บอร์ดควบคุมสำหรับระบบฝังตัว

Control Board for Embedded System

นาย บุญเลิศ ใสส่อง
นาย ไพศาล แก้วจันทร์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2545

บอร์ดควบคุมสำหรับระบบฝังตัว

Control Board for Embedded System

โดย

นาย บุญเลิศ ใสส่อง

นาย ไพศาล แก้วจันทร์

อาจารย์ที่ปรึกษา

อ. ประสาร ตังติสานนท์

อ. อวัชริน นาชิน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

ปริญญานิพนธ์ ปีการศึกษา 2545

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง บอร์ดควบคุมสำหรับระบบฝังตัว

Control Board for Embedded System

ผู้จัดทำ

นาย บุญเลิศ ใสส่อง รหัสประจำตัว 43015366

นาย ไพศาล แก้วจันทร์ รหัสประจำตัว 43015374

_____ อาจารย์ที่ปรึกษา

(อ. ประสาร ดังติสานนท์)

_____ อาจารย์ที่ปรึกษา

(อ. อวัชริน นาชิน)

บอร์ดควบคุมสำหรับระบบฝังตัว

นาย บุญเลิศ ใสส่อง 43015366
นาย ไพศาล แก้วจันทร์ 43015374
อ. ประสาร ตังติสานนท์ อาจารย์ที่ปรึกษา
อ. อวัชริน นาชิน อาจารย์ที่ปรึกษา
ปีการศึกษา 2545

บทคัดย่อ

โครงงานนี้ เป็นการพัฒนาระบบฝังตัว เพื่อนำมารองรับการใช้งานผ่านระบบเครือข่ายคอมพิวเตอร์ โดยจะแบ่งออกเป็น 2 ส่วน ส่วนแรกเป็นการพัฒนาบอร์ดควบคุม โดยได้ทำการออกแบบบอร์ดควบคุม ซึ่งประกอบด้วย ไมโครโปรเซสเซอร์ ARM7TDMI เป็นตัวประมวลผล หน่วยความจำแรมขนาด 1 เม็กกะไบท์ หน่วยความจำแฟลซขนาด 2 เม็กกะไบท์ และอีเทอร์เนตคอนโทรลเลอร์ 10 เม็กกะบิตต่อวินาที นอกจากนี้ยังมีอุปกรณ์ที่จำเป็นสำหรับบอร์ดควบคุม ได้แก่ ไทม์เมอร์ วอทช์ด็อก พอร์ตอนุกรมและพอร์ตขนานสำหรับติดต่ออุปกรณ์ภายนอก ส่วนที่สอง เป็นการพัฒนาโปรแกรม และระบบปฏิบัติการเรียลไทม์ โดยจะทำการพัฒนาจากระบบ ปฏิบัติการ eCos มาใช้งานร่วมกับบอร์ดควบคุม

Control Board for Embedded System

Mr. Boonlerd Saisong

Mr. Paisarn Keawchan

Advisor :

Mr. Prasarn Tungtisanon

Mr. Awacharin Nachin

2002

Abstract

This thesis develop embedded system for working network. This thesis consist of two parts. First part is develop the control board by design control board that used ARM7TDMI microprocessor is central processing unit , RAM 1 MB , Flash 2 MB , Ethernet Controller 10 Mb/s and used Timer , watchdog , serial port and parallel port for interface control external equipments. Second part is develop software programming by develop eCos Real – time OS for used on control board.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้อย่างดี ด้วยคำแนะนำและคำปรึกษาเกี่ยวกับระบบฝังตัว จาก อ. ประสาร ตังติสานนท์ และ อ. อวัชริน นาชิน ซึ่งเป็นอาจารย์ผู้ควบคุมปริญญานิพนธ์ ผู้วิจัยรู้สึกซาบซึ้งในความอนุเคราะห์จากท่านและขอกราบขอบพระคุณเป็นอย่างสูง ขอขอบคุณเพื่อนๆ นักศึกษาทุกคนที่ช่วยเหลือให้คำแนะนำต่างๆ พร้อมทั้งช่วยตรวจเทียบ และแก้ไขทฤษฎีและอื่นๆ ที่ผิดพลาด จนสำเร็จสมบูรณ์ยิ่งขึ้นและยังให้กำลังใจต่อผู้วิจัยอย่างใกล้ชิดตลอดมา

สุดท้ายขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ ที่ได้ให้ทุนสนับสนุนการทำปริญญานิพนธ์ครั้งนี้ คุณค่าและประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

บุญเลิศ ใสส่อง
ไพศาล แก้วจันทร์

สารบัญ

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูปภาพ

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

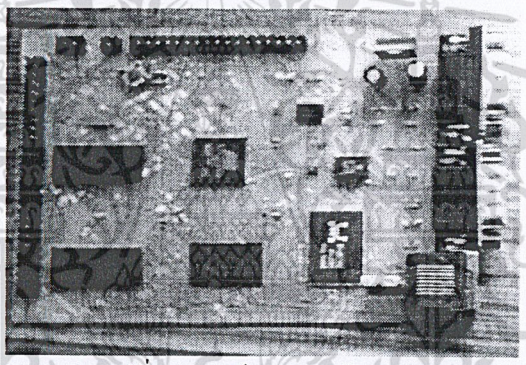
ปัจจุบันเทคโนโลยีของไมโครโปรเซสเซอร์ มีการพัฒนาอยู่อย่างต่อเนื่องเป็นแรงผลักดันให้เกิดประยุกต์ใช้งานใหม่ๆสำหรับระบบฝังตัว (Embedded System) จากเดิมที่มีใช้กันอยู่ในส่วนของงานภาคอุตสาหกรรม การควบคุมการผลิตหรืออุตสาหกรรมอื่นๆ ได้ขยายเข้ามาใช้งานด้านติดต่อสื่อสาร อุปกรณ์สำนักงาน อีกทั้งระบบรถยนต์ เครื่องเสียง วิดีโอ โทรทัศน์ โทรศัพท์มือถือ เครื่องเล่นเกม และเครื่องใช้ภายในบ้าน เช่น เครื่องล้างจาน เตาอบไมโครเวฟ เครื่องปรับอากาศ ระบบไฟฟ้าในบ้าน จะเห็นได้ว่าอุปกรณ์ไฟฟ้าที่อยู่รอบตัวเราควบคุมด้วยระบบฝังตัว

ในช่วงเวลา 2-3 ปีที่ผ่านมา แนวโน้มความต้องการที่จะให้ระบบฝังตัวสามารถทำงานร่วมกันทำงานได้หลายหน้าที่ ทำให้ระบบฝังตัวมีความซับซ้อนมากขึ้น ผลิตภัณฑ์ที่ทำขึ้นต้องมีการปรับปรุงเทคโนโลยีดิจิตอล ทั้งในส่วนของซอฟท์แวร์และฮาร์ดแวร์ ได้แก่ ระบบฝังตัวที่ต้องมีลักษณะเป็นเรียลไทม์(Real-time) ประสิทธิภาพของไมโครโปรเซสเซอร์ ความสามารถในการประมวลผล การสื่อสารข้อมูลในปริมาณมากขึ้นเพื่อรองรับความต้องการที่เพิ่มขึ้น การใช้งานซอฟท์แวร์ระบบฝังตัวที่เพิ่มขึ้นและแรงกดดันจากปัจจัยการตลาด ได้แก่ เส้นเขตการตลาด (Time-to-Market) ราคาต่อหน่วย และอื่นๆ เป็นต้น ทำให้การพัฒนาระบบฝังตัวที่แยกจากกันตามแต่การใช้งานนั้นอาจไม่คุ้มค่า การแทนที่ด้วยการพัฒนาสิ่งพื้นฐานที่ร่วมกันอาจจะทำให้ได้ผลที่ดีกว่า นั่นคือ ความต้องการแพลตฟอร์มที่ประกอบด้วยฮาร์ดแวร์และซอฟท์แวร์ที่ดีสำหรับใช้ในการพัฒนา โดยปกติแล้ว ในการที่จะผลิตสินค้าที่เป็นระบบฝังตัว มักจะ ต้องพัฒนาใหม่เกือบทั้งหมด ทั้งในส่วนของฮาร์ดแวร์และซอฟแวร์ จึงทำให้เวลาในการออกแบบและพัฒนาค่อนข้างนานและทำให้ ต้นทุนสูงขึ้น แต่ถ้าหากมีแพลตฟอร์ม ที่สามารถรองรับการพัฒนาแอพพลิเคชั่น ในด้านใดด้านหนึ่งหรือกลุ่ม ของแอพพลิเคชั่นที่มีความต้องการใกล้เคียงกัน โดยที่แพลตฟอร์มนั้นมี สภาพแวดล้อมที่เอื้อต่อการพัฒนาและสามารถให้ความต้องการที่แอพพลิเคชั่นนั้นๆ ต้องการได้ ทำให้ไม่ต้องพัฒนาใหม่ทั้งหมด สามารถพัฒนาแอพพลิเคชั่นต่อไปได้ทันที ส่งผลให้ระยะเวลาในการพัฒนาผลิตสั้นลง ลดต้นทุนได้มากและทำให้ช่วงเวลาในการนำผลิตภัณฑ์ลงสู่ตลาดสั้นลงในขณะที่ยังคงรักษาคุณภาพของผลิตภัณฑ์ไว้

ดังนั้นจากที่ได้กล่าวมาข้างต้น ถ้าหากสามารถที่จะออกแบบฮาร์ดแวร์ที่มีประสิทธิภาพรองรับกับความต้องการที่สามารถขยายได้ มีซอฟท์แวร์ที่สามารถทำงานได้หลายหน้าที่มีลักษณะเป็นเรียลไทม์ (real-time) ก็จะสามารถที่จะทำให้แข่งขันกันในด้านธุรกิจได้ จึงเป็นเหตุผลที่ทำให้เกิดงานวิจัยนี้ขึ้นมา

1.2 วัตถุประสงค์ของงานวิจัย

1. เพื่อศึกษาและพัฒนาแพลตฟอร์มสำหรับระบบฝังตัว ที่สามารถรองรับการทำงาน ผ่านเครือข่าย
2. คอมพิวเตอร์ ทำงานได้หลายงาน (Multi-tasking) และยืดหยุ่นสามารถรองรับการทำงานที่เปลี่ยนแปลงได้
3. เพื่อศึกษาและใช้งานไมโครโปรเซสเซอร์ อุปกรณ์เชื่อมต่อที่ใช้ในการออกแบบและพัฒนาให้
4. เหมาะสมกับระบบฝังตัว
5. เพื่อศึกษาและพัฒนาระบบปฏิบัติการเรียลไทม์(real-time Operating System) ที่เหมาะสมกับระบบฝังตัว สามารถรองรับความต้องการของผู้ใช้ ที่เปลี่ยนแปลงไป
6. เพื่อศึกษาการควบคุมอุปกรณ์ผ่านเครือข่ายคอมพิวเตอร์ การใช้งานโปรโตคอลทีซีพีไอพี(TCP/IP protocal)



รูปที่ 1.1 บอร์ดควบคุมต้นแบบ

1.3 ขอบเขตของงานวิจัย

1. ออกแบบและพัฒนาบอร์ดควบคุมเพื่อใช้เป็นแพลตฟอร์มในการพัฒนาระบบฝังตัว
2. พัฒนาระบบปฏิบัติการเรียลไทม์(Real-time Operating System)ที่ใช้ร่วมกับบอร์ดควบคุม
3. ประยุกต์ใช้งานแพลตฟอร์มระบบฝังตัวที่สร้างขึ้น เพื่อทดสอบการทำงาน โดยจะใช้งานเป็นการควบคุมอุปกรณ์ผ่านเครือข่ายคอมพิวเตอร์

1.4 แนวความคิด

เนื่องจากระบบฝังตัวมักจะมีขนาดเล็ก เป็นอุปกรณ์ที่เคลื่อนที่ได้ เช่น โทรศัพท์มือถือ พ็อกเก็จพีซี เป็นต้น ซึ่งส่วนของฮาร์ดแวร์มักจะถูกติดตั้งเข้ากับแผ่นวงจร และซอฟท์แวร์ที่จำเป็นต้องมีการปรับปรุงได้ ปัญหาที่เกิดขึ้นคือเมื่อทำการเปลี่ยนแปลงซอร์ฟแวร์ นั้นจะทำอย่างไรที่สามารถเปลี่ยนแปลงข้อมูลที่อยู่ในหน่วยความจำรอมในระบบฝังตัวได้ วิธีการก็คือการทำ On-Board Programming ดังนั้นการเลือกอุปกรณ์ที่

นำมาประกอบเป็นบอร์ดควบคุมจำเป็นต้องมีอินเตอร์เฟสที่สามารถติดต่ออุปกรณ์ในขณะที่อุปกรณ์ยังอยู่ในวงจรได้ กล่าวคือมี พอร์ต JTAG (Joint Test Action Group) อีกทั้งยังมีประโยชน์ในการตรวจสอบข้อผิดพลาดของระบบด้วย

1.5 วิธีการดำเนินการ

โครงงานนี้จะแบ่งการทำงานออกเป็นลำดับขั้นตอนดังนี้

1. ออกแบบบอร์ดควบคุมเพื่อที่จะนำมาสร้างและพัฒนาใช้งานเป็นแพลตฟอร์มที่เหมาะสมกับระบบฝังตัว โดยพิจารณาจากประสิทธิภาพ และ การทำงานโดยรวมของบอร์ดควบคุม
2. สร้างและพัฒนาบอร์ดควบคุมพร้อมทั้งปรับปรุงแก้ไขส่วนต่างๆ ให้สามารถใช้งานได้อย่างน่าเชื่อถือและประสิทธิภาพมากที่สุด
3. ศึกษาค้นคว้าเกี่ยวกับระบบปฏิบัติการที่จะนำมาใช้เป็นระบบฝังตัว
4. พัฒนาระบบฝังตัวที่เป็นเรียลไทม์(real-time OS) เพื่อที่จะนำมาใช้งานร่วมกับบอร์ดควบคุมที่ได้สร้างขึ้น โดยคำนึงถึงความเป็นไปได้ของการพัฒนาและสามารถนำไปใช้ให้เกิดประโยชน์สูงสุด
5. พัฒนาแอพพลิเคชั่นเพื่อมาทดสอบการทำงานของบอร์ด
6. สรุปผลการพัฒนาบอร์ดและการนำบอร์ดที่สร้างขึ้นไปใช้งาน ว่ามีประสิทธิภาพมากน้อยเพียงใด
7. รวมถึงข้อดีและข้อเสีย ในการนำบอร์ดไปใช้งาน

บทที่ 2
ทฤษฎีและหลักการ

2.1 ระบบฝังตัว (Embedded System)

ระบบฝังตัว หรือ Embedded system คือ ระบบปฏิบัติการคอมพิวเตอร์ที่มีขนาดเล็กซึ่งอยู่ในชิปและฝังอยู่กับอุปกรณ์ต่างๆ คอยทำหน้าที่ควบคุมการทำงานของระบบ โดยมีไมโครคอนโทรลเลอร์หรือไมโครโพรเซสเซอร์เป็นหัวใจหลักสำคัญคอยควบคุมการทำงานของระบบ ให้สามารถทำงานได้ตามความต้องการ เรามักจะพบเห็นและใช้งานอุปกรณ์ที่เป็นระบบฝังตัวมากมายในชีวิตประจำวันทั่วไปโดยไม่รู้ตัว ตัวอย่างเช่น เครื่องปรับอากาศ เครื่องซักผ้า เตาอบไมโครเวฟ เป็นต้น

2.2 ลักษณะของระบบฝังตัว

- ขนาดเล็ก มีน้ำหนักเบา เพราะต้องไปฝังติดอยู่กับอุปกรณ์ต่างๆ
- ใช้พลังงานต่ำ ทำให้เกิดความร้อนน้อย
- ระบบต้องมีความเชื่อถือได้
- งานที่นำไปใช้ ถ้ามีความสำคัญมาก ระบบจะต้องมีความแน่นอนสูงสุด เช่นระบบควบคุมการหยุดของรถในขณะเลี้ยว เป็นต้น
- ทนต่อสภาพแวดล้อมต่างๆได้ ความร้อน สั่นสะเทือน คลื่นแม่เหล็กไฟฟ้า
- ระบบจะต้องตอบสนอง ทำงานได้ในระยะเวลาที่คาดไว้ได้

2.3 การแบ่งระดับของระบบฝังตัว

ซึ่งสามารถจัดแบ่งระดับของระบบฝังตัว ตามระดับของประสิทธิภาพการทำงานของไมโครโปรเซสเซอร์ ในระบบฝังตัวได้ออกเป็น สามระดับดังนี้คือ

2.3.1 ระบบฝังตัวขนาดเล็ก (Low-end embedded system)

เป็นระบบฝังตัวที่มีขนาดเล็ก ใช้ไมโครโพรเซสเซอร์หรือไมโครคอนโทรลเลอร์ขนาด4 หรือ 8 บิต เป็นตัวควบคุมการทำงาน มักใช้งานในงานที่มีฟังก์ชันการทำงานง่ายไม่ซับซ้อนตัวอย่างของไมโครคอนโทรลเลอร์ เหล่านี้ได้แก่ MCS-51,PIC และ Z80 เป็นต้น

2.3.2 ระบบฝังตัวขนาดกลาง (Mid-end embedded system)

เป็นระบบฝังตัวที่มีขนาดใหญ่ขึ้นมาอีกระดับมีความสามารถในการประมวลผลสูงขึ้นมา ส่วนใหญ่จะถูกนำไปใช้ในงานที่ต้องการคุณสมบัติพิเศษ ที่ไมโครคอนโทรลเลอร์หรือไมโครโพรเซสเซอร์ขนาดเล็กไม่

สามารถให้ได้ เช่น ต้องการให้งานจำนวนหน่วยความจำขนาดมากขึ้น เป็นต้น ไมโครโพรเซสเซอร์ หรือ ไมโครคอนโทรลเลอร์เหล่านี้มักมีขนาด 16 และ 32 บิต เช่น ไมโครโพรเซสเซอร์ ตระกูล x86 ของบริษัท Intel และ AMD ตระกูลARM7 และ ตระกูล TMS320 เป็นต้น

2.3.3 ระบบฝังตัวขนาดใหญ่ (High-end embedded System)

เป็นระบบฝังตัวขนาดใหญ่ที่มีความสามารถในการประมวลผลมากเป็นพิเศษ ส่วนใหญจะใช้งาน ไมโครโพรเซสเซอร์ที่ได้รับการออกแบบสำหรับการประมวลผลเป็นหลัก มักจะอยู่ในรูปของระบบฝังตัวที่ใช้เครื่องคอมพิวเตอร์เป็นพื้นฐาน (PC-Base) ใช้ฮาร์ดแวร์และซอฟต์แวร์สำหรับเครื่องคอมพิวเตอร์ทั่วไปเช่นระบบปฏิบัติการวินโดวส์ ดอส หรือ ลินุกซ์เป็นต้น ในการประมวลผลงานทางด้านใดด้านหนึ่งโดยเฉพาะอยู่ตลอดเวลา จึงสามารถเรียกระบบฝังตัวประเภทนี้ได้อีกแบบว่า Application Specific PC ไมโครโพรเซสเซอร์ที่ใช้งานก็จะ เป็น 32 หรือ 64 บิตของคอมพิวเตอร์ในปัจจุบัน เช่น เพนเทียม, เพนเทียมทู เป็นต้น

2.4 องค์ประกอบของระบบฝังตัว

2.4.1 ไมโครโพรเซสเซอร์

ไมโครโพรเซสเซอร์นับเป็นหัวใจหลักที่สำคัญของระบบฝังตัว เพราะต้องทำหน้าที่ประมวลผลข้อมูล และ ควบคุมการทำงานของระบบทั้งหมด โดยไมโครโพรเซสเซอร์ของระบบฝังตัวนี้มีมากมายหลากหลายประสิทธิภาพการทำงานสำหรับการนำไปใช้งานในด้านต่างๆ ตามความต้องการความสามารถในการประมวลผลข้อมูลในการทำงาน นับตั้งแต่ไมโครโพรเซสเซอร์ขนาด 4 บิตที่ใช้ในระบบฝังตัวทั่วไปในยุค ต้นๆ เช่น ในวงจรรีโมตคอนโทรลเป็นต้นเรื่อยมาจนกระทั่งไมโครโพรเซสเซอร์ 32 บิตหรือ 64 บิตในยุคปัจจุบันที่ต้องการความสามารถในการประมวลผลที่สูงเป็นพิเศษเช่นในกรณีของโทรศัพท์มือถือ เป็นต้น

2.4.2 หน่วยความจำ

หน่วยความจำเป็นส่วนที่สำคัญส่วนหนึ่งในระบบฝังตัว เพราะจะเป็นส่วนที่กำหนดถึงความซับซ้อน ของโปรแกรมที่ใช้ในการทำงานในระบบฝังตัว และเป็นตัวกำหนดถึงวิธีการออกแบบและพัฒนาโปรแกรมสำหรับควบคุมการทำงานของระบบขึ้นมา ซึ่งหน่วยความจำที่ใช้ในระบบฝังตัวนี้มีหลายประเภทเช่น DRAM, SRAM EPROM,EEPROM หรือหน่วยความจำแฟลช (Flash memory) เป็นต้นซึ่งมีขนาดและรูปแบบแตกต่างกัน ตั้งแต่ขนาดเล็กมีอยู่ภายในตัวไมโครคอนโทรลเลอร์จนถึงแบบที่เป็นแผงหน่วยความจำแบบเดียวกับที่ใช้ในระบบคอมพิวเตอร์แบบตั้งโต๊ะในปัจจุบัน ซึ่งเลือกใช้ได้ตามความต้องการในการทำงานของแต่ละงาน

2.4.3 ส่วนอุปกรณ์เชื่อมต่อ (Peripherals)

ส่วนอุปกรณ์เชื่อมต่อเป็นส่วนที่ใช้ในการติดต่อสื่อสารระหว่างระบบฝังตัวกับโลกภายนอก ซึ่งเห็นได้ในรูปแบบต่างกันไปตามลักษณะของงานต่างๆ ในส่วนของอินพุทของระบบ ส่วนนี้มักเป็นอุปกรณ์ตรวจวัด (sensor) ชนิดต่างๆ เช่น อุปกรณ์ตรวจจับความร้อน อุปกรณ์ตรวจจับความชื้น หรือเป็นแป้นควบคุมการทำงานต่างๆ เช่น ปุ่มควบคุมตั้งเวลาและอุณหภูมิของเตาอบไมโครเวฟ หรือปุ่มกดหมายเลขและฟังก์ชันการทำ

งานต่างๆของโทรศัพท์ มือถือเป็นต้น นอกจากนี้อาจจะอยู่ในรูปของโมดูลการสื่อสารต่างๆกับระบบอื่นๆภายนอก เช่น โมดูลการเชื่อมต่อแบบ USB ,PCMCIA RS-232 หรือ RS-485 เป็นต้น ในส่วนของเอาท์พุทของระบบจะเป็นอุปกรณ์ที่ทำหน้าที่เป็นส่วนติดต่อสื่อสารกับผู้ใช้งานหรือระบบอื่นๆภายนอก เช่น หน้าจอแอลซีดีสำหรับแสดงผลของเตาอบไมโครเวฟหรือโทรศัพท์มือถือ เป็นต้น

2.4.4 ซอฟต์แวร์

ซอฟต์แวร์เป็นองค์ประกอบส่วนหนึ่งที่สำคัญในระบบฝังตัว ซึ่งเป็นตัวควบคุมกำหนดหน้าที่ฟังก์ชันการทำงานของระบบ ซึ่งภายในอาจประกอบด้วยส่วนที่เป็นตัวเริ่มต้นการทำงานและปรับแต่งระบบ (Initialization and configuration) ส่วนของระบบปฏิบัติการหรือส่วนจัดการสภาพแวดล้อมขณะทำงาน (Run-time environment) ส่วนของแอพลิเคชัน และ ส่วนจัดการความผิดพลาด ซึ่งในแต่ละระบบก็ซอฟต์แวร์ภายในก็จะประกอบด้วยส่วนต่างๆแตกต่างกันไปตามความจำเป็นของแต่ละชนิด

