

ระบบปฏิบัติการสำหรับชุดพัฒนาคอม86
OPERATING SYSTEM FOR COM86 PLATFORM



นายปรการ โกลากุล
นายภาคย์ ทองกุล
นายเสกศักดิ์ ลิ้มวิวัฒน์กุล

เลขหมู่.....
เลขทะเบียน...46195
วัน, เดือน, ปี...2 0 ส.ค. 2546

b.....
i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

641288620

ระบบปฏิบัติการสำหรับชุดพัฒนาคอม86
OPERATING SYSTEM FOR COM86 PLATFORM



โดย
นาย ปรการ โกลากุล
นาย ภาศย์ ทองกุล
นาย เตอศักดิ์ ลิ้มวิวัฒน์กุล

อาจารย์ที่ปรึกษา
ผศ. อภิเนตร อุณากุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบปฏิบัติการสำหรับชุดพัฒนาคอม86

OPERATING SYSTEM FOR COM86 PLATFORM

ผู้จัดทำ

1. นาย ปราการ โกลากุล

รหัสประจำตัว 41014257

2. นาย ภาคย์ ทองกุล

รหัสประจำตัว 41014332

3. นาย เลอศักดิ์ ลิ้มวิวัฒน์กุล

รหัสประจำตัว 41014360

อาจารย์ที่ปรึกษา

(ผศ. อภินทร อุณากุล)

ระบบปฏิบัติการสำหรับชุดพัฒนาคอม86

นาย ปราการ โกลากุล 41014257

นาย ภาสค์ ทองกุล 41014332

นาย เลอศักดิ์ ลิ้มวิวัฒน์กุล 41014360

ผศ.อภิเนตร อุนากุล อาจารย์ที่ปรึกษา

ปีการศึกษา 2544

บทคัดย่อ

ปัจจุบันเทคโนโลยีระบบฝังตัวได้กลายเป็นส่วนสำคัญของระบบควบคุมระยะไกลที่ใช้กันอย่างแพร่หลายในอุปกรณ์ต่างๆเช่น ยูพีเอส หรือแม้กระทั่งระบบควบคุมขนาดใหญ่ในโรงงาน แต่ในประเทศไทยนั้น การพัฒนาอุปกรณ์ระบบฝังตัวเหล่านี้ยังทำได้ลำบากทั้งที่ประเทศเรามีบุคลากรที่มีความสามารถอยู่มาก ทั้งนี้เนื่องจากต้องนำเข้าชุดพัฒนาจากต่างประเทศซึ่งมีราคาสูง

ปริญญานิพนธ์ฉบับนี้จึงนำเสนอระบบปฏิบัติการสำหรับชุดพัฒนาอุปกรณ์ระบบฝังตัวโดยชุดพัฒนานี้มีชื่อว่าคอม86 ซึ่งพัฒนาโดยคนไทยเพื่อใช้ในการตอบสนองต่อปัญหาที่ได้กล่าวมาข้างต้น โครงการนี้ได้นำเอาระบบปฏิบัติการมินิคซ์ซึ่งเป็นระบบปฏิบัติการบนพีซีมาพัฒนาดัดแปลงให้สามารถทำงานได้อย่างเหมาะสมบนบอร์ดคอม86 ทำให้เกิดชุดพัฒนาที่สมบูรณ์รองรับการพัฒนาอุปกรณ์ระบบฝังตัวที่มีความต้องการด้านการสื่อสาร เช่นระบบควบคุมระยะไกล

Operating System For Com86 Platform

Prakarn Kolakul

Parck Tongkul

Lersak Limwiwatkul

Asst. Prof. Apinetr Unakul Advisor

ABSTRACT

Today, embedded system technology become essential parts of remote-control system that is widely used not only in stand-alone equipments such UPS but also in the large control system of the industry. In Thailand, unfortunately, embedded system development in this field hasn't grown as it should to do, while our country has many efficient developers, because of high cost of imported development kits.

This thesis presents an operating system for embedded system development platform which called Com86. Com86 has been developed by Thai group in order to solving the problem mentioned above. In this project we have modified MINIX - an operating system for PC in order to run suitably on Com86 platform, result in creating a complete development kits for developing embedded system facilitate communication-capable application such as remote-control system.

กิตติกรรมประกาศ

การทำปริญญานิพนธ์ฉบับนี้ไม่อาจสำเร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือและร่วมมือจากหลายๆฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลงได้ก็คือ อาจารย์อภิเนตร อุณากุล ซึ่งเป็นอาจารย์ที่ปรึกษาวิทยานิพนธ์ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา อีกท่านที่อยากจะทำกล่าวถึงคือ คุณชนิษฐา ประสารสุข ที่ช่วยสนับสนุนการทำงานต่างๆให้ราบรื่น จึงขอขอบพระคุณเป็นอย่างสูง

ขอขอบคุณอาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ทุกท่านที่ให้ความรู้และคำปรึกษามาตลอดระยะเวลา 3 ปี

ขอขอบคุณสมาชิกทุกท่านในห้องปฏิบัติการ Embedded Systems Lab ให้คำปรึกษาช่วยเหลือให้คำแนะนำ และห้องปฏิบัติการ Embedded Systems Lab ที่เอื้อเฟื้อสถานที่ในการดำเนินโครงการ

ขอขอบคุณบริษัท เน็ตแก๊ตเจต จำกัด ที่เอื้อเฟื้อชุดพัฒนาคอม86มาใช้ในการโครงการนี้ ขอขอบคุณรุ่นพี่และเพื่อนๆทุกคนที่คอยให้คำปรึกษา ช่วยเหลือและให้กำลังใจตลอดมา สุดท้ายนี้ต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจเอาใจใส่เสมอมา ในทุกๆด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณและขอกราบขอบพระคุณมา ณ ที่นี้

ปราการ โกลากุล
ภาคย์ ทองกุล
เลอศักดิ์ ลิมวิวัฒน์กุล

สารบัญ

บทคัดย่อภาษาไทย.....	I
บทคัดย่อภาษาอังกฤษ.....	II
กิตติกรรมประกาศสารบัญ.....	III
สารบัญ	IV
สารบัญตาราง.....	VIII
สารบัญภาพ	IX
บทที่1 บทนำ.....	1
1.1 ความสำคัญและที่มา.....	1
1.2 วัตถุประสงค์ของงานวิจัย	1
1.3 ขอบเขตงานวิจัย	2
1.4 วิธีการดำเนินงาน	2
บทที่2 ระบบฝังตัว	3
2.1 ระบบฝังตัวคืออะไร	3
2.2 ลักษณะของระบบฝังตัว	4
2.3 องค์ประกอบของระบบฝังตัว	5
ไมโครโพรเซสเซอร์.....	5
หน่วยความจำ.....	6
ส่วนอุปกรณ์เชื่อมต่อ(Peripherals).....	6
ซอฟต์แวร์.....	6
2.4 แอปพลิเคชันของระบบฝังตัว	6
อุปกรณ์เครื่องใช้.....	7
ระบบควบคุมในโรงงาน ตึกอาคาร.....	7
ชุดพัฒนาที่ใช้ในด้านการศึกษา วิจัยและพัฒนา	8
2.5 แพลตฟอร์ม.....	8
ลักษณะของแพลตฟอร์มในแอปพลิเคชันต่างๆ	10
บทที่3 บอร์ดคอม86.....	11
3.1 สถาปัตยกรรมทางด้านฮาร์ดแวร์	11
ไมโครโพรเซสเซอร์.....	12
หน่วยความจำหลัก	13
อุปกรณ์ต่อเชื่อม	13
3.2 สถาปัตยกรรมทางด้านซอฟต์แวร์.....	15

3.3 จุดเด่นของบอร์ด คอม86	16
เป็นไมโครคอนโทรลเลอร์ที่มีความสามารถสูง	16
ง่ายในการเรียนรู้และพัฒนา	16
มีความสามารถในการเพิ่มขยายได้	17
มีเครื่องมือในการพัฒนาโปรแกรมมาก	18
3.4 แอปพลิเคชันเป้าหมายของชุดพัฒนา (Target Application)	18
Industrial control and Automation	18
USB peripheral	18
Embedded web server	18
Robotics	18
บทที่ 4 ระบบปฏิบัติการสำหรับระบบฝังตัว	19
4.1 ระบบปฏิบัติการสำหรับระบบฝังตัว	19
คุณสมบัติของระบบปฏิบัติการที่ใช้ในระบบฝังตัว	19
4.2 ประเภทของระบบปฏิบัติการที่ใช้ในระบบคอมพิวเตอร์ฝังตัว	19
Real-Time Operating System	19
Non Real-Time Operating System	20
4.3 ลักษณะของระบบปฏิบัติการที่เหมาะสมกับชุดพัฒนาคอม86	20
4.4 ระบบปฏิบัติการและเทคโนโลยีที่น่าสนใจสำหรับนำมาพัฒนาในโครงการ	20
4.4.1 ระบบปฏิบัติการมินิกซ์	20
4.4.1.1 ลักษณะเด่นของมินิกซ์เวอร์ชัน 2.0	22
4.4.1.2 ฮาร์ดแวร์อย่างต่ำที่ต้องการ	23
4.4.1.3 การใช้งานระบบปฏิบัติการมินิกซ์ร่วมกับระบบปฏิบัติการอื่น	23
4.4.1.4 ลิขสิทธิ์ของมินิกซ์	24
4.4.2 ระบบปฏิบัติการฟรีดอส (Freedos Operating System)	24
4.4.2.1 ประวัติการพัฒนาฟรีดอส	25
4.4.2.2 ลักษณะเด่นของระบบปฏิบัติการฟรีดอส	26
4.4.2.3 ฮาร์ดแวร์ที่ต้องการ	26
4.4.2.4 ซอฟต์แวร์ที่ฟรีดอสสนับสนุน	26
4.4.2.5 องค์ประกอบของระบบปฏิบัติการฟรีดอส	27
4.4.2.6 ความสามารถด้านเครือข่าย	28
4.4.2.7 ลิขสิทธิ์ของฟรีดอส	29
4.4.3 ระบบปฏิบัติการอีแอลเคส (ELKS Operating System)	29
4.4.3.1 เปรียบเทียบลักษณะกับอีแอลเคส	29
4.4.3.2 ลักษณะเด่นของอีแอลเคส	30

4.4.4 KVM	30
4.4.4.1 จุดมุ่งหมายของการออกแบบ KVM	30
4.4.4.2 ความสัมพันธ์ระหว่าง KVM กับ J2ME	31
บทที่ 5 ระบบปฏิบัติการมินิกซ์	33
5.1 เหตุผลในการเลือกมินิกซ์	33
5.2 มินิกซ์	36
5.2.1 โครงสร้างภายในของมินิกซ์	36
5.2.2 การจัดการโปรเซส	38
5.2.3 การสื่อสารระหว่างโปรเซส	38
5.2.4 การจัดการเวลาการทำงาน	39
5.2.5 กระบวนการบูตของมินิกซ์	39
5.2.6 โครงสร้างซอร์สโค้ดของเคอร์เนลระบบปฏิบัติการมินิกซ์	41
บทที่ 6 รูปแบบการพัฒนามินิกซ์บนคอม86	43
6.1 กระบวนการพัฒนามินิกซ์บนคอม86	43
6.2 เอคิเตอร์ที่ใช้ในการพัฒนา	45
6.3 ล็อกไฟล์สำหรับการพัฒนา	45
6.4 ข้อกำหนดของการตั้งชื่อไฟล์ diff	47
บทที่ 7 การพัฒนามินิกซ์บนคอม86	49
7.1 บทนำ	49
7.2 เวอร์ชันในการพัฒนา	49
7.3 มินิกซ์บนคอม86 เวอร์ชันแรก เวอร์ชัน 0.1.1	53
7.3.1 เริ่มต้นที่ 0.1.1/src/kernel/Makefile	54
7.3.2 พิจารณา 0.1.1/include/minix/config.h	55
7.3.3 ไดรเวอร์สำหรับดิสก์แบบสกซ์ซี่	60
7.3.4 ไดรเวอร์ที่สนับสนุนดีไวซ์ที่เป็นซีรอม	65
7.3.5 ไดรเวอร์ที่สนับสนุนดีไวซ์ที่คอมแพตทิเบิลกับการ์ดเสียงของชาวคัสเตอร์	65
7.3.6 ไดรเวอร์สำหรับดิสก์แบบมาตรฐาน	66
7.3.7 ไดรเวอร์สำหรับการ์ดอีเทอร์เน็ตของทรินคอม และเวสต์เทิร์นดิจิตอล	66
7.3.8 ดีไวซ์ไดรเวอร์ที่สนับสนุนการอ่านเขียนระบบไฟล์บนดอส เมื่อจำลองให้ทำงานบนดอส	67
7.3.9 ดีไวซ์ไดรเวอร์สำหรับการสื่อสารผ่านพอร์ตขนาน	67
7.3.10 ไดรเวอร์สำหรับฟลอปปี้ดิสก์	69
7.4 มินิกซ์บนคอม86 เวอร์ชัน 0.1.2	69
7.4.1 ประเมินแนวทางในการแก้ไขในเบื้องต้น	70
7.4.2 พิจารณาถึงข้อจำกัดของหน่วยความจำแบบโปรเทกต์เคด โหมด	72

7.4.3	พิจารณาฟังก์ชันแรก mem_vid_copy	74
7.4.4	พิจารณาฟังก์ชันในการแสดงผลอีกฟังก์ชันหนึ่ง vid_vid_copy	76
7.4.4.1	พิจารณาในฟังก์ชัน scroll_screen.....	77
7.4.4.2	พิจารณาฟังก์ชัน do_escape.....	78
7.4.4.3	พิจารณาฟังก์ชัน cons_org0.....	79
7.4.5	พิจารณาฟังก์ชันที่มีการติดต่อกับฮาร์ดแวร์โดยตรงในไฟล์ console.c.....	80
7.5	มินิซันคอม86 เวอร์ชัน 0.1.3.....	84
7.6	มินิซันคอม86 เวอร์ชัน 0.2.1.....	86
7.6.1	ศึกษาเรื่องการใช้งานอินเทอร์พรีตของโปรเซสเซอร์ของแพลตฟอร์มคอม86	87
7.6.2	ศึกษาเรื่องการใช้งานรีจิสเตอร์เวลาของโปรเซสเซอร์ของแพลตฟอร์มคอม86	90
7.6.3	ศึกษาเรื่องระบบอินเทอร์พรีตในระบบปฏิบัติการมินิซันและทำการแก้ไขและปรับปรุง.....	91
7.6.4	ศึกษาดีไวซ์ไครเวอร์เวลาของระบบปฏิบัติการมินิซัน เพื่อแก้ไขและปรับปรุง	96
7.7	มินิซันคอม86 เวอร์ชัน 0.1.4 และเวอร์ชัน 0.2.2	101
7.8	มินิซันคอม86 เวอร์ชัน 0.1.5 และ 0.2.3	103
7.9	มินิซันคอม86 เวอร์ชัน 0.2.4.....	104
7.10	มินิซันคอม86 เวอร์ชัน 1.0.0.....	105
7.11	สรุปผลการพัฒนาแต่ละเวอร์ชัน.....	105
7.12	ภาพผลการทดลอง.....	107
บทที่ 8	บทวิจารณ์และสรุปผล.....	113
8.1	สรุปผลงาน.....	113
8.2	ประโยชน์จากโครงการ	113
8.3	ปัญหาที่พบจากโครงการ.....	113
8.4	แนวทางการพัฒนาต่อ	114
8.5	การนำโครงการไปประยุกต์ใช้งาน	114
8.6	สรุปภาพรวมของโครงการ	114
บรรณานุกรม		115

สารบัญตาราง

ตาราง 1 แสดงข้อพิจารณาเบื้องต้นที่จำเป็น.....	33
ตาราง 2 แสดงคุณสมบัติที่สนับสนุนการใช้งานของมินิกซ์ อีแอลเอส และฟรีดอส.....	34
ตาราง 3 แสดงเครื่องมือในการพัฒนาของมินิกซ์และอีแอลเอส.....	35
ตาราง 4 เปรียบเทียบลักษณะต่างๆระหว่างมินิกซ์กับอีแอลเอส.....	36
ตาราง 5 เวอร์ชันและจุดประสงค์ในการพัฒนาในแต่ละเวอร์ชันของมินิกซ์บนคอม86.....	52
ตาราง 6 แสดงจุดประสงค์ในการพัฒนามินิกซ์เวอร์ชัน 0.1.1.....	54
ตาราง 7 ไฟล์ที่ทำการพิจารณาตัดออก.....	60
ตาราง 8 แสดงจุดประสงค์ในการพัฒนามินิกซ์เวอร์ชัน 0.1.2.....	69
ตาราง 9 แสดงบริการไบออสในเรื่องของการแสดงผลที่สนับสนุนโดยไบออสของคอม86.....	70
ตาราง 10 แสดงอักขระแอสเคปที่แสดงผลไม่ถูกต้องในเวอร์ชัน 0.1.2.....	79
ตาราง 11 แสดงจุดประสงค์ในการพัฒนามินิกซ์เวอร์ชัน 0.1.3.....	84
ตาราง 12 แสดงจุดประสงค์ในการพัฒนามินิกซ์เวอร์ชัน 0.2.1.....	86
ตาราง 13 แสดงจุดประสงค์ในการพัฒนามินิกซ์เวอร์ชัน 0.1.4 และเวอร์ชัน 0.2.2.....	102
ตาราง 14 แสดงจุดประสงค์ในการพัฒนามินิกซ์เวอร์ชัน 0.1.5 และ 0.2.3.....	103
ตาราง 15 แสดงจุดประสงค์ในการพัฒนามินิกซ์เวอร์ชัน 0.2.4.....	104
ตาราง 16 แสดงจุดประสงค์ในการพัฒนามินิกซ์เวอร์ชัน 1.0.0.....	105

สารบัญภาพ

รูปที่ 1 แสดงถึงตัวอย่างของระบบฝังตัวที่พบในปัจจุบัน.....	3
รูปที่ 2 แสดงถึงภาพตลาดของระบบ ฝังตัวเกี่ยวกับคอมพิวเตอร์ตั้งโต๊ะ	4
รูปที่ 3 อุปกรณ์ที่ออกแบบมาสำหรับใช้ท่องเที่ยวโดยเฉพาะ.....	7
รูปที่ 4 ระบบควบคุมโรงงานอัตโนมัติ	8
รูปที่ 5 ระบบคอมพิวเตอร์ประกอบไปด้วย ฮาร์ดแวร์ ซอฟต์แวร์ระบบ และ โปรแกรมแอปพลิเคชัน	9
รูปที่ 6 แพลตฟอร์มประกอบด้วยชั้นต่างๆของฮาร์ดแวร์และซอฟต์แวร์	10
รูปที่ 7 แสดงถึงส่วนประกอบต่างๆของบอร์ดคอม86.....	11
รูปที่ 8 บล็อกไดอะแกรมแสดงองค์ประกอบของไมโครคอนโทรลเลอร์ AM186CC	12
รูปที่ 9 ภาพบอร์ดคอม86.....	14
รูปที่ 10 แสดงถึงสถาปัตยกรรมทางด้านซอฟต์แวร์บนบอร์ดคอม86	15
รูปที่ 11 แสดงถึงรูปแบบการพัฒนาโปรแกรมโดยใช้บอร์ดคอม86.....	17
รูปที่ 12 แสดงการเชื่อมต่อของบอร์ดขยายกับบอร์ดคอม86	17
รูปที่ 13 แพลตฟอร์มของ Java 2 Micro Edition	32
รูปที่ 14 โครงสร้างโปรเซสของมินิกซ์.....	36
รูปที่ 15 กระบวนการบูต	40
รูปที่ 16 โครงสร้างแสดงฟังก์ชันการทำงานของซอร์สโค้ดไฟล์ต่างๆ	42
รูปที่ 17 รูปแบบการพัฒนา minixc.....	43
รูปที่ 18 ลำดับในการพัฒนามินิกซ์บนคอม86	53
รูปที่ 19 แสดงลำดับในการพัฒนามินิกซ์เวอร์ชัน 0.1.1	54
รูปที่ 20 องค์ประกอบของมินิกซ์เมื่อมองจากมุมมองในลักษณะของลำดับชั้นของโปรเซส.....	59
รูปที่ 21 แสดงลำดับในการพัฒนามินิกซ์เวอร์ชัน 0.1.2	70
รูปที่ 22 ทางเลือกในการพัฒนาวิธีการแสดงผล.....	71
รูปที่ 23 แสดงความสัมพันธ์ของฟังก์ชัน vid_vid_copy	77
รูปที่ 24 แสดงความสัมพันธ์ของฟังก์ชัน beep กับ stop_beep.....	81
รูปที่ 25 แสดงความสัมพันธ์ของฟังก์ชัน ga_program.....	82
รูปที่ 26 แสดงความสัมพันธ์ของฟังก์ชัน set_6845.....	83
รูปที่ 27 แสดงลำดับในการพัฒนามินิกซ์เวอร์ชัน 0.1.3	84
รูปที่ 28 แสดงความสัมพันธ์กับไฟล์ keyboard.c	85
รูปที่ 29 แสดงความสัมพันธ์ของฟังก์ชัน kb_init กับฟังก์ชันอื่น.....	85
รูปที่ 30 แสดงลำดับในการพัฒนามินิกซ์เวอร์ชัน 0.2.1	87

รูปที่ 31 ไดอะแกรมองค์ประกอบของระบบอินเทอร์เน็ตใน AM186CC	88
รูปที่ 32 รีจิสเตอร์ PRIMSK	89
รูปที่ 33 รีจิสเตอร์ IMASK	89
รูปที่ 34 รีจิสเตอร์ EO1	90
รูปที่ 35 ไดอะแกรมของไทมเมอร์	90
รูปที่ 36 การใช้งานไทมเมอร์บนแพลตฟอร์มคอม86	91
รูปที่ 37 แสดงลำดับการเกิดอินเทอร์เน็ตในระบบมินิกซ์	92
รูปที่ 38 ลำดับการทำงานในการบริการอินเทอร์เน็ตของมินิกซ์	95
รูปที่ 39 แสดงระบบจัดการเวลาในไอบีเอ็มพีซีคอมแพคทิเบิล	97
รูปที่ 40 แสดงค่าคงที่ที่ใช้ในการจัดการเวลาในไอบีเอ็มพีซีคอมแพคทิเบิล	98
รูปที่ 41 การจัดการเวลาในแพลตฟอร์มคอม86 โดยใช้มินิกซ์	98
รูปที่ 42 รีจิสเตอร์ T0CON	99
รูปที่ 43 รีจิสเตอร์ T2CON	99
รูปที่ 44 รีจิสเตอร์ CH0CON	99
รูปที่ 45 รีจิสเตอร์ IMASK	100
รูปที่ 46 แสดงลำดับในการพัฒนามินิกซ์เวอร์ชัน 0.1.4 และมินิกซ์เวอร์ชัน 0.2.2	103
รูปที่ 47 แสดงลำดับในการพัฒนามินิกซ์เวอร์ชัน 0.1.5 และ 0.2.3	104
รูปที่ 48 แสดงลำดับในการพัฒนามินิกซ์เวอร์ชัน 0.2.4	104
รูปที่ 49 แสดงลำดับในการพัฒนามินิกซ์เวอร์ชัน 1.0.0	105
รูปที่ 50 ขั้นตอนขณะถ่ายโอนเนื้อหาของระบบปฏิบัติการมินิกซ์ลงคอม86	107
รูปที่ 51 มินิกซ์บูตมอนิเตอร์บนคอม86	107
รูปที่ 52 มินิกซ์บูตมอนิเตอร์หลังจากกดคีย์แอสเคปเพื่อเข้าสู่มินิกซ์บูตมอนิเตอร์	108
รูปที่ 53 ขั้นตอนเมื่อทำการเริ่มต้นระบบปฏิบัติการมินิกซ์	108
รูปที่ 54 มินิกซ์รุ่น ปรับปรุงและเปลี่ยนแปลงดีไวส์ไครเวอร์ในส่วนการอ่านเขียนข้อมูลกับแฟลชดิสก์	109
รูปที่ 55 มินิกซ์ เวอร์ชัน 0.2.3 ปรับปรุงเรื่องการรับข้อมูลเข้า โดยใช้บริการไบออสเซอร์วิสยูทิลิตี้	109
รูปที่ 56 มินิกซ์บนคอม86 เวอร์ชัน 0.2.4 ปรับปรุงและเปลี่ยนแปลงดีไวส์ไครเวอร์ของอีเทอร์เน็ต	110
รูปที่ 57 ทำการทดสอบโดยใช้โปรโตคอล ICMP ในการทดลองใช้คำสั่ง ping	110
รูปที่ 58 จำนวนแพ็กเก็ตที่ทำการ ping ออก	111
รูปที่ 59 ทำการ ping เข้ามาจากภายนอก จะเห็นได้ว่า ping เข้ามาได้ถูกต้อง	111
รูปที่ 60 จำนวนแพ็กเก็ตที่ทำการรับเข้ามาเมื่อทำการ ping เข้ามา	112

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มา

จากการที่เทคโนโลยีในปัจจุบันได้มีการพัฒนาไปอย่างรวดเร็ว ข้อสังเกตอย่างหนึ่งที่ได้จากการเติบโตของเทคโนโลยีเหล่านั้น ทำให้ทราบว่าในไม่ช้าอุปกรณ์ต่างๆ ไม่ว่าจะเล็กหรือใหญ่ จะมีส่วนที่เป็นคอมพิวเตอร์เป็นองค์ประกอบอยู่ด้วยเสมอ ซึ่งเป็นแนวโน้มในอนาคตที่น่าจับตามอง ปัจจุบันเองก็มีอุปกรณ์ลักษณะนี้ให้เห็นเพิ่มมากขึ้นเรื่อยๆ จนบางครั้งเราแทบไม่รู้สึกละเลยว่ามีอยู่ด้วยหรือไม่ ยกตัวอย่างเช่น โทรศัพท์มือถือ ซึ่งนับวันจะมีลูกเล่น มีอะไรแปลกใหม่เพิ่มมากขึ้นทุกวัน สามารถต่อเชื่อมเข้ากับอินเทอร์เน็ตได้ สามารถเช็คเมลได้ เป็นต้น ตัวอย่างหนึ่งที่เห็นได้ชัดเจน คงหนีไม่พ้นอุปกรณ์จัดการข้อมูลส่วนบุคคล (Personal Data Assistant) เช่น พ็อกเก็ตพีซี (PocketPC) หรือ ปาล์ม (Palm) ซึ่งคงคุ้นตากันดี อุปกรณ์เหล่านี้ต้องมีคอมพิวเตอร์เป็นหัวใจหลักในการทำงาน สิ่งที่กำลังจะเกิดขึ้นในอนาคตคืออุปกรณ์กำลังจะผนวกรวมความสามารถของคอมพิวเตอร์ไว้ภายใน ซึ่งเราเรียกระบบคอมพิวเตอร์ที่ผนวกรวมกับอุปกรณ์ต่างๆ นี้ว่า ระบบฝังตัว (Embedded System)

ซึ่งในปัจจุบันตลาดทางด้านนี้ ได้มีการขยายตัวเจริญเติบโตอย่างรวดเร็ว การที่จะสร้างอุปกรณ์ทางด้านระบบฝังตัวจำเป็นต้องมีสภาพแวดล้อมที่เหมาะสมสำหรับการพัฒนาที่ต้องการช่วงเวลาในการนำผลิตภัณฑ์ลงสู่ตลาด (Time-to-market) ที่น้อย ต้นทุนในการพัฒนาดำ และรองรับต่อการพัฒนาแอปพลิเคชันที่เปลี่ยนแปลงอย่างรวดเร็ว จึงจำเป็นต้องการแพลตฟอร์มที่ดีสำหรับการพัฒนา โดยปกติแล้วในการที่จะผลิตสินค้าที่เป็นระบบฝังตัว มักจะต้องพัฒนาใหม่เกือบทั้งหมดทั้งส่วนของฮาร์ดแวร์และซอฟต์แวร์ จึงทำให้เวลาในการออกแบบและพัฒนาค่อนข้างนานและทำให้ต้นทุนราคาสูงขึ้น แต่ถ้าหากมีแพลตฟอร์มที่สามารถรองรับการพัฒนาแอปพลิเคชันในด้านใดด้านหนึ่งหรือกลุ่มของแอปพลิเคชันที่มีความต้องการใกล้เคียงกัน โดยที่แพลตฟอร์มนั้นมีสภาพแวดล้อมที่เอื้อต่อการพัฒนาและสามารถให้ความต้องการที่แอปพลิเคชันนั้นๆ ต้องการได้ ทำให้ไม่ต้องพัฒนาใหม่ทั้งหมด สามารถพัฒนาแอปพลิเคชันต่อไปได้ทันที ส่งผลให้ระยะเวลาในการพัฒนาผลิตภัณฑ์ลดลง ต้นทุนได้มาก และทำให้ช่วงเวลาในการนำผลิตภัณฑ์ลงสู่ตลาดสั้นลงในขณะที่ยังคงรักษาคุณภาพของผลิตภัณฑ์ไว้

แพลตฟอร์มจะประกอบไปด้วยฮาร์ดแวร์และซอฟต์แวร์โดยในส่วนซอฟต์แวร์นั้นมีระบบปฏิบัติการเป็นส่วนประกอบที่สำคัญ ดังนั้นการพัฒนาระบบปฏิบัติการสำหรับแพลตฟอร์มที่ดีจึงนับเป็นสิ่งที่สำคัญอย่างยิ่งสำหรับการพัฒนาระบบฝังตัว

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 เพื่อเป็นการพัฒนาแพลตฟอร์มสำหรับงานทางด้านระบบฝังตัว โดยคนไทยเพื่อคนไทย สามารถพัฒนาต่อเติมให้ตรงกับความต้องการในการนำไปใช้งานได้อย่างรวดเร็ว ไม่ต้องเรียนรู้ระบบมากนัก เกินไปนัก อันทำให้ง่ายต่อการนำไปพัฒนา เป็นการจุดประกาย ให้เกิดเป็นสังคมในการพัฒนาระบบฝังตัว ขึ้นในประเทศไทย

1.2.2 เพื่อเป็นการศึกษาถึงการทำงานของระบบปฏิบัติการอย่างละเอียด ว่ามีการทำงานในแต่ละส่วนเป็นอย่างไร มีทฤษฎีอะไรบ้างเกี่ยวข้อง ทำให้ทราบว่าในการพัฒนาระบบปฏิบัติการขึ้นมาต้องทำอย่างไร ต้อง เรียนรู้อะไรบ้าง

1.2.3 เพื่อเป็นการเรียนรู้ในการทดลอง ออกแบบ อิมพลีเมนต์ ทั้งในส่วนของการซอฟต์แวร์ระบบ (System Software) และคอมพิวเตอร์ฮาร์ดแวร์ในระดับลึก

1.2.4 เพื่อเป็นการพัฒนาองค์ความรู้ (Knowledge base) สำหรับนำไปใช้ในการวิจัยและพัฒนาทางด้าน ระบบฝังตัวต่อไปในอนาคต

1.3 ขอบเขตงานวิจัย

โครงการนี้จะสร้างระบบปฏิบัติการสำหรับบอร์ดคอม86ขึ้นมา โดยจะคัดเลือกเอาระบบปฏิบัติการที่ทำงานอยู่บนโครงสร้างสถาปัตยกรรม x86 แบบ 16 บิต มาพัฒนาดัดแปลง เพื่อให้เหมาะสมกับงานระบบฝังตัว รวมทั้งสร้างเครื่องมือช่วยในการพัฒนา ไบโบลาร์สำหรับรองรับการพัฒนาแอปพลิเคชัน และคู่มือการใช้งาน ซึ่งจะทำให้เกิดแพลตฟอร์มที่รองรับการพัฒนาแอปพลิเคชันที่ต้องการประสิทธิภาพการประมวลผลของโปรเซสเซอร์ในระดับกลางและมีความต้องการทางด้าน การติดต่อสื่อสาร เช่นงานทางด้านระบบควบคุมโรงงานอัตโนมัติ เป็นต้น

อนึ่งโครงการนี้ไม่ได้ครอบคลุมการถึงการออกแบบพัฒนาบอร์ดคอม86 ทั้งหมด แต่ได้มีการนำชุดพัฒนาคอม86 ของบริษัท Netgadgets มาสำหรับใช้เป็นแพลตฟอร์มในการวิจัยและพัฒนาระบบปฏิบัติการในโครงการนี้

1.4 วิธีการดำเนินงาน

งานวิจัยในโครงการนี้แบ่งการทำงานออกเป็นลำดับขั้นตอนดังนี้

1.4.1 ศึกษาสำรวจในงานวิจัยทางด้านระบบปฏิบัติการที่มีอยู่ในปัจจุบัน เพื่อเป็นแนวทางและแหล่งข้อมูลสำหรับการวิจัยและพัฒนาระบบปฏิบัติการของโครงการ

1.4.2 ศึกษาในระบบฝังตัว ทั้งในเรื่องของความหมายของระบบฝังตัวลักษณะสำคัญของระบบฝังตัวองค์ประกอบของระบบฝังตัว รวมไปถึงอุปกรณ์หรือระบบงานที่เป็นที่ระบบฝังตัว การพัฒนาทางด้านระบบฝังตัว

1.4.4 ศึกษาและเรียนรู้การทำงานของบอร์ดคอม86 ในเรื่องของการพัฒนาซอฟต์แวร์ การพัฒนาระบบปฏิบัติการในโครงการ การใช้งาน ข้อจำกัดต่างๆทางด้านฮาร์ดแวร์

1.4.4 พัฒนาโปรแกรมเครื่องมือ (Software Tools) สำหรับใช้ในการพัฒนาโปรแกรมบนบอร์ดคอม86

1.4.5 ศึกษาและพัฒนาระบบปฏิบัติการมินิกซ์ (Minix) ในเรื่องของสถาปัตยกรรมของระบบปฏิบัติการการทำวิศวกรรมย้อนกลับ (Reverse engineering) ของระบบปฏิบัติการ การปรับแต่งระบบปฏิบัติการให้เหมาะสมกับบอร์ดคอม86

1.4.6 สร้างแอปพลิเคชันขึ้นมาทดสอบการทำงานของระบบปฏิบัติการที่ได้วิจัยและพัฒนาขึ้นมา

1.4.7 รวบรวมข้อมูลที่ได้จากการวิจัยและพัฒนา ทำเป็นเอกสารต่างๆเช่นคู่มือการใช้งานระบบปฏิบัติการ คู่มือสำหรับการพัฒนาระบบปฏิบัติการ

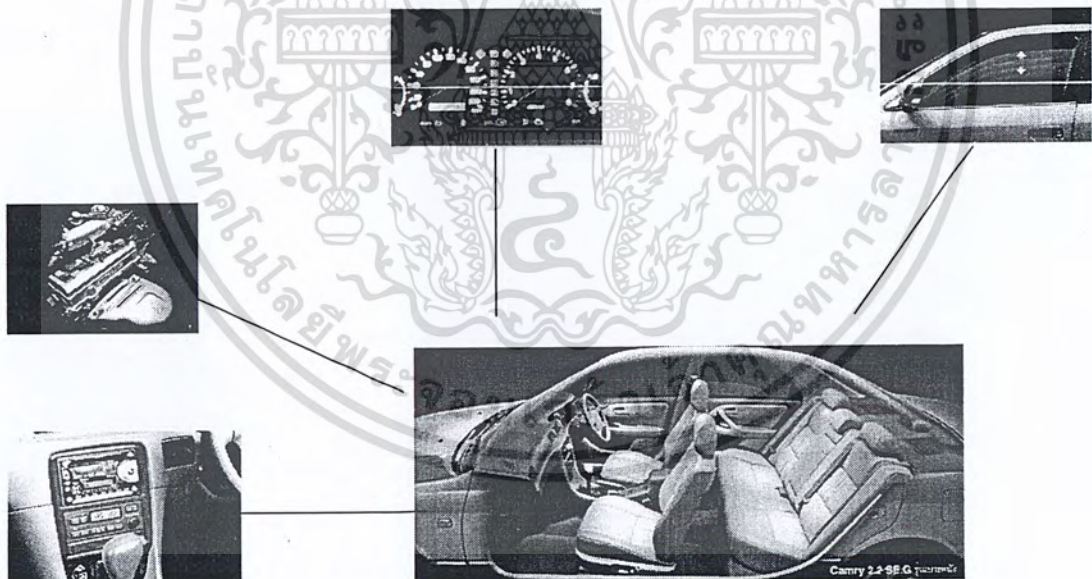
บทที่ 2

ระบบฝังตัว

2.1 ระบบฝังตัวคืออะไร

ในปัจจุบันคอมพิวเตอร์ได้เข้ามามีบทบาทต่อชีวิตประจำวันเราเป็นอย่างมาก เรามักจะคุ้นเคยและรู้จักกับคอมพิวเตอร์ที่ใช้งานอยู่ในปัจจุบันในรูปของแล็ปท็อปคอมพิวเตอร์หรือคอมพิวเตอร์ตั้งโต๊ะที่ใช้งานกันทั่วไป โดยมีหน่วยประมวลผลหรือไมโครโพรเซสเซอร์ที่คอยทำหน้าที่ประมวลผลข้อมูลจากผู้ใช้ในรูปของโปรแกรมคอมพิวเตอร์ต่างๆให้ทำงานตามที่ผู้ใช้ต้องการ

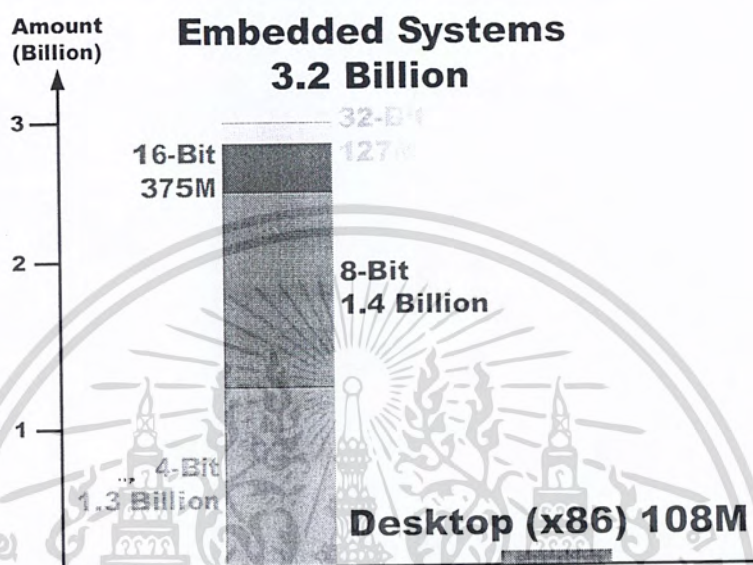
ระบบฝังตัว หรือ *Embedded system* คือ ระบบที่มีคอมพิวเตอร์ขนาดเล็กฝังอยู่กับอุปกรณ์ต่างๆ คอยทำหน้าที่ควบคุมการทำงานของระบบ โดยมีไมโครคอนโทรลเลอร์หรือไมโครโพรเซสเซอร์เป็นหัวใจหลักสำคัญคอยควบคุมการทำงานของระบบ ให้สามารถทำงานได้ตามความต้องการ เรามักจะพบเห็นและใช้งานอุปกรณ์ที่เป็นระบบฝังตัวมากมายในชีวิตประจำวันทั่วไปโดยไม่รู้ตัว ตัวอย่างเช่นในรถยนต์บางคันนั้นประกอบด้วยการทำงานประสานกันของไมโครโพรเซสเซอร์ที่ทำหน้าที่ควบคุมอุปกรณ์อำนวยความสะดวกต่างๆไม่ว่าจะเป็นถุงลมนิรภัย



รูปที่ 1 แสดงถึงตัวอย่างของระบบฝังตัวที่พบในปัจจุบัน

ระบบเบรก ระบบการจ่ายน้ำมัน ระบบควบคุมความเร็วรอบของเครื่องยนต์ ระบบกระจกไฟฟ้า ระบบเครื่องปรับอากาศ ระบบวิทยุ และระบบอื่นอีกมากมายในรถยนต์ หรือในบางครั้งอาจพบเห็นในรูปแบบของอุปกรณ์เครื่องใช้ต่างๆเช่นเครื่องซักผ้าที่มีระบบควบคุมตั้งโปรแกรมการซักผ้าในรูปแบบต่างๆ เตารีดไมโครเวฟ โทรศัพท์มือถือ หรือแม้กระทั่งในของเล่นบางชิ้น ซึ่งในบางครั้งผู้ใช้งานนั้นก็ไม่ทราบ

ว่าได้ใช้งานหรือเกี่ยวข้องกับระบบฝังตัวเหล่านี้อยู่เป็นประจำในชีวิตประจำวัน ซึ่งโลกในปัจจุบันเป็นยุคของการเปลี่ยนแปลงจากระบบคอมพิวเตอร์ธรรมดาเข้าสู่ระบบฝังตัว มากขึ้น ซึ่งจากบทความเรื่อง “Embedded Designs Move to Linux and Systems-on-a-Chip” ของวารสาร Embedded Linux Journal ประจำเดือนกรกฎาคม – สิงหาคม ปีค.ศ.2001 ได้แสดงถึงภาพตลาดของระบบฝังตัว ดังนี้



รูปที่ 2 แสดงถึงภาพตลาดของระบบฝังตัวเทียบกับคอมพิวเตอร์ตั้งโต๊ะ

ซึ่งจากภาพทำให้เห็นถึงสัดส่วนของจำนวนระบบฝังตัวที่มีใช้อยู่มากมาย เมื่อเทียบกับระบบคอมพิวเตอร์ธรรมดาโดยทั่วไป

2.2 ลักษณะของระบบฝังตัว

ระบบฝังตัวมีคุณสมบัติสำคัญดังนี้คือ

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

2.3 องค์ประกอบของระบบฝังตัว

ระบบฝังตัวนี้มักจะมีหน้าที่คอยควบคุมการทำงานของอุปกรณ์เครื่องใช้ต่างๆ ให้สามารถทำงานได้ตามฟังก์ชันการทำงานต่างๆ อย่างถูกต้อง โดยเป็นระบบที่มีคอมพิวเตอร์ขนาดเล็กฝังตัวอยู่ภายในซึ่งมีองค์ประกอบหลักดังนี้คือ

ไมโครโพรเซสเซอร์

ไมโครโพรเซสเซอร์นับเป็นหัวใจหลักที่สำคัญของระบบฝังตัว เพราะต้องทำหน้าที่ประมวลผลข้อมูลและควบคุมการทำงานของระบบทั้งหมด โดยไมโครโพรเซสเซอร์ของระบบฝังตัวนี้มีมากมายหลากหลายประสิทธิภาพการทำงาน สำหรับการนำไปใช้งานในด้านต่างๆ ตามความต้องการความสามารถในการประมวลผลข้อมูลในการทำงาน นับตั้งแต่ไมโครโพรเซสเซอร์ขนาด 4 บิตที่ใช้ในระบบฝังตัวทั่วไปในยุคต้นๆ เช่น ในวงจรรีโมตคอนโทรล เป็นต้นเรื่อยมาจนกระทั่งไมโครโพรเซสเซอร์ 32 บิต หรือ 64 บิต ในยุคปัจจุบันที่ต้องการความสามารถในการประมวลผลที่สูงเป็นพิเศษ เช่น ในกรณีของโทรศัพท์มือถือ เป็นต้น ซึ่งสามารถจัดแบ่งระดับของระบบฝังตัวตามระดับของประสิทธิภาพการทำงานของไมโครโพรเซสเซอร์ในระบบฝังตัวได้ออกเป็นสามระดับดังนี้คือ

- ระบบฝังตัวขนาดเล็ก (Low-end embedded system)

เป็นระบบฝังตัวที่มีขนาดเล็ก ใช้ไมโครโพรเซสเซอร์หรือไมโครคอนโทรลเลอร์ขนาด 4 หรือ 8 บิตเป็นตัวควบคุมการทำงาน มักใช้งานในงานที่มีฟังก์ชันการทำงานง่ายไม่ซับซ้อน ตัวอย่างของไมโครคอนโทรลเลอร์เหล่านี้ได้แก่ MCS-51, PIC และ Z80 เป็นต้น

- ระบบฝังตัวขนาดกลาง (Mid-end embedded system)

เป็นระบบฝังตัวที่มีขนาดใหญ่ขึ้นมาอีกระดับมีความสามารถในการประมวลผลสูงขึ้นมา ส่วนใหญ่จะถูกนำไปใช้ในงานที่ต้องการคุณสมบัติพิเศษ ที่ไมโครคอนโทรลเลอร์หรือไมโครโพรเซสเซอร์ขนาดเล็กไม่สามารถให้ได้ เช่น ต้องการให้งานจำนวนหน่วยความจำขนาดมากขึ้น เป็นต้น ไมโครโพรเซสเซอร์หรือไมโครคอนโทรลเลอร์เหล่านี้มักมีขนาด 16 และ 32 บิต เช่น ไมโครโพรเซสเซอร์ตระกูล x86 ของบริษัท Intel และ AMD ตระกูล ARM7 และ ตระกูล TMS320 เป็นต้น

- ระบบฝังตัวขนาดใหญ่ (High-end embedded System)

เป็นระบบฝังตัวขนาดใหญ่ที่มีความสามารถในการประมวลผลมากเป็นพิเศษ ส่วนใหญ่จะใช้งาน ไมโครโพรเซสเซอร์ที่ได้รับการออกแบบสำหรับการประมวลผลเป็นหลัก มักจะอยู่ในรูปของระบบฝังตัวที่ใช้เครื่องคอมพิวเตอร์เป็นพื้นฐาน (PC-Base) ใช้ฮาร์ดแวร์และซอฟต์แวร์สำหรับเครื่องคอมพิวเตอร์ทั่วไปเช่นระบบปฏิบัติการวินโดวส์ คอส หรือ ลินุกซ์ เป็นต้น ในการประมวลผลงานทางด้านใดด้านหนึ่งโดยเฉพาะอยู่ตลอดเวลา จึงสามารถเรียกระบบฝังตัวประเภทนี้ได้อีกแบบว่า Application Specific PC ไมโครโพรเซสเซอร์ที่ใช้งาน ก็จะเป็น 32 หรือ 64 บิตของคอมพิวเตอร์ในปัจจุบัน เช่น เพนเทียม, เพนเทียมทู เป็นต้น

หน่วยความจำ

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

ส่วนอุปกรณ์เชื่อมต่อ(Peripherals)

ส่วนอุปกรณ์เชื่อมต่อเป็นส่วนที่ใช้ในการติดต่อสื่อสารระหว่างระบบฝังตัวกับโลกภายนอก ซึ่งเห็นได้ในรูปแบบต่างกันไปตามลักษณะของงานต่างๆ ในส่วนของอินพุตของระบบ ส่วนนี้มักเป็นอุปกรณ์ตรวจวัด (sensor) ชนิดต่างๆ เช่น อุปกรณ์ตรวจจับความร้อน อุปกรณ์ตรวจจับความชื้น หรือเป็นเซ็นเซอร์ควบคุมการทำงานต่างๆ เช่น ปุ่มควบคุมตั้งเวลาและอุณหภูมิของเตาอบไมโครเวฟ หรือปุ่มกดหมายเลขและฟังก์ชันการทำงานต่างๆ ของโทรศัพท์มือถือ เป็นต้น นอกจากนี้อาจจะอยู่ในรูปของโมดูลการสื่อสารต่างๆ กับระบบอื่นๆ ภายนอก เช่น โมดูลการเชื่อมต่อแบบ USB, PCMCIA, RS-232 หรือ RS-485 เป็นต้น ในส่วนของเอาต์พุตของระบบจะเป็นอุปกรณ์ที่ทำหน้าที่เป็นส่วนติดต่อสื่อสารกับผู้ใช้งานหรือระบบอื่นๆ ภายนอกเช่น หน้าจอแอลซีดีสำหรับแสดงผลของเตาอบไมโครเวฟ หรือโทรศัพท์มือถือ เป็นต้น

ซอฟต์แวร์

ซอฟต์แวร์เป็นองค์ประกอบส่วนหนึ่งที่สำคัญในระบบฝังตัว ซึ่งเป็นตัวควบคุมกำหนดหน้าที่ฟังก์ชันการทำงานของระบบ ซึ่งภายในอาจประกอบด้วยส่วนที่เป็นตัวเริ่มต้นการทำงานและปรับแต่งระบบ (Initialization and configuration) ส่วนของระบบปฏิบัติการหรือส่วนจัดการสภาพแวดล้อมการทำงาน (Run-time environment) ส่วนของแอปพลิเคชัน และ ส่วนจัดการความผิดพลาด ซึ่งในแต่ละระบบก็ซอฟต์แวร์ภายในก็จะประกอบด้วยส่วนต่างๆ แตกต่างกันไปตามความจำเป็นของแต่ละชนิด

2.4 แอปพลิเคชันของระบบฝังตัว

ดังที่ได้กล่าวมาแล้วว่าระบบฝังตัว คือระบบที่มีคอมพิวเตอร์ขนาดเล็กอยู่ภายใน ทำงานเป็นส่วนหนึ่งของระบบ ทั้งนี้เนื่องจากความสามารถในการทำงานที่ซับซ้อน ได้อย่างถูกต้องและแม่นยำของคอมพิวเตอร์ ระบบฝังตัวจึงถูกนำไปประยุกต์ใช้ในงานต่างๆ เช่น ใช้ในอุปกรณ์เครื่องใช้ไฟฟ้าเพื่อให้ใช้งานได้ง่ายขึ้น ใช้ในระบบควบคุมเบรกและถุงลมนิรภัยของรถยนต์ซึ่งเป็นการที่ต้องการความแม่นยำสูง เป็นต้น นอกจากนี้คอมพิวเตอร์ยังตอบสนองความต้องการด้านการสื่อสารได้เป็นอย่างดีเนื่องจากมีโครงสร้างสถาปัตยกรรมเครือข่ายคอมพิวเตอร์ที่เป็นมาตรฐานและมีการใช้งานกันอย่างกว้างขวางอยู่แล้ว ทำให้ระบบฝังตัวสามารถนำไปใช้งานที่มีความต้องการด้านการติดต่อสื่อสาร เช่น ระบบควบคุมอุปกรณ์ระยะไกล ได้ เราอาจมองงานของระบบฝังตัวได้เป็นกลุ่มดังนี้ อุปกรณ์เครื่องใช้ต่างๆ ระบบควบคุมโรง

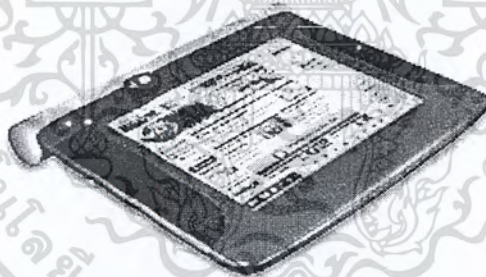
งาน อาคาร ซึ่งเป็นระบบใหญ่ที่มีการทำงานร่วมกันของอุปกรณ์จำนวนมาก และชุดพัฒนาต่างๆที่ใช้ในด้านการศึกษา วิจัย และพัฒนา

อุปกรณ์เครื่องใช้

เครื่องใช้ไฟฟ้าในบ้านทั่วไป เช่น เตาไมโครเวฟ ตู้เย็น โทรทัศน์ เครื่องซักผ้า ได้ถูกพัฒนามาอย่างต่อเนื่องให้มีความฉลาดมากขึ้น สามารถใช้งานได้ง่าย โดยใช้ไมโครคอนโทรลเลอร์หรือไมโครโปรเซสเซอร์เป็นหัวใจหลักคอยควบคุมการทำงานต่างๆเช่น ในเตาอบไมโครเวฟ ใช้ควบคุมช่วงเวลาและระดับความร้อนซึ่งมีความสัมพันธ์กันในการปรุงอาหาร เป็นต้น

