สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

เครื่องทดสอบสมรรถนะของ Rotary Positioner

ROTARY POSITIONER PERFORMANCE TESTER

T104013

นางสาววรรษมน นาคินทร์
นางสาวศุจินธร แฟงฝอย
นายสการย์ สายกระสุน

วิ ว 7 3 ก
2551

เลขหมู่.....
เลขทะเบียน 104013
วัน,เดือน,ปี.....

.b 12111259
.i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2551

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ROTARY POSITIONER PERFORMANCE TESTER

WATSAMON NAKARIN

SUJINTHON FANGFOY

SAKARN SAIKRASOON

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

DEPARTMENT OF INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUE OF TECHNOLOGY LADKRABANG

2008

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาควิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

.....

หัวข้อปริญญานิพนธ์ เครื่องทดสอบสมรรถนะของ Rotary Positioner

ROTARY POSITIONER PERFORMANCE TESTER

นักศึกษาผู้จัดทำ นางสาววรรษมน นาครินทร์ **รหัสนักศึกษา** 48010771

นางสาวสุจินธร แฟงฝอย **รหัสนักศึกษา** 48010904

นายสการย์ สายกระสุน **รหัสนักศึกษา** 48010924

ปริญญา วิศวกรรมศาสตรบัณฑิต

สาขาวิชา วิศวกรรมการวัดคุม

ปีการศึกษา 2551

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
รศ.อาจินต์ น่วมสำราญ	

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปริญญานิพนธ์	เครื่องทดสอบสมรรถนะของ Rotary Positioner		
	ROTARY POSITIONER PERFORMANCE TESTER		
นักศึกษาผู้จัดทำ	นางสาววรรษมน นาครินทร์	รหัสนักศึกษา	48010771
	นางสาวศุจินธร แฟงฝอย	รหัสนักศึกษา	48010904
	นายสการย์ สายกระสุน	รหัสนักศึกษา	48010924
อาจารย์ที่ปรึกษา	รศ.อาจินต์ น่วมสำราญ		
ปีการศึกษา	2551		

บทคัดย่อ

Rotary Positioner เป็นเซอร์โวมอเตอร์ ที่เป็นองค์ประกอบสำคัญในกระบวนการการเขียนสัญญาณเวอร์โวลงบนแผ่นฮาร์ดดิสก์ไดรฟ์ด้วยเครื่องเขียนสัญญาณแบบ NCH (Non Clock Head) ปริญญานิพนธ์เล่มนี้ได้นำเสนอวิธีการสร้างเครื่องทดสอบ Rotary Positioner เพื่อทดสอบหา Rotary Positioner ที่ดีและเสีย เครื่องทดสอบ Rotary Positioner ที่คณะผู้จัดทำได้ทำขึ้นอาศัยหลักการทดสอบแบบวงรอบเปิด และได้นำไมโครคอนโทรลเลอร์ ARM7 มาเป็นหน่วยควบคุมและประมวลผลสัญญาณที่ได้จาก Positiner Control Card ซึ่งสัญญาณที่นำมาประมวลผลเป็นสัญญาณพัลซ์เอ็นโค้ดเดอร์ของ Rotary Positioner วิธีการทดสอบสมรรถนะของ Rotary Positioner สามารถทำได้โดยการ move track ไปจำนวณ 100 และ 400 track เพื่อเปรียบเทียบหา Rotary Positiner ที่ดีและเสีย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Thesis Title	Rotary Positioner Performance Tester	
Authors	Miss Watsamon	Nakarin
	Miss Sujinthon	Fangfoy
	Mr.Sakarn	Saikrasoon
Thesis Advisor	Assoc.Prof.Arjin	Numsomran
Year	2008	

ABSTRACT

The Rotary Positioner is the servomotor which is the main factor process of writing servo signal on hard disk drive written by NCH (Non Clock Head) model. This degree offers writing books how to build a test Rotary Positioner for Rotary Positioner Rotary Positioner good and lose.

Rotary Positioner to the test team can create a live test of open loop's control method. The ARM7 microcontroller and a control unit and signal processing from Positiner Control Card. This signal is a signal processing Pluse of Rotary Positioner. How to test the performance of the Rotary Positioner can move track to remember the 100 track and 400 track for Rotary Positiner comparison to find a good and lose.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้ ด้วยความกรุณาของรองศาสตราจารย์ อาจินต์ น่วมสำราญ อาจารย์ที่ปรึกษาปริญญานิพนธ์ และคณาจารย์ ซึ่งได้ให้คำปรึกษา ข้อชี้แนะ และความช่วยเหลือในหลายสิ่งหลายอย่างจนกระทั่งลุล่วงไปได้ด้วยดี และทำให้ปริญญานิพนธ์เล่มนี้เสร็จสมบูรณ์จึงขอกราบขอบพระคุณเป็นอย่างสูงมา ณ ที่นี้

ขอบพระคุณ บริษัท ฮิตาชิ โกลบอล สตอเรจ เทคโนโลยีส์ (ประเทศไทย) จำกัด ที่ได้ให้การสนับสนุน สถานที่ อุปกรณ์ และความรู้ทางด้านเทคโนโลยีฮาร์ดดิสก์ และขอบพระคุณพี่ ๆ ที่บริษัททุกคน ที่ให้คำปรึกษา แนะนำ และช่วยเหลือตลอดการฝึกงานและการทำวิจัย

ขอบคุณ คุณไมตรี ที่ให้ความเอื้อเฟื้อ และความกรุณาในการแก้ไขข้อบกพร่องต่าง ๆ รวมทั้งให้คำแนะนำและกำลังใจ ขอบคุณ คุณเลอสันต์ คุณกฤษ ที่ให้ความรู้และแนะนำเรื่องต่าง ๆ ที่น้อง ๆ สรรหามาถามอยู่เสมอและที่ลืมไม่ได้ คือ คุณกฤษฎา ขอบคุณกฤษฎามาก ๆ ที่ช่วยเหลือทุกสิ่งทุกอย่างและให้กำลังใจพี่ ๆ มาตลอด

คุณค่าและประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ คณะผู้จัดทำขอมอบบูชาพระคุณคุณพ่อ คุณแม่ ครู - อาจารย์ ตลอดจนผู้มีพระคุณทุกท่านด้วยความเคารพยิ่ง

คณะผู้จัดทำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

หน้า

บทคัดย่อภาษาไทย I

บทคัดย่อภาษาอังกฤษ II

กิตติกรรมประกาศ III

สารบัญ IV

สารบัญตาราง VI

สารบัญภาพ VII

บทที่ 1 บทนำ 1

1.1 ความสำคัญของปริญญานิพนธ์ 1

1.2 วัตถุประสงค์ของปริญญานิพนธ์ 2

1.3 ขอบเขตของปริญญานิพนธ์ 2

1.4 ขั้นตอนการศึกษา 2

1.5 ประโยชน์ที่คาดว่าจะได้รับ 3

บทที่ 2 ทฤษฎี 4

2.1 ความรู้เกี่ยวกับฮาร์ดดิสก์ไดรฟ์ 4

2.1.1 โครงสร้างของฮาร์ดดิสก์ไดรฟ์ (Hard Disk Construction) 4

2.1.1.1 ชิ้นส่วนของฮาร์ดดิสก์ไดรฟ์ 5

2.1.2 หลักการทำงานของฮาร์ดดิสก์ไดรฟ์ 8

2.1.3 สัญญาณเซอร์โว (Servo) 10

2.1.4 กระบวนการเขียนสัญญาณเซอร์โว
และระบบควบคุมตำแหน่งของ Rotary Positioner 11

2.1.5 เครื่องเขียนสัญญาณเซอร์โว หรือ NCH (Non Clock Head) 18

2.1.5.1 เอ็นโค้ดเดอร์แบบเลเซอร์โรตารี่ (Laser Rotary Encoder) 19

2.1.6 วิธีการตรวจสอบสเต็ปการเคลื่อนที่ของหัวอ่าน/เขียน 20

2.2 ความรู้เกี่ยวกับ ARM7 22

2.2.1 สถาปัตยกรรมและคุณสมบัติของ ARM 22

2.2.2 ARM REGISTERS 22

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

หน้า

2.2.3 คุณสมบัติข้อเด่นของ ARM7 23
2.2.4 ไมโครคอนโทรลเลอร์ตระกูล ADUc ของ Analog Devices 23
2.2.4.1 การทำงานแบบ 16/32 บิต ของARM
และการทำงานโดยใช้ชุดคำสั่งแบบ Thumb 24
2.2.4.2 อินเทอร์รัฟท์ 24
2.2.5 อุปกรณ์เชื่อมต่ออินพุทเอาท์พุทแบบอนุกรมของ
ไมโครคอนโทรลเลอร์ตระกูล ADUc702x 24
2.2.5.1 การติดต่อพอร์ตและการควบคุมการทำงานของ
พอร์ตอเนกประสงค์ (General Purpose I/O : GPIO) 25
2.2.6 งานประยุกต์ต่างๆของไมโครคอนโทรลเลอร์ตระกูล ADUc702x 30
2.3 การเขียนโปรแกรมภาษา C เพื่อสั่งงาน ARM7 30
2.4 โปรแกรม PLA (Programmable Logic Array) 30
2.4.1 Input / Outputs Signals for PLAs 32

บทที่ 3 วิธีการดำเนินงาน 35
3.1 ศึกษากระบวนการทดสอบสมรรถนะของ Rotary Positioner แบบเดิม
(APC Test) 35
3.1.1 วิธีการและขั้นตอนการทำงาน 35
3.1.1.1 วิธีการทดสอบ Good/Bad Rotary Positioner 35
3.1.1.2 ตารางบันทึกผล 36
3.1.1.3 ตัวอย่างกราฟ APC จาก MATLAB 37
3.1.2 สรุปผลที่ได้จากการทดสอบด้วยวิธีแบบเดิม (APC Test) 38
3.2 ออกแบบเครื่องทดสอบสมรรถนะของ Rotary Positioner 38
3.2.1 อุปกรณ์ฮาร์ดแวร์ 38
3.2.1.1 LCD Graphpic 38
3.2.1.2 บอร์ด ET-BASE ARM7024 (ADUc7024) 39
3.2.1.3 บอร์ดสวิตซ์ และ LED 40
3.2.1.4 รางถ่าน AA ขนาด 4 ก้อน 41

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

หน้า

3.2.2 โปรแกรมที่ใช้ในการทดสอบสมรรถนะของ Rotary Positioner 41

3.2.3 แผนผังการทำงานของโปรแกรมที่ใช้ใน
เครื่องทดสอบสมรรถนะของ Rotary Positioner 43

3.3 ออกแบบการทดลองเครื่องทดสอบสมรรถนะของ Rotary Positioner 45

3.3.1 ขั้นตอนการทำงาน 45

บทที่ 4 การแสดงผล 51

4.1 เครื่องทดสอบสมรรถนะของ Rotary Positioner 51

4.2 การทำงานของ เครื่องทดสอบสมรรถนะของ Rotary Positioner 51

4.3 ผลการทดลอง 53

4.3.1 ผลการทดสอบ Rotary Positioner ตัวที่ดี Serial Number 10xxx-4x
เพื่อนำมาเป็นมาตรฐานในการยืนยันผลการทดลอง 53

4.3.2 บันทึกผลการทดลองที่ 100 track 55

4.3.3 บันทึกผลการทดลองที่ 400 track 55

4.3.4 ผลจากการเคลื่อนที่ไป 100 Track 58

4.3.5 ผลจากการเคลื่อนที่ไป 400 Track 61

4.4 ตรวจสอบและยืนยันผลการทดลองด้วยวิธีทดสอบแบบ APC 64

บทที่ 5 สรุปผลการวิจัยและข้อเสนอแนะ 68

5.1 สรุปผลการทดลอง 68

5.2 ข้อเสนอแนะ 68

บรรณานุกรม 69

ภาคผนวก 70

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

ตารางที่ **หน้า**

2.1 แสดง error code ของ Rotary Positioner 17

2.1 (ต่อ) แสดง error code ของ Rotary Positioner 17

2.2 GPIO Pin Function Descriptions 25

2.2 (ต่อ) GPIO Pin Function Descriptions 26

2.3 GPxCON Registers 27

2.4 GPxCON MMR Bit Descriptions 27

2.5 GPxDAT Registers 28

2.6 GPxDAT MMR Bit Descriptions 28

2.7 GPxSET Registers 28

2.8 GPxSET MMR Bit Descriptions 28

2.9 GPxCLR Registers 29

2.10 GPxCLR MMR Bit Descriptions 29

2.11 IRQEN Register 29

2.12 FIQCLR Register 29

3.1 Good Rotary Positioner Test 36

3.2 Bad Rotary Positioner Test 36

4.1 แสดงผลการทดลองที่ 100 track 55

4.2 แสดงผลการทดลองที่ 400 track 56

4.2 (ต่อ) แสดงผลการทดลองที่ 400 track 56

4.3 ตารางยืนยันผลการทดลองหา Rotary Positioner ดีหรือเสีย โดยใช้
หลักการควบคุมคุณภาพ (QC) ที่ 100 track 57

4.4 ตารางยืนยันผลการทดลองหา Rotary Positioner ดีหรือเสีย โดยใช้
หลักการควบคุมคุณภาพ (QC) ที่ 400 track 57

4.4 (ต่อ) ตารางยืนยันผลการทดลองหา Rotary Positioner ดีหรือเสีย โดยใช้
หลักการควบคุมคุณภาพ (QC) ที่ 400 track 58

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ

ภาพที่	หน้า
2.1 แสดงการประกอบ HDA และ PCBA บนฮาร์ดดิสก์ไดรฟ์	4
2.2 แสดงลักษณะ HDA และ PCBA	5
2.3 ส่วนประกอบของฮาร์ดดิสก์ 1	5
2.4 Head stack assembly (HSA)	6
2.5 วอยซ์คอยมอเตอร์ (VCM)	6
2.6 หัวอ่าน/เขียนฮาร์ดดิสก์ไดรฟ์	7
2.7 ส่วนประกอบของฮาร์ดดิสก์ 2	7
2.8 แผ่น Aluminum Alloy Platter	8
2.9 แบบจำลอง Track และ Sector	9
2.10 การลอยตัวของหัวอ่าน/เขียนบนแผ่นมีเดียหรือ Platter	9
2.11 แสดงการขยายสัญญาณเซอร์โว	10
2.12 แสดง OD และ ID	12
2.13 กระบวนเขียนสัญญาณเซอร์โว	12
2.14 Radial Card	13
2.15 R/W Motor Card	13
2.16 Clock Pattern Card	14
2.17 DE IF Card	14
2.18 Positioner Control Card	15
2.19 วงรอบการควบคุมตำแหน่งของ Rotary Positioner	16
2.20 NCH (Non Clock Head)	18
2.21 Head Stack Assembly (HSA)	18
2.22 หลักการทำงานของเอ็นโค้ดเดอร์แบบเลเซอร์โรตารี่	19
2.23 ความสัมพันธ์ระหว่างตำแหน่งของหัวและสัญญาณ APC	21
2.24 ค่าคงที่ของ Ww และ Wr	21
2.25 การคำนวณ	21
2.26 ภาษา C สำหรับควบคุม ARM7	30
2.27 ฟังก์ชันไดอะแกรมของ single PLA	31
2.28 Input / outputs signals for PLAs	32

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ(ต่อ)

ภาพที่ **หน้า**

2.29 หน้าต่างเริ่มต้นของโปแกรม 33

2.30 หน้าต่าง Element 33

2.31 แผนผังวงจรลอจิกเกต (Logic gate) ที่วาด 34

3.1 สัญญาณ APC Step ของ Good Rotary Positioner 37

3.2 สัญญาณ APC Step ของ Bad Rotary Positioner 37

3.3 LCD Graphpic 38

3.4 บอร์ด ET-BASE ARM7024 (ADUc7024) 40

3.5 ET-TEST 10P/INP-OUT 40

3.6 รางถ่าน AA ขนาด 4 ก้อน 41

3.7 หน้าต่างโปรแกรม Keil uVersion3 41

3.8 การกลับ Logic สัญญาณพัลส์ของ Encoder จาก Positioner card 42

3.9 ตัวอย่างการใส่ PLA 42

3.10 แผนผังการทำงานของโปรแกรมที่ใช้ในเครื่อง
ทดสอบสมรรถนะของ Rotary Positioner 44

3.11 ภาพประกอบขั้นตอนการถอด Rotary Positioner 45

3.12 ภาพประกอบขั้นตอนการใส่ Rotary Positioner 46

3.13 ภาพประกอบการติดตั้งบีค (Beak) 48

3.14 โซลินอยด์ควบคุมการ Load และ Unload Golden Unit 49

3.15 แสดงจุด Test Point บน Positioner Card 50

4.1 เครื่องทดสอบสมรรถนะของ Rotary Positioner 51

4.2 ผังการทำงานของ เครื่องทดสอบสมรรถนะของ Rotary Positioner 52

4.3 ภาพจากการ Test APC 53

4.4 ภาพจากการที่ใช้เครื่องทดสอบสมรรถนะของ Rotary Positioner ทดสอบ
วัดสัญญาณจากการ move track ไปจำนวน100 แทรค 54

4.5 ภาพจากการที่ใช้เครื่องทดสอบสมรรถนะของ Rotary Positioner ทดสอบ
วัดสัญญาณจากการ move track ไปจำนวน400 แทรค 54

4.6 การเคลื่อนที่ไป 100 Track ของ Good 1 58

4.7 การเคลื่อนที่ไป 100 Track ของ Good 2 59

4.8 การเคลื่อนที่ไป 100 Track ของ Good 3 59

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ(ต่อ)

ภาพที่ **หน้า**

4.9 การเคลื่อนที่ไป 100 Track ของ Bad 1 60

4.10 การเคลื่อนที่ไป 100 Track ของ Bad 2 60

4.11 การเคลื่อนที่ไป 100 Track ของ Bad 3 61

4.12 การเคลื่อนที่ไป 400 Track ของ Good 1 61

4.13 การเคลื่อนที่ไป 400 Track ของ Good 2 62

4.14 การเคลื่อนที่ไป 400 Track ของ Good 3 62

4.15 การเคลื่อนที่ไป 400 Track ของ Bad 1 63

4.16 การเคลื่อนที่ไป 400 Track ของ Bad 2 63

4.17 การเคลื่อนที่ไป 400 Track ของ Bad 3 64

4.18 กราฟ APC Step จาก Good 1 (S/N 33xxx-M4x) 64

4.19 กราฟ APC Step จาก Good 2 (S/N 35xxx-Mx) 65

4.20 กราฟ APC Step จาก Good 3 (S/N 31xxx-Mx) 65

4.21 กราฟ APC Step จาก Bad 1 (S/N 37xxx-Nx) 66

4.22 กราฟ APC Step จาก Bad 2 (S/N 33xxx-Mx) 66

4.23 กราฟ APC Step จาก Bad 3 (S/N 35xxx-Mx) 67

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความสำคัญของปริญญานิพนธ์

ในยุคปัจจุบันเป็นยุคที่เทคโนโลยีทางระบบคอมพิวเตอร์มีการแข่งขันกันสูง จึงจำเป็นที่แต่ละบริษัทจะต้องแข่งขันกัน ในเรื่องของการเพิ่มประสิทธิภาพของอุปกรณ์ให้มีศักยภาพที่สูงขึ้นเป็นไปตามความต้องการของผู้บริโภค ซึ่งฮาร์ดดิสก์เป็นส่วนหนึ่งของอุปกรณ์ทางคอมพิวเตอร์ที่มีความจำเป็นสำหรับการบันทึกข้อมูลที่นับวันข้อมูลต่าง ๆ ยิ่งมีขนาดใหญ่ และมีปริมาณมากขึ้นกว่าในอดีตรวมทั้งเป็นที่นิยมมากขึ้นเรื่อย ๆ ด้วยเหตุนี้ทางบริษัทผู้ผลิตต้องทำการพัฒนากระบวนการผลิตฮาร์ดดิสก์ไดรฟ์เพื่อแข่งขันและดึงดูดผู้บริโภค สิ่งที่เป็นจุดขายที่สำคัญที่สุดก็คือ ความจุและราคา ดังนั้นบริษัทผู้ผลิตฮาร์ดดิสก์ไดรฟ์ต่าง ๆ ล้วนแล้วแต่ให้ความสำคัญในเรื่องนี้ทั้งสิ้นและฮาร์ดดิสก์ไดรฟ์จะมีความจุมากหรือน้อยนั้น ปัจจัยหนึ่งที่มีสำคัญ คือ การเขียนสัญญาณเซอร์โว ซึ่งเป็นการเขียนสัญญาณลงบนแผ่นฮาร์ดดิสก์ไดรฟ์ โดยที่สัญญาณจะต้องไม่มีการทับหรือซ้อนกันในแต่ละแทรค (track) และไม่ผิดเพี้ยนไปจากรูปแบบที่ถูกต้องของสัญญาณเซอร์โว

กระบวนการเขียนสัญญาณเซอร์โวที่จะกล่าวถึงในปริญญานิพนธ์ฉบับนี้ เป็นวิธีการอาศัยการควบคุมการเคลื่อนที่ของ Rotary Positioner โดยการให้ pusher ของ Rotary Positioner ลากหัวอ่าน/เขียน (Read/Write : R/W) ไปยังตำแหน่งที่ต้องการเขียนสัญญาณเซอร์โวบนแผ่นบันทึกข้อมูลที่อยู่ในฮาร์ดดิสก์ไดรฟ์ ดังนั้นหาก Rotary Positioner ที่ใช้ไม่มีสมรรถนะเพียงพอหรือเกิดขัดข้องบางประการ ย่อมส่งผลเสียต่อการเขียนสัญญาณเซอร์โวและคุณภาพของฮาร์ดดิสก์ไดรฟ์

ปัญหาที่เกิดจากกระบวนการเขียนสัญญาณเซอร์โวนั้น เป็นปัญหาที่สำคัญและสร้างความเสียหายให้กับกระบวนการผลิตฮาร์ดดิสก์ไดรฟ์เป็นอย่างมาก ซึ่งทางบริษัทฮิตาชิก็ได้ตระหนักถึงปัญหานี้เช่นกัน จึงได้ให้ความร่วมมือ และเปิดโอกาสให้ผู้วิจัยได้มีส่วนร่วมทำการคิดวิเคราะห์และหาทางแก้ไขปัญหา ซึ่งจากการศึกษาข้อมูลของบริษัท ฮิตาชิ โกลบอล สตอเรจ เทคโนโลยีส์ (ประเทศไทย) จำกัด พบว่ากระบวนการเขียนสัญญาณเซอร์โวของบริษัทฮิตาชินั้น จะใช้เครื่อง NCH เป็นตัวควบคุมกระบวนการเขียนสัญญาณเซอร์โว รวมทั้งใช้ทดสอบสมรรถนะของ Rotary Positioner ด้วยวิธีการ APC ซึ่งวิธีการนี้ใช้เวลาทดสอบนานถึงครั้งละ 24 ชั่วโมง

จากปัญหาดังกล่าว ทำให้ผู้วิจัยต้องศึกษาหาข้อมูลในเรื่องต่าง ๆ ที่เกี่ยวกับฮาร์ดดิสก์ไดรฟ์ ไม่ว่าจะเป็นองค์ประกอบ โครงสร้าง หลักการทำงาน รวมไปถึงการผลิต วิธีการและการประกอบฮาร์ดดิสก์ไดรฟ์ ซึ่งข้อมูลเหล่านี้ล้วนแล้วแต่เป็นความรู้พื้นฐานสำคัญ แต่สิ่งที่จะทำการศึกษาและค้นคว้าเป็นพิเศษคือ เรื่องกระบวนการเขียนสัญญาณเซอร์โวและการวิเคราะห์สัญญาณจาก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Rotary Positioner เพื่อนำมาเป็นข้อมูลเบื้องต้นในการวิจัย และคิดค้นเครื่องมือทดสอบสมรรถนะของ Rotary Positioner

ดังนั้นปริญญานิพนธ์เล่มนี้ จึงเป็นส่วนสำคัญที่ช่วยลดปัญหาและเวลาในกระบวนการเขียนสัญญาณเซอร์โวที่เกิดจาก Rotary Positioner ให้น้อยลง จากการที่ผู้วิจัยได้ทำการคิดเครื่องทดสอบสมรรถนะของ Rotary Positioner ขึ้นมา ซึ่งเป็นส่วนสำคัญที่จะสามารถนำไปพัฒนาให้เกิดประโยชน์ได้ต่อไป

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. เพื่อศึกษาเทคโนโลยีของฮาร์ดดิสก์ไดรฟ์
2. ศึกษาองค์ประกอบพื้นฐานต่างๆที่เกี่ยวกับการเขียนสัญญาณเซอร์โว
3. คิดค้นเครื่องทดสอบสมรรถนะของ Rotary Positioner เพื่อนำไปทดสอบหา Rotary Positioner ดีและเสีย
4. นำเสนองานวิจัยที่เกี่ยวข้องกับฮาร์ดดิสก์ไดรฟ์ให้เป็นที่รู้จักแพร่หลายมากขึ้น

1.3 ขอบเขตของปริญญานิพนธ์

1. ศึกษากระบวนการทำงาน ส่วนประกอบต่างๆของฮาร์ดดิสไดรฟ์
2. ศึกษากระบวนการเขียนสัญญาณเซอร์โว
3. ออกแบบเครื่องทดสอบสมรรถนะของ Rotary Positioner ขึ้นมาโดยการใช้การวิเคราะห์สัญญาณ Pulse ที่มาจาก Encoder ด้วยบอร์ด ARM7
4. ทดสอบการทำงานของเครื่องทดสอบสมรรถนะของ Rotary Positioner ให้สามารถทำงานได้

1.4 ขั้นตอนการศึกษา

1. ศึกษาเกี่ยวกับส่วนประกอบต่างๆของฮาร์ดดิสก์ไดรฟ์
2. ศึกษาหลักการทำงานและการเขียนสัญญาณเซอร์โวลงบนแผ่นฮาร์ดดิสก์ไดรฟ์
3. ศึกษาหาปัจจัยที่ทำให้กระบวนในการเขียนสัญญาณเซอร์โวไม่เป็นไปตามกระบวนการที่ถูกต้อง ทำให้เกิดการสูญเสียทรัพย์สินและเวลา
4. ศึกษาส่วนประกอบของ Rotary Positioner
5. ศึกษาวิธีการทดสอบแบบเก่าหรือการทดสอบด้วยวิธี APC หาข้อบกพร่อง เพื่อนำมาพัฒนา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6. ศึกษาการเขียนโปรแกรมควบคุม ARM7 ด้วยภาษา C และเขียนโปรแกรมทดสอบสมรรถนะของ Rotary Positioner

7. ออกแบบเครื่องทดสอบสมรรถนะของ Rotary Positioner เพื่อนำไปทดลองการใช้งานและความถูกต้องในการทดสอบสมรรถนะของ Rotary Positioner

8. นำผลการทดลองมาสรุปผลการทำงานและทฤษฎี

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. เข้าใจกระบวนการทำงานพื้นฐานของฮาร์ดดิสก์ไดรฟ์รวมทั้งส่วนประกอบต่างๆ

2. เข้าใจกระบวนการเขียนสัญญาณเซอร์โว และรู้ถึงสาเหตุที่ทำให้การทำงานไม่เป็นไปตามกระบวนการที่ถูกต้อง และสามารถบอกได้ว่าลักษณะการทำงานวิธีการตามลำดับขั้นตอนของกระบวนการเป็นอย่างไร

3. นำไปพัฒนาวิธีการหรือเครื่องมือทดสอบสมรรถนะของ Rotary Positioner จากเดิมที่มีอยู่แล้วให้ดีขึ้น

4. เป็นการช่วยลดต้นทุนและเวลาของกระบวนการผลิตในอุตสาหกรรมฮาร์ดดิสก์ไดรฟ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ทฤษฎี

2.1 ความรู้เกี่ยวกับฮาร์ดดิสก์ไดรฟ์

2.1.1 โครงสร้างของฮาร์ดดิสก์ไดรฟ์ (Hard Disk Construction)

ฮาร์ดดิสก์ประกอบด้วย 2 ส่วนหลัก

1. HDA : Hard Disk Assembly

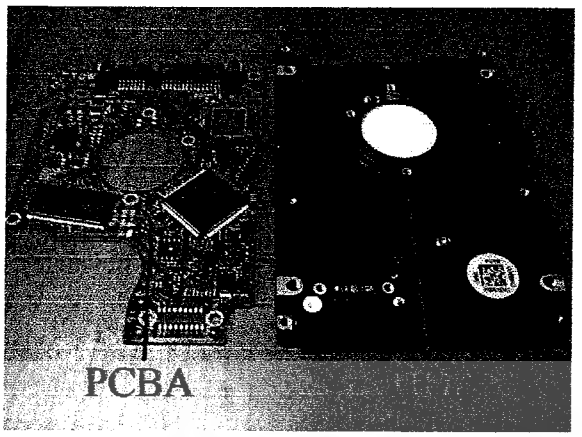
เป็นส่วนกลไกของฮาร์ดดิสก์ไดรฟ์ เช่น มอเตอร์, แผ่นมีเดีย, HSA (Head Stack Assembly) และอื่น ๆ การประกอบ HDA จะต้องทำในห้องคลีนรูม (Clean Room) ซึ่งควบคุมไฟฟ้าสถิตย์ (ESD) และฝุ่นละอองในอากาศ

2. PCBA : Printed Circuit Board Assembly

เป็นส่วนที่เป็นอิเล็กทรอนิกส์ ซึ่งใช้ควบคุมส่วนที่เป็นกลไกใน HDA รวมทั้งการติดต่อสื่อสารกับคอมพิวเตอร์

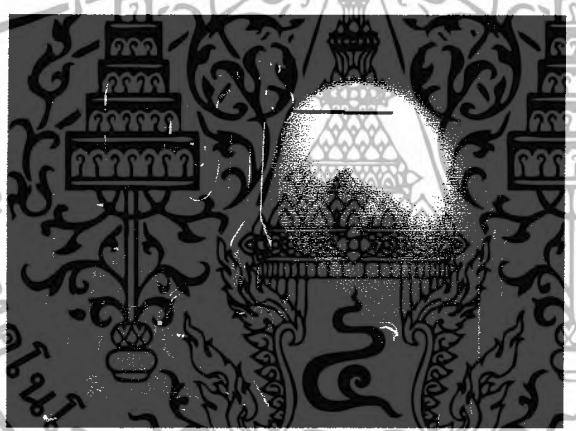
ภาพที่ 2.1 แสดงการประกอบ HDA และ PCBA บนฮาร์ดดิสก์ไดรฟ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 2.2 แสดงลักษณะ HDA และ PCBA

2.1.1.1 ชิ้นส่วนของฮาร์ดดิสก์ไดรฟ์



ภาพที่ 2.3 ส่วนประกอบของฮาร์ดดิสก์ 1

1. Top cover

   ส่วนของฝาปิดด้านบนของฮาร์สดิสก์ไดรฟ์

2. Disc clamp

   ส่วนที่ยึดแผ่นมีเดียเพื่อให้ติดกับสปินเดิลมอเตอร์

3. Disc / Media หรือ Platter

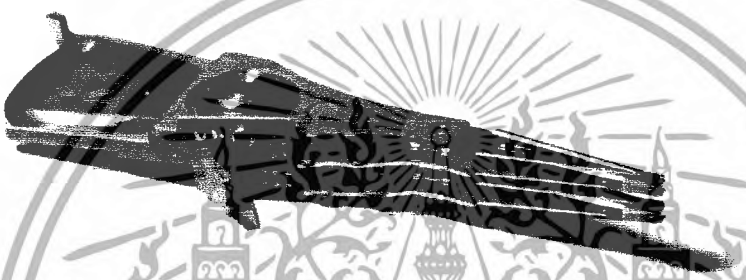
ส่วนของแผ่นเคลือบสารแม่เหล็กเพื่อใช้ในการบันทึกข้อมูล มีลักษณะเป็นแผ่นกลมแบนวางซ้อนกันหลายแผ่น แต่ละแผ่น เรียกว่า แพลทเทอร์ (Platter) แต่เดิมแพลทเทอร์ทำจากวัสดุจำพวกอะลูมิเนียม แต่มันมีข้อด้อย คือ มีความหนามากไป และมีค่าต้านทานความร้อนต่ำ ต่อมาเปลี่ยนมาใช้วัสดุผสมจากแก้วและซีเมนส์แทน วัสดุที่ใช้เหล่านี้ยังต้องได้รับการเคลือบด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ออกไซด์ที่เป็นสารแม่เหล็ก สารแม่เหล็กที่ใช้นี้จะต้องมีสมบัติคือ มีค่าสภาพลบล้างแม่เหล็ก (Coercivity) สูง คือ มีความเป็นแม่เหล็กสูง อันจะช่วยให้เกิดการเปลี่ยนแปลงสถานะแบบฉับพลัน ได้นำไปสู่การที่ฮาร์ดดิสก์ไดรฟ์ที่เขียนแนวลองจิจูด (Longitude) มีความจุสูงขึ้นได้

