยรองรับ Event ที่เกิดจากการกดที่รูป LED และเตรียมข้อมูลก่อนส่งเข้าสู่ระบบเว็บเซิร์ฟเวอร์แบบฝังตัว โดยสคริปต์ทั้งหมดแบ่งออกเป็น 2 ส่วนคือ สคริปต์ส่วนที่ฝังอยู่ในระบบเว็บเซิร์ฟเวอร์แบบฝังตัว พัฒนาด้วย PHP Lite และ หน้าเว็บส่วนที่ฝากไว้ยังเว็บเซิร์ฟเวอร์ภายนอก ซึ่งประกอบด้วยรูปภาพต่างๆ และ JavaScript Component สคริปต์ทั้ง 2 ส่วนนี้ สามารถแสดงได้ดังภาพที่ 41 และ 42 ตามลำดับ

```
<BODY BGCOLOR="lightyellow">
<H1>สาธิตการทำงาน Emweb/Thesis1.0</H1>
<form method="get">
<%
outp(0x2000,$wr);
inp(0x2000,$val); str($val);
%>
Current Value:
<INPUT TYPE=TEXT NAME=rd VALUE=<% echo ($val);%>><RB>
To Write Value:
<INPUT TYPE=TEXT NAME=wr><BR>
<INPUT TYPE=SUBMIT VALUE="SET">
</form>
<address>โดย เกริก ภิรมย์โสภา</address>
</BODY>
```

ภาพที่ 41 ตัวอย่างสคริปต์ส่วนที่ฝังในเว็บเซิร์ฟเวอร์แบบฝังตัว

จากภาพที่ 41 นั้นจะพบว่าสคริปต์จะรับค่าจากตัวแปร wr เพื่อเขียนออกสู่พอร์ต 0x2000 ดังนั้นหน้าที่ของสคริปต์ส่วนที่เป็น Component คือการสร้าง URL เพื่อร้องขอข้อมูลจากระบบเว็บเซิร์ฟเวอร์แบบฝังตัวให้มีการส่งผ่านค่าตัวแปร wr ตามต้องการเช่น กรณีต้องการตั้งค่า wr เป็น 5 สคริปต์ส่วน Component จะต้องสร้าง URL เป็น http://10.0.1.250/test?wr=05 เป็นต้น จากการทำงานในลักษณะดังกล่าว ผู้พัฒนาอาจพัฒนา Component นี้ด้วย JavaScript ดังแสดงในภาพที่ 42

```
<HTML><HEAD><TITLE>LED Module</TITLE></HEAD><BODY BGCOLOR="BLACK">
<SCRIPT LANGUAGE="JavaScript1.2">
function setimage(newval)
{ // Set Picture
document.led3.src= (newval & 8)? "1.gif":"0.gif";   document.led2.src= (newval & 4)? "1.gif":"0.gif";
document.led1.src= (newval & 2)? "1.gif":"0.gif";   document.led0.src= (newval & 1)? "1.gif":"0.gif"; }
function toggleval(val)
{ StrConst = new Array();
var newval;
StrConst[0] = "00";StrConst[1] = "01";StrConst[2] = "02";StrConst[3] = "03";
StrConst[4] = "04";StrConst[5] = "05";StrConst[6] = "06";StrConst[7] = "07";
StrConst[8] = "08";StrConst[9] = "09";StrConst[10] = "0J";StrConst[11] = "0K";
StrConst[12] = "0L";StrConst[13] = "0M";StrConst[14] = "0N";StrConst[15] = "0O";
Document.forms[0].dec.value ^= val;   Newval = document.forms[0].dec.value;
Document.forms[0].hex.value =StrConst[newval];   Setimage(newval); }
Function setvalue()
{ self.parent.frames["sub"].location = document.forms[0].burl.value + "?wr=" + document.forms
[0].hex.value; }
</SCRIPT><CENTER><FONT COLOR="YELLOW" SIZE=+3>Demo LED Board</FONT><BR>
<TABLE><TR><TD><FONT COLOR="CYAN"><P ALIGN=LEFT>Step:<BR>1.Click on the LED That
you want to toggle on/off.<BR>2.Click the Set Botton.</P></FONT></TD></TR></TABLE><BR>
<BR><FONT COLOR="RED">Light Status/Switch (Click to Toggle):</FONT><BR>
<A HREF="javascript:toggleval(8);"><IMG NAME=led3 SRC="" BORDER=0></A> 
<A HREF="javascript:toggleval(4);"><IMG NAME=led2 SRC="" BORDER=0></A> 
<A HREF="javascript:toggleval(2);"><IMG NAME=led1 SRC="" BORDER=0></A> 
<A HREF="javascript:toggleval(1);"><IMG NAME=led0 SRC="" BORDER=0></A>
<BR><FORM><A HREF="javascript:setvalue();"><IMG SRC="1.gif" ALT="SET" BORDER=0></a>
</CENTER><INPUT TYPE=HIDDEN NAME="dec" VALUE="0"><INPUT TYPE=HIDDEN NAME="hex"
VALUE="00"><CENTER><FONT COLOR="ORANGE">Base URL: </FONT><INPUT TYPE=TEXT
NAME="burl" SIZE=40></CENTER></FORM>
<SCRIPT LANGUAGE="JavaScript1.2">
setimage(document.forms[0].dec.value);
</SCRIPT></BODY></HTML>
```

