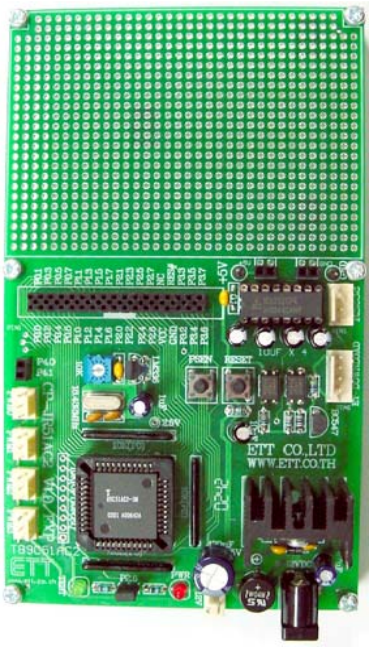
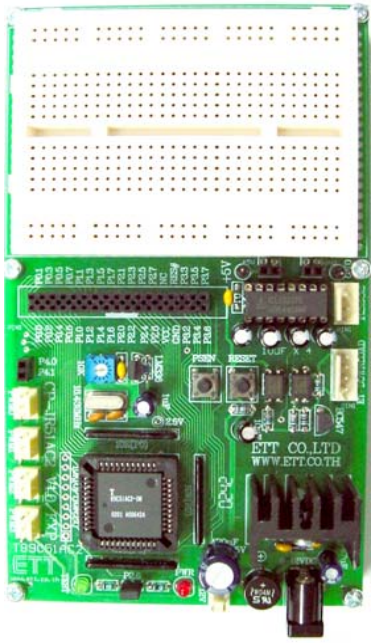
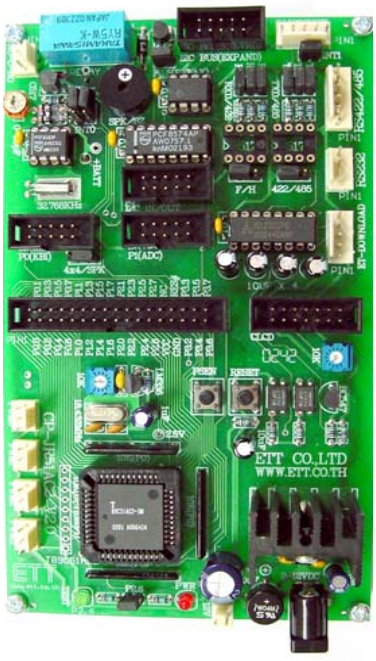
คู่มือการใช้งานบอร์ดไมโครคอนโทรลเลอร์

CP-JR51AC2 V1.0
CP-JR51AC2 V1.0 EXP
CP-JR51AC2 V2.0







ETT
www.ett.co.th

บริษัท อีทีที จำกัด

1112/96-98 ถนนสุขุมวิท แขวงพระโขนง เขตคลองเตย กรุงเทพ 10110 http://www.etteam.com

1112/96-98 Sukhumvit Rd., Phrakanong Klongtoey BANGKOK 10110 http://www.ett.co.th

TEL 02-712 1120 FAX 02-391 7216 e-mail:sale@etteam.com

ชื่อหนังสือ “คู่มือการใช้งานบอร์ดไมโครคอนโทรลเลอร์ CP-JR51AC2 V1&V2”

ISBN 974-90858-6-8

ผู้เขียน นายเอกชัย มะการ

พิมพ์ครั้งที่ 1

4 ธันวาคม 2545

จำนวน 72 หน้า

พิมพ์จำนวน 1000 เล่ม

(หากพบข้อผิดพลาดใดๆ ในหนังสือนี้ กรุณาแจ้งให้กับทาง บริษัท อีทีที จำกัด E-MAIL: sales@etteam.com)

จัดพิมพ์โดย
บริษัท อีทีที จำกัด
1112/96-98 ถนนสุขุมวิท แขวงพระโขนง
เขตคลองเตย กรุงเทพ 10110
โทร. (02)712-1120 - 1 FAX (02) 391-7216.

ETT
www.ett.co.th

คำนำ

บอร์ด CP-JR51AC2 นั้น ถูกพัฒนาขึ้น โดยได้แบ่งแยก รุ่นของบอร์ด ออกเป็น 3 รุ่น สำหรับตอบสนองต่อความต้องการของกลุ่มผู้ใช้ที่มีลักษณะของความต้องการที่แตกต่างกัน โดยทุกบอร์ดต่างก็อาศัยพื้นฐานทางโครงสร้างของวงจรที่เหมือนกัน แต่จะมีความแตกต่างกันในรายละเอียดและส่วนประกอบย่อยๆเพื่อให้คุณสมบัติของบอร์ดสามารถตอบสนองต่อความต้องการและตรงกับจุดประสงค์ของการนำไปใช้งานของผู้ใช้แต่ละกลุ่มมากที่สุด โดยลักษณะของบอร์ดนั้น จะออกแบบโครงสร้างของบอร์ดเพื่อ 3 จุดประสงค์หลัก คือ

- รุ่นแรก CP-JR51AC2 V1.0 EXPANSION สำหรับกลุ่มลูกค้าที่ต้องการนำบอร์ดไปใช้ศึกษาทดลองและเรียนรู้ทำความเข้าใจกับ CPU อย่างง่ายๆโดยมุ่งเน้นที่จะต่อทดลองวงจรส่วนของ I/O ต่างๆขึ้นมาทดลองเอง โดยใช้วิธีการต่อวงจรด้วยแผงทดลอง Photo Board เป็นหลัก
- รุ่นที่สอง คือ CP-JR51AC2 V1.0 นั้น จุดประสงค์ของการออกแบบบอร์ดเพื่อใช้ตอบสนองกลุ่มลูกค้าที่ต้องการนำบอร์ดไปใช้พัฒนางานต้นแบบ โดยมุ่งเน้นการออกแบบวงจรในส่วน I/O เอง ซึ่งภายในบอร์ดจะมีพื้นที่สำหรับต่อวงจร I/O ไว้สำหรับบัดกรีต่อวงจรเพิ่มเติม
- รุ่นที่สาม คือ CP-JR51AC2 V2.0 นั้นมุ่งเน้นสำหรับตอบสนองความต้องการของกลุ่มผู้ใช้ที่ต้องการสร้าง Application ของงาน โดยไม่สะดวกที่จะสร้างบอร์ดไมโครคอนโทรลเลอร์ขึ้นมาใช้งานเอง โดยบอร์ดจะเพิ่มเติมส่วนของวงจร I/O พื้นฐานต่างๆที่มีความจำเป็นสำหรับการนำไปประยุกต์ใช้งานในรูปแบบต่างๆไว้สำหรับตอบสนองความต้องการของผู้ใช้

สำหรับคู่มือเล่มนี้ เขียนขึ้นเพื่อจุดประสงค์สำหรับแนะนำการใช้งานบอร์ดเท่านั้น ไม่ได้มีการอธิบายถึงวิธีการเขียนโปรแกรมหรือรายละเอียดการใช้งาน CPU และ อุปกรณ์ Chips Support ต่างๆมากนัก โดยเนื้อหาได้มุ่งเน้นอธิบายถึงเฉพาะวิธีการ Setup Jumper และข้อกำหนดในการใช้งานส่วนประกอบต่างๆของบอร์ดเป็นหลัก แต่ก็ได้พยายามรวมรวมข้อมูลต่างๆที่เป็นประโยชน์ต่อการใช้งานไว้ให้ผู้ใช้พอสมควร ลักษณะของเนื้อหาเหมาะสำหรับกลุ่มผู้ใช้ที่มีพื้นฐานการใช้งานไมโครคอนโทรลเลอร์มาบ้างแล้ว และต้องการอาศัยบอร์ดเป็นเครื่องมือในการพัฒนาหรือทดลองศึกษาเรียนรู้เพื่อเพิ่มทักษะให้ตนเองมากขึ้น แต่อย่างไรก็ตามในการที่ผู้ใช้จะสามารถนำบอร์ดไปประยุกต์ใช้งานได้อย่างมีประสิทธิภาพนั้น อาจจำเป็นต้องศึกษารายละเอียดเพิ่มเติมอีกบางส่วน เช่น ถ้าต้องการใช้งาน RTC เบอร์ PCF8583 นั้น ถ้ายังไม่รู้จักวิธีการใช้งาน RTC ตัวนี้มาก่อนก็จำเป็นต้องศึกษารายละเอียดจากเอกสาร Data Sheet เพิ่มเติมด้วย ซึ่งทางทีมงาน ทีมงาน อีทีที ได้จัดทำตัวอย่างโปรแกรมต่างๆไว้ให้เป็นแนวทางสำหรับการใช้งานอุปกรณ์ต่างๆภายในบอร์ดอย่างครบถ้วนอยู่แล้ว โดยทางทีมงานหวังว่าคู่มือเล่มนี้คงสามารถเพิ่มความกระจ่างในการใช้งานบอร์ดได้พอสมควร

ทีมงานอีทีที

ธันวาคม 2545

สารบัญ

****

CP-JR51AC2

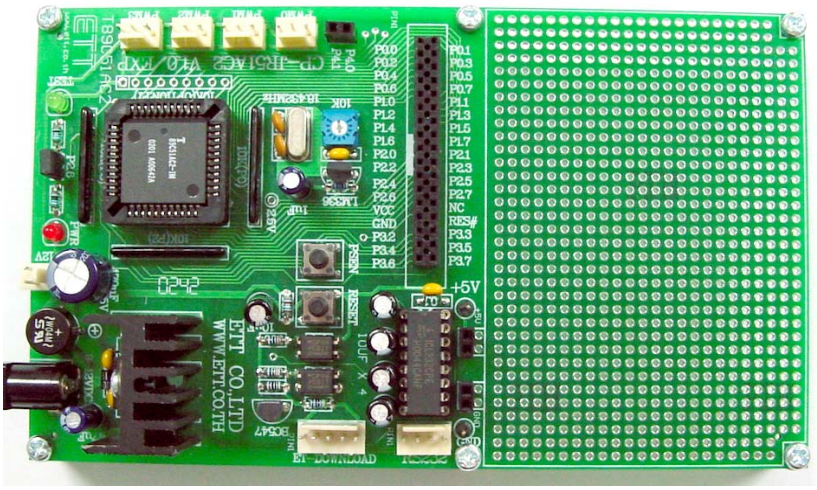
ลักษณะโดยทั่วไป

บอร์ดไมโครคอนโทรลเลอร์ในกลุ่ม CP-JR51AC2 เป็นบอร์ดไมโครคอนโทรลเลอร์ขนาดกลาง โดยเลือกใช้ CPU ไมโครคอนโทรลเลอร์ขนาด 8บิต ของ ATMEL ในตระกูล MCS51 เบอร์ T89C51AC2 เป็น CPU ประจำบอร์ด ซึ่ง CPU ตัวนี้บรรจุอยู่ภายในตัวถังแบบ PLCC ขนาด 44 ขา และมีทรัพยากรต่างๆบรรจุไว้ภายในตัว CPU อย่างครบถ้วน ไม่ว่าจะเป็น ADC/TIMER/COUNTER/PWM หรือ PORT I/O ต่างๆ ซึ่งมีความเหมาะสมในการนำไปประยุกต์ใช้งานในลักษณะต่างๆได้เป็นอย่างดี เนื่องจากสถาปัตยกรรมทางด้านฮาร์ดแวร์ของ CPU เบอร์นี้ จะมีความอ่อนตัวในการใช้งานได้ค่อนข้างดี กล่าวคือ ฟังก์ชั่นการทำงานต่างๆของฮาร์ดแวร์สามารถปรับเปลี่ยนการทำงานได้ด้วยโปรแกรม ดังนั้นผู้ใช้งานจึงสามารถนำระบบฮาร์ดแวร์แบบเดียวกันไปประยุกต์ใช้งานในลักษณะต่างๆกันได้โดยไม่ยากนัก โดยการปรับเปลี่ยนโปรแกรมสำหรับควบคุมการทำงานของบอร์ดเพียงเล็กน้อยเท่านั้น

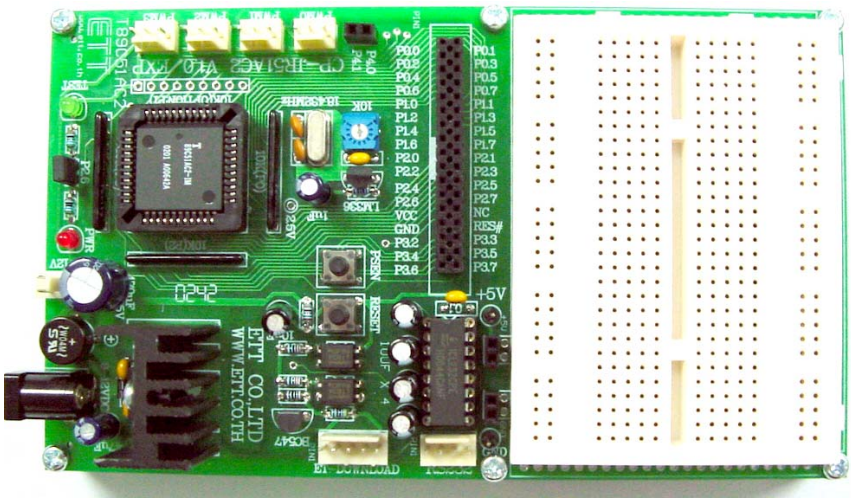
สำหรับอุปกรณ์ I/O ต่างๆ ซึ่งไม่ได้มีบรรจุไว้ในตัว CPU ด้วย ทางทีมงานอีทีที ก็ได้จัดหาและทำการออกแบบวงจรสำหรับเชื่อมต่อกับอุปกรณ์ต่างๆที่มีความจำเป็นไว้ให้ด้วยแล้ว ไม่ว่าจะเป็นจอแสดงผลแบบ LCD ระบบฐานเวลา RTC วงจร Line Driver สำหรับการสื่อสารข้อมูลอนุกรมแบบ RS232 และ RS422/485 และยังสามารถให้ผู้ใช้ทำการเพิ่มเติมอุปกรณ์ I/O อื่นๆเข้าไปได้อีกตามความจำเป็นในการใช้งาน

โดยลักษณะของบอร์ดไมโครคอนโทรลเลอร์ในกลุ่ม CP-JR51AC2 นี้ จะแบ่งออกเป็น 3 รุ่น ให้ผู้ใช้ได้เลือกใช้งานกันตามความเหมาะสมดังนี้คือ

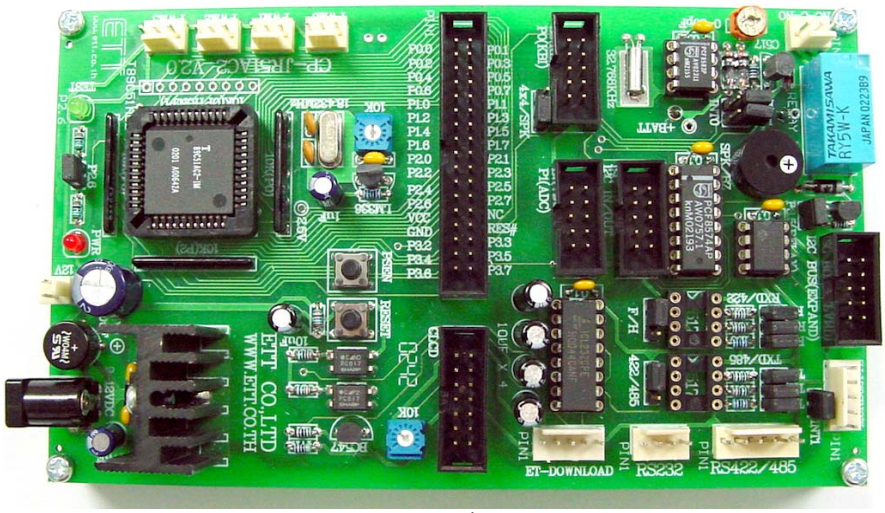
- CP-JR51AC2 V1.0 เป็นบอร์ดไมโครคอนโทรลเลอร์ ซึ่งออกแบบวงจรเฉพาะส่วนพื้นฐานที่จำเป็น เช่น แหล่งจ่ายไฟ วงจรรีเซ็ต วงจรกำเนิดความถี่สัญญาณนาฬิกา วงจรสำหรับ Download โปรแกรม และวงจรสื่อสารอนุกรม ส่วนวงจร I/O ภายนอกนั้น จะไม่ได้จัดเตรียมไว้ให้ด้วย แต่จะทำการต่อสัญญาณ I/O ต่างๆจาก CPU มาไว้ยังขั้วต่อ Connector สำหรับให้ผู้ใช้นำไปเชื่อมต่อกับอุปกรณ์ I/O ภายนอกได้โดยง่าย และยังมีพื้นที่เอนกประสงค์สำหรับให้ผู้ใช้ออกแบบวงจร I/O และต่อวงจร I/O เพิ่มเติมได้เอง เหมาะสำหรับผู้ใช้ที่ต้องการนำบอร์ดไปใช้พัฒนางานต้นแบบโดยการสร้าง I/O ต่างๆขึ้นมาใช้งานเอง
- CP-JR51AC2 V1.0 EXPANSION จะมีลักษณะเดียวกันกับบอร์ด CP-JR51AC2 V1.0 แต่จะมีแผง Photo Board สำหรับให้ผู้ใช้ต่อทดลองวงจร I/O อย่างง่ายๆได้เอง เหมาะสำหรับผู้ใช้ที่ต้องการศึกษาเรียนรู้และต้องการทดลองวงจร I/O ต่างๆ ร่วมกับ CPU อย่างง่ายๆ
- CP-JR51AC2 V2.0 เป็นบอร์ดไมโครคอนโทรลเลอร์ที่มีการออกแบบวงจรสำหรับเชื่อมต่อกับอุปกรณ์ I/O ภายนอกอื่นๆที่มีความจำเป็นไว้รองรับการใช้งานในลักษณะต่างๆ เพื่อให้ผู้ใช้งานสามารถนำบอร์ดไปใช้งานในลักษณะงานที่แตกต่างกันได้ โดยไม่ต้องดัดแปลงวงจร หรือ อาจดัดแปลงวงจรเพียงเล็กน้อยสำหรับงานบางอย่าง ซึ่งบอร์ดรุ่นนี้เหมาะสำหรับกลุ่มผู้ที่ต้องการนำบอร์ดไมโครคอนโทรลเลอร์ไปใช้งานจริงๆแต่ไม่สะดวกที่จะสร้างบอร์ดเอง



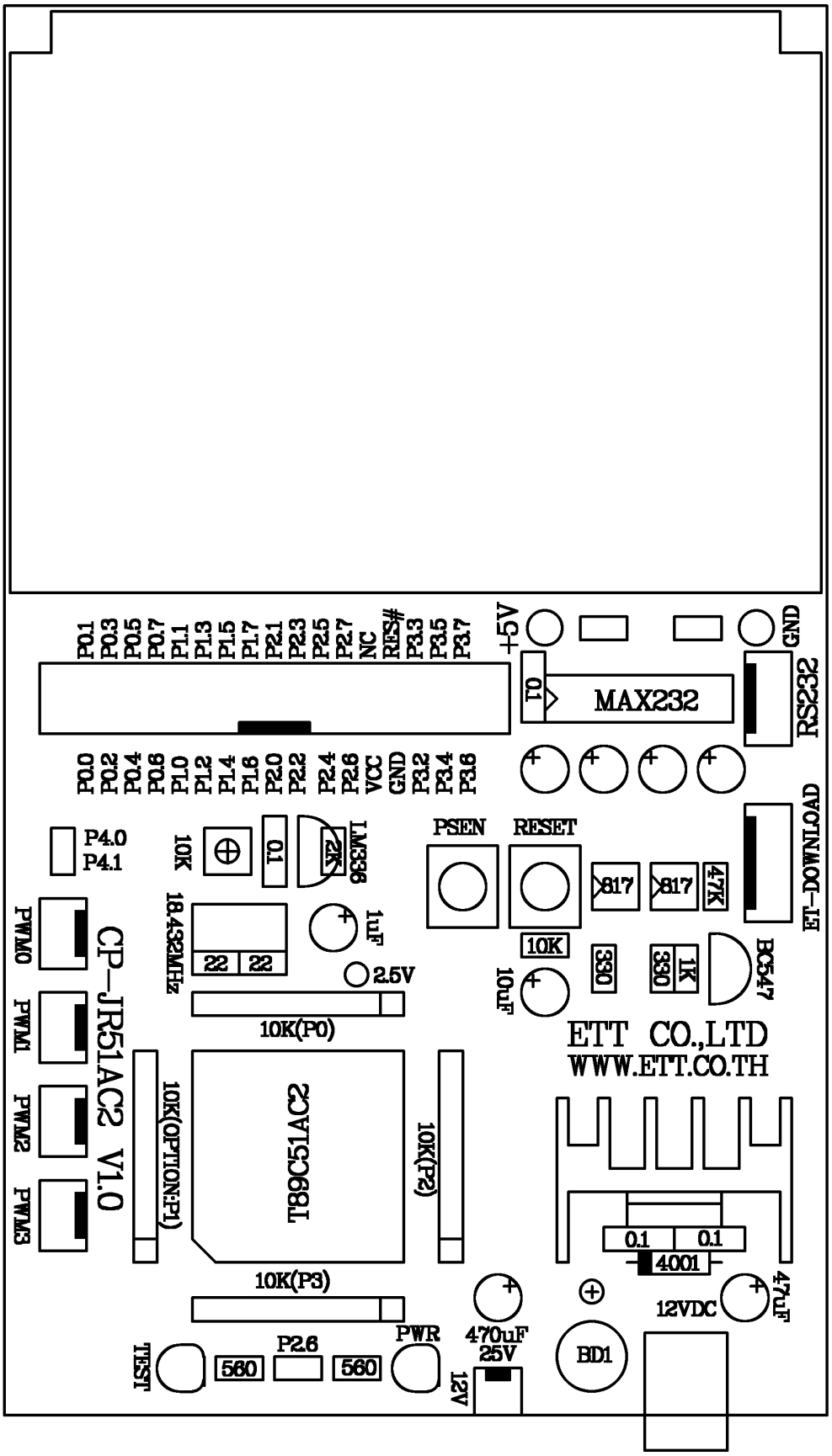
รูปแสดง ลักษณะของบอร์ด CP-JR51AC2 V1.0



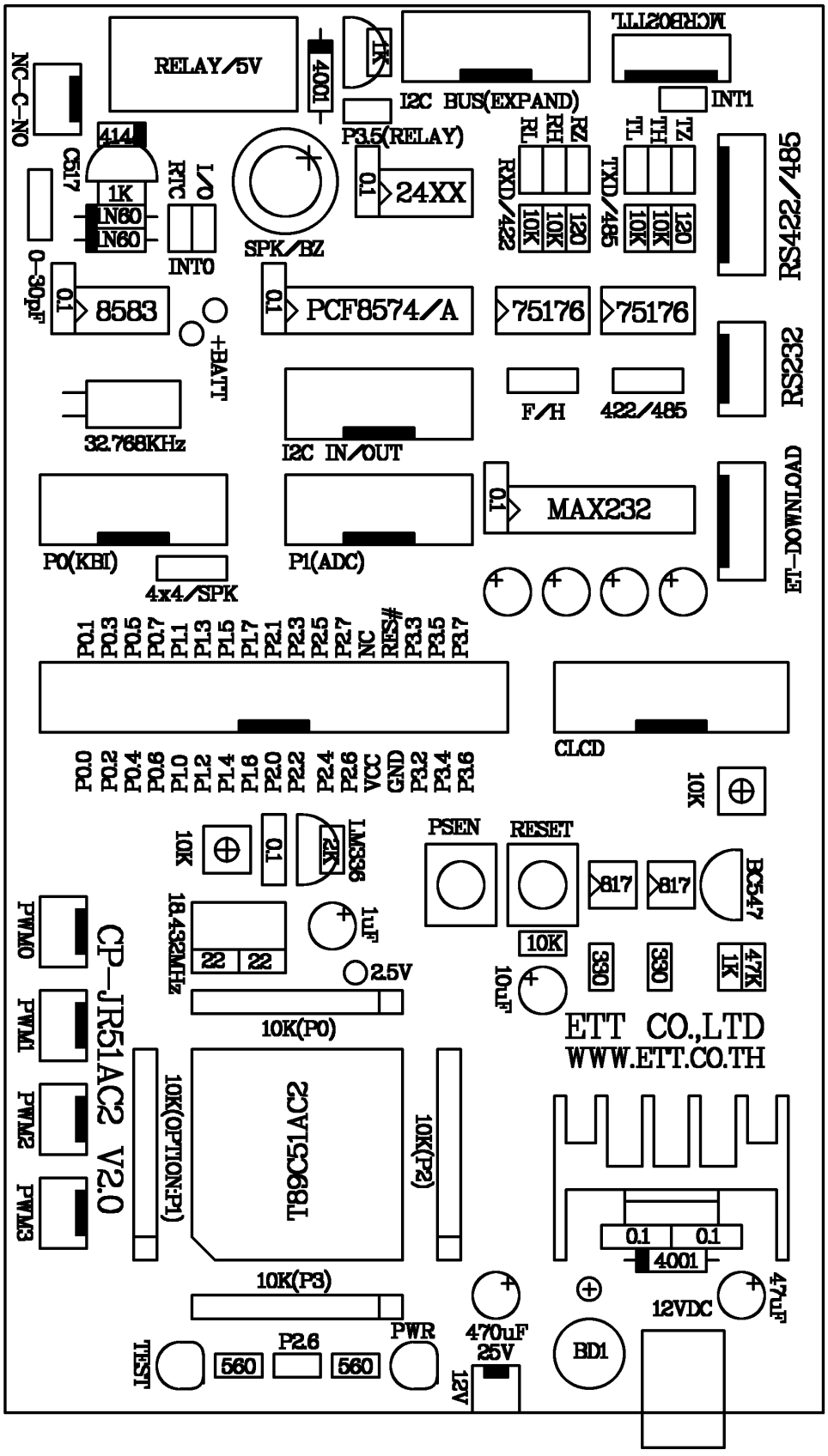
รูปแสดง ลักษณะของบอร์ด CP-JR51AC2 V1.0 EXPANSION