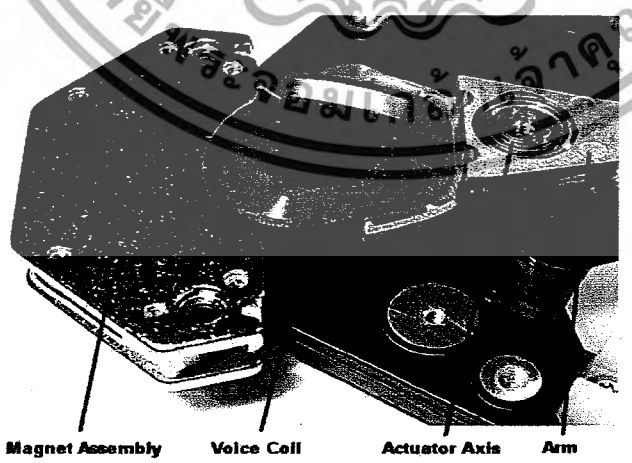
4. Head stack assembly (HSA)

เป็นส่วนของแขนที่ติดกับหัวอ่าน/หัวเขียน และวอยซ์คอยมอเตอร์ (VCM) ภายในตัว HSA จะมีอุปกรณ์ที่ทำหน้าที่ อ่านและเขียนสัญญาณแม่เหล็ก นั่นก็คือ Slider ซึ่งตัว Slider มีโครงสร้างภายในที่สลับซับซ้อนมาก เนื่องจากเป็นการนำชั้นสารหลายๆชั้นมารวมกัน ซึ่งถือว่าเป็นหัวใจหลักของการอ่าน/เขียนข้อมูล



ภาพที่ 2.4 Head stack assembly (HSA)

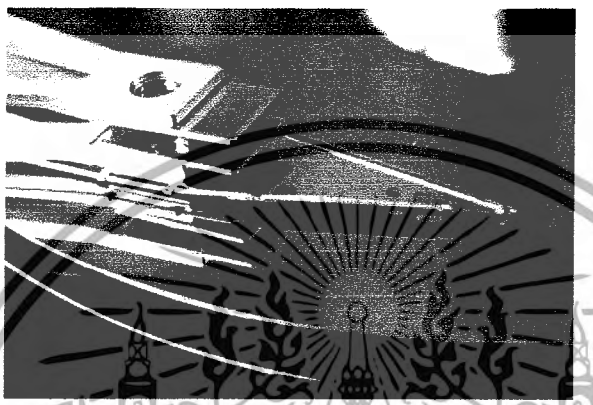
VCM เป็นส่วนที่เป็นขดลวดทองแดง และเมื่อกระแสผ่านขดลวดนี้ตัดกับสนามแม่เหล็กที่อยู่ด้านบนและล่างระหว่าง VCM ก็จะทำให้เกิดแรงที่ HSA ทำให้ส่วนของ HSA ซึ่งเป็นส่วนที่ติดกับหัวอ่าน/เขียน เคลื่อนที่ได้



ภาพที่ 2.5 วอยซ์คอยมอเตอร์ (VCM)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวเขียนจะเป็นลักษณะแบบ Inductive อันประกอบด้วยแม่เหล็กรูปเกือกม้าที่มีขดลวดพันอยู่รอบ ๆ โดยมีหน้าที่แปลงกระแสที่เข้ามาให้เป็นสนามแม่เหล็ก เพื่อไปจัดเรียงอนุภาคแม่เหล็กของแพลทเทอร์ ส่วนหัวอ่านจะทำหน้าที่แปลงฟลักซ์แม่เหล็กให้เป็นแรงดันไฟฟ้า ปัจจุบันหัวอ่านจะเป็นหัวเทปแม่เหล็กแบบ Giant Magneto-Resistive (GMR) หัวอ่านและเขียนจะมีสมบัติคือ มีค่าสภาพลบล้าง แม่เหล็กต่ำและค่าสภาพให้ซึมผ่านได้ (Permeability) สูง



ภาพที่ 2.6 หัวอ่าน/เขียนฮาร์ดดิสก์ไดรฟ์

ภาพที่ 2.7 ส่วนประกอบของฮาร์ดดิสก์ 2

5. Spindle motor

ส่วนที่เป็นมอเตอร์ใช้ขับให้แผ่นมีเดียหมุน ต้องมีประสิทธิภาพดีไม่ทำให้เกิดความร้อนและเสียงมากเกินไป และสามารถควบคุมความเร็วได้ดี

6. Base

ส่วนที่เป็นกล่องของฮาร์ดดิสก์ไดรฟ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7. Latch

เป็นส่วนที่ป้องกันไม่ให้หัวอ่าน/เขียนชนเข้ากับมอเตอร์และออกนอกเขตพื้นที่การอ่านเขียนข้อมูล

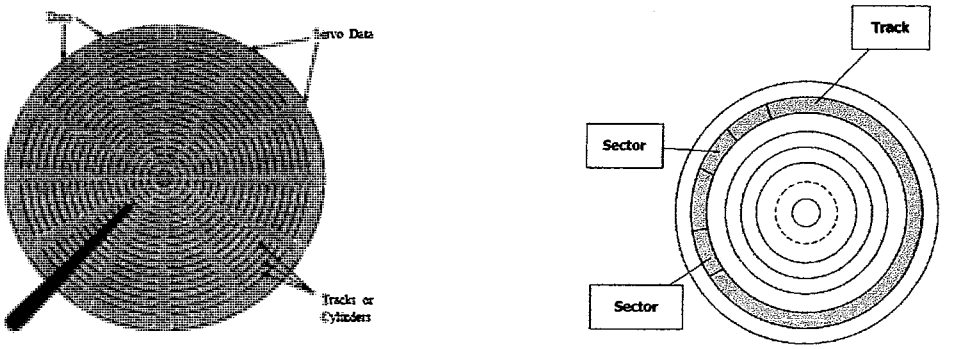
2.1.2 หลักการทำงานของฮาร์ดดิสก์ไดรฟ์

ภายในฮาร์ดดิสก์ไดรฟ์จะมีแผ่น Aluminum Alloy Platter หลายแผ่นหมุนอยู่ด้วยความเร็วสูง โดยจะมีจำนวนแผ่นขึ้นอยู่กับแต่ละรุ่นแต่ละยี่ห้อต่างกันไป การทำงานเขียนอ่านข้อมูลของฮาร์ดดิสก์ไดรฟ์ จะมีการทำงานคล้ายการทำงานของเทปคาสเซ็ท แพลทเทอร์ของฮาร์ดดิสก์ไดรฟ์นั้น จะเคลือบไปด้วยวัตถุจำพวกแม่เหล็ก ที่ขนาดความหนาเพียง 2-3 ในล้านส่วนของนิ้ว แตกต่างจากเทปทั่ว ๆ ไป คือฮาร์ดดิสก์ไดรฟ์นั้น จะใช้หัวอ่านเพียงหัวเดียวในการทำงานทั้งอ่านและเขียนข้อมูล การเขียนข้อมูลลงบนฮาร์ดดิสก์ไดรฟ์นั้น หัวอ่านจะได้รับกระแสไฟฟ้าผ่านเข้าสู่ คอยล์ของหัวอ่าน เพื่อสร้างรูปแบบแม่เหล็กบนสื่อที่เคลือบ อยู่บนแผ่นแพลทเทอร์ ซึ่งเท่ากับเป็นการเขียนข้อมูลลงบนฮาร์ดดิสก์ไดรฟ์

ภาพที่ 2.8 แผ่น Aluminum Alloy Platter

การอ่านนั้นก็จะเป็น การแปลงสัญญาณรูปแบบแม่เหล็ก ที่ได้บันทึกอยู่บนฮาร์ดดิสก์ไดรฟ์กลับแล้วเพิ่มสัญญาณ และทำการประมวลให้กลับมาเป็นข้อมูลอีกครั้ง การเก็บข้อมูลจะเก็บในรูปสัญญาณอิเล็กทรอนิกส์ โดยจะเก็บเป็นเลขฐานสอง คือ 0 และ 1 ฮาร์ดดิสก์ไดรฟ์จะเก็บข้อมูลไว้ใน Track หรือเส้นวงกลม โดยจะเริ่มเก็บข้อมูลด้านนอกสุดของฮาร์ดดิสก์ไดรฟ์ก่อน จากนั้นจึงไล่เข้ามาด้านในสุด โดยฮาร์ดดิสก์ไดรฟ์จะเป็นอุปกรณ์ที่สามารถสุ่มการเข้าถึงข้อมูลได้ นั่นคือหัวอ่านสามารถเคลื่อนที่ไปอ่านข้อมูลบนจุดใดของฮาร์ดดิสก์ไดรฟ์ก็ได้ ฮาร์ดดิสก์ไดรฟ์สามารถเก็บข้อมูลได้ทั้ง 2 ด้านของแพลทเทอร์ ถ้าหัวอ่านเขียนนั้นอยู่ทั้ง 2 ด้าน การเคลื่อนที่ของหัวอ่านเขียนนี้จะมีการเคลื่อนที่ไปพร้อม ๆ กัน โดยจะมีการเคลื่อนที่ที่ตรงกัน Track วงกลมนั้นจะถูกแบ่งออกเป็นหน่วยย่อย ๆ เรียกว่า Sector

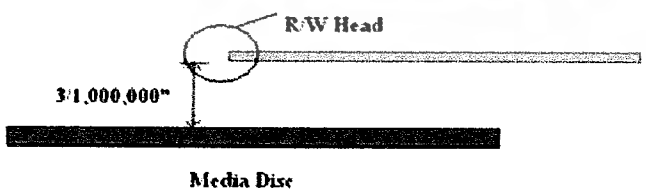
เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 2.9 แบบจำลอง Track และ Sector

การเขียนข้อมูลลงบนฮาร์ดดิสก์ไดรฟ์ เริ่มจากการเขียนข้อมูลที่รอบในสุดของฮาร์ดดิสก์-ไดรฟ์ก่อน จากนั้นเมื่อข้อมูลใน Track ในสุด ถูกเขียนจนเต็ม หัวอ่านก็จะเคลื่อนมายังแทรคถัดมาที่ว่าง แล้วทำการเขียนข้อมูลต่อไป การหมุนของแพลทเทอร์นั้น นับได้ว่าเร็วมาก ความเร็วต่ำสุดจะเท่ากับ 3,600 รอบต่อนาที และปัจจุบันสูงสุดนับหมื่นรอบ ซึ่งเป็นการทำงานที่เร็วกว่าฟลอปปี้ดิสก์หรือเทปมาก ด้วยความเร็วขนาดนี้ทำให้เกิดม่านอากาศบริเวณผิวของแพลทเทอร์ ทำให้หัวอ่าน/เขียนขนาดเล็กสามารถลอยหรือบินอยู่เหนือพื้นผิวได้

หัวอ่านเขียนนั้นได้รับการออกแบบให้บินอยู่เหนือแผ่นแพลทเทอร์ ที่กำลังหมุนด้วยความเร็วสูงนี้ ในความสูงเพียง 3 ในล้านส่วนของนิ้ว หากเกิดการกระแทกอย่างรุนแรงขึ้นกับ ฮาร์ดดิสก์-ไดรฟ์จนทำให้หัวอ่านเขียนสัมผัสกับแพลทเทอร์ จะทำให้พื้นผิวหรือหัวอ่านเขียนเกิดการเสียหาย ซึ่ง จะส่งผลให้เกิดปัญหาข้อมูลเสียหาย หรือถ้าโชคร้ายก็คือฮาร์ดดิสก์ไดรฟ์พัง อย่างไรก็ตาม ปัญหานี้ มักจะไม่เกิดกับฮาร์ดดิสก์ไดรฟ์ในปัจจุบัน ทั้งนี้เพราะฮาร์ดดิสก์ไดรฟ์ในปัจจุบันมีเทคโนโลยีการผลิตที่สูงขึ้น และได้รับการป้องกันเป็นอย่างดี โดยถูกสร้างให้สามารถรับแรงกระแทกได้สูงถึง 70-100 เท่าของแรงดึงดูด (70-100 G)



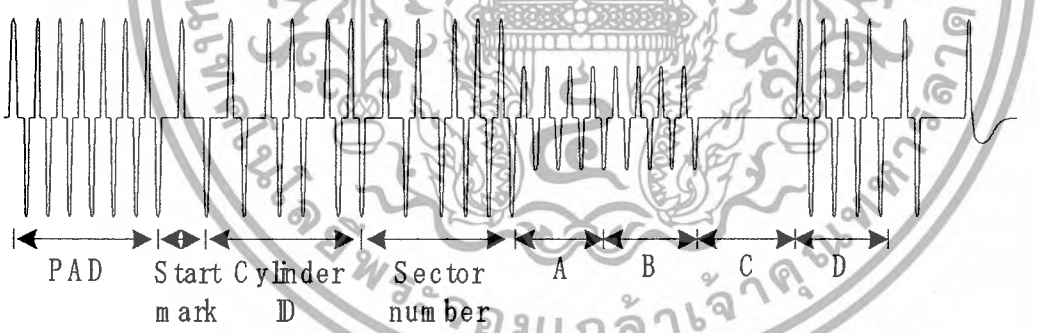
ภาพที่ 2.10 การลอยตัวของหัวอ่าน/เขียนบนแผ่นมีเดียหรือ Platter

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.3 สัญญาณเซอร์โว (Servo)

ในกระบวนการผลิตฮาร์ดดิสก์ไดรฟ์ มีหนึ่งในกระบวนการที่เป็นการเขียนสัญญาณลงบนแผ่นมีเดียเพื่อบอกตำแหน่งให้กับหัวอ่าน/เขียนของฮาร์ดดิสก์ไดรฟ์ ถ้าจะเปรียบเทียบก็คงเปรียบเทียบได้กับถนน บนถนนมีหลักกิโลเมตร และป้ายบอกทางที่คอยบอกว่าอีกกี่กิโลเมตรที่จะถึงตำบลนี้,อำเภอนี้,จะเลี้ยวซ้ายหรือขวา เป็นต้น ถ้าไม่มีหลักกิโลเมตรและป้ายบอกทางเหล่านี้ ผู้ที่เดินทางบนท้องถนนคงต้องหลงทางไปไม่ถึง และเสียเวลามากเพื่อจะเดาทาง เช่นเดียวกันเวลาหัวอ่าน/เขียนในฮาร์ดดิสก์ที่นำมาใช้ในคอมพิวเตอร์หรือในด้านต่างๆ ถ้าไม่มีอะไรบอกตำแหน่งว่าตัวเองอยู่ที่จุดใด และจุดที่จะไปคือจุดใด หัวอ่าน/เขียน ก็จะไม่สามารถไปตรงตำแหน่งนั้นๆที่จะต้องอ่านหรือเขียนข้อมูลลงไป ยิ่งระยะห่างของแทรค (Track) นั้นน้อยมากด้วย การที่หัวอ่าน/เขียน สามารถเคลื่อนย้ายตัวเองไปอยู่บนจุดต่าง ๆ บนแผ่นมีเดียนั้นได้อย่างถูกต้อง และแม่นยำก็เพราะสัญญาณที่เขียนบอกตำแหน่ง หรือ สัญญาณเซอร์โวนั่นเอง

สัญญาณเซอร์โวนี้ เป็นสัญญาณแม่เหล็กที่เขียนมาจากหัวเขียนบนฮาร์ดดิสก์ตัวนั้น ๆ เองเรียกตำแหน่งที่เขียนสัญญาณเซอร์โวนี้ลงไปว่า เซอร์โวเซกเตอร์ (Servo sector) โดยในเครื่องเขียนสัญญาณเซอร์โวจะมีโพสิชันเนอร์ (Rotary Positioner) เป็นตัวลากหัวเขียนไปเขียนสัญญาณ ณ จุดต่างๆบนฮาร์ดดิสก์ ดังนั้น โพสิชันเนอร์ที่ใช้ต้องมีความละเอียดในการหมุนสูงมากคือ 1085 ใน 1,000,000,000 ส่วน หรือถ้าเทียบเป็นระยะทางเชิงเส้นคือ 0.947 นาโนเมตร



ภาพที่ 2.11 แสดงการขยายสัญญาณเซอร์โว

จากภาพที่ 2.11 อธิบายได้ดังนี้

- สัญญาณแพด (Write recovery pad or PAD signal) เป็นสัญญาณที่ยาวที่สุดเท่าที่จะเป็นไปได้ เพราะสัญญาณนี้ใช้ในการปรับ Gain ของฮาร์ดดิสก์ไดรฟ์ Automatic Gain Control เพื่อการปรับอัตราการขยายให้เข้าสู่ค่าที่ดีที่สุด

- จุดเริ่มต้นของสัญญาณเซอร์โว (Servo Start Mark or SSM) เป็นรูปแบบพิเศษเพื่อใช้ในการระบุ บ่งบอกการเริ่มต้นของรูปแบบสัญญาณเซอร์โว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- สัญญาณไซเลนเดอร์ไอดี เป็นส่วนที่อ้างอิงถึงและบันทึกตำแหน่งของไซเลนเดอร์ไอดี
- เซกเตอร์ไอดี (Sector ID) เป็นส่วนที่อ้างอิงถึงและบันทึกตำแหน่งของเซกเตอร์ไอดี
- เซอร์โวเบิร์ส (Servo burst) รูปแบบพิเศษนี้จะบ่งบอกถึงจุดกึ่งกลางของแทรค แอมปลิจูดของเอและบี จะถูกทำให้สมดุลกันเมื่อหัวอ่าน/เขียนกำลังกระทำอยู่บริเวณจุดกึ่งกลางของแทรค แอมปลิจูดของซีและดี จะถูกทำให้สมดุลเมื่อ อยู่บริเวณเส้นขอบของแทรค
- เซอร์โวเอ็นแพด (Servo End PAD) เป็นสัญญาณปิดท้าย เพื่อแบ่งระหว่างเซอร์โวเซกเตอร์กับส่วนของข้อมูล

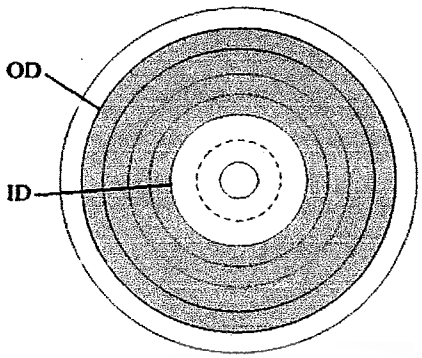
2.1.4 กระบวนการเขียนสัญญาณเซอร์โว และระบบควบคุมตำแหน่งของ Rotary Positioner

กระบวนการเขียนสัญญาณเซอร์โวคือ เมื่อวางฮาร์ดดิสก์ไดรฟ์ลงบนเครื่องเขียนสัญญาณเซอร์โวและสั่งให้เครื่องทำงานในโหมดอัตโนมัติ เครื่องจะเริ่มสตาร์ทให้มอเตอร์หมุนและคำนวณค่าเริ่มต้นต่างๆของฮาร์ดดิสก์ไดรฟ์ตัวนั้นๆ

1. คำนวณค่าความต้านทานของ VCM ซึ่งฮาร์ดดิสก์แต่ละตัวจะมีค่านี้ไม่เท่ากัน
2. อ่านค่าความเร็วที่มอเตอร์หมุนและปรับให้ความเร็วอยู่ที่ 4800 rpm
3. คำนวณและหาตำแหน่งจุดเริ่มต้นด้านในสุด (ID) และด้านนอกสุด (OD) ฮาร์ดดิสก์-ไดรฟ์แต่ละตัวจะมี OD และ ID ในตำแหน่งที่ไม่เหมือนกัน เป็นผลพวงจากขั้นตอนกระบวนการผลิตแผ่นมีเดีย
4. กระแสไบอัสถูกจ่ายไปยัง VCM เพื่อให้เกิดแรงบิดในทิศทวนเข็มนาฬิกา Rotary Positioner ก็เช่นเดียวกันแต่ในทิศตามเข็มนาฬิกา และมีแรงบิดมากกว่าจึงลากหัวเขียนไปยังจุดเริ่มต้น (OD) ได้
5. เริ่มเขียนสัญญาณนาฬิกาที่ OD เพื่อเป็นสัญญาณอ้างอิง
6. กระแสไบอัสถูกจ่ายไปยัง VCM และโพสิชันเนอร์อีกครั้งเพื่อเริ่มเขียนสัญญาณ แต่เครื่องเขียนสัญญาณจะเคลื่อนที่ทวนเข็มเช่นเดียวกับ VCM เพื่อประคองไม่ให้หัวอ่านเคลื่อนที่ออกนอกตำแหน่งที่ควรจะเป็น

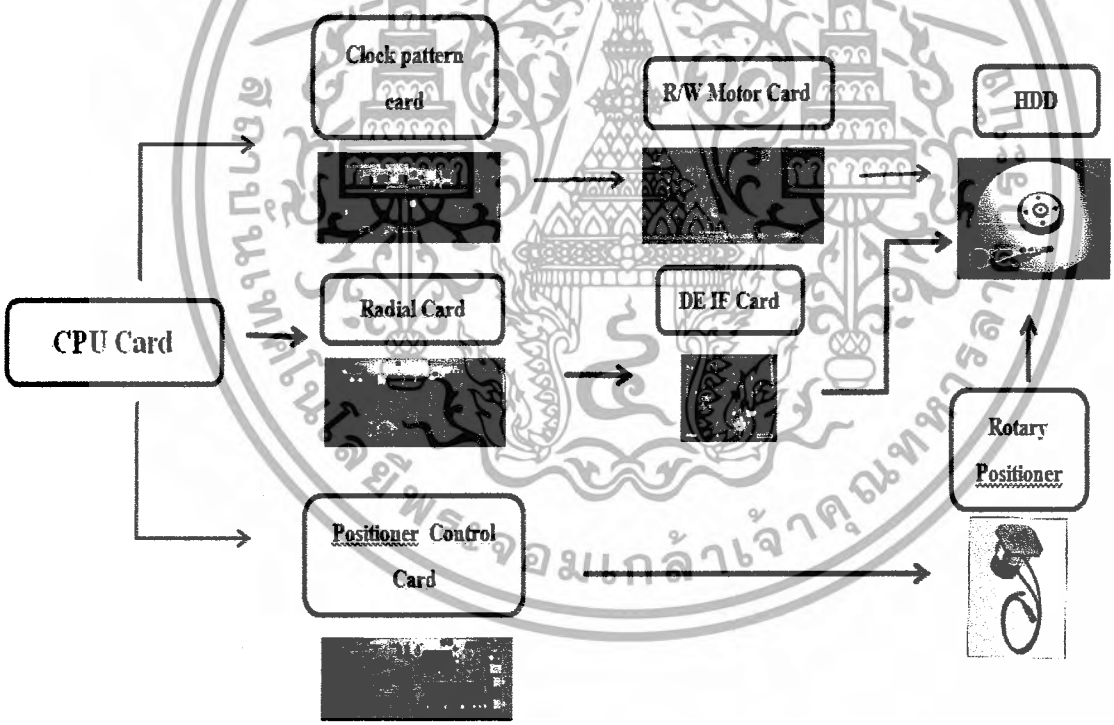
เมื่อเริ่มต้นการเขียน R/W motor Card จะสั่งให้มอร์เตอร์ในฮาร์ดดิสก์หมุนให้ได้ความเร็วที่ต้องการ ในที่นี้คือ 5400 rpm จากนั้น Radial Card จะเป็นส่วนที่กำหนดจุดเริ่มต้นที่หัวอ่าน/เขียนควรจะอยู่ โดยโพสิชันเนอร์จะถูกสั่งจาก Positioner Control Card เพื่อลากหัวอ่าน/เขียน ไปยังตำแหน่งเริ่มต้นดังกล่าว จากนั้น Clock Pattern Card จะให้จังหวะในการเขียนโดยสั่งผ่าน R/W motor Card และผ่านไปยังหัวเขียน จากนั้นโพสิชันเนอร์จะเริ่มลากหัวเขียนโดยความเร็วในการเคลื่อนที่ของโพสิชันเนอร์นั้นจะถูกควบคุมด้วย Positioner Control Card

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 2.12 แสดง OD และ ID

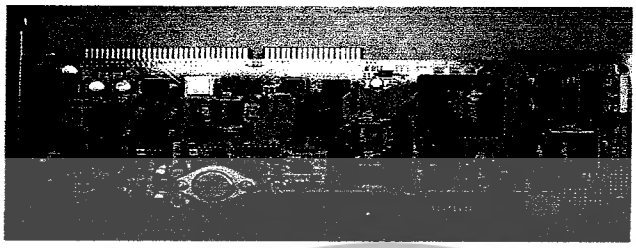
กระบวนเขียนสัญญาณเซอร์โว ประกอบด้วยส่วนต่างๆ ดังนี้



ภาพที่ 2.13 กระบวนเขียนสัญญาณเซอร์โว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- Radial Card เป็นการ์ดที่กำหนดจุดเริ่มต้นในการเขียนสัญญาณเซอร์โวและเป็นส่วนที่ใช้กำหนดตำแหน่งของหัวอ่าน/เขียน เพื่อให้ได้ตำแหน่งที่ถูกต้อง



ภาพที่ 2.14 Radial Card

คุณสมบัติ

1. ติดต่อกับ signal board computer (SBC) ผ่านสาย AT bus
2. DSP 32 bit
3. 12 bit เป็น ADC (A to D) สำหรับ ended signal และ diff. signal
4. 12 bit รับ DAC (D to A) แบบอนุกรม to Control high output amplifier
5. 8 bit แบบขนาน เพื่อใช้ DAC สำหรับ AGC control signal
6. ขับ VCM ด้วยกระแสขาออก (high output amplifier)
7. อ่านสัญญาณกลับ (EMF signal) จาก VCM ด้วย Diff. amp.

- R/W motor Card เป็นการ์ดที่รับสัญญาณต่อจาก Clock Pattern Card เพื่อเขียนสัญญาณเซอร์โวลงบนแผ่นมีเดีย และเป็นการ์ดที่ใช้ในการไดร์ฟให้มอเตอร์ในฮาร์ดดิสก์หมุน (ขับ spindle motor)

ภาพที่ 2.15 R/W motor Card

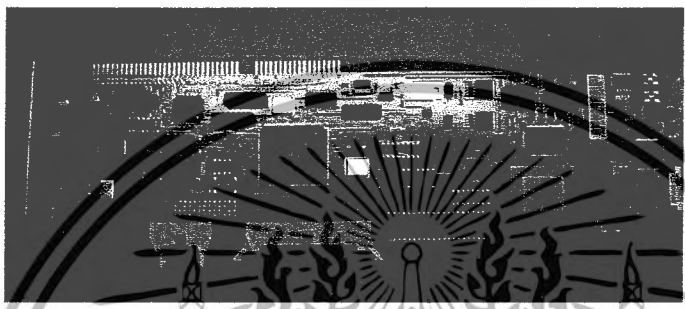
คุณสมบัติ

1. ประมวลผลแบบ DSP

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. ขับ spindle motor 3 เฟสด้วย สัญญาณคลื่น
3. clock pattern ได้จากตัวเอง เพื่อเขียน gate and data signal
4. สร้างข้อมูลแบบอนุกรมสำหรับ AE

- Clock Pattern Card วัดช่วงสัญญาณ read-out และสร้าง servo pattern + timing เป็นการ์ดที่ใช้สร้างจังหวะในการสั่งให้หัวเขียน เขียนสัญญาณลงบนแผ่นมีเดีย

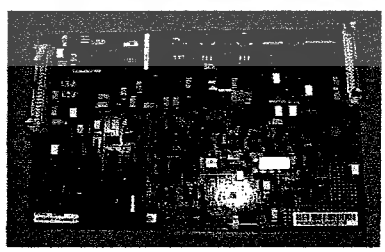


ภาพที่ 2.16 Clock Pattern Card

คุณสมบัติ

1. ประมวลผลแบบ DSP
2. ช่วงเวลาการวัด วัดด้วย digital counter และ เวอร์เนีย circuit system
3. สร้าง timing ด้วย Direct digital synthesis
4. programmable pattern generator

- DE IF Card เป็นการ์ดที่เชื่อมการทำงานระหว่าง Radial Card และฮาร์ดดิสก์ไดรฟ์ ใช้ต่อร่วมกับ SSW controller card ,Clock Pattern , Radial and R/W motor



ภาพที่ 2.17 DE IF Card

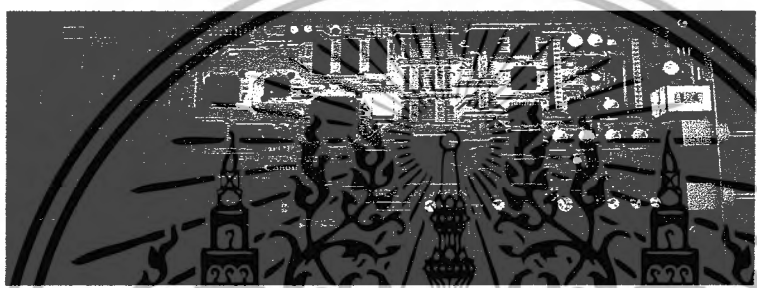
คุณสมบัติ

1. ควบคุม Gain สำหรับสัญญาณ read-out

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. ตรวจจับสัญญาณ timing mark จากสัญญาณ read-out
3. ตรวจจับสัญญาณ envelop จากสัญญาณ read-out
4. module จ่ายกำลัง แบบ enable/disable
5. รับส่งข้อมูลแบบอนุกรม ด้วย read-back function

- Positioner Control Card (System Controller: SC-xxH/PC) ควบคุมโพซิชันเนอร์ด้วยคำสั่งจากคอมพิวเตอร์ ผ่าน ISA bus เป็นการ์ดที่สั่งขับโพสิชันเนอร์ให้เคลื่อนที่เพื่อลากหัวเขียนไปเขียนสัญญาณลงบนฮาร์ดดิสก์ไดรฟ์



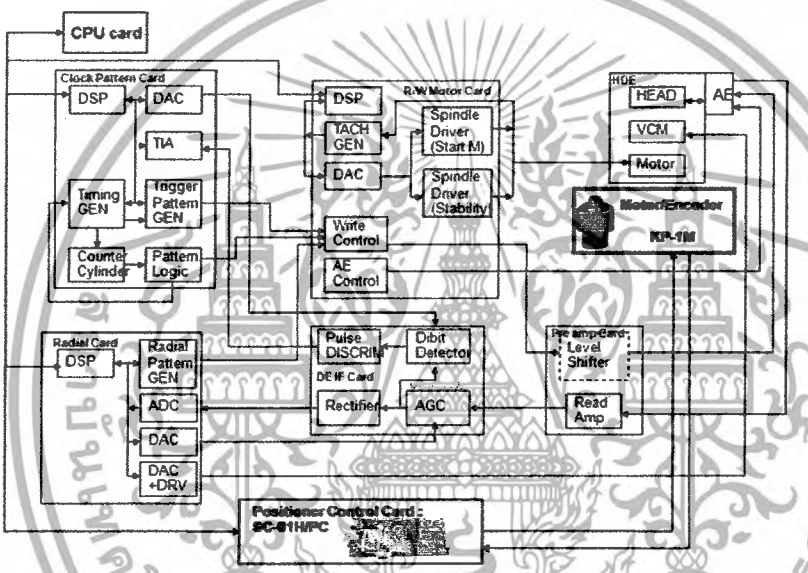
ภาพที่ 2.18 Positioner Control Card

คุณสมบัติ

1. ทำงานผ่าน ISA bus
2. DSP 150 MHz
3. sampling น้อยกว่า 50kHz
4. A to D converter 14 bit
5. interpolation 4096
6. Power supply + 5 V ± 5 %
   (max 1.5 A) + 12 V ± 5 %
   (max 1.5 A) - 12 V ± 5 % (max 100 mA)
7. stroke ±20 องศา
8. ความละเอียด 0.003906 arc-sec
9. ความแม่นยำบนแทร็ก ± 0.05 arc-sec
10. selting time น้อยกว่า 8 msec
11. หมุนเร็วสุด 60 องศา/วินาที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Positioner Control Card จะทำหน้าที่ เป็นตัวควบคุมที่จะสร้างสัญญาณควบคุมที่เหมาะสมส่งให้กับชุดขับและให้สัญญาณควบคุมเอาท์พุทส่งไปยัง Rotary Positioner เพื่อขับเคลื่อนตัว pusher ให้พาหัวอ่าน/เขียน เคลื่อนที่ไป โดยระยะการเคลื่อนที่เชิงมุมที่เกิดขึ้น จะถูกวัดและป้อนกลับจาก encoder เข้าสู่ชุดปรับแต่งสัญญาณ ภายในบอร์ดและส่งผ่านไปยังตัวควบคุม เพื่อคำนวณหาค่าสัญญาณควบคุมที่เหมาะสม ที่สามารถทำให้หัวอ่าน/เขียน อยู่ในตำแหน่งที่ต้องการซึ่งจะรับคำสั่งการทำงาน ค่าพารามิเตอร์ ในการควบคุม จาก PC ผ่านทาง execute software s.exe และไฟล์ที่ทำหน้าที่กำหนด ค่าพารามิเตอร์ SCP1.txt และเมื่อพารามิเตอร์สำหรับ การควบคุมถูกส่งผ่านไปยัง DSP โมดูลบนบอร์ด Positioner Control Card (System Controller: SC-xxH/PC)