ภาพที่ 42 ตัวอย่าง JavaScript เพื่อช่วยในการควบคุมที่ซับซ้อน

นอกจากนี้เพื่อความสะดวกในการแสดงผลลัพธ์ของหน้าเว็บส่วน Component ที่พัฒนาโดย JavaScript และส่วนที่ฝังในระบบเว็บเซิร์ฟเวอร์แบบฝังตัว จึงอาศัยการแบ่งเฟรม และแปลงค่าจาก Event ต่างๆ ของผู้ใช้เป็นค่าที่ต้องการเพื่อส่งผ่านไปยังระบบเว็บเซิร์ฟเวอร์แบบฝังตัว โดยการแบ่งเฟรมสามารถทำได้โดยการใช้ HTML Tag ดังแสดงเป็นตัวอย่างในภาพที่ 43

```
< frameset rows="*,150">
        <frame name="main" src="ledmod.html">
        <frame name="sub" src="none.html">
</frameset>
```

ภาพที่ 43 ตัวอย่างหน้าหลักเพื่อทำการแบ่งเฟรม

บทที่ 8
บทสรุป การประยุกต์ใช้งาน และ ข้อเสนอแนะ

การพัฒนาระบบเว็บเซิร์ฟเวอร์แบบฝังตัวที่ปรับรูปลักษณ์ใหม่ได้นั้น เป็นประโยชน์อย่างยิ่งในการพัฒนาระบบควบคุมแบบฝังตัวทั่วไป เนื่องจากระบบดังกล่าวสามารถดัดแปลงแก้ไขให้เข้ากับสภาวะหรืออุปกรณ์ต่างๆ ที่ต่อพ่วงอยู่ได้โดยง่าย และยังอำนวยประโยชน์ในการควบคุมจากระยะไกล ไม่ว่าจะอยู่ที่ใดก็ตามที่สามารถเข้าถึงอินเทอร์เน็ตได้ ผู้ใช้สามารถควบคุมและแก้ไขค่าต่างๆ ของระบบควบคุมที่เป็นเว็บเซิร์ฟเวอร์แบบฝังตัวได้ทันที อย่างไรก็ตามงานวิจัยดังกล่าวเป็นเพียงต้นแบบเท่านั้น ยังมีข้อบกพร่องหรือแนวทางที่ต้องปรับปรุงมากมาย

8.1. การประยุกต์ใช้งาน

การประยุกต์ใช้งานนั้น ผู้ใช้งานสามารถพัฒนา PHP Lite Script เพื่อใช้กำหนดการทำงานของตนเองให้เข้ากับอุปกรณ์ต่างๆ ที่ต่อพ่วงอยู่ได้โดยง่าย โดยระบบควบคุมแบบฝังตัวที่พัฒนาขึ้นนี้ มีความสามารถในการขับและรับสัญญาณแบบ TTL โดยการเชื่อมต่อผ่าน Memory Mapped I/O ดังกล่าวข้างต้น เช่น ระบบการขับเคลื่อนมอร์เตอร์ ระบบเปิดปิดวาล์วน้ำ ระบบการควบคุมเครื่องจักรอุตสาหกรรม หรือระบบไฟฟ้าในครัวเรือน เป็นต้น

8.2. แนวทางการปรับปรุง

แนวทางในการปรับปรุงระบบควบคุมแบบฝังตัวดังกล่าว ให้มีประสิทธิภาพมากขึ้นนั้น ประกอบด้วยการพัฒนาประสิทธิภาพทางด้านระบบเครือข่าย และ การพัฒนาระบบควบคุม ซึ่งสามารถสรุปเป็นหัวข้อได้ดังนี้

1. พัฒนาระบบเครือข่ายให้รองรับการทำงานแบบ Multi Session หรือมีผู้ใช้งานได้พร้อมกันมากกว่า 1 คน ณ เวลาเดียวกัน
2. จัดสร้างสคริปต์หรือ องค์ประกอบ (Component) ที่ซับซ้อนยิ่งขึ้น เพื่อช่วยให้สามารถควบคุมและเชื่อมต่อกับอุปกรณ์ต่างๆได้โดยง่าย ซึ่งอาจจัดทำในรูปแบบของวัตถุ (Object) บนมาตรฐานอื่นๆ ของโปรแกรมประยุกต์บนเว็บ เช่น JavaScript, Java หรือ VBScript เป็นต้น
3. พัฒนาระบบ Watch Dog เพื่อใช้ในการตรวจสอบสถานะของระบบหากมีความผิดปกติเกิดขึ้น ให้ทำการแก้ไขตัวเองได้ ทั้งนี้เพื่อเป็นประโยชน์ในการพัฒนาระบบควบคุมที่ยากแก่การเข้าไปตรวจสอบหรือแก้ไข

