ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า)

ปริญญา

วิศวกรรมไฟฟ้า	วิศวกรรมไฟฟ้า
สาขา	**ภาควิชา**

เรื่อง ระบบปฏิบัติการไบโอเมตริกบนตัวประมวลผลสัญญาณดิจิทัลผ่านทางเครือข่าย

Biometric Operating System on Digital Signal Processing via Network

นามผู้วิจัย นายธวัชชัย บานใหม่

ได้พิจารณาเห็นชอบโดย

ประธานกรรมการ .....

(ผู้ช่วยศาสตราจารย์วุฒิพงศ์ อารีกุล, Ph.D.)

กรรมการ .....

(อาจารย์ชูเกียรติ การะเกตุ, Ph.D.)

กรรมการ .....

(ผู้ช่วยศาสตราจารย์เขมะฑัต วิภาตะวนิช, Ph.D.)

หัวหน้าภาควิชา .....

(อาจารย์ชูเกียรติ การะเกตุ, Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

.....

(รองศาสตราจารย์วินัย อาจคงหาญ, M.A.)

คณบดีบัณฑิตวิทยาลัย

วันที่ 5 **เดือน** เมษายน **พ.ศ.** 2547

วิทยานิพนธ์

เรื่อง

ระบบปฏิบัติการไบโอเมตริกบนตัวประมวลผลสัญญาณดิจิทัลผ่านทางเครือข่าย

Biometric Operating System on Digital Signal Processing via Network

โดย

นายธวัชชัย บานใหม่

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า)
พ.ศ. 2549

ISBN 974-16-1385-7

ธวัชชัย บานใหม่ 2549: ระบบปฏิบัติการไบโอเมตริกบนตัวประมวลผลสัญญาณดิจิทัลผ่านทางเครือข่าย ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า) สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า ประธานกรรมการที่ปรึกษา: ผู้ช่วยศาสตราจารย์วุฒิพงศ์ อารีกุล, Ph.D. 135 หน้า
ISBN 974-16-1385-7

โครงงานวิทยานิพนธ์นี้เป็นการพัฒนาระบบไบโอเมตริกบนตัวประมวลผลสัญญาณดิจิทัล โดยสามารถแบ่งเป็น 2 งานหลัก ได้แก่ การออกแบบและพัฒนาระบบปฏิบัติการไบโอเมตริกบนตัวประมวลผลสัญญาณดิจิทัล และการออกแบบและสร้างฮาร์ดแวร์เชื่อมต่อเครือข่าย

ในงานแรกเป็นการออกแบบและพัฒนาระบบปฏิบัติการไบโอเมตริกเน้นการตรวจสอบลายนิ้วมือสำหรับการเข้าถึงบนตัวประมวลผลสัญญาณดิจิทัล ซึ่งระบบปฏิบัติการนี้สามารถแบ่งการเชื่อมต่อเป็นสองส่วนคือ ส่วนแรกเป็นตัวกลางเชื่อมต่อส่วนฮาร์ดแวร์ และส่วนที่สองเป็นตัวกลางเชื่อมต่อส่วนซอฟต์แวร์ขั้นตอนวิธีการตรวจสอบไบโอเมตริก โดยส่วนแรกจะใช้ซอฟต์แวร์ระบบปฏิบัติการพื้นฐานของบริษัท Texas Instruments (TI) ที่เรียกว่า DSP/BIOS (Digital Signal Processor/Basic Input Output System) มาจัดการทรัพยากรต่าง ๆ และส่วนที่สองจะใช้มาตรฐานขั้นตอนวิธี (Algorithm Standard) เพื่อให้การพัฒนาขั้นตอนวิธีแยกกันได้อย่างอิสระและนำกลับมารวมกันได้อย่างเป็นระบบ โดยระบบปฏิบัติการไบโอเมตริกนี้จะสามารถจัดรูปแบบตามมาตรฐานส่วนต่อประสานโปรแกรมประยุกต์ไบโอเมตริก (Biometric Application Programming Interface) หรือ BioAPI เพื่อสะดวกต่อการนำไปประยุกต์ใช้งานไบโอเมตริกโดยทั่วไปในอนาคต

งานที่สองคือการออกแบบและสร้างฮาร์ดแวร์เชื่อมต่อเครือข่ายต้นแบบเพื่อทำหน้าที่ส่งผ่านข้อมูลระหว่างตัวประมวลผลสัญญาณดิจิทัลกับระบบเครือข่ายพื้นที่ท้องถิ่น (Local Area Network) โดยใช้ตัวประมวลผลเครือข่ายเบอร์ IP2022 ของบริษัท Ubicom ซึ่งการพัฒนาโปรแกรมจะใช้ชุดซอฟต์แวร์โพรโทคอลเรียงทับซ้อนทีซีพี (TCP Protocol Stack) ที่มีในตัวประมวลผลเครือข่าย ทำให้สะดวกต่อการพัฒนาซอฟต์แวร์ประยุกต์สำหรับเชื่อมต่อระหว่างฮาร์ดแวร์ประมวลผลสัญญาณดิจิทัลกับระบบเครือข่ายได้อย่างมีประสิทธิภาพ

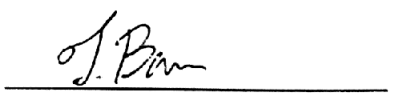
ลายมือชื่อนิสิต ลายมือชื่อประธานกรรมการ 3 / 4 / 49

Thawatchai Banmai 2006: Biometric Operating System on Digital Signal Processing via Network. Master of Engineering (Electrical Engineering), Major Field: Electrical Engineering, Department of Electrical Engineering. Thesis Advisor: Assistant Professor Vutipong Areekul, Ph.D. 135 pages.
ISBN 974-16-1385-7

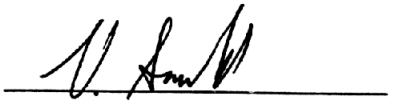
In this thesis, biometric operating system on digital signal processor (DSP) is developed. The thesis consists of 2 major parts; 1) design and development of biometric operating system on digital signal processor, and 2) design and implementation of network interface hardware.

The first part of this thesis is to design and develop biometric operating system for fingerprint access control on digital signal processor. The biometric operating system interface can be separated into two parts; hardware interface and algorithm software interface. In case of hardware interface, DSP/BIOS (Digital Signal Processor/Basic Input Output System), provided by Texas Instruments (TI), plays the important role in order to manage hardware resources. In another case, algorithm standard rules, provided by TI, are used for algorithm software interface in order to develop and integrate independent algorithms. In addition, this biometric operating system can be arranged to meet BioAPI standard (Biometric Application Programming Interface) for general biometric applications in the near future.

The second part of this thesis is to design and implement network interface hardware in order to exchange information between digital signal processor and LAN (Local Area Network). This hardware prototype uses IP2022 network processor by Ubicom Inc., which included TCP protocol stack facilities inside, provides easy and efficient DSP-LAN interface development.



Student's signature



Thesis Advisor's signature

3 / 4 / 49

กิตติกรรมประกาศ

ข้าพเจ้าขอกราบขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร. วุฒิพงศ์ อารีกุล ประธานกรรมการที่ปรึกษา ที่ได้ช่วยเหลือในการวางแผนงานวิจัยในวิทยานิพนธ์ฉบับนี้ ตลอดจนการให้คำปรึกษาแนะนำและตรวจแก้ไขข้อบกพร่องในวิทยานิพนธ์ให้สำเร็จลุล่วงไปด้วยดี

ข้าพเจ้าขอกราบขอบพระคุณ คุณแม่อุไร, คุณพ่อสมิทธิ์ บานใหม่ ที่สนับสนุนและให้กำลังใจในการทำวิทยานิพนธ์ฉบับนี้จนสำเร็จลุล่วงไปด้วยดี

ข้าพเจ้าขอขอบคุณศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ (NECTEC) ที่ได้ให้เงินทุนอุดหนุนในการทำโครงงานวิจัย และหน่วยวิจัยระบบสื่อสารแบบใช้สาย (Wireline Communication Section) ซึ่งเป็นหน่วยงานวิจัยหนึ่งของ NECTEC ที่ได้ช่วยพัฒนาบอร์ดฮาร์ดแวร์ประมวลผลสัญญาณดิจิทัลสำหรับเป็นตัวประมวลผลหลักในโครงงานวิจัย

ข้าพเจ้าขอขอบคุณเพื่อน ๆ ในห้องวิจัยด้านการประมวลผลสัญญาณและภาพ มหาวิทยาลัยเกษตรศาสตร์ (KSIP Lab) ที่ช่วยพัฒนาและทดสอบส่วนต่างในการทำวิทยานิพนธ์ฉบับนี้จนสำเร็จลุล่วงไปด้วยดี ขอขอบคุณเพื่อน ๆ ที่สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือที่ให้คำปรึกษาและให้คำแนะนำต่าง ๆ หากวิทยานิพนธ์ฉบับนี้มีข้อบกพร่องประการใด ข้าพเจ้ายินดีรับข้อเสนอแนะและขออภัยมา ณ ที่นี้ด้วย

ธวัชชัย บานใหม่

กุมภาพันธ์ 2549

สารบัญ

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

คำอธิบายสัญลักษณ์ คำย่อ และอักษรย่อ

API	=	Application Programming Interface
ARP	=	Address Resolution Protocol
ASCII	=	American Standard Code for Information Interchange
BioAPI	=	Biometric Application Programming Interface
BIOS	=	Basic Input Output System
BIR	=	Biometric Identification Record
BSP	=	Biometric Service Provider
CBEFF	=	Common Biometric Exchange File Format
CCS	=	Code Composer Studio
CRC	=	Cyclic Redundancy Check
CSL	=	Chip Support Library
CSMA/CD	=	Carrier Sense, Multiple Access with Collision Detection
DNA	=	Deoxyribo Nucleic Acid
DSP	=	Digital Signal Processor
EBCDIC	=	Extended Binary Coded Decimal Interchange Code
FTP	=	File Transfer Protocol
FVS	=	Fingerprint Verification Software
HPU	=	Host Processing Unit
HTTP	=	Hyper Text Transfer Protocol
IALG	=	Instance Algorithm
ICMP	=	Internet Control Message Protocol
IGMP	=	Internet Group Management Protocol
IP	=	Internet Protocol
ISDN	=	Integrated Services Digital Network
ISO	=	International Organization for Standardization
ISR	=	Interrupt Service Routine
JTAG	=	Joint Test Action Group

คำอธิบายสัญลักษณ์ คำย่อ และอักษรย่อ (ต่อ)

LAN	=	Local Area Network
LSB	=	Least Significant Bit
MSB	=	Most Significant Bit
OSI	=	Open System Interconnect
PCB	=	Print Circuit Board
PPP	=	Point to Point Protocol
RARP	=	Reverse Address Resolution Protocol
RISC	=	Reduce Instruction Set Computer
RTDX	=	Real Time Data Exchange
SLIP	=	Serial Line Interface Protocol
SMTP	=	Simple Mail Transfer Protocol
SNMP	=	Simple Network Management Protocol
SPI	=	Service Provider Interface
TCP	=	Transmission Control Protocol
TI	=	Texas Instruments
TTL	=	Time-to-Live
UART	=	Universal Asynchronous Receiver/Transmitter
UDP	=	User Datagram Protocol
XDAIS	=	TMS320 DSP Algorithm Interoperability Standard

ระบบปฏิบัติการไบโอเมตริกบนตัวประมวลผลสัญญาณดิจิทัลผ่านทางเครือข่าย

Biometric Operating System on Digital Signal Processing via Network

คำนำ

ในปัจจุบันงานวิจัยทางด้านการประมวลภาพดิจิทัลนั้นเป็นที่สนใจกันอย่างแพร่หลาย หนึ่งในงานวิจัยทางด้านนี้ ได้แก่ การตรวจสอบความเป็นบุคคลจากอัตลักษณ์หรือคุณลักษณะของร่างกายแต่ละบุคคลที่เรียกว่า ไบโอเมตริก (Biometric) ตัวอย่างเช่น การตรวจสอบภาพลายนิ้วมือ ลายม่านตา ใบหน้า ท่าทางการเดิน เส้นเลือด ฯลฯ โดยที่การตรวจสอบอัตลักษณ์นั้น สามารถนำไปประยุกต์ใช้ในงานต่าง ๆ ได้หลากหลายด้าน เช่น ด้านความปลอดภัย ด้านการแพทย์ ด้านอุตสาหกรรม เป็นต้น

สำหรับงานด้านระบบความปลอดภัยในปัจจุบันได้ใช้เทคโนโลยีทางด้านการประมวลภาพดิจิทัลเข้ามาเพิ่มประสิทธิภาพของระบบความปลอดภัย เนื่องจากข้อได้เปรียบของการตรวจสอบอัตลักษณ์ คือ ไม่สามารถปลอม ไม่โดนขโมย และไม่สูญหายเมื่อเปรียบเทียบกับบัตรเครดิตหรือสมาร์ตการ์ด อีกทั้งไม่ต้องจำรหัสผ่านหรือเลขรหัสต่าง ๆ ทำให้เป็นที่สนใจของผู้ที่ต้องการนำไปพัฒนาใช้งานกับระบบทางด้านความปลอดภัยเป็นอย่างมาก

ในการประยุกต์ใช้งานด้านประมวลภาพดิจิทัลในระบบไบโอเมตริกผู้พัฒนาจะต้องออกแบบระบบฮาร์ดแวร์ที่เหมาะสมกับการใช้งาน เช่น การตรวจสอบลายนิ้วมือสำหรับการเข้าถึงสถานที่หรือข้อมูล ผู้ออกแบบควรเลือกอุปกรณ์ฮาร์ดแวร์ที่มีขนาดเล็ก ใช้พลังงานต่ำ และราคาประหยัด ตัวประมวลผลสัญญาณดิจิทัล (Digital Signal Processor) หรือ DSP เป็นอุปกรณ์หนึ่งที่เหมาะสม เนื่องจากมีขนาดเล็กและสามารถออกแบบให้ติดต่อกับอุปกรณ์ภายนอกได้หลายรูปแบบ นอกจากนี้ยังมีเครื่องมือช่วยในการเขียนโปรแกรม เช่น เครื่องมือ DSP/BIOS ทำให้ง่ายต่อการพัฒนาซอฟต์แวร์ระบบด้วย

ในงานวิจัยนี้จะทำการออกแบบและพัฒนาซอฟต์แวร์ระบบปฏิบัติการโดยใช้เครื่องมือ DSP/BIOS (Digital Signal Processor/ Basic Input Output System) ของบริษัท Texas Instruments ให้สามารถทำงานได้ตามความต้องการของระบบไบโอเมตริกและใช้สำหรับการพัฒนาขั้นตอนวิธีการตรวจสอบลายนิ้วมือแบบเวลาจริงบนฮาร์ดแวร์ประมวลผลสัญญาณดิจิทัล รวมทั้งพัฒนาฮาร์ดแวร์เชื่อมต่อเครือข่ายเพื่อเพิ่มความสามารถให้กับระบบในการสื่อสารผ่านเครือข่ายพื้นที่ท้องถิ่น (Local Area Network) หรือ LAN ได้ ซึ่งระบบไบโอเมตริกที่พัฒนาขึ้นสามารถใช้เป็นต้นแบบเชิงอุตสาหกรรมของไทยในอนาคตอันใกล้นี้

วัตถุประสงค์ของการวิจัย

วัตถุประสงค์ของการพัฒนาระบบปฏิบัติการไบโอเมตริกแบบเวลาจริงบนตัวประมวลสัญญาณดิจิทัลผ่านเครือข่ายมีดังนี้

1. เพื่อพัฒนาระบบปฏิบัติการไบโอเมตริกแบบเวลาจริงบนประมวลผลสัญญาณดิจิทัลโดยใช้ DSP/BIOS ซึ่งเป็นเคอร์เนลมาตรฐานบนตัวประมวลผลสัญญาณดิจิทัลของบริษัท Texas Instruments ทำให้ระบบปฏิบัติการที่พัฒนาไม่ขึ้นอยู่กับตัวประมวลสัญญาณดิจิทัลเบอร์ใดเบอร์หนึ่งของบริษัทนี้ และสามารถนำไปใช้กับตัวประมวลสัญญาณดิจิทัลรุ่นใหม่ ๆ ของบริษัทนี้ในอนาคตได้ สาเหตุที่ใช้ตัวประมวลผลสัญญาณดิจิทัลของบริษัท Texas Instruments เนื่องจากเป็นบริษัทที่มีความก้าวหน้าและมั่นคงในการพัฒนาตัวประมวลผลสัญญาณดิจิทัลที่ประสิทธิภาพสูงในปัจจุบัน
2. เพื่อพัฒนาระบบปฏิบัติการไบโอเมตริกที่มีความยืดหยุ่นในการปรับเปลี่ยนขั้นตอนวิธีใหม่ได้โดยไม่ซับซ้อนและไม่ต้องพัฒนาซอฟต์แวร์ระบบใหม่
3. เพื่อลดเวลาในการพัฒนาระบบโดยรวม เนื่องจากผู้พัฒนาฮาร์ดแวร์ประมวลผลและผู้พัฒนาขั้นตอนวิธีสามารถพัฒนาได้อย่างอิสระ โดยมีซอฟต์แวร์ระบบปฏิบัติการไบโอเมตริกเป็นตัวกลางเชื่อมระหว่างฮาร์ดแวร์และขั้นตอนวิธี
4. เพื่อพัฒนาระบบไบโอเมตริกให้สามารถติดต่อสื่อสารผ่านทางเครือข่ายที่ใช้อยู่ในปัจจุบันได้

การตรวจเอกสาร

ไบโอเมตริก

การตรวจสอบพิสูจน์หรือระบุตัวบุคคลเป็นสิ่งสำคัญในงานที่ต้องการยืนยันว่าบุคคลนั้น ๆ ได้รับการอนุญาตให้สามารถใช้งานได้ ตัวอย่างเช่น การผ่านเข้าออกอาคาร การใช้งานคอมพิวเตอร์ การเข้าใช้เครื่องรับจ่ายเงินอัตโนมัติ เป็นต้น ซึ่งสิ่งที่จะใช้ในการตรวจสอบพิสูจน์หรือระบุตัวบุคคลนั้นจะต้องเป็นสิ่งที่มีเอกลักษณ์เฉพาะตัว ไม่สามารถปลอมแปลงได้หรือปลอมแปลงได้ยาก และไม่เปลี่ยนแปลงเมื่อเวลาผ่านไป อาทิเช่น ใบหน้า ลายม่านตา ลายนิ้วมือ เสียง ฯลฯ ซึ่งการใช้คุณสมบัติทางสรีรวิทยาหรือทางกายภาพในแต่ละบุคคลมาพิสูจน์นั้น เรียกว่า ไบโอเมตริก (Biometric)

คุณสมบัติทางสรีรวิทยาหรือลักษณะเฉพาะของมนุษย์นั้นเป็นสิ่งที่แตกต่างกันไปในแต่ละบุคคล คุณสมบัติดังกล่าวที่จะนำมาวิเคราะห์และทำการรู้จำจะต้องมีคุณสมบัติที่สำคัญ ดังต่อไปนี้ (Jain *et al*., 1999)

1. ลักษณะที่มีทั่วไป (Universality) แต่ละคนควรมีลักษณะนี้อยู่หรือมีอยู่ในคนทุกคน เช่น ใบหน้า ลายม่านตา รหัสดีเอนเอ (Deoxyribo Nucleic Acid, DNA) เป็นต้น

2. ลักษณะที่เฉพาะบุคคล (Distinctiveness) ควรจะมีรูปแบบของลักษณะที่ไม่เหมือนกันในแต่ละคน เช่น ลายนิ้วมือ ลายม่านตา หรือดีเอนเอ เป็นต้น

3. ลักษณะที่คงทน (Permanence) หมายถึงลักษณะที่ไม่มีเปลี่ยนแปลงเมื่อเวลาผ่านไป หรือเปลี่ยนแปลงน้อยเมื่อเทียบกับช่วงที่ใช้งานเปรียบเทียบไบโอเมตริก เช่น ลายนิ้วมือ ลายม่านตา ดีเอนเอ เป็นต้น

4. ลักษณะที่เก็บได้ (Collectability) ควรเป็นลักษณะที่เป็นปริมาณที่วัดและเก็บได้ในทางปฏิบัติ ซึ่งทำให้สามารถวิเคราะห์คุณลักษณะได้ เช่น ใบหน้า ลายมือชื่อ เป็นต้น

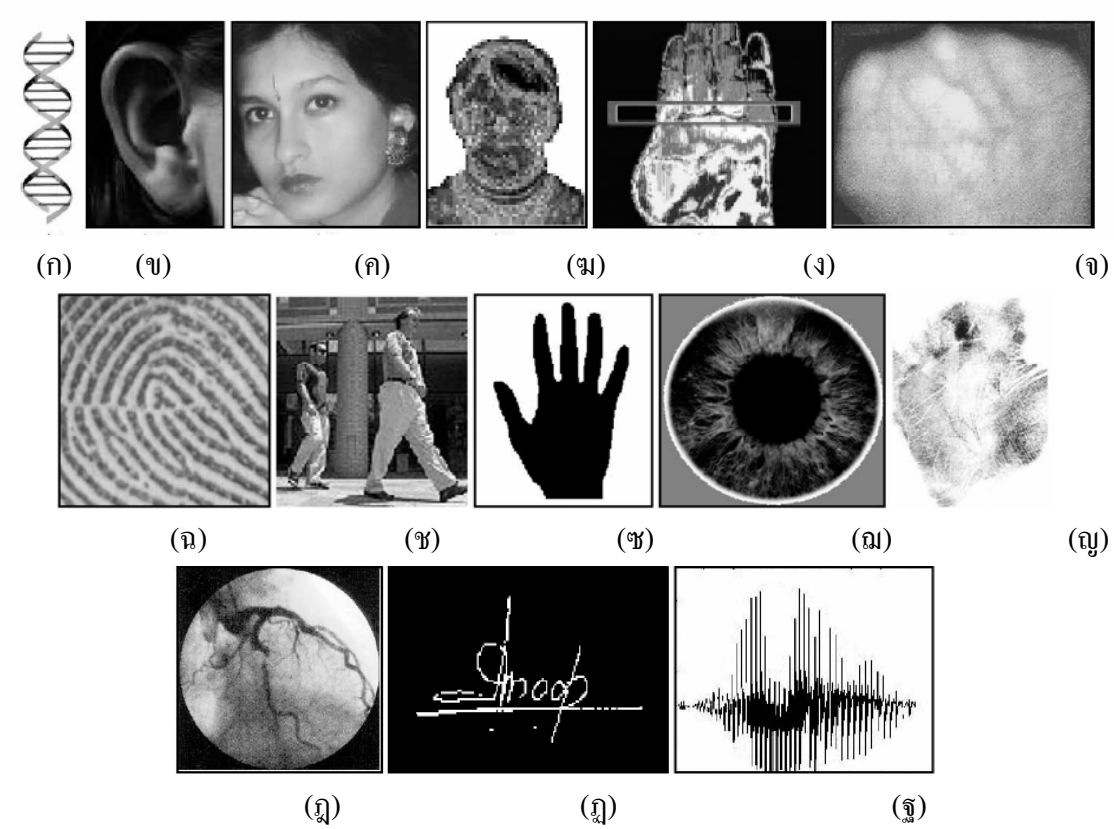
นอกจากนี้ในการเลือกใช้ระบบไบโอเมตริกในทางปฏิบัตินั้น ยังจะต้องมีคุณสมบัติของระบบที่จะต้องพิจารณา ได้แก่

1. ประสิทธิภาพ (Performance) หมายถึงความสามารถของระบบในการพิสูจน์บุคคลได้อย่างแม่นยำ ความรวดเร็ว และใช้ทรัพยากรของระบบน้อย เช่น การตรวจสอบลายนิ้วมือ การตรวจสอบลายม่านตา เป็นต้น

2. การยอมรับ (Acceptability) หมายถึงระบบที่ได้รับการยอมรับจากบุคคลส่วนใหญ่ว่าสะดวกและสามารถใช้งานได้ในชีวิตประจำวัน เช่น การตรวจสอบใบหน้า การตรวจสอบลายเซ็นต์ เป็นต้น ในขณะที่การตรวจสอบลายนิ้วมืออาจไม่เป็นที่ยอมรับในบางประเทศ เช่น ญี่ปุ่น ออสเตรเลีย

3. การหลอกลวงระบบ (Circumvention) หมายถึงความสามารถในการปลอมแปลงคุณลักษณะบนร่างกายเพื่อให้สามารถเข้าใช้งานระบบได้ เช่น นิ้วปลอมสำหรับการตรวจสอบลายนิ้วมือ ม่านตาปลอมสำหรับการตรวจสอบลายม่านตา เป็นต้น

จากลักษณะดังกล่าวในข้างต้นนั้น ได้มีการวิจัยและพัฒนาอย่างกว้างขวางเพื่อให้ได้คุณลักษณะสำคัญในร่างกายมนุษย์ที่จะนำมาใช้ในระบบไบโอเมตริก ซึ่งในแต่ละบุคคลนั้นจะมีคุณลักษณะที่แตกต่างกันไป ดังแสดงในภาพที่ 1 ได้แก่ การตรวจสอบดีเอนเอ, การตรวจสอบรูปร่างของใบหู, การตรวจสอบลักษณะของใบหน้า, การตรวจสอบลักษณะของความร้อนในร่างกาย, การตรวจสอบลักษณะของเส้นเลือดบนฝ่ามือ, การตรวจสอบลายนิ้วมือ, การตรวจสอบลักษณะของท่าทางการเดิน, การตรวจสอบลักษณะของฝ่ามือ, การตรวจสอบลักษณะของลายม่านตา, การตรวจสอบลักษณะของลายมือชื่อ, การตรวจสอบลักษณะของเสียงพูด เป็นต้น (Jain *et al*., 2004)



(ก) (ข) (ค) (ฆ) (ง) (จ)

(ฉ) (ช) (ซ) (ฌ) (ญ)

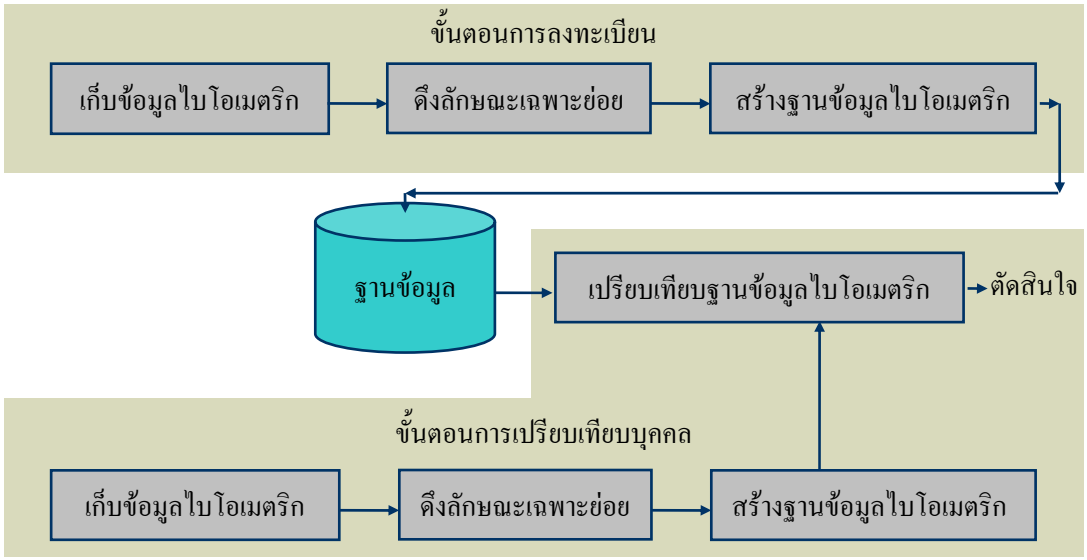
(ฎ) (ฏ) (ฐ)

ภาพที่ 1 ตัวอย่างคุณลักษณะทางไบโอเมตริก (ก) ดีเอนเอ (ข) ใบหู (ค) ใบหน้า (ฆ) ความร้อนจากใบหน้า (ง) ความร้อนบริเวณฝ่ามือ (จ) เส้นเลือดบริเวณฝ่ามือ (ฉ) ลายนิ้วมือ (ช) ท่าทางการเดิน (ซ) ฝ่ามือ (ฌ) ลายม่านตา (ญ) ลายฝ่ามือ (ฎ) เยื่อภายในลูกตาหรือเรตินา (Ratina) (ฏ) ลายมือชื่อ (ฐ) สัญญาณของเสียง

ที่มา: Jain *et al*. (2004)

ระบบไบโอเมตริกเป็นระบบพิสูจน์บุคคลที่ทำงานโดยอาศัยคุณลักษณะเฉพาะบนร่างกายหรืออาการท่าทางที่ทำให้เกิดความแตกต่างกันดังที่กล่าวในข้างต้นของแต่ละบุคคลมาแยกแยะว่าใครเป็นใคร ซึ่งเป็นเหมือนคำถามที่ว่า “คุณเป็นใคร” และนำสิ่งนั้นมาใช้ในการพิสูจน์บุคคล ต่างจากวิธีการแบบเดิมที่ใช้เพียง “คุณรู้อะไร” ตัวอย่างเช่น การใช้รหัสผ่าน หรือ “คุณมีอะไร” เช่น การใช้กุญแจหรือบัตรผ่าน เป็นต้น ดังนั้นระบบไบโอเมตริกจึงมีความสามารถในการแยกแยะระหว่างบุคคลที่เป็นตัวจริงกับบุคคลที่เป็นตัวปลอมซึ่งพยายามหลอกหลวงระบบ (Jain *et al*., 2004)

ในการทำงานของระบบไบโอเมตริกไม่ว่าจะนำคุณลักษณะเฉพาะแบบใดมาใช้งานจะมีหลักการทำงานพื้นฐานคล้ายกัน ดังแสดงในภาพที่ 2



ภาพที่ 2 หลักการทำงานของระบบไบโอเมตริก

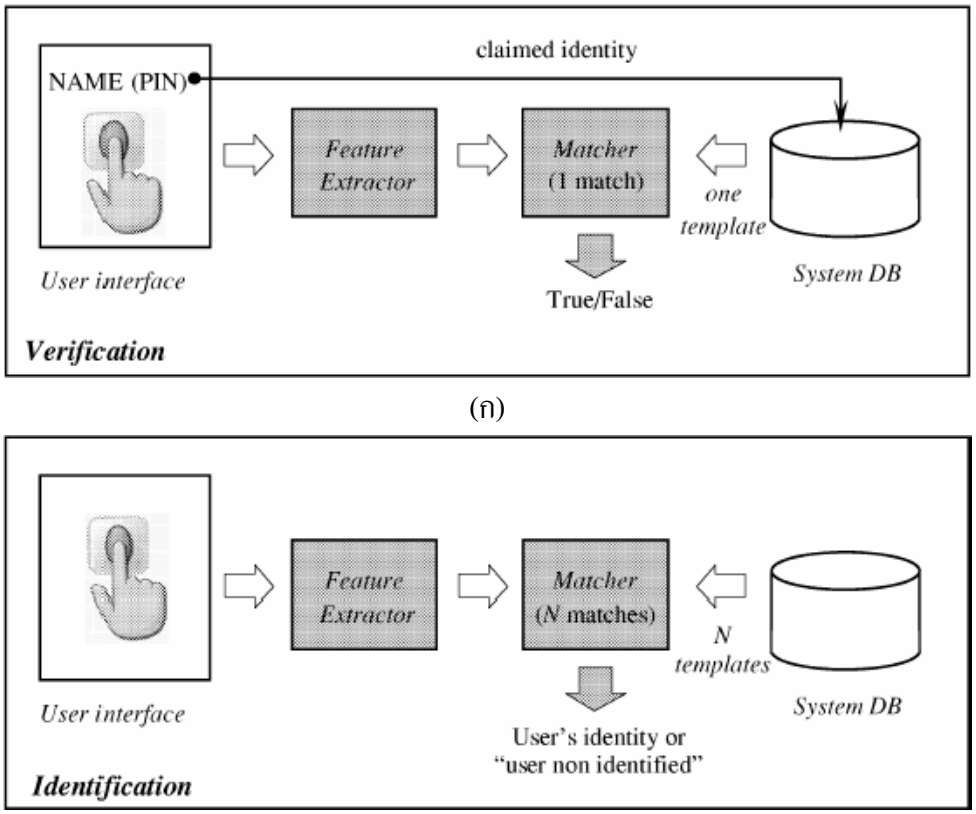
จากภาพสามารถแบ่งขั้นตอนการทำงานของระบบไบโอเมตริกออกเป็น 2 ขั้นตอน คือ

1. ขั้นตอนการลงทะเบียน (Enrollment Process)
2. ขั้นตอนการเปรียบเทียบบุคคล (Verification Process)

ในการทำงานของระบบไบโอเมตริกนั้น บุคคลที่ได้รับอนุญาตนั้นจะต้องทำการลงทะเบียน (Enrollment) ก่อน ซึ่งระบบจะเก็บข้อมูลไบโอเมตริกเข้ามาโดยใช้เครื่องอ่านคุณลักษณะทางไบโอเมตริกเฉพาะ โดยข้อมูลไบโอเมตริกที่ได้จะนำไปประมวลผลวิเคราะห์และแยกลักษณะเฉพาะย่อยออกมาแปลงเป็นโครงสร้างข้อมูลไบโอเมตริกที่เหมาะสม จากนั้นก็จะเก็บข้อมูลไบโอเมตริกที่ได้ลงในระบบฐานข้อมูลพร้อมกับรหัสชื่อของบุคคลที่ลงทะเบียน เพื่อเป็นแม่แบบ (Template) สำหรับนำไปเปรียบเทียบพิสูจน์บุคคลต่อไป

สำหรับขั้นตอนเปรียบเทียบบุคคลนั้น ในขั้นแรกระบบจะทำการเก็บข้อมูลไบโอเมตริกเข้ามา โดยใช้เครื่องอ่านคุณลักษณะทางไบโอเมตริกโดยเฉพาะในลักษณะเดียวกับการลงทะเบียน จากนั้นจะทำการประมวลผลวิเคราะห์และแยกลักษณะเฉพาะย่อยออกมาแปลงเป็นโครงสร้างข้อมูลไบโอเมตริกที่เหมาะสม แล้วทำการเปรียบเทียบคู่ลักษณะเฉพาะย่อย (Feature matching) ที่ได้

กับข้อมูลที่มีอยู่ในฐานข้อมูลเพื่อสอบพิสูจน์ตัวบุคคลนั้น ๆ ขั้นตอนการเปรียบเทียบบุคคลสามารถแบ่ง 2 ประเภท ได้แก่ การสอบพิสูจน์บุคคล (Verification) และการระบุตัวบุคคล (Identification) ดังภาพที่ 3 ซึ่งขึ้นอยู่กับลักษณะการใช้งาน โดยมีรายละเอียดดังต่อไปนี้



ภาพที่ 3 แผนภาพขั้นตอนการใช้งานไบโอเมตริก (ก) ขั้นตอนการตรวจสอบบุคคล (ข) ขั้นตอนการระบุตัวบุคคล

ที่มา: Jain *et al*. (2004)

1. การสอบพิสูจน์บุคคล (Verification) เป็นการเปรียบเทียบข้อมูลไบโอเมตริกที่เก็บเข้ามากับข้อมูลเฉพาะที่ถูกอ้างถึงเฉพาะในระบบฐานข้อมูล เพื่อเป็นการพิสูจน์ว่าเป็นบุคคลเดียวกับที่ได้รับอนุญาตจากระบบหรือไม่ ซึ่งการอ้างถึงข้อมูลเฉพาะที่เก็บไว้ในระบบฐานข้อมูลสามารถอ้างได้จากรหัสเฉพาะประจำตัว ได้แก่ ตัวเลขระบุตัวบุคคล (Personal Identification Number) บัตร RFID (Radio Frequency Identification Card) เป็นต้น จากนั้นระบบจะทำการเปรียบเทียบข้อมูลกับฐานข้อมูลแบบหนึ่งต่อหนึ่ง (One-to-one) และบอกว่าใช่บุคคลที่อ้างถึงหรือไม่ ตัวอย่างการใช้งานระบบไบโอเมตริกในรูปแบบนี้ เช่น การผ่านเข้าออกอาคารหรือสถานที่ที่มีการรักษาความปลอดภัย เป็นต้น

2. การระบุตัวบุคคล (Identification) เป็นการเปรียบเทียบข้อมูลทางไบโอเมตริกที่เก็บเข้ามากับข้อมูลไบโอเมตริกเฉพาะทั้งหมดที่เก็บอยู่ในระบบฐานข้อมูล เพื่อระบุว่าเป็นใครและอนุญาตให้ใช้งานหรือเข้าระบบได้ ซึ่งลักษณะของการเปรียบเทียบข้อมูลนั้นจะเป็นแบบหนึ่งต่อหลายหน่วย (One-to-many) การใช้งานไบโอเมตริกในรูปแบบนี้จะนำไปใช้กับระบบเกี่ยวกับการค้นหาบุคคล เช่น ในด่านตรวจคนเข้าเมือง ข้อมูลของคนที่ไม่อนุญาตเข้าเมืองจะเก็บในฐานข้อมูล เมื่อมีผู้ที่ไม่อนุญาตเข้ามาระบบจะแจ้งเตือนให้เจ้าหน้าทราบได้ เป็นต้น

งานประยุกต์ของระบบไบโอเมตริกดังที่กล่าวมาสามารถแบ่งออกได้เป็น 3 กลุ่มหลัก ๆ คือ (Jain *et al*., 2004)

1. ด้านการค้า (Commercial) ได้แก่ การผ่านเข้าออกอาคารหรือสถานที่ การลงบันทึกเข้าทำงาน การรักษาความปลอดภัยของข้อมูล การเข้าใช้เครื่องรับจ่ายเงินอัตโนมัติ เป็นต้น

2. ด้านการปกครอง (Government) ได้แก่ บัตรประชาชน ใบขับขี่ การควบคุมหนังสือเดินทาง การตรวจคนเข้าเมือง การลงคะแนนเสียงเลือกตั้ง เป็นต้น

3. ด้านนิติเวช (Forensic) ได้แก่ การตรวจสอบผู้เสียชีวิต การสืบสวนคดีฆาตรกรรม เช่น ตรวจสอบลายนิ้วมือฆาตรกร การสอบพิสูจน์ผู้ก่อการร้าย เป็นต้น

ส่วนต่อประสานโปรแกรมประยุกต์สำหรับระบบไบโอเมตริก

อุตสาหกรรมทางด้านไบโอเมตริกมีผู้ผลิตอยู่มากมายทั้งทางฮาร์ดแวร์และซอฟต์แวร์ โดยแต่ละผู้ผลิตนั้นมีการออกแบบระบบที่มีอัลกอริทึม รูปแบบการติดต่อ และโครงสร้างข้อมูลเป็นของตัวเอง เนื่องจากไม่มีมาตรฐานในการเชื่อมต่อทำให้ผลิตภัณฑ์ของผู้ผลิตที่ต่างกันไม่สามารถทำงานร่วมกันได้ ซึ่งเป็นปัญหาสำคัญที่ทำให้ผลิตภัณฑ์ของไบโอเมตริกไม่ถูกนำไปใช้อย่างกว้างขวาง ดังนั้นผู้ผลิตเหล่านี้จึงได้ร่วมกันตั้งมาตรฐานในการกำหนดส่วนต่อประสานโปรแกรมประยุกต์และการใช้ข้อมูลร่วมกัน ทำให้ผู้พัฒนาซอฟต์แวร์สามารถใช้อุปกรณ์หรือเทคโนโลยีต่าง ๆ ในรูปแบบเดียวกัน มาตรฐานที่กำหนดขึ้นทำให้เทคโนโลยีและอุตสาหกรรมทางด้านไบโอเมตริกมีการเติบโตอย่างรวดเร็ว (Bubeck and Sanchez, 2003)

ส่วนต่อประสานโปรแกรมประยุกต์สำหรับระบบไบโอเมตริก (Biometric Application Programming Interface) หรือไบโอเอพีไอ (BioAPI) เป็นมาตรฐานหนึ่งที่ถูกกำหนดขึ้นโดยสมาคมไบโอเอพีไอ ซึ่งเป็นสมาคมที่เกิดจากองค์กรต่าง ๆ ในปัจจุบันมีสมาชิกรวมทั้งหมดมากกว่า 100 องค์กร จากกลุ่มอุตสาหกรรม หน่วยงานของรัฐ หรือสถาบันการศึกษา (http://www.bioapi.org, 2004) เพื่อพัฒนาส่วนต่อประสานโปรแกรมประยุกต์สำหรับระบบไบโอเมตริกให้เป็นที่ยอมรับและใช้ได้อย่างกว้างขวาง ซึ่งสามารถใช้ได้กับเทคโนโลยีไบโอเมตริกประเภทต่าง ๆ ได้

มาตรฐานของส่วนต่อประสานโปรแกรมประยุกต์สำหรับระบบไบโอเมตริกได้นิยามวิธีการติดต่อระหว่างผู้พัฒนาโปรแกรมประยุกต์ (Application Developer) และผู้ให้บริการไบโอเมตริก (Biometric Service Provider, BSP) ซึ่งทำให้ผู้พัฒนาโปรแกรมประยุกต์สามารถพัฒนาโปรแกรมให้ทำงานกับเทคโนโลยีไบโอเมตริกหลากหลายประเภทภายใต้มาตรฐานเดียวกัน ประโยชน์ที่ได้รับจากการใช้มาตรฐานส่วนต่อประสานโปรแกรมประยุกต์สำหรับระบบไบโอเมตริกในการเขียนโปรแกรมบนระบบไบโอเมตริกมีดังนี้ (Tilton, 2003)

1. สามารถพัฒนาโปรแกรมประยุกต์ได้อย่างรวดเร็ว เนื่องจากผู้พัฒนาโปรแกรมไม่ต้องสนใจการทำงานภายในของฮาร์ดแวร์

2. สามารถเปลี่ยนแปลงเทคโนโลยีไบโอเมตริกแบบต่าง ๆ ได้ง่าย

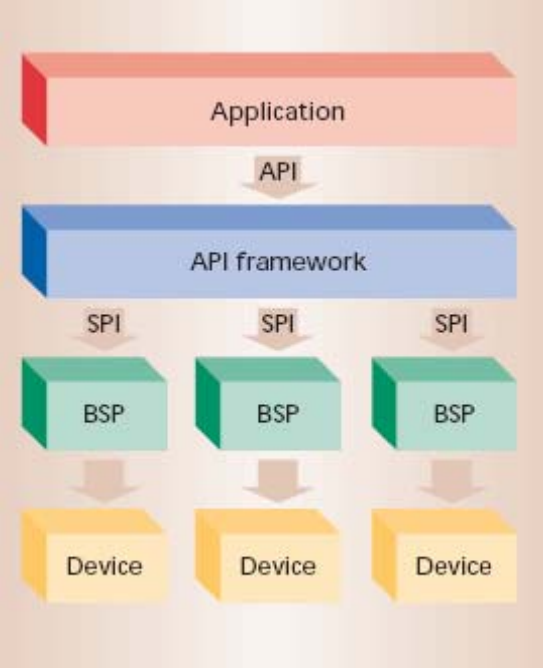
3. สามารถใช้งานไบโอเมตริกแบบต่าง ๆ พร้อมกันภายใต้การเชื่อมต่อหรือมาตรฐานเดียวกันได้

4. สามารถพัฒนาโปรแกรมทางด้านฮาร์ดแวร์หรือซอฟต์แวร์ผู้ให้บริการไบโอเมตริกโดยไม่มีผลกระทบต่อโปรแกรมประยุกต์

5. สำหรับงานประยุกต์งานเดียวสามารถใช้เทคโนโลยีไบโอเมตริกได้หลายประเภท ซึ่งทำให้สามารถเปรียบเทียบประสิทธิภาพของเทคโนโลยีไบโอเมตริกได้

6. สำหรับงานประยุกต์หลายงานสามารถใช้เทคโนโลยีไบโอเมตริกที่เหมือนกันได้ ซึ่งทำให้สามารถเปรียบเทียบประสิทธิภาพของงานประยุกต์สำหรับไบโอเมตริกได้

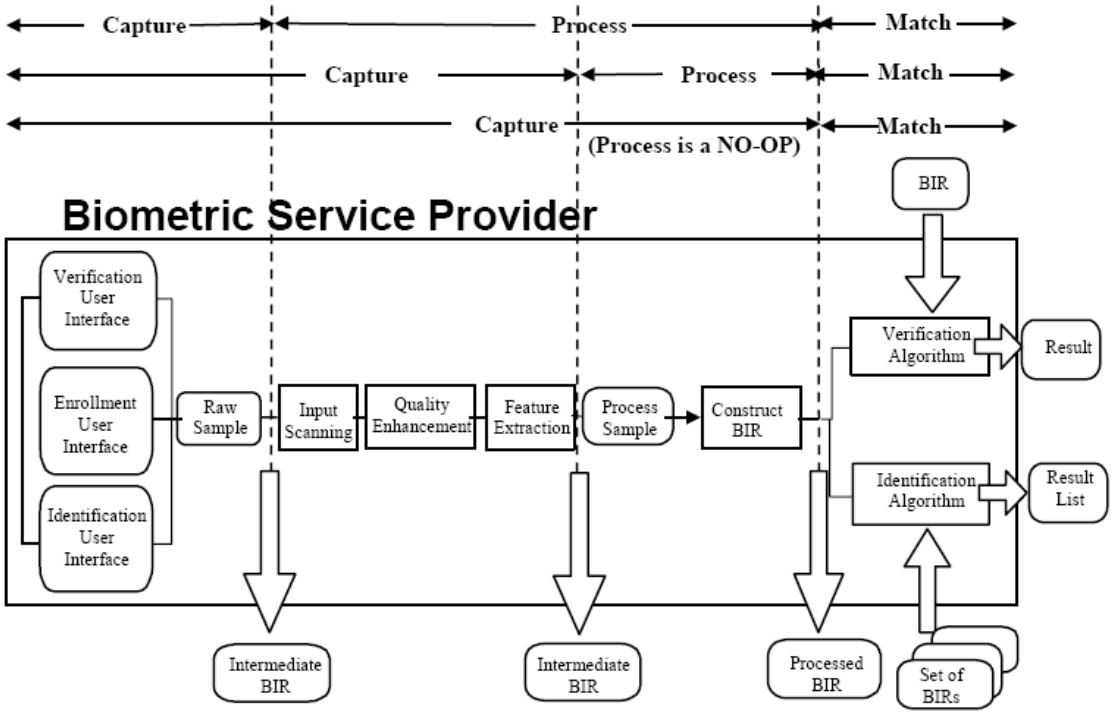
สถาปัตยกรรมพื้นฐานของมาตรฐานส่วนต่อประสานโปรแกรมประยุกต์สำหรับระบบไบโอเมตริกสามารถแสดง ๆ ได้ดังภาพที่ 4



ภาพที่ 4 สถาปัตยกรรมของมาตรฐานไบโอเอพีไอ
ที่มา: Tilton (2000)

จากภาพที่ 4 โปรแกรมประยุกต์ (Application) จะติดต่อกับระบบผ่านทางส่วนต่อประสานโปรแกรมประยุกต์ (API) ไปยังกรอบงานส่วนต่อประสานโปรแกรมประยุกต์ (API Framework) โดยกรอบงานส่วนต่อประสานโปรแกรมประยุกต์จะเชื่อมต่อกับส่วนผู้ให้บริการไบโอเมตริก (Biometric Service Provider) หรือ BSP ผ่านทางส่วนต่อประสานสำหรับผู้ให้บริการ (Service Provider Interface) หรือ SPI ซึ่งสามารถใช้งานได้หลายตัว โดยที่ BSP จะติดต่อกับฮาร์ดแวร์ระบบหรืออุปกรณ์ไบโอเมตริก (Biometric Device) โดยตรง

ผู้ให้บริการไบโอเมตริก (BSP) เป็นส่วนที่ผู้พัฒนาระบบไบโอเมตริกต้องเตรียมฟังก์ชันการทำงานสำหรับระบบไบโอเมตริกไว้ให้ผู้พัฒนาโปรแกรมประยุกต์เรียกใช้ ซึ่งการทำงานของผู้ให้บริการไบโอเมตริกจะแบ่งเป็นการลงทะเบียน (Enrollment) การตรวจสอบบุคคล (Verification) และการระบุตัวบุคคล (Identification) ดังที่ได้อธิบายไว้ก่อน และแต่ละส่วนจะมีช่วงการทำงานแบบต่าง ๆ ดังภาพที่ 5



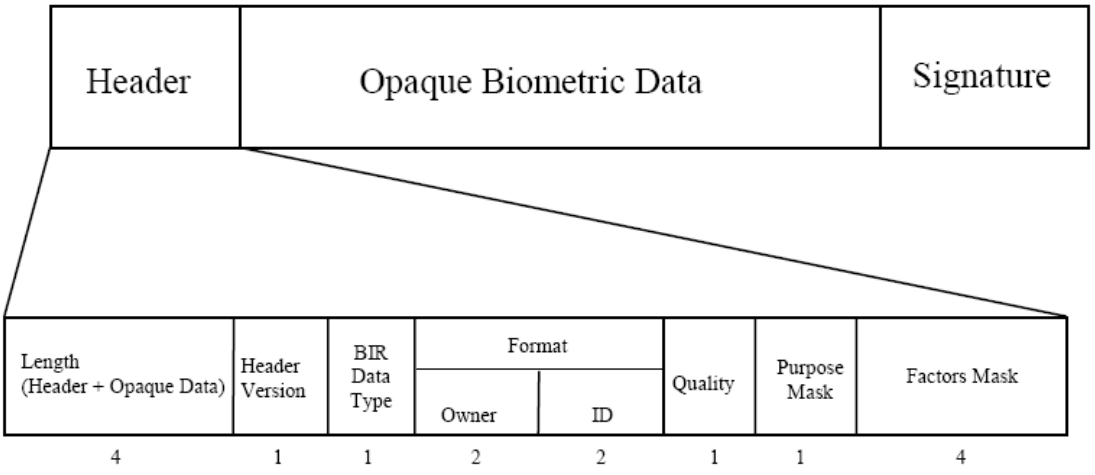
ภาพที่ 5 รูปแบบการทำงานของผู้ให้บริการไบโอเมตริก
ที่มา: Wilson (2001)

จากภาพที่ 5 ในบล็อกของผู้ให้บริการไบโอเมตริกประกอบด้วยขั้นตอนย่อยต่าง ๆ เช่น การเก็บข้อมูลดิบไบโอเมตริก (Raw Sample and Input Scanning) การปรับปรุงคุณภาพของข้อมูลดิบ (Quality Enhancement) การดึงลักษณะเฉพาะย่อย (Feature Extraction) การประมวลผลข้อมูล (Data Process) การสร้างข้อมูลบันทึกเฉพาะไบโอเมตริก (Biometric Identification Record Construction) หรือการสร้างข้อมูลบีไออาร์ (BIR Construction) การตรวจสอบข้อมูลที่ได้กับฐานข้อมูลที่อ้างถึง (Data Verification) การระบุหาข้อมูลในฐานข้อมูล (Data Identification) เป็นต้น โดยที่ขั้นตอนย่อย ๆ ดังกล่าวนี้สามารถจัดกลุ่มเป็นฟังก์ชันปฐมฐาน (Primitive Function) ได้แก่ การเก็บข้อมูลดิบ (Capture) การประมวลผล (Process) และการจับคู่ (Match) ซึ่งแสดงในส่วนบนของภาพที่ 5 โดยที่การกำหนดฟังก์ชันจะมีระดับขั้นความเสรี (Degree of Freedom) ในการกำหนดช่วงการทำงานของฟังก์ชัน กล่าวคือ ช่วงของการทำงานในแต่ละฟังก์ชันสามารถกำหนดได้จากผู้ให้บริการไบโอเมตริก

ในแต่ละช่วงการทำงานของฟังก์ชันปฐมฐาน ผู้ให้บริการไบโอเมตริกจะเก็บข้อมูลที่ได้รวมเป็นโครงสร้างข้องมูลที่เรียกว่า บันทึกข้อมูลเฉพาะไบโอเมตริก (Biometric Identification Record) หรือบีไออาร์ (BIR) เพื่อส่งให้โปรแกรมประยุกต์หรือเก็บไว้ใช้ในช่วงต่อไป ชนิดของข้อมูลบีไออาร์ขึ้นอยู่กับขั้นของการทำงานในฟังก์ชันปฐมฐาน ได้แก่ บันทึกข้อมูลเฉพาะไบโอเมตริกขั้นกลาง (Intermediate BIR) เป็นบีไออาร์ที่ในช่วงก่อนประมวลผล ซึ่งอาจเป็นข้อมูลดิบหรือข้อมูลที่ทำการประมวลผลบางส่วนมาก่อน (Pre-processing) แล้ว สำหรับบันทึกข้อมูลเฉพาะไบโอเมตริกหลังประมวลผล (Processed BIR) เป็นบีไออาร์ในช่วงหลังประมวลผลแล้ว ซึ่งเป็นข้อมูลบีไออาร์ที่พร้อมนำไปใช้ในการจับคู่เปรียบเทียบ

มาตรฐานของส่วนต่อประสานโปรแกรมประยุกต์สำหรับระบบไบโอเมตริกใช้คำว่าแม่แบบ (Template) ในการอ้างถึงข้อมูลที่ได้จากการลงทะเบียนไบโอเมตริกของผู้ใช้งานไว้แล้ว ซึ่งแม่แบบที่ได้จะนำไปจับคู่กับตัวอย่าง (Sample) ปัจจุบันที่ได้จากผู้ใช้ในการพิสูจน์บุคคล (Authentication) โดยทั่วไปแม่แบบดังกล่าว คือ ข้อมูลบันทึกข้อมูลเฉพาะไบโอเมตริกที่จะถูกจัดเก็บอย่างถาวรโดยโปรแกรมประยุกต์ ซึ่งจะต้องเป็นบันทึกข้อมูลเฉพาะไบโอเมตริกหลังการประมวลผลที่ได้จากการลงทะเบียนเท่านั้น

บันทึกข้อมูลเฉพาะไบโอเมตริกหรือบีไออาร์ มีโครงสร้างดังภาพที่ 6



ภาพที่ 6 โครงสร้างบันทึกข้อมูลเฉพาะไบโอเมตริกหรือบีไออาร์
ที่มา: Wilson (2001)

จากภาพที่ 6 โครงสร้างบันทึกข้อมูลเฉพาะไบโอเมตริกหรือบีไออาร์แบ่งออกเป็น 3 ส่วน ได้แก่ ส่วนหัว (Header) ข้อมูลไบโอเมตริกปกปิด (Opaque Biometric Data) และลายเซ็น (Signature) ในส่วนข้อมูลไบโอเมตริกปกปิดเป็นส่วนของข้อมูลที่จะใช้ในการรับส่ง โดยที่รูปแบบของข้อมูลจะถูกกำหนดจากเขตข้อมูลรูปแบบ (Format Field) ที่อยู่ในส่วนหัวข้อมูล ซึ่งจะเป็นรูปแบบตามมาตรฐานหรือไม่ก็ได้ นอกจากนี้ข้อมูลไบโอเมตริกปกปิดยังสามารถทำการเข้ารหัสความปลอดภัย (Encryption) ได้

ลายเซ็นเป็นทางเลือก (Optional) ซึ่งผู้ให้บริการไบโอเมตริกจะใส่ลงไปหรือไม่ก็ได้ ในกรณีที่มีลายเซ็นส่วนนี้จะเป็นค่าเกี่ยวกับการตรวจสอบความผิดพลาดของข้อมูล ซึ่งในส่วนนี้จะไม่มีข้อกำหนดเกี่ยวกับข้อมูลลายเซ็น

ในข้อมูลส่วนหัวจะมีการแบ่งเป็นเขตข้อมูล (Data Field) ต่าง ๆ โดยมีข้อกำหนดตามมาตรฐานของรูปแบบแฟ้มข้อมูลแลกเปลี่ยนไบโอเมตริกร่วม (Common Biometric Exchange File Format, CBEFF) ซึ่งส่วนต่อประสานโปรแกรมประยุกต์สำหรับระบบไบโอเมตริกเป็นหนึ่งในรูปแบบผู้สนับสนุนรูปแบบแฟ้มข้อมูลแลกเปลี่ยนไบโอเมตริกร่วม (CBEFF Patron Formats) มาตรฐานของรูปแบบแฟ้มข้อมูลแลกเปลี่ยนไบโอเมตริกร่วมถูกเขียนขึ้นโดยสถาบันเผยแพร่มาตรฐานแห่งชาติ (National Institute of Standards Publication) (Podio *et al*., 2001)

รูปแบบแฟ้มข้อมูลแลกเปลี่ยนไบโอเมตริกร่วมได้นิยามตัวแบบผู้สนับสนุนและไคล์เอนท์ (Patron/Client Model) โดยที่ส่วนแรกผู้สนับสนุน (Patron) คือกลุ่มองค์กรที่ร่วมกันพัฒนามาตรฐานหรือข้อกำหนดตัวข้อมูลไบโอเมตริก (Biometric Data Object) ตามความต้องการของรูปแบบแฟ้มข้อมูลแลกเปลี่ยนไบโอเมตริกร่วม (CBEFF Requirement) ตัวอย่างเช่น สมาคมไบโอเอพีไอ (BioAPI) และกลุ่มเอ็คซ์ 9.84 (X9.84) เป็นต้น ส่วนที่สองไคล์เอนท์ (Client) เป็นข้อกำหนดรูปแบบแฟ้มข้อมูลแลกเปลี่ยนไบโอเมตริกร่วม เช่น การกำหนดค่าในเขตข้อมูลต่าง ๆ ในส่วนหัวของโครงสร้างบันทึกข้อมูลเฉพาะไบโอเมตริก (Tilton, 2003)