ภาพที่ 2.19 วงรอบการควบคุมตำแหน่งของ Rotary Positioner

การควบคุมตำแหน่ง (เชิงมุม) ของ Rotary Positioner ให้อยู่ในตำแหน่งที่ต้องการได้อย่างเที่ยงตรงและแม่นยำหรือไม่นั้น ขึ้นอยู่กับหลายปัจจัยด้วยกัน ทั้งในส่วนความสามารถของตัวควบคุมสภาวะของกระบวนการ ณ จุดทำงาน และสัญญาณรบกวนที่เกิดขึ้นกับระบบขณะทำงานซึ่งอาจมีสาเหตุมาจากสิ่งรบกวนภายในระบบ (Internal disturbances) และจากสิ่งรบกวนภายนอก (External disturbances) เช่น disk rotations (bearing RRO/NRRO , windage) , Reaction forces , Noise (PES) , shock vibration ด้วยปัจจัยของ สัญญาณรบกวนต่างๆ ที่กล่าวมานี้ อาจส่งผลต่อกระบวนการเขียนสัญญาณเซอร์โวในส่วนของการควบคุมการเคลื่อนที่ของ Rotary Positioner ให้ไม่สามารถขับเคลื่อนหัวอ่านให้อยู่ในตำแหน่งที่ต้องการได้ภายในค่าเวลาคงที่ที่กำหนด โดยผลของความผิดพลาดที่เกิดขึ้นแสดงให้เห็นในรูปของ error code

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ตารางที่ 2.1 แสดง error code ของ Rotary Positioner

Main	Sub	Description
0	9999	Pass
9965	3301	Timing Error Too Many Retry
9965	3302	Timing Error Unexpected Missings or Extras Overflow
9965	3304	Timing Error Slip Overflow
9920	701	Positioner Out Position Overflow
9920	702	Positioner Seek Error

ตารางที่ 2.1 (ต่อ) แสดง error code ของ Rotary Positioner

Main	Sub	Description
9920	703	Positioner H2 Time Out
9920	705	Positioner Servo Off Error
9920	706	Positioner Parameter Not Found
9920	708	Positioner Status Error
9920	709	Positioner Inposition Over Time
9920	710	Positioner Alarm
9920	712	Positioner Program Version Error
9920	800	Positioner Hit DE at Clamp
9964	3307	Timing Error Invalid Count Overflow
9964	3308	Timing Error Index Delay Too Big
9964	3309	Timing Error Velocity

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
104013

ระบบควบคุมให้หัวอ่านอยู่บนแทรค (Track) ใดๆ

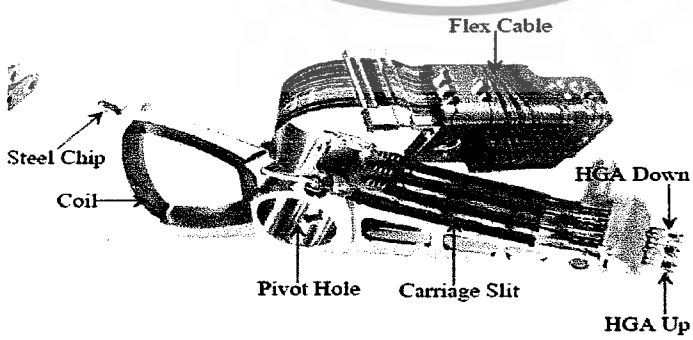
จะใช้ Voice coil motor ที่มีระบบนำร่อง ที่ชื่อว่า Servo เพื่อบอกตำแหน่งที่ถูกต้องของหัวอ่านว่าอยู่กลาง Track หรือไม่ VCM จะปรับตำแหน่งของหัวอ่านให้ถูกต้องถ้าระบบ Servo ระบุว่าหัวอ่านไม่อยู่กลาง Track วิธีการนี้มีชื่อเรียกทั่วไปว่า Servo – controlled mechanim

2.1.5 เครื่องเขียนสัญญาณเซอร์โว หรือ NCH (Non Clock Head)

ภาพที่ 2.20 NCH (Non Clock Head)

ใช้เพื่อเขียนสัญญาณเซอร์โวลงบนฮาร์ดดิสก์ไดรฟ์ ปัจจัยที่ต้องควบคุมขณะทำการเขียนสัญญาณ แยกเป็น 2 ส่วนที่สำคัญ คือ

1. Head Stack Assembly (HSA) ที่ทำการเขียนสัญญาณลงบนฮาร์ดดิสก์ ณ ตำแหน่งแทรคต่างๆ

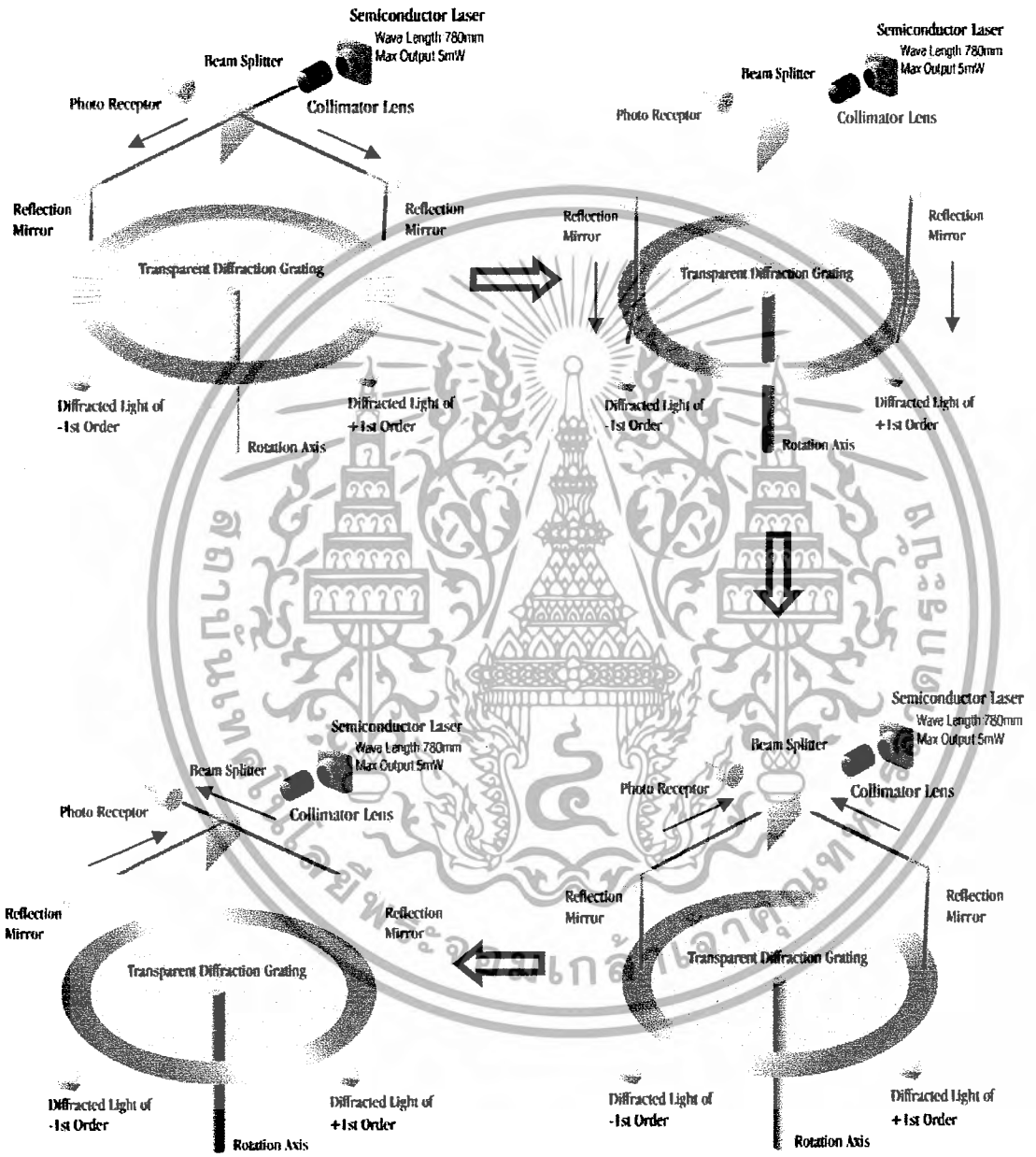


ภาพที่ 2.21 Head Stack Assembly (HSA)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. ช่วงเวลา (Timing) ของสัญญาณเซอร์โวที่จะถูกเขียนลงในแทรคทุก ๆ แทรค

2.1.5.1 เอ็นโค้ดเดอร์แบบเลเซอร์โรตารี่ (Laser Rotary Encoder)



ภาพที่ 2.22 หลักการทำงานของเอ็นโค้ดเดอร์แบบเลเซอร์โรตารี่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Rotary Positioner เป็นตัวช่วยควบคุมตำแหน่งการเขียนสัญญาณเซอร์โวโดยใช้เอ็นโค้ดเดอร์แบบเลเซอร์โรตารี่ จากภาพที่ 2.22 เป็นหลักการทำงานของเอ็นโค้ดเดอร์แบบเลเซอร์โรตารี่ โดยมีหลักการทำงานคือแสงเลเซอร์จะถูกยิงจาก Semiconductor Laser ผ่านCollimator Lens ซึ่งจะเป็นผลให้แสงมีขนาดเล็ก จากนั้น Beam Splitter จะทำการแบ่งลำแสงออกเป็น 2 ลำแสงในระยะทางที่เท่ากันและไปตกกระทบที่ Reflection Mirror ผ่านไปยังดิสก์กระจายแสง (Transperent diffraction Grating Disc) และตัวสะท้อนแสงกลับของอนุพันธ์อันดับหนึ่งทางด้านบวกและทางด้านลบ (Diffracted Light of ±1st Order)

การเคลื่อนที่ของมุมใน 1 รอบจะเท่ากับ $\pm 2\pi$ จากนั้นลำแสงจะสะท้อนกลับไปยังดิสก์กระจายแสงสู่ Reflection Mirror ซึ่ง Reflection Mirror จะทำหน้าที่รวมลำแสงทั้ง 2 ลำแสงจากทางด้านบวกและลบ ส่งกลับไป ณ ที่ตัวรับแสง (Photo Receptor) ทำให้มุมหมุนไปเท่ากับ $\pm 4\pi$ เป็นผลให้เอ็นโค้ดเดอร์แบบเลเซอร์โรตารี่ มีความแม่นยำในการควบคุมตำแหน่งสูงขึ้น

2.1.6 วิธีการตรวจสอบสเต็ปการเคลื่อนที่ของหัวอ่าน/เขียน

สัญญาณเซอร์โวที่เขียนลงบนแผ่นมีเดียนั้นถ้าเป็นสัญญาณที่ดี จะต้องไม่มีการซ้อนทับกันของสัญญาณ เช่น สัญญาณที่อยู่บนแทรคที่ 1 ต้องไม่ไปซ้อนทับกับสัญญาณในแทรคที่ 2 วิธีการตรวจสอบก็คือ ต้องใช้วิธีการทดสอบที่ชื่อว่า APC Test ซึ่งเป็นวิธีที่ใช้หัวอ่านไปอ่านสัญญาณเบิร์ส (Burst) ซึ่งจะถูกเขียนลงบนแผ่นมีเดียเมื่อใช้วิธีการทดสอบนี้ โดยแถบสัญญาณเบิร์สนี้จะบอกถึงตำแหน่งที่สัญญาณเซอร์โวควรจะอยู่ ถ้าหัวอ่านอ่านค่าที่ได้ซึ่งคำนวณได้จากสมการที่ (1) จะได้ค่า V_{APC} ออกมา ซึ่งค่าที่ได้ควรจะเป็น คือ 1 ซึ่งค่า V_{APC} นั้นจะนำไปคำนวณหาค่าระยะห่างระหว่างแทรค (T_P, Track Pitch) ต่อไป ดังสมการที่ (2) และสามารถหาค่า Positioner Error Signal ได้ดังสมการที่ (3)

สูตรที่ใช้ในการคำนวณ

สมการที่ (1)

$$V_{APC} = \frac{V_A+V_C}{V_B} = \frac{A+C}{B}$$

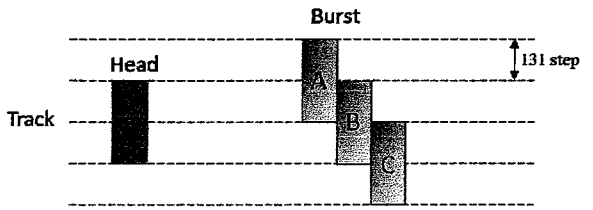
สมการที่ (3)

$$T_P = W_w+W_r(1\text{-}APC)$$

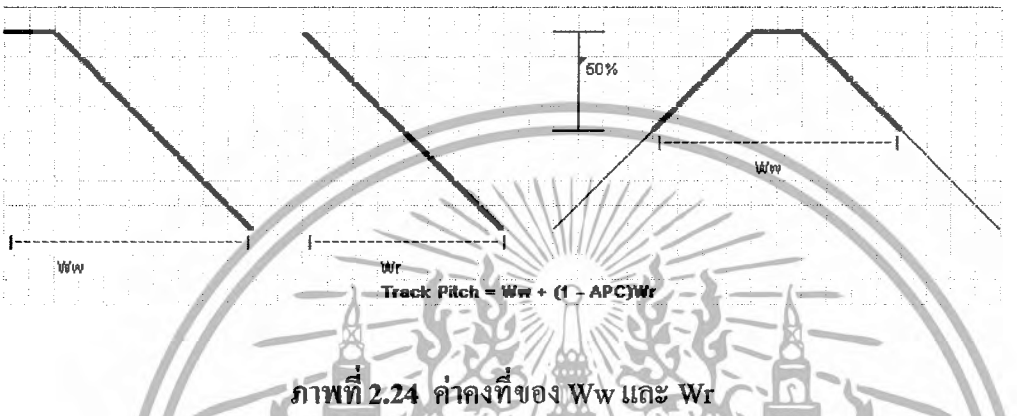
สมการที่ (3)

$$PES = \frac{\frac{1}{2}(V_A+V_B)}{2V_B+V_A+V_C}$$

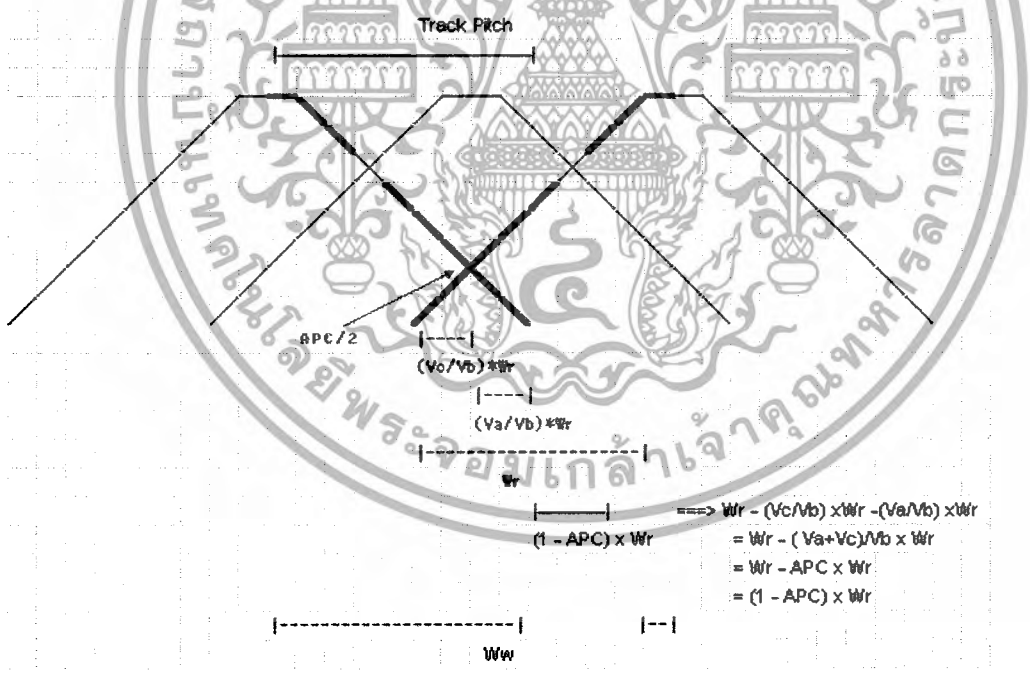
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 2.23 ความสัมพันธ์ระหว่างตำแหน่งของหัวและสัญญาณ APC



ภาพที่ 2.24 ค่าคงที่ของ Ww และ Wr



ภาพที่ 2.25 การคำนวณ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2 ความรู้เกี่ยวกับ ARM7

2.2.1 สถาปัตยกรรมและคุณสมบัติของ ARM

Arm เป็นไมโครโปรเซสเซอร์ที่ออกแบบโดยแนวความคิด RISC (Reduce Instruction Set Computer) หมายความว่าหน่วยประมวลผลที่ถูกออกแบบ โดยแนวความคิดที่จะทำให้สามารถลดจำนวนคำสั่งให้เหลือน้อยที่สุด เพื่อให้เกิดประมวลที่มีประสิทธิภาพ และรวดเร็วเพราะกำหนดให้ตัวประมวลผลแบบจุลภาค (microprocessor) แต่ละตัวทำงานจำกัดเฉพาะบางอย่าง โดยทั่วไปมีคุณสมบัติดังนี้

- จะต้องมีรีจิสเตอร์ให้สามารถใช้งานได้มากเพียงพอ
- ใช้รูปแบบคำสั่งแบบ load and store หมายความว่า ก่อนการประมวลผลจะต้องทำการโหลดข้อมูลจากรีจิสเตอร์หรือหน่วย ความจำก่อน หลังจากนั้นจึงทำการประมวลผลซึ่งผลลัพธ์ที่ได้อาจจะเก็บอยู่ในรีจิสเตอร์เพื่อไปสู่กระบวนการถัดไปหรืออาจจะบันทึกกลับลงสู่หน่วยความจำ
- มีโหมดการอ้างตำแหน่งแอดเดรสที่ไม่มากนัก โดยสามารถกระทำได้โดยอ้างตำแหน่งจากรีจิสเตอร์และคำสั่งเพียงคำสั่งเดียวเท่านั้น
- ขนาดของคำสั่งมีขนาดคงที่ (32 บิต)

สำหรับ ARM7 แล้วได้เพิ่มความสามารถเพิ่มเติม คือ

- มี shifter หรือตัวเลื่อนบิตแยกต่างหากก่อนที่จะเข้า ALU (Arithmetic Logic Unit) เพื่อเพิ่มความเร็วในการประมวลผล
- มีโหมดการอ้างตำแหน่งแบบ Auto-increment และ Auto-decrement เพื่อช่วยให้การทำงานแบบลูปง่ายขึ้น
- สามารถเพิ่มเงื่อนไขภายในแต่ละคำสั่งเพื่อลดการสูญเสียจากการทำงานของ pipeline ซึ่งจะส่งผลให้การทำงานเร็วขึ้น

2.2.2 ARM REGISTERS

ARM มีรีจิสเตอร์ทั้งหมด 37 ตัวเป็นแบบ 32 บิต แบ่งเป็น

รีจิสเตอร์ใช้งานทั่วไป 30 ตัว

รีจิสเตอร์แสดงสถานะ 6 ตัว

program counter 1 ตัว

ผู้ใช้สามารถใช้ได้ทั้งหมด 16 ตัว ตัวอื่น ๆ มีเพื่อเพิ่มประสิทธิภาพในการทำเอกเซปชั่น (exception processing) พอสรุปได้ดังนี้

- Stack pointer รีจีสเตอร์ R13 ทำหน้าที่เก็บข้อมูลชั่วคราว เช่น เวลาจะเรียกโปรแกรมย่อยเราจะทำการบันทึกค่าของรีจิสเตอร์เก็บไว้ในแสตก เพื่อป้องกันการเปลี่ยนค่าในรีจิสเตอร์ขณะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประมวลผลโปรแกรมย่อยนั้น ดังนั้นเมื่อออกจากโปรแกรมย่อยก็จะนำค่าในแสตกคืนมาที่รีจิสเตอร์ใหม่

- Link register รีจีสเตอร์ R14 ทำหน้าที่ในการเก็บค่าที่อยู่ของคำสั่งถัดไปหลังจากคำสั่ง Branch with Link (BL) ที่ใช้เวลาเรียกโปรแกรมย่อยของ ARM

- Program counter รีจีสเตอร์ R15 ทำหน้าที่เป็น Program Counter ซึ่งใช้กับคำสั่งทุกคำสั่งในการชี้ไปที่คำสั่งนั้น ใน ARM คำสั่งหนึ่งๆจะมีขนาด 32 บิต

- Other registers รีจีสเตอร์ที่เหลือไม่มีหน้าที่พิเศษโดยเฉพาะ

เมื่อเกิดเอกเซปชั่น รีจีสเตอร์บางตัวจะถูกกำหนดให้เป็นรีจีสเตอร์ ซึ่งเก็บค่าเฉพาะของเอกเซปชั่น ในทุกๆโหมดค่า R13 และ R14 เป็นของตัวเอง โดย R13 เก็บ stack pointer และ R14 เก็บ link register

- SPSR (Saved Program Status Register) เก็บค่าของ CPSR ก่อนจะเกิดเอกเซปชั่น ซึ่งในแต่ละโหมดจะมีรีจีสเตอร์ SPSR เป็นของตัวเองโดยใช้ CPSR ร่วมกัน

2.2.3 คุณสมบัติข้อเด่นของ ARM7

- มีความเร็วการประมวลผลสูงที่ 45 MIPS
- ใช้เวลาการคูณแบบ 32*8 บิตโดยใช้เวลา 2 ไซเคิล แบบ 32*32 5 ไซเคิล
- ใช้เวลาในการสำเนาข้อมูล 1kb block นาน 1.1 ms
- ใช้เวลาในการเข้าถึงหน่วยความจำ Flash นาน Erase : 20ms , Write : 20us
- มีความเที่ยงตรงสูง

2.2.4 ไมโครคอนโทรลเลอร์ตระกูล ADUc ของ Analog Devices

ไมโครคอนโทรลเลอร์ตระกูล ADUc7xxx ของ Analog Device เป็น MCU แบบ ARM7 core มีจุดเด่นอยู่สามส่วนหลักคือ มีส่วนเชื่อมสัญญาณแบบ Analog แบบหลายช่องทางที่มีประสิทธิภาพสูง และมีส่วนของไมโครคอนโทรลเลอร์ที่ได้มาตราฐานอุตสาหกรรม แผนกลยุทธ์การพัฒนาหน่วยประมวลผลหลักของไมโครคอนโทรลเลอร์ ตระกูล ADUc702x อยู่ในทิศทางของการพัฒนาเพื่อตอบสนองความต้องการใช้งานไมโครคอนโทรลเลอร์ที่มีประสิทธิภาพสูง และใช้พลังงานต่ำ

สำหรับไมโครคอนโทรลเลอร์ตระกูล ADUc702x ถูกสร้างขึ้นโดยใช้ ARM7TDMI MCU core มีสถาปัตยกรรมแบบ 16/32 บิต RISC , 48Hz (Flash & SRAM), Embeded JTAG สำหรับการดีบัก , วงจร PMWs แบบสามเฟส , ADC I/Ps LowDrift Vref ที่ยืดหยุ่นกว่า มี Programmable Logic Array (PLA) และถูกออกแบบให้มีขนาดเล็ก มีสถานะ ในราคาที่ย่อมเยาว์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.4.1 การทำงานแบบ 16/32 บิต ของARM และการทำงานโดยใช้ชุดคำสั่งแบบ Thumb

- ชุดคำสั่งมีความยาวที่ 32 บิต
- ในคำสั่งเดียวสามารถทำให้อยู่ในรูปของคำสั่งแบบ THUMB หรือคำสั่งในโหมด ARM ก็ได้
- มีความเร็วในการเอ็กซิคิวท์จากหน่วยความจำแบบ Flash/EE (CD>0) เทียบเท่ากับ SRAM (CD>=0)
- มีการเขียนตารางเวคเตอร์

2.2.4.2 อินเทอร์รัฟท์

ตารางเวกเตอร์ตั้งอยู่ที่แอดเดรส 0x00

1. มีระดับความสำคัญอยู่สองระดับ
   IRQ : เป็นการอินเทอร์รัฟท์ในในระดับมาตรฐาน
   FIQ : เป็นการอินเทอร์รัฟท์แบบ Fast IRQ (มีความสำคัญมากกว่า)
2. มีที่มาของการอินเทอร์รัฟท์ 23 แห่งไม่ขึ้นต่อกัน
   - อาทิ เช่น จาก ADC , PWM , PSM , พอร์ตสื่อสารอนุกรม
3. Flash/EE หรือ RAM ภายใน
4. เปิดใช้งานอินเทอร์รัฟท์ผ่านทาง IRQEN และ FIQEN เมมโมรี่แมปรีจิสเตอร์
   - 29 ไซเคิลจาก RAM
   - 34 ไซเคิลจาก Flash

2.2.5 อุปกรณ์เชื่อมต่ออินพุทเอาท์พุทแบบอนุกรมของไมโครคอนโทรลเลอร์ตระกูล ADUc702x

มีอุปกรณ์เชื่อมต่ออินพุทเอาท์พุทแบบอนุกรมที่ยืดหยุ่น (10 GPIOs) ซึ่งประกอบด้วย 3 ชนิดด้วย 3 ชนิด ได้แก่

1. Standard Serial Peripheral Interface (SPI)
2. Standard UART บวกกับคุณสมบัติที่เพิ่งเข้ามา คือ ตัวหารแบบมีเศษและโหมดอ้างอิงแอดเดรสแบบเน็ตเวิร์ค
3. สนับสนุนพอร์ต I^2C ถึง 2 พอร์ต โดยอุปกรณ์ฮาร์ดแวร์สามารถเป็นได้ทั้งมาสเตอร์และสเลฟ และ คุณสมบัติพิเศษสำหรับงานประยุกต์ออฟติคอล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.5.1 การติดต่อพอร์ตและการควบคุมการทำงานของพอร์ตอเนกประสงค์ (General Purpose I/O : GPIO)

พอร์ตอเนกประสงค์ (General Purpose input/output) ในไมโครคอนโทรลเลอร์ ADuC7024 เป็นพอร์ตอเนกประสงค์สำหรับใช้งานเป็นอินพุท เอาท์พุทได้ใน การจะกำหนดได้ว่าพอร์ตไหนเป็นเอาท์พุทและพอร์ตไหนเป็นอินพุท กำหนดได้จากรีจิสเตอร์ดังตารางที่ 2.2

ตารางที่ 2.2 GPIO Pin Function Descriptions

Port	Pin	Configuration			
		00	01	10	11
0	P0.0	GPIO	CMP	MS2	PLA[7]
	P0.1	GPIO	PWM2H	BLE	
	P0.2	GPIO	PWM2L	BHE	
	P0.3	GPIO	TRUST	A16	ADCbusy
	P0.4	GPIO/IRQ0	PWMtmp	MS1	PLAO[1]
	P0.5	GPIO/IRQ1	ADCbusy	MS0	PLAO[2]
	P0.6	GPIO/T1	MRST	AE	PLAO[3]
	P0.7	GPIO	ECLK/XCLK'	SIN	PLAO[4]
1	P1.0	GPIO/T1	SIN	SCLO	PLA[0]
	P1.1	GPIO	SOUT	SDAO	PLA[1]
	P1.2	GPIO	RTS	SCL1	PLA[2]
	P1.3	GPIO	CTS	SDA1	PLA[3]
	P1.4	GPIO/IRQ2	RI	CLK	PLA[4]
	P1.5	GPIO/IRQ3	DCD	MISO	PLA[5]
	P1.6	GPIO	DSR	MOSI	PLA[6]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.2 (ต่อ) GPIO Pin Function Descriptions

Port	Pin	Configuration			
		00	01	10	11
2	P2.0	GPIO	CONVstart	SOUT	PLAO[5]
	P2.1	GPIO	PWMOH	WS	PLAO[6]
	P2.2	GPIO	PWMOL	AS	PLAO[7]
	P2.3	GPIO		AE	
	P2.4	GPIO	PWMOH	MS0	
	P2.5	GPIO	PWMOL	MS1	
	P2.6	GPIO	PWM1H	MS2	
	P2.7	GPIO	PWM1L	MS3	
3	P3.0	GPIO	PWMOH	AD0	PLA[8]
	P3.2	GPIO	PWM1H	AD2	PLA[10]
	P3.3	GPIO	PWM1L	AD3	PLA[11]
	P3.4	GPIO	PWM2H	AD4	PLA[12]
	P3.5	GPIO	PWM2L	AD5	PLA[13]
	P3.6	GPIO	PWMtmp	AD6	PLA[14]
	P3.7	GPIO	PWMsytic	AD7	PLA[15]
4	P4.0	GPIO		AD8	PLA[8]
	P4.1	GPIO		AD9	PLA[9]
	P4.2	GPIO		AD10	PLA[10]
	P4.3	GPIO		AD11	PLA[11]
	P4.4	GPIO		AD12	PLA[12]
	P4.5	GPIO		AD13	PLA[13]
	P4.6	GPIO		AD14	PLA[14]
	P4.7	GPIO		AD15	PLA[15]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.3 GPxCON Registers

Name	Address	Default Value	Access
GP0CON	0xFFFFF400	0x00000000	R/W
GP1CON	0xFFFFF404	0x00000000	R/W
GP2CON	0xFFFFF408	0x00000000	R/W
GP3CON	0xFFFFF40C	0x00000000	R/W
GP4CON	0xFFFFF410	0x00000000	R/W

ตารางที่ 2.4 GPxCON MMR Bit Descriptions

Bit	Description
31:30	Reservde
29:28	Select Function of Px.7Pin
27:26	Reserved
25:24	Select Function of Px.6Pin
23:22	Reserved
21:20	Select Function of Px.5Pin
19:18	Reserved
17:16	Select Function of Px.4Pin
15:14	Reserved
13:12	Select Function of Px.3Pin
11:10	Reserved
9:8	Select Function of Px.2Pin
7:6	Reserved
5:4	Select Function of Px.1Pin
3:2	Reserved
1:0	Select Function of Px.0Pin

ค่าของบิทในรีจิสเตอร์ GPxCON จะมีความหมายในการกำหนดหน้าที่การทำงานของแต่ล่ะพอร์ตอยู่การที่เราจะกำหนดหน้าที่การทำงานได้ให้กำหนดจากรีจิสเตอร์ GPxCON

ในส่วนของการควบคุมบิทแต่ล่ะบิต เราสามารถสั่งได้จากรีจิสเตอร์ GPxDAT , GPxSET, GPxCLR และ GPxPAR จะเป็นการพูลอัพ (pull up) ภายในของไมโครคอนโทรลเลอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.5 GPxDAT Registers

Name	Address	Default Value	Access
GP0DAT	0xFFFFF420	0x000000XX	R/W
GP1DAT	0xFFFFF430	0x000000XX	R/W
GP2DAT	0xFFFFF440	0x000000XX	R/W
GP3DAT	0xFFFFF450	0x000000XX	R/W
GP4DAT	0xFFFFF460	0x000000XX	R/W

ตารางที่ 2.6 GPxDAT MMR Bit Descriptions

Bit	DESCRIPTION
31:24	Direction of the Data. Set to 1 by user to configure the GPIO pin as an output. Cleared to 0 by user to configure the GPIO pin as an input.
23:16	Port x Data Output.
15:8	Reflect the State of Port x Pins at Reset (read only).
7:0	Port x Data input (read only).