แนวทางในการปรับปรุงและพัฒนาดังกล่าวรวบรวมจากปัญหาและอุปสรรคต่างๆ ที่พบในระหว่างการวิจัยและทดสอบ โดยแนวทางการปรับปรุงมุ่งเน้นเพื่ออำนวยความสะดวกในการใช้งาน และ เพิ่มประสิทธิภาพการทำงานของระบบให้มีความทนทานมากขึ้น

8.3. บทสรุป

การพัฒนาระบบเว็บเซิร์ฟเวอร์แบบฝังตัวที่จัดรูปลักษณ์ใหม่ได้ เป็นการช่วยอำนวยความสะดวกในการพัฒนาระบบควบคุมผ่านระบบเครือข่ายเว็บที่มีความซับซ้อนมากยิ่งขึ้น ทั้งนี้อุปกรณ์ต่อพ่วงที่เป็นดิจิทัลต่างๆ สามารถพัฒนาเพื่อการเชื่อมต่อ และควบคุมได้อย่างสะดวก โดยอาศัยเพียงการพัฒนาสคริปต์และองค์ประกอบ (Component) ที่มีอยู่ ให้เหมาะสมกับสภาวะของอุปกรณ์นั้นๆ ได้ อย่างไรก็ตามอุปกรณ์และผลการวิจัยดังกล่าวยังมีจุดบกพร่องที่ควรปรับปรุงอีกหลายจุด ทั้งนี้เพื่อประโยชน์ ประสิทธิภาพและความสะดวกในการใช้งาน

8.4. ข้อเสนอแนะ

ระบบเว็บเซิร์ฟเวอร์แบบฝังตัวนั้น จะสามารถทำงานได้อย่างมีประสิทธิภาพหากมีการทำงานร่วมกับภาษาสคริปต์อื่นๆ ในฝังผู้ใช้งาน ทั้งนี้เพื่อช่วยอำนวยความสะดวกในการสร้างหน้าจอติดต่อกับผู้ใช้ที่ดูเข้าใจง่าย และเหมาะแก่การควบคุม อย่างไรก็ตามระบบควบคุมในงานวิจัยนี้ ไม่เหมาะสมที่จะใช้ควบคุมระบบงานที่ต้องการความรวดเร็วในการประมวลผล หรือ การประมวลผลแบบ Real-time ดังนั้น หากต้องใช้กับงานควบคุมที่ซับซ้อนมากยิ่งขึ้น ควรเลือกใช้ไมโครคอนโทรลเลอร์ที่มีประสิทธิภาพมากกว่านี้

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.

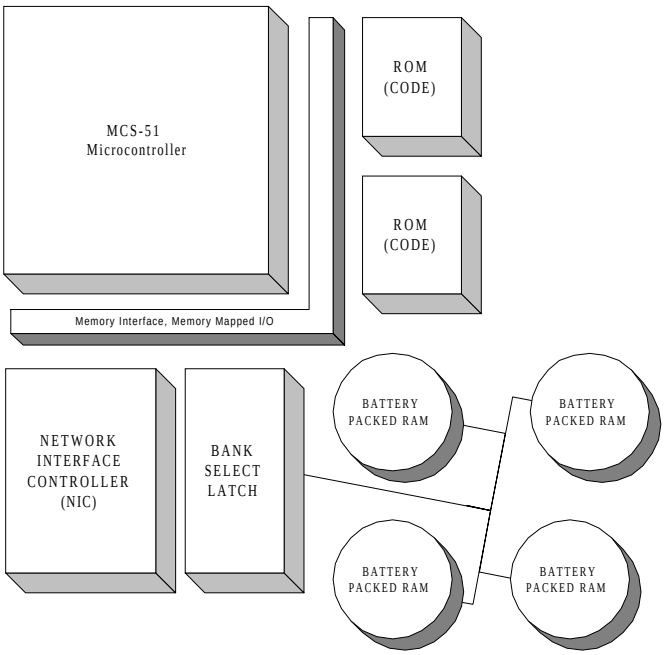


Figure 1 Hardware Block Diagram

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.