ในปัจจุบันเครื่องใช้ไฟฟ้าไม่เพียงแต่ใช้ระบบคอมพิวเตอร์ในการควบคุมเท่านั้น แต่ยังมีหน้าที่สำคัญในการติดต่อสื่อสาร ไม่ว่าจะเป็นการเชื่อมต่อระยะใกล้หรือแม้กระทั่งการเชื่อมต่อระยะไกลผ่านอินเทอร์เน็ต ทำให้เครื่องใช้ต่างๆสามารถใช้ประโยชน์จากอินเทอร์เน็ตได้ ตัวอย่างเช่น เตาอบไมโครเวฟ อินเทอร์เน็ตซึ่งมีจอแอลซีดี ระบบทัชสกรีนและโมเด็มในตัว ทำให้ผู้ใช้สามารถหาสูตรปรุงอาหารจากเว็บได้และสามารถดาวน์โหลดข้อมูลสำหรับการปรุงอาหารเช่น เวลาในการปรุง ระดับความร้อน วิธีการปรุง มาใช้ได้โดยตรง โดยที่ผู้ใช้ไม่ต้องใส่ข้อมูลใดๆเพิ่มเติม

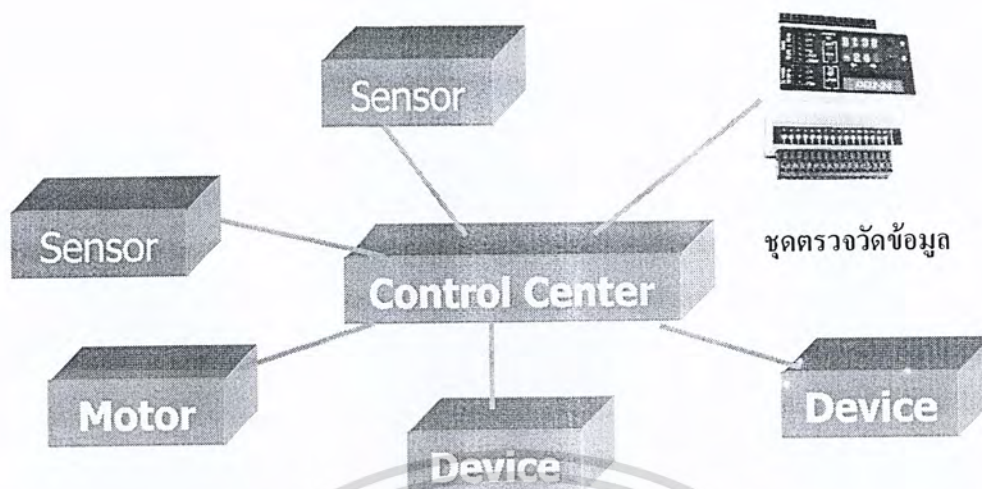
รูปแบบของระบบฝังตัวในกลุ่มนี้รวมถึง อุปกรณ์พกพาต่างๆ เช่น โทรศัพท์มือถือ อุปกรณ์จัดการข้อมูลส่วนบุคคล เป็นต้น นอกจากนี้ยังมีอุปกรณ์ที่ออกแบบมาสำหรับใช้งานอินเทอร์เน็ตเป็นหลัก โดยใช้ท่องเว็บหรือใช้รับส่งจดหมายอิเล็กทรอนิกส์โดยเฉพาะ



รูปที่ 3 อุปกรณ์ที่ออกแบบมาสำหรับใช้ท่องเว็บโดยเฉพาะ

ระบบควบคุมในโรงงาน ตึกอาคาร

ในปัจจุบันระบบระบบคอมพิวเตอร์ฝังตัวได้กลายเป็นส่วนประกอบหนึ่งของการทำงานต่างๆในหลายอุตสาหกรรมไม่ว่าจะเป็นการเกษตร การขนส่งก๊าซและน้ำมัน และในอุตสาหกรรมอื่นๆ ที่อาจจะเห็นได้ในรูปของอุปกรณ์ที่คอยควบคุมและตรวจสอบการทำงานจากระยะไกลดังเช่น SCADA (Supervisory Control and Data Acquisition) หรือเป็นอุปกรณ์ที่คอยตรวจวัดข้อมูลต่างๆจากระยะไกลที่เรียกว่า RTUs (Remote Telemetry Units) เป็นต้น อุปกรณ์เหล่านี้จะทำหน้าที่เสมือนกับเป็นคอมพิวเตอร์ขนาดเล็กทำหน้าที่คอยตรวจสอบการทำงานของชุดตรวจวัดต่างๆที่เชื่อมต่อทำการประมวลผลตามที่ได้ถูกโปรแกรมไว้และเก็บไว้หรือทำการส่งไปให้ศูนย์กลางเพื่อทำการรวบรวมประมวลผลต่อไป



รูปที่ 4 ระบบควบคุมโรงงานอัตโนมัติ

ในปัจจุบันระบบคอมพิวเตอร์ฝังตัวที่เป็นส่วนหนึ่งของระบบควบคุมในโรงงาน ได้ถูกพัฒนาอย่างต่อเนื่องเรื่อยมาเช่นจากเดิม เป็นเพียงอุปกรณ์ที่คอยตรวจวัดและสื่อสารกับศูนย์กลางผ่านทางสายสื่อสารแบบอนุกรม RS-232 ก็ได้พัฒนามาเป็นอุปกรณ์ที่มีเว็บเซิร์ฟเวอร์ขนาดเล็กภายในตัวสามารถเชื่อมต่อตรวจสอบข้อมูลและควบคุมการทำงานจากภายนอกผ่านทางเครือข่ายคอมพิวเตอร์ในรูปแบบของกราฟิกอินเทอร์เน็ตเฟสง่ายต่อการใช้งาน

งานอีกลักษณะหนึ่งที่คล้ายกับระบบควบคุมโรงงานอัตโนมัติก็คือ ระบบควบคุมอาคารอัตโนมัติ ซึ่งเป็นระบบที่ใช้คอมพิวเตอร์ รวมไปถึงการนำระบบเครือข่ายคอมพิวเตอร์เข้าใช้ในการควบคุมอุปกรณ์หรือระบบต่างๆ ภายในอาคาร เช่น ระบบไฟส่องสว่าง ระบบปรับอากาศและการไหลเวียนของอากาศ ระบบพลังงานไฟฟ้า ระบบความปลอดภัยในอาคาร เป็นต้น ทั้งนี้ระบบต่างๆ ยังได้มีการออกแบบให้มีการทำงานสอดคล้องกันเป็นระบบใหญ่อีกด้วย

ชุดพัฒนาที่ใช้ในการศึกษา วิจัยและพัฒนา

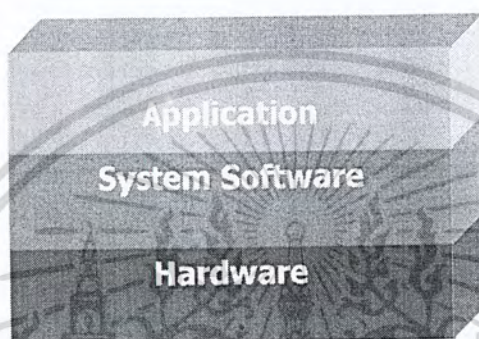
อีกรูปแบบหนึ่งของระบบฝังตัวที่พบได้ในปัจจุบันคือ อยู่ในรูปของชุดพัฒนาทางอุปกรณ์ที่มีไมโครโปรเซสเซอร์หรือไมโครคอมพิวเตอร์เป็นส่วนประกอบ เป็นลักษณะของแผ่นวงจรรวมที่ประกอบด้วยอุปกรณ์หลายอย่างใช้สำหรับพัฒนาทั้งซอฟต์แวร์และฮาร์ดแวร์ ในการสร้างผลิตภัณฑ์ขึ้นมา โดยเรียกว่า ชุดพัฒนา (Development Kit) ซึ่งจะมีทั้งฮาร์ดแวร์และซอฟต์แวร์ ที่จำเป็นต่อการพัฒนาระบบ นอกจากนี้ระบบ คอมพิวเตอร์แบบฝังที่พบเห็นในแวดวงของนักทดลองหรือนักศึกษา ไม่ว่าจะเป็นวงจรหุ่นยนต์หรืออุปกรณ์ต่างๆ ที่ได้สร้างขึ้นมาที่มีไมโครโปรเซสเซอร์หรือไมโครคอนโทรลเลอร์เป็นส่วนประกอบในระบบการทำงาน ซึ่งก็คือว่าเป็นระบบฝังตัวอย่างหนึ่งด้วยเช่นกัน

2.5 แพลตฟอร์ม

ระบบคอมพิวเตอร์นั้นมีความซับซ้อน ประกอบไปด้วยส่วนต่างๆ หลายส่วนได้แก่ โปรเซสเซอร์ หน่วยความจำ ดิสก์ เน็ตเวิร์กอินเตอร์เฟส และอินพุตเอาต์พุตอื่นๆ ทำให้ในการเขียนโปรแกรมพัฒนา

แอปพลิเคชันใดๆ ต้องมีส่วนที่จะมาจัดการกับส่วนต่างๆ เหล่านี้ทุกครั้ง ทำให้การเขียนโปรแกรมยุ่งยากในการที่จะ ต้องมาคอยคำนึงถึงการทำงานในระดับต่ำของอุปกรณ์ต่างๆ ว่าทำงานอย่างไร เป็นอุปสรรคต่อการเขียนโปรแกรมให้ถูกต้องและมีประสิทธิภาพในระยะเวลาสั้น

จากปัญหาดังกล่าวจึงต้องการสิ่งที่มาลดความซับซ้อนของฮาร์ดแวร์จากมุมมองของผู้เขียนโปรแกรม นั่นก็คือการใส่ชั้นของซอฟต์แวร์เอาไว้บนระดับของฮาร์ดแวร์เพื่อที่จะจัดการส่วนต่างๆ ของระบบและทำให้เกิดมุมมองที่ง่ายในการเขียนโปรแกรมต่อผู้พัฒนาแอปพลิเคชัน ชั้นนี้ก็คือ ซอฟต์แวร์ระบบ



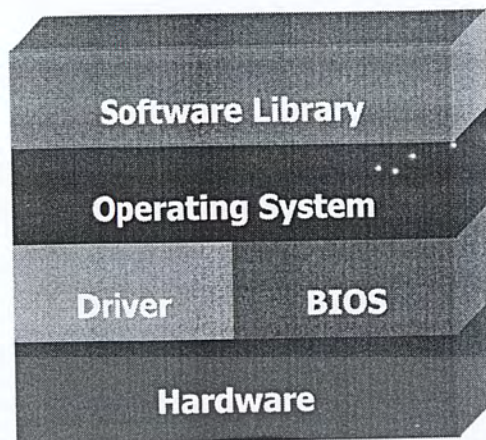
รูปที่ 5 ระบบคอมพิวเตอร์ประกอบไปด้วย ฮาร์ดแวร์ ซอฟต์แวร์ระบบ และโปรแกรมแอปพลิเคชัน

ซอฟต์แวร์โดยทั่วไปเราอาจแบ่งแบบหยาบๆ ได้เป็น 2 ชนิดคือ ซอฟต์แวร์ระบบ (system software) ซึ่งมีหน้าที่จัดการการดำเนินงานต่างๆ ของตัวคอมพิวเตอร์เอง และ ซอฟต์แวร์แอปพลิเคชัน (application software) ซึ่งเป็นโปรแกรมที่ทำงานจริงๆ ที่ผู้ใช้ต้องการ เช่น โปรแกรมสเปรดชีต โปรแกรมแท็กซ์เอดิเตอร์

ส่วนที่มีความสำคัญมากในซอฟต์แวร์ระบบก็คือ ระบบปฏิบัติการซึ่งทำหน้าที่ควบคุมทรัพยากรต่างๆ ของคอมพิวเตอร์ทั้งหมดและจัดสรรให้กับโปรแกรมแอปพลิเคชัน นอกจากนั้นซอฟต์แวร์ระบบยังต้องประกอบด้วยส่วนที่ใช้ในการพัฒนาโปรแกรมได้แก่ คอมไพเลอร์ แอสเซมเบเลอร์ และซอฟต์แวร์ไลบรารี (software library) เพื่อให้สามารถพัฒนาโปรแกรมได้ง่ายขึ้น

ในส่วนของซอฟต์แวร์ไลบรารีนั้น แบ่งออกได้เป็นส่วนที่เป็นมาตรฐาน กับส่วนที่ใช้เพื่อสนับสนุนการเขียนโปรแกรมแอปพลิเคชันในด้านต่างๆ โดยเฉพาะ ที่เรียกว่าไลบรารีแอปพลิเคชัน (library application) ทั้งนี้เรายังสามารถแบ่งไลบรารีแอปพลิเคชันออกได้เป็น 2 กลุ่ม คือ กลุ่มแนวนอน หรือกลุ่มของไลบรารีที่สนับสนุนกลุ่มแอปพลิเคชันที่มีความต้องการร่วมกัน และกลุ่มแนวตั้ง หรือกลุ่มของไลบรารีที่สนับสนุนแอปพลิเคชันด้านใดด้านหนึ่งโดยเฉพาะ

จากส่วนที่กล่าวมาทั้งหมดนั้นทั้งส่วนฮาร์ดแวร์และซอฟต์แวร์ระบบ เพื่อให้โปรแกรมแอปพลิเคชันถูกพัฒนาและทำงานได้ ประกอบกันเป็น แพลตฟอร์ม



รูปที่ 6 แพลตฟอร์มประกอบด้วยชั้นต่างๆของฮาร์ดแวร์และซอฟต์แวร์

ส่วนของฮาร์ดแวร์นั้นจะกล่าวถึง คอม86 ซึ่งเป็นบอร์ดที่ใช้เป็นฮาร์ดแวร์แพลตฟอร์มสำหรับใช้ในการพัฒนาในโครงการนี้ ซึ่งจะกล่าวรายละเอียดในบทที่ 3 ส่วนในด้านซอฟต์แวร์นั้นจะมีการกล่าวถึงระบบปฏิบัติการในบทที่ 5 ระบบปฏิบัติการของระบบฝังตัว

ลักษณะของแพลตฟอร์มในแอปพลิเคชันต่างๆ

ระบบฝังตัวนั้นมีการนำไปใช้ในแอปพลิเคชันต่างๆที่หลากหลาย ซึ่งต่างก็มีความต้องการทางด้านฮาร์ดแวร์ที่ต่างกัน จึงทำให้ซอฟต์แวร์ซึ่งทำหน้าที่ควบคุมฮาร์ดแวร์แตกต่างกันไป ด้วย จากหัวข้อ 2.4 แอปพลิเคชันของระบบฝังตัว ซึ่งแบ่งออกเป็น กลุ่มเครื่องใช้ไฟฟ้าต่างๆ ระบบควบคุมโรงงาน อาคาร และชุดพัฒนาที่ใช้ด้านการศึกษา วิจัยและพัฒนา ซึ่งจะมีลักษณะของแพลตฟอร์มของฮาร์ดแวร์พื้นฐานที่ใกล้เคียงกันได้แก่ RTC, PIO control, ฮาร์ดแวร์อินเทอร์เฟซต่างๆ โดยมีลักษณะของแพลตฟอร์มที่ต้องการในแต่ละกลุ่มดังนี้

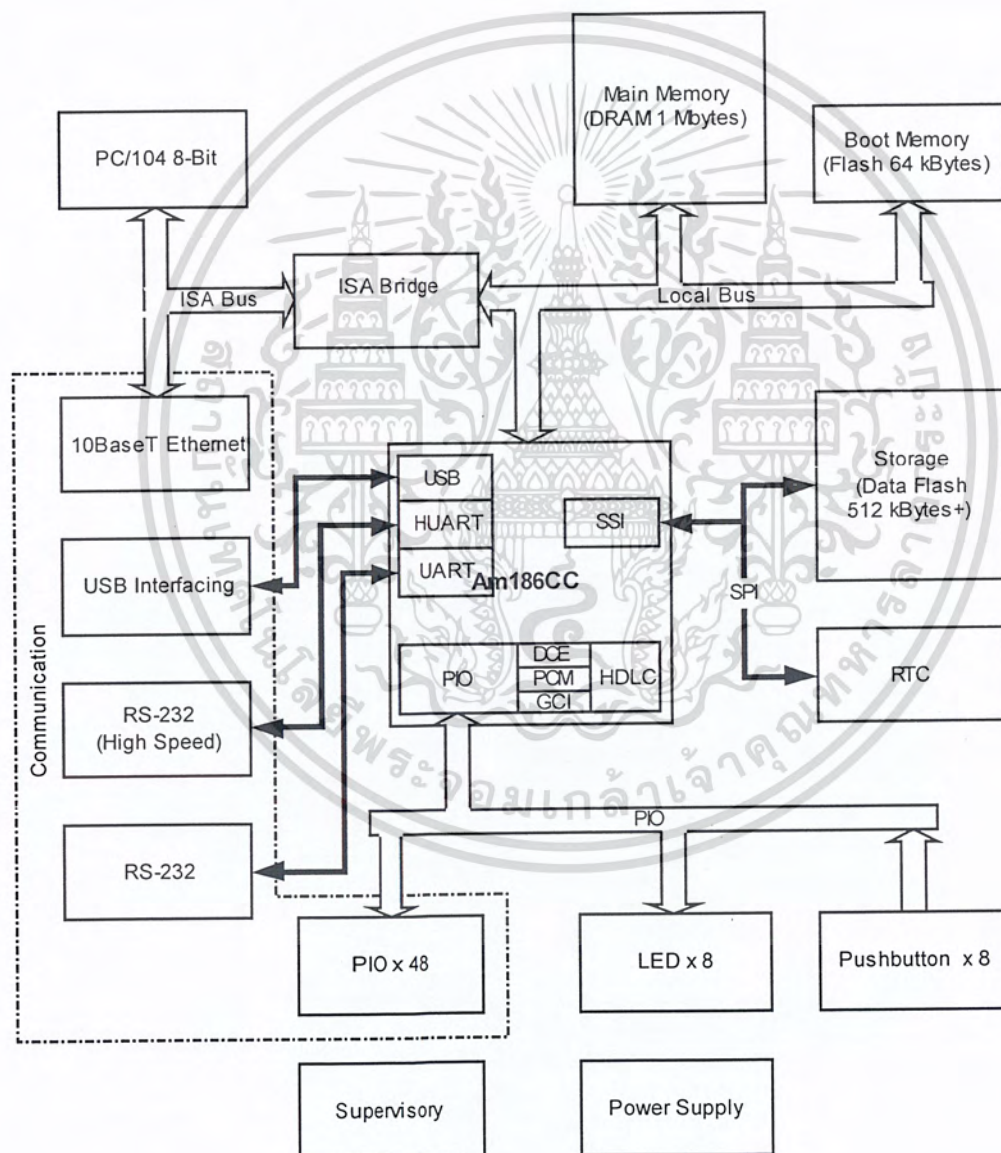
- **กลุ่มอุปกรณ์เครื่องใช้** มีอินเทอร์เฟซที่ใช้งานได้ง่าย ระบบปฏิบัติการใช้เวลาในการบูตน้อย หรืออาจจะไม่ต้องเลย ควรสนับสนุนความสามารถด้านการสื่อสารเช่น USB, TCP/IP, SNMP, SMTP, Bluetooth, HTTP, IR Communication
- **ระบบควบคุมในโรงงาน อาคาร** ความสำคัญที่จำเป็นสำหรับงานทางด้านนี้คือความสามารถทางด้านการสื่อสาร เช่น RS-232, RS-485, Ethernet หรือทางคลื่นวิทยุ เป็นต้น นอกจากนี้ยังต้องการความสามารถในการควบคุมต่างๆ เช่น PLC interface, Machine control เป็นต้น ซึ่งจะแตกต่างกันไปตามลักษณะงานที่นำไปใช้ที่สำคัญ
- **ชุดพัฒนาที่ใช้ด้านการศึกษา วิจัย พัฒนา** มีความต้องการที่แตกต่างกันไปตามลักษณะงานที่ต้องการเช่น Camera interface, Sensor control, Motor control, Ethernet communication, IR communication

บทที่ 3

บอร์ดคอม86

บอร์ดคอม86 (Com86 board) เป็นบอร์ดที่อยู่ในชุดพัฒนาคอม86 (Com86 platform) ซึ่งเป็นแพลตฟอร์มสำหรับการพัฒนาอุปกรณ์ทางด้านระบบฝังตัวโดยใช้ไมโครคอนโทรลเลอร์ขนาด 16 บิต ออกแบบและพัฒนาโดยห้องปฏิบัติการระบบควบคุมด้วยสมองกล เพื่อเป็นการสนับสนุนการเรียนรู้และพัฒนาอุปกรณ์ทางด้านระบบฝังตัวของประเทศไทย บอร์ดคอม86 มีส่วนประกอบต่าง ๆ ดังนี้

3.1 สถาปัตยกรรมทางด้านฮาร์ดแวร์

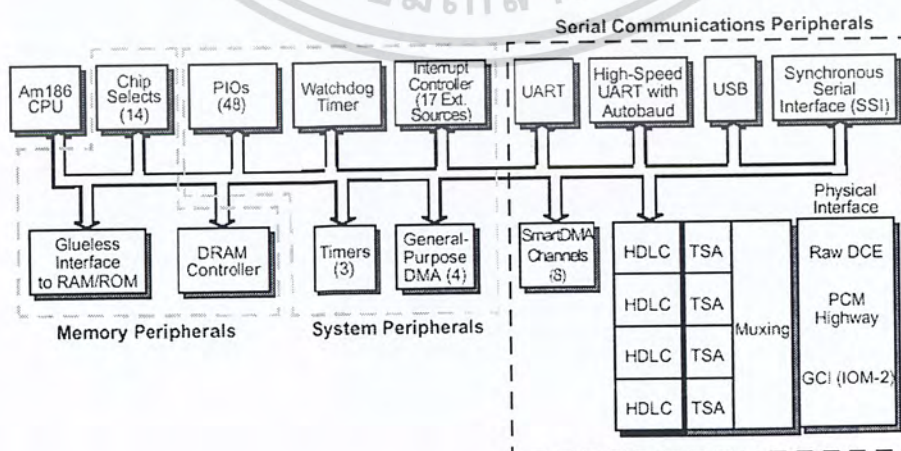


รูปที่ 7 แสดงถึงส่วนประกอบต่างๆของบอร์ดคอม86

ไมโครโปรเซสเซอร์

หน่วยประมวลผลกลางของบอร์ดคอม86 ได้ออกแบบโดยใช้ไมโครคอนโทรลเลอร์ขนาด16 บิต ในตระกูล E86 ของบริษัทเอเอ็มดีชื่อว่า Am186CC ซึ่งเป็นคอมมิวนิเคชันไมโครคอนโทรลเลอร์ (Communication microcontroller) ที่ใช้แกนหลัก (core) ของ 80186-โดยได้เพิ่มความสามารถทางด้านการสื่อสารและส่วนประกอบอื่นๆ โดยมีส่วนสำคัญที่ได้เพิ่มเข้าไปดังนี้

- ส่วน HDLC (High-level Data Link Control) สำหรับเป็นส่วนจัดการเรื่องการติดต่อสื่อสารแบบHDLCที่สามารถเชื่อมต่อกับ DCE, PCM Highway และ GCI (IOM-2) ผ่านทาง PIO โดยมี TSAs (Time Slot Assigners) สำหรับการทำงานในแบบมัลติเพล็กซ์ (Multiplex)
- USB Peripheral Controller สำหรับการเชื่อมต่อให้เป็นอุปกรณ์แบบ USB
- HUART (High-speed UART) สำหรับใช้งานในการเชื่อมต่อแบบอนุกรมความเร็วสูง โดยสามารถสื่อสารได้ในอัตราเร็วสูงสุด 460 Kbit/s สามารถตรวจสอบอัตราเร็วในการเชื่อมต่อได้อัตโนมัติ (Automatic Baud Rate Detection) และมีฮาร์ดแวร์ Handshaking
- UART (Universal Asynchronous Receiver/Transmitter) สำหรับการเชื่อมต่อแบบอนุกรม โดยสามารถสื่อสารได้ในอัตราเร็วสูงสุด 115.2 Kbit/s และมีฮาร์ดแวร์ Handshaking
- SSI (Synchronous Serial Interface) สำหรับการเชื่อมต่อสื่อสารแบบอนุกรมความเร็วสูง (25 Mbit/s) ซึ่งสามารถสื่อสารแบบ SPI ได้
- Programmable timer ขนาด 16 บิต 3 ตัว
- Hardware watchdog timer
- คีแรมคอนโทรลเลอร์ (DRAM Controller)
- อินเตอร์รัพท์คอนโทรลเลอร์ (Interrupt controller) 36 มาสต์เคเบิลอินเตอร์รัพท์



รูปที่ 8 บล็อกไดอะแกรมแสดงองค์ประกอบของไมโครคอนโทรลเลอร์ AM186CC

ไมโครคอนโทรลเลอร์ Am186CC นี้เป็นไมโครคอนโทรลเลอร์ที่มีสถาปัตยกรรมแบบ X86 มีชุดคำสั่งที่เข้ากันได้กับชุดคำสั่งของ 80286 (Real Mode) สนับสนุนการอ้างอิงหน่วยความจำหลักได้ถึง 1-เมกะไบต์ และไมโครโปรเซสเซอร์ทำงานด้วยความเร็ว 12 หรือ 24 เมกะเฮิร์ตซ์

หน่วยความจำหลัก

Main memory

หรือหน่วยความจำหลักของบอร์ดคอม 86เป็นหน่วยความจำแบบดีเรม (DRAM) ขนาด 1 เมกะไบต์เป็นหน่วยความจำสำหรับการทำงานของระบบปฏิบัติการและแอปพลิเคชัน

Boot Memory

เป็นหน่วยความจำแบบแฟลช (Flash) ที่สำหรับใช้เก็บ BIOS และโปรแกรมสำหรับเริ่มการทำงานของระบบ

Storage

บอร์ดคอม86มีส่วนเก็บข้อมูลที่เป็นหน่วยความจำแบบนอน โวลาทิล (Non-Volatile) เป็นหน่วยความจำแบบแฟลช (Flash memory) ในเบื้องต้นมีขนาด 512 กิโลไบต์ สามารถขยายได้โดยการเปลี่ยนหน่วยความจำ data flash บนบอร์ด

อุปกรณ์ต่อเชื่อม

RTC

เป็นส่วนของ Real-Time Clock มีหน้าที่สำหรับบอกวันและเวลาของระบบ

PIO

PIO (Programmable Input/Output) เป็นส่วนของการเชื่อมต่อกับอุปกรณ์ภายนอกเพื่อควบคุมหรืออ่าน สถานะต่างๆของอุปกรณ์โดยสามารถโปรแกรมได้

Push button

เป็นสวิตช์ชนิดกดปล่อยดับจำนวน 8 ตัว เชื่อมต่อกับ PIO สามารถเลือกใช้หรือไม่ใช้งานได้ตามต้องการ

LED

เป็นLED สำหรับใช้แสดงผลจำนวน 8 ดวง เชื่อมต่อกับ PIO สามารถเลือกใช้หรือไม่ใช้งานได้ตามต้องการ

Power supply

เป็นส่วนที่ทำการจ่ายกระแสไฟฟ้าในระดับแรงดันต่างๆให้แก่ระบบทั้งหมดโดยรับแรงดันจากภายนอก12โวลต์

Supervisory

เป็นส่วนของการสร้างสัญญาณรีเซตระบบและตรวจสอบแรงดันของ พาวเวอร์ ซัพพลาย

ISA Bridge

เป็นส่วนที่ทำการเชื่อมต่อ ISA บัสเข้ากับ โลคอลบัส (Local Bus) โดยทำหน้าที่จัดการ สัญญาณต่างๆใน การเชื่อมต่อสื่อสารกันระหว่างบัสทั้ง 2 ด้านให้มีสัญญาณตรงตามมาตรฐาน ของ ISA BUS

PC/104 8 bit Compatible connector

เป็นคอนเนคเตอร์สำหรับเชื่อมต่อในลักษณะเดียวกันกับมาตรฐาน PC/104 แบบ 8 บิต

10BaseT Etherne

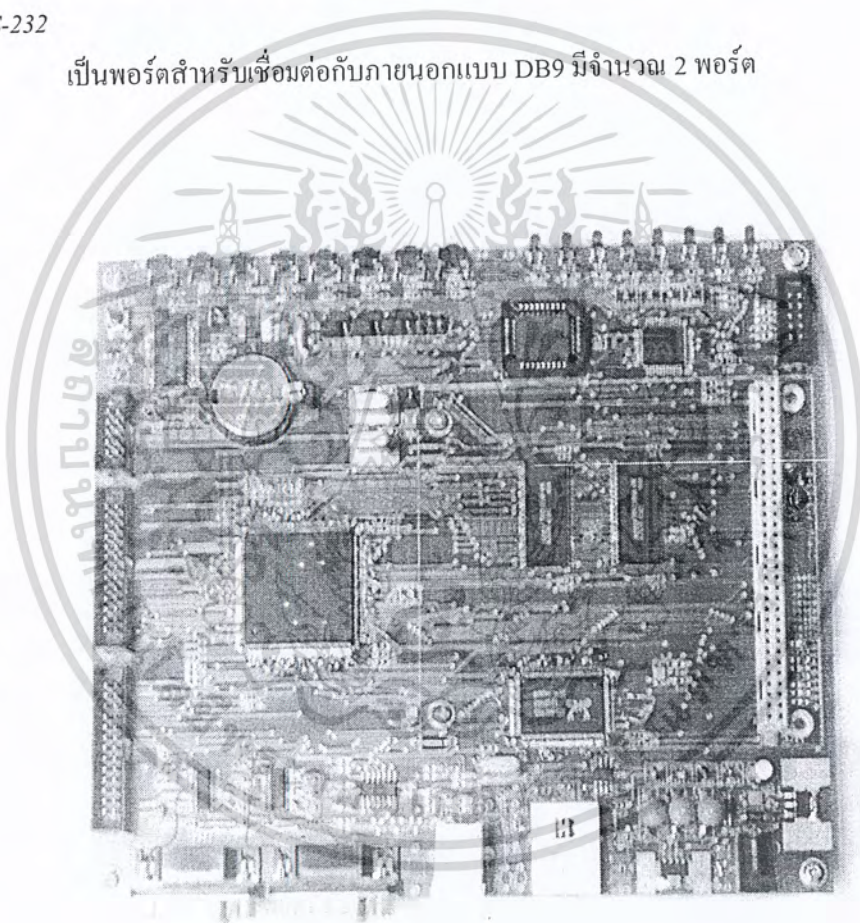
เป็นส่วนของการสื่อสารกับเครือข่าย Ethernet แบบ 10BaseT

USB Interface

เป็นส่วนเชื่อมต่อแบบ USB แบบ TYPE B

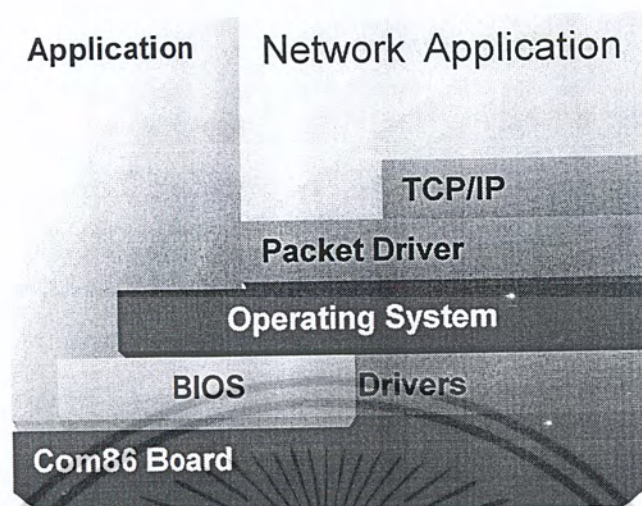
RS-232

เป็นพอร์ตสำหรับเชื่อมต่อกับภายนอกแบบ DB9 มีจำนวน 2 พอร์ต



รูปที่ 9 ภาพบอร์ดคอม86

3.2 สถาปัตยกรรมทางด้านซอฟต์แวร์



รูปที่ 10 แสดงถึงสถาปัตยกรรมทางด้านซอฟต์แวร์บนบอร์ดคอม86

จากภาพแสดงถึงลำดับชั้นต่างๆของซอฟต์แวร์ที่ทำงานอยู่บนบอร์ด คอม86โดยแต่ละชั้นมีความหมายต่างดังนี้

BIOS

เป็นโปรแกรมที่คอยควบคุมการทำงานของฮาร์ดแวร์ต่างๆบนบอร์ดคอม86 เป็นส่วนที่จัดการการสื่อสารระหว่างฮาร์ดแวร์กับแอปพลิเคชันซอฟต์แวร์ และระบบปฏิบัติการ

Driver

เป็นโปรแกรมขนาดเล็กที่ทำหน้าที่ติดต่อสื่อสารกับฮาร์ดแวร์ อื่นๆที่ไบออสไม่ได้รองรับการทำงานไว้เช่น ฮาร์ดแวร์ที่สร้างขึ้นมาเพิ่มเติมเพื่อเชื่อมต่อเข้ากับบอร์ดคอม86

Operating System

ระบบปฏิบัติการเป็นโปรแกรมที่ควบคุมการทำงานของโปรแกรมต่างๆในระบบ ทั้งในส่วนของการจัดการหน่วยความจำ เป็นส่วนสื่อสารระหว่างโปรแกรมแอปพลิเคชันของผู้ใช้และส่วนของฮาร์ดแวร์ระบบต่างๆ

Packet Driver

เป็นโปรแกรมจัดการเกี่ยวกับแพ็กเก็ตของการส่งข้อมูล สำหรับแอปพลิเคชันทางด้านการสื่อสาร

TCP/IP

เป็นโปรแกรมจัดการทางด้านโพรโตคอลทีซีพี/ไอพีสำหรับแอปพลิเคชันทางด้านการสื่อสาร

Network Application

แอปพลิเคชันทางการสื่อสาร เป็นโปรแกรมแอปพลิเคชันทางการสื่อสาร ต่างๆ เช่น โปรแกรมเว็บเซิร์ฟเวอร์ในระบบฝังตัว (Embedded web server) เป็นต้น

Application

เป็นโปรแกรมแอปพลิเคชันอื่นๆ ที่ไม่ได้ใช้ความสามารถทางการสื่อสาร เช่น โปรแกรมควบคุมการทำงานของหุ่นยนต์ หรือเครื่องจักร เป็นต้น

3.3 จุดเด่นของบอร์ด คอม86

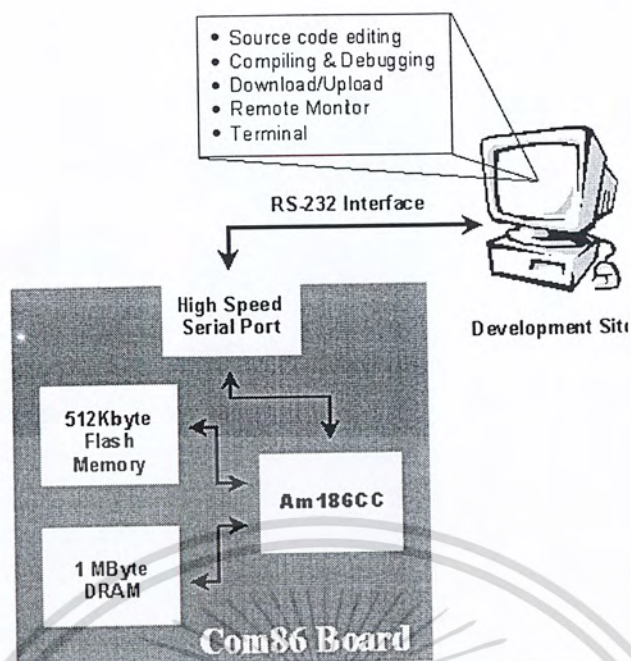
บอร์ดคอม86มีคุณสมบัติเด่นสำหรับการพัฒนาอุปกรณ์ทางด้านระบบฝังตัว ดังนี้

เป็นไมโครคอนโทรลเลอร์ที่มีความสามารถสูง

บอร์ดคอม86 เลือกใช้ไมโครคอนโทรลเลอร์ Am186 CC ของบริษัท AMD ซึ่งเป็นไมโครคอนโทรลเลอร์ขนาด 16 บิต สามารถทำงานได้ที่ ความเร็ว 25 40 และ 50 MHz ซึ่งทำให้มีความสามารถในการประมวลผลเพิ่มขึ้นมากเมื่อเทียบกับการพัฒนาโดยใช้ไมโครคอนโทรลเลอร์ขนาด 8 บิตที่นิยมใช้กันอยู่ในปัจจุบัน นอกจากนี้ยังได้รวมเอาอุปกรณ์พื้นฐานต่างๆ ที่จำเป็นเข้าไปในตัว เช่น DRAM controller, Interrupt controller, UART เป็นต้นซึ่งทำให้ง่ายต่อการออกแบบอุปกรณ์ เพื่อทำเป็นสินค้าหลังการพัฒนา นอกจากนี้ยังเพิ่มส่วนของการสื่อสารต่างๆ เข้าไป ทำให้มีความสามารถทางการสื่อสารมากขึ้นเหมาะกับการพัฒนาอุปกรณ์ที่ต้องใช้คุณสมบัติการสื่อสารต่างๆ เมื่อเทียบกับการใช้ไมโครคอนโทรลเลอร์ขนาดเล็กแบบเดิมซึ่งมีข้อจำกัดต่างๆ อยู่มากมาย

ง่ายในการเรียนรู้และพัฒนา

บอร์ดคอม86 มีสถาปัตยกรรมของไมโครคอนโทรลเลอร์แบบ x86 ซึ่งเป็นสถาปัตยกรรมเดียวกับคอมพิวเตอร์ส่วนบุคคลทั่วไป โดยมีชุดคำสั่งที่เข้ากันได้กับชุดคำสั่งของ 80286 (Real mode) ทำให้ง่ายต่อการเรียนรู้และพัฒนาโปรแกรม และยังสามารถพัฒนาและทดสอบซอฟต์แวร์ต่างๆ ได้บนคอมพิวเตอร์ ก่อนที่จะทดสอบกับบอร์ด คอม86 ทำให้สามารถพัฒนาอุปกรณ์ทางด้านระบบฝังตัวต่างๆ ได้อย่างรวดเร็ว

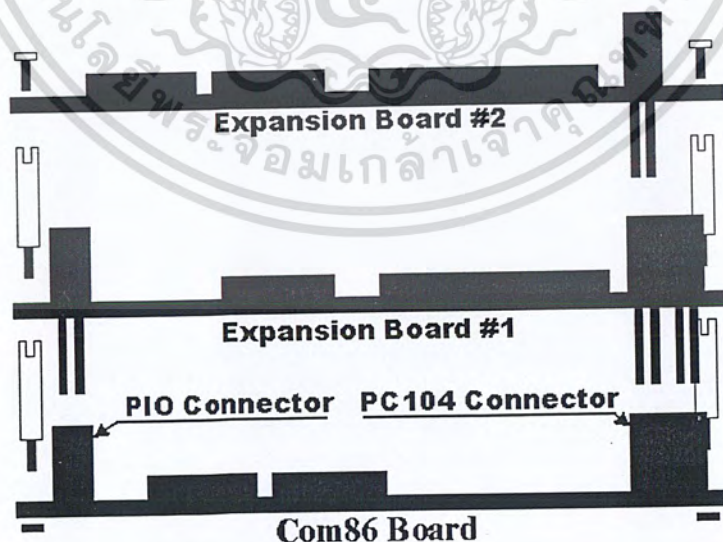


รูปที่ 11 แสดงถึงรูปแบบการพัฒนาโปรแกรมโดยใช้บอร์ดคอม86

มีความสามารถในการเพิ่มขยายได้

ในส่วนของการเพิ่มขยายของฮาร์ดแวร์สามารถทำได้โดยผ่านทางคอนเนคเตอร์ของบอร์ดที่ออก

แบบไว้ตามมาตรฐาน PC 104 ซึ่งจะมีการเชื่อมต่อคล้ายกับมาตรฐาน ISA BUS ของ คอมพิวเตอร์ส่วนบุคคล ซึ่งผู้พัฒนาสามารถออกแบบฮาร์ดแวร์ให้เป็นลักษณะของบอร์ดขยายและนำมาเชื่อมต่อกับบอร์ด คอม86 ได้ทันที และเขียนซอฟต์แวร์ไดรเวอร์สำหรับควบคุมส่วนส่วนเพิ่มเข้ามาใหม่ซึ่งจะทำงานร่วมกับระบบปฏิบัติการ ในการรองรับการทำงานของแอปพลิเคชันต่างๆ



รูปที่ 12 แสดงการเชื่อมต่อของบอร์ดขยายกับบอร์ดคอม86

มีเครื่องมือในการพัฒนาโปรแกรมมาก

เนื่องจากในการพัฒนาโปรแกรมเป็นเสมือนกับการพัฒนาบนสถาปัตยกรรม X86 ของคอมพิวเตอร์ส่วนบุคคลทั่วไป ดังนั้นจึงสามารถนำโปรแกรมเครื่องมือต่างๆที่ใช้พัฒนาโปรแกรมสำหรับคอมพิวเตอร์ส่วนบุคคลมาใช้งานได้มากมายและเป็นที่คุ้นเคยของนักพัฒนาทำให้สามารถพัฒนาอุปกรณ์ได้อย่างรวดเร็ว ลดต้นทุนในการพัฒนาต่างๆได้เป็นจำนวนมาก

3.4 แอปพลิเคชันเป้าหมายของชุดพัฒนา (Target Application)

จากความสามารถของไมโครคอนโทรลเลอร์ของบอร์ดคอม86 ทำให้การประยุกต์ใช้งานสามารถทำได้ซับซ้อนมากขึ้น โดยมีตัวอย่างแอปพลิเคชันต่างๆที่เหมาะสมแก่การพัฒนาโดยบอร์ดคอม86 ดังนี้

Industrial control and Automation

ตัวอย่างของอุปกรณ์ควบคุมทางด้านโรงงานเช่นการควบคุมเครื่องจักรจากระยะไกล การตรวจวัดสภาพแวดล้อมต่างๆภายในโรงงานและส่งกลับไปยังศูนย์กลางเป็นต้น

USB peripheral

อุปกรณ์ USB สามารถพัฒนาโดยใช้บอร์ดคอม86 โดยสามารถเชื่อมต่อเข้ากับคอมพิวเตอร์ ด้วยความเร็ว 12 Mbps โดยไม่ต้องมีวงจรเพิ่มเติม

Embedded web server

Embedded web server สามารถสร้างได้โดยใช้บอร์ดคอม86โดยไม่ต้องใช้อุปกรณ์อื่นเพิ่มเติม โดยสามารถสื่อสารกับภายนอกผ่านทาง Ethernet port ที่อยู่บนบอร์ดคอม86

Robotics

บอร์ดคอม86 มีไมโครคอนโทรลเลอร์ที่มีประสิทธิภาพสูงในการทำงาน การพัฒนาซอฟต์แวร์สามารถพัฒนาได้ซับซ้อนมากยิ่งขึ้น การเชื่อมต่อขยายทางด้านฮาร์ดแวร์ต่างๆสามารถทำได้โดยง่าย จึงเหมาะสำหรับการพัฒนาหุ่นยนต์ที่มีการทำงานที่ซับซ้อน

นอกจากนี้ตัวไมโครคอนโทรลเลอร์ยังรองรับการนำไปสร้างเป็นอุปกรณ์ต่างๆที่เกี่ยวข้องกับงานทางด้านการสื่อสารต่างๆ เช่น ISDN Modem and Terminal Adaptor, Low-end Router, Line card application, xDSL application, Digital corded phone เป็นต้นซึ่งสามารถบอร์ดคอม86ในการออกแบบพัฒนาได้เช่นกัน

บทที่ 4

ระบบปฏิบัติการสำหรับระบบฝังตัว

4.1 ระบบปฏิบัติการสำหรับระบบฝังตัว

ในการพัฒนาอุปกรณ์ที่เป็นระบบคอมพิวเตอร์ฝังตัวนั้น ระบบปฏิบัติการเป็นองค์ประกอบสำคัญ ส่วนหนึ่งของระบบซอฟต์แวร์ทั้งหมด เพราะเป็นส่วนที่ทำให้การพัฒนาซอฟต์แวร์แอปพลิเคชันสำหรับงานต่างๆทำได้รวดเร็วยิ่งขึ้น สามารถพัฒนาซอฟต์แวร์แอปพลิเคชันที่มีความซับซ้อนได้ง่ายยิ่งขึ้น สามารถดูแลแก้ไขได้ง่าย สามารถนำโค้ดของแอปพลิเคชันที่มีอยู่กลับมาใช้ใหม่ได้ในรูปของไลบรารี ส่งผลให้ระยะเวลาในการพัฒนาผลิตภัณฑ์ลดลง ต้นทุนได้มาก และทำให้เวลาในการเข้าสู่ตลาดของผลิตภัณฑ์สั้นลงในขณะที่ยังคงรักษาคุณภาพของผลิตภัณฑ์ไว้

คุณสมบัติของระบบปฏิบัติการที่ใช้ในระบบฝังตัว

ระบบปฏิบัติการในระบบฝังตัว ในปัจจุบันควรมีคุณสมบัติต่างๆดังนี้คือ

- *Reliability* คือ มีความคงทน เชื่อถือได้
- *Small kernel* คือ มีเคอร์เนลขนาดเล็ก เนื่องจากต้องนำไปใช้ในระบบที่มีข้อจำกัดทางด้านฮาร์ดแวร์ ทั้งในด้านหน่วยความจำที่ใช้เก็บข้อมูล
- *Modularity* คือ ระบบปฏิบัติการสามารถแบ่งออกเป็นโมดูลย่อยๆ ได้ เพื่อเหมาะสมกับการนำไปปรับใช้กับแต่ละงาน
- *Scalability* คือ ความสามารถในการรองรับการเพิ่มหรือลดขนาดของฮาร์ดแวร์
- *Multiplatform* คือ สามารถทำงานได้บน ไมโครโพรเซสเซอร์ หรือไมโครคอนโทรลเลอร์ ได้หลายตระกูล
- *Networking Support* คือ สนับสนุนความสามารถด้านเน็ตเวิร์ก
- *Power Management* คือ การสนับสนุนความสามารถในการประหยัดพลังงาน
- *Real-Time Support* คือ การสนับสนุนการทำงานแบบเรียลไทม์ สำหรับใช้ในบางลักษณะงานที่ต้องการ

4.2 ประเภทของระบบปฏิบัติการที่ใช้ในระบบคอมพิวเตอร์ฝังตัว

ระบบปฏิบัติการในระบบฝังตัว สามารถจัดแบ่งออกเป็นสองประเภทหลักๆ ได้ดังนี้ คือ

Real-Time Operating System

คือระบบปฏิบัติการที่รองรับการทำงานตามเงื่อนไขของเวลา ซึ่งต้องสามารถรองรับการทำงานที่ต้องการการตอบรับ ในเวลาที่ถูกต้องตามเหตุการณ์ที่เข้ามาจากภายนอก ตัวอย่างระบบปฏิบัติการเหล่านี้ในปัจจุบัน ได้แก่ QNX, VxWorks, LynxOS, RT-Kernel, Itron, pSOS เป็นต้น

Non Real-Time Operating System

คือระบบปฏิบัติการอื่นๆที่ไม่ได้เป็นระบบที่รองรับการทำงานแบบเรียลไทม์ แต่เป็นระบบ

ปฏิบัติการที่ใช้ในงานทางด้านระบบคอมพิวเตอร์ฝังตัวทั่วไปเช่น Embedded Linux, RomDos, ELKS เป็นต้น

4.3 ลักษณะของระบบปฏิบัติการที่เหมาะสมกับชุดพัฒนาคอม86

จากสถาปัตยกรรมและแอปพลิเคชันที่จะพัฒนาจากชุดพัฒนาคอม86 ทำให้สามารถกำหนดคุณสมบัติของระบบปฏิบัติการที่เหมาะสมจะนำมาใช้งานกับชุดพัฒนาคอม86ได้ดังนี้คือ

1. ควรเป็นระบบปฏิบัติการที่สามารถเรียนรู้การใช้งานได้อย่างรวดเร็ว
2. มีความเชื่อถือได้ (Reliability)
3. มีขนาดเล็กสามารถทำงานได้ในสถานะที่หน่วยความจำในการทำงานและการเก็บข้อมูลมีขนาดจำกัด
4. สามารถปรับแต่งได้ตามลักษณะของงานที่ชุดพัฒนาคอม86 ถูกนำไปใช้
5. สามารถรองรับการขยายขีดความสามารถทางด้านการฮาร์ดแวร์ในอนาคต
6. ควรสนับสนุนการทำงานทางด้านเน็ตเวิร์ก
7. รองรับการทำงานของแอปพลิเคชันที่พัฒนาโดยภาษาระดับสูง เช่น ภาษาC หรือ C++ เป็นต้น
8. ควรสนับสนุนการทำงานแบบมัลติทาสก์

4.4 ระบบปฏิบัติการและเทคโนโลยีที่น่าสนใจสำหรับนำมาพัฒนาในโครงการ

4.4.1 ระบบปฏิบัติการมินิ็กซ์

มินิ็กซ์ (Minix) ย่อมาจาก มินิ-ยูนิกซ์ (Mini-Unix) ถ้าแปลตามตัวแปลได้ว่ายูนิกซ์ฉบับย่อ มินิ็กซ์เป็นระบบปฏิบัติการที่เลียนแบบระบบปฏิบัติการยูนิกซ์ (Unix clone) ซึ่งแจกจ่ายฟรีพร้อมกับซอร์สโค้ดของระบบปฏิบัติการ มินิ็กซ์มีขนาดเล็ก ใช้สถาปัตยกรรมเคอร์เนลแบบไมโครเคอร์เนล (Microkernel) และเอกสารคู่มือการใช้งานอย่างครบครัน เหมาะสำหรับผู้ที่ต้องการใช้งานระบบปฏิบัติการที่มีลักษณะคล้ายกับยูนิกซ์บนเครื่องคอมพิวเตอร์ของตนเองเพื่อใช้ในการศึกษาการทำงานภายในของระบบปฏิบัติการ อีกทั้งยังเหมาะกับผู้ที่ไม่คุ้นเคยกับการใช้งานระบบปฏิบัติการก็สามารถเรียนรู้ได้ภายในเวลาไม่นาน

มินิ็กซ์เกิดจากความต้องการให้การศึกษาวิชาระบบปฏิบัติการคอมพิวเตอร์สามารถเรียนรู้ได้จากการศึกษาโครงสร้างการทำงานของระบบปฏิบัติการจริง ซึ่งโดยปกติการเรียนวิชาระบบปฏิบัติการคอมพิวเตอร์จะมีเฉพาะหลักการและทฤษฎีของระบบปฏิบัติการเท่านั้น ปัญหาที่เกิดขึ้นคือศึกษาได้เฉพาะในทฤษฎี ไม่ได้มีตัวอย่างให้ใช้ในการศึกษากันมากนัก เนื่องจากระบบปฏิบัติการในสมัยก่อนไม่ได้มีการเผยแพร่ซอร์สโค้ดทำให้การศึกษาและพัฒนาระบบปฏิบัติการ

เป็นไปได้ช้าและยากลำบาก ดังนั้น ศาสตราจารย์แอนดรู ทาเนนบาม (Prof. Andrew S. Tanenbaum) ที่มหาวิทยาลัยเวอร์จิ (Vrije University) ประเทศเนเธอร์แลนด์จึงได้พัฒนาระบบปฏิบัติการมินิกซ์ขึ้นเพื่อใช้ในการเรียนวิชาระบบปฏิบัติการ และได้มีการนำมาใช้ประกอบการเรียนวิชานี้กันอย่างแพร่หลายในเวลาต่อมา