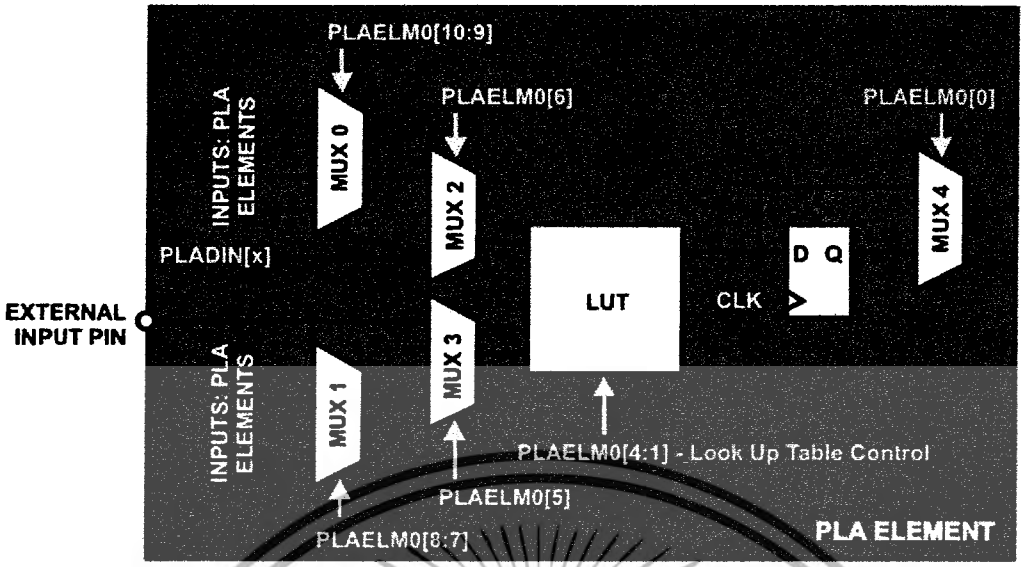
ตารางที่ 2.7 GPxSET Registers

Name	Address	Default Value	Access
GP0SET	0xFFFFF424	0x000000XX	W
GP1SET	0xFFFFF434	0x000000XX	W
GP2SET	0xFFFFF444	0x000000XX	W
GP3SET	0xFFFFF454	0x000000XX	W
GP4SET	0xFFFFF464	0x000000XX	W

ตารางที่ 2.8 GPxSET MMR Bit Descriptions

Bit	DESCRIPTION
31:24	Reserved.
23:16	Data PortxSet Bit. Set to 1 by user to set bit on portx;also sets the corresponding bit in the GPxDAT MMR. Cleared to 0by user; does not affect the data out.
15:0	Reserved

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 2.27 ฟังก์ชันไดอะแกรมของ single PLA

ฟังก์ชันไดอะแกรมของ single PLA ที่แสดงอยู่บนภาพที่ 2.27 เราจะเห็นอินพุทต่ออยู่กับ MUX ซึ่งสามารถให้สัญญาณอินพุทจาก

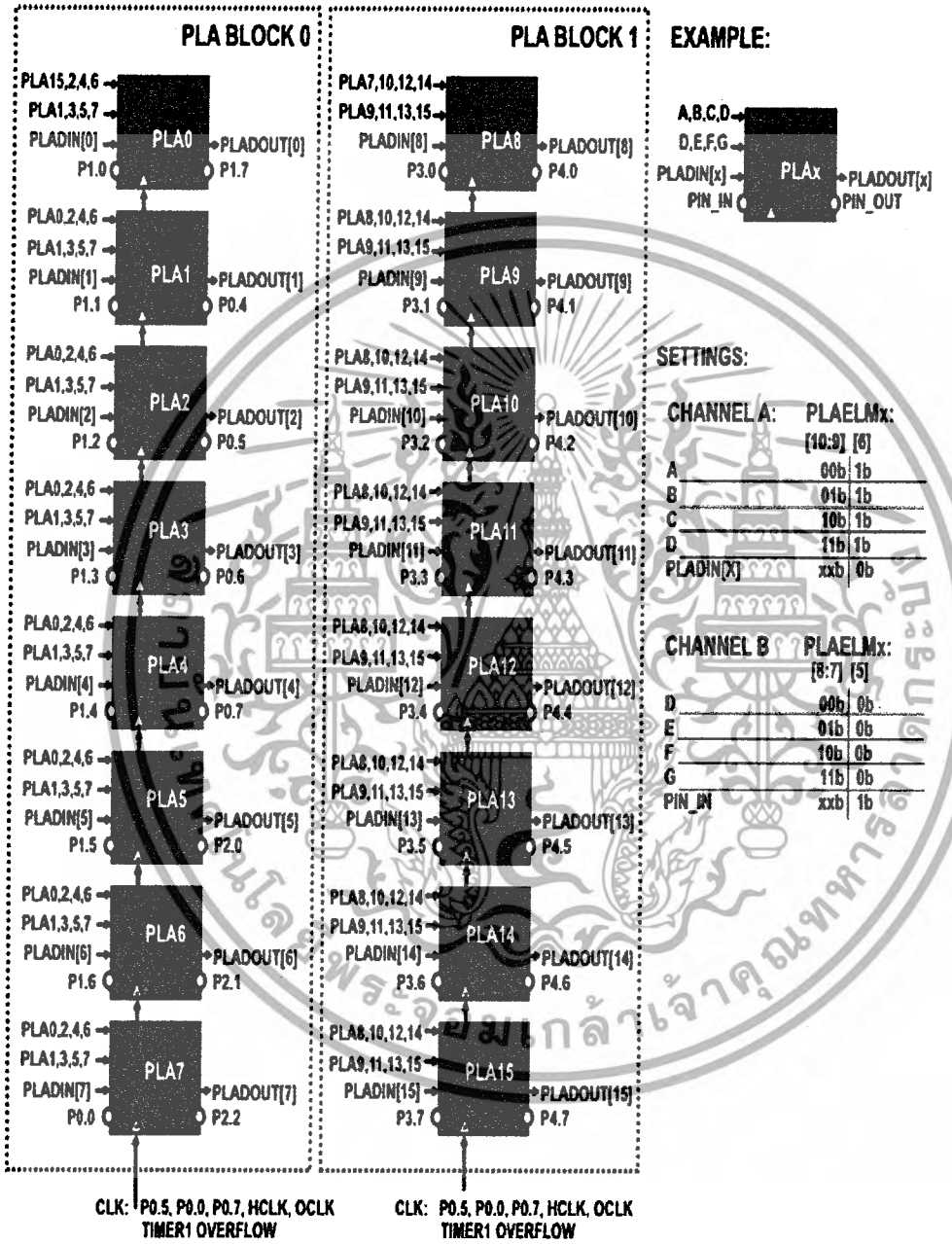
- external pin PLA
- other PLA output
- register PLADIN

ในบล็อกตรงกลาง PLA นั้นจะมี LUT(look up table) – LUT จะประกอบไปด้วยฟังก์ชันทั้งหมด 16 element โดยมันจะสามารถสร้างเกท XOR,OR,AND,NOT (จะมีค่าคงที่ 1 หรือ 0) LUT จะสามารถสร้างฟังชัน AND OR มาประกอบฟังก์ชันระหว่างสัญญาณ หรือสามารถกำหนด output ให้เป็น HI หรือ LOW ได้ (โดยไม่คำนึงถึง input) เอาพุทต์ของ LUT จะต่ออยู่กับ Flip-Flop Flip-Flop จะมีสัญญาณนาฬิกาอยู่ โดยสัญญาณนาฬิกาจะมาจาก GPIO, P0.5, P0.0 0.7, HCLK, OCLK เอาพุทต์ของ PLA จะต่ออยู่กับ PLADOUT register นั้นจะอ่านจากโปรแกรมโดยที่เอาท์พุทที่ขาภายนอก PLAO pin และตัวสัญญาณเอาท์พุทจะต่อกับ PLA ตัวอื่นอีกด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4.1 Input / Outputs Signals for PLAs

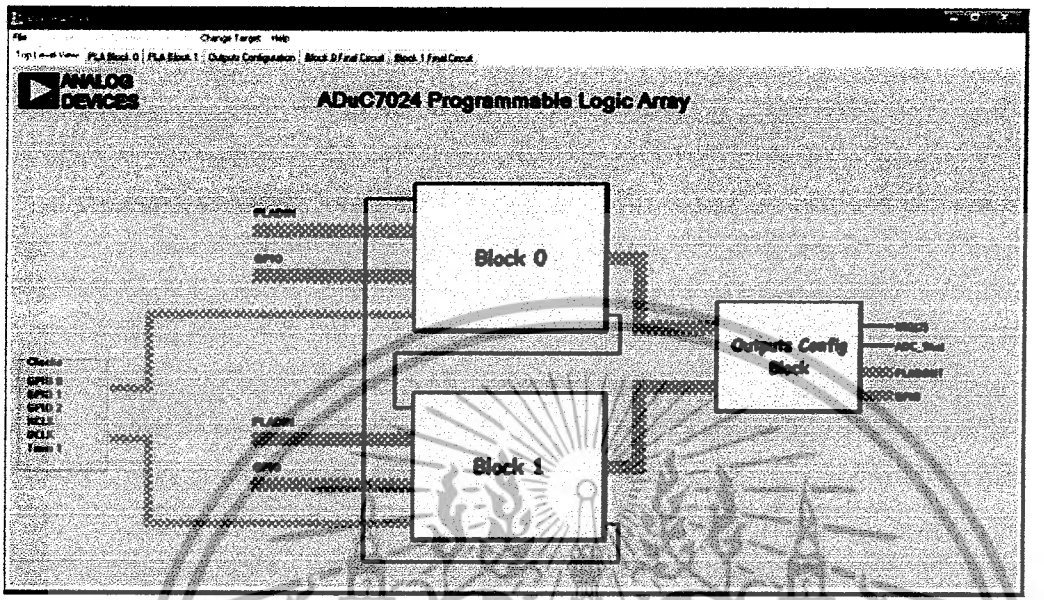
เพื่อให้เข้าใจง่ายอินพุทเอาท์พุทของทุกทุก PLA จะอยู่ในแผนภาพด้านล่าง สัญญาณจาก Channel A และ Channel B ไปที่ LUT



ภาพที่ 2.28 Input / outputs signals for PLAs

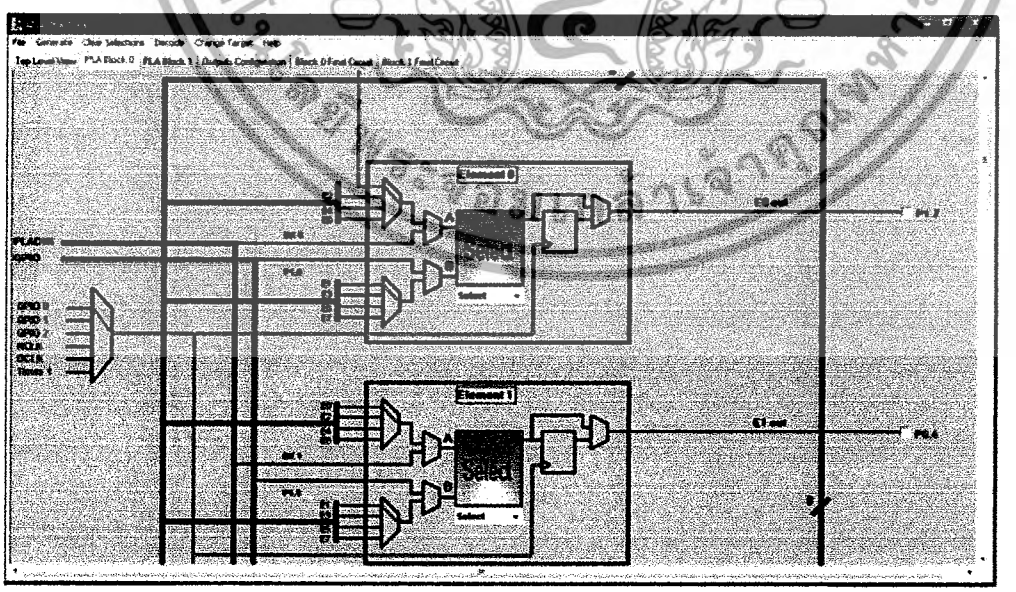
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในที่นี้จะเป็นการประยุกต์ใช้โปรแกรม ADUc PLA TOOL ในการออกแบบวงจรและกำหนดค่ารีจิสเตอร์ของ PLA ใน ไมโครคอนโทรลเลอร์ ADUc702x



ภาพที่ 2.29 หน้าต่างเริ่มต้นของโปรแกรม

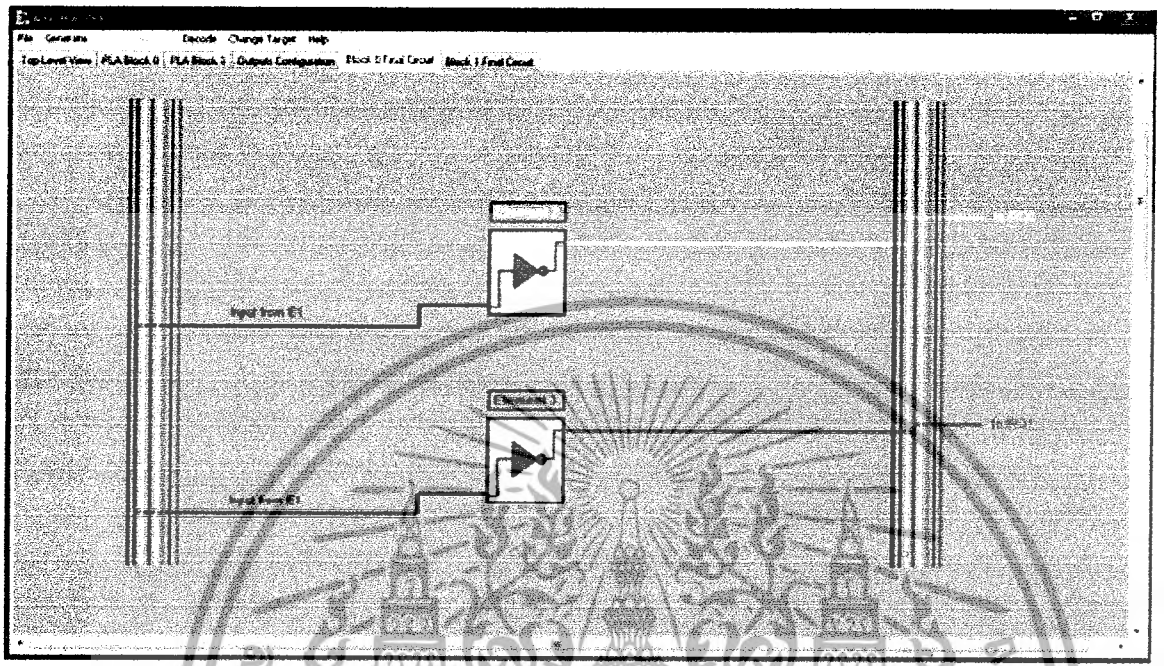
ในขั้นตอนแรกเมื่อเปิดโปรแกรมขึ้นมาจะเป็นการเลือก Block Element เพื่อทำการเขียน PLA ในที่นี้จะเลือก Block 0 เพื่อ ตรงกับขาอินพุทที่รับมา



ภาพที่ 2.30 หน้าต่าง Element

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อทำการเลือก Block แล้วจะเป็นการใส่สัญลักษณ์ Logic gate ลงไป ในที่นี้เราจะทำการใส่ NOT GATE เพื่อทำการกลับสัญญาณจาก positioner control card