ตัวอย่างของรูปแบบไคล์เอนท์ เช่น เขตข้อมูลรูปแบบเจ้าของ (Format Owner Field) ในส่วนหัวจากภาพที่ 6 จะถูกกำหนดโดยสมาคมอุตสาหกรรมไบโอเมตริกนานาชาติ (International Biometric Industry Association, IBIA) ซึ่งผู้ที่จะพัฒนาโปรแกรมส่วนผู้ให้บริการไบโอเมตริกจะต้องลงทะเบียนก่อน เพื่อให้ได้รหัสที่เป็นเอกลักษณ์ของแต่ละบริษัทที่ใช้มาตรฐานรูปแบบแฟ้มข้อมูลแลกเปลี่ยนไบโอเมตริกร่วม

เมื่อผู้ให้บริการไบโอเมตริกสร้างข้อมูลบันทึกข้อมูลเฉพาะไบโอเมตริกขึ้นมาใหม่นั้น ผู้ให้บริการไบโอเมตริกจะส่งเพียงค่าจัดกระทำ (Handle) หรือตัวชี้ (Pointer) ไปยังฟังก์ชันที่สร้างขึ้น กล่าวคือเป็นการส่งค่าโดยไม่จำเป็นต้องทำการย้ายข้อมูลบีไออาร์ในหน่วยความจำซึ่งบางครั้งข้อมูลบันทึกข้อมูลเฉพาะไบโอเมตริกอาจมีขนาดใหญ่มาก ทำให้ใช้เวลามากในการย้ายข้อมูลและประสิทธิภาพการทำงานลดลง ดังนั้นถ้าโปรแกรมประยุกต์ต้องการจัดการข้อมูลบีไออาร์ เช่น การเก็บข้อมูลลงฐานข้อมูลจะต้องทำงานผ่านทางค่าจัดกระทำ (Handle)

ในกรณีที่โปรแกรมประยุกต์ต้องการส่งข้อมูลบีไออาร์ไปให้โปรแกรมผู้ให้บริการไบโอเมตริก สามารถทำได้ 3 วิธี คือ

1. ส่งผ่านโดยอ้างอิงค่าจัดกระทำ (Handle)
2. ส่งผ่านโดยอ้างอิงค่าที่ได้จากการเปิดฐานข้อมูล เป็นการส่งข้อมูลบีไออาร์ในลักษณะการเข้าถึงฐานข้อมูล
3. ส่งผ่านโดยการแก้ข้อมูลบีไออาร์ เป็นการส่งข้อมูลบีไออาร์โดยสร้างข้อมูลบีไออาร์แล้วส่งไปรับข้อมูลมา

การทำงานของระบบตามมาตรฐานของส่วนต่อประสานโปรแกรมประยุกต์สำหรับระบบไบโอเมตริกนั้น จะมีรูปแบบการทำงานซึ่งแบ่งเป็น 2 แบบ คือ

1. การใช้ฟังก์ชันปฐมฐาน (Primitive Function) โดยฟังก์ชันปฐมฐานสำหรับส่วนต่อประสานโปรแกรมประยุกต์จะมีอยู่ 4 กลุ่มฟังก์ชัน ได้แก่ การเก็บข้อมูลดิบ (Capture) การประมวลผล (Process) การจับคู่ (Match) และการสร้างแม่แบบ (Create Template) มีรายละเอียดดังนี้

1.1 การเก็บข้อมูลดิบ (Capture) เป็นช่วงการเก็บตัวอย่างไบโอเมตริกเข้ามาในระบบ โดยที่ในช่วงการเก็บข้อมูลไบโอเมตริกอาจมีการประมวลผลข้อมูลด้วย ซึ่งถ้าการประมวลผลไม่สมบูรณ์ ฟังก์ชันจะส่งค่าบันทึกข้อมูลเฉพาะไบโอเมตริกขั้นกลางมาให้ เพื่อให้โปรแกรมเรียก

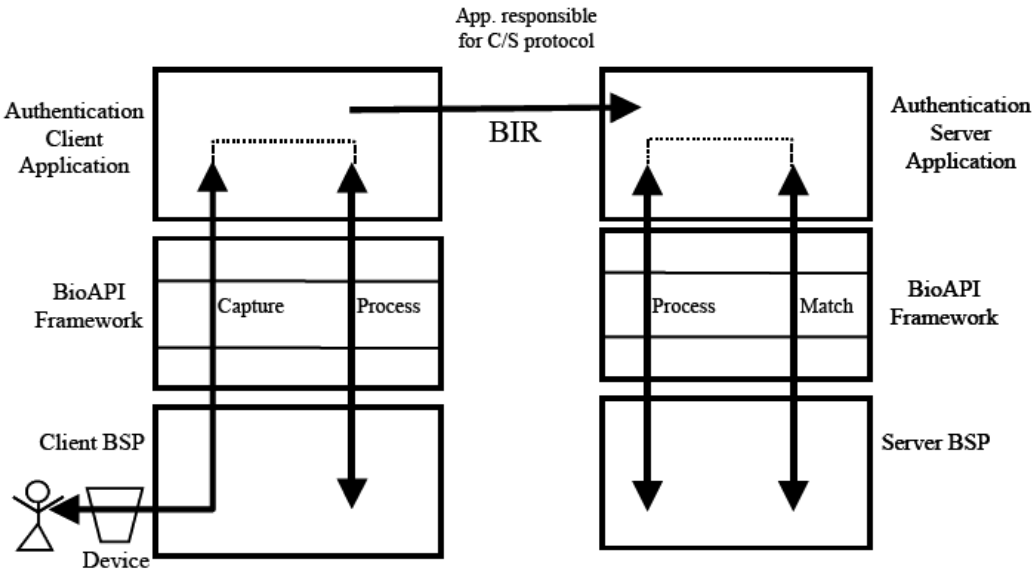
ฟังก์ชันประมวลผลต่อไป ในกรณีที่ประมวลผลเสร็จสมบูรณ์ ฟังก์ชันจะส่งค่าบันทึกข้อมูลเฉพาะไบโอเมตริกหลังประมวลผลมาให้

1.2 การประมวลผล (Process) จะใช้ซอฟต์แวร์อัลกอริทึมเฉพาะสำหรับแต่ละเทคโนโลยีไบโอเมตริก โดยฟังก์ชันจะรับบันทึกข้อมูลเฉพาะไบโอเมตริกขั้นกลางเข้ามาทำการประมวลผล หลังจากนั้นจะส่งบันทึกข้อมูลเฉพาะไบโอเมตริกหลังประมวลผลคืนไปยังโปรแกรมประยุกต์

1.3 การจับคู่ (Match) จะแบ่งเป็น 2 ประเภท คือ กรณีของการจับคู่ตรวจสอบบุคคล (Verify Match) เป็นการเปรียบเทียบของบันทึกข้อมูลเฉพาะไบโอเมตริกหลังประมวลผลกับแม่แบบ 1 แม่แบบ และกรณีของการจับคู่ระบุตัวบุคคล (Identify Match) เป็นการเปรียบเทียบบันทึกข้อมูลเฉพาะไบโอเมตริกหลังประมวลผลกับกลุ่มของแม่แบบ

1.4 การสร้างแม่แบบ (Create Template) ฟังก์ชันสร้างแม่แบบจะรับข้อมูลบันทึกข้อมูลเฉพาะไบโอเมตริกขั้นกลางนำไปสร้างเป็นแม่แบบ ซึ่งจะเป็นบันทึกข้อมูลเฉพาะไบโอเมตริกหลังประมวลผล โดยฟังก์ชันสร้างแม่แบบมีการทำงานคล้ายกับฟังก์ชันประมวลผล แต่ในฟังก์ชันสร้างแม่แบบจะใช้สำหรับการลงทะเบียนเท่านั้น นอกจากนี้ยังสามารถแทนที่แม่แบบเก่าด้วยแม่แบบใหม่จากข้อมูลบันทึกข้อมูลเฉพาะไบโอเมตริกขั้นกลางที่รับเข้ามาในกรณีที่แม่แบบใหม่มีความสมบูรณ์มากกว่าได้ด้วย

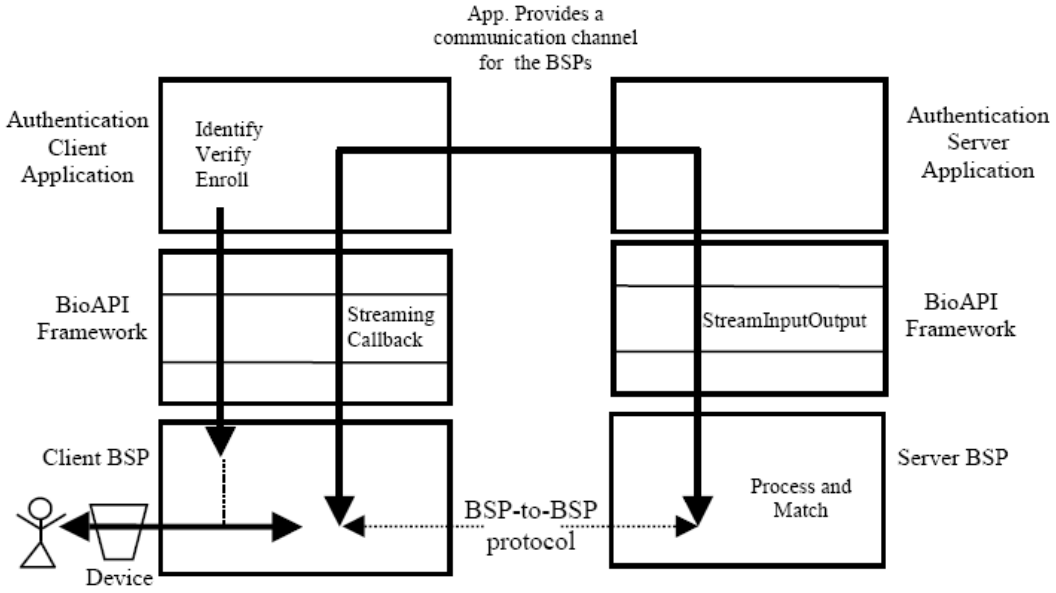
ในการติดต่อระหว่างระบบไบโอเมตริกนั้น ระบบจะติดต่อผ่านทางโปรแกรมประยุกต์โดยการรับส่งบันทึกข้อมูลเฉพาะไบโอเมตริกระหว่างกัน ดังภาพที่ 7 ในกรณีที่ระบบเป็นแบบเครื่องลูกข่ายและเครื่องบริการ (Client and Server) ขั้นตอนการทำงานอาจแบ่งอยู่คนละส่วน เนื่องจากความสามารถของระบบไม่เท่ากัน เช่น การจับคู่เปรียบเทียบระบุตัวบุคคล ถ้าจำนวนข้อมูลที่จะต้องเปรียบเทียบมีปริมาณมากควรส่งข้อมูลไปทำการจับคู่เปรียบเทียบที่เครื่องบริการ ซึ่งมีประสิทธิภาพสูงกว่า โดยที่จากภาพที่ 7 โปรแกรมประยุกต์ของเครื่องลูกข่ายจะทำการตรวจจับและประมวลผลก่อน หลังจากนั้นจะส่งข้อมูลในรูปบันทึกข้อมูลเฉพาะไบโอเมตริกไปยังโปรแกรมประยุกต์ของเครื่องบริการ เพื่อทำการประมวลผลและจับคู่เปรียบเทียบ



ภาพที่ 7 รูปแบบการใช้ฟังก์ชันปฐมฐานในระบบไบโอเมตริก
ที่มา: Wilson (2001)

2. การใช้วิธีเรียกคืนกระแสข้อมูล (Streaming Callback)

วิธีการนี้จะใช้ฟังก์ชันการกำหนดสาระสำคัญระดับสูง (High-level Abstraction Function) ได้แก่ ฟังก์ชันตรวจสอบบุคคล (Verify) ระบุตัวบุคคล (Identify) และลงทะเบียน (Enroll) แทนการใช้ฟังก์ชันปฐมฐาน โดยการทำงานทั้งหมดจะทำที่ผู้ให้บริการไบโอเมตริกผ่านทางการเรียกคืนกระแสข้อมูล (Streaming Callback) กล่าวอีกนัยหนึ่งคือ โปรแกรมประยุกต์จะเรียกใช้ฟังก์ชันในระดับการใช้งานของระบบเท่านั้น เช่น สั่งให้ระบบทำการลงทะเบียน ตรวจสอบบุคคล หรือระบุตัวบุคคล ซึ่งจะไม่เข้าไปจัดการขั้นตอนการทำงานต่าง ๆ ของระบบ เช่น การตรวจจับ การประมวลผลข้อมูล หรือการจับคู่เปรียบเทียบ เป็นต้น



ภาพที่ 8 รูปแบบการใช้วิธีเรียกคืนกระแสข้อมูลในระบบไบโอเมตริก
ที่มา: Wilson (2001)

ในการติดต่อระหว่างระบบไบโอเมตริกนั้นจะไม่ทำที่ส่วนของโปรแกรมประยุกต์ แต่โปรแกรมประยุกต์จะเตรียมช่องทางการติดต่อสื่อสารให้ โดยการเรียกฟังก์ชันขอรับข้อมูล ดังภาพที่ 8 ซึ่งการรับส่งข้อมูลนั้น ผู้ให้บริการไบโอเมตริกของแต่ละระบบจะทำการติดต่อกันเองผ่านเกณฑ์วิธีระหว่างผู้ให้บริการไบโอเมตริก (BSP-to-BSP Protocol) กรณีที่เป็นฝ่ายเรียกทำงานนั้นคำสั่งหรือฟังก์ชันสำหรับโปรแกรมประยุกต์ที่ขอรับส่งข้อมูลจะถูกเรียกผ่านคำสั่งเรียกคืนกระแสข้อมูล (Streaming Callback) ขณะที่อีกฝ่ายหนึ่งผู้ให้บริการไบโอเมตริกจะถูกเรียกผ่านฟังก์ชันกระแสข้อมูลเข้าออก (Stream Input Output)

จากภาพที่ 8 ผู้ให้บริการไบโอเมตริกของเครื่องลูกข่ายจะส่งข้อมูลที่ได้จากอุปกรณ์ตรวจจับไบโอเมตริกไปยังผู้ให้บริการไบโอเมตริกของเครื่องบริการ เพื่อทำการประมวลผลและจับคู่เปรียบเทียบข้อมูลผ่านทางฟังก์ชันเรียกคืนกระแสข้อมูล ในกรณีที่การทำงานของระบบไบโอเมตริกที่เป็นแบบเดี่ยว (Stand-alone) ตัวผู้ให้บริการไบโอเมตริกอาจไม่ต้องสนใจส่วนของการติดต่อแบบเรียกคืนกระแสข้อมูลได้ เนื่องจากระบบสามารถทำงานสมบูรณ์ด้วยตัวเองอยู่แล้ว

จากที่กล่าวมานั้น มาตรฐานส่วนต่อประสานโปรแกรมประยุกต์สำหรับระบบไบโอเมตริกหรือไบโอเอพีไอได้เตรียมฟังก์ชันต่าง ๆ มาสนับสนุนการทำงานของระบบ ประกอบด้วยฟังก์ชันพื้นฐาน ฟังก์ชันการกำหนดสาระสำคัญระดับสูง ฟังก์ชันปฐมฐาน ฯลฯ ซึ่งตารางที่ 1 ได้แสดงฟังก์ชันบางส่วนในมาตรฐานไบโอเอพีไอ

ตารางที่ 1 ฟังก์ชันบางส่วนในมาตรฐานไบโอเอพีไอ

Basic Functions	Primitive Functions
Module Management BioAPI_ModuleLoad BioAPI_ModuleAttach ***Data Handling*** BioAPI_GetBIRFromHandle BioAPI_GetHeaderFromHandle ***Callback & Event Operations*** BioAPI_SetStreamCallback ***Biometric Operations*** *BioAPI_Enroll* – Captures biometric data and creates template *BioAPI_Verify* – Captures live biometric data and matches it against one enrolled template *BioAPI_Identify* – Captures live biometric data and matches it against a set of enrolled templates	***BioAPI_Capture*** Captures raw/intermediate data from sensor ***BioAPI_Process*** Converts raw sample into processed template for matching ***BioAPI_CreateTemplate*** Converts raw sample(s) into processed template for enrolment ***BioAPI_VerifyMatch*** Performs a 1:1 match ***BioAPI_IdentifyMatch*** Performs a 1:N match ***BioAPI_Import*** Imports non-realtime data for processing

ที่มา: Tiltion (2003)

การพัฒนาระบบตามมาตรฐานไบโอเอพีไอนั้น แบ่งเป็น 2 ส่วน คือ ส่วนของโปรแกรมประยุกต์และส่วนของผู้ให้บริการไบโอเมตริก โดยส่วนของโปรแกรมประยุกต์ผู้พัฒนาโปรแกรมประยุกต์ต้องเขียนการทำงานให้ตรงกับข้อกำหนดที่วางเอาไว้ และค่าพารามิเตอร์ต่าง ๆ ที่จะส่งเข้าไปในฟังก์ชันต้องเป็นค่าที่มีจริงในโปรแกรม

ในส่วนผู้ให้บริการไบโอเมตริกจะต้องเตรียมฟังก์ชันต่าง ๆ ตามมาตรฐานที่กำหนดไว้ ซึ่งจะเป็นส่วนต่อประสานของผู้ให้บริการ (Service Provider Interface) ผู้ให้บริการไบโอเมตริกจะต้องรับค่าพารามิเตอร์ต่าง ๆ เข้ามาและคืนค่าพารามิเตอร์ตามข้อกำหนดของมาตรฐาน ส่วนฟังก์ชันหรือการทำงานที่เป็นทางเลือกนั้นไม่จำเป็นต้องมี แต่ถ้ามีการเตรียมฟังก์ชันที่เป็นทางเลือกต้องทำตามข้อกำหนดของมาตรฐานเช่นกัน

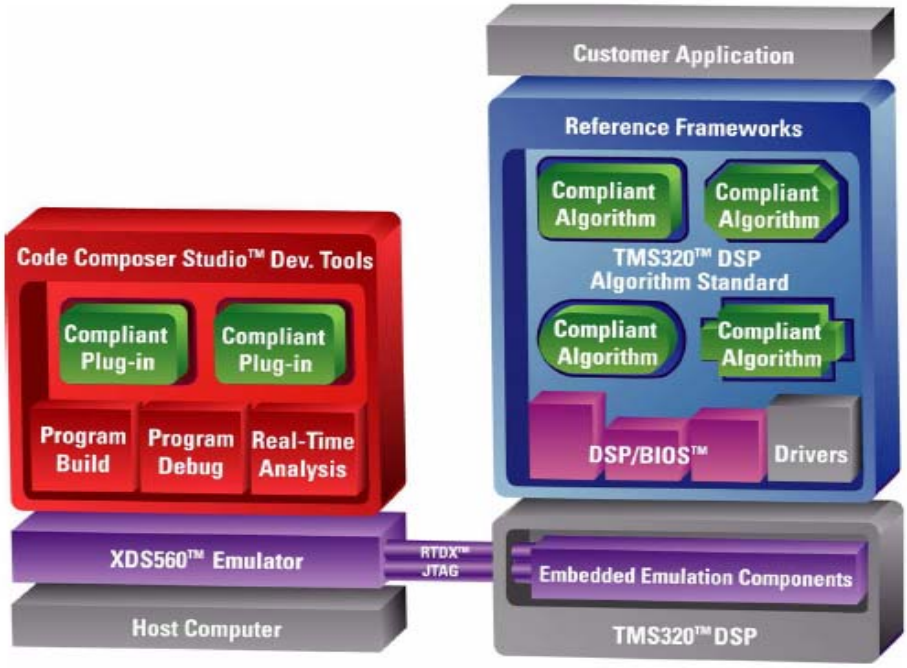
มาตรฐานขั้นตอนวิธีบนตัวประมวลผลสัญญาณดิจิทัล

ความก้าวหน้าทางเทคโนโลยีของตัวประมวลผลสัญญาณดิจิทัลช่วยให้เกิดการวิจัยและพัฒนางานประยุกต์ด้านต่าง ๆ เช่น ด้านโทรคมนาคม ด้านการประมวลผลภาพ วีดิทัศน์หรือเสียง เป็นต้น การพัฒนาขั้นตอนวิธี (Algorithm) บนตัวประมวลผลสัญญาณดิจิทัลในแต่ละงานประยุกต์มีความแตกต่างกันมาก เนื่องจากการออกแบบและใช้งานระบบที่แตกต่างกัน เช่น การจัดการหน่วยความจำที่ไม่เหมือนกันหรือการเขียนโปรแกรมที่แตกต่างกัน การที่ผู้พัฒนาขั้นตอนวิธี (Algorithm Developers) ไม่มีมาตรฐานในการเขียนโปรแกรมที่ตรงกัน ทำให้ไม่สามารถนำขั้นตอนวิธีที่พัฒนาแล้วในระบบหนึ่งไปยังอีกระบบหนึ่งได้ ตัวอย่างเช่น เมื่อต้องการย้ายขั้นตอนวิธีไปยังฮาร์ดแวร์แพลตฟอร์ม (Platform) ใหม่ ผู้พัฒนาระบบจะต้องพัฒนาขั้นตอนวิธีขึ้นมาใหม่ ทำให้เสียเวลาในการพัฒนา รวบรวมโปรแกรม และทดสอบระบบ (Torud, 2002)

มาตรฐานการทำงานร่วมกันสำหรับขั้นตอนวิธีบนตัวประมวลผลสัญญาณดิจิทัลตระกูล TMS320 (TMS320 DSP Algorithm Interoperability Standard) หรือรู้จักในชื่อ XDAIS เป็นส่วนหนึ่งของเครื่องมือ eXpressDSP ของบริษัท Texas Instruments ดังภาพที่ 9 มีจุดประสงค์ในการลดข้อแตกต่างระหว่างการเขียนหรือพัฒนาขั้นตอนวิธีการบนตัวประมวลผลสัญญาณดิจิทัล ซึ่งมาตรฐานขั้นตอนวิธี (Algorithm Standard) ที่ได้จะสามารถนำไปใช้กับตัวประมวลผลต่าง ๆ ในตระกูล TMS320 โดยที่ไม่ต้องพัฒนาขั้นตอนวิธีขึ้นมาใหม่ (Blonstein, 2002)

จากภาพที่ 9 ด้านซ้ายของภาพคือชุดเครื่องมือรวมสำหรับเขียนโปรแกรม (Code Composer Studio) ซึ่งช่วยให้ผู้พัฒนาซอฟต์แวร์สามารถเขียนโปรแกรมบนเครื่องคอมพิวเตอร์แม่ข่าย (Host Computer) ได้ ส่วนด้านขวาของภาพคือเครื่องมือ eXpressDSP บนตัวประมวลผลสัญญาณดิจิทัลตระกูล TMS320 ประกอบด้วย กรอบงานอ้างอิง (Reference Frameworks) เป็นโครงสร้างซอฟต์แวร์พื้นฐานที่เตรียมไว้สำหรับผู้พัฒนาระบบ ไบออสบนตัวประมวลผลสัญญาณดิจิทัล (DSP/BIOS) เป็นเครื่องมือที่ใช้สำหรับจัดการระบบในระดับฮาร์ดแวร์ และมาตรฐานขั้นตอนวิธี

(Algorithm Standard) เป็นข้อกำหนดมาตรฐานสำหรับผู้พัฒนาขั้นตอนวิธีบนตัวประมวลผลสัญญาณดิจิทัล



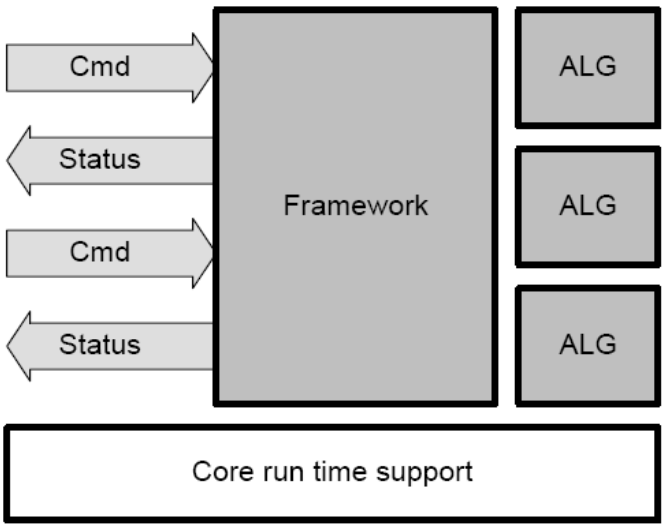
ภาพที่ 9 ส่วนประกอบของ eXpressDSP บนตัวประมวลผลสัญญาณดิจิทัลตระกูล TMS320
ที่มา: Blonstein (2002)

มาตรฐานขั้นตอนวิธีเป็นข้อกำหนดกลางที่ช่วยให้ผู้พัฒนาระบบและผู้พัฒนาขั้นตอนวิธีสามารถพัฒนาซอฟต์แวร์สำหรับงานประยุกต์ต่าง ๆ ร่วมกันได้ ซึ่งผู้ที่ได้รับประโยชน์จากมาตรฐานขั้นตอนวิธีนั้นมีอยู่ 2 กลุ่ม คือ

1. ผู้พัฒนาขั้นตอนวิธี (Algorithm Developers) กำหนดให้มีมาตรฐานในการพัฒนาขั้นตอนวิธีและเมื่อมีการเปลี่ยนฮาร์ดแวร์แพลตฟอร์มของระบบก็ไม่จำเป็นต้องพัฒนาขั้นตอนวิธีขึ้นมาใหม่ นอกจากนี้ยังสามารถทดสอบการทำงานในส่วนขั้นตอนวิธีได้โดยไม่ต้องรอให้การพัฒนาระบบทั้งหมดเสร็จสมบูรณ์

2. ผู้พัฒนาและรวบรวมระบบ (System Developers and Integrators) ได้ประโยชน์ในการลดเวลาในการรวบรวมระบบ เนื่องจากมีมาตรฐานที่แน่นอนในการติดต่อกับขั้นตอนวิธี ซึ่งผู้พัฒนาระบบไม่จำเป็นต้องรู้ว่าภายในขั้นตอนวิธีมีการทำงานอย่างไรบ้าง นอกจากนี้ผู้พัฒนาระบบยังสามารถเลือกใช้และเปรียบเทียบประสิทธิภาพการทำงานของขั้นตอนวิธีจากผู้พัฒนาขั้นตอนวิธีหลายคนได้

การเขียนซอฟต์แวร์ในลักษณะมาตรฐานขั้นตอนวิธีบนระบบประมวลผลสัญญาณดิจิทัล สามารถแบ่งส่วนต่าง ๆ ได้ดังภาพที่ 10



ภาพที่ 10 สถาปัตยกรรมซอฟต์แวร์สำหรับตัวประมวลผลสัญญาณดิจิทัล
ที่มา: Texas Instruments Inc. (2002 a)

จากภาพที่ 10 สถาปัตยกรรมซอฟต์แวร์แบ่งได้เป็น 3 ส่วน ได้แก่ ส่วนแรกคือส่วนสนับสนุนแก่นขณะทำงาน (Core Run-time Support) เป็นซอฟต์แวร์ที่ทำการติดต่อกับฮาร์ดแวร์ระบบ ส่วนที่ 2 คือส่วนขั้นตอนวิธี (Algorithm) หรือ ALG เป็นซอฟต์แวร์สำหรับประมวลผลข้อมูลต่าง ๆ ส่วนสุดท้ายคือกรอบงาน (Framework) เป็นตัวเชื่อมระหว่างซอฟต์แวร์สนับสนุนแก่นขณะทำงาน (Core Run-time Support) ซอฟต์แวร์ขั้นตอนวิธี (Algorithm) และซอฟต์แวร์ประยุกต์ (Application Software) ซึ่งจะคอยรับข้อมูลผ่านทางคำสั่ง (Command) และส่งค่าสถานะ (Status) กลับไปยังซอฟต์แวร์ประยุกต์

ในส่วนของมาตรฐานขั้นตอนวิธีนั้นได้มีการกำหนดกฎต่าง ๆ ในการเขียนโปรแกรมเพื่อให้เป็นไปตามมาตรฐานเดียวกัน โดยข้อกำหนดของมาตรฐานขั้นตอนวิธี ประกอบด้วย (Texas Instruments Inc., 2002 b)

1. กฎข้อบังคับ (Rules) เป็นข้อบังคับในการเขียนโปรแกรมซึ่งผู้พัฒนาขั้นตอนวิธีจะต้องทำตาม จำนวนข้อบังคับสำหรับผู้พัฒนาขั้นตอนวิธีมีอยู่ด้วยกัน 46 ข้อบังคับ ซึ่งลักษณะของข้อบังคับจะเป็นไปตามหลักการหรือความรู้สึกทั่วไป (Common Sense) ในการเขียนโปรแกรม ตัวอย่างเช่น ข้อบังคับที่ห้ามไม่ให้ผู้พัฒนาขั้นตอนวิธีเข้าใช้ฮาร์ดแวร์เองโดยตรง เพื่อป้องกันไม่ให้ระบบทำงานผิดพลาด เป็นต้น

2. ข้อแนะนำ (Guidelines) เป็นข้อชี้แนะสำหรับผู้พัฒนาขั้นตอนวิธีควรปฏิบัติซึ่งจะทำให้เพิ่มประสิทธิภาพในการทำงานของระบบ ตัวอย่างเช่น ข้อแนะนำให้หลีกเลี่ยงการใช้ข้อมูลแบบจุดลอย (Floating Point) เนื่องจากระบบบางฮาร์ดแวร์แพลตฟอร์มไม่สามารถใช้ข้อมูลชนิดนี้ได้ เป็นต้น ซึ่งผู้พัฒนาไม่จำเป็นต้องปฏิบัติตามก็ได้

การย้อนเข้าใหม่ (Reentrancy) เป็นหนึ่งในข้อบังคับสำคัญ ซึ่งซอฟต์แวร์ขั้นตอนวิธีจะต้องมีคุณสมบัตินี้ การย้อนเข้าใหม่คือลักษณะการเขียนโปรแกรมฟังก์ชันให้สามารถถูกเรียกใช้ได้หลายครั้งโดยไม่ทำให้ฟังก์ชันนั้นทำงานผิดพลาด เนื่องจากการเขียนซอฟต์แวร์บนตัวประมวลผลสัญญาณดิจิทัลนิยมเขียนแบบหลายงาน (Multitasking) ซึ่งแต่ละงานที่กำลังทำงานอาจมีการเรียกใช้ฟังก์ชันพร้อมกัน โดยการเขียนโปรแกรมให้เป็นแบบย้อนเข้าใหม่ได้นั้น ตัวฟังก์ชันที่ถูกเรียกจะต้องไม่มีการเก็บข้อมูลสถานะส่วนกลาง (Global State Information) ซึ่งตัวอย่างการเขียนโปรแกรมแบบย้อนเข้าใหม่สามารถแสดงได้ดังภาพที่ 11

```
int z0 = 0, z1 = 0;  /* previous input values */

void PRE_filter(int input[], int length)
{
   int i, tmp;

   for (i = 0; i < length; i++) {
      tmp = input[i] - z0 + (13 * z1 + 16) / 32;
      z1 = z0;
      z0 = input[i];
      input[i] = tmp;
   }
}
```

(ก)

```
void PRE_filter1(int input[], int length, int *z)
{
   int i, tmp;

   for (i = 0; i < length; i++) {
      tmp = input[i] - z[0] + (13 * z[1] + 16) / 32;
      z[1] = z[0];
      z[0] = input[i];
      input[i] = tmp;
   }
}
```

(ข)

ภาพที่ 11 ตัวอย่างการเขียนโปรแกรมแบบ (ก) ย้อนเข้าใหม่ไม่ได้ (ข) ย้อนเข้าใหม่ได้
ที่มา: Texas Instruments Inc. (2002 b)

จากภาพที่ 11(ก) โปรแกรมมีการใช้ตัวแปรส่วนกลาง (Global Variable) ในการเก็บค่าสถานะ ดังนั้นเมื่อมีการเรียกใช้ฟังก์ชันนี้ขณะที่ยังทำงานไม่เสร็จ เช่น ค่าสถานะ z0 และ z1 ในการเรียกครั้งแรกจะถูกนำไปใช้ในการเรียกครั้งที่สอง ทำให้การประมวลผลเกิดความผิดพลาด

จากภาพที่ 11(ข) เป็นการเขียนโปรแกรมในลักษณะย้อนเข้าใหม่ได้ ซึ่งตัวแปรที่ใช้เก็บค่าสถานะจะถูกส่งเข้ามาใช้ในฟังก์ชันด้วยโดยซอฟต์แวร์ที่เรียกใช้จะต้องเตรียมพื้นที่เก็บข้อมูลมาด้วย ดังนั้นการเรียกใช้โปรแกรมแต่ละครั้งจะมีพื้นที่เก็บค่าสถานะเป็นของตัวเอง ทำให้เป็นอิสระในการเรียกใช้งานและไม่ทำให้การทำงานผิดพลาด

ระเบียบการตั้งชื่อ (Naming Conventions) เป็นอีกหนึ่งข้อบังคับที่สำคัญของมาตรฐานขั้นตอนวิธี เนื่องจากเป็นสิ่งที่ใช้ในการต่อประสานระหว่างขั้นตอนวิธีและผู้พัฒนาระบบ ระเบียบการตั้งชื่อคือข้อบังคับในการตั้งชื่อตัวแปร ฟังก์ชัน หรือข้อมูลต่าง ๆ ที่ติดต่อกับโปรแกรมภายนอก เรียกว่าตัวชี้ภายนอก (External Identifiers) โดยมีโครงสร้างของชื่อเป็น PREFIX_Name ซึ่งแบ่งเป็น 2 ส่วน คือ ส่วนเติมหน้าสำหรับตัวชี้ภายนอก (Prefix of External Identifiers) และส่วนชื่อสำหรับตัวชี้ภายนอก (Name of External Identifiers)

ส่วนเติมหน้าสำหรับตัวชี้ภายนอกเป็นส่วนที่ใช้แยกความแตกต่างระหว่างขั้นตอนวิธีด้วยกัน ซึ่งจะขึ้นต้นด้วยชื่อของมอดูลขั้นตอนวิธีและตามด้วยชื่อของบริษัทหรือกลุ่มผู้พัฒนา โดยโครงสร้างจะมีลักษณะเป็น <MODULE>_<VENDOR>_ และชื่อที่ใช้จะต้องเป็นอักษรภาษาอังกฤษตัวพิมพ์ใหญ่ ตัวอย่างเช่น FIR_TI_ เป็นการบอกว่าขั้นตอนวิธีนี้พัฒนาจากบริษัท TI และเป็นมอดูลสำหรับการประมวลผลตัวกรองแบบ FIR นอกจากนี้ในกรณีที่ตัวชี้ภายนอกเป็นสิ่งที่มีเหมือนกัน (Common) ในแต่ละผู้พัฒนาจะสามารถเขียนส่วนเติมหน้าเฉพาะส่วนของมอดูลขั้นตอนวิธีได้ เช่น FIR_ เป็นต้น

ชื่อสำหรับตัวชี้ภายนอก เป็นชื่อที่ใช้สื่อความหมายในการทำงานของตัวชี้ภายนอก ซึ่งแบ่งได้เป็นหลายประเภทตามตารางที่ 2

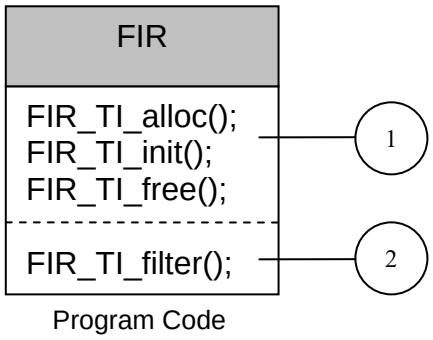
ตารางที่ 2 ตัวอย่างการตั้งชื่อในมาตรฐานขั้นตอนวิธีการ

Convention	Description	Example
Variables and functions	Variables and functions begin with lowercase (after the pre-fix).	FIR_apply()
Constants	Constants are all uppercase	G729_FRAMELEN
Types	Data types are in title case (after the prefix)	FIR_Handle
Structure fields	Structure fields begin with lowercase	buffer
macros	Macros follow the conven-tions of constants or functions as appropriate	FIR_create()

ที่มา: Texas Instruments Inc. (2002 b)

จากตารางที่ 2 ระเบียบการตั้งชื่อสำหรับตัวแปรและฟังก์ชัน (Variables and Functions) นั้น ตัวอักษรแรกหลังจากส่วนเติมหน้าจะต้องเป็นตัวอักษรภาษาอังกฤษตัวพิมพ์เล็ก ชื่อของค่าคงที่ (Constants) จะต้องเป็นตัวอักษรภาษาอังกฤษตัวพิมพ์ใหญ่ทั้งหมด ชื่อของชนิดข้อมูล (Data Types) จะขึ้นต้นด้วยตัวอักษรภาษาอังกฤษตัวพิมพ์ใหญ่ ส่วนเขตข้อมูลในโครงสร้างข้อมูล (Structure Fields) จะเริ่มต้นด้วยตัวอักษรภาษาอังกฤษตัวพิมพ์เล็ก และมาโคร (Macros) จะใช้ระเบียบการตั้งชื่อแบบเดียวกับตัวแปรและฟังก์ชัน

กลุ่มของโปรแกรมขั้นตอนวิธีที่อยู่ในซอฟต์แวร์ประยุกต์จะถูกเรียกว่ามอดูล (Module) ซึ่งจะถูกเก็บอยู่ในรูปอ็อบเจกต์มอดูล (Module Instance Objects) โดยภายในอ็อบเจกต์จะเก็บค่าที่อยู่ (Address) ที่ชี้ไปยังฟังก์ชันการทำงานต่าง ๆ ของขั้นตอนวิธีดังภาพที่ 12 ซึ่งอ็อบเจกต์มอดูลจะถูกเก็บไว้ในหน่วยความจำโปรแกรม (Program) ก่อนจะถูกนำมาใช้ในการสร้างอ็อบเจกต์ของขั้นตอนวิธีต่อไป (Texas Instruments Inc., 2002 a)

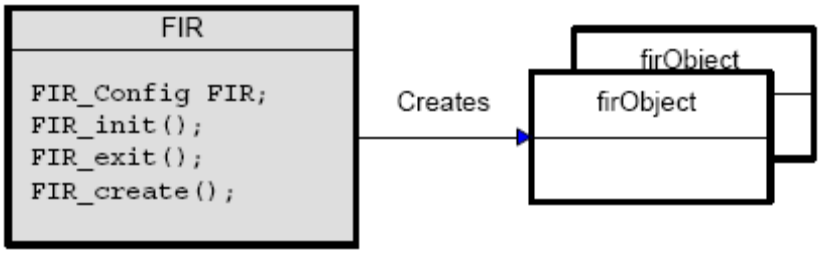


ภาพที่ 12 ตัวอย่างของอ็อบเจกต์มอดูล

จากภาพที่ 12 เป็นภาพแสดงตัวอย่างของอ็อบเจกต์มอดูล ซึ่งในอ็อบเจกต์จะเก็บค่าของฟังก์ชันขั้นตอนวิธี โดยจะแบ่งได้ 2 กลุ่ม กลุ่มแรกเป็นฟังก์ชันที่ผู้พัฒนาขั้นตอนวิธีได้เขียนขึ้นและต้องเป็นไปตามข้อกำหนดของมาตรฐานขั้นตอนวิธีด้วย โดยมีหน้าที่เกี่ยวกับการกำหนดค่าเริ่มต้นหรือการสิ้นสุดการทำงานให้กับมอดูลขั้นตอนวิธี ได้แก่ FIR_TI_alloc() เป็นฟังก์ชันที่ใช้จองหน่วยความจำสำหรับขั้นตอนวิธี กลุ่มที่ 2 เป็นฟังก์ชันพิเศษหรือฟังก์ชันเพิ่มเติม (Extended Function) เขียนโดยผู้พัฒนาขั้นตอนวิธี ทำหน้าที่ประมวลผลตามขั้นตอนวิธีที่เขียนขึ้น เช่น FIR_TI_filter() เป็นฟังก์ชันกรองสัญญาณที่มาจากข้อมูลขาเข้า

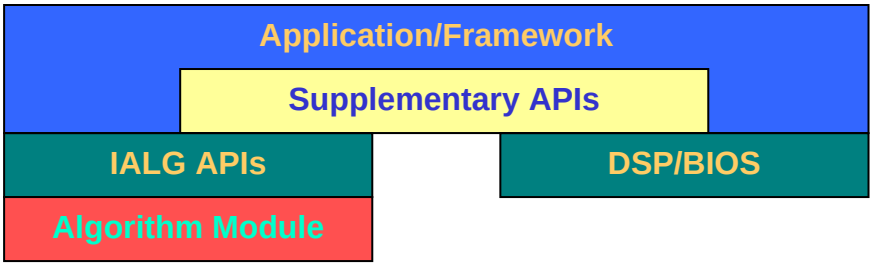
ก่อนที่จะนำมอดูลขั้นตอนวิธีไปใช้ในโปรแกรมประยุกต์ตัวมอดูลจะต้องถูกสร้างและกำหนดค่าเริ่มต้น (Initialized) ขึ้นมาก่อน และเมื่อโปรแกรมประยุกต์ใช้งานขั้นตอนวิธีเสร็จแล้วก็ต้องกำหนดการสิ้นสุด (Finalization) ให้มอดูลเช่นเดียวกัน วิธีการกำหนดค่าเริ่มต้นเป็นการกำหนดค่าให้กับข้อมูลส่วนกลาง (Global Data) ที่จะใช้ในโปรแกรมขั้นตอนวิธี ส่วนการกำหนดการสิ้นสุดเป็นการบอกให้ระบบรับรู้ว่ายกเลิกการใช้งานขั้นตอนวิธีนั้นแล้ว ซึ่งนิยมใช้ในการหาข้อผิดพลาด (Debug) ระหว่างการทำงานของโปรแกรม ในโปรแกรมแบบใช้งานจริงไม่จำเป็นต้องใช้วิธีการกำหนดการสิ้นสุด โดยเขียนฟังก์ชันสิ้นสุดให้เป็นลักษณะฟังก์ชันว่างเปล่า (Empty Function)

จากอ็อบเจกต์มอดูลที่กล่าวมาซึ่งเป็นส่วนที่อยู่ในหน่วยความจำโปรแกรม (Program Memory) เมื่อมีการเรียกฟังก์ชันสร้าง (Create Function) ซอฟต์แวร์ประยุกต์จะทำการสร้างอ็อบเจกต์ขั้นตอนวิธีขึ้นในหน่วยความจำข้อมูล (Data Memory) ดังภาพที่ 13 ซึ่งมีลักษณะเหมือนกับอ็อบเจกต์ นอกจากนั้นการสร้างอ็อบเจกต์ขั้นตอนวิธีสามารถสร้างได้หลายครั้ง เนื่องจากเป็นการทำสำเนา (Copy) ข้อมูลจากหน่วยความจำโปรแกรมไปยังหน่วยความจำข้อมูล ทำให้โปรแกรมประยุกต์สามารถเรียกใช้ขั้นตอนวิธีจากหลายอ็อบเจกต์ได้พร้อมกัน



ภาพที่ 13 การสร้างอ็อบเจกต์ขั้นตอนวิธีจากอ็อบเจกต์มอดูล
ที่มา: Texas Instruments Inc. (2002 a)

ในมาตรฐานขั้นตอนวิธีได้เตรียมโครงสร้างข้อมูลสำหรับอ็อบเจกต์มอดูลและวิธีการเรียกใช้ผ่านทางส่วนต่อประสานขั้นตอนวิธีทันด่วน (Instance Algorithm Interface) หรือส่วนต่อประสาน IALG (IALG Interface) ไว้ โดยลักษณะซอฟต์แวร์ที่มีการเขียนโดยใช้มาตรฐานขั้นตอนวิธีจะมีโครงสร้างดังภาพที่ 14



ภาพที่ 14 โครงสร้างซอฟต์แวร์สำหรับมาตรฐานขั้นตอนวิธี

จากภาพที่ 14 ส่วนต่อประสานขั้นตอนวิธีทันด่วน (Instance Algorithm Application Programming Interface) หรือ IALG APIs เป็นส่วนมาตรฐานที่ใช้เชื่อมต่อระหว่างโปรแกรมประยุกต์หรือกรอบงาน (Application/Framework) กับมอดูลขั้นตอนวิธี (Algorithm Module) ซึ่งโปรแกรมประยุกต์สามารถติดต่อกับขั้นตอนวิธีผ่านส่วนต่อประสานขั้นตอนวิธีทันด่วนและติดต่อกับฮาร์ดแวร์ระบบผ่านทางไบออสบนตัวประมวลผลสัญญาณดิจิทัล (DSP/BIOS) ในขณะที่ส่วนต่อประสานเพิ่มเติม (Supplementary APIs) เป็นการจัดกลุ่มของส่วนต่อประสานขั้นตอนวิธีทันด่วนที่ทำงานต่อเนื่องกันและส่วนจัดการทรัพยากรระบบ เช่น หน่วยความจำเข้าด้วยกัน เพื่อให้ผู้พัฒนาโปรแกรมประยุกต์เรียกใช้ได้ง่าย

จากภาพที่ 14 ส่วนต่อประสานขั้นตอนวิธีทันด่วน (IALG APIs) ไบออสบนตัวประมวลผลสัญญาณดิจิทัล (DSP/BIOS) และส่วนต่อประสานเพิ่มเติม (Supplementary APIs) เป็นส่วนที่บริษัท Texas Instruments ได้กำหนดและเตรียมไว้ให้แล้ว ส่วนของมอดูลขั้นตอนวิธี (Algorithm Module) เป็นส่วนที่เขียนโดยผู้พัฒนาขั้นตอนวิธี (Algorithm Developers) และในส่วนงานประยุกต์ (Application) เป็นส่วนที่เขียนขึ้นโดยผู้พัฒนาโปรแกรมประยุกต์ (Application Developers)

ส่วนต่อประสานขั้นตอนวิธีทันด่วนได้เตรียมมาตรฐานโครงสร้างของอ็อบเจกต์มอดูล ซึ่งมีการประกาศฟังก์ชันมาตรฐานสำหรับติดต่อกับขั้นตอนวิธีดังภาพที่ 15

```
typedef struct IALG_Fxns {
    Void *implementationId;
    Void (*algActivate)(IALG_Handle);
    Int (*algAlloc)(const IALG_Params *, struct IALG_Fxns **, IALG_MemRec *);
    Int (*algControl)(IALG_Handle, IALG_Cmd, IALG_Status *);
    Void (*algDeactivate)(IALG_Handle);
    Int (*algFree)(IALG_Handle, IALG_MemRec *);
    Int (*algInit)(IALG_Handle, const IALG_MemRec *, IALG_Handle, const IALG_Params *);
    Void (*algMoved)(IALG_Handle, const IALG_MemRec *, IALG_Handle, const IALG_Params *);
    Int (*algNumAlloc)(Void);
} IALG_Fxns;
```

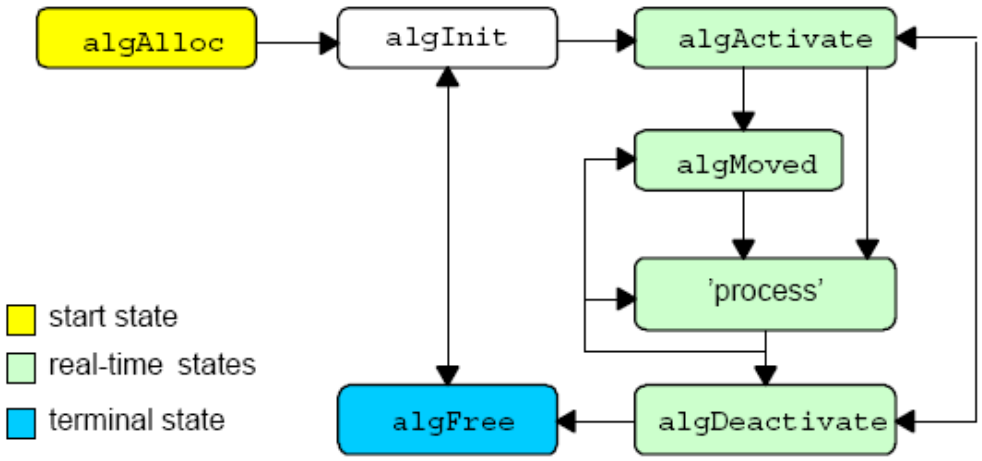
ภาพที่ 15 อ็อบเจกต์มอดูลในส่วนต่อประสานขั้นตอนวิธีทันด่วน
ที่มา: Texas Instruments Inc. (2002 a)

จากภาพที่ 15 ลักษณะอ็อบเจกต์มอดูลจะประกาศเป็นโครงสร้างข้อมูล ภายในประกอบด้วยตัวชี้อ็อบเจกต์ขั้นตอนวิธี และตัวชี้ไปยังฟังก์ชันต่าง ๆ แบ่งเป็น 2 กลุ่มฟังก์ชัน ได้แก่ กลุ่มที่จำเป็นต้องเขียนตามมาตรฐาน (Required Implementation) และกลุ่มทางเลือก (Option)

ฟังก์ชันกลุ่มที่จำเป็นต้องเขียนตามมาตรฐานเป็นกลุ่มของฟังก์ชันที่ผู้พัฒนาขั้นตอนวิธีต้องเขียนขึ้นตามมาตรฐานขั้นตอนวิธี กลุ่มของฟังก์ชันเหล่านี้จะทำหน้าที่เกี่ยวกับการเรียกขอจัดการกับหน่วยความจำไปยังโปรแกรมประยุกต์ โดยฟังก์ชันเหล่านี้คือ “algAlloc” เป็นฟังก์ชันที่บอกให้ซอฟต์แวร์ระบบหรือซอฟต์แวร์โปรแกรมประยุกต์รับรู้ว่ามอดูลขั้นตอนวิธีต้องการใช้บล็อกของหน่วยความจำ (Memory Block) หรือจุดตั้งระยะหน่วยความจำ (Memory Tab) จำนวนเท่าไร และมีขนาดเท่าไร ขณะที่ “algInit” เป็นฟังก์ชันที่กำหนดค่าเริ่มต้นให้กับอ็อบเจกต์ขั้นตอนวิธีโดยใช้ตัวชี้ข้อมูลที่ต้องการชี้ไปยังบล็อกของหน่วยความจำที่ระบบได้จัดสรรไว้ให้ สุดท้ายฟังก์ชัน “algFree” เป็นฟังก์ชันขอยกเลิกการจัดสรรหน่วยความจำที่เคยจัดสรรไว้ในฟังก์ชัน “algAlloc” ให้กับขั้นตอนวิธี ซึ่งใช้ในกรณีระบบไม่ได้ใช้ขั้นตอนวิธีนี้ในการทำงาน (Torud, 2002)

ฟังก์ชันกลุ่มทางเลือก ได้แก่ “algActivate,” “algDeactivate,” และ “algMove” เป็นกลุ่มฟังก์ชันที่ใช้สำหรับจัดการหน่วยความจำแบบเปลี่ยนแปลงได้ (Dynamic Memory Management) ใช้สำหรับการออกแบบซอฟต์แวร์ให้สามารถเคลื่อนย้ายขั้นตอนวิธีในหน่วยความจำได้ นอกจากนี้ยังมีฟังก์ชัน “algControl” และ “algNumAlloc” เป็นฟังก์ชันที่ใช้ในการรับส่งสถานะและหาจำนวนบล็อกของหน่วยความจำตามลำดับ

จากฟังก์ชันที่กล่าวมาในข้างต้น การเรียกใช้งานฟังก์ชันต่าง ๆ จะต้องมีลำดับการเรียกใช้ ดังภาพที่ 16



ภาพที่ 16 ลำดับการเรียกใช้ฟังก์ชันในส่วนต่อประสานขั้นตอนวิธีทันด่วน
ที่มา: Texas Instruments Inc. (2002 a)

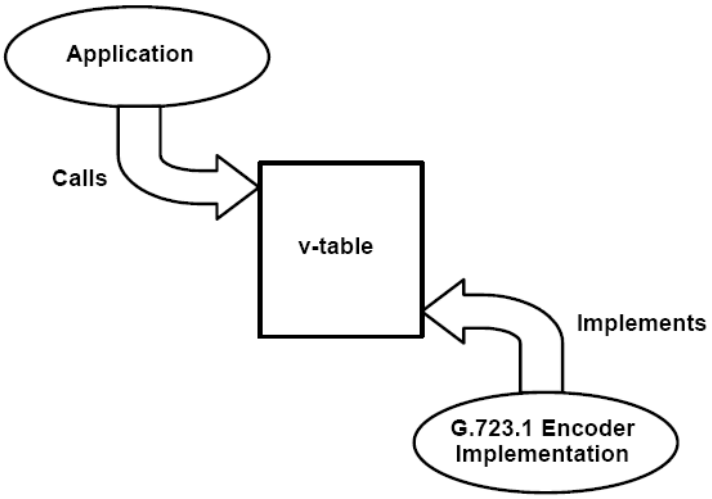
จากภาพที่ 16 ในการเรียกใช้งานขั้นตอนวิธีทุกครั้งสำหรับผู้พัฒนาโปรแกรมประยุกต์จะต้องเรียกฟังก์ชัน “algAlloc” ก่อนเพื่อหาจำนวนหน่วยความจำที่ต้องการใช้ จากนั้นจะเรียกฟังก์ชัน “algInit” เพื่อกำหนดค่าเริ่มต้น ต่อมาโปรแกรมประยุกต์สามารถเรียกฟังก์ชัน “algActivate” เพื่อเตรียมทำการประมวลผลหรือฟังก์ชัน “algFree” เพื่อจบการทำงานของขั้นตอนวิธี หลังจากเรียกฟังก์ชัน “algActivate” แล้วจะสามารถย้ายอ็อบเจกต์โดยคำสั่งฟังก์ชัน “algMoved” หรือเรียกประมวลผล (Process) ขั้นตอนวิธีได้ เมื่อเรียกฟังก์ชัน “algActivate” แล้วจะไม่สามารถเรียกฟังก์ชัน “algFree” ได้ต้องเรียกฟังก์ชัน “algDeactivate” ก่อน

ส่วนประมวลผล (Process) เป็นส่วนที่ผู้พัฒนาขั้นตอนวิธีจะต้องเตรียมฟังก์ชันเพิ่มเติม (Extended Function) ใช้ในการประมวลผลต่าง ๆ ลักษณะการประกาศฟังก์ชันเพิ่มเติมจะเป็นรูปแบบการสืบทอด (Inherits) มาจากอ็อบเจกต์มอดูลของส่วนต่อประสานขั้นตอนวิธีทันด่วน ดังภาพที่ 17

```
typedef struct IFIR_Fxns {
    IALG_Fxns   ialg;
    Void        (*filter)(IFIR_Handle handle, Int in[], Int out[]);
} IFIR_Fxns;
```

ภาพที่ 17 อ็อบเจกต์สำหรับฟังก์ชันเพิ่มเติมในส่วนต่อประสานขั้นตอนวิธีทันด่วน
ที่มา: Texas Instruments Inc. (2002 a)

ผู้พัฒนาโปรแกรมขั้นตอนวิธีสามารถจัดการเชื่อมโยงฟังก์ชันที่เขียนขึ้นมากับมาตรฐานขั้นตอนวิธีได้โดยการเรียกใช้ตารางของตัวชี้ฟังก์ชัน (Table of Function Pointers) หรือตารางวี (V-table) ดังภาพที่ 18 ซึ่งเป็นตัวอย่างตารางวีที่ใช้เชื่อมต่อระหว่างผู้พัฒนาโปรแกรมประยุกต์และผู้พัฒนาขั้นตอนวิธีสำหรับการเข้ารหัสสัญญาณเสียงแบบจี 723.1 (G.723.1 Encoder) ทุก ๆ มอดูลขั้นตอนวิธีจะต้องสร้างตารางวีเตรียมไว้ให้ผู้พัฒนาโปรแกรมประยุกต์ ซึ่งลักษณะการทำงานจะเหมือนเป็นตัวกลางระหว่างผู้พัฒนาโปรแกรมประยุกต์และผู้พัฒนาขั้นตอนวิธี (Torud, 2002)



ภาพที่ 18 ตัวอย่างการใช้งานตารางวี
ที่มา: Torud (2002)

จากตัวอย่างในภาพที่ 18 ในส่วนของตารางวีสามารถเขียนโปรแกรมได้ดังภาพที่ 19

```
#define IALGFXNS \
   &G723ENC_TI_IALG,              /* module ID */      \
   NULL,                          /* activate */       \
   G723ENC_TI_algAlloc,           /* alloc */          \
   NULL,                          /* control */        \
   NULL,                          /* deactivate */     \
   G723ENC_TI_algFree,            /* free */           \
   G723ENC_TI_algInit,            /* init */           \
   NULL,                          /* moved */          \
   NULL                           /* numAlloc */       \
IG723ENC_Fxns G723ENC_TI_IG723ENC ={
   IALGFXNS,                      /* IALG functions */
   G723ENC_TI_control,
   g723ENC_TI_encode
} G723ENC_TI_IG723ENC;
```

ภาพที่ 19 ตัวอย่างการเขียนโปรแกรมตารางวี
ที่มา: Torud (2002)