รูปแสดง ลักษณะของบอร์ด CP-JR51AC2 V2.0



รูปแสดง ลักษณะโครงสร้างของบอร์ด CP-JR51AC2 V1.0 และ V1.0 EXPANSION



รูปแสดง โครงสร้างของบอร์ด CP-JR51AC2 V2.0

แหล่งจ่ายไฟ (POWER SUPPLY)

สำหรับแหล่งจ่ายไฟของบอร์ดในกลุ่ม CP-JR51AC2 นั้น จะสามารถต่อใช้งานได้ทั้งกับไฟกระแสตรงและกระแสสลับ เนื่องจากในบอร์ดได้จัดเตรียมวงจร RECTIFIER แบบ BRIDGE พร้อมวงจร FILTER และ REGULATOR ขนาด +5V ไว้ให้อย่างครบถ้วนอยู่แล้ว โดยผู้ใช้สามารถป้อนแรงดันไฟตรงหรือไฟสลับที่มีระดับแรงดันประมาณ 9-12V ให้กับบอร์ด โดยสามารถเลือกต่อกับขั้ว CONNECTOR แบบ CPA ขนาด 2 ขา หรือจะต่อผ่านขั้ว CONNECTOR สำหรับ ADAPTER จ่ายไฟก็ได้เช่นกัน โดยการทำงานของแหล่งจ่ายไฟจะมีหลอดแสดงผล LED “PWR” สำหรับแสดงผลการทำงานให้ทราบด้วย

สัญญาณนาฬิกา CLOCK

ความถี่ของสัญญาณนาฬิกาที่จะป้อนให้กับ CPU เบอร์ T89C51AC2 นั้น ตามปรกติทั่วๆไปแล้ว จะสามารถป้อนค่าความถี่ของ Crystal ได้มากถึง 40MHz ในกรณีที่จะโปรแกรมโหมดการทำงานของ CPU ให้ทำงานใน Standard Mode (12 Clock / 1 Machine Cycle) แต่ในกรณีที่โปรแกรมโหมดการทำงานของ CPU ใน X2 Mode (6 Clock / 1 Machine Cycle) จะสามารถใช้ค่าความถี่สูงสุดได้ 20MHz ซึ่งเทียบเท่ากับความเร็ว 40 MHz ใน Standard Mode แต่สำหรับบอร์ด CP-JR51AC2 นั้นจะกำหนดให้ใช้ค่าความถี่ของ Crystal ที่ป้อนให้กับ CPU ด้วยค่าความถี่ Crystal เป็น 18.432MHz เพื่อให้การใช้งานพอร์ตสื่อสารอนุกรม สามารถหารค่า Baudrate ได้ลงตัวตามมาตรฐานของการสื่อสารอนุกรมทั่วๆไป ซึ่งค่าความเร็วการทำงานของ CPU ในบอร์ดจะอ้างอิงการทำงานจากความถี่ 18.432MHz นี้เป็นหลัก แต่อย่างไรก็ตามค่าความเร็วในการปฏิบัติงานของ CPU สามารถปรับเปลี่ยนได้จากโปรแกรมเพื่อให้การทำงานเร็วขึ้นเป็น 2 เท่า โดยกำหนดให้การทำงานของ CPU ทำงานใน X2 Mode ซึ่งจะเปรียบเทียบได้กับการทำงานด้วยความเร็วเท่ากับความถี่ 36.864MHz ใน Standard Mode โดยคุณสมบัติการทำงานของ สัญญาณนาฬิกามีดังนี้

- กำหนดให้ CPU ทำงานใน Standard Mode หรือ 12 Clock / 1 Machine Cycle ซึ่งคุณสมบัตินี้จะเหมือนกับ CPU ในตระกูล MCS51 มาตรฐานทั่วๆไป
- กำหนดให้ CPU ทำงานใน X2 Mode หรือ 6 Clock / 1 Machine Cycle ซึ่งจะทำให้การทำงานของ CPU เร็วกว่า CPU ในตระกูล MCS51 มาตรฐานทั่วๆไปถึง 2 เท่า เมื่อเปรียบเทียบโดยใช้ค่าความถี่ของ Crystal ด้วยค่าความถี่เดียวกัน

สำหรับการกำหนดโหมดการทำงานของสัญญาณนาฬิกาของ CPU นั้นสามารถกระทำได้ 2 วิธี ด้วยกัน คือ การกำหนดจากบิต X2 (บิต0) ของรีจิสเตอร์ CKCON ในคำสั่งของโปรแกรมที่ผู้ใช้เขียนขึ้น หรืออีกวิธีหนึ่งคือการกำหนดจาก Fuse Bit X2 จากขั้นตอนของการ Download โปรแกรมให้กับ CPU ใน Monitor Mode โดยสามารถสั่งเลือกเครื่องหมายถูก (√) หน้าบิต X2 ของโปรแกรม FLIP ซึ่งใช้สำหรับ Download โปรแกรมให้กับ CPU ใน Monitor Mode (ขอให้ดูรายละเอียดเพิ่มเติมเรื่องการใช้งานโปรแกรม FLIP) แล้วสั่ง Set Device Special Byte ซึ่งเมื่อสั่งเลือกบิต X2 ในโปรแกรม FLIP ไปแล้ว หลังจากรีเซ็ตทุกครั้งจะทำให้ CPU เริ่มต้นทำงานแบบ X2 Mode (6 Clock) แต่ถ้าไม่เลือกกำหนดบิต X2 ในโปรแกรม FLIP ไว้ หลังการรีเซ็ตทุกครั้งจะทำให้ CPU เริ่มต้นทำงานใน Standard Mode (12 Clock) แทน

แต่อย่างไรก็ตามถึงแม้ว่าจะมีการกำหนดบิต X2 ในส่วนของโปรแกรม FLIP ไว้อย่างไรก็ตาม ผู้ใช้สามารถสั่งเปลี่ยนแปลงการทำงานของ CPU ได้อีกหลังการรีเซ็ตแล้ว โดยเขียนโปรแกรมสั่งให้เซ็ตหรือเคลียร์บิต X2 (บิต0) ของรีจิสเตอร์ CKCON ได้อีกตามต้องการ แต่ถ้าในส่วนของโปรแกรมของผู้ใช้ไม่ได้ไปสั่งเปลี่ยนแปลงค่าของบิต X2 (บิต0) ในรีจิสเตอร์ CKCON เลย การทำงานของ CPU ก็จะขึ้นอยู่กับการเลือกบิต X2 จากโปรแกรม FLIP เพียงอย่างเดียวเท่านั้น

ดังนั้นเพื่อความแน่นอนและป้องกันความผิดพลาดในการกำหนดความเร็วการทำงานของ CPU นั้น ในส่วนเริ่มต้นโปรแกรมของผู้ใช้ควรสั่งจัดการกับบิต X2 ในรีจิสเตอร์ CKCON ด้วยเสมอ ทั้งนี้ก็เพื่อป้องกันความผิดพลาดจากขั้นตอนของการ Download โปรแกรมใน Monitor Mode โดยเพิ่มคำสั่งในการจัดการบิต X2 ของรีจิสเตอร์ CKCON ไว้ในส่วนเริ่มต้นของโปรแกรมด้วย ซึ่งมีวิธีการดังนี้

```
        ORG   0000H                 ; ตำแหน่งแอดเดรสเริ่มต้นของโปรแกรม
MAIN:   ORL   CKCON,#00000001B      ; สั่งเซ็ตค่า X2 บิตของ CKCON (8FH) ให้เป็น “1”
        คำสั่งอื่นๆของโปรแกรม
```

ตัวอย่างการกำหนดให้ CPU ทำงานแบบ X2 Mode (6 Clock / Machine Cycle)

```
        ORG   0000H                 ; ตำแหน่งแอดเดรสเริ่มต้นของโปรแกรม
MAIN:   ANL   CKCON,#11111110B      ; สั่งเคลียร์ค่า X2 บิตของ CKCON (8FH) ให้เป็น “0”
        คำสั่งอื่นๆของโปรแกรม
```

ตัวอย่างการกำหนดให้ CPU ทำงานแบบ Standard Mode (12 Clock / Machine Cycle)

การทำงานของรีจิสเตอร์ CKCON และผลต่อการทำงานของ Clock ของ CPU

CKCON เป็นรีจิสเตอร์สำหรับกำหนดความเร็วในการทำงานของ CPU และอุปกรณ์ต่างๆที่บรรจุไว้ภายในตัว CPU โดยหลังการรีเซ็ตทุกๆครั้งค่าของรีจิสเตอร์ตัวนี้จะมีค่าเป็น “0” ทุกบิต

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
-	WDX2	PCAX2	SIX2	T2X2	T1X2	T0X2	X2

รูปแสดง ลักษณะโครงสร้างของรีจิสเตอร์ CKCON (ตำแหน่ง 8FH)

- WDX2 เป็นบิต Watch Dog Clock ใช้สำหรับกำหนดคุณสมบัติโหมดการทำงานของสัญญาณนาฬิกาที่จะป้อนให้กับวงจร Watch Dog ซึ่งตามปรกติหลังการรีเซ็ตทุกๆครั้งนั้นบิตนี้จะมีค่าเป็น “0” เสมอ โดยบิตนี้จะมีผลเมื่อบิต X2 ถูกเซ็ตเป็น “1” ไว้แล้วเท่านั้น โดยถ้ากำหนดให้บิตนี้มีค่าเป็น “0” จะหมายถึง เลือกโหมดการทำงานของความถี่ของวงจร Watch Dog เป็นแบบ 6 Clock /

1 Watch Dog Clock แต่ถ้ากำหนดให้บิตนี้มีค่าเป็น “1” จะหมายถึง การเลือกโหมดการทำงานของความถี่ของจร Watch Dog เป็น 12 Clock / 1 Watch Dog Clock

- PCAX2 เป็นบิต Programmable Counter Array Clock ใช้สำหรับกำหนดคุณสมบัติโหมดการทำงานของสัญญาณนาฬิกาที่จะป้อนให้กับวงจร PCA ซึ่งตามปรกติหลังการรีเซ็ตทุกๆครั้งนั้นบิตนี้จะมีค่าเป็น “0” เสมอ โดยบิตนี้จะมีผลเมื่อบิต X2 ถูกเซ็ตเป็น “1” ไว้แล้วเท่านั้น โดยถ้ากำหนดให้บิตนี้มีค่าเป็น “0” จะหมายถึง เลือกโหมดการทำงานของความถี่ของวงจร PCA เป็นแบบ 6 Clock / 1 PCA Clock แต่ถ้ากำหนดให้บิตนี้มีค่าเป็น “1” จะหมายถึง การเลือกโหมดการทำงานของความถี่ของจร PCA เป็น 12 Clock / 1 PCA Clock
- SIX2 เป็นบิต Enhanced UART Clock (Mode 0 and 2) ใช้สำหรับกำหนดคุณสมบัติโหมดการทำงานของสัญญาณนาฬิกาที่จะป้อนให้กับวงจร UART หรือพอร์ตสื่อสารอนุกรม ซึ่งตามปรกติหลังการรีเซ็ตทุกๆครั้งนั้นบิตนี้จะมีค่าเป็น “0” เสมอ โดยบิตนี้จะมีผลเมื่อบิต X2 ถูกเซ็ตเป็น “1” ไว้แล้วเท่านั้น โดยถ้ากำหนดให้บิตนี้มีค่าเป็น “0” จะหมายถึง เลือกโหมดการทำงานของความถี่ของวงจร UART เป็นแบบ 6 Clock / 1 UART Clock แต่ถ้ากำหนดให้บิตนี้มีค่าเป็น “1” จะหมายถึง การเลือกโหมดการทำงานของความถี่ของจร UART เป็น 12 Clock / 1 UART Clock
- T2X2 เป็นบิต Timer2 Clock ใช้สำหรับกำหนดคุณสมบัติโหมดการทำงานของสัญญาณนาฬิกาที่จะป้อนให้กับวงจร Timer2 ซึ่งตามปรกติหลังการรีเซ็ตทุกๆครั้งนั้นบิตนี้จะมีค่าเป็น “0” เสมอ โดยบิตนี้จะมีผลเมื่อบิต X2 ถูกเซ็ตเป็น “1” ไว้แล้วเท่านั้น โดยถ้ากำหนดให้บิต T2X2 นี้ มีค่าเป็น “0” จะหมายถึง เลือกโหมดการทำงานของความถี่ของวงจร Timer2 เป็นแบบ 6 Clock แต่ถ้ากำหนดให้บิต T2X2 นี้มีค่าเป็น “1” จะหมายถึง การเลือกโหมดการทำงานของความถี่ของจร Timer2 เป็นแบบ 12 Clock
- T2X1 เป็นบิต Timer1 Clock ใช้สำหรับกำหนดคุณสมบัติโหมดการทำงานของสัญญาณนาฬิกาที่จะป้อนให้กับวงจร Timer1 ซึ่งตามปรกติหลังการรีเซ็ตทุกๆครั้งนั้นบิตนี้จะมีค่าเป็น “0” เสมอ โดยบิตนี้จะมีผลเมื่อบิต X2 ถูกเซ็ตเป็น “1” ไว้แล้วเท่านั้น โดยถ้ากำหนดให้บิต T1X2 นี้ มีค่าเป็น “0” จะหมายถึง เลือกโหมดการทำงานของความถี่ของวงจร Timer1 เป็นแบบ 6 Clock แต่ถ้ากำหนดให้บิต T1X2 นี้มีค่าเป็น “1” จะหมายถึง การเลือกโหมดการทำงานของความถี่ของจร Timer1 เป็นแบบ 12 Clock
- T0X2 เป็นบิต Timer0 Clock ใช้สำหรับกำหนดคุณสมบัติโหมดการทำงานของสัญญาณนาฬิกาที่จะป้อนให้กับวงจร Timer0 ซึ่งตามปรกติหลังการรีเซ็ตทุกๆครั้งนั้นบิตนี้จะมีค่าเป็น “0” เสมอ โดยบิตนี้จะมีผลเมื่อบิต X2 ถูกเซ็ตเป็น “1” ไว้แล้วเท่านั้น โดยถ้ากำหนดให้บิต T0X2 นี้ มีค่าเป็น “0” จะหมายถึง เลือกโหมดการทำงานของความถี่ของวงจร Timer0 เป็นแบบ 6 Clock แต่ถ้ากำหนดให้บิต T0X2 นี้มีค่าเป็น “1” จะหมายถึง การเลือกโหมดการทำงานของความถี่ของจร Timer0 เป็นแบบ 12 Clock
- X2 เป็นบิต CPU Clock ใช้สำหรับกำหนดโหมดการทำงานของสัญญาณนาฬิกาที่ป้อนให้กับ CPU ซึ่งตามปรกติหลังการรีเซ็ตทุกๆครั้งนั้นบิตนี้จะมีค่าเป็น “0” เสมอ โดยเมื่อกำหนดให้บิตนี้มีค่าเป็น “0” จะเป็นการกำหนดโหมดการทำงานของสัญญาณความถี่นาฬิกาของ CPU ให้ทำงาน

ใน Standard Mode (12 Clock / 1Machine Cycle) ซึ่งการทำงานของ CPU และอุปกรณ์ภายในตัว CPU ทั้งหมดจะอ้างอิงการทำงานจาก Machine Cycle แบบ 12 Clock ทั้งหมด แต่ถ้ากำหนดให้บิตนี้มีค่าเป็น “1” จะเป็นการกำหนดโหมดการทำงานของสัญญาณความถี่นาฬิกาของ CPU ให้ทำงานใน X2 Mode (6 Clock / 1Machine Cycle) ซึ่งการทำงานของ CPU จะอ้างอิงการทำงานจาก Machine Cycle แบบ 6 Clock ส่วนอุปกรณ์ภายในอื่นๆนั้นจะขึ้นอยู่กับการกำหนดบิตเลือกโหมดสัญญาณนาฬิกา ของแต่ละอุปกรณ์อีกครั้งหนึ่ง

หมายเหตุ ถ้ากำหนดให้บิต X2 มีค่าเป็น “0” ไว้ การเปลี่ยนแปลงค่าของบิต WDX2,PCAX2,SIX2,T2X2,T1X2 และ T0X2 จะไม่มีผลต่อการทำงานใดๆทั้งสิ้น

ความเกี่ยวข้องของ Hardware Security Byte กับ Clock และโหมดการทำงานของ CPU

Hardware Security Byte เป็นไบท์ข้อมูลพิเศษภายในตัว CPU เบอร์ T89C51AC2 มีขนาด 1 ไบท์ โดยไบท์ข้อมูลตำแหน่งนี้จะแบ่งออกเป็น 2ส่วน คือ 4บิตบน (MSB) สามารถอ่านหรือเขียนได้ทั้งจากคำสั่งของโปรแกรม และการเข้าถึงในโหมดของการโปรแกรม CPU ด้วยเครื่องโปรแกรม ส่วน 4บิตล่าง (LSB) นั้น ถ้าต้องการเข้าถึงจากโปรแกรมจะสามารถสั่งอ่านค่าออกมาได้อย่างเดียว ส่วนในการเขียนค่านั้นจะต้องกระทำในโหมดของการโปรแกรมข้อมูลให้กับ CPU ผ่านทางเครื่องโปรแกรม CPU ที่ใช้วิธีการแบบ Parallel Programming เท่านั้น

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
X2B	BLJB	-	-	-	LB2	LB1	LB0

รูปแสดงลักษณะโครงสร้างของ Hardware Security Byte

- **X2B หรือ บิต7** เป็นบิตสำหรับกำหนดโหมดการทำงานของ Oscillator ของ CPU หลังการรีเซ็ต โดยถ้าบิตนี้ถูกกำหนดให้เป็น “0”จะหมายถึงกำหนให้ CPU เริ่มต้นทำงานแบบ X2 Mode(6 Clock / Machine Cycle) แต่ถ้ากำหนดให้บิตนี้มีค่าเป็น “1” จะหมายถึงกำหนดให้ CPU เริ่มต้นทำงานแบบ Standard Mode (12 clock / Machine Cycle)
- **BLJB หรือ บิต6** เป็นบิต Boot Loader Jump Bit ใช้สำหรับกำหนดการทำงานของ CPU หลังการรีเซ็ต โดยถ้ากำหนดให้บิตนี้มีค่าเป็น “0” จะหมายถึง ให้ CPU เริ่มต้นทำงานในตำแหน่ง F800H ซึ่งเป็นตำแหน่งเริ่มต้นของ Monitor Mode แต่ถ้าบิตนี้ถูกกำหนดให้เป็น “1” จะหมายถึง ให้ CPU เริ่มต้นทำงานในตำแหน่งแอดเดรส 0000H ซึ่งเป็นตำแหน่งการทำงานของโปรแกรมที่ผู้ใช้เขียนขึ้นตามปรกติ

หมายเหตุ บิต X2B และ BLJB ของ Hardware Security Byte นี้สามารถเข้าถึงได้โดยโปรแกรม FLIP ในการติดต่อกับ CPU ใน Monitor Mode

ส่วนบิต X2 ของรีจิสเตอร์ CKCON นั้นจะสามารถเข้าถึงได้จากคำสั่งในโปรแกรมของผู้ใช้เอง โดยบิต X2 ของ CKCON และบิต X2B ของ Hardware Security Byte นี้จะแยกกันอยู่ ไม่ใช่บิตเดียวกัน

โหมดการทำงานของบอร์ด

การทำงานของบอร์ด CP-JR51AC2 นั้น สามารถกำหนดโหมดการทำงานของบอร์ดได้ 2 โหมดการทำงานด้วยกัน คือ Monitor Mode และโหมดการทำงานปรกติ User Mode (Run โปรแกรมในตำแหน่ง 0000H) โดยในการเลือกโหมดการทำงานของบอร์ดนั้นจะกระทำในขณะรีเซ็ต CPU โดยถ้าใช้วิธีการรีเซ็ต CPU ตามปรกติด้วยวิธีการจ่ายไฟเลี้ยงให้บอร์ดในครั้งแรก (Power-on Reset) หรือ ใช้วิธีการรีเซ็ตด้วยการกดสวิตช์รีเซ็ตเพียงอย่างเดียวนั้นบอร์ด CP-JR51AC2 จะเข้าทำงานใน User Mode หรือโหมดการทำงานปรกติ ตามโปรแกรมในตำแหน่ง 0000H แต่ถ้าใช้วิธีการรีเซ็ตโดยให้ขาสัญญาณ PSEN ของ CPU มีสภาวะทางลอจิกเป็น “0” ด้วย (มีการกดสวิตช์ PSEN ไว้ด้วย) บอร์ด CP-JR51AC2 จะเข้าทำงานใน Monitor Mode

โดยวิธีการตรวจสอบโหมดการทำงานของ CPU เบอร์ T89C51AC2 ของ ATMEL นั้น หลังจากขอบขาลง (Falling Edge) ของสัญญาณ RESET บิต ENBOOT ในรีจิสเตอร์ AUXR1 จะถูกกำหนดให้มีค่าเหมือนกับค่าของบิต BLJB (Boot Loader Jump Bit) หลังจากนั้น CPU จะตรวจสอบเงื่อนไขและสถานะทางฮาร์ดแวร์ของขาสัญญาณต่างๆในตัว CPU เพื่อตรวจสอบเงื่อนไขการทำงานของ CPU ว่าต้องทำงานใน User Mode (ตำแหน่ง 0000H) หรือ Monitor Mode (ตำแหน่ง F800H) ซึ่งเงื่อนไขการทำงานทางฮาร์ดแวร์มีดังนี้

- ขาสัญญาณ PSEN เป็น “0” หรือไม่
- ขาสัญญาณ EA เป็น “1” หรือไม่
- ขาสัญญาณ ALE เป็น “1” หรือปล่อยลอยหรือไม่

ซึ่งถ้าหากการตรวจสอบเงื่อนไขทางฮาร์ดแวร์ดังกล่าวข้างต้นเป็นจริงรีจิสเตอร์ FCON จะถูกกำหนดให้มีค่าเป็น 00H ส่วนค่าของรีจิสเตอร์ PC จะถูกกำหนดให้มีค่าเป็น F800H เพื่อชี้ไปยังตำแหน่งเริ่มต้นในการทำงานของโปรแกรมใน Monitor Mode ที่อยู่ในตัวของ CPU และเริ่มต้นทำงานตามคำสั่งของโปรแกรมที่อยู่ในตำแหน่ง F800H ทันที ซึ่งโปรแกรมในส่วนนี้จะถูกโปรแกรมมาพร้อมกับตัว CPU ทุกตัวจากโรงงานอยู่แล้ว โดยเมื่อ CPU ตรวจสอบเงื่อนไขการทำงานของฮาร์ดแวร์ข้างต้นแล้วพบว่าเงื่อนไขดังกล่าวเป็นจริง ก็จะเข้าทำงานใน Monitor Mode ในทันที โดยไม่สนใจเงื่อนไขของบิต BLJB ว่าเป็นอย่างไร

โดยวงจรของบอร์ด CP-JR51AC2 นั้นจะออกแบบให้ขาสัญญาณ EA ของ CPU มีค่าเป็น “1” ไว้ตลอด ส่วนขาสัญญาณ ALE ก็จะปล่อยลอยไว้ ส่วนขาสัญญาณ PSEN จะทำการ Pull-Up ให้มีค่าเป็น “1” ไว้โดยมีการต่อสวิตช์ PSEN เพื่อให้สามารถบังคับให้ขา PSEN เป็น “0” ได้ด้วยถ้ามีการกดสวิตช์ PSEN ไว้ ซึ่งจะเห็นได้ว่าสภาวะของขาสัญญาณต่างๆที่ใช้เลือกโหมดการทำงานของ CPU นั้นเตรียมพร้อมที่จะเข้าทำงานใน Monitor Mode อยู่แล้วขาดแต่เพียงขาสัญญาณ PSEN เพียงอย่างเดียว ดังนั้นการเลือกโหมดการทำงานของบอร์ดจึงขึ้นอยู่กับขาสัญญาณ PSEN โดยควบคุมจากสวิตช์ PSEN เท่านั้น ซึ่งถ้ามีการกดสวิตช์ PSEN รอไว้ก่อนแล้วจึง สั่งรีเซ็ต บอร์ดก็จะทำให้ CPU ตรวจสอบพบว่าขาสัญญาณ PSEN มีค่าเป็น “0” ซึ่งก็จะทำให้ CPU เข้าทำงานใน Monitor Mode ทันทีเนื่องจากขาสัญญาณ EA และ ALE นั้นตรงตามเงื่อนไขอยู่ก่อนแล้ว แต่ถ้าสั่งรีเซ็ต CPU โดยไม่มีการกดสวิตช์ PSEN ด้วย CPU ก็จะตรวจพบว่าขาสัญญาณ PSEN มีค่าเป็น “1” จึงทำให้เงื่อนไขทางฮาร์ดแวร์จะไม่ถูกต้องเนื่องจากขา PSEN จะมีค่าเป็น “1” ดังนั้น CPU จึงเข้าทำงานใน User Mode หรือโหมดการทำงานปรกติตามโปรแกรมในตำแหน่งแอดเดรส 0000H

แต่สำหรับในกรณีที่การตรวจสอบเงื่อนไขทางฮาร์ดแวร์แล้วเป็นเท็จ ค่าของ FCON จะถูกกำหนดให้มีค่าเป็น F0H หลังจากนั้นจึงจะมีการตรวจสอบค่าของบิต BLJB ว่าเป็นอย่างไร