ภาพที่ 2.31 แผนผังวงจรลอจิกเกท (Logic gate) ที่วาด

ทำการวาดเกทลงไปแล้วพร้อมทั้งเลือกแหล่งอินเตอร์รัปต์ ทั้งสองแหล่ง

หลังจากนั้นให้ทำการคำนวณค่ารีจิสเตอร์เพื่อนำมาเขียนในโปรแกรมภาษาซี หรือทำการกดไปที่ Generate เพื่อเรียกดูค่ารีจิสเตอร์ที่โปรแกรมคำนวณให้

```
#include "ADuC7024.h"
Void plaInitialize( )
{
        PLAELM2 = 0x002B;
        PLAELM3 = 0x002B;
        PLACLK = 0x0000;
        // IRQ output configuration
        PLAIRQ = 0x1312;
 }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
วิธีการดำเนินงาน

3.1 ศึกษากระบวนการทดสอบสมรรถนะของ Rotary Positioner แบบเดิม (APC Test)

การทดสอบ APC เป็นวิธีการทดสอบหา Rotary Positioner ดี และ Rotary Positioner เสีย แบบที่ใช้กันอยู่ในปัจจุบัน และผลที่ได้จากการทดสอบด้วยวิธีนี้ค่อนข้างที่จะแม่นยำ แต่ระยะเวลาที่ใช้ในการทดสอบแต่ละครั้งจะนานถึง 24 ชม. ซึ่งการทดสอบด้วยวิธีนี้จะเป็นการเขียนสัญญาณเซอร์โวลงบนแผ่นฮาร์ดดิสก์ทั้งแผ่น แล้วจะได้จำนวนแทรค (track) ทั้งหมดประมาณสองแสนกว่าแทรค (track) และจะได้ไฟล์นามสกุลแดต (.dat) ซึ่งจะเป็นรายละเอียดในการเขียนสัญญาณเซอร์โว จากนั้นนำข้อมูลที่ได้มาทำการป้อนค่าลงในโปรแกรม MATLAB จะปรากฏเป็นกราฟรูปสัญญาณ APC step และ Positioner step เราจะทราบว่าเป็น Rotary Positioner ดี หรือ Rotary Positioner เสีย นั้น วิเคราะห์ได้จากการดูรูปกราฟที่ปรากฏจาก APC step เท่านั้น กล่าวคือ ถ้าเส้นกราฟสีน้ำเงินอยู่ภายในบริเวณเส้นสีแดง แสดงว่า Rotary Positioner ตัวนั้นเป็นตัวที่ดี แต่ถ้าเส้นกราฟสีน้ำเงินอยู่นอกเส้นสีแดงก็จะถือว่า Rotary Positioner ตัวนั้นเป็นตัวที่เสีย

3.1.1 วิธีการและขั้นตอนการทำงาน

3.1.1.1 วิธีการทดสอบ Good/Bad Rotary Positioner

1. ศึกษาข้อมูลย้อนหลังของการเขียนสัญญาณเซอร์โว ลงบนแผ่นฮาร์ดดิสก์ไดร์ฟที่เกิดปัญหา (error) โดยเน้นเฉพาะปัญหาที่มี error code เกี่ยวกับ Rotary Positioner โดยตรง รวมทั้งศึกษาข้อมูลย้อนหลังของ Rotary Positioner ที่ไม่เกิดปัญหา คือ เมื่อเขียนงานสัญญาณเซอร์โวแล้วชิ้นงาน (ฮาร์ดดิสก์ไดร์ฟ) ผ่านตลอด

2. ทำการทดสอบ Rotary Positioner โดยใส่ชิ้นงาน (Golden Unit) สำหรับทำการทดสอบลงไป แล้วทำการเขียนสัญญาณเซอร์โวตามปกติ เมื่อเขียนเสร็จ บันทึกค่าผลการเขียนตามหน้าจอ (Monitor) โดยสังเกตว่า ผ่าน (Pass) หรือ ไม่ผ่าน (Fail)

3. นำ Golden Unit ไปลบสัญญาณก่อนการเขียนครั้งต่อไป

4. นำ Golden Unit ที่ทำการลบสัญญาณแล้วไปทดสอบการเขียนสัญญาณเซอร์โวกับ Rotary Positioner

5. ทำซ้ำข้อ 4 ทั้งหมด 3 ครั้ง เพื่อหา Good/ Bad Rotary Positioner ที่แน่นอน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1.1.2 ตารางบันทึกผล

ตารางที่ 3.1 Good Rotary Positioner Test

Good Rotary Positioner (serial No.)	**Test 1**	**Test 2**	**Test 3**	**Test 4**
12xxx – 7x	Pass	Pass	Pass	Pass
32xxx - Mx	Pass	Pass	Pass	Pass
32xxx - Mx	Pass	Pass	Pass	Pass

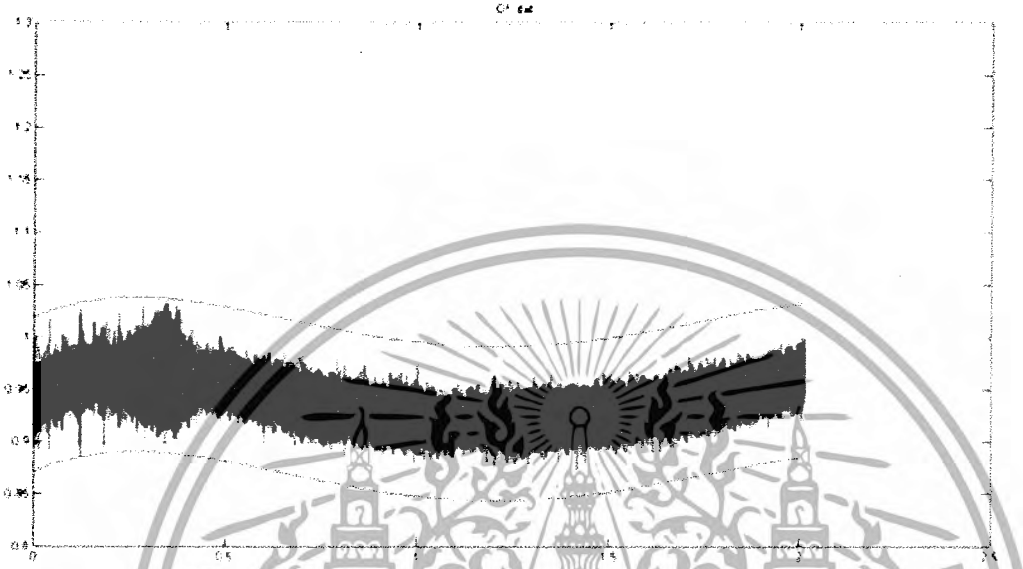
ตารางที่ 3.2 Bad Rotary Positioner Test

Good Rotary Positioner **(serial No.)**	**Test 1**	**Test 2**	**Test 3**
10xxx – 4x	Fail 9965 (3301)	Fail 9920 (0713)	Fail 9964 (3311)
32xxx – Mx	Fail 9064 (3308)	Fail 9964 (3308)	Fail 9920 (0709)
35xxx – Mx	Fail 9064 (3310)	Fail 9064 (3310)	Fail 9020 (0709)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

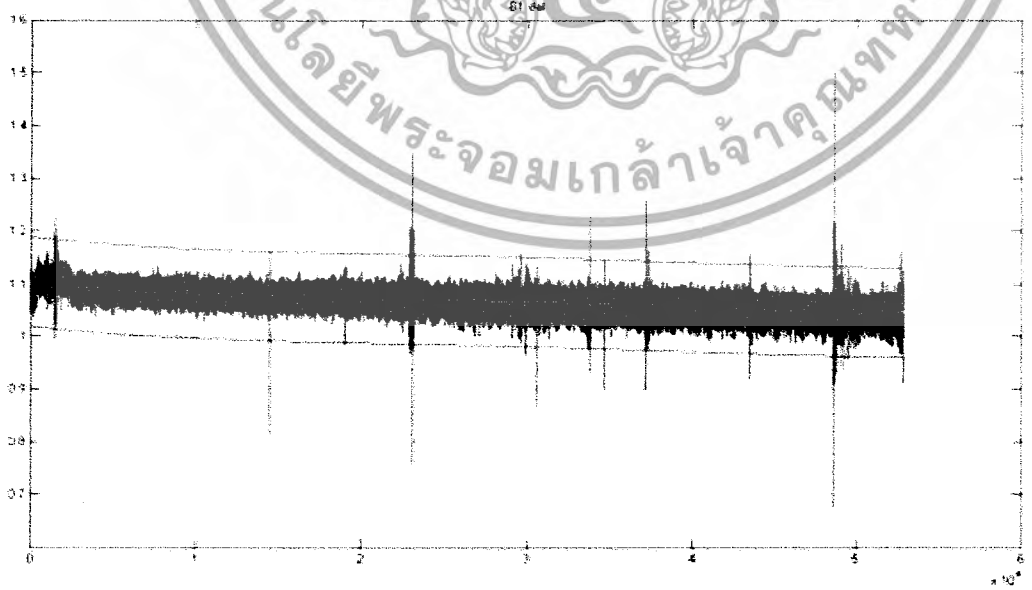
3.1.1.3 ตัวอย่างกราฟ APC จาก MATLAB

1. Good Rotary Positioner



ภาพที่ 3.1 กราฟ APC Step

2. Bad Rotary Positioner



ภาพที่ 3.2 กราฟ APC Step

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1.2 สรุปผลที่ได้จากการทดสอบด้วยวิธีแบบเดิม (APC Test)

ในการทดสอบแต่ละครั้งใช้เวลานานถึง 24 ชั่วโมง หรือ 1 วัน ซึ่งยังขาดความสะดวกและรวดเร็ว ดังนั้นคณะผู้จัดทำจึงมีแนวความคิดที่จะทำเครื่องทดสอบสมรรถนะของ Rotary Positioner แบบใหม่ขึ้นมา ให้มีประสิทธิภาพสูงกว่าแบบเดิม

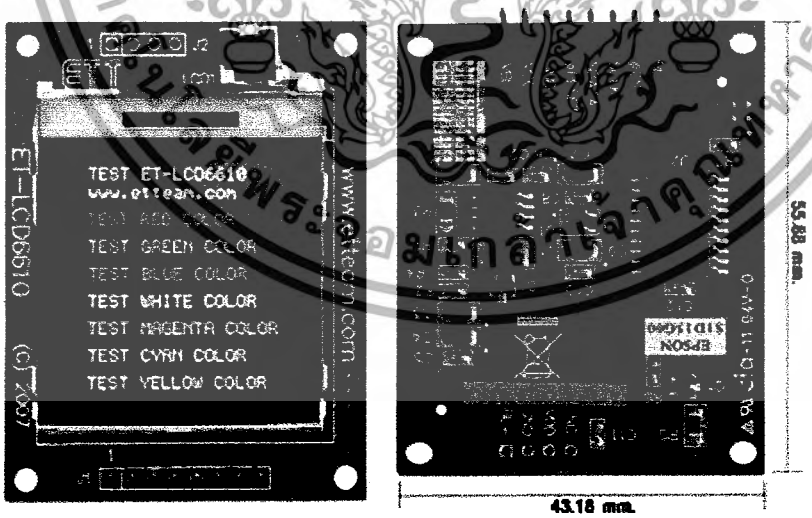
3.2 ออกแบบเครื่องทดสอบสมรรถนะของ Rotary Positioner

3.2.1 อุปกรณ์ฮาร์ดแวร์

3.2.1.1 LCD Graphpic

LCD Graphpic ที่ใช้ในการทดลองนี้ คือรุ่น ET- LCD6610 ซึ่งมีรายละเอียด ดังนี้

- ชิปประมวลผล Epson S1D15G10 Compatible Controller
- รองรับระบบไฟ 3.3 V ถึง 5.5 V
- ความละเอียดหน้าจอ 132x132 pixel
- ใช้การสื่อสารอนุกรมแบบ 9 Bit SPI Serial Interface
- เชื่อมต่อสัญญาณโดยตรงกับไมโครคอนโทรลเลอร์ได้ทั้งระบบไฟ 3.3 และ 5 โวลท์
- ความละเอียดสี 4,096 เฉดสี



ภาพที่ 3.3 LCD Graphpic

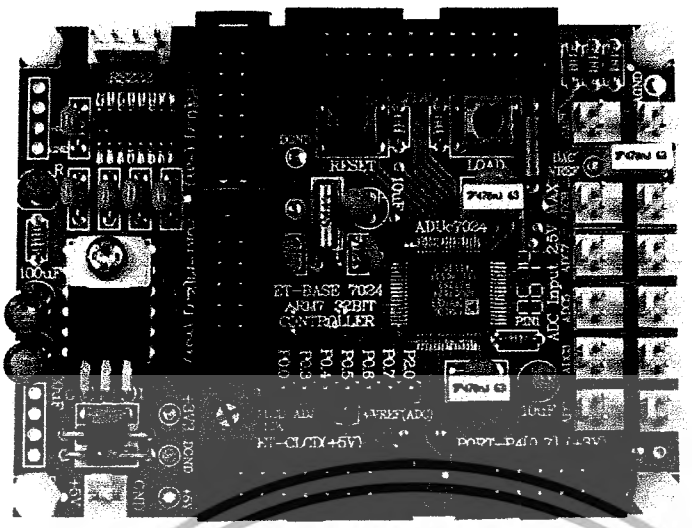
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.1.2 บอร์ด ET-BASE ARM7024 (ADUc7024)

จุดเด่นก็คือ มี A TO D ขนาด 12 BIT จำนวน 10 ช่องและ D TO A ขนาด 12 BIT 2 ช่อง ออกแบบเป็นบอร์ดขนาดมาตรฐาน ET-BASE สามารถนำไปใช้งานอิสระ หรือต่อทดลองกับบอร์ด ET-BASIC I/O V1 ก็ได้ ส่วนรายละเอียดอื่นๆมีดังนี้

- ใช้ MCU เบอร์ ADUC7024 ขนาด 64 PIN LQFP TYPE มีหน่วยความจำแบบ FLASH 62KBYTE, RAM 8KBYTE
- A TO D ขนาด 12 BIT 10 ช่อง (0 - 2.5V)
- D TO A ขนาด 12 BIT 2 ช่อง (0 - 2.5V)
- I/O PIN สามารถต่อกับ I/O ระดับสัญญาณ 5V ได้
- RUN X' TAL 32.768KHz สามารถกำหนดการทำงานร่วมกับ PHASE LOCK LOOP ให้สามารถทำงานได้ถึง 41.78MHz ได้
- 3 PORT I/O 10PIN ET
- 14 PIN LCD PORT แบบ CHARACTER TYPE
- TIMER/COUNTER , SPI , 16 BIT PWM , WATCHDOG
- ใช้ POWER SUPPLY 5 VDC (สามารถใช้กับชุด POWER SUPPLY แบบ ET-SWITCHING ADAPTER 5V 1.2A TYPE H)
- ขนาด PCB 6.2 x 8.1 cm.
- สามารถ DOWNLOAD โปรแกรมเข้าหน่วยความจำภายในแบบ FLASH ได้โดยตรงจากเครื่องคอมพิวเตอร์พีซี (PC) ผ่านทาง RS232 PORT ร่วมกับโปรแกรม ARMWSD ทำงานได้บน WINDOWS 98/ME/XP/2000/VISTA

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

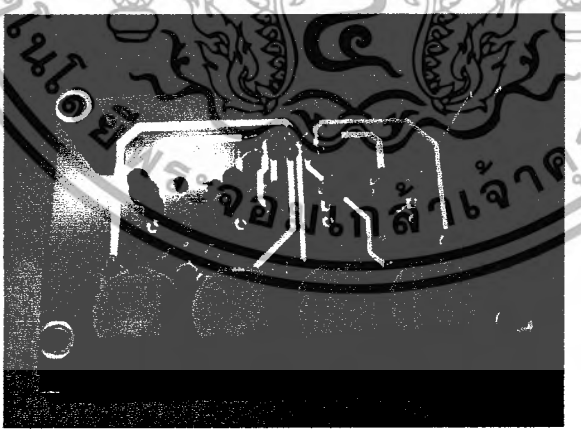


ภาพที่ 3.4 บอร์ด ET-BASE ARM7024 (ADUc7024)

3.2.1.3 บอร์ดสวิตซ์ และ LED

เป็นบอร์ดใช้สำหรับรับค่าอินพุทจากสวิตซ์ และแสดงค่าเอาท์พุทเป็นหลอด LED สีแดง

- อินพุท ใช้ SW 4 ตัว แบบ กดติด ปล่อยดับ
- เอาท์พุท ใช้ LED ขนาด 3 mm. กลม สีแดง 4 ตัว



ภาพที่ 3.5 ET-TEST 10P/INP-OUT

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

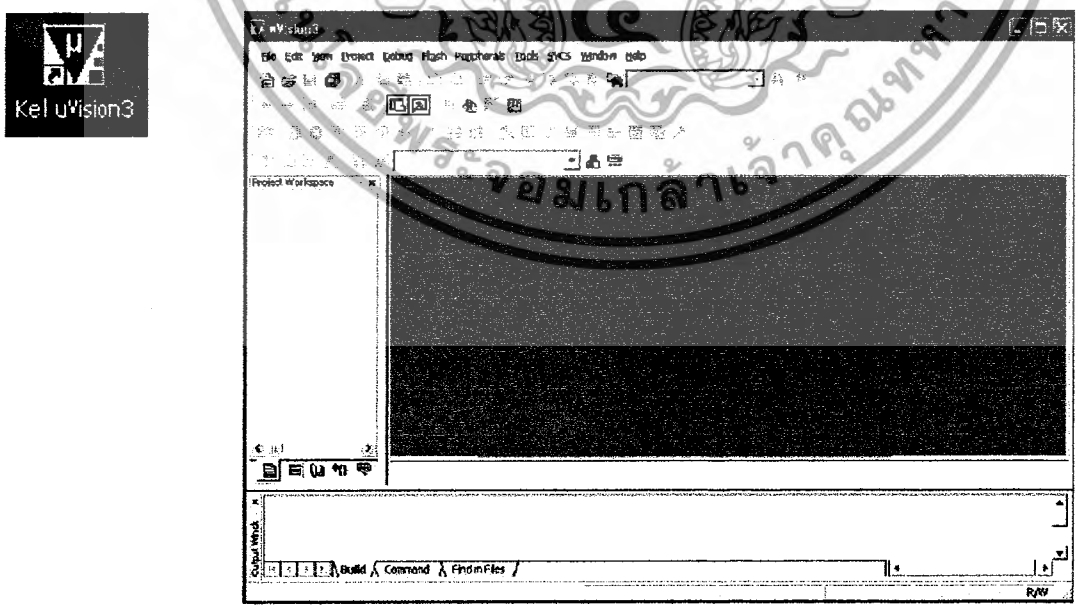
3.2.1.4 รางถ่าน AA ขนาด 4 ก้อน

เป็นแหล่งจ่ายไฟให้กับเครื่องทดสอบสมรรถนะของ Rotary Positioner

ภาพที่ 3.6 รางถ่าน AA ขนาด 4 ก้อน

3.2.2 โปรแกรมที่ใช้ในการทดสอบสมรรถนะของ Rotary Positioner

โปรแกรมที่ใช้ในการทดสอบสมรรถนะของ Rotary Positioner คือ Keil uVersion3 ซึ่งจะใช้การเขียนโปรแกรมภาษา C เพื่อสั่งงาน ARM7

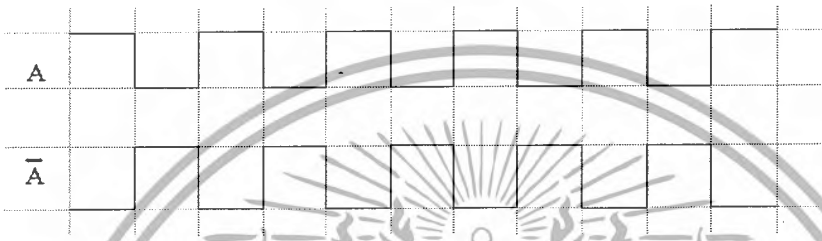


ภาพที่ 3.7 หน้าต่างโปรแกรม Keil uVersion3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

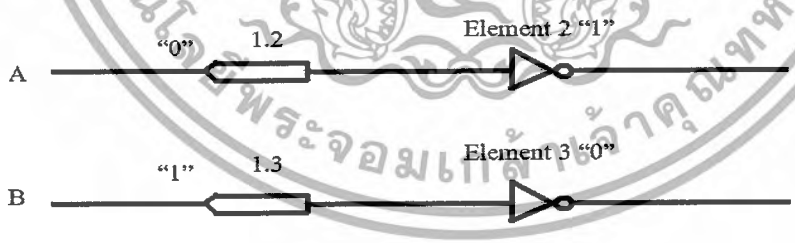
ในส่วนของการประยุกต์ใช้คุณสมบัติที่มีใน ARM7 สำหรับทดสอบ Rotary Positioner ตามทฤษฎีที่ได้กล่าวมานั้น สัญญาณพัลซ์จาก Positioner Control card จะมีความเร็วสูงมาก ดังนั้น ไมโครคอนโทรลเลอร์ที่เหมาะสมที่คณะผู้จัดทำได้เลือกมาใช้นั่นก็คือ ไมโครคอนโทรลเลอร์ ADUc7024 ซึ่งมีสถาปัตยกรรมของ ARM7

สัญญาณที่ได้จาก Positioner Control card จะมีลักษณะเป็นพัลซ์แบบ Active Low ดังนั้น การจะอ่านค่าพัลซ์ให้ถูกต้องนั้น จะต้องทำการกลับ Logic ของสัญญาณก่อน ดังภาพที่ 3.8



ภาพที่ 3.8 การกลับ Logic สัญญาณพัลส์ของ Encoder จาก Positioner Control card

ในการกลับสัญญาณ Logic นั้น เราจะต้องใส่ Not Gate เข้าไปตัวหนึ่ง ในทางปฏิบัติเรา สามารถสร้างวงจรขึ้นมาได้ แต่ในไมโครคอนโทรลเลอร์ ADUc7024 มีส่วนที่เรียกว่า PLA (Programmable Logic Control) ดังนั้นเราจึงนำค่าสัญญาณจาก Positioner Control card มาต่อกับ ขาอินพุท PLA ของไมโครคอนโทรลเลอร์ ADUc7024 ดังภาพที่ 3.9



ภาพที่ 3.9 ตัวอย่างการใส่ PLA

ในการนับพัลซ์นั้นจะใช้ส่วนของการนำสัญญาณที่ออกมาจาก PLA มาทำการอินเตอร์รัพ (Interrupt) ที่ไมโครคอนโทรลเลอร์ โดยจะเขียนโปรแกรมให้อ่านเฉพาะ ขอบขาขึ้นของสัญญาณ พัลซ์เท่านั้น (พัลซ์ที่ผ่าน PLA มาแล้ว) ดังนั้นในตอนนี้ไมโครคอนโทรลเลอร์ ADUc7024 จะรับรู้ สัญญาณพัลซ์ได้แล้ว ขั้นตอนต่อมา เมื่อโปรแกรมเริ่มนับค่าพัลซ์แล้ว จะเริ่ม Sampling ค่าทุก ๆ 200 µs เพื่อจะได้สามารถหาสมรรถนะของ Rotary Positioner ได้โดยการนำค่าไปเก็บไว้ใน Buffer

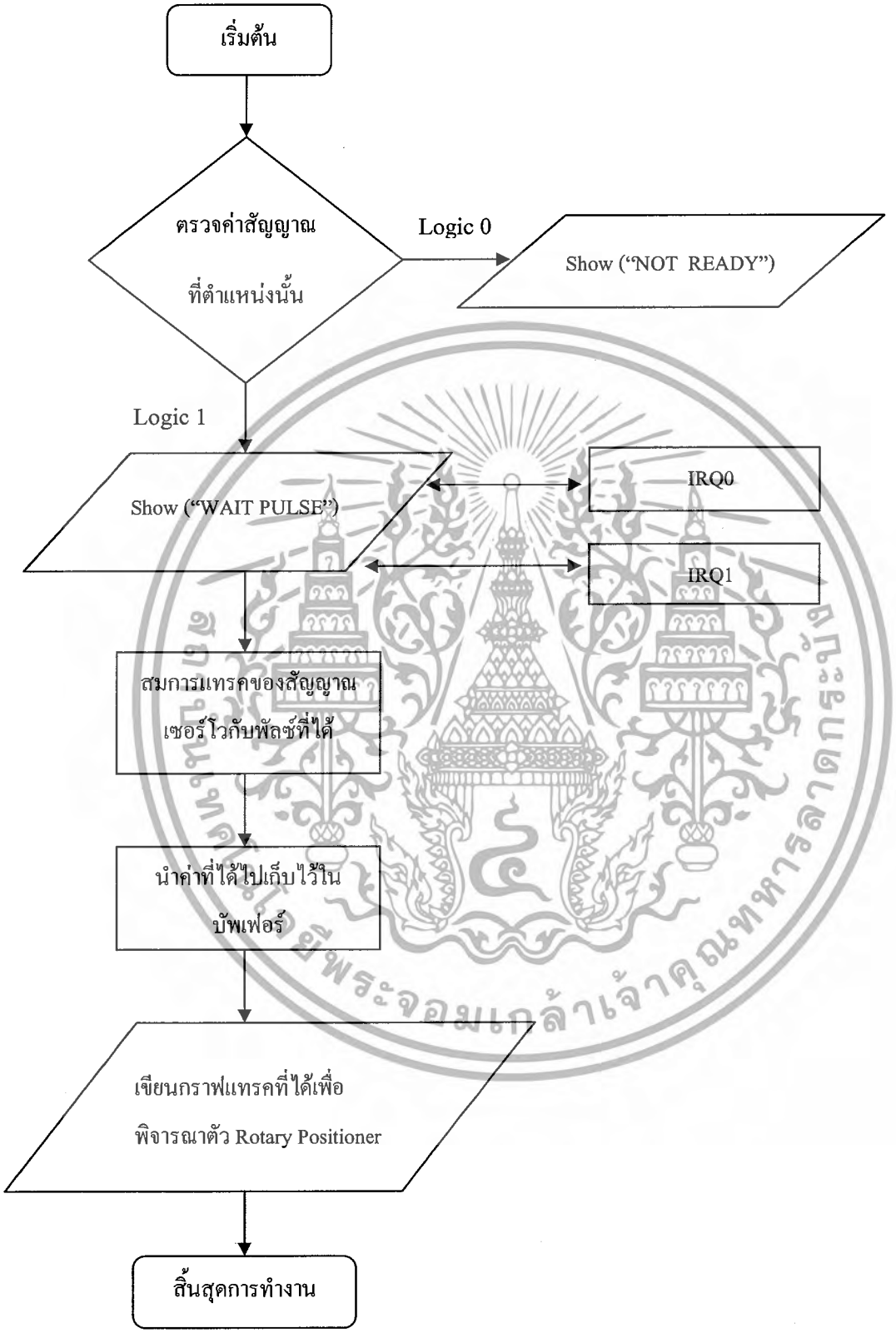
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เพื่อรอการนำค่าที่ Sampling ได้นี้ไป Plot Graph เมื่อกดปุ่ม Plot Graph ฟังก์ชัน Plot Graph ก็จะเรียกค่าที่เก็บไว้ใน Buffer ทุกๆจุดมา Plot Graph ให้ดู เพื่อที่จะสามารถวิเคราะห์ Rotary Positioner ต่อไป

3.2.3 แผนผังการทำงานของโปรแกรมที่ใช้ในเครื่องทดสอบสมรรถนะของ Rotary Positioner

ลำดับขั้นตอนการทำงานของโปรแกรมที่สามารถสรุปได้จากแผนผังการทำงาน คือ เมื่อเราเริ่มทำการเปิดเครื่องมา โปรแกรมจะทำการเช็คสัญญาณ ถ้าสัญญาณเป็น " 0 " โปรแกรมจะไปเรียกฟังก์ชัน " NOT READY " แต่ถ้าสัญญาณเป็น " 1 " โปรแกรมจะไปเรียกฟังก์ชัน " WAIT PULSE " หลังจากนั้นโปรแกรมจะเก็บค่าไว้เพื่อรอการอินเตอร์รัพ และเมื่อเกิดการอินเตอร์รัพแล้วก็จะทำการเก็บค่า Pulse ไว้ด้วยในตัว พร้อมทั้งทำการ sampling ค่า pulse ทุก ๆ 200 μs แล้วนำมาคำนวณในสมการ track เพื่อนำไปเก็บไว้ในบัพเฟอร์ จากนั้นการทำการกดปุ่มพล็อตกราฟ โปรแกรมก็จะทำการเรียกฟังก์ชันพล็อกราฟออกมาเพื่อทำการพล็อตกราฟตามสัญญาณ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 3.10 แผนผังการทำงานของโปรแกรมที่ใช้ในเครื่องทดสอบสมรรถนะของ Rotary Positioner

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3 ออกแบบการทดลองเครื่องทดสอบสมรรถนะของ Rotary Positioner

3.3.1 ขั้นตอนการทำงาน

1. ติดตั้งเครื่องเขียนสัญญาณเซอร์โวให้พร้อมเขียนสัญญาณจริงให้เรียบร้อย

1.1 การติดตั้ง (Set up) เครื่องเขียนสัญญาณเซอร์โว

ในการเขียนสัญญาณเซอร์โวนั้น ขั้นตอนการติดตั้งอุปกรณ์ก็เป็นอีกปัจจัยหนึ่งที่จะทำให้การเขียนประสบความสำเร็จหรือล้มเหลว ฉะนั้น จึงต้องติดตั้งอย่างถูกต้องตามขั้นตอนที่จะนำเสนอดังต่อไปนี้

- การถอดเปลี่ยน Rotary Positioner

1) ปิดสวิทช์เครื่องให้เรียบร้อย

2) ถอดสายสัญญาณของเครื่องเขียนสัญญาณเซอร์โวทั้ง 2 สายออก

3) ใช้กุญแจ L ขันสกรูของเซนเซอร์ตรวจสอบตำแหน่งการวางฮาร์ดดิสก์ไดร์ฟออก

4) คลายสกรูที่ Beak แล้วนำ Beak ออกด้วยกุญแจ L

5) ถอดสกรูที่ฐานรองฮาร์ดดิสก์ไดร์ฟทั้ง4มุมด้วยกุญแจ L

6) ถอดสกรูที่ยึดตัว Rotary Positioner กับเครื่องเขียนสัญญาณเซอร์โวออกทั้ง 4 ด้านโดยใช้มือดัน C-Clamp ด้านล่างไว้ด้วยเพื่อป้องกัน C-Clamp ร่วงลงกระแทกเครื่องเขียนสัญญาณเซอร์โว

ภาพที่ 3.11 ภาพประกอบขั้นตอนการถอด Rotary Positioner

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- การใส่ Rotary Positioner ลงบนเครื่องเขียนญญาณเซอร์โว

1) นำ Rotary Positioner ใส่เข้าไปใต้เครื่องเขียนสัญญาณเซอร์โวโดยให้ก้านที่ยึดกับ Beak โผล่ออกมา แล้วขันสกรูทั้ง 4 มุมเพื่อยึดตัว Rotary Positioner กับเครื่องเขียนสัญญาณเซอร์โว

2) ขั้นสกรูฐานรองฮาร์ดดิสก์ไดร์ฟ เข้ากับเครื่องเขียนสัญญาณ เซอร์โว

3) ต่อสายสัญญาณทั้ง 2 เข้ากับเครื่องเขียนสัญญาณเซอร์โวให้เรียบร้อย

ภาพที่ 3.12 ภาพประกอบขั้นตอนการใส่ Rotary Positioner

1.2 การติดตั้ง Beak ให้อยู่ในสภาพพร้อมใช้งาน

หลังจากการประกอบส่วนของ Rotary Positioner กับเครื่องเขียนสัญญาณเซอร์โวแล้ว จากนั้นเราจำเป็นต้องใช้คอมพิวเตอร์ช่วยในการหาตำแหน่งที่ถูกต้องในการติดตั้ง Rotary Positioner เสียก่อน

1) เปิดเครื่องเขียนสัญญาณเซอร์โวให้พร้อมใช้งาน

2) บนหน้าจอจะปรากฏหน้าต่างคำสั่ง ให้คีย์คำสั่งต่อไปนี้

Wait START... /* กด ESC 1ครั้ง */ เพื่อเข้าสู่การตั้งค่า

E:\XXX> /* พิมพ์ s แล้วกด Enter 1 ครั้ง เพื่อเข้าสู่ Manual Mode */

จากนั้นบนจอจะปรากฏข้อความดังต่อไปนี้

[XXX PROD version YYY110G]

Parameter file name is YYY4.PRM

XXX Manual Mode . .

SBC 400 MHz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

File Opened 2 Closed 2 Fail 0 เป็นการแสดงไฟล์ที่มาของค่าพารามิเตอร์ในการเซ็ตตำแหน่ง

posi /* พิมพ์ posi แล้วกด Enter 1 ครั้ง */

จากนั้นจอจะแสดงผลดังนี้และรอให้เครื่องโหลดค่าพารามิเตอร์สักครู่

[PosHardware Reset]

[PosLoadParameter]

[PosRezero]

Positioner initialize RC = 0000 /* ข้อสังเกตหากไม่ใช่ค่า RC=0000จะเกิดความผิดพลาดจนไม่สามารถตั้งค่าได้ในขั้นตอนต่อไปฉะนั้นต้องเริ่มทำใหม่ตั้งตั้งแต่ปิดและเปิดเครื่องใหม่ */

Posdiag /* พิมพ์ posdiag แล้วกด Enter 1 ครั้ง */

หน้าจอแสดงผล

Alarm code: 00000h

Status code: 0000Dh

SYNC

INPOSITION

SERV00N

ADR = 48000000

SC01 TEST – V2.0 > **move** /* พิมพ์ move แล้วกด Enter 1 ครั้ง */

จะปรากฏคำว่า ADDRESS ให้ใส่ตัวเลข 9000000 แล้ว Enter เพื่อเป็นการย้ายตำแหน่งไปอยู่ที่ตำแหน่ง 9000000

จากนั้นนำ Set Beak มาวาง เพื่อเป็นแนวแล้วนำ Beak มาวางโดยให้แนบกะ Set Beak พอดีดังรูปแล้วขันสกรูให้บีคยึดแน่นกับ Positioner แล้วถอด set Beak ออก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาพที่ 3.13 ภาพประกอบการติดตั้งบีค (Beak)

เมื่อใส่ Beek เสร็จเรียบร้อยแล้วให้เซ็ตค่าบีคไปสู่ตำแหน่งเริ่มต้นอีกครั้ง

SC01 TEST – V2.0 > **move** /* พิมพ์ move แล้วกด Enter 1 ครั้ง */

จะปรากฏคำว่า ADDRESS ให้ใส่ตัวเลข 48000000 แล้ว Enter

Posi /* พิมพ์ posi แล้วกด Enter 1 ครั้ง */

X /* พิมพ์ x แล้วกด Enter 1 ครั้ง เพื่อออก */

E:\XXX>**auto** /* พิมพ์ auto แล้วกด Enter 1 ครั้ง เพื่อรอ การใส่ Unit ต่อไป*/

1.3 การเขียนสัญญาณเซอร์โว

หลังจากการติดตั้ง Beak ให้อยู่ในสภาวะที่พร้อมใช้งานแล้วนั้น เราก็จะเริ่มทำการเขียนสัญญาณเซอร์โวลงบนแผ่นฮาร์ดดิสก์ไดร์ฟ โดยเราจะทำการโปรแกรมคำสั่งดังนี้

Ctrl +c

E\NCH:> **c:**

C:\>**cd dpc**

C:\DPC>**copy *.* e:**

C:\DPC> **e:**

E:\NCH> **dir**

E:\NCH> **autoexec**

โปรแกรมจะทำงานไปเรื่อยๆจนกระทั่งขึ้น Error ให้กด **Ctrl+C** แล้ววาง Golden Unit ลงไป

กด A ซึ่งอยู่ด้านขวาบน เพื่อ lock Golden Unit

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กด A ซึ่งอยู่ด้านซ้ายบน เพื่อ Load Golden Unit

E:\NCH> **makeramp** \\ timing Calibration

E:\NCH> **s**

E:\NCH> **s** \\ start motor

->head

โปรแกรมจะทำงานไปเรื่อยๆ จนสุดท้ายให้ตรวจดูว่า ค่า RC (Result Code) มีค่าเท่ากับ 0000 หรือไม่ ถ้าเท่ากับ 0000 ให้ทำการเขียนคำสั่งต่อไป แต่หากไม่เท่ากับ 0000 ให้ทำการเขียนคำสั่ง **motoff** เพื่อเป็นการหยุดมอเตอร์ แล้วทำการเขียนคำสั่งใหม่ ตั้งแต่ E:\NCH>s เป็นต้นมา ถ้าเท่ากับ 0000 ก็ทำการเขียนคำสั่งต่อไปได้เลย

->ppro \\ เขียนสัญญาณ servo

โปรแกรมจะถามหา serial ให้ใส่ **0** ลงไป

โปรแกรมจะถามว่า Erase before (ต้องการจะลบสัญยาณก่อนการเขียนหรือไม่) ใส่ **0** หรือ **1**

ต้องการจะเขียนจำนวน track เท่าไหร่ ก็ใส่จำนวนลงไป ในที่นี้จะใส่ **100** และ **400**

สุดท้ายก่อนที่จะทำการเขียนสัญญาณลงบน Golden Unit โปรแกรมจะถามว่า power off หรือไม่ (จะปิดเครื่องหลังจากการเขียนเสร็จเลยหรือไม่) ใส่ **0** หรือ **1**

โปรแกรมจะเริ่มการเขียนสัญญาณจนถึง track ที่กำหนด

เมื่อเขียนสัญญาณเสร็จให้ใช่คำสั่ง motoff เพื่อเป็นการหยุดมอเตอร์ แล้วจากนั้นก็ทำการ

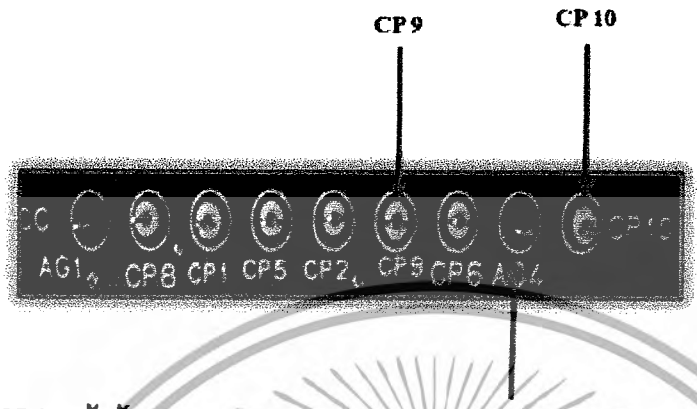
กด B ซึ่งอยู่ด้านขวาบน เพื่อ Unlock Golden Unit

กด B ซึ่งอยู่ด้านซ้ายบน เพื่อ UnLoad Golden Unit

ภาพที่ 3.14 โซลินอยด์ควบคุมการ Load และ Unload Golden Unit

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. เปิดฝาปิด ตรงส่วนของบอร์ดที่ต่อกับการ์ดต่าง ๆ เพื่อต่อเครื่องทดสอบสมรรถนะของ Rotary Positioner ที่สร้างขึ้น เข้ากับจุด Test Point



ภาพที่ 3.15 แสดงจุด Test Point บน Positioner Card

3. วัดสัญญาณที่ได้ และบันทึกผลลงในตาราง

4. ยืนยันว่าผลการทดลองนั้นมีประสิทธิภาพที่สามารถเชื่อถือได้ โดยนำหลักการของการควบคุมคุณภาพมาประยุกต์ใช้

5. ในการอ้างอิงว่าถูกต้องจริงหรือไม่ ต้องใช้ APC ทดสอบอีกครั้งกับเครื่องเดิมที่วัดสัญญาณเครื่องทดสอบสมรรถนะของ Rotary Positioner ที่สร้างขึ้น เพื่อยืนยันความถูกต้อง

6. สรุปผล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

การแสดงผล

4.1 เครื่องทดสอบสมรรถนะของ Rotary Positioner

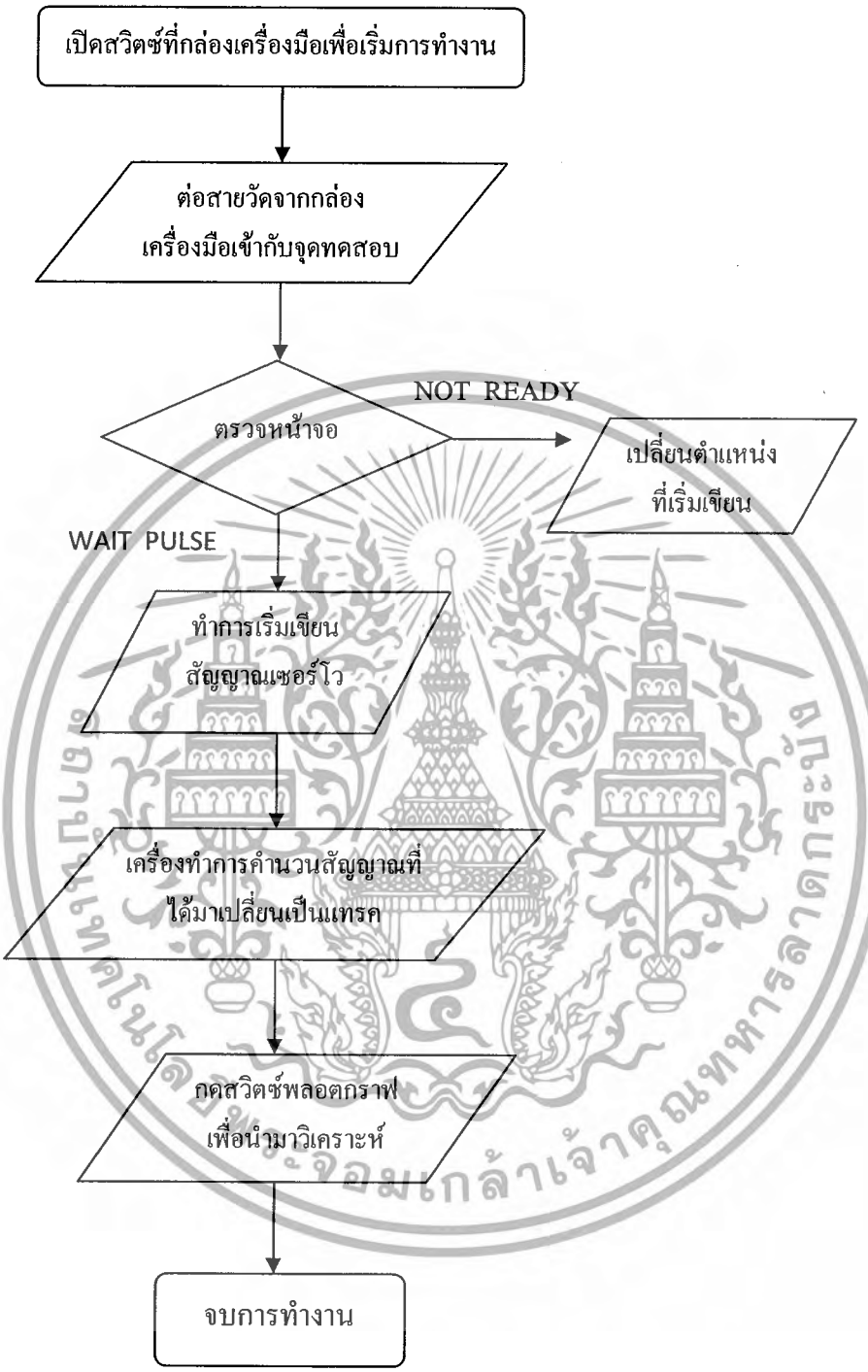
จากการศึกษาและร่วมมือกันออกแบบเครื่องทดสอบสมรรถนะของ Rotary Positioner ของผู้วิจัยนั้น ทำให้เกิดเครื่องมือที่จะสามารถนำไปใช้ทำการทดลอง ดังภาพที่ 4.1

ภาพที่ 4.1 เครื่องทดสอบสมรรถนะของ Rotary Positioner

4.2 การทำงานของ เครื่องทดสอบสมรรถนะของ Rotary Positioner

การทำงานของเครื่องทดสอบสมรรถนะของ Rotary Positioner นั้นมีหลักการทำงานคือ เมื่อเราทำการเปิดสวิตซ์ที่ตัวเครื่อง แล้วทำการต่อสายที่จะทำการวัดสัญญาณจาก Rotary Positioner เข้าไปที่ Positioner Control Card จากนั้นก็ให้สังเกตที่หน้าจอของเครื่องทดสอบสมมรรถนะของ Rotary Positioner ว่ามีอะไรปรากฏออกมาหรือไม่ ถ้าหน้าจอปรากฏคำว่า NOT READY แสดงว่าเครื่องทดสอบสมรรถนะของ Rotary Positioner ยังไม่พร้อมที่จะทำงานในตำแหน่งนั้น ให้ทำการ move track ไปอีกเพื่อเปลี่ยนตำแหน่ง หลังจากที่ทำการเปลี่ยนตำแหน่งเรียบร้อยแล้ว หน้าจอจะปรากฏคำว่า WAIT PULSE หมายความว่าเครื่องทดสอบสมรรถนะของ Rotary Positioner พร้อมที่จะใช้งานแล้ว ก็ให้ทำการ move track ไปเป็นจำนวน 100 และ 400 แทรค ตามลำดับ เครื่องทดสอบสมรรถนะของ Rotary Positioner ก็จะทำการหาจำนวน pulse แล้วนำมาพล็อตกราฟเพื่อทำการพิจารณาว่า Rotary Positioner ตัวนั้นดีหรือเสีย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 4.2 ผังการทำงานของเครื่องทดสอบสมรรถนะของ Rotary Positioner

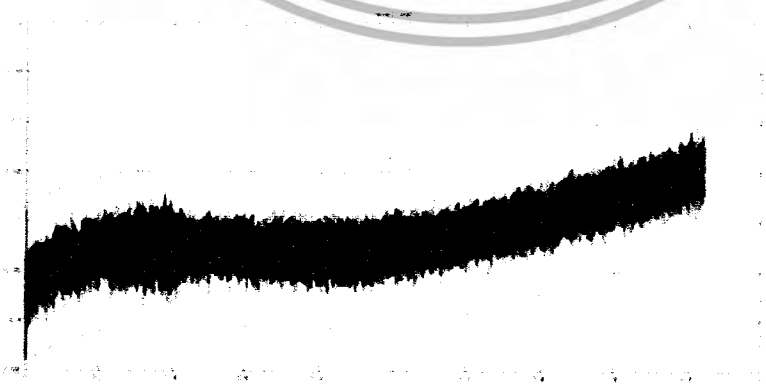
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3 ผลการทดลอง

จากการออกแบบการทดลองในบทที่ 3 เพื่อที่เราจะทำการหา Rotary Positioner ที่ดีและเสีย อย่างละ 3 ตัว เราจึงต้องทำการสุ่มหา Rotary Positioner จากการดูค่า Percent Yield ซึ่งค่า Percent Yield จะเป็นตัวบ่งบอกอัตราส่วนของ output ที่เขียนสัญญาณผ่าน / ชิ้นงาน input ทั้งหมด ถ้าเป็น Rotary Positioner ที่ดี จะมีค่า Percent Yield สูง ส่วน Rotary Positioner ที่เสียนั้น จะมีค่า Percent Yield ต่ำ โดยเราจะแทน Rotary Positioner ที่ดี ทั้ง 3 ตัวด้วย Good 1 , Good 2 , Good 3 และ Rotary Positioner เสียด้วย Bad 1 , Bad 2 , Bad 3

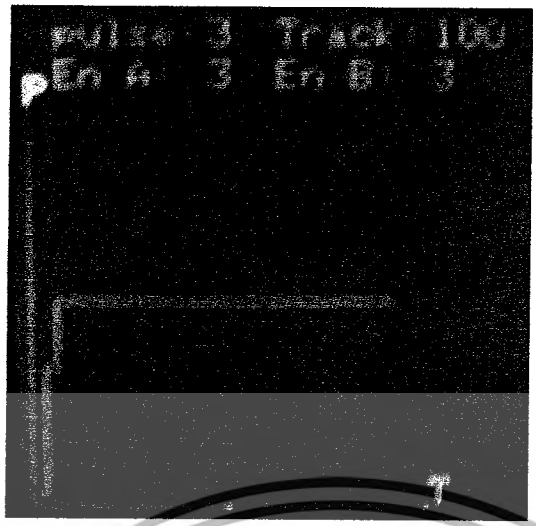
ผลการทดลองจากการที่เรานำเครื่องทดสอบสมรรถนะของ Rotary Positioner ไปทำการจับสัญญาณ Pulse จาก Positioner Control Card โดยใช้คำสั่ง move track ไปจำนวน100 แทรค และ 400 แทรค เครื่องทดสอบจะบอกจำนวน Pulse และจำนวนแทรคออกมาในรูปของตัวเลขแสดงผลที่จอ LCD Graphic พร้อมทั้งพล็อตกราฟเวลาและ Pulse ออกมา เพื่อเป็นการพิจารณาว่า กราฟที่ปรากฏออกมานั้น มีความราบเรียบหรือไม่ และจำนวนของ Pulse ที่ออกมานั้นตรงกับความเป็นจริงหรือไม่ กล่าวคือ ถ้ากราฟมีความราบเรียบแสดงว่าเป็น Rotary Positioner ที่ดี และถ้าในทางตรงกันข้าม ลักษณะกราฟที่ได้ มีความไม่ราบเรียบก็จะสามารถยืนยันได้ว่าเป็น Rotary Positioner เสีย ส่วนจำนวน Pulse ที่ออกมานั้น ถ้าสั่งให้ move track ไปจำนวน100 แทรค Pulse ที่ออกมาต้องได้ 3 Pulse และถ้า move track ไปจำนวน 400 แทรค Pulse ที่ออกมาต้องได้ 11 Pulse จึงจะถือว่าเป็น Rotary Positioner ที่ดี ซึ่ง 3 Pulse และ 11 Pulse นั้น เราได้มาจากการที่เราได้นำ Rotary Positioned ตัวที่ดี Serial Number 10xxx-4xมาทำการทดสอบเพื่อนำผลที่ได้มาเป็นมาตรฐานในการยืนยันผลการทดลอง ซึ่งผลการทดลองเป็นดังนี้

4.3.1 ผลการทดสอบ Rotary Positioner ตัวที่ดี Serial Number 10xxx-4x เพื่อนำมาเป็นมาตรฐานในการยืนยันผลการทดลอง

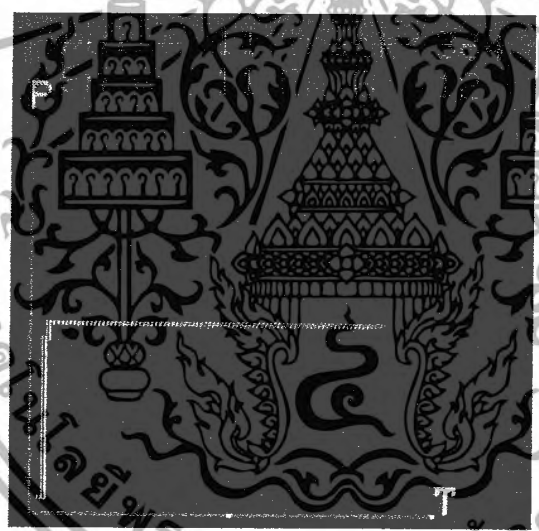


ภาพที่ 4.3 ภาพจากการ Test APC

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 4.4 ภาพจากการที่ใช้เครื่องทดสอบสมรรถนะของ Rotary Positioner ทดสอบวัดสัญญาณจากการ move track ไปจำนวน100 แทรค



ภาพที่ 4.5 ภาพจากการที่ใช้เครื่องทดสอบสมรรถนะของ Rotary Positioner ทดสอบวัดสัญญาณจากการ move track ไปจำนวน 400 แทรค

จากภาพข้างต้น จะเห็นว่า จากการที่ใช้เครื่องทดสอบสมรรถนะของ Rotary Positioner ทดสอบวัดสัญญาณจากการ move track ไปจำนวน100 แทรค จะได้ Pulse ออกมาจำนวน 3 Pulse และ Pulse ที่ออกมาจาก Encoder phase A และ Encoder phase B ก็จะเท่ากับ 3 Pulse ส่วนการทดสอบวัดสัญญาณจากการ move track ไปจำนวน400 แทรค จะได้ Pulse ออกมาจำนวน 11 Pulse และ Pulse ที่ออกมาจาก Encoder phase A และ Encoder phase B จะเท่ากับ 11 และ 12 Pulse ตามลำดับ ซึ่งผู้ทำการวิจัยจะนำค่าที่ได้นี้เป็นค่ามาตรฐานเพื่อยืนยันว่า Rotary Positioner เป็นตัวที่ดี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3.2 บันทึกผลการทดลองที่ 100 track

ตารางที่ 4.1 แสดงผลการทดลองที่ 100 track

Rotary Positioner Serial Number	การเขียนสัญญาณเซอร์โว				
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5
Good 1 (S/N 33xxx-M4x)	Pass	Pass	Pass	Pass	Pass
Good 2 (S/N 35xxx-Mx)	Pass	Pass	Fail	Pass	Pass
Good 3 (S/N 31xxx-Mx)	Pass	Pass	Pass	Pass	Pass
Bad 1 (S/N 37xxx-Nx)	Fail	Fail	Fail	Fail	Fail
Bad 2 (S/N 33xxx-Mx)	Fail	Fail	Fail	Fail	Fail
Bad 3 (S/N 35xxx-Mx)	Fail	Fail	Fail	Fail	Fail

4.3.3 บันทึกผลการทดลองที่ 400 track

ตารางที่ 4.2 แสดงผลการทดลองที่ 400 track

Rotary Positioner Serial Number	การเขียนสัญญาณเซอร์โว				
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5
Good 1 (S/N33xxx-M4x)	Pass	Pass	Pass	Pass	Pass
Good 2 (S/N 35xxx-Mx)	Pass	Pass	Pass	Pass	Pass
Good 3 (S/N 31xxx-Mx)	Pass	Pass	Pass	Pass	Pass

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 4.2 (ต่อ) แสดงผลการทดลองที่ 400 track

Rotary Positioner Serial Number	การเขียนสัญญาณเซอร์โว				
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5
Bad 1 (S/N 37xxx-Nx)	Fail	Fail	Fail	Pass	Fail
Bad 2 (S/N 33xxx-Mx)	Fail	Fail	Fail	Fail	Fail
Bad 3 (S/N 35xxx-Mx)	Fail	Fail	Fail	Fail	Pass

จากตารางบันทึกผลการทดลองเป็นการเก็บสถิติของการเขียนสัญญาณเซอร์โวจาก Rotary Positioner 6 ตัว โดยมี Rotary Positioner ตัวที่ดี 3 ตัว และตัวที่เสียอีก 3 ตัว เพื่อเป็นการยืนยันว่าผลการทดลองนั้นมีประสิทธิภาพที่สามารถเชื่อถือได้ ทางคณะผู้จัดทำได้นำหลักการของการควบคุมคุณภาพมาประยุกต์ใช้ นั่นก็คือ ทางคณะผู้จัดทำไม่สามารถที่จะไปทำการตรวจสอบสมรรถนะของ Rotary Positioner ทุกตัวในโรงงานได้ จึงได้นำเอาการตรวจสอบแบบสุ่มตัวอย่าง (Sampling Inspection) มาใช้ การสุ่มตัวอย่างเป็นการนำ เอาวิชาการทางสถิติมาประยุกต์ใช้หลักพื้นฐานง่าย ๆ ก็คือ ต้องกำหนดขนาดของชุด (Lot size) หรือ (population size) ขนาดของตัวอย่างที่จะสุ่ม (Sample size) และกำหนดระดับเป้าหมายมาตรฐานที่ยอมรับ ซึ่งในการควบคุมคุณภาพนี้เรียกว่า Assurance of quality หรือ Acceptable Quality Level (AQL) เมื่อได้กำหนดจุดมุ่งหมายและข้อกำหนดไว้แล้ว ก็จะดำเนินการตรวจสอบไปตามจุดมุ่งหมายที่วางไว้ ยกตัวอย่าง เช่น ถ้าเรากำหนด lot size ไว้ 100 ชิ้น Sample size 30 ชิ้น และ AQL 95% เมื่อตรวจสอบแล้วปรากฏว่า ได้ของคุณภาพถูกต้องจำนวน 25 ชิ้น (95% ของ 30 ชิ้นคือ 28.5 ชิ้น) หมายความว่าการตรวจสอบคุณภาพสำหรับ lot นั้น ไม่ผ่านการยอมรับ แต่ถ้าหากการทดสอบคุณภาพนั้นพบว่าอยู่ในคุณภาพที่ต้องการ 29 ชิ้น ย่อมหมายความว่าสินค้าทั้ง lot นั้นเป็นที่ยอมรับ

ในการทดลองครั้งนี้นั้น ทางคณะผู้จัดทำได้กำหนดให้ทำการเขียนสัญญาณเซอร์โวไว้ที่ 5 ครั้ง ต่อ Rotary Positioner 1 ตัว และกำหนดค่า AQL ให้เป็น 80% ก็จะได้ว่า ถ้า Rotary Positioner สามารถเขียนสัญญาณเซอร์โวผ่าน 4 ครั้งขึ้นไป (80% ของ 5ครั้ง คือ 4 ครั้ง) เราก็จะยอมรับว่า Rotary Positioner เป็นตัวที่ดี และถ้าน้อยกว่า 4 ครั้ง ก็จะเป็นตัวที่เสีย ซึ่งจะเป็นการยืนยันผลการทดลองอีกทางหนึ่ง ว่า Rotary Positioner ที่ได้นำมาทำการทดลองนั้นเป็นตัวที่ดี และเสียจริง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 4.3 ตารางยืนยันผลการทดลองหา Rotary Positioner ดีหรือเสีย โดยใช้หลักการควบคุมคุณภาพ (QC) ที่ 100 track

Rotary Positioner	จำนวนครั้งที่ผ่าน	จำนวนครั้งที่ไม่ผ่าน	สรุป Rotary Positioner ดี หรือ เสีย
Good 1 (S/N 33xxx-M4x)	5	-	ดี
Good 2 (S/N 35xxx-Mx)	4	1	ดี
Good 3 (S/N 31xxx-Mx)	5	-	ดี
Bad 1 (S/N 37xxx-Nx)	-	5	เสีย
Bad 2 (S/N 33xxx-Mx)	-	5	เสีย
Bad 3 (S/N 35xxx-Mx)	-	5	เสีย

ตารางที่ 4.4 ตารางยืนยันผลการทดลองหา Rotary Positioner ดีหรือเสีย โดยใช้หลักการควบคุมคุณภาพ (QC) ที่ 400 track

Rotary Positioner	จำนวนครั้งที่ผ่าน	จำนวนครั้งที่ไม่ผ่าน	สรุป Rotary Positioner ดี หรือ เสีย
Good 1 (S/N 33xxx-M4x)	5	-	ดี
Good 2 (S/N 35xxx-Mx)	5	-	ดี
Good 3 (S/N 31xxx-Mx)	5	-	ดี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 4.4 (ต่อ) ตารางยืนยันผลการทดลองหา Rotary Positioner ดีหรือเสีย โดยใช้หลักการควบคุมคุณภาพ (QC) ที่ 400 track

Rotary Positioner	จำนวนครั้งที่ผ่าน	จำนวนครั้งที่ไม่ผ่าน	สรุป Rotary Positioner ดี หรือ เสีย
Bad 1 (S/N 37xxx-Nx)	1	4	เสีย
Bad 2 (S/N 33xxx-Mx)	-	5	เสีย
Bad 3 (S/N 35xxx-Mx)	1	4	เสีย

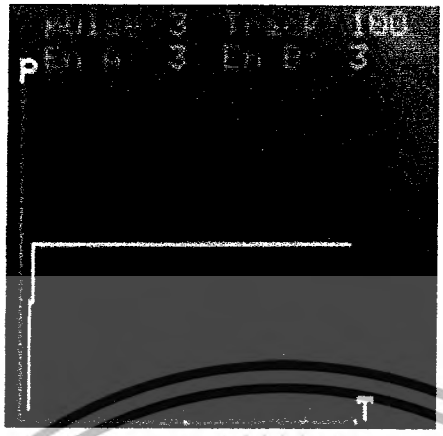
4.3.4 ผลจากการเคลื่อนที่ไป 100 Track

- Good 1 (S/N 33xxx-M4x)

ภาพที่ 4.6 การเคลื่อนที่ไป 100 Track ของ Good 1 (S/N 33xxx-M4x)

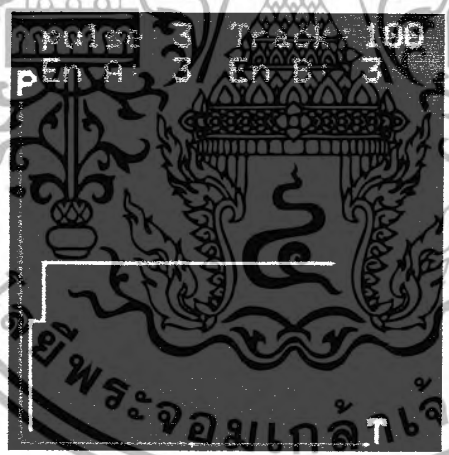
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- Good 2 (S/N 35xxx-Mx)



ภาพที่ 4.7 การเคลื่อนที่ไป 100 Track ของ Good 2 (S/N 35xxx-Mx)

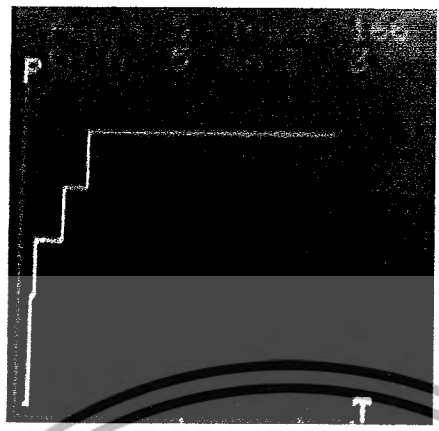
- Good 3 (S/N 31xxx8-Mx)



ภาพที่ 4.8 การเคลื่อนที่ไป 100 Track ของ Good 3 (S/N 31xxx-Mx)

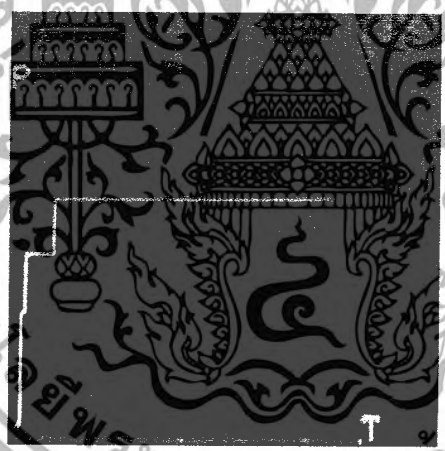
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- Bad 1 (S/N 37xxx-Nx)



ภาพที่ 4.9 การเคลื่อนที่ไป 100 Track ของ Bad 1(S/N 37xxx-Nx)

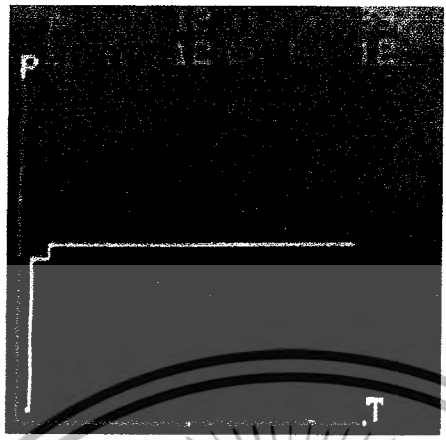
- Bad 2 (S/N 33xxx-Mx)



ภาพที่ 4.10 การเคลื่อนที่ไป 100 Track ของ Bad 2 (S/N 33xxx-Mx)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- Bad 1 (S/N 37xxx-Nx)



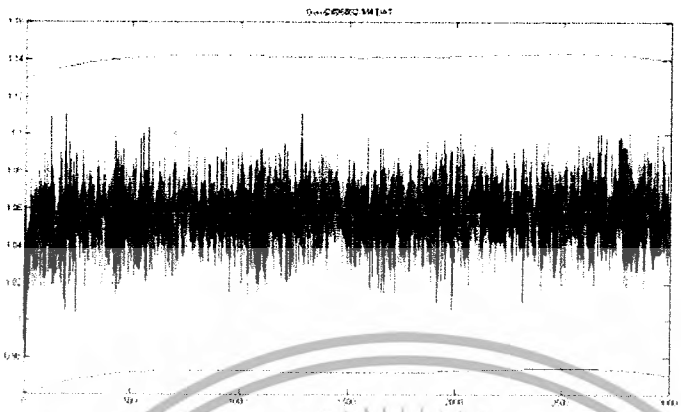
ภาพที่ 4.15 การเคลื่อนที่ไป 400 Track ของ Bad 1 (S/N 37xxx-Nx)

- Bad 2 (S/N 33xxx-Mx)

ภาพที่ 4.16 การเคลื่อนที่ไป 400 Track ของ Bad 2 (S/N 33xxx-Mx)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- Good 2 (S/N 35xxx-Mx)



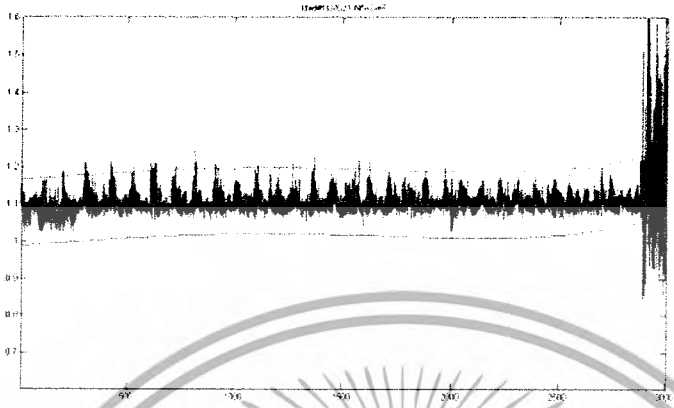
ภาพที่ 4.19 กราฟ APC Step จาก Good 2 (S/N 35xxx-Mx)

- Good 3 (S/N 31xxx-Mx)

ภาพที่ 4.20 กราฟ APC Step จาก Good 3 (S/N 31xxx-Mx)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- Bad 1 (S/N 37xxx-Nx)



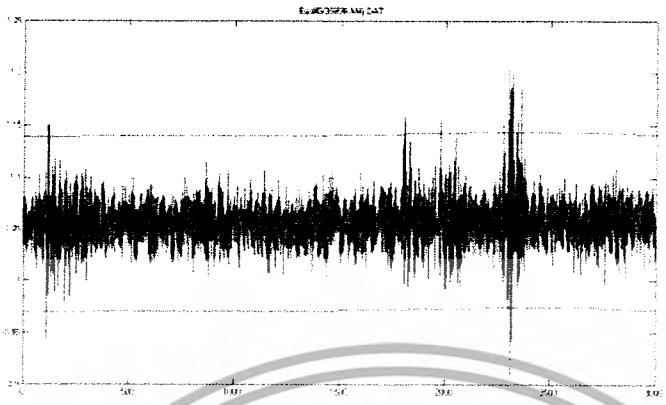
ภาพที่ 4.21 กราฟ APC Step จาก Bad 1 (S/N 37xxx-Nx)

- Bad 2 (S/N 33xxx-Mx)

ภาพที่ 4.22 กราฟ APC Step จาก Bad 2 (S/N 33xxx-Mx)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- Bad 3 (S/N 35xxx-Mx)



ภาพที่ 4.23 กราฟ APC Step จาก Bad 3 (S/N 35xxx-Mx)

จากกราฟ APC Step ข้างต้น สามารถอธิบายได้ว่า ถ้าเส้นกราฟสีน้ำเงินอยู่ภายในบริเวณเส้นสีแดงแสดงว่า Rotary Positioner ตัวนั้นเป็นตัวที่ดี แต่ถ้าเส้นกราฟสีน้ำเงินอยู่นอกเส้นสีแดงก็จะถือว่า Rotary Positioner ตัวนั้นเป็นตัวที่เสีย

จากการยืนยันผลการทดลองด้วยวิธีการทดสอบแบบ APC อีกครั้งหนึ่งนั้น ตามผลการทดลองดังข้างต้น สามารถสรุปได้ว่า เครื่องทดสอบสมรรถนะของ Rotary Positioner ที่ผู้วิจัยได้ทำการคิดค้นขึ้นมานั้นสามารถยืนยันได้ว่า Rotary Positioner ดีและเสียจริง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

การทดลองเพื่อทดสอบเครื่องทดสอบสมรรถนะของ Rotary Positioner โดยทดลอง move track จำนวน 100 track และ 400 track เครื่องทดสอบสมรรถนะของ Rotary Positioner จะแสดงจำนวน Pulse และจำนวนแทรค แสดงผลที่จอ LCD Graphic พร้อมทั้งพล็อตกราฟการเคลื่อนที่ของ Rotary Positioner พอสรุปได้ดังนี้

1. พิจารณาเส้นกราฟที่ได้ ถ้าเส้นกราฟที่ได้มีความราบเรียบตลอดช่วงการวัด สรุปได้ว่า Rotary Positioner ตัวที่นำมาทดสอบ มีสมรรถนะที่ดี ถ้าเส้นกราฟที่ได้มีการกระโดด แสดงว่า Rotary Positioner ตัวที่นำมาทดสอบเสีย

2. จำนวน pulse ที่ปรากฏบนจอแสดงผล LCD Graphic เป็นไปตามมาตรฐานของ Rotary Positioner ตัวที่ดี (Serial Number 10xxx-4x) จากการทดลองสรุปได้ว่า ถ้าเป็น Rotary Positioner ตัวที่ดี สัญญาณ Pulse ที่แสดงออกมา จะมีค่า 3 Pulse (สำหรับ move track 100 track) และ 11 Pulse (สำหรับ move track 400 track) ถ้า Rotary Positioner ตัวที่เสีย ค่าของสัญญาณ Pulse ที่แสดงจะไม่เป็นไปตามค่าที่ได้กล่าวมา

5.2 ข้อเสนอแนะ

การทดสอบตามที่ทางกลุ่มการวิจัยของเราได้คิดค้นขึ้นมานั้น เป็นวิธีที่จะช่วยให้การตรวจสอบเครื่อง Rotary Positioner ของทางบริษัทฮิตาชิ โกลบอล สตอเรจ เทคโนโลยีส์ (ประเทศไทย) จำกัด มีประสิทธิภาพมากยิ่งขึ้นทั้งเป็นการประหยัดเวลาช่วยในการลดต้นทุนและยังสามารถเพิ่มกำลังการผลิตได้อีกด้วย โดยกลุ่มผู้ทำการวิจัยคาดว่าโครงงานชิ้นนี้จะมีความสมบูรณ์มากขึ้นหากพัฒนาโครงงานชิ้นนี้ต่อไปดังนี้

1. จะเป็นการดียิ่งขึ้นหากสามารถพัฒนาให้เครื่องทดสอบสมรรถนะ Rotary Positioner สามารถที่จะวัดสัญญาณจากการ move track ไปที่แทรคเท่าใดก็ได้ภายในแผ่นดิสก์

2. เพิ่มฟังก์ชั่นในการทดสอบสมรรถนะแบบต่างๆ เพื่อเพิ่มความเชื่อมั่นมากยิ่งขึ้น

3. ปรับแต่งกล่องเครื่องมือให้มีขนาดกระทัดรัด หน้าจอแสดงผลมีขนาดใหญ่และละเอียดมากยิ่งขึ้นและสามารถใช้งานได้ สะดวกต่อการนำไปใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

[1] Hitachi inspire the next, "Hard Disk Drive SpecificationDeskstar7K1000, CinemaStar7K1000,UltraStarA7K1000" on 27 June 2007.

[2] Canon, "ROTARY POSITIONING SYSTEM" OPERATION MANUAL, 2001.11.27, pp. 29-41.

[3] STW Manufacturing Maintenance Instruction , " Theory " , IBM.

[4] http://www.canon.com/optoelectro/1_Laser_Rotary/1-F.html

[5] http://thaieleczone.com/index.php?option=com_content&task=view&id=29&Itemid=54

[6] http://202.28.94.55/web/322461/2550/report/g4/15result1.html

[7] http://web.ku.ac.th/schoolnet/snet1/hardware/hdisk.htm

[8] http://www.tws.ac.th/thoenwit/Library/ruen%20computer/www.ruencom.com/hdd.htm

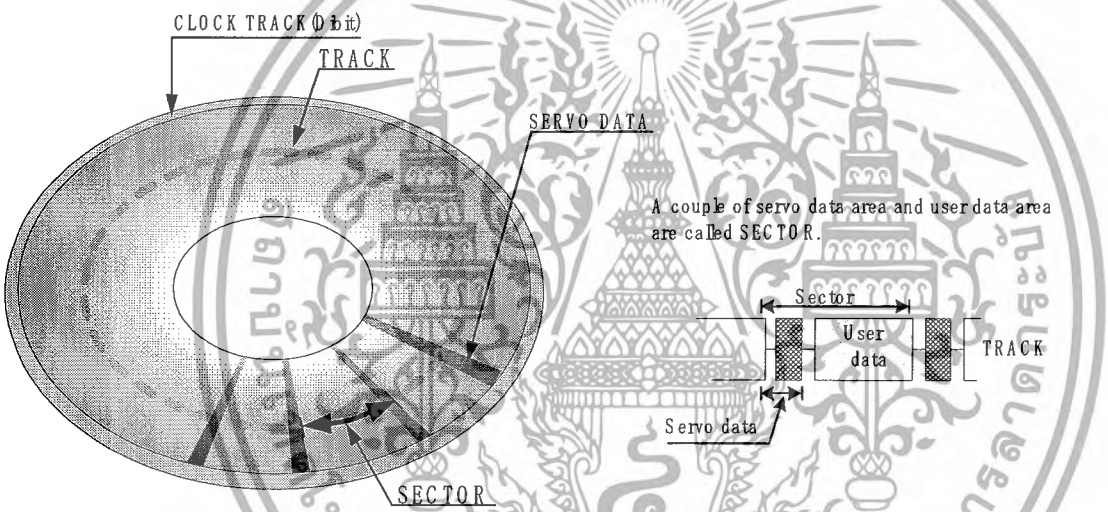
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

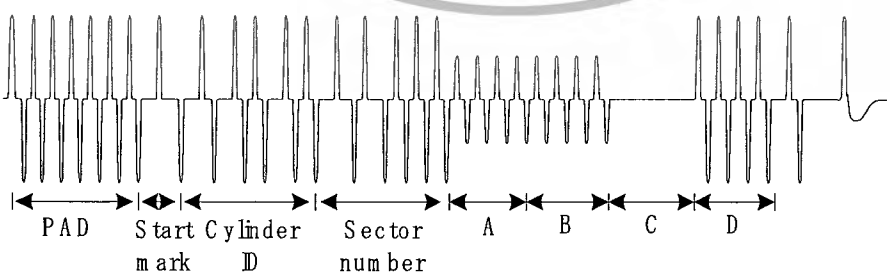
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.1 What is Servo Track Writer?

It is designed to write sector servo information on every disk of a newly assembled Disk Enclosure.
The servo information is required so that the file electronics are able to position the product heads correctly to read or write data.



The image of each servo pattern is shown as below.



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

PAD: To fill the area between servo area and servo area, simple positive and negative pulse are written.

Start mark: Special pattern to indicate start of servo pattern.

Cylinder ID: Physical cylinder ID is recorded on this area.

Sector number: Sector number is recorded on this area.

Burst A,B,C,D: This special pattern indicates track center position. Amplitude of A and B is balanced, when head is positioning on center of track. Amplitude C and D is balanced, when head is positioning on boundary of track.

This amplitude unbalance of A and B is measured by 256 bit resolution.

(Resolution of track pitch 3.5um is 3.5/256 = 0.014um)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

APPENDIX 1 SPECIFICATIONS

Configuration

1) System controller SC-01H/PC

2) Encoder-Motor KP-1M

Performance

1) Stroke(guaranteed)	±20 degrees (Z phase center)
2) Resolution	0.003906arc-sec
3) On-track stability	±0.05 arc-sec
4) Load inertia	100 gcm2 or less (SI system of units)
5) Settling time	Less than 8 msec (at 36 arc-sec step , LQ MODE) (with a 80 gcm^2 inertia load attached)
6) Maximum Angular Speed	60 degree/sec
7) Inertia of motor & encoder	48 gcm^2 (SI system of units)
8) External load	250gfcm or less
9) Error between pitches	Smaller than or equal to 20[nm] at 20[mm] radius

Operating Environment

1) Power supply	+5V: 1.5A +12V: max. 1.5A -12V: 50mA
2) Operating temperature	23 ± 5 °C
3) Storage temperature	0~60 °C , 20~85% RH (no condensation of moisture)

Weight and External Dimensions (L x W x H mm)

1) SC-01H/PC	330W x120W x *36H approx. 0.4 kg * requires two slot space.
2) Encoder-Motor KP-1M	80L x 80W x 117H cable length;0.5 m approx. 1.2 kg

Attachments

1. Extension cable for Motor/Encoder (cable length ; 2.0 m x 2)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

APPENDIX 2 CHECK PIN INFORMATION

MAIN BOARD

CP1 Test input (same as BNC)

CP2 Encoder phaseA after gain calibration

CP3 Encoder phaseB

CP4 Encoder phaseA

CP5 Monitor output (same as BNC)

CP6 Encoder phaseB after gain calibration

CP7 2.0V reference

CP8 Motor current reference

CP9 Encoder phaseB pulse

CP10 Encoder phaseA pulse

CP11 Z phase

CP12 MOVE status output (High when moving phs1.2 or later)

CP13 Sample clock of servo loop

CP14 Reserved

CP15 Z phase pulse (High at CCW)

CP16 Counter clear pulse

SUB BOARD (AH9-0037)

CP4 /INPOS

CP12 Thermistor output (4.7V at normal, 3.3V at alarm)

CP3 BUSY (phs1.2c or later)

CP2 TOUCH (phs1.2c or later)

SUB BOARD (AG9-0343)

TP8 Source input point of loop frequency response

TP10 Output of loop frequency response

TP17 /INPOSITION

TP18 BUSY

TP19 TOUC

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Code Program