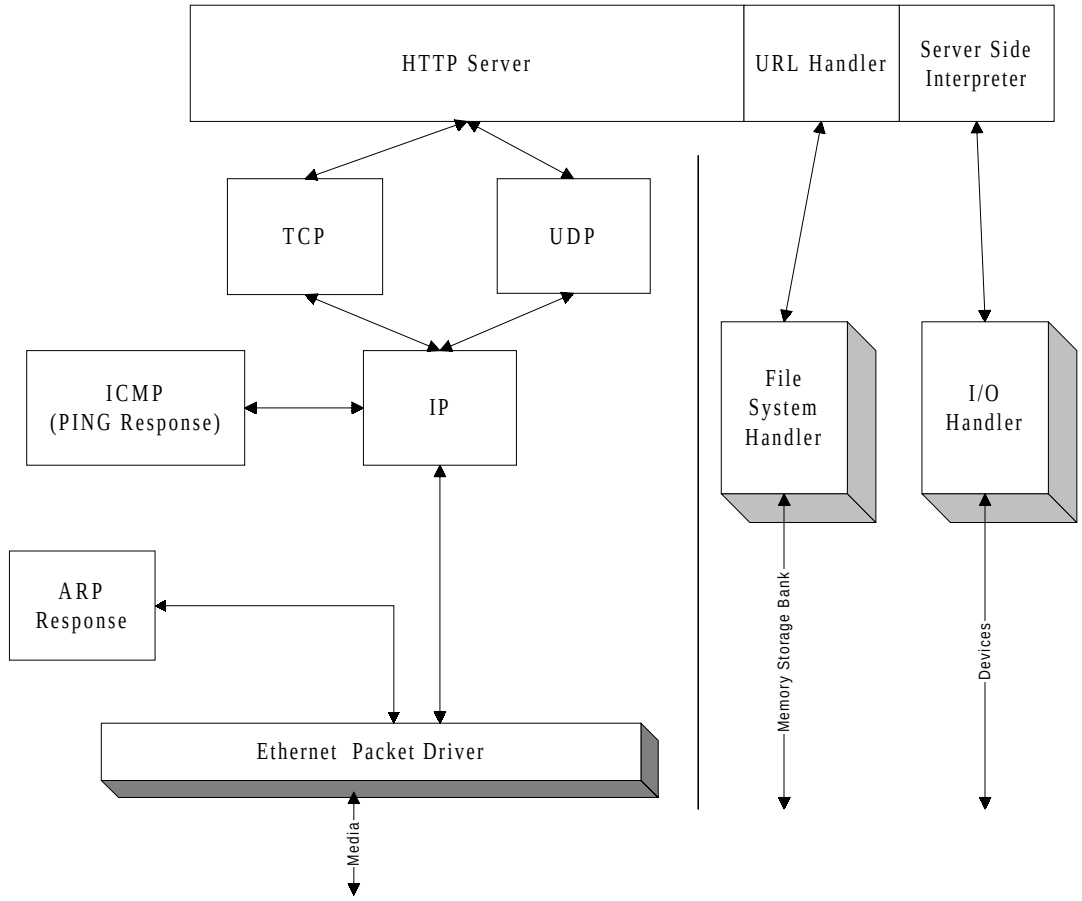


Figure 2 The software component in embedded web server.

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

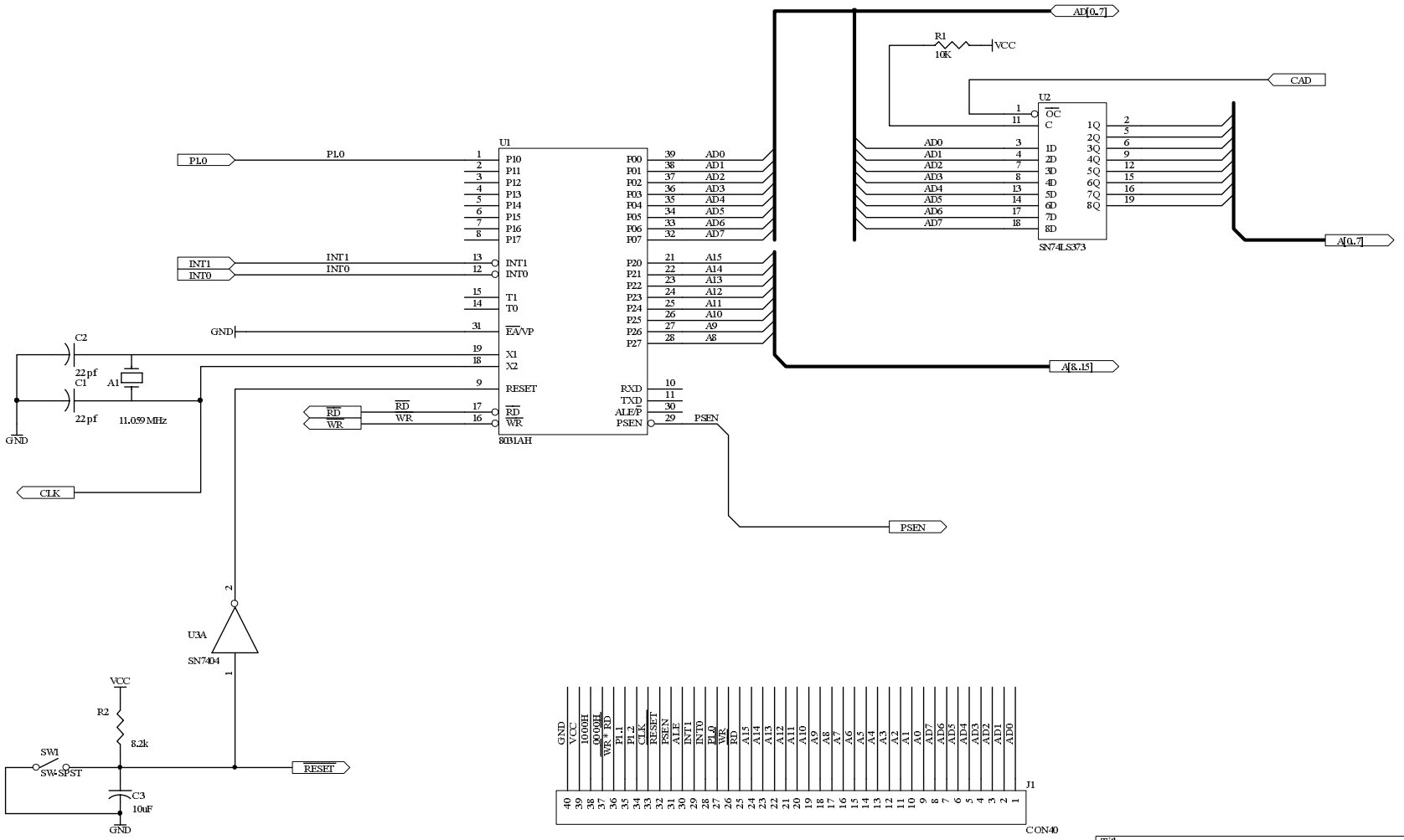
[9] H.Frystyk, J. Gettys, J. Mogul, R. Fielding, and T. Berners-Lee, 1997. "Hypertext Transfer Protocol – HTTP/1.1," RFC 2068,162 pages