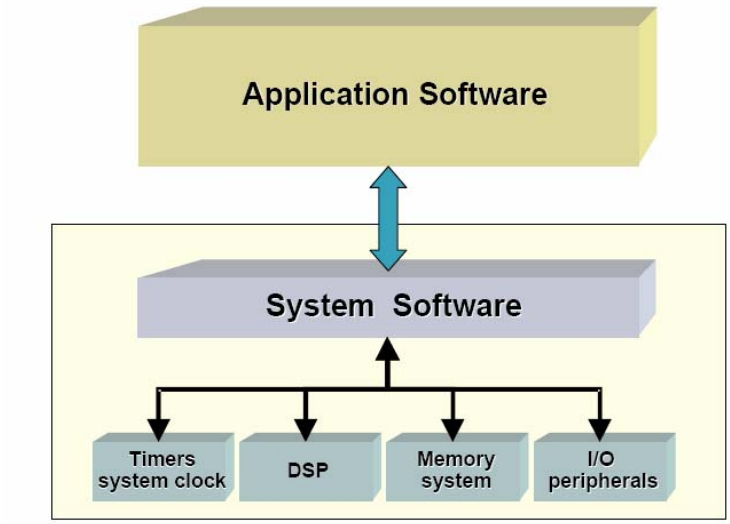
ตัวอย่างจากภาพที่ 19 ค่าคงที่ IALGFXNS จะทำการเชื่อมต่อฟังก์ชันที่ผู้พัฒนาขั้นตอนวิธีที่เขียนขึ้นมาเข้ากับมาตรฐานขั้นตอนวิธีที่กล่าวในข้างต้น โดยฟังก์ชันทางเลือกที่ไม่ได้เขียนขึ้นมาให้ตั้งค่าที่ตัวชี้เป็นค่าว่าง (Null) ส่วนฟังก์ชันเพิ่มเติมจะประกาศในลักษณะการสืบทอดโครงสร้างข้อมูล ซึ่งข้อมูล G723ENC_TI_IG723ENC จะส่วนที่สืบทอดมาจาก IALGFXNS ซึ่งภายในจะประกอบด้วยฟังก์ชันเพิ่มเติม G723ENC_TI_control และ g723ENC_TI_encode เป็นต้น

การพัฒนาซอฟต์แวร์ทั้งหมดบนระบบประมวลผลสัญญาณได้มีการกำหนดมาตรฐานขั้นตอนวิธีเพื่อเป็นมาตรฐานในการเขียนโปรแกรมประยุกต์และโปรแกรมขั้นตอนวิธี ทำให้การพัฒนาโปรแกรมทั้งสองส่วนเป็นอิสระจากกันและลดเวลาในการพัฒนาระบบลง เนื่องจากสามารถพัฒนาโปรแกรมไปพร้อม ๆ กันหรือขนานกันได้

ไบออสบนตัวประมวลผลสัญญาณดิจิทัล

ในการเขียนโปรแกรมในเวลาจริง (Real-time Software Application) บนตัวประมวลผลสัญญาณดิจิทัล (Digital Signal Processor) ของบริษัท Taxes Instruments หรือ TI นั้น ทางบริษัท TI ได้เตรียมเคอร์เนล (Kernel) ไบออสบนตัวประมวลผลสัญญาณดิจิทัล ซึ่งเป็นเครื่องมือที่ผู้พัฒนาโปรแกรมบนตัวประมวลผลสัญญาณดิจิทัลใช้จัดการโครงสร้างการทำงานของโปรแกรมบนตัวประมวลผลสัญญาณดิจิทัลและจัดการทรัพยากรต่าง ๆ ภายในระบบ นอกจากนั้นยังสามารถทำการวิเคราะห์ประสิทธิภาพของระบบได้อีกด้วย

ซอฟต์แวร์บนตัวประมวลผลสัญญาณดิจิทัลนั้นจะแบ่งตามหน้าที่การทำงานออกได้เป็น 2 ส่วน ดังภาพที่ 20 ซึ่งได้แก่ ซอฟต์แวร์ระบบ (System Software) และซอฟต์แวร์ประยุกต์ (Application Software) โดยส่วนของซอฟต์แวร์ระบบจะทำหน้าที่จัดการทรัพยากรต่าง ๆ ของระบบ รวมทั้งอุปกรณ์รอบข้าง (Peripherals) ในระบบด้วย ส่วนซอฟต์แวร์ประยุกต์นั้นเป็นส่วนที่ทำงานทางด้านประมวลผลต่าง ๆ



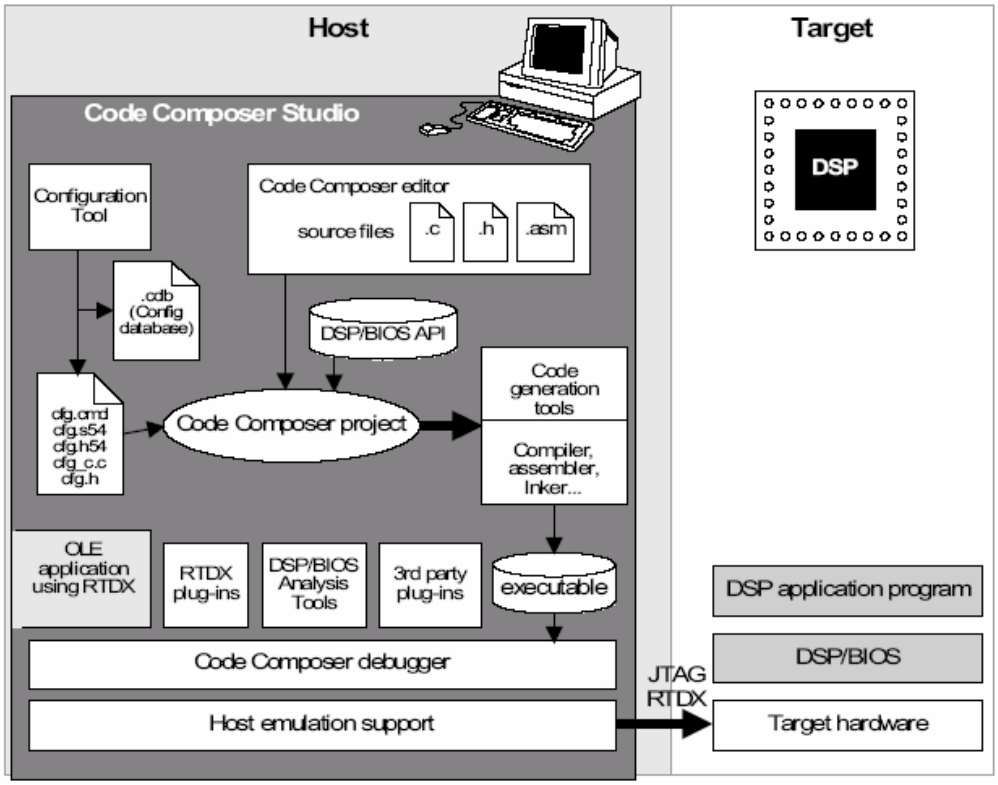
ภาพที่ 20 ส่วนประกอบของซอฟต์แวร์ระบบฝังตัว (Embedded System Software Components)
ที่มา: Texas Instruments Inc. (2001)

งานต่าง ๆ บนตัวประมวลผลสัญญาณดิจิทัลนั้นต้องการซอฟต์แวร์ระบบเพื่อช่วยในการจัดการกับทรัพยากรในระบบ รวมทั้งทำหน้าที่กำหนดค่าเริ่มต้น (Initializaion) และควบคุมส่วนของฮาร์ดแวร์ในระบบด้วย ซึ่งอาจกล่าวได้ว่าซอฟต์แวร์ระบบเป็นตัวกลางในการเข้าถึงระหว่างซอฟต์แวร์ประยุกต์และทรัพยากรระบบ ดังนั้นเมื่อมีการนำซอฟต์แวร์ประยุกต์ไปใช้กับฮาร์ดแวร์แพลตฟอร์ม (Platform) อื่น ๆ จะต้องคำนึงถึงคุณลักษณะของระบบนั้น ๆ ด้วย

ในระบบทั่วไปซอฟต์แวร์ระบบนั้นประกอบด้วยฟังก์ชันการกำหนดค่าเริ่มต้นฮาร์ดแวร์ การใช้งานอุปกรณ์รอบข้างและงานบริการการขัดจังหวะของฮาร์ดแวร์ (Hardware Interrupt Service) นอกจากนี้ยังทำการจัดลำดับ (Scheduling) การทำงานของงานโยงใยย่อย (Thread) หรือเรียกสั้น ๆ ว่า งานย่อยต่าง ๆ แบบเวลาจริง (Real-time) และควบคุมการใช้ทรัพยากรให้สามารถใช้งานได้ในลักษณะพร้อมกัน (Concurrent) การจัดการและควบคุมระบบของซอฟต์แวร์ระบบนั้นสามารถเขียนโปรแกรมโดยใช้ไบออสบนตัวประมวลผลสัญญาณดิจิทัล ซึ่งจะมีเครื่องมือในการจัดการควบคุมส่วนต่าง ๆ ในฮาร์ดแวร์ตัวประมวลผลสัญญาณดิจิทัล โดยมีข้อดีต่าง ๆ ดังนี้

- สามารถช่วยจัดการประมวลผลให้เป็นหลายงานโยงใยย่อย (Multi-thread) ได้ง่าย
- มีฟังก์ชันมาตรฐานในการติดต่อกับอุปกรณ์ฮาร์ดแวร์
- สามารถกำหนดการใช้งานทรัพยากรระบบได้
- สามารถทำงานและวิเคราะห์การทำงานในเวลาจริงได้
- ง่ายต่อการย้ายโปรแกรมไปทำงานที่ตัวประมวลผลดิจิทัลแพลตฟอร์มอื่น เนื่องจากมีการใช้ไบออสบนตัวประมวลผลสัญญาณดิจิทัลที่อยู่ในมาตรฐานเดียวกัน

ไบออสบนตัวประมวลผลสัญญาณดิจิทัลเป็นเครื่องมือที่มาพร้อมกับชุดเครื่องมือรวมสำหรับการเขียนโปรแกรม (Code Composer Studio) ของบริษัท TI โดยที่ชุดเครื่องมือรวมสำหรับการเขียนโปรแกรมนั้น เป็นโปรแกรมที่ใช้ในการเขียนและสร้างซอฟต์แวร์สำหรับทำงานบนตัวประมวลผลสัญญาณดิจิทัลและแก้จุดบกพร่อง (Debug) ต่าง ๆ ของซอฟต์แวร์ ซึ่งส่วนประกอบของไบออสบนตัวประมวลผลสัญญาณดิจิทัลสามารถแสดงได้ดังภาพที่ 21



ภาพที่ 21 ส่วนประกอบต่าง ๆ ของไบออสบนตัวประมวลผลสัญญาณดิจิทัล
ที่มา: Texas Instruments Inc. (2002 c)

จากภาพที่ 21 การพัฒนาซอฟต์แวร์บนตัวประมวลผลสัญญาณดิจิทัลจะแบ่งเป็น 2 ส่วน คือ เครื่องแม่ข่าย (Host) และอุปกรณ์เป้าหมาย (Target) โดยที่แม่ข่ายคือชุดเครื่องมือรวมสำหรับการเขียนโปรแกรมบนเครื่องคอมพิวเตอร์ ซึ่งผู้พัฒนาสามารถเขียนโปรแกรมด้วยภาษาซี (C) หรือซีพลัสพลัส (C++) หรือแอสเซมบลี (Assembly) ได้ และในชุดเครื่องมือรวมสำหรับการเขียนโปรแกรมจะประกอบด้วยส่วนต่อประสานโปรแกรมประยุกต์สำหรับไบออสบนตัวประมวลผลสัญญาณดิจิทัล (DSP/BIOS API) ซึ่งเป็นฟังก์ชันในการติดต่อหรือควบคุมไบออสบนตัวประมวลผลสัญญาณดิจิทัล เครื่องมือโครงแบบ (Configuration tool) เป็นเครื่องมือที่ใช้ในการกำหนดคุณลักษณะของอ็อบเจกต์ (Objects) ต่าง ๆ ในโปรแกรม ตัวแปลโปรแกรม (Compiler) ตัวแปลโปรแกรมภาษาแอสเซมบลี (Assembler) และโปรแกรมเชื่อมโยง (Linker) เครื่องมือต่าง ๆ เหล่านี้ใช้ในการเขียนโปรแกรมภาษาซี หรือแอสเซมบลี และเชื่อมโยงไฟล์ต่าง ๆ ตามลำดับ เมื่อได้ไฟล์ที่สามารถทำงานบนตัวประมวลผลได้แล้ว ไฟล์นั้นจะถูกส่งไปยังอุปกรณ์เป้าหมายหรือตัวประมวลผลสัญญาณดิจิทัลผ่านทางเจแทค (JTAG) หรืออาร์ทีดีเอ็กซ์ (RTDX) และในขณะที่ตัว

ประมวลผลกำลังทำงานอยู่ ชุดเครื่องมือรวมสำหรับการเขียนโปรแกรมได้เตรียมเครื่องมือวิเคราะห์ไบออสบนตัวประมวลผลสัญญาณดิจิทัล (DSP/BIOS Analysis Tool) ในการวิเคราะห์ระบบขณะทำงานในเวลาจริงได้

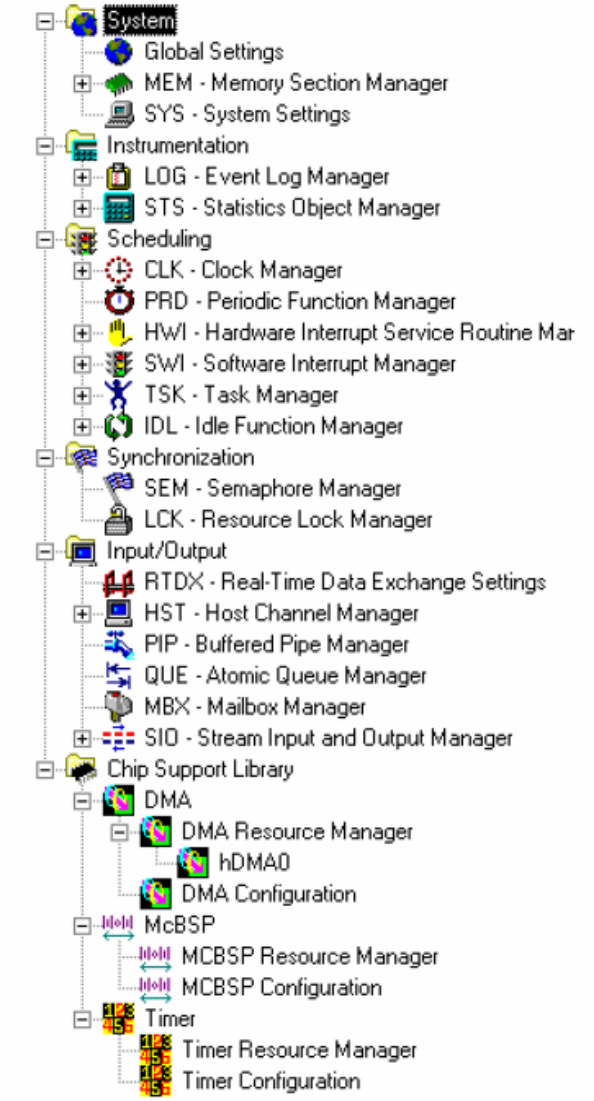
ส่วนต่อประสานโปรแกรมประยุกต์สำหรับไบออสบนตัวประมวลผลสัญญาณดิจิทัลนั้นจะแบ่งเป็นมอดูลต่าง ๆ ตามหน้าที่การทำงานที่ได้กำหนดไว้ ซึ่งสามารถแสดงได้ดังตารางที่ 3 โดยที่โปรแกรมประยุกต์ (Application Program) จะใช้งานมอดูลต่าง ๆ ผ่านส่วนต่อประสานโปรแกรมประยุกต์ (Application Programming Interface) ซึ่งมีลักษณะการเขียนแบบเดียวกับภาษาซี (C-callable Interface) ในขณะที่ภายในมอดูลอาจจะเขียนขึ้นโดยใช้ภาษาแอสเซมบลี (Assembly) เพื่อให้สะดวกในการเรียกใช้และมีความรวดเร็วในการทำงาน

ตารางที่ 3 มอดูลต่าง ๆ ของไบออสบนตัวประมวลผลสัญญาณดิจิทัลของบริษัท TI

Module	Description	Module	Description
ATM	Atomic functions written in assembly language	MEM	Memory segment manager
Cxx	Target-specific functions, platform dependent	PIP	Buffered pipe manager
CLK	Clock manager	PRD	Periodic function manager
CSL	Chip Support Library	QUE	Atomic queue manager
DEV	Device driver interface	RTDX	Real-time data exchange settings
GBL	Global setting manager	SEM	Semaphore manager
HOOK	Hook function manager	SIO	Stream I/O manager
HST	Host channel manager	STS	Statistics object manager
HWI	Hardware interrupt manager	SWI	Software interrupt manager
IDL	Idle function manager	SYS	System services manager
LCK	Resource lock manager	TRC	Trace manager
LOG	Event log manager	TSK	Multitasking manage
MBX	Mailbox manager		

ที่มา: Texas Instruments Inc. (2002 c)

ในส่วนของเครื่องมือโครงแบบ (Configuration Tool) เป็นเครื่องมือที่เตรียมไว้สำหรับให้ผู้พัฒนาโปรแกรมสามารถกำหนดค่าคุณลักษณะและการทำงานให้กับมอดูลต่าง ๆ ของไบออสบนตัวประมวลผลสัญญาณดิจิทัล โดยลักษณะการติดต่อนั้นจะคล้ายกับการใช้งานโปรแกรมค้นหาหน้าต่าง (Window Explorer) ของระบบปฏิบัติการวินโดว์ (WINDOWS) บนคอมพิวเตอร์ ดังภาพที่ 22 ซึ่งประกอบด้วยส่วนของการตั้งค่าส่วนกลาง (Global Settings) และมอดูลต่าง ๆ ของไบออสบนตัวประมวลผลสัญญาณดิจิทัล



ภาพที่ 22 การติดต่อกับไบออสบนตัวประมวลผลสัญญาณดิจิทัลโดยใช้เครื่องมือโครงแบบ

ที่มา: Texas Instruments Inc. (2002 c)

เครื่องมือวิเคราะห์ไบออสบนตัวประมวลผลสัญญาณดิจิทัลเป็นเครื่องมือที่ใช้สำหรับการวิเคราะห์การทำงานของระบบในเวลาจริง โดยการดูค่าพารามิเตอร์จากเครื่องมือแบบต่าง ๆ ได้แก่ กราฟภาระซีพียู (CPU Load Graph) จะแสดงกราฟของการใช้งานหน่วยประมวลผลเทียบกับช่วงที่ไม่ใช้งาน กราฟการปฏิบัติงาน (Execution Graph) จะแสดงกราฟการทำงานของมอดูลของงานย่อย (Thread) ต่างๆในระบบ บันทึกข้อความ (Message Log) ใช้แสดงข้อความทางหน้าจอ ภาพแสดงข้อมูลสถิติ (Statistics View) จะแสดงการนับจำนวนครั้งในการเรียกใช้มอดูลต่าง ๆ ของระบบ และภาพแสดงเคอร์เนลหรืออ็อบเจกต์ (Kernel/Object View) ใช้แสดงการทำงานและสถานะของงานย่อยต่าง ๆ ในระบบ เป็นต้น

เคอร์เนลไบออสบนตัวประมวลผลสัญญาณดิจิทัลได้เตรียมส่วนประกอบเคอร์เนลหรือบริการระบบในเวลาจริง (Real-time System Services) สำหรับการเขียนโปรแกรมบนตัวประมวลผลสัญญาณดิจิทัลตระกูล TMS320C6000 ซึ่งบริการและส่วนประกอบเหล่านั้นสามารถใช้ได้ผ่านทางเครื่องมือโครงแบบและคลังโปรแกรมขณะทำงาน (Run-time Library) ในเครื่องมือโครงแบบจะมีมอดูลของเคอร์เนลไบออสบนตัวประมวลผลสัญญาณดิจิทัล ซึ่งสามารถจัดกลุ่มตามประเภทของการทำงานได้เป็น 6 กลุ่ม ซึ่งได้แสดงไว้ในภาพที่ 22 ได้แก่ (Dart, 2001)

1. กลุ่มให้บริการระบบ (System Service) เป็นส่วนที่ใช้ในการกำหนดคุณลักษณะโดยรวมของระบบ โดยมีมอดูล MEM จัดการหน่วยความจำของระบบ ซึ่งใช้ในการจองหน่วยความจำทั้งแบบคงที่ (Static) และเปลี่ยนแปลงได้ (Dynamic) นอกจากนี้จะมีมอดูลสำหรับจัดการกับความผิดพลาดต่าง ๆ ของระบบ ซึ่งได้แก่ มอดูล SYS

2. กลุ่มเครื่องมือ (Instrumentation) หรือการวิเคราะห์เวลาจริง (Real-time Analysis, RTA) เป็นส่วนที่ผู้พัฒนาโปรแกรมสามารถใช้ในการติดต่อกับตัวประมวลผลสัญญาณดิจิทัลในการโต้ตอบ (Interaction) หรือตรวจสอบความผิดพลาด (Diagnostics) ในเวลาจริง โดยที่มอดูลประเภทนี้ ได้แก่ LOG STS และ TRC

3. กลุ่มจัดกำหนดการ (Scheduling) เป็นมอดูลสำหรับจัดการทำงานของงานย่อย (Thread) ซึ่งจะมีการแบ่งประเภทของงานย่อย ได้เป็น 4 ประเภท ได้แก่ HWI SWI TSK และ IDL แต่ละประเภทของงานย่อยจะมีขีดจำกัดและความสามารถที่แตกต่างกัน นอกจากนั้นไบออสบนตัวประมวลผลสัญญาณดิจิทัลยังสนับสนุนให้แต่ละงานย่อยมีความสามารถยึดครองการทำงาน

(Preemptive) ตามระดับความสำคัญ (Priority) ซึ่งทำให้ผู้พัฒนาสามารถออกแบบโปรแกรมที่ทำงานพร้อม ๆ กันแบบหลายงานย่อย (Multi-thread) ได้

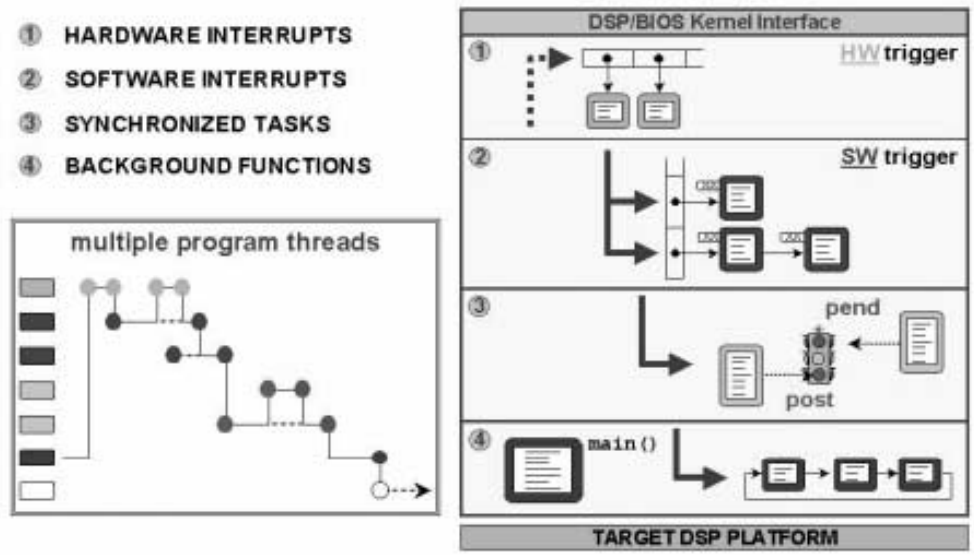
นอกจากการทำงานแบบหลายงานย่อยแล้วไบออสบนตัวประมวลผลสัญญาณดิจิทัลยังเตรียมการทำงานแบบไม่มีการยึดครองการทำงาน (Non-preemptive) ด้วย ได้แก่ ฟังก์ชันแบบ (Atomic) ซึ่งจะทำงานโดยไม่สนใจการขัดจังหวะของระบบ งานทั่วไปเกี่ยวกับการ AND OR INC DEC SET และ CLEAR ค่าของตัวแปรจะมีมอดูล ATM ทำหน้าที่จัดการงานเหล่านี้

4. กลุ่มการประสานเวลา (Synchronization) หรือเรียกว่าการสื่อสารระหว่างงานย่อย (Inter-thread Communication) เป็นมอดูลสำหรับช่วยการทำงานของหลายงานย่อย (Multi-thread) ให้สามารถทำงานได้โดยที่ไม่มีการแย่งใช้ทรัพยากรกัน หนึ่งในนั้นได้แก่ การใช้สัญญาณมือ (Semaphore) ซึ่งช่วยบริหารการใช้ทรัพยากรของระบบ โดยสั่งการควบคุมผ่านมอดูล SEM ส่วนมอดูล LCK จะจัดการการใช้ทรัพยากรร่วมกันเด็ดขาด (Arbitration) หรือแยกกันแบบเด็ดขาด (Mutual Exclusion) นอกจากนั้นยังมีมอดูลที่จัดการการสื่อสารหรือการส่งข้อความระหว่างงานย่อยด้วยกัน ได้แก่ QUE และ MBX ซึ่ง QUE สามารถทำงานโดยไม่ต้องรอการประสานเวลา แต่ MBX จะเป็นการส่งข้อความที่มีการประสานเวลากัน

5. กลุ่มข้อมูลเข้าออก (Input/Output) เป็นมอดูลที่เตรียมไว้สำหรับการส่งผ่านข้อมูลระหว่างตัวประมวลผลกับอุปกรณ์รอบข้าง (Peripherals) หรือระหว่างงานย่อยด้วยกัน ซึ่งเคอร์เนลไบออสบนตัวประมวลผลสัญญาณดิจิทัลได้เตรียมการส่งผ่านข้อมูลในแบบท่อข้อมูล (Data pipes) และตัวแบบทางเลือกเข้าออก (Alternative I/O Model) หรือเรียกอีกอย่างว่ากระแสข้อมูล (Data Streams) โดยที่ท่อข้อมูลถูกจัดการด้วยมอดูล PIP ซึ่งมีขนาดเล็กและสามารถส่งผ่านข้อมูลได้เร็วกว่า เหมาะสำหรับส่งข้อมูลระหว่างงานย่อย ขณะที่กระแสข้อมูลหรือมอดูล SIO นั้นจะมีขนาดใหญ่ แต่มีความยืดหยุ่นมากกว่าในการติดต่อกับระบบภายนอก ซึ่งต้องใช้ร่วมกับมอดูล DEV หรือตัวขับอุปกรณ์ (Device Drivers) ในการส่งผ่านข้อมูลด้วย ซึ่งสามารถทำการประมวลผลในขณะส่งผ่านข้อมูลได้เรียกวิธีการนี้ว่าการเรียงทับซ้อน (Stacking) ทำให้เพิ่มความสามารถในการประมวลผลแบบสายท่อ (Pipeline) การทำสเกล (Scaling) หรือการทำกรองข้อมูล (Filtering) ได้ นอกจากนี้ยังเตรียมมอดูลสำหรับจัดการการส่งผ่านข้อมูลระหว่างเครื่องแม่ข่าย (Host) และอุปกรณ์เป้าหมาย (Target) ได้แก่ HST และ RTDX ตามลำดับ

6. กลุ่มคลังโปรแกรมสนับสนุนชิป (Chip Support Library, CSL) เป็นส่วนที่เตรียมไว้สำหรับกำหนดค่าและควบคุมอุปกรณ์รอบข้าง (Peripherals) บนตัวประมวลผล โดยมีการติดต่อด้วยภาษาซี ซึ่งจะมีการจัดเก็บไว้ในไฟล์คลังข้อมูล (Library) โดยการใช้งานจะต้องมีการประกาศฟังก์ชัน CSL_init() ก่อน

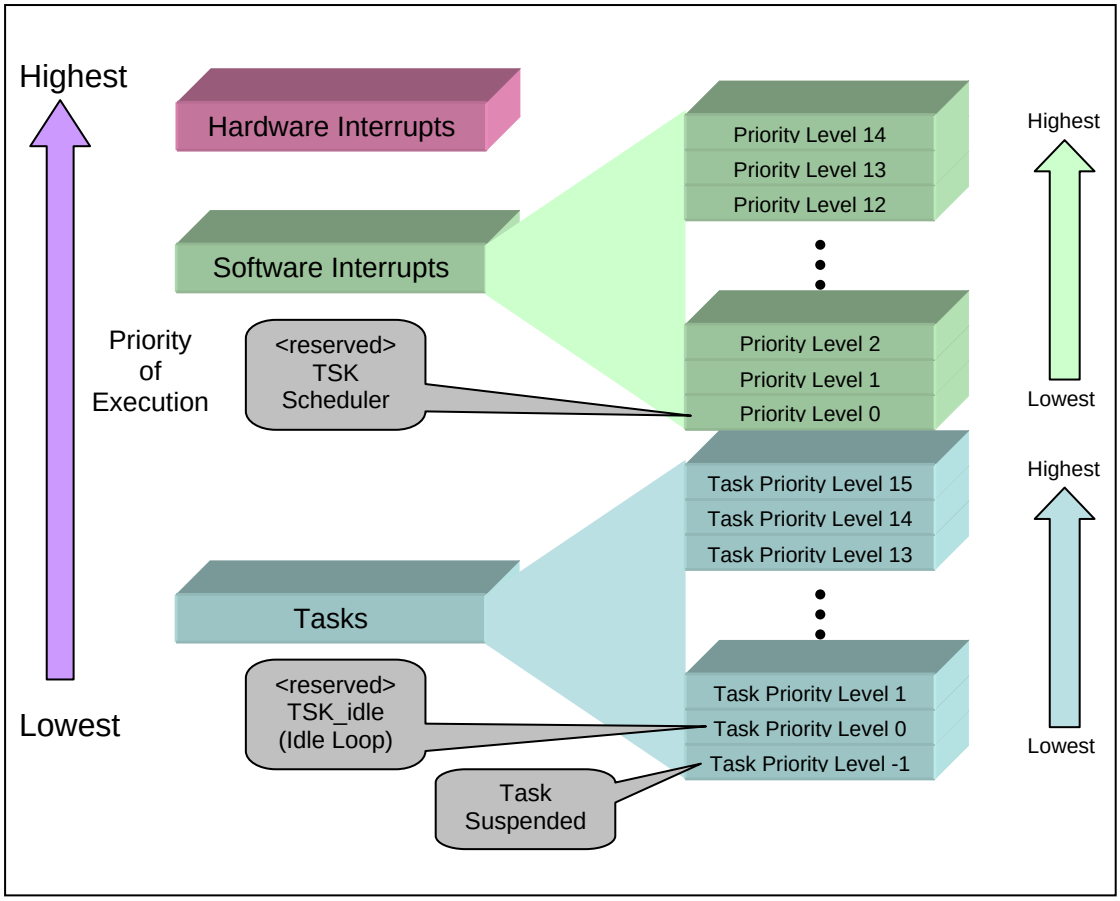
การทำงานของซอฟต์แวร์ที่ใช้เคอร์เนลไบออสบนตัวประมวลผลสัญญาณดิจิทัลได้แบ่งกลุ่มของงานได้เป็น 4 กลุ่ม ดังภาพที่ 23 ได้แก่ ฟังก์ชันขัดจังหวะทางฮาร์ดแวร์ (Hardware Interrupts) ฟังก์ชันขัดจังหวะทางซอฟต์แวร์ (Software Interrupts) ฟังก์ชันงานสำหรับการประสานเวลา (Synchronized Tasks) และฟังก์ชันพื้นหลัง (Background Functions)



ภาพที่ 23 ประเภทของงานย่อยสำหรับเคอร์เนลไบออสบนตัวประมวลผลสัญญาณดิจิทัล
ที่มา: Dart (2001)

จากภาพที่ 23 ฟังก์ชันขัดจังหวะทางฮาร์ดแวร์เป็นฟังก์ชันที่จะทำงานเมื่อมีการขัดจังหวะจากฮาร์ดแวร์ของระบบ ซึ่งการทำงานของฟังก์ชันจะเป็นลักษณะทำงานจนเสร็จ (Run-to-complete) กล่าวคือ จะไม่เปลี่ยนไปทำงานอย่างอื่นจนกว่าจะทำงานในฟังก์ชันจนเสร็จสมบูรณ์ ฟังก์ชันขัดจังหวะทางซอฟต์แวร์เป็นฟังก์ชันที่จะทำงานเมื่อมีการเรียกขัดจังหวะจากซอฟต์แวร์ ซึ่งมีลักษณะคล้ายกับฟังก์ชันขัดจังหวะทางฮาร์ดแวร์แต่มีความสำคัญน้อยกว่าและไม่สามารถเรียกขัดจังหวะจากตัวเองได้ ฟังก์ชันงานสำหรับการประสานเวลาเป็นฟังก์ชันที่ทำงานด้วยตัวเองตามเวลาที่กำหนด (ไม่มีการเรียกแบบฟังก์ชันเรียก) และอาจจะทำงานอยู่ตลอดเวลา เรียกว่างาน (Task)

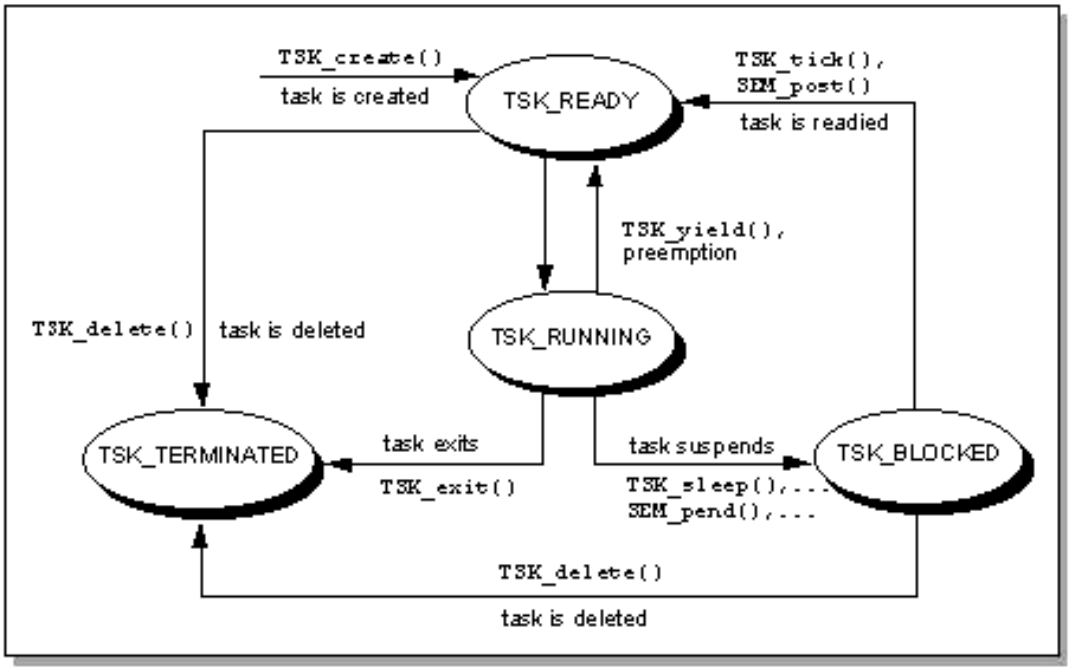
โดยที่แต่ละงานจะสามารถทำการติดต่อระหว่างกันได้ สุดท้ายฟังก์ชันพื้นหลังเป็นฟังก์ชันที่ทำงานได้ก็ต่อเมื่อไม่มีงานประเภทใดทำงานอยู่ ในแต่ละกลุ่มของงานจะมีการจัดลำดับความสำคัญ ดังภาพที่ 24 เนื่องจากแต่ละงานมีความสามารถในการเข้ายึดครองการทำงานของงานที่มีความสำคัญต่ำกว่า



ภาพที่ 24 ลำดับความสำคัญของงานประเภทต่าง ๆ
ที่มา: Dart (2001)

จากภาพที่ 24 ฟังก์ชันขัดจังหวะทางฮาร์ดแวร์จะมีความสำคัญสูงสุด ตามด้วยฟังก์ชันขัดจังหวะทางซอฟต์แวร์ ฟังก์ชันงานสำหรับการประสานเวลา และฟังก์ชันพื้นหลังซึ่งมีความสำคัญน้อยที่สุด ตามลำดับ ในส่วนของฟังก์ชันขัดจังหวะทางซอฟต์แวร์และฟังก์ชันงานสำหรับการประสานเวลาจะมีการแบ่งระดับความสำคัญภายในด้วย เนื่องจากฟังก์ชันขัดจังหวะทางซอฟต์แวร์หรือฟังก์ชันงานสำหรับการประสานเวลาที่มีความสำคัญสูงกว่าสามารถเข้ายึดครองฟังก์ชันขัดจังหวะทางซอฟต์แวร์หรือฟังก์ชันงานสำหรับการประสานเวลาที่มีความสำคัญต่ำกว่าได้

ซึ่งระดับความสำคัญย่อยสำหรับฟังก์ชันขัดจังหวะทางซอฟต์แวร์จะมี 15 ขั้น โดยขั้นที่ 0 จะสำรองไว้สำหรับจัดกำหนดการของงานสำหรับการประสานเวลา (Task Scheduler) และฟังก์ชันงานสำหรับการประสานเวลาจะมี 17 ขั้น โดยขั้นที่ 0 จะเป็นฟังก์ชันพื้นหลัง และขั้นที่ -1 จะเป็นงานที่ถูกพักไว้ (Task Suspended)



ภาพที่ 25 สถานะต่าง ๆ ของงานประสานเวลา
ที่มา: Dart (2001)

ในการทำงานแบบหลายงานร่วมกัน (Multitasking) ไบออสบนตัวประมวลผลสัญญาณดิจิทัลได้เตรียมมอดูลที่ชื่อ TSK สำหรับใช้จัดการงานต่าง ๆ ที่ต้องการมีการประสานเวลา (Synchronization) และการจัดกำหนดการ (Scheduling) โดยแต่ละอ็อบเจกต์งาน TSK จะมีการแบ่งสถานะได้ดังภาพที่ 25 ซึ่งแบ่งเป็น 4 สถานะ คือ

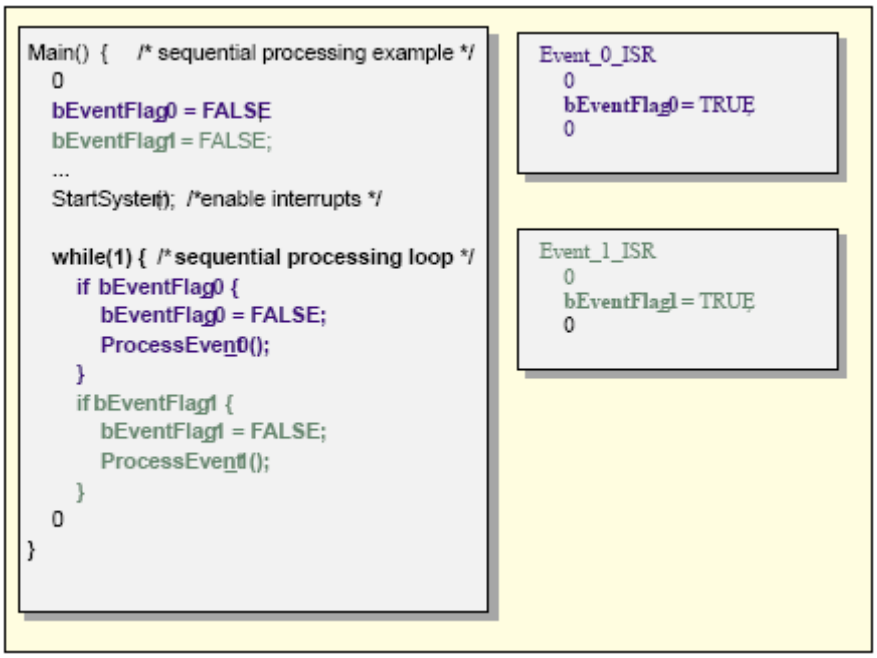
1. สถานะกำลังดำเนินงาน (Running) ซึ่งหมายถึงงานที่กำลังทำงานบนระบบประมวลผล ซึ่งจะมีอยู่งานเดียวเท่านั้นที่กำลังทำงานอยู่

2. สถานะเตรียมพร้อม (Ready) ซึ่งหมายถึงงานที่ถูกจัดกำหนดการไว้และรอให้ตัวประมวลผลว่างหรือไม่มีงานใดอยู่ในสถานะกำลังดำเนินงาน

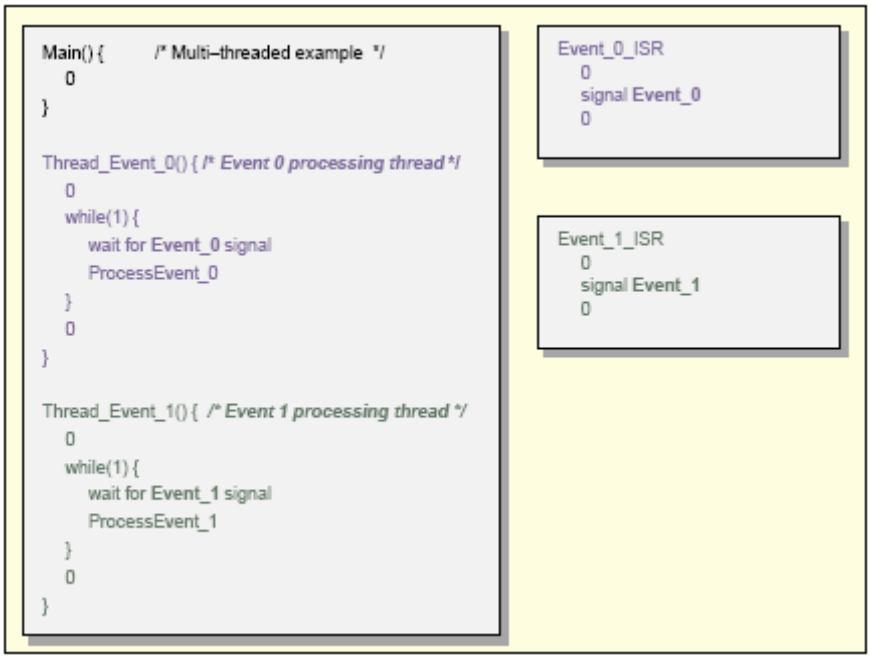
3. สถานะถูกขัดขวางหรือถูกพัก (Blocked or Suspended) หมายถึงงานที่ไม่สามารถทำงานได้จนกว่าจะเกิดเหตุการณ์บางอย่างในระบบมาสั่งให้ทำงานต่อ ซึ่งสถานะงานจะเปลี่ยนไปเป็นสถานะเตรียมพร้อม

4. สถานะถูกทำให้สิ้นสุด (Terminated) หมายถึงงานนั้น ๆ ถูกสั่งให้สิ้นสุดการทำงานและไม่สามารถทำงานต่อได้อีก

เนื่องจากการที่ตัวประมวลผลสัญญาณดิจิทัลของบริษัท Texas Instruments ได้เตรียมเครื่องมือสำหรับเขียนและจัดการระบบปฏิบัติการ ทำให้การเขียนโปรแกรมโดยใช้ไบออสบนตัวประมวลผลสัญญาณดิจิทัลมีลักษณะโครงสร้างแตกต่างจากการเขียนโปรแกรมทั่วไป ดังภาพที่ 26 (Texas Instruments Inc., 2001)



(ก)



(ข)

ภาพที่ 26 โครงสร้างการเขียนโปรแกรม (ก) แบบทั่วไป (ข) แบบใช้ไบออสบนตัวประมวลผลสัญญาณดิจิทัล

ที่มา: Texas Instruments Inc. (2001)

จากภาพที่ 26(ก) เป็นการเขียนโปรแกรมทั่วไป โดยในฟังก์ชันหลักจะทำการกำหนดค่าเริ่มต้นต่าง ๆ รวมทั้งเปิดรับการขัดจังหวะ (Interrupt Enable) จากระบบ จากนั้นระบบจะเข้าสู่การวนรอบทำงานไม่มีที่สิ้นสุด (Forever Loop) เพื่อรอการขัดจังหวะจากระบบภายนอก ซึ่งโปรแกรมการทำงานทั้งหมดจะอยู่ภายในฟังก์ชันหลัก (Main Function)

ส่วนภาพที่ 26(ข) เป็นลักษณะการเขียนโปรแกรมโดยใช้ไบออสบนตัวประมวลผลสัญญาณดิจิทัล โดยที่ในฟังก์ชันหลักจะทำแค่การกำหนดค่าเริ่มต้นเท่านั้น เมื่อออกจากฟังก์ชันหลักระบบจะเข้าสู่รอบการทำงานว่างเปล่า (Idle Loop) ฟังก์ชันในส่วนให้บริการการขัดจังหวะจะแยกอิสระจากฟังก์ชันหลัก ฟังก์ชันเหล่านี้จะถูกเรียกโดยมอดูลต่าง ๆ ในเคอร์เนลไบออสบนตัวประมวลผลสัญญาณดิจิทัล

การเขียนโปรแกรมโดยใช้เคอร์เนลไบออสบนตัวประมวลผลสัญญาณดิจิทัล ผู้พัฒนาโปรแกรมบนตัวประมวลผลสัญญาณดิจิทัลสามารถพัฒนางานทางด้านเวลาจริงได้ โดยแบ่งงานต่าง ๆ ออกเป็นงานย่อยและกำหนดคุณลักษณะของงานย่อยต่าง ๆ ซึ่งจะทำให้ระบบสามารถทำงานต่าง ๆ ได้พร้อมกัน นอกจากนั้นไบออสบนตัวประมวลผลสัญญาณดิจิทัลยังเตรียมเครื่องมือในการวิเคราะห์และแก้จุดบกพร่อง (Debug) ของซอฟต์แวร์ที่ทำงานในเวลาจริงได้

การสื่อสารเครือข่ายสำหรับระบบไบโอเมตริก

ในอดีตจนถึงปัจจุบันมนุษย์มีการกำหนดภาษามาใช้ร่วมกัน เช่น ภาษาไทย ภาษาอังกฤษ ทำให้สามารถสื่อสารกันได้อย่างมีประสิทธิภาพ ถ้าภาษาที่ใช้ต่างกันก็ไม่สามารถสื่อสารให้เข้าใจกันได้ คอมพิวเตอร์ก็เช่นเดียวกันการที่คอมพิวเตอร์เครื่องหนึ่งจะสามารถติดต่อกับคอมพิวเตอร์อีกเครื่องหนึ่งได้จำเป็นต้องใช้ภาษาเดียวกัน ภาษาที่ว่านี้ศัพท์ทางคอมพิวเตอร์จะเรียกว่า โพรโทคอล (Protocol) เช่น คอมพิวเตอร์ที่เชื่อมต่อเข้ากับเครือข่ายอินเตอร์เน็ตจะใช้โพรโทคอล TCP/IP ส่วนคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการเครือข่ายเน็ตแวร์ (NETWARE) ก็ใช้โพรโทคอล IPX/SPX เป็นต้น โดยกลุ่มคอมพิวเตอร์หรืออุปกรณ์สื่อสารที่ติดต่อแลกเปลี่ยนข้อมูลกันโดยใช้โพรโทคอลต่าง ๆ เรียกว่า เครือข่าย (Network)

ระบบเครือข่าย (Network System) คือระบบที่มีเครื่องคอมพิวเตอร์หรืออุปกรณ์สื่อสารหลายเครื่องเชื่อมต่อและสามารถสื่อสารแลกเปลี่ยนข้อมูลซึ่งกันและกันได้อย่างมีประสิทธิภาพ โดยผ่านสื่อกลางประเภทต่าง ๆ เช่น สายโทรศัพท์ สายนำสัญญาณ สายเคเบิลโคแอกเชียล (Co-axial Cable) เส้นใยนำแสง (Fiber Optic) หรือแม้กระทั่งอากาศ เนื่องจากระบบการทำงานบนเครื่องคอมพิวเตอร์หรือสื่อกลางที่ใช้ในแต่ละที่มีความแตกต่างกัน ภายในระบบเครือข่ายจึงมีการกำหนดโพรโทคอลมาตรฐานต่าง ๆ มาช่วยในการติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ ซึ่งผู้พัฒนาระบบเครือข่ายจะต้องศึกษาและเข้าใจการทำงานของโพรโทคอลมาตรฐานที่ใช้ในเครือข่ายด้วย

ในการออกแบบและวิเคราะห์เครือข่ายโดยทั่วไปจะอ้างอิงสถาปัตยกรรมเครือข่ายการเชื่อมต่อระบบเปิด OSI (Open System Interconnect) ซึ่งเป็นมาตรฐานที่กำหนดขึ้นโดยองค์การมาตรฐานนานาชาติ (International Organization for Standardization) หรือเรียกว่า ISO จุดมุ่งหมายของการพัฒนามาตรฐานนี้เพื่อให้คอมพิวเตอร์และอุปกรณ์เครือข่ายที่ผลิตโดยบริษัทต่าง ๆ สามารถทำงานร่วมกันได้ ชุดโพรโทคอลนี้จะเรียกว่า แบบอ้างอิง OSI (OSI Reference Model) เหตุที่เรียกแบบอ้างอิง เนื่องจากแบบอ้างอิง OSI มีการออกแบบโครงสร้างที่ค่อนข้างสมบูรณ์มากที่สุด จึงใช้โพรโทคอลชุดนี้เป็นแบบอ้างอิงในการพัฒนาโพรโทคอลชุดอื่น ๆ อีกทั้งยังเป็นระบบที่ง่ายต่อการอธิบายถึงกลไกการทำงานของโพรโทคอลในเครือข่าย (จตุชัย และอนุโชต, 2546)

ชั้นประยุกต์ (Application Layer)
ชั้นนำเสนอ (Presentation Layer)
ชั้นเซสชัน (Session Layer)
ชั้นเคลื่อนย้ายข้อมูล (Transport Layer)
ชั้นเครือข่าย (Network Layer)
ชั้นเชื่อมโยงข้อมูล (Data Link Layer)
ชั้นกายภาพ (Physical Layer)

ภาพที่ 27 แบบอ้างอิง OSI

แบบอ้างอิง OSI จะแบ่งขั้นตอนการสื่อสารระหว่างคอมพิวเตอร์ออกเป็น 7 ชั้นหรือเลเยอร์ (Layers) ดังภาพที่ 27 ดังนี้

1. ชั้นกายภาพ (Physical Layer) เป็นชั้นที่อยู่ล่างสุดของแบบอ้างอิง OSI รับผิดชอบการรับส่งข้อมูลระดับบิต ในชั้นนี้จะเห็นข้อมูลเป็นระดับบิตเท่านั้น โดยไม่สนใจความหมายของข้อมูล ซึ่งจะทำหน้าที่แปลงข้อมูลให้เป็นสัญญาณไฟฟ้า คลื่นวิทยุ หรือแสงขึ้นอยู่กับสื่อกลางที่ใช้

2. ชั้นเชื่อมโยงข้อมูล (Data Link Layer) เป็นชั้นที่รับผิดชอบในการรับส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ของผู้รับและผู้ส่งให้ถูกต้อง โดยมีการตรวจสอบความผิดพลาดของข้อมูลด้วย ในกรณีที่ข้อมูลผิดพลาดก็จะสามารถตรวจสอบและแก้ไขข้อผิดพลาดได้

3. ชั้นเครือข่าย (Network Layer) เป็นชั้นที่ทำหน้าจัดเส้นทางให้กับกลุ่มข้อมูล (Packet) ระหว่างเครื่องส่งและเครื่องรับที่อยู่ในเครือข่าย ในชั้นนี้จะไม่มีกลไกในการตรวจสอบข้อผิดพลาดของข้อมูล ซึ่งส่วนใหญ่จะทำอยู่แล้วในชั้นเชื่อมโยงข้อมูล

4. ชั้นเคลื่อนย้ายข้อมูล (Transport Layer) เป็นชั้นที่รับผิดชอบในการรับส่งข้อมูลระหว่างโพรเซสส์ของผู้รับและผู้ส่ง โพรเซสส์ (Process) ในที่นี้หมายถึงโปรแกรมที่กำลังรัน (Run) หรือกำลังทำงานบนเครื่องคอมพิวเตอร์ ซึ่งในชั้นนี้จะทำการรับส่งข้อมูลให้ถึงโพรเซสส์ที่ต้องการ โดยมีหน้าที่หลัก ๆ ได้แก่ การแบ่งข้อมูลเป็นกลุ่มข้อมูล (Data Packet) การตรวจสอบกลุ่มข้อมูล และการส่งข้อมูลใหม่อีกครั้ง (Retransmit) เป็นต้น นอกจากนี้ยังสามารถให้บริการโปรแกรมประยุกต์ได้หลาย ๆ โปรแกรมประยุกต์ในเวลาเดียวกัน โดยมีการกำหนดที่อยู่เฉพาะในการรับส่งข้อมูลในแต่ละโปรแกรมประยุกต์ ซึ่งเรียกว่า พอร์ต (Port) โดยแต่ละการเชื่อมต่อเข้ากับพอร์ตจะเรียกว่าซ็อกเก็ต (Socket)

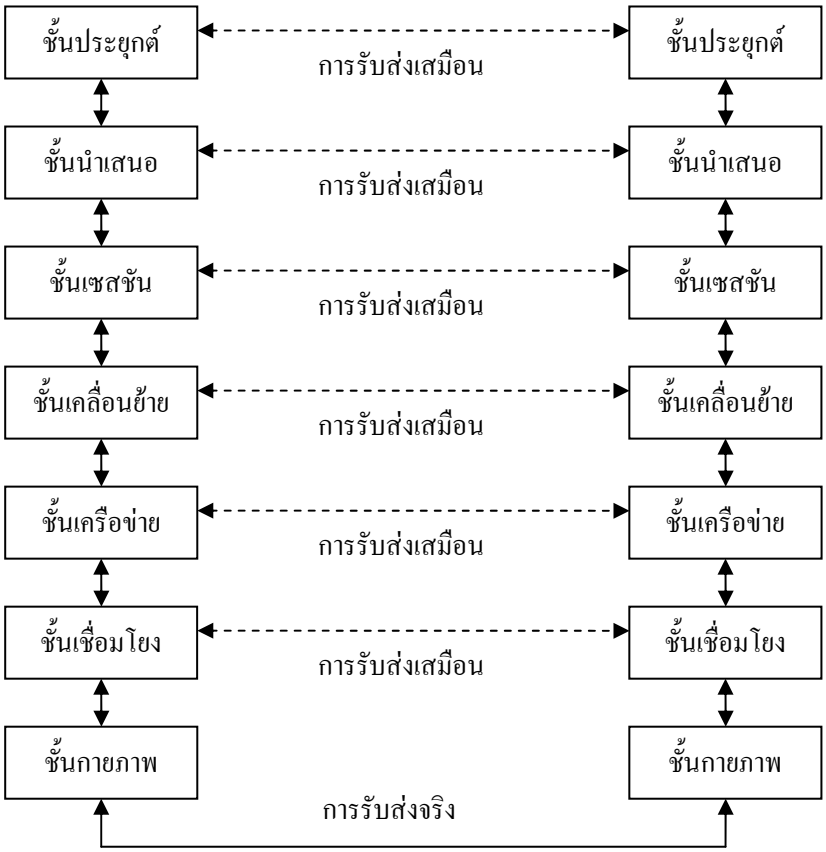
5. ชั้นเซสชัน (Session Layer) เป็นชั้นที่ทำหน้าที่ควบคุมการสื่อสารผ่านเครือข่ายที่กำลังเกิดขึ้น ซึ่งการสื่อสารที่กำลังเกิดขึ้นในช่วงขณะใดขณะหนึ่งจะเรียกว่า เซสชัน (Session) ในชั้นนี้จะรับผิดชอบเกี่ยวกับการสร้างเซสชัน ควบคุมการแลกเปลี่ยนข้อมูล และยกเลิกเซสชันเมื่อการสื่อสารสิ้นสุด นอกจากนี้ยังจัดการเกี่ยวกับการควบคุมจังหวะรับส่งข้อมูล (Synchronization) ได้

6. ชั้นนำเสนอ (Presentation Layer) เป็นชั้นที่รับผิดชอบเกี่ยวกับรูปแบบข้อมูลในการรับส่งผ่านเครือข่าย เนื่องจากในแต่ละเครื่องคอมพิวเตอร์จะมีรูปแบบรหัสข้อมูลที่แตกต่างกัน เช่น บางเครื่องมีรูปแบบรหัสข้อมูลแบบ ASCII (American Standard Code for Information Interchange) หรือบางเครื่องอาจใช้รูปแบบรหัสข้อมูลแบบ EBCDIC (Extended Binary Coded Decimal Interchange Code) ซึ่งข้อมูลเหล่านี้จะถูกแปลงให้อยู่ในรูปแบบมาตรฐานก่อนที่จะส่งไปในเครือข่าย

7. ชั้นประยุกต์ (Application Layer) เป็นชั้นที่อยู่บนสุดของแบบอ้างอิง OSI ถึงแม้ว่าจะชื่อว่าชั้นประยุกต์แต่ก็ไม่ใช่ส่วนที่เป็นโปรแกรมประยุกต์ของผู้ใช้งาน ลักษณะการทำงานของชั้นนี้จะเป็นจุดเชื่อมต่อระหว่างโปรแกรมประยุกต์ของผู้ใช้กับกระบวนการสื่อสารผ่านเครือข่าย