```
#include "ADuC7024.H"
#include <stdio.h>
#include <math.h>
#include "font.h"
#include "pcf8833.h"

int j=0,m=0;
unsigned int xaxis=8,yaxis=8,p_xaxis=8,p_yaxis=8,i;
const unsigned char FONT6x8[97][8];
const unsigned char FONT8x8[97][8];
const unsigned char FONT8x16[97][16];
void SetTime(void);
void analog_graph(void);
void WriteSpiCommand(unsigned int data);
void WriteSpiData(unsigned int data);
void Backlight(unsigned char state);
void InitLcd(void);
void LCDWrite130x130bmp(void);
void LCDClearScreen(void);
void LCDSetXY(int x, int y);
void LCDSetPixel(int x, int y, int color);
void LCDSetLine(int x1, int y1, int x2, int y2, int color);
void LCDSetRect(int x0, int y0, int x1, int y1, unsigned char fill, int color);
void LCDSetCircle(int x0, int y0, int radius, int color);
void LCDPutChar(char c, int x, int y, int size, int fcolor, int bcolor);
void LCDPutStr(char *pString, int x, int y, int Size, int fColor, int bColor);
void Delay (unsigned long a);
void Pla_setup(void);
void monitor(void);
unsigned char Tx[10],Ax[10],Bx[10],Cx[10],Dx[10];
unsigned int i=0,k=0;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
unsigned int count_encoder=0,count_encoder1=0;
unsigned int Track_count=0;
void IRQ_Handler (void) __irq;
int count=0,Data=0;
int Track[200];
unsigned long sw;
int Track_Mode=1;
int Datacheck;
int Plot,Sampling=0;
int True_Track=0;
unsigned int TrackData=100;
unsigned int Time=0;
unsigned int check=0;
void Time_Out(void);
void Calculate(void);
void WriteSpiCommand(volatile unsigned int command)
{
 unsigned char Bit = 0;                    // Bit Counter

LCD_CS_HIGH();                             // Makesure CS = Disable
 LCD_SCLK_LOW();                           // Prepared SCLK

 LCD_CS_LOW();                             // Start CS = Enable
 LCD_SDATA_LOW();                          // D/C# = 0(Command)
 LCD_SCLK_HIGH();                          // Strobe Signal Bit(SDATA)
 LCD_SCLK_LOW();                           // Standby SCLK

 for (Bit = 0; Bit < 8; Bit++)             // 8 Bit Write
 {
  LCD_SCLK_LOW();                          // Standby SCLK

  if((command&0x80)>>7) LCD_SDATA_HIGH();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        else LCD_SDATA_LOW();

        LCD_SCLK_HIGH();                    // Strobe Signal Bit(SDATA)

        command <<= 1;                      // Next Bit Data
 }
 LCD_SCLK_LOW();                            // Standby SCLK
 LCD_CS_HIGH();                             // Stop CS = Disable
}
void WriteSpiData(volatile unsigned int data)
{
 unsigned char Bit = 0;                     // Bit Counter

 LCD_CS_HIGH();                             // Makesure CS = Disable
 LCD_SCLK_LOW();                            // Standby SCLK

 LCD_CS_LOW();                              // Start CS = Enable
 LCD_SDATA_HIGH();                          // D/C# = 1(Data)
 LCD_SCLK_HIGH();                           // Strobe Signal Bit(SDATA)
 LCD_SCLK_LOW();                            // Standby SCLK

 for (Bit = 0; Bit < 8; Bit++)              // 8 Bit Write
 {
  LCD_SCLK_LOW();                           // Standby SCLK

        if((data&0x80)>>7) LCD_SDATA_HIGH();
        else LCD_SDATA_LOW();

        LCD_SCLK_HIGH();                    // Strobe Signal Bit(SDATA)

        data <<= 1;                         // Next Bit Data
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 LCD_SCLK_LOW();                    // Standby SCLK
 LCD_CS_HIGH();                     // Stop CS = Disable
}
void Backlight(unsigned char state)
{
 if (state == 1)
   LCD_BL_HIGH();                   // Black Light ON
 else
   LCD_BL_LOW();                    // Black Light OFF
}
void InitLcd(void)
{
 // Hardware reset
 LCD_RESET_LOW();                   // Start Reset
 Delay(20000);                      // Reset Pulse Time
 LCD_RESET_HIGH();                  // Release Reset
 Delay(20000);