โค้ดของระบบปฏิบัติการมินิกซ์ไม่ได้มีส่วนหนึ่งส่วนใดเหมือนโค้ดกับระบบปฏิบัติการยูนิกซ์ที่มีลิขสิทธิ์อื่นๆ อย่างเช่น AT&T แต่ได้เขียนขึ้นมาใหม่ทั้งหมด ทั้งเคอร์เนล คอมไพเลอร์ โลบารี่ใช้งานของระบบ และโปรแกรมอรรถประโยชน์ ทำให้มินิกซ์สามารถเผยแพร่ซอร์สโค้ดของระบบปฏิบัติการได้ ซึ่งสามารถหาดาวน์โหลดได้จากอินเทอร์เน็ต

มินิกซ์ได้รับการพัฒนาให้มีความสามารถเพิ่มขึ้นอย่างต่อเนื่องในช่วงหลายปีที่ผ่านมา ทำให้

มินิกซ์ได้รับการพัฒนาออกมาหลายเวอร์ชัน ซึ่งในปัจจุบัน¹ มีสองเวอร์ชันคือ

- มินิกซ์เวอร์ชัน 2.0 สนับสนุนการทำงานบนชิพยูนิกซ์ในตระกูลอินเทล (Intel) ตั้งแต่ 8088 ขึ้นไป และ
- มินิกซ์เวอร์ชัน 1.5 ซึ่งนอกเหนือจากสนับสนุนชิพยูนิกซ์ในตระกูลอินเทลแล้ว ยังสามารถทำงานได้ในเครื่องตระกูลแมคอินทอช (Macintosh) อาตารี (Atari) อะไมกา (Amiga) และสปาร์ค (SPARC)

ในปี 1987 เป็นจุดเริ่มต้นของมินิกซ์เวอร์ชันแรก 1.0 ซึ่งสามารถทำงานได้บนเครื่องคอมพิวเตอร์ที่มีความเร็วเพียง 4.77 เมกะเฮิรตซ์ มีหน่วยความจำ 256 กิโลไบต์ มีฟลอปปี้ดิสก์ขนาด 360 กิโลไบต์และไม่มีฮาร์ดดิสก์ ซึ่งคงหาเครื่องคอมพิวเตอร์รุ่นเดียวกันนี้ในปัจจุบันได้ยากเต็มที มินิกซ์ได้รับการพัฒนาอย่างต่อเนื่อง จนในเวอร์ชันปัจจุบันหรือมินิกซ์เวอร์ชัน 2.0 มินิกซ์เวอร์ชัน 2.0 สามารถคอมไพล์ให้มีได้ทั้ง 16 บิตหรือ 32 บิต ถ้าคอมไพล์เป็นแบบ 32 บิตต้องนำไปใช้ในเครื่องที่มีชิพยูนิกซ์ในตระกูลอินเทลตั้งแต่ 386 หรือ 486 ขึ้นไป

มินิกซ์เวอร์ชัน 2.0 ได้เพิ่มขีดความสามารถไปจากมินิกซ์เวอร์ชันแรกหรือเวอร์ชัน 1.0 ไปมาก มินิกซ์เวอร์ชัน 2.0 ยังคงสามารถทำงานได้บนเครื่องตั้งแต่ 8088 ขึ้นไป สนับสนุนอุปกรณ์ต่างๆ มากขึ้น สนับสนุนไครเวอร์อีเทอร์เน็ต โมเด็ม สนับสนุนการทำงานในแบบ 32 บิตของหน่วยความจำโปรเทกต์เค็ดโหมด (Protected mode) ระบบไฟล์สนับสนุนดิสก์ที่มีขนาดใหญ่ระดับกิกะไบต์ได้ และที่สำคัญคือเพิ่มความสามารถด้านเครือข่าย โดยสนับสนุนโพรโทคอล TCP/IP และเพิ่มโปรแกรมอรรถประโยชน์ทั้งด้านเครือข่าย ด้านดูแลระบบ ด้านใช้งานทั่วไป ซึ่งสามารถนำไปประยุกต์ใช้ได้อย่างกว้างขวาง

¹ มินิกซ์เวอร์ชัน 2.0 เปิดตัวเมื่อปี 1996 ในขณะที่เขียนปฏิญานพนธ์นี้ มินิกซ์ได้ออกเวอร์ชันใหม่มาแล้วคือ 2.0.3 แต่เป็นเพียงการแก้บั๊กเล็กน้อยของระบบปฏิบัติการเท่านั้น ดังนั้นในการอ้างอิงถึงระบบปฏิบัติการมินิกซ์ในปฏิญานพนธ์นี้จะยึดตามมินิกซ์เวอร์ชัน 2.0.2 เป็นหลัก

เคอร์เนลของมินิกซ์เป็นแบบไมโครเคอร์เนล ไมโครเคอร์เนลเป็นโครงสร้างของเคอร์เนลที่ทำให้สามารถเพิ่มเติมหรือลดความสามารถของเคอร์เนลได้โดยไม่จำเป็นต้องทำการคอมไพล์เคอร์เนลใหม่ มีองค์ประกอบในส่วนหน่วยควบคุมหน่วยความจำและระบบไฟล์ของระบบแยกเป็นอิสระจากส่วนเคอร์เนลของระบบปฏิบัติการ เป็นผลให้เคอร์เนลจริงๆ ของระบบไม่มีความซับซ้อนแต่อย่างใด ทำให้ง่ายต่อการศึกษาทำความเข้าใจ นอกจากนี้ระบบยังมีความเสถียรภาพมากกว่าโครงสร้างในแบบเก่าซึ่งเรียกโครงสร้างแบบเก่าว่าเคอร์เนลแบบเคอร์เนลโมโนลิธิก (Monolithic Kernel) เช่นในยูนิกซ์ ลินุกซ์ (Linux) และดอส (Dos) เป็นต้น โครงสร้างแบบโมโนลิธิกนี้ ถ้าต้องการเพิ่มหรือลดความสามารถของเคอร์เนลต้องทำการคอมไพล์ใหม่ ไมโครเคอร์เนลได้นำมาใช้เป็นโครงสร้างของระบบปฏิบัติการใหม่ๆ ในปัจจุบันอย่างแพร่หลายเช่น มัค (Mach) คอรัส (Chorus) อโมบา (Amoeba) และระบบปฏิบัติการที่นิยมใช้กันอย่างแพร่หลายในปัจจุบันอย่างไมโครซอฟท์วินโดวส์ (Microsoft Windows) เป็นต้น

ถ้าเปรียบเทียบมินิกซ์กับลินุกซ์ ต้องกล่าวว่าต่างกันตั้งแต่จุดเริ่มต้นของแนวความคิดในการออกแบบ ในการออกแบบมินิกซ์นั้นมีแนวความคิดหลักที่ต้องการออกแบบเพื่อให้นักศึกษาใช้ประกอบการเรียนวิชาระบบปฏิบัติการ แนวคิดหลักนี้ทำให้มีเงื่อนไขในการออกแบบที่ต้องออกแบบระบบปฏิบัติการมินิกซ์ให้มีขนาดเล็กและง่ายต่อการศึกษาทำความเข้าใจมากที่สุด ส่วนลินุกซ์นั้นออกแบบมาเพื่อให้ใช้เป็นระบบปฏิบัติการที่ใช้งานได้ทั้งเป็นเซิร์ฟเวอร์และไคลเอนต์ ทำให้มีความซับซ้อนมากกว่าไม่เหมาะที่จะนำมาใช้ในการศึกษา แต่ถึงกระนั้นก็ตาม นับตั้งแต่ได้มีการพัฒนามินิกซ์อย่างต่อเนื่อง มีแอปพลิเคชันที่สามารถทำงานได้บนมินิกซ์มากขึ้น ทำให้นอกจากนำมินิกซ์มาใช้ประกอบการศึกษาวิชาระบบปฏิบัติการแล้ว ยังได้มีการนำมินิกซ์มาใช้ในวงการอุตสาหกรรมคอมพิวเตอร์อีกด้วย

4.4.1.1 ลักษณะเด่นของมินิกซ์เวอร์ชัน 2.0

- สนับสนุนมัลติโปรแกรมมิ่ง หลายโปรแกรมสามารถทำงานได้พร้อมกันในเวลาเดียวกัน
- สนับสนุนทำงานในโปรเทกต์ดโหมด ได้ในซีพียู 286, 386, 486 และเพนเทียม
- สนับสนุนหน่วยความจำเพิ่มขยายได้มากที่สุด 16 เมกะไบต์ในซีพียู 286 และ 4 กิกะไบต์ใน 386, 486 และเพนเทียม
- สนับสนุนพอร์ตอนุกรม RS-232 และเทอร์มินอลอิมูเลชัน (Terminal Emulation) เช่นเคอร์มิต (Kermit) แซดโมเด็ม (zmodem) เป็นต้น
- สนับสนุนระบบประมวลผลแบบกระจายผ่านอีเทอร์เน็ต
- สนับสนุนไครเวอร์อีเทอร์เน็ตของการ์ด SMC/WD80*3, NE1000/NE2000 และ 3Com 3C503
- ไครเวอร์ซีดีรอม และพรีนเตอร์
- รองรับผู้ใช้งานเข้ามาใช้งานในเวลาเดียวกันได้มากที่สุด 3 คน

- ฟังก์ชันเรียกเข้าหาระบบ (System call) สนับสนุนตามมาตรฐาน POSIX เวอร์ชัน P1003.1 และเวอร์ชัน P1003.2
- ซอร์สโค้ดของระบบปฏิบัติการภาษาซี ทั้งเคอร์เนล โปรแกรมอรรถประโยชน์ และไลบรารี
- คอมไพเลอร์ภาษาซีสนับสนุนตามมาตรฐาน ANSI
- สนับสนุนการเชื่อมต่อเครือข่ายผ่านโพรโทคอล TCP/IP
- เชลล์พื้นฐานสนับสนุนตามรูปแบบบอร์นเชลล์ (Bourne shell)
- เอดิเตอร์พื้นฐานที่จำเป็นต่อการใช้งานมีทั้ง emacs, vi, ex และ ed
- โปรแกรมอรรถประโยชน์มากกว่า 200 โปรแกรมและไลบรารีพื้นฐานมากกว่า 300 ไลบรารี
- คู่มือประกอบการใช้ระบบปฏิบัติการครบทุกคำสั่ง (Manual Page)

4.4.1.2 ฮาร์ดแวร์อย่างต่ำที่ต้องการ

- เครื่องคอมพิวเตอร์ที่คอมแพคทีเบิ้ลกับ IBM PC AT, XT และ PS/2
- ซีพียู 8088, 286, 386, 486 หรือเพนเทียมขึ้นไป
- หน่วยความจำอย่างต่ำขนาด 512 กิโลไบต์ ขนาดที่เหมาะสมสำหรับมินิกซ์แบบ 16 บิตคือ 640 กิโลไบต์ และสำหรับมินิกซ์แบบ 32 บิตคือ 2 เมกะไบต์
- ฮาร์ดดิสก์ขนาด 30 เมกะไบต์ สำหรับลงมินิกซ์ทั้งหมด รวมซอร์สโค้ดและโปรแกรมอรรถประโยชน์
- จอแบบ CGA, EGA, VGA, monochrome หรือเออร์คิวิลิสการ์ด
- สนับสนุนฟลอปปีทั้งขนาด 5.25" และ 3.5"

4.4.1.3 การใช้ระบบปฏิบัติการมินิกซ์ร่วมกับระบบปฏิบัติการอื่น

มินิกซ์มีจุดประสงค์ในการออกแบบเพื่อใช้ในการศึกษาหลักการและทฤษฎีของระบบปฏิบัติการ ทำให้ต้องออกแบบมินิกซ์ให้ง่ายและมีขนาดเล็กเพื่อให้เหมาะสมต่อการศึกษาและสามารถเรียนรู้ได้ภายในเวลาไม่นาน ดังนั้นมินิกซ์จึงไม่ได้มีความสามารถในการเข้าถึงระบบไฟล์ของระบบปฏิบัติการอื่นๆ ได้เช่น ระบบไฟล์แบบ FAT ในดอส หรือระบบไฟล์แบบ EXT2 ในลินุกซ์เป็นต้น จะเข้าถึงได้แต่เพียงระบบไฟล์ของมินิกซ์เองเท่านั้น

พื้นฐานการใช้งานระบบไฟล์ของระบบปฏิบัติการยูนิกซ์ซึ่งรวมไปถึงมินิกซ์ด้วย เพราะใช้แนวความคิดอย่างเดียวกัน เรียกวิธีการในการเพิ่มเติมหรือลดขนาดของระบบไฟล์ว่า การเมาดิง (Mounting) หมายความว่าก่อนที่จะใช้ระบบไฟล์ใดๆ ไม่ว่าจะเป็นระบบไฟล์ของมินิกซ์เอง ต้องเมาดิงก่อนจึงสามารถเข้าไปใช้งานอ่านหรือเขียนกับระบบไฟล์ใดๆ ได้ นั่นหมายความว่าก่อนที่จะอ่านเขียนกับระบบไฟล์ใดต้องทำการเมาดิงก่อนจากนั้นจึงสามารถอ่านหรือเขียนได้ จากที่

กล่าวไปตอนต้นบอกว่ามินิกซ์ไม่สามารถเข้าถึงระบบไฟล์ของระบบปฏิบัติการอื่นนอกจากมินิกซ์เอง แต่การเข้าถึงไม่ได้ไม่ได้หมายความว่าอ่านหรือเขียนระบบไฟล์ของระบบปฏิบัติการอื่นไม่ได้ ซึ่งตัวมินิกซ์เองไม่สามารถเข้าถึงได้แต่สามารถอ่านหรือเขียนได้โดยใช้โปรแกรมอรรถประโยชน์ที่มาพร้อมกับมินิกซ์ ในขณะนี้สามารถอ่านระบบไฟล์ของ คอส, ไมโครซอฟท์ วินโดวส์, ISO-9660 ซึ่งเป็นระบบไฟล์บนซีดีรอม และระบบไฟล์ของลินุกซ์ก็สามารถอ่านได้เช่นเดียวกัน

- ระบบไฟล์ของคอสและวินโดวส์ใช้โปรแกรม dosdir, dosread และ doswrite อ่านหรือเขียนได้
- ระบบไฟล์ของซีดีรอมหรือ ISO-9660 ใช้โปรแกรม isoinfo, isodir และ isoread อ่านหรือเขียนได้
- ระบบไฟล์ของลินุกซ์ใช้โปรแกรม e2cp, e2cat และ e2fs ในอ่านหรือเขียนได้

4.4.1.4 ลิขสิทธิ์ของมินิกซ์

แม้ว่ามินิกซ์จะเปิดเผยซอร์สโค้ดของระบบปฏิบัติการแต่มินิกซ์เป็นโปรแกรมที่อยู่ภายใต้ลิขสิทธิ์ ซึ่งผู้เป็นเจ้าของลิขสิทธิ์คือ Prantice-Hall ซึ่งในปี 1997 ได้เปลี่ยนลิขสิทธิ์อีกครั้งถึงแม้เจ้าของลิขสิทธิ์เหมือนเดิมแต่ข้อบ่งชี้ในการนำมินิกซ์ไปใช้ ได้กว้างขวางขึ้นจากตัวลิขสิทธิ์ที่เปลี่ยนไป เพื่อให้สามารถนำมินิกซ์ไปใช้งานได้กว้างขวางมากขึ้นทั้งระดับบนและระดับล่าง ในระดับล่างแล้วจะเห็นว่าความที่มินิกซ์มีขนาดเล็กทำให้เหมาะสำหรับนำไปใช้ในระบบฝังตัว ซึ่งลิขสิทธิ์ใหม่นี้อนุญาตให้ผู้ที่นำมินิกซ์ไปใช้ในผลิตภัณฑ์ของตนนั้นสามารถที่จะปกปิดซอร์สโค้ดของมินิกซ์ที่ได้รับการนำไปประยุกต์เข้ากับผลิตภัณฑ์ของตนได้ หรือจะเปิดเผยซอร์สโค้ดก็ได้ ซึ่งจะเห็นว่าลิขสิทธิ์ใหม่นี้มีจุดประสงค์เพื่อให้สามารถนำมินิกซ์ไปใช้ให้เกิดผลทางการค้าได้มากขึ้น

4.4.2 ระบบปฏิบัติการฟรีดอส (Freedos Operating System)

จุดมุ่งหมายของโครงการระบบปฏิบัติการฟรีดอสคือการสร้างระบบปฏิบัติการที่เลียนแบบระบบปฏิบัติการคอสของไมโครซอฟท์หรือรู้จักกันในชื่อเอ็มเอสดอส (MS-DOS) โครงการนี้เริ่มต้นขึ้นในปี 1994 โดยนายจิม ฮอลล์ (Jim Hall) เนื่องจากความหลงใหลในระบบปฏิบัติการนี้ และในขณะนั้น ไมโครซอฟท์ได้ตัดสินใจหยุดพัฒนาคอสหันไปพัฒนาวินโดวส์อย่างจริงจัง จึงได้มีการรวมกลุ่มผู้ที่สนใจและ ชอบดอส ในที่สุดก็เริ่มต้นเป็นโครงการฟรีดอสขึ้นมา

ฟรีดอสไม่ได้ใช้หรือเอาส่วนหนึ่งส่วนของเอ็มเอสดอสหรือไมโครซอฟท์ดอส แต่ใช้ในการอ้างอิงเพื่อให้สามารถนำแอปพลิเคชันที่ทำงานบนเอ็มเอสดอสมาใช้งานฟรีดอสได้ เป้าหมายของฟรีดอสคือการเปิดเผยซอร์สโค้ดของระบบปฏิบัติการและคอมแพคทีเบิลกับเอ็มเอสดอส ฟรีดอสใช้ลิขสิทธิ์ของ กนู (GNU) ซึ่งหมายความว่าสามารถนำซอร์สโค้ดไปได้โดยไม่ต้อง

เสียค่าใช้จ่ายใดๆ ทั้งเคอร์เนลและโปรแกรม นอกจากนี้ฟรีดอสสามารถนำไปใช้กับเครื่องคอมพิวเตอร์รุ่นเก่าๆ ได้เนื่องจากโครงสร้างของระบบปฏิบัติการที่เลียนแบบดอสทำให้ไม่มีปัญหาเรื่องขนาดฮาร์ดแวร์ที่เหมาะสม

4.4.2.1 ประวัติการพัฒนาฟรีดอส

ฟรีดอสเริ่มต้นจากการนำเอา DOS/NT มาพัฒนาต่อให้สามารถใช้งานได้มากขึ้นนอกเหนือไปจากใช้งานในระบบฝังตัวเหมือนอย่าง DOS/NT เป็น DOS-C ซึ่งขณะนั้นทั้งตัวเคอร์เนลและส่วนประมวลผลคำสั่งหรือคอมมานด์โปรเซสซึ่งอยู่ร่วมกัน จนต่อมาได้มีการแยกทั้งสองส่วนในการพัฒนามีชื่อในการพัฒนาว่า FreeDos Kernel เป็นการพัฒนาเคอร์เนลของดอส และ FreeCom เป็นการพัฒนาเฉพาะส่วนที่ทำหน้าที่เป็นส่วนประมวลผลคำสั่งที่ผู้ใช้ป้อนเข้าไปหรือคอมมานด์โปรเซสซึ่งทั้งสองส่วนนี้เป็นองค์ประกอบหลักของฟรีดอส

สิ่งที่แตกต่างกันระหว่างเคอร์เนล (Kernel) และคอมมานด์คอตคอม (Command.com) ลองเปรียบเทียบเคอร์เนลได้กับหัวใจหลักของระบบปฏิบัติการ ในเอ็มเอสดอสมีสองไฟล์ที่ทำหน้าที่เป็นเคอร์เนลคือ IO.SYS และ MSDOS.SYS แต่ในฟรีดอสจะมีเพียงไฟล์เดียวคือ KERNEL.SYS ส่วนคอมมานด์คอตคอมหรือคอมมานด์โปรเซสซึ่งนั้นเป็นตัวประมวลผลคำสั่งที่รับมาจากผู้ใช้งาน ทำหน้าที่เป็นตัวกลางติดต่อระหว่างผู้ใช้กับระบบคอมพิวเตอร์ซึ่งควบคุมและดูแลโดยเคอร์เนลอีกต่อหนึ่ง ในฟรีดอสจะเรียก COMMAND.COM ว่า FreeCOM ซึ่งใช้ในการทำงานเป็นคอมมานด์โปรเซสของฟรีดอส

DOS/NT เป็นเวอร์ชันของดอสที่ไม่เปิดเผยโค้ดสำหรับใช้ในวงการธุรกิจ เป็นการพัฒนาเพื่อใช้งานระบบฝังตัวโดยเฉพาะ ในขณะนั้นขนาดของ DOS/NT มีขนาดเพียง 68 กิโลไบต์ สนับสนุนการทำงานทั้งโหมดใช้หน่วยความจำจริง (Real Mode) และโหมดหน่วยความจำป้องกัน (Protected Mode) มีองค์ประกอบหลักอยู่ที่ส่วนคือเคอร์เนลเป็นสถาปัตยกรรมแบบไมโครเคอร์เนล ระบบจัดการไฟล์ ระบบจัดการหน่วยความจำ และระบบหน่วยจัดการงานต่างๆ ที่เกิดขึ้นในระบบ ในเคอร์เนลสนับสนุนทั้งมัลติทาสกิงและการส่งผ่านเมสเสจซึ่งเป็นวิธีการที่จำเป็นต้องใช้ในเคอร์เนลแบบนี้ ขณะนั้น DOS/NT ออกมาสามเวอร์ชัน เวอร์ชันที่ได้นำเอามาพัฒนาต่อเป็น DOS-C นั้นเป็น DOS/NT เวอร์ชันที่เป็นแชร์แวร์โดยได้ลงทะเบียนและมีสิทธิใช้งานอย่างถูกต้องแล้ว นำมาพัฒนาให้มีความสามารถสูงขึ้น

ดอส-ซี (DOS-C) เป็นเคอร์เนลของฟรีดอส พัฒนาโดยนายแพท วิลลานี (Pat Villani) ซึ่งเป็นเคอร์เนลที่ได้พัฒนามาจาก DOS/NT ซึ่งเป็นดอสที่ใช้ในระบบฝังตัว ตัวดอส-ซีซึ่งเป็นเคอร์เนลของฟรีดอสมีความเป็นอิสระต่อแพลตฟอร์มสูงมาก หมายความว่าสามารถนำไปใช้ในแพลตฟอร์มอื่นๆ ได้โดยง่าย ซอร์สโค้ดเป็นภาษาซี และมีฟังก์ชันเรียกเข้าหาระบบเหมือนกับเอ็มเอสดอสทุกประการ

4.4.2.2 ลักษณะเด่นของระบบปฏิบัติการฟรีดอส

จากที่ได้กล่าวไปในตอนต้นว่าฟรีดอสมีจุดมุ่งหมายเพื่อสร้างระบบปฏิบัติการที่เลียนแบบระบบปฏิบัติการดอสของไมโครซอฟท์หรือที่เรียกกันว่าเอ็มเอสดอสนั้น ดังนั้นเป้าหมายคือต้องมีความคอมแพททิเบิลกับเอ็มเอสดอสเวอร์ชันใดเวอร์ชันหนึ่ง จากการศึกษาพบว่าเอ็มเอสดอสเวอร์ชัน 4 ขึ้นไปเป็นการเพิ่มความสามารถและแก้ไขข้อบกพร่องในด้านต่างๆจากเอ็มเอสดอสเวอร์ชัน 3.3 ขึ้นไป แต่ส่วนของเคอร์เนลหลักยังคงมีลักษณะไม่ได้เปลี่ยนแปลงไปมากมายนักไปจากเอ็มเอสดอสเวอร์ชัน 3.3 จึงได้เลือกที่พัฒนาฟรีดอสให้คอมแพททิเบิลกับเอ็มเอสดอสเวอร์ชัน 3.3 เป็นหลัก

4.4.2.3 ฮาร์ดแวร์ที่ต้องการ

ฟรีดอสออกแบบให้สามารถทำงานได้บนสถาปัตยกรรมพีซี ตั้งแต่ระดับล่างสุดอย่างรุ่น XT ที่มีหน่วยความจำเพียง 640 กิโลไบต์จนสูงกว่าได้ทั้งหมด อันเนื่องมาจากการออกแบบฟรีดอสให้คอมแพททิเบิลกับเอ็มเอสดอสเวอร์ชัน 3.3 เป็นหลักนั่นเอง ทำให้สามารถทำงานบนพีซีได้เกือบทุกรุ่น มีเรื่องที่ต้องพิจารณาคือขนาดของฮาร์ดดิสก์ที่รองรับได้โดยฟรีดอส ซึ่งถ้าเป็นไปตามเอ็มเอสดอสเวอร์ชัน 3.3 แล้วจะรองรับดิสก์ได้สูงสุดเพียง 32 เมกะไบต์ ซึ่งฟรีดอสสามารถรองรับขนาดของฮาร์ดดิสก์ได้มากกว่านั้น ซึ่งจริงๆแล้วควรรองรับขนาดของฮาร์ดดิสก์ได้มากที่สุดถึง 2 กิกะไบต์ แต่เนื่องจากข้อบกพร่องของฟรีดอสเอง ทำให้รองรับฮาร์ดดิสก์ได้เพียงไม่เกิน 512 เมกะไบต์

4.4.2.4 ซอฟต์แวร์ที่ฟรีดอสสนับสนุน

ซอฟต์แวร์ที่สามารถทำงานได้บนเอ็มเอสดอสหรือคอมแพททิเบิลดอส สามารถทำงานได้บนฟรีดอสได้เกือบทั้งหมด จะมีเพียงโปรแกรมไครเวอร์บางตัวเท่านั้นที่ไม่สามารถทำงานได้ เช่น ซีรอมไครเวอร์ของบางรุ่นยังมีปัญหาอยู่บ้าง แต่จากข้อมูลโดยส่วนใหญ่ได้รับการแก้ไขในเวอร์ชันถัดมาให้สามารถใช้งานได้อย่างถูกต้องโดยส่วนใหญ่แล้วฟรีดอสได้แบ่งออกเป็นสามส่วนหลักคือ

- ส่วนแรกคือส่วนพัฒนาโปรแกรมรรถประโยชน์ จุดประสงค์เพื่อเขียนโปรแกรมรรถประโยชน์ที่เป็นมาตรฐานตามดอสเป็นหลักแต่เขียนขึ้นมาใหม่ทั้งหมด
- ส่วนที่สองคือส่วนพัฒนาไครเวอร์สนับสนุนที่เขียนขึ้นมาใหม่เช่น ANSI.SYS และ CONTRY.SYS เป็นต้น

- ส่วนสุดท้ายคือส่วนที่พัฒนาเคอร์เนลของฟรีดอสซึ่งได้มีการพัฒนาเพิ่มเติมความสามารถและแก้ไขข้อบกพร่องที่เกิดขึ้นในฟรีดอส ซึ่งรองรับต่อแอปพลิเคชันจำนวนมากได้

4.4.2.5 องค์ประกอบของระบบปฏิบัติการฟรีดอส

1. FreeDOS Kernal หรือ DOS-C ต้องคอมแพททิเบิลกับเอ็มเอสดอสเวอร์ชัน 3.3 ซึ่งต้องรองรับอินเทอร์พรีตของดอสทั้งหมด
2. คำสั่งใน Config.sys และ Autoexec.bat ต้องคอมแพททิเบิลกับเอ็มเอสดอสทุกคำสั่ง
3. FreeDOS Commands เรียกว่า FreeCOM เป็นเซลล์ของระบบหรือคอมมานด์โปรเซสซึ่งมีความสามารถเทียบเท่ากับ COMMAND.COM รับคำสั่งจากผู้ใช้และทำงานตามคำสั่งที่ผู้ใช้ป้อนเข้ามา ซึ่งต้องรองรับทั้งคำสั่งภายในหรือคำสั่งที่ FreeCOM รู้จัก และคำสั่งภายนอกหรือโปรแกรมภายนอก คำสั่งภายในต้องสนับสนุนเทียบเท่ากับเอ็มเอสดอสเวอร์ชัน 3.3 ทุกประการ
4. รูปแบบวันที่ ฟรีดอสและโปรแกรมที่ใช้งานบนฟรีดอสต้องรองรับการทำงานในปีสองพันหรือวายุหศ หมายถึงรองรับจำนวนปีที่มีสี่อักขระได้ ซึ่งสามารถเลือกได้ว่าจะใช้ปีแบบใด แบบสองอักขระหลัง หรือแบบสี่อักขระเต็มก็ได้ โดยถ้าเลือกแบบสองอักขระจะใช้วิธีการที่เรียกว่าหน้าต่างในการกำหนดขอบเขตปีก่อนและหลังปีสองพัน หลักการคือถ้ามีปีตั้งแต่ 80 ถึง 99 จะหมายถึงปี 1980 ถึง 1999 แต่ถ้าเป็นปี 00 ถึง 79 จะหมายถึงปี 2000 ถึง 2079 นอกจากนี้ต้องรองรับการรูปแบบวันที่ตามประเทศต่างๆ ได้ด้วย
5. ภาษาที่ใช้เขียนระบบปฏิบัติการฟรีดอสและโปรแกรมอรรถประโยชน์คือภาษาซีและแอสเซมบลี
6. คอมไพเลอร์และเครื่องมือสำหรับการเขียนโปรแกรมบนฟรีดอส คอมไพเลอร์ที่ใช้ในการเขียนฟรีดอสคือ Borland C++ เวอร์ชัน 3.1 เป็นเวอร์ชันสุดท้ายที่สามารถคอมไพล์แล้วได้ออบเจกต์คอมแพททิเบิลกับ 8088 และ Microsoft MASM ซึ่งจะยึดถือวิธีการเขียนโค้ดตามคอมไพเลอร์สองตัวนี้เป็นหลัก จะเห็นได้ว่าเครื่องมือที่ใช้ในการเขียนฟรีดอสไม่ใช่โปรแกรมที่แจกจ่ายให้ใช้งานฟรี จากจุดนี้เองทำให้ได้นำฟรีดอสไปคอมไพล์กับคอมไพเลอร์ที่แจกจ่ายให้ใช้งานฟรีได้ ซึ่งจากที่ได้กล่าวไปก่อนหน้านี้ว่ายังอิงก็ตามจะยึดตามรูปแบบการเขียนโค้ดตามคอมไพเลอร์ที่ใช้ในข้างต้น ซึ่งพบว่ามีคอมไพเลอร์ที่สามารถใช้ทดแทนได้คือ Turbo C เวอร์ชัน 2.1 และเน็ตไรว์มาโครแอสเซมเบลเลอร์หรือ NASM อย่างไรก็ตามยังมีคอมไพเลอร์ภาษาซีอีกหลายตัวที่สามารถใช้ได้เช่นแปซิฟิกซี คอมไพเลอร์ (Pacific C Compiler) ส่วนกนู ซี คอมไพเลอร์ (GNU C Compiler) นั้นไม่สนับสนุนการทำงานในเรยลไทม์ขนาด 16 บิตทำให้ไม่สามารถนำไปใช้ได้

4.4.2.6 ความสามารถด้านเครือข่าย

เคอร์เนลของฟรีดอสสนับสนุนการทำงานด้านเครือข่ายซึ่งเป็นไปตามเอ็มเอสคอสเวอร์ชัน 3.3 แต่การจะทำให้ฟรีดอสสามารถติดต่อกับเครือข่ายได้ต้องมีองค์ประกอบเพิ่มอีกสามส่วนคือ อะแดปเตอร์อีเทอร์เน็ต ไดรเวอร์อีเทอร์เน็ต และโพรโทคอล TCP/IP สแต็ก จะขาดส่วนหนึ่งส่วนใดไม่ได้

1. อะแดปเตอร์อีเทอร์เน็ต ในปัจจุบันหาได้ไม่ยาก เพราะมีราคาถูกลงมากเมื่อเทียบกับแต่ก่อน แต่สิ่งที่ต้องพิจารณาคือไดรเวอร์สนับสนุนที่ทำงานบนฟรีดอสได้ และรู้จักอะแดปเตอร์อีเทอร์เน็ตนี้
2. ไดรเวอร์อีเทอร์เน็ต ซึ่งต้องสามารถทำงานได้บนฟรีดอสและรู้จักอะแดปเตอร์อีเทอร์เน็ต ซึ่งสามารถหาไดรเวอร์ได้จากบริษัทผู้ผลิตโดยตรง ซึ่งโดยส่วนใหญ่จะแจกจ่ายฟรีอยู่แล้ว ทำให้ลดปัญหาเรื่องการหาไดรเวอร์สนับสนุนที่สามารถทำงานได้บนฟรีดอสได้มากขึ้น
3. TCP/IP สแต็กสำหรับดอส ที่พบในปัจจุบันมีดังนี้
 - Trumpet TCP/IP สแต็ก เป็นสแต็กอย่างง่ายที่สุด ทั้งง่ายต่อการตั้งค่าและการเรียนรู้ใช้งาน
 - NCSA TCP/IP สแต็ก พัฒนามาจาก NET14 ของระบบ BSD ซึ่งเป็นโพรโทคอล TCP/IP สแต็ก อย่างง่ายรองรับโพรโทคอลทั้ง UDP, TCP และ ICMP รวมไปถึง ARP เนื่องมาจากการใช้อะแดปเตอร์อีเทอร์เน็ตอย่างในปัจจุบันด้วย
 - WATTCP TCP/IP สแต็ก พัฒนามาจาก NCSA TCP/IP สแต็ก เพิ่มเติมความสามารถ ซึ่งสามารถหาควินโหลดซอร์สโค้ดได้บนอินเทอร์เน็ต แต่เอกสารคู่มือสำหรับโปรแกรมเมอร์ต้องซื้อแต่ไม่ได้แจกจ่ายฟรี การติดตั้งสามารถทำได้โดยง่าย สนับสนุนโพรโทคอลมากกว่าใน NCSA TCP/IP สแต็ก มีซอร์สโค้ดตัวอย่างการใช้งานอย่างละเอียดอีกด้วย
 - FreePASCAL TCP/IP สแต็ก เป็นโพรโทคอลสแต็กที่เขียนโดยใช้ภาษาปาสคาล และแจกจ่ายฟรีทางอินเทอร์เน็ต ข้อเสียคือเอกสารคู่มือการใช้และคู่มือโปรแกรมเมอร์มีน้อยและยังไม่สนับสนุนกับการใช้แอปพลิเคชันบางประเภทด้วย
 - Microsoft TCP/IP สแต็ก ง่ายต่อการปรับแต่งและติดตั้งในการนำไปใช้งาน แต่ต้องการหน่วยความจำมากกว่าโพรโทคอลสแต็กอื่นๆ
 - WATTCP 32 ต่างกับ WATTCP TCP/IP สแต็กตรงที่ใช้ซ็อกเก็ตตามแบบ BSD ทุกรูปแบบ สนับสนุนการทำงานทั้งหน่วยความจำจริงและในหน่วยความจำป้องกันทั้งแบบ 16 บิตและ 32 บิต ได้รับความนิยมนำไปใช้งานมากที่สุด

4.4.2.7 ลิขสิทธิ์ของฟรีดอส

ซอร์สโค้ดของฟรีดอสสามารถเปิดเผยได้ทั้งหมด ซึ่งเป็นไปตามข้อตกลง GPL หรือ GNU General Public License ซึ่งสามารถเปิดเผยทุกส่วนของระบบปฏิบัติการได้ไม่ว่าในรูปแบบใดๆ นำไปใช้ นำไปต่อเติมความสามารถได้ไม่จำกัด แต่โปรแกรมที่พัฒนาขึ้นภายใต้ข้อตกลงนี้จะเป็นลิขสิทธิ์ของ GPL นี้ สามารถเพิ่มเติมมูลค่าของโปรแกรมได้ไม่จำกัด ลิขสิทธิ์ของ GPL ออกแบบมาเพื่อปกป้องผู้พัฒนาและการนำซอฟต์แวร์ที่แจกจ่ายฟรีไปใช้ในทางที่ไม่เป็นไปตามลักษณะเดิมของโปรแกรมนั้นหรือใช้ในทางผิดกฎหมาย และเน้นไปที่ความมีอิสระในการใช้งานโปรแกรมได้ไม่จำกัด

4.4.3 ระบบปฏิบัติการอีแอลเคส (ELKS Operating System)

อีแอลเคส (ELKS) ย่อมาจาก The Embeddable Linux Kernel Subset เป็นโครงการสร้างเคอร์เนล ขนาดเล็กที่คล้ายคลึงกับระบบปฏิบัติการลินุกซ์ สามารถทำงานได้บนเครื่องคอมพิวเตอร์ที่มีขนาดชิพ (CPU) จำกัด มีหน่วยความจำจำกัด เวอร์ชันแรกสามารถทำงานได้บนเครื่องที่ใช้ชิพของอินเทล 8086 เหมาะสำหรับนำไปใช้ในระบบฝังตัว เป้าหมายของการพัฒนาอีแอลเคสคือการสร้างทางเลือกของระบบปฏิบัติการลินุกซ์บนเครื่องรุ่นเก่าที่ใช้ชิพขนาด 8086 และ 80286 บนเครื่องปาล์ม ไมโครคอมพิวเตอร์บอร์ดเดียว ระบบควบคุมขนาดเล็ก และเครื่องคอมพิวเตอร์รุ่นเก่า

เคอร์เนลของอีแอลเคสมีขนาดเพียง 200 กิโลไบต์เท่านั้น เมื่อทำงานจริงจะใช้หน่วยความจำเพียง 400 ถึง 512 กิโลไบต์ และใช้ระบบปฏิบัติการลินุกซ์เป็นแพลตฟอร์มในการพัฒนา

แนวความคิดในการพัฒนานั้นสืบเนื่องมาจากการพัฒนาระบบปฏิบัติการให้สามารถทำงานได้ในสภาพแวดล้อมที่จำกัดหรือสภาพแวดล้อมที่ใช้ในระบบฝังตัว ซึ่งในปัจจุบันมีระบบปฏิบัติการเพียงมินิक्सและดอสเท่านั้นที่สนับสนุนชิพรุ่นเก่าตั้งแต่ 8086 ซึ่งในปัจจุบันหาไม่ได้อีกแล้ว

อีแอลเคสเวอร์ชันที่มีความเสถียรภาพที่สุดในปัจจุบันคือเวอร์ชัน 0.1.0 ซึ่งเป็นเวอร์ชันแรกมาก สนับสนุนเพียงฟอร์ตอ努กรมและฟอร์ตชานานเท่านั้น หากจะมีการนำมาใช้จริงแล้วคงต้องพิจารณาถึงองค์ประกอบที่มากกว่านี้เนื่องจากยังมีแอปพลิเคชันสนับสนุนไม่มาก รวมไปถึงการพัฒนาของระบบปฏิบัติการเพราะยังเป็นระบบปฏิบัติการที่ยังใหม่อยู่มาก

4.4.3.1 เปรียบเทียบลินุกซ์กับอีแอลเคส

อีแอลเคสออกแบบมาเพื่อเป็นระบบปฏิบัติการที่คล้ายคลึงกับลินุกซ์ให้มากที่สุด แต่มีขนาดเล็กสามารถศึกษาทำความเข้าใจกับระบบได้โดยใช้เวลาไม่นาน อย่างไรก็ตามอีแอลเคสยังอยู่ในสถานะเริ่มต้นของการพัฒนา จะเห็นได้ว่ายังมีอีกหลายจุดที่ต้องพัฒนาอีกมากกว่าจะใช้งานได้จริง โค้ดในส่วนต่างๆ หลายส่วนเช่นส่วนจัดการเรื่องหน่วยความจำ ไบรารีสนับสนุน ความสามารถด้านการเชื่อมต่อเครือข่าย ในขณะนี้ทำ

ได้เพียงเริ่มต้นระบบได้โดยนำพื้นฐานเคอร์เนลมาจากลินุกซ์ และใช้โปรแกรมได้บ้างเล็กน้อย

4.4.3.2 ลักษณะเด่นของอีแอลเอส

- สามารถทำงานได้บนเครื่องคอมพิวเตอร์ที่มีขนาดจำกัด ใช้ซีพียูของอินเทล 8086 ขึ้นไป
- เคอร์เนลของอีแอลเอสมีขนาดเล็กเพียง 200 กิโลไบต์ และใช้หน่วยความจำในการทำงานน้อยมาก
- สนับสนุนตามมาตรฐาน POSIX
- ใช้ระบบไฟล์ตามแบบมินิกซ์ เพราะมีขนาดเล็กและมีความสามารถในการทำงานได้ดีในระบบเล็กๆ
- มีไบโอสสนับสนุนในขณะนี้คือไบโอสของไบออส (BIOS)
- ใช้ VT52 เป็นมาตรฐานในการติดต่อกับอินพุตและเอาต์พุต
- คอมไพเลอร์พื้นฐานจากมินิกซ์ที่ทำงานได้บนลินุกซ์
- ไลบรารีสนับสนุนบางไลบรารีเท่าที่จำเป็น
- สนับสนุนเวอร์ชวลคอนโซล (Virtual console)

4.4.4 KVM

KVM (K Virtual Machine หรือ Kjava Virtual Machine) เป็นจาวาเวอร์ชวลแมชีน (Java Virtual Machine) ที่ถูกออกแบบมาพิเศษเพื่อให้มีความสามารถในการพอร์ตได้ง่าย มีขนาดเล็ก กระทัดรัด และสามารถใช้ในอุปกรณ์ที่มีทรัพยากรอยู่อย่างจำกัดเช่น โทรศัพท์มือถือ เพจเจอร์ อุปกรณ์จัดการข้อมูลส่วนบุคคล อุปกรณ์อินเทอร์เน็ตแบบพกพา เครื่องใช้ไฟฟ้าในบ้าน และระบบคอมพิวเตอร์ฝังตัวอื่นๆ ซึ่งอุปกรณ์เหล่านี้ส่วนใหญ่จะมี ไมโครโปรเซสเซอร์ขนาด 16 บิต หรือ 32 บิต และมีหน่วยความจำอย่างต่ำประมาณ 128 กิโลไบต์ เท่านั้น

4.4.4.1 จุดมุ่งหมายของการออกแบบ KVM

จุดมุ่งหมายของการออกแบบ KVM ก็คือ การสร้าง จาวาเวอร์ชวลแมชีนที่มีขนาดเล็กที่สุดเท่าที่จะเป็นไปได้ โดยยังคงไว้ซึ่งลักษณะซึ่งเป็นหัวใจหลักของภาษาจาวา และยังสามารถทำงานได้บนอุปกรณ์ที่มีข้อจำกัดทางด้านทรัพยากรซึ่งมีหน่วยความจำในระดับไม่กี่สิบกิโลไบต์ได้ โดยจุดมุ่งหมายของการออกแบบ KVM สามารถกล่าวให้ชัดได้เป็นข้อๆดังนี้

- มีขนาดเล็ก ใช้หน่วยความจำขั้นต่ำ 60 กิโลไบต์ (ขึ้นอยู่กับแพลตฟอร์มเป้าหมาย และตัวเลือกในการคอมไพล์)

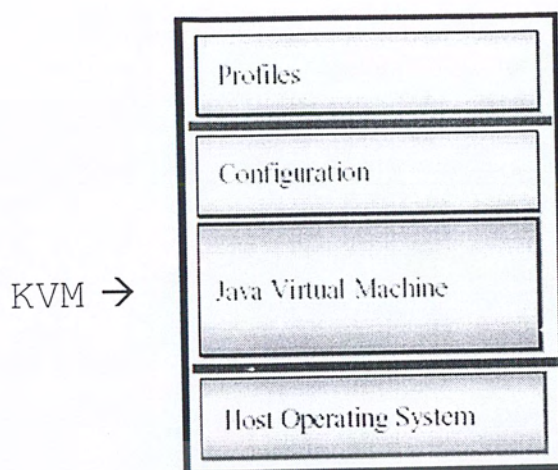
- ซอสโค้ดสามารถอ่านเข้าใจได้ง่าย และมีความสามารถในการพอร์ตไปยังแพลตฟอร์มต่างๆได้ง่าย
- มีการจัดแบ่งโปรแกรมออกเป็นส่วนๆอย่างดีและมีความสามารถในการปรับแต่งได้
- มีความสมบูรณ์และเร็วเท่าที่จะเป็นไปได้โดยที่ไม่ทำให้เป้าหมายข้ออื่นๆต้องด้อยลง

4.4.4.2 ความสัมพันธ์ระหว่าง KVM กับ J2ME

KVM ถูกพัฒนาขึ้นเป็นส่วนหนึ่งของ Java 2 Micro Edition (J2ME) ซึ่งเป็นแพลตฟอร์มที่มีความเป็นโมดูลาร์และมีความสามารถในการขยายตัวได้ สำหรับการพัฒนาแอปพลิเคชันที่มีความยืดหยุ่น ปรับเปลี่ยนได้และความปลอดภัย ในอุปกรณ์ระบบฝังตัวต่างๆ

J2ME แพลตฟอร์มประกอบไปด้วยชั้นต่างๆคือ ระบบปฏิบัติการโฮสต์ จาวาเวอร์ชวลแมชีน คอนฟิกูเรชัน (Configuration) และ โปรไฟล์ (Profiles) ซึ่งแต่ละส่วนมีรายละเอียดดังนี้

- ระบบปฏิบัติการโฮสต์ ก็คือระบบปฏิบัติการที่รันอยู่บนอุปกรณ์ต่างๆ
- จาวาเวอร์ชวลแมชีน เป็นชั้นของเวอร์ชวลแมชีนที่เขียนมาใหทำงานได้บนสำหรับระบบปฏิบัติการนั้นๆ และสนับสนุนคอนฟิกูเรชันที่อยู่ระดับบน
- คอนฟิกูเรชัน เป็นตัวกำหนดแพลตฟอร์มขั้นต่ำสำหรับ กลุ่มในแนวนอนหรือกลุ่มของอุปกรณ์ ที่แต่ละอุปกรณ์ มีความต้องการทางด้านหน่วยความจำและความสามารถในการประมวลผลที่ใกล้เคียงกัน คอนฟิกูเรชันจะเป็นตัวกำหนดภาษาจาวาและพีเจอาร์ของเวอร์ชวลแมชีนและ คลาสขั้นต่ำที่ผู้ผลิตอุปกรณ์หรือซอฟต์แวร์จะคาดหวังว่าจะมีในทุกๆอุปกรณ์ที่อยู่ในกลุ่มเดียวกัน
- โปรไฟล์ เป็นชั้นที่อยู่บนคอนฟิกูเรชัน โปรไฟล์จะกำหนดแนวทางความต้องการของส่วนกลุ่มตลาดในแนวตั้ง หรือในตระกูลของอุปกรณ์นั้นๆ จุดประสงค์หลักของโปรไฟล์ก็คือ การรับประกันว่า จะมีการทำงานร่วมกันได้ภายในอุปกรณ์ตระกูลเดียวกันหรือ ในโดเมนเดียวกัน โดยการกำหนดแพลตฟอร์มจาวาที่เป็นมาตรฐานสำหรับตลาดนั้น โปรไฟล์โดยปกติแล้วจะรวมคลาสไลบรารีที่อยู่นอกเหนือว่าโดเมนที่กำหนดระดับในคอนฟิกูเรชัน



รูปที่ 13 แพลตฟอร์มของ Java 2 Micro Edition

ในปัจจุบัน KVM และ คอนฟิกูเรชัน CLDC (Connected, Limited Device Configuration) ซึ่งเป็นคอนฟิกูเรชันที่กำหนดเกี่ยวกับอุปกรณ์ที่มีทรัพยากรจำกัด มีความสัมพันธ์กันอย่างใกล้ชิด เนื่องจากในขณะนี้เพียง CLDC เท่านั้นที่อยู่บน KVM อย่างไรก็ดีคาดว่า ในอนาคต KVM อาจจะสนับสนุนคอนฟิกูเรชันตัวอื่นด้วย

บทที่ 5

ระบบปฏิบัติการมินิกซ์

5.1 เหตุผลในการเลือกมินิกซ์

ระบบปฏิบัติการที่มีอยู่อย่างมากมายนั้น ต่างก็ได้ถูกออกแบบมาเพื่อให้ใช้งานในด้านต่างๆกัน และมีความสามารถในการทำงานได้บนฮาร์ดแวร์ที่ต่างกันด้วย หนึ่งในเนื่องจากโครงการนี้เป็นการพัฒนาแพลตฟอร์มสำหรับระบบฝังตัวซึ่งมีพื้นฐานอยู่บนบอร์ดคอม86 จึงทำให้มีข้อพิจารณาในเบื้องต้นในการคัดเลือกระบบปฏิบัติการคือเรื่องของลิขสิทธิ์และความเข้ากันได้กับบอร์ดคอม86

ลิขสิทธิ์ ระบบปฏิบัติการที่จะนำมาใช้ได้ ในโครงการนี้ต้องแจกฟรีและมีการเผยแพร่ซอร์สโค้ดเพื่อที่จะได้นำมาศึกษาและพัฒนาได้อย่างอิสระ

ความเข้ากันได้กับบอร์ดคอม86 ความต้องการของฮาร์ดแวร์ส่งผลโดยตรงต่อการพิจารณาเลือกระบบปฏิบัติการ บอร์ดคอม86 เป็นบอร์ดที่ใช้ไมโครโปรเซสเซอร์ตระกูล x86 แบบ 16 บิต ซึ่งทำงานในโหมดหน่วยความจำจริงเท่านั้น และมีหน่วยความจำแบบคงอยู่เพียง 512 กิโลไบต์

ความต้องการที่จำเป็นข้างต้นทั้ง 2 ข้อนั้นตรงกับคุณลักษณะของมินิกซ์ ซึ่งก็คือ เป็นระบบที่มีการแจกจ่ายฟรีและเผยแพร่ซอร์สโค้ด สามารถนำไปพัฒนาและเผยแพร่ต่อได้ภายใต้ลิขสิทธิ์ของโอเพ่นบีเอสดี (OpenBSD License) ส่วนเรื่องความเข้ากันได้กับบอร์ดคอม86 นั้น มินิกซ์มีเคอร์เนลขนาดเล็กเพียง 186 กิโลไบต์ (ทั้งนี้ขนาดจะขึ้นอยู่กับฟังก์ชันที่ได้กำหนดให้กับระบบ) ซึ่งเหมาะกับหน่วยความจำแบบคงอยู่ของบอร์ดคอม86 ที่มีอยู่อย่างจำกัด มินิกซ์สามารถทำงานได้บนไมโครโปรเซสเซอร์ตั้งแต่ 8086 และใช้หน่วยความจำน้อยกว่า 512 กิโลไบต์ ในการเริ่มต้นระบบ

นอกจากนี้ยังมีระบบปฏิบัติการอื่นๆที่มีคุณลักษณะที่สอดคล้องกัน ได้แก่ ฟรีดอส และอีแอลเคส ดังที่ได้กล่าวถึงคุณลักษณะเบื้องต้นในบทที่ 4

ข้อพิจารณา	มินิกซ์	อีแอลเคส	ฟรีดอส
แจกจ่ายฟรี เผยแพร่ซอร์สโค้ด	ใช่	ใช่	ใช่
ทำงานบนไมโครโปรเซสเซอร์แบบ 16 บิต	ใช่	ใช่	ใช่
ใช้หน่วยความจำไม่เกิน 1 เมกะไบต์	ใช่	ใช่	ใช่