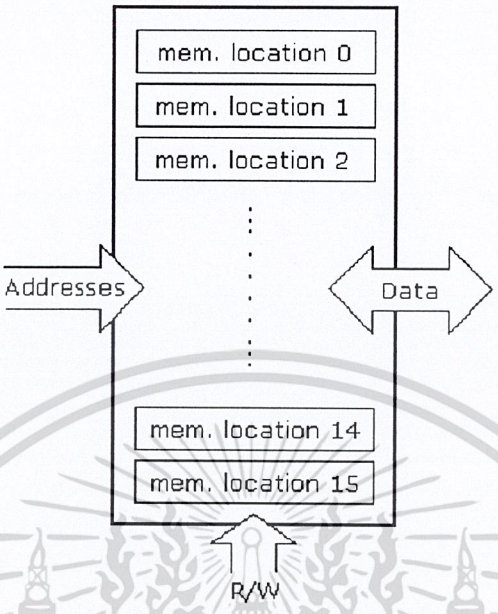
2.5 แอพพลิเคชันของระบบฝังตัว

ดังที่ได้กล่าวมาแล้วว่าระบบฝังตัว คือระบบที่มีคอมพิวเตอร์ขนาดเล็กอยู่ภายใน ทำงานเป็นส่วนหนึ่งของระบบ ทั้งนี้เนื่องจากความสามารถในการทำงานที่ซับซ้อน ได้อย่างถูกต้องและแม่นยำของคอมพิวเตอร์ระบบฝังตัวจึงถูกนำไปประยุกต์ใช้ในงานต่างๆ เช่น ใช้ในอุปกรณ์เครื่องใช้ไฟฟ้าเพื่อให้ใช้งานได้ง่ายขึ้น ใช้ในระบบควบคุมเบรคและถุงลมนิรภัยของรถยนต์ซึ่งเป็นงานที่ต้องการความแม่นยำสูง เป็นต้น นอกจากนี้คอมพิวเตอร์ยังตอบสนองความต้องการด้านการสื่อสารได้เป็นอย่างดีเนื่องจากมีโครงสร้างสถาปัตยกรรมเครือข่ายคอมพิวเตอร์ที่เป็นมาตรฐานและมีการใช้งานกันอย่างกว้างขวางอยู่แล้ว ทำให้ระบบฝังตัวสามารถนำไปใช้งานที่มีความต้องการด้านการติดต่อสื่อสาร เช่น ระบบควบคุมอุปกรณ์ระยะไกล ได้ เราอาจมองงานของระบบฝังตัวได้เป็นกลุ่มดังนี้อุปกรณ์เครื่องใช้ต่างๆ ระบบควบคุมโรงงาน อาคาร ซึ่งเป็นระบบใหญ่ที่มีการทำงานร่วมกันของอุปกรณ์จำนวนมาก และชุดพัฒนาต่างๆที่ใช้ในด้านการศึกษา วิจัย และพัฒนา

2.6 องค์ประกอบพื้นฐานของบอร์ดควบคุม

2.6.1 หน่วยความจำ (Memory unit)

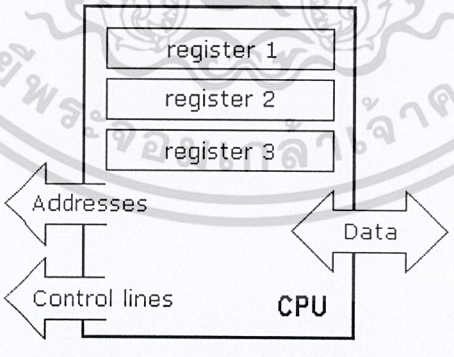
หน่วยความจำเป็นส่วนที่ไมโครโปรเซสเซอร์ใช้ ทำหน้าที่เก็บข้อมูล จากรูปข้างล่างเป็นตัวอย่างที่เป็นรูปแบบง่ายๆ ของหน่วยความจำ โดยใช้สัญญาณควบคุม R/W เป็นตัวกำหนดการติดต่อเมื่อเราต้องการอ่านหรือ เขียน



รูปที่ 2.1 บล็อกไดอะแกรมของหน่วยความจำ

2.6.2 หน่วยประมวลผลกลาง (Central Processing Unit)

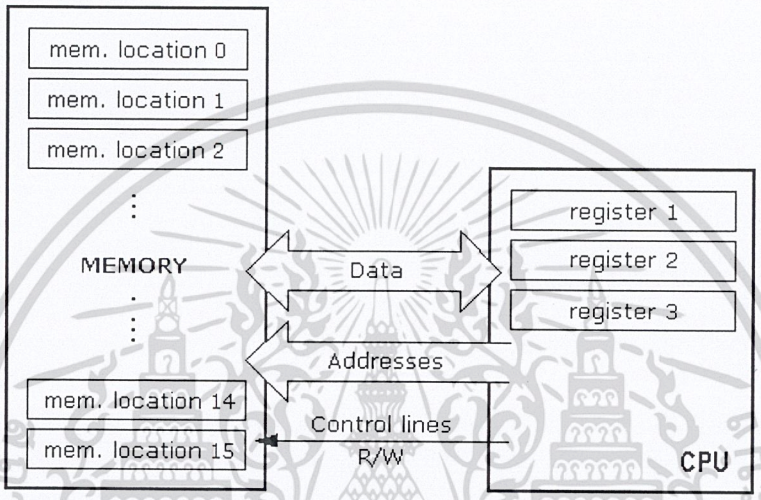
หน่วยประมวลผลกลางหรือที่เรียกว่า ซีพียู (CPU) ซึ่งมีความสามารถในกระบวนการทำงานทางคณิตศาสตร์และลอจิก จากรูป ซีพียูจะประกอบด้วยรีจิสเตอร์สามตัว โดยรีจิสเตอร์เหล่านี้เปรียบเสมือนหน่วยความจำมีหน้าที่หลายอย่างในการการทำงานทางด้านคณิตศาสตร์ หรือการทำงานอื่นๆที่เกี่ยวกับข้องกับข้อมูล (data)



รูปที่ 2.2 บล็อกไดอะแกรมของหน่วยประมวลผลกลาง

2.6.3 บัส (Bus)

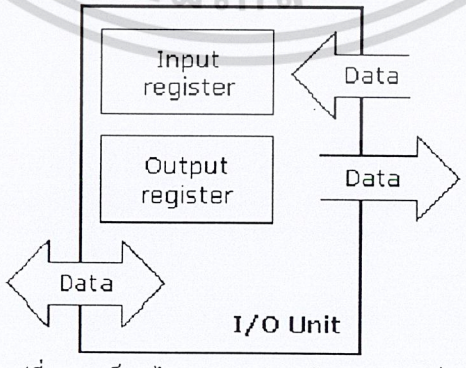
เป็นกลุ่มทางเดินข้อมูลที่มีขนาด 8,16 หรือมากกว่านั้น โดยบัสแบ่งออกเป็นสองชนิด คือแอดเดรสบัสและดาต้าบัสจากรูปเป็นการแสดงการเชื่อมต่อบัส โดยส่วนแรกเป็นการส่งแอดเดรสจากซีพียูไปยัง หน่วยความจำ และอีกส่วนเป็นการต่อดาต้าบัส



รูปที่ 2.3 บล็อกไดอะแกรมของบัส

2.6.4 อินพุท – เอาท์พุท (Input-output unit)

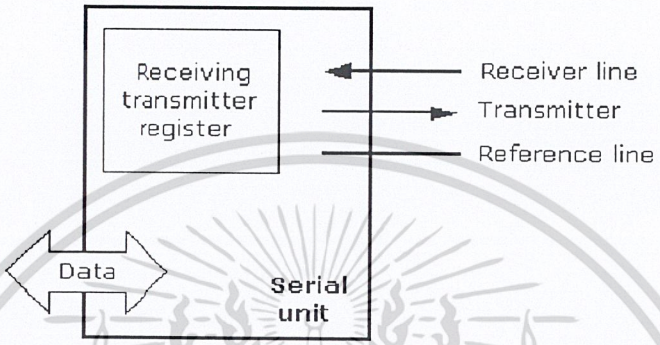
เป็นส่วนที่เราเรียกว่า " พอร์ต " ใช้ในการสื่อสารกับอุปกรณ์ภายนอก พอร์ตมีหลายแบบ ทั้งที่เป็นอินพุต , เอาท์พุต หรือเป็นได้ทั้ง อินพุตและ เอาท์พุท เมื่อมีการทำงานกับพอร์ตสิ่งแรกคือ จะมีการกำหนดพอร์ตที่ต้องการทำงานและ จากนั้นจากนั้นจะมีการส่งข้อมูลหรือ รับข้อมูลเกิดขึ้นผ่านพอร์ตดังกล่าว จากรูปเป็นตัวอย่าง หน่วยอินพุท-เอาท์พุท ที่ใช้ในการติดต่อกับส่วนภายนอก



รูปที่ 2.4 บล็อกไดอะแกรมของอินพุต – เอาท์พุท

2.6.5 การติดต่อสื่อสารแบบอนุกรม (Serial communication)

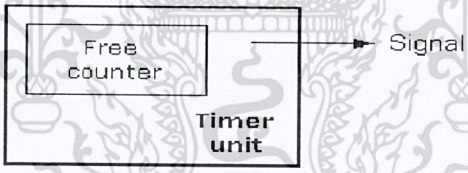
ใช้สำหรับการรับและส่งข้อมูล โดยจะมีการส่งข้อมูลในเป็นลำดับ จากรูปมีสายสามเส้นซึ่งเส้นแรกใช้สำหรับในการส่งข้อมูล เส้นที่สองใช้ในการรับข้อมูล ส่วนเส้นที่สุดท้ายใช้ทั้งในการส่งและรับข้อมูล



รูปที่ 2.5 บล็อกไดอะแกรมของการติดต่อสื่อสารแบบอนุกรม

2.6.6 ตัวกำเนิดสัญญาณเวลา (Timer unit)

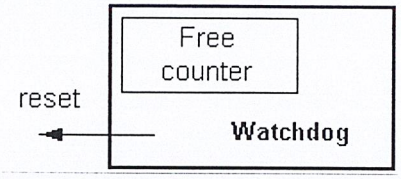
เป็นตัวกำเนิดสัญญาณช่วงเวลาให้กับการทำงานของไมโครคอนโทรเลอร์



รูปที่ 2.6 บล็อกไดอะแกรมของ Timer unit

2.6.7 วอทช์ด็อก (Watchdog)

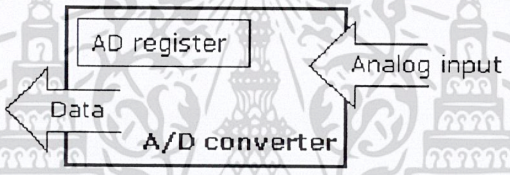
เป็นตัวคอยดูแลควบคุมการทำงานของไมโครคอนโทรลเลอร์โดยไม่ให้เกิดข้อผิดพลาด เช่น ไมโครคอนโทรลเลอร์อาจเกิดการทำงานผิดพลาด ในบางครั้งมีการหยุดเอ็กซ์ซิคิวโปรแกรม ทำให้เกิดผลผิดพลาด หรือในตอนเริ่มทำงานมีสิ่งที่ผิดพลาดเกิดขึ้น ซึ่งถ้าข้อผิดพลาดเหล่านี้ เกิดขึ้นกับเครื่องคอมพิวเตอร์ เราสามารถแก้ ปัญหาได้โดยง่ายด้วยการกดปุ่มรีเซตแล้วมันก็จะสามารถเริ่มทำงานใหม่ได้ แต่สำหรับไมโครคอนโทรลเลอร์แล้วมันไม่มีปุ่มรีเซต จึงต้องใช้ตัว watchdog ในการแก้ปัญหาดังกล่าว



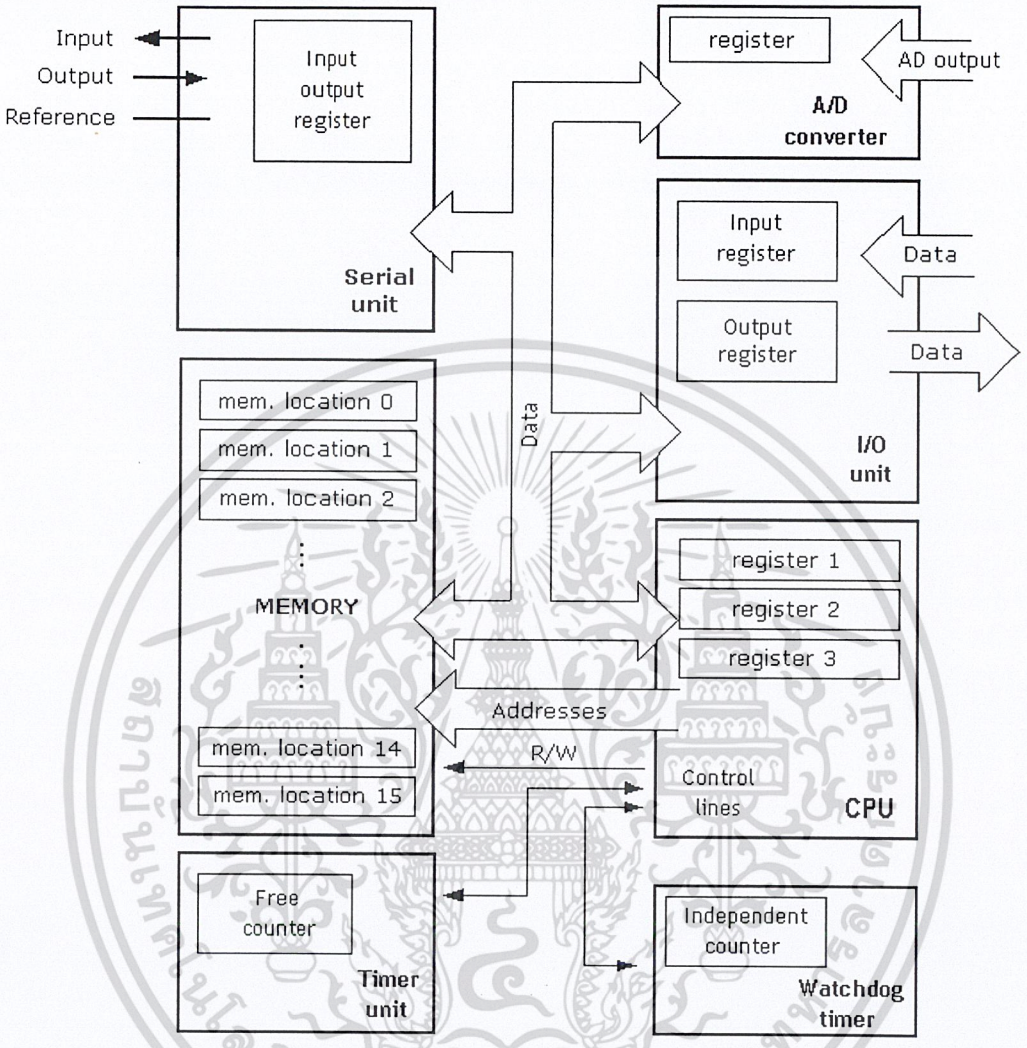
รูปที่ 2.7 บล็อกไดอะแกรมของ Watchdog

2.6.8 หน่วยแปลงสัญญาณอนาล็อกเป็นดิจิตอล (Analog – Digital Converter)

การเปลี่ยนสัญญาณอนาล็อกให้อยู่ในรูปของสัญญาณดิจิตอล (0 หรือ 1) ที่ไมโครคอนโทรลเลอร์สามารถเข้าใจได้ เป็นหน้าที่การทำงานในส่วนของหน่วยแปลงสัญญานอนาล็อกเป็นดิจิตอล หรือ AD converter ซึ่งทำหน้าที่แปลงข้อมูล (information) ที่เป็นค่าอนาล็อก ให้อยู่ในรูปของค่าไบนารี



รูปที่ 2.8 บล็อกไดอะแกรมของ A/D converter



รูปที่ 2.9 ส่วนประกอบโดยรวมของไมโครคอนโทรลเลอร์

2.7 ระบบปฏิบัติการสำหรับระบบฝังตัว

ในการพัฒนาระบบฝังตัวนั้น ระบบปฏิบัติการเป็นองค์ประกอบอย่างหนึ่งที่สำคัญ ของระบบซอร์ฟแวร์ทั้งหมด เพราะเป็นส่วนที่ทำให้การพัฒนาซอร์ฟแวร์แอพพลิเคชันสำหรับงานต่างๆ ทำได้รวดเร็วยิ่งขึ้น สามารถพัฒนาซอร์ฟแวร์แอพพลิเคชันที่มีความซับซ้อนได้ง่ายยิ่งขึ้น สามารถดูแลแก้ไขได้ง่าย สามารถนำโค้ดของแอพพลิเคชันที่มีอยู่กลับมาใช้ใหม่ได้ในรูปของไลบรารี ส่งผลให้ระยะเวลาในการพัฒนาผลิต สั้นลง ลดต้นทุนได้มาก และทำให้เวลาในการเข้าสู่ตลาดของผลิตภัณฑ์ สั้นลงในขณะที่ยังคงรักษาคุณภาพของผลิตภัณฑ์ไว้

2.7.1 คุณสมบัติของระบบปฏิบัติการในงานระบบฝังตัว

- Reliability คือ มีความเชื่อถือได้
- Small kernel คือ มีเคอร์เนลขนาดเล็ก เนื่องจากต้องนำไปใช้ในระบบที่มีข้อจำกัดทางด้านฮาร์ดแวร์ ทั้งในด้านหน่วยความจำที่ใช้เก็บข้อมูล
- Modularity คือ ระบบปฏิบัติการสามารถแบ่งออกเป็นโมดูลย่อยๆ ได้ เพื่อเหมาะสมกับการนำไปปรับใช้กับแต่ละงาน
- Scalability คือ ความสามารถในการรองรับการเพิ่มหรือลดขนาดของฮาร์ดแวร์
- Multiplatform คือ สามารถทำงานได้บนไมโครโพรเซสเซอร์ หรือไมโครคอนโทรเลอร์ ได้หลายตระกูล
- Networking Support คือ สนับสนุนความสามารถด้านเน็ตเวิร์ก
- Power Management คือ การสนับสนุนความสามารถในการประหยัดพลังงาน
- Real-Time Support คือการสนับสนุนการทำงานแบบเรียลไทม์ สำหรับใช้ในบางลักษณะงานที่ต้องการ
- multi-tasking สนับสนุนการทำงานได้หลายๆ งาน

2.7.2 ประเภทของระบบปฏิบัติการที่ใช้ในระบบฝังตัว

2.7.2.1 Real-Time Operating System

คือระบบปฏิบัติการที่รองรับการทำงานตามเงื่อนไขของเวลา ซึ่งต้องสามารถรองรับการทำงานที่ต้องการการตอบรับ ในเวลาที่ถูกต้องตามเหตุการณ์ที่เข้ามาจากภายนอก ตัวอย่างระบบปฏิบัติการเหล่า นี้ในปัจจุบันได้แก่ QNX, VxWorks, LynxOS, RT-Kernel, Itron,pSOS เป็นต้น

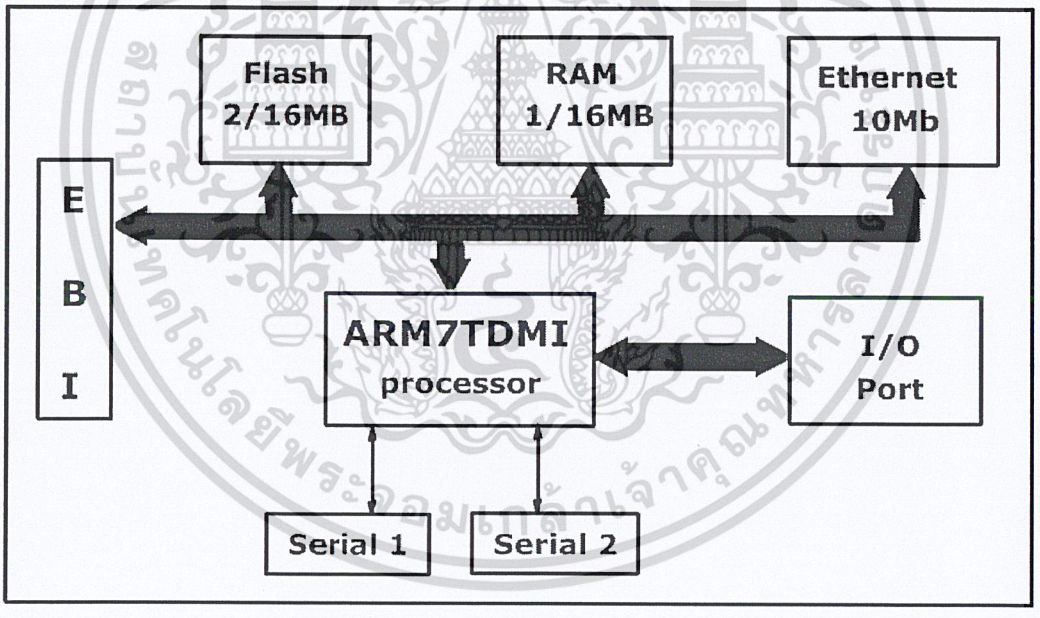
2.7.2.2 Non Real-Time Operating System

คือระบบปฏิบัติการอื่นๆที่ไม่ได้เป็นระบบที่รองรับการทำงานแบบเรียลไทม์ แต่เป็นระบบ ปฏิบัติการที่ใช้ในงานทางด้านระบบคอมพิวเตอร์ฝังตัวทั่วไปเช่น Embedded Linux, RomDos, ELKS เป็นต้น

บทที่ 3
การวิเคราะห์และการออกแบบ

3.1 ภาพรวมของโครงงาน

โครงงานนี้เป็นการพัฒนาบอร์ดควบคุมสำหรับระบบฝังตัว ซึ่งจากการวิเคราะห์โครงสร้างโดยรวมของโครงงานจะประกอบด้วย 2 ส่วนหลักๆ คือ ส่วนของฮาร์ดแวร์และซอร์ฟแวร์ โดยส่วนของฮาร์ดแวร์เป็นการพัฒนาบอร์ดควบคุมที่ใช้กับระบบฝังตัว โดยโครงสร้างจะประกอบด้วย ไมโครโปรเซสเซอร์ ARM7TDMI , หน่วยความจำแฟลช 2 เมกกะไบต์ หน่วยความจำแรม 1 เมกกะไบต์ อีเทอร์เน็ตคอลโทรลเลอร์ความเร็ว 10 เมกกะบิตต่อวินาที , บัสภายนอก(External Bus Interface) สำหรับต่อขยายหน่วยความจำสามารถที่จะขยายสูงสุดได้ถึง 64 เมกกะไบท์, พอร์ตขนานที่ใช้สำหรับติดต่อกับอุปกรณ์ภายนอก , พอร์ตอนุกรมจำนวน 2 พอร์ตโดยพอร์ตหนึ่งสำหรับติดต่อกับโฮสต์(Host System)และอีกพอร์ตสำหรับใช้งานทั่วไป ดังรูป



รูปที่ 3.1 โครงสร้างของบอร์ดควบคุม

ในส่วนของซอร์ฟแวร์จะเป็นการพัฒนาเครื่องมือที่ใช้ในการโปรแกรมมิ่งหน่วยความจำแฟลช(Onboard-Programming) และระบบปฏิบัติการที่ใช้บนบอร์ดควบคุม คือ อีคอส(eCos-Embedded Configuration Operating System)

3.2 ส่วนประกอบทางด้านฮาร์ดแวร์

ในการวิเคราะห์ส่วนประกอบทางด้านฮาร์ดแวร์ เป็นการวิเคราะห์ถึงส่วนที่จำเป็นในการเลือกอุปกรณ์ที่จะนำ มาใช้เป็นส่วนประกอบของบอร์ด ให้มีความเหมาะสมและเป็นไปตามเป้าหมายของโครงงาน โดยในการเลือกจะพิจารณาถึงลักษณะของการเชื่อมต่อ ความสามารถและคุณสมบัติที่ต้องการ รวมทั้งต้องพิจารณาถึงความยากง่ายในการนำมาใช้