- ถ้าบิต BLJB มีค่าเป็น "1" แล้ว CPU ก็จะกระโดดไปทำงานในตำแหน่ง 0000H ซึ่งเป็นตำแหน่งการทำงานตามปรกติของ CPU เหมือนกับ CPU ในตระกูล MCS51 ปรกติทั่วๆไป
- ถ้าบิต BLJB มีค่าเป็น "0" แล้ว CPU จึงจะตรวจสอบค่าของ BSB และ SBV เพื่อตัดสินใจกระโดดไปเริ่มต้นทำงานยังตำแหน่งแอดเดรสต่างๆดังตาราง

BLJB	BSB	SBV	**โหมดการทำงานหลังการทำงานแบบปรกติ**
( ) ไม่เลือก	XX	XX	User Mode (เริ่ม Run ตำแหน่ง 0000H)
(√) เลือก	00H	XX	User Mode (เริ่ม Run ตำแหน่ง 0000H)
(√) เลือก	ไม่ใช่ 00H	ไม่ใช่ FCH	User Mode (เริ่ม Run จากตำแหน่ง [SBV:00H])
(√) เลือก	ไม่ใช่ 00H	FCH	Monitor Mode (เริ่ม Run ตำแหน่ง F800H)

ตาราง แสดงโหมดการทำงานของ CPU หลังการรีเซ็ตแบบปรกติ

*****หมายเหตุ*****

- BLJB **หมายถึง** Boot Loader Jump Bit
- BSB **หมายถึง** Boot Status Byte
- SBV **หมายถึง** Software Boot Vector
- XX **หมายถึง ค่าใดๆ**

ซึ่งจะเห็นได้ว่าถ้ามีการเลือกบิต BLJB ไว้ แล้วค่าของ BSB ไม่ได้ถูกกำหนดให้มีค่าเป็น "00H" ไว้ด้วยแล้ว จะทำให้ CPU กระโดดไปทำงานที่ตำแหน่งอื่นๆที่ไม่ใช่ 0000H ซึ่งจะขึ้นอยู่กับค่าของ SBV โดยถ้าค่าของ SBV มีค่าเป็น FCH จะทำให้ CPU กลับเข้าไปทำงานใน Monitor Mode ที่ตำแหน่ง F800H แต่ถ้าค่าของ SBV เป็นค่าอื่นๆที่ไม่ใช่ FCH จะทำให้ CPU กระโดดไปทำงานยังตำแหน่งที่ชี้โดย SBV โดยค่าที่กำหนดให้ SBV จะเป็นค่าแอดเดรสไบต์สูงส่วนค่าแอดเดรสไบท์ต่ำจะมีค่าเป็น 00H เสมอ

ตัวอย่างเช่น ถ้ากำหนดให้ค่าของ SBV มีค่าเป็น 70H ไว้(มีการเลือกบิต BLJBไว้ และ BSB ไม่ใช่ 00H) หลังจากรีเซ็ตแบบปรกติทุกครั้งจะทำให้ CPU กระโดดไปทำงานยังตำแหน่ง 7000H เสมอ

แต่ถ้ากำหนดให้ค่าของ SBV มีค่าเป็น FCH ไว้ (มีการเลือกบิต BLJBไว้ และ BSB ไม่ใช่ 00H) หลังจากรีเซ็ตแบบปรกติทุกครั้งจะทำให้ CPU กลับเข้าไปทำงานใน Monitor Mode ที่ตำแหน่ง F800H แทน

*****หมายเหตุ***** หลังจากการรีเซ็ตแบบปรกตินั้น ถ้าต้องการให้ CPU เริ่มต้นทำงานที่ตำแหน่ง 0000H ทุกครั้ง เพื่อป้องกันความผิดพลาดต่างๆควรกระทำดังนี้

- อย่าเลือกบิต BLJB (อย่าใส่เครื่องหมายถูก (√) หน้าบิต BLJB ในโปรแกรม FLIP)
- กำหนดให้ค่าของ BSB(Boot Status Byte) มีค่าเป็น 00H ไว้เสมอ
- กำหนดให้ค่าของ SBV(Software Boot Vector) มีค่าเป็น 00H ไว้

การทำงานใน MONITOR MODE

ในโหมดนี้จะใช้สำหรับในกรณีที่ต้องการพัฒนาโปรแกรมของบอร์ด หรือการ Download โปรแกรมแบบ HEX File จากเครื่องคอมพิวเตอร์ PC ให้กับหน่วยความจำโปรแกรมของ CPU ซึ่งตามปรกติแล้วจะต้องใช้ร่วมกับโปรแกรม FLIP (Flexible In-system Programmer) ของ ATMEL โดยใช้การติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ PC กับ CPU เบอร์ T89C51AC2 ในบอร์ดด้วยพอร์ตสื่อสารอนุกรม RS232 ซึ่งในโหมดนี้ผู้ใช้สามารถสั่งจัดการหน่วยความจำ FLASH ภายในตัวของ CPU ได้โดยตรงไม่ว่าจะเป็นการสั่งลบข้อมูล หรือเขียนข้อมูลใหม่ให้กับหน่วยความจำ FLASH ของ CPU

ข้อจำกัดของการใช้งานบอร์ดใน MONITOR MODE

1. สัญญาณ ALE ของ CPU ต้องปล่อยลอยไว้ หรือกำหนดให้มีสภาวะเป็น “1”
2. สัญญาณ EA ของ CPU ต้องกำหนดให้มีสภาวะเป็น “1”
3. สัญญาณ PSEN ของ CPU ต้องมีสภาวะเป็นลอจิก “0” ก่อนปล่อยสัญญาณรีเซ็ตจาก “1” เป็น “0”

โดยบอร์ด CP-JR51AC2 นั้นจะออกแบบให้ใช้ Push Button Switch จำนวน 2 ตัว คือ PSEN และ RESET สำหรับร่วมกันกำหนดโหมดการทำงานของบอร์ด โดยวิธีการกำหนดโหมดการทำงานของ CPU เบอร์ T89C51AC2 ของบอร์ด CP-JR51AC2 ให้เข้าทำงานใน Monitor Mode นั้นสามารถทำได้ดังขั้นตอนต่อไปนี้

- กดสวิตช์แบบ Push Button ของ PSEN ค้างไว้
- กดสวิตช์แบบ Push Button ของ RESET ค้างไว้
- ปล่อยสวิตช์ RESET ในขณะที่สวิตช์ PSEN ยังถูกกดค้างอยู่
- ปล่อยสวิตช์ PSEN เป็นลำดับสุดท้าย

ซึ่งหลังการปฏิบัติตามขั้นตอนดังกล่าวข้างต้นเรียบร้อยแล้ว บอร์ด CP-JR51AC2 ก็พร้อมทำงานใน Monitor Mode แล้ว ซึ่งในจุดนี้ผู้ใช้ก็สามารถสั่งงาน CPU ใน Monitor Mode ด้วยฟังก์ชั่นต่างๆของโปรแกรม FLIP ได้ตามต้องการแล้ว สำหรับการใช้งานโปรแกรม FLIP นั้นจะกล่าวถึงโดยละเอียดในหัวข้อการพัฒนาโปรแกรมของบอร์ด

การทำงานใน USER MODE หรือ RUN MODE

การทำงานในโหมดนี้เป็นโหมดการทำงานปรกติของบอร์ด โดยจะใช้สำหรับในกรณีที่ผู้ใช้ทำการโปรแกรมข้อมูลให้กับหน่วยความจำ FLASH ของ CPU เรียบร้อยแล้ว ซึ่งในโหมดการทำงานนี้ สามารถจะใช้งานทรัพยากรต่างๆของ CPU ได้อย่างครบถ้วนโดยไม่มีข้อจำกัดใดๆ โดยวิธีการกำหนดโหมดการทำงานของบอร์ด CP-JR51AC2 เป็น RUN MODE หลังการรีเซ็ตบอร์ดทุกๆครั้ง ถ้าสวภาวะของขาสัญญาณ PSEN มีค่าเป็น “1” อยู่ CPU ก็จะกระโดดเข้ามาทำงานในโหมดนี้โดยอัตโนมัติอยู่แล้ว

ซึ่งหลังจาก CPU พ้นสภาวะจากการรีเซ็ตแล้วตรวจพบว่าขาสัญญาณ PSEN มีค่าเป็น “1” อยู่ CPU ก็จะเริ่มทำงานตามโปรแกรมที่บรรจุไว้ในหน่วยความจำของ CPU ในทันที

การจัดสรร I/O ของบอร์ด CP-JR51AC2 V2.0

บอร์ด CP-JR51AC2 V2.0 จะใช้ CPU เบอร์ T89C51AC2 เป็น CPU ประจำบอร์ด โดยตัว CPU เบอร์นี้จะมีขาสัญญาณที่สามารถนำมาใช้งานเป็น I/O Port ได้ทั้งหมด 34 เส้นสัญญาณ ประกอบด้วย

- P0[0..7] จำนวน 8 เส้นสัญญาณ
- P1[0..7] จำนวน 8 เส้นสัญญาณ
- P2[0..7] จำนวน 8 เส้นสัญญาณ
- P3[0..7] จำนวน 8 เส้นสัญญาณ
- P4[0..1] จำนวน 2 เส้นสัญญาณ

โดยการออกแบบวงจรของบอร์ด CP-JR51AC2 V2.0 นั้น ได้พยายามออกแบบวงจรโดยวางโครงสร้างของบอร์ด ให้มีความอ่อนตัวในในการใช้งานมากที่สุด เพื่อให้ผู้ใช้งานสามารถนำบอร์ดไปประยุกต์ใช้งานในหลายๆลักษณะได้โดยไม่ต้องดัดแปลงโครงสร้างวงจรของบอร์ดไปจากเดิมมากนัก ดังนั้นจึงได้มีการจัดสรรขาสัญญาณ Port I/O ของ CPU ให้สามารถทำงานได้หลายหน้าที่ โดยให้ผู้ใช้สามารถเลือกได้ตามต้องการ โดยบางขาสัญญาณสามารถกำหนดได้จากโปรแกรม แต่บางขาสัญญาณก็อาจต้องกำหนดจาก Jumper ด้วย โดยหน้าที่การใช้งาน Port I/O ของ CPU ในบอร์ด CP-JR51AC2 V2.0 นั้น สามารถสรุปได้ดังต่อไปนี้

P0.0-P0.7 สำหรับขาสัญญาณเหล่านี้สามารถใช้งานเป็น Input หรือ Output ได้ตามต้องการ โดยในบอร์ด CP-JR51AC2 V2.0 นั้น ขาสัญญาณของ P0 ทั้งหมด จะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN และขั้วต่อ P0(KBI) ไว้ด้วย โดยที่ P0.0-P0.6 จะต่อตรงไปยังขั้วต่อ P0(KBI) ทั้งหมด แต่ P0.7 นั้นจะต่อผ่าน Jumper 4X4/SPK โดยถ้าเลือก Jumper ไว้ด้าน 4x4 สัญญาณ P0.7 ก็จะต่อไปยังขั้ว P0(KBI) ด้วย แต่ถ้า Jumper 4x4/SPK ถูกเลือกไว้ทางด้าน SPK สัญญาณ P0.7 ก็จะถูกต่อไปควบคุมการทำงานของลำโพงแทน

P1.0-P1.7 สำหรับขาสัญญาณเหล่านี้จะสามารถใช้งานได้หลายหน้าที่ เช่น ใช้งานเป็น ADC ใช้งานเป็น Input หรือ Output และบางขายังสามารถใช้งานในหน้าที่พิเศษของระบบ Timer และ PCA ได้อีกด้วย โดยขาสัญญาณทั้งหมดจะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN และขั้วต่อ P1(ADC) ไว้ให้เลือกใช้งานตามต้องการ นอกจากนี้แล้วยังมีขาสัญญาณบางขาของ P1 ถูกจัดสรรไปยังวงจรอื่นๆด้วยดังนี้

- P1.3 ถูกต่อไปยังขั้วต่อ PWM0
- P1.4 ถูกต่อไปยังขั้วต่อ PWM1
- P1.5 ถูกต่อไปยังขั้วต่อ PWM2
- P1.6 ถูกต่อไปยังขั้วต่อ PWM3

P2.0-P2.7 สำหรับขาสัญญาณเหล่านี้จะสามารถใช้งานได้ทั้งเป็น Input และ Output โดยขาสัญญาณทั้งหมดของ P2 จะถูกต่อไปยังขั้วต่อ 34PIN ส่วน P2.0-P2.5 นั้นนอกจากจะต่อไปยังขั้วต่อ 34PIN แล้วยังต่อไปยังขั้วต่อ CLCD เพื่อใช้ควบคุมการทำงานของ LCD อีกด้วยดังนี้

- P2.0 ทำหน้าที่เป็น D4 ของ LCD
- P2.1 ทำหน้าที่เป็น D5 ของ LCD
- P2.2 ทำหน้าที่เป็น D6 ของ LCD
- P2.3 ทำหน้าที่เป็น D7 ของ LCD
- P2.4 ทำหน้าที่เป็น EN ของ LCD
- P2.5 ทำหน้าที่เป็น RS ของ LCD

P3.0 ทำหน้าที่เป็น RXD ของวงจรสื่อสารอนุกรม RS232/422/485

P3.1 ทำหน้าที่เป็น TXD ของวงจรสื่อสารอนุกรม RS232/422/485

P3.2-P3.7 สำหรับขาสัญญาณเหล่านี้จะสามารถใช้งานได้ทั้งเป็น Input และ Output โดยขาสัญญาณทั้งหมดของ P3 ทั้ง 6 เส้นนี้ จะถูกต่อไปยังขั้วต่อ 34PIN ทั้งหมด แต่ยังมีขาสัญญาณของ P3 อีกบางขาที่มีการจัดสรรหน้าที่ออกไปใช้งานยังส่วนอื่นๆด้วยดังนี้

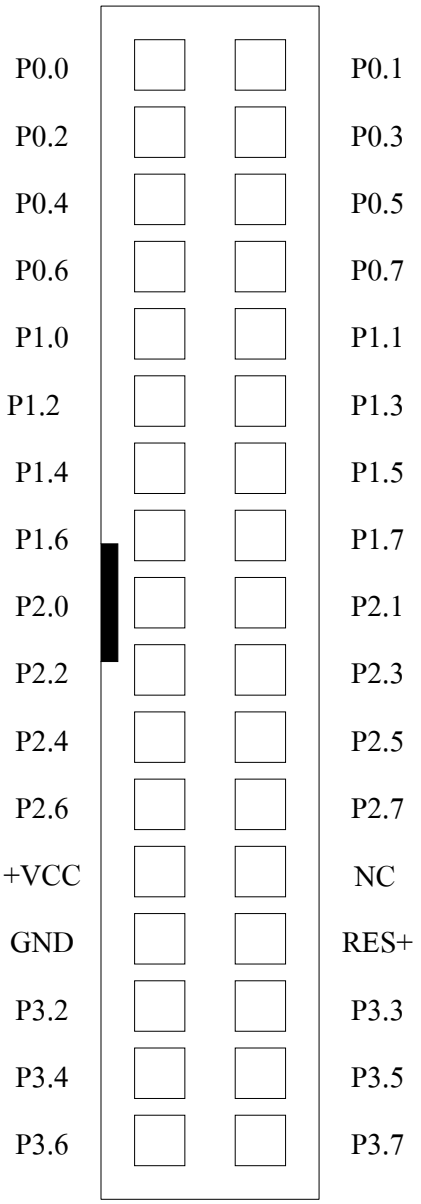
- P3.3 ถูกต่อไปยังขั้วต่อของเครื่องอ่านบัตรแถบแม่เหล็กโดยทำหน้าที่เป็น PRESENT
- P3.4 ถูกต่อไปควบคุมทิศทางการ รับ-ส่ง ข้อมูลของ RS485 โดยผ่าน Jumper 422/485
- P3.5 ถูกต่อไปควบคุมการทำงานของ RELAY โดยเลือกผ่าน Jumper P3.5(RELAY)
- P3.6 ถูกต่อไปยังขั้วต่อเครื่องอ่านบัตรแถบแม่เหล็กโดยทำหน้าที่เป็น DATA
- P3.6 ถูกต่อไปยังขั้วต่อเครื่องอ่านบัตรแถบแม่เหล็กโดยทำหน้าที่เป็น CLOCK

P4.0 จะใช้ทำหน้าที่ติดต่อกับอุปกรณ์ I2C BUS โดยทำหน้าที่เป็น SCL

P4.1 จะใช้ทำหน้าที่ติดต่อกับอุปกรณ์ I2C BUS โดยทำหน้าที่เป็น SDA

การใช้งานขั้วต่อ 34PIN (72IOZ80)

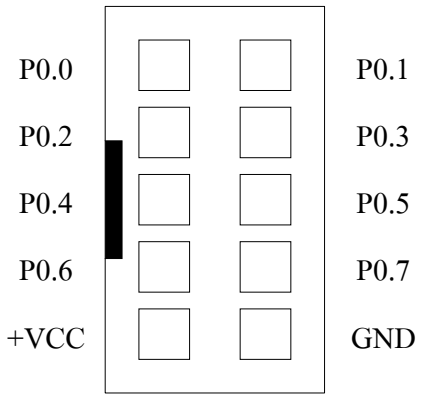
สำหรับขั้วต่อ Connector ขนาด 34 PIN ของบอร์ด CP-JR51AC2 นั้น จะมีอยู่ด้วยกัน 2 แบบ คือ ถ้าเป็นบอร์ด CP-JR51AC2 V1.0 และ CP-JR51AC2 V2.0 นั้น จะเป็นขั้วแบบ IDE ขนาด 34 PIN ตัวผู้ ซึ่งขั้วต่อนี้ออกแบบไว้สำหรับให้ผู้ใช้เชื่อมต่อสัญญาณต่างๆของ CPU ออกไปใช้งานกับบอร์ดอื่นๆ ซึ่งอาจเป็นบอร์ดที่ผู้ใช้ออกแบบและสร้างขึ้นเอง หรืออาจใช้บอร์ด I/O ต่างๆที่ ทาง บริษัท อีทีที จำกัด สร้างขึ้นไว้สนับสนุนการใช้งานก็ได้ โดยวิธีการเชื่อมต่อนั้นขอแนะนำให้ใช้สายแพร์ขนาด 34 PIN จะสะดวกที่สุดเพราะสามารถทำการเชื่อมต่อหรือแยกบอร์ดออกจากกันได้ง่าย ส่วนในกรณีที่ใช้บอร์ดรุ่น CP-JR51AC2 V1.0 EXPANSION นั้น ขั้วต่อ 34 PIN จะเป็นแบบ IDE ตัวเมีย เพื่อให้ผู้ใช้สามารถใช้สาย Jumper ต่อสัญญาณต่างๆจากขั้วต่อนี้ไปยังแผงทดลอง Photo Board เพื่อต่อร่วมกับวงจรต่างๆได้โดยง่าย โดยลักษณะการจัดเรียงสัญญาณเป็นดังนี้



รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้ว 34PIN

การใช้งานขั้วต่อ P0(KBI)

พอร์ต P0(KBI) ถูกจัดไว้ที่ขั้ว Connector ขนาด 10 PIN แบบ IDE ซึ่งขั้วต่อนี้จะมีอยู่เฉพาะ ในบอร์ดรุ่น CP-JR51AC2 V2.0 เท่านั้น โดยขั้วต่อนี้จะเชื่อมต่อสัญญาณมาจาก P0 ของ CPU ทั้ง 8 เส้น โดย P0.0-P0.6 จะถูกนำมาจัดเรียงไว้โดยตรงอยู่แล้ว ส่วน P0.7 จะต้องเลือกจาก Jumper (4x4 / SPK) อีกครั้งหนึ่ง โดยลักษณะของขาสัญญาณที่จัดเรียงไว้ที่ขั้ว P0(KBI) จะเป็นดังรูป



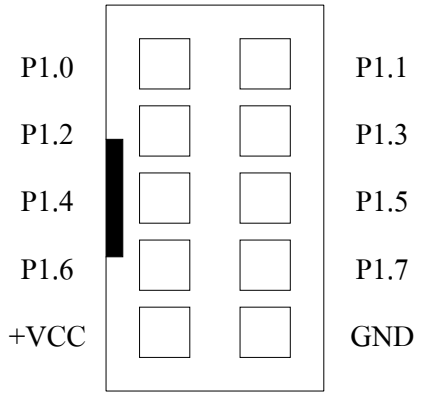
รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้ว P0(KBI)

โดยจุดประสงค์ในการออกแบบของบอร์ด CP-JR51AC2 V2.0 นั้น ขั้วต่อ P0(KBI) จะใช้สำหรับให้ผู้ใช้เชื่อมต่อกับวงจรคีย์บอร์ดแบบ Matrix ซึ่งสามารถจะใช้ได้กับคีย์บอร์ดแบบ Matrix ขนาด 4x3 หรือ 4x4 ก็ได้ ซึ่งในกรณีที่ใช้กับคีย์บอร์ดขนาด 4x3 จะเหลือสัญญาณไว้ 1 เส้นคือ P0.7 ซึ่งสามารถนำไปใช้ควบคุมการกำเนิดเสียงของลำโพงขนาดเล็กหรือ BUZZER เพื่อกำเนิดเสียงได้

สำหรับในกรณีที่ไม่มีความจำเป็นต้องใช้งานคีย์บอร์ดแล้ว ขั้วต่อ P0(KBI) นี้ก็ยังสามารถนำไปต่อใช้งานเป็น Input หรือ Output ทั่วๆไปได้อีกด้วย

การใช้งานขั้วต่อ P1(ADC)

พอร์ต P1(ADC) นี้จะถูกเชื่อมต่อออกมารอไว้ยังขั้ว Connector ขนาด 10 PIN แบบ IDE ซึ่งขั้วต่อนี้จะมีอยู่เฉพาะบอร์ดในรุ่น CP-JR51AC2 V2.0 เท่านั้น โดยขั้วต่อ P1(ADC) นี้ สามารถโปรแกรมหน้าที่การใช้งานได้ หลายหน้าที่ เช่น โปรแกรมให้ทำหน้าที่เป็น ADC ขนาด 10 บิต 8 ช่อง เพื่อรับค่าสัญญาณ Input แบบ Analog ขนาด 0-3VDC จากภายนอกเข้ามาและเปลี่ยนเป็นค่าข้อมูล (000H-3FFH) ให้กับ CPU นำไปประมวลผลตามต้องการ หรือถ้าไม่ต้องการใช้งานเป็น ADC พอร์ต P1(ADC) นี้ก็ยังสามารถโปรแกรมหน้าที่การทำงานสำหรับใช้งานเป็น Input / Output ทั่วไปได้อีกด้วย โดยลักษณะของขาสัญญาณที่จัดเรียงไว้ที่ขั้ว P1(ADC) จะเป็นดังรูป



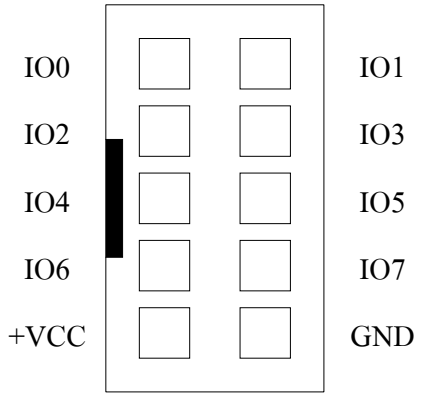
รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้ว P1(ADC)

นอกจากนี้แล้วสัญญาณ P1.3,P1.4,P1.5 และ P1.6 ของ พอร์ต P1 นั้นยังจัดสรรเป็นทางเลือกสำหรับนำไปใช้ประโยชน์อื่นๆได้อีกเช่น

- P1.3 ในกรณีที่ไม่ต้องการใช้งานเป็น ADC สามารถนำไปใช้งานเป็นขาสัญญาณ PWM0 เพื่อสร้างสัญญาณ PWM หรือการทำงานอื่นๆจาก PCA ช่อง0
- P1.4 ในกรณีที่ไม่ต้องการใช้งานเป็น ADC สามารถนำไปใช้งานเป็นขาสัญญาณ PWM1 เพื่อสร้างสัญญาณ PWM หรือการทำงานอื่นๆจาก PCA ช่อง1
- P1.5 ในกรณีที่ไม่ต้องการใช้งานเป็น ADC สามารถนำไปใช้งานเป็นขาสัญญาณ PWM2 เพื่อสร้างสัญญาณ PWM หรือการทำงานอื่นๆจาก PCA ช่อง2
- P1.6 ในกรณีที่ไม่ต้องการใช้งานเป็น ADC สามารถนำไปใช้งานเป็นขาสัญญาณ PWM3 เพื่อสร้างสัญญาณ PWM หรือการทำงานอื่นๆจาก PCA ช่อง3

การใช้งานขั้วต่อ I²C IN/OUT

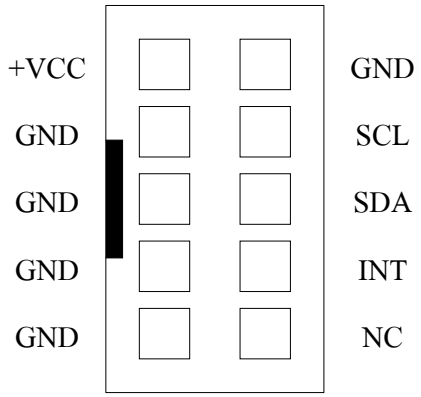
พอร์ต I²C IN/OUT นี้จะถูกเชื่อมต่อออกมารอไว้ยังขั้ว Connector ขนาด 10 PIN แบบ IDE ซึ่งขั้วต่อนี้จะมีอยู่เฉพาะบอร์ดในรุ่น CP-JR51AC2 V2.0 เท่านั้น โดยขั้วต่อ I²C IN/OUT นี้ จะเชื่อมต่อมาจากขาสัญญาณ I/O Port ของ PCF8574/A ซึ่งสามารถโปรแกรมหน้าที่การใช้งาน ให้เป็น Input หรือ Output ก็ได้ตามต้องการจากโปรแกรม แต่ต้องกำหนดหน้าที่ให้เป็น Input หรือ Output อย่างใดอย่างหนึ่งเท่านั้น ไม่สามารถใช้งานทั้งสองหน้าที่พร้อมๆกันได้ และในการกำหนดหน้าที่ให้เป็น Input หรือ Output ก็ต้องกำหนดให้เหมือนกันทั้ง 8 บิตด้วย โดยลักษณะของขาสัญญาณที่จัดเรียงไว้ที่ขั้ว I²C IN/OUT จะเป็นดังรูป



รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้ว I²C IN/OUT

การใช้งานขั้วต่อ I²C BUS EXPAND

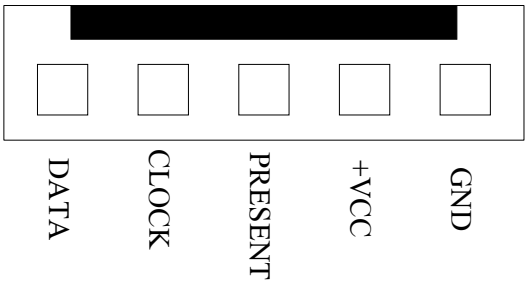
พอร์ต I²C BUS EXPAND นี้จะถูกเชื่อมต่อออกมารอไว้ยังขั้ว Connector ขนาด 10 PIN แบบ IDE ซึ่งขั้วต่อนี้จะมีอยู่เฉพาะบอร์ดในรุ่น CP-JR51AC2 V2.0 เท่านั้น โดยขั้วต่อ I²C BUS EXPAND นี้ จะใช้สำหรับทำการขยายหรือเพิ่มเติมจำนวนอุปกรณ์ที่ใช้การติดต่อสื่อสารแบบ I2C ให้กับบอร์ด



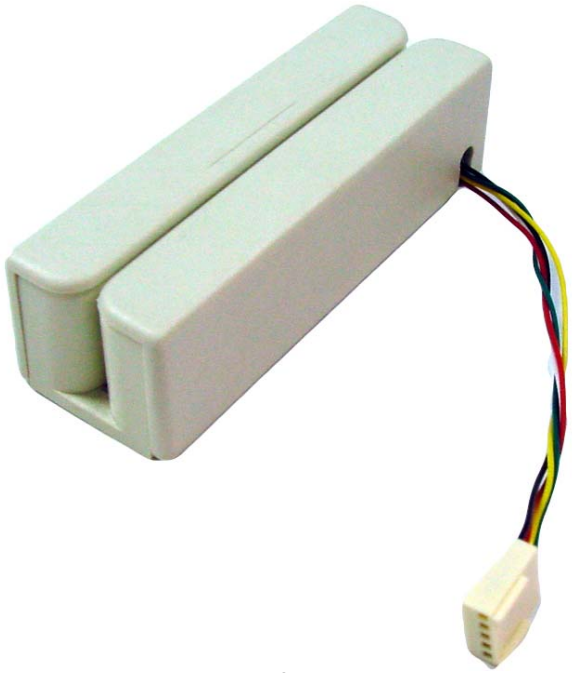
รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้ว I²C BUS EXPAND

การใช้งานเครื่องอ่านบัตรแถบแม่เหล็ก (MAGNETIC-CARD READER)

สำหรับบอร์ด CP-JR51AC2 V2.0 นั้น จะออกแบบให้สามารถเชื่อมต่อกับเครื่องอ่านบัตรแถบแม่เหล็ก รุ่น “MCR-B02TTL” ได้โดยตรง โดยไม่ต้องดัดแปลงวงจรใดๆทั้งสิ้น โดยในบอร์ดจะจัดเตรียมขั้วแบบ CPA ขนาด 5 PIN ไว้รองรับอยู่แล้ว ผู้ใช้สามารถนำขั้วต่อของเครื่องอ่านบัตรแถบแม่เหล็ก รุ่น “MCR-B02TTL” ของบริษัท อีทีที ต่อเข้าไปได้ทันที สำหรับในการเขียนโปรแกรมเพื่อเชื่อมต่อระหว่างบอร์ด CP-JR51AC2 V2.0 กับเครื่องอ่านบัตรแถบแม่เหล็กนั้น จะสามารถทำได้ 2 แบบ คือ ใช้วิธีการวนรอบตรวจสอบสัญญาณจากเครื่องอ่านบัตรแถบแม่เหล็กเอง ซึ่งวิธีการนี้จะใช้สัญญาณจาก CPU เพียง 2 เส้นสัญญาณคือ P3.6 ทำหน้าที่เป็น DATA และ P3.7 ทำหน้าที่เป็น CLOCK โดยต้องกำหนดคุณสมบัติของสัญญาณทั้ง 2 เส้นให้มีทิศทางเป็น Input ไว้ ส่วนอีกวิธีหนึ่งคือการใช้วิธีการ Interrupt ซึ่งสามารถทำได้โดยการ SHORT JUMPER “INT1” ที่อยู่ใกล้ๆกับขั้วต่อเครื่องอ่านบัตรแถบแม่เหล็ก ซึ่งการ SHORT JUMPER จะเป็นการเชื่อมต่อสัญญาณ INT1 ของ CPU เข้ากับสัญญาณ PRESENT ของเครื่องอ่านบัตรแถบแม่เหล็ก ดังนั้นเมื่อมีการนำบัตรแถบแม่เหล็กไปรูดผ่านเครื่องอ่านบัตรแถบแม่เหล็กก็จะมีการส่งสัญญาณ Interrupt ออกมายัง CPU ซึ่งผู้ใช้ก็เพียงแต่เขียนโปรแกรมสำหรับบริการการ Interrupt ของ INT1 ไว้ก็สามารถใช้งานได้แล้ว



รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้วต่อเครื่องอ่านบัตรแถบแม่เหล็ก MCR-B02TTL

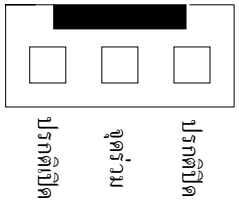


รูปแสดง ลักษณะของเครื่องอ่านบัตรแถบแม่เหล็ก

การใช้งาน OUTPUT RELAY

ภายในบอร์ด CP-JR51AC2 V2.0 นั้น จะออกแบบวงจรควบคุม RELAY ไว้ให้ผู้ใช้งานสามารถนำไปประยุกต์ใช้งานทั่วไปได้ด้วย จำนวน 1 ชุด โดยวงจร RELAY ดังกล่าวสามารถใช้งานได้ ทั้งหน้าสัมผัสแบบปรกติเปิด (Normal Open : NO) และแบบหน้าสัมผัสปรกติปิด (Normal Close : NC)

สำหรับสัญญาณ Output ในการควบคุมการทำงานของ REALY นั้น จะแบ่งมาจาก P3.5 ของ CPU ซึ่งเมื่อต้องการใช้งาน RELAY จะต้องทำการ SHORT JUMPER P3.5(RELAY) ไว้ด้วยเพื่อเชื่อมต่อสัญญาณ P3.5 มาทำการควบคุมการทำงานของ RELAY และต้องแน่ใจว่าไม่ได้ต่อสัญญาณ P3.5 จากขั้วต่ออื่นๆออกไปใช้งานกับอุปกรณ์ใดๆนอกเหนือจาก RELAY เนื่องจากสัญญาณ P3.5 นั้น นอกจากจะต่อมาใช้ควบคุมการทำงานของ RELAY แล้วยังต่อไปยังขั้วต่อ 34PIN อีกด้วย โดยการทำงานของ RELAY จะถูกควบคุมการทำงานจากขาสัญญาณ P3.5 โดยผู้ใช้ต้องกำหนดคุณสมบัติของสัญญาณ P3.5 ให้ทำหน้าที่เป็น OUTPUT ไว้ด้วย ซึ่งเมื่อขาสัญญาณ P3.5 มีสภาวะเป็น OUTPUT และให้สถานะเป็น "1" จะทำให้ RELAY ทำงาน แต่ถ้าสถานะของสัญญาณ P3.5 มีค่าเป็น "0" จะทำให้ RELAY หยุดทำงาน



รูปแสดง ลักษณะขั้วต่อสัญญาณจากหน้าสัมผัสของ RELAY

*****หมายเหตุ***** เนื่องจากสัญญาณ P3.5 ที่นำมาใช้ควบคุมการทำงานของ RELAY จะเป็นสัญญาณเส้นเดียวกับ P3.5 ที่ต่อไว้ยังขั้ว 34PIN ด้วย ดังนั้นเมื่อต้องการใช้งาน RELAY โดยการควบคุมจาก P3.5 แล้วต้องแน่ใจว่าไม่ได้ต่อใช้งานสัญญาณ P3.5 ในจุดอื่นๆดังกล่าว ไปใช้งานด้วย แต่ถ้าหากมีความจำเป็นต้องใช้งาน P3.5 พร้อมกับการใช้งาน RELAY ด้วยในเวลาเดียวกัน ก็อาจดัดแปลงวงจรได้โดยการ OPEN JUMPER P3.5(RELAY) ออก แล้วใช้วิธีการเชื่อมสายสัญญาณเส้นอื่นๆจาก PORT I/Oของ CPU ที่ไม่ได้ใช้งานมาเข้ากับวงจรควบคุม RELAY แทนก็ได้เช่นเดียวกัน โดยให้เชื่อมต่อสายสัญญาณที่ต้องการไปยัง Jumper P3.5(RELAY) ด้านที่ต่อกับตัวต้านทานค่า 1KOhm แต่การดัดแปลงวิธีนี้ควรถอดตัว JUMPER P3.5(RELAY) ออกจากบอร์ดเสียก่อน เพื่อจะได้ไม่หลงลืมทำการ SHORT JUMPER นี้ซ้ำอีกในภายหลัง เนื่องจากจะเป็นการ SHORT สัญญาณ P3.5 เข้ากับสัญญาณเส้นใหม่ที่บัดกรีมายังวงจร RELAY นี้อีก

การใช้งานลำโพงขนาดเล็ก หรือ BUZZER

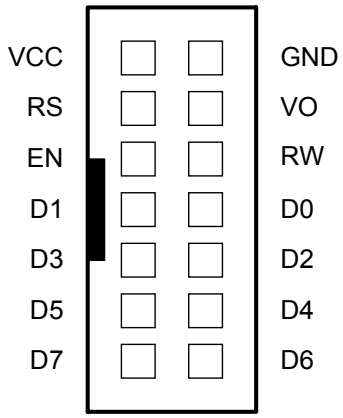
ภายในบอร์ด CP-JR51AC2 V2.0 จะมีวงจรกำเนิดเสียงรวมอยู่ด้วย 1 จุด ซึ่งในตำแหน่งนี้สามารถเลือกใส่อุปกรณ์กำเนิดเสียงแบบลำโพงขนาดเล็ก หรือ จะเลือกใส่ BUZZER แทนก็ได้เช่นเดียวกัน โดยในกรณีที่เลือกใช้ลำโพงจะมีข้อดีคือ สามารถสร้างความถี่เสียงได้หลากหลายความถี่ตามต้องการแต่การเขียนโปรแกรมจะยุ่งยากกว่า BUZZER เนื่องจากต้องสร้างเป็นสัญญาณความถี่จึงจะสามารถทำให้ลำโพงกำเนิดเสียงให้ได้ ส่วนในกรณีที่เลือกใช้ BUZZER นั้น จะมีข้อดีคือ เขียนโปรแกรมควบคุมการกำเนิดเสียงได้ง่ายกว่าลำโพง เนื่องจากใช้วิธีการ ON หรือ OFF เท่านั้น โดยการ ON บิตควบคุม BUZZER ให้มีสภาวะเป็น “1” เท่านั้น BUZZER ก็จะกำเนิดเสียงให้แล้ว แต่ความถี่เสียงของ BUZZER จะไม่สามรถเลือกได้ เหมือนกับลำโพง

สำหรับสัญญาณ Output ในการควบคุมการทำงานของ ลำโพง หรือ BUZZER นั้น จะแบ่งมาจาก P0.7 ของ CPU ซึ่งเมื่อต้องการใช้งาน ลำโพง หรือ BUZZER จะต้องทำการเลือก JUMPER 4x4/SPK มารอไว้ยังตำแหน่ง SPK ด้วย เพื่อให้สามารนำสัญญาณ P0.7 มาทำการควบคุมการทำงานของลำโพง หรือ BUZZER ได้ โดยผู้ใช้ต้องทำการกำหนดคุณสมบัติของสัญญาณ P0.7 ให้ทำหน้าที่เป็น OUTPUT ไว้ด้วย ซึ่งเมื่อขาสัญญาณ P0.7 มีสภาวะเป็น OUTPUT และให้สถานะเป็น “1” จะทำให้ ลำโพง หรือ BUZZER ทำงาน แต่ถ้าสถานะของสัญญาณ P0.7 มีค่าเป็น “0” จะทำให้ ลำโพง หรือ BUZZER หยุดทำงาน

*****หมายเหตุ***** เนื่องจากสัญญาณ P0.7 ที่นำมาใช้ควบคุมการทำงานของ ลำโพง หรือ BUZZER นั้น จะเป็นสัญญาณเส้นเดียวกับ P0.7 ที่ต่อไว้ยังขั้ว 34PIN และขั้วต่อ 10PIN แบบ IDE ของพอร์ต P0(KBI) ด้วย ดังนั้นเมื่อต้องการใช้งาน ลำโพง หรือ BUZZER โดยการควบคุมจาก P0.7 แล้ว ต้องแน่ใจว่าไม่ได้ต่อใช้งานสัญญาณ P0.7 ในจุดอื่นๆทั้งสอง ดังกล่าวด้วย แต่ถ้าหากมีความจำเป็นต้องใช้งาน P0.7 พร้อมกับการใช้งานลำโพง หรือ BUZZER ด้วยในเวลาเดียวกัน ก็อาจดัดแปลงวงจรได้ โดยการใช้วิธีการบัดกรีเชื่อมสายสัญญาณเส้นอื่นๆจาก PORT I/Oของ CPU ที่ไม่ได้ใช้งานมาเข้ากับวงจรควบคุมลำโพงหรือ BUZZER แทนก็ได้เช่นเดียวกัน โดยให้เชื่อมต่อสายสัญญาณที่ต้องการไปยัง Jumper 4x4/SPK ด้าน SPK แต่การดัดแปลงวิธีนี้ควรถอดตัว JUMPER 4x4/SPK ออกจากบอร์ดเสียก่อนแล้วทำการเชื่อมต่อขาสัญญาณ Jumper 4x4/SPK ด้าน ตำแหน่ง 4x4 ไว้ด้วยกันเลย เพื่อจะได้ไม่หลงลืมทำการเลือก JUMPER กลับมายังด้าน SPK นี้อีกในภายหลัง เนื่องจากจะเป็นการ SHORT สัญญาณ P0.7 เข้ากับสัญญาณเส้นใหม่ที่บัดกรีมายังวงจรควบคุมลำโพงหรือ BUZZER นี้อีก

การใช้งานจอแสดงผลแบบ LCD (Dot-Matrix Character LCD)

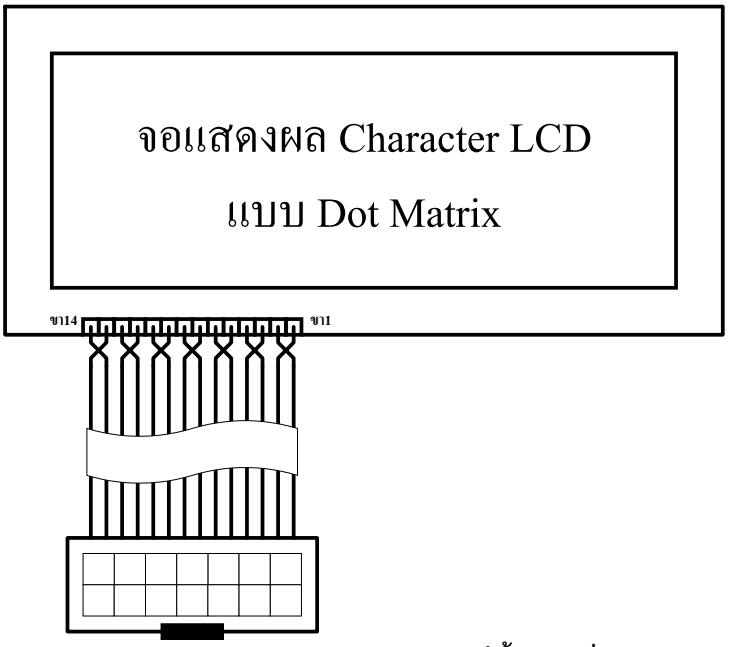
บอร์ด CP-JR51AC2 V2.0 สามารถใช้เชื่อมต่อกับจอแสดงผล LCD แบบ Dot-Matrix โดยเชื่อมต่อผ่านทาง Connector ขนาด 14 PIN และใช้ตัวต้านทานปรับค่าได้แบบเกือกม้าขนาด 10K สำหรับปรับระดับความสว่างของหน้าจอ LCD โดยวงจรในการเชื่อมต่อ LCD ของบอร์ดนี้จะออกแบบวงจรให้ใช้วิธีการควบคุมการทำงานแบบ "DATA 4-BIT"



รูปแสดง ขาสัญญาณของขั้วต่อ CLCD

สำหรับวิธีการเชื่อมต่อสัญญาณจากขั้วต่อ LCD ของบอร์ดไปเข้ากับตัว LCD นั้น เพื่อความสะดวกขอแนะนำให้ใช้สายแพร์ ขนาด 14 เส้น เป็นตัวเชื่อมต่อสัญญาณระหว่างบอร์ดและตัว LCD จะสะดวกมากที่สุด ซึ่งในปัจจุบันพบว่า ลักษณะขั้วสัญญาณที่อยู่ทางด้านของจอแสดงผล LCD เองนั้น ที่พบเห็นได้ทั่วๆไป จะมีอยู่ด้วยกัน 2 แบบ คือ

- แบบที่เป็นขั้วต่อแบบแถวเดี่ยว ขนาด 14PIN (HEADER 14X1) โดยการต่อสายของ LCD แบบนี้จะใช้สายแพร์ขนาด 14 เส้น แบบเข้าหัว CONNECTOR ไว้ด้านเดียว สำหรับเสียบกับขั้วต่อ CLCD ภายในบอร์ด CP-JR51AC2 V2.0 ส่วนปลายสายอีกด้านหนึ่งของสายแพร์ทั้ง 14 เส้น จะต้องนำไปบัดกรีเข้ากับขั้วต่อของตัว LCD ให้ครบทั้ง 14 เส้น โดยในการบัดกรีจะต้องสลับปลายสายเป็นคู่ๆเรียงลำดับกันไป คือ ขา 2 สลับกับ 1,ขา4 สลับกับ 3...ขา14 สลับกับ 13 ตามลำดับ กล่าวคือ สายเส้นที่1 ต่อกับ PIN2 ของ LCD ส่วนสายเส้นที่2 จะต้องต่อกับ PIN1 ของ LCD และในทำนองเดียวกันสายเส้นที่3 ก็จะต้องต่อกับ PIN4 ของ LCD อย่างนี้เรื่อยไปจนครบทั้ง 14 เส้น
- แบบที่เป็นขั้วต่อแบบแถวคู่ 14PIN (HEADER ขนาด 7X2) โดยการต่อสายของ LCD แบบนี้ จะใช้สายแพร์ขนาด 14 เส้น แบบเข้าหัว CONNECTOR ไว้ทั้งสองด้าน โดยในการเชื่อมต่อนั้น ให้ต่อสายแต่ละด้านเข้ากับขั้วต่อ โดยให้ตำแหน่งของ PIN1 ของขั้วต่อแต่ละด้านอยู่ในตำแหน่งที่ตรงกันก็สามารถใช้งานได้แล้ว



รูปแสดงวิธีการต่อสาย LCD แบบใช้ขั้วแถวเดี่ยว

*****หมายเหตุ***** ใน CLCD บางรุ่นนั้น อาจมีขั้วต่อสัญญาณเพิ่มเป็น ขนาด 16 PIN ซึ่งในกรณีนี้ ก็จะยังคงใช้วิธีการเชื่อมต่อแบบเดิม คือจะใช้สัญญาณในการเชื่อมต่อระหว่าง CLCD กับบอร์ด เพียงแค่ 14 PIN เท่านั้น ส่วนสัญญาณขา 15 และ 16 ที่เพิ่มเข้ามานั้นจะเป็นขาไฟเลี้ยงของวงจร LED Back-Light (A และ K) ซึ่งถ้า LCD ที่ซื้อมาใช้งานมี LED Back-Light รวมอยู่ด้วย ขอแนะนำให้แยกต่อไฟเลี้ยง LED Back-light เข้ากับแหล่งจ่ายไฟ +5V โดยตรงต่างหากก็ได้ หรือถ้าต้องการให้ LED Back-Light ทำงานตลอดเวลาก็อาจทำการต่อขาสัญญาณ (A) หรือขา 15 เข้ากับ ขา 2 ของ LCD ส่วนขา (K) ก็ให้ต่อเข้ากับขา 1 ของ LCD ก็ได้เช่นกัน

การเชื่อมต่อกับอุปกรณ์ I²C BUS

สำหรับอุปกรณ์แบบ I²C Bus ที่ใช้ในบอร์ด CP-JR51AC2 V2.0 นั้น จะออกแบบให้สามารถติดตั้งใช้งานอุปกรณ์ I²C ได้พร้อมกันในบอร์ดทั้งหมดด้วยกัน 3 ตัว คือ

- I²C RTC เบอร์ PCF8583 ของ PHILIPS
- EEPROM ในตระกูล 24XX ซึ่งสามารถเลือกใช้ได้หลายเบอร์หลายผู้ผลิต ขึ้นอยู่กับขนาดความจุของหน่วยความจำที่ต้องการจะใช้ ซึ่งในบอร์ด CP-JR51AC2 V2.0 นั้น สามารถติดตั้งใช้งานหน่วยความจำ EEPROM แบบ I²C นี้ได้ตั้งแต่ เบอร์ 24XX32,24XX64,24XX128,24XX256 หรือ 24XX512 เป็นต้น
- I²C I/O เบอร์ PCF8574 หรือ PCF8574A ของ Phillips

โดยอุปกรณ์ I²C ทั้ง 3 ตัวนี้จะต่อร่วมกันอยู่ภายในบัสเดียวกัน และใช้สัญญาณ P4.1 เป็นขาสัญญาณ SDA และใช้สัญญาณ P4.0 เป็นสัญญาณ SCL ในการควบคุมบัส ซึ่ง CPU จะทำหน้าที่เป็นตัวแม่ในการควบคุมบัส นอกจากนี้แล้วยังสามารถขยายอุปกรณ์จำพวก I²C นี้ได้อีก แต่ต้องเป็นอุปกรณ์ที่มีรหัสควบคุม Control Word ไม่ซ้ำกันกับอุปกรณ์ที่มีอยู่แล้วภายในบอร์ดด้วย โดยอาจเชื่อมต่อผ่านทางขั้วต่อ “I²C EXPANSION” ที่บอร์ดจัดเตรียมไว้ให้แล้วก็ได้

สำหรับในการใช้งานอุปกรณ์ I²C นั้น เนื่องจาก CPU เบอร์ T89C51AC2 ไม่มีส่วนของฮาร์ดแวร์ที่ทำหน้าที่ติดต่อสื่อสารแบบ I²C บรรจุไว้ในตัว CPU ด้วย ดังนั้นจึงต้องใช้วิธีการนำ I/O Port ของ CPU มาใช้ติดต่อกับอุปกรณ์แบบ I²C แทน ซึ่งอุปกรณ์ I²C นั้นต้องการสัญญาณในการติดต่อสื่อสารกันจำนวน 2 เส้นสัญญาณ คือ SCL(Clock) และ SDA(Data) โดยสัญญาณของ I/O Port ที่ใช้ในการติดต่อสื่อสารกับอุปกรณ์แบบ I²C ของบอร์ด CP-JR51AC2 V2.0 นั้น จะออกแบบให้สามารถเลือกใช้งานได้ 2 แบบ คือ

- **SDA** ทำหน้าที่เป็นสัญญาณข้อมูลแบบ 2 ทิศทาง ใช้สำหรับรับส่งข้อมูลจาก CPU กับอุปกรณ์ I²C โดยเมื่อ CPU ต้องการเขียนข้อมูลไปยังอุปกรณ์ ผู้ใช้จะต้องกำหนดให้สัญญาณนี้ทำหน้าที่เป็น Output แต่เมื่อ CPU ต้องการอ่านหรือรับข้อมูลจากอุปกรณ์ ผู้ใช้ก็ต้องกำหนดให้สัญญาณนี้ทำหน้าที่เป็น Input แทน โดยในบอร์ดกำหนดให้ P4.1 ทำหน้าที่เป็น SDA

- **SCL** ทำหน้าที่เป็นสัญญาณนาฬิกา Clock สำหรับใช้ติดต่อสื่อสารกับอุปกรณ์ I²C โดยสัญญาณที่ทำหน้าที่เป็น SCL นี้จะเป็นสัญญาณ Output จาก CPU เพื่อใช้ควบคุมการรับส่งข้อมูลระหว่าง CPU กับอุปกรณ์ โดยสัญญาณ I/O Port ของ CPU ที่จะสามารถใช้ทำหน้าที่เป็นสัญญาณ SCL สำหรับติดต่อสื่อสารกับอุปกรณ์ I²C นี้ โดยในบอร์ดจะใช้สัญญาณ P4.0 ทำหน้าที่เป็น SCL

การใช้งาน Interrupt ของอุปกรณ์ I^2C

สำหรับสัญญาณ Interrupt จากอุปกรณ์ I^2C นั้น เนื่องจากการออกแบบวงจรของบอร์ดจะกำหนดให้สัญญาณที่ใช้สำหรับรับการ Interrupt จากภายนอก INT0 ต่อร่วมกันกับสัญญาณ Interrupt ของ RTC และ Port I/O ไว้ ดังนั้นผู้ใช้จำเป็นต้องจัดสรรและเลือกว่า จะให้อุปกรณ์ตัวใดใช้สัญญาณ Interrupt ดังกล่าว โดยบอร์ด CP-JR51AC2 V2.0 นั้น จะออกแบบให้ผู้ใช้สามารถเลือกการใช้สัญญาณ Interrupt จาก INT0 ได้ว่า จะทำการเชื่อมต่อสัญญาณ INT0 เข้ากับอุปกรณ์ตัวใด โดยมีจุดเลือกการต่อสัญญาณ Interrupt ให้กับ INT0 ได้ทั้งหมด 3 แหล่ง คือ