ตาราง 1 แสดงข้อพิจารณาเบื้องต้นที่จำเป็น

นอกจากความจำเป็นทั้งสองข้อแล้วยังต้องมีการพิจารณาความเหมาะสมในการนำไปใช้งาน ระบบปฏิบัติการควรมีคุณสมบัติอื่นๆที่สนับสนุนฟังก์ชันการทำงานของบอร์ดคอม86 ได้แก่ ความสามารถในการทำงานด้านเน็ตเวิร์ก และคุณสมบัติอื่นๆที่เอื้อประโยชน์ต่อการพัฒนาแอปพลิเคชันให้มีประสิทธิภาพได้ เช่น ความสามารถในการทำงานหลายงานพร้อมกันได้ ความสามารถในการรองรับผู้ใช้หลายคนพร้อมกัน

ข้อพิจารณา	มินิกซ์	อีแอลเคส	ฟรีดอส
สนับสนุนการทำงานด้านเน็ตเวิร์ก	ใช่	ยังไม่สมบูรณ์	ใช่
สามารถทำงานหลายงานพร้อมกัน	ใช่	ใช่	ไม่ใช่
รองรับการทำงานแบบผู้ใช้หลายคน	ใช่	ใช่	ไม่ใช่

ตาราง 2 แสดงคุณสมบัติที่สนับสนุนการใช้งานของมินิกซ์ อีแอลเคส และฟรีดอส

ความสามารถในการสนับสนุนการทำงานด้านเน็ตเวิร์กหมายถึงการที่ระบบปฏิบัติการสนับสนุนการเชื่อมต่อ โปรโตคอลที่เป็นมาตรฐาน เช่น โปรโตคอลชุดทีซีพี/ไอพี และมีไลบรารีให้ใช้ในการพัฒนาแอปพลิเคชันทางด้านเน็ตเวิร์ก

โดยทั่วไปแล้วในแอปพลิเคชันด้านเน็ตเวิร์กที่มียุคแรกๆ ที่มีการทำงานแบบไคลเอนต์-เซิร์ฟเวอร์นั้น ฟังก์ชันเซิร์ฟเวอร์มักจะมีความต้องการที่จะรองรับการให้บริการไคลเอนต์พร้อมๆ กันหลายๆ ไคลเอนต์ ซึ่งการที่จะพัฒนาแอปพลิเคชันในลักษณะนี้ได้จำเป็นต้องได้รับการสนับสนุนจากระบบปฏิบัติการที่มีความสามารถในการทำงานหลายงานพร้อมกัน

จากตาราง 2 แสดงคุณสมบัติที่สนับสนุนการใช้งานของทั้งมินิกซ์ อีแอลเคส และฟรีดอส จะเห็นว่ามีเพียงมินิกซ์และอีแอลเคสที่การสนับสนุนทั้งสามข้อ แต่อีแอลเคสนั้นปัจจุบันยังสนับสนุนโปรโตคอลในชุดทีซีพี/ไอพีได้บางโปรโตคอลเท่านั้น เนื่องจากยังอยู่ในช่วงเริ่มแรกของการพัฒนา

เนื่องจากการใช้ชุดพัฒนาก็คือการพัฒนาแอปพลิเคชันและผู้ใช้ก็เป็นนักพัฒนา ดังนั้นระบบจึงต้องเอื้อต่อการพัฒนาโปรแกรมโดยมี การสนับสนุนต่อมาตรฐานต่างๆ เช่น POSIX และ ANSI C ในการเขียนโปรแกรม อินเตอร์เฟซของฟังก์ชันและคอมไพเลอร์ที่สนับสนุนข้อกำหนดของภาษาที่เป็นมาตรฐาน ทำให้ผู้ใช้สามารถเขียนโปรแกรมได้สะดวกรวดเร็ว

มินิกซ์มีเครื่องมือในการพัฒนาโปรแกรมได้แก่ คอมไพเลอร์ภาษาซี ACK-CC ซึ่งสนับสนุนมาตรฐาน ANSI C ดีบั๊กเกอร์ MDB มีไครเวอร์สำหรับอุปกรณ์ต่างๆ มีฟังก์ชันไลบรารีที่เป็นไปตามมาตรฐาน POSIX และมีแหล่งข้อมูลมากมายที่เป็นประโยชน์ต่อการพัฒนาโปรแกรม

	มินิกซ์	อีแอลเอส
เครื่องมือในการพัฒนา		
• คอมไพเลอร์	ACK CC	BCC
สนับสนุน ANSI C	ใช่	-
• ดีบั๊กเกอร์	MDB	-
ไดรเวอร์	มีมาก	มีน้อย
ไลบรารี	มีมาก ประมาณ 300	มีน้อย ประมาณ 50
สนับสนุน POSIX	ใช่	-
แหล่งข้อมูล	มีมาก	มีน้อย

ตาราง 3 แสดงเครื่องมือในการพัฒนาของมินิกซ์และอีแอลเอส

นอกจากข้อพิจารณาที่จำเป็นในเบื้องต้น ข้อพิจารณาในด้านความสามารถที่สนับสนุนการใช้งานต่างๆแล้ว ยังมีข้อพิจารณาที่สำคัญที่ต้องคำนึงถึงนั่นคือความเหมาะสมในการพัฒนา ในการพัฒนาในระบบฝังตัวมักมีความยากในการพัฒนา และมีปัญหาคือ ความไม่คุ้นเคยในการพัฒนาบนระบบฝังตัว ดังนั้นสิ่งสำคัญที่จะต้องคำนึงถึงก็คือ เป็นระบบปฏิบัติการที่สามารถเรียนรู้ได้อย่างรวดเร็ว มีเอกสารและแหล่งข้อมูลให้ศึกษา ถ้ามีน้อย ต้องเสียเวลาศึกษานานก่อนจะพัฒนา ข้อต่อมาก็คือระบบปฏิบัติการควรจะปรับแต่งได้ตามลักษณะของงานที่ชุดพัฒนาคอม86 ถูกนำไปใช้ สามารถรองรับการขยายขีดความสามารถทางด้านการฮาร์ดแวร์ในอนาคต

เนื่องจากเคอร์เนลของมินิกซ์มีโครงสร้างเป็นแบบโมโครเคอร์เนล มินิกซ์ประกอบที่แบ่งส่วนหน้าที่ชัดเจนระหว่างเคอร์เนลจริงกับส่วนที่ให้บริการต่างๆ เช่น ส่วนหน่วยควบคุมหน่วยความจำและระบบไฟล์แยกเป็นอิสระจากส่วนเคอร์เนล มินิกซ์ได้รับข้อดีของโครงสร้างแบบโมโครเคอร์เนลมา ได้แก่ความสามารถในการเพิ่มเติมหรือลดความสามารถของเคอร์เนลได้ง่าย มีความยืดหยุ่นสูงสามารถปรับแต่งได้ง่าย ทั้งนี้การที่เคอร์เนลมีขนาดเล็กและมีการแบ่งออกเป็นส่วนๆยังทำให้ระบบไม่มีความซับซ้อน ทำให้ง่ายต่อการศึกษาทำความเข้าใจ

จากตาราง 4 แสดงลักษณะต่างๆของมินิกซ์และอีแอลเอส จะสังเกตเห็นว่าเวอร์ชันปัจจุบันอีแอลเอสอยู่ที่เวอร์ชัน 0.0.84 เท่านั้น เนื่องจากยังอยู่ในช่วงเริ่มต้นพัฒนา ต่างจากมินิกซ์ซึ่งพัฒนามาตั้งแต่ปี 1987 และได้พัฒนาต่อเนื่องมาจนถึงปัจจุบัน ทำให้ตัวเคอร์เนลนั้นมีความสมบูรณ์กว่า และมีแอปพลิเคชันต่างๆมากกว่า จากเหตุผลข้างต้นทั้งหมดพบว่ามินิกซ์นั้นมีความเหมาะสมมากที่สุดที่จะนำมาใช้พัฒนาในโครงการนี้ แต่ในอนาคตนั้นอีแอลเอสก็ระบบที่น่าติดตามมากสำหรับการพัฒนาในระบบฝังตัว

	มินิกซ์	อีแอลเอส
เคอร์เนล	ไมโครเคอร์เนล	โมนอลิธิก
ขนาดเคอร์เนล	186 กิโลไบต์	200 กิโลไบต์
โปรเซสเซอร์ที่ทำงานได้	ตั้งแต่ 8086	8086 – 80286
หน่วยความจำ	512	400
คอมไพเลอร์	ACK-CC	BCC
ANSI C	ใช่	-
POSIX	ใช่	-
ระบบไฟล์	MINIX FS	MINIX FS, ELKS FS
ภาษาที่ใช้เขียน	C, assembly	C, assembly
เอดิเตอร์	ex, vi, ed, emacs	vi, sed
จำนวนโปรแกรมอรรถประโยชน์	200	~140
เชลล์	ash, sh	sash, ash
เวอร์ชันปัจจุบัน	2.0.3	0.0.84

ตาราง 4 เปรียบเทียบลักษณะต่างๆระหว่างมินิกซ์กับอีแอลเอส

5.2 มินิกซ์

โครงสร้างของมินิกซ์เป็นแบบไมโครเคอร์เนล เมื่อนมองอย่างกว้างๆจะประกอบไปด้วยชั้น 4 ชั้น แต่ละชั้นได้ถูกกำหนดให้มีหน้าที่ในการทำงานของแต่ละชั้นอย่างชัดเจน ในแต่ละชั้นยังแบ่งส่วนการทำงานต่างๆเป็นแต่ละโปรเซส โปรเซสต่างๆสื่อสารกันระหว่างโปรเซสโดยใช้กลไกของการส่งผ่านเมสเสจ (Message passing) รูปที่ 14 แสดงระดับชั้นทั้ง 4 ของมินิกซ์

Init	User process 1		User process 2		...	4	User Process
Mm	Fs	Inet	...			3	Server Process
Disk	Tty	Clock	System	Ethernet	...	2	Task
Process Management						1	

รูปที่ 14 โครงสร้างโปรเซสของมินิกซ์

5.2.1 โครงสร้างภายในของมินิกซ์

ในชั้นล่างสุดมีหน้าที่หลักอยู่ 2 อย่าง อย่างแรกคือ จัดการเรื่องแทริป(trap)และการทำอินเทอร์รัปต์ ทำการเซฟและคืนค่าให้กับรีจิสเตอร์ ทำหน้าที่จัดตารางเวลาทำงาน(scheduling) และ หน้าที่อื่นๆที่ทำให้เกิดการดำเนินงานแบบโครงสร้างที่เป็นระดับชั้น หน้าที่หลักอย่างที่สองคือ ทำให้โปรเซสในชั้นบนเป็นอิสระต่อกัน โดยการทำหน้าที่ในการสื่อสารระหว่างโปรเซสให้ ผ่านกลไกการส่งผ่านเมสเสจ ได้แก่ การ

ตรวจสอบจุดหมายปลายทาง หากที่อยู่ในหน่วยความจำของบัฟเฟอร์เพื่อที่จะรับส่ง และถือปี่ข้อมูลจากผู้ส่งไปยังผู้รับ โค้ดในชั้นนี้ทำงานอยู่ในระดับล่างสุดซึ่งเกี่ยวข้องกับระบบอินเทอร์พรีตจะถูกโค้ดด้วยภาษาแอสซิมบลี ส่วนที่เหลือในชั้นนี้และชั้นบนจะถูกโค้ดด้วยภาษาซีทั้งหมด

ในชั้นที่สอง เป็นชั้นของโปรเซสที่ทำหน้าที่เกี่ยวกับอินพุตเอาต์พุต แต่ละชนิดของอินพุตเอาต์พุตจะมีโปรเซสรองรับการทำงานอยู่หนึ่งโปรเซส โปรเซสในชั้นนี้จะเรียกว่า ทาสก์ (task) ซึ่งมีหน้าที่เปรียบได้กับดีไวซ์ไดรเวอร์ของอุปกรณ์ต่างๆ ทาสก์จำเป็นต้องใช้สำหรับอุปกรณ์แต่ละชนิดได้แก่ ดิสก์, พรินเตอร์, เทอร์มินอล, การ์ดเน็ตเวิร์ก และนาฬิกา เป็นต้น หากมีการเพิ่มอุปกรณ์อื่นๆเข้าไป ต้องมีการเพิ่มทาสก์เพื่อเข้าไปรองรับการทำงานของอุปกรณ์นั้นๆด้วย

ทุกทาสก์ในชั้นที่สองและโค้ดในชั้นล่างสุดจะถูกลิงก์เข้าด้วยกันเป็นไบนารีโปรแกรมเพียงโปรแกรมเดียวเรียกว่า เคอร์เนล อาจมีบางทาสก์ที่มีการแชร์การทำงานกับส่วนอื่นๆบ้าง แต่ส่วนใหญ่จะทำงานเป็นอิสระต่อกัน สื่อสารผ่านกันโดยใช้กลไกการส่งผ่านเมสเซจ และถูกจัดตารางการทำงานแยกต่อกัน ในโปรเซสเซอร์ของอินเทลตั้งแต่รุ่น 286 เป็นต้นมา ได้มีการกำหนดลำดับสิทธิการเข้าถึง (privilege) ไว้ 4 ระดับ สำหรับแต่ละโปรเซส ถึงแม้ว่าทาสก์และเคอร์เนลจะคอมไพล์รวมกัน แต่ตอนที่เคอร์เนลและส่วนอินเทอร์พรีตทำงาน จะมีสิทธิสูงกว่าทาสก์ ดังนั้นส่วนที่เป็นเคอร์เนลจริงๆ สามารถที่จะเข้าถึงส่วนใดๆในหน่วยความจำและรีจิสเตอร์ต่างๆ ได้ ซึ่งจริงๆแล้วเคอร์เนลสามารถจะเรียกส่วนของโค้ดใดๆในระบบทั้งหมดได้ ส่วนทาสก์นั้นมีสิทธิต่ำกว่าเคอร์เนล(ในระดับล่างสุด)ไม่สามารถเรียกโค้ดในส่วนต่างๆได้ทั้งหมด และไม่สามารถเข้าถึงรีจิสเตอร์และหน่วยความจำทั้งหมดได้ อย่างไรก็ตามทาสก์สามารถเข้าถึงโปรเซสส่วนที่มีสิทธิลำดับต่ำกว่าได้ มีทาสก์อยู่หนึ่งคือ ทาสก์ระบบ (system task) ซึ่งไม่ได้ทำงานเกี่ยวกับอินพุตเอาต์พุตแต่มีหน้าที่ในการให้บริการโปรเซสอื่นๆเกี่ยวกับการคัดลอกข้อมูลระหว่างพื้นที่หน่วยความจำต่างๆ ให้กับโปรเซสที่มีสิทธิในระดับต่ำกว่าซึ่งไม่อนุญาตให้ทำการเข้าหน่วยความจำส่วนสำคัญอื่นๆโดยตรง แต่สำหรับเครื่องที่ไม่ได้มีการกำหนดลำดับสิทธิการเข้าถึงอย่างไโปรเซสเซอร์ของอินเทลรุ่นเก่าก่อนรุ่น 286 นั้นจึงไม่สามารถจำกัดการเข้าถึงของโปรเซสต่างๆได้

ในชั้นที่สาม เป็นส่วนของโปรเซสที่ทำหน้าที่ให้บริการแก่อยูสเซอร์โปรเซสในระบบ เซิร์ฟเวอร์เหล่านี้จะทำงานในลำดับสิทธิต่ำกว่าส่วนเคอร์เนลและทาสก์ จึงไม่สามารถเข้าถึงพอร์ตอินพุตเอาต์พุตต่างๆได้โดยตรง และไม่สามารถเข้าถึงหน่วยความจำส่วนอื่นๆที่เคอร์เนลไม่ได้กำหนดไว้ให้ ส่วนจัดการหน่วยความจำ(MM – Memory Manager) รองรับซิสเต็มคอลที่เกี่ยวกับการจัดการหน่วยความจำได้แก่ FORK, EXEC และ BRK เป็นต้น ส่วนจัดการระบบไฟล์ (FS – File System) รองรับซิสเต็มคอลที่เกี่ยวข้องกับระบบไฟล์ เช่น READ, MOUNT และ CHDIR เป็นต้น ส่วนจัดการระบบไฟล์นั้นได้ถูกออกแบบมาให้ทำหน้าที่เป็นไฟล์เซิร์ฟเวอร์และสามารถย้ายไปยังระบบที่อยู่อื่นได้ โดยที่ไม่ต้องเปลี่ยนแปลงส่วนอื่นๆมากนัก เนื่องจากเซิร์ฟเวอร์ต่างๆนั้นมีอิสระต่อกัน เซิร์ฟเวอร์เหล่านี้จะแตกต่างจากยูสเซอร์โปรเซสคือ จะเริ่มทำงานตั้งแต่ตอนเริ่มต้นระบบ และทำงานจนกว่าระบบจะหยุดทำงาน ถึงแม้ว่าเซิร์ฟเวอร์เหล่านี้จะทำงานในลำดับสิทธิการเข้าถึงเดียวกับยูสเซอร์โปรเซสในระดับบนแต่เคอร์เนลนั้นจะให้สิทธิในการทำงานที่สูงกว่า คือกำหนดให้มีลำดับในการทำงาน (priority) ก่อนยูสเซอร์โปรเซสในชั้นที่สี่

ในชั้นที่สี่ เป็นส่วนของ ยูสเซอร์โพรเซส ได้แก่ เซลล์, เอดิเตอร์, คอมไพเลอร์, และโปรแกรมใช้งานอื่นๆ ระบบที่ทำงานอยู่ก็มีโพรเซสบางโพรเซสที่เริ่มต้นเมื่อบูตระบบและทำงานอยู่ตลอดเวลาจนกว่าระบบจะไม่ทำงาน เรียกว่า เดมอน (daemon) ซึ่งเป็นโพรเซสที่ทำงานในฉากหลัง ที่จะรอคอยเหตุการณ์ที่จะเกิดขึ้น เช่น แพ็กเก็ตข้อมูลที่มาจากเน็ตเวิร์ก แล้วจึงทำงาน เป็นต้น ถึงแม้ว่าเดมอนนั้นจะเป็นทำหน้าที่เป็นเซิร์ฟเวอร์ที่เริ่มต้นขึ้นอย่างอิสระและทำงานเป็นยูสเซอร์โพรเซส แต่อย่างไรก็ดี เดมอนจะต่างจากเซิร์ฟเวอร์ที่กล่าวถึงในชั้นที่สามที่มีสิทธิเหนือกว่า เดมอนนั้นจะถูกมองเหมือนเป็นยูสเซอร์โพรเซสธรรมดา ไม่ได้รับสิทธิพิเศษจากเคอร์เนลอย่างที่ส่วนจัดการหน่วยความจำและส่วนไฟล์ระบบได้รับ

5.2.2 การจัดการโพรเซส

โพรเซสในมินิกซ์มีรูปแบบใกล้เคียงกับหลักการของโพรเซสทั่วไปคือ สามารถสร้างโพรเซสลูกได้ และโพรเซสลูกที่สร้างขึ้นนั้นสามารถที่จะสร้างโพรเซสลูกต่อได้อีก ทำให้เกิดเป็นทรีของโพรเซสขึ้น ซึ่งในมินิกซ์นั้นยูสเซอร์โพรเซสทั้งหมดจะอยู่ภายใต้รูทของทรีเพียงอันเดียวเท่านั้นก็คือ init (ดูรูปที่ 14 ประกอบ) ในตอนเริ่มต้นระบบ หลังจากระบบได้เริ่มต้นเซิร์ฟเวอร์ต่างๆในชั้นที่สามได้แก่ ส่วนจัดการหน่วยความจำ, ส่วนจัดการระบบไฟล์ และเซิร์ฟเวอร์อื่นๆในกรณีที่มี หหมดแล้ว ระบบจะเริ่มรัน init เป็นยูสเซอร์โพรเซสอันดับแรก และเป็นรูทของยูสเซอร์โพรเซสทั้งหมด หลังจากนั้นจึงรันโปรแกรมอื่นๆเป็นโพรเซสลูกของ init ซึ่งอาจได้แก่ โปรแกรมเซลล์ เป็นต้น

การจัดการกับโพรเซสในมินิกซ์นั้นจะมีซีสเต็มคอลหลักอยู่สองอันคือ FORK และ EXEC FORK เป็นวิธีเดียวในการสร้างโพรเซสขึ้นมาใหม่ และ EXEC ใช้ในการเรียกโปรแกรมให้ทำงาน เมื่อโปรแกรมเริ่มทำงานจะมีการกำหนดส่วนของหน่วยความจำให้กับโพรเซสนั้นๆโดยดูจากค่าในส่วนหัวของโปรแกรม นอกจากนั้นยังต้องมีการเก็บข้อมูลต่างๆของโพรเซสอีกหลายอย่างเพื่อใช้ในการจัดการโพรเซส เช่น สถานะของโพรเซส เป็นต้น

ข้อมูลเกี่ยวกับโพรเซสทั้งหมดจะถูกเก็บอยู่ในตารางโพรเซส ตารางโพรเซสจะมีฟิลด์ที่เก็บข้อมูลของโพรเซสในส่วนที่เกี่ยวข้องกับเคอร์เนล, ส่วนจัดการหน่วยความจำ, ส่วนจัดการระบบ เมื่อมีการสร้างโพรเซสใหม่หรือยกเลิกการทำงานของโพรเซส ส่วนจัดการหน่วยความจำจะอัปเดตค่าในส่วนที่เกี่ยวข้องในตารางโพรเซสก่อนแล้วจึงส่งเมสเซจไปยังส่วนจัดการระบบไฟล์และเคอร์เนลเพื่อให้อัปเดตค่าในส่วนที่เกี่ยวข้องในตารางโพรเซสต่อไป

5.2.3 การสื่อสารระหว่างโพรเซส

การสื่อสารระหว่างโพรเซสจะใช้กลไกการส่งผ่านเมสเซจ ได้แก่ การส่ง, การรับ และการส่งและรอรับจากโพรเซสนั้น ในการรับส่งเมสเซจนั้นโพรเซสจะส่งที่อยู่ข้อมูลของเมสเซสเป็นพารามิเตอร์ กลไกการจัดการการส่งผ่านเมสเซจที่อยู่ในเคอร์เนลจะทำการคัดลอกเมสเซจจากผู้ส่งไปยังผู้รับ ซึ่งกลไกในเคอร์เนลนี้อาจถูกแทนด้วยฟังก์ชันที่ทำการส่งผ่านเมสเซจระหว่างเน็ตเวิร์กเพื่อสร้างเป็นระบบแบบกระจายได้

แต่ละโพรเซสหรือทาสก์สามารถรับส่งเมสเซจระหว่างกันในระดับเดียวกันได้ และสามารถรับจากชั้นที่ติดต่อกันอยู่ด้านล่างโดยตรงได้ ยูสเซอร์โพรเซสไม่สามารถติดต่อกับทาสก์ส่วนอินพุตเอาพุตได้โดยตรง ต้องติดต่อผ่านส่วนเซิร์ฟเวอร์ที่ให้บริการในชั้นที่สามเช่น ส่วนจัดการระบบไฟล์

เพื่อหลีกเลี่ยงปัญหาเรื่องการจัดการบัฟเฟอร์ มินิกซ์ใช้วิธีเรนเดวูส์ (rendezvous) ในการรับส่งเมสเซจ นั่นคือ เมื่อโพรเซสส่งเมสเซจไปยังโพรเซสที่ไม่ได้กำลังรอรับเมสเซจอยู่ ผู้ส่งจะบล็อกไปจนกว่าผู้รับจะพร้อมรับ ซึ่งวิธีนี้อาจทำให้มีความยืดหยุ่นน้อยลงแต่ก็เพียงพอและง่ายในการอินพลีเมนต์สำหรับระบบเล็กๆอย่างมินิกซ์

5.2.4 การจัดการตารางเวลาการทำงาน

ระบบอินเทอร์พรีตเป็นส่วนที่ทำให้เกิดการทำงานแบบมัลติโปรแกรมมิง โพรเซสจะบล็อกเมื่อมีการเรียกใช้ส่วนอินพุต เพื่อให้โพรเซสอื่นทำงานระหว่างรอส่วนอินพุต เมื่ออินพุตพร้อมโพรเซสที่กำลังทำงานอยู่จะถูกอินเทอร์พรีตโดยคิสก์, คีบอร์ด, หรือฮาร์ดแวร์อื่นๆ เพื่อให้มาทำงานในส่วนของทาสก์นั้นๆ นาฬิกาทำให้เกิดอินเทอร์พรีตเพื่อให้โพรเซสอื่นๆ ได้ทำงานโดยแบ่งกันทำงานตามช่วงเวลาที่กำหนด จากรูปที่ 14 ชั้นล่างสุดมีหน้าที่ในการปิดบังรายละเอียดเรื่องอินเทอร์พรีตโดยการเปลี่ยนให้อยู่ในรูปแบบของการส่งเมสเซจ เช่นเมื่อทาสก์จัดการอุปกรณ์อินพุตเอาท์พุตเสร็จ ก็จะส่งเมสเซจไปปลุกโพรเซสที่ทำการเรียกและบล็อกรออยู่กลับมาทำงานอีกครั้ง

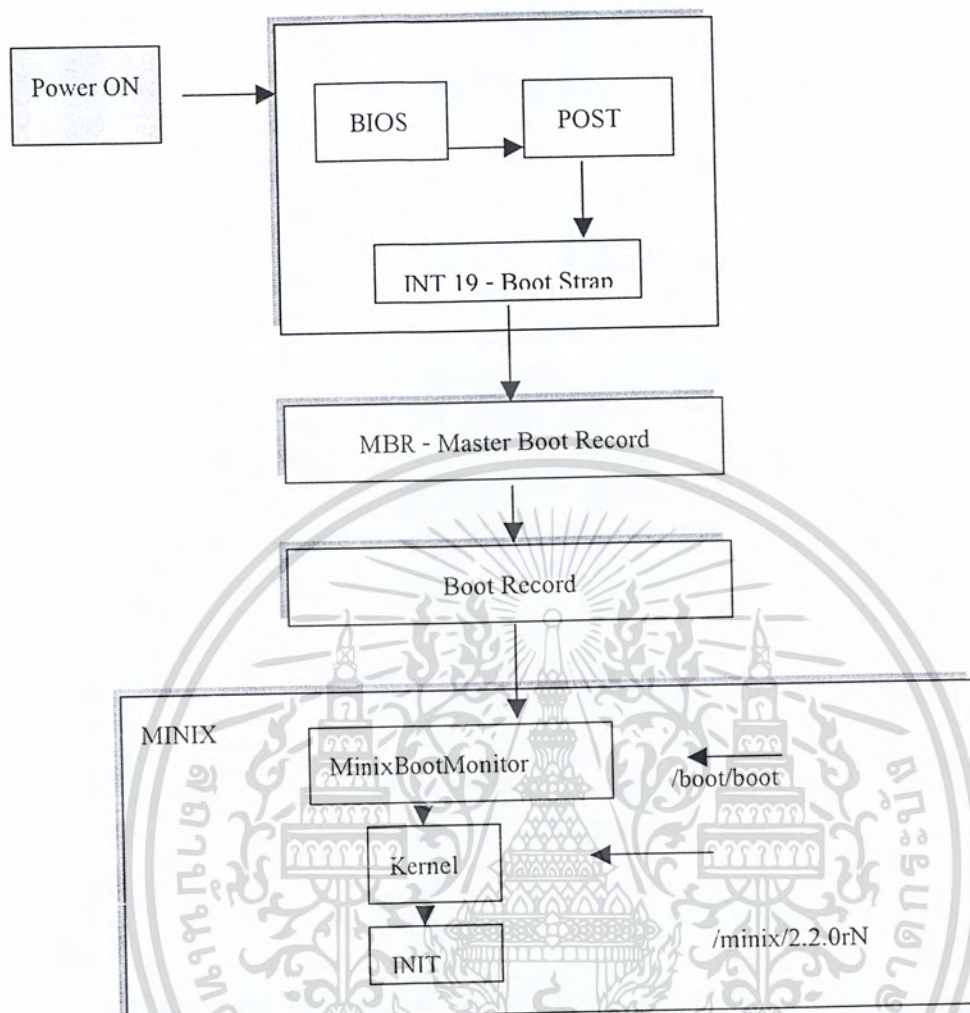
การจัดการตารางการทำงานจะใช้ระบบคิว 3 คิว ที่มีลำดับความสำคัญต่างกัน 3 ระดับ จากมากไปน้อยได้แก่ ทาสก์, เซิร์ฟเวอร์ และยูสเซอร์โพรเซส ตามลำดับ ภายในคิวของทาสก์และเซิร์ฟเวอร์โพรเซสโพรเซสที่ทำงานจะทำงานไปจนกว่าจะบล็อก แต่ในคิวของยูสเซอร์โพรเซสนั้นจะมีการวนผลัดกันทำงานไปเรื่อยๆ ตามช่วงเวลาที่ได้กำหนดไว้ หรือที่เรียกว่า ราวด์โรบิน (round robin)

ในการเลือกโพรเซสขึ้นทำงานนั้น ตัวจัดการจะตรวจดูว่าก่อนว่ามีทาสก์ที่พร้อมจะทำงานอยู่หรือไม่ ถ้ามีก็จะเลือกทาสก์ที่เข้าคิวอยู่ก่อนขึ้นมาทำงาน ถ้าไม่มีทาสก์ใดพร้อม ตัวจัดการก็จะไปดูว่ามีเซิร์ฟเวอร์โพรเซสที่พร้อมจะทำงานหรือไม่ ถ้าไม่มีก็จะไปดูยูสเซอร์โพรเซสต่อไป ถ้าในขณะนั้นไม่มีโพรเซสใดพร้อมที่จะทำงานอยู่เลย ตัวจัดการก็จะเลือกโพรเซส IDLE ขึ้นมา ซึ่งโพรเซส IDLE นี้ไม่มีการทำงานใดๆ เพียงแต่ทำการวนลูปเพื่อรอให้เกิดอินเทอร์พรีตขึ้นเท่านั้น

ในแต่ละช่วงสัญญาณนาฬิกา ตัวจัดการจะตรวจว่าถ้ามียูสเซอร์โพรเซสใดกำลังทำงานอยู่ และโพรเซสที่ทำงานอยู่นี้ได้ทำงานมาเกินระยะเวลา 100 มิลลิวินาทีหรือยัง ถ้าเกิน ตัวจัดการก็จะดูว่ามีโพรเซสอื่นที่รอจะทำงานอยู่หรือไม่ ถ้ามีตัวจัดการจะย้ายโพรเซสปัจจุบันไปต่อท้ายคิว และจะเลือกโพรเซสต่อมาในคิวขึ้นมาทำงานแทน สำหรับทาสก์และเซิร์ฟเวอร์โพรเซสนั้นจะไม่ถูกพรีเอมต์โดยสัญญาณนาฬิกา แต่จะทำงานไปเรื่อยจนกว่าจะบล็อก

5.2.5 กระบวนการบูตของมินิกซ์

เป็นไปดังนี้



รูปที่ 15 กระบวนการบูต

เมื่อเปิดเครื่องพีซีขึ้น ระบบจะเริ่มดำเนินการทำงานของไบออส ซึ่งเป็นการตรวจสอบระบบเบื้องต้นว่าใช้งานได้หรือไม่ หรือที่เรียกกันว่ากระบวนการ POST ซึ่งจะทำไปพร้อมๆ กับการเริ่มต้นฮาร์ดแวร์ต่างๆ ของระบบ (Initialize System) จากนั้นจึงเริ่มต้นโหลดระบบปฏิบัติการเพื่อส่งต่อการทำงานไปให้ระบบปฏิบัติการ ขั้นตอนนี้จะเริ่มจากโหลดส่วนแรกของดิสก์หรือที่เรียกกันว่า Boot Record จากนั้นส่วน Boot Strap (ซึ่งเป็นส่วนประกอบของ Boot Record เอง) จะโหลดโปรแกรมเล็กๆ ตัวหนึ่งใน /boot หรือ Minix Boot Monitor ซึ่งมีหน้าที่ในการโหลดส่วนเคอร์เนลของมินิกซ์ขึ้นมาจาก /minix จากนั้นจึงเป็นการทำงานในส่วนระบบปฏิบัติการแล้ว

ข้อพิเศษของระบบปฏิบัติการมินิกซ์ในกระบวนการบูตนี้อยู่ที่โปรแกรมใน /boot หรือ Minix Boot Monitor ซึ่งนอกจากจะทำการโหลดส่วนเคอร์เนลของระบบขึ้นมาแล้ว ยังสามารถส่งผ่านพารามิเตอร์

มิเตอร์ไปให้เคอร์เนลได้ เช่นตำแหน่ง IRQ และ IO ของ การ์ดเพิ่มเติมของระบบต่างๆ เช่นการ์ดอีเทอร์เน็ต มากไปกว่านั้นคือการเลือกระบบปฏิบัติการในการบูตได้ แนวคิดนี้ได้นำไปใช้ในระบบปฏิบัติการรุ่นน้องอย่างลินุกซ์ด้วย ซึ่งรู้จักกันในชื่อของ LILO หรือ Linux Loader นั่นเอง

5.2.6 โครงสร้างซอร์สโค้ดของเคอร์เนลระบบปฏิบัติการมินิกซ์

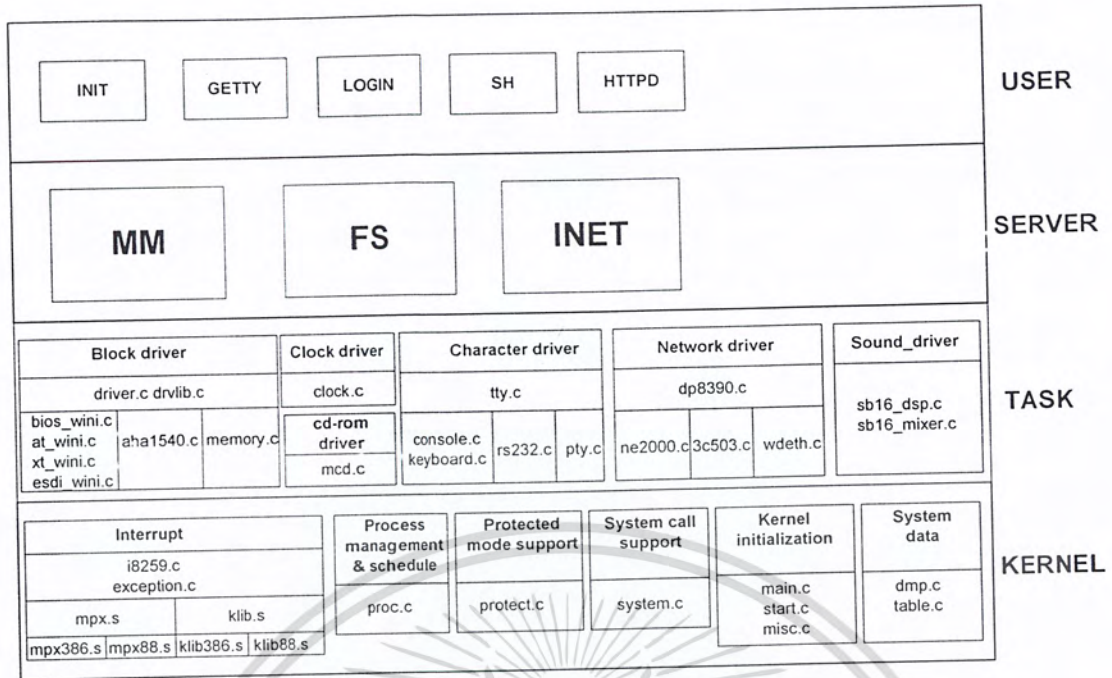
แบ่งออกเป็นสองที่คือ /usr/include กับ /usr/src แบ่งรายละเอียดได้ดังนี้

1. ใน /usr/include ประกอบไปด้วย POSIX Standard Header File ภายในแบ่งเป็น

sys/	POSIX Header เพิ่มเติม
minix/	Header File ที่ใช้ในมินิกซ์เท่านั้น
ibm/	เป็น Header File ที่ใช้สำหรับ IBM-C Specific Definition

2. ใน /usr/src ประกอบไปด้วยซอร์สโค้ดของมินิกซ์

kernel/	ประกอบไปด้วยโค้ดในชั้นที่หนึ่งและสอง
mm/	ส่วนจัดการหน่วยความจำ
fs/	ส่วนจัดการระบบไฟล์
lib/	สำหรับไลบรารีของมินิกซ์
tools/	โค้ดของ INIT
boot/	โค้ดของ Minix boot Monitor และ Boot block
commands/	โค้ดของคอมมานด์ต่างๆ
test/	โค้ดรูทีนในการทดสอบระบบปฏิบัติการ
inet/	โค้ดในส่วนสนับสนุนเน็ตเวิร์ก



รูปที่ 16 โครงสร้างแสดงฟังก์ชันการทำงานของซอร์สโค้ดไฟล์ต่างๆ

เมื่อทำการคอมไพล์แล้วจะได้ไฟล์ที่สามารถเรียกทำงานได้ ออกมาดังนี้



จะได้ไฟล์ออกมา 5 ไฟล์คือ kernel, mm, fs, inet และ init ทั้งหมดจะถูกโปรแกรม installboot รวมเป็นไฟล์อิมเมจเพียงไฟล์เพื่อใช้ในการเริ่มต้นระบบและเก็บไว้ใน /minix

ส่วน <minix src> เป็นซอร์สโค้ดในแพลตฟอร์มมินิกซ์ใช้ในการคอมไพล์
 ฐานข้อมูลทั้งหมด ไม่ขึ้นต่อกันทั้งฐานข้อมูลเองและผู้พัฒนา (หมายความว่า ผู้พัฒนาแต่ละคนจะมีฐานข้อมูลดังกล่าวนี้แยกกันอิสระ ไม่ขึ้นต่อกัน แต่ใช้วิธีการ patch และ diff ในการซิงค์โคโรไนซ์ไฟล์ทั้งหมดในฐานข้อมูลของแต่ละผู้พัฒนา)

- ทางฝั่ง target แบ่งออกเป็นสอง target คือ
 1. แพลตฟอร์มมินิกซ์ที่ทำงานอยู่บนเครื่องพีซี-คอมพิวเตอร์เพื่อบริการให้ง่ายในการอ้างอิง จะเรียกแพลตฟอร์มนี้ว่า minix-pc สามารถใช้ได้ทั้ง vmware หรือเครื่องจริงๆ ก็ได้
 2. แพลตฟอร์มมินิกซ์บนบอร์ดคอม86 โดยเรียกว่ามินิกซ์บนคอม86
- รูปแบบในการพัฒนาเป็นไปตามรูปแบบดังนี้ (ดูตามรูป)
 1. การแก้ไขไฟล์ใน <master src> นั้นทำได้สองทางคือ
 - ทางแรกใช้เอดิเตอร์
 - 1.1. ใช้เอดิเตอร์บนวินโดวส์ ทำการแก้ไขซอร์สโค้ดของมินิกซ์ การบันทึกให้บันทึกในรูปแบบของยูนิกซ์เทกซ์ไฟล์ (Unix-text-file Format)
 - 1.2. ใช้โปรแกรม diff บนโปรแกรม cygwin ทำการสร้างไฟล์ diff จาก <master src> (โปรแกรม cygwin คือโปรแกรมอิมูเลเตอร์ระบบปฏิบัติการยูนิกซ์ในสภาพแวดล้อมของวินโดวส์ สามารถหารายละเอียดและหาดาวน์โหลดได้ที่ www.redhat.com/cygwin)
 - 1.3. ไฟล์ diff ที่ได้เก็บไว้ใน <master diff>
 - 2. ทางที่สองคือการนำไฟล์ diff จากนักพัฒนาคนอื่นมาทำการ patch เข้ากับซอร์สโค้ดของตนเองเพื่อให้ได้ซอร์สโค้ดที่มีการปรับปรุงเป็นปัจจุบันอย่างเสมอ
 - 2.1. นำไฟล์ diff มาทำการ patch กับ <master src>
 - 2.2. นำไฟล์ diff ที่ได้รับมาเก็บใน <master diff>
 3. ส่งไฟล์ diff ไปที่ minix-pc เพื่อทำการคอมไพล์
 4. ทำการคอมไพล์เคอร์เนลของมินิกซ์ บน minix-pc
 5. สำหรับเคอร์เนลของ มินิกซ์ ที่ทำการคอมไพล์นั้น แบ่งการใช้งานออกได้เป็นสองแบบคือ
 - 5.1. นำไปใช้บนมินิกซ์บนคอม86 ซึ่งทำได้โดยส่งไฟล์มาเก็บไว้ที่ <master kernel> จากนั้นจึงส่งไปที่บอร์ดคอม86 เพื่อทดสอบการทำงานของเคอร์เนล (ในทางปฏิบัตินั้นสามารถเลือกที่จะไม่ส่งไฟล์กลับมาที่โฮส ก็ได้ สามารถส่งไฟล์ไปที่บอร์ดคอม86 ได้โดยตรงโดยใช้โปรแกรม term หรือ minccm บนมินิกซ์ส่งไฟล์ก็สามารถทำได้)
 - 5.2. ยังสามารถทำงานได้บน minix-pc

ตรงนี้ข้อสังเกตได้ว่า ในการพัฒนานั้น เราจะแบ่งการพัฒนาออกเป็นสองทาง คือ

- ทางแรก เป็นการพัฒนาเพื่อ ลดขนาด และปรับปรุงซอร์สโค้ดของ มินิกซ์ แต่ยังคงไว้ซึ่งส่วนที่ยังทำงานได้บนพีซี เช่น ไม่ได้ปรับปรุงเรื่องอินเทอร์รัปต์ (interrupt) เป็นต้น เพื่อใช้ในการทำการทดสอบบนเครื่องพีซีได้

- ทางที่สองเป็นการพัฒนาเพื่อปรับปรุงให้สามารถทำงานได้บนบอร์ดคอม86 โดยเน้นไปที่ส่วนการเชื่อมต่อกับฮาร์ดแวร์ (Hardware interface) ระหว่างมินิกซ์กับบอร์ดคอม86

จากการพัฒนาทั้งสองทาง จะรวมกันเมื่อมินิกซ์ สามารถทำงานได้บนบอร์ดคอม86 ได้ตามทางหลัง ซึ่งเมื่อเสร็จสมบูรณ์ จะได้มินิกซ์บนคอม86 ที่มีความสามารถดังต่อไปนี้

- ✓ มีขนาดเล็กและทำงานได้บนบอร์ดคอม86 ได้อย่างราบรื่น
- ✓ มีเฉพาะคิโวลซ์ไครเวอร์ที่จำเป็นต่อการทำงาน และเหมาะสมต่อบอร์ดคอม86 มากที่สุด
- ✓ มีความสามารถด้านเน็ตเวิร์ก สามารถติดต่อสื่อสารโดยใช้โพรโทคอล TCP/IP ได้
- ✓ สามารถคอมไพล์โปรแกรมขนาดเล็กได้บนบอร์ดคอม86

6 นำเคอร์เนลที่ได้จากการคอมไพล์ไปทดสอบบนบอร์ดคอม86

กระบวนการดังกล่าวนี้ ถ้าเลือกมินิกซ์เป็นแพลตฟอร์มในการพัฒนา สามารถพัฒนาได้ในเครื่องเดียวได้ทั้งหมด ไม่ว่าจะเป็นเอดิเตอร์เช่น vi จากนั้นใช้โปรแกรม patch หรือ diff เมื่อพัฒนาแล้วก็ทำการคอมไพล์บนมินิกซ์ แต่อย่างไรก็ตาม การพัฒนาจะขออ้างถึงการพัฒนามนวินโดวส์เป็นหลัก

6.2 เอดิเตอร์ที่ใช้ในการพัฒนา

เนื่องมาจากระบบปฏิบัติการมินิกซ์เป็นระบบปฏิบัติการที่ทำงานคล้ายกับระบบยูนิกซ์ ดังนั้นเพื่อให้รูปแบบไฟล์ทั้งหมด ใช้งานร่วมกับระบบมินิกซ์ได้ จึงกำหนดให้บันทึกไฟล์ในรูปแบบยูนิกซ์เทกซ์ไฟล์

ดังนั้นเอดิเตอร์ที่ใช้ในการพัฒนาจะต้องมีความสามารถในการบันทึกไฟล์ในรูปแบบยูนิกซ์เทกซ์ไฟล์ได้ เช่น โปรแกรมเอดิเตอร์อย่าง EditPlus มนวินโดวส์ หรือเอดิเตอร์พื้นฐานบนระบบยูนิกซ์อย่าง vi เป็นต้น

6.3 ล็อกไฟล์สำหรับการพัฒนา

ล็อกไฟล์ที่ใช้ในการพัฒนานั้นให้ใช้ชื่อว่า logfile.m การที่ตั้งนามสกุลไฟล์เป็น .m เนื่องมาจากใช้วิธีการบันทึกไฟล์ในรูปแบบแบบยูนิกซ์เทกซ์ไฟล์ ดังนั้นเพื่อหลีกเลี่ยงความสับสนกับไฟล์นามสกุล .txt จึงตั้งให้ใช้นามสกุลอื่นคือ .m (ย่อมาจาก minixC)

ล็อกไฟล์ไม่จำเป็นต้องมีไฟล์เดียว เนื่องมาจากการพัฒนาครั้งหนึ่ง ไม่ได้ทำการปรับปรุงแก้ไขทุกไฟล์ ในไครเทอรี แต่โดยมากมักจะเกี่ยวข้องกับไครเทอรีเพียงหนึ่งหรือสองเท่านั้น เช่น การพัฒนาในช่วงเริ่มต้นจะเกี่ยวข้องกับไครเทอรี /src/kernel ดังนั้นล็อกไฟล์ logfile.m จะมีอยู่ในไครเทอรี /src/kernel ด้วย การพัฒนาครั้งต่อไปอาจจะไปพัฒนาใน /src/boot ซึ่งจะมีล็อกไฟล์อยู่ในไครเทอรี /src/boot ด้วยเช่นเดียวกัน

เนื่องจากล็อกไฟล์แต่ละล็อกไฟล์อยู่ในไครเทอรีของการพัฒนา ดังนั้นการรวบรวมข้อมูลของการพัฒนาทั้งหมด สามารถทำได้ง่าย รวดเร็ว ไม่ซับซ้อน เพราะสามารถหาข้อมูลทั้งหมดได้จากล็อกไฟล์เพียงไฟล์เดียว นอกจากจะทำให้เข้าถึงข้อมูลของการเปลี่ยนแปลงต่างๆ ได้โดยง่ายแล้ว ยังทำให้ซอร์สโค้ด

ของมินิกซ์ไม่ต้องเพิ่มข้อมูลที่เกี่ยวข้องกับการแก้ไข (เอาข้อมูลการเปลี่ยนแปลงแก้ไขไปรวมไว้ที่ล็อกไฟล์เพียงไฟล์เดียว) นั่นหมายความว่าซอร์สโค้ดของมินิกซ์ยังคงสภาพเหมือนเดิมมากที่สุดซึ่งเป็นสิ่งที่ดีต่อการเผยแพร่ซอร์สโค้ดของมินิกซ์ภายหลัง

ล็อกไฟล์แต่ละไฟล์ จะถูกบันทึกโดยนักพัฒนาแต่ละคนเพื่อบอกถึงความเปลี่ยนแปลงที่ได้ทำโดยมีรูปแบบเดียวกัน ยกตัวอย่างเช่นไฟล์ logfile.m ในไดเรกทอรี /src/kernel มีรูปแบบคือ (ให้ใช้ TAB ในการเว้นช่องว่าง)

ข้อสังเกตอีกข้อหนึ่งคือล็อกไฟล์ก็เป็นไฟล์ที่อยู่ในไดเรกทอรีต่างๆ ในการพัฒนา ดังนั้นเมื่อเรียกใช้โปรแกรม diff ในการหาความแตกต่างของไฟล์ ล็อกไฟล์จะเป็นส่วนหนึ่งด้วย ดังนั้นในระหว่างที่มีการพัฒนา นักพัฒนาแต่ละคนไม่จำเป็นต้องส่งล็อกไฟล์ไปมาระหว่างกัน เพื่อบอกว่าทำการแก้ไขอะไรไปบ้าง นักพัฒนาคนนั้นเพียงแค่ทำการแก้ไขไฟล์ แล้วบันทึกการเปลี่ยนแปลงลงในล็อกไฟล์ จากนั้นจึงใช้กระบวนการปกติคือ การใช้โปรแกรม diff จากนั้นเมื่อนักพัฒนาคนอื่นเอาไฟล์ diff ไปแล้วใช้โปรแกรม

```
/src/kernel/logfile.m
main.c      12/10/2001  12:30AM      JOJOEd
    -change to show string
    -add some comment concern with starting up
    ..
the operating system on com86 board
i8259.c     13/10/2001  4:01AM      POM
    -modify to work with com86 in function
    interrump()
    ..
    ..
```

patch กับซอร์สโค้ดของदनนอกจากจะได้ซอร์สโค้ดที่มีการปรับปรุงแล้ว จะได้ล็อกไฟล์ที่บันทึกข้อมูล

ตำแหน่งของไฟล์ล็อก

ชื่อไฟล์ที่ทำการแก้ไข DD/MM/YYYY HH:MM(AM,PM) ชื่อนักพัฒนา
 -แก้ไขอะไรไปบ้าง ในฟังก์ชันใด ทำอะไรได้บ้าง
 -ยังบอกรายละเอียดเยอะยิ่งดี
 ...