[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

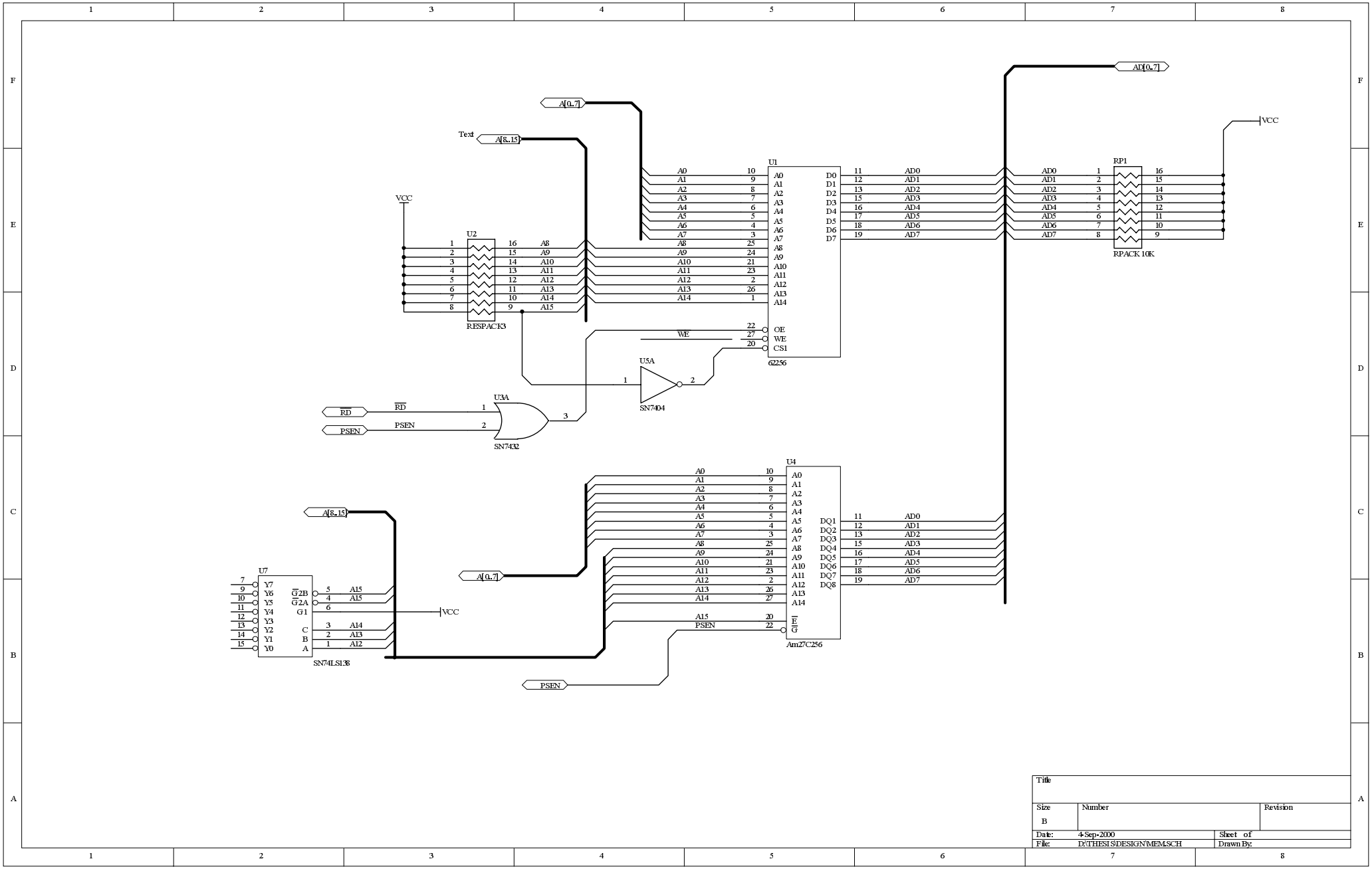
[11] Steve Wingard, Spyglass, Inc. "Embedding HTTP Functionality for Web-Based Configutaion and Management of Devices," Embedded System Conference, Class 460