3.2.1 เปรียบเทียบไมโครคอนโทรลเลอร์

การเลือกโปรเซสเซอร์จะเลือกเป็นไมโครคอนโทรลเลอร์ มากกว่าที่จะเป็นไมโครโปรเซสเซอร์เพราะได้รวมเอาอุปกรณ์พื้นฐานที่จำเป็นสำหรับบอร์ดควบคุมไว้ ทำให้บอร์ดควบคุมมีขนาดเล็กลง

ตารางที่ 3.1 การเปรียบเทียบคุณสมบัติไมโครคอนโทรลเลอร์ AT91-4x , ElanSC4xx , Motorola 68K

คุณสมบัติ\ซีพียู	Elan TM SC4xx	AT91-4x	Integated 68K MC683xx
ผู้ผลิต(ตระกูล)	AMD(486SX)	Atmel(ARM7TDMI)	Motorola(MC68K)
ชื่ออุปกรณ์	ElanSC400 ElanSC410	AT91M40800 AT91R40807	MC68332 MC68334
สถาปัตยกรรม	CISC	RISC	CISC
ขนาดบัสภายใน/นอก	32/32,16,8	32/16,8	32/32
DRAM Controller	ใช่	ไม่ใช่	ไม่ใช่
MIPS/Bus Frequency	63.5@100Mhz/33,66,100	36@40MHz/40 MAX	9.8@25MHZ
Interfaces/ Address Space	ISA Bus,VL Bus/ 4GB	External Bus/ 64MB	Async SCI and queued sync (SPI) / 4GB
Peripherals Interfaces	Interrupt Controllers UART, Keyboard IF, 32 Gen Purpose I/O, PCMCIA (400), LCD Controller (400)	Timers, Watchdog, Interrupt Controller, Power Management, Oscillator, PLL	Timer , Watchdog , Bus Monitor, Periodic Interrupt Timer, Background Debug Mode, PIT, TPU TimeBase Counter
Vdd (core I/O) Power Dissipation	750mW@33MHz/ 5V 1.8W @ 100MHz	46mW/MHz @ 3.3V	125-700mW @ 5V and 16-25MHz
ราคาประมาณ	452 – 1654 บาท	633 – 856 บาท	966-1850 บาท

เนื่องจากโครงงานนี้ต้องการที่จะพัฒนาบอร์ดควบคุมที่สามารถรองรับการประมวลผลทางด้านเครือข่ายคอมพิวเตอร์ บวกกับความสามารถในการทำมัลติทาสก์กิ้ง(Multi-tasking) ของระบบปฏิบัติการซึ่งต้องการโปรเซสเซอร์ที่มีความเร็วในการประมวลผลสูง แต่ก็ยังคงพิจารณาถึงความเหมาะสมกับระบบฝังตัวขนาดกลาง การสิ้นเปลืองพลังงาน และราคาก็เป็นสิ่งจำเป็น ที่ต้องคำนึงถึง ตารางข้างบนเป็นการเปรียบเทียบไมโครคอนโทรเลอร์ที่มีคุณสมบัติดังได้กล่าวไปแล้ว และมีการใช้งานอย่างกว้างขวาง

3.2.1.1 ไมโครคอนโทรลเลอร์ Elan TM SC4xx Processor

เป็นซีพียูตระกูลอินเทล 486DX ขนาดโปรเซสเซอร์ 32 บิต ใช้พลังงาน 5 โวลต์ ทำงานที่ 33,66,100 MHz มีสถาปัตยกรรมแบบ CISC สนับสนุนบัสภายนอก(external data bus) ทั้ง 8,16 และ 32 บิต มีความสามารถใช้โค้ดร่วมกันได้กับ 8086 , 80286 และโปรเซสเซอร์ตระกูลอินเทล 386 และสามารถเอ็กซิคิวหลายๆ คำสั่งด้วยสัญญาณคล็อกเพียงลูกเดียว มีการทำงานแบบไปป์ไลน์แบบ fetching , decoding , address translation และ execution ข้อดีของไมโครคอนโทรเลอร์เบอร์นี้คือรวมอุปกรณ์ที่มีอยู่ในพีซี(Personal Computer) เกือบทั้งหมดเอาไว้ ทำให้มีจำนวนขามากและแพ็กเกจที่ค่อนข้างยากในการประกอบเป็นบอร์ดควบคุม เบอร์ที่สนใจคือ ElanSC400 เพราะรวมเอาวงจรขับแอลซีดีเข้าไว้ด้วย ทำงานที่ ความเร็วคล็อก 33 MHz สามารถต่อหน่วยความจำแบบดีแรม(DRAM) ได้

3.2.1.2 ATMEL ARM7TDMI (AT91-4x)

AT91M40800 เป็นไมโครคอนโทรลเลอร์ 32 บิต และมีโปรเซสเซอร์ภายในเป็น ARM7TDMI ทีมีสถาปัตยกรรมแบบ RISC ใช้พลังงาน 3.3 โวลท์ ทำงานที่ 33 MHz มีชุดคำสั่งเป็น 16 บิต ใช้พลังงานต่ำ สามารถเอ็กซิคิวหลายๆ คำสั่งด้วยสัญญาณคล็อกเพียงลูกเดียว มีการกำหนดการทำงานของอุปกรณ์ที่ดีสำหรับรองรับแอพพลิเคชั่นแบบ เรียลไทม์(real – time)และมีการจัดระดับความสำคัญ (Priorty) ของอินเตอร์รัพท์เวกเตอร์ 8 ระดับ มีพอร์ตไว้สำหรับต่อขยายหน่วยความจำ (EBI = external bus interface) เหมาะสำหรับที่จะนำไปใช้พัฒนาระบบฝังตัว เนื่องจากเป็นโปรเซสเซอร์ที่เลือกใช้จะได้กล่าวในรายละเอียดต่อไป

3.2.1.3 Integated 68K MC683xx

เป็นไมโครโปรเซสเซอร์ 32 บิต ซึ่งอยู่ในตระกูลไมโครโปรเซสเซอร์ 68000 ของ Motolola ใช้เทคโนโลยี VLSI (Very Large Scale Integration) คาต้าบัส32 บิต และ แอดเดรส 32 บิต สนับสนุนภาษาระดับสูง, แคชคำสั่ง , มีการติดต่อภายในไมโครโปรเซสเซอร์เป็นแบบขนาน สามารถเอกซิคิวได้หลายคำสั่งพร้อมกัน มีโครงสร้างบัสเป็นแบบอะซิงโครนัส โปรเซสเซอร์สามารถขยายขนาดของบัสได้ เป็นไมโครโปรเซสเซอร์ที่ออกแบบมาใช้สำหรับระบบฝังตัว ข้อดีของไมโครคอนโทรเลอร์เบอร์นี้คือรวมอุปกรณ์ต่างเข้าไว้หลายตัวด้วยกัน แต่ก็ยังมีความเร็วไม่ค่อยดีนักเมื่อเปรียบเทียบกับโปรเซสเซอร์ที่มีอยู่ในตาราง อีกทั้งราคาก็ยังแพงอยู่มาก

3.2.2 เหตุผลที่ใช้ในการเลือกไมโครโปรเซสเซอร์

ไมโครโปรเซสเซอร์ที่มีอยู่อย่างมากมายนั้น ต่างก็ได้ถูกออกแบบมาเพื่อให้ใช้งานในด้านต่างๆ กัน และมีความสามารถในการทำงานต่างกันด้วย หลักที่ใช้ในการพิจารณามีหลายอย่างไม่ว่าจะเป็น ด้านประสิทธิภาพการทำงาน การใช้พลังงาน ความยากง่ายในการพัฒนา รวมไปถึงการมีเครื่องมือสนับสนุนที่ใช้ในการพัฒนาและราคาของตัวไมโครโปรเซสเซอร์เมื่อเทียบกับการนำมาใช้งาน เนื่องจากโครงงานนี้เป็นการพัฒนาบอร์ดที่ใช้เป็นต้นแบบสำหรับระบบฝังตัว จึงทำให้มีข้อพิจารณาในเบื้องต้นในการคัดเลือกไมโครโปร เซสเซอร์ ดังนี้

ตารางที่ 3.2 การเปรียบเทียบไมโครคอนโทรลเลอร์ AT91M40800 , ElanSC400 ,MC68332

ข้อพิจารณา	AT91M40800	ElanSC400	MC68332
ขนาด (บิต)	32	32	32
สถาปัตยกรรม	RISC	CISC	CISC
พลังงานที่ใช้(วัตต์)	0.15	1.14	1.2
ความเร็ว	30MIPS@33MHz	7.6MIPS@33MHz	9.8MIPS@25MHZ
อุปกรณ์พื้นฐาน Timers, Watchdog, Interrupt Controller, Serial Port	มี	มี	มี
พอร์ต JTAG	มี	มี	มี
พื้นที่หน่วยความจำ	64MB	4GB	4G
ราคา (บาท)	633	1,386	966

เนื่องจากปัจจุบันนี้การทำงานของชิปและอุปกรณ์โดยทั่วไปจะสนับสนุนการทำงานที่ขนาด 32 บิตมากขึ้นราคาไม่แตกต่างกับโปรเซสเซอร์มากนัก และความต้องการของผู้ใช้งานที่ต้องการความเร็วมากขึ้น จึงให้ความสำคัญกับโปรเซสเซอรขนาด 32 บิต เนื่องจากต้องการให้ง่ายในการออกแบบจึงต้องหาอุปกรณ์ที่รวมเอาสิ่งจำเป็นพื้นฐานเอาไว้ การสิ้นเปลืองพลังงานก็เป็นสิ่งที่สำคัญ นอกจากนี้อุปกรณ์ในระบบฝังตัวมักจะไม่มีขนาดเล็กจึงจำเป็นต้องมีสิ่งที่ใช้ในการตรวจสอบข้อผิดพลาด การทำ On-Board Programming เช่น พ็อต JTAG(Joint Test Action Group)

ราคาก็เป็นอีกสิ่งหนึ่งที่ใช้เป็นปัจจัยในการพิจารณา จากตารางด้านล่างจะเห็นได้ว่า ARM7TDMI เป็นไมโครคอนโทรลเลอร์ ที่มีราคาถูกกว่าตัวอื่น ๆ และเมื่อได้พิจารณาประสิทธิภาพโดยรวมกับการนำมาใช้งานสำหรับโครงงานนี้ ARM7TDMI ก็เป็นไมโครคอนโทรลเลอร์ที่น่าจะนำมาใช้มากที่สุด

3.3 องค์ประกอบของบอร์ด

ซึ่งบอร์ดที่พัฒนาขึ้นจะใช้ไมโครคอนโทรลเลอร์ AT91M40800 ซึ่งเป็นของบริษัท ATMEL ที่มี ไมโครโพรเซสเซอร์เป็น ARM7TDMI

3.3.1 ส่วนประกอบของบอร์ด

- ไมโครคอนโทรลเลอร์ AT91M40800 โดยมีโพรเซสเซอร์ภายในเป็น ARM7TDMI
- ตัวสร้างสัญญาณนาฬิกา (Clock Generator)
- SRAM 1 เมกกะไบต์ สามารถต่อขยายได้ 16 เมกกะไบต์
- Flash Memory 2 เมกกะไบต์ สามารถต่อขยายได้ 16 เมกกะไบต์
- EBI บัส สำหรับต่อขยายหน่วยความจำ
- พอร์ตขนานที่ใช้สำหรับติดต่อกับอุปกรณ์ภายนอก
- ซีเรียลพอร์ตจำนวน 2 พอร์ต
- พอร์ตขนาน
- ปุ่มรีเซ็ต

3.3.2 ไมโครโปรเซสเซอร์ ARM7TDMI

เป็นไมโครโปรเซสเซอร์ขนาด 32 บิต ตระกูล ARM (Advanced RISC Machines) มีประสิทธิภาพสูง สิ้นเปลืองพลังงานน้อย ราคาถูก

สถาปัตยกรรม ARM ใช้หลักการของ RISC (Reduced Instruction Set Computer) โดยมีชุดคำสั่งและวิธีการดีโค้ด (Decode) ที่ง่ายกว่า CISC (Complex Instruction Set Computer) ทำให้คำสั่งเหล่านี้มี Throughput ที่สูง และเหมือนกับว่ามีการตอบรับการอินเตอร์รัพท์ในลักษณะของเรียลไทม์จากชิปที่มีขนาดเล็กและทำให้คุมค่าในการลงทุน

มีการนำหลักการไปป์ไลน์ (Pipeline) ไปใช้กับทุกส่วนของโพรเซสเซอร์และกับระบบหน่วยความจำซึ่งสามารถทำการโอเปอเรชั่นได้อย่างต่อเนื่อง อาทิเช่น เมื่อมีคำสั่งหนึ่งถูกเอ็กซีคิวต์ โพรเซสเซอร์จะเริ่มทำการดีโค้ดและคำสั่งต่อไปจะถูกดึงมาจากหน่วยความจำ

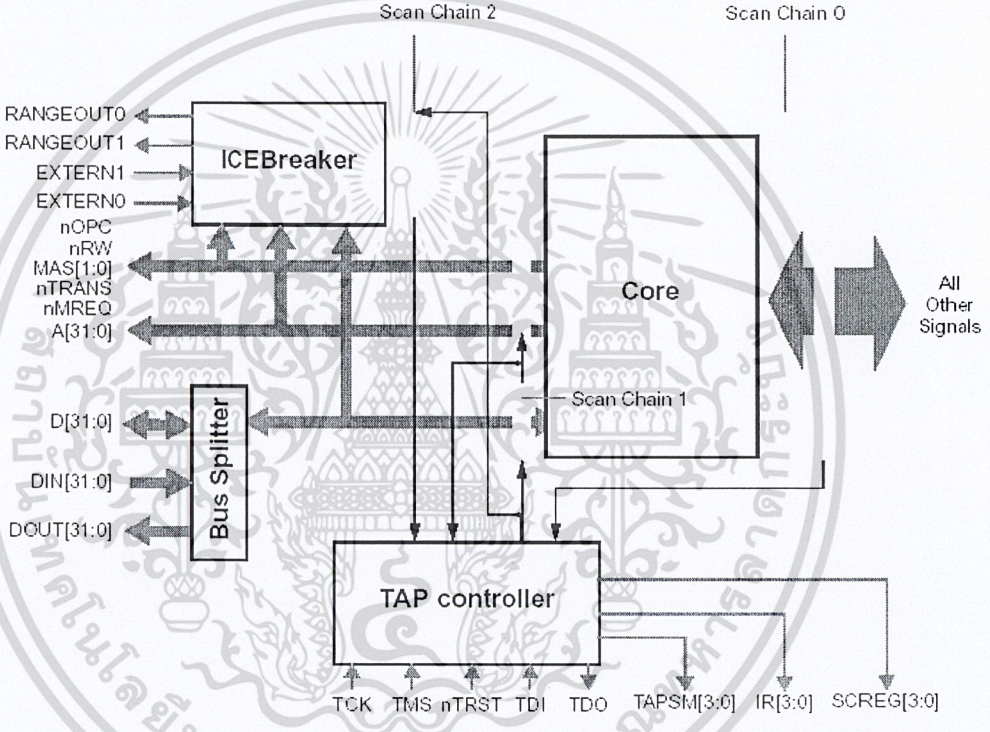
ในการเชื่อมต่อกับหน่วยความจำใน ARM ถูกออกแบบให้ใช้งานได้อย่างมาประสิทธิภาพกับระบบหน่วยความจำที่ราคาไม่สูง สัญญาณควบคุมที่ถูกควบคุมโดยไปป์ไลน์นั้นใช้ตรรกกะมาตรฐานแบบพลังงานต่ำและรูปแบบของสัญญาณ ที่สามารถใช้ได้กับโหมดการเข้าถึงแบบเร็วของหน่วยความจำมาตรฐานในอุตสาหกรรม

ส่วนคุณสมบัติอื่นๆเพิ่มเติมนอกจากนี้ได้แก่

- ดีบักแบบบนชิปโดย ICEBreaker
- ไปป์ไลน์ 3 สถานะ
- โพรเซสเซอร์ RISC 32 บิต เป็นสถาปัตยกรรมแบบ VON Neuman load/stroe ซึ่งมีบัส สำหรับแอดเดรสและ ข้อมูลสำหรับชุดคำสั่งและชุดข้อมูล

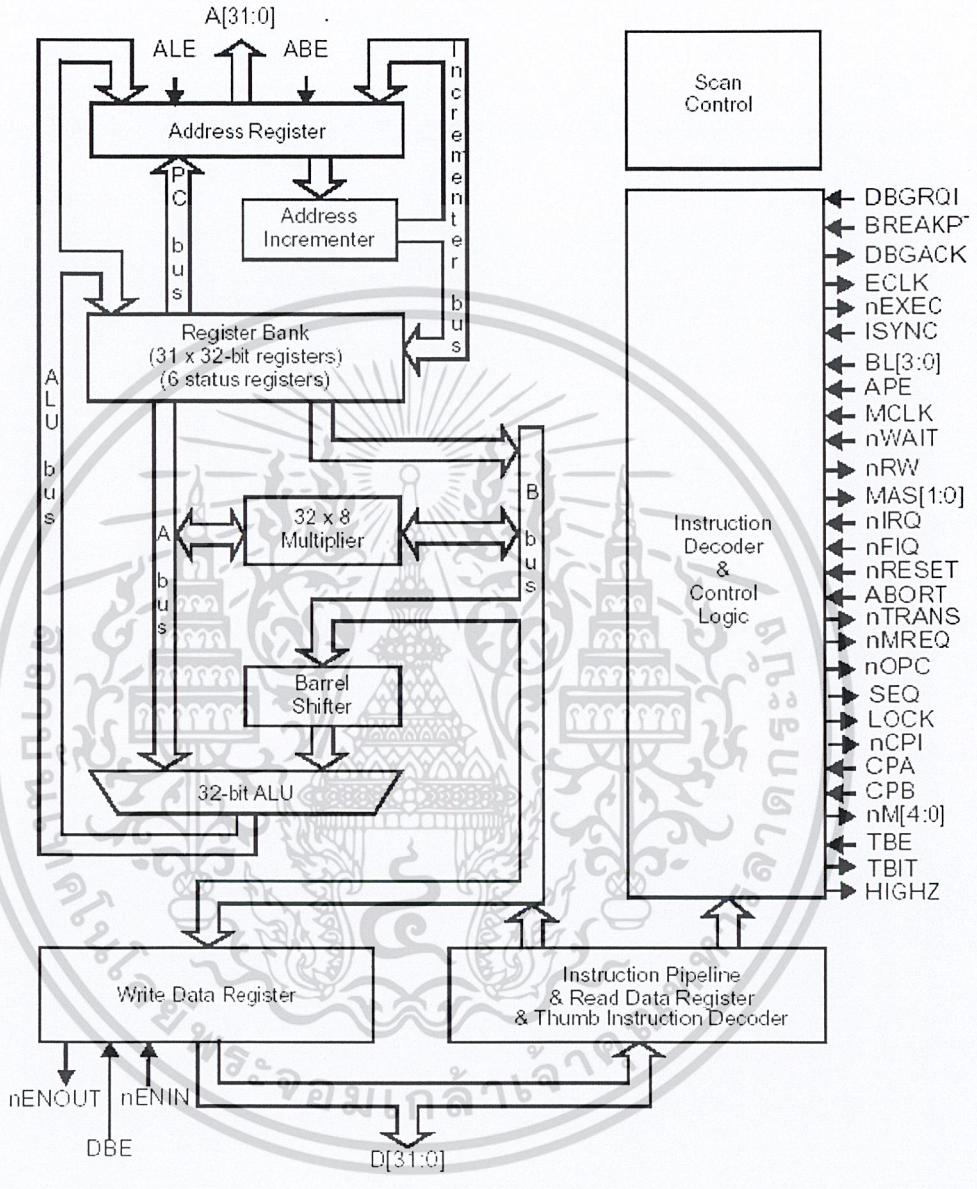
- หน่วยประมวลผลมีชุดคำสั่ง 2 แบบ ได้แก่ ชุดคำสั่ง ARM 32 บิต และชุดคำสั่ง Thumb ขนาด 16 บิต โดย คำสั่งเหล่านี้สามารถที่ทำการประมวลผลกับข้อมูลได้หลายขนาด เช่น 8 , 16 , 32 บิต
- หน่วยประมวลผลมีโหมดการทำงาน 7 โหมด โดยแต่ละโหมดมีรีจิสเตอร์เพื่อเพิ่มความเร็วในการจัดการแอ็กซ์เซปชั่น (Exception)
- โพรเซสเซอร์มีรีจิสเตอร์ขนาด 32 จำนวน 37 ตัว รวมถึง 6 รีจิสเตอร์สถานะ

ARM7TDMI Block Diagram



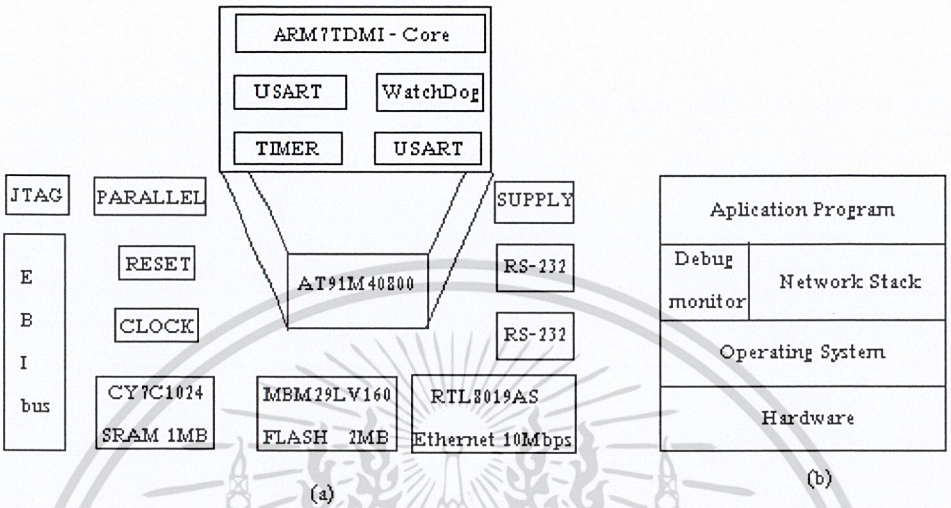
รูปที่ 3.2 โครงสร้างของ ARM7TDMI

ARM7TDMI Core Diagram



รูปที่ 3.3 โครงสร้างภายในของ ARM7TDM

3.3.3 สถาปัตยกรรม



รูปที่ 3.4 (a) โครงสร้าง (b) สถาปัตยกรรมทางด้านซอฟต์แวร์

3.3.4 ไมโครคอนโทรลเลอร์ AT91M40800

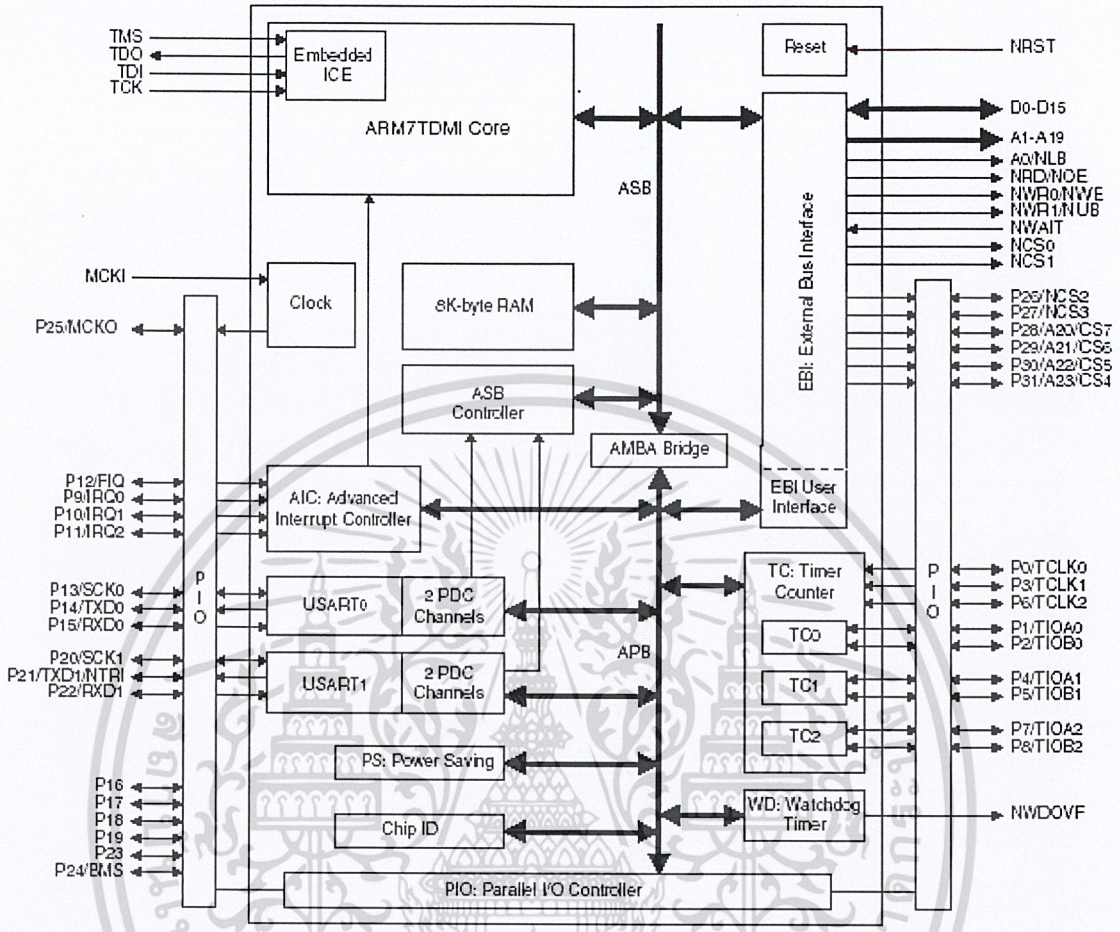
ไมโครโปรเซสเซอร์ที่เลือกใช้ในการออกแบบบอร์ดควบคุม AT91M40800 เป็นไมโครคอนโทรเลอร์ ARM7TDMI 16/32 บิต สถาปัตยกรรม RISC ความเร็ว 30 MIPS ที่สัญญาณคล็อก 33 MHz ขณะที่สิ้นเปลืองกำลังงานน้อยกว่าหนึ่งวัตถ์ ภายในได้รวมเอาอุปกณ์ที่เป็นพื้นฐานในการพัฒนาระบบฝังตัว ได้แก่