 // Sleep out (command 0x11)
 WriteSpiCommand(SLEEPOUT);

 //Inversion on (command 0x20)
 WriteSpiCommand(INVOFF); // seems to be required for this controller

 // Color Interface Pixel Format (command 0x3A)
 WriteSpiCommand(COLMOD);
 WriteSpiData(0x03); // 0x03 = 12 bits-per-pixel

 // Memory access controler (command 0x36)
 WriteSpiCommand(MADCTL);
 WriteSpiData(0x40); // 0xC0 = mirror x and y, reverse rgb
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  // Write contrast (command 0x25)
  WriteSpiCommand(SETCON);
  WriteSpiData(0x30); // contrast 0x30
  Delay(2000);

  // Display On (command 0x29)
  WriteSpiCommand(DISPON);
}
void LCDClearScreen(void)
{
  long i; // loop counter

  // Row address set (command 0x2B)
  WriteSpiCommand(PASET);
  WriteSpiData(0);
  WriteSpiData(131);

  // Column address set (command 0x2A)
  WriteSpiCommand(CASET);
  WriteSpiData(0);
  WriteSpiData(131);

  // set the display memory to BLACK
  WriteSpiCommand(RAMWR);
  for (i = 0; i < ((132 * 132) / 2); i++)
  {
    WriteSpiData((BLACK >> 4) & 0xFF);
    WriteSpiData(((BLACK & 0xF) << 4) | ((BLACK >> 8) & 0xF));
    WriteSpiData(BLACK & 0xFF);
  }
}
void LCDSetXY(int x, int y)
```

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
{
  // Row address set (command 0x2B)
  WriteSpiCommand(PASET);
  WriteSpiData(x);
  WriteSpiData(x);

  // Column address set (command 0x2A)
  WriteSpiCommand(CASET);
  WriteSpiData(y);
  WriteSpiData(y);
}
void LCDSetPixel(int x, int y, int color)
{
  LCDSetXY(x, y);
  WriteSpiCommand(RAMWR);
  WriteSpiData((unsigned char)((color >> 4) & 0xFFFF));
  WriteSpiData((unsigned char)(((color & 0x0F) << 4) | 0x00));
  WriteSpiCommand(NOP);
}
void LCDSetLine(int x0, int y0, int x1, int y1, int color)
{
  int dy = y1 - y0;
  int dx = x1 - x0;
  int stepx, stepy;
  if (dy < 0) { dy = -dy; stepy = -1; } else { stepy = 1; }
  if (dx < 0) { dx = -dx; stepx = -1; } else { stepx = 1; }
  dy <<= 1; // dy is now 2*dy
  dx <<= 1; // dx is now 2*dx
  LCDSetPixel(x0, y0, color);
  if (dx > dy)
  {
    int fraction = dy - (dx >> 1); // same as 2*dy - dx
```

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    while (x0 != x1)
    {
     if (fraction >= 0)
     {
      y0 += stepy;
      fraction -= dx;                    // same as fraction -= 2*dx
     }
     x0 += stepx;
     fraction += dy;                     // same as fraction -= 2*dy
     LCDSetPixel(x0, y0, color);
    }
   }
   else
   {
    int fraction = dx - (dy >> 1);
    while (y0 != y1)
          {
     if (fraction >= 0)
           {
      x0 += stepx;
      fraction -= dy;
     }
     y0 += stepy;
     fraction += dx;
     LCDSetPixel(x0, y0, color);
    }
   }
}
void LCDSetRect(int x0, int y0, int x1, int y1, unsigned char fill, int color)
{
 int xmin, xmax, ymin, ymax;
 int i;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// check if the rectangle is to be filled
if (fill == FILL)
{
 // best way to create a filled rectangle is to define a drawing box
 // and loop two pixels at a time
 // calculate the min and max for x and y directions
 xmin = (x0 <= x1) ? x0 : x1;
 xmax = (x0 > x1) ? x0 : x1;
 ymin = (y0 <= y1) ? y0 : y1;
 ymax = (y0 > y1) ? y0 : y1;

 // specify the controller drawing box according to those limits
 // Row address set (command 0x2B)
 WriteSpiCommand(PASET);
 WriteSpiData(xmin);
 WriteSpiData(xmax);

 // Column address set (command 0x2A)
 WriteSpiCommand(CASET);
 WriteSpiData(ymin);
 WriteSpiData(ymax);

 // WRITE MEMORY
 WriteSpiCommand(RAMWR);

 // loop on total number of pixels / 2
 for (i = 0; i < ((((xmax - xmin + 1) * (ymax - ymin + 1)) / 2) + 1); i++)
        {
  // use the color value to output three data bytes covering two pixels
  WriteSpiData((color >> 4) & 0xFF);
  WriteSpiData(((color & 0xF) << 4) | ((color >> 8) & 0xF));
  WriteSpiData(color & 0xFF);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    }
   }
   else
   {
    // best way to draw un unfilled rectangle is to draw four lines
    LCDSetLine(x0, y0, x1, y0, color);
    LCDSetLine(x0, y1, x1, y1, color);
    LCDSetLine(x0, y0, x0, y1, color);
    LCDSetLine(x1, y0, x1, y1, color);
   }
}
void LCDSetCircle(int x0, int y0, int radius, int color)
{
 int f = 1 - radius;
 int ddF_x = 0;
 int ddF_y = -2 * radius;
 int x = 0;
 int y = radius;
 LCDSetPixel(x0, y0 + radius, color);
 LCDSetPixel(x0, y0 - radius, color);
 LCDSetPixel(x0 + radius, y0, color);
 LCDSetPixel(x0 - radius, y0, color);
 while (x < y)
 {
  if (f >= 0)
        {
    y--;
    ddF_y += 2;
    f += ddF_y;
   }
  x++;
  ddF_x += 2;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    f += ddF_x + 1;
    LCDSetPixel(x0 + x, y0 + y, color);
    LCDSetPixel(x0 - x, y0 + y, color);
    LCDSetPixel(x0 + x, y0 - y, color);
    LCDSetPixel(x0 - x, y0 - y, color);
    LCDSetPixel(x0 + y, y0 + x, color);
    LCDSetPixel(x0 - y, y0 + x, color);
    LCDSetPixel(x0 + y, y0 - x, color);
    LCDSetPixel(x0 - y, y0 - x, color);
   }
}
void LCDPutChar(char c, int x, int y, int size, int fColor, int bColor)
{
 extern const unsigned char FONT6x8[97][8];
 extern const unsigned char FONT8x8[97][8];
 extern const unsigned char FONT8x16[97][16];
 int i,j;
 unsigned int nCols;
 unsigned int nRows;
 unsigned int nBytes;
 unsigned char PixelRow;
 unsigned char Mask;
 unsigned int Word0;
 unsigned int Word1;
 unsigned char *pFont;
 unsigned char *pChar;
 unsigned char *FontTable[] = {(unsigned char *)FONT6x8,
 (unsigned char *)FONT8x8,
 (unsigned char *)FONT8x16};

 // get pointer to the beginning of the selected font table
 pFont = (unsigned char *)FontTable[size];
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// get the nColumns, nRows and nBytes
nCols = *pFont;
nRows = *(pFont + 1);
nBytes = *(pFont + 2);

// get pointer to the last byte of the desired character
pChar = pFont + (nBytes * (c - 0x1F)) + nBytes - 1;

// Row address set (command 0x2B)
WriteSpiCommand(PASET);
WriteSpiData(x);
WriteSpiData(x + nRows - 1);

// Column address set (command 0x2A)
WriteSpiCommand(CASET);
WriteSpiData(y);
WriteSpiData(y + nCols - 1);

// WRITE MEMORY
WriteSpiCommand(RAMWR);
// loop on each row, working backwards from the bottom to the top
for (i = nRows - 1; i >= 0; i--)
{
 // copy pixel row from font table and then decrement row
 PixelRow = *pChar--;

 // loop on each pixel in the row (left to right)
 // Note: we do two pixels each loop
 Mask = 0x80;
 for (j = 0; j < nCols; j += 2)
        {
 // if pixel bit set, use foreground color; else use the background color
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      // now get the pixel color for two successive pixels
      if ((PixelRow & Mask) == 0)
        Word0 = bColor;
      else
        Word0 = fColor;
      Mask = Mask >> 1;

      if ((PixelRow & Mask) == 0)
        Word1 = bColor;
      else
        Word1 = fColor;
      Mask = Mask >> 1;

      // use this information to output three data bytes
      WriteSpiData((Word0 >> 4) & 0xFF);
      WriteSpiData(((Word0 & 0xF) << 4) | ((Word1 >> 8) & 0xF));
      WriteSpiData(Word1 & 0xFF);
    }
  }

  // terminate the Write Memory command
  WriteSpiCommand(NOP);
}
void LCDPutStr(char *pString, int x, int y, int Size, int fColor, int bColor)
{
  // loop until null-terminator is seen
  while (*pString != 0x00)
  {
    // draw the character
    LCDPutChar(*pString++, x, y, Size, fColor, bColor);

    // advance the y position
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    if (Size == SMALL)
    y = y + 6;
    else if (Size == MEDIUM)
    y = y + 8;
    else
    y = y + 8;

    // bail out if y exceeds 131
    if (y > 131) break;
  }
}
void Delay (unsigned long a)
{
  while (--a!=0);
}
void main(void)
{
 int TempColor[11] = { WHITE, BLACK, RED, GREEN, BLUE, CYAN,
                MAGENTA, YELLOW, BROWN, ORANGE, PINK };

 char *TempChar[11] = { "White ","Black ","Red   ","Green ","Blue  ","Cyan  ",
                "Magenta", "Yellow ", "Brown  ", "Orange ", "Pink  " };

 /* Config P2[0..7] Interface LCD */
 GP3CON  =   0x00;                    // P2[0..7] = GPIO
 GP3DAT  =   0xFF000000;              // P2[0..7] = Output
 LCD_BL_DIR();                        // BL = Output
 LCD_CS_DIR();                        // CS = Output
 LCD_SCLK_DIR();                      // SCLK = Output
 LCD_SDATA_DIR();                     // SDATA = Output
 LCD_RESET_DIR();                     // RESET = Output
 LCD_SCLK_LOW();                      // Standby SCLK
```

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
LCD_CS_HIGH();                          // Disable CS
LCD_SDATA_HIGH();                       // Standby SDATA
LCD_BL_HIGH();
InitLcd();                              // Initial LCD
LCDClearScreen();
Track[0]= 1;
Data = 1;
if(Data <= 200)Pla_setup();

while(1)
 {
        sw=GP4DAT & 0xF0;
 i=0,k=0;
        if(sw == 0xF0)                  //no press switch
 {
         Datacheck = 0;                 //reset Data
  GP4SET = 0x00030000;

        if((count<100)&&j==1)
        { LCDPutStr("     ",120,50,SMALL ,BLACK,BLACK);j=0; }       //Clear colum

        if((count<10)&&m==1)
        { LCDPutStr("     ",120,50,SMALL ,BLACK,BLACK);m=0; }      //Clear colum
         sprintf(Dx,"%d",TrackData);
         LCDPutStr("Wait Pulse",  80, 20 , LARGE,YELLOW ,BLUE);
         LCDPutStr("MODE:",  40,10 ,LARGE ,YELLOW ,BLUE);
         LCDPutStr(Dx ,40,50,LARGE ,YELLOW ,BLUE);                  //Mode Setup
         LCDPutStr("TRACK", 40 ,74,LARGE,YELLOW,BLUE);
         monitor();
         check = 1;                     //call Monitor
        }                               // switch two press
        if(sw == 0xD0)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        {                                                   // software reset
         RSTSTA = 0x04;
        }
        if(sw == 0xB0)
        {
         Track_Mode = 1;                                    //Mode 100 Track
         TrackData  = 100;
        }
        if(sw == 0x70)
        {
         Track_Mode = 4;
         TrackData  = 400;                                  //Mode 400 Track
        }

        if(sw == 0xE0)
        {
         GP4CLR = 0x00030000;
    LCDClearScreen();                                       //Setpoint

        if(Plot>=200) Plot=0;
                //Plot in Buffer
        for(Plot=0;Plot<200;Plot++)
{
        if((count<100)&&j==1)
        { LCDPutStr("    ",120,50,SMALL ,BLACK,BLACK);j=0; }
        if((count<10)&&m==1)
        { LCDPutStr("    ",120,50,SMALL ,BLACK,BLACK);m=0; }

        if(Track[Plot]==0)      Track[Plot]=Track[Plot-1];
        xaxis = Track[Plot];
        monitor();
        analog_graph();                                     //call function
```

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        }
        }
        }
}

void analog_graph(void)
{
  if(i==0)                                        // Function plot graph
          {
           p_xaxis = xaxis;
           i++;
          }
        LCDSetPixel((8+xaxis),yaxis, GREEN);
  while(p_xaxis<xaxis)
        {
          LCDSetPixel((8+p_xaxis),p_yaxis,GREEN);
          p_xaxis++;
        }
        while(p_xaxis>xaxis)
        {
          LCDSetPixel((8+p_xaxis),p_yaxis,GREEN);
          p_xaxis--;
        }
        while(p_yaxis<yaxis)
        {
          LCDSetPixel((8+xaxis),p_yaxis,GREEN);
          p_yaxis++;
        }
  p_xaxis=xaxis;
        p_yaxis=yaxis;
        yaxis++;
        if(yaxis>105)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    {
     yaxis=8;
            p_yaxis=8;
            i=0;
            LCDClearScreen();
           }
          if(xaxis>105)
    {
     xaxis=100;
            p_xaxis=8;
            i=0;
           }
          if(p_xaxis>105)p_xaxis=8;

}
void SetTime(void)
{
//IRQ = My_IRQ_Function;
 IRQEN |= 0x00000008;                    // Enable Timer1 Trigger IRQ
Interrupt
 T1LD  =  8696;                          // 200us
 ;
 T1CON &= 0xFFFFFFF0;                    // Prescale = HCLK / 1
 T1CON &= 0xFFFFFFCF;                    // Format = Binary Counter
 T1CON |= 0x00000040;                    // Timer1 Mode = Periodic
 T1CON &= 0xFFFFFEFF;                    // Timer1 = Count Down
 T1CON &= 0xFFFFF1FF;                    // Timer1 Clock Source = HCLK
 T1CON |= 0x00000080;

 GP4CON = 0x00000000;                    // P4 = GPIO
 GP4DAT = 0x0F000000;                    // P4[7] = Output
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}
void Pla_setup(void)

{

   GP1CON = 0x3300;
        GP1DAT = 0x00;
   // Configure individual elements
   PLAELM2 = 0x002B;
        PLAELM3 = 0x002B;
   // Clk Source configuration
   PLACLK = 0x0000;
   // IRQ output configuration
   PLAIRQ = 0x1312;
   IRQEN  |= 0x00180000;
  //IRQSTA = 0x20;
}
void monitor(void)
{
   LCDPutStr("P", 106, 5, SMALL, WHITE, BLACK);
        LCDPutStr("T",   5,106,SMALL, WHITE, BLACK);
        sprintf(Tx,"%d",count);
        sprintf(Ax,"%d",count_encoder);
        sprintf(Bx,"%d",count_encoder1);
        sprintf(Cx,"%d",True_Track);
   LCDPutStr(Tx,120,50,SMALL ,MAGENTA ,BLACK);
        LCDPutStr(Ax,110,50,SMALL ,MAGENTA ,BLACK);
        LCDPutStr(Bx,110,104,SMALL ,MAGENTA ,BLACK);
        LCDPutStr(Cx,120,104,SMALL,MAGENTA ,BLACK);
        LCDPutStr("pulse:",120,12,SMALL ,ORANGE ,BLACK);
        LCDPutStr("En A:",110,12,SMALL ,ORANGE ,BLACK);
        LCDPutStr("En B:",110,66,SMALL ,ORANGE ,BLACK);
```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        LCDPutStr("Track:",120,66,SMALL ,ORANGE ,BLACK);
        LCDSetLine (5,5,105,5,RED);
        LCDSetLine (5,5,5,105,RED);


        LCDSetPixel(5,55,YELLOW);
    LCDSetPixel(5,105,YELLOW);
}
void Time_Out(void)
{
if(Data == 200)
{
  IRQCLR = 0xFFFFFFFF;
  IRQEN  = 0x00;
  PLAIRQ = 0x00;
  T1CLRI = 0;
}
}

void Calculate(void)
{
 Track_count =(((33*count)+1)/(Track_Mode*2));                    //Sampling
 True_Track = ((33*count)+1);
 if(Track_count>105)Track_count=105;
 if(count<0) Track_count=0;
 Track[Data] = Track_count;
 Data++;
 Time_Out();
 if(Data>200)
{
 Data=200;
 IRQCLR = 0xffffffff;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}
T1CLRI = 0;
}


void IRQ_Handler (void)__irq
{  if((IRQSIG & 0x00080000)!=0)                    //PLA interrupt
        {
    if(check==0) LCDPutStr("NOT READY",  80, 20 , LARGE,YELLOW ,BLUE);
                if(check==1)
                {
                 SetTime();
                 count_encoder++;
                 if((IRQSIG & 0x00080000)!=0)
                {
                    i=1;
                    while((i^k)==1)
                    {
                    count++;
                if(count>100)j=1;
                if(count>10) m=1;
                        break;
                        }
                        while((i^k)==0)
                        {
                        count--;
                         break;
                        }
                while((IRQSIG & 0x00080000)!=0)
                {
                while ((IRQSTA & 0x00000008) != 0) Calculate();
                }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            }
        }
}
  if((IRQSIG & 0x00100000)!=0)
        {
                if(check==0) LCDPutStr("NOT READY",  80, 20 , LARGE,YELLOW
,BLUE);
            if(check==1)
        {
                        SetTime();
                count_encoder1++;
                k=1;
                while((IRQSIG & 0x00100000)!=0)
                  {
                    while ((IRQSTA & 0x00000008) != 0) Calculate();
                  }
        }
        }
 if(((IRQSIG & 0x00080000)!=0)&&((IRQSIG & 0x00100000)==0)) k=0;

 if ((IRQSTA & 0x00000008) != 0)
        {
         while((IRQSTA & 0x00000008) != 0) Calculate();
        }
       return ;
        }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Font.h

```
const unsigned char FONT6x8[97][8] = {
 0x06,0x08,0x08,0x00,0x00,0x00,0x00,0x00, // columns, rows, num_bytes_per_char
 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,      // space 0x20
 0x20,0x20,0x20,0x20,0x20,0x00,0x20,0x00,      // !
 0x50,0x50,0x50,0x00,0x00,0x00,0x00,0x00,      // "
 0x50,0x50,0xF8,0x50,0xF8,0x50,0x50,0x00,      // #
 0x20,0x78,0xA0,0x70,0x28,0xF0,0x20,0x00,      // $
 0xC0,0xC8,0x10,0x20,0x40,0x98,0x18,0x00,      // %
 0x40,0xA0,0xA0,0x40,0xA8,0x90,0x68,0x00,      // &
 0x30,0x30,0x20,0x40,0x00,0x00,0x00,0x00,      // '
 0x10,0x20,0x40,0x40,0x40,0x20,0x10,0x00,      // (
 0x40,0x20,0x10,0x10,0x10,0x20,0x40,0x00,      // )
 0x00,0x20,0xA8,0x70,0x70,0xA8,0x20,0x00,      // *
 0x00,0x20,0x20,0xF8,0x20,0x20,0x00,0x00,      // +
 0x00,0x00,0x00,0x00,0x30,0x30,0x20,0x40,      // ,
 0x00,0x00,0x00,0xF8,0x00,0x00,0x00,0x00,      // -
 0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x00,      // .
 0x00,0x08,0x10,0x20,0x40,0x80,0x00,0x00,      // / (forward slash)
 0x70,0x88,0x88,0xA8,0x88,0x88,0x70,0x00,      // 0 0x30
 0x20,0x60,0x20,0x20,0x20,0x20,0x70,0x00,      // 1
 0x70,0x88,0x08,0x70,0x80,0x80,0xF8,0x00,      // 2
 0xF8,0x08,0x10,0x30,0x08,0x88,0x70,0x00,      // 3
 0x10,0x30,0x50,0x90,0xF8,0x10,0x10,0x00,      // 4
 0xF8,0x80,0xF0,0x08,0x08,0x88,0x70,0x00,      // 5
 0x38,0x40,0x80,0xF0,0x88,0x88,0x70,0x00,      // 6
 0xF8,0x08,0x08,0x10,0x20,0x40,0x80,0x00,      // 7
 0x70,0x88,0x88,0x70,0x88,0x88,0x70,0x00,      // 8
 0x70,0x88,0x88,0x78,0x08,0x10,0xE0,0x00,      // 9
 0x00,0x00,0x20,0x00,0x20,0x00,0x00,0x00,      // :
 0x00,0x00,0x20,0x00,0x20,0x20,0x40,0x00,      // ;
 0x08,0x10,0x20,0x40,0x20,0x10,0x08,0x00,      // <
```

เอกสารนี้สงวนลิขสิทธิ์ สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
0x00,0x00,0xF8,0x00,0xF8,0x00,0x00,0x00,    // =
0x40,0x20,0x10,0x08,0x10,0x20,0x40,0x00,    // >
0x70,0x88,0x08,0x30,0x20,0x00,0x20,0x00,    // ?
0x70,0x88,0xA8,0xB8,0xB0,0x80,0x78,0x00,    // @ 0x40
0x20,0x50,0x88,0x88,0xF8,0x88,0x88,0x00,    // A
0xF0,0x88,0x88,0xF0,0x88,0x88,0xF0,0x00,    // B
0x70,0x88,0x80,0x80,0x80,0x88,0x70,0x00,    // C
0xF0,0x88,0x88,0x88,0x88,0x88,0xF0,0x00,    // D
0xF8,0x80,0x80,0xF0,0x80,0x80,0xF8,0x00,    // E
0xF8,0x80,0x80,0xF0,0x80,0x80,0x80,0x00,    // F
0x78,0x88,0x80,0x80,0x98,0x88,0x78,0x00,    // G
0x88,0x88,0x88,0xF8,0x88,0x88,0x88,0x00,    // H
0x70,0x20,0x20,0x20,0x20,0x20,0x70,0x00,    // I
0x38,0x10,0x10,0x10,0x10,0x90,0x60,0x00,    // J
0x88,0x90,0xA0,0xC0,0xA0,0x90,0x88,0x00,    // K
0x80,0x80,0x80,0x80,0x80,0x80,0xF8,0x00,    // L
0x88,0xD8,0xA8,0xA8,0xA8,0x88,0x88,0x00,    // M
0x88,0x88,0xC8,0xA8,0x98,0x88,0x88,0x00,    // N
0x70,0x88,0x88,0x88,0x88,0x88,0x70,0x00,    // O
0xF0,0x88,0x88,0xF0,0x80,0x80,0x80,0x00,    // P 0x50
0x70,0x88,0x88,0x88,0xA8,0x90,0x68,0x00,    // Q
0xF0,0x88,0x88,0xF0,0xA0,0x90,0x88,0x00,    // R
0x70,0x88,0x80,0x70,0x08,0x88,0x70,0x00,    // S
0xF8,0xA8,0x20,0x20,0x20,0x20,0x20,0x00,    // T
0x88,0x88,0x88,0x88,0x88,0x88,0x70,0x00,    // U
0x88,0x88,0x88,0x88,0x88,0x50,0x20,0x00,    // V
0x88,0x88,0x88,0xA8,0xA8,0xA8,0x50,0x00,    // W
0x88,0x88,0x50,0x20,0x50,0x88,0x88,0x00,    // X
0x88,0x88,0x50,0x20,0x20,0x20,0x20,0x00,    // Y
0xF8,0x08,0x10,0x70,0x40,0x80,0xF8,0x00,    // Z
0x78,0x40,0x40,0x40,0x40,0x40,0x78,0x00,    // [
0x00,0x80,0x40,0x20,0x10,0x08,0x00,0x00,    // \ (back slash)
```

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
0x78,0x08,0x08,0x08,0x08,0x08,0x78,0x00,    // ]
0x20,0x50,0x88,0x00,0x00,0x00,0x00,0x00,    // ^
0x00,0x00,0x00,0x00,0x00,0x00,0xF8,0x00,    // _
0x60,0x60,0x20,0x10,0x00,0x00,0x00,0x00,    // ` 0x60
0x00,0x00,0x60,0x10,0x70,0x90,0x78,0x00,    // a
0x80,0x80,0xB0,0xC8,0x88,0xC8,0xB0,0x00,    // b
0x00,0x00,0x70,0x88,0x80,0x88,0x70,0x00,    // c
0x08,0x08,0x68,0x98,0x88,0x98,0x68,0x00,    // d
0x00,0x00,0x70,0x88,0xF8,0x80,0x70,0x00,    // e
0x10,0x28,0x20,0x70,0x20,0x20,0x20,0x00,    // f
0x00,0x00,0x70,0x98,0x98,0x68,0x08,0x70,    // g
0x80,0x80,0xB0,0xC8,0x88,0x88,0x88,0x00,    // h
0x20,0x00,0x60,0x20,0x20,0x20,0x70,0x00,    // i
0x10,0x00,0x10,0x10,0x10,0x90,0x60,0x00,    // j
0x80,0x80,0x90,0xA0,0xC0,0xA0,0x90,0x00,    // k
0x60,0x20,0x20,0x20,0x20,0x20,0x70,0x00,    // l
0x00,0x00,0xD0,0xA8,0xA8,0xA8,0xA8,0x00,    // m
0x00,0x00,0xB0,0xC8,0x88,0x88,0x88,0x00,    // n
0x00,0x00,0x70,0x88,0x88,0x88,0x70,0x00,    // o
0x00,0x00,0xB0,0xC8,0xC8,0xB0,0x80,0x80,    // p 0x70
0x00,0x00,0x68,0x98,0x98,0x68,0x08,0x08,    // q
0x00,0x00,0xB0,0xC8,0x80,0x80,0x80,0x00,    // r
0x00,0x00,0x78,0x80,0x70,0x08,0xF0,0x00,    // s
0x20,0x20,0xF8,0x20,0x20,0x28,0x10,0x00,    // t
0x00,0x00,0x88,0x88,0x88,0x98,0x68,0x00,    // u
0x00,0x00,0x88,0x88,0x88,0x50,0x20,0x00,    // v
0x00,0x00,0x88,0x88,0xA8,0xA8,0x50,0x00,    // w
0x00,0x00,0x88,0x50,0x20,0x50,0x88,0x00,    // x
0x00,0x00,0x88,0x88,0x78,0x08,0x88,0x70,    // y
0x00,0x00,0xF8,0x10,0x20,0x40,0xF8,0x00,    // z
0x10,0x20,0x20,0x40,0x20,0x20,0x10,0x00,    // {
0x20,0x20,0x20,0x00,0x20,0x20,0x20,0x00,    // |
```

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
0x40,0x20,0x20,0x10,0x20,0x20,0x40,0x00,      / }
0x40,0xA8,0x10,0x00,0x00,0x00,0x00,0x00,      // ~
0x70,0xD8,0xD8,0x70,0x00,0x00,0x00,0x00};     // DEL


const unsigned char FONT8x8[97][8] = {
0x08,0x08,0x08,0x00,0x00,0x00,0x00,0x00,      // columns, rows, num_bytes_per_char
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,      // space 0x20
0x30,0x78,0x78,0x30,0x30,0x00,0x30,0x00,      // !
0x6C,0x6C,0x6C,0x00,0x00,0x00,0x00,0x00,      // "
0x6C,0x6C,0xFE,0x6C,0xFE,0x6C,0x6C,0x00,      // #
0x18,0x3E,0x60,0x3C,0x06,0x7C,0x18,0x00,      // $
0x00,0x63,0x66,0x0C,0x18,0x33,0x63,0x00,      // %
0x1C,0x36,0x1C,0x3B,0x6E,0x66,0x3B,0x00,      // &
0x30,0x30,0x60,0x00,0x00,0x00,0x00,0x00,      // '
0x0C,0x18,0x30,0x30,0x30,0x18,0x0C,0x00,      // (
0x30,0x18,0x0C,0x0C,0x0C,0x18,0x30,0x00,      // )
0x00,0x66,0x3C,0xFF,0x3C,0x66,0x00,0x00,      // *
0x00,0x30,0x30,0xFC,0x30,0x30,0x00,0x00,      // +
0x00,0x00,0x00,0x00,0x00,0x18,0x18,0x30,      // ,
0x00,0x00,0x00,0x7E,0x00,0x00,0x00,0x00, /    / -
0x00,0x00,0x00,0x00,0x00,0x18,0x18,0x00,      // .
0x03,0x06,0x0C,0x18,0x30,0x60,0x40,0x00,      // / (forward slash)
0x3E,0x63,0x63,0x6B,0x63,0x63,0x3E,0x00,      // 0 0x30
0x18,0x38,0x58,0x18,0x18,0x18,0x7E,0x00,      // 1
0x3C,0x66,0x06,0x1C,0x30,0x66,0x7E,0x00,      // 2
0x3C,0x66,0x06,0x1C,0x06,0x66,0x3C,0x00,      // 3
0x0E,0x1E,0x36,0x66,0x7F,0x06,0x0F,0x00,      // 4
0x7E,0x60,0x7C,0x06,0x06,0x66,0x3C,0x00,      // 5
0x1C,0x30,0x60,0x7C,0x66,0x66,0x3C,0x00,      // 6
0x7E,0x66,0x06,0x0C,0x18,0x18,0x18,0x00,      // 7
0x3C,0x66,0x66,0x3C,0x66,0x66,0x3C,0x00,      // 8
0x3C,0x66,0x66,0x3E,0x06,0x0C,0x38,0x00,      // 9
```