- ต่อสัญญาณการ Interrupt INT0 ให้กับ P3.2 ทางขั้วต่อ 34PIN
- ต่อสัญญาณการ Interrupt INT0 ให้กับ RTC PCF8583
- ต่อสัญญาณการ Interrupt INT0 ให้กับ I/O Port PCF8574/A

สำหรับในส่วนของอุปกรณ์ I^2C ที่อยู่ภายในบอร์ดนั้น จะมีอยู่ 2 อุปกรณ์ด้วยกัน ที่สามารถสร้างสัญญาณ Interrupt INT0 ให้กับ CPU ได้ คือ RTC เบอร์ PCF8583 และ I/O Port เบอร์ PCF8574/A ซึ่งในการเลือก Interrupt ให้กับอุปกรณ์ทั้งสองนั้น ให้เลือกจาก Jumper INT0(I/O และ RTC) ระหว่าง I/O หรือ RTC อย่างใดอย่างหนึ่ง โดยถ้าเลือก Short Jumper INT0 ของ I/O จะหมายถึง ให้ I/O Port PCF8574/A ใช้งาน INT0 แต่ถ้าเลือก Short Jumper INT0 ของ RTC จะหมายถึง ให้ RTC PCF8583 ใช้งาน INT0

*****หมายเหตุ***** ไม่ควรทำการ Short Jumper ของ INT0 เข้ากับอุปกรณ์มากกว่า 1 อุปกรณ์ เนื่องจากจะเป็นการยุ่งยากในการเขียนโปรแกรมเพื่อตรวจสอบแหล่งที่มาของการ Interrupt ว่ามาจากอุปกรณ์ตัวใด

แอดเดรสของอุปกรณ์ I^2C

เนื่องจากคุณสมบัติของ BUS แบบ I^2C นั้น สามารถเชื่อมต่ออุปกรณ์ต่างๆที่ใช้วิธีการสื่อสารแบบ I^2C ได้มากมากหลายตัวภายในบัสเดียวกันได้ เพียงแต่มีข้อแม้ว่า อุปกรณ์ที่จะนำมาต่อร่วมกันภายในบัสเดียวกันนั้น จะต้องมีรหัสตำแหน่งในการติดต่อสื่อสาร (Control Byte) ที่ไม่ซ้ำกัน ซึ่งอุปกรณ์บางตัวนั้น ผู้ผลิตได้มีการออกแบบให้สามารถกำหนดค่ารหัสตำแหน่ง Control Byte ได้มากกว่า 1 ค่าเพื่อให้สามารถเชื่อมต่ออุปกรณ์ประเภทเดียวกันร่วมกันภายในบัสเดียวกันได้มากกว่า 1 ตัว โดยใช้วิธีการกำหนดค่าโลจิกให้กับขาสัญญาณสำหรับใช้ระบุตำแหน่ง (Address) ของอุปกรณ์เบอร์นั้นๆได้เอง เช่น I/O Port เบอร์ PCF8574 นั้น สามารถต่อร่วมกันภายในบัสเดียวกันได้มากถึง 8 ตัว และยังสามารถเชื่อมต่ออุปกรณ์ I/O Port ที่มีคุณสมบัติเหมือนกันแต่มีรหัสตำแหน่งที่แตกต่างกันคือ PCF8574A เพิ่มเติมได้อีก 8 ตัว ซึ่งจะเห็นได้ว่าอุปกรณ์ I/O Port นั้นสามารถทำการเพิ่มเติมเข้าไปให้ระบบบัสเดียวกันได้มากถึง 16 ตัว และในทำนองเดียวกัน หน่วยความจำ E^2PROM เบอร์ 24LC256 ก็สามารถต่อร่วมกันภายในบัสเดียวกันได้มากถึง 8 ตัว จากตัวอย่างข้างต้นจะเห็นได้ว่าภายในบัสเดียวกันของ I^2C นั้น อุปกรณ์เพียง 2 ประเภท คือ I/O และ E^2PROM สามารถต่อร่วมกันภายในบัสเดียวกันได้มากถึง 24 ตัว คือ I/O PCF8574 8 ตัว ,PCF8574A 8ตัว และ 24LC256 อีก 8 ตัว และยังสามารถนำอุปกรณ์ I^2C อื่นๆที่มีรหัสตำแหน่งของ Control Byte ไม่ซ้ำกันมาต่อเพิ่มเติมได้อีก แต่อย่างไรก็ตามถึงแม้ว่าอุปกรณ์แบบ I^2C นี้ยอมให้มีการเชื่อมต่อร่วมกันภายในบัสเดียวกันได้หลายตัวภายในระบบบัสเดียวกันก็ตาม แต่ในทางปฏิบัติแล้วอาจเกิดข้อจำกัดในเรื่องของโหลด (FAN-IN/FAN-OUT) เนื่องจากคุณสมบัติของ Port I/O ของ CPU เอง ก็มีข้อจำกับในการขับกระแสให้กับโหลดได้ประมาณ 15mA เท่านั้น ซึ่งคงไม่สามารถต่ออุปกรณ์ร่วมกันในบัสได้โดยไม่จำกัดจำนวนเหมือนในทฤษฎีบอกไว้ ซึ่งในความเป็นจริงอาจต้องพิจารณาตามความเหมาะสมและความจำเป็นในการใช้งานจริงๆด้วยว่าในระบบบัสหนึ่งๆของ I^2C นั้นควรต่ออุปกรณ์ในบัสจำนวนเท่าใด

หน้าที่และเบอร์ของอุปกรณ์ I^2C	**รหัสตำแหน่งมาตรฐานในการอ่าน/เขียน**	**รหัสตำแหน่งของบอร์ด CP-JR51AC2 V2.0**	
		รหัสตำแหน่งในการอ่าน	รหัสตำแหน่งในการเขียน
RTC : PCF8583	[1][0][1][0][0][0][X][?]	[1][0][1][0][0][0][1][1]	[1][0][1][0][0][0][1][0]
E^2PROM:24XX	[1][0][1][0][X][X][X][?]	[1][0][1][0][1][0][0][1]	[1][0][1][0][1][0][0][0]
I/O : PCF8574	[0][1][0][0][X][X][X][?]	[0][1][0][0][0][0][0][1]	[0][1][0][0][0][0][0][0]
I/O : PCF8574A	[0][1][1][1][X][X][X][?]	[0][1][1][1][0][0][0][1]	[0][1][1][1][0][0][0][0]

ตารางแสดง รหัสตำแหน่งของอุปกรณ์ I2C ภายในบอร์ด CP-JR51AC2 V2.0

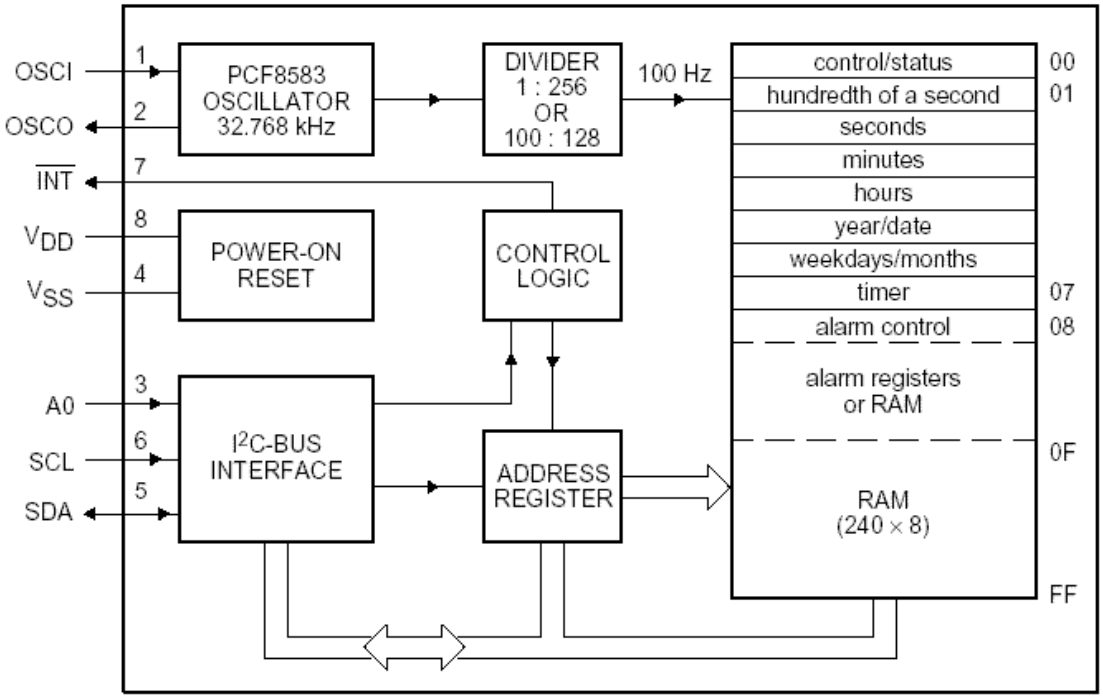
*****หมายเหตุ***** 1. ค่า X หมายถึงค่าโลจิกของขาสัญญาณ Address ของอุปกรณ์ ที่กำหนดในวงจร

2. ค่า ? หมายถึงบิตสำหรับกำหนดว่าต้องการเขียน หรือ อ่าน ข้อมูลกับอุปกรณ์

3. เนื่องจาก RTC เบอร์ PCF8583 และ E^2PROM เบอร์ในกลุ่ม 24XX นั้นมีรหัสตำแหน่ง 4 บิตแรก ซ้ำกัน หรือ อยู่ในกลุ่มเดียวกัน ซึ่งในบอร์ด CP-JR51AC2 V2.0 นั้นออกแบบให้ RTC เบอร์ PCF8583 มีรหัสตำแหน่งของ Control Byte คงที่เป็น 1010001X ไว้ ส่วน E^2PROM ก็กำหนดรหัสตำแหน่ง Control Byte ไว้ที่ 1010100X ดังนั้นถ้าต้องการเพิ่มเติมอุปกรณ์ใดๆเข้าไปอีกต้องกำหนดให้ค่า Control Byte ของอุปกรณ์ที่จะต่อเพิ่มเข้าไปไม่ซ้ำกับค่า Control Byte ทั้งสองดังกล่าวนี้ด้วย

การใช้งาน I^2C RTC เบอร์ PCF8583

สำหรับวงจรฐานเวลา RTC นั้น ในบอร์ด CP-JR51AC2 V2.0 นั้น จะเลือกใช้ Chips Support ของ PHILIPS เบอร์ PCF8583 ซึ่งเป็นชิพฐานเวลาแบบ I^2C-Bus และมีฐานเวลาให้ใช้งานอย่างครบถ้วน ตั้งแต่ วินาที/นาที/ชั่วโมง/วันที่/เดือน/วันในสัปดาห์ และปีคศ. นอกจากนี้ยังมีความอ่อนตัวในการใช้งานค่อนข้างดีเกี่ยวกับระบบเวลา เช่น ค่าของชั่วโมงสามารถกำหนดได้จากโปรแกรมว่าจะให้เป็นระบบ 12 ชั่วโมง หรือ 24 ชั่วโมง และในส่วนของวันที่และวันในสัปดาห์ก็สามารถปรับเปลี่ยนได้เองว่า เดือนใดมี 28/29/30 หรือ 31 วันอย่างอัตโนมัติ ซึ่งนอกจากจะใช้งานเป็นฐานเวลา RTC แล้ว PCF8583 นี้ยังมีฟังก์ชั่นพิเศษในการตั้งเวลาสำหรับเปิดปิดการทำงานของอุปกรณ์ต่างๆ (ALARM FUNCTION) ได้อีกด้วย นอกจากนี้แล้วในตัวของ RTC เองยังมีหน่วยความจำ RAM ขนาด 8บิต จำนวน 240ไบท์ สำหรับให้ผู้ใช้นำไปใช้งานเก็บข้อมูลได้อย่างอิสระ เช่น อาจนำไปใช้ในการเก็บค่าการตั้ง เวลา เพื่อใช้ ตั้งเวลาเปิด-ปิด อุปกรณ์ไฟฟ้า เป็นต้น



รูปแสดงโครงสร้างภายในของ RTC เบอร์ PCF8583

จะเห็นได้ว่า PCF8583 ประกอบขึ้นจากวงจรหลายๆส่วน เช่น วงจร Power-on Reset วงจรเชื่อมต่อแบบ I^2C วงจรถอดรหัสตำแหน่งแอดเดรส วงจรหารความถี่ และวงจรกำเนิดความถี่ขนาด 32.768KHz โดยต้องต่อคริสตัลจากภายนอกให้กับขา OSCI และ OSCO ด้วย สำหรับหน่วยความจำนั้น PCF8583 จะมีโครงสร้างของหน่วยความจำขนาด 8บิต จำนวน 256 ไบท์ โดยจัดสรรสำหรับแบ่งออกเป็นรีจิสเตอร์ของส่วนที่เป็นฐานเวลาจำนวน 16ไบท์(00H-0FH) และใช้งานเป็น หน่วยความจำ RAM ทั่วไปได้อีก 240ไบท์(10H-FFH) ซึ่งในการประยุกต์ใช้งานนั้น ตามปรกติแล้วจะสามารถใช้งานในหน้าที่ของ RTC(Clock Mode) หรือใช้งานเป็นตัวนับ Counter (Event Counter) สำหรับนับความถี่จากขาสัญญาณ OSCI ก็ได้ แต่สำหรับวงจรของ PCF8583 ภายในบอร์ด CP-JR51AC2 V2.0 นั้นจะออกแบบให้ใช้งาน PCF8583 ในโหมด RTC หรือ Clock Mode เท่านั้น

การติดต่อสื่อสารกับ RTC เบอร์ PCF8583

ในการเขียนโปรแกรมติดต่อกับ RTC นั้น จะใช้วิธีการเชื่อมต่อแบบมาตรฐาน I^2C Bus โดยใน RTC เบอร์ PCF8583 นี้จะมีตำแหน่งแอดเดรสในการติดต่อภายในบัส หรือ Control Byte เป็น “1010001X” ดังนั้นในการติดต่อกับ RTC ไม่ว่าจะเป็นการเขียนข้อมูลหรืออ่านข้อมูลจากตัว RTC ก็ตามที หลังจากสร้างสภาวะเริ่มต้น (Start Condition) แล้วจะต้องส่งค่า Control Byte ของตัว RTC ในบัส ด้วยค่า “1010001X” เพื่อบอกให้ RTC รับรู้ว่า CPU ต้องการจะอ่านหรือเขียนข้อมูลให้กับ RTC จากนั้นจึงส่งรหัส ไบท์แอดเดรส เพื่อระบุตำแหน่งแอดเดรสเริ่มต้นภายในตัว RTC ที่ต้องการจะอ่านหรือเขียนข้อมูลให้กับ RTC เป็นลำดับต่อไป โดยถ้าเป็นตำแหน่งแอดเดรสของฐานเวลาภายในตัว RTC จะมีค่าตำแหน่งแอดเดรสอยู่ระหว่าง 00H-0FH แต่ ถ้าเป็นตำแหน่งแอดเดรสของ RAM ภายในตัว RTC จะมีค่าอยู่ระหว่าง 10H-FFH ตามลำดับ โดยรหัส Control Byte ของ RTC นั้นมีลักษณะโครงสร้างดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
1	0	1	0	0	0	A0	$R/\overline{W}$

รูปแสดง โครงสร้างของ Control Byte ของ PCF8583

*****หมายเหตุ***** บิต0 (R/W) นั้นจะใช้สำหรับกำหนดว่าจะอ่านหรือเขียนข้อมูลจากอุปกรณ์

ซึ่งจะเห็นว่าตามสภาวะปรกติแล้ว Control Byte ของ PCF8583 นั้น สามารถเลือกได้ 2 ค่า โดยการกำหนดโลจิกให้กับขาสัญญาณ A0 ของ PCF8583 เอง ดังนั้นในระบบบัสเดียวกัน จึงสามารถทำการติดตั้ง RTC เบอร์ PCF8583 นี้ได้ 2 ตัว โดยต้องกำหนดให้ขาสัญญาณ A0 ของแต่ละตัวมีสภาวะเป็น “0” และ “1” ซึ่ง ตัวที่ขาสัญญาณ A0 มีสภาวะเป็น “0” ก็จะมีรหัส Control Byte เป็น “1010000X” ส่วนตัวที่ขาสัญญาณ A0 ได้รับสภาวะลอจิกเป็น “1” ก็จะมีรหัส Control Byte เป็น “1010001X” แทน

แต่สำหรับบอร์ด CP-JR51AC2 V2.0 นั้น จะกำหนดให้ขาสัญญาณ A0 ของ PCF8583 มีสภาวะทางโลจิกเป็น “1” คงที่ไว้เลย ดังนั้น RTC เบอร์ PCF8583 ในบอร์ด CP-JR51AC2 V2.0 นั้นจึงมีรหัส Control Byte คงที่เป็น “1010001X” เสมอ

*****หมายเหตุ***** ค่า X หรือ บิต0 ใน Control Byte เป็นบิตสำหรับกำหนดคุณสมบัติในการอ่านหรือเขียนข้อมูลกับอุปกรณ์ I^2C โดยถ้าหากว่าบิต0 มีค่าเป็น “0” จะหมายถึง CPU ต้องการเขียนค่าไปยังอุปกรณ์ แต่ถ้าค่าในบิต0 มีค่าเป็น “1” จะหมายถึง CPU ต้องการอ่านค่าจากอุปกรณ์ เช่นรหัส Control Byte ของ RTC เบอร์ PCF8583 มีค่า “1010001X” ถ้าต้องการเขียนค่าไปยัง RTC จะต้องส่งรหัส Control Byte เป็น “10100010” แต่ถ้าต้องการอ่านค่าจาก RTC ก็จะต้องส่งรหัส Control Byte ด้วยค่า “10100011” แทน เป็นต้น

การใช้งานหน่วยความจำ E^2PROM (24XX)

หน่วยความจำ Serial EEPROM ที่ใช้ในบอร์ดจะใช้การเชื่อมต่อแบบ I^2C-Bus ในตระกูล 24XX ซึ่งหน่วยความจำแบบนี้มีคุณสมบัติที่น่าสนใจหลายประการคือ มีตัวถังขนาดเล็ก ใช้สัญญาณในการเชื่อมต่อน้อยเส้น และสามารถเก็บรักษาข้อมูลไว้ได้นานกว่า 200 ปี นอกจากนี้ยังสามารถลบและเขียนซ้ำได้ถึง 1 ล้านครั้ง (อ้างอิงจาก Microchip) จึงสามารถนำไปประยุกต์ใช้งาน ในด้านที่เกี่ยวข้องกับการเก็บรักษาข้อมูลสำหรับงานต่างๆได้ดี

โดยผู้ใช้เองสามารถเลือกติดตั้งหน่วยความจำเพื่อใช้งาน กับบอร์ดได้มากมายหลายเบอร์ ขึ้นอยู่กับจุดประสงค์และขนาดของหน่วยความจำที่ต้องการ โดยให้เลือกใช้ E^2PROMในตระกูล 24XX (I^2C Bus) ในกลุ่มที่สามารถตำแหน่งรหัส Control Byte ของหน่วยความจำจากฮาร์ดแวร์ (ขาสัญญาณ A2,A1 และ A0) ได้ เช่น เบอร์ 24XX32,64,128 และ 24XX256 ของ Microchip เป็นต้น

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
1	0	1	0	A2	A1	A0	$R/\overline{W}$

รูปแสดง รหัส Control Byte ของ 24XX32/64/128/256 ของ Microchip

สำหรับหน่วยความจำเบอร์ 24XX32,24XX64,24XX128 และ 24XX256 ของ Microchip นั้น จะเห็นได้ว่ารหัส Control Byte ในตำแหน่ง 4บิตบน (บิต7,6,5 และ 4) จะมีค่าเป็น “1010” ส่วน บิต3 บิต2 และ บิต1 นั้นจะขึ้นอยู่กับสภาวะทางลอจิกของขาสัญญาณ A2,A1 และ A0 ในวงจร ซึ่งจากคุณสมบัติดังกล่าวจะทำให้สามารถทำการต่อหน่วยความจำดังกล่าวได้มากถึง 8 ตัวภายในระบบบัสเดียวกัน โดยกำหนดสภาวะของขาสัญญาณ ลอจิก แอดเดรสที่แตกต่างกันออกไป โดยสามารถสรุปให้เห็นได้ดังตารางต่อไปนี้

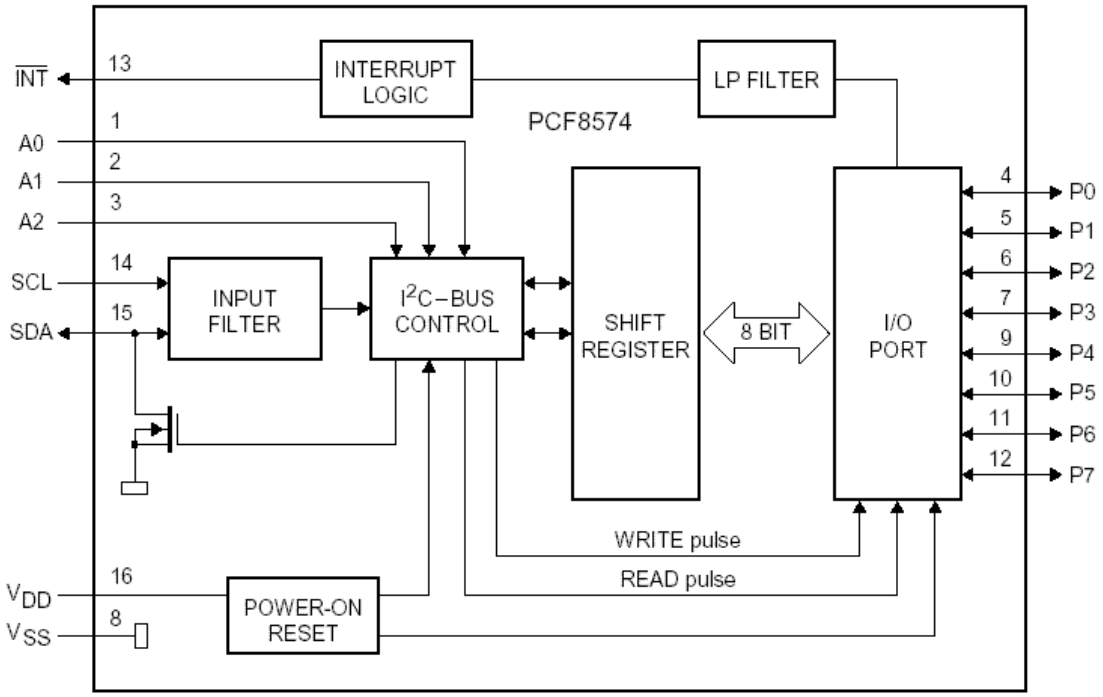
เบอร์(ความจุ)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
24XX32 (4Kx8)	1	0	1	0	A2	A1	A0	R/W
24XX64 (8Kx8)	1	0	1	0	A2	A1	A0	R/W
24XX128 (16Kx8)	1	0	1	0	A2	A1	A0	R/W
24XX256 (32Kx8)	1	0	1	0	A2	A1	A0	R/W

ตารางแสดงรหัส Control Byte ของหน่วยความจำแบบ I^2C Bus ของ Microchip

จากตารางจะเห็นได้ว่า หน่วยความจำ E^2PROM แบบ I^2C-BUS นั้น 24XX32/64/128/256 ของ Microchip นั้นจะมีรหัส Control Code ที่เหมือนกันทุกเบอร์ แต่จะมีความแตกต่างกันที่ ดังนั้นเมื่อทำการติดตั้งใช้งานหน่วยความจำเบอร์เหล่านี้กับบอร์ด CP-JR51AC2 V2.0 แล้วจะมีรหัส Control Byte เป็น “1010100X” คงที่ตลอด แต่ถ้ามีการต่อหน่วยความจำเหล่านี้เพิ่มเติมจากภายนอกบอร์ดแล้วรหัส Control Byte ก็จะขึ้นอยู่กับการกำหนดสภาวะทางลอจิกให้กับขาสัญญาณ A2,A1 และ A0 ของหน่วยความจำที่ต่อไว้

การใช้งาน I/O Port แบบ I^2C (PCF8574/A)

ตามปรกติแล้ว CPU เบอร์ T89C51AC2 นั้นจะมีพอร์ต I/O สำหรับให้ผู้ใช้สามารถนำไปใช้งานได้มากถึง 5 พอร์ต อยู่แล้ว ซึ่งในส่วนของบอร์ด CP-JR51AC2 V1.0 นั้น พอร์ต I/O ทั้งหมดของ CPU จะปล่อยว่างไว้ให้ผู้ใช้เลือกใช้งานอย่างอิสระตามต้องการ แต่สำหรับบอร์ดรุ่น CP-JR51AC2 V2.0 นั้น พอร์ต I/O ต่างๆของ CPU จะถูกจัดสรรออกไปใช้งานในวงจรต่างๆ ดังได้กล่าวอธิบายมาแล้วในข้างต้น แต่ถ้าหากว่าผู้ใช้งานมีความจำเป็นต้องใช้งานพอร์ต I/O จำนวนมาก และจำนวนพอร์ต I/O ของ CPU ที่มีอยู่ไม่เพียงพอต่อการใช้งานแล้ว ผู้ใช้ก็สามารถทำการเพิ่มเติมพอร์ต I/O ได้อีก โดยในบอร์ด CP-JR51AC2 V2.0 นั้นจะออกแบบให้ผู้ใช้สามารถทำการเพิ่มเติม พอร์ต I/O แบบ I^2C ซึ่งมีขนาด I/O จำนวน 8 บิต I/O โดยใช้ไอซี สำหรับทำหน้าที่เป็นพอร์ต I/O ของ Phillips เบอร์ PCF8574 หรือ PCF8574A โดย PCF8574/A มีโครงสร้างดังรูป