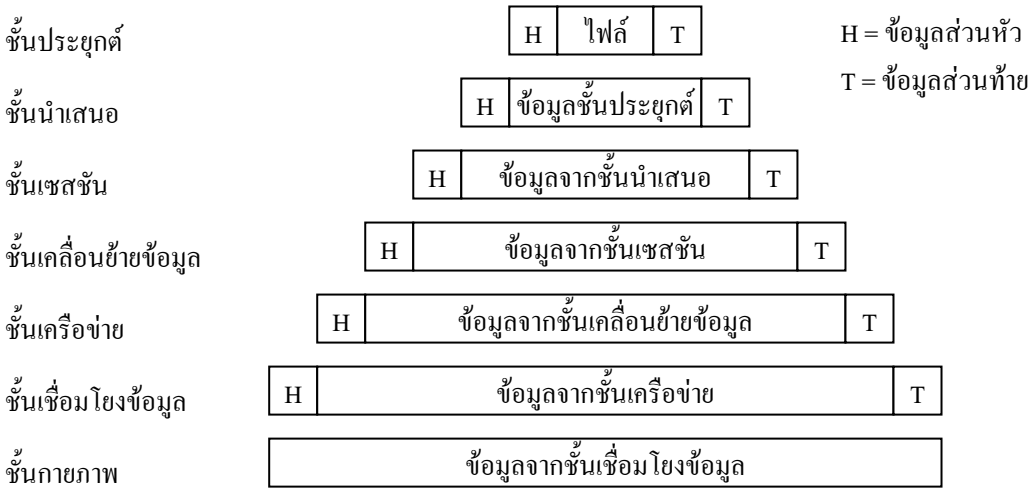
การสื่อสารระหว่างคอมพิวเตอร์สองเครื่องใด ๆ ในเครือข่ายจะเริ่มจากผู้ใช้มีข้อมูลที่ต้องการส่งไปยังผู้ใช้อีกคนหนึ่งที่ใช้คอมพิวเตอร์อีกเครื่องหนึ่ง ข้อมูลนั้นก็จะส่งผ่านไปยังชั้นประยุกต์ซึ่งในเลเยอร์นี้ข้อมูลจะถูกดัดแปลงและใส่ข้อมูลบางอย่างเพิ่มเติม แล้วจะถูกส่งต่อไปยังเลเยอร์ที่อยู่ต่ำกว่า จะทำอย่างนี้ไปเรื่อย ๆ จนถึงเลเยอร์ที่อยู่ต่ำสุด ข้อมูลจะถูกแปลงเป็นสัญญาณเพื่อส่งผ่านสายสัญญาณหรือสื่อกลางที่เชื่อมกันระหว่างคอมพิวเตอร์สองเครื่องนี้จนถึงเครื่องรับ ส่วนกระบวนการในการรับข้อมูลของเครื่องรับนั้นก็จะทำในทางตรงกันข้ามกับเครื่องที่ส่งข้อมูล กล่าวคือ จะเริ่มกระบวนการรับข้อมูลจากชั้นกายภาพก่อน และส่งต่อไปเรื่อย ๆ จนถึงชั้นประยุกต์และส่งต่อให้โปรแกรมประยุกต์ของผู้ใช้ต่อไป การทำงานในแต่ละเลเยอร์จะเป็นไปในลักษณะการให้บริการ (Service) กับเลเยอร์ที่อยู่สูงกว่าและติดกัน โดยที่เลเยอร์ที่ให้บริการจะไม่สนใจในรายละเอียดว่าเลเยอร์ที่เรียกบริการ (เลเยอร์ที่อยู่สูงกว่า) จะมีวิธีการทำงานอย่างไร เพียงแค่รู้ว่าข้อมูลนั้นจะส่งไปถึงผู้รับปลายทางในระดับชั้นของตัวเองเท่านั้น

การจัดเรียงโพรโทคอลเป็นลำดับชั้นนี้ก็เพื่อจำลองการไหลของข้อมูลจากเครื่องส่งถึงเครื่องรับ โดยโพรโทคอลแต่ละชั้นจะส่งข้อมูลไปยังชั้นที่อยู่ติดกัน เช่น ถ้าเป็นการส่งข้อมูล ข้อมูลจะถูกส่งต่อไปยังชั้นต่ำกว่าถัดลงไป แต่ถ้าเป็นการรับข้อมูล ข้อมูลก็จะส่งจากข้างล่างขึ้นข้างบน แต่ละชั้นจะมีจุดเชื่อมต่อกับชั้นที่อยู่ใกล้เคียงเพื่อให้การติดต่อสื่อสารสำเร็จได้ การติดต่อสื่อสารในแต่ละชั้นจะเป็นแบบเพียร์ทูเพียร์ (Peer-to-Peer) หมายความว่าโพรโทคอลชั้นหนึ่งที่อยู่ฝั่งส่งจะติดต่อกับโพรโทคอลในชั้นเดียวกับที่อยู่ฝั่งรับดังภาพที่ 28 ซึ่งข้อมูลที่อยู่ในชั้นจะมีความหมายเฉพาะกับโพรโทคอลที่อยู่ในระดับเดียวกันของฝั่งตรงกันข้ามเท่านั้น



ภาพที่ 28 การสื่อสารระหว่างคอมพิวเตอร์ 2 เครื่องโดยใช้แบบอ้างอิง OSI

ถึงแม้ว่าข้อมูลจะถูกส่งไปยังโพรโทคอลชั้นที่อยู่ติดกันแต่โพรโทคอลในแต่ละชั้นจะคิดว่าเป็นการส่งข้อมูลไปยังชั้นเดียวกันที่อยู่อีกฝั่งหนึ่ง เพื่อให้การสื่อสารแบบนี้เกิดขึ้นได้โพรโทคอลในแต่ละชั้นจะทำการเพิ่มข้อมูลบางอย่างให้กับข้อมูลที่ได้รับมาจากชั้นที่อยู่ข้างบน โดยข้อมูลส่วนที่เพิ่มเข้ามานี้เรียกว่า ข้อมูลส่วนหัว (Header) และข้อมูลส่วนท้าย (Trailer) ดังแสดงในภาพที่ 29 ซึ่งข้อมูลในส่วนนี้จะเข้าใจได้เฉพาะโพรโทคอลที่อยู่ในระดับเดียวกันเท่านั้น โดยในกระบวนการรับส่งข้อมูลนั้น ทางฝั่งส่งจะเพิ่มข้อมูลส่วนหัวและข้อมูลส่วนท้ายเข้าไปในข้อมูลที่จะส่ง ส่วนทางฝั่งรับจะนำข้อมูลส่วนหัวและข้อมูลส่วนท้ายออกจากข้อมูลที่รับเข้ามา



ภาพที่ 29 การเข้าข้อมูลส่วนหัวและข้อมูลส่วนท้ายในโพรโทคอลชั้นต่าง ๆ

จากภาพที่ 29 ข้อมูลในชั้นประยุกต์ (Application Layer) จะประกอบด้วยข้อมูลที่ถูกส่งมาจากโปรแกรมประยุกต์ เช่น การรับส่งไฟล์ข้อมูล จากนั้นจะเพิ่มข้อมูลส่วนหัวและข้อมูลส่วนท้ายเพื่อเป็นข้อมูลให้ชั้นประยุกต์ของอีกฝั่งตรงกันข้ามรับรู้ เสร็จแล้วข้อมูลชุดใหม่นี้ก็จะถูกส่งไปยังชั้นนำเสนอ (Presentation Layer) เมื่อชั้นนำเสนอได้รับข้อมูลก็จะทำการเพิ่มข้อมูลส่วนหัวและข้อมูลส่วนท้ายให้ โดยการส่งในแต่ละลำดับชั้นจะทำเช่นนี้ไปเรื่อย ๆ จนไปถึงชั้นเชื่อมโยงข้อมูล (Data Link Layer) ซึ่งชั้นนี้ก็จะเพิ่มข้อมูลส่วนหัวและข้อมูลส่วนท้าย และเปลี่ยนรูปแบบให้เป็นเฟรม (Frame) จากนั้นเฟรมข้อมูลจะถูกส่งต่อยังชั้นกายภาพ (Physical Layer) ซึ่งจะทำการแปลงเฟรมให้เป็นบิตต่อเนื่อง เพื่อทำการส่งต่อไปบนสื่อนำสัญญาณไปยังปลายทางต่อไป ส่วนการทำงานของฝั่งรับก็จะมีลักษณะตรงกันข้ามกับฝั่งส่ง

ในปัจจุบันระบบเครือข่ายมีโพรโทคอลหลากหลายประเภท ได้แก่ ชุดโพรโทคอล TCP/IP ชุดโพรโทคอล IPX/SPX ชุดโพรโทคอล AppleTalk หรือชุดโพรโทคอล NetBEUI ซึ่งพัฒนาโดยบางบริษัท โครงสร้างของโพรโทคอลเหล่านี้จะแบ่งเป็นชั้น ๆ หรือเลเยอร์คล้ายกับแบบอ้างอิง OSI แต่อาจจะไม่เหมือนกันทุกเลเยอร์บางชุดโพรโทคอลอาจแบ่งขั้นตอนการรับส่งข้อมูลแค่ 4-5 ชั้นเท่านั้น แทนที่จะเป็น 7 ชั้นเหมือนแบบอ้างอิง OSI ซึ่งการทำงานของแต่ละชั้นอาจจะไม่เป็นเหมือนของ OSI ทุกอย่าง อย่างไรก็ตามแบบอ้างอิง OSI ก็ถือได้ว่าเป็นชุดโพรโทคอลที่เป็นต้นแบบสำหรับการศึกษาโพรโทคอลชุดอื่นได้ดี

ชุดโพรโทคอล TCP/IP

ชุดโพรโทคอล TCP/IP (TCP/IP Protocol Suite) เป็นหนึ่งในชุดโพรโทคอลที่นิยมใช้กันอย่างแพร่หลาย ซึ่งเริ่มจากโครงการวิจัยเครือข่าย Advanced Research Project Agency Network หรือ ARPANET ที่สนับสนุนโดยกระทรวงกลาโหมสหรัฐฯ มีจุดประสงค์เพื่อเชื่อมต่อคอมพิวเตอร์ที่มีแพลตฟอร์ม (Platform) หรือระบบปฏิบัติการที่แตกต่างกันให้สามารถสื่อสารกันผ่านเครือข่ายได้ เนื่องจากชุดโพรโทคอล TCP/IP เป็นระบบที่เปิดเผยได้และข้อกำหนดต่าง ๆ ที่ตั้งขึ้นไม่มีการเปลี่ยนแปลงหรือมีการเปลี่ยนแปลงน้อย ดังนั้นชุดโพรโทคอลนี้จึงถูกนำมาใช้เป็นหลักพื้นฐาน (Basis) ของระบบอินเตอร์เน็ตสากลโลก (Worldwide Internet) หรืออินเตอร์เน็ต (Internet) ที่ใช้เป็นเครือข่ายเชื่อมต่อคอมพิวเตอร์หรืออุปกรณ์สื่อสารทั่วโลกในปัจจุบันนั้นเอง (Stevens, 1994)

ชุดโพรโทคอล TCP/IP ถูกพัฒนาในลักษณะลำดับชั้นเช่นเดียวกับแบบอ้างอิง OSI โดยในแต่ละลำดับชั้นจะรับผิดชอบหน้าที่ของการสื่อสารที่แตกต่างกัน ซึ่งแสดงได้ดังภาพที่ 30

<table>
<tr><th>แบบอ้างอิง OSI</th><th>ชุดโพรโทคอล TCP/IP</th></tr>
<tr><td>ชั้นประยุกต์ (Application Layer)</td><td rowspan="2">ชั้นประยุกต์ (Application Layer)</td></tr>
<tr><td>ชั้นนำเสนอ (Presentation Layer)</td></tr>
<tr><td>ชั้นเซสชัน (Session Layer)</td><td rowspan="2">ชั้นเคลื่อนย้ายข้อมูล (Transport Layer)</td></tr>
<tr><td>ชั้นเคลื่อนย้ายข้อมูล (Transport Layer)</td></tr>
<tr><td>ชั้นเครือข่าย (Network Layer)</td><td>ชั้นเครือข่าย (Network Layer)</td></tr>
<tr><td>ชั้นเชื่อมโยงข้อมูล (Data Link Layer)</td><td rowspan="2">ชั้นเชื่อมโยง (Link Layer)</td></tr>
<tr><td>ชั้นกายภาพ (Physical Layer)</td></tr>
</table>

ภาพที่ 30 ลำดับชั้นของชุดโพรโทคอล TCP/IP และการเปรียบเทียบกับแบบอ้างอิง OSI

จากภาพที่ 30 แสดงลำดับชั้นของชุดโพรโทคอล TCP/IP เมื่อเปรียบเทียบกับแบบอ้างอิง OSI โดยที่ชุดโพรโทคอล TCP/IP จะโพรโทคอลออกเป็น 4 ชั้น ได้แก่

1. ชั้นเชื่อมโยง (Link Layer) บางครั้งอาจเรียกว่าชั้นต่อประสานเครือข่าย (Network Interface Layer) ซึ่งจะรวมถึงตัวขับอุปกรณ์ (Device Driver) บนระบบปฏิบัติการและฮาร์ดแวร์ต่อประสานเครือข่าย (Network Interface Hardware) ในคอมพิวเตอร์ด้วย สำหรับชุดโพรโทคอล TCP/IP ไม่ได้กำหนดมาตรฐานสำหรับทำงานในชั้นนี้ แต่อย่างไรก็ตามชุดโพรโทคอล TCP/IP ก็สามารถใช้งานได้กับเครือข่ายหลายประเภท ได้แก่ อีเธอร์เน็ต (Ethernet) FDDI (Fiber Distributed Data Interface) PPP (Point to Point Protocol) SLIP (Serial Line Internet Protocol) และ ISDN (Integrated Services Digital Network) เป็นต้น

2. ชั้นเครือข่าย (Network Layer) บางครั้งอาจเรียกว่าชั้นอินเตอร์เน็ต (Internet Layer) ซึ่งจะจัดการเกี่ยวกับการเคลื่อนที่ของกลุ่มข้อมูลภายในเครือข่ายเรียกว่า การค้นหาเส้นทาง (Routing) ในชั้นนี้จะประกอบด้วยโพรโทคอลต่าง ๆ ได้แก่ โพรโทคอลอินเตอร์เน็ต (Internet Protocol) หรือ IP โพรโทคอลข้อความควบคุมอินเตอร์เน็ต (Internet Control Message Protocol) หรือ ICMP และโพรโทคอลจัดการกลุ่มอินเตอร์เน็ต (Internet Group Management Protocol) หรือ IGMP เป็นต้น

3. ชั้นเคลื่อนย้ายข้อมูล (Transport Layer) ทำหน้าที่ควบคุมการไหล (Flow Control) ของข้อมูลระหว่างเครื่องคอมพิวเตอร์หรืออุปกรณ์สื่อสาร ในชั้นนี้จะประกอบด้วยโพรโทคอลที่แตกต่างกัน 2 โพรโทคอล คือ โพรโทคอล TCP (Transmission Control Protocol) และโพรโทคอล UDP (User Datagram Protocol) ซึ่งการเลือกใช้ชนิดโพรโทคอลในชั้นนี้จะขึ้นอยู่กับความต้องการความน่าเชื่อถือของข้อมูลในโปรแกรมประยุกต์ข้างบน

โพรโทคอล TCP เป็นการรับส่งข้อมูลที่มีความน่าเชื่อถือระหว่างผู้ส่งและผู้รับ ซึ่งการรับส่งข้อมูลจะมีการส่งกลุ่มข้อมูลตอบรับเพื่อยืนยันว่าผู้รับได้รับกลุ่มข้อมูลแล้ว ดังนั้นความน่าเชื่อถือของข้อมูลในชั้นนี้ทำให้โพรโทคอลชั้นที่สูงกว่า (เช่น ชั้นประยุกต์) ไม่จำเป็นต้องตรวจสอบความน่าเชื่อถือเอง ทำให้เหมาะสำหรับนำไปใช้งานกับโปรแกรมประยุกต์ที่ต้องการความน่าเชื่อถือสูง

โพรโทคอล UDP เป็นการรับส่งข้อมูลแบบง่าย ซึ่งจะทำการส่งข้อมูลที่เรียกว่า ดาต้าแกรม (Datagram) ไปยังผู้รับโดยไม่รับประกันว่าข้อมูลจะไปถึงผู้รับหรือไม่ ซึ่งถ้าโพรโทคอลชั้นประยุกต์ต้องการความน่าเชื่อถือจะต้องทำการตรวจสอบความน่าเชื่อถือเอง เนื่องจากผู้ส่งไม่ต้องรอการยืนยันจากผู้รับทำให้สามารถส่งข้อมูลได้เร็วกว่าโพรโทคอล TCP ซึ่งเหมาะกับโปรแกรมประยุกต์ต้องการความเร็วในการรับส่งสูงแต่ต้องการความน่าเชื่อถือไม่มากนัก

4. ชั้นประยุกต์ (Application Layer) ทำหน้าที่จัดการรายละเอียดการทำงานของแต่ละโปรแกรมประยุกต์ ซึ่งโปรแกรมประยุกต์ที่สำคัญบน TCP/IP ได้แก่

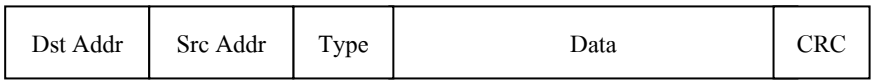
- Telnet ใช้สำหรับการล็อกอินระยะไกล (Remote Login)
- FTP (File Transfer Protocol) ใช้สำหรับถ่ายโอนไฟล์ระหว่างแม่ข่าย (Host)
- SMTP (Simple Mail Transfer Protocol) ใช้สำหรับการรับส่งอีเมล (E-mail) ระหว่างเครื่องบริการไปรษณีอิเล็กทรอนิกส์ (Mail Server)
- SNMP (Simple Network Management Protocol) ใช้ควบคุมและจัดการเครือข่าย
- HTTP (Hyper Text Transfer Protocol) ใช้สำหรับการรับส่งไฟล์เว็บเพจ (Webpage) ระหว่างเว็บบราวเซอร์ (Web Browser) และเครื่องให้บริการเว็บ (Web Server)

<u>โพรโทคอลชั้นเชื่อมโยงและอีเธอร์เน็ต</u>

ชั้นเชื่อมโยง (Link Layer) เป็นชั้นโพรโทคอลสำหรับการเชื่อมโยงข้อมูลในระดับล่าง หน้าที่ของโพรโทคอลชั้นนี้ คือ การรับส่งชุดข้อมูลหรือดาต้าแกรม (Datagram) จากโพรโทคอลชั้นอินเตอร์เน็ตเพื่อส่งออกไปยังเครือข่าย นอกจากนี้ยังทำหน้าที่ค้นหาหรือแปลงค่าระหว่างเลขที่อยู่ไอพี (IP Address) และเลขที่อยู่แม็ก (MAC Address) ของเครื่องคอมพิวเตอร์ที่ต้องการติดต่อด้วย โดยใช้โพรโทคอล ARP (Address Resolution Protocol) และโพรโทคอล RARP (Reverse Address Resolution Protocol) ตามลำดับ โพรโทคอลในการรับส่งข้อมูลระหว่างชั้นอินเตอร์เน็ตกับอุปกรณ์สื่อสารภายนอกสามารถใช้งานได้กับเครือข่ายหลายประเภท ได้แก่ อีเธอร์เน็ต (Ethernet) FDDI (Fiber Distributed Data Interface) PPP (Point to Point Protocol) SLIP (Serial Line Internet Protocol) และ ISDN (Integrated Services Digital Network) เป็นต้น ขึ้นอยู่กับฮาร์ดแวร์ต่อประสานเครือข่ายที่ใช้ในคอมพิวเตอร์

อีเธอร์เน็ต (Ethernet) เป็นมาตรฐานการสื่อสารที่ตั้งขึ้นในปี ค.ศ. 1982 โดยความร่วมมือของบริษัท Digital Equipment Corp., Intel Corp. และ Xerox Corp. ซึ่งเป็นรูปแบบการสื่อสารที่โดดเด่นในเทคโนโลยีเครือข่ายพื้นที่ท้องถิ่น (Local Area Network) และถูกนำมาใช้ในชุดโพรโทคอล TCP/IP ในปัจจุบัน ซึ่งอีเธอร์เน็ตใช้วิธีการเข้าถึงแบบ CSMA/CD (Carrier Sense, Multiple Access with Collision Detection) ที่มีความเร็วในการรับส่งข้อมูลเท่ากับ 10 เมกกะบิตต่อวินาที และใช้เลขที่อยู่ขนาด 48 บิต (Stevens, 1994)

ในชุดโพรโทคอล TCP/IP ได้กำหนดรูปแบบการเข้าเฟรมข้อมูลอีเธอร์เน็ต (Ethernet Encapsulation) ของข้อมูลจากโพรโทคอลชั้นอินเตอร์เน็ตไว้ในข้อกำหนด RFC894 (Request for Comment) ซึ่งแสดงได้ดังภาพที่ 31



ภาพที่ 31 รูปแบบการเข้าเฟรมข้อมูลอีเธอร์เน็ต
ที่มา: Stevens (1994)

จากภาพที่ 31 เลขที่อยู่ต้นทาง (Source Address) และเลขที่อยู่ปลายทาง (Destination Address) มีขนาด 48 บิตหรือ 6 ไบต์ ซึ่งเรียกกันว่าเลขที่อยู่ฮาร์ดแวร์ (Hardware Address) หรือเลขที่อยู่แม็ก (MAC Address) โดยถือว่าเป็นเลขที่อยู่ระดับล่างสุด ในชุดโพรโทคอล TCP/IP ใช้โพรโทคอล ARP และ RARP ในการจับคู่ระหว่างหมายเลขไอพีขนาด 32 บิตกับเลขที่อยู่ฮาร์ดแวร์ขนาด 48 บิต

ส่วนชนิดข้อมูล (Type) เป็นส่วนที่ต่อจากเลขที่อยู่ฮาร์ดแวร์ซึ่งจะบอกชนิดของข้อมูลที่ตามหลังมา ตัวอย่างเช่น ค่า 0800 หมายถึงข้อมูลที่ตามมาเป็นข้อมูลจากโพรโทคอลชั้นอินเตอร์เน็ต 0806 หมายถึงข้อมูลสำหรับโพรโทคอล ARP และ 0835 หมายถึงข้อมูลสำหรับโพรโทคอล RARP เป็นต้น ส่วนข้อมูลที่ตามมานั้นจะมีขนาดตั้งแต่ 46 ไบต์จนถึง 1500 ไบต์ ถ้าข้อมูลที่ต้องการรับส่งมีขนาดน้อยการ 46 ไบต์ อีเธอร์เน็ตก็จะเพิ่มข้อมูลที่มีค่าเป็น 0 ให้ครบ 46 ไบต์ หรือถ้าข้อมูลที่ต้องการรับส่งมีขนาดมากกว่า 1500 ไบต์ อีเธอร์เน็ตก็จะทำการแบ่งข้อมูลเป็นหลาย ๆ เฟรม

ส่วนสุดท้ายของเฟรมอีเธอร์เน็ต คือ CRC (Cyclic Redundancy Check) ซึ่งถือว่าเป็นข้อมูลส่วนท้ายเรียกว่า ลำดับการตรวจสอบเฟรม (Frame Check Sequence, FCS) ทำหน้าที่ตรวจสอบความผิดพลาดของข้อมูล

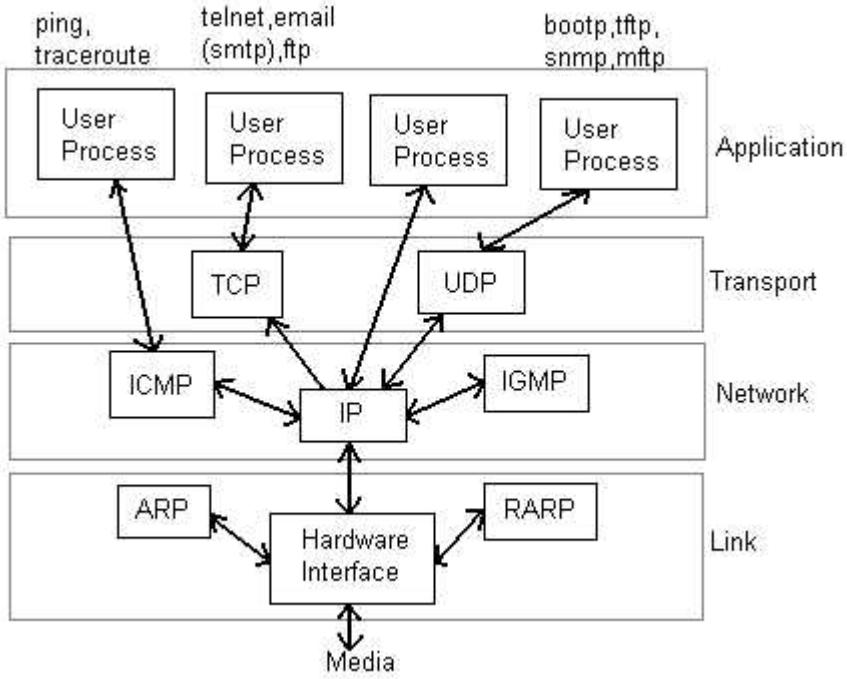
<u>โพรโทคอลชั้นอินเตอร์เน็ต</u>

ชั้นเครือข่าย (Network Layer) หรือชั้นอินเตอร์เน็ต (Internet Layer) ทำหน้าที่ควบคุมการเคลื่อนที่ของกลุ่มข้อมูลในเครือข่าย ในชั้นนี้ประกอบด้วยโพรโทคอลต่าง ๆ ได้แก่ โพรโทคอลอินเตอร์เน็ต (Internet Protocol) โพรโทคอลข้อความควบคุมอินเตอร์เน็ต (Internet Control Message Protocol) และโพรโทคอลจัดการกลุ่มอินเตอร์เน็ต (Internet Group Management Protocol) เป็นต้น ซึ่งมีรายละเอียดดังนี้

1. โพรโทคอลจัดการกลุ่มอินเตอร์เน็ต (Internet Group Management Protocol) หรือ IGMP เป็นโพรโทคอลที่ทำหน้าที่แจ้งให้อุปกรณ์จัดเส้นทาง (Router) ทราบเกี่ยวกับกลุ่มของหมายเลขไอพีแบบมัลติคาสต์ (Multicast) ซึ่งเป็นกลุ่มของเครื่องแม่ข่ายที่มีหมายเลขไอพีเหมือนกัน โดยข้อมูลนี้จะถูกส่งต่อ ๆ กันไปยังอุปกรณ์จัดเส้นทางต่าง ๆ ที่อยู่ในเครือข่ายเพื่อให้เครือข่ายสามารถรองรับการรับส่งข้อมูลแบบมัลติคาสต์ได้ การส่งกลุ่มข้อมูลของ IGMP จะส่งผ่านโพรโทคอลอินเตอร์เน็ต (Internet Protocol) ไปยังโพรโทคอลชั้นเชื่อมโยงเพื่อส่งข้อมูลออกไปสู่เครือข่ายภายนอก

2. โพรโทคอลข้อความควบคุมอินเตอร์เน็ต (Internet Control Message Protocol) หรือ ICMP ทำหน้าที่รายงานข้อผิดพลาดต่าง ๆ ที่เกิดขึ้นในระหว่างการรับส่งข้อมูลในเครือข่าย ซึ่งข้อความบางอย่างอาจส่งไปให้ผู้ใช้ได้รับรู้ด้วย ดังนั้นข้อความต่าง ๆ ในโพรโทคอล ICMP จึงมีประโยชน์มากสำหรับการวิเคราะห์และค้นหาจุดบกพร่องของเครือข่าย การส่งกลุ่มข้อมูลของ ICMP จะส่งผ่านโพรโทคอลอินเตอร์เน็ต (Internet Protocol) ไปยังโพรโทคอลชั้นเชื่อมโยงเพื่อส่งข้อมูลออกไปสู่เครือข่ายภายนอกเช่นเดียวกับโพรโทคอล IGMP

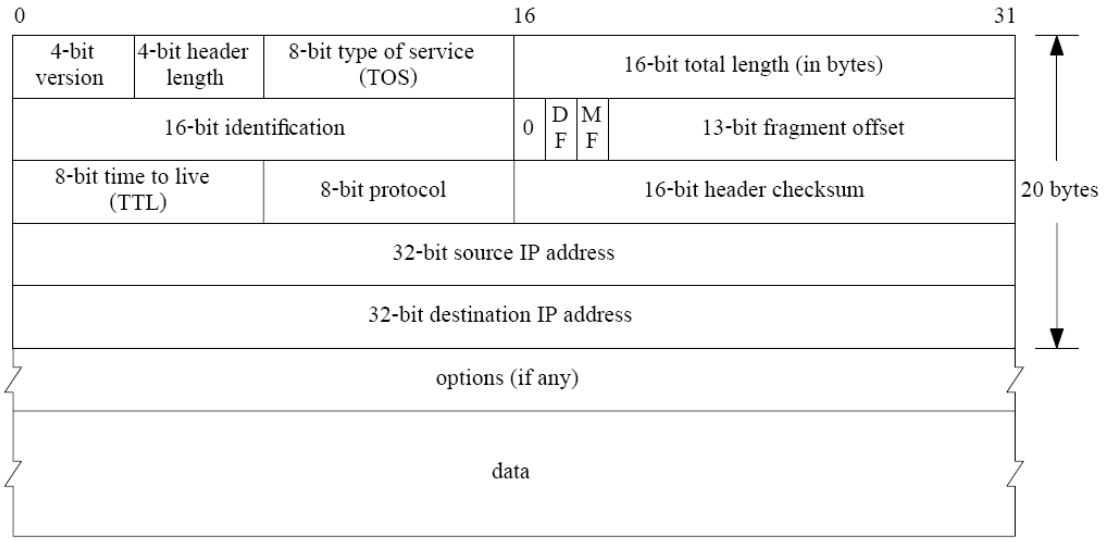
3. โพรโทคอลอินเตอร์เน็ต (Internet Protocol) หรือ IP เป็นโพรโทคอลรับส่งข้อมูลในระบบเครือข่าย โดยโพรโทคอลทั้งหมดในระดับเดียวกันหรือสูงกว่าที่ต้องการส่งข้อมูลไปออกยังเครือข่ายจะต้องส่งผ่านโพรโทคอลอินเตอร์เน็ตในลักษณะกลุ่มข้อมูล ดังภาพที่ 32 ตัวอย่างเช่น โพรโตคอล TCP UDP ICMP หรือ IGMP โดยจะต้องติดต่อกับโพรโทคอลอินเตอร์เน็ตในการส่งข้อมูลไปยังส่วนเชื่อมต่อฮาร์ดแวร์ (Hardware Interface) เท่านั้น นอกจากนี้โพรโทคอลอินเตอร์เน็ตยังรับผิดชอบการจัดเส้นทาง (Routing) ให้กลุ่มข้อมูลที่จะส่งไปยังผู้รับด้วย



ภาพที่ 32 โพรโทคอลในแต่ละลำดับชั้นของชุดโพรโทคอล TCP/IP
ที่มา: Stevens (1994)

โพรโทคอลอินเตอร์เน็ตเป็นโพรโทคอลที่ให้บริการแบบไม่น่าเชื่อถือ (Unreliable) และไม่เชื่อมต่อกัน (Connectionless) ซึ่งความไม่น่าเชื่อถือหมายความว่าโพรโทคอลนี้ไม่รับประกันว่าข้อมูลดาต้าแกรมจะส่งไปถึงปลายทางหรือไม่ ขณะที่ความไม่เชื่อมต่อกันหมายความว่าโพรโทคอลนี้ไม่รับรู้ลำดับการส่งของกลุ่มข้อมูลซึ่งอาจทำให้ลำดับการส่งผิดพลาดได้ อย่างไรก็ตามโพรโทคอลอินเตอร์เน็ตสามารถตรวจสอบข้อผิดพลาดได้จากโพรโทคอล ICMP ส่วนปัญหาความน่าเชื่อถือของการรับส่งข้อมูลจะเป็นหน้าที่โพรโทคอลที่มีลำดับชั้นสูงกว่า ได้แก่ โพรโทคอล TCP

ภาพที่ 33 แสดงรูปแบบโครงสร้างข้อมูลของโพรโทคอลอินเตอร์เน็ต ซึ่งประกอบด้วยข้อมูลส่วนหัวขนาด 20 ไบต์ โดยไม่รวมส่วนฟิลด์ตัวเลือก (Option)



ภาพที่ 33 รูปแบบโครงสร้างข้อมูลของโพรโทคอลอินเตอร์เน็ต
ที่มา: Stevens (1994)

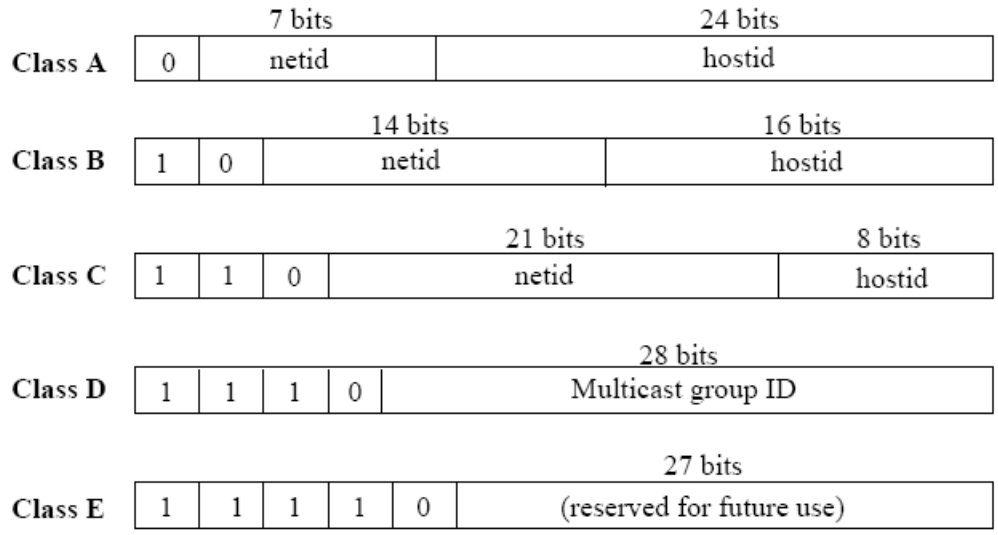
จากภาพที่ 33 ข้อมูลบิตที่มีความสำคัญมากที่สุด (Most Significant Bit, MSB) จะแทนด้วยหมายเลข 0 ในขณะที่ข้อมูลบิตที่มีความสำคัญน้อยที่สุด (Least Significant Bit, LSB) จะแทนด้วยหมายเลข 31 ข้อมูล 4 ไบต์จากความกว้าง 32 บิตในแต่ละแถวจะถูกส่งไปทีละ 1 ไบต์โดยเริ่มจากบิต 0 ถึง 7 ตามด้วย 8 ถึง 15 ตามด้วย 16 ถึง 23 และสุดท้าย 24 ถึง 31 ตามลำดับ โดยการเรียงลำดับการส่งแบบนี้ว่า Big Endian หรือลำดับไบต์เครือข่าย (Network Byte Order)

ข้อมูลส่วนหัวของโพรโทคอลอินเตอร์เน็ตประกอบด้วยฟิลด์ข้อมูลต่าง ๆ ดังนี้

- เวอร์ชัน (Version) เป็นข้อมูลที่บอกถึงเวอร์ชันของโพรโทคอลอินเตอร์เน็ต ซึ่งในปัจจุบันจะใช้เวอร์ชัน 4 หรือเรียกว่า IPv4 ในอนาคตอันใกล้อาจมีการเปลี่ยนไปใช้เวอร์ชันใหม่ คือ เวอร์ชัน 6 เนื่องจากปัญหาหมายเลขไอพีไม่เพียงพอต่อการใช้งาน
- ความยาวข้อมูลส่วนหัว (Header Length) เป็นตัวเลขที่บอกความยาวของข้อมูลในส่วนหัว มีขนาด 4 บิต
- รูปแบบการบริการ (Type of Service) แสดงรูปแบบบริการที่ต้องการใช้ ได้แก่ บริการความล่าช้าน้อยที่สุด (Minimized Delay) บริการอัตรารับส่งสูงที่สุด (Maximized Throughput) บริการความน่าเชื่อถือมากที่สุด (Maximized Reliability) เป็นต้น
- ความยาว (Length) เป็นความยาวทั้งหมดของกลุ่มข้อมูลไอพีมีหน่วยเป็นไบต์ เนื่องจากฟิลด์ข้อมูลมีขนาด 16 บิตทำให้ขนาดสูงสุดของกลุ่มข้อมูลไอพีเท่ากับ 65,535 ไบต์

- ตัวชี้ (Identification) เป็นหมายเลขเฉพาะที่กำหนดขึ้นโดยผู้ส่ง ซึ่งจะไม่ซ้ำกันในแต่ละกลุ่มข้อมูล
- ฟิลด์แฟลก (Flag) เป็นฟิลด์ที่ใช้จัดการเกี่ยวกับการแบ่งข้อมูลเป็นกลุ่มข้อมูลย่อย ประกอบด้วย DF และ MF
- ส่วนชดเชยข้อมูลย่อย (Fragment Offset) เป็นส่วนที่บอกว่ากลุ่มข้อมูลย่อยนี้อยู่ห่างจากจุดเริ่มต้นของข้อมูลทั้งหมดเท่าใดในกรณีที่ข้อมูลถูกแบ่งเป็นกลุ่มข้อมูลย่อย
- อายุของกลุ่มข้อมูล (Time-to-Live หรือ TTL) เป็นการกำหนดโดยผู้ส่งว่ากลุ่มข้อมูลแต่ละกลุ่มข้อมูลจะอยู่ในเครือข่ายได้นานเท่าใด ซึ่งจะบอกเป็นจำนวนครั้งที่ผ่านอุปกรณ์จัดเส้นทาง โดยจะลดค่า TTL ทุกครั้งที่ผ่านอุปกรณ์จัดเส้นทาง เมื่อค่านี้เป็นศูนย์กลุ่มข้อมูลนี้ก็จะถูกลบทิ้งไป
- โพรโทคอล (Protocol) เป็นข้อมูลที่บอกถึงโพรโทคอลที่สูงกว่าติดต่อด้วย เช่น TCP และ UDP เป็นต้น
- ข้อมูลตรวจผลรวมส่วนหัว (Header Checksum) เป็นข้อมูลที่ใช้ในการตรวจสอบข้อผิดพลาดในข้อมูลส่วนหัวของกลุ่มข้อมูล
- เลขที่อยู่ไอพีเครื่องต้นทาง (Source IP Address) เป็นหมายเลขไอพีของผู้ส่ง
- เลขที่อยู่ไอพีเครื่องปลายทาง (Destination IP Address) เป็นหมายเลขไอพีของผู้รับ
- ข้อมูล (Data) เป็นข้อมูลของโพรโทคอลที่อยู่สูงกว่า ได้แก่ โพรโทคอล TCP หรือโพรโทคอล UDP เป็นต้น

เครื่องคอมพิวเตอร์หรืออุปกรณ์ต่าง ๆ ในเครือข่ายอินเตอร์เน็ตจะมีหมายเลขเฉพาะสำหรับสื่อสารกัน คือ เลขที่อยู่ไอพี (IP Address) หรือเรียกให้เข้าใจง่ายว่า หมายเลขไอพี ซึ่งมีขนาด 32 บิต การกำหนดหมายเลขไอพีจะมีการแบ่งเป็นกลุ่มเรียกว่า คลาส (Class) ดังภาพที่ 34 แทนการกำหนดเรียงลำดับ 1 2 และ 3 ไปเรื่อย ๆ เนื่องจากแต่ละเครือข่ายจะมีลักษณะแบ่งเป็นกลุ่มอยู่แล้ว



ภาพที่ 34 การแบ่งคลาสของหมายเลขไอพี
ที่มา: Stevens (1994)

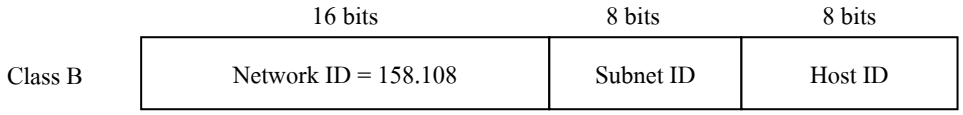
จากภาพที่ 34 หมายเลขไอพีสามารถแบ่งได้เป็น 5 กลุ่ม ได้แก่ คลาส A B C D และ E ซึ่งมีลักษณะที่แตกต่างกันดังนี้

- คลาส A มีบิตแรกเป็น 0 เท่านั้น ส่วนหมายเลขเครือข่าย (Network ID) มี 7 บิต ทำให้มีเครือข่ายได้ทั้งหมด 126 เครือข่าย โดยไม่นับค่าที่เป็น 0 หมดและ 1 หมด ส่วนอีก 24 บิตที่เหลือจะเป็นหมายเลขแม่ข่าย (Host ID) ซึ่งสามารถมีจำนวนแม่ข่ายได้ถึง 16,777,214 เครื่อง โดยไม่นับค่าที่เป็น 0 หมดและ 1 หมดเช่นเดียวกับหมายเลขเครือข่าย
- คลาส B มีสองบิตแรกเป็น 10 เท่านั้น ส่วนหมายเลขเครือข่ายมี 14 บิต ทำให้มีเครือข่ายได้ทั้งหมด 16,382 เครือข่าย ส่วนอีก 16 บิตที่เหลือจะเป็นหมายเลขแม่ข่ายซึ่งสามารถมีจำนวนแม่ข่ายได้ถึง 65,534 เครื่อง
- คลาส C มีสามบิตแรกเป็น 110 เท่านั้น ส่วนหมายเลขเครือข่ายมี 21 บิต ทำให้มีเครือข่ายได้ทั้งหมด 2,097,150 เครือข่าย ส่วนอีก 8 บิตที่เหลือจะเป็นหมายเลขแม่ข่ายซึ่งสามารถมีจำนวนแม่ข่ายได้ 254 เครื่อง
- คลาส D มีสี่บิตแรกเป็น 1110 ซึ่งจะเป็นคลาสสำหรับหมายเลขไอพีที่ใช้สำหรับการมัลติคาสต์ (Multicasting) หรือสำหรับส่งข้อมูลแบบมีแม่ข่ายปลายทางหลายเครื่องแต่อาจจะอยู่คนละเครือข่ายกัน
- คลาส E มีห้าบิตแรกเป็น 11110 ซึ่งสำรองไว้ใช้ในอนาคต

การเขียนหมายเลขไอพีจะเขียนในรูปตัวเลขฐานสิบ 4 ตัว โดยแต่ละตัวจะแทน 1 ไบต์ ซึ่งเรียกว่าสัญลักษณ์ Dotted-decimal ตัวอย่างของการเขียนหมายเลขไอพีสำหรับคลาส B ได้แก่ 158.108.47.36 เป็นต้น

จากที่กล่าวในข้างต้นการกำหนดหมายเลขไอพีจะต้องเป็นหมายเลขเฉพาะไม่ซ้ำกัน ดังนั้นจะต้องมีหน่วยงานกลางสำหรับกำหนดหมายเลขไอพี ซึ่งได้แก่ หน่วยงานข่าวสารเครือข่ายอินเตอร์เน็ตกลาง (Internet Network Information Center หรือ InterNIC) โดย InterNIC จะกำหนดเฉพาะฟิลด์หมายเลขสำหรับเครือข่าย (Network ID) เท่านั้น ส่วนฟิลด์หมายเลขสำหรับแม่ข่าย (Host ID) จะกำหนดโดยผู้ดูแลเครือข่าย (Administrator) ของแต่ละระบบเครือข่าย

การแบ่งหมายเลขไอพีให้เป็นคลาสเป็นการแบ่งใช้กลุ่มของหมายเลขไอพีในแต่ละเครือข่ายที่มีขนาดแตกต่างกันได้อย่างมีประสิทธิภาพ โดยแต่ละเครือข่ายจะได้รับเฉพาะฟิลด์หมายเลขสำหรับเครือข่าย (Network ID) ในกรณีที่เครือข่ายมีขนาดใหญ่ ผู้ดูแลเครือข่าย (Administrator) สามารถแบ่งหมายเลขสำหรับแม่ข่าย (Host ID) ให้เป็นเครือข่ายย่อยได้อีก ซึ่งวิธีการนี้เรียกว่าการกำหนดหมายเลขเครือข่ายย่อย (Subnet Addressing) โดยผู้ดูแลเครือข่ายสามารถแบ่งหมายเลขสำหรับแม่ข่าย (Host ID) ในภาพที่ 34 ให้เป็นฟิลด์ย่อย 2 ฟิลด์ ได้แก่ ฟิลด์สำหรับหมายเลขเครือข่ายย่อย (Subnet ID) และฟิลด์หมายเลขสำหรับแม่ข่าย (Host ID) ดังภาพที่ 35 โดยขนาดของฟิลด์ทั้งสองจะเป็นเท่าใดก็ได้ขึ้นอยู่กับผู้ดูแลเครือข่าย



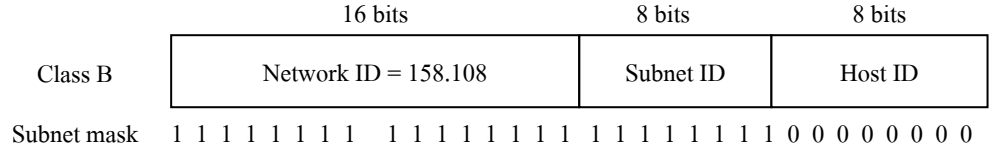
ภาพที่ 35 การกำหนดหมายเลขเครือข่ายย่อยสำหรับคลาส B

ภาพที่ 35 เป็นตัวอย่างการกำหนดหมายเลขเครือข่ายย่อยสำหรับหมายเลขไอพีคลาส B ที่มีหมายเลขเครือข่ายเป็น 158.108 โดยหมายเลขที่เหลืออีก 16 บิต จะแบ่งเป็นหมายเลขเครือข่ายย่อย (Subnet ID) 8 บิตและหมายเลขสำหรับแม่ข่าย (Host ID) 8 บิต ซึ่งทำให้ผู้ดูแลเครือข่ายสามารถแบ่งเครือข่ายภายในได้ 254 เครือข่ายย่อยและแต่ละเครือข่ายย่อยจะมีเครื่องลูกข่ายได้สูงสุด 254 เครื่อง เนื่องจากหมายเลขไอพีจะเป็นศูนย์หมดหรือหนึ่งหมดไม่สามารถใช้ได้

การกำหนดหมายเลขเครือข่ายย่อยนิยมกำหนดในหมายเลขไอพีคลาส A และ B เนื่องจากมีหมายเลขไอพีในเครือข่ายมาก ส่วนคลาส C นั้นสามารถทำได้แต่คลาส C มีหมายเลขเครื่องน้อยอยู่แล้วจึงไม่นิยมแบ่งเป็นเครือข่ายย่อย

การแบ่งหมายเลขไอพีเป็นแบบเครือข่ายย่อยนั้นนอกจากช่วยให้ผู้ดูแลเครือข่ายสามารถจัดการกับกลุ่มของอุปกรณ์สื่อสารหรือเครื่องคอมพิวเตอร์ในเครือข่ายได้แล้ว ยังช่วยลดขนาดของตารางการหาเส้นทาง (Routing Table) ภายในอุปกรณ์จัดเส้นทางได้ เนื่องจากเครื่องคอมพิวเตอร์ภายในเครือข่ายย่อยจะถูกมองเป็นกลุ่มเดียวกัน

สำหรับการติดต่อภายในเครือข่าย เครื่องคอมพิวเตอร์หรืออุปกรณ์สื่อสารแต่ละตัวต้องรู้ว่าหมายเลขไอพีของตัวเองมีการแบ่งจำนวนบิตเป็นหมายเลขเครือข่ายย่อยและหมายเลขเครื่องเท่าใด โดยการแยกบิตหมายเลขเครือข่ายย่อยและหมายเลขเครื่อง กล่าวคือกำหนดค่า 32 บิต โดยให้ตัวเลขหนึ่งแทนกลุ่มของหมายเลขเครือข่าย (Network ID) และหมายเลขเครือข่ายย่อย (Subnet ID) และให้ตัวเลขศูนย์แทนกลุ่มหมายเลขแม่ข่าย ดังภาพที่ 36 ซึ่งค่า 32 บิตดังกล่าวเรียกว่า ตัวพรางเครือข่ายย่อย (Subnet Mask)



ภาพที่ 36 ตัวอย่างหมายเลขตัวพรางเครือข่ายย่อยของเครือข่ายคลาส B

วิธีการเขียนหมายเลขตัวพรางเครือข่ายย่อยจะเขียนในลักษณะเดียวกับการเขียนหมายเลขไอพี คือ ใช้สัญลักษณ์ Dotted-decimal เช่นเดียวกัน จากตัวอย่างในภาพที่ 36 สามารถเขียนได้เป็น 255.255.255.0 เป็นต้น

<u>โพรโทคอลชั้นเคลื่อนย้ายข้อมูล</u>

ชั้นเคลื่อนย้ายข้อมูล (Transport Layer) ทำหน้าที่ควบคุมการไหล (Flow Control) ของข้อมูลระหว่างเครื่องคอมพิวเตอร์หรืออุปกรณ์สื่อสาร ในชั้นนี้จะประกอบด้วยโพรโทคอลที่แตกต่างกัน 2 โพรโทคอล คือ โพรโทคอล TCP (Transmission Control Protocol) และโพรโทคอล UDP (User Datagram Protocol) โพรโทคอลทั้ง 2 มีรายละเอียดดังนี้

1. โพรโทคอล TCP เป็นโพรโทคอลการรับส่งข้อมูลที่มีความน่าเชื่อถือระหว่างผู้ส่งและผู้รับ กล่าวคือ เมื่อผู้ส่งทำการส่งกลุ่มข้อมูล (Packet) ไปยังผู้รับแล้ว ผู้รับจะส่งกลุ่มข้อมูลตอบรับกลับมายังผู้ส่งเพื่อยืนยันว่าได้รับกลุ่มข้อมูลแล้ว ดังนั้นจึงเป็นการรับประกันว่ากลุ่มข้อมูลส่งไปถึงปลายทางแน่นอน ความน่าเชื่อถือของข้อมูลในชั้นนี้ทำให้โพรโทคอลชั้นที่สูงกว่า (เช่น ชั้นประยุกต์) ไม่จำเป็นต้องตรวจสอบความน่าเชื่อถือเอง ซึ่งเหมาะกับการนำไปใช้งานกับโปรแกรมประยุกต์ที่ต้องการความน่าเชื่อถือสูง

สำหรับวิธีการรับส่งข้อมูลของโพรโทคอล TCP นั้น ก่อนที่จะส่งข้อมูลโพรโทคอล TCP จะมีการสร้างเซสชัน (Session) เพื่อเชื่อมต่อกับแม่ข่ายปลายทางก่อน ซึ่งการสร้างเซสชันคือการสร้างการสนทนาอย่างเป็นรูปแบบระหว่างทั้งสองแม่ข่ายเพื่อใช้สำหรับการกู้คืนข้อมูลเมื่อเกิดข้อผิดพลาดระหว่างการรับส่งข้อมูล ขั้นตอนในการสร้างเซสชันนี้จะมีอยู่ 3 ขั้นตอนซึ่งเรียกว่าการตรวจสอบสามทาง (Three-way Handshake)

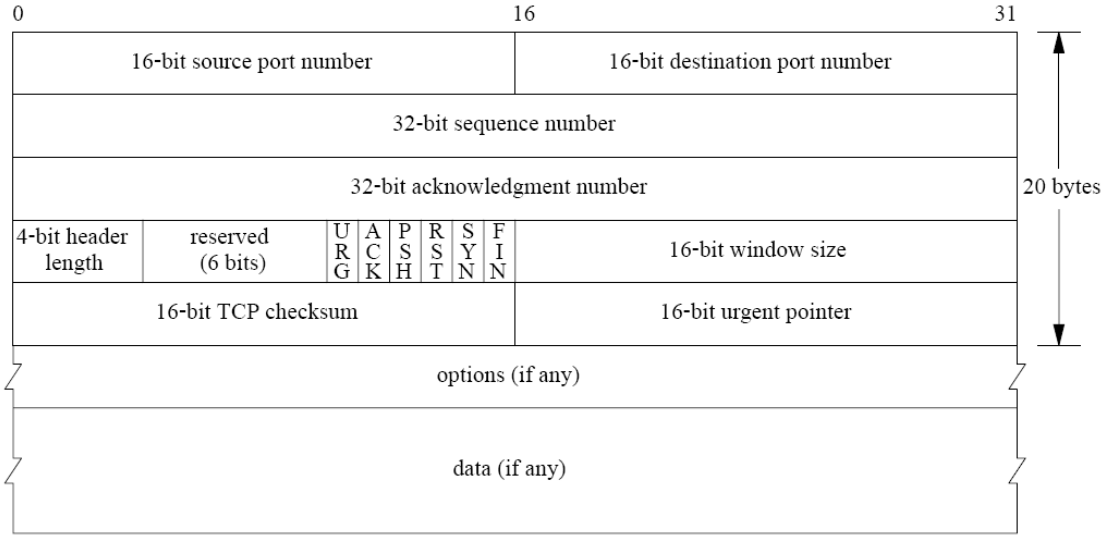
1. แม่ข่ายที่ต้องการส่งข้อมูลจะส่งกลุ่มข้อมูลไปยังแม่ข่ายปลายทางเพื่อแจ้งให้ทราบว่าต้องการส่งข้อมูล
2. แม่ข่ายปลายทางก็จะตอบตกลงกลับมาพร้อมทั้งรหัสที่จะใช้ในการรับส่งข้อมูล
3. แม่ข่ายต้นทางก็จะส่งกลุ่มข้อมูลพร้อมรหัสที่ได้รับเพื่อเป็นการยืนยันการเชื่อมต่อ

หลังจากที่ได้มีการสร้างเซสชันสำเร็จแล้วแม่ข่ายทั้งสองจึงจะเริ่มขบวนการรับส่งข้อมูลที่ต้องการ โดยที่การรับส่งข้อมูลแต่ละครั้งก็จะส่งกลุ่มข้อมูลยืนยันการรับข้อมูลจากแม่ข่ายปลายทางทุกครั้ง เมื่อรับส่งข้อมูลเสร็จก็จะทำการยกเลิกเซสชันซึ่งจะมีขั้นตอนคล้ายกับการสร้างเซสชัน

ในการรับส่งข้อมูลของโพรโทคอล TCP นั้นแม่ข่ายฝ่ายรับจะต้องตอบกลับทุก ๆ กลุ่มข้อมูลที่ได้รับภายในเวลาที่กำหนดเพื่อเป็นการยืนยันการรับข้อมูลในทุก ๆ กลุ่มข้อมูล โดยแม่ข่ายฝ่ายรับจะทำการเช็คความถูกต้องของกลุ่มข้อมูลทุกครั้งและแจ้งให้แม่ข่ายฝ่ายส่งทราบถึงผลการตรวจสอบนั้น ถ้าแม่ข่ายฝ่ายส่งไม่ได้รับการตอบรับจากฝ่ายรับภายในเวลาที่กำหนด แม่ข่ายฝ่ายส่งจะคาดเอาเองว่ากลุ่มข้อมูลสูญหายระหว่างทาง จากนั้นแม่ข่ายฝ่ายส่งก็จะทำการส่งกลุ่มข้อมูลนั้นใหม่อีกครั้ง เพื่อให้มั่นใจได้ว่าข้อมูลทุก ๆ กลุ่มข้อมูลส่งถึงปลายทางได้อย่างสมบูรณ์ นอกจากนี้การแบ่งข้อมูลขนาดใหญ่ออกเป็นกลุ่มข้อมูลย่อย ๆ โพรโทคอล TCP ก็จะกำหนดหมายเลขลำดับ (Sequence Number) ให้แต่ละกลุ่มข้อมูล เพื่อใช้สำหรับรวมกลุ่มข้อมูลย่อย ๆ เหล่านั้นให้เป็นข้อมูลเหมือนเดิม นอกจากนี้หมายเลขลำดับยังใช้สำหรับการตรวจสอบว่าข้อมูลส่งถึงปลายทางครบทุกกลุ่มข้อมูลหรือไม่

กลไกการรับส่งข้อมูลในโพรโทคอล TCP เรียกว่า การเลื่อนหน้าต่าง (Sliding Window) คือ เมื่อแม่ข่ายฝ่ายส่งทำการส่งกลุ่มข้อมูลชุดหนึ่งตามขนาดของหน้าต่าง (Window Size) หลังจากนั้นก็จะรอการตอบกลับจากแม่ข่ายฝ่ายรับ ซึ่งแม่ข่ายฝ่ายรับสามารถยืนยันการรับกลุ่มข้อมูลโดยการส่งเพียงกลุ่มข้อมูลเดียวสำหรับการยืนยันการรับหลายกลุ่มข้อมูล จากนั้นแม่ข่ายฝ่ายส่งจะเลื่อนหน้าต่างไปยังข้อมูลชุดที่ยังไม่ได้ส่งแล้วค่อยส่งกลุ่มข้อมูลต่อไป ถ้าไม่ได้รับการตอบกลับภายในเวลาที่กำหนดก็จะส่งกลุ่มข้อมูลทั้งหมดอีกครั้งหรือส่งในส่วนที่ยังไม่ได้รับกลุ่มข้อมูลตอบกลับ ซึ่งวิธีนี้จะช่วยลดจำนวนกลุ่มข้อมูลที่ต้องไหลเวียนในเครือข่ายและฝ่ายส่งสามารถส่งทีละหลาย ๆ กลุ่มข้อมูลก่อนที่จะรอการตอบรับ

ขบวนการส่งข้อมูลแบบนี้จะทำให้มั่นใจได้ว่าข้อมูลจะส่งไปถึงปลายทางอย่างแน่นอนและถูกต้อง ซึ่งการให้บริการแบบนี้เรียกว่าบริการเชิงการเชื่อมต่อ (Connection Oriented)



ภาพที่ 37 รูปแบบกลุ่มข้อมูลของโพรโทคอล TCP
ที่มา: Stevens (1994)