การเปลี่ยนแปลงจากนักพัฒนาคนอื่นๆ ได้อีกด้วย จะเห็นได้ว่าเป็นประโยชน์ทั้งสองทาง

การสร้างล็อกไฟล์ครั้งแรกภายใต้ไดเรกทอรีต่างๆ ให้บอกในไฟล์ต่างหาก ซึ่งขอให้แนบมากับไฟล์ที่เกิดจากการ diff ด้วย (เนื่องมาจากโปรแกรม diff ไม่สามารถสร้างไฟล์ขึ้นมาได้เอง ดังนั้น ถ้ามีการเพิ่มไฟล์หรือลบไฟล์ diff จะบอกได้แต่เพียงว่าไฟล์นี้เพิ่มหรือลบเข้ามาหรือออกไป แต่ไม่สามารถสร้างหรือลบให้ได้) ดังนั้นทุกครั้งที่มีการเพิ่มหรือลบไฟล์ในไดเรกทอรี ในการพัฒนาครั้งใดๆ ให้ส่งไฟล์แนบมาบอกด้วยว่าไฟล์ใดบ้าง โดยให้ส่งแนบมากับ diff ไฟล์ในครั้งนั้นๆ

เช่นในการพัฒนาครั้งนั้นๆ นักพัฒนา ก. พัฒนาในไดเรกทอรี /src/fs ซึ่งยังไม่มีนักพัฒนาคนใดไปยุ่งเกี่ยว ดังนั้นในไดเรกทอรีนี้จะไม่มียล็อกไฟล์ นักพัฒนา ก. จึงจำเป็นต้องสร้างล็อกไฟล์ logfile.m ในไดเรกทอรี /src/fs ขึ้นเพื่อใช้ในการบันทึกถึงข้อมูลการเปลี่ยนแปลงต่างๆ ที่เกิดขึ้นในการพัฒนา จาก

นั่นจึงทำการพัฒนาไปตามปกติ เมื่อทำการพัฒนาได้จนได้ผลเป็นที่น่าพอใจ จึงใช้โปรแกรม diff ในการหาข้อแตกต่าง แล้วจึงส่งไฟล์ diff ไปให้นักพัฒนาคนอื่น นอกจากนักพัฒนา ก. จะต้องส่งไฟล์ diff ไปให้นักพัฒนาคนอื่นแล้ว ยังต้องบอกอีกด้วยว่า ก่อนจะทำการพัฒนาต่อไปนั้น ให้สร้างไฟล์ logfile.m ในไดเรกทอรี /src/fs เสียก่อนจึงสามารถใช้โปรแกรม patch ได้ (จากที่ได้กล่าวไปแล้ว ว่าคู่โปรแกรม diff-patch ไม่สามารถสร้างหรือลบไฟล์ใดๆ ได้ ดังนั้นจึงต้องเป็นภาระหน้าที่ของนักพัฒนาเองที่จะต้องดูแลจัดการเรื่องเหล่านี้) การพัฒนาต่อไปจึงจะสามารถดำเนินไปได้อย่างถูกต้อง

6.4 ข้อกำหนดของการตั้งชื่อไฟล์ diff

มีข้อกำหนดการตั้งชื่อเป็น

<main directory>-[<old version>]-<new version>

- main directory คือ ไดเรกทอรีที่ใช้ในการเปรียบเทียบครั้งนั้นๆ
- old version เป็นเวอร์ชันเก่า
- new version เป็นเวอร์ชันใหม่

ตัวอย่างเช่นในการพัฒนาครั้งหนึ่ง ทำการพัฒนาในไดเรกทอรี /src/kernel และมีเวอร์ชันจาก 0.1.1 ไปเป็น 0.1.2 ดังนั้น ตั้งชื่อไฟล์ diff ได้เป็น

kernel-0.1.1-0.1.2

อย่างไรก็ตามสามารถละ old version ได้ หรือ สามารถตั้งชื่อไฟล์แบบนี้ได้

kernel-0.1.2

หมายความว่าไฟล์ diff นี้สร้างมาจากการเปรียบเทียบระหว่างเวอร์ชันก่อนหน้าเวอร์ชันที่ใช้เป็นชื่อไฟล์ หรือ 0.1.2 ดังนั้นเวอร์ชันก่อนหน้าคือ 0.1.1 (ลดลงหนึ่ง) นั่นคือเปรียบเทียบกันระหว่างเวอร์ชัน 0.1.1 กับ 0.1.2 เป็นต้น

หรือ

kernel-0.1.3-0.1.10

การตั้งชื่อแบบนี้มีความหมายว่า ทำการเปรียบเทียบกันระหว่างสองเวอร์ชันคือ 0.1.3 กับ 0.1.10

บทที่ 7

การพัฒนามินิคลับบนคอม86

7.1 บทนำ

การนำซอฟต์แวร์ระบบมาปรับปรุงเพื่อให้ทำงานได้บนแพลตฟอร์มใหม่ได้นั้น มีองค์ประกอบเพื่อให้บรรลุเป้าหมายในการพัฒนาหลายอย่าง อาทิเช่น ความแตกต่างระหว่างแพลตฟอร์มเดิมกับแพลตฟอร์มใหม่ ข้อจำกัดของแพลตฟอร์มใหม่ คุณลักษณะโดยละเอียดของแพลตฟอร์มใหม่และความเข้าใจในซอฟต์แวร์ระบบที่นำมาพัฒนาที่เป็นสิ่งสำคัญที่ไม่สามารถละเลยได้เช่นเดียวกัน

ซอฟต์แวร์ระบบที่กล่าวถึงคือระบบปฏิบัติการมินิคลับ เป็นระบบปฏิบัติการเพื่อการศึกษา ซึ่งได้กล่าวไว้แล้วถึงที่มาในการเลือกระบบปฏิบัติการมินิคลับและคุณลักษณะเบื้องต้น ปัจจุบันได้มีการนำมินิคลับไปพัฒนาให้สามารถทำงานได้ในหลายแพลตฟอร์ม อาทิเช่น โอเพนเอ็มพีซีคอมแพคทีเบิล สปาร์ก แมคอินทอช อะมิกา หรืออาตาริ เป็นต้น

แพลตฟอร์มที่ใช้ในโครงการนี้คือแพลตฟอร์มคอม86 เป็นแพลตฟอร์มที่ได้พัฒนาขึ้น โดยห้องปฏิบัติการระบบควบคุมด้วยสมองกลสถาบันพระจอมเกล้าเจ้าคุณทหารลาดกระบัง และเลือกระบบปฏิบัติการมินิคลับเวอร์ชัน 2.0.2 บนแพลตฟอร์มโอเพนเอ็มพีซีคอมแพคทีเบิลเป็นฐานในการพัฒนา เนื่องจากเป็นแพลตฟอร์มที่มีความคล้ายคลึงทางฮาร์ดแวร์กับแพลตฟอร์มคอม86 มากที่สุด

จุดประสงค์หลักของโครงการคือการพัฒนาและปรับปรุงระบบปฏิบัติการมินิคลับให้สามารถทำงานได้บนแพลตฟอร์มคอม86 และให้คำนิยามคำว่า “ทำงานได้” มีความหมายในเชิงปฏิบัติดังนี้

- สามารถเริ่มต้นระบบได้อย่างถูกต้องและสมบูรณ์บนแพลตฟอร์มคอม86
- สามารถแสดงผลผ่านโปรแกรมเทอร์มินอลได้อย่างถูกต้อง
- สามารถรับข้อมูลที่ป้อนผ่านทางคีย์บอร์ดผ่านโปรแกรมเทอร์มินอลได้อย่างถูกต้อง
- สามารถเชื่อมต่อเข้าสู่ระบบเครือข่ายคอมพิวเตอร์ได้อย่างถูกต้อง
- ระบบมีความเสถียรภาพเทียบเท่ากับระบบปฏิบัติการมินิคลับบนแพลตฟอร์มโอเพนเอ็มพีซีคอมแพคทีเบิลที่ทำงานในสถานะแวดล้อมแบบ 16 บิต และในโหมดหน่วยความจำจริง

7.2 เวอร์ชันในการพัฒนา

ในการพัฒนามินิคลับบนคอม86 ใช้รูปแบบการพัฒนาในลักษณะการพัฒนาต่อยอดจากที่มีอยู่แล้วไปเรื่อยๆ หรือที่เรียกว่าการพัฒนาในลักษณะเวอร์ชัน พัฒนาต่อเติมไปจากเวอร์ชันเดิมขึ้นไปยังเวอร์ชันใหม่ โดยมุ่งเป้าไปที่เวอร์ชันนั้นๆ ว่าต้องมีการบรรลุจุดประสงค์ใดจุดประสงค์หนึ่งตามที่ได้ออกแบบไว้ อย่างครบถ้วน เช่นออกแบบให้เวอร์ชัน 0.2.1 สามารถทำงานได้บนแพลตฟอร์มคอม86 เป็นต้น ทำให้ในการพัฒนาสามารถตรวจสอบได้ว่าดำเนินการไปถึงจุดใดแล้ว จุดใดที่ยังไม่สามารถทำได้ เกิดปัญหาอะไรขึ้น สามารถย้อนกลับมาตรวจสอบความผิดพลาดได้ตลอดเวลา และเป็นการสำเนาข้อมูลไปในตัว ทั้งนี้และทั้งนั้นเพื่อลดปัญหาและข้อผิดพลาดที่อาจจะเกิดขึ้นได้ในขณะพัฒนาให้เหลือน้อยที่สุดเท่าที่จะมี

โอกาสในการเกิดขึ้นได้ รวมไปถึงการสนับสนุนการพัฒนาโดยโปรแกรมเมอร์มากกว่าหนึ่งคนที่สามารถพัฒนาร่วมกันได้เป็นอย่างดี

ระบบปฏิบัติการมินิซ์เวอร์ชัน 2.0.2 บนแพลตฟอร์มไอบีเอ็มพีซีคอมพิวเตอร์ทีเบิลที่เป็นฐานในการนำมาพัฒนาและปรับปรุงเป็นมินิซ์บนคอม86 มีปรัชญาในการออกแบบโค้ดในส่วนล่างสุดของระบบหรือส่วนที่ติดต่อกับฮาร์ดแวร์โดยพยายามใช้ไบออส (BIOS) ของระบบให้น้อยที่สุด โดยออกแบบให้จัดการกับฮาร์ดแวร์โดยตรงโดยระบบปฏิบัติการ จุดประสงค์เพื่อให้ระบบทำงานได้อย่างมีประสิทธิภาพสูงสุดและมีเสถียรภาพสูงสุด จากเหตุผลดังกล่าว ทำให้โค้ดในส่วนที่ติดต่อกับฮาร์ดแวร์ทั้งหมดต้องกระทำการพิจารณาปรับปรุงและเปลี่ยนแปลง โดยยึดแนวทางในการปรับปรุงให้ทำงานร่วมกับไบออสที่อยู่บนแพลตฟอร์มคอม86 ให้มากที่สุด เพื่อให้การพัฒนาสามารถทำได้อย่างรวดเร็วและง่ายต่อการแก้ไขปัญหาเมื่อมีปัญหาเกิดขึ้น

อนึ่งไบออสในแพลตฟอร์มคอม86 ได้ออกแบบให้เข้ากันได้กับไบออสของไอบีเอ็มพีซีคอมพิวเตอร์ทีเบิลมากที่สุดเท่าที่จะเป็นไปได้ นอกจากนี้ยังเพิ่มในส่วนที่ช่วยในการพัฒนา ไม่ว่าจะเป็นการโอนถ่ายอิมเมจ (Image) ของดิสก์ลงในดิสก์ของแพลตฟอร์มคอม86 การแสดงผลและรับข้อมูลผ่านทางโปรแกรมเทอร์มินอล รวมไปถึงไบออสเซอร์วิสรูทีน (Bios Service Routine) หรือบริการของไบออส ซึ่งได้ออกแบบให้สนับสนุนเซอร์วิสรูทีนที่จำเป็นเกือบจะครบทั้งหมด ทั้งหมดนี้ช่วยให้การพัฒนาระบบปฏิบัติการใหม่บนแพลตฟอร์มคอม86 ทำได้ง่ายขึ้น และลดเวลาในการพัฒนาดังได้

จากการระดมความคิด และการค้นคว้าหาข้อมูลเบื้องต้นของระบบปฏิบัติการมินิซ์ สามารถแจกแจงงานที่จะต้องจัดการเพื่อให้ระบบปฏิบัติการมินิซ์สามารถทำงานบนแพลตฟอร์มคอม86 ได้ดังต่อไปนี้

1. พิจารณาคัดดีไวซ์ใดเวอร์ที่ไม่จำเป็นออก เพื่อลดขนาดของเคอร์เนล
2. พิจารณาปรับปรุงและเปลี่ยนแปลงเรื่องการแสดงผลให้ใช้บริการของไบออส
3. พิจารณาปรับปรุงและเปลี่ยนแปลงเรื่องการรับข้อมูลเข้าให้ใช้บริการไบออส
4. พิจารณาปรับปรุงและเปลี่ยนแปลงในส่วนที่ขึ้นอยู่กับฮาร์ดแวร์ทั้งหมด
5. พิจารณาปรับปรุงและเปลี่ยนแปลงดีไวซ์ใดเวอร์ของเวลาของระบบ เป็นส่วนสำคัญที่สุดของระบบ
6. พิจารณาปรับปรุงและเปลี่ยนแปลงดีไวซ์ใดเวอร์ในส่วนการอ่านเขียนข้อมูลกับแฟลชดิสก์
7. พิจารณาปรับปรุงและเปลี่ยนแปลงดีไวซ์ใดเวอร์ของอีเทอร์เน็ต

พิจารณาปรับปรุงและเปลี่ยนแปลงในส่วนการสนับสนุนโปรเทกต์โหมดเนื่องจากโปรเซสเซอร์ของแพลตฟอร์มคอม86 ไม่สนับสนุนการทำงานในโหมดนี้

ให้สังเกตว่าในข้อสองและข้อสี่นั้น จำเป็นที่จะต้องทำไปพร้อมๆ กัน เนื่องจากว่า ถ้าทำเฉพาะข้อสองที่ว่าทำให้สามารถแสดงผลได้ แต่ไม่ทำข้อสี่ที่ว่าแก้ไขปรับปรุงส่วนที่ขึ้นกับฮาร์ดแวร์ทั้งหมด ก็ไม่สามารถทำให้มินิซ์ทำงานได้บนคอม86 หรือกลับกันถ้าทำเฉพาะข้อสี่ไม่ทำข้อสองก็ไม่สามารถแสดงผลได้ ไม่สามารถทราบได้เช่นเดียวกันว่าทำข้อสี่ถูกต้องหรือไม่ โดยมากส่วนติดต่อกับฮาร์ดแวร์ที่มีการปรับปรุงนั้น จะเป็นส่วนที่เกี่ยวข้องกับระบบอินเทอร์พรีตของระบบ เพราะมีรายละเอียดแตกต่างกับไอบีเอ็มพีซีคอมพิวเตอร์ทีเบิลมากพอสมควร

จากปัญหาเบื้องต้นในเรื่องนี้ ทั้งที่มาจากระบบปฏิบัติการมินิกซ์ที่มีส่วนที่ติดต่อกับฮาร์ดแวร์โดยพยายามใช้ไบออสของระบบให้น้อยที่สุด เช่นในเรื่องของการแสดงผลหรือรับข้อมูล และส่วนที่เกี่ยวข้องกับระบบอินเทอร์พรีตของระบบ จากการพิจารณาเราสามารถลดปัญหาในเบื้องต้นนี้ได้โดยการพัฒนาในส่วนที่ติดต่อกับฮาร์ดแวร์โดยตรงให้มาใช้บริการผ่านไบออส และทำให้สามารถทำงานได้อย่างถูกต้องในเครื่องไอบีเอ็มพีซีคอมแพคทิเบิลเสียก่อน เช่นเรื่องของการแสดงผลให้มาใช้บริการของไบออสในการแสดงผล ซึ่งจะเห็นว่า ถ้าเราสามารถแก้ไขให้แสดงผลโดยใช้ไบออสของระบบแทนที่จะจัดการกับฮาร์ดแวร์โดยตรง ก็สามารถที่จะแสดงผลได้อย่างถูกต้องได้ในคอม86 เช่นเดียวกัน ซึ่งเราสามารถลดปัญหาในส่วนนี้ไปได้ และเมื่อสามารถแสดงผลได้ ก็สามารถพัฒนาต่อในส่วนที่เกี่ยวข้องกับระบบอินเทอร์พรีตของระบบได้อย่างถูกต้องต่อไป

ดังนั้นจากวิธีการแก้ไขปัญหาในเบื้องต้นนี้ ทำให้มีวิธีในการพัฒนาโดยแบ่งสายของการพัฒนาออกเป็นสองสาย คือ

สายแรก เป็นการพัฒนาเพื่อลดขนาดและปรับปรุงซอร์สโค้ดของมินิกซ์ แต่ยังคงไว้ซึ่งส่วนที่ยังทำงานได้บนเครื่องไอบีเอ็มพีซีคอมแพคทิเบิล โดยมุ่งเน้นในการพัฒนาให้ระบบทำงานผ่านไบออสให้มากที่สุด และไม่ได้ปรับปรุงเรื่องอินเทอร์พรีต เพื่อให้สามารถใช้ในการทำการทดสอบการทำงานบนเครื่องไอบีเอ็มพีซีคอมแพคทิเบิลได้

สายที่สอง เป็นการพัฒนาเพื่อปรับปรุงให้สามารถทำงานได้บนบอร์ดคอม86 โดยมุ่งเน้นไปที่ส่วนการเชื่อมต่อกับฮาร์ดแวร์ของแพลตฟอร์มคอม86 กับระบบปฏิบัติการมินิกซ์ เช่น การพัฒนาปรับปรุงในส่วนที่เกี่ยวข้องกับระบบอินเทอร์พรีตของระบบเป็นต้น

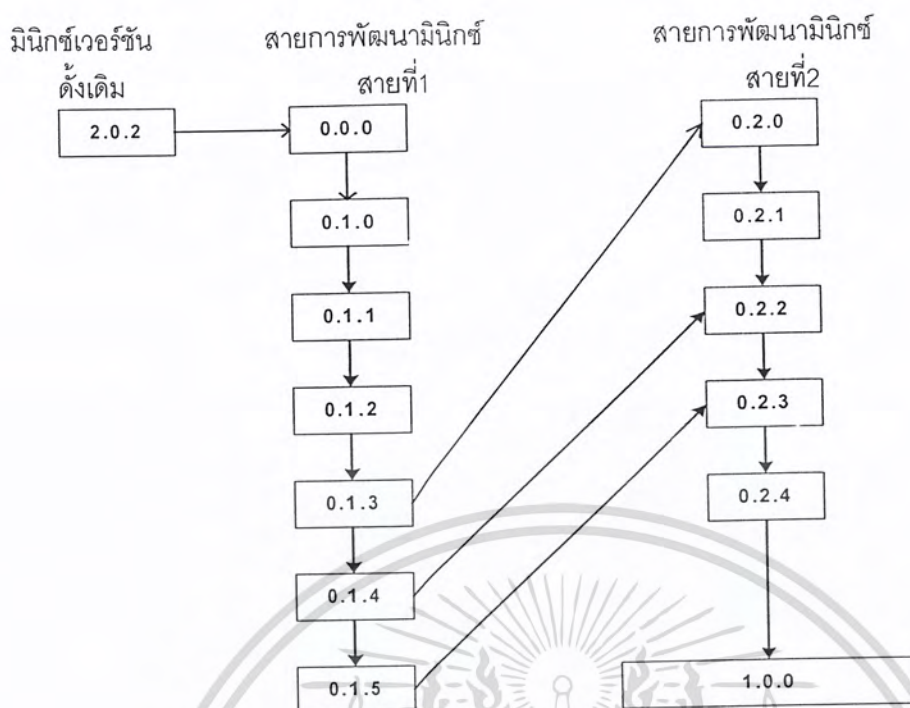
จากแนวทางดังกล่าวนี้เองทำให้สามารถสรุปแนวทางและจุดประสงค์ของการพัฒนาในแต่ละเวอร์ชันได้ดังต่อไปนี้

เวอร์ชันในการพัฒนา	จุดประสงค์ในการพัฒนา
สายการพัฒนาสายที่ 1	
0.0.0	เวอร์ชันต้นฉบับ สำเนาจากมินิกซ์ เวอร์ชัน 2.0.2
0.1.0	เวอร์ชันสำเนาข้อมูล เป็นเวอร์ชันที่สำเนามาจากเวอร์ชัน 0.0.0
0.1.1	ตัดดีไวส์ไดรเวอร์ที่ไม่จำเป็นออก และลบไฟล์ที่ไม่ได้ใช้
0.1.2	ปรับปรุงและเปลี่ยนแปลงเรื่องการแสดงผล ให้ใช้บริการของไบออส
0.1.3	ปิดการทำงานของคีย์บอร์ดดีไวส์ไดรเวอร์

	มุ่งเน้นไปที่การพัฒนาในส่วนที่เกี่ยวข้องกับ อินเทอร์เน็ตบนแพลตฟอร์มคอม86 ก่อน
0.1.4	ปรับปรุงและเปลี่ยนแปลงดีไวส์ไดรเวอร์ใน ส่วนการอ่านเขียนข้อมูลกับดิสก์ให้มาใช้ บริการของไบออส
0.1.5	ปรับปรุงและเปลี่ยนแปลงเรื่องการรับข้อมูล เข้า โดยใช้บริการไบออส
0.1.6	พิจารณาตัดโค้ดในส่วนของการสนับสนุน โปรเทกต์เทคใหม่
สายการพัฒนาสายที่ 2	
0.2.0	เวอร์ชันสำเนาจากเวอร์ชัน 0.1.3
0.2.1	ปรับปรุงและแก้ไขส่วนที่เกี่ยวข้องกับระบบ อินเทอร์เน็ตของระบบ ปรับปรุงและเปลี่ยนแปลงดีไวส์ไดรเวอร์ ของเวลาของระบบ
0.2.2	ปรับปรุงและเปลี่ยนแปลงดีไวส์ไดรเวอร์ใน ส่วนการอ่านเขียนข้อมูลกับแฟลชดิสก์ใน แพลตฟอร์มคอม86 เพื่อให้อ่านแฟลชดิสก์ ได้
0.2.3	ปรับปรุงและเปลี่ยนแปลงเรื่องการรับข้อมูล เข้า โดยใช้บริการไบออส
0.2.4	ปรับปรุงและเปลี่ยนแปลงดีไวส์ไดรเวอร์ ของอินเทอร์เน็ต
1.0.0	มินิซันคอม86 ที่สามารถ "ทำงาน" ได้ อย่างสมบูรณ์

ตาราง 5 เวอร์ชันและจุดประสงค์ในการพัฒนาในแต่ละเวอร์ชันของมินิซันคอม86

เพื่อให้่ายในการทำความเข้าใจลำดับในการพัฒนา มีลำดับในการพัฒนาในแต่ละเวอร์ชันดังรูป



รูปที่ 18 ลำดับในการพัฒนามินิซันคอม86

และสัญลักษณ์ลูกศร มีความหมายคือ
 → เป็นการไล่เนาะจากต้นฉบับ
 → เป็นการพัฒนาขึ้นมาในแต่ละรุ่น

เนื้อหาและรูปแบบที่ใช้ในการบอกเล่าถึงวิธีการและรายละเอียดในการพัฒนามินิซันคอม86 จะใช้วิธีการบอกเล่าในรูปแบบของการไล่ไปตามลำดับของแต่ละเวอร์ชันในการพัฒนา ตามลำดับตั้งแต่เวอร์ชัน 0.0.0 จนถึงเวอร์ชันสุดท้ายคือ 1.0.0 เพื่อให้เข้าใจจุดประสงค์และวิธีการในการพัฒนาได้อย่างถูกต้อง จะไม่ใช้รูปแบบของการสรุป เพราะจะทำให้เนื้อหาขาดส่วนที่สำคัญไป

โดยในแต่ละเวอร์ชัน จะมีทั้งวิธีการ ทฤษฎีที่เกี่ยวข้อง ขั้นตอนการทำงานของระบบปฏิบัติการมินิซันในแต่ละส่วน วิธีการในการใช้งานและติดต่อกับฮาร์ดแวร์ของแพลตฟอร์มคอม86 โดยได้สอดแทรกโค้ดในส่วนต่างๆ ที่ได้มีการเพิ่มเติม ตัดออกหรือปรับปรุงไว้ เพื่อให้ง่ายต่อการอ้างอิง ทั้งนี้จะพยายามอ้างอิงถึงโค้ดให้มากที่สุด เพื่อให้สามารถทำความเข้าใจถึงวิธีการทำงาน และเหตุผลในการปรับปรุงนั้นได้อย่างเข้าใจ เข้าใจถึงที่มาที่ไปได้อย่างถูกต้อง สามารถนำไปพัฒนาต่อยอดได้ทันทีในโอกาสอันเหมาะสมต่อไปในอนาคต

เริ่มต้นที่เวอร์ชัน 0.1.1 ไปจนถึงเวอร์ชัน 1.0.0 เป็นอันจบบทนี้

7.3 มินิซันคอม86 เวอร์ชันแรก เวอร์ชัน 0.1.1

จากการรวบรวมความคิดและการศึกษาในเบื้องต้น ทำให้สรุปรูปแบบการพัฒนาในลักษณะเป็นเวอร์ชัน และจากจุดนี้ทำให้การออกแบบลำดับถัดมาคือการกำหนดจุดประสงค์ให้การพัฒนาในแต่ละ

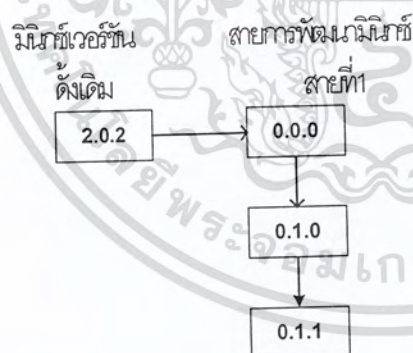
เวอร์ชันจากงานที่ได้แจกแจงไว้ สามารถทำได้อย่างรวดเร็วและมีขั้นตอนอย่างเป็นระบบ แสดงให้เห็นถึงคำกล่าวที่ว่าเริ่มต้นได้ดีเหมือนได้ชัยไปกว่าครึ่งเป็นคำกล่าวที่ไม่เกินความจริงเลยแม้แต่น้อย ทำให้สรุปได้ว่าการพัฒนามินิซันคอม86 ได้จำแนกสายของการพัฒนาออกเป็นสองสาย มุ่งเน้นไปในแต่ละด้าน และในแต่ละเวอร์ชันก็มีเป้าหมายในการพัฒนาอย่างชัดเจน ทำให้สามารถตรวจสอบการทำงานและตรวจสอบถึงความถูกต้องในการพัฒนาได้ตลอดเวลา

ในเวอร์ชัน 0.1.1 มีจุดประสงค์ในการพัฒนาคือ

เวอร์ชันในการพัฒนา	จุดประสงค์ในการพัฒนา
0.0.0	เวอร์ชันต้นฉบับ สำเนาจากมินิซันเวอร์ชัน 2.0.2
0.1.0	เวอร์ชันสำเนาข้อมูล เป็นเวอร์ชันที่สำเนาจากเวอร์ชัน 0.0.0
0.1.1	ตัดดีไวซ์ใดเวอร์ที่ไม่จำเป็นออก และลบไฟล์ที่ไม่ได้ใช้

ตาราง 6 แสดงจุดประสงค์ในการพัฒนามินิซันเวอร์ชัน 0.1.1

ซึ่งมีลำดับที่มา เริ่มต้นจากการสำเนามินิซันเวอร์ชัน 2.0.2 มาเป็นเวอร์ชัน 0.0.0 และ 0.1.0 และ 0.1.1 ในที่สุด จากนั้นจึงทำการแก้ไขมินิซันเวอร์ชัน 0.1.1 ให้บรรลุจุดประสงค์ที่ตั้งไว้ มีลำดับในการพัฒนาดังรูป



รูปที่ 19 แสดงลำดับในการพัฒนามินิซันเวอร์ชัน 0.1.1

7.3.1 เริ่มต้นที่ 0.1.1/src/kernel/Makefile

เพื่อให้การคอมไพล์เป็นไปได้อย่างถูกต้อง และคอมไพเลอร์สามารถหาไฟล์เฮดเดอร์ภาษาซีที่ใช้ได้ถูกต้อง ต้องทำการแก้ไขพารามิเตอร์ในการค้นหาไฟล์เฮดเดอร์ในเมคไฟล์ (Makefile) ซึ่งเป็นไฟล์ที่กำหนดวิธีการในการคอมไพล์ซอร์สโค้ด พารามิเตอร์ในการค้นหา พารามิเตอร์ที่ส่งผ่านให้คอมไพเลอร์ รวมไปถึงไฟล์ต่างๆ ที่จะนำมารวมกันเพื่อให้เป็นไฟล์ที่ทำงานได้ โดยเราทำการแก้ไขเป็น


```
# Directories
m = ../../
u = /usr
i = $m/include
s = $i/sys
h = $i/minix
b = $i/libm
l = $u/lib
```

ตัวแปรที่เพิ่มเข้าไปคือ m โดยเห็นว่าจะใช้วิธีการในการค้นหาไฟล์แบบสัมพันธ์กับตำแหน่งที่อยู่ในปัจจุบัน ซึ่งในที่นี้เราให้ m = ../../ เมื่อขณะทำการคอมไพล์ตำแหน่งที่อยู่ในปัจจุบันคือ 0.1.1/src/kemcl ดังนั้นเราก็จะได้ i = ../../include ซึ่งคอมไพเลอร์สามารถที่จะหาเฮดเดอร์ไฟล์ที่ 0.1.1/include ได้อย่างถูกต้อง นอกจากนั้น ยังทำให้แต่ละเวอร์ชัน แยกเฮดเดอร์ไฟล์อย่างอิสระในแต่ละเวอร์ชัน ไม่สับสนว่าจะใช้เฮดเดอร์ไฟล์ในเวอร์ชันใด แยกกันโดยเด็ดขาด ทำให้การคอมไพล์ถูกต้องและได้ผลลัพธ์ตามที่ต้องการ

7.3.2 พิจารณา 0.1.1/include/minix/config.h

ในการคอมไพล์เคอร์เนลของระบบปฏิบัติการมินิกซ์ มีไฟล์ที่ทำหน้าที่กำหนดคุณลักษณะของเคอร์เนลคือไฟล์ include/minix/config.h หรือคอนฟิกไฟล์ (Config File) ของเคอร์เนลของมินิกซ์ ซึ่งสามารถปรับแต่งค่าต่างๆ ได้ที่ไฟล์นี้ เพิ่มหรือตัดคิไวซ์ใดเวอร์ ปรับเปลี่ยนค่าพารามิเตอร์ที่ใช้ในระบบปฏิบัติการ รวมไปถึงค่าต่างๆ ที่จำเป็นต้องใช้ในระบบ

ก่อนอื่นเราจะเพิ่มเวอร์ชันในการพัฒนา ซึ่งขอให้คำว่า มินิกซ์บนคอม86 คือ MINIXC

```
/* Minix on COM86 release and version numbers. */
#define MINIXC_RELEASE "0.1"
#define MINIXC_VERSION "1"

/* Minix release and version numbers. */
#define OS_RELEASE "2.0"
#define OS_VERSION "2"
```

โดยเพิ่ม MINIXC_RELEASE กับ MINIXC_VERSION เข้าไป เพื่อบอกให้ทราบว่า เป็นมินิกซ์บนคอม86 เวอร์ชันเท่าใด และจะใช้คำว่าเลเบลแทนคำว่า #define ในการกล่าวถึงในลำดับถัดไป

ก่อนจะกล่าวต่อไป จะกล่าวถึงการพิจารณาโค้ดที่ไม่มีส่วนเกี่ยวข้องกับการพิจารณาการแก้ไขซอร์สโค้ดนั้น หมายถึงว่าโค้ดที่เมื่อตัดออกแล้วจะไม่มีผลกระทบเลย หรือถ้ามีก็น้อยมาก เราพบว่าถ้าจะทำการพิจารณาตัดโค้ดที่ไม่เกี่ยวข้องออกไป จะไม่พิจารณาตัดออกไปครั้งเดียวในเวอร์ชันใดเวอร์ชันหนึ่งทั้งหมด แต่ถ้าพบหรือเจอในเวอร์ชันใดก็จะพิจารณาตัดออกในทันทีเพื่อป้องกันความผิดพลาดและความเสี่ยงจากการตัดออกที่ไม่ถูกต้อง โดยพบว่ามีลักษณะโค้ดที่ไม่เกี่ยวข้องซึ่งแบ่งออกได้เป็น 2 แบบหลักคือ

```
# Directories
m = ../..
u = /usr
i = $m/include
s = $i/sys
h = $i/minix
b = $i/ibm
l = $u/lib
```

ตัวแปรที่เพิ่มเข้าไปคือ m โดยเห็นว่าจะใช้วิธีการในการค้นหาไฟล์แบบสัมพันธ์กับตำแหน่งที่อยู่ในปัจจุบัน ซึ่งในที่นี้เราให้ m = ../.. เมื่อขณะทำการคอมไพล์ตำแหน่งที่อยู่ในปัจจุบันคือ 0.1.1/src/kernel ดังนั้นเราก็จะได้ i = ../include ซึ่งคอมไพเลอร์สามารถที่จะหาเฮดเดอร์ไฟล์ที่ 0.1.1/include ได้อย่างถูกต้อง นอกจากนั้น ยังทำให้แต่ละเวอร์ชัน แยกเฮดเดอร์ไฟล์อย่างอิสระในแต่ละเวอร์ชัน ไม่สับสนว่าจะใช้เฮดเดอร์ไฟล์ในเวอร์ชันใด แยกกันโดยเด็ดขาด ทำให้การคอมไพล์ถูกต้องและได้ผลลัพธ์ตามที่ต้องการ

7.3.2 พิจารณา 0.1.1/include/minix/config.h

ในการคอมไพล์เคอร์เนลของระบบปฏิบัติการมินิกซ์ มีไฟล์ที่ทำหน้าที่กำหนดคุณลักษณะของเคอร์เนลคือไฟล์ include/minix/config.h หรือคอนฟิกไฟล์ (Config File) ของเคอร์เนลของมินิกซ์ ซึ่งสามารถปรับแต่งค่าต่างๆ ได้ที่ไฟล์นี้ เพิ่มหรือตัดโค้ดใดเวอร์ ปรับเปลี่ยนค่าพารามิเตอร์ที่ใช้ในระบบปฏิบัติการ รวมไปถึงค่าต่างๆ ที่จำเป็นต้องใช้ในระบบ

ก่อนอื่นเราจะเพิ่มเวอร์ชันในการพัฒนา ซึ่งย่อให้คำว่า มินิกซ์บนคอม86 คือ MINIXC

```
/* Minix on COM86 release and version numbers. */
#define MINIXC_RELEASE "0.1"
#define MINIXC_VERSION "1"

/* Minix release and version numbers. */
#define OS_RELEASE "2.0"
#define OS_VERSION "2"
```

โดยเพิ่ม MINIXC_RELEASE กับ MINIXC_VERSION เข้าไป เพื่อบอกให้ทราบว่าเป็นมินิกซ์บนคอม86 เวอร์ชันเท่าใด และจะใช้คำว่าเลขแทนคำว่า #define ในการกล่าวถึงในลำดับถัดไป

ก่อนจะกล่าวต่อไป จะกล่าวถึงการพิจารณาตัดโค้ดที่ไม่มีส่วนเกี่ยวข้องกับการพิจารณาการแก้ไขซอร์สโค้ดนั้น หมายถึงว่าโค้ดที่เมื่อตัดออกแล้วจะไม่มีผลกระทบเลย หรือถ้ามีก็น้อยมาก เราพบว่าถ้าจะทำการพิจารณาตัดโค้ดที่ไม่เกี่ยวข้องออกไป จะไม่พิจารณาตัดออกไปครั้งเดียวในเวอร์ชันใดเวอร์ชันหนึ่งทั้งหมด แต่ถ้าพบหรือเจอในเวอร์ชันใดก็จะพิจารณาตัดออกในทันทีเพื่อป้องกันความผิดพลาดและความเสี่ยงจากการตัดออกที่ไม่ถูกต้อง โดยพบว่ามีลักษณะโค้ดที่ไม่เกี่ยวข้องซึ่งแบ่งออกได้เป็น 2 แบบหลักคือ

เนื่องจากแพลตฟอร์มคอม86 เป็นแพลตฟอร์มขนาด 16 บิตเท่านั้น ซึ่งโค้ดในมินิกซ์ที่ระบุถึงการคอมไพล์ให้เป็น 16 หรือ 32 บิตนั้น จะถูกเลือกโดยตัวแปร WORD_SIZE ถ้าเป็น 4 จะหมายถึง 32 บิต ถ้าเป็น 2 จะหมายถึง 16 บิต ดังนั้นโค้ดที่มีคำว่า WORD_SIZE = 4 จะพิจารณาตัดออกทันทีเช่น

```
#if (MACHINE == IBM_PC && _WORD_SIZE == 4)
#define NR_BUFS      80 /* # blocks in the buffer cache */
#define NR_BUF_HASH 128 /* size of buf hash table;
#endif
```

ตัวแปร MACHINE และ CHIP ที่ไม่ใช่ IBM_PC กับ INTEL จะพิจารณาตัดออกทันที เช่น

```
#if (MACHINE == ATARI)
/* The next define says if you have an ATARI ST or TT */
#define ATARI_TYPE TT
#define ST          1 /* all ST's and Mega ST's */
#define STE         2 /* all STE and Mega STE's */
#define TT          3
```

ในคอนฟิกไฟล์ มีการกำหนดคิไวซ์ไครเวอร์ที่จะคอมไพล์รวมไปกับเคอร์เนล เราสามารถกำหนดให้มีการคอมไพล์คิไวซ์ไครเวอร์ใด ก็สามารถกำหนดให้เป็น 1 ได้ รวมไปถึงความหมายในการคอมไพล์ให้เคอร์เนลนั้นสนับสนุนคิไวซ์นั้นด้วย

ในมินิกซ์คิไวซ์ไครเวอร์ที่สนับสนุนโดยพิจารณาจากค่าที่กำหนดในไฟล์คอนฟิกดังนี้

```
/* Include or exclude device drivers. 1 to include */
#define ENABLE_AT_WINI 1 /* AT winchester driver */
#define ENABLE_BIOS_WINI 1 /* BIOS winchester driver */
#define ENABLE_ESDI_WINI 1 /* ESDI winchester driver */
#define ENABLE_XT_WINI 0 /* XT winchester driver */
#define ENABLE_AHA1540_SCSI 1 /* Adaptec 1540 SCSI driver */
#define ENABLE_MITSUMI_CDROM 0 /* Mitsumi CD-ROM driver */
#define ENABLE_DOSFAT 0 /* DOS FAT file virtual */
#define ENABLE_DOSFILE 1 /* DOS file virtual disk driver */
#define ENABLE_SB_AUDIO 0 /* Soundblaster audio driver */

/* DMA_SECTORS may be increased to speed up DMA based drivers. */
#define DMA_SECTORS 1 /* DMA buffer size (must be >= 1) */

/* Enable or disable networking code (TCP/IP task & drivers). */
#define ENABLE_NETWORKING 0 /* TCP/IP code */
#define ENABLE_WDETH 1 /* Western Digital WD80x3 */
#define ENABLE_NE2000 1 /* Novell NE1000/NE2000 */
#define ENABLE_3C503 1 /* 3Com Etherlink II (3C503) */
```

ตัวหาคือคิไวซ์ไครเวอร์ที่ไม่เกี่ยวข้องและไม่สนับสนุนการทำงานบนแพลตฟอร์มคอม86 ซึ่งต่อไปในโค้ดถ้ามีตัวหาคือจะหมายถึงว่า โค้ดส่วนนี้จะมีการพิจารณาตัดออกหรือมีการเปลี่ยนแปลง

ต้องการให้คอร์เนลสนับสนุนดิสก์ไดรเวอร์ใด ให้เปลี่ยนค่าเป็น 1 ถ้าไม่ต้องการก็เป็น 0 สำหรับดิสก์ไดรเวอร์ที่ไม่เกี่ยวข้องและไม่สนับสนุนการทำงานบนแพลตฟอร์มคอม86 ไม่มีความจำเป็นที่ต้องใช้ นั่นคือไม่ต้องคอมไพล์รวมไปในคอร์เนล ซึ่งจากการพิจารณาเราจะทำการลบเลเบลเหล่านี้ออก อย่างไรก็ตามเลเบลที่ทำการลบออกนี้ มีความเกี่ยวข้องเชื่อมโยงกันไปในแต่ละส่วนในโครงสร้างของซอร์สโค้ด พิจารณาความสัมพันธ์ของเลเบลที่ทำการลบออกในคอนฟิกไฟล์ มีความสัมพันธ์คือ

```
#define ENABLE_WINI      (ENABLE_AT_WINI || ENABLE_BIOS_WINI || \
                           ENABLE_ESDI_WINI || ENABLE_XT_WINI)
#define ENABLE_SCSI      (ENABLE_AHA1540_SCSI)
#define ENABLE_CDROM     (ENABLE_MITSUMI_CDROM)
#define ENABLE_AUDIO     (ENABLE_SB_AUDIO)
#define ENABLE_DOSDSK    (ENABLE_DOSFAT || ENABLE_DOSFILE)
```

เช่นถ้าเราลบเลเบล ENABLE_SB_AUDIO เราก็ต้องพิจารณาลบเลเบล ENABLE_AUDIO ออกด้วย เพราะมันจะคอมไพล์ไม่ผ่าน จากตัวอย่างจะเห็นได้ว่าต้องลบเลเบล ENABLE_AUDIO ออกด้วย เช่นเดียวกันเป็นต้น ดังนั้นจึงต้องมีการพิจารณาอย่างละเอียดรอบคอบก่อนที่จะทำการลบเลเบลใดๆ ออก วิธีการในการพิจารณาจะลบเลเบลใด จะทำการพิจารณาในแต่ละดิสก์ไดรเวอร์แยกกันไป โดยต้องคำนึงถึงเลเบลที่สัมพันธ์กันเสมอ

จนถึงขั้นตอนนี้ สามารถสรุปถึง ไฟล์และหน้าที่ที่เกี่ยวข้องได้ โดยจะขอยกภาพองค์ประกอบของมินิคซ์ซึ่งประกอบไปด้วยสี่ชั้น มาประกอบเพื่อทำการอธิบายอีกครั้ง โดยจะเน้นไปที่ชั้นทาสก์ (Task) หรือดิสก์ไดรเวอร์ของระบบปฏิบัติการมินิคซ์ ซึ่งจะเห็นได้ว่าสามารถจัดกลุ่มได้ดังนี้

1.Clock device driver เป็นดิสก์ไดรเวอร์ที่จัดการดูแลกับระบบเวลาของระบบมีไฟล์เดียวคือ clock.c

2.Block device driver เป็นดิสก์ไดรเวอร์ที่สนับสนุนดิสก์ที่มีลักษณะการถ่ายโอนข้อมูลเป็นบล็อก เป็นกลุ่มของข้อมูล มีไฟล์และหน้าที่ที่เกี่ยวข้องคือ

- driver.c drvlib.c เป็นส่วนที่ไม่ขึ้นกับฮาร์ดแวร์ ใช้ติดต่อกับเลเซอร์บนเช่นไฟล์ซิสเต็ม
- bios_wini.c at_wini.c xt_wini.c esdi_wini.c เป็นส่วนที่ขึ้นกับฮาร์ดแวร์ ไดรเวอร์สำหรับดิสก์แบบมาตรฐาน อนึ่ง bios_wini.c เป็นดิสก์ไดรเวอร์ที่ใช้ไบออสในการอ่านหรือเขียนกับดิสก์ ดังนั้นจึงจะไม่ทำการตัดออก เพื่อพิจารณาถึงความเป็นไปได้ในการนำมาพัฒนาเป็นไดรเวอร์สำหรับอ่านหรือเขียนกับแฟลชดิสก์บนคอม86
- floppy.c เป็นส่วนที่ขึ้นกับฮาร์ดแวร์ ไดรเวอร์สำหรับฟลอปปีดิสก์
- aha1540.c เป็นส่วนที่ขึ้นกับฮาร์ดแวร์ ไดรเวอร์สำหรับดิสก์แบบสก็ซี (SCSI)
- memory.c เป็นส่วนที่ขึ้นกับฮาร์ดแวร์ ไดรเวอร์สำหรับดิสก์แบบแรมดิสก์ (Ram disk)

3.Character device driver เป็นดิสก์ไดรเวอร์ที่สนับสนุนดิสก์ที่เป็นสตริมหรือสายของอักขระ มีไฟล์และหน้าที่ที่เกี่ยวข้องคือ

- tty.c เป็นส่วนที่ไม่ขึ้นกับฮาร์ดแวร์ ใช้ติดต่อกับเลเยอร์บนเช่นไฟล์ซิสเต็ม ดูแลเรื่องอินพุตเอาต์พุตทั้งหมด
- console.c keyboard.c เป็นส่วนที่ขึ้นกับฮาร์ดแวร์ ไดรเวอร์สำหรับจอภาพและคีย์บอร์ด
- rs232.c เป็นส่วนที่ขึ้นกับฮาร์ดแวร์ ไดรเวอร์สำหรับการสื่อสารผ่านพอร์ตอนุกรมมาตรฐาน
- pty.c เป็นส่วนที่ขึ้นกับฮาร์ดแวร์ ไดรเวอร์สำหรับการสื่อสารผ่านเครือข่าย โดยเป็นเทอร์มินอลเมื่อทำการติดต่อเข้ามาทางระบบเครือข่ายคอมพิวเตอร์

4. Printer device driver เป็นไดรเวอร์ที่สนับสนุนไดรเวอร์สำหรับการสื่อสารผ่านพอร์ตนานมาตรฐาน มีไฟล์ที่เกี่ยวข้องเพียงไฟล์เดียวคือ printer.c

5. CD-Rom device driver เป็นไดรเวอร์ที่สนับสนุนไดรเวอร์ที่เป็นซีดีรอม ปัจจุบันสนับสนุนซีดีจากผู้ผลิตมิตซูบิเท่านั้น มีไฟล์เดียวคือ mcd.c

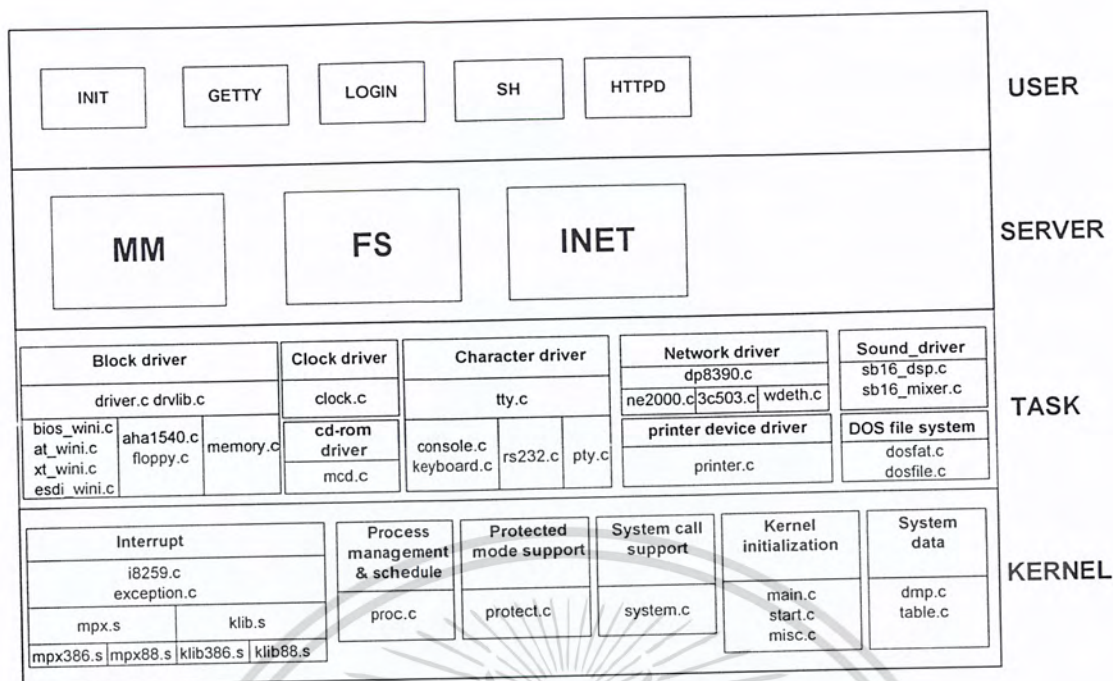
6. Sound device driver เป็นไดรเวอร์ที่สนับสนุนไดรเวอร์ที่เกี่ยวกับเสียง ปัจจุบันสนับสนุนไดรเวอร์ที่คอมแพคทิเบลกับการ์ดเสียงของชาวล์ลาสเตอร์เท่านั้น มีไฟล์คือ sb16_dsp.c และ sb16_mixer.c

7. Network device driver เป็นไดรเวอร์ที่สนับสนุนการ์ดเครือข่ายแบบอีเทอร์เน็ต จากสามผู้ผลิต

- dp8390.c เป็นไฟล์หลัก เนื่องจากการ์ดทั้งสามนั้นมีการทำงานในส่วนหลักเหมือนกัน
- 3c503.c เป็นไดรเวอร์สำหรับการ์ดอีเทอร์เน็ตของทริคอม (3COM)
- wdeth.c เป็นไดรเวอร์สำหรับการ์ดอีเทอร์เน็ตของเวสต์เทิร์นดิจิทัล (Western Digital)
- ne2000.c เป็นไดรเวอร์สำหรับการ์ดอีเทอร์เน็ตที่คอมแพคทิเบลกับการ์ดอีเทอร์เน็ตของโนเวล 2000 จากการทดสอบฮาร์ดแวร์ พบว่าฮาร์ดแวร์ที่สนับสนุนการทำงานทางด้านเครือข่ายของคอม86 สนับสนุนไดรเวอร์ตัวนี้ ดังนั้นเราจะเก็บไว้เพื่อพิจารณาถึงความเหมาะสมในการนำไปใช้ต่อไป

8. Dos file system device driver เป็นไดรเวอร์ที่สนับสนุนการอ่านเขียนระบบไฟล์บนดอส ไม่ได้ให้สามารถอ่านเขียนไฟล์ได้ โดยในมินิคซ์เวอร์ชัน 2.0.2 นั้นได้เพิ่มความสามารถให้สามารถทำงานได้บนระบบปฏิบัติการดอสได้ หมายถึงว่าทำงานเหมือนเป็นแอปพลิเคชันหนึ่ง ซึ่งไดรเวอร์ตัวนี้ออกแบบมาสำหรับให้สามารถอ่านและเขียนไฟล์ได้ในขณะที่ทำงานอยู่บนดอส ซึ่งไม่ได้สนับสนุนการอ่านเขียนไฟล์โดยตรง มีไฟล์ที่เกี่ยวข้องคือ dosfile.c กับ dosfat.c

อ้างอิงได้ในตามรูปในเลเยอร์ของทาสก์



รูปที่ 20 องค์ประกอบของมินิกซ์เมื่อมองจากมุมมองในลักษณะของลำดับชั้นของโปรเซส

พิจารณาถึงไฟล์ที่ต้องทำการตัดออกในชั้นล่างสุดหรือชั้นเคอร์เนล พิจารณาที่ไฟล์ mpx.s ซึ่งมีโค้ดดังนี้

```

#! Chooses between the 8086 and 386 versions of the Minix.

#include <minix/config.h>
#if _WORD_SIZE == 2
#include "mpx88.s"
#else
#include "mpx386.s"
#endif

```

ตัวหนาคือโค้ดที่ตัดออกจากที่กล่าวไปแล้วว่า จะพิจารณาเฉพาะโค้ดที่เป็น WORD_SIZE = 2 เท่านั้น ดังนั้นจึงสามารถตัดไฟล์ mpx386.s ได้โดยตรง และใน klib.s ก็มีโค้ดลักษณะเดียวกัน ดังนั้นไฟล์ klib386.s ก็สามารถตัดออกได้เช่นเดียวกัน

สรุปถึงไฟล์ที่ต้องทำการพิจารณาตัดออก ได้ดังนี้

ไฟล์ที่ต้องทำการพิจารณาตัดออก	หน้าที่
aha1540.c	ไดรเวอร์สำหรับดิสก์แบบสกซ์ซี
mcd.c	ไดรเวอร์ที่สนับสนุนดีไวซ์ที่เป็นซีดีรอม
sb16_dsp.c	ไดรเวอร์ที่สนับสนุนดีไวซ์ที่คอมแพคทีเบิล

sb16_mixer.c sb16.h sound.h	กับการดเสียงของชาวดีบลัสเตอร์
at_wini.c xt_wini.c esdi_wini.c	ไดรเวอร์สำหรับดิสก์แบบมาตรฐาน
wdeth.c wdeth.h 3c503.c 3c503.h	ไดรเวอร์สำหรับการ์ดอีเทอร์เน็ตของทีรคอม และเวสต์เทิร์นดิจิตอล
dosfile.c dosfat.c	เป็นดีไวส์ไดรเวอร์ที่สนับสนุนการอ่านเขียน ระบบไฟล์บนดอส เมื่อจำลองให้ทำงานบน ดอส
printer.c	ดีไวส์ไดรเวอร์สำหรับการสื่อสารผ่านพอร์ต ขนาน
floppy.c	ไดรเวอร์สำหรับฟลอปปีดิสก์
mpx386.s klib386.s	โค้ดภาษาแอสเซมบลีใช้ในระดับที่มีการติด ต่อกับฮาร์ดแวร์โดยตรง

ตาราง 7 ไฟล์ที่ทำการพิจารณาตัดออก

7.3.3 ไดรเวอร์สำหรับดิสก์แบบสกซ์ซี่

จุดใดที่สามารถเอาออกได้ จะพิจารณาเอาออกทันทีในหัวข้อนี้ พิจารณาลABEL
ENABLE_AHA1540 SCSI และเลเบลที่สัมพันธ์กันคือ ENABLE_SC SI โดยพิจารณาลABELทั้งสองนี้
ออก ดังนั้นก่อนอื่นลบไฟล์ aha1540.c ออกไปก่อน จากนั้น ลบเลเบลนี้ใน 0.1.1/include/minix/config.h
ออกทั้งสองจุด

```
#define ENABLE_AHA1540 SCSI 1 /* Adaptec 1540 SCSI driver */
...
#define ENABLE_SC SI (ENABLE_AHA1540 SCSI)
```