- 8 Kbyte SRAM 32 บิตสามารถอ่านและเขียนใน single cycle clock
- อินเตอร์รัพท์คอนโทรลเลอร์ 8-level Priority, Individually Maskable, Vectored Interrupt Controller และอินเตอร์รัพท์จากภายนอก 4 เส้น
- พอร์ท Parallel I/O สามารถโปรแกรมได้ 32 เส้น
- 2 USART (Universal Asynchronous Receiver/Transmitter) สำหรับการสื่อสารอนุกรมความเร็วสูง
- Programmable Timer/Counter 16 บิต 3 ตัว
- Programmable Watchdog Timer
- Power Saving Controller

สถาปัตยกรรมของ AT91M40800

ประกอบด้วยบัสหลัก 2 เส้นซึ่งเชื่อมต่อกันด้วยโดย AMBA Bridge คือ

ASB (The Advanced System Bus) ถูกออกแบบสำหรับประสิทธิภาพที่สูงที่สุด ทำการเชื่อมต่อ



รูปที่ 3.5 บล็อกไดอะแกรมของไมโครคอนโทรเลอร์ AT91M40800

โพรเซสเซอร์ หน่วยความจำบนชิพขนาด 32 บิตและหน่วยความจำภายนอกรวมทั้งอุปกรณ์ต่างๆผ่าน EBI (External Bus Interface)

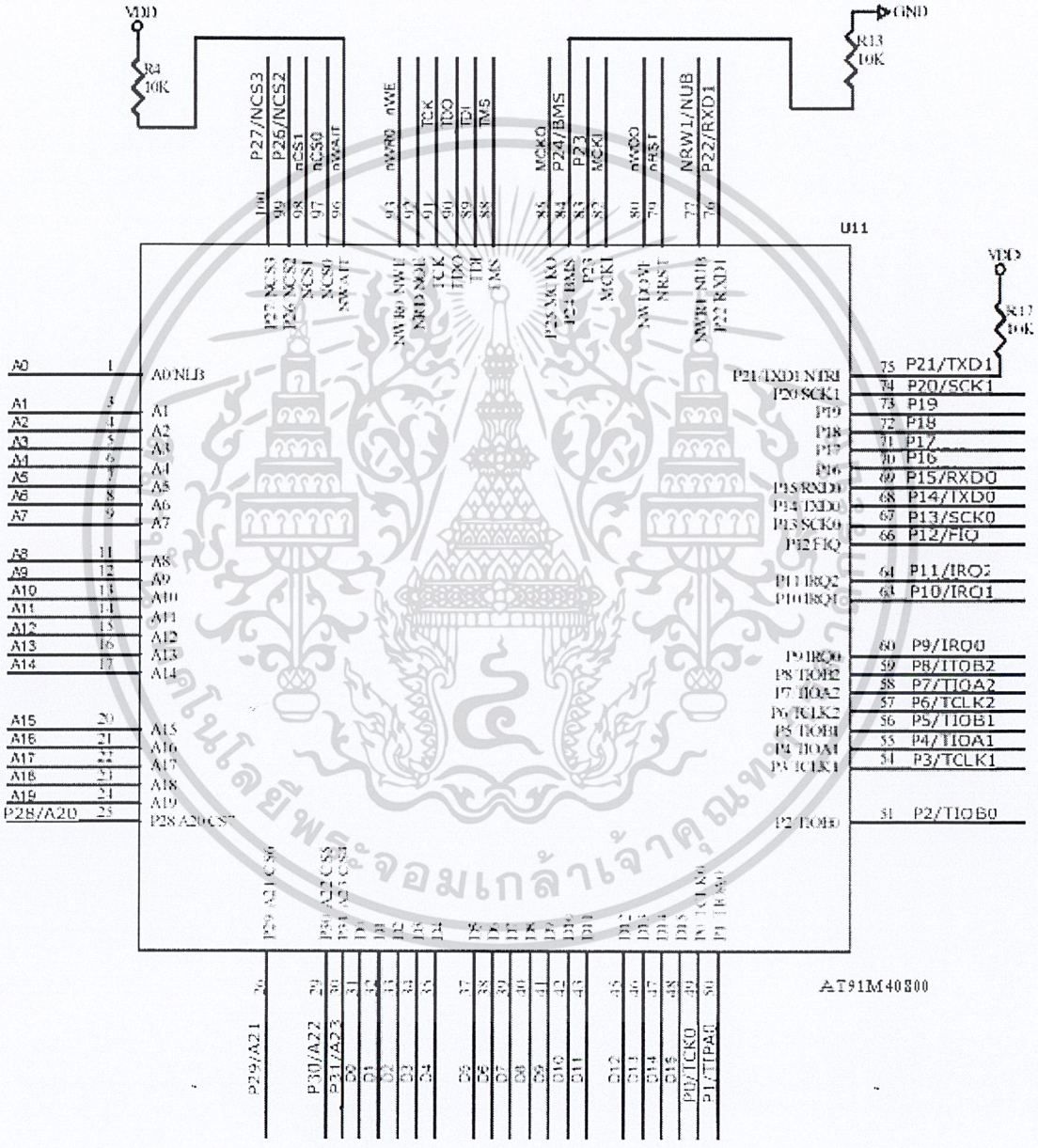
APB (The Advanced Peripheral Bus) ถูกออกแบบสำหรับเข้าถึงอุปกรณ์ต่างๆบนชิพและใช้พลังงานน้อย

PDC (Peripheral Data Controller) ทำการโอนย้ายข้อมูลระหว่าง USARTs ที่อยู่บนชิปกับหน่วยความจำทั้งภายในและนอกชิพที่สำคัญ PDCจะทำการลดโอเวอร์ฮีดที่โปรเซสเซอร์ทำการอินเตอร์ แฮนดิ้ง และลดจำนวนไซเคิลของสัญญาณนาฬิกา ที่ต้องการสำหรับโอนย้ายข้อมูล ซึ่งสามารถทำการโอนย้ายขยาด 64 กิโลไบต์ได้อย่างต่อเนื่อง ทำให้เพิ่มประสิทธิภาพและลดการสิ้นเปลืองพลังงานลงได้

อุปกรณ์รอบข้างภายในถูกออกแบบให้ใช้ชุดคำสั่งในการทำงานน้อยในการทำงานแต่โดยแต่ละตัวจะมีช่วงแอดเดรสขนาด 16 กิโลไบต์ในขอบเขต 3 เมกกะไบต์บนช่วงของ 4 แอดเดรสกิกกะไบต์ เพื่อให้ประ สิทธิ

ภาพในการดำเนินงานกับบิตยังได้มีการแยกรีจิสเตอร์ ที่มีการใช้บ่อยเป็น 3 ตำแหน่งและให้ตำแหน่งแรก เป็นการเซตบิต ตำแหน่งที่สอง เป็นการเคลียร์บิต และตำแหน่งที่สามเป็นการอ่านค่าของรีจิสเตอร์นั้นๆ

สัญญาณภายนอกทั้งหมดของอุปกรณ์รอบข้างจะถูกควบคุมโดย PIO (Parallel I/O Controller) ซึ่งสามารถโปรแกรมให้ ตัวมันเองเพิ่มตัวกรองอินพุตได้



รูปที่ 3.6 ไมโครโปรเซสเซอร์ Schematic diagram

3.3.5 หน่วยความจำระบบ

บอร์ดควบคุมที่สร้างขึ้นมีหน่วยความจำระบบดังนี้

1. MBM29LV160 เป็นหน่วยความจำแฟลชขนาด 2 เมกกะไบต์ มีความสามารถโปรแกรมได้ไม่น้อยกว่า 100,000 ครั้ง การเข้าถึงข้อมูล 90 นาโนวินาที(ns) สำหรับเก็บการทำงานเริ่มต้นของระบบ
2. SRAM เบอร์ CYC1049 ขนาด 512 เมกกะไบต์ 2 ตัว เวลาในการเข้าถึง 15 นาโนวินาที ทำให้ไม่ต้องใช้ wait state

3.3.6 การจัดการหน่วยความจำ

ไมโครคอนโทรเลอร์ AT91M40800 สามารถโปรแกรมแต่เลือกชิพได้ 8 ตัว หรือ 4 ตัว แต่ในบอร์ดควบคุมที่ได้ออกแบบไว้ให้เลือกการทำงานได้กับชิพ 4 ตัว แต่ละตัวสามารถอ้างหน่วยความจำได้ 16 MB ทาให้สามารถอ้างหน่วยความจำได้ทั้งหมด 64 MB สัญญานที่ใช้ในการเลือกชิพคือ NCS0 ถึง NSC3 สามารถที่จะโปรแกรมเลือกขนาด Data bus ได้ทั้ง 8 หรือ 16 บิต ตารางข้างล่างแสดงการทำ Memmory Map

ตารางที่ 3.3 การจัดการหน่วยความจำ (Memory Map)

Address Range	Function	Chip Select
0x00000000 – 0x00000200	Internal SRAM 8 Kbyte	-
0x01000000 - 0x011FFFFF	Flash Memory 2Mbyte	NCS0
0x02000000 – 0x020FFFFF	External SRAM 1 Mbyte	NCS1
0x03000000 - 0x03000400	Ethernet Controller	NCS2
0xFFC00000 – 0xFFFFFFFF	On-Chip peripherals	-

3.3.7 หน่วยความจำ I/O

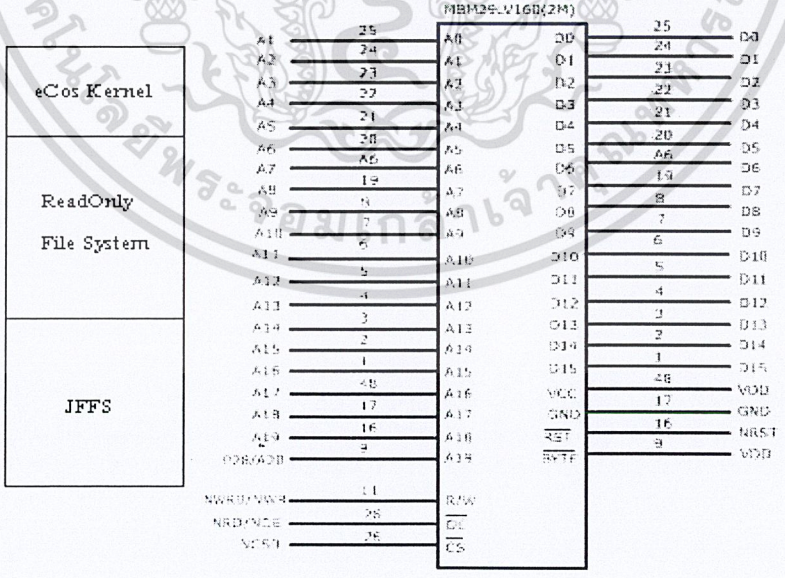
หน่วยความจำ I/O สามารถแบ่งได้เป็น 2 ส่วนในแอดเดรสทั้งหมดของ ARM7TDMI 4 GB คือ พื้นที่แอดเดรสของ EBI-bus (Externale Bus Interface) ตั้งแต่ 0x00400000 – 0xFFBFFFFF ซึ่งสามารถเลือกโดย NSC3 ซึ่งไม่ถูกใช้งานในขณะนี้ และส่วนที่สองเป็น On-Chip Periperals ตั้งแต่แอดเดรส 0xFFC00000 – 0xFFFFFFFF แต่ไม่ได้ถูกใช้งานทั้งหมด การจัดหน่วยความจำของ I/O อ้างถึงใน DATA Sheet ของ AT91x4x ของ Atmel ARM7TDMI ไมโครคอนโทรเลอร์ ดังนี้

ตารางที่ 3.4 การจัดการหน่วยความจำ I/O (Memory Map I/O)

Address Range	Function	Chip Select
0xFFE00000 – 0xFFE03FFF	EBI-external bus interface 16 kb	-
0xFFF00000 – 0xFFF03FFF	Special Function 16 Kbyte	-
0xFFFCC00 - 0xFFFC3FFF	USART1 16 Kbyte	
0xFFFD0000 – 0xFFFD3FFF	USART0 16 KByte	-
0xFFFE0000 – 0xFFFE3FFF	Timer Counter 16 KByte	-
0xFFFF0000 – 0xFFFF3FFF	PIO 16 Kbyte	-
0xFFFF4000 – 0xFFFF7FFF	Power saving	-
0xFFFF8000 – 0xFFFFBFFF	Watchdog Timer	-
0xFFFFF000 – 0xFFFFFFFF	Advance interrupt Controller	-

3.3.8 หน่วยความจำแฟลช (Flash Memory)

หน่วยความจำแฟลช(Flash ROM) ที่ใช้งานกับบอร์ดควบคุมคือ ฟูจิซึ (Fujishu) เบอร์ MBM29LV160 อ่านและเขียนด้วยไฟเลี้ยง 3.0V มีขนาดของ 2 Mbyte จะถูกกำหนดไว้ที่แอดเดรส 0x01000000 หลังจากทำการรีแม็บ (Remap memory) รูปที่ 3.7 เป็นการแสดงวงจรที่ต่อใช้งาน



รูปที่ 3.7 แฟลช Schematics Diagram และ Memory Layout

Layout ของหน่วยความจำแฟลซ

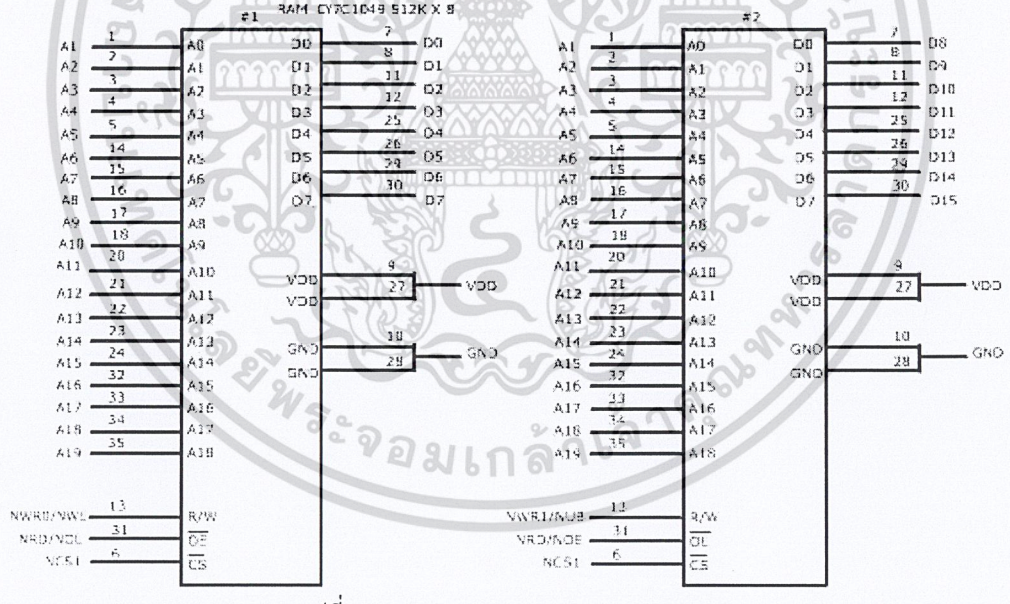
สำหรับบอร์ดควบคุมที่ได้ออกแบบไว้จะเก็บข้อมูลสำหรับระบบดังนี้

- eCos Kernel ที่เริ่มต้นของหน่วยความจำแฟลซ
- ไฟล์ระบบจะถูกเรียกใช้โดย eCos Kernel สามารถอ่านได้อย่างเดียว
- JFFS(Journaling Flash File System) ใช้ในการเข้าใช้งานของแอพพลิเคชั่น

สิ่งที่ต้องพิจารณาเป็นกรณีพิเศษคือ หน่วยความแฟลซจะถูกอ่านในครั้งแรกเมื่อจ่ายไฟให้กับบอร์ดควบคุมหรือทำการรีเซท ซึ่งถูกเรียกใช้งานโดย NCS0 อยู่ที่แอดเดรส 0x00000000 ดังนี้ตำแหน่งเริ่มต้นของแฟลซจะเก็บโค้ดโปรแกรมบูทระบบ หลังจากทำการรีแม็บแล้ว หน่วยความจำแฟลซ จึงจะถูกกำหนดที่แอดเดรส 0x01000000 และหน่วยความจำที่ตำแหน่ง 0x00000000 จะกลายเป็น SRAM ภายใน 8Kbyte

3.3.9 หน่วยความจำแรม

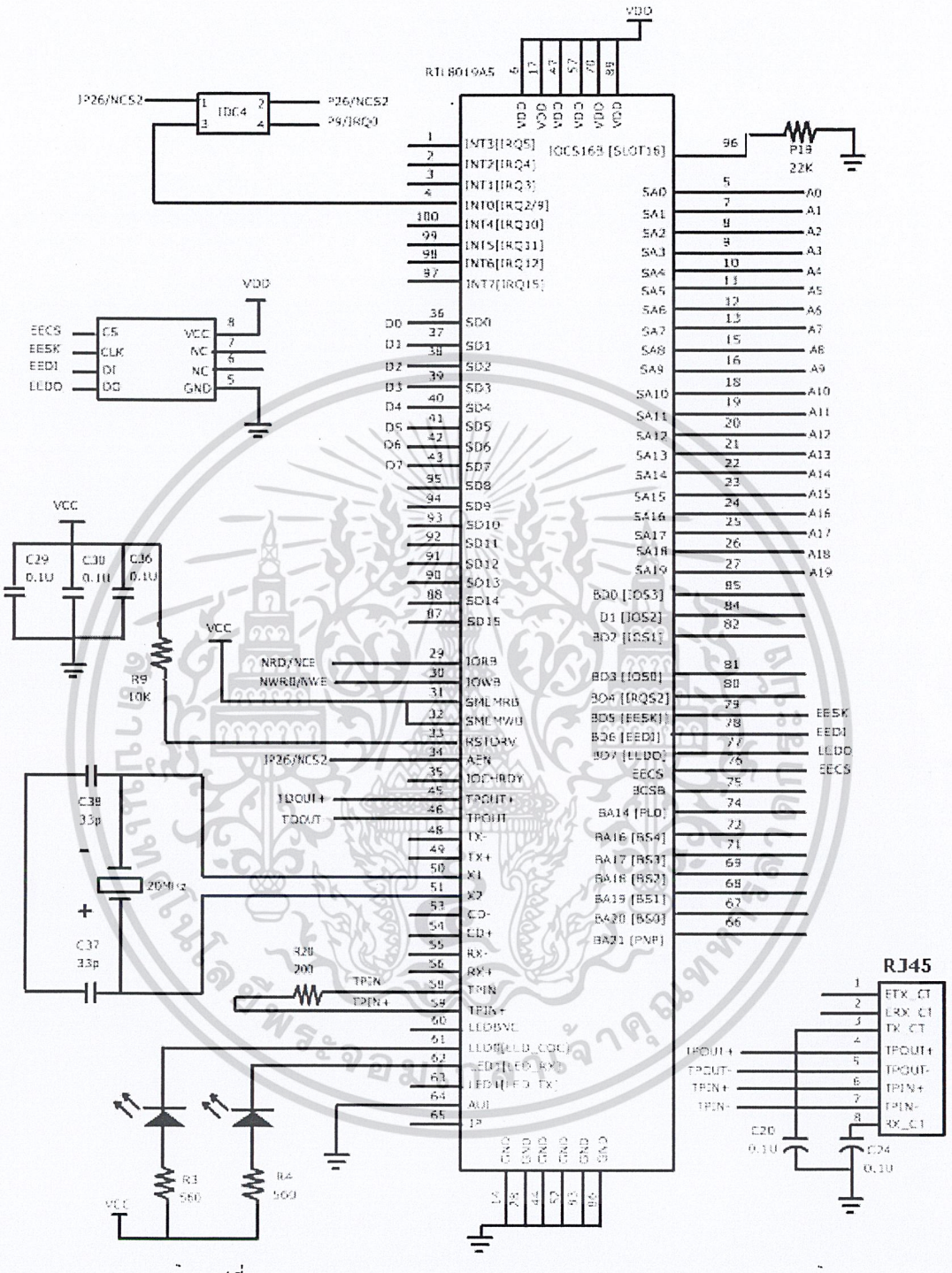
แรมที่ใช้ในบอร์ดควบคุมคือ CY7C1024 ขนาด 512K x 8 ทั้งหมด 2 ตัวรวมเป็น 1 MB ใช้ไฟเลี้ยง 3.0V สามารถทำงานร่วมกับไมโครคอนโทรเลอร์โดยไม่มี Wait State ซึ่งจะถูกกำหนดไว้ที่ แอดเดรส 0x02000000 จะเรียกใช้ในขณะรันโปรแกรม รูปที่ แสดงการ Schematic ของ SRAM



รูปที่ 3.8 Schematic diagram ของ SRAM

3.3.10 Ethernet Controller

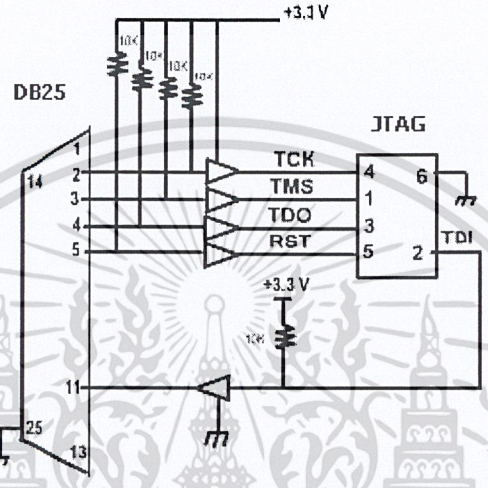
ในบอร์ดควบคุมได้ทำการเพิ่มความสามารถในการติดต่อผ่านเครือข่ายคอมพิวเตอร์ โดยติดตั้ง Ethernet Controller เบอร์ RealTek RTL8019AS ความเร็ว 10 Mbps ต่อเข้า data bus 8 bit ถูกกำหนดในพื้นที่ I/O แอดเดรส 0x03000000 โดยใช้งานสัญญาณเลือกชิพ NCS2 รูปที่ ค แสดงวงจรที่ใช้ในการออกแบบ