ข้อมูลในส่วนหัวของโพรโทคอล TCP นั้นสามารถแสดงได้ดังภาพที่ 37 ซึ่งแต่ละฟิลด์มีความหมายดังนี้

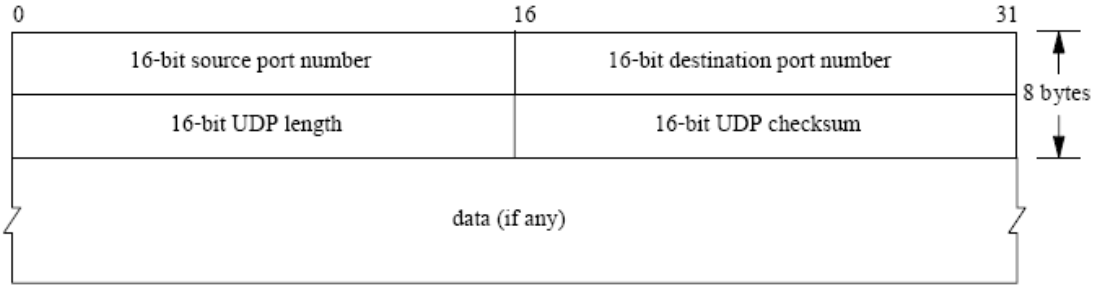
- หมายเลขพอร์ตเครื่องต้นทาง (Source Port Number) เป็นหมายเลขพอร์ตสำหรับโปรแกรมประยุกต์บนเครื่องต้นทาง
- หมายเลขพอร์ตเครื่องปลายทาง (Destination Port Number) เป็นหมายเลขพอร์ตสำหรับโปรแกรมประยุกต์บนเครื่องปลายทาง
- หมายเลขลำดับ (Sequence Number) เป็นหมายเลขที่บอกลำดับกลุ่มข้อมูลที่จะใช้โดยฝั่งรับในการเรียงข้อมูลให้อยู่ในรูปแบบเดิม ซึ่งการส่งข้อมูลผ่านเครือข่ายนั้นกลุ่มข้อมูลแต่ละชุดอาจถูกส่งไปบนเส้นทางที่ต่างกันเนื่องจากลักษณะการส่งข้อมูลของโพรโทคอลชั้นเครือข่าย ดังนั้นกลุ่มข้อมูลอาจจะไปถึงปลายทางไม่เป็นไปตามลำดับ ซึ่งหมายเลขนี้จะใช้ในการจัดเรียงกลุ่มข้อมูลให้อยู่ในลำดับเดิม
- หมายเลขตอบรับ (Acknowledgment Number) เป็นหมายเลขลำดับกลุ่มข้อมูลถัดไปที่ทางฝั่งรับคาดหวัง ซึ่งเป็นการบอกเป็นนัยว่ากลุ่มข้อมูลที่มีหมายเลขลำดับก่อนหน้านี้ได้รับหมดแล้วนั้นเอง
- ความยาวข้อมูลส่วนหัว (Header Length) เป็นส่วนที่บอกความยาวของข้อมูลส่วนหัว ซึ่งมีหน่วยเป็น 32 บิตหรือคำ (Word)

- ฟิลด์สำรอง (Reserved) ส่วนนี้จะถูกกำหนดให้มีค่าเป็นศูนย์ตลอด ซึ่งข้อมูลส่วนนี้ไม่มีความหมายอะไรเพียงแต่เป็นการสำรองไว้ใช้ในอนาคต
- แฟลก (Flag) เป็นข้อมูลสำหรับควบคุมการรับส่งกลุ่มข้อมูลหรือบอกสถานะของข้อมูลที่ใช้รับส่ง ซึ่งประกอบด้วย 6 บิต ได้แก่ URG ACK PSH RST SYN และ FIN เป็นต้น
- ขนาดหน้าต่างรับส่ง (Window Size) เป็นตัวเลขที่เครื่องปลายทางบอกให้เครื่องต้นทางทราบขนาดข้อมูลที่เครื่องปลายทางสามารถรับได้
- ค่าตรวจสอบผลรวม (Checksum) เป็นข้อมูลที่ใช้ตรวจสอบข้อผิดพลาดของข้อมูลในส่วนหัว
- ตัวชี้ความเร่งด่วน (Urgent Pointer) เป็นฟิลด์ข้อมูลที่ใช้ได้ก็ต่อเมื่อมีการตั้งค่าบิต URG ในฟิลด์แฟลกให้เป็น 1 ซึ่งค่าในตัวชี้ความเร่งด่วนจะเป็นการบอกให้เครือข่ายรับรู้ว่าเป็นข้อมูลที่ถูกส่งไปเป็นข้อมูลแบบฉุกเฉิน
- ตัวเลือก (Option) เป็นข้อมูลเพิ่มเติมที่ใช้ในการรับส่ง ซึ่งส่วนใหญ่จะใช้ในการบอกขนาดส่วนแบ่งสูงสุด (Maximum Segment Size) หรือ MSS
- ข้อมูล (Data) เป็นข้อมูลที่รับมาจากโพรโทคอลชั้นที่สูงกว่า

2. โพรโทคอล UDP เป็นบริการที่มีลักษณะตรงข้ามกับโพรโทคอล TCP ซึ่งเป็นการรับส่งข้อมูลที่เชื่อถือไม่ได้ โดยโพรโทคอลนี้จะพยายามส่งข้อมูลให้ดีที่สุด (Best Effort) ไปยังผู้รับ แต่ไม่รับประกันว่าข้อมูลจะไปถึงผู้รับหรือไม่ ดังนั้นถ้าโพรโทคอลชั้นประยุกต์ต้องการความน่าเชื่อถือจะต้องทำการตรวจสอบความน่าเชื่อถือเอง ซึ่งการส่งกลุ่มข้อมูลของโพรโทคอลนี้เรียกว่าดาต้าแกรม (Datagram)

ถึงแม้ว่าโพรโทคอล UDP จะมีความหน้าเชื่อถือน้อยแต่ก็มีข้อดีหลายอย่าง เช่น ทำให้ประสิทธิภาพการส่งข้อมูลสูงเนื่องจากไม่มีกลไกในการตรวจสอบความถูกต้องของการรับส่ง นอกจากนี้ยังสามารถใช้ในการส่งข้อมูลแบบกระจายเสียง (Broadcast) และมัลติคาสต์ (Multicast) เนื่องจากการส่งข้อมูลลักษณะนี้ไม่จำเป็นต้องรอการตอบกลับจากเครื่องปลายทาง

ในการรับส่งของโพรโทคอล UDP นั้นถ้าโปรแกรมประยุกต์ต้องการความน่าเชื่อถือของการส่งข้อมูล โปรแกรมประยุกต์นั้นจะต้องควบคุมและตรวจสอบข้อผิดพลาดเอง แต่ส่วนใหญ่โปรแกรมประยุกต์ที่ใช้โพรโทคอลนี้จะไม่ต้องการยืนยันการตอบรับ เช่น โพรโทคอลของระบบเครือข่ายของไมโครซอฟต์ ตัวอย่างเช่น การลงบันทึกเปิด (Log on) การค้นผ่าน (Browsing) และการจำแนกชื่อ (Name Resolution) เป็นต้น (จตุชัย และอนุโชต, 2546)



ภาพที่ 38 รูปแบบกลุ่มข้อมูลของโพรโทคอล UDP
ที่มา: Stevens (1994)

ข้อมูลในส่วนหัวของโพรโทคอล UDP นั้นสามารถแสดงได้ดังภาพที่ 38 ซึ่งแต่ละฟิลด์มีความหมายดังนี้

- หมายเลขพอร์ตเครื่องต้นทาง (Source Port Number) เป็นหมายเลขพอร์ตสำหรับโปรแกรมประยุกต์บนเครื่องต้นทาง
- หมายเลขพอร์ตเครื่องปลายทาง (Destination Port Number) เป็นหมายเลขพอร์ตสำหรับโปรแกรมประยุกต์บนเครื่องปลายทาง
- ค่าตรวจสอบผลรวม (Checksum) เป็นข้อมูลที่ใช้ตรวจสอบข้อผิดพลาดของข้อมูล
- ความยาวข้อความ (Message Length) เป็นความยาวของกลุ่มข้อมูลทั้งหมด โดยความยาวต่ำสุดของกลุ่มข้อมูลเท่ากับ 8 ไบต์ ซึ่งเป็นความยาวของข้อมูลส่วนหัวอย่างเดียว
- ข้อมูล (Data) เป็นข้อมูลที่รับมาจากโพรโทคอลชั้นที่สูงกว่า

การรับส่งข้อมูลในชั้นเคลื่อนย้ายข้อมูลสำหรับชุดโพรโทคอล TCP/IP จะใช้หมายเลขพอร์ตในการบอกถึงชนิดของโปรแกรมประยุกต์ในโพรโทคอลชั้นที่สูงกว่า โดยที่โปรแกรมประยุกต์จะเลือกหมายเลขพอร์ตซึ่งมีขนาด 16 บิตเอง

การเลือกหมายเลขพอร์ตใช้งานนั้นโดยทั่วไปทางด้านเครื่องบริการ (Server) จะต้องรู้ว่าโปรแกรมประยุกต์ประเภทไหนใช้พอร์ตหมายเลขอะไร ซึ่งหมายเลขพอร์ตมาตรฐานที่ถูกกำหนดค่าสำหรับโปรแกรมประยุกต์แล้วเรียกว่า Well-known Port Number ตัวอย่างเช่น เครื่องบริการ FTP จะใช้หมายเลขพอร์ต TCP เท่ากับ 21 หรือเครื่องบริการ Telnet จะใช้หมายเลขพอร์ต TCP เท่ากับ 23 หรือโปรแกรมประยุกต์ของ TFTP (Trivial File Transfer Protocol) จะใช้หมายเลข

พอร์ต UDP เท่ากับ 69 เป็นต้น โดยหมายเลขพอร์ตมาตรฐานเหล่านี้จะมีค่าอยู่ระหว่าง 1 ถึง 1023 ซึ่งกำหนดโดย IANA (Internet Assigned Numbers Authority)

สำหรับทางด้านเครื่องรับบริการ (Client) ไม่จำเป็นต้องสนใจในการเลือกใช้งานหมายเลขพอร์ต แต่โปรแกรมประยุกต์ต่าง ๆ บนแม่ข่ายเครื่องเดียวกันจะต้องเลือกใช้งานพอร์ตที่ไม่ซ้ำกัน โดยหมายเลขพอร์ตสำหรับเครื่องรับบริการเรียกว่าพอร์ต Ephemeral ซึ่งหมายความว่าเป็นพอร์ตที่ใช้ชั่วคราวในการรับส่งข้อมูลเท่านั้น ขณะที่พอร์ตในเครื่องบริการจะใช้งานตลอดเวลา ในการใช้งานจริงหมายเลขพอร์ต Ephemeral จะอยู่ระหว่าง 1024 ถึง 5000 ส่วนหมายเลขพอร์ตตั้งแต่ 5001 ขึ้นไปจะใช้สำหรับโปรแกรมประยุกต์บนเครื่องบริการที่ไม่ได้กำหนดเป็นหมายเลขพอร์ตมาตรฐานไว้ (Stevens, 1994)

อุปกรณ์และวิธีการ

อุปกรณ์

1. เครื่องคอมพิวเตอร์ส่วนบุคคล
2. ชุดเครื่องมือรวมสำหรับการเขียนโปรแกรม (Code Composer Studio) รุ่นที่ 2.0
3. ชุดเขียนโปรแกรมบนตัวประมวลผลเครือข่าย Unity
4. โปรแกรมออกแบบลายวงจร PowerPCB
5. สายดาวน์โหลดข้อมูล JTAG Emulator ของบริษัท Digital Spectrum Incorporate
6. ฮาร์ดแวร์ประมวลสัญญาณดิจิทัลพัฒนาโดยหน่วยวิจัยระบบสื่อสารแบบใช้สาย (Wireline Communication Section) ซึ่งเป็นหน่วยงานวิจัยของ NECTEC
7. ส่วนประกอบฮาร์ดแวร์สำหรับพัฒนาฮาร์ดแวร์จัดการเครือข่าย เช่น ชิปตัวประมวลผลเครือข่าย ชิปหน่วยความจำ เป็นต้น
8. ชุดเครื่องมือลงอุปกรณ์ฮาร์ดแวร์ เช่น เครื่องบัดกรี
9. เครื่องมือวัดต่าง ๆ เช่น ออสซิลโลสโคป และดิจิทัลมัลติมิเตอร์

วิธีการ

แผนการทำวิจัย

ในการออกแบบและพัฒนาระบบปฏิบัติการไบโอเมตริกสำหรับสื่อสารผ่านเครือข่าย สามารถแบ่งการพัฒนาได้หลายส่วน ซึ่งกำหนดเป็นแผนการวิจัยไว้ดังต่อไปนี้

1. ออกแบบระบบปฏิบัติการสำหรับไบโอเมตริก เป็นขั้นตอนการออกแบบซอฟต์แวร์สำหรับไบโอเมตริกบนฮาร์ดแวร์ประมวลผลสัญญาณดิจิทัล โดยซอฟต์แวร์จะมีลักษณะเป็นระบบปฏิบัติการซึ่งทำหน้าที่จัดการการใช้ทรัพยากรณ์ต่าง ๆ ภายในระบบ นอกจากนี้ในการออกแบบซอฟต์แวร์ดังกล่างยังมีการพัฒนารูปแบบการเขียนโปรแกรมให้เข้ากับมาตรฐานขั้นตอนวิธีบนตัวประมวลผลสัญญาณดิจิทัลและมาตรฐานส่วนต่อประสานโปรแกรมประยุกต์อีกด้วย

2. ออกแบบฮาร์ดแวร์เชื่อมต่อเครือข่าย เป็นขั้นตอนการออกแบบฮาร์ดแวร์สำหรับเชื่อมต่อระหว่างฮาร์ดแวร์ประมวลสัญญาณดิจิทัลและระบบเครือข่าย

3. ทดสอบฮาร์ดแวร์เชื่อมต่อเครือข่าย เป็นการเขียนโปรแกรมควบคุมบนฮาร์ดแวร์ประมวลผลเครือข่าย เพื่อทดสอบการทำงานในการเชื่อมต่อกับระบบเครือข่าย

4. ทดสอบประสิทธิภาพการทำงานของระบบ เป็นขั้นตอนการรวมฮาร์ดแวร์ประมวลสัญญาณดิจิทัลและฮาร์ดแวร์เชื่อมต่อเครือข่าย เพื่อทดสอบประสิทธิภาพการทำงานโดยรวมของระบบทั้งหมด

รายละเอียดในแต่ละขั้นตอนสามารถอธิบายในหัวข้อถัดไป

<u>การออกแบบระบบปฏิบัติการสำหรับไบโอเมตริก</u>

ในการออกแบบและพัฒนาระบบปฏิบัติการบนฮาร์ดแวร์ประมวลผลสัญญาณดิจิทัล สามารถแบ่งการทำงานได้หลายประเภท เช่น การประมวลผลภาพลายนิ้วมือ การติดต่อผู้ใช้งาน การอ่านค่าลายนิ้วมือ การควบคุมการเปิดประตู การจัดการฐานข้อมูล และการสื่อสารผ่านพอร์ตอนุกรม RS232 เป็นต้น ซึ่งการทำงานในแต่ละส่วนจะค่อนข้างเป็นอิสระต่อกัน ทำให้สามารถออกแบบส่วนต่าง ๆ ได้ง่าย การออกแบบระบบปฏิบัติการบนฮาร์ดแวร์ประมวลผลสัญญาณดิจิทัลประกอบด้วยต่าง ๆ ของดังนี้

1. <u>การจัดการหน่วยความจำ</u>

การเขียนโปรแกรมบนตัวประมวลผลสัญญาณดิจิทัลในขั้นแรกนั้นจะต้องมีการกำหนดพื้นที่หน่วยความจำให้เหมาะสมกับงานแต่ละประเภทก่อน ซึ่งการจัดการเกี่ยวกับพื้นที่หน่วยความจำนั้นบริษัท Texas Instruments ได้เตรียมเครื่องมือ DSP/BIOS สำหรับจัดการพื้นที่ใช้งานหน่วยความจำ ในหัวข้อนี้จะกล่าวถึงหน่วยความจำและวิธีจัดการหน่วยความจำบนตัวประมวลผลสัญญาณดิจิทัลสำหรับโปรแกรมของระบบไบโอเมตริก ซึ่งจะกล่าวในรายละเอียดของส่วนต่าง ๆ ดังต่อไปนี้

1.1 การจัดการหน่วยความจำบนตัวประมวลผลสัญญาณดิจิทัล

หน่วยความจำในตัวประมวลผลสัญญาณดิจิทัลตระกูล TMS320C6000 ใช้สำหรับเก็บโปรแกรมและข้อมูลในการทำงานของระบบ รวมทั้งใช้ควบคุมส่วนต่าง ๆ ของระบบโดยกำหนดเป็นรีจิสเตอร์ควบคุม (Control Register) ทำหน้าที่ควบคุมส่วนต่าง ๆ ซึ่งตัวประมวลผลสัญญาณดิจิทัลได้จัดแบ่งเป็นส่วนต่าง ๆ ดังตารางที่ 4

ตารางที่ 4 การแบ่งพื้นที่หน่วยความจำบนตัวประมวลผลสัญญาณดิจิทัล

MEMORY BLOCK DESCRIPTION	BLOCK SIZE (BYTES)	HEX ADDRESS RANGE
Internal RAM (L2)	64K	0000 0000 – 0000 FFFF
Reserved	24M – 64K	0001 0000 – 017F FFFF
External Memory Interface (EMIF) Registers	256K	0180 0000 – 0183 FFFF
L2 Registers	256K	0184 0000 – 0187 FFFF
HPI Registers	256K	0188 0000 – 018B FFFF
McBSP 0 Registers	256K	018C 0000 – 018F FFFF
McBSP 1 Registers	256K	0190 0000 – 0193 FFFF
Timer 0 Registers	256K	0194 0000 – 0197 FFFF
Timer 1 Registers	256K	0198 0000 – 019B FFFF
Interrupt Selector Registers	512	019C 0000 – 019C 01FF
Device Configuration Registers [C6711C/C6711D only]	4	019C 0200 – 019C 0203
Reserved	256K – 516	019C 0204 – 019F FFFF
EDMA RAM and EDMA Registers	256K	01A0 0000 – 01A3 FFFF
Reserved	768K	01A4 0000 – 01AF FFFF
GPIO Registers [C6711C/C6711D only]	16K	01B0 0000 – 01B0 3FFF
Reserved	480K	01B0 4000 – 01B7 BFFF
PLL Controller Registers [C6711C/C6711D only]	8K	01B7 C000 – 01B7 DFFF
Reserved	4M + 520K	01B7 E000 – 01FF FFFF
QDMA Registers	52	0200 0000 – 0200 0033
Reserved	736M – 52	0200 0034 – 2FFF FFFF
McBSP 0 Data/Peripheral Data Bus	64M	3000 0000 – 33FF FFFF
McBSP 1 Data/Peripheral Data Bus	64M	3400 0000 – 37FF FFFF
Reserved	64M	3800 0000 – 3BFF FFFF
Reserved	1G + 64M	3C00 0000 – 7FFF FFFF
EMIF CE0†	256M	8000 0000 – 8FFF FFFF
EMIF CE1†	256M	9000 0000 – 9FFF FFFF
EMIF CE2†	256M	A000 0000 – AFFF FFFF
EMIF CE3†	256M	B000 0000 – BFFF FFFF
Reserved	1G	C000 0000 – FFFF FFFF

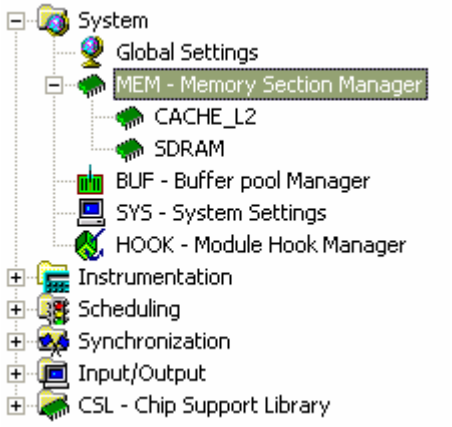
† The number of EMIF address pins (EA[21:2]) limits the maximum addressable memory (SDRAM) to 128MB per CE space. To get 256MB of addressable memory, additional general-purpose output pin or external logic is required.

ที่มา: Texas Instruments Inc. (2001)

จากตารางที่ 4 ค่าตำแหน่งของหน่วยความจำจะแสดงด้วยตัวเลขฐาน 16 โดยเริ่มจากตำแหน่ง 0 ซึ่งเป็นพื้นที่ของหน่วยความจำภายในสามารถใช้เป็นหน่วยความจำสำหรับเก็บโปรแกรมหรือข้อมูลได้หรือใช้เป็น Cache ระดับสอง เพื่อเพิ่มความเร็วในการเข้าถึงข้อมูล ในส่วนต่อมาตั้งแต่ตำแหน่ง 0x01800000 ถึง 0x37FFFFF เป็นตำแหน่งที่ใช้สำหรับเข้าถึงรีจิสเตอร์

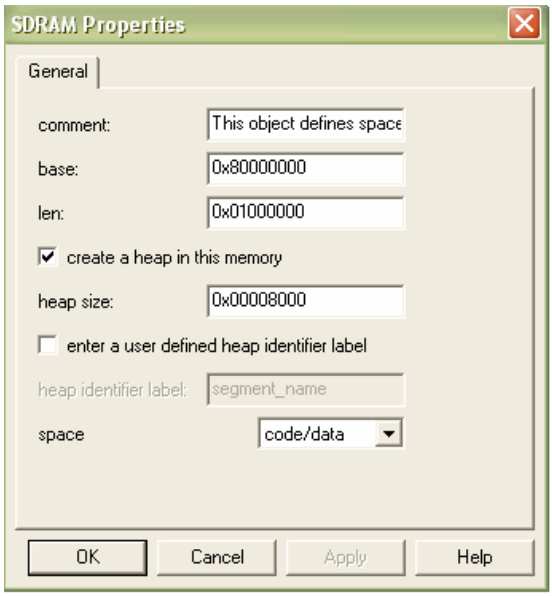
ควบคุมแบบต่าง ๆ และสุดท้ายตั้งแต่ตำแหน่ง 0x80000000 ขึ้นไปจะถูกจัดให้เป็นพื้นที่ของหน่วยความจำภายนอก

สำหรับการเขียนโปรแกรมโดยใช้ไบออสบนตัวประมวลผลสัญญาณดิจิทัลนั้นชุดเครื่องมือรวมสำหรับเขียนโปรแกรม (Code Composer Studio) ได้เตรียมเครื่องมือในการออกแบบและจัดการกับหน่วยความจำของระบบ ดังภาพที่ 39



ภาพที่ 39 ตัวอย่างเครื่องมือ DSP/BIOS

จากภาพที่ 39 มอดูล MEM หรือตัวจัดการสัดส่วนหน่วยความจำ (Memory Section Manager) ของเครื่องมือ DSP/BIOS เป็นส่วนที่ใช้กำหนดหน่วยความจำของโปรแกรมทั้งหมดที่เขียนขึ้น โดยที่ผู้เขียนโปรแกรมสามารถกำหนดตำแหน่งและขนาดของหน่วยความจำได้ดังภาพที่ 40 แต่ต้องสอดคล้องกับการแบ่งหน่วยความจำของระบบที่แสดงในตารางที่ 4



ภาพที่ 40 การกำหนดคุณสมบัติของหน่วยความจำ

1.2 มอดูล MEM บนตัวประมวลผลสัญญาณดิจิทัล

มอดูล MEM ได้เตรียมฟังก์ชันสำหรับกำหนดหรือจองหน่วยความจำได้โดยใช้เครื่องมือ DSP/BIOS โดยที่ขนาดการจองหน่วยความจำนั้นจะมีหน่วยเป็น MADUs (Minimum Addressable Data Units) ซึ่งเป็นหน่วยเล็กที่สุดของข้อมูลในหน่วยความจำที่ตัวประมวลผลจะสามารถอ่านหรือเขียนได้ โดยที่ตัวจัดการสัดส่วนหน่วยความจำ (Memory Section Manager) นั้นจะจัดเก็บข้อมูลที่ได้จากการแปลโปรแกรม (Compile) ด้วยชุดเครื่องมือรวมสำหรับเขียนโปรแกรม (Code Composer Studio) ลงไปในหน่วยความจำที่กำหนดขึ้นมา ซึ่งการจัดเก็บข้อมูลหน่วยความจำเหล่านี้มีความสำคัญมากในการทำงานของโปรแกรม เช่น ข้อมูลที่มีการเข้าถึงบ่อย ๆ ควรจัดให้อยู่ในบริเวณของหน่วยความจำภายใน แต่ถ้ามีข้อมูลโปรแกรมของระบบที่ใช้งานในบางครั้งก็ให้อยู่ในบริเวณหน่วยความจำภายนอกแบบชั่วคราว หรือถ้าข้อมูลมีความสำคัญมากและต้องการบันทึกค่าเก็บไว้แม้ไฟดับก็ควรจัดให้อยู่ในบริเวณหน่วยความจำภายนอกแบบถาวร (External Flash ROM) เป็นต้น

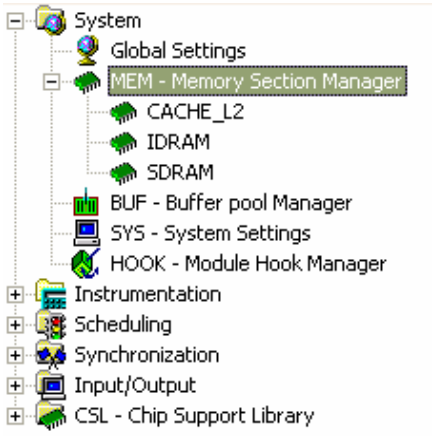
ในขณะที่แปลโปรแกรม (Compile) ด้วยชุดเครื่องมือรวมสำหรับเขียนโปรแกรมนั้น โปรแกรมทั้งหมดที่เขียนขึ้นมาจะถูกแบ่งเป็นส่วนประกอบโปรแกรม (Segment) เพื่อนำไปจัดลงบนหน่วยความจำของตัวประมวลผล ในที่นี้จะกล่าวถึงส่วนประกอบโปรแกรมที่สำคัญ ได้แก่

- .arg เป็นส่วนสำหรับเก็บค่าอาร์กิวเมนต์ที่ใช้ในโปรแกรม
- .stack เป็นส่วนสำหรับเก็บค่าทับซ้อนส่วนกลาง (Global stack) ของโปรแกรม
- .bios เป็นส่วนสำหรับเก็บโปรแกรม DSP/BIOS ที่ใช้สำหรับตัวประมวลผล
- .text เป็นส่วนสำหรับเก็บโปรแกรมที่เขียนขึ้นมาหลังจากแปลโปรแกรมแล้ว
- .switch เป็นส่วนสำหรับเก็บข้อมูลที่เกี่ยวกับการกระโดด (Jump) ไปทำงานในโปรแกรมส่วนอื่น
- .bss เป็นส่วนสำหรับเก็บค่าตัวแปรส่วนกลาง (Global Variable) ในโปรแกรม
- .const เป็นส่วนสำหรับเก็บค่าคงที่ต่าง ๆ ในโปรแกรม
- .data เป็นส่วนสำหรับเก็บข้อมูลชั่วคราวในโปรแกรม

1.3 การออกแบบการจัดการหน่วยความจำสำหรับระบบไบโอเมตริก

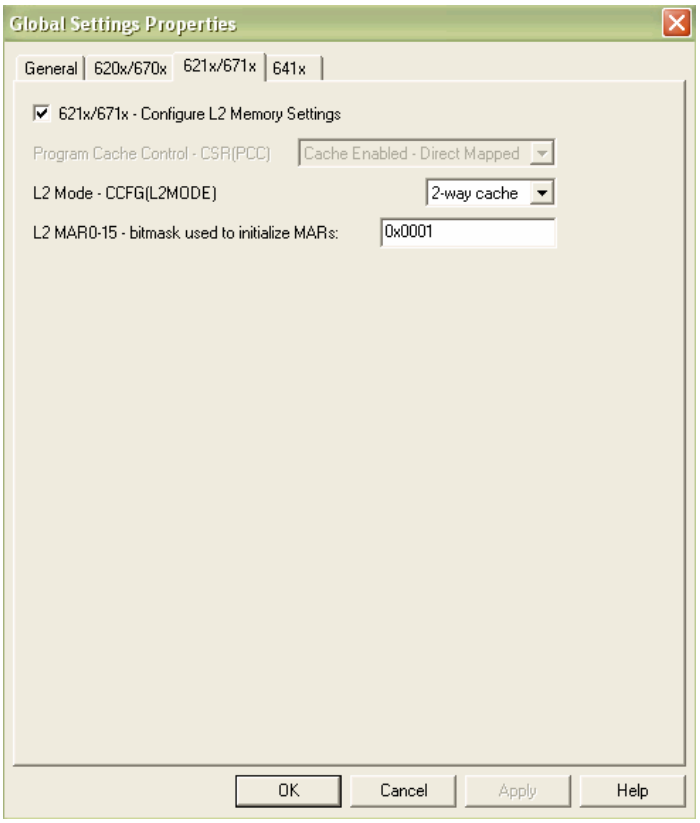
การออกแบบโปรแกรมในส่วนของการจัดการหน่วยภายจำของระบบ ในที่นี้จะกล่าวเฉพาะการเขียนโปรแกรมบนชุดประมวลผลสัญญาณดิจิทัล TMS320C6713 ซึ่งมีหน่วยความจำภายนอกเป็น SDRAM ขนาด 16 MB และใช้ DSP/BIOS ในการจัดการหน่วยความจำของระบบ

การออกแบบหน่วยความจำบนชุดประมวลผลสัญญาณดิจิทัลสำหรับระบบไบโอเมตริกนั้น จะทำการแบ่งพื้นที่หน่วยความจำใช้งานเป็น 3 ส่วน ดังภาพที่ 41



ภาพที่ 41 การกำหนดค่า DSP/BIOS สำหรับระบบไบโอเมตริก

จากภาพที่ 41 ในส่วนของตัวจัดการสัดส่วนหน่วยความจำ (Memory Section Manager) จะแบ่งหน่วยความจำที่ออกเป็น 3 ส่วนได้แก่ CACHE_L2 เป็นหน่วยความจำภายในซึ่งทำหน้าที่เป็น Cache ระดับที่ 2 เพื่อเพิ่มประสิทธิภาพในการประมวลผลของโปรแกรม ต่อมา IDRAM เป็นพื้นที่หน่วยความจำภายในสำหรับเก็บข้อมูล ใช้สำหรับข้อมูลที่ต้องใช้ความเร็วสูงในการเข้าถึง สุดท้าย SDRAM เป็นหน่วยความจำภายนอกของระบบ ใช้ในการเก็บข้อมูลโปรแกรมหรือส่วนประกอบโปรแกรม (Segment) ที่มีขนาดใหญ่ เป็นต้น โดยที่การกำหนดค่าต่าง ๆ ของหน่วยความจำ IDRAM และ SDRAM สามารถทำได้เช่นเดียวกับภาพที่ 40 ส่วนการตั้งค่าภายในส่วน CACHE_L2 นั้นไม่สามารถกำหนดได้โดยตรง แต่สามารถตั้งค่าได้ผ่านทางการตั้งค่าส่วนกลาง (Global Settings) ดังภาพที่ 42

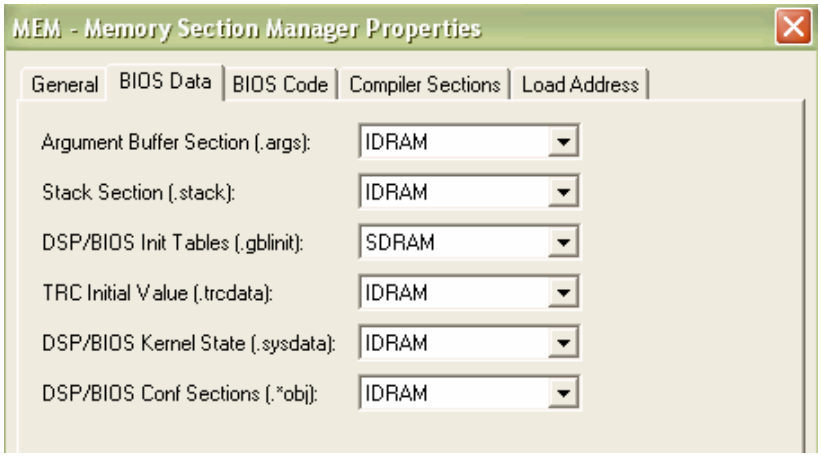


ภาพที่ 42 การกำหนดค่าในโปรแกรมตั้งค่าส่วนกลาง (Global Settings)

จากภาพที่ 42 ขนาดของ CACHE_L2 ขึ้นอยู่กับการตั้งค่า L2 Mode – CCFG (L2MODE) ซึ่งเป็นการเลือกลักษณะการทำงานของ Cache ได้แก่ Direct Mapping, 2-way Set, 3-way Set, 4-way Set หรือไม่ใช้ระบบ Cache เป็นต้น ตัวอย่างเช่น ถ้ามีการกำหนดค่า L2 Mode

เป็น 2-way Set Associative จะทำให้ CACHE_L2 มีขนาด 0x8000 และมีตำแหน่งเริ่มต้นที่ 0x8000 เป็นต้น ซึ่งขนาดและตำแหน่งจะถูกกำหนดขึ้นเองอัตโนมัติ

หลังจากแบ่งพื้นที่หน่วยความจำแล้วต่อไปก็จะทำการระบุส่วนประกอบโปรแกรม (Segment) ต่าง ๆ ลงไปยังพื้นที่หน่วยความจำที่แบ่งไว้ผ่านทางตัวจัดการสัดส่วนหน่วยความจำ (Memory Section Manager) ดังภาพที่ 43 ซึ่งหลักการระบุตำแหน่งของส่วนประกอบโปรแกรมนั้นควรคำนึงถึงความเร็วและขนาดของส่วนประกอบโปรแกรมนั้น



ภาพที่ 43 การกำหนดค่าในมอดูล MEM

จากที่กล่าวมาในหัวข้อนี้การเขียนโปรแกรมบนตัวประมวลผลสัญญาณดิจิทัลโดยใช้ DSP/BIOS สิ่งที่ต้องพิจารณาเป็นอันดับแรก ๆ นั้น คือ การจัดสรรพื้นที่ของหน่วยความจำของระบบ เพื่อเพิ่มประสิทธิภาพและป้องกันความผิดพลาดในการทำงานของระบบซึ่งการกำหนดค่าโดยใช้ DSP/BIOS นั้นสามารถทำได้ผ่านมอดูล MEM และการตั้งค่าส่วนกลาง (Global Settings) ของระบบ

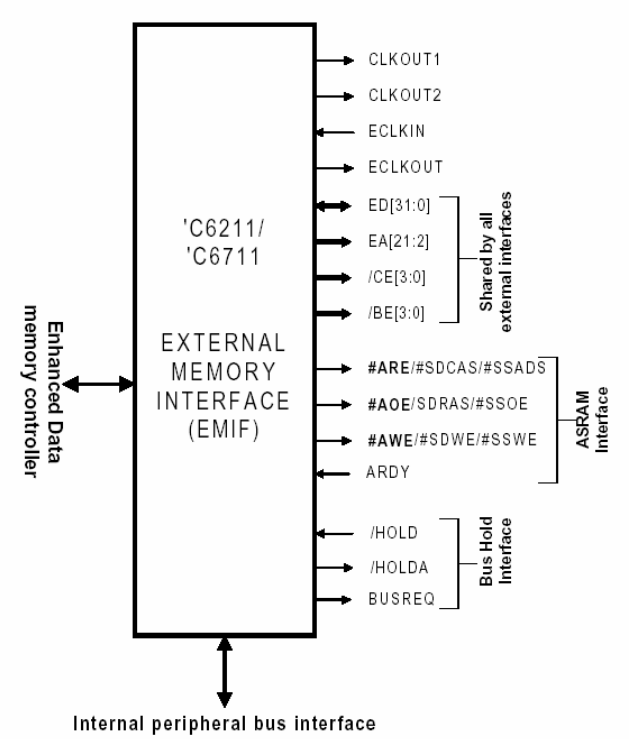
2. การติดต่อตัวกราดตรวจลายนิ้วมือ

การติดต่อกับอุปกรณ์ภายนอกแบบขนานของตัวประมวลผลสัญญาณดิจิทัลตระกูล TMS320C6000 นั้น สามารถทำได้โดยใช้มอดูล EMIF ลักษณะวิธีการสำหรับการติดต่อกับอุปกรณ์ต่าง ๆ นั้นมีอยู่หลายรูปแบบ เช่น การติดต่อแบบ ASRAM SBSRAM SDRAM หรือ FLASH ROM ขึ้นอยู่กับอุปกรณ์ที่นำมาเชื่อมต่อ สำหรับการติดต่อกับชุดกราดตรวจภาพลายนิ้วมือ จะใช้วิธีการ

เข้าถึงแบบ ASRAM (Asynchronous SRAM) ซึ่งในหัวข้อนี้จะกล่าวถึงวิธีการออกแบบและเขียนโปรแกรมทางด้านซอฟต์แวร์บนตัวประมวลผลสัญญาณดิจิทัลสำหรับการติดต่อกับชุดกราดตรวจภาพลายนิ้วมือ รวมถึงวิธีการเชื่อมต่อทางด้านฮาร์ดแวร์ด้วย ซึ่งจะกล่าวในรายละเอียดของส่วนต่าง ๆ ดังต่อไปนี้

2.1 การติดต่อตัวกราดตรวจลายนิ้วมือ

การติดต่อกับอุปกรณ์ภายนอกแบบไม่เข้าจังหวะของ EMIF นั้นเป็นลักษณะการเข้าถึงข้อมูลแบบไม่ต้องรอการเข้าจังหวะกับสัญญาณนาฬิกา โดยทั่วไปนิยมใช้ในการเข้าถึงข้อมูลของ SRAM ซึ่งขาสัญญาณทั้งหมดในส่วนของมอดูล EMIF นั้น สามารถแสดงได้ดังภาพที่ 44



ภาพที่ 44 แผนภาพของมอดูล EMIF

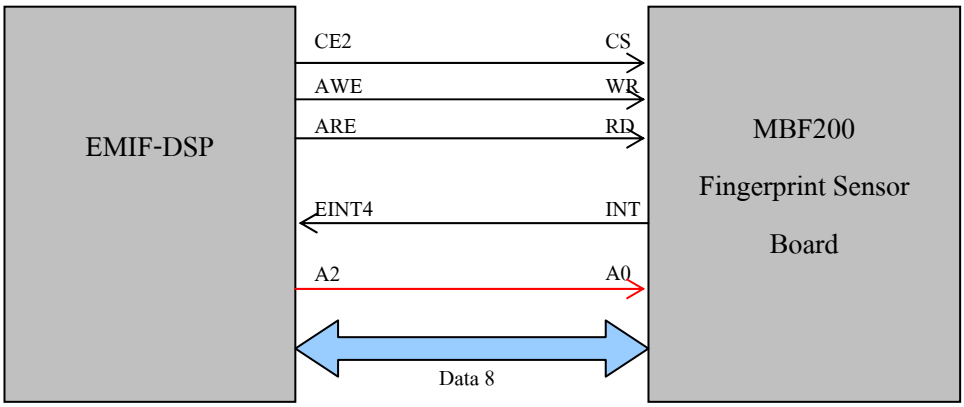
ที่มา: Texas Instruments Inc. (2001)

จากภาพที่ 44 มอดูล EMIF ได้เตรียมขาสัญญาณต่าง ๆ ไว้เพื่อให้สามารถเข้าถึงอุปกรณ์ได้หลายประเภท การใช้มอดูล EMIF ในการเข้าถึงข้อมูลแบบ ASRAM จะใช้ขาสัญญาณในส่วนของ ASRAM เท่านั้น ซึ่งได้แก่ ขาสัญญาณข้อมูล (Data Bus, ED[31:0]) ขาสัญญาณตำแหน่ง (Address Bus, EA[21:2], CE[3:0], BE[3:0]) และขาสัญญาณควบคุม (#ARE, #AOE, #AWE and ARDY) เป็นต้น

2.2 การติดต่อกับตัวกราดตรวจลายนิ้วมือโดยใช้มอดูล EMIF

ในหัวข้อนี้จะกล่าวถึงการเชื่อมต่อทางด้านฮาร์ดแวร์ของชุดกราดตรวจลายนิ้วมือและ EMIF ของตัวประมวลผลสัญญาณดิจิทัล โดยที่ตัวกราดตรวจลายนิ้วมือเป็นวงจรรวมของ Fujitsu เบอร์ MBF200 ซึ่งเป็นตัวกราดตรวจลายนิ้วมือแบบวัดค่าประจุ (Capacitive) และมีการเข้าถึงของมูลแบบขนานและสามารถเชื่อมต่อกับมอดูล EMIF ได้

สำหรับการทำงานของชุดกราดตรวจลายนิ้วมือนั้น ชุดกราดตรวจลายนิ้วมือจะส่งสัญญาณขัดจังหวะ (Interrupt Signal) ไปยังตัวประมวลผลเพื่อบอกว่ามีการวางนิ้วมือแล้ว ดังนั้นส่วนของการเชื่อมต่อจะต้องเพิ่มส่วนของการรับสัญญาณขัดจังหวะด้วย จากนั้นตัวประมวลผลจะอ่านค่าลายนิ้วมือผ่านยังส่วนของมอดูล EMIF ซึ่งลักษณะการเชื่อมต่อสัญญาณระหว่างชุดกราดตรวจลายนิ้วมือและตัวประมวลผลสัญญาณดิจิทัลสามารถแสดงได้ดังภาพที่ 45



ภาพที่ 45 การติดต่อกับตัวกราดตรวจลายนิ้วมือโดยใช้มอดูล EMIF

จากภาพที่ 45 นั้น EMIF จะใช้ขาสัญญาณข้อมูล 8 บิตในการรับและส่งข้อมูล ส่วนขาสัญญาณ A2 จะใช้ในการบอกให้ตัวกราดตรวจลายนิ้วมือให้รับรู้ว่าตัวประมวลผลต้องการอ่านหรือเขียนข้อมูล (ใช้เฉพาะในกรณีของตัวกราดตรวจลายนิ้วมือเบอร์ MBF200) ส่วนสัญญาณ CE2 AWE และ ARE ใช้สำหรับควบคุมการอ่านหรือเขียนข้อมูลบนตัวกราดตรวจลายนิ้วมือ และสุดท้ายคือสัญญาณ EINT4 เป็นสัญญาณขัดจังหวะของตัวประมวลผลเพื่อให้รู้ว่ามีการวางนิ้วมือแล้วนั้นเอง

2.3 มอดูล EMIF บนตัวประมวลผลสัญญาณดิจิทัลและการปรับแต่ง

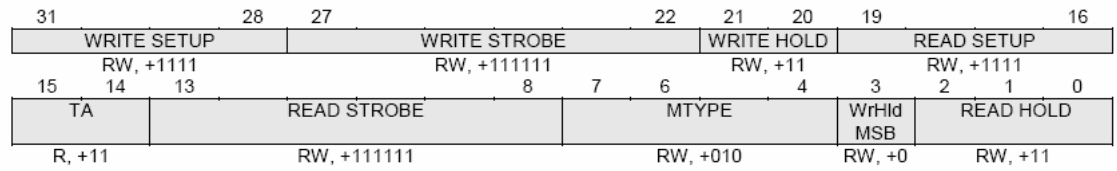
ในการปรับแต่งคุณลักษณะของมอดูล EMIF นั้นตัวประมวลผลสัญญาณดิจิทัลได้เตรียมรีจิสเตอร์ควบคุม (Control Register) ต่าง ๆ ไว้ ซึ่งใช้ในการปรับแต่งความกว้างของสัญญาณที่ออกมา โดยที่ตำแหน่งของรีจิสเตอร์สำหรับการเข้าถึงข้อมูลแบบไม่เข้าจังหวะของ ASRAM (Asynchronous Static RAM) นั้น จะแสดงได้ดังตารางที่ 5

ตารางที่ 5 รีจิสเตอร์ EMIF ที่ตำแหน่งต่าง ๆ ของหน่วยความจำ

Byte Address	Name
0x0180 0000	EMIF global control
0x0180 0004	EMIF CE1 space control
0x0180 0008	EMIF CE0 space control
0x0180 000C	Reserved
0x0180 0010	EMIF CE2 space control
0x0180 0014	EMIF CE3 space control

จากตารางจะเห็นว่าพื้นที่ของหน่วยความจำสำหรับ EMIF จะแบ่งเป็น 4 ส่วน คือ ส่วนของ CE0 CE1 CE2 และ CE3 เพื่อเพิ่มความยืดหยุ่นของตัวประมวลผลในกรณีที่มีการต่อกับอุปกรณ์หลายประเภทที่มีลักษณะการเข้าถึงข้อมูลที่แตกต่างกัน สำหรับรีจิสเตอร์ EMIF Global Control เป็นรีจิสเตอร์ควบคุมสัญญาณนาฬิกาและขาสัญญาณอื่น ๆ ซึ่งการเข้าถึงข้อมูลแบบ ASRAM ไม่จำเป็นต้องใช้ในการทำงาน

สำหรับรีจิสเตอร์ CEx Spece Control โดยที่ x มีค่าเป็น 0 1 2 หรือ 3 นั้นสามารถแสดงได้ดังภาพที่ 46



ภาพที่ 46 รีจิสเตอร์ CEx Space Control สำหรับ EMIF

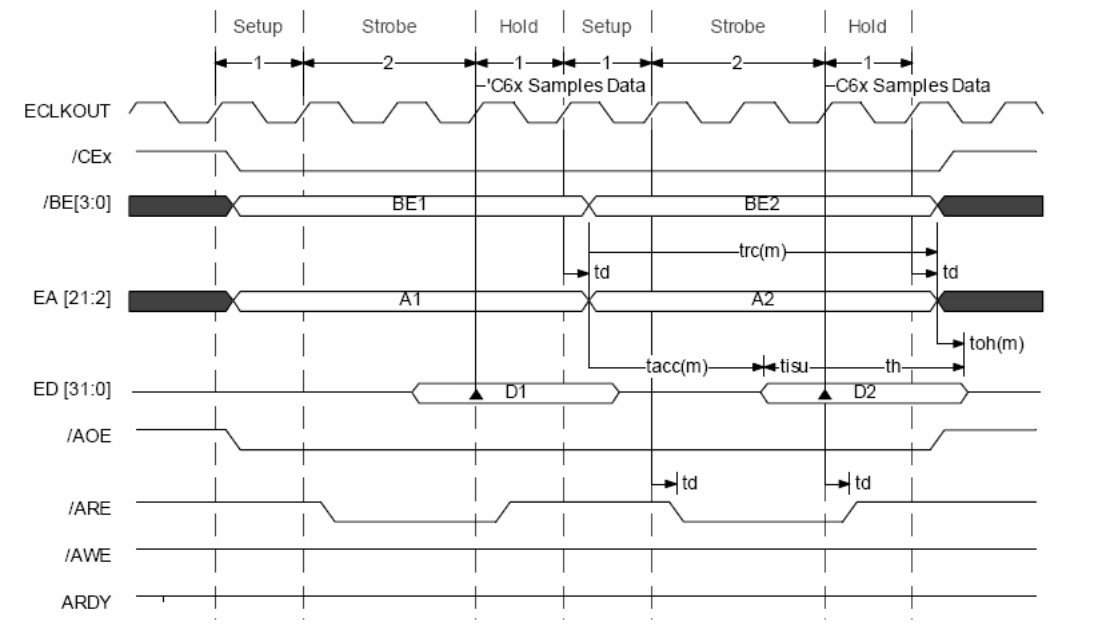
ที่มา: Texas Instruments Inc. (2001)

จากภาพที่ 46 ภายในรีจิสเตอร์จะประกอบด้วยเขตข้อมูลควบคุม (Control Field) ต่าง ๆ ที่ใช้กำหนดค่าสำหรับควบคุมคุณลักษณะของมอดูล EMIF ซึ่งเขตข้อมูลควบคุมของรีจิสเตอร์ CEx Space Control ประกอบด้วย MTYPE ใช้สำหรับเลือกวิธีการเข้าถึงข้อมูลกับอุปกรณ์แบบต่าง ๆ TA ใช้กำหนดจำนวนคาบสัญญาณนาฬิการะหว่างการอ่านและเขียนข้อมูล หรือระหว่างการอ่านข้อมูล ส่วนเขตข้อมูลควบคุมอื่น ๆ เป็นการกำหนดความกว้างของรูปแบบสัญญาณสำหรับอ่านหรือเขียนข้อมูล โดยกำหนดเทียบกับจำนวนของสัญญาณนาฬิกาของมอดูล EMIF

การโปรแกรมค่าพารามิเตอร์สำหรับการเข้าถึงข้อมูลแบบไม่เข้าจังหวะนั้นสามารถทำได้โดยการกำหนดค่า (Configuration) ลงในรีจิสเตอร์ควบคุมในส่วนของมอดูล EMIF ดังที่กล่าวมาแล้วในข้างต้น ซึ่งค่าพารามิเตอร์เหล่านั้นสามารถแบ่งได้เป็น 2 กลุ่ม คือ กลุ่มสำหรับการอ่านและเขียนซึ่งจะมีพารามิเตอร์เหมือนกัน โดยที่พารามิเตอร์สำหรับการอ่านและเขียนข้อมูลแบบไม่เข้าจังหวะ ได้แก่

- SETUP เป็นเวลาในช่วงแรกของการเข้าถึงข้อมูลจนถึงสัญญาณอ่านหรือเขียนเริ่มทำงาน
- STROBE เป็นเวลาที่สัญญาณอ่านหรือเขียนเริ่มทำงานจนถึงสิ้นสุด
- HOLD เป็นเวลาหลังจากสัญญาณอ่านหรือเขียนทำงานเสร็จแล้วจนกระทั่งสิ้นสุดการเข้าถึงข้อมูล

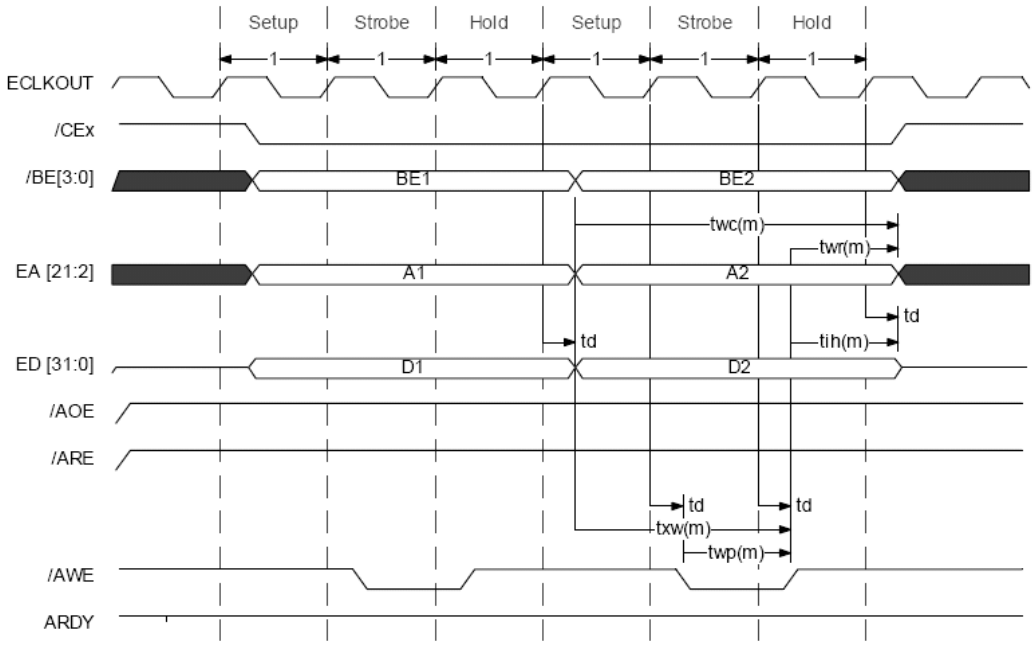
สำหรับตัวประมวลผลสัญญาณดิจิทัลตระกูล TMS320C67x นั้นค่าพารามิเตอร์ที่กำหนดลงไปจะมีหน่วยเป็นจำนวนเท่าของคาบสัญญาณนาฬิกา (ECLKOUT Cycles) นอกจากนี้ยังมีพารามิเตอร์ Turn-around (TA) ซึ่งเป็นเวลาระหว่างการอ่านและการเขียนข้อมูล หรือระหว่างการอ่านข้อมูลเพื่อเพิ่มความหน่วงของเข้าถึงข้อมูลได้ โดยที่รูปแบบการอ่านและเขียนข้อมูลของมอดูล EMIF สามารถเขียนแผนภาพเวลาได้ดังภาพที่ 47 และ 48



ภาพที่ 47 แผนภาพเวลาในการอ่านข้อมูลแบบไม่เข้าจังหวะบน TMS320C6713

ที่มา: Texas Instruments Inc. (2001)

จากภาพที่ 47 เป็นตัวอย่างการอ่านข้อมูลของตัวประมวลผลโดยมีการกำหนดค่าของ SETUP/STROBE/HOLD เป็น 1/2/1 โดยที่ในช่วงแรก (Setup Time) สัญญาณ CE AOE BE และ EA เริ่มทำงาน ต่อมาในช่วงที่สอง (Strobe Time) สัญญาณ ARE จะทำงานจนกระทั่งสิ้นสุดช่วงที่ 2 ที่จุดเริ่มต้นของช่วงสุดท้าย (Hold Time) จะมีการอ่านข้อมูล และที่จุดปลายของช่วงสุดท้ายสัญญาณต่าง ๆ ก็จะสิ้นสุดการทำงาน



ภาพที่ 48 แผนภาพเวลาในการเขียนข้อมูลแบบไม่เข้าจังหวะบน TMS320C6713
ที่มา: Texas Instruments Inc. (2001)

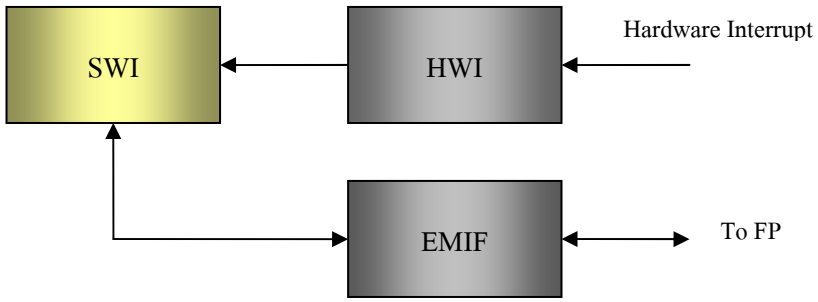
จากภาพที่ 48 เป็นตัวอย่างการเขียนข้อมูลของตัวประมวลผลโดยมีการกำหนดค่าของ SETUP/STROBE/HOLD เป็น 1/1/1 โดยที่ในช่วงแรก (Setup Time) สัญญาณ CE BE EA และ ED เริ่มทำงาน ต่อมาในช่วงที่สอง (Strobe Time) สัญญาณ AWE จะทำงานจนกระทั่งสิ้นสุดช่วงที่ 2 ที่จุดเริ่มต้นของช่วงสุดท้าย (Hold Time) จะมีการอ่านข้อมูล และที่จุดปลายของช่วงสุดท้าย สัญญาณต่าง ๆ ก็จะสิ้นสุดการทำงาน

จากที่กล่าวมาในข้างต้นนั้นเป็นการคำนวณลักษณะของสัญญาณสำหรับอ่านและเขียนข้อมูลของมอดูล EMIF บนตัวประมวลผลสัญญาณดิจิทัลผ่านรีจิสเตอร์ควบคุมบนตัวประมวลผลเท่านั้น ซึ่งในทางปฏิบัตินั้นสัญญาณที่ออกมาจริงยังมีผลมาจากค่าความหน่วงของตัวประมวลผลที่ใช้อีกด้วย ซึ่งขึ้นอยู่ค่าเฉพาะ (Specification) ของตัวประมวลผลแต่ละชนิด

2.4 โครงสร้างซอฟต์แวร์สำหรับติดต่อกับตัวกราดตรวจลายนิ้วมือ

ในหัวข้อนี้จะกล่าวถึงลักษณะและโครงสร้างที่ใช้ในการเขียนโปรแกรมสำหรับติดต่อกับชุดกราดตรวจลายนิ้วมือ โดยที่จะใช้วิธีการเขียนโปรแกรมบนตัวประมวลผลสัญญาณดิจิทัล TMS320C6713 แบบใช้ DSP/BIOS เพื่อใช้ประยุกต์กับงานประเภทเวลาจริง (Real Time)

วิธีรับรู้ว่ามีการวางนิ้วมือลงบนตัวกราดตรวจลายนิ้วมือใช้การรอสัญญาณขัดจังหวะที่ส่งมาจากชุดกราดตรวจลายนิ้วมือ จากนั้นตัวประมวลผลก็เริ่มทำการอ่านข้อมูลเข้ามา ดังนั้นโครงสร้างซอฟต์แวร์ของระบบนั้นสามารถแสดงดังภาพที่ 49