รูปแสดง Block Diagram ของ PCF8574/A

นอกจากนี้แล้วผู้ใช้งานยังสามารถทำการขยาย จำนวนพอร์ต I/O ของ PCF8574/A นี้ได้อีกมากถึง 15 ตัว (120 บิต I/O) ทางขั้วต่อ “**I^2C EXPAND**” ของบอร์ด เนื่องจาก PCF8574 หรือ PCF8574A นั้น สามารถต่อร่วมกันภายในระบบบัสเดียวกันได้มากถึงอย่างละ 8 ตัว กล่าวคือ ในระบบบัสของ I^2C นั้น จะสามารถต่อใช้งาน PCF8574 ได้มากถึง 8 ตัว และยังสามารถต่อพอร์ต I/O เบอร์ PCF8574A ได้อีก 8 ตัว รวมเป็น 16 ตัวภายในบัสเดียวกัน โดยการกำหนดตำแหน่งแอดเดรสของอุปกรณ์แต่ละตัวให้มีความแตกต่างกัน ซึ่งตามปรกติแล้ว PCF8574 หรือ PCF8574A นั้น จะมีขาสัญญาณแอดเดรสจำนวน 3 เส้น คือ A0,A1 และ A2 โดยการกำหนดสภาวะทางโลจิกให้กับขาสัญญาณแอดเดรสทั้ง 3 ให้มีค่าไม่ซ้ำกัน โดย PCF8574 และ PCF8574A นั้น จะมีคุณสมบัติและวิธีการใช้งานที่เหมือนกันทุกประการ แตกต่างกันเพียงรหัส Control Byte เท่านั้น โดยโครงสร้างของรหัส Control Byte ของ PCF8574 และ PCF8574A สามารถแสดงให้เห็นได้ดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	1	0	0	A2	A1	A0	$R/\overline{W}$

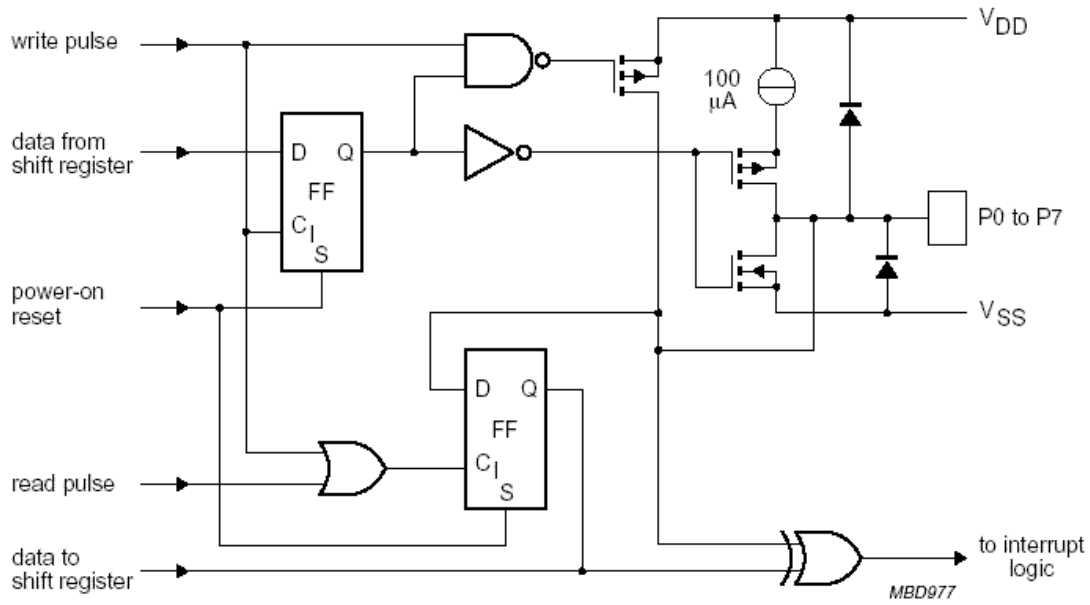
รูปแสดง รหัส Control Byte **ของ** PCF8574

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	1	1	1	A2	A1	A0	$R/\overline{W}$

รูปแสดง รหัส Control Byte **ของ** PCF8574A

สำหรับรหัส Control Byte ของพอร์ต I/O เบอร์ PCF8574 หรือ PCF8574A ของบอร์ดนั้น จะถูกกำหนดไว้ตายตัว โดยขาสัญญาณแอดเดรส A0,A1 และ A2 จะถูกกำหนดสภาวะโลจิกเป็น “0” ไว้ทั้งหมด ซึ่งในกรณีที่ผู้ใช้ทำการติดตั้งพอร์ต I/O เบอร์ PCF8574 จะมีรหัส Control Byte เป็น “0100000X” แต่สำหรับกรณีที่ผู้ใช้ทำการติดตั้งพอร์ต I/O เบอร์ PCF8574A จะมีรหัส Control Byte เป็น “0111000X” แทน

โดยที่ พอร์ต I/O เบอร์ PCF8574/A นั้น ตามปรกติแล้วจะสามารถใช้งานเป็น Input หรือ Output ก็ได้ตามต้องการ แต่จำเป็นต้องเลือกกำหนดหน้าที่เพียงหน้าที่เดียวเท่านั้น ไม่สามารถใช้งานเป็นทั้ง Input และ Output ในเวลาเดียวกันได้ โดยลักษณะโครงสร้างภายในของขาสัญญาณ I/O ของ PCF8574 เป็นดังนี้



รูปแสดง ลักษณะโครงสร้างของขาสัญญาณ I/O **แต่ละขาของ** PCF8574/A

การใช้งานพอร์ตสื่อสารอนุกรม RS232/RS422/RS485

ภายในตัว CPU เบอร์ T89C51AC2 ที่ใช้กับบอร์ด CP-JR51AC2 นั้น จะมีวงจรสื่อสารแบบอนุกรม (UART) บรรจุรวมไว้ด้วยแล้ว ซึ่งวงจรส่วนนี้ผู้ใช้งานสามารถทำการเขียนโปรแกรมควบคุมการสื่อสารข้อมูลของ CPU กับอุปกรณ์อื่นๆได้ตามต้องการ โดยในส่วนของโปรแกรมนั้น ผู้ใช้สามารถกำหนดรูปแบบของการสื่อสารข้อมูลได้เองจากโปรแกรมที่เขียนขึ้น ไม่ว่าจะเป็นความเร็วในการสื่อสาร (Baudrate) จำนวนบิตข้อมูลในการรับส่ง (Data Bit) การกำหนดบิตรวจสอบความถูกต้องข้อมูล (Parity) และคุณสมบัติอื่นๆ ซึ่งในรายละเอียดส่วนนี้จะไม่ขอกล่าวถึง ขอให้ผู้ใช้ศึกษาจากคู่มือสถาปัตยกรรมทางฮาร์ดแวร์ หรือ Data Sheet ของ CPU เบอร์ T89C51AC2 เอง

ซึ่งปรกติแล้วขาสัญญาณสำหรับ รับ-ส่ง ข้อมูลของ CPU นั้น สามารถนำไปเชื่อมต่อกับขาสัญญาณรับ-ส่ง ของอุปกรณ์อื่นๆได้แล้ว โดยขาส่ง (TX) ของ CPU ต้องนำไปต่อกับขารับ (RX) ของอุปกรณ์ที่จะนำมาสื่อสารกัน ส่วนขารับข้อมูล (RX) ของ CPU ก็ต้องต่อกับขาส่งข้อมูล (TX) จากอุปกรณ์ที่จะนำมาสื่อสารกัน แต่เนื่องจากขาสัญญาณ RX และ TX ของ CPU นั้น จะสามารถเชื่อมต่อกับสัญญาณที่มีคุณสมบัติเป็นแบบ ระดับลอจิก TTL เท่านั้น ซึ่งถ้าใช้วิธีการเชื่อมต่อสัญญาณรับส่งของ CPU กับอุปกรณ์โดยตรงนั้น จะสามารถสื่อสารกันได้เพียงระยะทางใกล้ๆหรือภายในแผงวงจรเดียวกันเท่านั้น ไม่สามารถสื่อสารกันด้วยระยะทางไกลๆได้ ดังนั้นบอร์ด CP-JR51AC2 จึงได้ออกแบบวงจร Line Driver สำหรับทำหน้าที่เป็น Buffer เพื่อเปลี่ยนแปลงระดับสัญญาณทางไฟฟ้าของขาสัญญาณ รับ-ส่ง ข้อมูลของ CPU ที่เป็น แบบ TTL ให้สามารถรับส่งข้อมูลกันได้ในระยะทางที่ไกลมากขึ้น (สามารถอ่านรายละเอียดเพิ่มเติมได้จากหัวข้อ “ความรู้ทั่วไปเกี่ยวกับการสื่อสารอนุกรม” ในภาคผนวกท้ายเล่มของคู่มือนี้) โดยบอร์ด CP-JR51AC2 นั้น จะสามารถเลือกกำหนดรูปแบบของวงจร Line Driver สำหรับการสื่อสารอนุกรมได้ 3 แบบด้วยกัน คือ

การสื่อสารอนุกรมแบบ RS232

ในกรณีนี้จะต้องทำการติดตั้งไอซี Line Driver เพื่อเปลี่ยนระดับสัญญาณทางไฟฟ้าของขาสัญญาณสำหรับ รับ-ส่ง ข้อมูลแบบ TTL ของ CPU (RX และ TX) ให้เป็นระดับสัญญาณทางไฟฟ้าแบบ RS232 (±12V) โดยการติดตั้งไอซีเบอร์ MAX232 เพื่อทำหน้าที่เปลี่ยนระดับสัญญาณ TTL จากขาสัญญาณส่งข้อมูล (TX) ของ CPU ให้เป็นระดับสัญญาณ ±12V สำหรับส่งไปยังขารับสัญญาณ (RX) ของอุปกรณ์ภายนอก และในทางกลับกัน ก็จะทำหน้าที่เปลี่ยนระดับสัญญาณส่ง (TX) แบบ RS232 (±12V) จากอุปกรณ์ภายนอก ให้กลับมาเป็นระดับ TTL เพื่อส่งให้กับขารับข้อมูล (RX) ของ CPU ด้วย โดยเมื่อเปลี่ยนระดับสัญญาณในการรับส่งข้อมูลจาก TTL มาเป็นแบบ RS232 นี้แล้วจะทำให้สามารถทำการ รับ-ส่ง ข้อมูลกับอุปกรณ์ภายนอกที่ใช้ระดับสัญญาณทางไฟฟ้าในการ รับ-ส่ง แบบเดียวกัน (RS232) ได้ไกลขึ้น ประมาณ 50ฟุต หรือ ประมาณ 15 เมตร โดยสามารถทำการ รับ-ส่ง ข้อมูลกับอุปกรณ์ต่างๆได้ในลักษณะของตัวต่อตัว (Point-to-Point) เท่านั้น

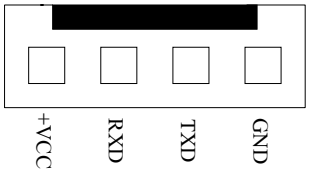
สำหรับสายสัญญาณที่จะนำมาใช้สำหรับทำการสื่อสารแบบ RS232 นั้น จะใช้สัญญาณเพียง 2-3 เส้นเท่านั้น ทั้งนี้ขึ้นอยู่กับความต้องการในการสื่อสารว่าต้องการสื่อสารแบบทิศทางเดียวหรือสองทิศทาง

- **การสื่อสาร RS232 แบบสองทิศทาง** ซึ่งจะมีทั้งการรับข้อมูลและส่งข้อมูลไปมา ระหว่างด้านรับและด้านส่ง โดยในกรณีนี้จะต้องใช้สายสัญญาณจำนวน 3 เส้น สัญญาณรับข้อมูล (RXD) สัญญาณส่งข้อมูล(TXD) และสัญญาณอ้างอิง (GND) โดยในการเชื่อมต่อสายนั้นจะต้องทำการ

สลับสัญญาณกับอุปกรณ์ปลายทางด้วย คือ สัญญาณส่ง (TXD) จากบอร์ด CP-JR51AC2 จะต้องต่อเข้ากับสัญญาณรับ (RXD) ของอุปกรณ์ และสัญญาณส่ง (TXD) จากอุปกรณ์ก็ต้องต่อกับสัญญาณรับ (RXD) ของบอร์ด ส่วนสัญญาณอ้างอิง (GND) จะต้องต่อตรงถึงกัน จึงจะสามารถทำการ รับ-ส่ง ข้อมูลกันได้

- **การสื่อสาร** RS232 **แบบทิศทางเดียว** ซึ่งอาจเป็นการรอรับข้อมูลจากด้านส่งเพียงอย่างเดียว หรืออาจเป็นการส่งข้อมูลออกไปยังปลายทางเพียงอย่างเดียว โดยไม่มีการโต้ตอบข้อมูลซึ่งกันและกัน ซึ่งวิธีนี้จะใช้สายสัญญาณเพียง 2 เส้น เท่านั้น โดยถ้าเป็นทางด้านส่ง ก็จะต่อเพียงสัญญาณส่ง (TXD) และสัญญาณอ้างอิง (GND) แต่ถ้าเป็นทางด้านรับ ก็จะต่อเพียงสัญญาณรับ (RXD) และ สัญญาณอ้างอิง (GND) เท่านั้น

โดยขั้วต่อของสัญญาณ RS232 ของบอร์ด CP-JR51AC2 ทั้ง 2 รุ่น นั้น จะเป็นจุดเชื่อมต่อของสัญญาณ รับ-ส่ง ข้อมูล ที่เปลี่ยนระดับสัญญาณเป็นแบบ RS232 แล้ว ซึ่งจะมีลักษณะเป็นแบบขั้ว CPA ขนาด 4 PIN สำหรับใช้เป็นจุดเชื่อมต่อสัญญาณ รับ-ส่ง ข้อมูลกับอุปกรณ์ภายนอก โดยมีลักษณะการจัดเรียงสัญญาณดังนี้



แสดงขั้วต่อสัญญาณ RS232 **ของบอร์ด** CP-JR51AC2 V1.0 & V2.0

ซึ่งจะเห็นได้ว่าขั้วต่อสัญญาณ RS232 ของบอร์ดนั้น จะมีทั้งหมด 4 เส้น แต่ในการ รับ-ส่ง ข้อมูลแบบปรกตินั้น จะใช้สัญญาณเพียงแค่ 3 เส้น คือ RXD,TXD และ GND เท่านั้น ส่วน +VCC ซึ่งเป็นไฟเลี้ยงวงจร +5V นั้น จะไม่จำเป็นต้องนำมาใช้ในการสื่อสารกันแต่อย่างใด โดย +VCC หรือ +5V นี้ จะออกแบบเผื่อไว้ในกรณีที่อุปกรณ์ปลายทางเป็นวงจรขนาดเล็กและไม่สะดวกที่จะหาแหล่งจ่ายไฟให้กับอุปกรณ์ปลายทางด้วย ก็อาจต่อไฟเลี้ยงวงจร +VCC นี้ออกไปให้กับอุปกรณ์ปลายทางด้วยก็ได้เช่นกัน

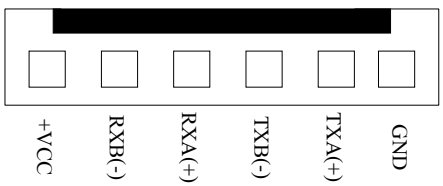
******หมายเหตุ***** สำหรับไอซี Line Drive แบบ RS232 นั้น จะจัดเป็นอุปกรณ์มาตรฐานของบอร์ดในตระกูล CP-JR51AC2 ซึ่งจะมีติดตั้งให้ไปกับบอร์ดอยู่แล้ว ผู้ใช้ไม่ต้องจัดหาเพิ่มเติม แต่พึงระลึกไว้เสมอว่า จะต้องทำการติดตั้งไอซี Line Driver สำหรับเลือกชนิดสัญญาณทางไฟฟ้าของการสื่อสารอนุกรม ได้เพียงอย่างใดอย่างหนึ่งเท่านั้น เช่น เมื่อเลือกติดตั้งไอซี Line Driver เป็นแบบ RS232 โดยการติดตั้ง MAX232 ในบอร์ดแล้ว จะต้องไม่ติดตั้งไอซี Line Driver แบบอื่น เช่น RS422 หรือ RS485 เข้าไปด้วย เพราะจะทำให้ไม่สามารถรับส่งข้อมูลกันได้อย่างถูกต้อง ผู้ใช้ต้องเลือกติดตั้งไอซี Line Driver อย่างใดอย่างหนึ่งเท่านั้น

การสื่อสารอนุกรมแบบ RS422

ในกรณีนี้จะต้องทำการติดตั้งไอซี Line Driver เบอร์ 75176 หรือ MAX3088 จำนวน 1-2 ตัว เพื่อทำหน้าที่เปลี่ยนระดับสัญญาณการไฟฟ้าในการ รับ-ส่ง แบบ TTL จาก CPU ให้เป็นระดับสัญญาณแบบ Balance Line เพื่อ รับ-ส่งสัญญาณกับอุปกรณ์ที่มีระดับสัญญาณทางไฟฟ้าเป็นแบบ Balance Line เหมือนกัน โดยถ้าต้องการใช้การสื่อสารแบบ 2 ทิศทาง ก็จะต้องติดตั้งไอซี Line Driver จำนวน 2 ตัว โดยแบ่งเป็นตัวแปลงสัญญาณทางด้านรับ 1 ตัว และตัวแปลงสัญญาณด้านส่งอีก 1 ตัว แต่ถ้าต้องการสื่อสารแบบทิศทางเดียวก็อาจทำการติดตั้งไอซี Line Driver เพียงตัวเดียว โดยถ้าต้องการให้เป็นฝ่ายรับข้อมูลเพียงอย่างเดียวก็ให้ติดตั้งไอซี Line Driver เฉพาะในตำแหน่งของ “RXD/422” เพียงตัวเดียว แต่ถ้าต้องการให้เป็นฝ่ายส่งข้อมูลเพียงอย่างเดียวก็ให้ทำการติดตั้งไอซี Line Driver เฉพาะในตำแหน่ง “TXD/485” เพียงตัวเดียวเท่านั้น

ซึ่งการสื่อสารแบบ RS422 นี้ สามารถนำไปทดแทนการสื่อสารแบบ RS232 ได้ทันที โดยไม่ต้องดัดแปลงหรือแก้ไขโปรแกรมเลย ซึ่งการสื่อสารโดยใช้ระดับสัญญาณในการ รับ-ส่ง แบบ RS422 นี้จะมีข้อดี คือ สามารถทำการสื่อสารกันได้ในระยะทางที่ไกลขึ้นกว่าแบบ RS232 มาก กล่าวคือ สามารถจะทำการ รับ-ส่ง ข้อมูลกันได้ในระยะทางประมาณ 4000 ฟุต หรือ 1200 เมตร หรือ 1.2 กิโลเมตรเลยทีเดียว เพียงแต่ต้องใช้สายสัญญาณที่ออกแบบมาสำหรับรองรับการใช้งานในด้านการสื่อสารแบบนี้โดยเฉพาะ ซึ่งได้แก่ สายสัญญาณแบบ UTP (Un-Shiled Twist Pair) หรือ STP (Shiled Twist Pair) โดยการสื่อสารด้วยระดับสัญญาณทางไฟฟ้าแบบ RS422 นี้ ถ้าเป็นการสื่อสารแบบ 2 ทิศทาง คือ ทั้งรับข้อมูลและส่งข้อมูล จะสามารถทำการรับส่งข้อมูลกับอุปกรณ์ต่างๆได้ในลักษณะของตัวต่อตัว (Point-to-Point) เหมือนกับ RS232 ทุกประการ แต่ในกรณีที่เป็นการสื่อสารแบบทิศทางเดียวนั้น สามารถจะทำการต่อขนานสัญญาณทางด้านรับ จำนวนหลายๆจุด เข้ากับสัญญาณส่งเพียงจุดเดียวได้ โดยถ้าเลือกใช้ไอซี Line Driver เบอร์ 75176 จะสามารถต่อขนานจำนวนอุปกรณ์สำหรับด้านรับข้อมูลได้ประมาณ 32จุด แต่ถ้าเลือกใช้ไอซี Line Driver เบอร์ MAX3088 นั้น จะสามารถต่อขนานจำนวนอุปกรณ์ทางด้านรับข้อมูลได้มากถึง 256 จุด เลยทีเดียว แต่ถ้าเป็นอุปกรณ์ทางด้านส่งนั้น จะไม่สามารถนำมาต่อขนานสัญญาณส่งข้อมูลเข้าด้วยกันมากกว่า 1 จุด เหมือนทางด้านฝ่ายรับได้ ซึ่งวงจร Line Driver แบบ RS422 นี้จะมีอยู่เฉพาะในบอร์ดรุ่น CP-JR51AC2 V2.0 เท่านั้น

สำหรับลักษณะของขั้วต่อของสัญญาณ RS422 นั้น จะเป็นแบบ CPA ขนาด 6 PIN ดังรูป โดยในการสื่อสารกันนั้น จะใช้สายสัญญาณในการ รับ-ส่ง ข้อมูลกัน จำนวน 4 เส้น สัญญาณ คือ สัญญาณในการรับข้อมูลจำนวน 2 เส้น คือ RXA (RX+) และ RXB (RX-) และสัญญาณในการส่งข้อมูลอีก 2 เส้น คือ TXA (TX+) และ TXB (TX-) ซึ่งในการต่อสัญญาณนั้น จะต้องทำการต่อสัญญาณในลักษณะของการสลับกัน คือ สัญญาณส่งจะต้องต่อเข้ากับสัญญาณรับ นั่นก็คือ สัญญาณ RXA (RX+) จะต้องต่อกับ TXA (TX+) ส่วน RXB (RX-) ก็จะต้องต่อกับ TXB (TX-) ด้วยเช่นกัน โดยลักษณะของขั้วต่อสัญญาณ RS422 เป็นดังรูป



แสดงขั้วต่อสัญญาณ RS422/485 ของบอร์ด CP-JR51AC2 V2.0 เมื่อเลือกเป็น RS422

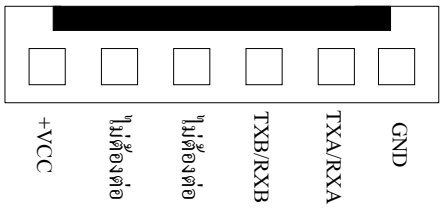
การสื่อสารอนุกรมแบบ RS485

ในการสื่อสารแบบ RS485 นี้จะมีคุณสมบัติของสัญญาณทางไฟฟ้าเหมือนกับ RS422 ทุกประการ เพียงแต่ว่าในการสื่อสารแบบ RS485 นี้จะใช้สายสัญญาณในการรับส่งข้อมูลกันเพียง 2 เส้น เท่านั้น แต่จะมีความพิเศษกว่าแบบ RS422 ตรงที่ ทิศทางของสัญญาณจะสามารถปรับเปลี่ยนได้จากโปรแกรม กล่าวคือ สัญญาณทั้ง 2 เส้น นี้สามารถจะสลับหน้าที่เป็นด้านส่ง และ เป็นด้านรับได้ ตามต้องการ โดยการควบคุมจาก CPU โดยจากบอร์ด CP-JR51AC2 V2.0 นั้น จะกำหนดให้สัญญาณ PTC3 ทำหน้าที่สำหรับควบคุมทิศทางของข้อมูลว่าจะให้เป็นรับหรือส่ง โดยถ้าควบคุมให้ PTC3 มีสภาวะเป็น “1” จะเป็นการกำหนดทิศทางให้เป็นฝ่ายส่งข้อมูล แต่ถ้าสภาวะของ PTC3 เป็น “0” จะเป็นการกำหนดทิศทางให้เป็นฝ่ายรับข้อมูล ซึ่งจากคุณสมบัติข้อนี้จะทำให้การสื่อสารแบบ RS485 สามารถทำการต่อขนานอุปกรณ์ร่วมกันในสายส่งเดียวกันได้จำนวนหลายๆจุด โดยถ้าใช้ไอซี Line Driver เบอร์ 75176 จะสามารถต่อขนานอุปกรณ์กันได้ จำนวน 32 จุด แต่ถ้าเลือกใช้ไอซี Line Driver เบอร์ MAX3088 แล้วจะสามารถต่อขนานอุปกรณ์ในสายคู่เดียวกันได้มากถึง 256 จุด เลยทีเดียว แต่มีข้อแม้ว่า เมื่อมีการต่ออุปกรณ์ขนานกันในสายสัญญาณคู่เดียวกันมากกว่า 2 จุดแล้ว จะต้องเขียนโปรแกรมควบคุมให้มีการส่งข้อมูลออกมาในสายครั้งละ 1 จุดเท่านั้น เพราะถ้ามีการกำหนดทิศทางของข้อมูลให้เป็นส่งในเวลาเดียวกันมากกว่า 1 จุดแล้วจะทำให้เกิดการชนกันของข้อมูลและไม่สามารถสื่อสารกันได้อย่างถูกต้อง

โดยเมื่อต้องการใช้วิธีการสื่อสารแบบ RS485 นี้ จะต้องทำการติดตั้งไอซี Line Driver เบอร์ 75176 หรือ MAX3088 ในตำแหน่งของ “TXD/485” เพียงตัวเดียว พร้อมกับเลือกกำหนดเป็นแบบ RS485 ดังนี้