รูปที่ 3.9 Schematics Diagram ของภาค Ethernet Controller

3.3.11 พอร์ท JTAG

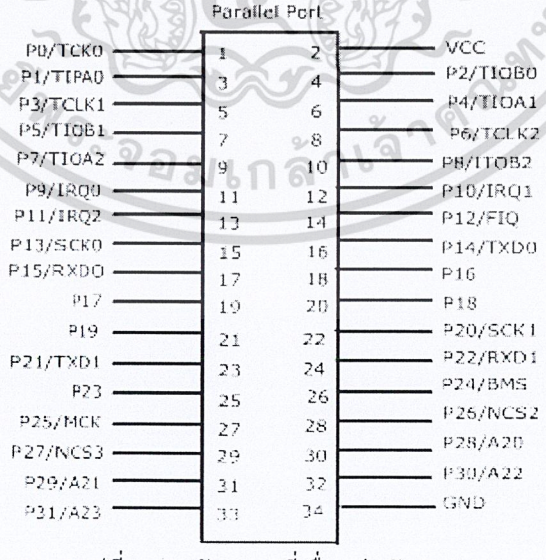
สำหรับในระบบฝังตัวแล้วอุปกรณ์ทุกอย่างจะยึดเข้ากับแผงวงจร เพื่อให้มีขนาดเล็ก และราคาถูก ดังนั้น เมื่อต้องการที่จะตรวจสอบข้อผิดพลาด หรือแม้กระทั่งการปรับปรุงซอฟแวร์ต้องมีอินเตอร์เฟสสำหรับติดต่อกับอุปกรณ์ JTAG(Joint Test Access Group) รูปที่ 3.10 แสดงการต่อพอร์ท JTAG



รูปที่ 3.10 การเชื่อมต่อ JTAG พอร์ท

3.3.12 พอร์ทขนาน (PIO)

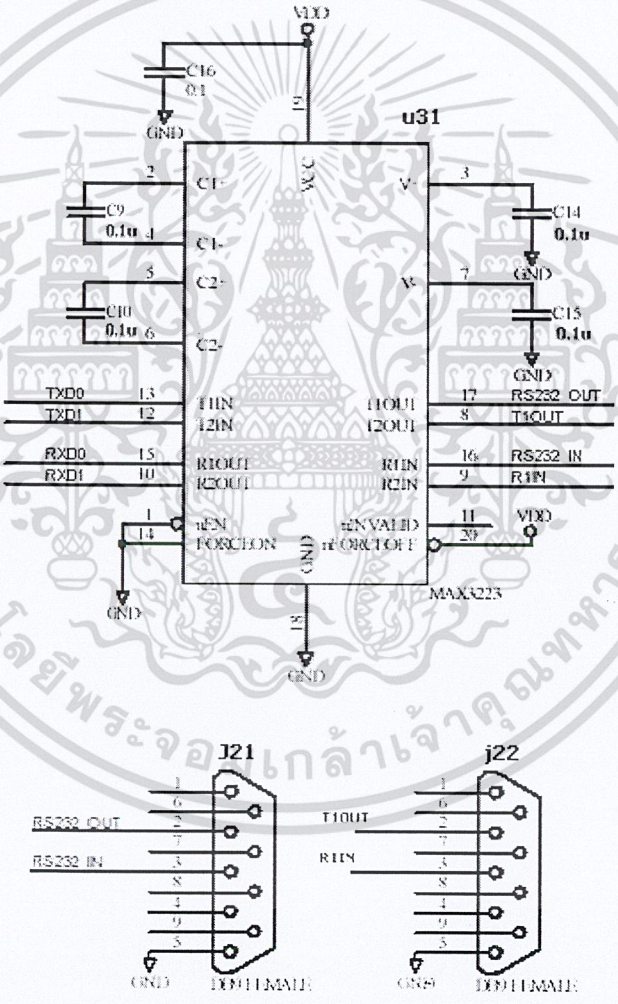
พอร์ทขนานเป็นอีก I/O พอร์ทที่มีอยู่ในบอร์ดควบคุมมีขนาด 32 เส้น ซึ่งสามารถที่จะโปรแกรมใช้งานได้ทั้งหมด สำหรับแอดเดรสที่ใช้การอ้าง PIO อยู่ที่ 0xFFFF0000 – 0xFFFF3FFF



รูปที่ 3.11 สัญญาณที่เชื่อมต่อกับ PIO

3.3.13 USART

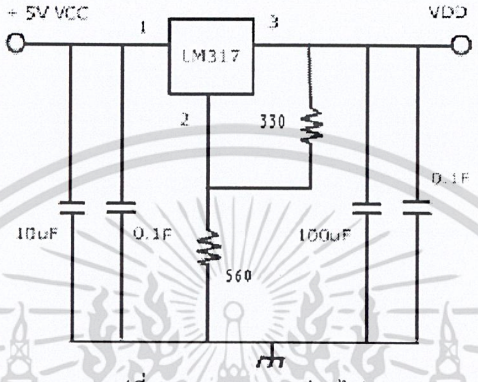
ในบอร์ดควบคุมที่ได้ออกแบบขึ้นที่อินเตอร์เฟสกับ USART (Universal Asynchronous Receiver/Transmitter) สำหรับการสื่อสารแบบอนุกรม ซึ่งจะถูกต่อเข้ากับ DB9-RS232 สามารถรองรับการสื่อสารที่ความเร็วสูงสุด 230400 bps เป็นการเชื่อมต่อแบบ NULL Modem และ แอดเดรสของ USART ทั้งสอง 2 ช่องจะอยู่ที่ 0xFFFD0000 – 0xFFFD3FFF สำหรับ USART0 และ 0xFFFCC00 - 0xFFFC3FFF สำหรับ USART1 ในการติดต่อกับโฮส(Host System) จะกำหนดไว้ที่ USART0 รูปที่ 3.12 เป็นวงจรที่ใช้งานการติดต่อสื่อสารอนุกรม โดยมีไดร์เวอร์เป็นไอซี เบอร์ MAX3223



รูปที่ 3.12 USART Schematic Diagram

3.3.1.4 ภาคจ่ายไฟ(Power Suppy)

เนื่องจากไฟที่เลี้ยงวงจรส่วนใหญ่ใช้แรงดัน 3.3 โวลต์ แต่ไฟที่จ่ายให้กับบอร์ดควบคุมนั้นมีค่า 5 โวลต์ ดังนั้นต้องทำการควบคุมแรงดังให้เหลือเพียง 3.3 โวล์ต IC ที่ใช้คือ LM317 รูปที่ 3.12 . แสดงวงจรของภาคจ่ายไฟที่ใช้งานจริง



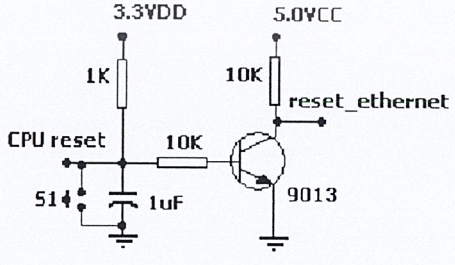
รูปที่ 3.13 วงจรภาคจ่ายไฟ

3.3.15 ภาคกำเนิดสัญญานคล็อก(Clock Generator)

ทำหน้าที่กำเนิดสัญญาณคล็อกให้กับไมโครคอนโทรเลอร์ ความถี่ 33 Mhz รูปข้างล่างแสดงการต่อใช้งาน

รูปที่ 3.14 Schematic Diagram ของภาคกำเนิดสัญญาณคล็อก

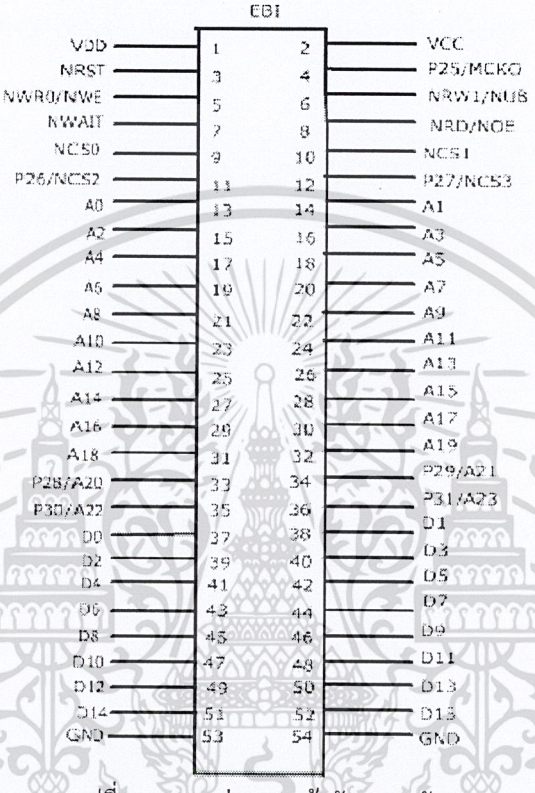
3.3.16 ภาคสัญญาณรีเซท (Reset signal)



รูปที่ 3.14 Schematic Diagram ภาคสัญญาณรีเซท

3.3.17 พอร์ท EBI(External Bus Interface)

EBI Ports จะถูกใช้ในการขยายหน่วยความจำและอุปกรณ์ I / O จากบอร์ดควบคุมรูปที่ 11 พอร์ท EBI ซึ่งจะรวมเอาสัญญานควบคุม ,แอดเดรสบัส และ ดาต้าบัส ไว้ จะมีทั้งหมด 54 ขา



รูปที่ 3.16 การต่อวงจงเข้ากับ EBI-บัส

3.3.18 คุณสมบัติทางไฟฟ้า

- ทำงานได้กับแหล่งจ่าย 5 VDC เท่านั้น
- กินกระแสสูงสุด 750 mA@5VDC

3.3.19 อุณหภูมิทำงาน

- อุณหภูมิทำงาน 0 ถึง +70 องศาเซลเซียส

3.4 ส่วนประกอบทางด้านซอร์ฟแวร์

ในส่วนนี้ได้ทำการวิ เคราะห์เพื่อหาซอร์ฟแวร์ที่ใช้ในการพัฒนาที่สะดวกและเหมาะสม โดยสามารถที่จะทำการพัฒนาได้บนเครื่องคอมพิวเตอร์ทั่วไปได้ รวมทั้งมีเครื่องมือให้ได้ง่าย โดยตัวเครื่องที่ใช้พัฒนาจะต้องทำ งานด้วยระบบปฏิบัติการลีนุกซ์ และมีการใช้คอมไพล์เลอร์ที่สามารถที่จะแปลงไฟล์จากระบบ x86 ไปเป็น ARM ได้ รวมทั้งมีเครื่องมือในการโหลดข้อมูลลงไปทำงานบนบอร์ดผ่านทางพอร์ตขนานของเครื่องพีซีได้

3.4.1 ระบบปฏิบัติการที่น่าสนใจสำหรับระบบฝังตัว

3.4.1.1 uClinux

พัฒนามาจากลีนุกซ์เคอร์เนล 2.0 มีจุดประสงค์เพื่อใช้ในงานไมโครคอนโทรเลอร์โดยไม่ต้องอาศัย การจัดการของหน่วยความจำ (MMUs) มีความสามารถในการทำงานแบบ multitasking นอกจากนั้นไฟล์ไบนารี่และซอร์สโค้ดสำหรับเคอร์เนล ยังถูกเขียนใหม่ให้มีขนาดเล็กลงและลดจำนวนโค้ดที่ใช้ให้น้อยที่สุด มีความเสถียร สนับสนุนการทำงานด้านเน็ตเวิร์ก และไฟล์ระบบ มีขนาดของเคอร์เนล ไม่เกิน 512 กิโลไบท์ และเมื่อรวมส่วนประกอบที่เป็นเครื่องมืออื่นๆ ไปด้วยแล้วจะไม่เกิน 900 กิโลไบท์

คุณสมบัติของ uClinux

- MMU non-dependent
- สนับสนุนโปรโตคอล TCP/IP, SLIP และ PPP
- Embedded Web servers
- Embedded uC library collection uClibc, uClibns, uCcrypt
- ไฟล์ระบบที่สนับสนุน : NFS / JFFS / bFLAT EXT2FS / MINIX / SMB
- fully customizable

แพลตฟอร์มที่รองรับ

- Motorola
- Atmel
- Samsung
- Conexant
- Analog Devices
- Hitachi
- SPARC
- Intel
- MIPS 4K
- และอื่น ๆ

ขนาดของโค้ด

- เคอร์เนลโดยปกติมีขนาดประมาณ 512 กิโลไบท์
- เมื่อรวมองค์ประกอบทั้งหมดจะมีขนาดประมาณ 1 เมกกะไบท์

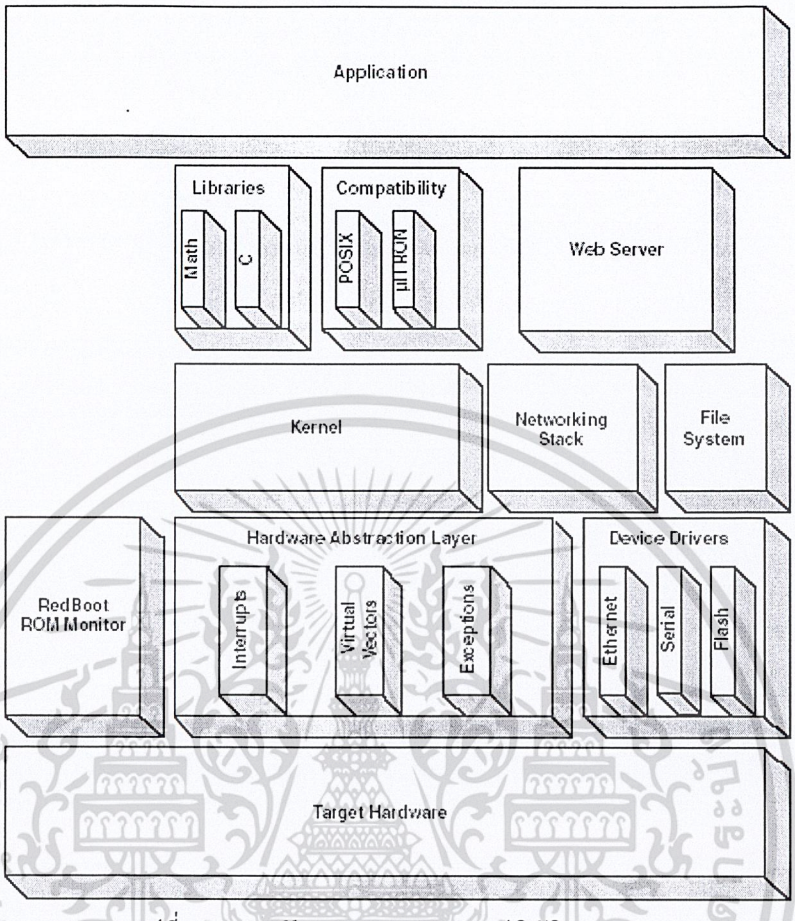
3.4.1.2 eCos (Embedded Configurable Operating System)

eCos เป็นระบบปฏิบัติการพัฒนาเป็นครั้งแรกในปี 1997 เป้าคือเพื่อพัฒนาระบบฝังตัวประสิทธิภาพสูง ต้องการทรัพยากรที่น้อยลง นอกจากนี้ยังเป็นระบบปฏิบัติการที่มีการเผยแพร่แจกจ่ายซอร์สโค้ดฟรี ถูกออกแบบให้สามารถรองรับแอพพลิเคชั่นต่างๆ ตามที่ผู้พัฒนาต้องการต้องการไม่ว่าจะเป็นด้านการติดต่อสื่อสาร การควบคุมเครื่องจักร และแอพพลิเคชั่นอื่นๆ ในระดับสูง

eCos เป็นระบบปฏิบัติการที่เป็นเรียลไทม์ (Real – Time Operathing System)มีความเหมาะสมกับระบบฝังตัวระดับลึกที่ระบบปฏิบัติการลินุกซ์ทั่วไปไม่สามารถทำได้ โดยทั่วไประบบปฏิบัติการลินุกซ์ จะมีขนาดของเคอร์เนล อยู่ที่ประมาณ 500 กิโลไบท์ ถึง 1.5 เมกะไบท์ แต่ eCos จะมีขนาดของเคอร์เนลประมาณ 10 - 100 กิโลไบท์ และอาจขึ้นอยู่กับความต้องการของระบบด้วย

สถาปัตยกรรม

eCos ถูกออกแบบมาให้สนับสนุนกับแอพพลิเคชั่นที่มีการทำงานแบบเรียลไทม์(real-time) เหมาะที่จะนำมาพัฒนาแอพพลิเคชั่นของระบบฝังตัว ที่ระบบปฏิบัติการลินุกซ์ทั่วไปไม่สามารถทำได้ โดยทั่วไประบบปฏิบัติการลินุกซ์ จะมีขนาดของเคอร์เนล อยู่ที่ประมาณ 500 กิโลไบท์ ถึง 1.5 เมกะไบท์ แต่ eCos จะมีขนาดของเคอร์เนลประมาณ 10 - 100 กิโลไบท์ และอาจขึ้นอยู่กับความต้องการของระบบด้วย รองรับกับไดรเวอร์ของอุปกรณ์ ,ไฟล์ระบบ ,โปรโตคอล TCP/IP การจัดการหน่วยความจำ , exception handling , ภาษาซีและไลบรารี , และอื่น ๆ eCos มีเครื่องมือที่ใช้ในการ



รูปที่ 3.17 สถาปัตยกรรมของระบบปฏิบัติการ eCos

พัฒนาแอพพลิเคชันระบบฝังตัว โดยได้รับการสนับสนุนตัวคอมไพเลอร์ , แอสแซมเบลอร์ , ลิ้งเกอร์ , ดีบักเกอร์ และซิมมูเลเตอร์ จาก กนู(GNU) ดังนี้

- สนับสนุนไดรว์เวอร์อีเทอร์เน็ต USB , flash , serial และอื่นๆ
- สนับสนุนการทำงาน TCP/IP แบบ SNMP
- สนับสนุนภาษาซีและไลบรารี่
- รองรับมาตรฐาน POSIX EL/XI ระดับ 1
- สนับสนุน µITRON 3.02
- ได้รับการสนับสนุนเครื่องมือจากกนู(GNU)
- สนับสนุนดีบักมอนิเตอร์ RedBoot
- มี Hardware Abstraction Layer (HAL) ทำให้สามารถทำงานได้กับโปรเซสเซอร์หลายแพลตฟอร์ม
- ส่วนประกอบของเคอร์เนลที่เป็นเรียลไทม์ :

- Interrupt handling
- Exception handling
- Choice of schedulers
- Thread support
- Rich set of synchronization primitives
- Timers, counters, and alarms
- Choice of memory allocators
- Debug and instrumentation support

สนับสนุนระบบปฏิบัติการโฮสต์

- Red Hat Linux
- Sun Solaris and ระบบ UNIX อื่นๆ
- Microsoft Windows

แพลตฟอร์มที่สนับสนุน

- ARM
- Hitachi SH3, SH4
- MIPS
- NEC V850
- Panasonic AM3x
- PowerPC
- SPARC
- x86
- SuperH,Matsushita AM3x,IA32,และอื่นๆ

3.4.2 เหตุผลที่เลือกใช้ระบบปฏิบัติการ eCos

จากที่ได้ลองศึกษาระบบปฏิบัติการที่ผ่านมาของโครงงานนี้ มีระบบปฏิบัติการอยู่ 2 ตัว ที่นำมาพิจารณาเลือกใช่ คือ eCos และ uClinux โดยระบบปฏิบัติการทั้ง 2 ตัว นี้มีคุณสมบัติที่แตกต่างกันอยู่ ดังนั้นการพิจารณาเลือกระบบปฏิบัติการที่จะนำมาใช้ ต้องมีความเหมาะสมและเข้ากันได้กับโครงงานนี้

จากที่ได้ลองพิจารณา พบว่าระบบปฏิบัติการ eCos มีข้อได้เปรียบกว่า uClinux อยู่หลายด้าน เนื่องจาก eCos เป็นระบบปฏิบัติการที่พัฒนามาเพื่อระบบฝังตัวโดย เฉพาะ มีคุณสมบัติที่ดีกว่าเหมาะที่จะนำมาใช้ในการพัฒนาระบบฝังตัว และดีกว่า uClinux ที่ใช้กันอยู่ในปัจจุบันนี้ uClinux พัฒนามาจากลินุกซ์และ ทำให้มีคุณสมบัติที่เหมาะสมน้อยกว่า เมื่อเทียบกับ eCos เนื่องจาก uClinux ไม่ได้พัฒนามาใช้สำหรับระบบฝังตัวโดยเฉพาะ

ตารางที่ 3.5 แสดงการเปรียบเทียบระหว่างระบบปฏิบัติการ eCos และ uClinux

ข้อพิจารณา	eCos	uClinux
แจกจ่ายฟรี เผยแพร่ซอร์สโค้ด	ใช่	ใช่
ทำงานบนไมโครโพรเซสเซอร์แบบ 32 บิต	ใช่	ใช่
สนับสนุนการทำงานด้านเน็ตเวิร์ก	ใช่	ใช่
แหล่งข้อมูล	มีมาก	มีน้อย
การทำงานแบบ Real - time	ใช่	ไม่ใช่
ขนาดของเคอร์เนล (กิโลไบท์)	10 - 100	600

จากตารางจะเห็นว่า ขนาดของเคอร์เนลที่พัฒนามาจาก uClinux มีขนาดอย่างน้อยประมาณ 600 กิโลไบท์ ซึ่งไม่ เหมาะสมที่นำมาใช้กับระบบฝังตัว แต่ eCos จะมีขนาดแค่เพียง 10 - 100 กิโลไบท์เท่านั้น โดยจะขึ้นอยู่ตามความต้องการของระบบ นอกจากนั้น uClinux ยังไม่มีคุณสมบัติทางด้าน "Real – time" ถึงแม้ว่าจะนำมาใช้งานแบบเดียวกัน แต่สำหรับ eCos จะมีคุณสมบัติการทำงานแบบ "Real – time" พร้อมอยู่ในตัวแล้ว นอกจากนั้น eCos มีเครื่องมือที่ใช้ในการพัฒนาระบบฝังตัวที่ดีกว่าของ uClinux

จากที่ได้ลองพิจารณาระบบปฏิบัติการ eCos และ uClinux จึงทำให้ได้ข้อสรุปว่า ระบบปฏิบัติการ eCos น่าสนใจที่จะนำมาใช้ในการพัฒนากับโครงงานนี้มากที่สุด