ภาพที่ 49 รูปแบบโครงสร้างซอฟต์แวร์การอ่านภาพลายนิ้วมือ

จากภาพที่ 49 เป็นลักษณะโครงสร้างของโปรแกรมในส่วนของการติดต่อกับชุดกราดตรวจลายนิ้วมือ โดยระบบจะทำงานก็ต่อเมื่อได้รับสัญญาณขัดจังหวะมาจากชุดกราดตรวจลายนิ้วมือ ซึ่งสัญญาณที่รับเข้ามาจะถูกประมวลผลโดยมอดูล HWI หลังจากนั้นจะส่งการทำงานไปที่มอดูล SWI เนื่องจากตัวประมวลผลจัดให้มอดูล HWI มีระดับความสำคัญสูงที่สุด ดังนั้นต้องใช้เวลาในการประมวลผลน้อยที่สุด ซึ่งวิธีลดเวลาในการประมวลผลคือส่งข้อมูลไปยังมอดูล SWI ที่มีความสำคัญน้อยกว่าในการประมวลผล ในส่วนของมอดูล SWI จะเป็นส่วนควบคุมการเขียนและอ่านข้อมูล ซึ่งจะสั่งการผ่านทางมอดูล EMIF

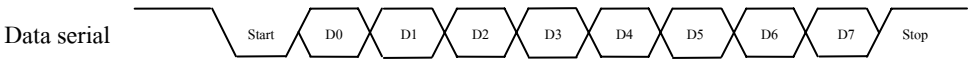
จากที่กล่าวมาในหัวข้อนี้การติดต่อระหว่างชุดกราดตรวจลายนิ้วมือและตัวประมวลผลสัญญาณดิจิทัลนั้น ตัวประมวลผลสัญญาณดิจิทัล TMS320C6713 สามารถใช้มอดูล EMIF และขาสัญญาณขัดจังหวะมาควบคุมการอ่านหรือเขียนข้อมูลในแบบไม่เข้าจังหวะ (Asynchronous) ได้ และการเขียนโปรแกรมบนตัวประมวลผลนั้นต้องมีการแบ่งหน้าที่การทำงานของระบบเป็นมอดูล HWI และมอดูล SWI เพื่อให้ระบบยังสามารถจัดตารางการทำงานในเวลาจริงได้

3. การติดต่อผ่านพอร์ตอนุกรมแบบ UART บนฮาร์ดแวร์ประมวลผลสัญญาณดิจิทัล

ในหัวข้อนี้จะกล่าวถึงหลักและวิธีการใช้งานมอดูล McBSP (Multi-channel Buffered Serial Port) สำหรับติดต่อกับ UART (Universal Asynchronous Receiver/Transmitter) เนื่องจากลักษณะของพอร์ตอนุกรมบนตัวประมวลผลสัญญาณดิจิทัลเป็นแบบเข้าจังหวะ (Synchronous) ทำให้ไม่สามารถติดต่อกับอุปกรณ์ที่ใช้ UART ได้โดยตรง นอกจากนี้จะกล่าวถึงวิธีการออกแบบและเขียนโครงสร้างโปรแกรมทางด้านซอฟต์แวร์บนตัวประมวลผลสัญญาณดิจิทัลสำหรับการติดต่อกับ UART รวมทั้งวิธีการเชื่อมต่อทางด้านฮาร์ดแวร์ด้วย ซึ่งจะกล่าวในรายละเอียดของส่วนต่าง ๆ ดังต่อไปนี้

3.1 ลักษณะของ UART

มาตรฐานกระบวนการส่งผ่านข้อมูลแบบ UART นั้นเป็นกระบวนการส่งสัญญาณข้อมูลอนุกรมแบบไม่เข้าจังหวะหรือเป็นการรับส่งที่ไม่ใช้สัญญาณนาฬิกาในการเข้าจังหวะข้อมูล โดยโครงสร้างทางสัญญาณของข้อมูลแบบ UART สามารถแสดงได้ดังภาพที่ 50



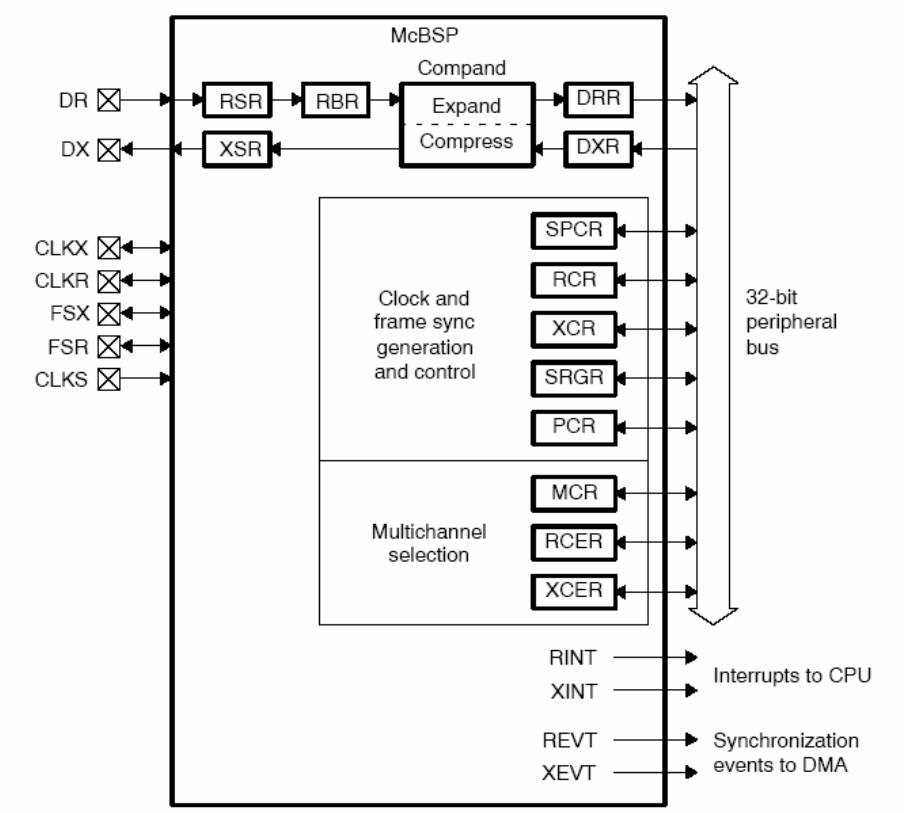
ภาพที่ 50 แผนภาพเวลาของ UART

กระบวนการส่งของข้อมูลแบบ UART นั้นจะเริ่มจากบิตเริ่มต้น (Start Bit) แล้วปิดท้ายด้วยบิตหยุด (Stop Bit) เสมอ โดยที่บิตเริ่มต้นจะมีระดับแรงดันเป็นลอจิกต่ำ (Logic Low) ส่วนบิตหยุดจะมีระดับแรงดันเป็นลอจิกสูง (Logic High) และข้อมูลที่จะส่งนั้นจะถูกส่งแบบอนุกรมออกมา โดยที่จะเรียงลำดับจากบิตที่มีความสำคัญต่ำสุด (Least Significant Bit) ไปยังบิตที่มีความสำคัญสูงสุด (Most Significant Bit) ซึ่งในการส่งแต่ละครั้งจะส่งข้อมูลทีละไม่เกิน 1 ไบต์ สำหรับการติดต่อกับอุปกรณ์ภายนอก เช่น คอมพิวเตอร์หรือตัวควบคุมระบบ สัญญาณข้อมูลอนุกรมที่ได้จะถูกเปลี่ยนจากระดับสัญญาณแบบ CMOS ไปเป็นระดับสัญญาณ RS232 ซึ่งจะมีช่วงระดับแรงดัน +3 ถึง +25 โวลต์ เรียกว่า Space (Logic 0) และระดับแรงดัน -3 ถึง -25 โวลต์ เรียกว่า Mark (Logic 1)

จากที่กล่าวมาในข้างต้นนั้นจะเห็นได้ว่ามาตรฐานการส่งข้อมูลแบบ UART นั้นเป็นการส่งข้อมูลอนุกรมแบบไม่เข้าจังหวะ (Asynchronous) ซึ่งจะไม่สามารถต่อเข้ากับมอดูล McBSP บนตัวประมวลผลสัญญาณดิจิทัล (DSP) ได้โดยตรง เนื่องจากตัวประมวลผลสัญญาณดิจิทัลมีการส่งข้อมูลอนุกรมแบบเข้าจังหวะ (Synchronous) ดังนั้นการติดต่อระหว่างตัวประมวลผลสัญญาณดิจิทัลและ UART นั้นจะต้องมีการปรับแต่งทางด้านฮาร์ดแวร์และซอฟต์แวร์ที่เหมาะสม ซึ่งจะได้กล่าวไว้ในส่วนต่อไป

3.2 การติดต่อสื่อสารทางพอร์ตอนุกรมแบบ UART บนตัวประมวลผลสัญญาณดิจิทัล

ในหัวข้อนี้จะกล่าวถึงลักษณะทางด้านฮาร์ดแวร์ของอุปกรณ์ UART และ McBSP ของตัวประมวลผลสัญญาณดิจิทัล โดยที่ฮาร์ดแวร์ของอุปกรณ์ UART สำหรับต่อกับอุปกรณ์ควบคุมหรือคอมพิวเตอร์จะมีขาสัญญาณที่จำเป็นอยู่ 3 ขาสัญญาณ ได้แก่ ขาส่ง (Tx) ขารับ (Rx) และขาอ้างอิง (GND) ส่วนมอดูล McBSP ของตัวประมวลผลสัญญาณดิจิทัลจะประกอบด้วยขาสัญญาณใช้งานทั้งหมดรวมทั้งรีจิสเตอร์ภายใน ดังภาพที่ 51

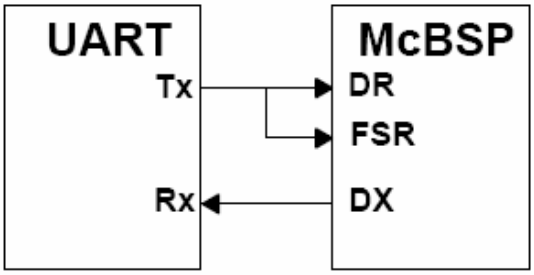


ภาพที่ 51 แผนภาพของมอดูล McBSP

ที่มา: Texas Instruments Inc. (2001)

จากภาพนั้นมอดูล McBSP จะมีขาสัญญาณ 7 ขา โดยแบ่งเป็น 2 กลุ่ม กลุ่มแรกใช้ในการส่งและรับข้อมูล ได้แก่ DX และ DR ตามลำดับ ส่วนกลุ่มที่สองจะใช้ในการเข้าจังหวะสัญญาณ (Synchronization) ได้แก่ CLKX CLKR FSX FSR และ CLKS โดยที่สัญญาณ CLKX และ CLKR ใช้สำหรับรับส่งสัญญาณนาฬิกาเพื่อทำการเข้าจังหวะบิตสัญญาณระหว่างการส่งและรับข้อมูล ตามลำดับ สัญญาณ CLKS เป็นสัญญาณนาฬิกาขาเข้าของมอดูล McBSP ส่วนสัญญาณ FSX และ FSR ใช้สำหรับรับส่งสัญญาณที่ใช้ในการเข้าจังหวะเฟรมของสัญญาณส่งและรับตามลำดับ

ในการติดต่อทางด้านฮาร์ดแวร์ระหว่างมอดูล McBSP บนตัวประมวลผลสัญญาณดิจิทัลกับอุปกรณ์ UART นั้น ที่มอดูล McBSP จะเลือกใช้เฉพาะขาสัญญาณ DX DR และ FSR ในการติดต่อกับอุปกรณ์ UART ดังภาพที่ 52



ภาพที่ 52 การเชื่อมต่อระหว่างมอดูล McBSP กับอุปกรณ์ UART
ที่มา: Texas Instruments Inc. (2001)

จากภาพที่ 52 ข้อมูลจาก McBSP ไปยัง UART จะส่งเข้า Rx ผ่านทางขา DX โดยตรง ส่วนข้อมูลจาก UART ไปยัง McBSP จะส่งผ่าน Tx ไปยังขา DR และ FSR โดยขาสัญญาณ FSR จะทำหน้าที่ตรวจจับแบบไม่เข้าจังหวะ (Asynchronous) ว่ามีข้อมูลถูกส่งมาหรือไม่ ซึ่งจะใช้ซอฟต์แวร์ในการปรับแต่งคุณลักษณะของมอดูล McBSP เพื่อให้สามารถใช้ติดต่อกับ UART ได้

3.3 มอดูล McBSP บนตัวประมวลผลสัญญาณดิจิทัลและการปรับแต่ง

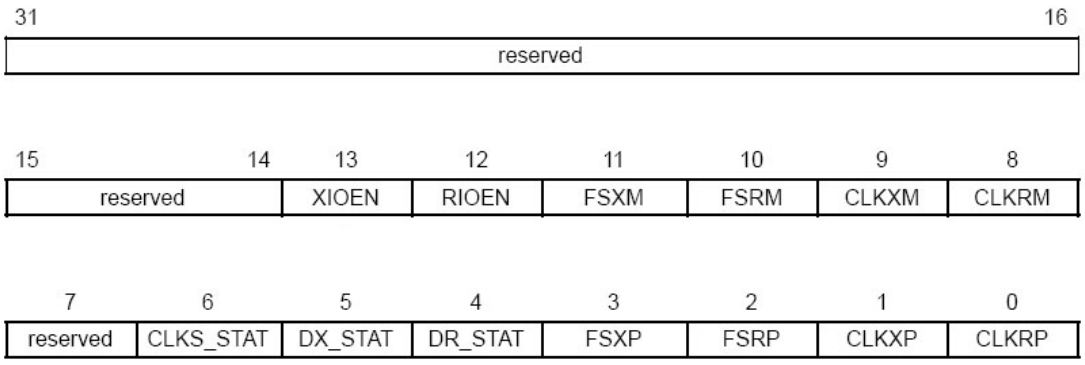
ในการปรับแต่งคุณลักษณะของมอดูล McBSP นั้นตัวประมวลผลสัญญาณดิจิทัลได้เตรียมรีจิสเตอร์สำหรับควบคุมคุณลักษณะต่าง ๆ ดังภาพที่ 51 รีจิสเตอร์ควบคุมสำหรับมอดูล McBSP แบ่งเป็น 2 กลุ่ม โดยที่กลุ่มแรกใช้ควบคุมลักษณะทั่วไปของการรับส่งข้อมูล ได้แก่ SPCR XCR RCR SRGR และ PCR ซึ่งจะกล่าวถึงการใช้งานอย่างคร่าว ๆ ในส่วนต่อไป ส่วนกลุ่มที่ 2 ไม่จำเป็นในงานส่วนนี้ ได้แก่ MCR XCER และ RCER ซึ่งใช้ในการจัดสรรช่องสัญญาณของการ

รับส่งสัญญาณ เนื่องจากมอดูล McBSP มีความสามารถในการจัดการรับส่งสัญญาณแบบหลายช่องสัญญาณ (Multi-channel) ได้ด้วย

สิ่งที่ต้องทราบก่อนในการปรับคุณลักษณะของมอดูล McBSP ได้แก่ จำนวนชุดของบิตข้อมูลและอัตรารับส่งข้อมูล ซึ่งจำนวนบิตข้อมูลที่ใช้ในการรับส่งนั้นเท่ากับ 10 บิตข้อมูล โดยจะต้องนับรวมในส่วนของบิตเริ่มต้นและบิตสุดท้ายด้วย ซึ่งจะได้ว่าในการส่งหนึ่งครั้งหรือหนึ่งเฟรมข้อมูลเท่ากับ 10 คำ (word) ส่วนอัตรารับส่งข้อมูลของ UART นั้นจะถูกนำมาคำนวณ เพื่อใช้ในการหารความถี่ของสัญญาณนาฬิกาในการรับส่งข้อมูล ซึ่งจะกล่าวในส่วนต่อไป

ในการปรับแต่งคุณลักษณะของมอดูล McBSP นั้น จะสามารถปรับแต่งรีจิสเตอร์ที่กล่าวมาแล้วในข้างต้นได้ ซึ่งส่วนต่อไปจะกล่าวถึงความหมายและการปรับแต่งค่าของรีจิสเตอร์ต่าง ๆ ของมอดูล McBSP สำหรับติดต่อกับอุปกรณ์ UART อย่างคร่าว ๆ ได้ดังนี้

- PCR (Pin Control Register) เป็นรีจิสเตอร์สำหรับควบคุมรูปแบบการใช้งานของขาสัญญาณต่าง ๆ ในมอดูล McBSP ซึ่งจะประกอบด้วยฟิลด์ต่าง ๆ ดังภาพที่ 53 โดยจะกล่าวถึงเฉพาะฟิลด์ที่สำคัญเท่านั้น



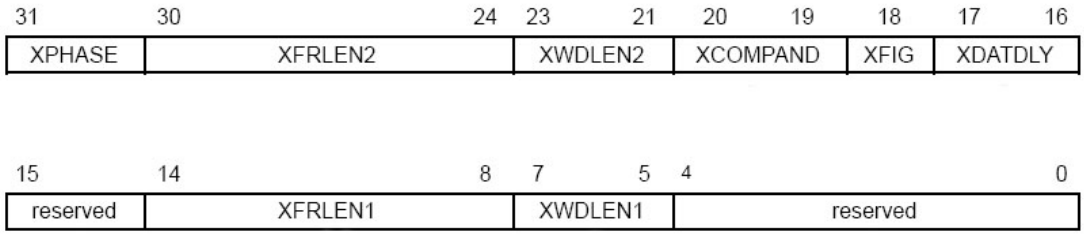
ภาพที่ 53 รีจิสเตอร์ควบคุม Pin

ที่มา: Texas Instruments Inc. (2001)

จากภาพที่ 53 มีฟิลด์ที่สำคัญ ได้แก่ FSXM(11) และ FSXP(3) ทำหน้าที่ปรับค่าการส่งสัญญาณเข้าจังหวะเฟรมข้อมูลสัญญาณ ซึ่งสามารถตั้งค่าให้เป็นสัญญาณจากภายในหรือภายนอกได้ และการทำงานที่ลอจิกสูงหรือต่ำได้ตามลำดับ โดยในที่นี้จะทำการตั้งค่าเป็น 1, 1 ต่อมา

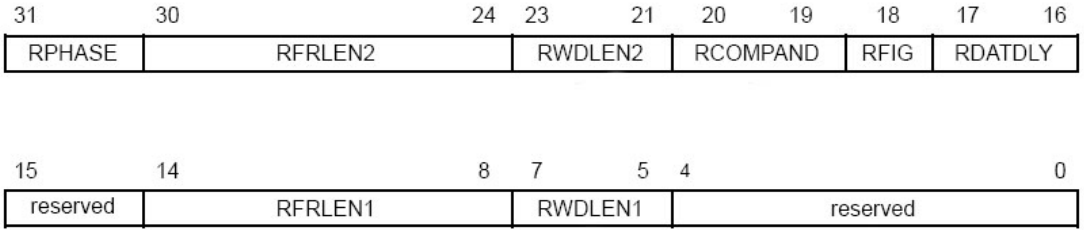
เป็นฟิลด์ FSRM(10) และ FSRP(2) ทำหน้าที่ปรับค่าการรับสัญญาณเข้าจังหวะเฟรมข้อมูลสัญญาณ ซึ่งสามารถตั้งค่าให้เป็นสัญญาณจากภายในหรือภายนอกได้ และการทำงานที่ลอจิกสูงหรือต่ำได้ตามลำดับเช่นกัน โดยในที่นี้จะทำการตั้งค่าเป็น 0, 1 สุดท้าย CLKXM(9) และ CLKRM(8) เป็นการเลือกว่าจะใช้สัญญาณนาฬิกาจากภายในหรือภายนอก โดยในที่นี้จะทำการตั้งค่าเป็น 1, 1 เป็นต้น

- XCR/RCR (Transmit/Receive Control Registers) เป็นรีจิสเตอร์สำหรับควบคุมลักษณะต่าง ๆ ของการส่งและรับในลักษณะเฟรมข้อมูล ซึ่งจะประกอบด้วยฟิลด์ต่าง ๆ ดังภาพที่ 54 และ 55 ตามลำดับ โดยที่จะทำการกำหนดการรับส่งแบบ 1 เฟรมข้อมูล ซึ่งประกอบด้วย 10 คำ (Word) ข้อมูลและแต่ละคำ จะมีความยาว 32 บิต



ภาพที่ 54 รีจิสเตอร์ควบคุมการส่ง

ที่มา: Texas Instruments Inc. (2001)



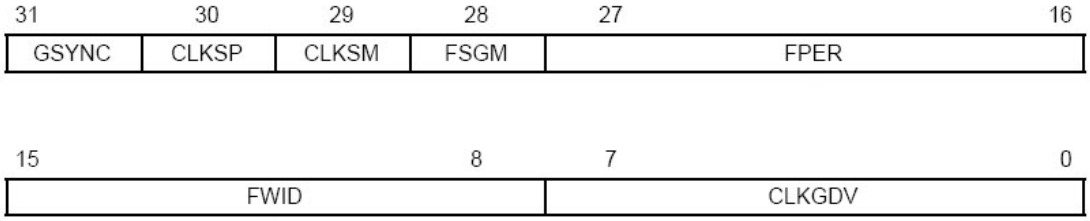
ภาพที่ 55 รีจิสเตอร์ควบคุมการรับ

ที่มา: Texas Instruments Inc. (2001)

จากภาพที่ 54 และ 55 มีฟิลด์ที่สำคัญ ได้แก่ (X/R)PHASE, (X/R)FRLEN1 และ (X/R)WDLEN1 เป็นการกำหนดค่าความยาวและจำนวนข้อมูลดังที่กล่าวมาในข้างต้น ซึ่งจะมีการตั้งค่าเป็น 0, 0xA, 0x5 ตามลำดับ และ (X/R) FIG ใช้ในการกำหนดให้มอดูล McBSP ไม่สนใจการเกิดสัญญาณเข้าจังหวะเฟรมข้อมูลภายในการรับเฟรมข้อมูลจากอุปกรณ์ UART โดยที่จะตั้งค่าให้

เท่ากับ 1 เนื่องจากสัญญาณที่มาจากอุปกรณ์ UART นั้นเป็นแบบไม่เข้าจังหวะแต่ที่ตัวประมวลผลสัญญาณดิจิทัลนั้นจะต้องทำการเข้าจังหวะ ดังนั้นการตรวจสอบบิตเริ่มของข้อมูลจะดูได้จากการเปลี่ยนระดับสัญญาณจากลอจิกสูงไปต่ำ (ตรวจสอบบิตเริ่มต้น) มาตรวจสอบการเริ่มรับข้อมูล แต่เนื่องจากภายในการส่งข้อมูลของ UART ก็มีการเปลี่ยนระดับสัญญาณจากลอจิกสูงไปต่ำเช่นกัน จึงต้องกำหนดให้มอดูล McBSP ไม่สนใจการเกิดสัญญาณเข้าจังหวะเฟรมข้อมูลภายในการรับเฟรมข้อมูลจากอุปกรณ์ UART ด้วย

- SRGR (Sample Rate Generator Register) เป็นรีจิสเตอร์สำหรับควบคุมอัตราการรับส่งข้อมูลซึ่งจะประกอบด้วยฟิลด์ต่าง ๆ ดังภาพที่ 56 โดยจะสามารถคำนวณอัตราการรับและส่งให้สอดคล้องกับ UART ได้

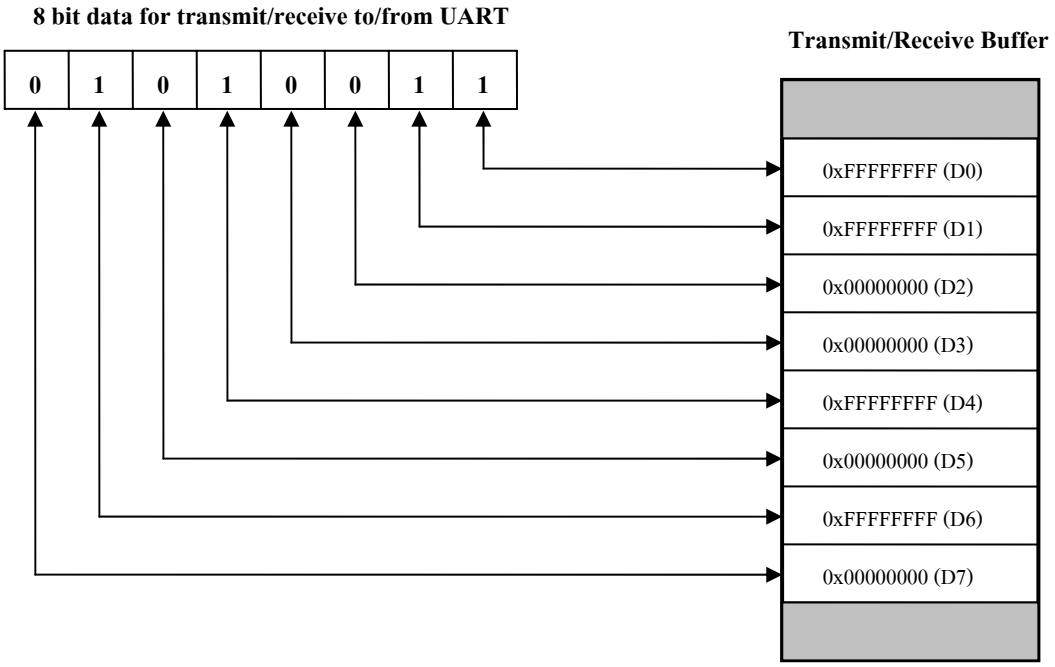


ภาพที่ 56 รีจิสเตอร์ตัวสร้างอัตราสุ่มตัวอย่าง
ที่มา: Texas Instruments Inc. (2001)

จากภาพที่ 56 มีฟิลด์ที่สำคัญ คือ CLKSM(29) เป็นการกำหนดว่าจะต้องการคำนวณอัตราการรับส่งข้อมูลจากแหล่งกำเนิดสัญญาณนาฬิกาภายในหรือภายนอก และ CLKGDV(7:0) ใช้ในการคำนวณอัตราการส่งข้อมูล ซึ่งจะกำหนดให้ CLKSM เท่ากับ 1 เป็นการเลือกแหล่งกำเนิดสัญญาณนาฬิกาภายใน และมีค่าเท่ากับ CPU Clock/2 = 75 MHz ส่วนอัตราการรับส่งข้อมูลคำนวณได้จาก Baud Rate = 75 MHz/(32x(CLKGDV+1)) ซึ่งจากการคำนวณจะได้ค่าที่ต่ำสุดในการรับส่งข้อมูลนั้นเท่ากับ 9600 และจะได้ค่า CLKGDV = 0xF3

3.4 การแปลงข้อมูล

ก่อนที่ข้อมูลแต่ละไบต์จะถูกส่งหรือรับเข้าทางมอดูล McBSP นั้น ข้อมูลจะถูกขยายและถอดรหัสข้อมูล กล่าวคือข้อมูล 1 บิตสำหรับส่งออกทางมอดูล McBSP จะถูกขยายให้เป็น 1 คำ (Word) หรือ 32 บิตในบัฟเฟอร์ข้อมูล ดังภาพที่ 57



ภาพที่ 57 การแปลงข้อมูลที่ได้จาก UART

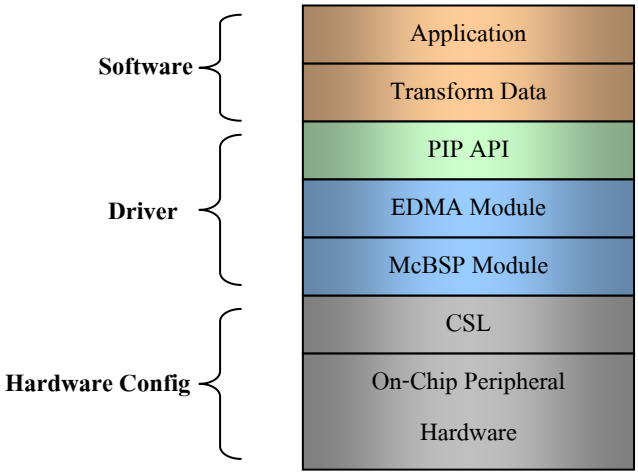
ในทางกลับกันข้อมูลที่รับเข้ามานั้นจะถูกเก็บไว้ในลักษณะเดียวกัน ซึ่งจะทำการถอดรหัสที่ได้ออกมาว่าจะเป็นบิต 1 หรือ 0 โดยขั้นตอนในการขยายข้อมูลที่จะส่งออกและถอดรหัสข้อมูลที่รับเข้ามานั้นจะใช้ซอฟต์แวร์ช่วยในการประมวลผล

เหตุผลสำคัญในการทำการขยายและถอดรหัสข้อมูลนั้น คือ การเพิ่มขนาดของบิตโดยรวมที่จะทำการรับส่งทางมอดูล McBSP เนื่องจากการคำนวณสัญญาณนาฬิกาในที่นี้ใช้สัญญาณนาฬิกาที่มาจากภายใน ซึ่งมีความถี่ที่สูงมาก (75 MHz) แต่สัญญาณที่จะทำการติดต่อกับอุปกรณ์ UART นั้นใช้อัตราการส่งที่ค่อนข้างต่ำ (ประมาณ 300 – 115200 bps) ดังนั้นการขยายข้อมูลจะทำให้สามารถลดอัตราการรับส่งข้อมูลให้เท่ากับอุปกรณ์ UART ได้

3.5 ซอฟต์แวร์การรับส่งสำหรับ UART

ในหัวข้อนี้จะกล่าวถึงลักษณะและโครงสร้างที่ใช้ในการเขียนโปรแกรมสำหรับติดต่อกับอุปกรณ์ UART โดยที่ในที่นี้จะใช้วิธีการเขียนโปรแกรมบนตัวประมวลผลสัญญาณดิจิทัล TMS320C6713 แบบใช้ DSP/BIOS เพื่อใช้ประยุกต์กับงานประเภทเวลาจริง (Real Time) โดยที่จะกล่าวในส่วนพื้นฐานที่เป็นตัวขับสัญญาณ (Driver) ขาเข้าและขาออกจากระบบ

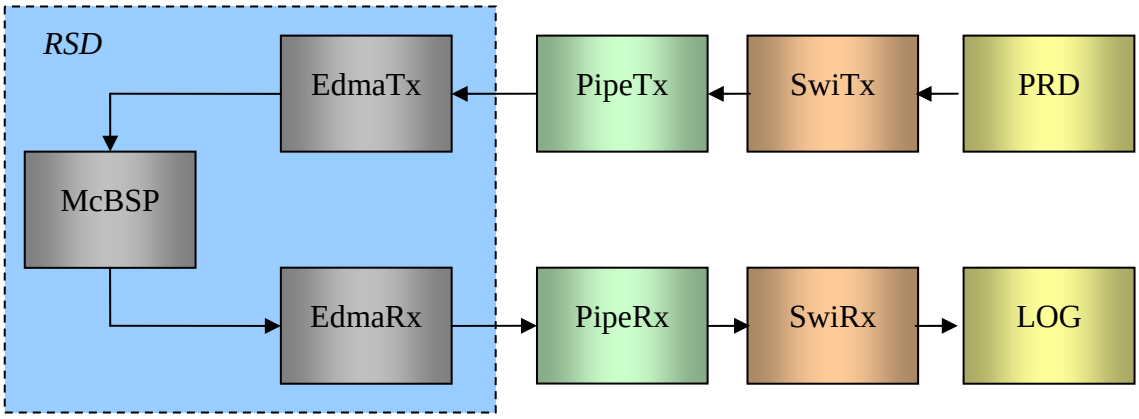
โครงสร้างซอฟต์แวร์โดยรวมของระบบนั้นสามารถแบ่งเป็นลำดับชั้นต่าง ๆ เพื่อให้ง่ายต่อการพัฒนาต่อไปนั้น สามารถแสดงดังภาพที่ 58



ภาพที่ 58 แผนภาพแบบโครงสร้างของตัวขับข้อมูลแบบอนุกรม

จากภาพแสดงลักษณะโครงสร้างของโปรแกรมในส่วนของการขับสัญญาณติดต่อกับอุปกรณ์ UART โดยเริ่มจากชั้นประยุกต์ (Application) ทำการผ่านข้อมูลไปยังชั้นแปลงข้อมูล (Transform Data) เพื่อทำการจัดเรียงข้อมูลให้สอดคล้องกับบัฟเฟอร์ (Buffer) ที่ตั้งไว้ในหัวข้อที่ผ่านมา จากนั้นมอดูล PIP ซึ่งเป็นส่วนต่อประสาน DSP/BIOS จะทำการส่งผ่านข้อมูลระหว่างชั้นแปลงข้อมูลไปยังมอดูล EDMA เพื่อทำหน้าที่ส่งข้อมูลแทน CPU ไปยังมอดูล McBSP ในชั้น CSL จะทำหน้าที่แปลงคำสั่งเพื่อติดต่อกับฮาร์ดแวร์วงจรรวมรอบข้าง (On-Chip Peripheral) โดยเป็นส่วนที่ DSP/BIOS ได้เตรียมไว้ให้ติดต่อกับฮาร์ดแวร์วงจรรวมรอบข้าง (On-Chip Peripheral)

จากโครงสร้างลำดับชั้นที่ได้กล่าวไว้แล้วนั้น ต่อไปจะทำการออกแบบรายละเอียดของการเขียนโปรแกรมสำหรับตัวประมวลผลสัญญาณดิจิทัลที่มีลักษณะการทำงานเชิงระบบปฏิบัติการ ซึ่งจะแบ่งเป็นอ๊อบเจกต์หรืองานโยงใยย่อย (Thread) ต่าง ๆ โดยใช้เครื่องมือใน DSP/BIOS ในการออกแบบ และสามารถแสดงได้ดังภาพที่ 59

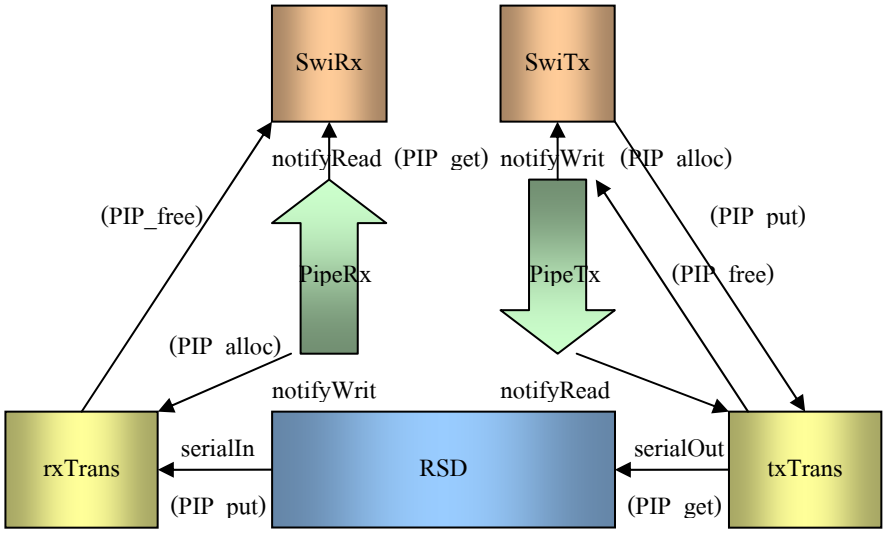


ภาพที่ 59 แผนภาพมอดูลอ็อบเจกต์ของการรับส่งข้อมูลแบบอนุกรม

จากภาพที่ 59 มอดูลที่ใช้ ได้แก่ มอดูล PRD SWI PIP และ LOG โดยที่มอดูล PRD LOG นั้นเป็นส่วนที่ใช้ในการรับส่งข้อมูลไปยังซอฟต์แวร์ประยุกต์ โดยที่การทำงานจะเริ่มจากมอดูล PRD เปรียบเสมือนข้อมูลที่ใช้ส่งซึ่งจะกำหนดคาบการส่งข้อมูลไปให้มอดูล SWI หรือ SwiTx ซึ่งทำหน้าที่จัดรูปแบบข้อมูลที่ใช้ในการส่ง จากนั้นมอดูล PIP หรือ PipeTx จะทำหน้าที่ส่งผ่านข้อมูลที่เหมาะสมไปให้กับมอดูล EDMA ใช้ในการส่งข้อมูลออกไปทางมอดูล McBSP ในการรับข้อมูลนั้น ข้อมูลจากอุปกรณ์ UART ที่ส่งมาจะถูกรับโดยมอดูล McBSP เช่นกัน และมอดูล EDMA จะทำหน้าที่รับข้อมูลเข้ามาเก็บไว้ในบัฟเฟอร์ (Buffer) หลังจากนั้นมอดูล PIP หรือ PipeRx จะส่งผ่านข้อมูลไปยังมอดูล SWI หรือ SwiRx เพื่อถอดรหัสสัญญาณข้อมูลที่ได้ แล้วส่งไปแสดงผลยังมอดูล LOG หรือรับข้อมูลไปประมวลผลต่อไป

จากภาพที่ 59 ในบริเวณส่วนที่เป็นเส้นประนั้นเรียกว่า มอดูล RSD (RS232 Driver) ซึ่งถูกเขียนโดยผ่านส่วนต่อประสาน CSL ในการควบคุมอุปกรณ์ที่อยู่ภายในตัวประมวลผลดิจิทัลในที่นี้ ได้แก่ มอดูล EDMA และมอดูล McBSP ซึ่งในส่วนนี้จะเป็นการเขียนโปรแกรมเพื่อปรับแต่งคุณลักษณะและโครงสร้างของมอดูล EDMA และมอดูล McBSP ให้เหมาะสมกับการติดต่ออุปกรณ์ UART

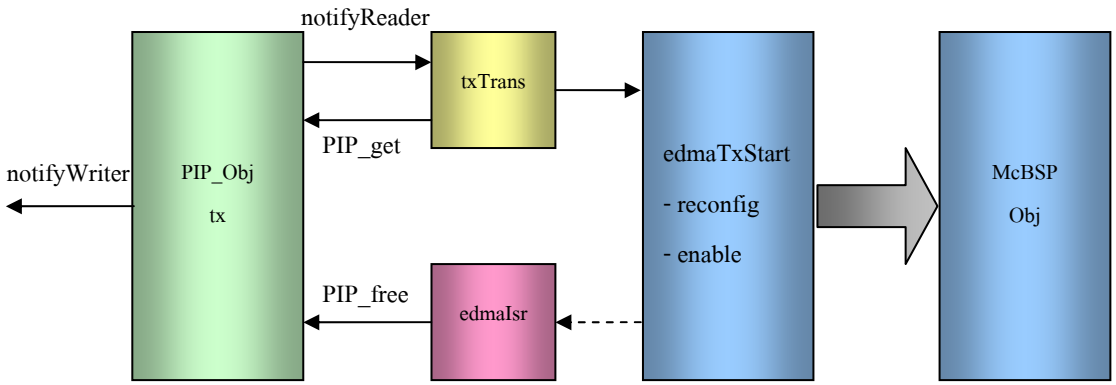
จากการออกแบบโครงสร้างความสัมพันธ์ของแต่ละอ็อบเจกต์สำหรับตัวประมวลผลสัญญาณดิจิทัลแล้ว ต่อไปจะกล่าวถึงฟังก์ชันที่ใช้และความสัมพันธ์ระหว่างการติดต่อระหว่างมอดูล SWI และมอดูล RSD โดยการใช้มอดูล PIP ในการส่งผ่านข้อมูล ซึ่งแสดงความสัมพันธ์ได้ดังภาพที่ 60



ภาพที่ 60 การส่งข้อมูลระหว่างมอดูล SWI และมอดูล RSD

ในขั้นแรกหลังจากคำสั่งคืนค่า (Return) ในฟังก์ชันหลัก (Main Function) เพื่อเข้าสู่ DSP/BIOS นั้น notifyWriter ของ PipeTx และ PipeRx จะถูกเรียกขึ้นมาเพื่อจองขนาดของหน่วยความจำและรอให้เรียกใช้งาน เมื่อข้อมูลใน PipeTx หรือ PipeRx เต็มแล้ว notifyReader ก็จะเรียกฟังก์ชันใหม่ขึ้นมาเพื่อนำข้อมูลออกไปใช้งานและจะยกเลิกหน่วยความจำที่จองไว้ หลังจากนั้นถ้ามีจำนวนท่อ (Pipe) ที่ไม่ถูกใช้งานเหลืออยู่นั้น notifyWriter ของ PipeTx และ PipeRx จะถูกเรียกขึ้นมาเพื่อจองขนาดของหน่วยความจำอีกครั้ง ซึ่งการทำงานในลักษณะนี้จะกระทำไปเรื่อย ๆ ไม่มีที่สิ้นสุด ถ้ายังมีการส่งผ่านข้อมูลอยู่ จากภาพที่ 60 มอดูล PIP มีฟังก์ชันหลักสำหรับจัดการและควบคุมมอดูล PIP ได้แก่ PIP_alloc, PIP_put, PIP_get และ PIP_free ซึ่งทำหน้าต่าง ๆ เกี่ยวกับการควบคุมหน่วยความจำของตัวประมวลผลสัญญาณดิจิทัล โดยที่ PIP_alloc นั้นจะจองหน่วยความจำสำหรับมอดูล PIP ขึ้น หลังจากนั้น PIP_put จะส่งข้อมูลไปยังมอดูล PIP เมื่อข้อมูลในท่อ (Pipe) เต็มนั้น จะใช้ PIP_get ในการนำข้อมูลจาก PIP มอดูลมาใช้งาน และ PIP_free เป็นการยกเลิกหน่วยความจำที่ได้จองไว้

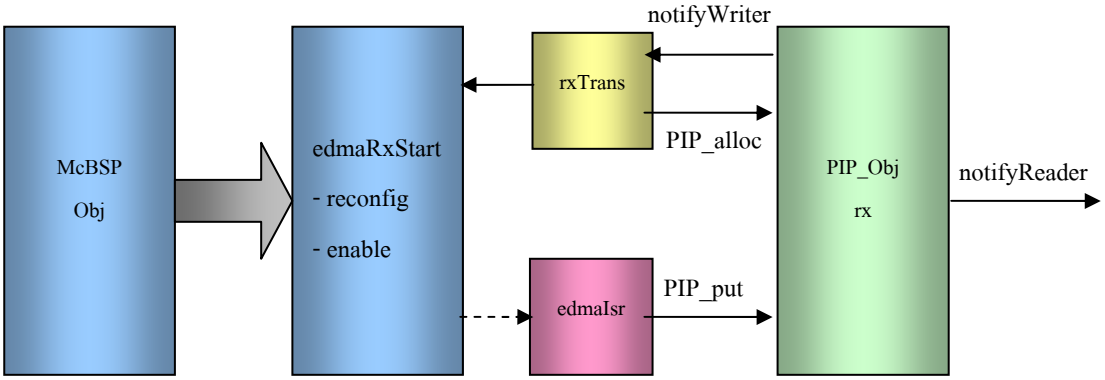
การรับส่งข้อมูลระหว่างมอดูล PIP และมอดูล RSD สามารถแบ่งได้เป็นสองส่วน คือ ส่วนส่งข้อมูลและส่วนรับข้อมูล โดยส่วนของการส่งข้อมูลสามารถเขียนได้ดังภาพที่ 61



ภาพที่ 61 การส่งข้อมูลของมอดูล PIP และมอดูล RSD

จากภาพที่ 61 หลังจากที่ข้อมูลในท่อ (Pipe) เต็มแล้ว notifyReader จะเรียกฟังก์ชัน txTrans เพื่อนำข้อมูลออกไปใช้ ซึ่ง txTrans จะใช้คำสั่ง PIP_get ในการดึงข้อมูลออกมา หลังจากนั้นก็จะส่งต่อไปยังฟังก์ชันสำหรับควบคุม EDMA (edmaTxStart) ซึ่งจะตั้งค่าเริ่มต้นให้ในการส่งผ่านข้อมูลเพื่อออกไปยังมอดูล McBSP สำหรับส่งข้อมูลออกไปสู่ภายนอก หลังจากที่ EDMA ส่งข้อมูลครบตามค่าที่ได้กำหนดไว้แล้วก็จะส่งสัญญาณขัดจังหวะ (Interrupt) เพื่อบอกหน่วยประมวลผลว่าส่งข้อมูลเรียบร้อยแล้ว ซึ่ง edmaIsr จะยกเลิกหน่วยความจำที่จองไว้สำหรับมอดูล PIP โดยผ่านคำสั่ง PIP_free

ส่วนขั้นตอนการรับข้อมูลนั้น จะสามารถเขียนในรูปแผนภาพได้ดังภาพที่ 62



ภาพที่ 62 การรับข้อมูลของมอดูล PIP และมอดูล RSD

จากภาพเมื่อเริ่มต้นเข้าไบออสนั้น notifyWriter จะเรียกฟังก์ชัน rxTrans เพื่อจองหน่วยความจำสำหรับเก็บข้อมูล ซึ่ง txTrans จะใช้คำสั่ง PIP_alloc ในการจองพื้นที่หน่วยความจำสำหรับเก็บข้อมูล หลังจากนั้นก็จะเรียกฟังก์ชันสำหรับควบคุม EDMA (edmaRxStart) ซึ่งจะตั้งค่าเริ่มต้นให้ในการรับข้อมูลไปยังมอดูล McBSP ซึ่งใช้รอรับข้อมูลภายนอกที่มาจากอุปกรณ์ UART หลังจากที่มอดูล McBSP ได้รับข้อมูลจากภายนอกแล้ว EDMA จะรับข้อมูลที่ได้มาเก็บไว้ในบัฟเฟอร์ของมอดูล PIP ที่ได้เตรียมไว้ เมื่อ EDMA เก็บข้อมูลที่ได้ครบตามค่าที่ได้กำหนดไว้แล้วก็จะส่งสัญญาณขัดจังหวะ (Interrupt) เพื่อบอกหน่วยประมวลผลว่ารับข้อมูลมาครบแล้วจากนั้น edmaIsr ส่งข้อมูลที่ได้มาผ่านมอดูล PIP โดยผ่านคำสั่ง PIP_put

จากที่กล่าวมาในหัวข้อนี้การติดต่อกันระหว่างอุปกรณ์ UART และตัวประมวลผลสัญญาณดิจิทัลนั้น ตัวประมวลผลสัญญาณดิจิทัล TMS320C6713 นั้นไม่สามารถเรียกใช้คำสั่งในการรับส่งข้อมูลแบบ UART ได้ เนื่องจากลักษณะการรับส่งข้อมูลของพอร์ตอนุกรมบนตัวประมวลผลสัญญาณดิจิทัลนั้นเป็นแบบเข้าจังหวะ (Synchronous) จึงต้องมีการปรับแต่งโครงสร้างทางฮาร์ดแวร์และเขียนโปรแกรมขึ้นเพื่อช่วยในการรับส่งข้อมูลแทน

สำหรับการปรับแต่งโครงสร้างทางฮาร์ดแวร์จะใช้ขาสัญญาณเพียง 3 ขาและมีการต่อร่วมกันระหว่างขา FSR และ DR ใช้ในการรับสัญญาณแบบไม่เข้าจังหวะ (Asynchronous) ซึ่งไม่ค่อยยุ่งยากในทางปฏิบัติมากนัก ส่วนการเขียนโปรแกรมขึ้นนั้นเป็นลักษณะการเขียนแบบใช้ DSP/BIOS อาจเกิดความยุ่งยากในการรวมโปรแกรมเนื่องจากการปรับแต่งข้อมูลในส่วนของ DSP/BIOS นั้นมีการสื่อสารที่ซับซ้อนมาก แต่การแบ่งโปรแกรมในลักษณะชั้นและมอดูลนั้นจะช่วยลดความยุ่งยากในการพัฒนาและการนำไปใช้ร่วมกับส่วนอื่นได้

4. <u>เคอร์เนลสำหรับระบบไบโอเมตริก</u>

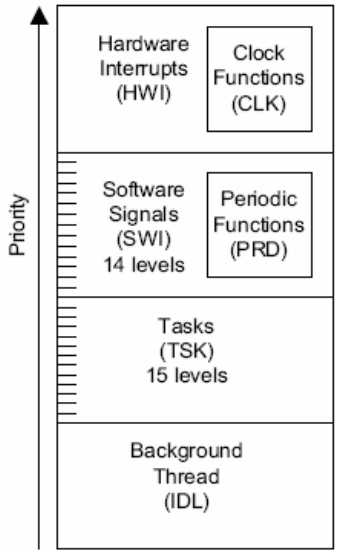
ระบบปฏิบัติการไบโอเมตริกบนตัวประมวลผลสัญญาณดิจิทัลนั้นมีการทำงานอยู่หลายอย่าง ได้แก่ การประมวลผลภาพลายนิ้วมือ การติดต่อผู้ใช้งาน การอ่านค่าลายนิ้วมือ การควบคุมการเปิดประตู การจัดการฐานข้อมูล และการสื่อสารผ่านพอร์ตอนุกรม RS232 เป็นต้น ดังนั้นการเขียนโปรแกรมระบบไบโอเมตริกในลักษณะเวลาจริง (Real-time) บนตัวประมวลผลสัญญาณดิจิทัลทำได้โดยการออกแบบโปรแกรมให้เป็นแบบหลายงานสายโยงใย (Multi-thread) นั้นคือระบบจะแบ่งงานทั้งหมดออกเป็นงานสายโยงใยย่อย (Thread) ในที่นี้จะเรียกว่า งานย่อย เพื่อให้สั่นและได้ใจความ แล้วจัดลำดับความสำคัญและการทำงานของแต่ละงานย่อย ทำให้งาน

ต่าง ๆ ในระบบสามารถทำงานได้อย่างมีประสิทธิภาพ โดยที่ตัวประมวลผลสัญญาณดิจิทัลของบริษัท Texas Instruments ได้เตรียม DSP/BIOS มาช่วยให้ผู้พัฒนาโปรแกรมสามารถจัดลำดับการทำงานและความสำคัญของงานในระบบได้สะดวกขึ้น ซึ่งในหัวข้อนี้จะกล่าวถึงวิธีการออกแบบและเขียนโปรแกรมบนตัวประมวลผลสัญญาณดิจิทัลของระบบไบโอเมตริกโดยใช้ DSP/BIOS ซึ่งจะกล่าวในรายละเอียดของส่วนต่าง ๆ ดังต่อไปนี้

4.1 หลักการจัดการงานย่อยบนตัวประมวลผลสัญญาณดิจิทัล

งานทางด้านประมวลผลแบบเวลาจริงส่วนใหญ่มักมีการแบ่งงานย่อยหลายงานที่สามารถทำงานได้ในช่วงเวลาเดียวกัน โดยงานย่อยต่าง ๆ ที่จะทำงานได้นั้นขึ้นอยู่กับเหตุการณ์ภายนอกหรือได้ข้อมูลมาจากงานอื่นหรือถูกสั่งให้ทำงาน ซึ่งจะมีเคอร์เนลของระบบมาทำการจัดลำดับหรือจัดทรัพยากรสำหรับงานนั้น ๆ สำหรับตัวประมวลผลสัญญาณดิจิทัลของบริษัท Texas Instruments สามารถกำหนดงานต่าง ๆ ให้เป็นงานย่อย (Subroutine) หรืองานบริการสัญญาณขัดจังหวะ (Interrupt Service Routine, ISR) หรือฟังก์ชันย่อยที่ถูกเรียก (Function Call) ได้

การควบคุมการทำงานของระบบแบบหลายงานโยงใยย่อย (Multi-thread) นั้น แต่ละงานย่อย (Thread) จะต้องกำหนดระดับความสำคัญ (Priority) ซึ่งจะเหมือนกันหรือต่างกันก็ได้ขึ้นอยู่กับความเหมาะสมของงาน โดยแต่ละงานย่อยจะมีความสามารถยึดครอง (Preemption) ขึ้นอยู่กับความสำคัญของแต่ละงานย่อย ตัวอย่างเช่น งานย่อยที่มีความสำคัญสูงกว่าจะสามารถทำงานได้ทันทีแม้ว่าจะมีงานย่อยที่มีความสำคัญต่ำกว่าทำงานอยู่หรือไม่ก็ตาม ซึ่งงานย่อยที่มีความสำคัญต่ำกว่าจะต้องรอให้งานย่อยที่มีความสำคัญสูงกว่าที่งานเสร็จสิ้นก่อน สำหรับตัวประมวลผลสัญญาณดิจิทัลของบริษัท Texas Instruments นั้น ได้แบ่งประเภทของงานย่อยออกเป็น 4 ประเภท (เรียงจากความสำคัญสูงสุดไปต่ำสุด) ได้แก่ HWI (Hardware Interrupts) SWI (Software Interrupts) TSK (Tasks) และ IDL (Background Thread) ซึ่งแสดงได้ดังภาพที่ 63



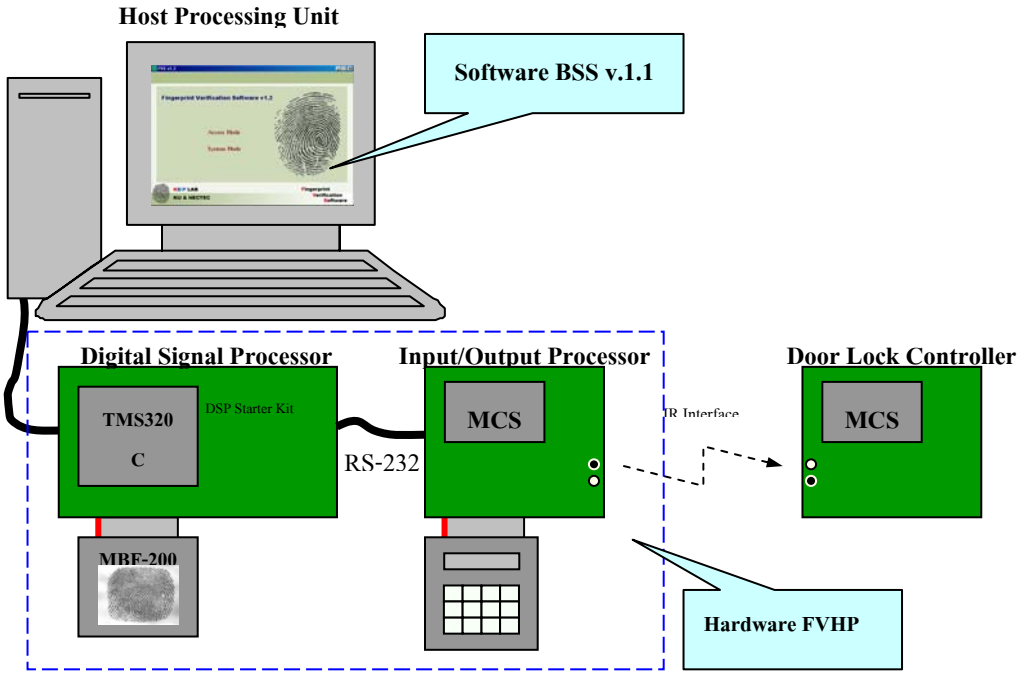
ภาพที่ 63 ลำดับความสำคัญของงานย่อย (Thread)
ที่มา: Texas Instruments Inc. (2001)

จากภาพที่ 63 งานย่อยที่เป็น HWI จะมีความสำคัญสูงที่สุด เนื่องจากเป็นงานที่ต้องใช้เวลาน้อยที่สุดในการทำงาน ในขณะที่งานย่อยที่เป็น IDL จะมีความสำคัญต่ำสุด ในส่วนของงานย่อยที่เป็น SWI และ TSK นั้นจะมีการแบ่งระดับภายในย่อยได้อีก โดยที่ SWI จะแบ่งได้ 14 ระดับ และ TSK จะแบ่งย่อยได้ 15 ระดับ เพื่อเพิ่มความยืดหยุ่นของการออกแบบระบบในกรณีที่ประเภทของงานย่อยหลายงานเหมือนกัน แต่ต้องการให้มีการเข้ายึด (Preempt) กันเองได้ CLK และ PRD เป็นฟังก์ชันของการทำงานในลักษณะเป็นไปตามเวลา ซึ่งเป็นส่วนหนึ่งของงานย่อยใน HWI และใน SWI ตามลำดับ ซึ่งสามารถใช้กับงานที่มีลักษณะเป็นคาบเวลาได้

4.2 การออกแบบระบบไบโอเมตริก

การออกแบบระบบไบโอเมตริกโดยใช้เครื่องมือ DSP/BIOS จะเริ่มจากการกำหนดคุณลักษณะโดยรวมของระบบก่อน ได้แก่ การกำหนดชนิดของตัวประมวลผล ความเร็วสูงสุดในการทำงานของระบบ การจัดแบ่งหน่วยความจำของระบบ หรือจัดแบ่งงานต่าง ๆ ที่ต้องจัดการในระบบ เป็นต้น

ระบบไบโอเมตริกที่ได้ออกแบบไว้นั้นสามารถแสดงได้ดังภาพที่ 64 ซึ่ง IOP (Inpu/Output Processor) จะทำหน้าที่ติดต่อกับผู้ใช้งาน เช่น แสดงผลที่หน้าจอ LCD (Liquid Crystal Display) หรือรับหมายเลข PIN (Personal Identification Number) หรือรหัสผ่าน จากนั้น IOP จะส่งคำสั่งสื่อสารมายัง DSP (Digital Signal Processor) ขึ้นอยู่กับเหตุการณ์ที่เกิดขึ้นเช่นเดียวกัน ในกรณีที่เชื่อมต่อกับระบบใหญ่ระบบก็สามารถที่จะถูกควบคุมได้ผ่านทาง HPU เช่น การตั้งค่าเริ่มต้น (Reset) การส่งฐานข้อมูลจาก HPU มายัง DSP เป็นต้น นอกจากนี้บอร์ด DSP จะต้องทำหน้าที่ประมวลสัญญาณขัดจังหวะที่ส่งมาจากอุปกรณ์กราดตรวจลายนิ้วมืออีกด้วย