ต่อมาไฟล์ 0.1.1/src/kernel/driver.c เปลี่ยนจาก

```
#if ENABLE_AHA1540 SCSI && DMA_BUF_SIZE < 2048
/* A bit extra scratch for the Adaptec driver. */
#define BUF_EXTRA (2048 - DMA_BUF_SIZE)
#else
#define BUF_EXTRA 0
#endif
```

เป็น

```
#define BUF_EXTRA 0
```

พิจารณาไฟล์ 0.1.1/src/kernel/table.c เป็นไฟล์เก็บรายละเอียดของโครงสร้างฐานข้อมูลของมินิกซ์

```
...
#if ENABLE_AHA1540_SCSI
#define scsi_task          aha1540_scsi_task
#endif
...
    { mixer_task,          MIXER_STACK,      "MIXER"
      },
#endif
#if ENABLE_SCSI
    { scsi_task,           SCSI_STACK, "SCSI"
      },
#endif
#if ENABLE_WINI
    { winchester_task,     WINCH_STACK,      "WINCH"
      },
...
#if ENABLE_AHA1540_SCSI
    { "aha1540",           aha1540_scsi_task },
#endif
...
#if ENABLE_SCSI
    { "sd",                SCSI_STACK,
      },
#endif
...
```

ให้ลบออก และจุดเดียวกัน เป็นการกำหนดคสแด้ของไครเวอร์ เปลี่ยนจาก

```
#define PRINTER_STACK SMALL_STACK
#define SCSI_STACK (2 * SMALL_STACK * ENABLE_SCSI)
#define CDROM_STACK (4 * SMALL_STACK * ENABLE_CDROM)
#define DOSDSK_STACK (3 * SMALL_STACK * ENABLE_DOSDSK)
#define AUDIO_STACK (4 * SMALL_STACK * ENABLE_AUDIO)
#define MIXER_STACK (4 * SMALL_STACK * ENABLE_AUDIO)
#define FLOP_STACK (3 * SMALL_STACK)
```

เป็น

```
#define PRINTER_STACK 0
#define SCSI_STACK 0
#define CDROM_STACK 0
#define DOSDSK_STACK 0
#define AUDIO_STACK 0
#define MIXER_STACK 0
#define FLOP_STACK 0
```

ต่อมาแก้ไข 0.1.1/src/kernel/Makefile เอาตัวหนาออก เอาไฟล์ที่ต้องการจะลบออกไปพร้อมกัน ด้วย เพราะอย่างไรก็ลบไฟล์ออกอย่างแน่นอน

มีไฟล์ที่สำคัญอีกหนึ่งไฟล์ทำหน้าที่ในการกำหนดฟังก์ชันทั้งหมด คือ 0.1.1/src/kernel/proto.h เนื่องมาจากการที่พิจารณาจบไฟล์ออก ดังนั้นต้องทำการลบฟังก์ชันที่ได้ประกาศไว้ออกไปด้วย โดยตัวหนาคือที่เอาออกทั้งหมด

```
...
_PROTOTYPE( void at_winchester_task, (void) );
_PROTOTYPE( void bios_winchester_task, (void) );
_PROTOTYPE( void esdi_winchester_task, (void) );
_PROTOTYPE( void ps_winchester_task, (void) );
_PROTOTYPE( void xt_winchester_task, (void) );
...
/* aha1540.c */
_PROTOTYPE( void aha1540_scsi_task, (void) );

/* dosfat.c, dosfile.c */
_PROTOTYPE( void dosfat_task, (void) );
_PROTOTYPE( void dosfile_task, (void) );
_PROTOTYPE( void dosfile_stop, (void) );
...
/* floppy.c, stfloppy.c */
_PROTOTYPE( void floppy_task, (void) );
_PROTOTYPE( void floppy_stop, (void) );
...
/* mcd.c */
_PROTOTYPE( void mcd_task, (void) );
...
/* printer.c, stprint.c */
_PROTOTYPE( void printer_task, (void) );
...
/* sb16_dsp.c */
_PROTOTYPE( void dsp_task, (void) );
...
/* sb16_mixer.c */
_PROTOTYPE( void mixer_task, (void) );
...
/* 3c503.c */
_PROTOTYPE( int el2_probe, (struct dpeth *dep) );
...
/* printer.c */
_PROTOTYPE( void pr_restart, (void) );
...
/* wdeh.c */
_PROTOTYPE( int wdeth_probe, (struct dpeth *dep) );
```

ยังมีจุดที่สำคัญอีกจุดหนึ่งเป็นจุดที่สำคัญมาก คือในไฟล์ 0.1.1/include/minix/const.h คือ

```
/* Number of tasks. */
#define NR_TASKS (9 + ENABLE_WINI + ENABLE_SCSI + ENABLE_CDROM
+ ENABLE_DOSDSK + ENABLE_NETWORKING + 2 * ENABLE_AUDIO)
```

NR_TASKS เป็นตัวแปรที่บอกว่าในมินิกซ์มีดีไวซ์ไครเวอร์ทั้งหมดเป็นจำนวนเท่าใด จำเป็นอย่างยิ่งที่จะต้องแก้ไขให้ถูกต้องตามจำนวนดีไวซ์ไครเวอร์ที่มีการกำหนดอยู่จริง สามารถนับได้จากโครงสร้าง task_tab ในไฟล์ src/kernel/table.c เมื่อทำการพิจารณาจบเลเบลแล้วจะได้เป็น

```
/* Number of tasks. */
#define NR_TASKS (7 + ENABLE_WINI + ENABLE_NETWORKING)
```

สังเกตตรงตัวเลขจาก 9 ลดลงเป็น 7 เนื่องจากเราได้ลบดีไวซ์ไครเวอร์ออกอีกสองดีไวซ์คือพรินเตอร์และฟลอปปีดิสก์ เนื่องจากดีไวซ์สองดีไวซ์นี้ไม่ได้มีเลเบลในการที่จะกำหนดว่าจะให้คอมไพเลอร์รวมไปกับเคอร์เนลไหม หรือกล่าวได้ว่าไม่มีเลเบลให้เลือก แต่จะเป็นดีไวซ์ที่มีนิกซ์กำหนดให้ต้องมีเป็นมาตรฐาน แต่เนื่องจากว่า ดีไวซ์ไครเวอร์ทั้งพรินเตอร์และฟลอปปีดิสก์ไม่ได้สนับสนุนโดยแพลตฟอร์มคอม86 ดังนั้นการพิจารณาตัดออกจึงสามารถทำได้ ซึ่งต้องลบค่าใน NR_TASKS ออกให้สอดคล้องด้วย

อีกไฟล์หนึ่งที่มีความสำคัญเช่นเดียวกันคือไฟล์ 0.1.1/include/minix/com.h เป็นไฟล์ที่กำหนดหมายเลขประจำตัวสำหรับดีไวซ์ไครเวอร์ ดีไวซ์ไครเวอร์ในมินิกซ์ทั้งหมดนอกจากจะมีการกำหนดจำนวนดีไวซ์ที่มีได้ทั้งหมดโดยค่าคงที่ NR_TASKS แล้วยังมีการกำหนดหมายเลขประจำดีไวซ์ไครเวอร์เป็นค่าคงที่อีก เหตุผลเนื่องจากความต้องการให้ระบบปฏิบัติการมีโครงสร้างไม่ซับซ้อนจนเกินไป

```
#define TTY (DL_ETH - 1)
...
#define DL_ETH (DOSDSK - ENABLE_NETWORKING)
...
#define DOSDSK (CDROM - ENABLE_DOSDSK)
...
#define CDROM (AUDIO - ENABLE_CDROM)
...
#define AUDIO (MIXER - ENABLE_AUDIO)
...
#define MIXER (SCSI - ENABLE_AUDIO)
...
#define SCSI (WINCHESTER - ENABLE_SCSI)
...
#define WINCHESTER (SYN_ALARM_TASK - ENABLE_WINI)
...
#define SYN_ALARM_TASK -8
...
#define IDLE -7
...
#define PRINTER -6
...
#define FLOPPY -5
...
#define MEM -4
...
#define CLOCK -3
...
#define SYSTASK -2
...
#define HARDWARE -1
```

แก้ไขได้เป็น

```

#define TTY                (DL_ETH - 1)
...
#define DL_ETH              (WINCHESTER - ENABLE_NETWORKING)
...
#define WINCHESTER          (SYN_ALARM_TASK - ENABLE_WINI)
...
#define SYN_ALARM_TASK      -6
...
#define IDLE                 -5
...
#define MEM                  -4
...
#define CLOCK                -3
...
#define SYSTASK              -2
...
#define HARDWARE             -1

```

7.3.4 ไดรเวอร์ที่สนับสนุนดีไวซ์ที่เป็นซีดีรอม

ก่อนอื่นลบไฟล์ mcd.c จากนั้นพิจารณาถึงเลเบล ENABLE_CDROM และ ENABLE_MITSUMI_CDROM ซึ่งจะเอาออกในไฟล์ 0.1.1/src/kernel/table.c แสดงเป็นตัวหนาในโค้ดดังนี้

```

...
* WINCHESTER, but for now the choice is fixed.
*/
#define cdrom_task          mcd_task
#define audio_task          dsp_task
#endif
...
#if ENABLE_CDROM
    { cdrom_task,          CDROM_STACK,          "CDROM"      },
#endif
#if ENABLE_AUDIO
...

```

7.3.5 ไดรเวอร์ที่สนับสนุนดีไวซ์ที่คอมแพคทีเบิ้ลกับการ์ดเสียงของชาวคัสเตอร์

มีไฟล์ที่สามารถลบออกได้ถึงสี่ไฟล์คือ sb16_dsp.c sb16_mixer.c sb16.h และ 0.1.1/include/minix/sound.h ต่อมาพิจารณาถึงเลเบล ENABLE_AUDIO กับ ENABLE_SB_AUDIO พิจารณาในไฟล์ 0.1.1/src/kernel/table.c ที่ตัวหนาคือลบออก

```

...
#define audio_task          dsp_task
...
#if ENABLE_AUDIO
    { audio_task,          AUDIO_STACK,          "AUDIO"      },
    { mixer_task,          MIXER_STACK,          "MIXER"      },

```

```
#endif
...
```

7.3.6 ไดรเวอร์สำหรับดิสก์แบบมาตรฐาน

ตัวเลขเบลที่เกี่ยวข้องในไฟล์ 0.1.1/src/kernel/table.c ได้ดังนี้

```
...
/* Default choices for tasks with more than one driver. */
#if ENABLE_AT_WINI
#define winchester_task at_winchester_task
#elif ENABLE_BIOS_WINI
#define winchester_task bios_winchester_task
#elif ENABLE_ESDI_WINI
#define winchester_task esdi_winchester_task
#elif ENABLE_XT_WINI
#define winchester_task xt_winchester_task
#endif
...
#if ENABLE_AT_WINI
    { "at",          at_winchester_task      },
#endif
#if ENABLE_BIOS_WINI
    { "bios",        bios_winchester_task    },
#endif
#if ENABLE_ESDI_WINI
    { "esdi",        esdi_winchester_task    },
#endif
#if ENABLE_XT_WINI
    { "xt",          xt_winchester_task      },
#endif
```

7.3.7 ไดรเวอร์สำหรับการดื่เทอร์เน็ตของทรินคอม และเวสต์เทิร์นดิจิตอล

พิจารณาโดยใช้หลักเช่นเดียวกัน ก่อนอื่นลบไฟล์ได้ทั้งหมดคือไฟล์คือ 3c503.h 3c503.c wdeth.h wdeth.c แต่คราวนี้ไฟล์ที่ต้องแก้ไขไม่ใช่ 0.1.1/src/kernel/table.c แล้วแต่จะเป็นไฟล์ 0.1.1/src/kernel/dp8390.c ให้พิจารณาโค้ดที่ทำการตัดออกก่อนคือ

```
...
/* Card inits configured out? */
#if !ENABLE_WDETH
#define wdeth_probe(dep)          (0)
#endif
#if !ENABLE_NE2000
#define ne_probe(dep)             (0)
#endif
#if !ENABLE_3C503
#define el2_probe(dep)            (0)
```



```
#endif
...
```

เนื่องจาก ENABLE_WDETH และ ENABLE_3C503 ได้ทำการพิจารณาออกไปจากคอนฟิกไฟล์ ดังนั้นจึงต้องทำการลบออกในจุดนี้ด้วย

7.3.8 ดีไวซ์ไดรเวอร์ที่สนับสนุนการอ่านเขียนระบบไฟล์บนดอส เมื่อจำต้องให้ทำงานบนดอสทำการลบไฟล์ dosfat.c dosfile.c ออกได้ พิจารณาใน 0.1.1/src/kernel/table.c

```
...
#if ENABLE_DOSFILE
#define dosdsk_task      dosfile_task
#elif ENABLE_DOSFAT
#define dosdsk_task      dosfat_task
#endif
...
#if ENABLE_DOSDSK
{ dosdsk_task, DOSDSK_STACK, "DOSDSK" },
#endif
...
#if ENABLE_DOSFILE
{ "file", dosfile_task },
#endif
#if ENABLE_DOSFAT
{ "fat", dosfat_task },
#endif
...
#if ENABLE_DOSDSK
{ "dosd", DOSDSK },
#endif
...
```

และในไฟล์ 0.1.1/src/kernel/keyboard.c ลบโค้ดที่เรียกใช้ dosfile_stop() ออก

```
...
floppy_stop();
clock_stop();
#if ENABLE_DOSFILE
dosfile_stop();
#endif
...
```

7.3.9 ดีไวซ์ไดรเวอร์สำหรับการสื่อสารผ่านพอร์ตนาน

```
...
PUBLIC struct dmap dmap[] = {
/* ?   Open      Read/Write  Close   Task #   Device  File
-   -   -   -   -   -   -   -   -   -   -   -
---- */
```

```

    DT(1, no_dev,    no_dev,    no_dev,    0)          /* 0 = not
used */
    DT(1, dev_opcl,  call_task,  dev_opcl, MEM)          /* 1 =
/dev/mem */
    DT(1, dev_opcl,  call_task,  dev_opcl, FLOPPY)       /* 2 =
/dev/fd0 */
    DT(ENABLE_WINI,
        dev_opcl,  call_task,  dev_opcl, WINCHESTER) /* 3 =
/dev/hd0 */
    DT(1, tty_open,  call_task,  dev_opcl, TTY)          /* 4 =
/dev/tty00 */
    DT(1, cttty_open, call_ctty,  cttty_close, TTY) /* 5 = /dev/tty
*/
    DT(1, dev_opcl,  call_task,  dev_opcl,  PRINTER) /* 6 =
/dev/lp */
...

```

แก้ไขเป็น

```

...
PUBLIC struct dmap dmap[] = {
/* ?   Open      Read/Write  Close      Task #   Device  File
-----
---- */
    DT(1, no_dev,    no_dev,    no_dev,    0)          /* 0 = not
used */
    DT(1, dev_opcl,  call_task,  dev_opcl, MEM)          /* 1 =
/dev/mem */
    DT(0, 0,         0,         0,         0)          /* 2 =
/dev/fd0 */
    DT(ENABLE_WINI,
        dev_opcl,  call_task,  dev_opcl, WINCHESTER) /* 3 =
/dev/hd0 */
    DT(1, tty_open,  call_task,  dev_opcl, TTY)          /* 4 =
/dev/tty00 */
    DT(1, cttty_open, call_ctty,  cttty_close, TTY) /* 5 = /dev/tty
*/
    DT(0, 0,         0,         0,         0)          /* 6 = /dev/lp
*/
...

```

เนื่องจากการที่ดิสก์ไดรเวอร์ทั้งพรีนเตอร์และฟลอปปีดิสก์ไม่ได้สนับสนุนโดยแพลตฟอร์มคอม86 และพิจารณาตัดออก ดังนั้นจึงควรให้ไม่มีการเรียกใช้ดิสก์ไดรเวอร์ทั้งสองดิสก์นี้

7.3.10 ไดรเวอร์สำหรับฟลอปปีดิสก์

ลบไฟล์ 0.1.1/src/kernel/floppy.c 0.1.1/include/ibm/diskparm.h โดยมากเราได้ตัดเลเบลที่ไม่จำเป็นออกแล้ว ในขั้นตอนนี้อาจเหลือแค่การแก้ไขไฟล์เพียงไฟล์เดียวคือ 0.1.1/src/kernel/keyboard.c โดยลบ

```

#if ENABLE_NETWORKING
    dp8390_stop();

```



```
#endif
floppy_stop();
clock_stop();
```

เพียงขั้นตอนเดียว

7.3.11 แก้ไขขั้นตอนสุดท้าย

เปลี่ยนแปลงโลโกของมินิกซ์บนคอม86 โดยแก้ไขไฟล์ 0.1.1/src/kernel/tty.c เป็น

```
/* Display the Minix startup banner. */
printf("MinixC (Minix on Com86) Version %s.%s\n",
        MINIXC_RELEASE, MINIXC_VERSION);
printf("MinixC is based on Minix Version %s.%s Copyright 1998 Prentice-Hall, Inc.\n",
        OS_RELEASE, OS_VERSION);
printf("Copyright (C) Embedded System Lab and CATOS Team, 2001. All right
reserved.\n\n");
```

และเพิ่มการตรวจสอบคอมไพเลอร์ที่เป็น ACK คอมไพเลอร์แบบ 16 บิตเท่านั้นใน
0.1.1/include/minix/config.h

```
#if (_WORD_SIZE != 2)
error "Please use 16-bit CC Compiler"
#endif
```

โดยถ้าตรวจสอบว่าไม่ใช่จะไม่ทำการคอมไพล์ซอร์สโค้ดและแสดงข้อความผิดพลาดขึ้นมา

7.4 มินิกซ์บนคอม86 เวอร์ชัน 0.1.2

ในเวอร์ชัน 0.1.2 มีจุดประสงค์ในการพัฒนาคือ

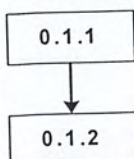
เวอร์ชันในการพัฒนา	จุดประสงค์ในการพัฒนา
0.1.2	ปรับปรุงและเปลี่ยนแปลงเรื่องการแสดงผลให้ใช้บริการของไบออส

ตาราง 8 แสดงจุดประสงค์ในการพัฒนามินิกซ์เวอร์ชัน 0.1.2

ลำดับในการพัฒนาเป็นไปตามลำดับในสายของการพัฒนาจากเวอร์ชัน 0.1.1 ลงมาเป็น 0.1.2

สายการพัฒนามินิกซ์

สายที่ 1



รูปที่ 21 แสดงลำดับในการพัฒนามินิกซ์เวอร์ชัน 0.1.2

7.4.1 ประเมินแนวทางในการแก้ไขในเบื้องต้น

บริการของไบออสในคอม86 ในเรื่องของการแสดงผลสนับสนุนบริการผ่านทางอินเทอร์รัปต์ หมายเลข 10 ตามสืบทก ทั้งสามบริการคือ

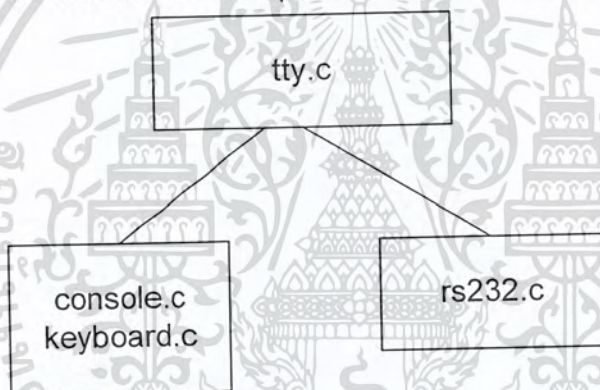
ฟังก์ชันย่อย	รายละเอียดของฟังก์ชัน
AH = 09h	Write attributes and characters at cursor position
	AL = Character BH = Display page BL = Attributes of character CX = Number of times to write character all characters are displayed, including CR, LF, and BS
AH = 0Ah	Write characters only at cursor position
	AL = Character BH = Display page BL = Color of character CX = Number of times to write character all characters are displayed, including CR, LF, and BS
AH = 0Eh	Write characters and advance cursor position
	AL = Character BH = Display page BL = foreground color characters 07h (BEL), 08h (BS), 0Ah (LF), and 0Dh (CR) are interpreted and do the expected things

ตาราง 9 แสดงบริการไบออสในเรื่องของการแสดงผลที่สนับสนุนโดยไบออสของคอม86

เนื่องจากแพลตฟอร์มคอม86 เป็นแพลตฟอร์มที่จัดอยู่ในระบบฝังตัว มีรูปแบบในการพัฒนาโดยการส่งผ่านข้อมูลผ่านพอร์ตอนุกรมมาตรฐาน หมายถึงว่าระบบปฏิบัติการที่อยู่บนคอม86 จะสามารถแสดงผลและรับข้อมูลได้ทางเดียวคือผ่าน โปรแกรมเทอร์มินอลที่ติดต่อผ่านพอร์ตอนุกรมมาตรฐานเท่านั้น ดังนั้นในการพัฒนาระบบปฏิบัติการเพื่อให้สามารถแสดงผลได้อย่างถูกต้อง ต้องทำให้การแสดงผลทั้งหมด แสดงผ่านพอร์ตอนุกรมแทนที่จะไปแสดงผลที่จอภาพ ข้อความจึงจะถูกส่งมาแสดงที่โปรแกรมเทอร์มินอลที่สื่อสารอยู่กับพอร์ตอนุกรมได้ รวมไปถึงการรับข้อมูลด้วยเช่นเดียวกัน ระบบปฏิบัติการต้องออกแบบให้รองรับการรับข้อมูลผ่านคีย์บอร์ดผ่าน โปรแกรมเทอร์มินอลผ่าน ไปยังพอร์ตอนุกรมและรับข้อมูลผ่านพอร์ตอนุกรมแล้วจึงนำไปประมวลผลตามกระบวนการทำงานปกติต่อไป

บริการไบออสบนแพลตฟอร์มคอม86 ได้ออกแบบให้บริการไบออสที่ใช้ในการแสดงผลทางจอภาพหรือบริการผ่านทางอินเทอร์พรีต์หมายเลข 10 ฐานสิบหกของไบออส และบริการไบออสที่ใช้ในการรับข้อมูลจากคีย์บอร์ดหรือบริการผ่านทางอินเทอร์พรีต์หมายเลข 16 ฐานสิบหกของไบออส ให้สามารถแสดงผลและรับข้อมูลผ่านพอร์ตอนุกรมได้ หมายความว่าถ้าระบบปฏิบัติการใด หรือแอปพลิเคชันใดที่นำไปทำงานอยู่บนคอม86 และได้มีการออกแบบให้มีการแสดงผลผ่านบริการผ่านทางอินเทอร์พรีต์หมายเลข 10 ฐานสิบหกของไบออส และรับข้อมูลจากคีย์บอร์ดหรือบริการผ่านทางอินเทอร์พรีต์หมายเลข 16 ฐานสิบหกของไบออส ระบบปฏิบัติการนั้นหรือแอปพลิเคชันนั้นก็สามารถที่จะแสดงผลผ่านโปรแกรมเทอร์มินอลและรับข้อมูลจากคีย์บอร์ดผ่านโปรแกรมเทอร์มินอลได้อย่างถูกต้อง

เพื่อให้การพัฒนาในขั้นตอนนี้เป็นไปอย่างรวดเร็ว เราได้พิจารณาที่จะเลือกพัฒนาและปรับปรุงฟังก์ชันที่ทำหน้าที่ในการติดต่อกับการแสดงผลผ่านจอภาพ จากที่ได้กล่าวไปแล้วว่าระบบปฏิบัติการมีนักซ์ถูกออกแบบมาให้ใช้บริการไบออสน้อยที่สุด ทำให้ฟังก์ชันที่ทำหน้าที่ติดต่อกับจอภาพไม่ได้เรียกใช้บริการไบออสในการแสดงผล แต่ทำการติดต่อกับหน่วยความจำจอภาพโดยตรง เพื่อให้การส่งข้อมูลออกมาแสดงผลทำได้รวดเร็วกว่า ให้พิจารณาจากจุดนี้มีทางเลือกในการเลือกที่จะพัฒนาได้สองทางคือ



รูปที่ 22 ทางเลือกในการพัฒนาวิธีการแสดงผล

จากรูปเห็นได้ว่า มีทางเลือกสองทางคือ

1. นำดีไวซ์ไครเวอร์ที่ติดต่อกับจอภาพและคีย์บอร์ดหรือ console.c กับ keyboard.c มาปรับปรุงให้ใช้บริการผ่านไบออสแทนที่จะติดต่อกับฮาร์ดแวร์โดยตรง หรือ
2. นำดีไวซ์ไครเวอร์ที่ติดต่อสื่อสารผ่านพอร์ตอนุกรมหรือ rs232.c มาพิจารณาแก้ไขปรับปรุงส่วนที่ติดต่อกับฮาร์ดแวร์เพื่อนำมาใช้ในการแสดงผลผ่านพอร์ตอนุกรมสื่อสาร

จากการประเมินผลโดยใช้เกณฑ์ในการประเมินเบื้องต้น เราเลือกที่นำดีไวซ์ไครเวอร์ที่ติดต่อกับจอภาพและคีย์บอร์ดมาทำการแก้ไขปรับปรุงให้ใช้บริการผ่านไบออสแทนที่จะติดต่อกับฮาร์ดแวร์โดยตรง มีเหตุผลในการเลือกดังนี้

1. มาจากความต้องการให้การพัฒนาช่วงนี้เป็นไปอย่างรวดเร็ว เนื่องจากยังมีส่วนหลักในภายหลังที่จะต้องทำการแก้ไข เช่น ส่วนที่ติดต่อกับฮาร์ดแวร์โดยตรงหรือส่วนที่เกี่ยวข้องกับอินเทอร์พรีต์ของระบบ ส่วนที่ติดต่อกับอ็อบเจกต์เน็ต เป็นต้น

2.ความรู้ในเรื่องการแก้ไขให้มาใช้บริการผ่านไบออสสามารถทำได้รวดเร็วและง่ายกว่าการนำดีไวซ์ที่ติดต่อกับพอร์ตอนุกรมมาทำการแก้ไขปรับปรุงในขั้นตอนเบื้องต้น จากการระดมความคิดมีความเห็นว่าจะนำมาแก้ไขก็ต่อเมื่อได้แก้ปัญหาในจุดสำคัญจากเหตุผลข้อแรกไปก่อนแล้ว จุดที่สำคัญในการแก้ไขจัดเป็นส่วนที่เร่งด่วนไม่สามารถปล่อยปละละเลยได้

ดังนั้นเราจึงเลือกพิจารณาโดยเลือกนำดีไวซ์ไครเวอร์ที่ติดต่อกับจอภาพและคีย์บอร์ดมาพัฒนา เพื่อพัฒนาให้ใช้บริการของไบออสแทนที่จะติดต่อกับหน่วยความจำแสดงผลโดยตรง เราพบว่ามีฟังก์ชันสองฟังก์ชันหลักที่ใช้ในการติดต่อกับหน่วยความจำจอภาพโดยตรงเพื่อส่งข้อมูลไปแสดงผล ฟังก์ชันที่กล่าวถึงอยู่ในไฟล์ 0.1.2/src/kernel/klib88.s ซึ่งเป็นฟังก์ชันในภาษาแอสเซมบลี คือ

- mem_vid_copy() ทำหน้าที่ในการนำข้อมูลจากหน่วยความจำของระบบไปยังหน่วยความจำที่ใช้ในการแสดงผลของจอภาพโดยตรง
- vid_vid_copy ทำหน้าที่ในการนำข้อมูลภายในหน่วยความจำแสดงผลไปยังตำแหน่งอื่นในหน่วยความจำแสดงผลของจอภาพ ใช้ทำหน้าที่ในการเลื่อนข้อมูลในจอภาพ การติดต่อกับหน่วยความจำจอภาพโดยตรงจะทำให้การแสดงผลทำได้อย่างรวดเร็วและไม่มีการกระตุก

จุดหลักในการที่จะทำให้เวอร์ชันนี้บรรลุจุดประสงค์คือการพัฒนาปรับปรุงให้สองฟังก์ชันหลักและฟังก์ชันที่มีการเกี่ยวข้อง ให้เรียกใช้บริการไบออส ซึ่งจะทำได้สามารถแสดงผลผ่านโปรแกรมเทอร์มินอลได้อย่างถูกต้อง

7.4.2 พิจารณาถึงข้อจำกัดของหน่วยความจำแบบโปรเทกต์เต็ดโหมด

เนื่องจากว่า ถ้าระบบปฏิบัติการมินิกซ์เริ่มได้ระบบภายใต้การจัดการหน่วยความจำแบบโปรเทกต์เต็ดโหมด การเรียกใช้บริการไบออสโดยวิธีการปกติจะไม่สามารถทำได้โดยตรง ต้องมีขั้นตอนปฏิบัติเพิ่มเติมไปจากขั้นตอนปกติ เพื่อให้สามารถเรียกใช้บริการไบออสได้ อย่างไรก็ตามโค้ดในส่วนนี้ในภายหลังจำเป็นต้องตัดออกอยู่แล้ว เนื่องจากการที่แพลตฟอร์มคอม86 ไม่สนับสนุนการจัดการหน่วยความจำแบบโปรเทกต์เต็ดโหมด

เราจะพิจารณาคัดโค้ดที่ใช้ในการเริ่มต้นระบบโดยใช้การจัดการหน่วยความจำแบบโปรเทกต์เต็ดโหมดออกไป พิจารณาที่ไฟล์แรก 0.1.2/src/kernel/mpx88.s โค้ดตัวหนา คือ โค้ดที่ตัดออก

```

push    cs
call    _cstart      ! cstart(cs, ds, mds, parmoff, parmlen)
add     sp, #5*2

cmp     _protected_mode, #0
jz      nosw         ! ok to switch to protected mode?

call    klib_init_prot ! initialize klib functions call
          real2prot     ! switch to protected mode

push    #0           ! set flags to known good state

```



```

    popf

nosw:
    jmp    _main    ! main()

```

ตัวแปร `protected_mode` เป็นตัวแปรที่ส่งผ่านมาจากโปรแกรมมอเอนเตอร์ ซึ่งพิจารณาเลือกที่จะเริ่มต้นระบบในโหมดไหน ถ้าการตรวจสอบในขั้นตอนนี้พบว่าต้องการเริ่มต้นระบบในโปรเทกต์โหมด จะเรียกฟังก์ชัน `klib_init_prot` และ `real2prot` เพื่อทำการสลับการใช้หน่วยความจำจากรีลโหมดไปเป็นโปรเทกต์โหมดโดยขั้นตอนนี้ ซึ่งเราไม่ต้องการดังนั้นจึงพิจารณาตัดโค้ดส่วนนี้ออกไป ต่อมาพิจารณาที่ไฟล์ `0.1.2/src/kernel/start.c` เพื่อสังเกตถึงการตรวจสอบค่าของตัวแปร `protected_mode`

```

/* Processor? */
processor = boot_parameters.bp_processor; /* 86, 186, ... */
...
/* Decide if mode is protected. */
#if _WORD_SIZE == 2
protected_mode = processor >= 286;
if (!protected_mode) mon_return = 0;
#endif

```

ตรวจสอบค่าจากตัวแปร `processor` อีกครั้ง โดยถ้าค่าในตัวแปร `processor` มากกว่า 286 ตัวแปร `protected_mode` จะเป็นหนึ่งหมายถึงสามารถเริ่มต้นระบบในโปรเทกต์โหมดได้ เราแก้ไขโค้ดตรงนี้ ได้เป็น

```

/* Processor? */
processor = boot_parameters.bp_processor; /* 86, 186, ... */
...
/* Decide if mode is protected. */
#if _WORD_SIZE == 2
protected_mode = 0;
mon_return = 0;
#endif

```

กำหนดให้ตัวแปร `protected_mode` เป็นศูนย์เสมอ ระบบก็จะไม่เริ่มต้นระบบในโปรเทกต์โหมดอย่างแน่นอน

เพื่อให้โค้ดลงขนาดลงอีกเราลบข้อมูลในการแสดงผลในไฟล์ `0.1.2/src/kernel/tty.c` ที่เป็นตัวนำออกคือ

```

#if (CHIP == INTEL)
/* Real mode, or 16/32-bit protected mode? */
#if _WORD_SIZE == 4
printf("Executing in 32-bit protected mode\n\n");
#else
printf("Executing in %s mode\n\n",
    protected_mode ? "16-bit protected" : "real");
#endif
#endif

```

7.4.3 พิจารณาฟังก์ชันแรก mem_vid_copy

พิจารณาโค้ดต้นฉบับคือ

```

_mem_vid_copy:
    push    si
    push    di
    push    es
    mov     bx, sp
    mov     si, MVC_ARGS(bx)          ! source
    mov     di, MVC_ARGS+2(bx)        ! destination
    mov     dx, MVC_ARGS+2+2(bx)      ! count
    mov     es, _vid_seg              ! destination is video segment
    cld                                ! make sure direction is up

mvc_loop:
    (1)    and     di, _vid_mask        ! wrap address
           mov     cx, dx               ! one chunk to copy
           mov     ax, _vid_size
           sub     ax, di
           cmp     cx, ax
           jbe     0f                  ! cx = min(cx, vid_size - di)
0:         sub     dx, cx               ! count -= cx
           shl     di, #1              ! byte address
           test    si, si              ! source == 0 means blank the screen
           jz      mvc_blank

mvc_copy:
    (2)    rep     ! copy words to video memory
           movs
           jmp     mvc_test

mvc_blank:
    mov     ax, _blank_color          ! ax = blanking character
    rep
    stos
    ! jmp     mvc_test                ! copy blanks to video memory

mvc_test:
    shr     di, #1                    ! word addresses
    test    dx, dx
    jnz     mvc_loop

mvc_done:
    pop     es
    pop     di
    pop     si
    ret

```

โค้ดในจุดที่ (1) ตัวหนาคือโค้ดที่เพิ่มเข้าไปดังนี้

```

mvc_loop:
    push    ax
    push    bx
    push    dx
    mov     ax, di                    ! screen address
    xor     dx, dx                    ! prepare to division

```



```

mov    bx, _scr_width          ! BX is a scr_width
div    bx                      ! AX div by BX
movb   ah, #0x02               ! set cursor position routine
movb   bh, #0x00               ! page 0
movb   dh, al                  ! row in DH, column in DL
int     0x10
pop     dx
pop     bx
pop     ax
and     di, _vid_mask          ! wrap address
mov     cx, dx                 ! one chunk to copy

```

หลักการคือคำนวณหาตำแหน่งของเคอร์เซอร์ปัจจุบันก่อน จากค่าใน di จากนั้นจึงเรียกบริการผ่านทางอินเทอร์รัปต์หมายเลข 10 ฐานสิบหกของไบออส ฟังก์ชันย่อยคือ AH = 02 เป็นการกำหนดตำแหน่งของเคอร์เซอร์เพื่อให้สามารถแสดงผลได้อย่างถูกต้อง

พิจารณาในโค้ดในจุดที่ (2) เปลี่ยนตัวหนาเป็นโค้ดต่อไปนี้

```

        jz     mvc_blank
mvc_copy:
    push    ax
    push    bx
    movb    bh, #0x00
mvc_print:
    cmp     cx, #0              ! end of char to print
    jz      mvc_endprint
    dec     cx                  ! one char to print
    lods    ds, [di]            ! get char from DS:SI
    movb    bl, ah              ! char attribute
    movb    ah, #0x0E           ! print screen routine
    int     0x10
    jmp     mvc_print
mvc_endprint:
    pop     bx
    pop     ax
    jmp     mvc_test
mvc_blank:
    push    ax
    push    bx
    movb    ah, #0x0E
    movb    al, #0x00
    mov     bx, #0x0007
mvc_pblank:
    cmp     cx, #0              ! end of char to print
    jz      mvc_endpblank
    dec     cx                  ! one char to print
    int     0x10
    jmp     mvc_pblank
mvc_endpblank:
    pop     bx
    pop     ax
mvc_test:
    shr     di, #1              ! word addresses

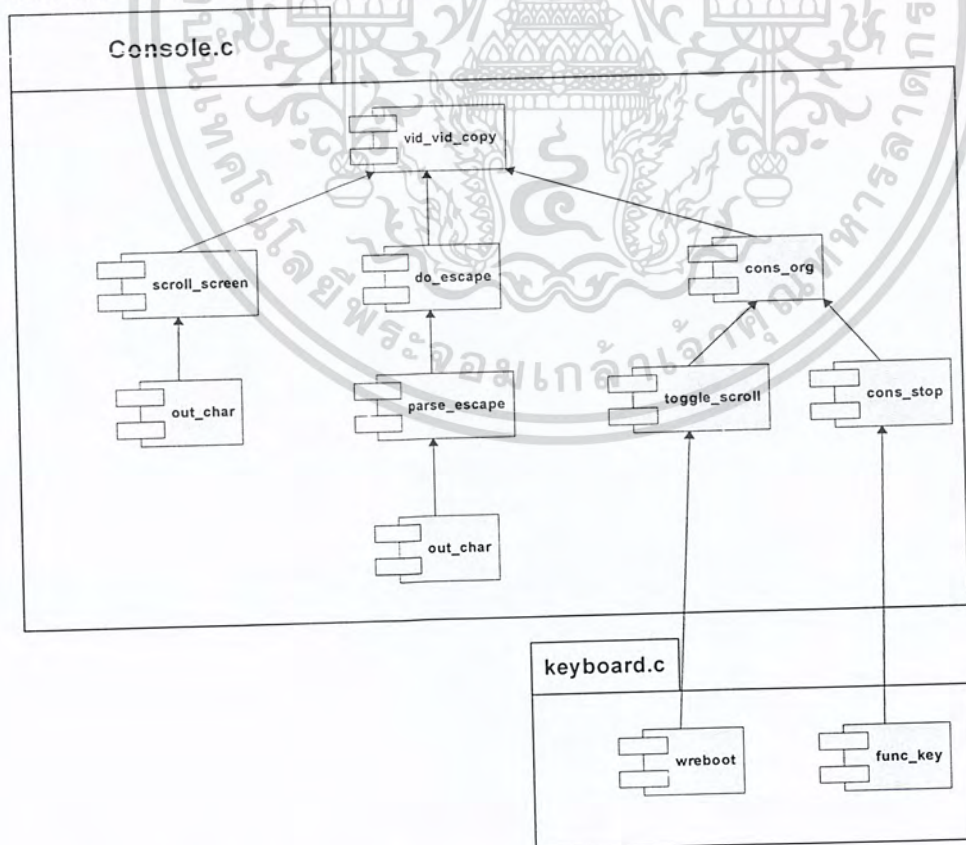
```

โค้ดในตัวหนาคือโค้ดที่เพิ่มเข้าไป มีหลักการทำงานคือ คำนวณตัวอักษรที่ยังมีอยู่ ที่ต้องการจำพิมพ์ วนจนพิมพ์หมด โดยนำค่ามาจากรีจิสเตอร์ si โดยใช้คำสั่ง lods นำค่ามาเก็บไว้ที่ al จากนั้นจึงส่งพิมพ์โดยเรียกบริการผ่านทางอินเทอร์รัปต์หมายเลข 10 ฐานสิบหกของไบออสฟังก์ชันย่อยคือ AH = 0Eh โดยฟังก์ชันย่อยนี้สามารถแสดงอักขระพิเศษได้ตามความหมายของอักขระนั้นๆ ได้ทางจอภาพ เช่น ถ้าเป็นอักขระให้ขึ้นบรรทัดใหม่ การแสดงผลก็จะทำการขึ้นบรรทัดใหม่ให้ด้วย

ฟังก์ชันนี้ยังทำหน้าที่ในการเคลียร์สกรีนอีกด้วย ในกรณีที่ค่าอินพุต si เป็นศูนย์ ฟังก์ชันนี้ก็ต้องทำการพิมพ์ตัวอักษรเปล่าบนหน้าจอเพื่อทำการล้างหน้าจอ จะเห็นได้จากในรูป mvc_pblank จนถึง mvc_endpblank ซึ่งเรียกใช้ผ่านบริการของไบออสในการแสดงผลเช่นเดียวกัน

7.4.4 พิจารณาฟังก์ชันในการแสดงผลอีกฟังก์ชันหนึ่ง vid_vid_copy

vid_vid_copy ทำหน้าที่ในการนำข้อมูลภายในหน่วยความจำแสดงผลไปยังตำแหน่งอื่นในหน่วยความจำแสดงผลของจอภาพ ใช้ทำหน้าที่ในการเคลื่อนข้อมูลในจอภาพ การติดต่อกับหน่วยความจำจอภาพโดยตรง ซึ่งทำให้การแสดงผลทำได้อย่างรวดเร็วและไม่มีการกระตุก จะเห็นได้ว่าหน่วยความจำในการแสดงผลนั้น ไม่มีอยู่จริงในแพลตฟอร์มคอม86 ถ้าใช้ฟังก์ชันในการทำงานจะทำให้ค่าในหน่วยความจำเกิดความผิดพลาดขึ้นอย่างแน่นอน เนื่องมาจากการไม่มีอยู่จริงของหน่วยความจำจอภาพในคอม86 แต่หน่วยความจำส่วนนี้เป็นหน่วยความจำที่อาจจะอยู่ในโปรแกรมอะไรก็ได้ ดังนั้นจึงจำเป็นที่จะต้องพิจารณาตัดฟังก์ชันนี้ออก โดเนลบทฟังก์ชันนี้ออกจาก klib88.s ทั้งหมด รวมไปถึงการประกาศฟังก์ชันด้วย จากนั้นพิจารณาถึงความสัมพันธ์ของฟังก์ชัน vid_vid_copy กับฟังก์ชันอื่น



รูปที่ 23 แสดงความสัมพันธ์ของฟังก์ชัน vid_vid_copy

7.4.4.1 พิจารณาในฟังก์ชัน `scroll_screen`

จากการพิจารณา ฟังก์ชันนี้ทำหน้าที่ในการเลื่อนจอภาพขึ้นและลง โดยเรียกฟังก์ชัน `vid_vid_copy` เพื่อทำหน้าที่ในการเลื่อนขึ้นและลงอีกครั้งหนึ่ง โดยมีลักษณะในการเลื่อนแบ่งออกเป็นสามแบบคือ

- Software scrolling ใช้กับการ์ดแสดงผลแบบเซอร์กิวลิส
- Hardware scrolling ใช้กับการ์ดแสดงผลแบบโมโนหรือซีจีเอการ์ด (CGA)
- Hardware scrolling without wrapping ใช้กับการ์ดแสดงผลตั้งแต่อีจีเอขึ้นไป (EGA)

สำหรับมินิคอมพิวเตอร์ 86 นั้นเป็นโมโน เนื่องจากโค้ดในส่วนที่เป็น Hardware scrolling ไม่ได้มีการเรียกใช้ `vid_vid_copy` ดังนั้นโค้ดที่มีการเรียกใช้ `vid_vid_copy` จากทั้ง Software scrolling และ Hardware scrolling without wrapping ออกได้ โค้ดตัวหนาถือคือโค้ดที่ตัดออก

```
if (dir == SCROLL_UP) {
    /* Scroll one line up in 3 ways: soft, avoid wrap, use origin.*/
    if (softscroll) {
        vid_vid_copy(cons->c_start + scr_width, cons->c_start, chars);
    } else
    if (!wrap && cons->c_org + scr_size + scr_width >= cons->c_limit) {
        vid_vid_copy(cons->c_org + scr_width, cons->c_start, chars);
        cons->c_org = cons->c_start;
    } else {
        cons->c_org = (cons->c_org + scr_width) & vid_mask;
    }
    new_line = (cons->c_org + chars) & vid_mask;
} else {
    /* Scroll one line down in 3 ways: soft, avoid wrap, use origin.*/
    if (softscroll) {
        vid_vid_copy(cons->c_start, cons->c_start + scr_width, chars);
    } else
    if (!wrap && cons->c_org < cons->c_start + scr_width) {
        new_org = cons->c_limit - scr_size;
        vid_vid_copy(cons->c_org, new_org + scr_width, chars);
        cons->c_org = new_org;
    } else {
        cons->c_org = (cons->c_org - scr_width) & vid_mask;
    }
    new_line = cons->c_org;
}
```

7.4.4.2 พิจารณาฟังก์ชัน `do_escape`

ฟังก์ชัน `do_escape` ทำหน้าที่ในการแสดงตัวอักษรตามมาตรฐาน ANSI โดยใช้อักขระแอสเคปในการขึ้นต้นตัวอักษรเพื่อบอกให้เทอร์มินอลหรือจอภาพที่ใช้แสดงผล แสดงผลตามที่ต้องการ พิจารณาการทำงานของฟังก์ชัน `vid_vid_copy` ในโค้ดตัวหนาต่อไปนี้

```
case 'L':          /* ESC [nL inserts n lines at cursor */
    n = value;
    if (n < 1) n = 1;
```

```

if (n > (scr_lines - cons->c_row))
    n = scr_lines - cons->c_row;

src = cons->c_org + cons->c_row * scr_width;
dst = src + n * scr_width;
count = (scr_lines - cons->c_row - n) * scr_width;
vid_vid_copy(src, dst, count);
blank_color = cons->c_blank;
mem_vid_copy(BLANK_MEM, src, n * scr_width);
break;

case 'M':          /* ESC [nM deletes n lines at cursor */
    n = value;
    if (n < 1) n = 1;
    if (n > (scr_lines - cons->c_row))
        n = scr_lines - cons->c_row;

    dst = cons->c_org + cons->c_row * scr_width;
    src = dst + n * scr_width;
    count = (scr_lines - cons->c_row - n) * scr_width;
    vid_vid_copy(src, dst, count);
    blank_color = cons->c_blank;
    mem_vid_copy(BLANK_MEM, dst + count, n * scr_width);
    break;

case '@':          /* ESC [n@ inserts n chars at cursor */
    n = value;
    if (n < 1) n = 1;
    if (n > (scr_width - cons->c_column))
        n = scr_width - cons->c_column;

    src = cons->c_cur;
    dst = src + n;
    count = scr_width - cons->c_column - n;
    vid_vid_copy(src, dst, count);
    blank_color = cons->c_blank;
    mem_vid_copy(BLANK_MEM, src, n);
    break;

case 'P':          /* ESC [nP deletes n chars at cursor */
    n = value;
    if (n < 1) n = 1;
    if (n > (scr_width - cons->c_column))
        n = scr_width - cons->c_column;

    dst = cons->c_cur;
    src = dst + n;
    count = scr_width - cons->c_column - n;
    vid_vid_copy(src, dst, count);
    blank_color = cons->c_blank;
    mem_vid_copy(BLANK_MEM, dst + count, n);
    break;

```

จากการลบบึงก์ชัน vid_vid_copy ออกทำให้มีผลให้การแสดงอักขระต่อไปนี้ไม่ถูกต้อง

อักขระ	ความหมาย
ESC [nL	inserts n lines at cursor
ESC [nM	deletes n lines at cursor
ESC [nP	deletes n chars at cursor
ESC [n@	inserts n chars at cursor

ตาราง 10 แสดงอักขระแอสเคปที่แสดงผลไม่ถูกต้องในเวอร์ชัน 0.1.2

7.4.4.3 พิจารณาฟังก์ชัน `cons_org0`

พิจารณาความสัมพันธ์ขึ้นไปจะเห็นว่าสัมพันธ์กับ `toggle_scroll` ฟังก์ชันนี้ลบออกได้ แต่ต้องไปจัดการลบออกที่ฟังก์ชัน `func_key` ใน `keyboard.c` ด้วย

```
...
case F1:    p_dmp(); break;          /* print process table */
case F2:    map_dmp(); break; /* print memory map */
case F3:    toggle_scroll(); break;
#if ENABLE_NETWORKING
case F5:    dp_dump(); break;      /* network statistics */
...