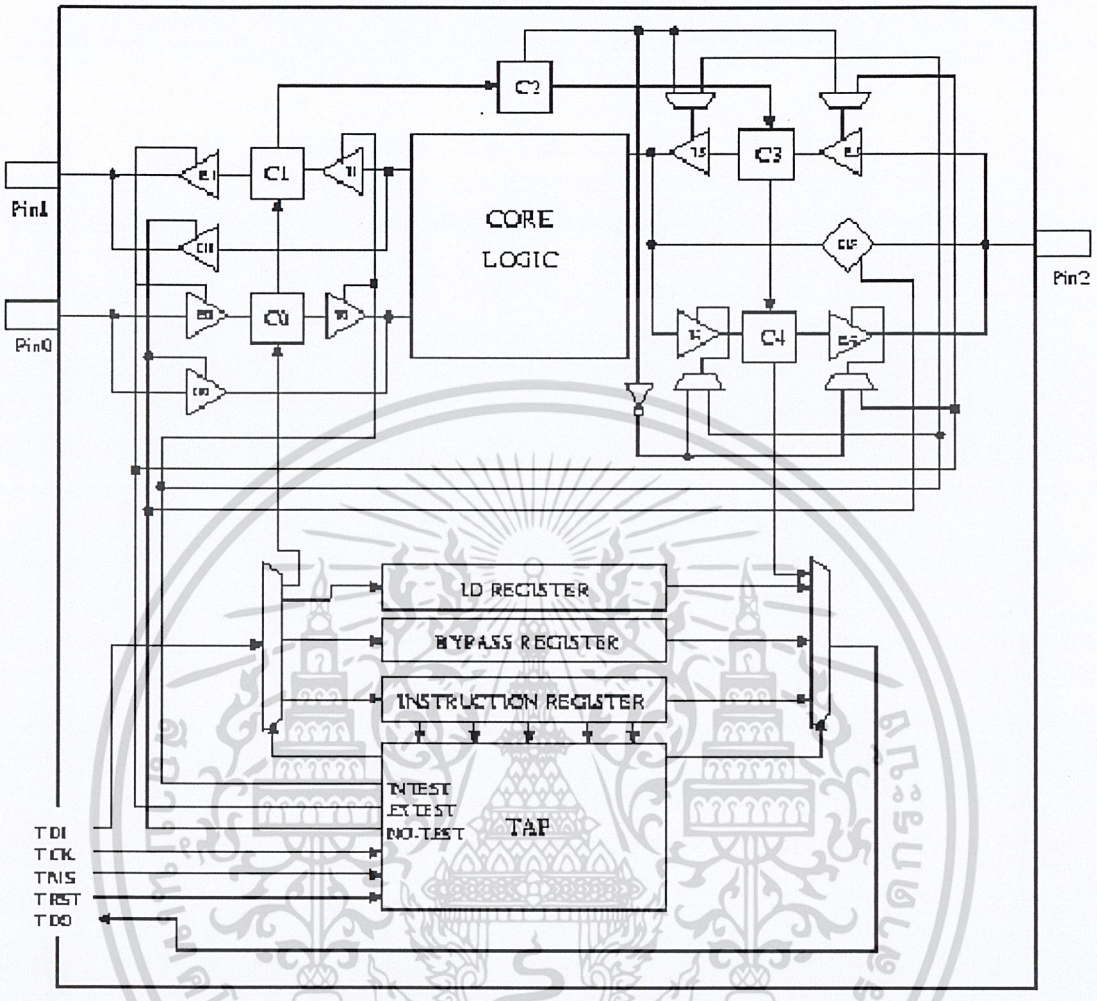
3.4.3 เครื่องมือที่ใช้ในการพัฒนา

การพัฒนาโปรแกรมทั้งหมดของโครงงานนี้ใช้ภาษาซี ซึ่งเครื่องมือที่เลือกใช้ในการพัฒนาโปรแกรมบนระบบปฏิบัติการลินุกซ์ เป็นเครื่องมือที่อยู่ในชุด GNU ของ Free Software Foundation คือ binutils gcc และ gdb ซึ่งเครื่องมือทั้งคู่สามารถที่จะใช้งานกับโพรเซสเซอร์ในตระกูล ARM7TDMI ได้อยู่แล้ว ทำให้สามารถใช้เป็นเครื่องมือหลักในการพัฒนาโปรแกรมรวมทั้งไดร์เวอร์ให้ทำงานบนบอร์ดควบคุมได้

3.4.3.1 JTAG Boundary Scan Interface

ปัจจุบันอุปกรณ์อิเล็คทรอนิคส์มีขนาดเล็กลงมาก ทำให้PCB (Printed – Circuit Boards)ที่นำมาใช้พัฒนามีขนาดเล็กลงด้วย และส่งผลให้เกิดวิธีการพัฒนาบอร์ดที่เป็นลักษณะ On – Board Programming (OBP) โดยใช้เทคนิคที่เรียกว่า Joint Test Action Group Test Access Port (JTAG TAP)

JTAG TAP เป็นมาตรฐานของ IEEE 1149.1 ซึ่งกล่าวอ้างถึง การทดสอบการทำงานของพอร์ตและสถาปัตยกรรม Boundary – Scan



รูปที่ 3.18 สถาปัตยกรรมของ JTAG Boundary Scan

การทำงาน

สัญญาณทั้งหมดที่อยู่ระหว่าง core logic และเส้นทางเดินของสัญญาณขา ของตัวชิพซึ่งเรียกว่า *"Boundary Scan Register"* (*BSR*) จากรูปแสดง ด้วยเซลล์ C1 , C2 , C3 , C4 และมีเส้นทางต่อจากขาไปยัง core logic ในการทำงานในโหมด external-test mode ขาจะไม่สามารถต่อถึงเข้ากับ core logic ได้ จากรูปมี ขา Pin1 กับขา Pin2 เป็นเอาท์พุต และจะทำการอ่านค่าแล้วทำการ latch โดยขาอินพุตคือ ขา Pin0 กับขา Pin2 แต่ถ้าเป็นการทำงานใน internal-test mode ขาจะไม่ต่อถึงกับ core logic แต่ core logic จะเป็นตัวรับสัญญาณอินพุต และจะทำการอ่านค่าและ latch สัญญาณเอาท์พุตที่ core logic

จากรูป สมมุติให้ JTAG ทำงานในโหมด external-test mode โดย C0 เป็น BSR เซลล์ แล้วทำการอ่านค่าที่ขาอินพุต Pin0 , C1 เป็น BSR เซลล์เอาท์พุตที่ขา Pin1 , C2 ไม่มีการตอบสนองการทำงานกับขาใดๆ แต่มันจะ "enable" BSR เซลล์ ซึ่งควบคุมการทำงานแบบทางเดียว จากการทำงานที่เป็นสองทางที่ขา Pin2 , C3

เป็นอินพุต BSR แบบสองทางที่ต่อกับขา Pin2 และ C4 เป็นเอาท์พุต BSRเซลล์ แบบสองทางที่ต่อกับขา Pin2 เราสามารถสรุปการทำงานของ BSRเซลล์ได้ 3 อย่าง ดังนี้

- Input Cells คือ C0 , C3 ซึ่งจะทำงานสัมพันธ์กับขาที่ต่ออยู่ในการ Captureเมื่อต่อกับ JTAG ขณะทำงานใน external test mode
- Output Cells คือ C1 , C4 ซึ่งจะทำงานสัมพันธ์กับขาที่ต่ออยู่ในการให้ค่าออกมา ขณะทำงานใน external test mode
- Enable Cells คือ C2 จะไม่ขึ้นอยู่กับขาใดๆ แต่จะควบคุมให้มีการไหลของข้อมูลเพียงทางเดียวหรือว่าสองทาง หรือจะกำหนดให้ขาสัญญาณเป็นอินพุตหรือเอาท์พุต

ส่วนเกท E0 , E3 และ E4 การทำงานจะถูกควบคุมโดย TAP (และอาจควบคุมด้วยการ “enable” เซลล์เหมือนกับ C2) การอ่านค่าหรือนำค่าไปใช้ จะเป็นไปตามลำดับของสถานะ(State)อินพุตหรือเอาท์พุต , เซลล์ , จาก ขาของตัวชิพ สถานะ(State)ในการอ่านค่า (capture) หรือ การนำค่าไปใช้ (apply) ซึ่งจะมีช่วงเปลี่ยนการทำงานที่แน่นอนในตาม TAP state-machine ซึ่งอาจจะเป็นตอนที่ IR (instruction register) มีการโหลดค่าก่อนแล้วนำมาเก็บ เป็น opcode (EXTEST)

เกท I0,I3และ I4 การทำงานจะถูกควบคุมโดย TAP (และอาจควบคุมด้วยการ “enable” เซลล์ เหมือนกับ C2) การอ่านค่าหรือนำค่าไปใช้ จะเป็นไปตามลำดับของสถานะ(State)อินพุตหรือเอาท์พุต , เซลล์ , จากสัญญาณภายในลอจิกของตัวชิพสถานะ(state)ในการอ่านค่า(capture) หรือ การนำค่าไปใช้ (apply) ซึ่งจะมีช่วงเปลี่ยนการทำงานที่แน่นอนใน TAP state-machine state-machine ซึ่งอาจจะเป็นตอนที่ IR (instruction register) มีการโหลดค่าก่อน แล้วนำมาเก็บ เป็น opcode (INTEST).

เกท N0 , N1 และ N3 จะทำงานเมื่อระบบอยู่โหมดปกติ (normal-operation mode) เท่านั้น และขาของชิปต่อถึงกับ core-logic ค่ารีจิสเตอร์ของ BSR สามารถเขียนและอ่านได้บิตต่อบิต ตามลำดับ โดยใช้สัญญาณ TDI และ TDO ของJTAG ในความเป็นจริงในการอ่านหรือเขียน BSR ในช่วงเวลาเดียวกันของค่าใหม่จากขาสัญญาณ TDI ขณะก่อนที่มีการ shift ค่าออกจากขาสัญญาณ TDO จะใช้เทคนิคเดียวกันในการอ่านและเขียนค่าของ JTAG registers ตัวอื่นๆ ด้วย โดยจะใช้ TAP controller เป็นตัวต่อระหว่างขา TDI , TDO กับตัว BSR

สัญญาณที่ใช้ในการติดต่อ

JTAG ประกอบด้วยขาสัญญาณที่ใช้สำหรับควบคุมในการทำงาน 5 ขา คือ

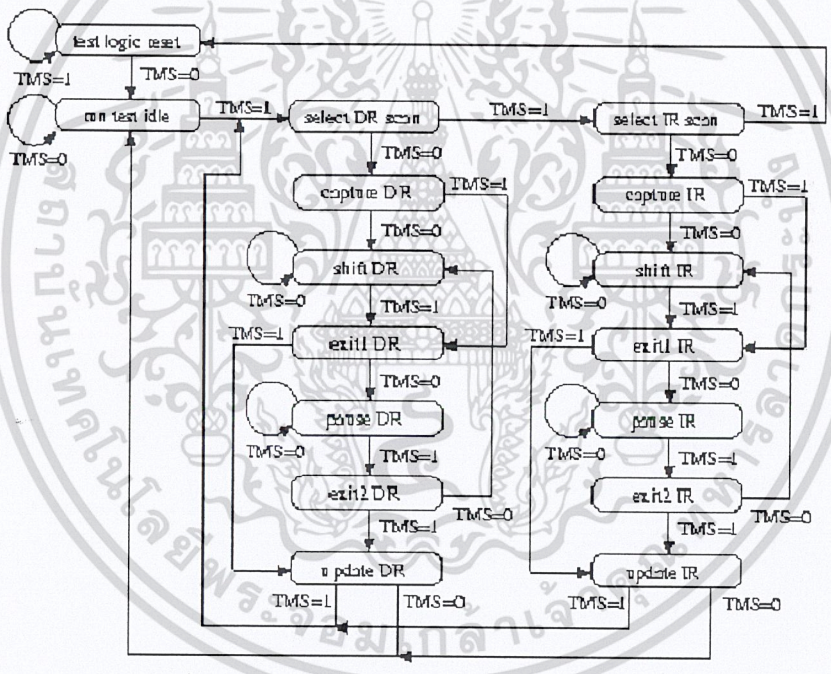
- TRSC - Test Reset Input (active low) เป็นสัญญาณอะซิงโครนัสของ TAP controller ที่ใช้ในช่วงเริ่มต้นการติดต่อและยกเลิกการติดต่อ
- TCK - Test Clock Input เป็นสัญญาณคล็อคจากภายนอกไม่ได้ขึ้นกับระบบ
- TMS - Test Mode Select Input ใช้ควบคุมการเปลี่ยนโหมดการทำงานของ state machine

- TDI - Test Data Input เป็นขาที่ใช้ชิพข้อมูลไปยัง JTAG register หรือ Boundary Scan chain (Boundary Scan Register, หรือ other data registers อื่นๆ) โดยข้อมูลจะผ่านทางขา TDI
- TDO - Test Data Output เป็นขาที่ใช้ชิพข้อมูลออกจาก JTAG register หรือ Boundary Scan chain

โดยปกติ การทดสอบการทำงานของบอร์ดที่ประกอบขึ้นด้วยจำนวนชิพหลายตัว ที่สนับสนุนการทำงานแบบ JTAG จะต่อสัญญาณ TRST ,TCK และ TMS แบบขนานกับชิพทุกตัว และต่อขาสัญญาณ TDO จากชิพตัวหนึ่งไปยัง TDI อีกตัวในลักษณะลูป

TAP controller

เป็นตัวที่ใช้สำหรับควบคุมการติดต่อขาสัญญาณของ JTAG กับ BSR ของชิพ ซึ่งมีขั้นตอนการทำงานตาม state-machine ดังรูปข้างล่าง โดยการเปลี่ยนของสถานะ(State) จะขึ้นกับขาสัญญาณ TMS



รูปที่ 3.19 ภาพแสดงไดอะแกรมของการเปลี่ยนสถานะ

จากภาพไดอะแกรมข้างบน จะเห็นว่าการออกจากสถานะ(State)มีสองทาง การเปลี่ยนของสถานะ(State) จะขึ้นอยู่กับสัญญาณ TMS เพียงสัญญาณเดียว ในไดอะแกรมจะมีสองทางหลักในการเปลี่ยนการทำงานของ (State)ซึ่งเป็นการควบคุมการทำงานของ Data Register (ID regidter , Bypass register , BSR register) และ Instruction register โดย Data Register จะมีการทำงานทุกครั้งเมื่อทางเดินของ Data Register ถูกเลือกด้วยการโหลดค่าลงใน Instruction Register

การเปลี่ยนเส้นทางตามลำดับของโปรแกรมข้างล่าง (โดยปกติเรียกว่า IR path) เป็นการโหลดค่าใหม่ใน Instruction Register และ อ่านค่าเดิมออก

*-> test logic reset
--> run test idle
--> select dr scan
--> select ir scan
--> capture ir
--> shift ir --> ... n times ... --> shift ir
--> exit1 ir
--> update ir
--> run test idle ->*

ค่าใหม่จะถูก shift ลงใน Instruction Register จากขาสัญญาณ TDI เป็นจำนวน 1 bit ซึ่งขึ้นอยู่กับสถานะ "shift ir" ทั้งหมด ส่วนค่าเก่าของ Instruction Register ถูก shift ออกทางขาสัญญาณ TDO เป็นจำนวน 1 บิต ซึ่งเป็นการออกจากสถานะ "shift ir"

การเปลี่ยนเส้นทางตามลำดับของโปรแกรมข้างล่าง (โดยปกติเรียกว่า DR path) เป็นการโหลดค่าใหม่ใน Data Register ที่ได้เลือกไว้ในขณะนั้นแล้ว อ่านค่าเดิมออกไป

*-> test logic reset
--> run test idle
--> select dr scan
--> capture dr
--> shift dr --> ... n times ... --> shift dr
--> exit1 dr
--> update dr
--> run test idle ->*

ค่าใหม่ถูก shift ลงใน Data Register ที่เลือกไว้ในขณะนั้น จากขาสัญญาณ TDI เป็นจำนวนหนึ่งบิต ซึ่งการทำงานขึ้นอยู่กับสถานะ "shift dr" ทั้งหมด ส่วนค่าเก่าจะออกจาก Data Register ที่เลือกไว้ในขณะนั้นจะถูก shift ออกจากขาสัญญาณ TDO เป็นจำนวนหนึ่งบิต ซึ่งเป็นการออกจากสถานะ "shift dr" และค่าของ Instruction Register จะถูก shift ออกไป การ captured ขึ้นอยู่กับสถานะ "capture ir" ค่าของ Data Register ที่เลือกไว้ในขณะนั้น จะถูก shift ออกไป การ captured ขึ้นอยู่กับสถานะ "capture dr" ค่าใหม่จะถูก shift ลงใน

Instruction Register ซึ่งเป็นการนำค่ามาใช้ (และคำสั่งจะเป็นผล) ซึ่งขึ้นอยู่กับสถานะ "update ir" ค่าของ Data Register ที่เลือกไว้ในขณะนั้นจะถูก นำไปใช้ (คือ การส่งข้อมูลออกที่ขาเอาท์พุต ในกรณีของ BSR) ซึ่งขึ้นอยู่กับสถานะ "update dr"

Data register

เมื่อมีการพูดถึงวิธีการทดสอบการทำงาน จะเป็นผลของการเก็บค่าของ instruction register ซึ่งเป็นการเลือกระหว่างความแตกต่าง ของ data registers ที่ทำงานในช่วง "dr path" ซึ่งต่อไปนี้เป็นการแสดง data-registers ที่ใช้ใน JTAG

- *Device ID* register (*IDR*) reads-out เป็นการกำหนดหมายเลขให้กับชิปที่นำมาเชื่อมต่อ
- *Bypass* register (*BR*) 1-cell pass-through register ซึ่งเป็นการต่อ TDI ด้วย TDO กับหน่วงเวลาไป 1clock เพื่อให้ง่ายในการติดต่อกับอุปกรณ์อื่นที่อยู่บนแผ่น PCB เดียวกัน
- *Boundary Scan* register (*BSR*) ซึ่งอธิบายรายละเอียดเกี่ยวกับสัญญาณที่อยู่ระหว่าง core-logic กับขา

Instructions

โดยปกติของการทดสอบด้วย JTAG เป็นการเข้าสู่คำสั่งซึ่ง เป็นลักษณะชนิดของการทดสอบที่จะกระทำต่อไป และ Data Register ถูกใช้ในช่วงที่มีการทดสอบ ค่าภายใน Instruction Register (จากวิธีการทำงานของ TAP ตลอดถึงการใช้ "ir path") และการใช้ Data Register ในการทดสอบ (จากวิธีการทำงานของ TAP ตลอดถึงการใช้ "dr path" เพียงหนึ่งตัวหรือมากกว่า)

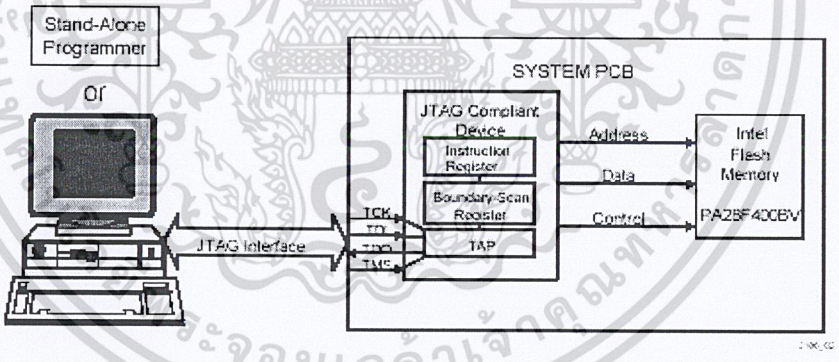
มีคำสั่งที่เป็นคำสั่งเฉพาะ(Private instructions) และ คำสั่งทั่วไป(Public instructions) โดยคำสั่งทั่วไปได้มาจากเอกสารของโรงงานผู้ผลิตชิพ และใช้กันโดยทั่วไป ส่วนคำสั่งเฉพาะจะไม่เปิดเผย ซึ่งมาตรฐานของ IEEE – 1149 เป็นการกำหนดการใช้ชุดคำสั่งทั่วไปที่แสดงใน JTAG ทั้งหมด โดยข้างล่างนี้เป็นชุดคำสั่งดังกล่าว

- *BYPASS* : นี้เป็นขาสัญญาณ TDI และ TDO ที่ต่อผ่านเข้าไปในรีจิสเตอร์เพียง 1 บิต (single-bit pass-through register) นี้เป็นคำสั่งที่ยอมให้ต่อเข้ากับอุปกรณ์ตัวอื่นในลักษณะที่เหมือนกับลูป
- *EXTEST* : นี้เป็นคำสั่งเกี่ยวกับ boundary scan register (BSR) ที่ต่อระหว่าง สัญญาณ TDI และ TDO มีการ capture ค่าที่ขาของชิพ โดย BSR เซลล์ ซึ่งทั้งหมดจะเป็นการทำงานในสถานะ "capture dr" (ดูการเปลี่ยน สถานะในdiagram) ค่าที่เก็บใน BSR register จะถูก shift ออกไป ผ่านทางขาสัญญาณ TDO แล้วจะออกจากสถานะ "shift dr" ค่าที่เก็บใน BSR register (ที่ได้จาการ capture) จะถูก shift ออกไป แล้วค่าใหม่จะถูก shift เข้ามา ซึ่งการทำงานทั้งหมดจะอยู่ในสถานะ " shift dr" ค่าใหม่ที่เก็บใน BSR และถูกนำมาใช้ โดยเป็นสัญญาณที่ขาของชิพ และจะทำงานในช่วงสถานะ "update dr"
- *IDCODE* : โดย ID register ที่ต่อระหว่าง TDI และ TDO ทั้งหมดจะเป็นการทำงานในสถานะ "capture dr" เป็น Device ID Code (เป็นการเชื่อมต่ออุปกรณ์ โดยมีการกำหนดหมายเลขจากโรงงานผู้ผลิต)

- *INTEST* : นี้เป็นคำสั่งเกี่ยวกับ boundary scan register (BSR) ที่ต่อระหว่าง สัญญาณ TDI และ TDO สัญญาณภายใน core-logic ของชิพ จะถูกcapture โดย BSR เซลล์ ซึ่งการทำงานทั้งหมดจะอยู่ในสถานะ "capture dr" (ดูการเปลี่ยน สถานะในdiagram) ค่าที่เก็บใน BSR register จะถูก shifted ออกผ่านทางขาสัญญาณ TDO ซึ่งจะออกจากสถานะ "shift dr" ค่าที่เก็บใน BSR register (ที่ได้จาการ capture) **จะถูก** shift ออกไป แล้วค่าใหม่จะถูก shift เข้ามา ซึ่งการทำงานทั้งหมดจะอยู่ในสถานะ " shift dr" ค่าใหม่ที่เก็บใน BSR และถูกนำมาใช้ เป็นสัญญาณให้กับ core-logic ในช่วงที่ทำงานในสถานะ "update dr"

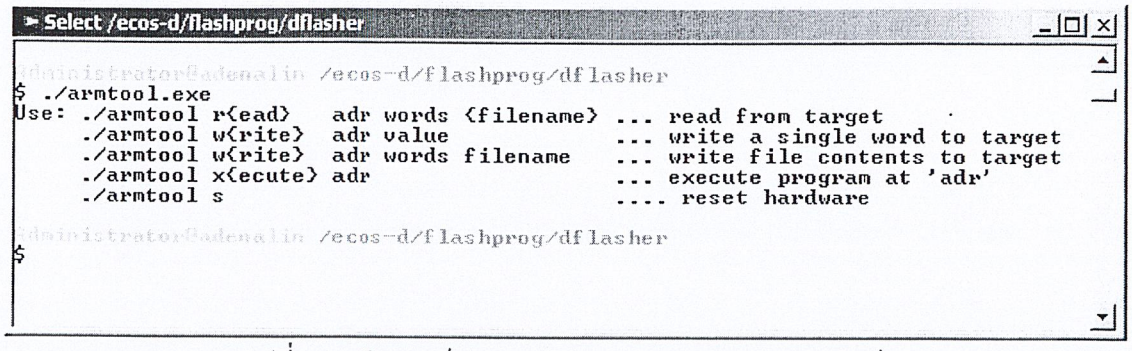
3.4.3.2 On-Board Programming

สิ่งจำเป็นอันดับแรกก่อนที่จะนำโปรแกรมที่พัฒนาไปทำงานยังที่แพลตฟอร์มที่ได้ออกแบบขึ้นจำเป็นต้องมีโปรแกรมที่มีการติดต่อระหว่างเครื่องโฮส(Host)และเครื่องแพลตฟอร์ม(Target) หรือบอร์ดควบคุม(On-Board Programming) ขั้นตอนนี้ได้ทำการพัฒนาเครื่องมือที่ใช้ในการโปรแกรมหน่วยความจำแฟลซ (Flasher) ซึ่งพัฒนาต่อจากโปรแกรม “Armtool” โดยจะส่งข้อมูลผ่าน พอร์ทขนานทำการโปรแกรมผ่านพอร์ท JTAG ของไมโครคอนโทรเลอร์ ดังรูปข้างล่างแสดงการทำงานของ On-Board Programming