ภาพที่ 64 โครงสร้างของระบบไบโอเมตริก

จากภาพจะเห็นได้ว่าการทำงานของ DSP จะถูกสั่งจากคำสั่งสื่อสารภายนอกเป็นหลัก ดังนั้นหากไม่สามารถประมวลผลคำสั่งสื่อสารได้ทันเวลาจริงระบบอาจจะทำงานผิดพลาดได้ ซึ่งการออกแบบซอฟต์แวร์ควรจะจัดลำดับความสำคัญของการสื่อสารกับ HPU และ IOP ให้อยู่ในระดับต้น ๆ ในขณะที่การประมวลภาพลายนิ้วมือซึ่งใช้เวลาในการประมวลผลมากกว่าควรจะถูกจัดให้มีลำดับความสำคัญรองลงมา จากตารางที่ 6 เป็นการกำหนดลำดับความสำคัญของงานต่าง ๆ โดยผ่านทางการเลือกใช้มอดูล DSP/BIOS แต่ละชนิด ซึ่งงานย่อยที่เป็น HWI จะมีลำดับความสำคัญสูงที่สุด รองลงมาจะเป็นงานย่อยที่เป็น SWI TSK และ IDL ตามลำดับ

ตารางที่ 6 การกำหนดมอดูล DSP/BIOS ให้กับงานย่อยต่าง ๆ

ลักษณะงาน	DSP/BIOS module ที่เลือกใช้
การประมวลผลสัญญาณขัดจังหวะจากอุปกรณ์กราดตรวจลายนิ้วมือ	HWI
การรับข้อมูลบิตจาก HPU ไว้ในบัฟเฟอร์ (Buffer) เพื่อรอการประมวลผล	HWI
การรับข้อมูลบิตจาก IOP ไว้ใน Buffer เพื่อรอการประมวลผล	HWI
การส่งข้อมูลบิตจาก Buffer ออกไปยัง HPU	HWI
การส่งข้อมูลบิตจาก Buffer ออกไปยัง IOP	HWI
การตีความคำสั่งสื่อสารจาก HPU และประมวลผล	SWI
การตีความคำสั่งสื่อสารจาก IOP และประมวลผล	SWI
การเขียนคำสั่งสื่อสารลงในบัฟเฟอร์ (Buffer) เพื่อรอการส่งออกไปยัง HPU	SWI
การเขียนคำสั่งสื่อสารลงในบัฟเฟอร์ (Buffer) เพื่อรอการส่งออกไปยัง IOP	SWI
การอ่านภาพลายนิ้วมือจากอุปกรณ์กราดตรวจลายนิ้วมือ	SWI
การจัดการระบบ เมื่อเปิดเครื่อง	TSK
การตรวจสอบลายนิ้วมือ	TSK

การทำงานของระบบโดยทั่วไปจะทำงานอยู่ใน 2 โหมด ได้แก่ โหมดใช้งาน (Accession Mode) และโหมดควบคุมระบบ (System Mode) ในโหมดใช้งานนั้นระบบจะรอคำสั่งสื่อสารหรือสัญญาณขัดจังหวะจากอุปกรณ์กราดตรวจลายนิ้วมือ เช่น เมื่อได้รับการกดหมายเลข PIN ของผู้ใช้ ระบบจะทำการตรวจสอบหมายเลข PIN นั้น ๆ และตรวจสอบว่าเป็นการลงทะเบียนครั้งแรก (Registration) หรือไม่ ถ้าใช่ก็จะทำการลงทะเบียนฐานข้อมูลไว้ แต่ถ้าหมายเลข PIN นั้นได้ถูกลงทะเบียนไว้แล้ว ระบบจะพิจารณาว่าเป็นการร้องขอการผ่านเข้าใช้งานและจะทำการตรวจสอบลายนิ้วมือ (Verification) ต่อไป หรือเมื่อได้รับสัญญาณขัดจังหวะจากอุปกรณ์กราดตรวจลายนิ้วมือระบบก็จะทำการตรวจสอบลายนิ้วมือกับฐานข้อมูลลายนิ้วมือทั้งหมดที่มีอยู่ในระบบ (Identification) เป็นต้น ส่วนโหมดควบคุมระบบจะสามารถเข้าถึงได้โดยผู้ดูแลระบบเท่านั้น โดยในโหมดนี้ระบบจะรอคำสั่งจาก IOP (Input/Output Processor) เช่น คำสั่งขอดูฐานข้อมูลที่มีอยู่ในระบบ ซึ่งระบบจะส่งข้อมูลออกไปเพื่อให้แสดงผลลัพธ์ที่หน้าจอ LCD เป็นต้น

สำหรับการจัดการโหมดของระบบนั้นจะมีตัวแปรส่วนกลาง (Global Variable) เป็นตัวบอกสถานะของระบบว่าอยู่ในสถานะโหมดใช้งานหรือโหมดควบคุมระบบ โดยที่ตัวแปรที่ใช้เก็บสถานะในระบบไบโอเมตริกจะใช้แสดงสถานะการทำงานปัจจุบันของระบบ นอกจาก

สถานะโหมดใช้งานและโหมดควบคุมระบบยังมีการกำหนดสถานะการทำงานภายในระบบด้วย ได้แก่

- ACCESS_MODE เป็นค่าคงที่ที่ใช้แสดงการใช้งานทั่วไป
- SYSTEM_MODE เป็นค่าคงที่ที่ใช้แสดงการใช้งานเกี่ยวกับการจัดการระบบ สำหรับผู้ดูแลระบบเท่านั้น
- SETUP_MODE เป็นค่าคงที่ที่ใช้แสดงการกำหนดค่าพารามิเตอร์ในระบบ
- PROCESS_MODE เป็นค่าคงที่ที่ใช้แสดงการประมวลผลข้อมูล
- DATABASE_MENU เป็นค่าคงที่ที่ใช้แสดงการจัดการข้อมูลเกี่ยวกับฐานข้อมูล
- HISTORY_MENU เป็นค่าคงที่ที่ใช้แสดงการจัดการข้อมูลเกี่ยวกับบันทึกเวลา
- WAIT_FP_MODE เป็นค่าคงที่ที่ใช้แสดงการรอข้อมูลลายนิ้วมือ
- WAIT_FP_PWD เป็นค่าคงที่ที่ใช้แสดงการรอข้อมูลรหัสผ่าน
- WAIT_MODE เป็นค่าคงที่ที่ใช้แสดงการรอคำสั่งทั่วไป
- HALT_MODE เป็นค่าคงที่ที่ใช้แสดงการหยุดระบบ

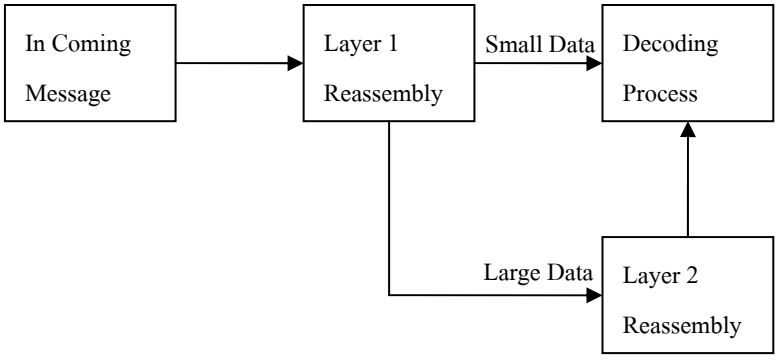
4.3 หลักการทำงานของงานย่อยในระบบไบโอเมตริก

จากการออกแบบและจำแนกประเภทของงานในตารางที่ 6 นั้น ในหัวข้อนี้จะกล่าวถึงรายละเอียดการทำงานของแต่ละงานย่อยที่ออกแบบไว้ เริ่มจากงานที่มีความสำคัญสูงสุดก่อน ดังนี้

4.3.1 การประมวลผลสัญญาณขัดจังหวะจากอุปกรณ์กราดตรวจลายนิ้วมือ (HWI) เป็นฟังก์ชันที่จะทำงานก็ต่อเมื่อได้รับสัญญาณขัดจังหวะจากอุปกรณ์กราดตรวจลายนิ้วมือ เนื่องจากมีการกำหนดงานนี้มีความสำคัญสูงที่สุด ดังนั้นเมื่อระบบได้รับสัญญาณขัดจังหวะจากอุปกรณ์กราดตรวจลายนิ้วมือ (มีการวางนิ้วมือ) ระบบจะหยุดงานทุกชนิดที่กำลังทำอยู่ เพื่อมาทำงานในส่วนของการประมวลผลสัญญาณขัดจังหวะก่อน ซึ่งการเขียนโปรแกรมต้องเขียนให้มีการประมวลผลน้อยที่สุด

4.3.2 การรับส่งข้อมูลผ่านพอร์ตอนุกรม (HWI) เป็นส่วนที่ในการใช้ในการรับหรือส่งข้อมูลไปยัง HPU หรือ IOP โดยจะรับส่งในลักษณะของมาตรฐาน RS-232 ซึ่งในการทำงานจะกำหนดให้มีความสำคัญรองจากการประมวลผลสัญญาณขัดจังหวะเท่านั้น

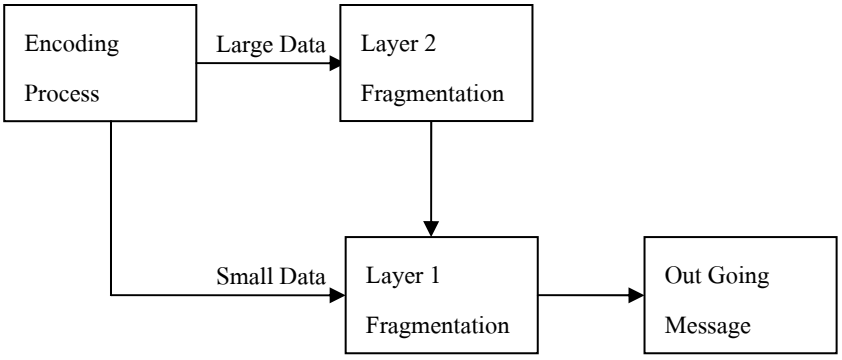
4.3.3 การถอดรหัสข้อความที่รับเข้ามา (SWI) เป็นส่วนที่ใช้รวบรวมและถอดรหัสข้อมูลที่รับเข้ามาทางพอร์ตอนุกรมออกมาเป็นคำสั่งต่าง ๆ ที่ได้ตกลงกันไว้ หลังจากนั้นก็จะทำการประมวลผลตามคำสั่งที่ได้รับเข้ามา ซึ่งจะสามารถแสดงได้ดังภาพที่ 65



ภาพที่ 65 แผนภาพขั้นตอนการถอดรหัสและรับข้อมูล

จากภาพที่ 65 โปรแกรมจะทำการรับข้อความเข้ามายังส่วนของขั้นตอนการรับข้อมูล (Incoming Message) และส่งต่อไปยังส่วนของขั้นตอนการรวมข้อมูลชั้นที่ 1 (Layer 1 Reassembly) เพื่อรวบรวมชุดข้อมูล เนื่องจากการรับส่งข้อมูลแบบ RS-232 ทำให้ตัวประมวลผลสามารถรับข้อมูลได้ทีละ 1 ไบต์หรือ 8 บิต ดังนั้นตัวโปรแกรมจะต้องสามารถรวบรวมชุดคำสั่งทั้งหมดก่อนแล้วจึงส่งไปยังส่วนของการถอดรหัส ในส่วนของการรวบรวมข้อมูลมาเป็นชุดคำสั่งนั้นจะแบ่งเป็นสองระดับ เนื่องจากข้อมูลที่รับเข้ามานั้น จะแบ่งเป็น 2 ประเภท คือ ข้อมูลขนาดเล็ก ได้แก่ ชุดคำสั่งต่าง ๆ และข้อมูลขนาดใหญ่ ได้แก่ ข้อมูลรหัสลายนิ้วมือ โดยที่ข้อมูลขนาดเล็กนั้นจะถูกรวบรวมชุดคำสั่งในส่วนของขั้นตอนการรวมข้อมูลชั้นที่ 1 จากนั้นก็จะส่งไปถอดรหัสที่ขั้นตอนการถอดรหัส (Decoding Process) เลย ในขณะที่ข้อมูลขนาดใหญ่นั้นจะถูกส่งไปยังขั้นตอนการรวมข้อมูลชั้นที่ 2 (Layer 2 Reassembly) ก่อน ซึ่งเป็นในการรวบรวมชุดข้อมูลขนาดเล็กหลายชุดมารวมเป็นชุดข้อมูลขนาดใหญ่จากนั้นจึงส่งไปยังขั้นตอนการถอดรหัสต่อไป

4.3.4 การเข้ารหัสข้อความสำหรับส่งออก (SWI) เป็นส่วนที่ใช้เข้ารหัสชุดข้อมูลสำหรับติดต่อกับอุปกรณ์ภายนอก เช่น HPU หรือ IOP ที่ได้ตกลงกันไว้แล้วส่งออกทางพอร์ตอนุกรม ซึ่งสามารถแสดงได้ดังภาพที่ 66

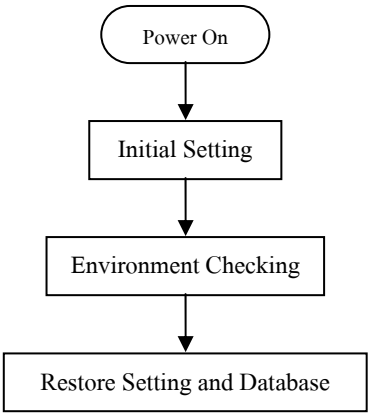


ภาพที่ 66 แผนภาพขั้นตอนการเข้ารหัสและส่งข้อมูล

จากภาพที่ 66 โปรแกรมจะทำการเข้ารหัสชุดข้อมูลที่ต้องการส่งไปยัง HPU หรือ IOP จากนั้นชุดข้อมูลที่เข้ารหัสแล้วจะถูกส่งไปยังขั้นตอนการแยกข้อมูลชั้นที่ 2 (Layer 2 Fragmentation) หรือขั้นตอนการแยกข้อมูลชั้นที่ 1 (Layer 1 Fragmentation) ขึ้นอยู่กับขนาดของข้อมูล โดยที่ข้อมูลขนาดใหญ่จะถูกส่งไปยังขั้นตอนการแยกข้อมูลชั้นที่ 2 เพื่อทำการแบ่งข้อมูลให้เหมาะสมก่อนที่จะส่งออกไป ในขณะที่ข้อมูลขนาดเล็กจะถูกส่งมายังขั้นตอนการแยกข้อมูลชั้นที่ 1 เลยเพื่อลดเวลาในการประมวลผล จากนั้นจึงส่งไปยังส่วนของขั้นตอนการส่งข้อมูล (Out Going Message) ซึ่งข้อมูลจะถูกส่งออกไปด้วยมาตรฐาน RS-232 ต่อไป

4.3.5 การอ่านภาพลายนิ้วมือจากอุปกรณ์กราดตรวจลายนิ้วมือ (SWI) เป็นฟังก์ชันสำหรับการอ่านค่าของลายนิ้วมือจากตัวกราดตรวจลายนิ้วมือ เนื่องจากการจำกัดด้านเวลาประมวลผลของ HWI ทำให้หลังจากมีการรับสัญญาณขัดจังหวะมาจากตัวกราดตรวจลายนิ้วมือ HWI จะประมวลผลเพียงเล็กน้อยและส่งมาประมวลผลต่อที่ SWI ซึ่งในส่วนนี้จะทำการอ่านค่าลายนิ้วมือทีละค่าจากตัวกราดตรวจลายนิ้วมือ

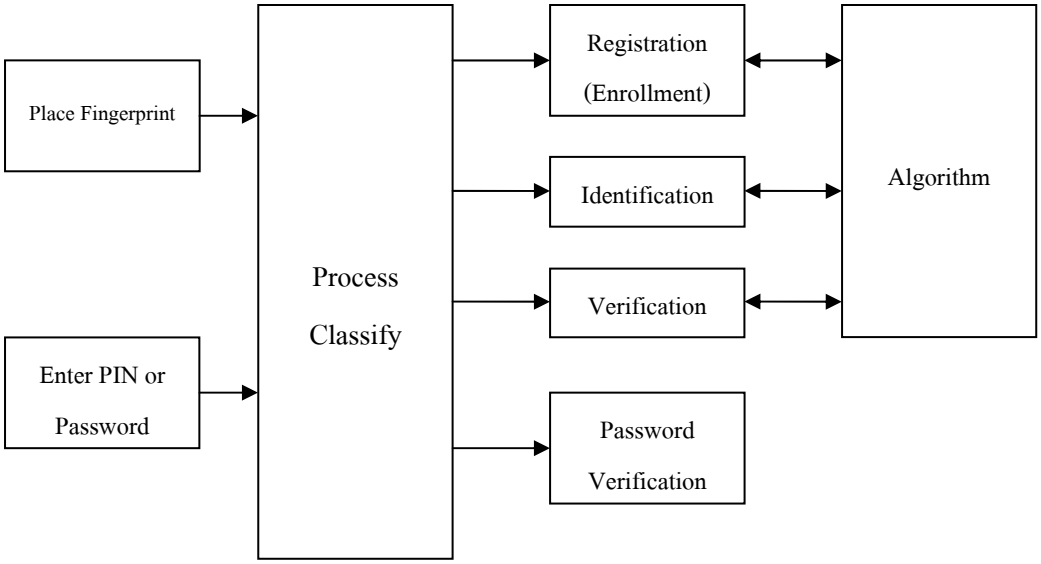
4.3.6 การจัดการระบบเมื่อเปิดเครื่อง (TSK) เป็นงานย่อยสำหรับจัดการระบบในช่วงเปิดเครื่อง ซึ่งจะทำงานเพียงครั้งเดียวต่อการเปิดเครื่อง 1 ครั้งเท่านั้น โดยที่ขั้นตอนการทำงานของการจัดระบบเมื่อเปิดเครื่องแสดงได้ดังภาพที่ 67



ภาพที่ 67 แผนภาพขั้นตอนการจัดระบบเมื่อเปิดเครื่อง

จากภาพที่ 67 เมื่อทำการเปิดเครื่องขึ้นมานั้นงานแรกสุดของระบบที่จะทำงานก็คือ งานการจัดการระบบเมื่อเปิดเครื่อง ซึ่งระบบจะกำหนดค่าเริ่มต้นต่าง ๆ ที่ต้องใช้ในระบบ ต่อจากนั้นระบบจะทำการตรวจสอบอุปกรณ์ภายนอกว่ามีการเชื่อมต่อกันอยู่หรือไม่ เพื่อจำกัดการทำงานของระบบหรือหยุดการทำงานเมื่อไม่มีการเชื่อมต่อกับอุปกรณ์ที่สำคัญ เช่น ตัวควบคุมการเปิด-ปิดประตู ในขั้นสุดท้ายจะเป็นการกู้ค่าตัวแปรระบบและฐานข้อมูลคืนมา ซึ่งใช้ในกรณีที่ไฟดับ โดยที่ค่าต่าง ๆ จะถูกเก็บไว้ที่หน่วยความจำถาวร

4.3.7 การตรวจสอบลายนิ้วมือ (TSK) เป็นงานย่อยสำหรับควบคุมการทำงานของระบบไบโอเมตริกที่เกี่ยวข้องกับการตรวจสอบลายนิ้วมือ โดยใช้ขั้นตอนวิธี (Algorithm) ที่พัฒนาโดยห้องวิจัย KSIP Lab ในการลงทะเบียนลายนิ้วมือ (Enrollment) การระบุลายนิ้วมือ (Identification) และการตรวจสอบลายนิ้วมือ (Verification) โดยจะมีกระบวนการในการพิจารณาว่าผู้ใช้ต้องการทำงานในรูปแบบใด ซึ่งกระบวนการทำงานของการตรวจสอบลายนิ้วมือสามารถแสดงได้ดังภาพที่ 68



ภาพที่ 68 แผนภาพขั้นตอนการตรวจสอบลายนิ้วมือ

จากภาพที่ 68 การตรวจสอบลายนิ้วมือนั้นจะทำงานเมื่อได้รับข้อมูลหรือคำสั่งจากขั้นตอนการวางลายนิ้วมือ (Push Fingerprint) หรือขั้นตอนการกด PIN หรือรหัสผ่าน (Enter PIN or Password) โดยที่ในส่วนขั้นตอนการวางลายนิ้วมือจะทำงานเมื่อระบบรับรู้ว่ามีการวางนิ้วมือลงบนตัวกราดตรวจลายนิ้วมือผ่านทางสัญญาณขัดจังหวะ ซึ่งอธิบายไว้หัวข้อการประมวลผลสัญญาณขัดจังหวะของตัวกราดตรวจลายนิ้วมือ ในขณะที่ส่วนของขั้นตอนการกด PIN หรือรหัสผ่านจะทำงานเมื่อระบบได้รับค่า PIN หรือรหัสผ่านมาจาก IOP จากนั้นระบบจะเลือกประเภทการทำงานโดยพิจารณาจากข้อมูลที่เข้ามาและสถานะในขณะนั้น ตัวอย่างเช่น ขณะที่ระบบอยู่ในสถานะโหมดใช้งานนั้นเมื่อผู้ใช้ใส่รหัส PIN และวางนิ้วมือลงบนเครื่องกราดตรวจลายนิ้วมือ ระบบจะทำงานแบบตรวจสอบลายนิ้วมือที่ได้เปรียบเทียบกับข้อมูลลายนิ้วมือของผู้ใช้ที่มีรหัส PIN นั้น ๆ ซึ่งจะเรียกว่า การตรวจสอบลายนิ้วมือ (Verification) หรือในกรณีที่ผู้ใช้งานวางนิ้วมือบนตัวกราดตรวจลายนิ้วมือเพียงอย่างเดียว ระบบจะเปลี่ยนการทำงานเป็นแบบตรวจสอบลายนิ้วมือที่ได้เปรียบเทียบกับผู้ใช้ทั้งหมด (ตาม PIN) ที่เคยลงทะเบียนไว้ ซึ่งเรียกว่า การระบุลายนิ้วมือ (Identification) เป็นต้น

ประเภทการทำงานของระบบไบโอเมตริกนั้นแบ่งได้ออกเป็น 4 ประเภท ได้แก่ การลงทะเบียน (Registration) การระบุลายนิ้วมือ (Fingerprint Identification หรือ Identification) การตรวจสอบลายนิ้วมือ (Fingerprint Verification หรือ Verification) การตรวจสอบรหัสผ่าน (Password Verification) โดยที่การลงทะเบียน การระบุลายนิ้วมือ และการตรวจสอบ

ลายนิ้วมือเป็นการทำงานที่เกี่ยวข้องกับการประมวลผลภาพ ดังนั้นจึงมีการเรียกใช้ขั้นตอนวิธี (Algorithm) ในการตรวจสอบลายนิ้วมือด้วย ในส่วนของขั้นตอนวิธีนั้นเป็นส่วนที่สามารถพัฒนาได้อย่างอิสระ โดยจะมีการกำหนดรูปแบบการติดต่อกับขั้นตอนวิธีไว้ก่อน ส่วนการตรวจสอบรหัสผ่านนั้นจะถูกใช้เมื่อผู้ใช้เป็นผู้ควบคุมระบบเท่านั้น ซึ่งสามารถใช้ได้ในกรณีฉุกเฉินที่ไม่ต้องการตรวจสอบลายนิ้วมือ

4.4 การจัดการฐานข้อมูลผู้ใช้และบันทึกเวลา

ระบบฐานข้อมูลในระบบไบโอเมตริกสามารถแบ่งเป็นสองประเภทคือ ฐานข้อมูลผู้ใช้และฐานข้อมูลบันทึกเวลา โดยการเก็บข้อมูลจะอยู่ในลักษณะอ็อบเจกต์ข้อมูล ซึ่งภายในจะประกอบด้วยฟิลด์ข้อมูลต่าง ๆ ซึ่งลักษณะโครงสร้างของข้อมูลผู้ใช้จะมีการเก็บค่าทั่วไปเกี่ยวกับผู้ใช้งาน แสดงได้ดังตารางที่ 7

ตารางที่ 7 โครงสร้างฐานข้อมูลผู้ใช้ในระบบไบโอเมตริกที่พัฒนาขึ้น

ชื่อเขตข้อมูล	ข้อมูลที่ถูกบันทึก
pin	หมายเลข PIN ของผู้ใช้
name	ชื่อของผู้ใช้
lastname	นามสกุลของผู้ใช้
pwd	รหัสผ่านของผู้ใช้
fp_ptr	ตัวชี้แม่แบบลายนิ้วมือของผู้ใช้
fp_size	ขนาดแม่แบบลายนิ้วมือของผู้ใช้
first_time	สถานะการลงทะเบียนของผู้ใช้
admin_user	สถานะการใช้งานระบบของผู้ใช้
idenPermit	สถานะอนุญาตในการนำลายนิ้วมือมาระบุตัวบุคคลของผู้ใช้

ในระบบฐานข้อมูลของผู้ใช้จะแบ่งข้อมูลเป็นสองส่วน คือ ข้อมูลทั่วไปผู้ใช้งาน (Profiles) และข้อมูลแม่แบบลายนิ้วมือของผู้ใช้ (Templates) โดยในฟิลด์ข้อมูลทั่วไปของผู้ใช้จะมีฟิลด์ fp_ptr เป็นตัวชี้ไปยังข้อมูลแม่แบบลายนิ้วมือของผู้ใช้ และมีฟิลด์ fp_size บอกขอบเขตของแม่แบบลายนิ้วมือ

สำหรับข้อมูลบันทึกเวลาเป็นการทำการบันทึกเวลาการเข้าใช้งานของผู้ใช้ในระบบเมื่อสามารถเข้าระบบได้ ซึ่งโครงสร้างข้อมูลนั้นจะเก็บ PIN ชื่อ และเวลาเข้าออกเท่านั้น เพื่อใช้ในการแสดงผล

การจัดการเกี่ยวกับฐานข้อมูลนั้นโปรแกรมของระบบได้เตรียมฟังก์ชันการจัดการไว้ ซึ่งในการจัดการฐานข้อมูลผู้ใช้จะเรียกฟังก์ชันจากไฟล์ DatabaseMan.h และการจัดการฐานข้อมูลบันทึกเวลาจะเรียกฟังก์ชันจากไฟล์ HistMan.h

4.5 ความน่าเชื่อถือของระบบ

ในการพัฒนาระบบไบโอเมตริกเพื่อนำไปผลิตในเชิงพานิชย์นั้น สิ่งหนึ่งที่สำคัญที่ต้องคำนึงถึงคือความน่าเชื่อถือของระบบ เช่น ในกรณีไฟดับหรือระบบขัดข้องหรือมีปัญหา ตัวระบบจะต้องสามารถกลับมาทำงานได้เหมือนเดิม โดยในระบบไบโอเมตริกที่พัฒนาขึ้นมานั้น ได้มีการสำรองข้อมูล (Data Backup) เก็บเอาไว้ ซึ่งเมื่อระบบมีปัญหาจะสามารถทำการการตั้งค่าเริ่มต้นข้อมูลให้กลับมาทำงานได้เหมือนเดิม

การสำรองข้อมูลในระบบไบโอเมตริกจะมีการสำรองข้อมูล 2 ที่ คือ การสำรองข้อมูลบนเครื่องแม่ข่าย (HPU) และการสำรองข้อมูลที่ตัวลูกข่าย (RAU) ทำให้ระบบที่ออกแบบมีความน่าเชื่อถือสูง

สำหรับการสำรองข้อมูลบนเครื่องแม่ข่ายนั้น จะทำได้ก็ต่อเมื่อระบบถูกตั้งค่าให้เป็นแบบเครือข่าย โดยข้อมูลที่ส่งไปสำรองบนเครื่องแม่ข่าย คือ ฐานข้อมูลผู้ใช้และฐานข้อมูลบันทึกเวลา โดยข้อมูลฐานข้อมูลผู้ใช้จะถูกส่งไปสำรองที่เครื่องแม่ข่ายเมื่อระบบมีเปลี่ยนแปลงฐานข้อมูล ส่วนฐานข้อมูลบันทึกเวลาจะถูกส่งไปสำรองที่เครื่องแม่ข่ายเมื่อมีผู้ใช้เข้าใช้งานระบบ

การสำรองข้อมูลที่ตัวลูกข่าย (RAU) ระบบจะทำการสำรองข้อมูลเก็บเข้าไปไว้ในหน่วยความจำถาวรของระบบ ได้แก่ หน่วยความจำแฟลช (Flash Memory) โดยข้อมูลที่จะทำการสำรองสามารถแบ่งได้ 4 ส่วน คือ ข้อมูลการปรับแต่งของระบบ (System Configuration Data) ข้อมูลทั่วไปผู้ใช้งาน (Profiles) ข้อมูลแม่แบบลายนิ้วมือของผู้ใช้ (Templates) และข้อมูลบันทึกเวลา (History)

การทำงานของซอฟต์แวร์ระบบในการสำรองข้อมูลนั้น โปรแกรมจะเตรียมมอดูล PRD ชื่อ systemMan ในการนับเวลาเพื่อตรวจสอบว่าระบบไม่มีการทำงาน เมื่อระบบไม่ได้ทำงานเป็นเวลาที่กำหนดไว้ ระบบจะตรวจสอบว่ามีการเปลี่ยนแปลงข้อมูลที่จะสำรองหรือไม่ ถ้ามีก็จะทำการสำรองข้อมูลใหม่โดยการส่งการทำงานไปยังมอดูล TSK ชื่อ storeData เพื่อทำการสำรองข้อมูลต่าง ๆ

เมื่อระบบมีการตั้งค่าเริ่มต้น (Reset) หรือไฟดับ ซอฟต์แวร์จะเริ่มทำงานแรกที่มอดูล TSK ชื่อ powerup ซึ่งฟังก์ชันในการดึงข้อมูลจากหน่วยความจำแฟลชที่เก็บไว้ในช่วงก่อนการตั้งค่าเริ่มต้น ทำให้ระบบสามารถทำงานได้เหมือนหรือใกล้เคียงกับช่วงก่อนการตั้งค่าเริ่มต้นหรือไฟดับ

จากที่กล่าวมาในหัวข้อการออกแบบระบบปฏิบัติการสำหรับไบโอเมตริกนั้น ผู้ออกแบบจะต้องแบ่งการทำงานของระบบต่าง ๆ ออกเป็นงานย่อยที่อิสระจากกันและมีการส่งผ่านข้อมูลระหว่างกัน โดยชุดเครื่องมือรวมสำหรับเขียนโปรแกรม (Code Composer Studio) นั้นได้เตรียมความสามารถสำหรับติดต่อสื่อสารระหว่างงานย่อยต่าง ๆ ไว้ให้

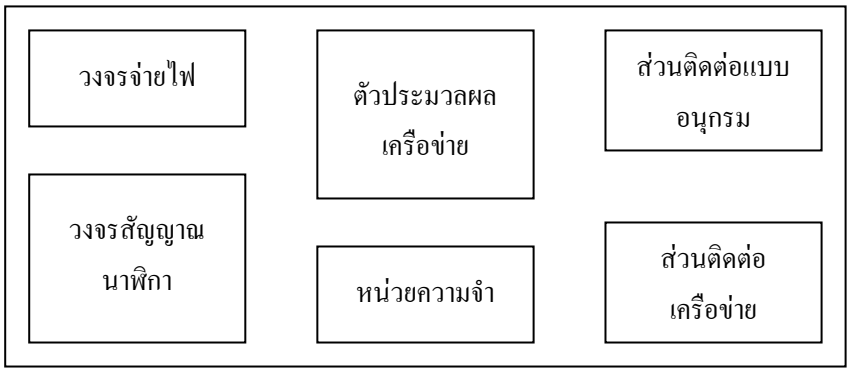
สำหรับการเขียนซอฟต์แวร์ของระบบไบโอเมตริก นอกจากการแบ่งงานย่อยต่าง ๆ แล้ว ผู้พัฒนาจะต้องเขียนโปรแกรมในส่วนการจัดการฐานข้อมูลและความน่าเชื่อถือของระบบด้วย

<u>การออกแบบฮาร์ดแวร์เชื่อมต่อเครือข่าย</u>

ฮาร์ดแวร์เชื่อมต่อเครือข่ายเป็นฮาร์ดแวร์ที่ใช้ในการเชื่อมต่อระหว่างฮาร์ดแวร์ประมวลผลสัญญาณดิจิทัลกับเครือข่ายภายนอก ซึ่งลักษณะการเชื่อมต่อของฮาร์ดแวร์ทั้งหมดสามารถแสดงได้ดังภาพที่ 69 โดยจะทำการแปลงรูปแบบการติดต่อแบบเครือข่ายให้เป็นการรับส่งแบบอนุกรม เนื่องจากส่วนติดต่ออุปกรณ์ภายนอกของฮาร์ดแวร์ประมวลผลสัญญาณดิจิทัลเป็นแบบพอร์ตอนุกรม

<u>ภาพที่ 69</u> โครงสร้างการเชื่อมต่อฮาร์ดแวร์ในระบบไบโอเมตริก

สำหรับฮาร์ดแวร์เชื่อมต่อเครือข่ายได้ถูกออกแบบลักษณะเหมือนบอร์ดประมวลผลทั่วไป ซึ่งมีส่วนประกอบต่าง ๆ ได้แก่ ตัวประมวลผล หน่วยความจำ พอร์ตติดต่อกับอุปกรณ์ภายนอก วงจรจ่ายไฟ วงจรสัญญาณนาฬิกา เป็นต้น ดังภาพที่ 70



ภาพที่ 70 สถาปัตยกรรมของฮาร์ดแวร์เชื่อมต่อเครือข่าย

จากภาพที่ 70 ตัวประมวลผลเครือข่ายจะทำหน้าที่แปลงข้อมูลระหว่างการรับส่งข้อมูลแบบอนุกรมและการรับส่งข้อมูลผ่านเครือข่ายผ่านทางส่วนติดต่อแบบอนุกรม (Serial Interface) และส่วนติดต่อเครือข่าย (Network Interface) ตามลำดับ ซึ่งในการประมวลผลจะใช้หน่วยความจำมาเก็บข้อมูลชั่วคราวด้วย วงจรสัญญาณนาฬิกาทำหน้าที่ส่งสัญญาณนาฬิกาให้กับตัวประมวลผลเครือข่าย และวงจรจ่ายไฟทำหน้าที่ป้อนกระแสไฟให้กับอุปกรณ์ทุกตัวที่อยู่ในบอร์ด

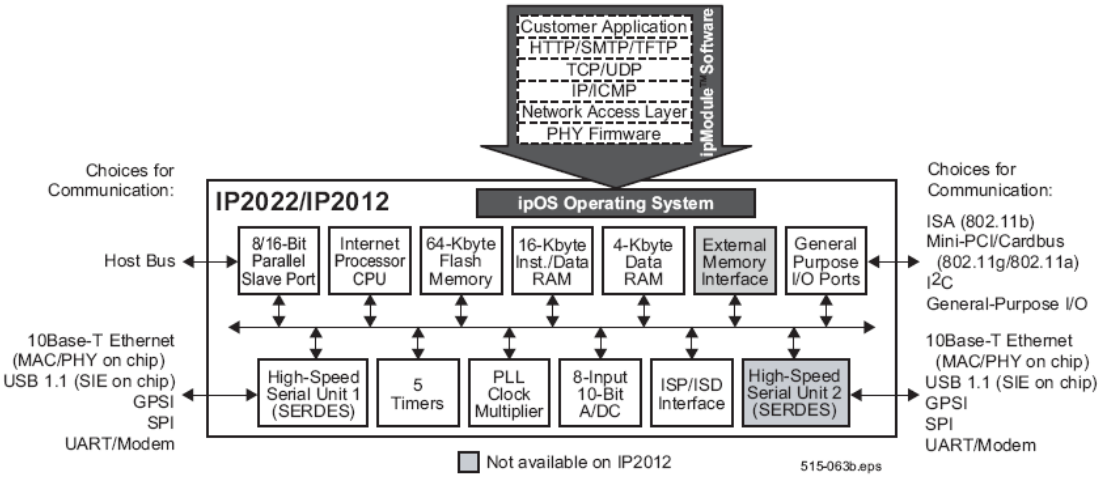
ตัวประมวลผลเครือข่ายเป็นตัวประมวลผลที่สามารถติดต่อแบบเครือข่ายได้ ซึ่งในท้องตลาดมีอยู่หลายบริษัท ดังตารางที่ 8

ตารางที่ 8 เปรียบเทียบชนิดของตัวประมวลผลเครือข่าย

ชนิดของตัวประมวลผล	ราคาต่อ 1 หน่วย (บาท)	ต้องการตัวควบคุมอีเธอร์เน็ตเพิ่ม	หน่วยความจำ	มีชุดโพรโทคอล TCP/IP
MC68EZ328	480.00	ต้องการ	2 MB	ไม่มี
16F877	360.00	ต้องการ	112 KB	ไม่มี
IP2022	570.00	ไม่ต้องการ	64 KB	มี
RCM2000	360.00	ต้องการ	512 KB	มี

จากตารางที่ 8 ตัวประมวลผลเครือข่ายมีอยู่หลายชนิด ได้แก่ MC68EZ328 ของบริษัท Motorola, 16F877 ของบริษัท Microchip, IP2022 ของบริษัท Ubicom และ RCM2000 ของบริษัท rabbit เป็นต้น ในวิทยานิพนธ์นี้ได้เลือกตัวประมวลผลเครือข่าย IP2022 ของบริษัท Ubicom เนื่องจากใช้งานง่ายกว่า คือ ไม่ต้องการตัวควบคุมอีเธอร์เน็ตเพิ่ม ทำให้ลดฮาร์ดแวร์ในส่วนจัดการอีเธอร์เน็ต และมีชุดโพรโทคอล TCP/IP อยู่ภายในทำให้สามารถพัฒนาโปรแกรมได้เร็ว

ตัวประมวลผลเครือข่าย IP2022 เป็นตัวประมวลผลแบบ RISC ประสิทธิภาพ 120 MIPS มีพอร์ตอนุกรม 2 พอร์ตสามารถเลือกติดต่อแบบต่าง ๆ ได้ ดังภาพที่ 71



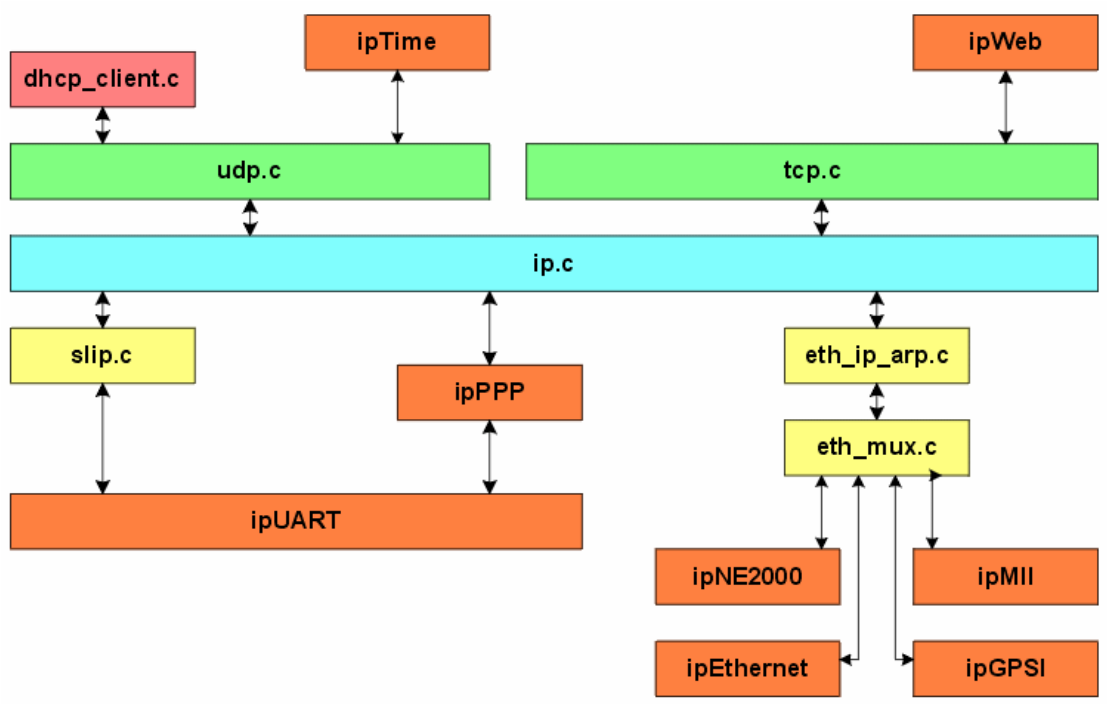
ภาพที่ 71 บล็อกไดอะแกรมของ IP2022
ที่มา: Ubicom Inc. (2001)

จากภาพที่ 71 พอร์ตอนุกรมสามารถเลือกการทำงานได้หลายประเภท เช่น 10Base-T Ethernet, USB1.1, GPSI, SPI หรือ UART ในการทำงานของตัวประมวลผลจะมีระบบปฏิบัติการ ipOS (ipOS Operating System) ซึ่งจะจัดการทรัพยากรต่าง ๆ ภายใน และการเขียนโปรแกรมบนตัวประมวลผลเครือข่าย IP2022 จะมีซอฟต์แวร์ ipModule ซึ่งมีลักษณะเป็น TCP/IP Stack ทำให้ง่ายต่อการพัฒนาโปรแกรมบนเครือข่าย

การทดสอบฮาร์ดแวร์เชื่อมต่อเครือข่าย

หลังจากออกแบบและสร้างฮาร์ดแวร์เชื่อมต่อเครือข่ายแล้ว ขั้นต่อไปจะเป็นการทดสอบฮาร์ดแวร์โดยการเขียนโปรแกรมทดสอบบนฮาร์ดแวร์เครือข่าย ซึ่งจะทดสอบผ่านทางเครือข่ายพื้นที่ท้องถิ่น (Local Area Network)

ในการเขียนโปรแกรมบนตัวประมวลผลเครือข่าย IP2022 ทางบริษัท Ubicom ได้เตรียมชุดเขียนโปรแกรม Unity IDE ซึ่งเป็นชุดโปรแกรมสำหรับพัฒนา แปลโปรแกรม รวมทั้งดาวน์โหลดโปรแกรมลงบนฮาร์ดแวร์ประมวลผลเครือข่าย นอกจากนี้ยังเตรียมมอดูล ipStack ซึ่งมีลักษณะเป็นชั้นโพรโทคอล TCP/IP (TCP/IP Protocol Stack) ดังภาพที่ 72 ทำให้ตัวประมวลผล IP2022 มีความสามารถในการจัดการข้อมูลทางเครือข่ายได้อย่างมีประสิทธิภาพ

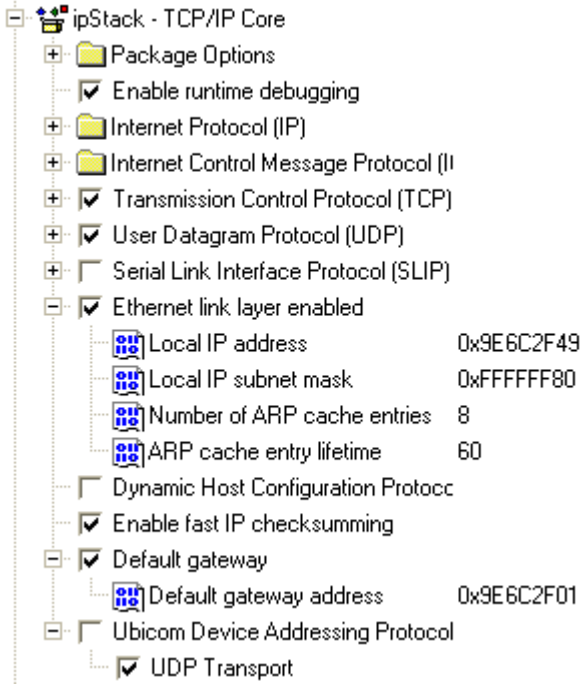


ภาพที่ 72 ลำดับชั้นโปรแกรมของมอดูล ipStack

จากภาพที่ 72 ชุดซอฟต์แวร์จะประกอบด้วยโปรแกรมตัวขับต่าง ๆ ได้แก่ ipUART ipPPP ipEthernet เป็นต้น ซึ่งจะสอดคล้องกับโครงสร้างฮาร์ดแวร์ที่เลือกใช้ เช่น ถ้าต้องการใช้พอร์ตอนุกรมก็จะเลือกใช้ ipUART หรือถ้าต้องการติดต่อสื่อสารผ่านเครือข่ายท้องถิ่นก็ต้องเลือก ipEthernet เป็นต้น จากนั้นโปรแกรมตัวขับจะรับส่งข้อมูลไปยังโปรแกรม ip.c ซึ่งทำงานใน

ลักษณะเดียวกันกับโพรโทคอลอินเตอร์เน็ต (Internet Protocol) จากนั้นข้อมูลจากโพรโทคอลอินเตอร์เน็ตจะส่งไปยังโพรโทคอลชั้นเคลื่อนย้ายข้อมูล ซึ่งได้แก่ TCP และ UDP เป็นต้น

การปรับแต่งข้อมูลบนซอฟต์แวร์มอดูล ipStack สามารถกำหนดได้ผ่านทางส่วนติดต่อผู้ใช้ทางภาพ (Graphic User Interface) หรือ GUI ดังภาพที่ 73 ซึ่งผู้พัฒนาสามารถปรับแต่งข้อมูลใช้ชั้นโพรโทคอลต่าง ๆ ได้