- ทำการเลือก Jumper สำหรับเลือก “422/485” ไว้ทางด้าน 485 (RS485)
- ทำการเลือก Jumper “F/H” ไว้ทางด้าน H (Half Duplex)
- ทำการ Short Jumper สำหรับต่อตัวต้านทาน Fail Safe Resister คือ “TL” ไว้
- ทำการ Short Jumper สำหรับต่อตัวต้านทาน Fail Safe Resister คือ “TH” ไว้
- สายสัญญาณที่ใช้จะต่อจาก TXB(TX-) และ TXA(TX+) เพียง 2 เส้น ออกไปใช้งาน

ซึ่งในการสื่อสารข้อมูลแบบ RS485 นี้ จะต้องเขียนโปรแกรมขึ้นมารองรับการสื่อสารโดยเฉพาะ เนื่องจากทิศทางของข้อมูลสามารถจะกำหนดจากโปรแกรมได้โดยตรง ซึ่งการสื่อสารวิธีนี้จะมีข้อดีคือ ใช้สายสัญญาณในการรับส่งน้อยเส้น แต่จะเสียเวลาในการสื่อสารมากกว่าวิธีอื่นๆ เนื่องจากว่า การสื่อสารแบบนี้จะไม่สามารถทำการรับและส่งข้อมูลในเวลาเดียวกันได้ แต่จะต้องใช้วิธีการ ผลัดกันรับ ผลัดกันส่ง แทน ซึ่งในความเป็นจริงแล้วในปัจจุบันนี้ ราคาของสายสัญญาณแบบ 2 เส้น และ 4 เส้น แทบจะไม่มีความแตกต่างกันเลย ดังนั้นเพื่อลดความยุ่งยากในการเขียนโปรแกรมสำหรับควบคุมการรับส่งข้อมูลของ CPU ขอแนะนำให้เลือกใช้วิธีการสื่อสารแบบ RS422 จะง่ายและสะดวกรวดเร็วกว่ากันมาก



แสดงขั้วต่อสัญญาณ RS422/485 **ของบอร์ด** CP-JR51AC2 V2.0 **เมื่อเลือกเป็น** RS485

การกำหนด Jumper สำหรับการสื่อสารแบบ RS422/485

เนื่องจากวงจร Line Driver ของพอร์ตสื่อสารอนุกรมของบอร์ดนั้น ออกแบบให้ผู้ใช้สามารถเลือกกำหนดได้หลายแบบ ดังนั้น จึงต้องมีการใช้ Jumper สำหรับเป็นตัวเลือกรูปแบบการสื่อสารร่วมด้วย โดยจะมี Jumper ที่เกี่ยวข้องกับการใช้งานการสื่อสารแบบ RS422 และ RS485 ดังต่อไปนี้ คือ

- **Jumper 422/485** เป็น Jumper สำหรับเลือกกำหนดการทำงานของไอซี Line Driver ในตำแหน่ง TXD/485 ให้ทำงานเป็นแบบ RS422 หรือ RS485 โดยถ้าต้องการให้เป็นแบบ RS422 จะต้องกำหนด Jumper ไว้ทางด้าน “422” ซึ่งจะทำให้ไอซี Line Driver ตำแหน่ง “TXD/485” ทำหน้าที่เป็นฝ่ายส่งข้อมูลเพียงอย่างเดียว แต่ถ้าต้องการใช้งานแบบ RS485 จะต้องกำหนด Jumper ไว้ทางด้าน “485” เพื่อกำหนดให้ไอซี Line Driver ในตำแหน่ง “TXD/485” ทำหน้าที่เป็นทั้งฝ่ายรับและฝ่ายส่ง ตามการควบคุมของสัญญาณ PTC3
- **Jumper F/H** (Full/Half) เป็น Jumper ใช้สำหรับเลือกกำหนดรูปแบบการสื่อสารให้เป็นแบบ Full Duplex (F) หรือ Half Duplex (H) โดยถ้าต้องการใช้งานแบบ RS422 จะต้องเลือกกำหนด Jumper นี้ไว้ทางด้านด้าน F(Full Duplex) แต่ถ้าต้องการใช้งานเป็นแบบ RS485 จะต้องเลือกกำหนด Jumper นี้ไว้ทางด้าน H(Half Duplex) แทน
- **Jumper RL** เป็น Jumper ใช้สำหรับเลือกกำหนดการเชื่อมต่อ ตัวต้านทานสำหรับทำหน้าที่คงสถานะของสัญญาณ RXB (RX-) หรือ Fail Safe Resister เพื่อให้สัญญาณ RXB (RX-) มีสภาวะแน่นอนเมื่อไม่มีการส่งสัญญาณใดๆออกมาในสายเลย ซึ่งถ้าหากว่ามีการต่อสายสัญญาณระยะทางไกลๆหรือมีการต่อสายระยะทางไกล้ๆแต่ไม่ได้มีการส่งข้อมูลออกมาในสายตลอดเวลาแล้วควรที่จะทำการ Short Jumper นี้ไว้ด้วยเสมอ โดยเฉพาะอย่างยิ่งตัวอุปกรณ์ที่อยู่ในตำแหน่งต้นสายและปลายสายควรทำการ Short Jumper นี้ไว้เสมอ ส่วนอุปกรณ์ที่อยู่ในตำแหน่งอื่นๆที่มีระยะไม่ไกลจากจุดต้นสายและปลายสายมากนักก็อาจ Open Jumper นี้ออกก็ได้ แต่อย่างน้อยที่สุด ควรมีการ Short Jumper นี้ให้กับอุปกรณ์ที่ต่อร่วมอยู่ในสายสัญญาณจำนวน 1 จุดเสมอ
- **Jumper RH** เป็น Jumper ใช้สำหรับเลือกกำหนดการเชื่อมต่อ ตัวต้านทานสำหรับทำหน้าที่คงสถานะของสัญญาณ RXA (RX+) หรือ Fail Safe Resister เพื่อให้สัญญาณ RXA (RX+) มีสภาวะแน่นอนเมื่อไม่มีการส่งสัญญาณใดๆออกมาในสายเลย ซึ่งถ้าหากว่ามีการต่อสายสัญญาณระยะทางไกลๆหรือมีการต่อสายระยะทางไกล้ๆแต่ไม่ได้มีการส่งข้อมูลออกมาในสายตลอดเวลาแล้ว ควรที่จะทำการ Short Jumper นี้ไว้ด้วยเสมอ โดยเฉพาะอย่างยิ่งตัวอุปกรณ์ที่อยู่ในตำแหน่งต้นสายและปลายสายควรทำการ Short Jumper นี้ไว้เสมอ ส่วนอุปกรณ์ที่อยู่ในตำแหน่งอื่นๆที่มีระยะไม่ไกลจากจุดต้นสายและปลายสายมากนักก็อาจ Open Jumper นี้ออกก็ได้ แต่อย่างน้อยที่สุด ควรมีการ Short Jumper นี้ให้กับอุปกรณ์ที่ต่อร่วมอยู่ในสายสัญญาณจำนวน 1 จุดเสมอ
- **Jumper RZ** เป็น Jumper สำหรับเลือกกำหนดการต่อตัวต้านทาน RZ เพื่อชดเชย ค่าความต้านทานของสายสัญญาณ (Impedance) ทางด้านรับ ซึ่งถ้าหากว่ามีการต่อสายสัญญาณในการรับส่งเป็นระยะทางไกลๆแล้วก็ควรทำการ Short Jumper นี้ไว้ด้วยเนื่องจากเมื่อสายมีความยาวมากๆจะเกิดค่าความต้านทานในสายขึ้น ดังนั้นจึงต้องทำการต่อค่าความต้านทานจากภายนอก

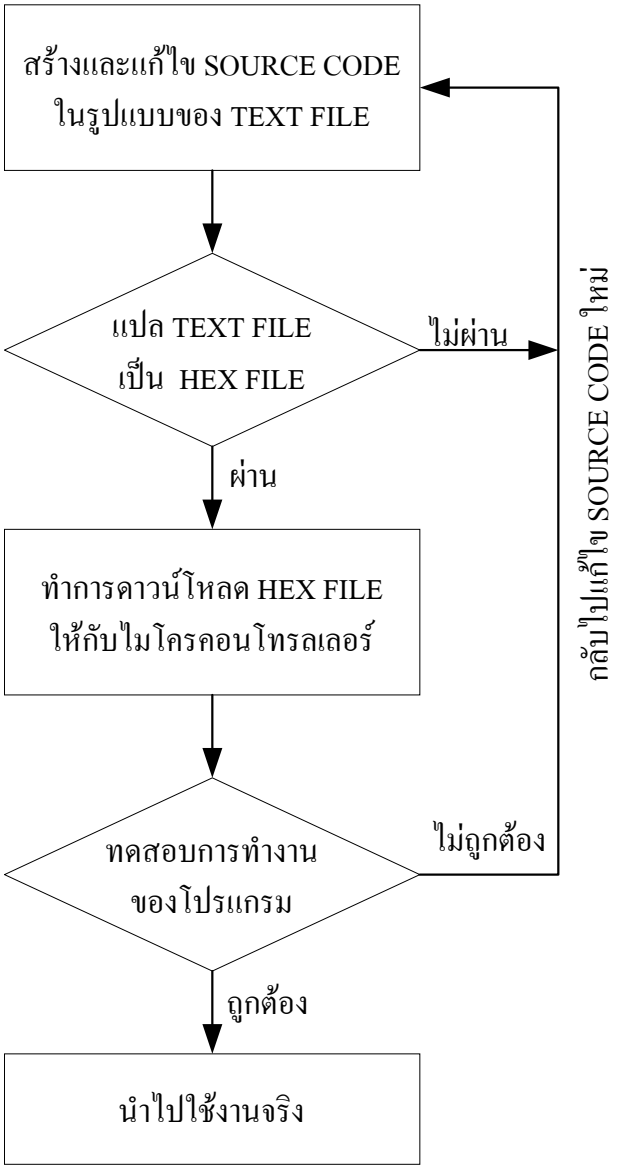
ไปชดเชยค่าความต้านทานของสายสัญญาณด้วย โดยเมื่อทำการ Short Jumper ตำแหน่ง RZ นี้ไว้ก็จะเป็นการต่อตัวต้านทานคร่อมระหว่าง RXA (RX+) และ RXB (RX-) ไว้ แต่ถ้าหากว่าต่อสายสัญญาณในระยะทางที่ไม่ไกลมากนัก ก็ให้ทำการ Open Jumper นี้ออกก็ได้

- **Jumper TL** เป็น Jumper ใช้สำหรับเลือกกำหนดการเชื่อมต่อ ตัวต้านทานสำหรับทำหน้าที่คงสถานะของสัญญาณ TXB (TX-) หรือ Fail Safe Resister เพื่อให้สัญญาณ TXB (TX-) มีสภาวะแน่นอนเมื่อไม่มีการส่งสัญญาณใดๆออกมาในสายเลย ซึ่งถ้าหากว่ามีการต่อสายสัญญาณระยะทางไกลๆหรือมีการต่อสายระยะทางไกล้ๆแต่ไม่ได้มีการส่งข้อมูลออกมาในสายตลอดเวลาแล้วควรที่จะทำการ Short Jumper นี้ไว้ด้วยเสมอ โดยเฉพาะอย่างยิ่งเมื่อใช้งานเป็นแบบ RS485 หรือใช้งานเป็นตัวอุปกรณ์ที่อยู่ในตำแหน่งต้นสายและปลายสายควรทำการ Short Jumper นี้ไว้เสมอ ส่วนอุปกรณ์ที่อยู่ในตำแหน่งอื่นๆที่มีระยะไม่ไกลจากจุดต้นสายและปลายสายมากนักก็อาจ Open Jumper นี้ออกก็ได้ แต่อย่างน้อยที่สุด ควรมีการ Short Jumper นี้ให้กับอุปกรณ์ที่ต่อร่วมอยู่ในสายสัญญาณจำนวน 1 จุดเสมอ
- **Jumper TH** เป็น Jumper ใช้สำหรับเลือกกำหนดการเชื่อมต่อ ตัวต้านทานสำหรับทำหน้าที่คงสถานะของสัญญาณ TXA (TX+) หรือ Fail Safe Resister เพื่อให้สัญญาณ TXA (TX+) มีสภาวะแน่นอนเมื่อไม่มีการส่งสัญญาณใดๆออกมาในสายเลย ซึ่งถ้าหากว่ามีการต่อสายสัญญาณระยะทางไกลๆหรือมีการต่อสายระยะทางไกล้ๆแต่ไม่ได้มีการส่งข้อมูลออกมาในสายตลอดเวลาแล้วควรที่จะทำการ Short Jumper นี้ไว้ด้วยเสมอ โดยเฉพาะอย่างยิ่งเมื่อใช้งานเป็นแบบ RS485 หรือใช้งานเป็นตัวอุปกรณ์ที่อยู่ในตำแหน่งต้นสายและปลายสายควรทำการ Short Jumper นี้ไว้เสมอ ส่วนอุปกรณ์ที่อยู่ในตำแหน่งอื่นๆที่มีระยะไม่ไกลจากจุดต้นสายและปลายสายมากนักก็อาจ Open Jumper นี้ออกก็ได้ แต่อย่างน้อยที่สุด ควรมีการ Short Jumper นี้ให้กับอุปกรณ์ที่ต่อร่วมอยู่ในสายสัญญาณจำนวน 1 จุดเสมอ
- **Jumper TZ** เป็น Jumper สำหรับเลือกกำหนดการต่อตัวต้านทาน TZ เพื่อชดเชย ค่าความต้านทานของสายสัญญาณ (Impedance) ทางด้านส่ง ซึ่งถ้าหากว่ามีการต่อสายสัญญาณในการรับส่งเป็นระยะทางไกลๆแล้วก็ควรทำการ Short Jumper นี้ไว้ด้วยเนื่องจากเมื่อสายมีความยาวมากๆจะเกิดค่าความต้านทานในสายขึ้น ดังนั้นจึงต้องทำการต่อค่าความต้านทานจากภายนอกไปชดเชยค่าความต้านทานของสายสัญญาณด้วย โดยเมื่อทำการ Short Jumper ตำแหน่ง TZ นี้ไว้ก็จะเป็นการต่อตัวต้านทานคร่อมระหว่าง TXA (TX+) และ TXB (TX-) ไว้ แต่ถ้าหากว่าต่อสายสัญญาณในระยะทางที่ไม่ไกลมากนัก ก็ให้ทำการ Open Jumper นี้ออกก็ได้

*****ข้อสังเกตุ***** จะเห็นได้ว่าวงจร Line Driver ทั้งแบบ RS422 และ RS485 นั้นจะมีความใกล้เคียงกันมาก แต่มีข้อแตกต่างอย่างหนึ่งที่เห็นได้ชัดเจนที่สุด คือ ถ้าเป็นแบบ RS422 จะไม่สามารถสั่งเปลี่ยน ทิศทางการรับส่งข้อมูลด้วยโปรแกรมได้ ซึ่งทิศทางการรับส่งจะกำหนดตายตัวจากวงจร แต่ถ้าเป็นแบบ RS485 นั้น จะสามารถสั่งควบคุมทิศทางการรับส่งจากโปรแกรมได้ว่าจะให้ทำหน้าที่เป็นฝ่ายรับ หรือฝ่ายส่ง อย่างใดอย่างหนึ่งได้ตามต้องการได้

การพัฒนาโปรแกรมของบอร์ด CP-JR51AC2

สำหรับการพัฒนาโปรแกรมนั้น บอร์ด CP-JR51AC2 จะออกแบบวงจรให้สามารถทำการพัฒนาโปรแกรมของบอร์ดได้เอง โดยไม่ต้องใช้เครื่องมือใดๆ นอกจากเครื่องคอมพิวเตอร์ PC และสายต่อพอร์ตสื่อสารอนุกรม RS232 สำหรับเชื่อมต่อสั่งงาน CPU จากเมนูคำสั่งต่างๆของโปรแกรม FLIP ซึ่งในการพัฒนาโปรแกรมนั้นจะมีลำดับขั้นตอนในการพัฒนาโปรแกรมดังแผนผังต่อไปนี้



แผนผัง แสดงขั้นตอนในการพัฒนาโปรแกรมของบอร์ด CP-JR51AC2

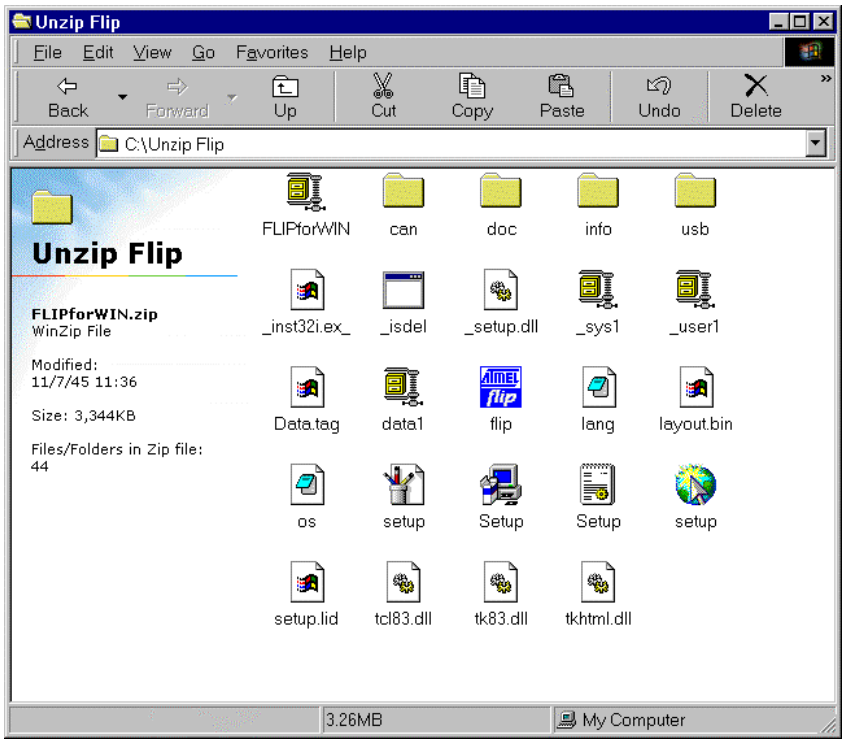
สำหรับในที่นี้จะขอกล่าวถึงเฉพาะขั้นตอนและวิธีการ Download Hex File ให้กับบอร์ดเท่านั้น ส่วนวิธีการเขียนโปรแกรมและการสั่งแปลคำสั่งให้ได้เป็น Hex File นั้น ขอให้ผู้ใช้ศึกษาจากข้อกำหนดของโปรแกรมแปลภาษาที่จะนำมาใช้ในการเขียนโปรแกรมเอง

การติดตั้งโปรแกรม FLIP สำหรับ Download โปรแกรม

สำหรับโปรแกรม FLIP นั้นสามารถทำการ Download จากเว็บไซต์ของ ATMEL มาใช้งานได้โดยไม่ต้องเสียค่าใช้จ่าย โดยในปัจจุบัน (เดือนธันวาคม 2545) จะเป็น Version 1.8.2 โดยลักษณะของไฟล์จะเป็นไฟล์ที่ถูกบีบอัดข้อมูลเป็น ZIP ไฟล์ไว้ มีขนาด 3.26Mbyte แต่อย่างไรก็ตามในกรณีที่ซื้อบอร์ดรุ่น CP-JR51AC2 ของบริษัท อีทีที ไปใช้งานนั้น โปรแกรมตัวนี้จะมีแถมให้ไปด้วยอยู่แล้วโดยจะบรรจุไว้ในแผ่น CD ROM ซึ่งจะแถมไปกับบอร์ด ผู้ใช้สามารถสั่งติดตั้งโปรแกรมโดยการสั่ง Run ไฟล์ Setup จากแผ่น CD ROM แล้วทำตามขั้นตอนต่างๆที่โปรแกรม Setup กำหนดไว้จนจบขั้นตอนการติดตั้ง

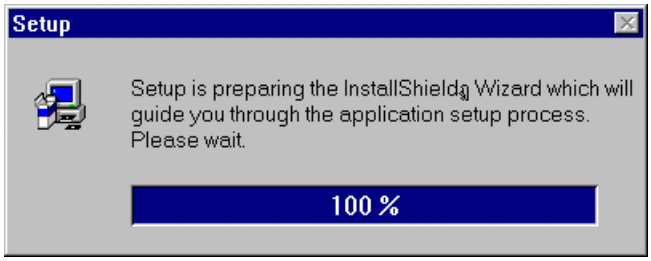
สำหรับในที่นี้จะขอแนะนำขั้นตอนการติดตั้งโปรแกรมแบบพอสังเขปเพื่อเป็นแนวทางสำหรับผู้ใช้ที่อาจยังไม่ค่อยชำนาญเกี่ยวกับการติดตั้งโปรแกรมภายใต้ระบบปฏิบัติการของ Windows ซึ่งในกรณีที่จะทำการติดตั้งโปรแกรมจากไฟล์ที่จัดเตรียมไว้ในแผ่น CD ROM ที่แถมไปกับบอร์ด CP-JR51AC2 นั้นให้ข้ามขั้นตอนของการ Unzip ไฟล์ไปได้เลย แต่สำหรับผู้ที่ทำการ Download ไฟล์จากเว็บไซต์เองนั้น อันดับแรกเมื่อทำการ Download ไฟล์มาได้เรียบร้อยแล้วจะต้องทำการแตกตัวไฟล์ โดยการ Unzip ไฟล์ที่ Download มาไว้ใน Harddisk เสียก่อน สมมุติว่าผู้ใช้ทำการ Download ไฟล์แล้วบันทึกเป็นไฟล์ชื่อ “FLIPforWIN.ZIP ไว้ให้ทำตามขั้นตอนดังนี้

1. สร้างไฟล์โฟล์เดอร์ไว้ใน Hard Disk โดยสามารถตั้งชื่อโฟล์เดอร์ได้ตามต้องการ โดยจากตัวอย่างจะสร้างโฟล์เดอร์ชื่อ Unzip Flip ซึ่งโฟล์เดอร์นี้จะใช้สำหรับเป็นที่แตกตัวของไฟล์ต่างๆที่ถูกบีบอัดไว้ใน FLIPforWIN.ZIP เท่านั้น เมื่อทำการติดตั้งเสร็จแล้วอาจลบทิ้งไปเลยในภายหลังก็ได้ จากนั้นจึงสั่ง Unzip ไฟล์ FLIPforWIN.ZIP เพื่อทำการแตกตัวไฟล์ต่างๆที่ถูกบีบอัดไว้ออกมาเก็บไว้ยังโฟล์เดอร์ที่สร้างเตรียมรอไว้แล้ว ซึ่งจะได้ผลดังรูป

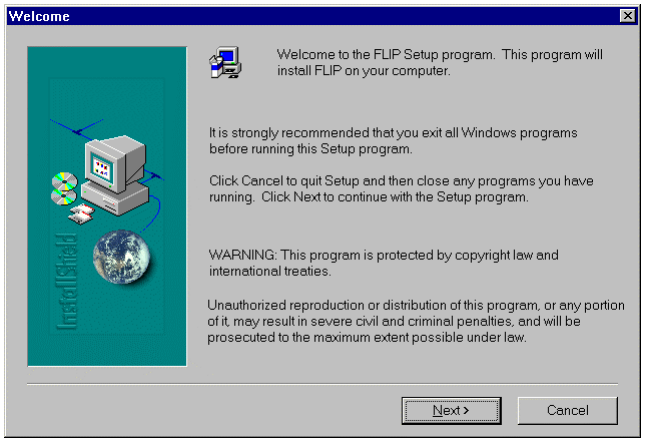


รูปแสดงไฟล์ต่างๆที่ได้จากการ Unzip

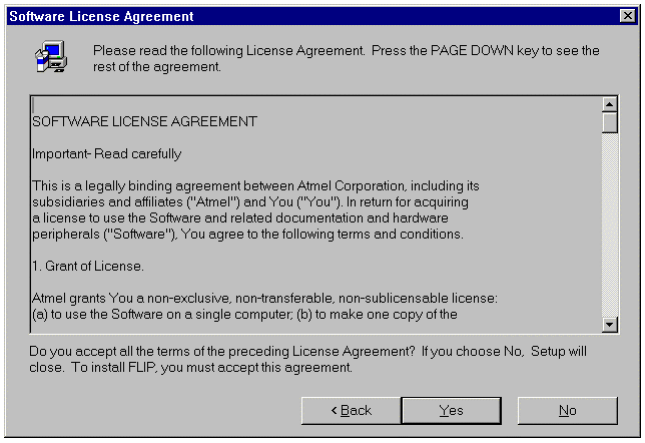
2. สั่ง Run ไฟล์สำหรับทำการ Install โปรแกรม FLIP โดยการ Double คลิกเมาส์ไปที่ไฟล์ชื่อ Setup ที่ทำการ Unzip ออกมาแล้วจากข้อ 1 ซึ่งโปรแกรม Setup จะเข้าสู่ขั้นตอนของการเตรียมการ Install โปรแกรมดังรูป