รูปที่ 3.20 การทำ On-Board Pramming

โปรแกรม “Armtool” จะเป็นมีอินเตอร์เฟสเป็นคอมมานด์ไลน์ทำงานได้ทั้งในระบบปฏิบัติการวินโดว์ Windows + Cygwin(Cygwin= เป็นโปรแกรมที่จำลองสภาแวดล้อมของระบบปฏิบัติการยูนิกส์บนวินโดว์) และระบบปฏิบัติการลินุกซ์ ดังรูป



รูปที่ 3.21 อินเตอร์เฟสของโปรแกรม Armtool บนวินโดว์

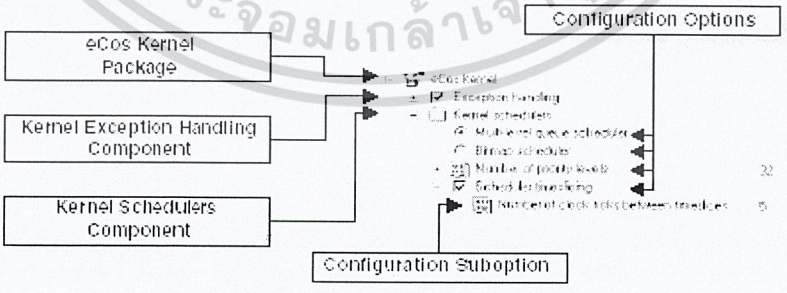
โปรแกรม "Armtool" สามารถที่จะทำงานเป็นตัวโหลด หรือ อ่านข้อมูลของหน่วยความจำ ยกเว้นหน่วยความจำแฟลซ ซึ่งต้องเขียนโปรแกรมเพิ่มขึ้นมา คำสั่งที่เป็นประโยชน์อีกคำสั่งคือ คำสั่ง เอกซิคิว เพื่อเป็นการสั่งให้ ARM7TDMI ทำงานตามที่ระบุในแอดเดรส สำหรับการโปรแกรมหน่วยความจำแฟลซจะใช้โปรแกรม Flasher ซึ่งอาศัยการทำงานของ Armtool และ Flaher code ที่เขียนขึ้น

3.4.3.3 eCos Configuration tool

eCos Configuration tool เป็นเครื่องมือที่ใช้ในการพัฒนาระบบปฏิบัติการอีคอส(eCos) ซึ่งเป็นตัวที่จะใช้ในการปรับ แต่งแก้ไขซอร์สโค้ด ก่อนที่จะทำการรวบรวมเป็นส่วนประกอบหรือคอมโพเนนท์ของไฟล์ตามที่ต้องการ โดยซอร์สโค้ดดังกล่าวจะเก็บรวบรวมไว้ในคอมโพเนนท์ที่ชื่อว่า "*component repository*" คอมไพล์เพื่อเก็บเป็นไลบรารี สำหรับนำไปลิงค์รวมกับ โปรแกรมแอพพลิเคชั่นที่เขียนขึ้น

วิธีการของ eCos Configuration

ในระบบฝังตัวต้องมีขนาดเล็ก เร็ว ถูก สามารถทำได้โดยการควบคุมให้มีซอฟท์แวร์ที่จำเป็นอยู่ในระบบเท่านั้น ซึ่งจะแตกต่างจากวิธีการที่กำหนดให้มีคอนโพเนนท์เข้าไปรวมกับแอพพลิชั่น

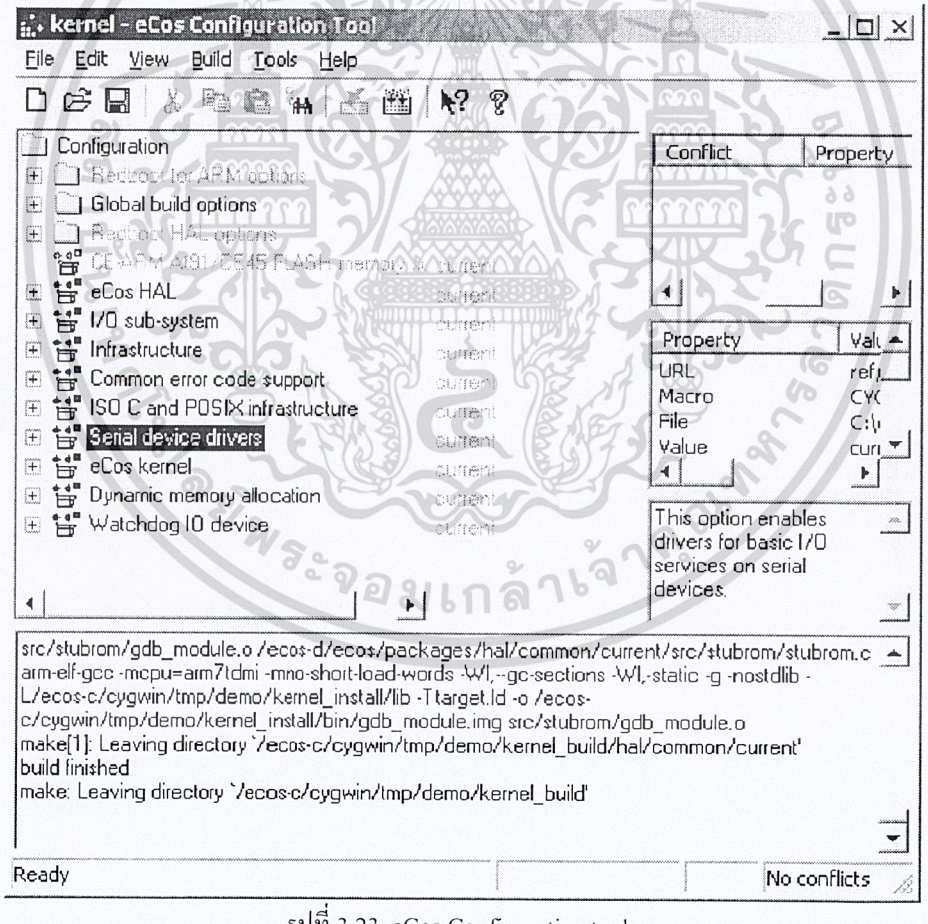


รูปที่ 3.22 การ configuration eCos

การลดขนาดแอพพลิเคชั่นระบบฝังตัว eCos จะใช้งาน Compile-time control โดยให้นักพัฒนาระบบจัดการกับคอมโพเนนท์ และเรียกใช้คอมโพเนนท์นั้นๆ สำหรับสร้างแอพพลิเคชั่นตามที่ตั้งใจ ซึ่งจะทำให้ลด

ขนาดของโค้ดได้ดีที่สุด เพราะว่าเป็นการควบคุมที่ระดับ Statement ใน source code มากกว่าที่จะไปควบคุมที่ Fucntion หรือ object การใช้งาน Compile-time จึงเหมาะสำหรับการพัฒนาระบบฝังตัว นอกจากโค้ดจะมีขนาดเล็กแล้วยังมีข้อดีที่สำคัญคือ

- แอพพลิเคชั่นทำงานได้เร็วกว่าเพราะว่าไม่มีการตรวจสอบตัวแปรในระหว่าง run-time เพื่อหาว่าต้องทำอะไร
- โปรแกรมมีการตอบสนองมากกว่า และ ส่วนที่ซ้ำซ้อนจะถูก reused ซึ่งเป้าหมายในการสร้างระบบตรวจสอบสำหรับ real-time device
- โค้ดที่ง่ายกว่าจะถูกสร้างขึ้น ทำให้ตรวจสอบ และ ทดสอบทำได้ง่ายขึ้น
- ทำให้ลดค่าใช้จ่ายเพราะว่าทรัพยากรที่ใช้จะถูก optimize และการใช้งาน processor cycle อย่างมีประสิทธิภาพ ด้วยเหตุนี้จึงต้องการ hardware ที่มีราคาถูกกว่าในการออกแบบ



รูปที่ 3.23 eCos Configuration tool

3.5 วิธีการที่จะนำโปรแกรมลงไปทำงานบนบอร์ดทำได้ 2 วิธี ดังนี้

3.5.1 การ Burning

คือ การนำเอาโปรแกรมและ Rom Filesystem image ที่ผ่านการคอมไพล์เรียบร้อยแล้วไปใส่ในหน่วยความจำประเภทคงอยู่ (Nonvolatile Memory) เช่น หน่วยความจำแฟลช (Flash Memory)

3.5.2 การ Downloading

คือการนำเอาโปรแกรมและ Rom Filesystem image ที่ผ่านการคอมไพล์เรียบร้อยแล้วโหลดผ่านทางสายสื่อสารไปใส่ในหน่วยความจำ เช่น พอร์ตขนาน หรือ พอร์ตอนุกรม เป็นต้น โดยการเรียกให้ทำงานให้เริ่มต้นทำงานในตำแหน่งที่ได้โหลดลงไป

ในวิธีแรกนั้นเหมาะสมกับระบบที่ได้มีการพัฒนาเรียบร้อยแล้วและพร้อมที่จะนำไปใช้งาน ดังนั้นจึงได้เลือกใช้วิธีที่สองแทน เพราะในขั้นตอนของการทดลองและทดสอบการทำงานจะต้องมีการแก้ไขอยู่เป็นประจำ รวมทั้งต้องมีการดีบักดูข้อผิดพลาดต่างๆ อยู่เสมอ ถ้ามีข้อมูลอยู่ในแรมอยู่แล้วจะทำให้สามารถที่จะดึงเอาข้อมูลที่อยู่ในแรมออกมาดูได้อย่างสะดวกยิ่งขึ้น ส่วนการโหลดอิมเมจลงไปทำงานบนบอร์ดนั้น ได้เลือกใช้การโหดผ่านทางพอร์ตขนานของเครื่องพีซี ซึ่งเชื่อมต่อกับพอร์ต JTAG ของบอร์ด เพราะได้มีการพัฒนาทั้งในส่วนของฮาร์ดแวร์ และซอร์ฟแวร์ ที่ใช้การติต่อกับ JTAG เรียบร้อยอยู่แล้ว

3.6 การพัฒนา eCos สำหรับบอร์ดควบคุม

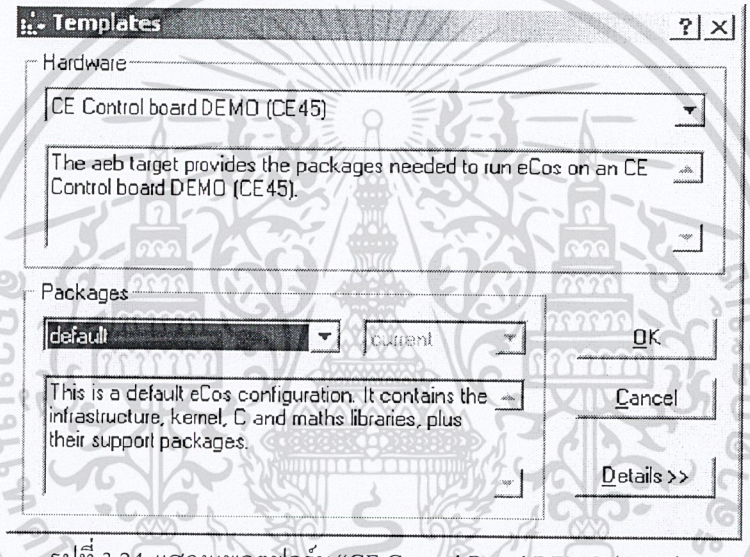
สิ่งที่ต้องทำเพื่อให้ eCos สามารถทำงานได้กับบอร์ดควบคุมที่ออกแบบขึ้นโดยการปรับปรุงในส่วนของ HAL(Hardware Abstraction Layer) โดยการเพิ่มแพลตฟอร์มใหม่เข้าไป ดังนี้

- เพิ่มข้อมูลเกี่ยวกับบอร์ดควบคุมเข้าไปใน Database ให้กับ eCos configuration tool
- เพิ่มรายละเอียดของ Memory Layout
- เพิ่มรายละเอียดของ Memory Controller Initialize
- เพิ่มรายละเอียดของ Interrupt Controller Handling
- เพิ่ม Serial Port device driver สำหรับ GDB และ โปรแกรมสำหรับตรวจสอบ
- เพิ่ม System Timer Initialize และ Control
- เพิ่ม Wallclock driver

หลังจากได้ทำการพัฒนาแพลตฟอร์มใหม่เข้าไป ชื่อ “CE Control board DEMO (CE45) “ โดยการปรับปรุงแพลตฟอร์มที่ใกล้เคียงกันคือ ATMEL EB40 ได้ผลเป็นดังรูป (เลือก Template เมนูของ eCos Configurationtool) จากนั้นใช้ eCos Configuration tool ทำการปรับปรุงส่วนของ option ที่ต้องการสำหรับแอพพลิเคชั่นระบบฝัง ทำการคอมไพล์ซึ่งจะได้เป็นไลบรารีเพื่อนำไปลิงค์กับแอพพลิเคชั่นที่เขียนขึ้น

ตัวอย่างของโปรแกรมแอพพลิเคชั่น จะถูกเรียกใช้จากฟังก์ชั่น cyg_user_start()

```
/* File Application.c */
void cyg_user_start()
{
    printf("Hello eCos World with CE Control Board DEMO(CE45)");
   /* To DO โค้ด สำหรับ แอพพลิเคชั่น  */
}
```



รูปที่ 3.24 แสดงแพลตฟอร์ม "CE Control Board DEMO (CE45)"

บทที่ 4
การทดสอบและวิเคราะห์ผล

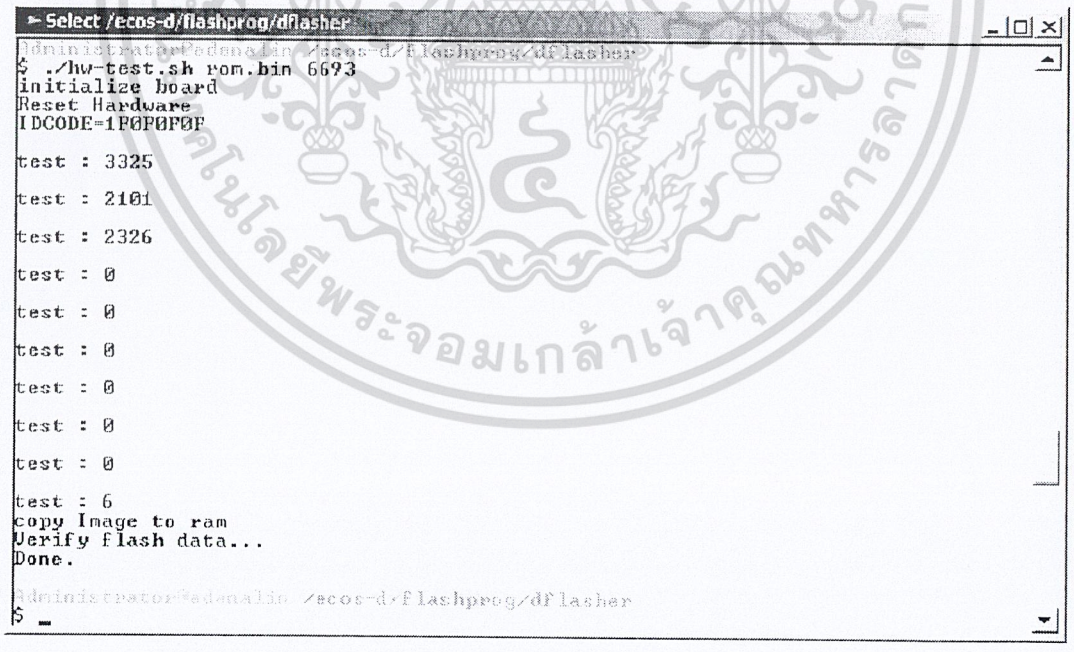
ในแต่ละขั้นตอนในการพัฒนา โครงงานนั้น มีการทดสอบและวิเคราะห์ผลว่าสิ่งที่ได้ที่ทำไปแล้วนั้นเป็นไปตามเป้าหมายของแต่ละขั้นหรือไม่ ซึ่งจำเป็น ต้องให้เป็นไปตามเป้าหมายนั้น แต่ถ้าไม่สามารถทำให้เป็นไปได้ตามเป้าหมายได้ จะมีการพิจารณาในสิ่งที่เกิดขึ้นและหาทางแก้ไขในโครงสร้างของระบบ หรือขั้นตอนและเป้าหมายต่างๆ แต่ไม่ให้กระทบถึงเป้าหมายของโครงงาน เมื่อได้ทำตามขั้นตอนต่างๆ เป็นที่เรียบร้อยนั้นจะมีการทดสอบและวิเคราะห์ผลอีกที ว่าระบบที่พัฒนาขึ้นมานั้นเป็นไปตามเป้าหมายของโครงงานหรือไม่

4.1 การทดสอบการทำงานของฮาร์ดแวร์

ในการทดสอบการทำงานของฮาร์ดแวร์ทำได้โดยการตรวจสอบที่แอลอีดีของภาคอีเทอร์เนตคอนโทรเลอร์ ว่าติดหรือดับ ขณะที่ต่อแหล่งจ่ายไฟเข้ากับบอร์ดควบคุม จากนั้นทำการตรวจสอบการทำงานของไมโครคอนโทรเลอร์ หน่วยความจำแรมโดยทำการอ่านและเขียนข้อมูลจากโฮสต์ ไปยังบอร์ดควบคุม โดยโปรแกรม Armtool ทำการเปรียบเทียบข้อมูลที่เขียนลงไปและข้อมูลที่อ่านออกมา ทำให้สามารถตรวจสอบได้ว่าบอร์ดควบคุมนั้นยังทำงานได้อย่างถูกต้อง ดังคำสั่งข้างล่างสามารถที่จะนำไปเขียนเป็น script เพื่อตรวจสอบการทำงานของบอร์ดควบคุมได้ ดังนี้

```
#!/bin/sh
filename=$1
word=$2
# board initializaton for ARM Control-board for Embedded System
# FLASH: 0x01000000          2MB     ROM    <----start
# RAM:   0x02000000          1MB     RAM    <----start
#############  reset hardware ############################
echo "initialize board"
./armtool.exe s
############EBI initialize memory ########################
./armtool.exe write 0xffe00000 0x01003325  #Flash initialize 0x1000000,16MB,2 Wait State
./armtool.exe write 0xffe00004 0x02002101  #Ram initialize  0x2000000,16MB,0 Wait State
./armtool.exe write 0xffe00008 0x03002326  #Ehernet initialize 0x3000000,16MB,2Wait State
```

```
./armtool.exe write 0xffe0000c 0x04000000  #not use
./armtool.exe write 0xffe00010 0x05000000  #not use
./armtool.exe write 0xffe00014 0x06000000  #not use
./armtool.exe write 0xffe00018 0x07000000  #not use
./armtool.exe write 0xffe0001c 0x08000000  #not use
./armtool.exe write 0xffe00020 1                ##Remap command
./armtool.exe write 0xffe00024 6               #Memory Controller
#################### Write SRAM ########################
echo  "copy Image to ram"
./armtool.exe write 0x2000000 $word  $filename
################### Read Sram  ########################
./armtool.exe read 0x02000000  $word  $filename.save
echo "Verify flash data..."
cmp -c $filename $filename.save
echo "Done."
```



รูปที่ 4.1 แสดงการตรวจสอบการทำงานของฮาร์ดแวร์

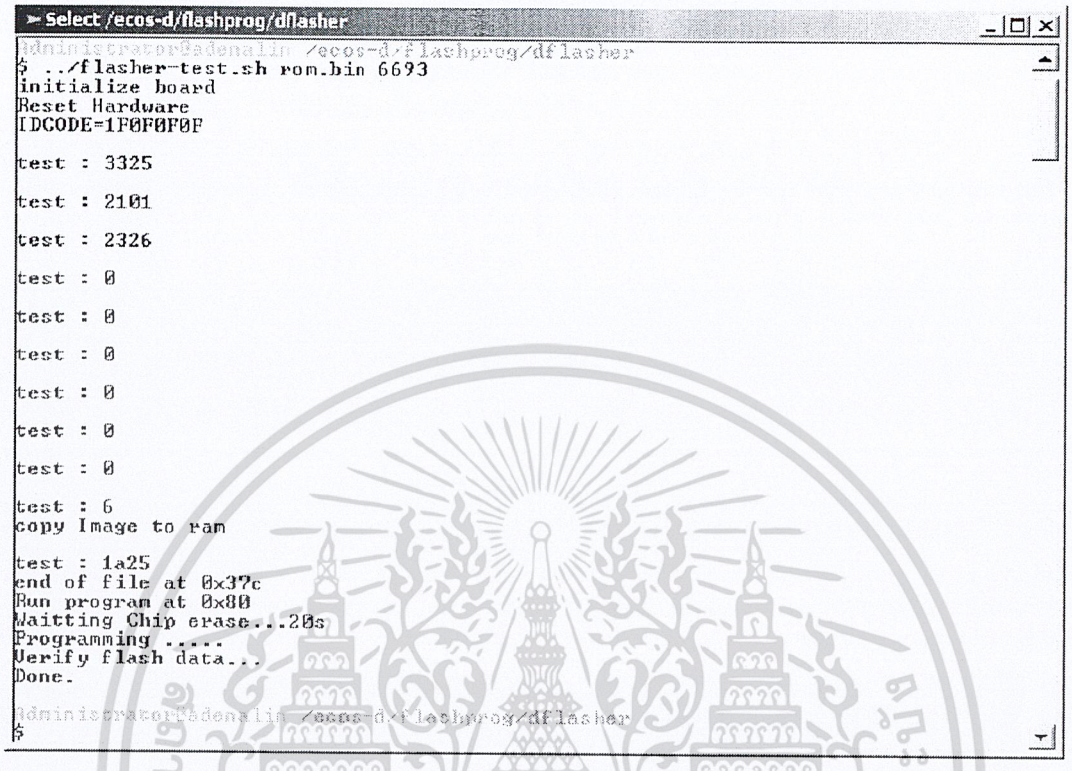
จากรูปที่ 4.1 แสดงให้เห็นการผลลัพธ์ที่ได้จากการรันสคริปท์ เพื่อทดสอบฮาร์ดแวร์ ในบรรทัดที่สามของเอาท์พุทถัดจากคำสั่ง เห็นได้ว่า IDCODE=1F0F0F0F ซึ่งเป็นชื่อของ ARM7TDMI ได้มาจากคำสั่ง ./armtool.exe s คือการรีเซทฮาร์ดแวร์ แสดงให้เห็นว่าการติดต่อการบอร์ดควบคุมนั้นไม่มีปัญหาแต่อย่างใด ใน บรรทัดถัดมาของ script เป็นการตั้งค่าเริ่มต้นให้กับ EBI bus จนก่อนจะถึง คำสั่งทำการคัดลอกข้อมูลที่เก็บอยู่ในไฟล์ที่ชื่อ rom.bin เข้าไปยัง ram เนื่องจาก AT91M40800 เป็นไมโครคอนโทรเลอร์ที่จำเป็นต้องมีการตั้งค่าเริ่มต้นทุกครั้งหลังจากที่มีรีเซทหรือเปิดเครื่องขึ้นมาเพื่อให้สามารถอ้างแอดเดรสได้อย่างถูกต้อง เมื่อทำการอ่านค่าได้ทำการโปรแกรมเข้าไปแล้ว มาตรวจสอบหากไม่มีข้อผิดพลาดคือ ข้อมูลนั้นเหมือนกันแล้ว ก็ไม่มีข้อผิดพลาดอะไรขึ้นมาแสดงผลจากคำสั่ง cmd –C $filename $filename.save เมื่อถึงจุดนี้ก็สามารถที่จะบอกได้ว่า โปรเซสเซอร์กับ หน่วยความจำนั้นไม่มีปัญหาแต่อย่างใด