เอกสารนี้สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
0x00,0x18,0x18,0x00,0x00,0x18,0x18,0x00,      // :
0x00,0x18,0x18,0x00,0x00,0x18,0x18,0x30,      // ;
0x0C,0x18,0x30,0x60,0x30,0x18,0x0C,0x00,      // <
0x00,0x00,0x7E,0x00,0x00,0x7E,0x00,0x00,      // =
0x30,0x18,0x0C,0x06,0x0C,0x18,0x30,0x00,      // >
0x3C,0x66,0x06,0x0C,0x18,0x00,0x18,0x00,      / ?
0x3E,0x63,0x6F,0x69,0x6F,0x60,0x3E,0x00,       // @ 0x40
0x18,0x3C,0x66,0x66,0x7E,0x66,0x66,0x00,      // A
0x7E,0x33,0x33,0x3E,0x33,0x33,0x7E,0x00,       // B
0x1E,0x33,0x60,0x60,0x60,0x33,0x1E,0x00,      // C
0x7C,0x36,0x33,0x33,0x33,0x36,0x7C,0x00,      // D
0x7F,0x31,0x34,0x3C,0x34,0x31,0x7F,0x00,      // E
0x7F,0x31,0x34,0x3C,0x34,0x30,0x78,0x00,      // F
0x1E,0x33,0x60,0x60,0x67,0x33,0x1F,0x00,      // G
0x66,0x66,0x66,0x7E,0x66,0x66,0x66,0x00,       // H
0x3C,0x18,0x18,0x18,0x18,0x18,0x3C,0x00,      // I
0x0F,0x06,0x06,0x06,0x66,0x66,0x3C,0x00,      // J
0x73,0x33,0x36,0x3C,0x36,0x33,0x73,0x00,      // K
0x78,0x30,0x30,0x30,0x31,0x33,0x7F,0x00,      // L
0x63,0x77,0x7F,0x7F,0x6B,0x63,0x63,0x00,      // M
0x63,0x73,0x7B,0x6F,0x67,0x63,0x63,0x00,      // N
0x3E,0x63,0x63,0x63,0x63,0x63,0x3E,0x00,      // O
0x7E,0x33,0x33,0x3E,0x30,0x30,0x78,0x00,      // P 0x50
0x3C,0x66,0x66,0x66,0x6E,0x3C,0x0E,0x00,      // Q
0x7E,0x33,0x33,0x3E,0x36,0x33,0x73,0x00,      // R
0x3C,0x66,0x30,0x18,0x0C,0x66,0x3C,0x00,      // S
0x7E,0x5A,0x18,0x18,0x18,0x18,0x3C,0x00,      // T
0x66,0x66,0x66,0x66,0x66,0x66,0x7E,0x00,      // U
0x66,0x66,0x66,0x66,0x66,0x3C,0x18,0x00,      // V
0x63,0x63,0x63,0x6B,0x7F,0x77,0x63,0x00,      // W
0x63,0x63,0x36,0x1C,0x1C,0x36,0x63,0x00,      // X
0x66,0x66,0x66,0x3C,0x18,0x18,0x3C,0x00,      // Y
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
0x7F,0x63,0x46,0x0C,0x19,0x33,0x7F,0x00,      // Z
0x3C,0x30,0x30,0x30,0x30,0x30,0x3C,0x00,      // [
0x60,0x30,0x18,0x0C,0x06,0x03,0x01,0x00,      // \ (back slash)
0x3C,0x0C,0x0C,0x0C,0x0C,0x0C,0x3C,0x00,      // ]
0x08,0x1C,0x36,0x63,0x00,0x00,0x00,0x00,       // ^
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,      // _
0x18,0x18,0x0C,0x00,0x00,0x00,0x00,0x00,      // ` 0x60
0x00,0x00,0x3C,0x06,0x3E,0x66,0x3B,0x00,      // a
0x70,0x30,0x3E,0x33,0x33,0x33,0x6E,0x00,      // b
0x00,0x00,0x3C,0x66,0x60,0x66,0x3C,0x00,      // c
0x0E,0x06,0x3E,0x66,0x66,0x66,0x3B,0x00,      // d
0x00,0x00,0x3C,0x66,0x7E,0x60,0x3C,0x00,      // e
0x1C,0x36,0x30,0x78,0x30,0x30,0x78,0x00,      // f
0x00,0x00,0x3B,0x66,0x66,0x3E,0x06,0x7C,      // g
0x70,0x30,0x36,0x3B,0x33,0x33,0x73,0x00,      // h
0x18,0x00,0x38,0x18,0x18,0x18,0x3C,0x00,      // i
0x06,0x00,0x06,0x06,0x06,0x66,0x66,0x3C,      // j
0x70,0x30,0x33,0x36,0x3C,0x36,0x73,0x00,      // k
0x38,0x18,0x18,0x18,0x18,0x18,0x3C,0x00,      // l
0x00,0x00,0x66,0x7F,0x7F,0x6B,0x63,0x00,      // m
0x00,0x00,0x7C,0x66,0x66,0x66,0x66,0x00,      // n
0x00,0x00,0x3C,0x66,0x66,0x66,0x3C,0x00,      // o
0x00,0x00,0x6E,0x33,0x33,0x3E,0x30,0x78,      // p 0x70
0x00,0x00,0x3B,0x66,0x66,0x3E,0x06,0x0F,      // q
0x00,0x00,0x6E,0x3B,0x33,0x30,0x78,0x00,      // r
0x00,0x00,0x3E,0x60,0x3C,0x06,0x7C,0x00,      // s
0x08,0x18,0x3E,0x18,0x18,0x1A,0x0C,0x00,      // t
0x00,0x00,0x66,0x66,0x66,0x66,0x3B,0x00,      // u
0x00,0x00,0x66,0x66,0x66,0x3C,0x18,0x00,      // v
0x00,0x00,0x63,0x6B,0x7F,0x7F,0x36,0x00,      // w
0x00,0x00,0x63,0x36,0x1C,0x36,0x63,0x00,      // x
0x00,0x00,0x66,0x66,0x66,0x3E,0x06,0x7C,      // y
```

ขอสงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
0x00,0x00,0x7E,0x4C,0x18,0x32,0x7E,0x00,     // z
0x0E,0x18,0x18,0x70,0x18,0x18,0x0E,0x00,     // {
0x0C,0x0C,0x0C,0x00,0x0C,0x0C,0x0C,0x00,     // |
0x70,0x18,0x18,0x0E,0x18,0x18,0x70,0x00,     // }
0x3B,0x6E,0x00,0x00,0x00,0x00,0x00,0x00,     // ~
0x1C,0x36,0x36,0x1C,0x00,0x00,0x00,0x00};    // DEL


 const unsigned char FONT8x16[97][16] = {
 0x08,0x10,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   //
columns, rows, nbytes
 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   //
space 0x20
 0x00,0x00,0x18,0x3C,0x3C,0x3C,0x18,0x18,0x18,0x00,0x18,0x18,0x00,0x00,0x00,0x00, // !
 0x00,0x63,0x63,0x63,0x22,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    // "
 0x00,0x00,0x00,0x36,0x36,0x7F,0x36,0x36,0x36,0x7F,0x36,0x36,0x00,0x00,0x00,0x00,   // #
 0x0C,0x0C,0x3E,0x63,0x61,0x60,0x3E,0x03,0x03,0x43,0x63,0x3E,0x0C,0x0C,0x00,0x00, // $
 0x00,0x00,0x00,0x00,0x00,0x61,0x63,0x06,0x0C,0x18,0x33,0x63,0x00,0x00,0x00,0x00,   // %
 0x00,0x00,0x00,0x1C,0x36,0x36,0x1C,0x3B,0x6E,0x66,0x66,0x3B,0x00,0x00,0x00,0x00, // &
 0x00,0x30,0x30,0x30,0x60,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   // '
 0x00,0x00,0x0C,0x18,0x18,0x30,0x30,0x30,0x30,0x18,0x18,0x0C,0x00,0x00,0x00,0x00,   // (
 0x00,0x00,0x18,0x0C,0x0C,0x06,0x06,0x06,0x06,0x0C,0x0C,0x18,0x00,0x00,0x00,0x00, // )
 0x00,0x00,0x00,0x00,0x42,0x66,0x3C,0xFF,0x3C,0x66,0x42,0x00,0x00,0x00,0x00,0x00, // *
 0x00,0x00,0x00,0x00,0x18,0x18,0x18,0xFF,0x18,0x18,0x18,0x00,0x00,0x00,0x00,0x00,   // +
 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x18,0x18,0x30,0x00,0x00,   // ,
 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   // -
 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x18,0x00,0x00,0x00,0x00,   // .
 0x00,0x00,0x01,0x03,0x07,0x0E,0x1C,0x38,0x70,0xE0,0xC0,0x80,0x00,0x00,0x00,0x00, // /
(forward slash)
 0x00,0x00,0x3E,0x63,0x63,0x63,0x6B,0x6B,0x63,0x63,0x63,0x3E,0x00,0x00,0x00,0x00,   // 0
0x30
 0x00,0x00,0x0C,0x1C,0x3C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x3F,0x00,0x00,0x00,0x00, //
```


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

0x00,0x00,0x3E,0x63,0x03,0x06,0x0C,0x18,0x30,0x61,0x63,0x7F,0x00,0x00,0x00,0x00, // 2

0x00,0x00,0x3E,0x63,0x03,0x03,0x1E,0x03,0x03,0x03,0x63,0x3E,0x00,0x00,0x00,0x00, // 3

0x00,0x00,0x06,0x0E,0x1E,0x36,0x66,0x66,0x7F,0x06,0x06,0x0F,0x00,0x00,0x00,0x00, // 4

0x00,0x00,0x7F,0x60,0x60,0x60,0x7E,0x03,0x03,0x63,0x73,0x3E,0x00,0x00,0x00,0x00, // 5

0x00,0x00,0x1C,0x30,0x60,0x60,0x7E,0x63,0x63,0x63,0x63,0x3E,0x00,0x00,0x00,0x00, // 6

0x00,0x00,0x7F,0x63,0x03,0x06,0x06,0x0C,0x0C,0x18,0x18,0x18,0x00,0x00,0x00,0x00, // 7

0x00,0x00,0x3E,0x63,0x63,0x63,0x3E,0x63,0x63,0x63,0x63,0x3E,0x00,0x00,0x00,0x00, // 8

0x00,0x00,0x3E,0x63,0x63,0x63,0x63,0x3F,0x03,0x03,0x06,0x3C,0x00,0x00,0x00,0x00, // 9

0x00,0x00,0x00,0x00,0x00,0x18,0x18,0x00,0x00,0x00,0x18,0x18,0x00,0x00,0x00,0x00, // :

0x00,0x00,0x00,0x00,0x00,0x18,0x18,0x00,0x00,0x00,0x18,0x18,0x18,0x30,0x00,0x00, // ;

0x00,0x00,0x00,0x06,0x0C,0x18,0x30,0x60,0x30,0x18,0x0C,0x06,0x00,0x00,0x00,0x00, // <

0x00,0x00,0x00,0x00,0x00,0x00,0x7E,0x00,0x00,0x7E,0x00,0x00,0x00,0x00,0x00,0x00, // =

0x00,0x00,0x00,0x60,0x30,0x18,0x0C,0x06,0x0C,0x18,0x30,0x60,0x00,0x00,0x00,0x00, // >

0x00,0x00,0x3E,0x63,0x63,0x06,0x0C,0x0C,0x0C,0x00,0x0C,0x0C,0x00,0x00,0x00,0x00, // ?

0x00,0x00,0x3E,0x63,0x63,0x6F,0x6B,0x6B,0x6E,0x60,0x60,0x3E,0x00,0x00,0x00,0x00, // @

0x40

0x00,0x00,0x08,0x1C,0x36,0x63,0x63,0x63,0x7F,0x63,0x63,0x63,0x00,0x00,0x00,0x00, // A

0x00,0x00,0x7E,0x33,0x33,0x33,0x3E,0x33,0x33,0x33,0x33,0x7E,0x00,0x00,0x00,0x00, // B

0x00,0x00,0x1E,0x33,0x61,0x60,0x60,0x60,0x60,0x61,0x33,0x1E,0x00,0x00,0x00,0x00, // C

0x00,0x00,0x7C,0x36,0x33,0x33,0x33,0x33,0x33,0x33,0x36,0x7C,0x00,0x00,0x00,0x00, // D

0x00,0x00,0x7F,0x33,0x31,0x34,0x3C,0x34,0x30,0x31,0x33,0x7F,0x00,0x00,0x00,0x00, // E

0x00,0x00,0x7F,0x33,0x31,0x34,0x3C,0x34,0x30,0x30,0x30,0x78,0x00,0x00,0x00,0x00, // F

0x00,0x00,0x1E,0x33,0x61,0x60,0x60,0x6F,0x63,0x63,0x37,0x1D,0x00,0x00,0x00,0x00, // G

0x00,0x00,0x63,0x63,0x63,0x63,0x7F,0x63,0x63,0x63,0x63,0x63,0x00,0x00,0x00,0x00, // H

0x00,0x00,0x3C,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x3C,0x00,0x00,0x00,0x00, // I

0x00,0x00,0x0F,0x06,0x06,0x06,0x06,0x06,0x06,0x66,0x66,0x3C,0x00,0x00,0x00,0x00, // J

0x00,0x00,0x73,0x33,0x36,0x36,0x3C,0x36,0x36,0x33,0x33,0x73,0x00,0x00,0x00,0x00, // K

0x00,0x00,0x78,0x30,0x30,0x30,0x30,0x30,0x30,0x31,0x33,0x7F,0x00,0x00,0x00,0x00, // L

0x00,0x00,0x63,0x77,0x7F,0x6B,0x63,0x63,0x63,0x63,0x63,0x63,0x00,0x00,0x00,0x00, // M

0x00,0x00,0x63,0x63,0x73,0x7B,0x7F,0x6F,0x67,0x63,0x63,0x63,0x00,0x00,0x00,0x00, // N

0x00,0x00,0x1C,0x36,0x63,0x63,0x63,0x63,0x63,0x63,0x36,0x1C,0x00,0x00,0x00,0x00, // O

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  0x00,0x00,0x7E,0x33,0x33,0x33,0x3E,0x30,0x30,0x30,0x30,0x78,0x00,0x00,0x00,0x00, // P
0x50
  0x00,0x00,0x3E,0x63,0x63,0x63,0x63,0x63,0x63,0x6B,0x6F,0x3E,0x06,0x07,0x00,0x00, // Q
  0x00,0x00,0x7E,0x33,0x33,0x33,0x3E,0x36,0x36,0x33,0x33,0x73,0x00,0x00,0x00,0x00, // R
  0x00,0x00,0x3E,0x63,0x63,0x30,0x1C,0x06,0x03,0x63,0x63,0x3E,0x00,0x00,0x00,0x00, // S
  0x00,0x00,0xFF,0xDB,0x99,0x18,0x18,0x18,0x18,0x18,0x18,0x3C,0x00,0x00,0x00,0x00, // T
  0x00,0x00,0x63,0x63,0x63,0x63,0x63,0x63,0x63,0x63,0x63,0x3E,0x00,0x00,0x00,0x00, // U
  0x00,0x00,0x63,0x63,0x63,0x63,0x63,0x63,0x63,0x63,0x36,0x1C,0x08,0x00,0x00,0x00, // V
  0x00,0x00,0x63,0x63,0x63,0x63,0x63,0x6B,0x6B,0x7F,0x36,0x36,0x00,0x00,0x00,0x00, // W
  0x00,0x00,0xC3,0xC3,0x66,0x3C,0x18,0x18,0x3C,0x66,0xC3,0xC3,0x00,0x00,0x00,0x00, // X
  0x00,0x00,0xC3,0xC3,0xC3,0x66,0x3C,0x18,0x18,0x18,0x18,0x3C,0x00,0x00,0x00,0x00, // Y
  0x00,0x00,0x7F,0x63,0x43,0x06,0x0C,0x18,0x30,0x61,0x63,0x7F,0x00,0x00,0x00,0x00, // Z
  0x00,0x00,0x3C,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x3C,0x00,0x00,0x00,0x00, // [
  0x00,0x00,0x80,0xC0,0xE0,0x70,0x38,0x1C,0x0E,0x07,0x03,0x01,0x00,0x00,0x00,0x00, // \
(back slash)
  0x00,0x00,0x3C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x3C,0x00,0x00,0x00,0x00, //
]
  0x08,0x1C,0x36,0x63,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // ^
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0x00,0x00,0x00, // _
  0x18,0x18,0x0C,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // `
0x60
  0x00,0x00,0x00,0x00,0x00,0x3C,0x46,0x06,0x3E,0x66,0x66,0x3B,0x00,0x00,0x00,0x00, // a
  0x00,0x00,0x70,0x30,0x30,0x3C,0x36,0x33,0x33,0x33,0x33,0x6E,0x00,0x00,0x00,0x00, // b
  0x00,0x00,0x00,0x00,0x00,0x3E,0x63,0x60,0x60,0x60,0x63,0x3E,0x00,0x00,0x00,0x00, // c
  0x00,0x00,0x0E,0x06,0x06,0x1E,0x36,0x66,0x66,0x66,0x66,0x3B,0x00,0x00,0x00,0x00, // d
  0x00,0x00,0x00,0x00,0x00,0x3E,0x63,0x63,0x7E,0x60,0x63,0x3E,0x00,0x00,0x00,0x00, // e
  0x00,0x00,0x1C,0x36,0x32,0x30,0x7C,0x30,0x30,0x30,0x30,0x78,0x00,0x00,0x00,0x00, // f
  0x00,0x00,0x00,0x00,0x00,0x3B,0x66,0x66,0x66,0x66,0x3E,0x06,0x66,0x3C,0x00,0x00, // g
  0x00,0x00,0x70,0x30,0x30,0x36,0x3B,0x33,0x33,0x33,0x33,0x73,0x00,0x00,0x00,0x00, // h
  0x00,0x00,0x0C,0x0C,0x00,0x1C,0x0C,0x0C,0x0C,0x0C,0x0C,0x1E,0x00,0x00,0x00,0x00, // i
  0x00,0x00,0x06,0x06,0x00,0x0E,0x06,0x06,0x06,0x06,0x06,0x66,0x66,0x3C,0x00,0x00, // j
  0x00,0x00,0x70,0x30,0x30,0x33,0x33,0x36,0x3C,0x36,0x33,0x73,0x00,0x00,0x00,0x00, // k
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  0x00,0x00,0x1C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x1E,0x00,0x00,0x00,0x00, //
l
  0x00,0x00,0x00,0x00,0x00,0x6E,0x7F,0x6B,0x6B,0x6B,0x6B,0x6B,0x00,0x00,0x00,0x00, // m
  0x00,0x00,0x00,0x00,0x00,0x6E,0x33,0x33,0x33,0x33,0x33,0x33,0x00,0x00,0x00,0x00, // n
  0x00,0x00,0x00,0x00,0x00,0x3E,0x63,0x63,0x63,0x63,0x63,0x3E,0x00,0x00,0x00,0x00, // o
  0x00,0x00,0x00,0x00,0x00,0x6E,0x33,0x33,0x33,0x33,0x3E,0x30,0x30,0x78,0x00,0x00, // p
0x70
  0x00,0x00,0x00,0x00,0x00,0x3B,0x66,0x66,0x66,0x66,0x3E,0x06,0x06,0x0F,0x00,0x00, // q
  0x00,0x00,0x00,0x00,0x00,0x6E,0x3B,0x33,0x30,0x30,0x30,0x78,0x00,0x00,0x00,0x00, // r
  0x00,0x00,0x00,0x00,0x00,0x3E,0x63,0x38,0x0E,0x03,0x63,0x3E,0x00,0x00,0x00,0x00, // s
  0x00,0x00,0x08,0x18,0x18,0x7E,0x18,0x18,0x18,0x18,0x1B,0x0E,0x00,0x00,0x00,0x00, // t
  0x00,0x00,0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x66,0x66,0x3B,0x00,0x00,0x00,0x00, // u
  0x00,0x00,0x00,0x00,0x00,0x63,0x63,0x36,0x36,0x1C,0x1C,0x08,0x00,0x00,0x00,0x00, // v
  0x00,0x00,0x00,0x00,0x00,0x63,0x63,0x63,0x6B,0x6B,0x7F,0x36,0x00,0x00,0x00,0x00, // w
  0x00,0x00,0x00,0x00,0x00,0x63,0x36,0x1C,0x1C,0x1C,0x36,0x63,0x00,0x00,0x00,0x00, // x
  0x00,0x00,0x00,0x00,0x00,0x63,0x63,0x63,0x63,0x63,0x3F,0x03,0x06,0x3C,0x00,0x00, // y
  0x00,0x00,0x00,0x00,0x00,0x7F,0x66,0x0C,0x18,0x30,0x63,0x7F,0x00,0x00,0x00,0x00, // z
  0x00,0x00,0x0E,0x18,0x18,0x18,0x70,0x18,0x18,0x18,0x18,0x0E,0x00,0x00,0x00,0x00, // {
  0x00,0x00,0x18,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x18,0x00,0x00,0x00, // |
  0x00,0x00,0x70,0x18,0x18,0x18,0x0E,0x18,0x18,0x18,0x18,0x70,0x00,0x00,0x00,0x00, // }
  0x00,0x00,0x3B,0x6E,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // ~
  0x00,0x70,0xD8,0xD8,0x70,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}; //
DEL
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Pcf8833.h

```
#ifndef pcf8833_h
#define pcf8833_h
#define NOP 0x00                    // nop
#define SWRESET 0x01                // software reset
#define BSTROFF 0x02                // booster voltage OFF
#define BSTRON 0x03                 // booster voltage ON
#define RDDIDIF 0x04                // read display identification
#define RDDST 0x09                  // read display status
#define SLEEPIN 0x10                // sleep in
#define SLEEPOUT 0x11               // sleep out
#define PTLON 0x12                  // partial display mode
#define NORON 0x13                  // display normal mode
#define INVOFF 0x20                 // inversion OFF
#define INVON 0x21                  // inversion ON
#define DALO 0x22                   // all pixel OFF
#define DAL 0x23                    // all pixel ON
#define SETCON 0x25                 // write contrast
#define DISPOFF 0x28                // display OFF
#define DISPON 0x29                 // display ON
#define CASET 0x2A                  // column address set
#define PASET 0x2B                  // page address set
#define RAMWR 0x2C                  // memory write
#define RGBSET 0x2D                 // colour set
#define PTLAR 0x30                  // partial area
#define VSCRDEF 0x33                // vertical scrolling definition
#define TEOFF 0x34                  // test mode
#define TEON 0x35                   // test mode
#define MADCTL 0x36                 // memory access control
#define SEP 0x37                    // vertical scrolling start address
#define IDMOFF 0x38                 // idle mode OFF
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define IDMON 0x39                  // idle mode ON
#define COLMOD 0x3A                 // interface pixel format
#define SETVOP 0xB0                 // set Vop
#define BRS 0xB4                    // bottom row swap
#define TRS 0xB6                    // top row swap
#define DISCTR 0xB9                 // display control
#define DOR 0xBA                    // data order
#define TCDFE 0xBD                  // enable/disable DF temperature compensation
#define TCVOPE 0xBF                 // enable/disable Vop temp comp
#define EC 0xC0                     // internal or external oscillator
#define SETMUL 0xC2                 // set multiplication factor
#define TCVOPAB 0xC3                // set TCVOP slopes A and B
#define TCVOPCD 0xC4                // set TCVOP slopes c and d
#define TCDF 0xC5                   // set divider frequency
#define DF8COLOR 0xC6               // set divider frequency 8-color mode
#define SETBS 0xC7                  // set bias system
#define RDTEMP 0xC8                 // temperature read back
#define NLI 0xC9                    // n-line inversion
#define RDID1 0xDA                  // read ID1
#define RDID2 0xDB                  // read ID2
#define RDID3 0xDC                  // read ID3

// backlight control
#define BKLGHT_LCD_ON 1
#define BKLGHT_LCD_OFF 2

// Booleans
#define NOFILL 0
#define FILL 1

// 12-bit color definitions
#define WHITE 0xFFF
```

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define BLACK 0x000
#define RED 0xF00
#define GREEN 0x0F0
#define BLUE 0x00F
#define CYAN 0x0FF
#define MAGENTA 0xF0F
#define YELLOW 0xFF0
#define BROWN 0xB22
#define ORANGE 0xFA0
#define PINK 0xF6A
// Font sizes
#define SMALL 0
#define MEDIUM 1
#define LARGE 2

// Define LCD-NOKIA6610 PinIO Interface Mask Bit
#define  LCD_BL_PIN   0x01010000   // P2.0  (0000 0000 0000 0000 0000 0000 0000 000x)
#define  LCD_CS_PIN   0x02020000   // P2.1  (0000 0000 0000 0000 0000 0000 0000 00x0)
#define  LCD_SCLK_PIN  0x04040000 // P2.2  (0000 0000 0000 0000 0000 0000 0000 0x00)
#define  LCD_SDATA_PIN  0x08080000  // P2.3  (0000 0000 0000 0000 0000 0000 0000 x000)
#define  LCD_RESET_PIN  0x10100010 // P2.4  (0000 0000 0000 0000 0000 0000 000x 0000)

#define  LCD_BL_DIR()         GP3DAT |= LCD_BL_PIN       // BL     = Output
#define  LCD_CS_DIR()         GP3DAT |= LCD_CS_PIN       // CS#    = Output
#define  LCD_SCLK_DIR()       GP3DAT |= LCD_SCLK_PIN     // SCLK   = Output
#define  LCD_SDATA_DIR()      GP3DAT |= LCD_SDATA_PIN    // SDATA  = Output
#define  LCD_RESET_DIR()      GP3DAT |= LCD_RESET_PIN    // RESET# = Output

#define  LCD_BL_HIGH()        GP3SET  = LCD_BL_PIN       // BL    = '1'
#define  LCD_BL_LOW()         GP3CLR  = LCD_BL_PIN       // BL    = '0'
#define  LCD_CS_HIGH()        GP3SET  = LCD_CS_PIN       // CS#   = '1'
#define  LCD_CS_LOW()         GP3CLR  = LCD_CS_PIN       // CS#   = '0'
```

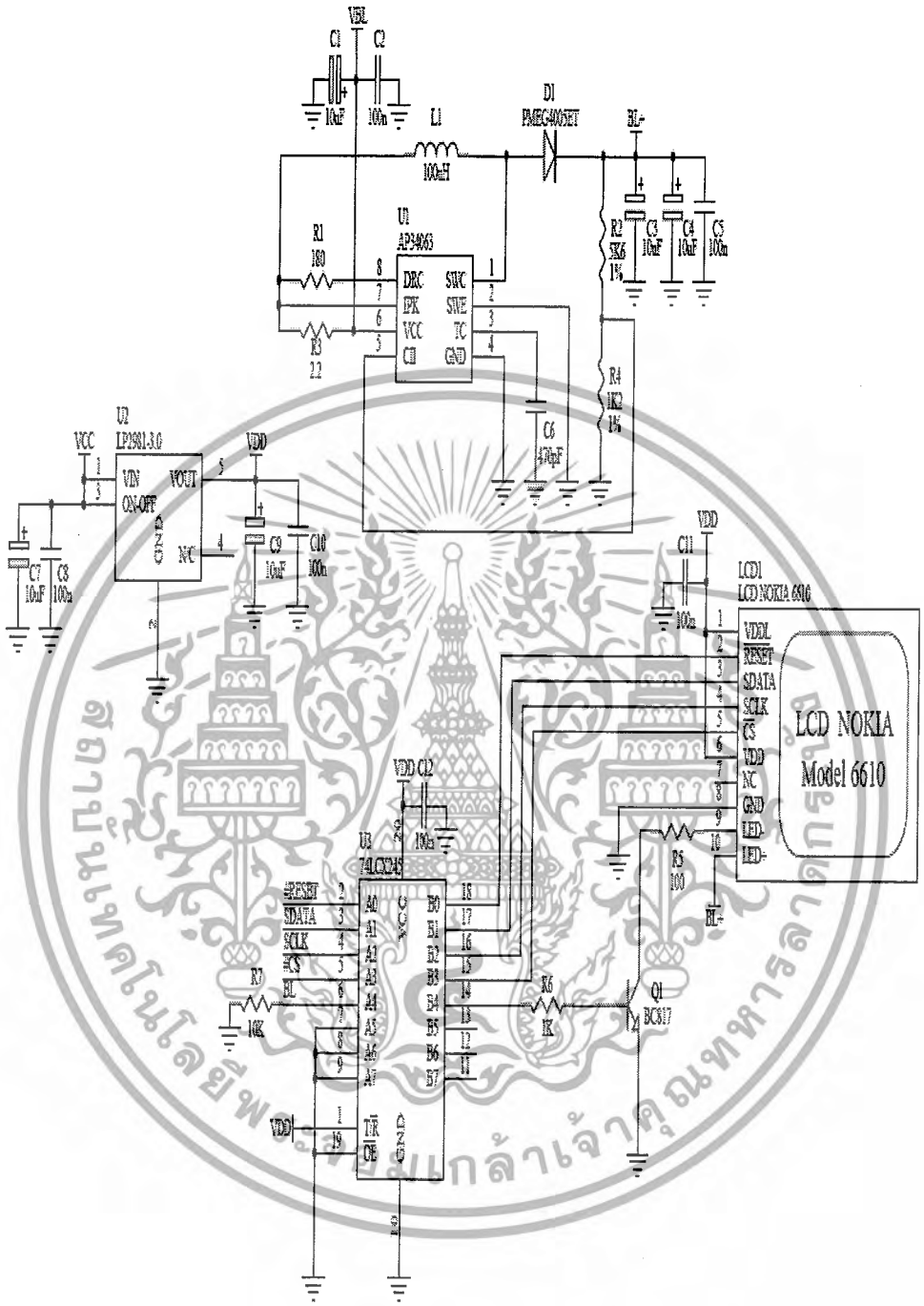
เอกสารนี้สงวนไว้สำหรับใช้ในการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define  LCD_SCLK_HIGH()    GP3SET = LCD_SCLK_PIN     // SCLK  = '1'
#define  LCD_SCLK_LOW()     GP3CLR = LCD_SCLK_PIN     // SCLK  = '0'
#define  LCD_SDATA_HIGH()   GP3SET = LCD_SDATA_PIN    // SDATA = '1'
#define  LCD_SDATA_LOW()    GP3CLR = LCD_SDATA_PIN    // SDATA = '0'
#define  LCD_RESET_HIGH()   GP3SET = LCD_RESET_PIN    // RESET# = '1'
#define  LCD_RESET_LOW()    GP3CLR = LCD_RESET_PIN    // RESET# = '0'
// End of Define For LCD-NOKIA6610

// mask definitions
#define BIT0  0x00000001
#define BIT1  0x00000002
#define BIT2  0x00000004
#define BIT3  0x00000008
#define BIT4  0x00000010
#define BIT5  0x00000020
#define BIT6  0x00000040
#define BIT7  0x00000080
#define BIT8  0x00000100
#define BIT9  0x00000200
#define BIT10 0x00000400
#define BIT11 0x00000800
#define BIT12 0x00001000
#define BIT13 0x00002000
#define BIT14 0x00004000
#define BIT15 0x00008000
#define BIT16 0x00010000
#define BIT17 0x00020000
#define BIT18 0x00040000
#define BIT19 0x00080000
#define BIT20 0x00100000
#define BIT21 0x00200000
#define BIT22 0x00400000
```

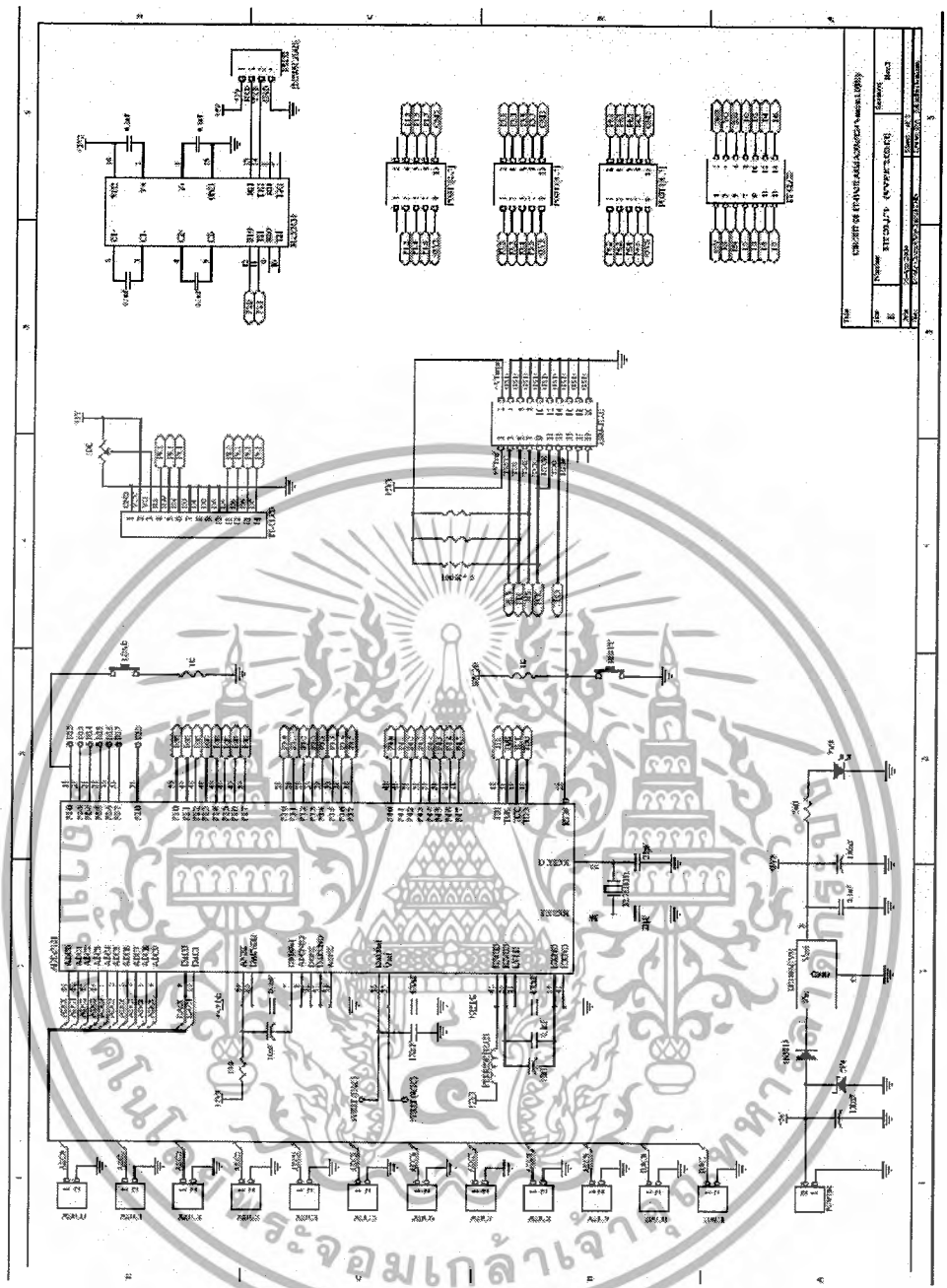
เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define BIT23 0x00800000
#define BIT24 0x01000000
#define BIT25 0x02000000
#define BIT26 0x04000000
#define BIT27 0x08000000
#define BIT28 0x10000000
#define BIT29 0x20000000
#define BIT30 0x40000000
#define BIT31 0x8000000

#endif // Lcd_h
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้