ภาพที่ 73 การปรับแต่งมอดูล ipStack

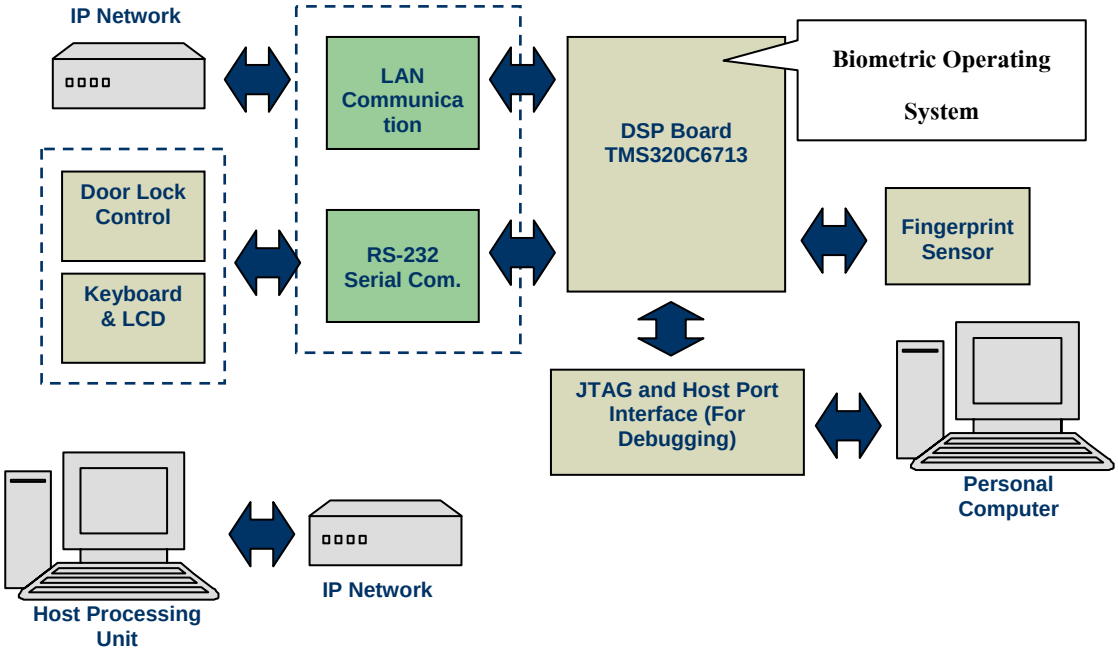
เมื่อทำการติดตั้งฮาร์ดแวร์เชื่อมต่อเครือข่ายแล้ว ผู้ใช้สามารถทดสอบการเชื่อมต่อของฮาร์ดแวร์เชื่อมต่อเครือข่ายได้โดยใช้คำสั่ง ping จากเครื่องคอมพิวเตอร์ในเครือข่ายนั้น ดังภาพที่ 74

```
D:\>ping 158.108.47.73

Pinging 158.108.47.73 with 32 bytes of data:

Reply from 158.108.47.73: bytes=32 time<1ms TTL=128
Reply from 158.108.47.73: bytes=32 time<1ms TTL=128
Reply from 158.108.47.73: bytes=32 time<1ms TTL=128
Reply from 158.108.47.73: bytes=32 time<1ms TTL=128

Ping statistics for 158.108.47.73:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

ภาพที่ 74 ทดสอบการเชื่อมต่อของฮาร์ดแวร์กับเครือข่าย

การทดสอบประสิทธิภาพของระบบโดยรวม

ในการทดสอบประสิทธิภาพของระบบโดยรวมเป็นขั้นตอนสุดท้ายของการพัฒนาระบบปฏิบัติการไบโอเมตริกบนตัวประมวลผลสัญญาณดิจิทัลผ่านทางเครือข่าย โดยนำส่วนต่าง ๆ ที่พัฒนาแล้วมาทำงานร่วมกัน ดังภาพที่ 75



ภาพที่ 75 การเชื่อมต่อของระบบไบโอเมตริกโดยรวม

จากภาพที่ 75 บอร์ด DSP TMS320C6713 ทำหน้าที่ประมวลผลภาพลายนิ้วมือจากตัวกราดตรวจลายนิ้วมือ (Fingerprint Sensor) นอกจากนี้ยังสามารถติดต่อกับชุดควบคุมประตู (Door Lock Controller) และชุดคีย์บอร์ดและจอแสดงผล (Keyboard & LCD) ผ่านทางการสื่อสารอนุกรมแบบ RS232 และติดต่อกับเครือข่าย (IP Network) ผ่านฮาร์ดแวร์เชื่อมต่อเครือข่าย โดยบอร์ด DSP หรือฮาร์ดแวร์ประมวลผลจะทำงานด้วยระบบปฏิบัติการไบโอเมตริก ซึ่งสามารถทดสอบและแก้จุดบกพร่องโปรแกรม (Debug) ด้วยคอมพิวเตอร์ส่วนบุคคล (Personal Computer) ผ่านทาง JTAG ได้ ในขณะที่หน่วยประมวลผลแม่ข่าย (Host Processing Unit) สามารถติดต่อกับฮาร์ดแวร์ประมวลผลผ่านทางเครือข่ายได้เช่นกัน

ผลและวิจารณ์ผลการทดลอง

ระบบตรวจสอบลายนิ้วมือต้นแบบที่ใช้ในการทดสอบได้ถูกออกแบบและพัฒนาขึ้นโดยคณะวิจัย KSIP Lab ร่วมกับหน่วยงานวิจัย NECTEC ซึ่งเป็นฮาร์ดแวร์ที่สร้างขึ้นโดยคนไทยทั้งหมด โดยระบบตรวจสอบลายนิ้วมือต้นแบบดังกล่าวมีคุณลักษณะต่าง ๆ ดังตารางที่ 9

ตารางที่ 9 คุณลักษณะของระบบตรวจสอบลายนิ้วมือ

คุณลักษณะของฮาร์ดแวร์ตรวจสอบลายนิ้วมือต้นแบบ	FVHPv2
ทีมผู้ผลิตและพัฒนา/ประเทศที่ผลิต	KSIP Lab&NECTEC/ ประเทศไทย
รุ่นของฮาร์ดแวร์ตรวจสอบลายนิ้วมือต้นแบบ	รุ่นที่ 2
รูปแบบของตัวกราดตรวจลายนิ้วมือ	Capacitive Sensor MBF200 (Fujitsu)
ตัวประมวลผลบนฮาร์ดแวร์	DSP TMS320C6713 (Texas Instruments)
ความเร็วของตัวประมวลผล	200 MHz
ขนาดสูงสุดของแม่แบบ	1,500 ไบต์
ความเร็วในการส่งจากตัวประมวลผลสัญญาณดิจิทัลผ่านเครือข่าย	< 40 kbps
เวลาในการรับส่งฐานข้อมูลขนาดประมาณ 1,500 ไบต์ผ่านเครือข่าย	2 วินาที
เวลาในการส่งภาพลายนิ้วมือขนาด 76,800 ไบต์ผ่านเครือข่าย	10 วินาที
ขั้นตอนวิธีการตรวจสอบลายนิ้วมือ (Fingerprint Algorithm)	FVS version 1.4

สำหรับการทดสอบการทำงานของระบบตรวจสอบลายนิ้วมือได้ใช้ลายนิ้วมือที่แตกต่างกันในการทดสอบประสิทธิภาพการทำงานของฮาร์ดแวร์ตรวจสอบลายนิ้วมือต้นแบบ ดังตารางที่ 10

ตารางที่ 10 ผลการทดสอบการทำงานของขั้นตอนวิธีตรวจสอบลายนิ้วมือ

คุณลักษณะที่ต้องการทดสอบ	ผลการทดสอบ (sec)		
	เวลาต่ำสุด	เวลาเฉลี่ย	เวลาสูงสุด
เวลาในการลงทะเบียน	2.304	4.289	5.132
เวลาในการตรวจสอบลายนิ้วมือ	2.340	4.419	5.904
เวลาในการจับคู่เปรียบเทียบ	0.001	0.249	0.982
เวลาในการระบุลายนิ้วมือจำนวน 1000 แม่แบบ	2.716	191.621	483.747

ขนาดของซอฟต์แวร์ที่ใช้ในการเขียนโปรแกรมระบบปฏิบัติการไบโอเมตริกบนตัวประมวลผลสัญญาณดิจิทัล สามารถสรุปได้ดังตารางที่ 11

ตารางที่ 11 ขนาดของซอฟต์แวร์ระบบปฏิบัติการไบโอเมตริก

พื้นที่หน่วยความจำ	ขนาดที่ใช้ (ไบต์)
หน่วยความจำภายใน	145,487
หน่วยความจำภายนอก	264,370

การพัฒนาระบบตรวจสอบลายนิ้วมือในอนาคตยังสามารถออกแบบฮาร์ดแวร์สำหรับตัวประมวลผลที่เป็นแบบ Ball Grid Array ซึ่งจะทำให้ความเร็วในการทำงานสูงขึ้นได้เป็นเท่าตัว ส่วนความถูกต้องของการตรวจสอบลายนิ้วมือยังสามารถพัฒนาประสิทธิภาพของซอฟต์แวร์ขั้นตอนวิธีให้ดียิ่งขึ้นได้ ซึ่งกำลังอยู่ในช่วงพัฒนา FVS version 1.5 ที่มีประสิทธิภาพสูงกว่าเดิม เนื่องจากการออกแบบระบบปฏิบัติการที่ยืดหยุ่นทำให้สามารถพัฒนาทั้งฮาร์ดแวร์ประมวลผลและซอฟต์แวร์ขั้นตอนวิธีพร้อมกันได้ (ณัฐพงศ์ และคณะ, 2548)

ในการติดต่อกับคอมพิวเตอร์ส่วนบุคคลโดยใช้ฮาร์ดแวร์เชื่อมต่อเครือข่ายนั้น พบว่าใช้เวลาในการรับส่งข้อมูลมากกว่าประสิทธิภาพสูงสุดของเครือข่ายเนื่องจากข้อมูลที่รับส่งระหว่างฮาร์ดแวร์เชื่อมต่อเครือข่ายและฮาร์ดแวร์ประมวลผลเป็นแบบ RS232 ซึ่งมีความเร็วในการส่งข้อมูลที่จำกัดทำให้เกิดปัญหาคอขวดที่ฮาร์ดแวร์เชื่อมต่อเครือข่าย ซึ่งในอนาคตปัญหานี้สามารถแก้ไขได้โดยการเปลี่ยนวิธีการรับส่งข้อมูลระหว่างฮาร์ดแวร์เชื่อมต่อเครือข่ายและฮาร์ดแวร์ประมวลผลให้เร็วขึ้น เช่น SPI (Serial Peripheral Interface) เป็นต้น

สรุป

งานวิจัยระบบไบโอเมตริกบนตัวประมวลผลสัญญาณดิจิทัลผ่านทางเครือข่ายสามารถแบ่งเป็น 2 งานหลัก ได้แก่ การออกแบบและพัฒนาระบบปฏิบัติการไบโอเมตริกบนตัวประมวลผลสัญญาณดิจิทัล และการออกแบบและสร้างฮาร์ดแวร์เชื่อมต่อเครือข่าย

ในงานแรกเป็นการออกแบบและพัฒนาระบบปฏิบัติการไบโอเมตริกบนตัวประมวลผลสัญญาณดิจิทัล ซึ่งสามารถแบ่งการเชื่อมต่อเป็นสองส่วนคือ ส่วนแรกเป็นตัวกลางเชื่อมต่อส่วนฮาร์ดแวร์ และส่วนที่สองเป็นตัวกลางเชื่อมต่อส่วนซอฟต์แวร์ขั้นตอนวิธีการตรวจสอบไบโอเมตริก โดยส่วนแรกจะใช้เครื่องมือ DSP/BIOS ซึ่งเครื่องมือสำหรับการเขียนโปรแกรมในลักษณะหลายงานสายโยงใย (Multi-thread) และใช้จัดการทรัพยากรต่าง ๆ ภายในระบบ โดยการแบ่งการทำงานต่าง ๆ เป็นงานโยงใยย่อย (Thread) และจัดลำดับ (Scheduling) งานย่อยต่าง ๆ ทำให้ระบบทำงานแบบเวลาจริง (Real-time) ได้ และส่วนที่สองจะใช้มาตรฐานขั้นตอนวิธี (Algorithm Standard) ซึ่งเป็นมาตรฐานในการเรียกใช้ซอฟต์แวร์ขั้นตอนวิธี (Algorithm Software) ทำให้ระบบสามารถจัดสรรทรัพยากรได้อย่างมีประสิทธิภาพ นอกจากนี้ระบบปฏิบัติการไบโอเมตริกนี้ยังจัดรูปแบบการทำงานตามมาตรฐานส่วนต่อประสานโปรแกรมประยุกต์ไบโอเมตริก (Biometric Application Programming Interface) หรือ BioAPI เพื่อสะดวกต่อการนำไปประยุกต์ใช้งานไบโอเมตริก ได้แก่ การลงทะเบียนบุคคล การตรวจสอบบุคคล และการระบุตัวบุคคล เป็นต้น

งานที่สองคือการออกแบบและสร้างฮาร์ดแวร์เชื่อมต่อเครือข่ายสามารถแบ่งได้ 2 ส่วน คือ ส่วนแรกเป็นการพัฒนาฮาร์ดแวร์โดยใช้ตัวประมวลผลเครือข่ายเบอร์ IP2022 ทำการแปลงสัญญาณข้อมูลอนุกรมแบบ UART จากตัวประมวลผลสัญญาณดิจิทัลไปเป็นสัญญาณข้อมูลแบบเครือข่ายโพรโทคอล TCP ส่วนที่สองเป็นการพัฒนาโปรแกรมโดยใช้ชุดซอฟต์แวร์ทับซ้อนของโพรโทคอล TCP (TCP Protocol Stack) บนตัวประมวลผลเครือข่ายในการจัดการข้อมูลจะตัวประมวลผลสัญญาณดิจิทัลและเครือข่ายได้อย่างมีประสิทธิภาพ

จากผลการทดสอบพบว่าระบบตรวจสอบลายนิ้วมือสามารถใช้งานได้จริงแต่ยังไม่สมบูรณ์ เนื่องจากการรับส่งข้อมูลผ่านเครือข่ายใช้เวลานานเมื่อเทียบกับประสิทธิภาพสูงสุดของการรับส่งข้อมูลผ่านเครือข่าย ซึ่งในอนาคตปัญหานี้สามารถแก้ไขได้โดยการเปลี่ยนวิธีการรับส่งข้อมูลระหว่างฮาร์ดแวร์เชื่อมต่อเครือข่ายและฮาร์ดแวร์ประมวลผลให้เร็วขึ้น เช่น SPI (Serial Peripheral Interface) เพื่อให้เป็นต้นแบบเชิงอุตสาหกรรมที่แท้จริงได้

เอกสารและสิ่งอ้างอิง

คณะกรรมการและอนุกรรมการบัญญัติศัพท์คอมพิวเตอร์. 2540. **ศัพท์คอมพิวเตอร์ ฉบับราชบัณฑิตยสถาน.** พิมพ์ครั้งที่ 4. โรงพิมพ์ มหาจุฬาลงกรณราชวิทยาลัย, กรุงเทพฯ.

จตุชัย แพงจันทร์ และ อนุโชต วุฒิพรพงษ์. 2546. **เจาะระบบ Network ฉบับสมบูรณ์.** บริษัท ไอดีซี อินโฟ ดิสทริบิวเตอร์ เซ็นเตอร์ จำกัด, นนทบุรี.

ณัฐพงศ์ เศียรสวัสดิ์ ธวัชชัย บานใหม่ และ วุฒิพงศ์ อารีกุล. 2548. การพัฒนาต้นแบบฮาร์ดแวร์ตรวจสอบลายนิ้วมือบนตัวประมวลผลสัญญาณดิจิทัล, น. 1009-1012. ใน **การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 28.** มหาวิทยาลัยธรรมศาสตร์, กรุงเทพฯ.

Andy, T. and D. W. Dart, updated by S. Dirksen. 2001. **How to get started with the DSP/BIOS kernel.** Literature no. SPRA782.

BDTI, Inc. 2000. **Evaluating DSP Processor Performance.** Berkeley Design Technology, Inc.

_____. 2003. **A BDTI Analysis of the Texas Instruments TMS320C67x.** Berkeley Design Technology, Inc.

_____. 2004. **BDTImark2000™/BDTIsimmark2000™ Scores for Packaged Processors.** Berkeley Design Technology, Inc.

Blonstein, S. 2002. **The TMS320 DSP Algorithm Standard.** Literature no. SPRA581C.

Bubeck, U. and D. Sanchez. 2003. **Biometric Authentication - Technology and Evaluation -.** Term Project CS574. San Diego State University.

Chassaing, R. 2002. **DSP Applications Using C and TMS320C6x DSK.** John Wiley & Sons, Inc., New York.

Dart, D., updated by S. Dirksen. 2001. **Using the DSP/BIOS Kernel in Real-Time DSP Applications.** Literature no. SPRA781.

Jain, A.K., R. Bolle, and S. Pankanti. 1999. **Biometrics Personal Identification in Networked Society.** Kluwer Academic Publishers.

Jain, A.K., A. Ross, and S. Prabhakar. 2004. An introduction to biometric recognition. **IEEE Trans. on circuits and systems for video tech** 14 (1): 4-20.

Kehtarnavaz, N. and M. Keramat. 2001. **DSP system design using the TMS320C6000.** Prentice-Hall, Inc.

Kurose, J.F. and K.W. Ross. 2003. **Computer Networking: A Top-Down Approach Featuring the Internet.** Second Edition. Pearson Education, Inc., USA.

Liu, S and M. Silverman. 2001. A practical guide to biometric security technology. **IEEE IT Professional** 3(1): 27-32.

Podio, F.L., J.S. Dunn, L. Reinert, C.J. Tilton, L. O'Gorman, M.P. Collier, M. Jerde, and B. Wirtz. 2001. CBEFF Common Biometric Exchange File Format. **NISTIR 6529.** NIST.

Podio, F.L., J.S. Dunn, L. Reinert, C.J. Tilton, B. Struif, F. Herr, J. Russell, M.P. Collier, M. Jerde, L. O'Gorman, and B. Wirtz. 2004. CBEFF Common Biometric Exchange Formats Framework. **NISTIR 6529-A.** NIST.

Sathappan, R. 2003. **DSP for Smart Biometric Solutions.** Literature no. SPRA894.

Silberschatz, Galvin and Gagne. 2003. **OPERATING SYSTEM CONCEPTS.** Sixth edition. John Wiley & Sons, Inc.

Stapleton, J and KPMG LLP. 2003. **Biometrics.** PKI Note.

Stevens, W.R. 1994. **TCP/IP Illustrated, Volume 1: The Protocols.** Addison-Wesley Publishing Company, USA.

Tanenbaum, A.S. 1996. **Computer network.** Pearson Education, Inc.

Texas Instruments Inc. 2001. **TMS320C6000 Peripherals Reference Guide.** Literature no. SPRU190D.

_____. 2002 a. **TMS320 DSP Algorithm Standard API Reference.** Literature no. SPRU423B.

_____. 2002 b. **TMS320 DSP Algorithm Standard Rules and Guidelines.** Literature no. SPRU423B.

_____. 2002 c. **TMS320 DSP/BIOS User's Guide.** Literature no. SPRU423B.

Tilton, C.J. 2000. An emerging biometric API industry standard. **IEEE Computer** 3 (2): 130-132.

Tilton, C.J. 2003. **Biometric Industry Standards.** SAFLINK Corporation.

Torud, S. 2002. **A Technical Overview of eXpressDSP-Compliant Algorithms for DSP Software Producers.** Literature no. SPRA782.

Ubicom, Inc. 2001. **IP2022 Internet Processor™ Features and Performance Optimized for Network Connectivity.** Ubicom, Inc.

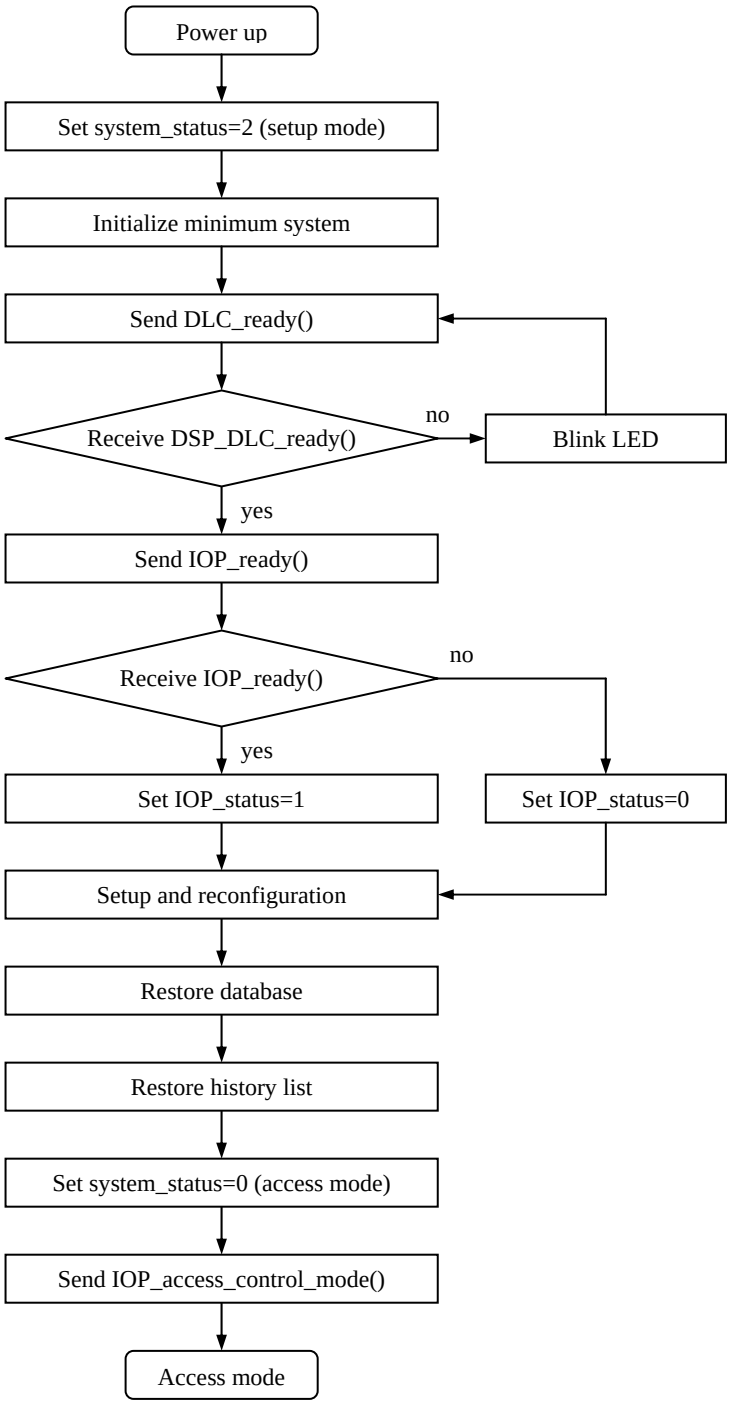
Wayman, J.L. 2001. Fundamentals of biometric authentication technologies. **Int. J. Image Graphics** vol. 1, no. 1, pp. 93–113.

Wilson, J.H. 2001. **BioAPI Specification Version 1.1.** The BioAPI Consortium.

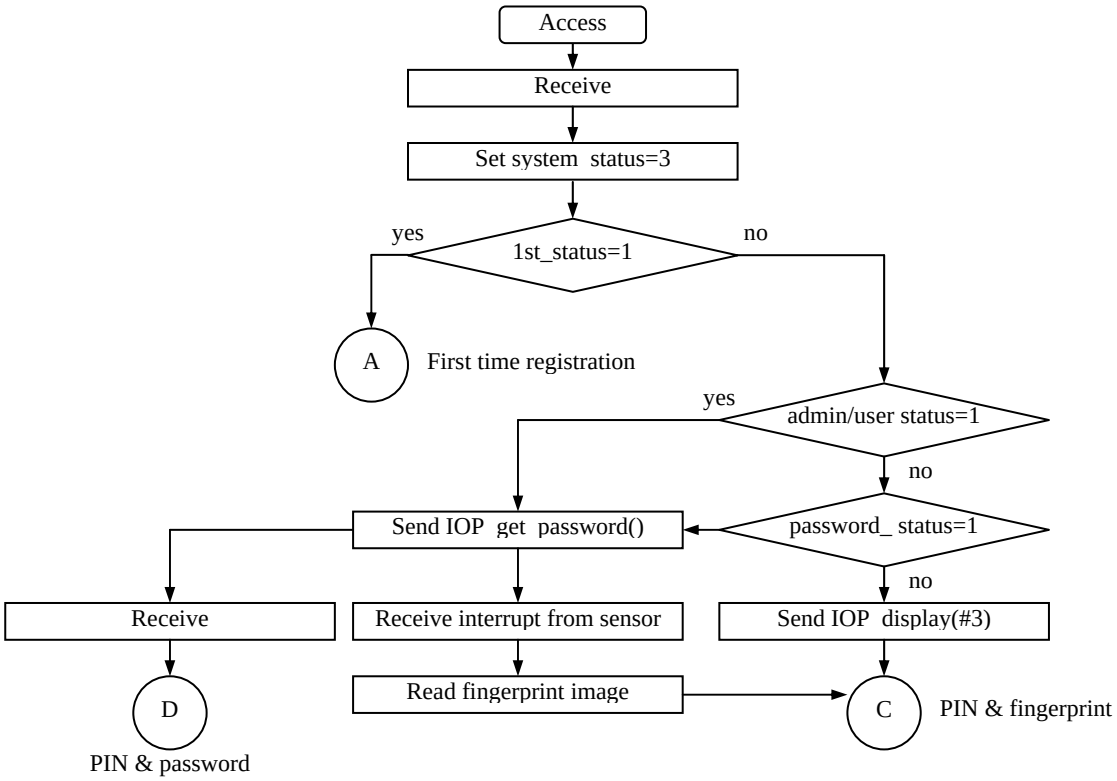
ภาคผนวก

ภาคผนวก ก

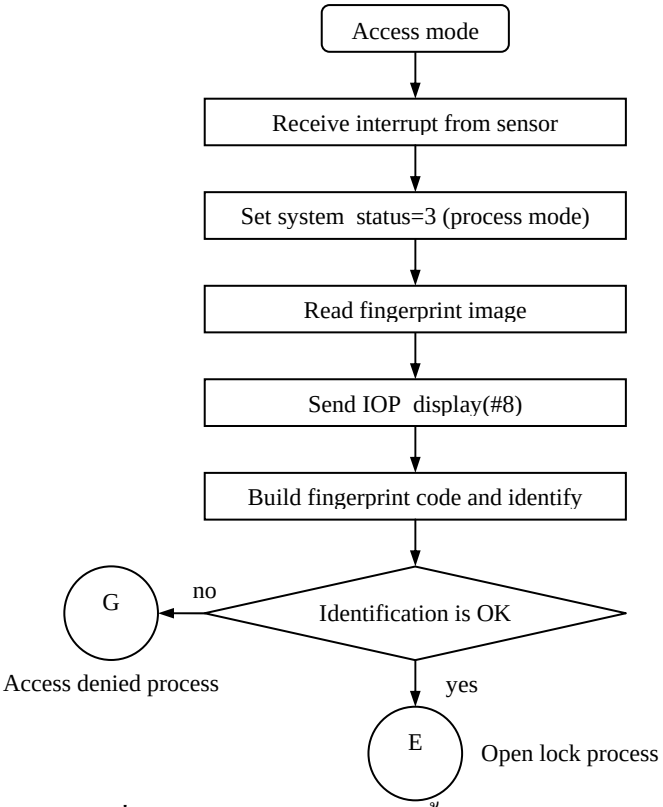
แผนผังซอฟต์แวร์ระบบปฏิบัติการไบโอเมตริก



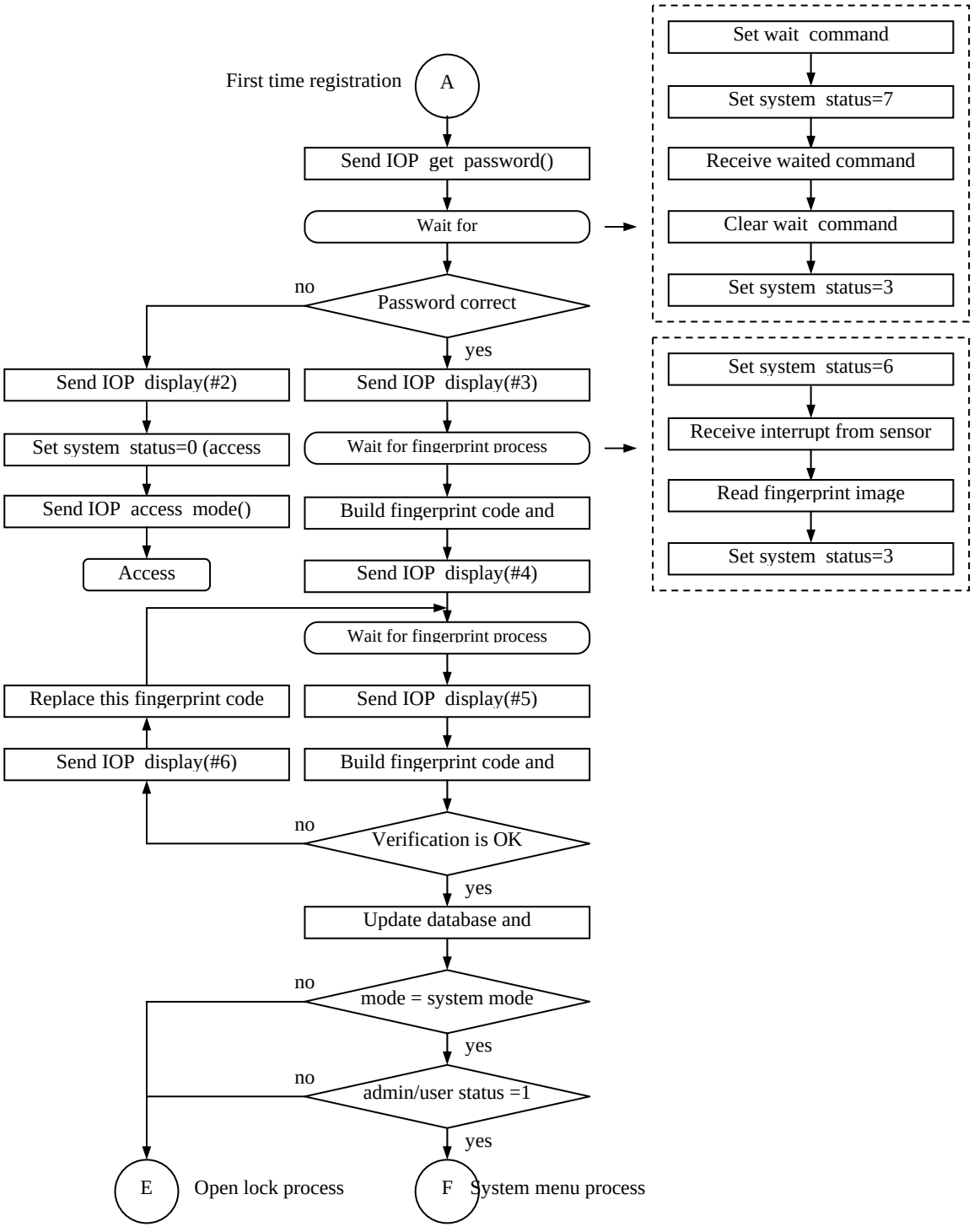
ภาพผนวกที่ 1 แผนผังซอฟต์แวร์ส่วนเริ่มต้นการทำงานของฮาร์ดแวรตรวจสอบลายนิ้วมือ



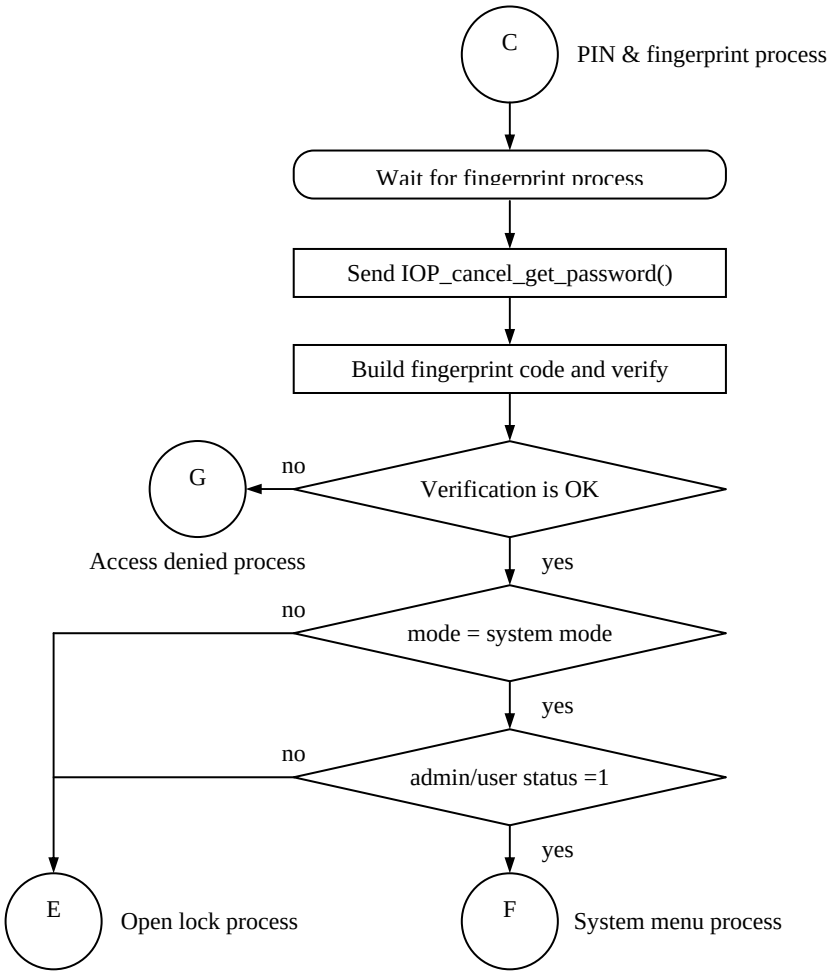
ภาพผนวกที่ 2 แผนผังซอฟต์แวร์ส่วนขั้นตอนการรับหมายเลข PIN



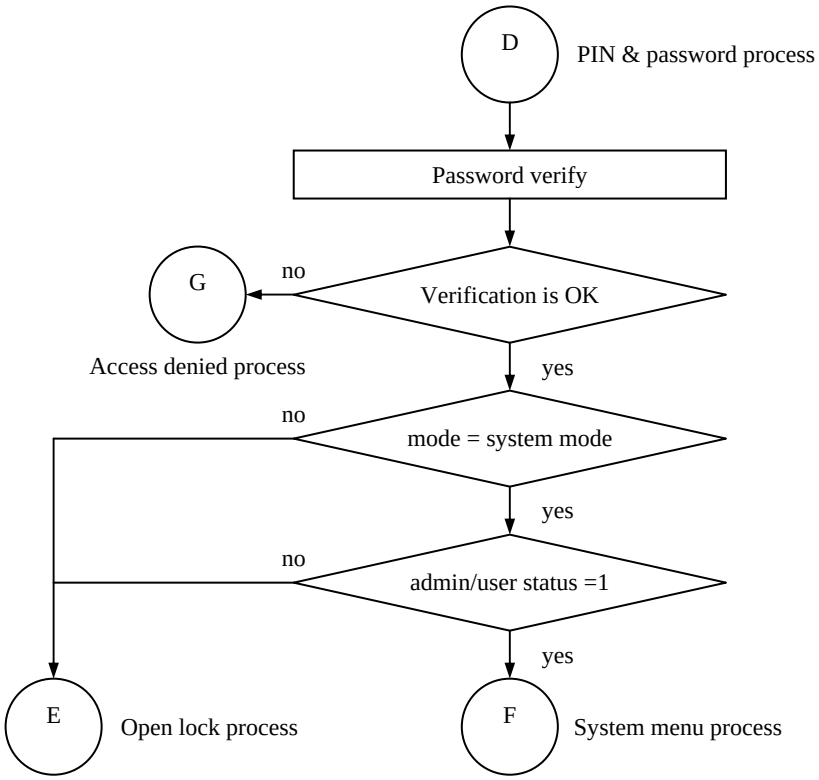
ภาพผนวกที่ 3 แผนผังซอฟต์แวร์ส่วนขั้นตอนการอ่านค่าลายนิ้วมือ



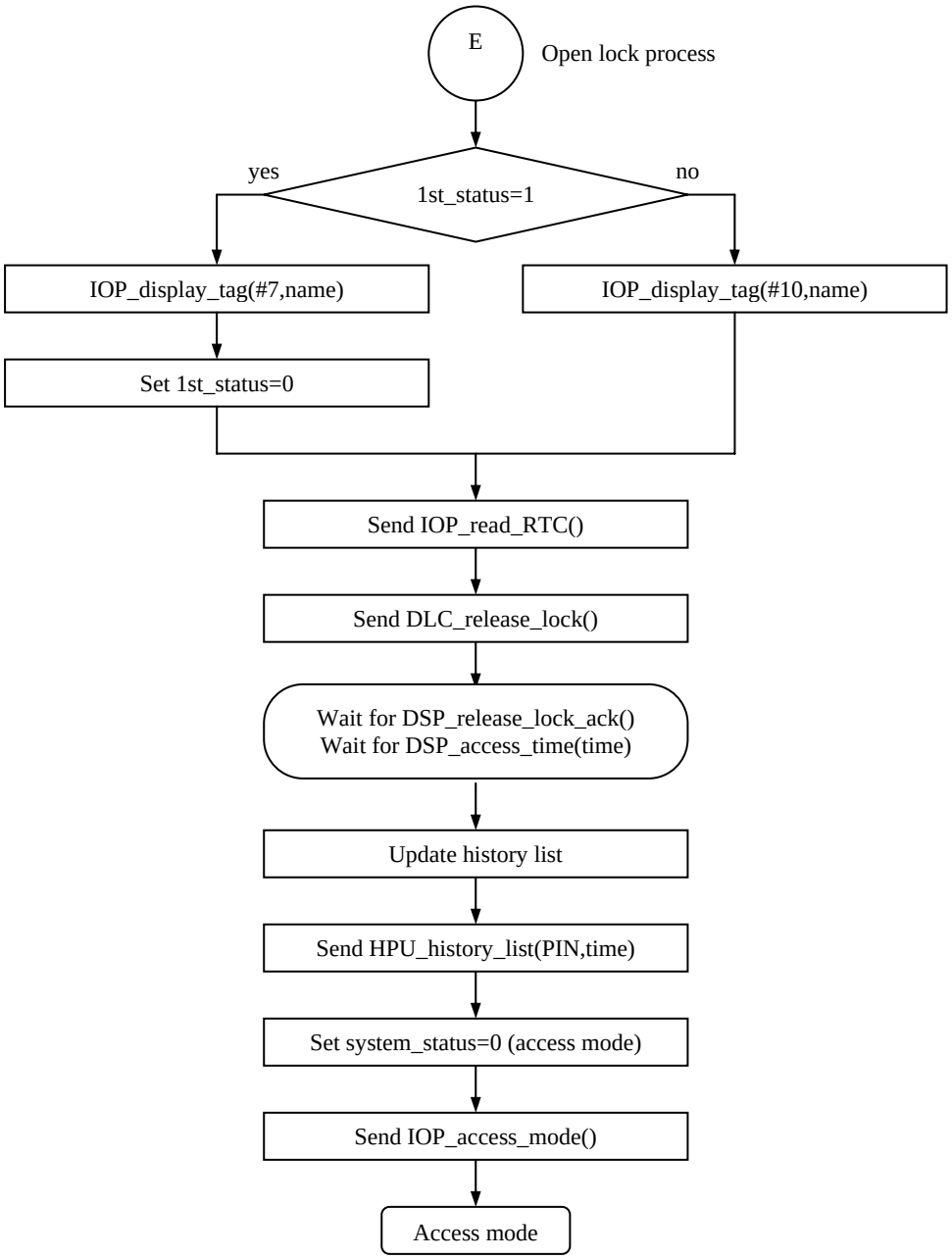
<u>ภาพผนวกที่ 4</u> แผนผังซอฟต์แวร์ส่วนขั้นตอนการลงทะเบียนครั้งแรก



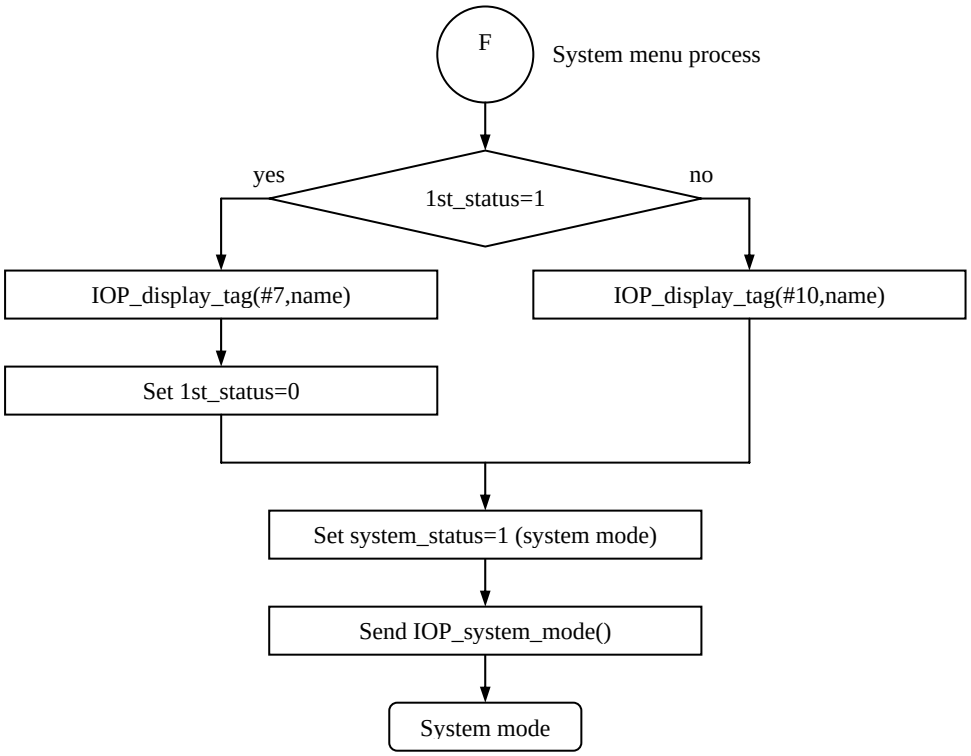
<u>ภาพผนวกที่ 5</u> แผนผังซอฟต์แวร์ส่วนขั้นตอนการตรวจสอบลายนิ้วมือ



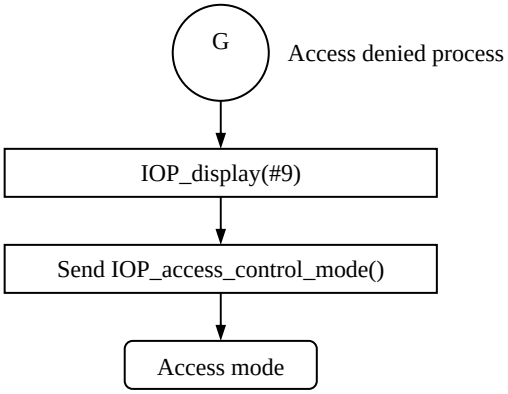
ภาพผนวกที่ 6 แผนผังซอฟต์แวร์ส่วนขั้นตอนการประมวลผลรหัสผ่าน



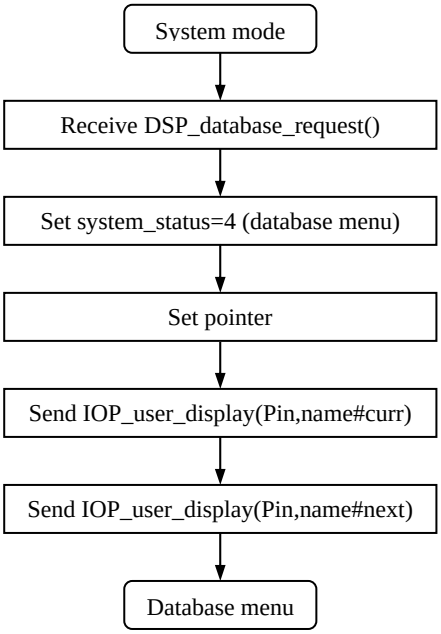
ภาพผนวกที่ 7 แผนผังซอฟต์แวร์ส่วนขั้นตอนการสั่งเปิดประตู



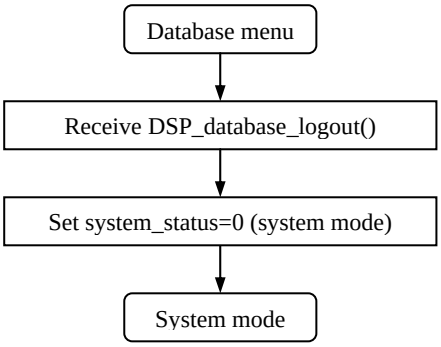
ภาพผนวกที่ 8 แผนผังซอฟต์แวร์ส่วนขั้นตอนการเข้าจัดการระบบ



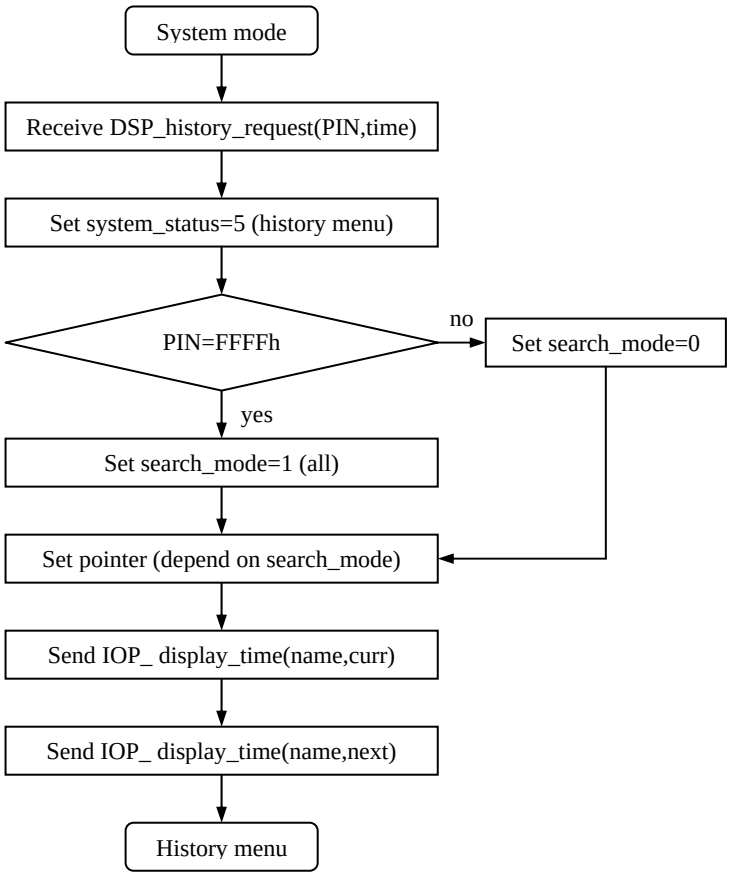
ภาพผนวกที่ 9 แผนผังซอฟต์แวร์ส่วนขั้นตอนการปฏิเสธเข้าใช้งาน



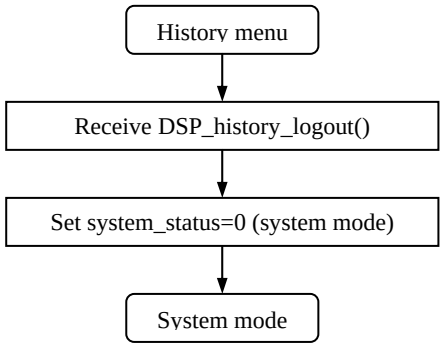
ภาพผนวกที่ 10 แผนผังซอฟต์แวร์ส่วนขั้นตอนการจัดการฐานข้อมูลบุคคล



ภาพผนวกที่ 11 แผนผังซอฟต์แวร์ส่วนขั้นตอนการออกจากฐานข้อมูลบุคคล



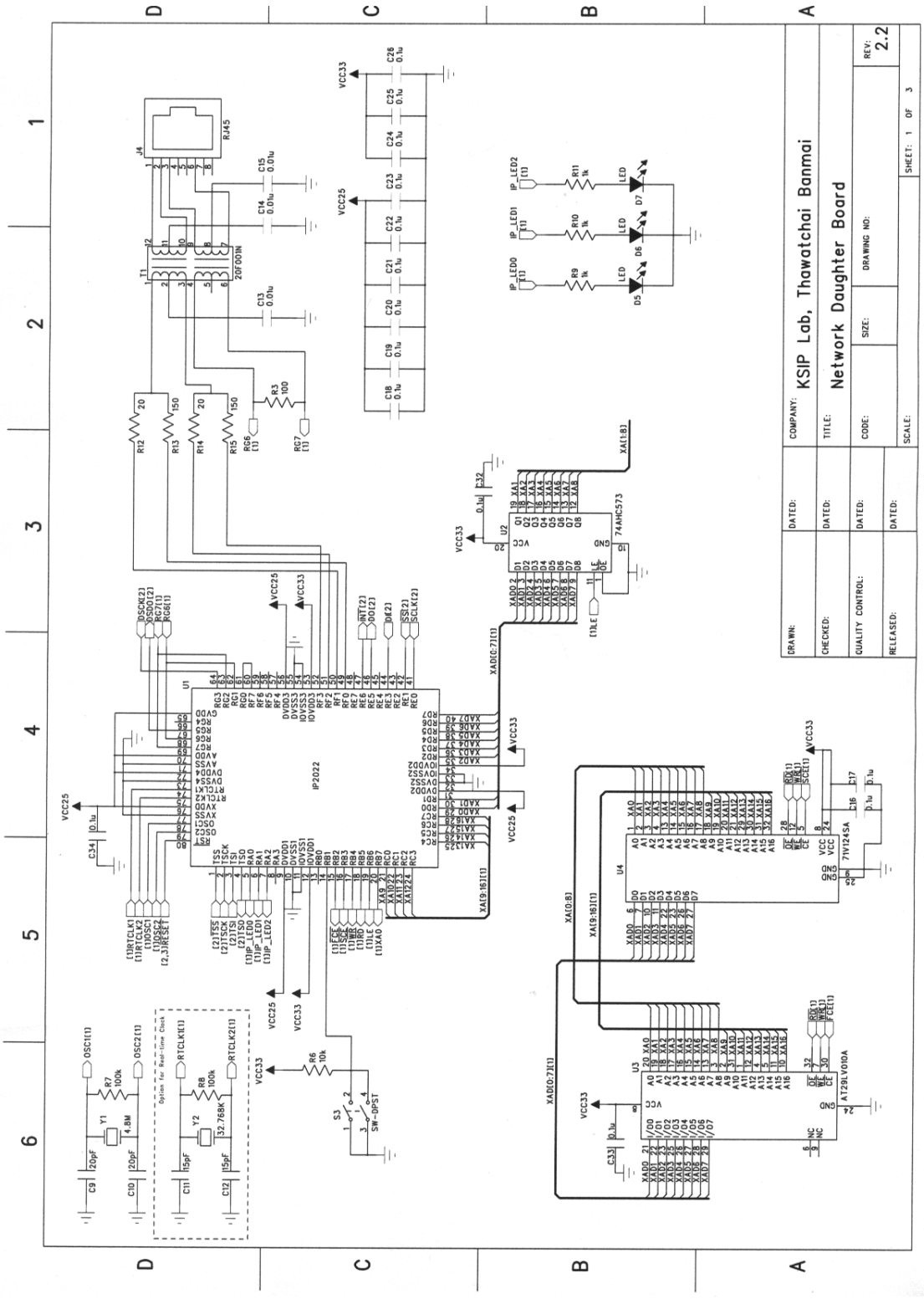
ภาพผนวกที่ 12 แผนผังซอฟต์แวร์ส่วนจัดการฐานข้อมูลประวัติ



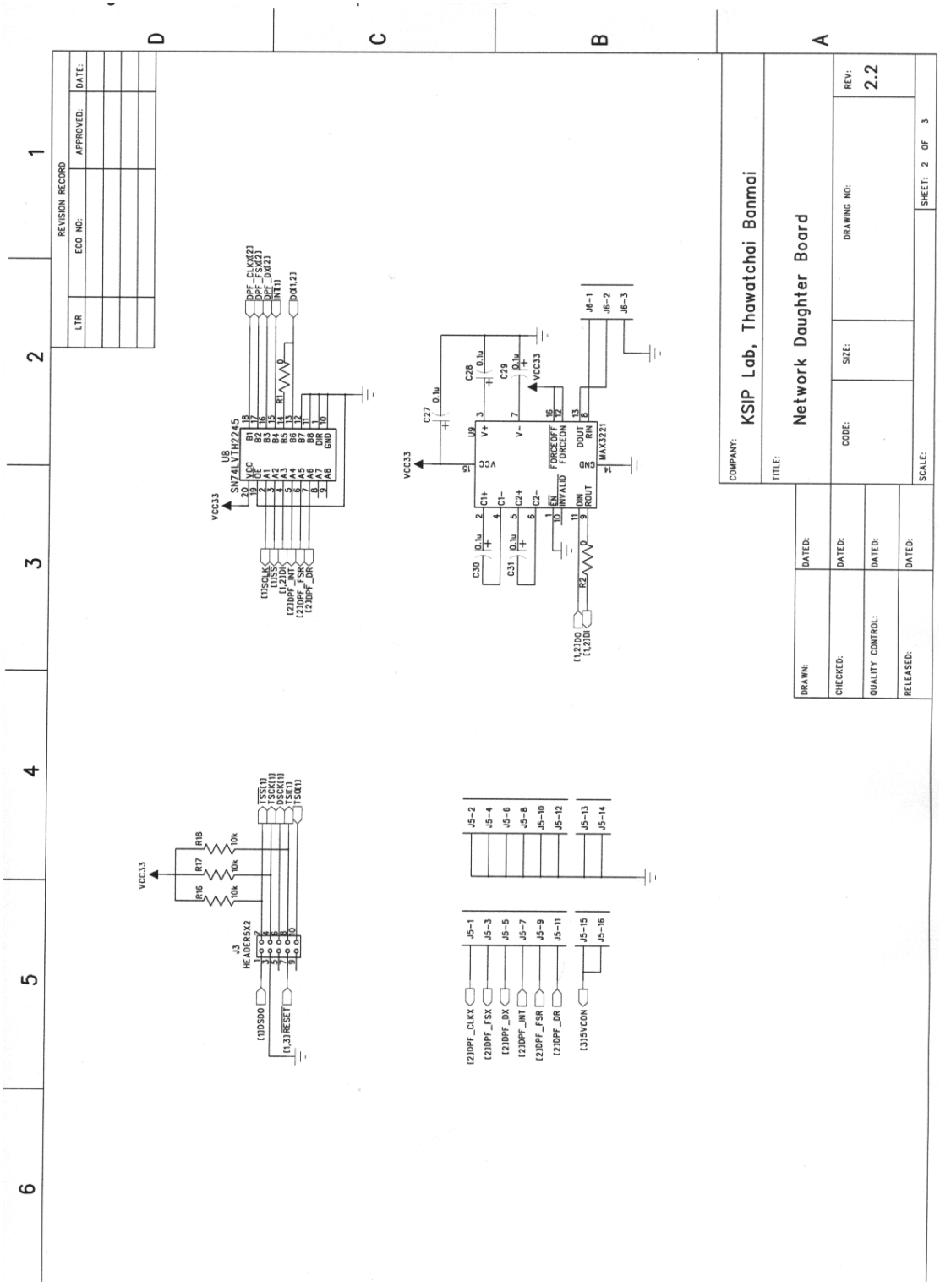
ภาพผนวกที่ 13 แผนผังซอฟต์แวร์ส่วนออกจากฐานข้อมูลประวัติ

ภาคผนวก ข

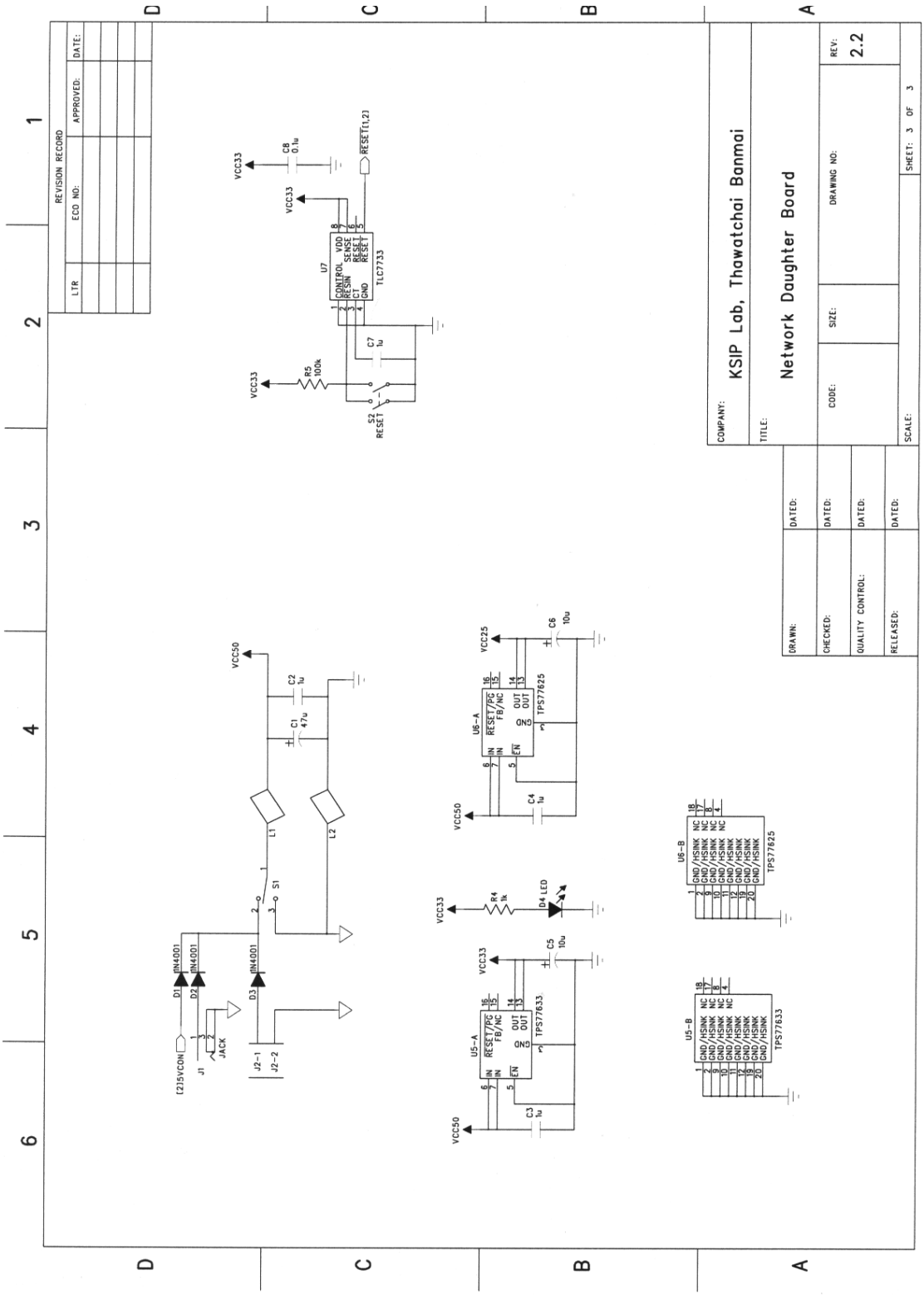
แผนภาพวงจรฮาร์ดแวร์เชื่อมต่อเครือข่าย



ภาพผนวกที่ 14 แผนผังวงจรฮาร์ดแวร์ส่วนตัวประมวลผลเครือข่าย



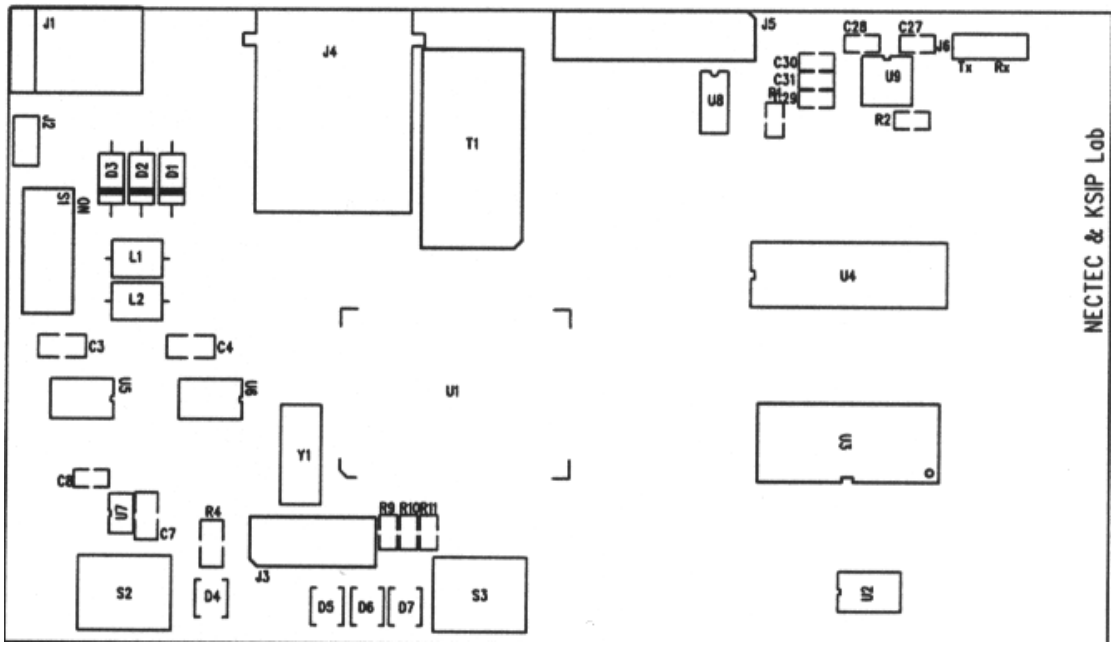
ภาพผนวกที่ 15 แผนผังวงจรฮาร์ดแวร์ส่วนเชื่อมต่อสัญญาณภายนอก



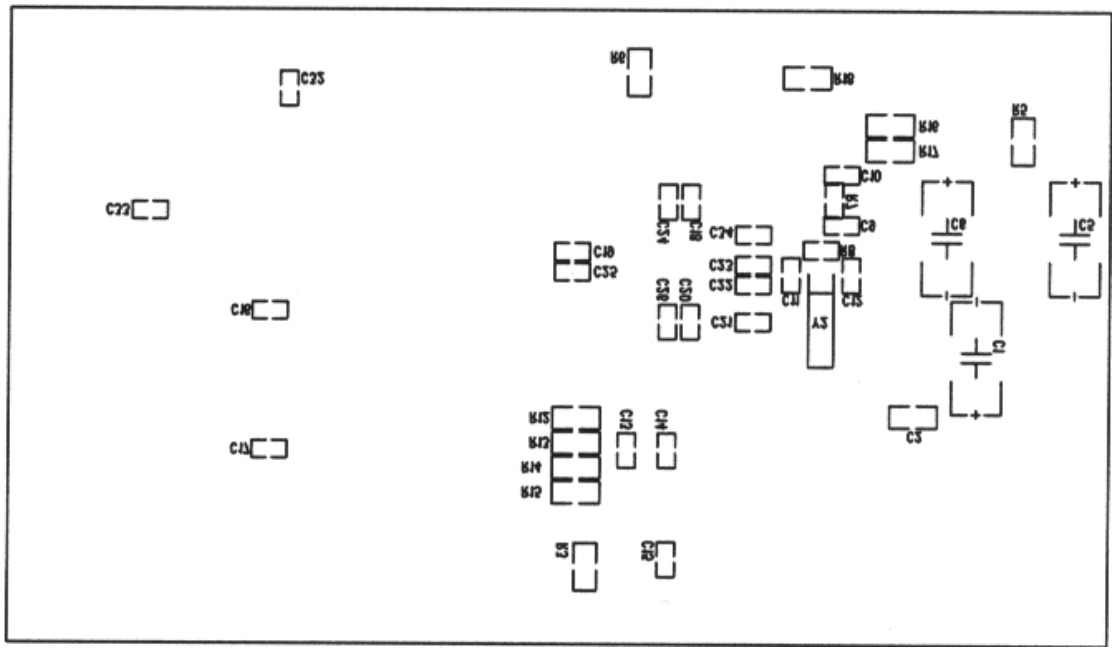
ภาพผนวกที่ 16 แผนผังวงจรฮาร์ดแวร์ส่วนวงจรจ่ายไฟ

ภาคผนวก ค

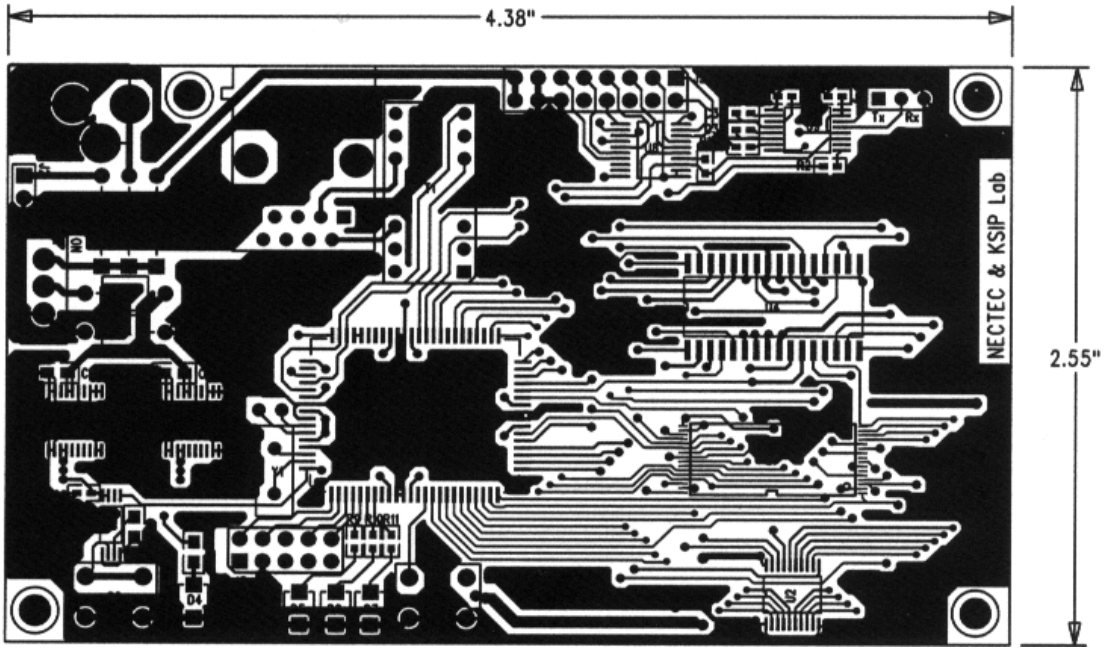
ลายวงจรฮาร์ดแวร์เชื่อมต่อเครือข่าย



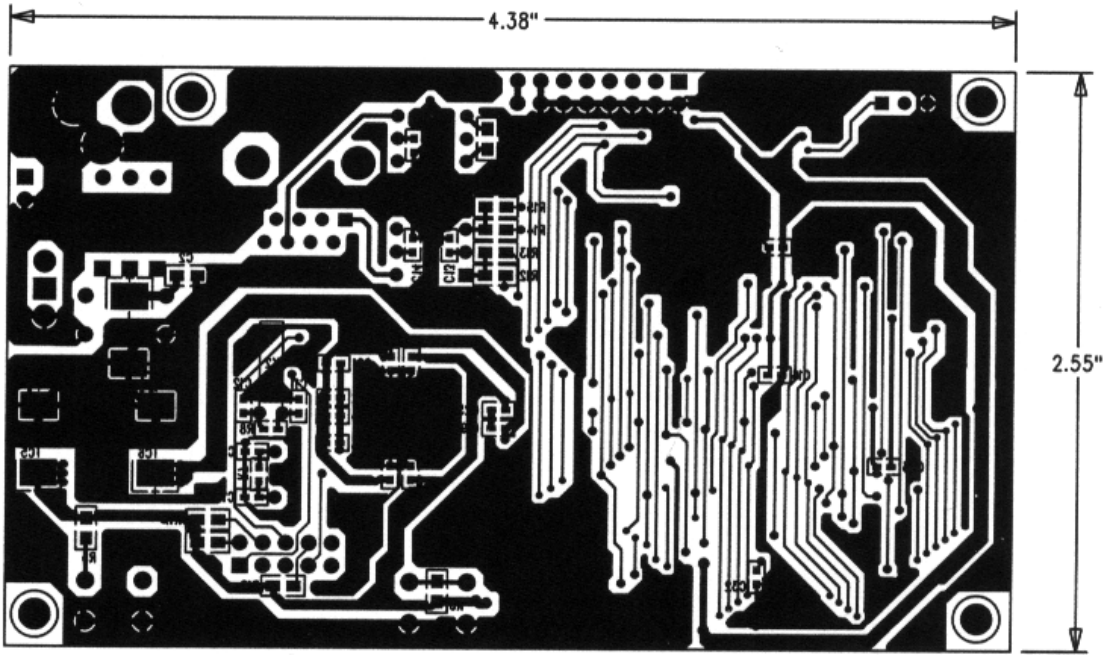
<u>ภาพผนวกที่ 17</u> ตำแหน่งอุปกรณ์บนลายวงจรฮาร์ดแวร์เชื่อมต่อเครือข่ายด้านบน



<u>ภาพผนวกที่ 18</u> ตำแหน่งอุปกรณ์บนลายวงจรฮาร์ดแวร์เชื่อมต่อเครือข่ายด้านล่าง



ภาพผนวกที่ 19 ลายวงจรฮาร์ดแวร์เชื่อมต่อเครือข่ายด้านบน



ภาพผนวกที่ 20 ลายวงจรฮาร์ดแวร์เชื่อมต่อเครือข่ายด้านล่าง