จากนั้นจะเป็นการทดสอบการโปรแกรมหน่วยความความจำแฟลซ โดยจะทำการโปรแกรม ข้อมูลเข้าไปในแฟลซ สามารถทำได้โดยรันคำสั่งดังต่อไปนี้

```
#!/bin/sh
filename=$1
word=$2
# board initializaton for ARM Control-board for Embedded System
# FLASH: 0x01000000          2MB     ROM    <----start
# RAM:   0x02000000          1MB     RAM    <----start
#############  reset hardware  ##############################
echo "initialize board"
./armtool.exe s
############EBI initialize memory ########################
./armtool.exe write 0xffe00000 0x01003325  #Flash initialize 0x1000000,16MB,2 Wait State
./armtool.exe write 0xffe00004 0x02002101  #Ram initialize  0x2000000,16MB,0 Wait State
./armtool.exe write 0xffe00008 0x03002326  #Ehernet initialize 0x3000000,16MB,2Wait State
./armtool.exe write 0xffe0000c 0x04000000  #not use
./armtool.exe write 0xffe00010 0x05000000  #not use
./armtool.exe write 0xffe00014 0x06000000  #not use
./armtool.exe write 0xffe00018 0x07000000  #not use
./armtool.exe write 0xffe0001c 0x08000000  #not use
./armtool.exe write 0xffe00020 1                ##Remap command
```

```
./armtool.exe write 0xffe00024 6                #Memory Controller
################### Write SRAM ########################
echo  "copy Image to ram"
./armtool.exe write 0x2000000 $word  $filename
################### Write Number Word ##################
./armtool.exe w 0x1ffc $word
###################Write Flaher  ########################
./armtool.exe w 0x80 9999 flasher.bin
################### Running Flaher ########################
./armtool.exe x 0x80
###################Delay chip erase about 20 secound#########
echo "Waitting Chip erase...20s"
sleep 20s
################ Delay chip Progrimming depending on file ####
echo "Programming ....."
sleep 20s
################### Read Sram  ########################
./armtool.exe read 0x01000000  $word  $filename.save
echo "Verify flash data..."
cmp -c $filename $filename.save
echo "Done."
#################### E nd  ##############################
```

จากสคริปท์นี้จะสังเกตได้ว่าแตกต่างจาก มีการเพิ่มคำสั่งให้โหลดแฟลซโปรแกรมเข้าไปที่แอดเดรด 0x80 ซึ่งจะเป็นส่วนของหน่วยความจำเอสแรม(SRAM)ที่อยู่ภายในไมโครคอนโทรเลอร์ซึ่งหลังจากที่ได้ทำการรีแม็บ (Remap) หน่วยความจำแล้ว โดยแฟลซโปรแกรม(Flasher)จะทำหน้าที่ในการคัดลอกข้อมูลที่อยู่ในหน่วย ความจำแรม เข้าไปหน่วยความจำแฟลซ จากการทดลองจำเป็นต้องเป็นเวลาอย่างน้อย 20 วินาที ขณะทำการลบข้อมูลอยู่ในแฟลซ และต้องอีกระยะเวลาอย่างน้อยเป็นเวลา 10 วินาทีสำหรับโปรแกรมข้อมูลขนาดประมาณ 300 กิโลไบท์ นั่นหมายความว่าถ้ามีข้อมูลขนาด 1 เมกกะไบท์ สคริบท์ที่เขียนขึ้นก็ต้องทำการแก้ไขส่วนของหน่วยเวลาในการโปรแกรม (Delay) อย่างน้อย ประมาณ 30 วินาที จึงจะทำการโปรแกรมแฟลซได้เสร็จ ดังรูปข้างล่างแสดงการ โปรแกรมแฟลซ ขนาด 26,772 ไบท์



รูปที่ 4.2 แสดงการตรวจสอบการทำงานโปรแกรมแฟลช

หากไม่มีการรายงานข้อผิดพลาดขึ้นมาในขณะทำการตรวจสอบข้อมูลที่ได้ทำการโปรแกรม แสดงว่าทุกส่วนของบอร์ดควบคุมนั้น สามารถทำงานได้ไม่มีปัญหาแต่อย่างใด เพราะว่าสามารถทำการโปรแกรมแฟลชได้โดยอาศัยการทำงานของ ไมโครคอนโทรเลอร์ และหน่วยความจำที่อยู่ภายในบอร์ดควบคุมเป็นตัวทำงาน

ซึ่งเพียงเท่านี้ก็สามารถที่จะเขียนโปรแกรมเข้าไปควบคุมการทำงานของบอร์ดควบคุมได้แล้ว แต่การทำงานที่ได้ทดลองไปแล้วนั้น ยังคงต้องทำงานที่หน่วยความจำแรมเท่านั้น หมายถึงว่า หากถอดแหล่งจ่ายหรือหรือกดปุ่มรีเซทแล้ว โปรแกรมก็ไม่สามารถทำงานได้

4.2 การทดสอบการทำงานของซอร์ฟแวร์

การเขียนโปรแกรมเข้าไปทำงานที่หน่วยความจำแรมโดยอาศัย Armtool ที่ได้พัฒนาขึ้น จะเหมาะสมสำหรับงานในการพัฒนาโปรแกรม ไม่เหมาะสมที่ต้องไปโปรแกรมแฟลชทุกครั้ง เพราะการอ่านหรือเขียนหน่วยความจำแรม ไม่มีข้อจำกัดเหมือนการเขียนแฟลช แม้ว่าจะสามารถโปรแกรมได้ใหม่เป็น 100,000 ครั้งก็ตาม จำเป็นต้องคำนึงการใช้งานของแอพพลิเคชั่น ที่บางครั้งจำเป็นต้องเขียนข้อมูลในแฟลช เพื่อเก็บสถานะของงานเอาไว้ ดังนั้นการพัฒนาโปรแกรมจำเป็นต้องทดสอบการทำงานของโปรแกรมที่เขียนขึ้นโดยให้

ทำงานที่แรม เมื่อโปรแกรมทำงานถูกต้องและพร้อมที่จะให้ทำงานได้ด้วยตัวเอง จึงจะทำการโปรแกรมเข้าไปที่หน่วยความจำแฟลซ

การพัฒนาโปรแกรมให้บอร์ดควบคุมสามารถทำงานได้ด้วยตัวเอง (Stand Alone) นั้น สามารถทำได้ดังนี้

1. ต้องเขียนบูทโค้ด (Boot Code) ขึ้นมาเพื่อทำการกำหนดค่าเริ่มต้นให้กับฮาร์ดแวร์ ทดสอบและตรวจสอบข้อผิดพลาดของฮาร์ดแวร์ จากนั้นให้กระโดดไปทำงานที่แอพพลิเคชั่นที่เขียนขึ้น ปกติแล้วบูทโค้ดจะเป็นภาษาแอสเซมบลี หน้าที่หลักสามารถแบ่งออกได้ดังนี้

- ทำการรีแม็บหน่วยความจำ โดยคำสั่งแรกของเมื่อเปิดบอร์ดควบคุมให้ทำงานจะอยู่ที่แอดเดรส 0x00 ซึ่งเป็นรีเซทเวคเตอร์ของโมโครโปรเซสเซอร์ ARM7TDMI โดยก่อนคำสั่งรีแม็บแอดเดรส 0x00 จะถูกแม็บไปยังหน่วยความจำแฟลซขนาด 1 เมกกะไบท์ ซึ่งหลังจากทำการรีแม็บแล้ว หน่วยความจำแฟลซจะถูกแม็บไปที่แอดเดรส 0x01000000

-ทำการตรวจสอบอุปกรณ์ที่อยู่บนบอร์ด เพื่อรายงานข้อผิดพลาดที่เกิดขึ้นกับตัวฮาร์ดแวร์ จากนั้นก็ทำการกระโดดไปทำงานที่แอพพลิเคชั่นโปรแกรม ที่เขียนขึ้น

2. แอพพลิเคชั่นโปรแกรม เป็นส่วนของโปรแกรมที่เขียนขึ้นให้บอร์ดควบคุมนั้นทำงานได้ตามความต้องการการ

โปรแกรมการใช้งานให้บอร์ดควบคุมสามารถทำงานได้ด้วยตัวเอง(Stand Alone)

```
/* ชื่อไฟล์ boot.s เป็นส่วนที่ทำหน้าที่ในการรีแม็บหน่วยความจำ */
    .equ        EBI_BASE,0xFFE00000          /* EBI แอดเดรส */
        .global  __main
    __main:     .long    InitReset           /* reset vector */
    undefvec:   .long    undefvec            /* Undef  vector*/
    swivec:     .long    swivec              /* SW  interrupt vector*/
    pabtvec:    .long    pabtvec             /* Prefetch  abort vector */
    dabtvec:    .long    dabtvec             /* Data  abort  vector*/
    rsvdvec:    .long    rsvdvec             /* reserved */
    irqvec:     ldr      pc, [pc,#-0xF20]    /* IRQ : read the AIC */
    fiqvec:     ldr      pc, [pc,#-0xF20]    /* FIQ : read the AIC */
    InitReset:  mov      sp,#0x400           /*| Initialise Stack ที่ 0x020ffffc สิ้นสุด ram*/
                ldr      r10, PtInitTableEBI /* get the address of the chip select register image */
                movs     r0, pc, LSR #20     /* pc > 0x100000 */
                moveq    r10, r10, LSL #12   /* Mask the 12 highest bits of the address */
```

```
                moveq    r10, r10, LSR #12
                ldr      r12, PtInitRemap          /* get the real jump address ( after remap ) */
                ldmia    r10!, {r0-r9,r11}     /* load the complete image and the EBI base */
                stmia    r11!, {r0-r9}         /* store the complete image with the remap command */
                mov      pc, r12               /* jump and break the pipeline */
InitRemap:      b        main
PtInitTableEBI: .long InitTableEBI          /* Table for EBI initialization */
PtInitRemap:    .long InitRemap             /* address where to jump after REMAP */
InitTableEBI:   .long  0x01003325       /*0x01000000, 16MB, 0 hold, 16 bits,  2 WS */
                .long  0x02002121       /*0x02000000, 16MB, 0 hold, 16 bits, 0 WS  */
                .long  0x03002326       /*0x03000000, 16MB, 0 hold, 16 bits,  2 WS */
                .long  0x40000000           /* unused */
                .long  0x50000000           /* unused */
                .long  0x60000000           /* unused */
                .long  0x70000000           /* unused */
                .long  0x80000000           /* unused */
                .long  0x00000001           /* REMAP command */
                .long  0x00000006           /* 7 memory regions, standard read */
                .long  EBI_BASE             /* EBI Base address */
/* สิ้นสุดไฟล์ boot.s  */
```

ส่วนของไฟล์สำหรับแอพพลิเคชั่นโปรแกรมสามารถเขียนเป็นภาษาซีได้ดังนี้

```
/*   ไฟล์แอพพลิเคชั่นโปรแกรม  application.c  */
        int  main()
        {
           /* โค้ดสำหรับ แอพพลิเคชั่นโปรแกรม  */
            ..............
           return 0 ;
        }
/* สิ้นสุดไฟล์ application.c  */
```

ส่วนของไฟล์ Makefile ที่จะใช้ในการคอมไพล์โค้ดโปรแกรม

```
/* ชื่อไฟล์ Makefile สำหรับการคอมไพล์โค้ดอัตโนมัติ */
All:    main.bin
Main.bin :  boot.s main.c
        arm-elf-gcc –O2 –c –mcpu=arm7tdmi  boot.s  main.c
        arm-elf-ld   -Ttext 0x01000000 –o main.elf boot.o main.o
        arm-elf-objcopy –O binary main.elf $@
/* สิ้นสุด Makefile */
```

เมื่อถึงขั้นตอนนี้แล้วจะได้เอาท์พุทไฟล์ ชื่อ main.bin จากนั้น ใช้แฟลชโปรแกรมมิ่ง โปรแกรมข้อมูลเข้าไปยังแฟลช ทดสอบโปรแกรมโดยการรีเซทฮาร์ดแวร์ ดูว่าสามารถทำงานได้ถูกต้องตามต้องการได้หรือไม่ เป็นอันสิ้นสุดกระบวนการการสร้าง Stand Alone แอพพลิเคชั่น

บทที่ 5
บทวิจารณ์และสรุปผล

5.1 ผลที่ได้รับจากโครงงาน

1. บอร์ดควบคุมที่นำมาใช้งานสำหรับระบบฝังตัว โดยสามารถที่จะรองรับการทำงานได้ดังนี้
   - ควบคุมอุปกรณ์ต่างๆได้ โดยการสร้างเป็นแอพพลิเคชั่นติดต่อผ่านทางพอร์ตอนุกรมหรือพอร์ตขนาน
   - สามารถรองรับการทำงานผ่านระบบเครือข่ายคอมพิวเตอร์ ความเร็ว 10 เม็กกะบิตต่อวินาที
2. ไลบารีพื้นฐานของระบบปฏิบัติการอีคอส(eCos) ที่ใช้ในการสร้างแอพพลิเคชั่นควบคุมการทำงานของบอร์ด ทำให้การพัฒนาโปรแกรมสามารถทำได้ง่ายขึ้น
3. เครื่องมือที่ใช้ในการพัฒนาระบบฝังตัว
   - โปรแกรมที่ใช้สำหรับโปรแกรมหน่วยความจำแบบแฟลช
   - เครื่องมือในการคอมไพล์ , ดีบัก จากกนู (GNU)

5.2 ปัญหาที่พบ

จากการที่ได้พัฒนาโครงงานนี้มีส่วนที่จำเป็นต้องทำการศึกษาค้นคว้าอยู่มาก แต่เนื่องจากโครงงานเป็นลักษณะของการพัฒนาทั้งฮาร์ดแวร์และซอร์ฟแวร์ ให้มีการทำงานควบคู่กัน ซึ่งเป็นเรื่องเรื่องที่ค่อนข้างยาก ต้องใช้เวลามากและหาข้อมูลได้ยากอีกทั้งผู้พัฒนาไม่มีความรู้ประสบการณ์และความเชี่ยวชาญมาก่อน ปัญหาที่พบในโครง งานจึงมีมากพอสมควร ซึ่งสามารถที่จะแบ่งปัญหาออกมา 3 ส่วนใหญ่ๆ คือ

- ปัญหาทางด้านฮาร์ดแวร์
- ปัญหาทางด้านซอร์ฟแวร์
- ปัญหาทางด้านเทคนิคต่างๆ

5.2.1 ปัญหาทางด้านฮาร์ดแวร์

5.2.1.1 เนื่องจากการพัฒนาบอร์ดในช่วงเริ่มแรก ประสบปัญหาเกี่ยวกับการออกแบบหาอุปกรณ์ที่จะนำมาประกอบเป็นบอร์ดส่งผลทำให้การพัฒนาล่าช้าไปมาก

5.2.1.2 อุปกรณ์ที่ใช้ในการประกอบเป็นบอร์ดหายาก และต้องสั่งซื้อมาจากต่างประเทศซึ่งต้องประสบกับปัญหาความล้าช้า และด้านเงินทุนรวมถึงวิธีการสั่งซื้ออุปกรณ์

5.2.1.3 การขาดประสบการณ์ของผู้พัฒนา ในการต่ออุปกรณ์ทางด้านฮาร์ดแวร์ส่งผลให้ระยะเวลาในการพัฒนาช้าไปกว่าที่กำหนด

5.2.2 ปัญหาทางด้านซอฟต์แวร์

5.2.2.1 ต้องใช้ระยะเวลานานพอสมควร ในการศึกษาเครื่องมือที่จะต้องนำมาใช้ในการพัฒนาบอร์ด ในช่วงเริ่มต้นการพัฒนาซอฟต์แวร์

5.2.2.2 เนื่องจากผู้พัฒนามีประสบการณ์ด้านเขียนโปรแกรมน้อย ทำให้ต้องใช้ระยะเวลานานกว่าที่กำหนดไว้

5.2.3 ปัญหาทางด้านเทคนิคต่างๆ

5.2.3.1 ขาดแคลนเครื่องมือที่ทันสมัยที่ใช้ในการพัฒนา

5.2.3.2 การใช้งานเครื่องมือจาก กนู(GNU) ซึ่งจำเป็นต้องศึกษาอีกมาก เนื่องจากไม่เคยศึกษามาก่อน ทั้งในเรื่องการทำครอส-คอมไพล์เลอ(C Cross-Compiler)

5.2.3.3 แหล่งข้อมูลหรือตัวอย่างในการพัฒนาแอพพลิเคชั่นมีน้อยมาก ซึ่งเท่าที่หาได้ก็ไม่ตรงกับความต้องการ ดังนั้นจึงต้องทดลองด้วยตัวเอง ทำให้ต้องใช้เวลาค่อนข้างมากในการพัฒนา

5.3 แนวทางในการพัฒนาต่อ

- พัฒนาในส่วนของระบบปฏิบัติการให้สามารถใช้งานร่วมกับบอร์ดควบคุมให้ดียิ่งขึ้น
- พัฒนาเครื่องมือที่ใช้ในการโปรแกรมหน่วยความจำแฟลช ให้สามารถใช้งานได้ง่ายยิ่งขึ้น
- พัฒนาไดรเวอร์อีเทอร์เน็ตคอนโทรลเลอร์ให้สามารถทำงานได้กับระบบปฏิบัติการอีคอส (eCos)
- พัฒนาในตัวดีบักมอนิเตอร์ที่ใช้สำหรับควบคุมและแสดงผลการทำงานของบอร์ด
- พัฒนาแอพพลิเคชันที่ใช้กับอีเทอร์เน็ตคอนโทรลเลอร์ให้สามารถติดต่อกับอุปกรณ์ที่ต้องการได้
- นำไปประยุกต์ใช้งาน

5.4 สรุปผลโครงงาน

การพัฒนาโครงงานโดยภาพรวม ซึ่งเริ่มจากการศึกษาและค้นคว้าความเป็นไปได้ในด้านการนำไปใช้งานและการออกแบบทางด้านฮาร์ดแวร์และซอร์ฟแวร์ ตลอดจนส่วนประกอบต่างๆ ของบอร์ดควบคุมว่ามีอะไรบ้าง จาก นั้นก็ได้ทำการออกแบบ และสร้างเป็นบอร์ดตามที่ได้กล่าวไปแล้วในบทที่ผ่านมา

จากการศึกษาและค้นคว้า ทำให้ได้ข้อสรุปของส่วนประกอบหลักที่จะนำมาใช้ในการสร้างและพัฒนาบอร์ด ได้แก่ ไมโครโปรเซสเซอร์ ARM7TDMI , หน่วยความจำแฟลช 2 เมกกะไบต์ สามารถต่อขยายได้ 16 เมกกะไบต์ , หน่วยความจำ SRAM 1 เมกกะไบต์ สามารถต่อขยายได้ 16 เมกกะไบต์ , อีเทอร์เน็ทคอนโทรเลอร์ความเร็ว 10 เมกกะบิตต่อวินาที , EBI บัส สำหรับต่อขยายหน่วยความจำ , พอร์ตขนานที่ใช้สำหรับติดต่อกับอุปกรณ์ภายนอก , พอร์ตอนุกรมจำนวน 2 พอร์ตโดยพอร์ตหนึ่งสำหรับไว้ติดต่อกับโฮสต์และอีกพอร์ตหนึ่งเอาไว้สำหรับติดต่อควบคุมกับอุปกรณ์ภายนอก

ในส่วนของการพัฒนาซอร์ฟแวร์ หลังจากที่สร้างบอร์ดเสร็จเรียบร้อย ก็ได้เริ่มพัฒนาในส่วนของซอฟแวร์ ซึ่งได้แบ่งออกเป็นสองส่วน คือ On-board Prongamming และส่วนของระบบปฏิบัติการ ในส่วนของตัว On-board programming ได้พัฒนาโปรแกรมที่ชื่อ Armtool ทำงานร่วมกับโปรแกรม flasher สำหรับโปรแกรม Firmware ให้กับแอพพลิเคชั่นทำงานได้ ทางด้านของระบบปฏิบัติการ สามารถที่จะพัฒนาอีคอสให้ทำงานได้บนแพลตฟอร์มที่สร้างขึ้นมาได้ แต่ผลที่ได้ยังทำงานไม่สมบูรณ์มากนัก เพียงแค่รองรับการทำงานในระดับพื้นฐานยังและไม่สามารถพัฒนาใช้งานแอพพลิเคชั่นที่มีการทำงานซับซ้อนได้ เนื่องจากเป็นเรื่องที่ยากและมีเวลาไม่พอ

เมื่อได้พิจารณาถึงผลงานที่ได้จากโครงงานนี้ อาจจะเห็นว่าผลที่ได้ยังอยู่ในขั้นที่ไม่ดีมากนัก เพราะยังไม่สามารถพัฒนาส่วนของฮาร์ดแวร์และแอพพลิเคชั่นที่นำมาใช้ควบคู่ได้อย่างสมบูรณ์ ซึ่งในส่วนนี้ที่จริงเป็นความตั้งใจอย่างหนึ่งที่ทางผู้จัดทำหวังไว้ว่าจะทำให้สำเร็จ แต่เนื่องด้วยจากปัญหาต่างๆ ที่ประสบมา ทำให้ไม่สามารถมีเวลาที่จะพัฒนาประสิทธิภาพ การทำงานของบอร์ดโดยรวมให้ดีขึ้นกว่านี้

แต่หากได้พิจารณาถึงช่วงระยะเวลาที่ใช้ในการพัฒนา ทั้งในส่วนของการศึกษาค้นคว้าทฤษฎีและทำความเข้าใจกับส่วนต่างๆ ที่นำมาใช้ในการพัฒนาในส่วนของฮาร์ดแวร์และซอร์ฟแวร์ เพื่อให้สามารถทำงานร่วมกันได้นั้นยังไม่ค่อยสมบูรณ์ แต่โดยภาพรวมของโครงงานนี้ได้ให้ประโยชน์ต่อผู้ที่ได้พัฒนาในการเรียนรู้และเข้าใจหลักการต่างๆ ของการพัฒนาระบบฝังตัวทั้งทางด้านฮาร์ดแวร์และซอร์ฟแวร์เป็นอย่างมาก หลักการณ์ และแนวทางบางอย่างที่ผู้พัฒนานำมาใช้ในการพัฒนาระบบ เกิดจากการค้นคว้าหาข้อมูลและหาข้อสรุปสำหรับหลักการและแนวทางนั้นๆ เพื่อเลือกวิธีการที่เหมาะสมในการพัฒนา รวมทั้งสามารถที่จะนำไปพัฒนาต่อไปได้เป็นอย่างดี อีกทั้งผู้พัฒนายังได้ให้แนวทางในการพัฒนาต่อเพื่อให้ผู้ที่สนใจสามารถพัฒนาประสิทธิภาพการทำงานของบอร์ดได้อย่างอย่างสมบูรณ์ ด้วย

สุดท้ายนี้ผู้พัฒนามีความคาดหวังเป็นอย่างยิ่งว่าความรู้ต่างๆ ที่ได้จากการศึกษาคนคว้าและการวิจัยออกมานั้นจะเป็นประโยชน์สำหรับผู้ที่สนใจในด้านนี้ไม่มากก็น้อย และสามารถใช้เป็นแนวทางในการพัฒนาระบบฝังตัวต่อไป

บรรณานุกรม

1. "Embedded Systems Lab. KMITL" : http://www.esl.in.th/index.html
2. "**Thai Embedded Systems Association (TESA)**" : http://www.thaiesf.org/default.asp
3. "AT91M40800 Microcontrollers" : http://www.atmel.com/
4. "Flash memory" : http://www.fme.fujitsu.com/datasheets/e520846.pdf
5. "eCos Configuration Tool " : http://sources.redhat.com/ecos/docs-1.3.1/
6. "uClinux Operathing System" : http://www.uclinux.org
7. "eCos Operathing System" : http://sources.redhat.com/ecos/
8. "embedded Ultra-Low Power (ULP) Intel486 GX processor" : http://www.intel.com/design/intarch/datashts/272755.htm
9. "ตารางเปรียบเทียบประสิทธิภาพไมโครคอนโทรลเลอร์ " : http://www.arrow.com/www_engineers/arrowtech_solutions/high_end_embedded/high_end_fulldocument.pdf
10. "สั่งซื้ออุปกรณ์ " : http://www.findchips.com