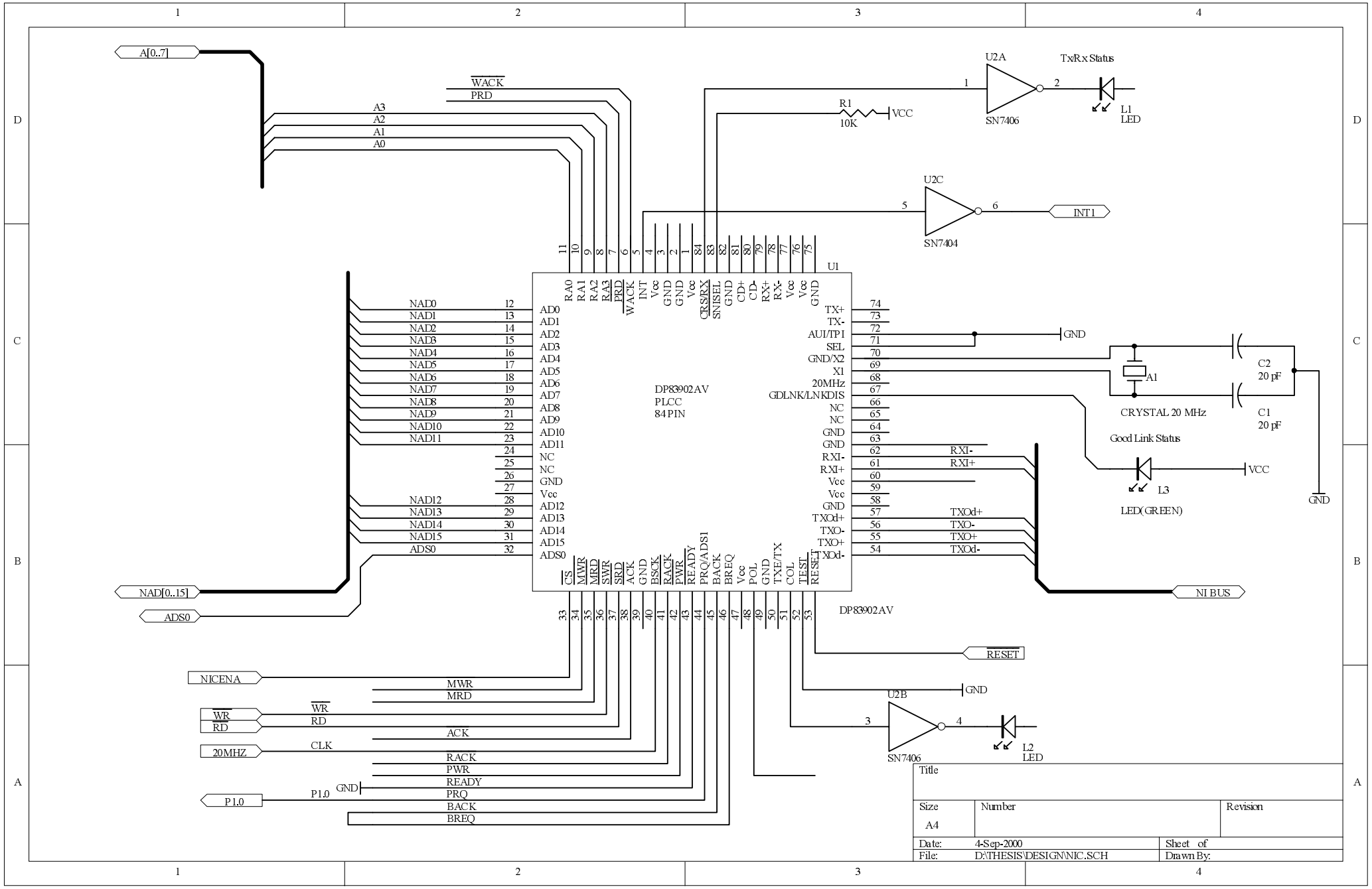
[12] Michael O'Brien, GoAhead Software, "Open Source Embedded Web Servers," Embedded System Conference, Class 407