```

เหลืออีกฟังก์ชันคือ `cons_stop` ให้ลบการเรียกใช้งาน `cons_org0` ออกได้

```
PUBLIC void cons_stop()
{
/* Prepare for halt or reboot. */
cons_org0();
softscroll = 1;
select_console(0);
cons_table[0].c_attr = cons_table[0].c_blank = BLANK_COLOR;

```

จากนั้นจึงทำการลบฟังก์ชัน `cons_org0` ออกและรวมไปถึงลบเลเบลต่อไปนี้ไปด้วย

```
FORWARD _PROTOTYPE( void cons_write, (struct tty *tp) );
FORWARD _PROTOTYPE( void cons_echo, (tty_t *tp, int c); );
FORWARD _PROTOTYPE( void out_char, (console_t *cons, int c) );
FORWARD _PROTOTYPE( void beep, (void) );
FORWARD _PROTOTYPE( void do_escape, (console_t *cons, int c) );
);
FORWARD _PROTOTYPE( void flush, (console_t *cons) );
);

```

```

FORWARD _PROTOTYPE( void parse_escape, (console_t *cons, int c)
);
FORWARD _PROTOTYPE( void scroll_screen, (console_t *cons, int
dir));
FORWARD _PROTOTYPE( void set_6845, (int reg, unsigned val)
);
FORWARD _PROTOTYPE( void stop_beep, (void)
);
FORWARD _PROTOTYPE( void cons_org0, (void)
);
FORWARD _PROTOTYPE( void ga_program, (struct sequence *seq)
);
FORWARD _PROTOTYPE( void cons_ioctl, (tty_t *tp)
);

```

7.4.5 พิจารณาฟังก์ชันที่มีการติดต่อกับฮาร์ดแวร์โดยตรงในไฟล์ *console.c*

มีดังต่อไปนี้

- ❑ `set_6845()` ทำหน้าที่ในการติดต่อกับพอร์ตสื่อสารของจอภาพโดยตรง
- ❑ `beep()` ทำหน้าที่ในการสร้างเสียง ติดต่อกับฮาร์ดแวร์โดยตรงเช่นเดียวกัน
- ❑ `stop_beep()` หยุดเสียงที่ทำการสร้างโดย `beep()`
- ❑ `ga_program()` ทำหน้าที่ในการนำรูปฟอนต์ใหม่ขึ้นไปเก็บไว้บนหน่วยความจำภายในจอภาพ

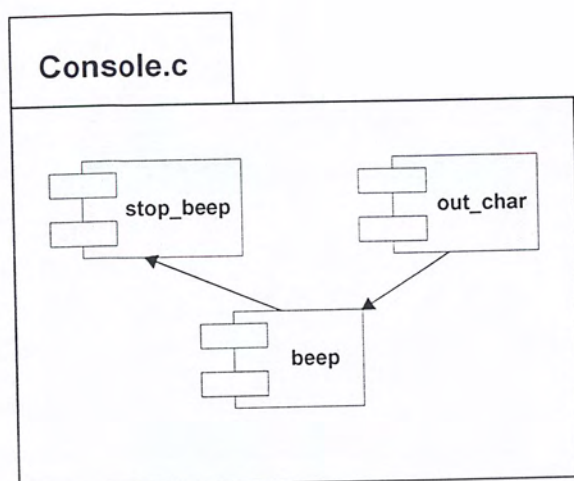
ฟังก์ชันทั้งหมดนี้ พิจารณาตัดออกทั้งหมด เพราะไม่มีความจำเป็นต้องใช้ ก่อนอื่นลบการประกาศฟังก์ชันในไฟล์ *console.c*

```

FORWARD _PROTOTYPE( void cons_write, (struct tty *tp) );
FORWARD _PROTOTYPE( void cons_echo, (tty_t *tp, int c); );
FORWARD _PROTOTYPE( void out_char, (console_t *cons, int c) );
FORWARD _PROTOTYPE( void beep, (void) );
FORWARD _PROTOTYPE( void do_escape, (console_t *cons, int c)
);
FORWARD _PROTOTYPE( void flush, (console_t *cons)
);
FORWARD _PROTOTYPE( void parse_escape, (console_t *cons, int c)
);
FORWARD _PROTOTYPE( void scroll_screen, (console_t *cons, int
dir));
FORWARD _PROTOTYPE( void set_6845, (int reg, unsigned val)
);
FORWARD _PROTOTYPE( void stop_beep, (void)
);
FORWARD _PROTOTYPE( void ga_program, (struct sequence *seq)
);
FORWARD _PROTOTYPE( void cons_ioctl, (tty_t *tp)
);

```

จากนั้นพิจารณาความสัมพันธ์ของแต่ละฟังก์ชัน เพื่อให้มั่นใจได้ไม่มีการเรียกใช้ฟังก์ชันเหล่านี้ เริ่มต้นที่ `beep` กับ `stop_beep` มีความสัมพันธ์ดังรูป

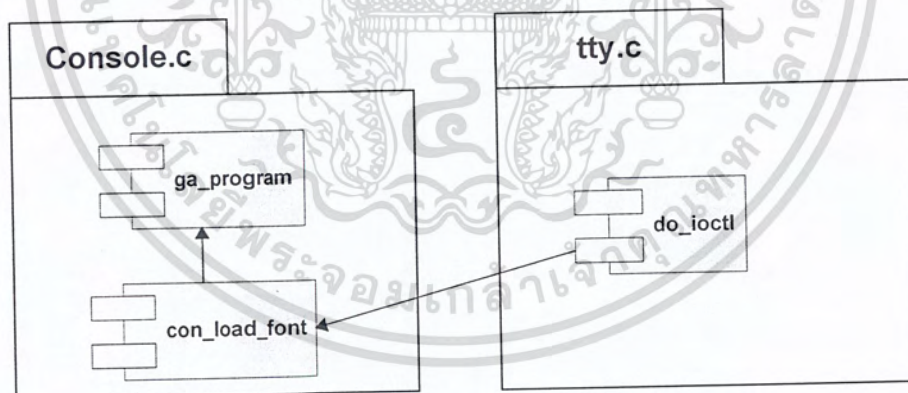


รูปที่ 24 แสดงความสัมพันธ์ของฟังก์ชัน beep กับ stop_beep

ทั้งฟังก์ชัน beep และ stop_beep ลบออกได้และลบการเรียกใช้ beep ในฟังก์ชัน out_char

```
case 007:      /* ring the bell */
    flush(cons); /* print any chars queued for output */
    beep ();
    return;
```

ต่อมาพิจารณาฟังก์ชัน ga_program มีความสัมพันธ์คือ



รูปที่ 25 แสดงความสัมพันธ์ของฟังก์ชัน ga_program

พิจารณาลบทั้งสองฟังก์ชันออกคือ ga_program และ con_load_font ผลกระทบที่เกิดขึ้นคือการที่ระบบไม่สามารถโหลดฟอนต์ใหม่ได้โดยใช้โปรแกรม load_font หรืออื่นๆ ในการติดตั้งฟอนต์ใหม่ให้ระบบ จากความสัมพันธ์ต้องไปทำการลบการเรียกใช้ฟังก์ชัน con_load_font ใน do_ioctl ในไฟล์ tty.c ด้วย ตัวหนาคือ โค้ดที่ลบออกดังนี้

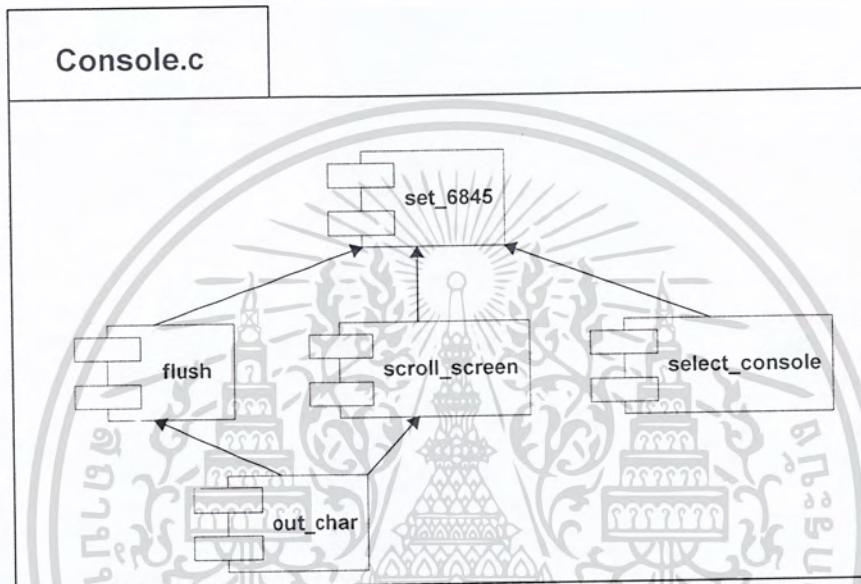
...

```

case TIOCSFON:
    /* Load a font into an EGA or VGA card (hs@hck.hr) */
    if (isconsole(tp)) r = con_loadfont(user_phys);
    break;
#endif
...

```

พิจารณาฟังก์ชันที่ติดต่อกับพอร์ตของจอภาพโดยตรงคือ set_6845 พิจารณาความสัมพันธ์ของฟังก์ชันได้
คือ



รูปที่ 26 แสดงความสัมพันธ์ของฟังก์ชัน set_6845

จะเห็นได้ว่ามีความสัมพันธ์โดยการเรียกใช้ในหลายฟังก์ชัน ทำการพิจารณาถึงโค้ดที่มีการเรียกใช้ทั้งหมด พิจารณาถึงฟังก์ชัน flush ลบโค้ดตัวหนาออก

```

cur = cons->c_org + cons->c_row * scr_width + cons->c_column;
if (cur != cons->c_cur) {
    if (cons == curcons) set_6845(CURSORM, cur);
    cons->c_cur = cur;
}
}

```

ต่อมาใน scroll_screen ลบโค้ดที่เป็นตัวหนาออก

```

mem_vid_copy(BLANK_MEM, new_line, scr_width);

/* Set the new video origin. */
if (cons == curcons) set_6845(VID_ORG, cons->c_org);
flush(cons);
}

```


และสุดท้ายใน select_console

```
{
/* Set the current console to console number 'cons_line'. */

if (cons_line < 0 || cons_line >= nr_cons) return;
current = cons_line;
curcons = &cons_table[cons_line];
set_6845(VID_ORG, curcons->c_org);
set_6845(CURSOR, curcons->c_cur);
}
```

เป็นอันเสร็จสิ้นในเวอร์ชัน 0.1.2

7.5 มินิغبั่นคอม86 เวอร์ชัน 0.1.3

ในเวอร์ชัน 0.1.3 มีจุดประสงค์ในการพัฒนาคือ

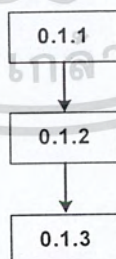
เวอร์ชันในการพัฒนา	จุดประสงค์ในการพัฒนา
0.1.3	ปิดการทำงานของคีย์บอร์ดวีซีไครเวอร์ มุ่งเน้นไปที่การพัฒนาในส่วนที่เกี่ยวข้องกับอินเทอร์พรีตบนแพลตฟอร์มคอม86 ก่อน

ตาราง 11 แสดงจุดประสงค์ในการพัฒนามินิغبั่นเวอร์ชัน 0.1.3

ลำดับในการพัฒนาเป็นไปตามลำดับในสายของการพัฒนาจากเวอร์ชัน 0.1.2 ลงมาเป็น 0.1.3

สายการพัฒนามินิغبั่น

สายที่ 1

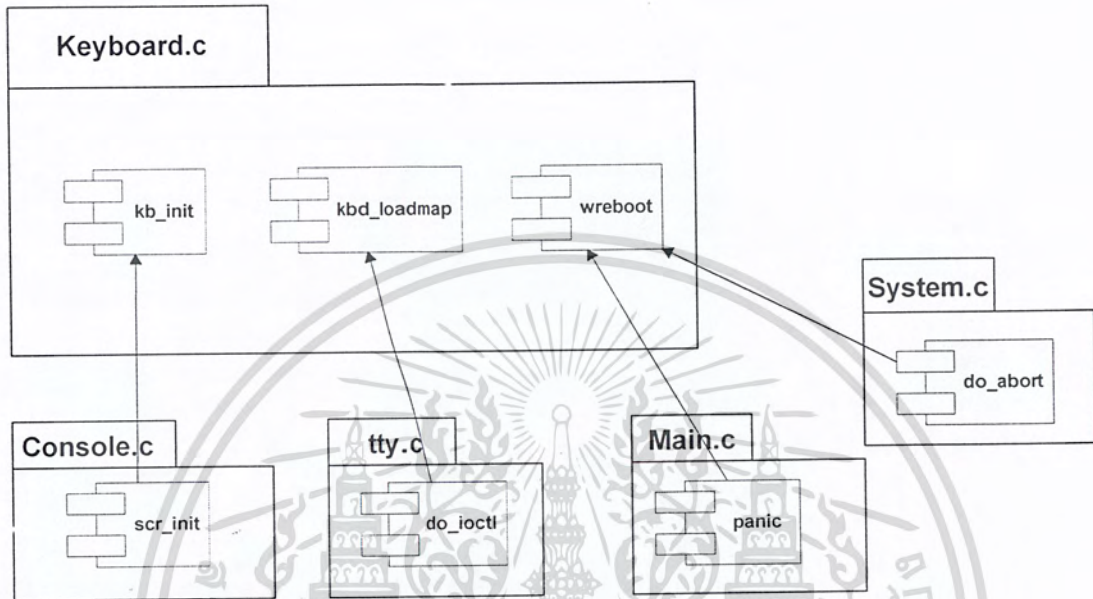


รูปที่ 27 แสดงลำดับในการพัฒนามินิغبั่นเวอร์ชัน 0.1.3

จุดประสงค์หลักของเวอร์ชันนี้คือการปิดการทำงานของคีย์บอร์ดวีซีไครเวอร์ เนื่องจากคีย์บอร์ดวีซีไครเวอร์ยังมีส่วนที่ติดต่อกับฮาร์ดแวร์โดยตรง จะมุ่งเน้นไปที่การพัฒนาในส่วนที่เกี่ยวข้องกับอินเทอร์พรีตบนแพลตฟอร์มคอม86 ก่อน แล้วจึงค่อยกลับมาพัฒนาในส่วนรับข้อมูลผ่านคีย์บอร์ดอีกครั้งในเวอร์ชันถัดไป

เพิ่มเลเบล ENABLE_KEYBOARD ไปไว้ในต้นไฟล์และท้ายไฟล์ keyboard.c เพื่อให้สามารถเลือกได้ และเพิ่ม ENABLE_KEYBOARD ไว้ใน 0.1.3/include/minix/config.h

พิจารณาความสัมพันธ์ฟังก์ชันที่มีการเรียกใช้โดยไฟล์อื่น พบสามฟังก์ชันคือ kb_init kbd_loadmap และ wreboot พิจารณาความสัมพันธ์ของทั้งสามฟังก์ชันนี้กับฟังก์ชันอื่นในไฟล์อื่น



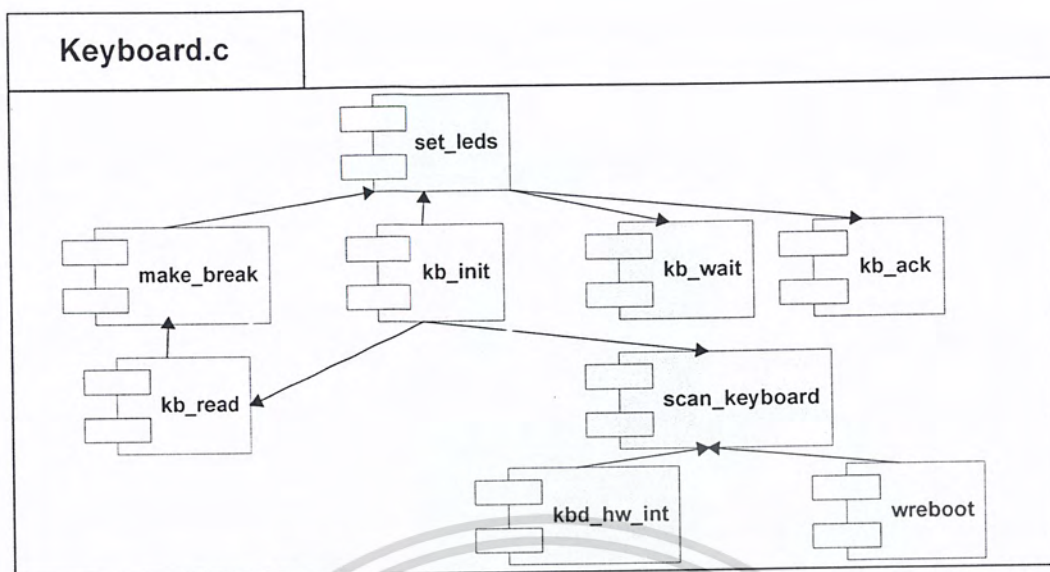
รูปที่ 28 แสดงความสัมพันธ์กับไฟล์ keyboard.c

แทรก

```
#if ENABLE_KEYBOARD
...
#endif
```

ในฟังก์ชันที่มีการเรียกใช้ kbd_loadmap wreboot คือฟังก์ชัน do_ioctl ในไฟล์ tty.c และฟังก์ชัน panic ใน main.c และ do_abort ในไฟล์ system.c

ส่วนฟังก์ชัน kb_init นั้นไม่สามารถแทรกได้โดยตรงในฟังก์ชัน scr_init ในไฟล์ console.c เนื่องจากมีตัวแปรที่ต้องเริ่มต้นหลายตัวดังนั้นพิจารณาความสัมพันธ์ย่อยของฟังก์ชัน kb_init



รูปที่ 29 แสดงความสัมพันธ์ของฟังก์ชัน `kb_init` กับฟังก์ชันอื่น

เราจะพิจารณาแทรก

```
#if ENABLE_KEYBOARD
...
#endif
```

ในฟังก์ชัน `set_leds` ทั้งหมด

ในฟังก์ชัน `scan_keyboard` ทั้งหมด

ในฟังก์ชัน `wreboot` ทั้งหมด

จนถึงเวอร์ชันนี้เป็นการเตรียมความพร้อมก่อนที่จะนำไปพัฒนาในส่วนที่เกี่ยวข้องกับระบบอินเทอร์พรีตของระบบ ขั้นตอนตั้งแต่เวอร์ชัน 0.1.1 จนถึง 0.1.3 เป็นการตัดโค้ดที่เกี่ยวข้องกับส่วนของฮาร์ดแวร์ทั้งหมดในไอบีเอ็มพีซีคอมแพคทิเบิล หรือปิดการทำงานในส่วนที่ยังไม่ได้พิจารณาไว้ ซึ่งโค้ดจนถึงขั้นตอนนี้จะเหลือเพียงโค้ดในส่วนระบบอินเทอร์พรีตของแพลตฟอร์มคอม86 และดีไวส์ไครเวอร์เกี่ยวกับเวลาเท่านั้น ซึ่งจะได้รับการพิจารณาแก้ไขในเวอร์ชันถัดไป คือ เวอร์ชัน 0.2.1 ซึ่งจะขึ้นเป็นสายการพัฒนาที่แก้ไขโค้ดในส่วนที่เกี่ยวข้องกับฮาร์ดแวร์ทั้งหมดให้เป็นไปตามคุณลักษณะของแพลตฟอร์มคอม86 เพื่อให้สามารถทำงานได้อย่างสมบูรณ์บนแพลตฟอร์มคอม86 ต่อไป

7.6 มินิซันคอม86 เวอร์ชัน 0.2.1

ในเวอร์ชัน 0.2.1 มีจุดประสงค์ในการพัฒนาคือ

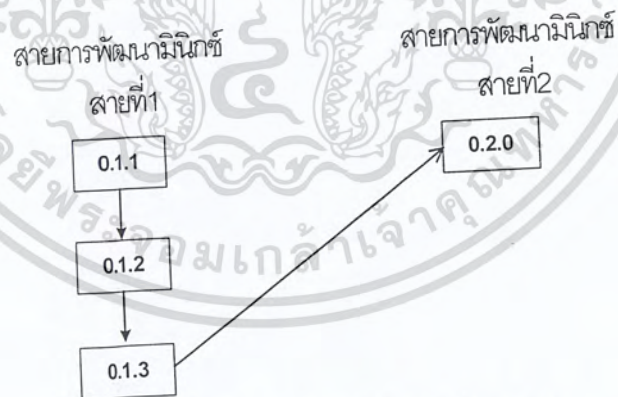
เวอร์ชันในการพัฒนา	จุดประสงค์ในการพัฒนา
0.2.0	เวอร์ชันสำเนาจากเวอร์ชัน 0.1.3
0.2.1	ปรับปรุงและแก้ไขส่วนที่เกี่ยวข้องกับระบบอินเทอร์พรีตของระบบ ปรับปรุงและเปลี่ยนแปลงดีไวส์ไดรเวอร์ของเวลาของระบบ

ตาราง 12 แสดงจุดประสงค์ในการพัฒนามินิซ์เวอร์ชัน 0.2.1

เป็นการเริ่มต้นการพัฒนาในสายของการพัฒนาสายที่สองแล้ว ซึ่งเป็นสายที่มุ่งแก้ไขและปรับปรุงโค้ดที่เกี่ยวข้องกับฮาร์ดแวร์ของระบบทั้งหมดให้สามารถทำงานได้บนแพลตฟอร์มคอม86

เวอร์ชัน 0.2.0 นั้นเป็นเวอร์ชันสำเนาจากเวอร์ชัน 0.1.3 ซึ่งจะเห็นได้ว่าการพัฒนาสายที่สองจะพัฒนาต่อจากสายที่หนึ่งที่เวอร์ชัน 0.1.3 ซึ่งได้เตรียมความพร้อมในเรื่องของโค้ดที่ไม่เกี่ยวข้องได้ถูกตัดหรือปิดการทำงานไว้ทั้งหมดแล้ว จากเวอร์ชัน 0.2.0 จะพัฒนาให้เป็น 0.2.1 โดยมุ่งเน้นในการปรับปรุงและแก้ไขโค้ดในส่วนที่เกี่ยวข้องกับระบบอินเทอร์พรีตของระบบ และ ส่วนที่เกี่ยวข้องกับดีไวส์ไดรเวอร์ของเวลาของระบบ ต้องทำสองส่วนนี้ไปพร้อมๆ กัน เพราะถ้าแยกก็ไม่สามารถทราบได้เลยว่า แก้ไขได้ถูกต้องหรือไม่ ขาดส่วนหนึ่งส่วนใดไปหรือแก้ไขส่วนหนึ่งส่วนใดไม่ถูกต้อง มินิซก็ไม่สามารถเริ่มต้นระบบบนแพลตฟอร์มคอม86 ได้

มีลำดับในการพัฒนาดังรูป



รูปที่ 30 แสดงลำดับในการพัฒนามินิซ์เวอร์ชัน 0.2.1

และมีงานที่ต้องกระทำเพื่อให้บรรลุจุดประสงค์ในการพัฒนาเวอร์ชันนี้ดังต่อไปนี้

1. ศึกษาเรื่องการใช้งานอินเทอร์พรีตของโปรเซสเซอร์ของแพลตฟอร์มคอม86 ซึ่งใช้โปรเซสเซอร์ AM186CC เป็นโปรเซสเซอร์ในระบบฝังตัวจากบริษัทเอเอ็มดี (AMD)
2. ศึกษาเรื่องการใช้งานรีจิสเตอร์เวลาของโปรเซสเซอร์ของแพลตฟอร์มคอม86
3. ศึกษาเรื่องระบบอินเทอร์พรีตในระบบปฏิบัติการมินิซและทำการแก้ไขและปรับปรุง
4. ศึกษาดีไวส์ไดรเวอร์เวลาของระบบปฏิบัติการมินิซ เพื่อแก้ไขและปรับปรุง

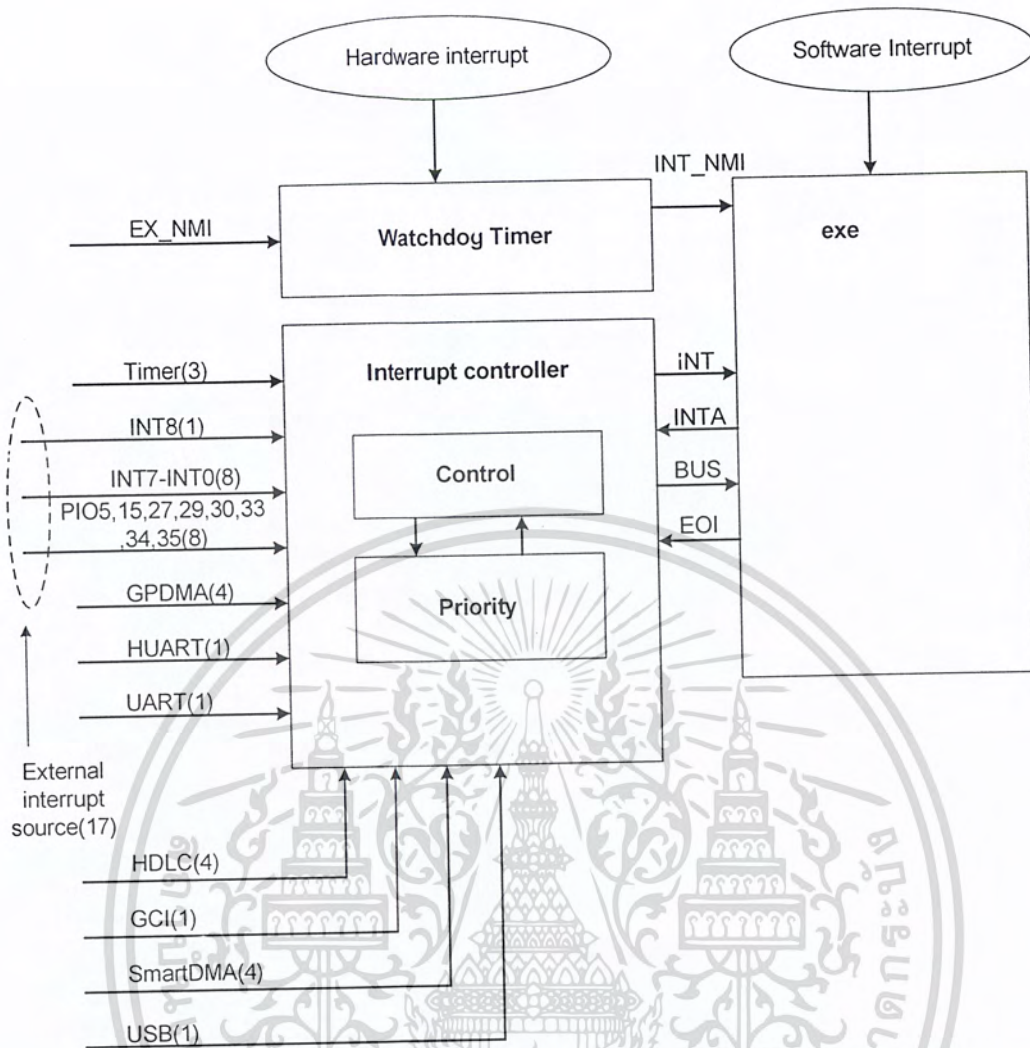
7.6.1 ศึกษาเรื่องการใช้งานอินเทอร์รัปต์ของโปรเซสเซอร์ของแพลตฟอร์มคอม86

โปรเซสเซอร์ AM186CC เป็นโปรเซสเซอร์ระบบฝังตัวจากบริษัทเอเอ็มดี มีอินเทอร์รัปต์ภายในที่สามารถเปิดการทำงานได้จากผู้ใช้ (Maskable Interrupt) ทั้งหมด 19 อินเทอร์รัปต์ คือ

- 3 Timers
- 2 UARTs (Hight Speed UARTs and Normal UARTs)
- 4 HDCL Channels
- 1 GCI
- 4 couple of SmartDMA
- 4 General purpose DMA
- 1 USB

สำหรับอินเทอร์รัปต์ภายในนั้นมีความหมายว่าเป็นอินเทอร์รัปต์ที่เป็นกึ่งประกอบในตัวแกนของโปรเซสเซอร์ เนื่องจากโปรเซสเซอร์ AM186CC เป็นโปรเซสเซอร์ในระบบฝังตัวทำให้ได้รวมระบบหลายๆ ระบบเช่น ระบบเวลา ระบบอินเทอร์รัปต์ ระบบติดต่อผ่านพอร์ตอนุกรม ไว้ภายในตัวโปรเซสเซอร์ ถ้าเทียบกับไอบีเอ็มคอมแพคทีเบิลแล้ว ระบบที่ยกตัวอย่างมานี้จะแยกอยู่ในหน่วยประมวลผลต่างหากเช่น ระบบเวลาก็ 8253 หรือระบบอินเทอร์รัปต์ก็ 8259 เป็นต้นซึ่งจะเป็นหน่วยประมวลผลที่แยกออกมาจากแกนของโปรเซสเซอร์ แต่อินเทอร์รัปต์ภายในที่กล่าวมานี้ โปรเซสเซอร์ AM186CC~ นี้ได้รวมระบบต่างๆ เหล่านี้ไว้ภายในตัวโปรเซสเซอร์ไว้เสร็จสรรพ พร้อมใช้งาน

มีไดอะแกรมแสดงองค์ประกอบของหน่วยประมวลผลอินเทอร์รัปต์ใน AM186CC เป็น



รูปที่ 31 ไดอะแกรมองค์ประกอบของระบบอินเทอร์รัปต์ใน AM186CC

จากไดอะแกรมจะเห็นได้ว่า ระบบอินเทอร์รัปต์ได้รวมไว้ภายในแกนของโปรเซสเซอร์ AM186CC แล้ว และมากกว่านั้นยังมีอินเทอร์รัปต์ภายนอกอีกได้ 17 อินเทอร์รัปต์ นับว่ามากเลยทีเดียว

สำหรับการใช้งานระบบอินเทอร์รัปต์ของ AM186CC เป็นไปตามขั้นตอนดังนี้

1. สำหรับอินเทอร์รัปต์ภายนอกคือ INT0 ถึง INT8 และ PIO ต้องใช้คำสั่ง CLI ก่อนเสมอ
2. สำหรับอินเทอร์รัปต์ที่เป็น PIO ให้ปรับแต่งค่าในรีจิสเตอร์ PIODIRx และ PIOINDEX ให้เรียบร้อย
3. สำหรับอินเทอร์รัปต์ที่เป็น INT0 ถึง INT8 ให้ตั้งค่าโพลาไรตีให้ถูกต้องในรีจิสเตอร์ INTPOL
4. ปรับแต่งค่าใน CHxCON เพื่อเลือกว่าต้องการจะใช้อินเทอร์รัปต์ใดให้ทำงาน กำหนดค่าความสำคัญต่ำสุดที่ยอมให้เกิดได้ในระบบ กำหนดคตรีจิสเตอร์ PRIMSK

Priority Mask (PRIMSK)**Offset 328h**

Register 0																
Bit Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res													PRM			
Software Read/Write	R												R/W			
Hardware Set/Clear	—												—			
Chip Reset Default	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

รูปที่ 32 รีจิสเตอร์ PRIMSK

- โดยค่าปกติจะเป็นค่ามาตรฐานอยู่แล้ว ดังนั้นขั้นตอนนี้อาจจะไม่ต้องทำก็ได้
- กำหนดลำดับความสำคัญในบิต PRI ใน CHxCON
- เปิดหรือปิด (Unmask or Mask) อินเทอร์รัปต์ได้ที่รีจิสเตอร์ IMASK

Interrupt Mask (IMASK)**Offset 326h**

Interrupt Mask (IMASK)																
Bit Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Res	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
Software Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Hardware Set/Clear	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Chip Reset Default	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

รูปที่ 33 รีจิสเตอร์ IMASK

- ตั้งค่าในรีจิสเตอร์ SHMASK ในการเปิด INT0 ถึง INT8 หรือ PIO ร่วมกันใน Channel 14
- กำหนดให้อินเทอร์รัปต์ทำงานโดยใช้คำสั่ง STI

และเมื่อเกิดกระบวนการอินเทอร์รัปต์ จะมีลำดับการทำงานดังนี้

- ร้องขออินเทอร์รัปต์ โดยขอหมายเลขอินเทอร์รัปต์ที่เกิดและคำนวณหาอินเทอร์รัปต์เวกเตอร์เพื่อจะได้กระโดดไปทำงานที่ตำแหน่งของอินเทอร์รัปต์เซอร์วิสรูทีนที่เกิดขึ้นได้
- บริการอินเทอร์รัปต์ โดยจะเกิดกระบวนการคือ โพรเซสเซอร์จะเก็บค่าตำแหน่งการทำงานปัจจุบันลงสแต็ก จากนั้นเคลียร์ค่า IF เพื่อปิดการทำงานของอินเทอร์รัปต์ จากนั้น โพรเซสเซอร์จะกระโดดไปทำงานที่ตำแหน่งของอินเทอร์รัปต์เซอร์วิสรูทีนที่เกิดขึ้น
- ตอบรับอินเทอร์รัปต์ ตั้งค่าภายในเพื่อบอกว่าอินเทอร์รัปต์ใดกำลังให้บริการอยู่ จะเกิดบริการซ้อนไม่ได้
- ส่งสัญญาณเพื่อจบกระบวนการอินเทอร์รัปต์ ซอฟต์แวร์ต้องเขียนค่าลงในรีจิสเตอร์ EOI เพื่อบอกว่าอินเทอร์รัปต์ใดเสร็จสิ้นหรือทำงานจนหมดแล้ว

End-of-Interrupt (EOI)

Offset 320h

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit Name	NSPEC	Res											S			
Software Read/Write	W	R											W			
Hardware Set/Clear	C	—											C			
Chip Reset Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

รูปที่ 34 รีจิสเตอร์ EOI

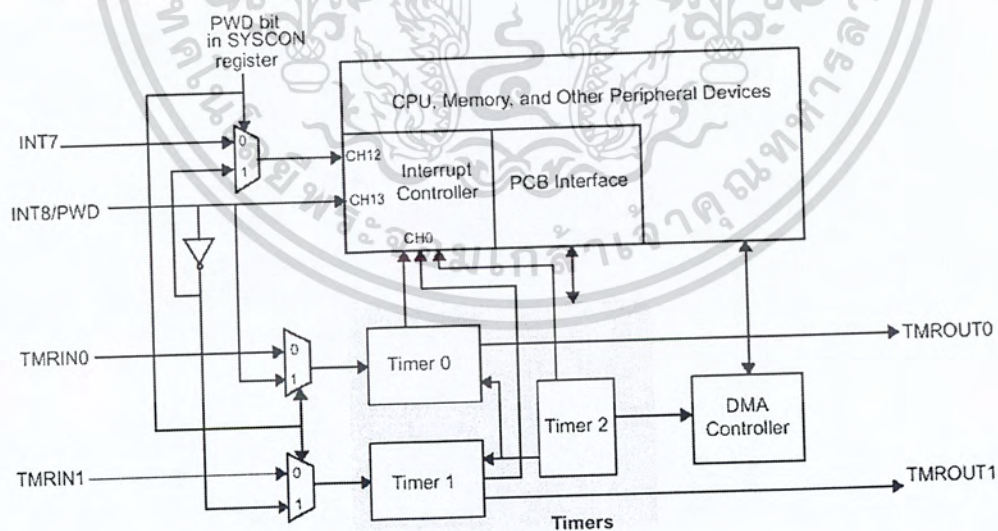
5. กลับมาจากอินเทอร์รัปต์ โคนโปรเซสเซอร์จะคืนค่าแฟล็กที่เก็บไว้แล้วกลับไปทำงานที่ตำแหน่งที่ถูกขัดจังหวะ

7.6.2 ศึกษาเรื่องการเรียกใช้งานรีจิสเตอร์เวลาของโปรเซสเซอร์ของแพลตฟอร์มคอม86

ไทมเมอร์ (Timer) ในโปรเซสเซอร์ AM186CC มีลักษณะที่เราต้องทราบคือ

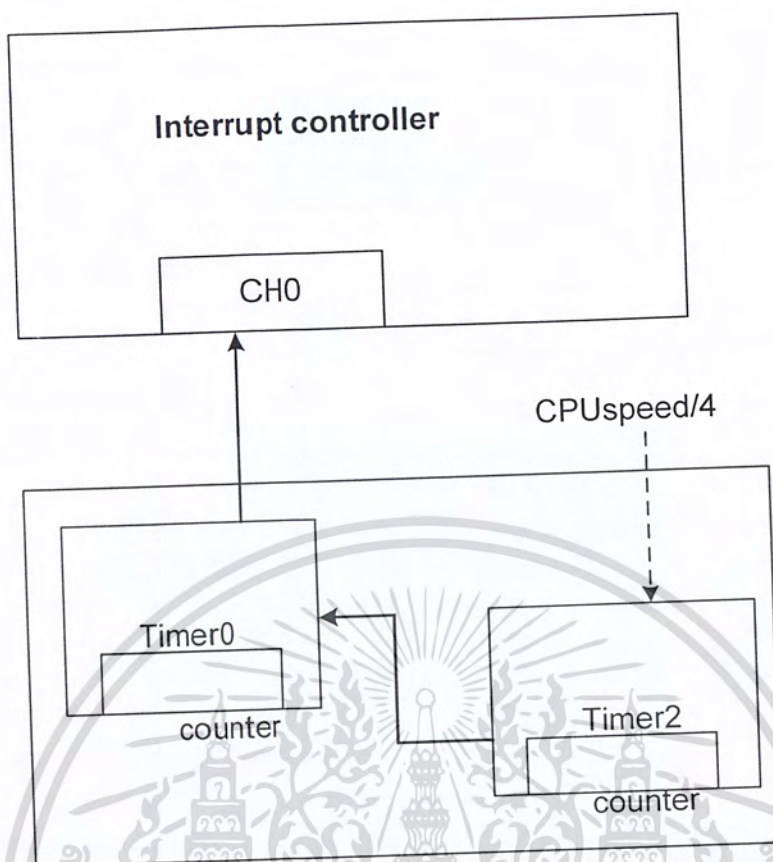
1. มีไทมเมอร์ทั้งหมดสามตัวคือ ไทมเมอร์ศูนย์ ไทมเมอร์หนึ่งและไทมเมอร์สอง ทั้งหมดมีขนาด 16 บิต
2. ไทมเมอร์สองสามารถใช้ร่วมกับไทมเมอร์ศูนย์หรือหนึ่ง เพื่อให้มีขนาดรวมไป 32 บิตได้
3. อินพุตของไทมเมอร์สองคือความเร็วของโปรเซสเซอร์หารด้วยสี่
4. อินพุตของไทมเมอร์ศูนย์หรือหนึ่งเป็นอินพุตที่สามารถกำหนดได้จากผู้ใช้งาน

มีไดอะแกรมการทำงานดังรูป



รูปที่ 35 ไดอะแกรมของไทมเมอร์

แต่เพื่อให้่ายในการทำความเข้าใจ จะใช้รูป



รูปที่ 36 การใช้งานไทมเมอร์บนแพลตฟอร์มคอมพิวเตอร์คอม86

เนื่องจากเราต้องการไทมเมอร์ตัวนับที่มีขนาด 32 บิต ดังนั้นในการใช้งานเราจะใช้ไทมเมอร์สองกับไทมเมอร์ศูนย์ร่วมกันเพื่อเป็นตัวนับขนาด 32 บิต

7.6.3 ศึกษาเรื่องระบบอินเทอร์รัปต์ในระบบปฏิบัติการมินิกซ์และทำการแก้ไขและปรับปรุง พิจารณาในไฟล์ 0.2.1/src/kernel/i8259.c เพื่อทำการศึกษาระบบอินเทอร์รัปต์ในมินิกซ์

```
PRIVATE vecaddr_t int_vec[] = {
    int00, int01, int02, int03, int04, int05, int06, int07,
};

PRIVATE vecaddr_t irq_vec[] = {
    hwint00, hwint01, hwint02, hwint03, hwint04, hwint05,
    hwint06, hwint07, hwint08, hwint09, hwint10, hwint11, hwint12,
    hwint13, hwint14, hwint15,
};
```

โครงสร้าง int_vec คืออินเทอร์รัปต์เวกเตอร์สำหรับอินเทอร์รัปต์หมายเลขศูนย์ถึงหกในระบบ ซึ่งเป็นอินเทอร์รัปต์ที่ไม่สามารถปิดได้ ส่วนโครงสร้าง irq_vec คืออินเทอร์รัปต์ที่สอดคล้องกับ IRQ นั้นของระบบ

พิจารณาฟังก์ชัน intr_init มีการทำงานดังนี้

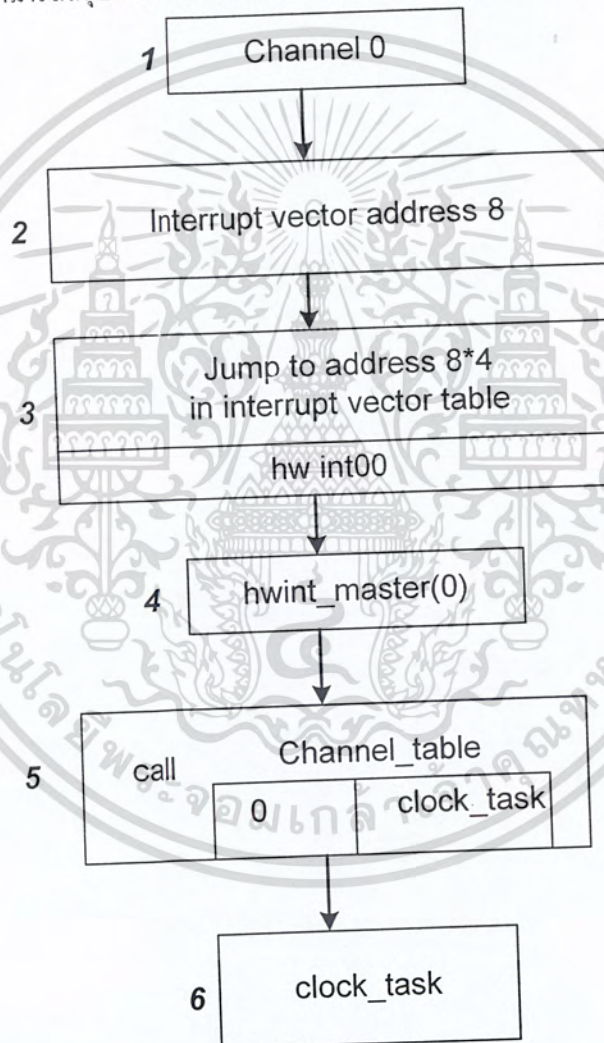
```

for (i = 0; i < 8; i++) set_vec(i, int_vec[i]);
set_vec(SYS_VECTOR, s_call);
/* Initialize the table of interrupt handlers. */
for (i = 0; i < NR_IRQ_VECTORS; i++)
    irq_table[i] = spurious_irq;

```

เห็นได้ว่าเป็นการตั้งค่าอินเทอร์รัปต์เวกเตอร์ให้มาชี้ในอินเทอร์รัปต์ของมินิกซ์ สำหรับอินเทอร์รัปต์สำหรับ IRQ ของระบบนั้นจะมีตารางอีกหนึ่งตารางใช้สำหรับอ้างอิงคือ irq_table โดยค่าเริ่มต้นจะชี้ไปที่ spurious_irq ทุกๆ IRQ

จากตรงนี้เราสามารถสรุปลำดับการเกิดของอินเทอร์รัปต์ของมินิกซ์ได้เป็น



รูปที่ 37 แสดงลำดับการเกิดอินเทอร์รัปต์ในระบบมินิกซ์

จากรูปสมมติว่า เกิด IRQ0 ระบบก็จะจัดการหาอินเทอร์รัปต์เวกเตอร์ซึ่งจะชี้มาที่รูทีน hwint00 ของมินิกซ์ ซึ่งในรูทีนนี้จะเรียกชี้ไปยังตาราง irq_table เพื่อหาตำแหน่งของเซอวิสิวูทีนของ IRQ0 นี้ ซึ่งจะชี้ไปยัง clock_handler จากนั้นจึงกระโดดไปทำงานได้อย่างถูกต้อง

เราจะเริ่มลงมือเปลี่ยนแล้ว เราพบว่ามียาการที่ต้องเปลี่ยนดังนี้

- put_irq_handler เปลี่ยนเป็น put_channel_handler เนื่องจากใน AM186CC จะใช้คำแทนคำว่า IRQ ว่า Channel และมีรูปแบบการเรียกดังนี้ put_channel_handler(vector, ch, handler)
- เปลี่ยน set_vec ใน i8259.c เป็น set_vec(vector, ch_vec[ch])
- เปลี่ยนชื่อ spurious_irq เป็น spurious_channel ให้สอดคล้องกัน
- เปลี่ยน NR_IRQ_VECTORS เป็น NR_CHANNEL_VECTORS

และเปลี่ยนฟังก์ชันต่างๆ ในไฟล์ i8259.c ได้เป็น เริ่มที่ intr_init

```
PUBLIC void intr_init(mine)
int mine;
{
    int i;
    lock();
    /* Use the BIOS interrupt vectors in real mode. */
    for (i = 0; i < NR_EXCEPTION; i++) set_vec(i, exception_vec[i]);
    set_vec(SYS_VECTOR, s_call);

    /* Initialize the table of interrupt handlers. */
    for (i = 0; i < NR_CHANNEL_VECTORS; i++)
        channel_table[i] = spurious_channel;
}
```

ต่อมา spurious_channel

```
PRIVATE int spurious_channel(ch)
int ch;
{
    /* Default interrupt handler. It complains a lot. */

    if (ch < 0 || ch >= NR_CHANNEL_VECTORS)
        panic("invalid call to spurious_channel", ch);

    printf("spurious channel %d\n", ch);

    return 1; /* Reenable interrupt */
}
```

สุดท้าย put_channel_handler

```
PUBLIC void put_channel_handler(vec_nr, ch, handler)
int vec_nr;
int ch;
irq_handler_t handler;
{
    /* Register an interrupt handler. */

    if (ch < 0 || ch >= NR_CHANNEL_VECTORS)
        panic("invalid call to put_channel_handler", ch);
```

```

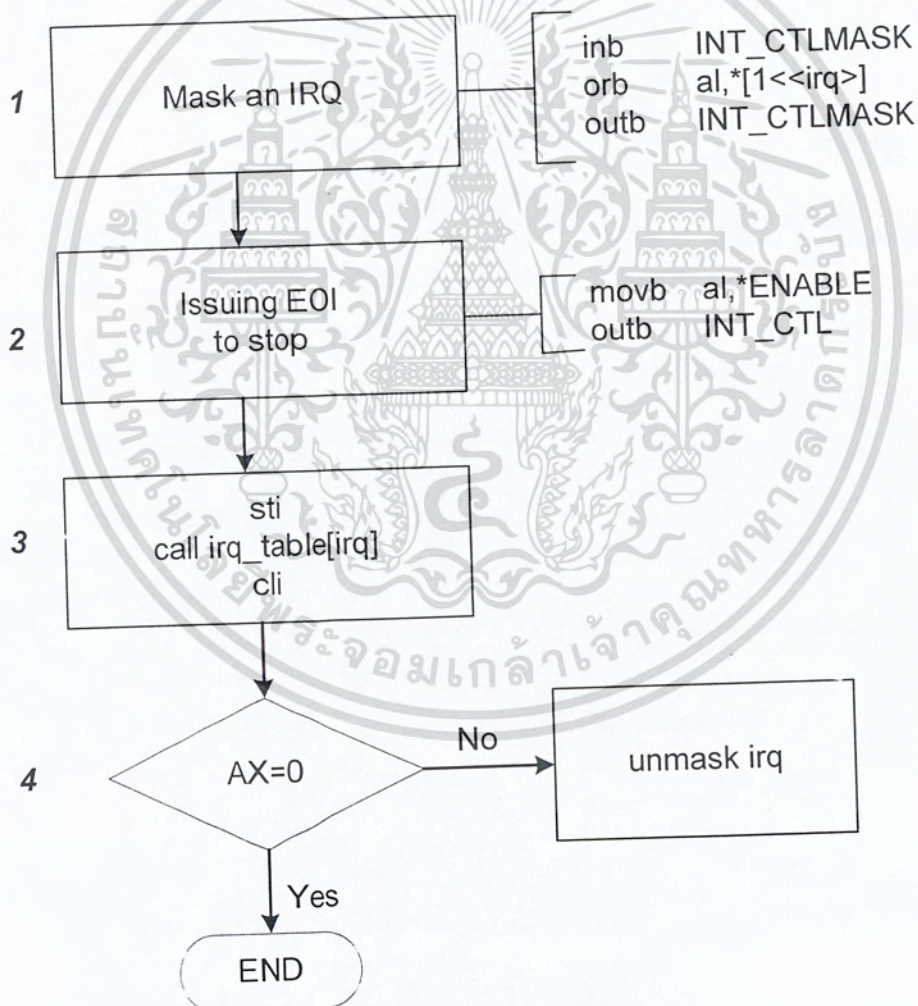
if (channel_table[ch] == handler)
    return; /* extra initialization */

if (channel_table[ch] != spurious_channel)
    panic("attempt to register second channel handler for channel",
ch);

disable_channel(ch);
set_vec(vec_nr, channel_vec[ch]);
channel_table[ch] = handler;
channel_use |= 1 << ch;
}

```

ต่อมาพิจารณา rutin ที่เกี่ยวข้องกับอินเทอร์รัปต์ ซึ่งอยู่ชื่อว่า hwint_master อยู่ในไฟล์ mp88.s มีลำดับการทำงานดังรูป



รูปที่ 38 ลำดับการทำงานในการบริการอินเทอร์รัปต์ของมินิซ

เราเปลี่ยนจาก hwint_master และ hwint_slave เป็น hwint ได้เป็น

```
#define hwint(ch) \
```



```

call    save                /* save interrupted process state */;\
cli                                           ;\
inw     IMASK                ;\
or      ax, *[1<<ch]        ;\
outw    IMASK                /* disable the channel */;\
sti                                           /* enable interrupts */;\
mov     ax, *ch              ;\
push    ax                  /* channel */;\
call    @_channel_table + 2*ch ;\
pop     cx                  ;\
cli                                           /* disable interrupts */;\
test    ax, ax              /* need to reenale channel? */;\
jz      0f                  ;\
inw     IMASK                ;\
and     ax, *~[1<<ch]       ;\
outw    IMASK                /* enable the channel */;\
0:      ret                  /* restart (another) process */

```

```

_hwint00:
    hwint(0)

```

```

_hwint01:
    hwint(1)
...

```

พิจารณาอีกสอง routine คือ enable_channel กับ disble_channel ในไฟล์ klib88.s แก้ไขได้เป็น

```

_enable_channel:
    mov     bx, sp
    mov     cx, 2(bx)        ! channel
    pushf
    cli
    mov     ax, #1
    rol     ax, cl
    not     ax
    mov     bx, ax
    inw     IMASK
    and     ax, bx
    outw    IMASK
    popf
    ret

_disable_channel:
    mov     bx, sp
    mov     cx, 2(bx)        ! ch
    pushf
    cli
    mov     ax, #1
    rol     ax, cl
    mov     bx, ax           ! save ax
    inw     IMASK
    test    ax, bx
    jnz     dis_already     ! already disabled?
    or      ax, bx
    outw    IMASK
    popf
    mov     ax, #1           ! disabled by this function

```

```

ret
dis_already:
    popf
    xor    ax, ax
    ret
! already disabled
ret

```

เป็นอันเสร็จสิ้น

7.6.4 ศึกษาวิธีใช้ไครเวอร์เวลาของระบบปฏิบัติการมินิกซ์ เพื่อแก้ไขและปรับปรุง

พิจารณาค่าคงที่ที่กำหนดค่าคงที่เวลาของมินิกซ์ในไฟล์ clock.c พบว่า

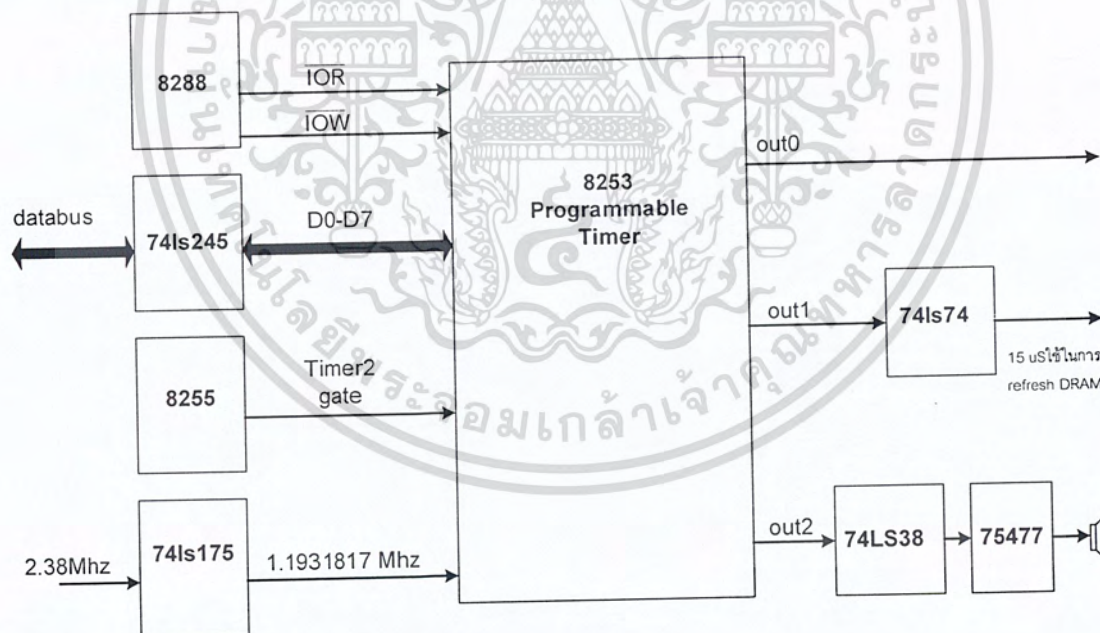
```

#define COUNTER_FREQ (2*TIMER_FREQ)
#define LATCH_COUNT 0x00
#define SQUARE_WAVE 0x36
#define TIMER_COUNT ((unsigned) (TIMER_FREQ/HZ))
#define TIMER_FREQ 1193182L
#define HZ 60

```

สังเกตว่าค่า TIMER_FREQ มีที่มาเนื่องจากข้อกำหนดทางฮาร์ดแวร์ของไอบีเอ็มพีซีคอมพิวเตอร์

เปิดผังรูป



รูปที่ 39 แสดงระบบจัดการเวลาในไอบีเอ็มพีซีคอมพิวเตอร์

เมื่อทำความเข้าใจแล้วก็มาถึงขั้นตอนการเปลี่ยนแปลง
เริ่มต้นเพิ่มค่าคงที่ไปดังนี้ในคอนฟิกไฟล์ดังนี้


```

/* Timer for AM186CC */
#define HZ 60 /* clock freq, software setable on minix */
#define CPUSPEED 2000000L /* CPU Speed */
#define TIMER2INPUT (CPUSPEED/4)
#define TIMER2DIV 20833 /* use in clock.c */
#define TIMER0INPUT ((unsigned) (TIMER2INPUT/TIMER2DIV))
#define TIMER0DIV ((unsigned) (TIMER0INPUT/HZ))

```

เพื่อกำหนดค่าคงที่เวลา

จากนั้นปรับปรุง init_clock พิจารณาโค้ดเดิมเป็น

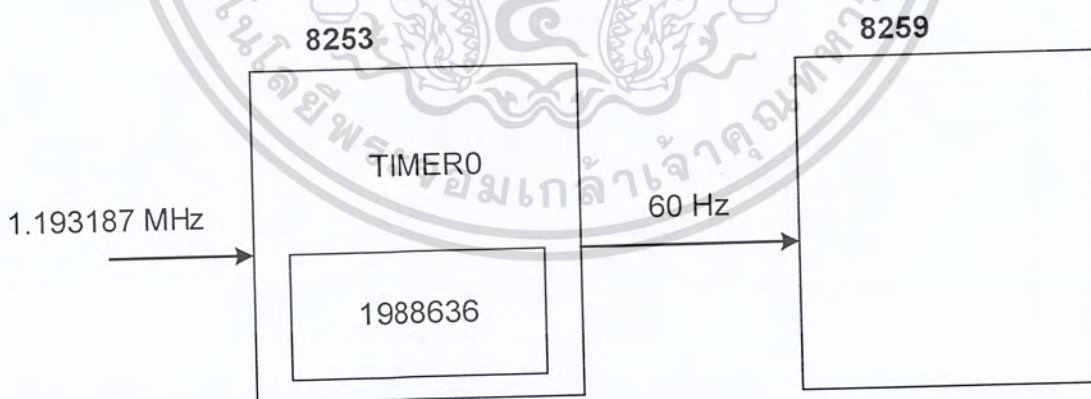
```

PRIVATE void init_clock()
{
/* Initialize channel 0 of the 8253A timer to e.g. 60 Hz. */

    out_byte(TIMER_MODE, SQUARE_WAVE); /* set timer to run
continuously */
    out_byte(TIMER0, TIMER_COUNT); /* load timer low byte */
    out_byte(TIMER0, TIMER_COUNT >> 8); /* load timer high byte */
    put_irq_handler(CLOCK_IRQ, clock_handler); /* set the interrupt
handler */
    enable_irq(CLOCK_IRQ); /* ready for clock interrupts */
}

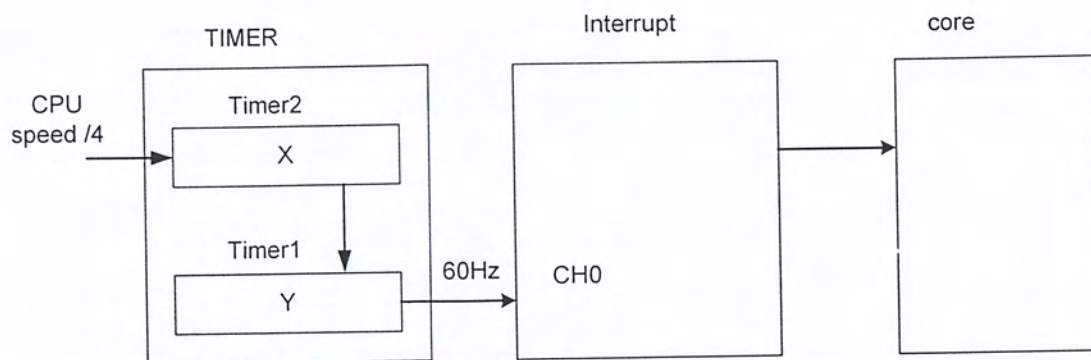
```

ซึ่งตัวแปร $TIMER_COUNT = TIMER_FREQ/HZ$ นั่นก็คือ $1.193187\text{ MHz}/60\text{ Hz} = 19886.36$
ซึ่งเป็นวิธีการจัดการเวลาในไอพีเอ็มพีซีคอมพิวเตอร์แบบดังรูป



รูปที่ 40 แสดงค่าคงที่ที่ใช้ในการจัดการเวลาในไอพีเอ็มพีซีคอมพิวเตอร์แบบดังรูป

จากรูปจะเห็นว่า ถ้าต้องการให้เอาต์พุตออกมา 60 ครั้งต่อวินาที ค่าที่เก็บในไทเมอร์ศูนย์ยังสามารถเก็บค่าได้ เพราะค่า $TIMER_COUNT$ มีค่าไม่เกิน 16 บิต
แต่พิจารณาการตั้งค่าตัวหารเวลาในแพลตฟอร์มคอมพิวเตอร์ 86 บ้าง จะเห็นได้ว่า



รูปที่ 41 การจัดการเวลาในแพลตฟอร์มคอม86 โดยใช้มินิซ

เนื่องจากอินพุตเมื่อหารแล้วไม่สามารถเก็บได้ในตัวแปรแบบ 16 บิตได้ ดังนั้นการใช้ค่าเวลาในโปรเซสเซอร์ AM186CC จึงต้องใช้ไทมเมอร์สองตัวช่วยกันเพื่อให้มีตัวนับมีขนาดมากกว่า 16 บิตหรือเป็น 32 บิตซึ่งพอเพียงต่อการใช้งานแล้ว

พิจารณารีจิสเตอร์ที่เกี่ยวข้องทั้งหมดที่ต้องใช้

Timer 0 Mode/Control (T0CON)

Offset 0340h

1514131211109876543210																
Bit Name	EN	INT	INT	RIU	Res						MC	RTG	P	EXT	ALT	CONT
Software Read/Write	R/W	W	R/W	R	R						R/W0	R/W	R/W	R/W	R/W	R/W
Hardware Set/Clear	C	—	—	S/C	—						S	—	—	—	—	—
Chip Reset Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

รูปที่ 42 รีจิสเตอร์ T0CON

Timer 2 Mode/Control (T2CON)

Offset 0350h

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit Name	EN	FIH	INT	Res							MC	Res				CONT
Software Read/Write	R/W	W	R/W	R							R/W0	R				R/W
Hardware Set/Clear	C	—	—	—							S	—				—
Chip Reset Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

รูปที่ 43 รีจิสเตอร์ T2CON

Channel 0 Control (CH0CON)**Offset 300h**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit Name	Res												MSK	PR		
Software Read/Write	R												R/W	R/W		
Hardware Set/Clear	—												—	—		
Chip Reset Default	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

รูปที่ 44 รีจิสเตอร์ CH0CON

Interrupt Mask (IMASK)**Offset 326h**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit Name	Res	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
Software Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Hardware Set/Clear	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Chip Reset Default	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

รูปที่ 45 รีจิสเตอร์ IMASK

จากรีจิสเตอร์ทั้งหมดและวิธีการใช้งานรีจิสเตอร์และแนวความคิดในการแก้ไข แก้ไขได้ทั้งหมด
ดังนี้

เริ่มที่ init_clock แก้ได้เป็น

```
PRIVATE void init_clock()
{
    out_word(T2CNT, 0);
    out_word(T2CMPA, TIMER2DIV);
    out_word(T2CON, (TCON_EN | TCON_INH | TCON_CONT));
    out_word(T0CNT, 0);
    out_word(T0CMPA, TIMER0DIV);
    out_word(T0CON, (TCON_P | TCON_CONT));
    out_word(CH0CON, (CHCON_MASK | CHCON_PR7));
    put_channel_handler(ITYPE_TMR0, TIMER_CH, clock_handler);
    out_word(EOI, ITYPE_TMR0);
    out_word(T0CON, (TCON_EN | TCON_INH | TCON_INT));
    enable_channel(TIMER_CH);
}
```

clock_stop แก้ได้เป็น

```
PUBLIC void clock_stop()
{
```

```

/* Reset the clock to the BIOS rate. (For rebooting) */

    out_word(T2CNT, 0);
    out_word(T0CNT, 0);
}

```

ส่วน clock_handler เพิ่มโค้ดตัวหนาเข้าไปได้เป็น

```

...
    (rdy_head[USER_Q] != NIL_PROC || rdy_head[SHADOW_Q] != NIL_PROC) {
#endif
    interrupt(CLOCK);
    out_word(EOI, ITYPE_TMR0);
    return 1;    /* Reenable interrupts */
}

if (--sched_ticks == 0) {
    sched_ticks = SCHED_RATE;    /* reset quantum */
    prev_ptr = bill_ptr;    /* new previous process */
}
out_word(EOI, ITYPE_TMR0);
return 1;    /* Reenable clock interrupt */

```

และใน milli_elapsed

```

...
count = in_word(T2CNT);

msp->accum_count += count <= msp->prev_count ? (count - msp-
>prev_count) : 1;
msp->prev_count = count;

return msp->accum_count / (TIMER2INPUT / 1000);

```

สุดท้ายแก้ไขค่าเริ่มต้นในไฟล์ console.c เนื่องจากไม่ได้สนับสนุนโดยไบออสของคอม86 ดังนั้น
ต้องแก้ไขให้เริ่มต้นค่าในฟังก์ชัน scr_init

```

scr_width = 80;
font_lines = 14;
scr_lines = 25;

vid_base = MONO_BASE;
vid_size = MONO_SIZE;
wrap = !ega;

vid_seg = physb_to_hclick(vid_base);
init_dataseg(&gdt[VIDEO_INDEX], vid_base, (phys_bytes) vid_size,
TASK_PRIVILEGE);
vid_size >>= 1;    /* word count */
vid_mask = vid_size - 1;

```


มินิซก็สามารถเริ่มต้นระบบได้อย่างถูกต้องบนคอม86

7.7 มินิซบนคอม86 เวอร์ชัน 0.1.4 และเวอร์ชัน 0.2.2

ข้อดีหนึ่งของการพัฒนาที่เป็นลำดับขั้นตอนในลักษณะที่นำมาใช้กับการพัฒนามินิซบนคอม86 คือการตรวจสอบถึงการบรรลุวัตถุประสงค์ในแต่ละขั้นตอนในการพัฒนา ทำให้รู้ถึงจุดที่กำลังยืนอยู่ได้เสมอ อันทำให้เราตระหนักได้เป็นอย่างดีว่า ขณะนี้ยังเหลืองานที่ต้องทำอีกมากน้อยเท่าใด

ในเวอร์ชัน 0.2.1 นั้นถือว่าเป็นเวอร์ชันหลักในการพัฒนามินิซบนคอม86 เลขที่ว่าได้ เนื่องจากเป็นการแก้ไขส่วนฐานทั้งหมดของระบบปฏิบัติการมินิซให้สามารถทำงานได้บนแพลตฟอร์มคอม86 จากความหมายของคำว่า “ทำงานได้” ที่เราได้ให้คำนิยามเชิงปฏิบัติไว้ตั้งแต่ต้น ขณะนี้เราสามารถบรรลุถึงคำว่าทำงานได้ถึงสองข้อด้วยกันแล้วคือ

- สามารถเริ่มต้นระบบได้อย่างถูกต้องและสมบูรณ์บนแพลตฟอร์มคอม86
- สามารถแสดงผลผ่านโปรแกรมเทอร์มินอลได้อย่างถูกต้อง

ทั้งสองจุดที่ผ่านมาได้นั้นถือว่าเป็นจุดหลักของการบรรลุวัตถุประสงค์ของคำว่า “ทำงานได้” และเป็นส่วนสำคัญที่เป็นรากฐานให้การพัฒนาในลำดับถัดไปมีความมั่นใจ มีส่วนช่วยให้การพัฒนาทำได้อย่างรวดเร็วมากขึ้น ถ้าเทียบเป็นเปอร์เซ็นต์ของการพัฒนาอาจนับได้ว่าเราได้เดินมาถึงครึ่งทางของการพัฒนามินิซบนคอม86 ก็ไม่ได้ผิดไปจากความจริงเลยแม้แต่น้อย

เวอร์ชันของการพัฒนาดังแต่เวอร์ชันนี้เป็นต้นไป จะมุ่งเน้นพัฒนาในส่วนของดีไวซ์ใดเวอร์ต่างๆ ของระบบปฏิบัติการมินิซ เพื่อให้สามารถใช้ดีไวซ์ใดเวอร์นั้น ได้อย่างสมบูรณ์บนระบบปฏิบัติการรุ่นลายครามตัวนี้ มินิซ

จากการออกแบบให้มีสายของการพัฒนาแบ่งออกได้เป็นสองสายจากที่ได้กล่าวไปแล้ว ในเวอร์ชันนี้ จะกล่าวรวมไปพร้อมๆ กันทั้งสองเวอร์ชันคือเวอร์ชัน 0.1.4 ที่ยังคงทำงานได้บนแพลตฟอร์มไอบีเอ็มพีซีคอมแพคทีเบิล และเวอร์ชัน 0.2.2 ซึ่งเป็นมินิซบนคอม86 เหตุผลที่กล่าวรวมกันได้ก็เพราะเป็นการพัฒนาดีไวซ์ใดเวอร์เดียวกัน แตกต่างกันตรงลักษณะการนำไปใช้เท่านั้น ถ้าเป็นเวอร์ชัน 0.1.4 จะแก้ไขให้สามารถอ่านฟลอปปี้ดิสก์ได้โดยใช้ดีไวซ์ใดเวอร์นี้ ส่วนเวอร์ชัน 0.2.2 นั้นก็สามารถอ่านแฟลชดิสก์ได้ตามที่ต้องการ

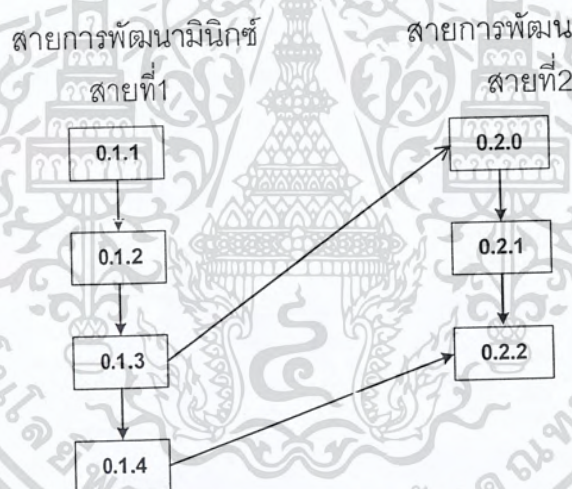
ดีไวซ์ใดเวอร์สำหรับดิสก์ที่เลือกคือ bios_wini.c เหตุผลที่เลือกเนื่องมาจากเป็นดีไวซ์ใดเวอร์สำหรับติดต่อกับดิสก์เพียงตัวเดียวที่ใช้บริการดิสก์ของไบออส จากที่ได้กล่าวไปแล้วว่าไบออสบนคอม86 ได้ออกแบบให้ใกล้เคียงกับไบออสบนไอบีเอ็มพีซีคอมแพคทีเบิลมากที่สุด บริการไบออสในเรื่องของดิสก์ก็เช่นเดียวกับบริการทางจอภาพและคีย์บอร์ดของไบออสบนคอม86 ที่ได้ทำการออกแบบให้เหมาะสมกับแพลตฟอร์มคอม86 บริการติดต่อกับดิสก์ของไบออสของคอม86 ก็เช่นเดียวกันได้ทำการออกแบบให้สามารถอ่านและเขียนกับแฟลชดิสก์บนแพลตฟอร์มคอม86 ได้ นั่นหมายความว่าถ้าเราสามารถเรียกใช้บริการดิสก์ผ่านบริการไบออสของคอม86 ได้ การอ่านเขียนข้อมูลกับแฟลชดิสก์ก็สามารถทำได้เช่นเดียวกับการอ่านเขียนฟลอปปี้ดิสก์ได้เหมือนกัน

มีจุดประสงค์ในการพัฒนาเวอร์ชัน 0.1.4 และเวอร์ชัน 0.2.2 คือ

เวอร์ชันในการพัฒนา	จุดประสงค์ในการพัฒนา
0.1.4	ปรับปรุงและเปลี่ยนแปลงดีไวซ์ไดรเวอร์ในส่วนการอ่านเขียนข้อมูลกับดิสก์ให้มาใช้บริการของไบออส
0.2.2	ปรับปรุงและเปลี่ยนแปลงดีไวซ์ไดรเวอร์ในส่วนการอ่านเขียนข้อมูลกับแฟลชดิสก์ในแพลตฟอร์มคอม86 เพื่อให้อ่านแฟลชดิสก์ได้

ตาราง 13 แสดงจุดประสงค์ในการพัฒนามินิซ์เวอร์ชัน 0.1.4 และเวอร์ชัน 0.2.2

โดยมีลำดับในการพัฒนาเริ่มจากการพัฒนาที่เวอร์ชัน 0.1.4 ให้สามารถใช้ดีไวซ์ไดรเวอร์ bios_wini.c สามารถอ่านเขียนกับฟลอปปีดิสก์ได้ หลังจากนั้นจึงนำวิธีการแก้ไขมาแก้ไขในเวอร์ชัน 0.2.2 เพื่อให้สามารถอ่านเขียนกับแฟลชดิสก์ได้ต่อไป มีลำดับในการพัฒนาดังรูป



รูปที่ 46 แสดงลำดับในการพัฒนามินิซ์เวอร์ชัน 0.1.4 และมินิซ์เวอร์ชัน 0.2.2

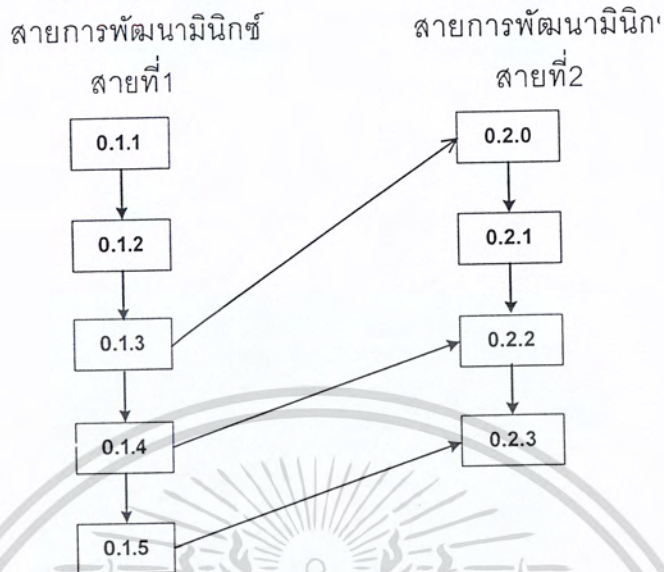
7.8 มินิซ์บนคอม86 เวอร์ชัน 0.1.5 และ 0.2.3

ในเวอร์ชัน 0.1.5 และ 0.23 มีจุดประสงค์ในการพัฒนาคือ

เวอร์ชันในการพัฒนา	จุดประสงค์ในการพัฒนา
0.1.5 และ 0.2.3	ปรับปรุงและเปลี่ยนแปลงเรื่องการรับข้อมูลเข้า โดยใช้บริการไบออส

ตาราง 14 แสดงจุดประสงค์ในการพัฒนามินิซ์เวอร์ชัน 0.1.5 และ 0.2.3

ลำดับในการพัฒนาเป็นไปตามลำดับในสายของการพัฒนาจากเวอร์ชัน 0.1.4 ลงมาเป็น 0.1.5 และจากเวอร์ชัน 0.2.2 ลงมาเป็น 0.2.3



รูปที่ 47 แสดงลำดับในการพัฒนามินิ็กซ์เวอร์ชัน 0.1.5 และ 0.2.3

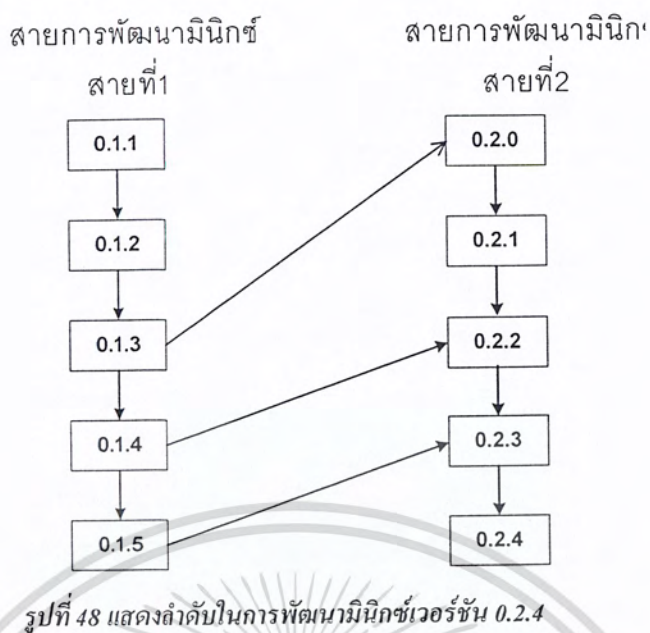
7.9 มินิ็กซ์บนคอม86 เวอร์ชัน 0.2.4

ในเวอร์ชัน 0.2.4 มีจุดประสงค์ในการพัฒนาคือ

เวอร์ชันในการพัฒนา	จุดประสงค์ในการพัฒนา
0.2.4	ปรับปรุงและเปลี่ยนแปลงดีไวส์ไดรเวอร์ของฮาร์ดแวร์เน็ต

ตาราง 15 แสดงจุดประสงค์ในการพัฒนามินิ็กซ์เวอร์ชัน 0.2.4

ลำดับในการพัฒนาเป็นไปตามลำดับในสายของการพัฒนาจากเวอร์ชัน 0.2.3 ลงมาเป็น 0.2.4



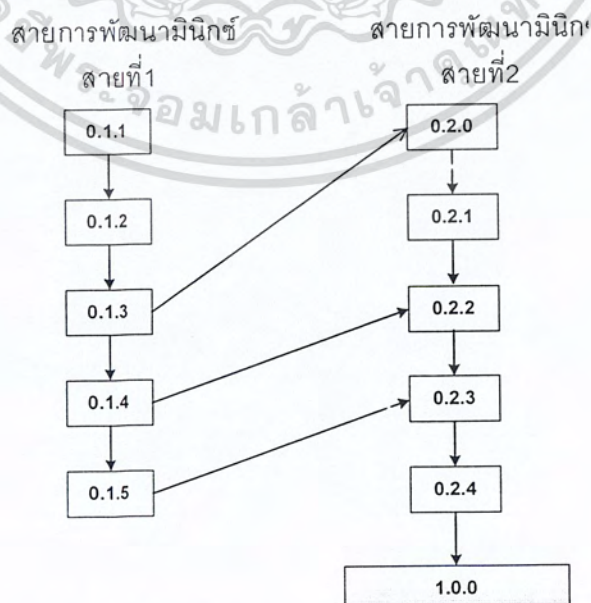
7.10 มินิฟิกซ์บนคอม86 เวอร์ชัน 1.0.0

ในเวอร์ชัน 1.0.0 มีจุดประสงค์ในการพัฒนาคือ

เวอร์ชันในการพัฒนา	จุดประสงค์ในการพัฒนา
1.0.0	มินิฟิกซ์บนคอม86 ที่สามารถ "ทำงาน" ได้อย่างสมบูรณ์

ตาราง 16 แสดงจุดประสงค์ในการพัฒนามินิฟิกซ์เวอร์ชัน 1.0.0

ลำดับในการพัฒนาเป็นไปตามลำดับในสายของการพัฒนาจากเวอร์ชัน 0.2.4 ลงมาเป็น 1.0.0



7.11 สรุปผลการพัฒนาแต่ละเวอร์ชัน

รายละเอียดแต่ละเวอร์ชันมีการพัฒนาต่าง ๆ ดังนี้

มินิغبนคอม86 เวอร์ชัน 0.1.1 ตัดดีไวส์ใดเวอร์ที่ไม่จำเป็นออก และลบไฟล์ที่ไม่ได้ใช้ เพื่อลดขนาดของเครื่องในเบื้องต้น และลดโค้ดในส่วนที่ติดต่อกับฮาร์ดแวร์โดยตรงลงให้เหลือน้อยที่สุด เพื่อไม่ให้เกิดปัญหาในภายหลัง

มินิغبนคอม86 เวอร์ชัน 0.1.2 ได้ทำการปรับปรุงและเปลี่ยนแปลงเรื่องการแสดงผลให้ใช้บริการของไบออสเซอรัวีสรูทินหมายเลข 10 ฐานสิบหก

มินิغبนคอม86 เวอร์ชัน 0.1.3 ปิดการทำงานของคีย์บอร์ดดีไวส์ใดเวอร์ มุ่งเน้นไปที่การพัฒนาในส่วนที่เกี่ยวข้องกับอินเทอร์พรีตบนแพลตฟอร์มคอม86 ก่อน

มินิغبนคอม86 เวอร์ชัน 0.1.4 ทำการปรับปรุงและเปลี่ยนแปลงดีไวส์ใดเวอร์ในส่วนการอ่านเขียนข้อมูลกับดิสก์ให้มาใช้บริการของไบออสเซอรัวีสรูทินหมายเลข 13 ฐานสิบหกเพื่อให้สามารถอ่านเขียนกับฟลอปปีดิสก์ได้

มินิغبนคอม86 เวอร์ชัน 0.1.5 ทำการปรับปรุงและเปลี่ยนแปลงเรื่องการรับข้อมูลเข้า โดยใช้บริการไบออสเซอรัวีสรูทินหมายเลข 16 ฐานสิบหกเป็นบริการรับข้อมูลผ่านทางคีย์บอร์ด

มินิغبนคอม86 เวอร์ชัน 0.2.1 ปรับปรุงและแก้ไขส่วนที่เกี่ยวข้องกับระบบอินเทอร์พรีตของระบบปฏิบัติการมินิغبให้สามารถทำงานได้บนแพลตฟอร์มคอม86 และทำการปรับปรุงและเปลี่ยนแปลงดีไวส์ใดเวอร์ของเวลาของระบบปฏิบัติการมินิغبให้ทำงานได้สอดคล้องกับแพลตฟอร์มคอม86

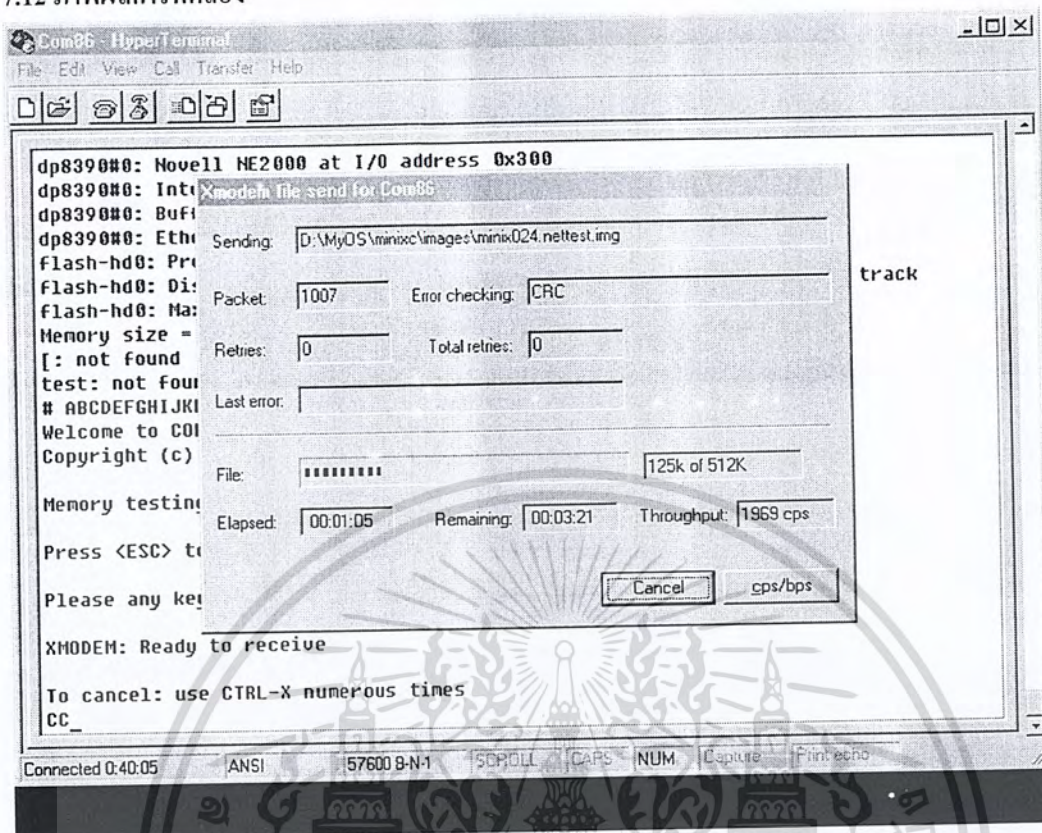
มินิغبนคอม86 เวอร์ชัน 0.2.2 ปรับปรุงและเปลี่ยนแปลงดีไวส์ใดเวอร์ในส่วนการอ่านเขียนข้อมูลกับแฟลชดิสก์ในแพลตฟอร์มคอม86 เพื่อให้อ่านแฟลชดิสก์ได้

มินิغبนคอม86 เวอร์ชัน 0.2.3 ปรับปรุงและเปลี่ยนแปลงเรื่องการรับข้อมูลเข้า โดยใช้บริการไบออสเซอรัวีสรูทินหมายเลข 16 ฐานสิบหกเป็นบริการรับข้อมูลผ่านทางคีย์บอร์ดผ่านโปรแกรมเทอร์มินอลได้

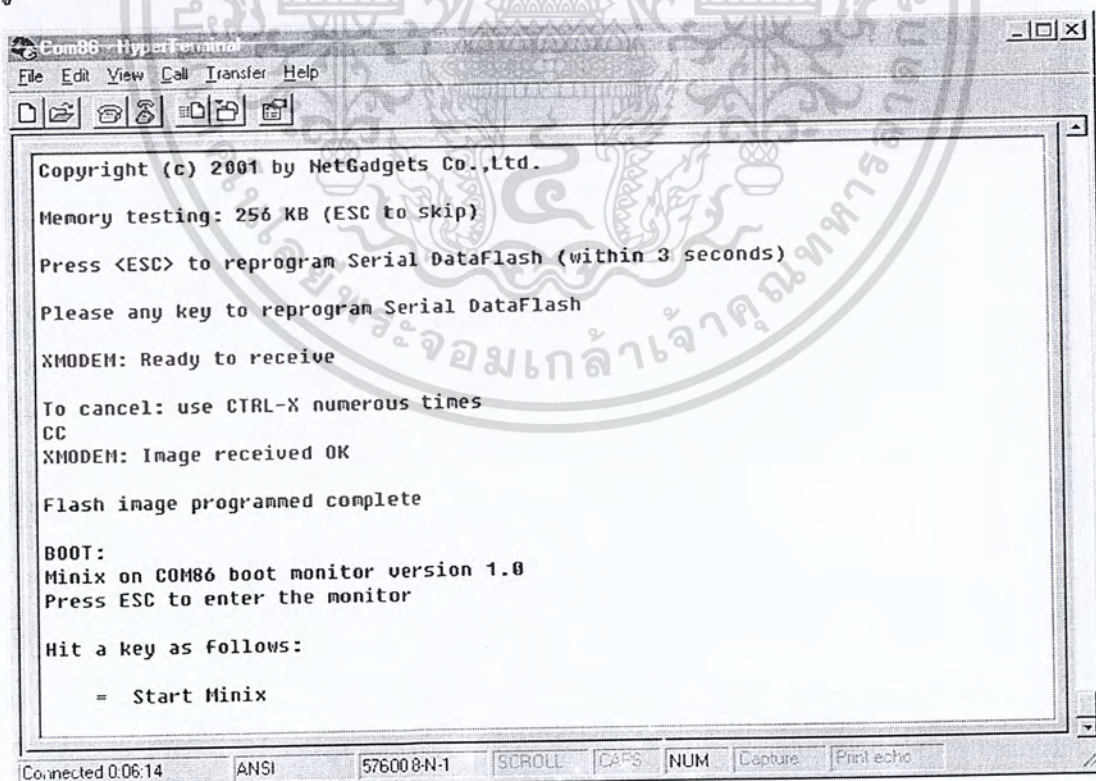
มินิغبนคอม86 เวอร์ชัน 0.2.4 ปรับปรุงและเปลี่ยนแปลงดีไวส์ใดเวอร์ของอีเทอร์เน็ตให้สามารถติดต่อกับระบบเครือข่ายคอมพิวเตอร์ได้

มินิغبนคอม86 เวอร์ชัน 1.0.0 มินิغبนคอม86 ที่สามารถ “ทำงาน” ได้อย่างสมบูรณ์

7.12 ภาพผลการทดลอง



รูปที่ 50 ขั้นตอนการถ่ายโอนอิมเมจของระบบปฏิบัติการมินิกซ์ลงคอม86



รูปที่ 51 มินิกซ์บูตมอนิเตอร์บนคอม86


```

Com86 - HyperTerminal
File Edit View Call Transfer Help

Hit a key as follows:

= Start Minix
[ESC]
Fd0>
Fd0>set
rootdev = (hd0)
ramdisksdev = (hd0)
ramsize = (0)
processor = (86)
bus = (xt)
video = (mda)
chrome = (mono)
memory = (800:9F800)
DPETH0 = (300:1)
memsize = (640)
enssize = (0)
image = (minix)
leader() echo \nMinix on COM86 boot monitor version \v\nPress ESC to enter the monitor
main() menu
trailer() echo \n
Fd0>_

Connected 0.36:14 ANSI 57600 8-N-1 SCROLL CAPS NUM Capture Print echo

```

รูปที่ 52 มินิกซ์บูตมอนิเตอร์หลังจากกดคีย์เอสเคปเพื่อเข้าสู่มินิกซ์บูตมอนิเตอร์

```

Com86 - HyperTerminal
File Edit View Call Transfer Help

ROOT:
Minix on COM86 boot monitor version 1.0
Press ESC to enter the monitor

Hit a key as follows:

= Start Minix

Loading Minix.

cs      ds      text    data     bss      stack
000800  00a000  38880   6204    20250    0 kernel
010800  013700  11904   1056    13024    512 mm
017000  01da00  27040   1890    52544   1024 fs
02b300  03aa00  63008   1274    41718   2048 inet
045a00  045a00  6154    1894    1090    384 init

Starting Minix on COM86

Minix on COM86 Release 0.2 Version 4
Minix on COM86 i_

Connected 0.37:25 ANSI 57600 8-N-1 SCROLL CAPS NUM Capture Print echo

```

รูปที่ 53 ขั้นตอนเมื่อทำการเริ่มต้นระบบปฏิบัติการมินิกซ์


```

Com86 - HyperTerminal
File Edit View Call Transfer Help

Loading Minix.

   cs      ds      text    data     bss      stack
000800  007200  26896   3600    14690      0  kernel
000a00  00e900  11904   1056   13024     512  mm
012200  018b00  26832   1898   52544    1024  fs
026400  026400    6154   1894    1090     384  init

Minix on COM86 Version 0.2.3
Minix on COM86 is based on Minix Version 2.0.2 Copyright 1998 Prentice-Hall,
Copyright (C) ESL and CATOS Team, 2001-2002. All right reserved.

Executing in real mode
flash_wini: Probe the disk
Flash-hd0: 80 cylinders, 2 heads, 18 sectors per track
Memory size = 638K  MINIX = 161K  RAM disk = 0K  Available = 478K
Starting standard daemons: update.

Welcome to Minix on COM86

Minix Release 2.0 Version 2

athena login: _

```

Connected 0:11:46 ANSI 57600 8-N-1 SCROLL CAPS NUM Capture Print echo

รูปที่ 54 มินิคซ์รุ่น ปรับปรุงและเปลี่ยนแปลงดีไวซ์ไดรเวอร์ในส่วนการอ่านเขียนข้อมูลกับแฟลชดิสก์

```

Com86 - HyperTerminal
File Edit View Call Transfer Help

Starting standard daemons: update.

Welcome to Minix on COM86

Minix Release 2.0 Version 2

athena login: root
# ls
.ellepro.b1 .exrc .profile
# cd /
# ls
bin boot dev etc minix mnt root tmp usr
# ls -l
total 118
drwxr-xr-x 2 bin operator 192 Jan 31 2002 bin
-rwxr-xr-x 1 root operator 27324 Jan 31 2002 boot
drwxr-xr-x 2 root operator 1568 Jan 30 2002 dev
drwxr-xr-x 2 root operator 272 Jan 29 2002 etc
-rw-r--r-- 1 root operator 84480 Jan 31 2002 minix
drwxr-xr-x 2 root operator 32 Jan 11 2002 mnt
dr-xr-xr-x 2 root operator 80 Dec 30 2001 root
drwxrwxrwx 2 root operator 32 Jan 10 2002 tmp
drwxr-xr-x 8 root operator 128 Jan 14 2002 usr
#

```

Connected 0:12:59 ANSI 57600 8-N-1 SCROLL CAPS NUM Capture Print echo

รูปที่ 55 มินิคซ์ เวอร์ชัน 0.2.3 ปรับปรุงเรื่องการรับข้อมูลเข้า โดยใช้บริการไบออสเซอร์วิสยูทิล


```

Com86 - Hyperterminal
File Edit View Call Transfer Help
[Icons]

02b300 03aa00 63008 1274 41718 2048 inet
045a00 045a00 6154 1894 1090 384 init

Starting Minix on COM86

Minix on COM86 Release 0.2 Version 4
Minix on COM86 is developed from Minix Release 2.0 Version 2
Copyright (C) 2001-2002 Embedded Systems Lab and CATOS. All right reserved.
kernel: Minix on COM86 is executing in real mode
dp8390#0: Finding for RTL8019AS Signature...0x5070
dp8390#0: Novell NE2000 compatible ethernet card operating in 8 bit mode
dp8390#0: Novell NE2000 at I/O address 0x300
dp8390#0: Interrupt Channel 1 and Memory size is 0x2000
dp8390#0: Buffer page address range from 0x4600 to 0x6000
dp8390#0: Ethernet address 0:20:18:61:81:34
Flash-hd0: Probing the flash disk
Flash-hd0: Disk geometry is 80 cylinders, 2 heads and 18 sectors per track
Flash-hd0: Maximum disk size support by BIOS is 1440 KBytes
Memory size = 638K MINIX = 286K RAM disk = 0K Available = 352K
IP Host Address is 192.168.1.101
Starting network daemons: .
Starting network services: .
# _

Connected 0:45:22 ANSI 57600 8-N-1 SCROLL CAPS NUM Capture Print echo

```

รูปที่ 56 มินิคซ์บนคอม86 เวอร์ชัน 0.2.4 ปรับปรุงและเปลี่ยนแปลงดีไวซ์ไดรเวอร์ของอีเทอร์เน็ต

ให้สามารถติดต่อกับระบบเครือข่ายคอมพิวเตอร์ได้ เห็นได้ว่ามินิคซ์สามารถหาดีไวซ์ไดรเวอร์สำหรับอีเทอร์เน็ตได้อย่างถูกต้อง

```

Com86 - Hyperterminal
File Edit View Call Transfer Help
[Icons]

dp8390#0: Novell NE2000 at I/O address 0x300
dp8390#0: Interrupt Channel 1 and Memory size is 0x2000
dp8390#0: Buffer page address range from 0x4600 to 0x6000
dp8390#0: Ethernet address 0:20:18:61:81:34
Flash-hd0: Probing the flash disk
Flash-hd0: Disk geometry is 80 cylinders, 2 heads and 18 sectors per track
Flash-hd0: Maximum disk size support by BIOS is 1440 KBytes
Memory size = 638K MINIX = 286K RAM disk = 0K Available = 352K
IP Host Address is 192.168.1.101
Starting network daemons: .
Starting network services: .
ping 192.168.1.1
192.168.1.1 is alive
ping 192.168.3.4
no answer from 192.168.3.4
ping 203.233.22.22
no answer from 203.233.22.22
ping 192.168.1.1
192.168.1.1 is alive
ping 192.168.1.2
write: Destination not reachable
ping 192.168.1.1
192.168.1.1 is alive
# _

Connected 0:20:27 ANSI 57600 8-N-1 SCROLL CAPS NUM Capture Print echo

```

รูปที่ 57 ทำการทดสอบโดยใช้โปรโตคอล ICMP ในการทดสอบใช้คำสั่ง ping


```

Com86 - HyperTerminal
File Edit View Call Transfer Help

fifoUnder :      0    fifoOver :      0    CDheartbeat:      0
OVC         :      0    dp_isr = 0x0 + 0x0, de_flags = 0x390
dp8390 port 1 is in sink mode

dp8390 statistics of port 0:
recvErr     :      0    sendErr      :      0    OUVW         :      0
CRCerr      :      0    frameAll   :      0    missedP      :      0
packetR     :    366    packetT    :    411    transDef     :    411
collision   :      0    transAb    :      0    carrSense    :      0
fifoUnder   :      0    fifoOver    :      0    CDheartbeat:      0
OVC         :      0    dp_isr = 0x0 + 0x0, de_flags = 0x390
dp8390 port 1 is in sink mode

#
dp8390 statistics of port 0:
recvErr     :      0    sendErr      :      0    OUVW         :      0
CRCerr      :      0    frameAll   :      0    missedP      :      0
packetR     :    381    packetT    :    426    transDef     :    426
collision   :      0    transAb    :      0    carrSense    :      0
fifoUnder   :      0    fifoOver    :      0    CDheartbeat:      0
OVC         :      0    dp_isr = 0x0 + 0x0, de_flags = 0x390
dp8390 port 1 is in sink mode

#

```

Connected 0:28:06 ANSI 57600 8N-1 SCROLL CUES NUM Capture Print Echo

รูปที่ 60 จำนวนแพ็กเก็ตที่ทำการรับเข้ามาเมื่อทำการ ping เข้ามา

จะเห็นว่ามินิคัมพลอม 86 สามารถทำงานได้บนระบบเครือข่ายคอมพิวเตอร์ได้เป็นอย่างดี

บทที่ 8

บทวิจารณ์และสรุปผล

8.1 สรุปผลงาน

จากการพัฒนาระบบปฏิบัติการสำหรับชุดพัฒนาคอม86สามารถสรุปผลงานได้ดังนี้

8.1.1 ได้มีการปรับปรุงระบบปฏิบัติการมินิกซ์จากเวอร์ชัน2.0.2 สำหรับ ไอบีเอ็มพีซีเพื่อนำมาใช้งานกับบอร์ด

คอม86โดยการตัดดีไวส์ไดรเวอร์ที่ไม่จำเป็นสำหรับการใช้งานบนบอร์ดคอม86ออก

8.1.2 ปรับปรุงในส่วนของ อีเธอร์เน็ตดีไวส์ไดรเวอร์ เพื่อให้สามารถทำงานกับชิพอีเธอร์เน็ตของบอร์ดคอม86

8.1.3 ปรับปรุงในส่วนของการสร้างฐานเวลาสำหรับใช้ในการทำงานของระบบปฏิบัติการ ให้มีความสัมพันธ์กับฮาร์ดแวร์ของบอร์ดคอม86

8.1.4 ปรับปรุงในส่วนของการติดต่อรับส่งอินพุต/เอาต์พุตของระบบปฏิบัติการ ให้ย้ายไปทำงานโดยผ่านทาง serial terminal แทนการติดต่อผ่านทางคีย์บอร์ดและหน้าจอ

8.1.5 ปรับปรุงในส่วนของการจัดการในเรื่องของอินเตอร์รัพต์ต่างๆจากซอฟต์แวร์และฮาร์ดแวร์ ให้สอดคล้องกับระบบอินเตอร์รัพต์ของไมโครโพรเซสเซอร์บนบอร์ดคอม86

8.1.6 พัฒนาระบบปฏิบัติการมินิกซ์ที่สามารถทำงานได้บนบอร์ดคอม86สำหรับใช้ในชุดพัฒนาอุปกรณ์ทางด้านระบบฝังตัว

8.1.7 พัฒนารูปแบบของการพัฒนาระบบปฏิบัติการสำหรับผู้พัฒนาหลายคน โดยใช้เวอร์ชันคอนโทรล

8.2 ประโยชน์จากโครงการ

8.2.1 ก่อให้เกิดองค์ความรู้ทางด้านการพัฒนาซอฟต์แวร์ระบบ วิธีการพัฒนาโปรแกรมติดต่อใช้งานฮาร์ดแวร์ในระดับลึก

8.2.2 ก่อให้เกิดองค์ความรู้ทางด้านระบบฝังตัว ทั้งในเรื่องของการนำไปประยุกต์ใช้งานและการพัฒนาอุปกรณ์อุปกรณ์ทางด้านระบบฝังตัว

8.2.3 ได้รับระบบปฏิบัติการสำหรับใช้ในชุดพัฒนาคอม86 เพื่อการนำไปพัฒนาอุปกรณ์ทางด้านระบบฝังตัว

8.2.4 เป็นการเพิ่มความสามารถของชุดพัฒนาคอม86 ในการทำให้นักพัฒนาสามารถพัฒนาซอฟต์แวร์สำหรับระบบฝังตัวที่มีความซับซ้อนได้สะดวกรวดเร็วยิ่งขึ้น

8.3 ปัญหาที่พบจากโครงการ

8.3.1 เนื่องจากทีมพัฒนาขาดประสบการณ์ในการพัฒนา ซอฟต์แวร์สำหรับระบบฝังตัว จึงส่งผลให้การพัฒนาระบบปฏิบัติการล่าช้ากว่าที่ได้วางแผนไว้ในโครงการพอสมควร

8.3.2 ความไม่เข้ากันของซอฟต์แวร์ที่คลาดเคลื่อนจากการที่ได้ประเมินไว้ในขั้นต้น เช่น อีเธอร์เน็ตดีไวส์ไดรเวอร์ ทำให้เสียเวลากับการทดสอบแก้ไขปัญหาในจุดต่างๆเหล่านี้เป็นเวลานาน

8.3.3 ความล่าช้าของการพัฒนาแพลตฟอร์มสำหรับใช้ในโครงการ จึงส่งผลกระทบให้เกิดความล่าช้าในการพัฒนาระบบปฏิบัติการ

8.4 แนวทางการพัฒนาต่อ

8.4.1 พัฒนาซอฟต์แวร์ไลบรารีสำหรับอุปกรณ์เชื่อมต่ออื่นๆ เช่น USB หรือ PIO เป็นต้นเพื่อเพิ่มความสะดวกสำหรับการใช้งานในการพัฒนาโปรแกรม

8.4.2 ลดขนาดของระบบปฏิบัติการ ให้มีขนาดเล็กลง

8.4.3 พัฒนาซอฟต์แวร์เพิ่มเติม สำหรับใช้ช่วยในการพัฒนาซอฟต์แวร์บนชุดพัฒนาคอม86

8.4.4 ปรับปรุงเรื่องเสถียรภาพในการทำงานของระบบปฏิบัติการให้มีความคงทนยิ่งขึ้น

8.5 การนำโครงการไปประยุกต์ใช้งาน

โครงการนี้เป็นการพัฒนาระบบปฏิบัติการสำหรับชุดพัฒนาคอม86ที่ถูกออกแบบมาให้ใช้เป็นแพลตฟอร์มในการพัฒนาอุปกรณ์ระบบฝังตัว ดังนั้นการนำไปใช้งานจะอยู่ในรูปของการอยู่เป็นส่วนหนึ่งของชุดพัฒนาคอม86 โดยเป็นซอฟต์แวร์พื้นฐานสำหรับช่วยให้นักพัฒนาสามารถพัฒนาโปรแกรมสำหรับระบบฝังตัวได้ง่ายยิ่งขึ้น โดยเฉพาะจุดแข็งของระบบปฏิบัติการที่มีรูปแบบคล้ายระบบยูนิกซ์มีความสามารถในการทำงานแบบขนานกันได้อย่างพร้อมๆกันหลายๆโปรเซส ซึ่งจะทำให้การพัฒนาซอฟต์แวร์สำหรับระบบฝังตัวที่มีความซับซ้อน เช่น โปรแกรมควบคุมการทำงานที่มีการสื่อสารผ่านระบบ network สามารถทำได้ง่ายขึ้นเมื่อเทียบกับการพัฒนาโปรแกรมโดยปกติที่ไม่มีระบบปฏิบัติการรองรับ

8.6 สรุปภาพรวมของโครงการ

โดยภาพรวมของโครงการนี้จะเป็นการศึกษาและพัฒนาระบบปฏิบัติการเพื่อใช้งานกับบอร์ดคอม86 เพื่อสร้างเป็นชุดพัฒนาทางด้านระบบฝังตัว โดยได้เลือกระบบปฏิบัติการมินิซมาเป็นต้นแบบในการแก้ไขและพัฒนาเป็นหลักแต่เนื่องจากปัญหาของความแตกต่างทางด้านฮาร์ดแวร์ระบบที่ใช้งานและปัญหาของความไม่เข้ากันของซอฟต์แวร์ในบางจุดที่ไม่ตรงกับตามที่ได้ประเมินไว้ในขั้นตอนของการประเมินเบื้องต้น เช่นอีเธอร์เน็ตไวน์ไดรเวอร์จึงทำให้การโครงการล่าช้ากว่ากำหนดที่ตั้งไว้ แต่ยังคงสามารถพัฒนาได้ตรงตามความต้องการพื้นฐาน คือสามารถทำงานได้จริงบนบอร์ดคอม86 ยังขาดแต่การพัฒนาซอฟต์แวร์ไลบรารีสำหรับสนับสนุนการทำงานของอุปกรณ์เชื่อมต่อต่างๆเพิ่มเติม จึงนับว่าโครงการนั้นประสบความสำเร็จอยู่ในระดับที่น่าพอใจ

บรรณานุกรม

- [1] Steve Heath: "Embedded Systems Design", Newnes, 1997.
- [2] ผศ. อภินทรอนุภาณุ : "เทคโนโลยีระบบฝังตัวที่เหมาะสมกับประเทศไทย", 2001
- [3] "Embedded Designs Move to Linux and Systems-on-a-Chip", embedded
Linux Journal issue 4, July-August 2001
- [4] Andrew S. Tanenbaum: "Operating Systems, Design and Implementation", 1997.
- [5] สุริยันต์ เลหาประภานนท์: "186CC Board Communication Controller board",
Embedded Systems Lab, 2001
- [6] "Am186CC™ High performance, 80C186-compatible 16-bit Embedded
Communications Controller", AMD datasheet
- [7] เว็บไซต์หลักของมินิกซ์ที่ <http://www.cs.vu.nl/~ast/minix.html>
- [8] เนื้อหารายละเอียดของมินิกซ์ที่ <http://minix1.hampshire.edu>
- [9] มินิกซ์ 32 บิตที่ <http://www.minix-vmd.org/>
- [10] แหล่งข้อมูลของมินิกซ์ที่ <ftp://ftp.cs.vu.nl/pub/minix>.
- [11] คำถามที่ถามบ่อยๆที่ <http://harmless.rinet.com/Minix/FAQ-17.html>
- [12] จดหมายข่าวมินิกซ์ที่ <http://listserv.nodak.edu/archives/minix-l.html>
- [13] รายละเอียดของฟรีดอสทั้งหมด <http://www.freedos.org>
- [14] ซอร์สโค้ดและเอกสารอ้างอิงที่ <http://www.sourceforge.net>
- [15] คำถามที่ได้รับการถามบ่อยๆ, "Freedos FAQ", เว็บไซต์ของฟรีดอส
- [16] เอกสารอ้างอิงหลักของฟรีดอส, "Freedos Information Sheet"
- [17] เอกสารคู่มือการใช้งานเครือข่ายของฟรีดอส, "Freedos network".
- [18] เอกสารกำกับแบบลักษณะของฟรีดอส, "Freedos spec"
- [19] ซอร์สโค้ดของอีแอลเคสที่ <ftp://linux.mit.edu/pub/ELKS>
- [20] เอกสารอ้างอิงที่ <ftp://ftp.ecs.soton.ac.uk/pub/elks/>
- [21] หน้าเว็บหลักของอีแอลเคสที่ <http://www.ecs.soton.ac.uk/elks>
- [22] Sun Micro System, Inc: "Java 2 Platform Micro Edition Technology for Creating
Mobile Devices", 2000