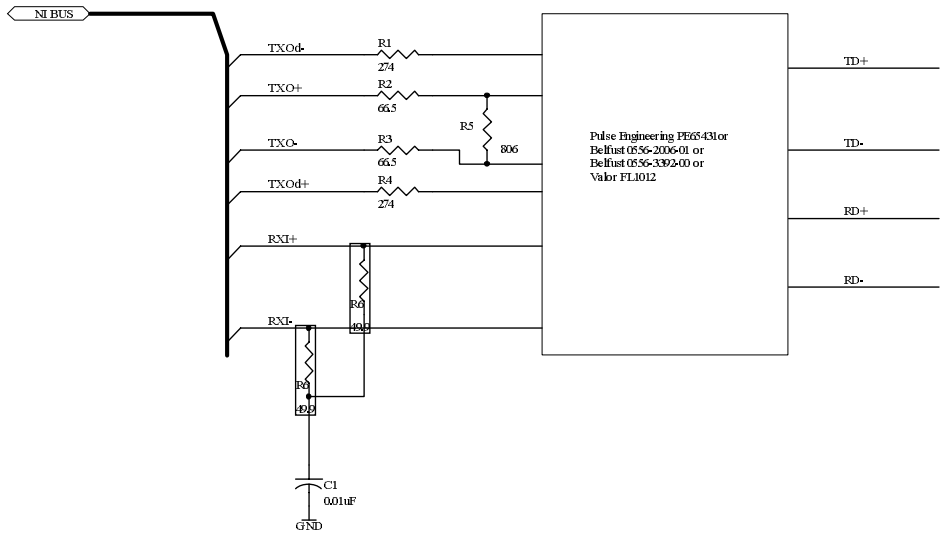
ภาคผนวก ข



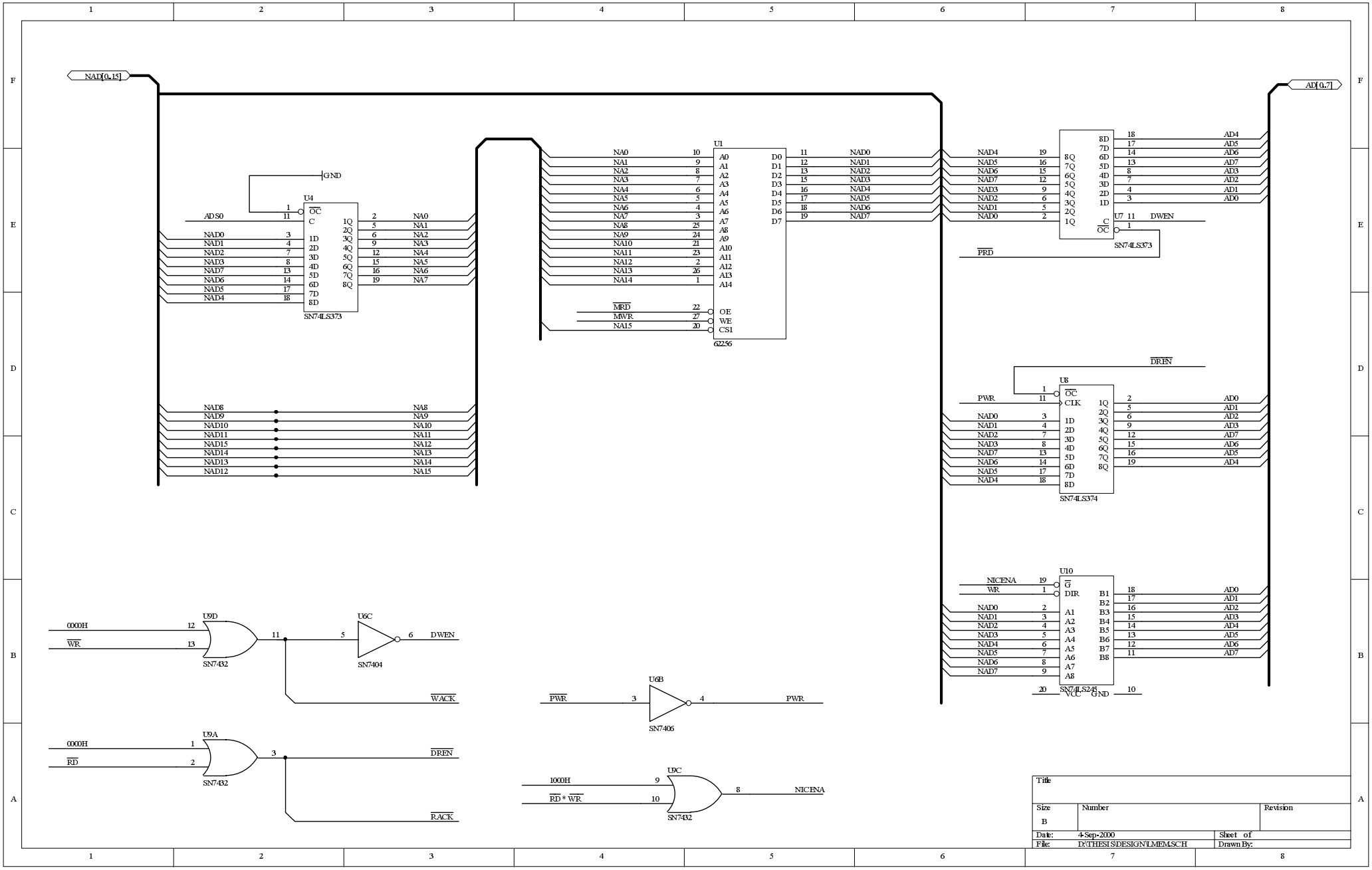
Title		
Size: B	Number	Revision
Date: 4-Sep-2000		Sheet of
File: D:\THESIS\DESIGN\CPU.SCH		Drawn By:

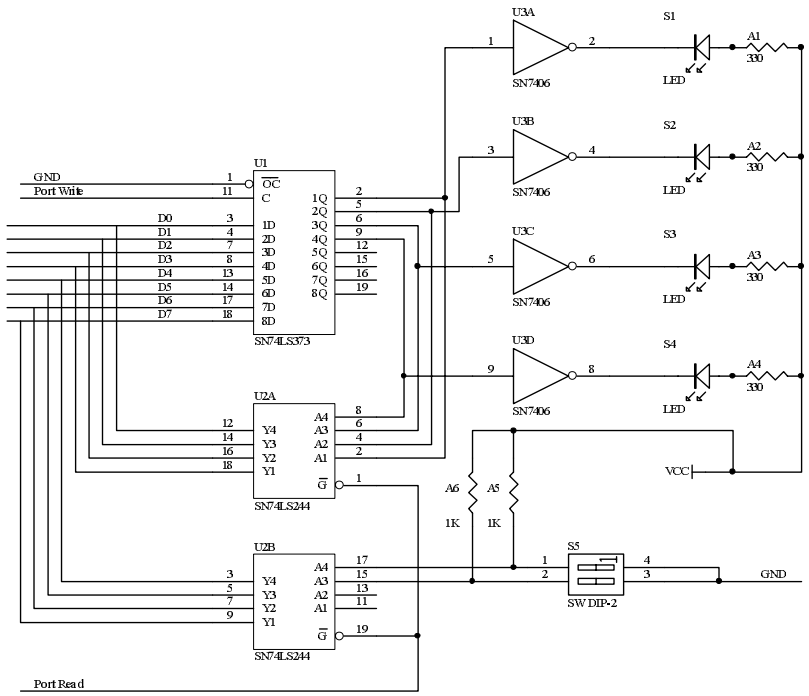

AD[0..7]
A[0..7]
Text
A[8..15]
VCC
U1
U2
RP1
RESPACK3
RPACK 10K
62256
U5A
SN7404
U3A
SN7432
RD
PSEN
WE
U4
Am27C256
U7
SN74LS138
A[8..15]
A[0..7]
VCC
PSEN
Title
Size
B
Number
Revision
Date: 4-Sep-2000
Sheet of
File: D:\THESIS\DESIGN\MEM.SCH
Drawn By:


A[0..7]
WACK
PRD
A3
A2
A1
A0
U2A
TxRx Status
R1
10K
VCC
SN7406
L1
LED
U2C
INT1
SN7404
U1
DP83902AV
PLCC
84 PIN
NAD0
NAD1
NAD2
NAD3
NAD4
NAD5
NAD6
NAD7
NAD8
NAD9
NAD10
NAD11
NAD12
NAD13
NAD14
NAD15
ADS0
NAD[0..15]
GND
A1
C2
20 pF
C1
20 pF
CRYSTAL 20 MHz
Good Link Status
L3
LED(GREEN)
RXI-
RXI+
TXOd+
TXO-
TXO+
TXOd-
NI BUS
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:
4-Sep-2000
Sheet of
File:
D:\THESIS\DESIGN\NIC.SCH
Drawn By:



Title			
Size B	Number		Revision
Date:	4-Sep-2000	Sheet of	
File:	D:\THESIS\DESIGN\NI.SCH	Drawn By:	


NAD[0..15]
AD[0..7]
GND
U4
SN74LS373
U1
62256
U7
SN74LS373
U8
SN74LS374
U10
SN74LS245
U9D
SN7432
U6C
SN7404
U6B
SN7406
U9A
SN7432
U9C
SN7432
0000H
1000H
WR
RD
PWR
DWEN
WACK
RACK
DREN
NICENA
RD * WR
Title
Size
B
Number
Revision
Date: 4-Sep-2000
Sheet of
File: D:\THESIS\DESIGN\LMEM.SCH
Drawn By:



Title		
Size: B	Number	Revision
Date: 4-Sep-2000		Sheet of
File: D:\THESIS\DESIGN\DEMO.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                                            ;           MOV     DPH,#HIGH(OBUF)
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
        ;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     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:
        ;
        00000000001111111112222222222333333333344444444444
        ;
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:
        ;
00000000001111111111222222222233333
33333344444444455555555556
        ;
01234567890123456789012345678901234
45678901234567890123456789
        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