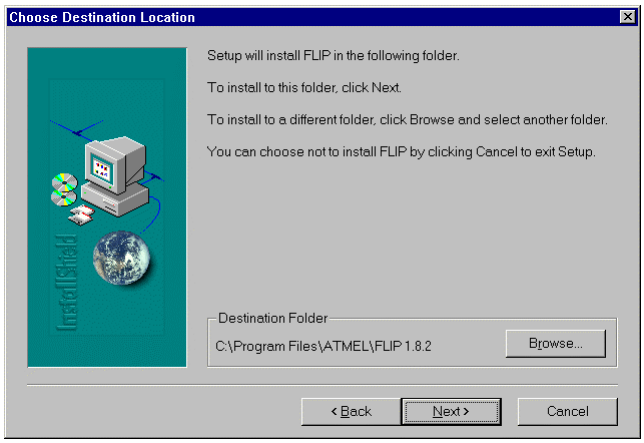
↓



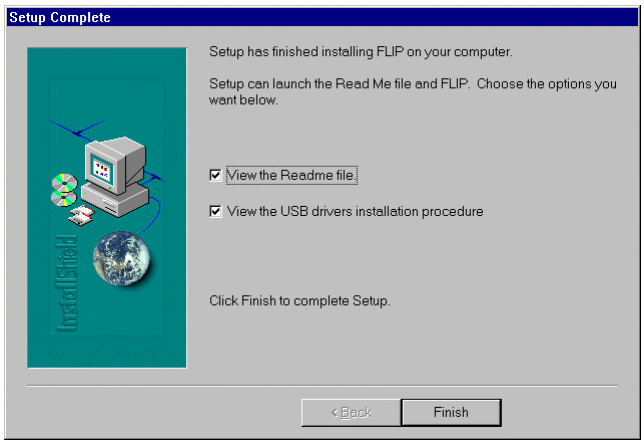
3. ให้เลือก **Next** เพื่อไปยังขั้นตอนต่อไปของการ Install โปรแกรม ซึ่งโปรแกรมการติดตั้งจะแสดงเงื่อนไขการใช้งานโปรแกรม FLIP ให้ทราบ ให้เลือก **YES** เพื่อยอมรับเงื่อนไขดังกล่าว เพื่อข้ามไปยังขั้นตอนต่อไปของการติดตั้งโปรแกรมซึ่งจะได้ผลดังรูป



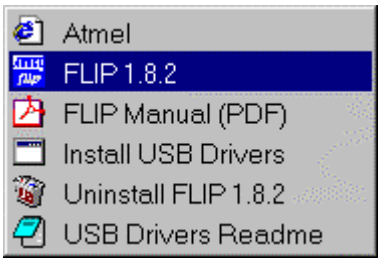
4. โปรแกรมจะให้กำหนดไฟล์โฟล์เดอร์สำหรับทำการติดตั้งโปรแกรมให้ โดยครั้งแรกจะกำหนดให้ทำการติดตั้งไว้ยัง “C:\Program Files\ATMEL\FLIP 1.8.2” ซึ่งถ้าต้องการติดตั้งโปรแกรมไว้ยังโฟล์เดอร์ดังกล่าวให้ทำการเลือก **NEXT** เพื่อไปยังขั้นตอนต่อไปของการติดตั้งโปรแกรมได้เลย แต่ถ้าต้องการเปลี่ยนแปลงก็ให้เลือก Browse เพื่อกำหนดตำแหน่งไฟล์โฟล์เดอร์ของการติดตั้งตามต้องการ แต่เพื่อความสะดวกขอให้เลือก NEXT เลย ดังรูป



ซึ่งหลังจากทำการคลิกเมาส์ที่ปุ่มคำสั่ง NEXT แล้วโปรแกรมจะเริ่มทำการติดตั้งโปรแกรมต่างๆทันที โดยในขั้นตอนนี้ให้รอจนกว่าโปรแกรมจะทำการติดตั้งเสร็จเรียบร้อยดังรูป



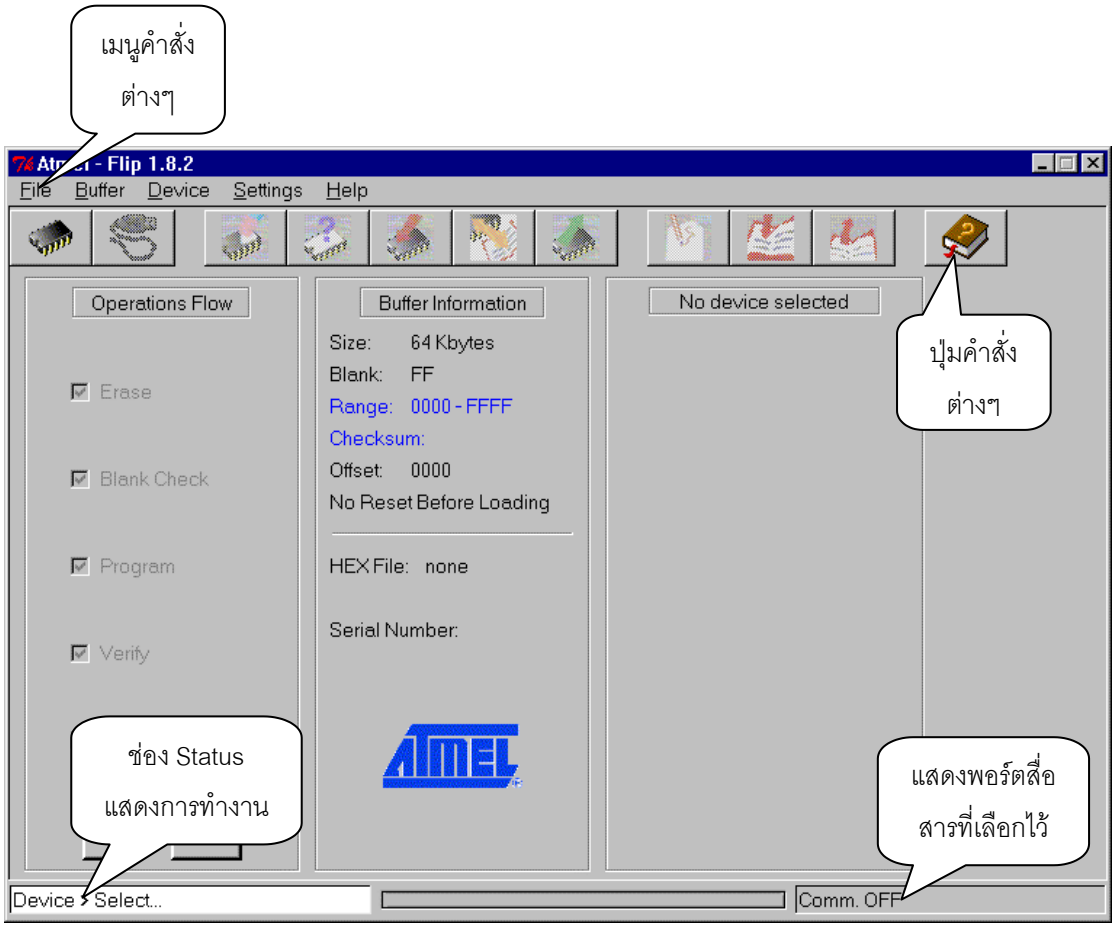
5. เมื่อโปรแกรมทำการติดตั้งเรียบร้อยแล้วให้เลือก Finish เพื่อจบการติดตั้งโปรแกรม ซึ่งหลังจากนี้ก็สามารถสั่ง Run โปรแกรม FLIP ได้ทันที โดยการเรียกจากเมนู Start →Programs → Atmel → FLIP 1.8.2 ซึ่งจะเกิดหน้าต่างเมนูสำหรับให้เลือก Run โปรแกรมต่างๆของ FLIP ที่ติดตั้งไว้แล้ว ให้ทำการเลือก FLIP 1.8.2 เพื่อเรียกใช้งานโปรแกรม FLIP ดังรูป



รูปแสดง หน้าต่างเมนูคำสั่งสำหรับเรียกใช้งานโปรแกรม FLIP

การใช้งานโปรแกรม FLIP (Flexible In-system Programmer) สำหรับDownload โปรแกรม

โปรแกรม FLIP (Flexible In-system Programmer) เป็นโปรแกรมสำหรับพัฒนาระบบของไมโครคอนโทรลเลอร์ของ ATMEL โดยสามารถใช้สนับสนุนการพัฒนาโปรแกรมของไมโครคอนโทรลเลอร์ ตระกูล MCS51 ในกลุ่มที่ใช้การพัฒนาแบบ ISP ซึ่งรวมถึงเบอร์ T89C51AC2 ด้วย โดยโปรแกรมจะทำงานภายใต้ระบบปฏิบัติการของ Windows9X/Me/NT/2000 และ Windows XP โดยสนับสนุนการเชื่อมต่อกับระบบฮาร์ดแวร์ที่ใช้การเชื่อมต่อแบบ RS232 หรือ CAN หรือ USB ซึ่งวิธีการเชื่อมต่อของโปรแกรม FLIP กับระบบฮาร์ดแวร์ของไมโครคอนโทรลเลอร์นั้น จะขึ้นอยู่กับความสามารถของตัวไมโครคอนโทรลเลอร์ที่จะนำมาทำการพัฒนาว่าสามารถใช้การติดต่อสื่อสารด้วยวิธีใดได้บ้าง แต่สำหรับไมโครคอนโทรลเลอร์เบอร์ T89C51AC2 นั้นจะสามารถใช้การเชื่อมต่อผ่านทางพอร์ตอนุกรม RS232 เท่านั้น ไม่สามารถเชื่อมต่อผ่านระบบการสื่อสารของ CAN หรือ USB ได้ โดยโปรแกรม FLIP จะใช้สำหรับ Download ข้อมูลให้กับหน่วยความจำของไมโครคอนโทรลเลอร์ที่ทำงานใน Monitor Mode เพื่อให้ผู้ใช้สั่งจัดการกับหน่วยความจำภายในตัว CPU ไม่ว่าจะเป็นการ สั่งลบข้อมูล(Erase) สั่งตรวจสอบข้อมูลในหน่วยความจำ(Blank Check) สั่งโปรแกรมข้อมูลให้กับหน่วยความจำโปรแกรมของ CPU (Program) สั่งเปรียบเทียบข้อมูลจาก Buffer กับหน่วยความจำในตัว CPU (Verify) หรือสั่งอ่านข้อมูลจากหน่วยความจำของ CPU (Read) เป็นต้น โดยลักษณะของโปรแกรม FLIP เมื่อสั่ง Run เป็นดังรูป



รูปแสดง ลักษณะหน้าต่างโปรแกรมของ FLIP Version 1.8.2

ซึ่งเมื่อต้องการให้โปรแกรม FLIP ติดต่อกับ CPU ใน Monitor Mode นั้น จะต้องสั่ง Reset ให้ CPU เข้าทำงานใน Monitor Mode ก่อนด้วยจึงจะสามารถสั่งงาน CPU ผ่านทางโปรแกรม FLIP ได้ ซึ่งหลักการสำหรับ Reset ให้ CPU เข้าทำงานใน Monitor Mode จะต้องกำหนดให้ขาสัญญาณ PSEN มีสภาวะเป็น “0” ในขณะที่ CPU หลุดพ้นจากสภาวะของการ Reset ซึ่งตามปรกติแล้วหลังการ Reset ทุกครั้ง CPU เบอร์ T89C51AC2 จะตรวจสอบสภาวะของขาสัญญาณ PSEN ว่าเป็น “0” หรือไม่ถ้าไม่ใช่ก็จะทำงานในโหมดการทำงานปรกติแต่ถ้าใช่ก็จะตรวจสอบสถานะของสัญญาณอื่นๆที่เกี่ยวข้องกับการทำงานใน Monitor Mode ถ้าเงื่อนไขอื่นๆถูกต้องก็จะเข้าทำงานใน Monitor Mode ทันที สำหรับบอร์ดรุ่น CP-JR51AC2 นั้น การที่จะสั่ง Reset ให้ CPU เบอร์ T89C51AC2 ของ ATMEL เข้าทำงานใน Monitor Mode นั้นจะต้องทำตามลำดับขั้นตอนดังต่อไปนี้

1. กดสวิตช์ PSEN ค้างไว้เพื่อกำหนดสถานะขาสัญญาณ PSEN ให้เป็น “0”
2. กดสวิตช์ RESET เพื่อส่งสัญญาณ RESET ให้กับ CPU โดยสวิตช์ PSEN ต้องกดค้างอยู่
3. ปล่อยสวิตช์ RESET เพื่อปล่อยให้ CPU พ้นจากสภาวะการ Reset (สวิตช์ PSEN ยังกดค้างอยู่)
4. ปล่อยสวิตช์ PSEN เป็นลำดับสุดท้าย

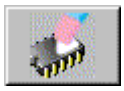
โดยเมื่อสั่ง Reset ให้บอร์ด CP-JR51AC2 เข้าทำงานใน Monitor Mode ตามขั้นตอนดังกล่าวข้างต้นแล้ว หลังจากกำหนดรูปแบบการติดต่อสื่อสารในโปรแกรมถูกต้องตามความเป็นจริงแล้ว จะสามารถทำการสั่ง Download ข้อมูล HEX File จากเครื่องคอมพิวเตอร์ PC ให้กับหน่วยความจำ FLASH ของ CPU ได้ทันที ซึ่งในขั้นตอนนี้ผู้ใช้จะสามารถสั่งจัดการหน่วยความจำของ CPU ได้ตามต้องการ ไม่ว่าจะเป็นการสั่งลบข้อมูลในหน่วยความจำ หรือ สั่งเขียนข้อมูลหรือโปรแกรมข้อมูลให้กับหน่วยความจำ ซึ่งถ้าหากไฟล์ที่นำมาโปรแกรมให้กับหน่วยความจำของ CPU นั้น มีความสมบูรณ์ถูกต้อง ก็สามารถนำบอร์ดไปใช้งานจริงได้ทันที

ซึ่งจะเห็นได้ว่า โปรแกรม FLIP นี้จะมีคำสั่งอยู่มากมายหลายคำสั่ง เพื่ออำนวยความสะดวกต่อผู้ใช้ โดยผู้ใช้สามารถเลือกสั่งงานคำสั่งต่างๆได้ตามต้องการ ซึ่งการเรียกใช้งานคำสั่งต่างๆนั้นอาจเลือกจากเมนูคำสั่งโดยตรง หรืออาจเลือกจากปุ่มคำสั่งที่มีให้ก็ได้ เนื่องจากคำสั่งที่มีความจำเป็นต้องใช้งานบ่อยๆนั้นถ้าใช้วิธีการสั่งงานจากเมนูคำสั่ง อาจทำให้เกิดความไม่สะดวกต่อการใช้งานเนื่องจากต้องผ่านขั้นตอนหลายขั้นตอน ซึ่งโปรแกรม FLIP เองก็ได้สร้างปุ่มคำสั่งพิเศษขึ้นมาให้เลือกใช้งาน นอกจากนั้นแล้วยังสามารถเรียกใช้งานคำสั่งด้วยการใช้คีย์ลัด (Short Key) ได้ด้วย เช่น ถ้าต้องการจะกำหนดเบอร์ของ CPU (Device Select) นั้น แทนที่จะต้องเลือกจากคำสั่งในเมนูคำสั่ง Device → Select ก็อาจใช้วิธีการกดคีย์ “F2” หรือเลือกคลิกเมาส์ที่ปุ่มคำสั่ง Select Device เพียงปุ่มเดียวก็ได้เช่นกัน โดยวิธีการใช้งานต่างๆโดยละเอียดนั้นสามารถดูได้จาก คู่มือของโปรแกรมเอง หรืออาจศึกษาจาก Help ในเมนูคำสั่ง Help ก็ได้ ซึ่งในที่นี้จะขอกล่าวและอธิบายถึงคำสั่งที่มีความจำเป็นต่อการใช้งานเพื่อเป็นแนวทางให้ทราบดังต่อไปนี้

การทำงานของปุ่มคำสั่งต่าง ๆในโปรแกรม FLIP

 **Select Device** ใช้สำหรับเลือกกำหนดเบอร์ CPU ที่จะใช้งานร่วมกับโปรแกรม FLIP

 **Select Communication** ใช้สำหรับกำหนดรูปแบบการเชื่อมต่อโปรแกรม FLIP กับ CPU

 **Erase Device** ใช้สำหรับสั่งลบข้อมูลจากหน่วยความจำของ CPU

 **Blank Check Device** ใช้สำหรับตรวจสอบว่าข้อมูลในหน่วยความจำของ CPU ว่างอยู่หรือไม่

 **Program Device** ใช้สำหรับสั่ง Program ข้อมูลให้กับหน่วยความจำของ CPU

 **Verify Device** ใช้สำหรับสั่งเปรียบเทียบข้อมูลในหน่วยความจำของ CPU กับข้อมูลที่อยู่ใน Buffer ของโปรแกรม FLIP ในเครื่องคอมพิวเตอร์ PC ว่ามีค่าตรงกันทั้งหมดหรือไม่

 **Read Device** ใช้สำหรับสั่งอ่านข้อมูลจากหน่วยความจำของ CPU ไปเก็บไว้ใน Buffer

 **Edit Buffer** ใช้สำหรับสั่งแก้ไขข้อมูลใน Buffer

 **Load HEX File** ใช้สำหรับสั่งโหลดข้อมูลแบบ HEX File มาเก็บไว้ใน Buffer

 **Save HEX File** ใช้สำหรับสั่งบันทึกข้อมูลใน Buffer เป็น HEX File เก็บไว้

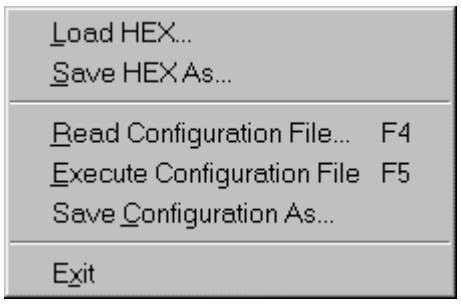
 **Help** ใช้สำหรับเรียกไฟล์ Help สำหรับดูรายละเอียดการใช้งานโปรแกรม

*****หมายเหตุ***** คำว่า Buffer ในที่นี้ หมายถึง พื้นที่สำหรับใช้พักข้อมูลชั่วคราวของโปรแกรม FLIP ซึ่งโปรแกรม FLIP จะใช้พื้นที่ส่วนนี้ในการพักข้อมูลและใช้เป็นตัวกลางในการทำงานของโปรแกรม เช่นเมื่อสั่ง Load HEX File ข้อมูลใน HEX File ก็จะถูกพักรอไว้ใน Buffer เมื่อสั่ง Program Device โปรแกรม FLIP ก็จะนำข้อมูลใน Buffer นี้เขียนเข้าไปยังหน่วยความจำของ CPU เป็นต้น

การทำงานของคำสั่งต่าง ๆในเมนูคำสั่ง

สำหรับการเรียกใช้งานคำสั่งต่างๆของโปรแกรม FLIP อีกวิธีหนึ่งคือการเรียกใช้งานผ่านทางเมนูคำสั่ง โดยเมื่อต้องการใช้งานคำสั่งใดก็ให้ทำการคลิกเมาส์ไปยังเมนูคำสั่งนั้นแล้วเลื่อนเมาส์ให้แถบสว่างไปปรากฏอยู่เหนือคำสั่งที่ต้องการแล้วคลิกเมาส์โปรแกรม FLIP ก็จะทำงานตามคำสั่งนั้นๆทันที ซึ่งโปรแกรม FLIP จะจัดกลุ่มหรือหมวดหมู่ของเมนูคำสั่งต่างๆไว้ด้วยกัน 5 กลุ่มดังนี้คือ

เมนู File ใช้สำหรับสั่งจัดการเกี่ยวกับไฟล์ ไม่ว่าจะเป็นการสั่งเปิด HEX ไฟล์ การสั่งบันทึก HEX ไฟล์ การสั่งอ่าน Configuration File การสั่งบันทึก Configuration File หรือการปิดโปรแกรมเป็นต้น โดยลักษณะเมนูคำสั่งของ File จะเป็นดังรูป

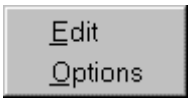


- **Load HEX** ใช้สำหรับสั่งอ่านข้อมูลแบบ Intel HEX มาพักรอไว้ใน Buffer ของโปรแกรม FLIP เพื่อรอการสั่ง Program Device
- **Save HEX As..** ใช้สำหรับสั่งบันทึกข้อมูลใน Buffer เก็บเป็น File แบบ Intel HEX ไว้
- **Read Configuration File** ใช้สำหรับสั่งอ่าน Configuration File เพื่อกำหนดคุณสมบัติการทำงานของโปรแกรม FLIP เหมือนกับที่กำหนดไว้ใน Configuration File ซึ่งวิธีการนี้จะช่วยอำนวยความสะดวกต่อการพัฒนาโปรแกรมมาก ในกรณีที่มีการเรียกใช้งานโปรแกรม FLIP เพื่อพัฒนาโปรแกรมให้กับบอร์ด ซึ่งในบางครั้งอาจจำเป็นต้องกระทำซ้ำๆกันหลายครั้งกว่าจะใช้งานได้ ซึ่งการกระทำดังกล่าวตามปรกติแล้วมักจะใช้เงื่อนไขหรือข้อกำหนดต่างๆที่เหมือนๆกันด้วย ไม่ว่าจะเป็น การกำหนดเบอร์ CPU การกำหนดรูปแบบการติดต่อสื่อสาร หรือชื่อ HEX ไฟล์ ที่จะนำมาใช้เป็นต้น ซึ่งโปรแกรม FLIP สามารถให้ผู้ใช้ทำการบันทึกคุณสมบัติต่างๆเหล่านี้ไว้ในรูปแบบของ Configuration File ซึ่งเมื่อทำการเปิดโปรแกรม FLIP ขึ้นมาใช้งานในครั้งต่อๆไปก็ไม่จำเป็นต้องเสียเวลาไปกำหนดตัวเลือกหรือเงื่อนไขต่างๆให้กับโปรแกรมให้ยุ่งยาก สามารถสั่งอ่าน Configuration File ที่สั่งบันทึกหรือสร้างไว้ก่อนหน้านั้นขึ้นมา ซึ่งโปรแกรม FLIP จะทำการกำหนดคุณสมบัติและเงื่อนไขการทำงานต่างๆของโปรแกรมให้เหมือนกับที่บันทึกไว้ใน Configuration File ทั้งหมดในทันที ซึ่งจะช่วยประหยัดเวลาและลดความยุ่งยากไปได้มากเลยทีเดียว
- **Execute Configuration File** ใช้สำหรับสั่งให้โปรแกรม FLIP ทำงานตามคำสั่งที่บันทึกไว้แล้วใน Configuration File ที่ถูกสั่งอ่านออกมาครั้งสุดท้าย(ปัจจุบัน)
- **Save Configuration As…** ใช้สำหรับสั่งบันทึกค่าตัวเลือกและคุณสมบัติต่างๆที่กำหนดให้กับโปรแกรม FLIP ไว้ในปัจจุบัน ไม่ว่าจะเป็นการกำหนดเบอร์ CPU การกำหนดการสื่อสาร หรือการ

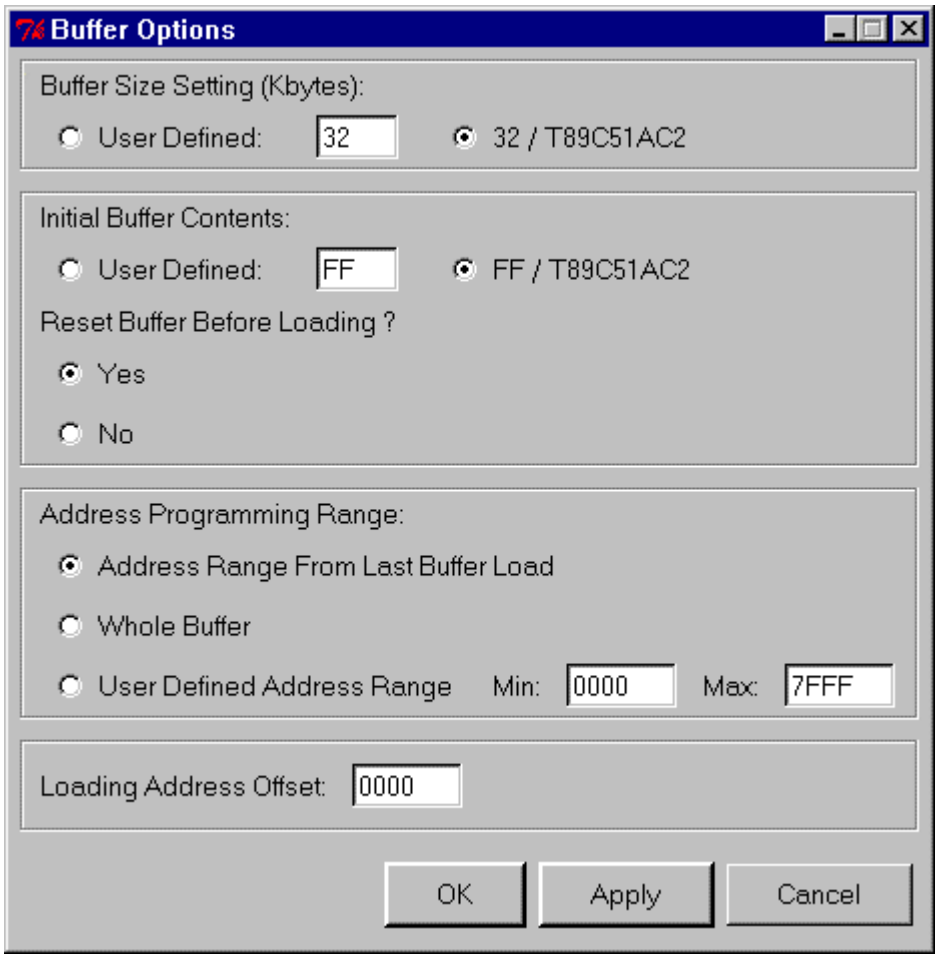
สั่งเปิดไฟล์ HEX สำหรับนำมาโปรแกรมให้กับ CPU เป็นต้น โดยคุณสมบัติต่างๆทั้งหมดจะถูกบันทึกเป็น Configuration File ไว้ให้ตามชื่อที่กำหนดตอนสั่งบันทึก

- EXIT ใช้สำหรับปิดโปรแกรม FLIP หรือจบการทำงานของโปรแกรม FLIP

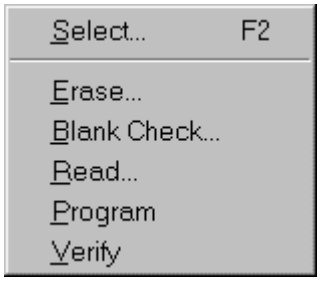
เมนู Buffer ใช้สำหรับสั่งจัดการเกี่ยวกับ Buffer



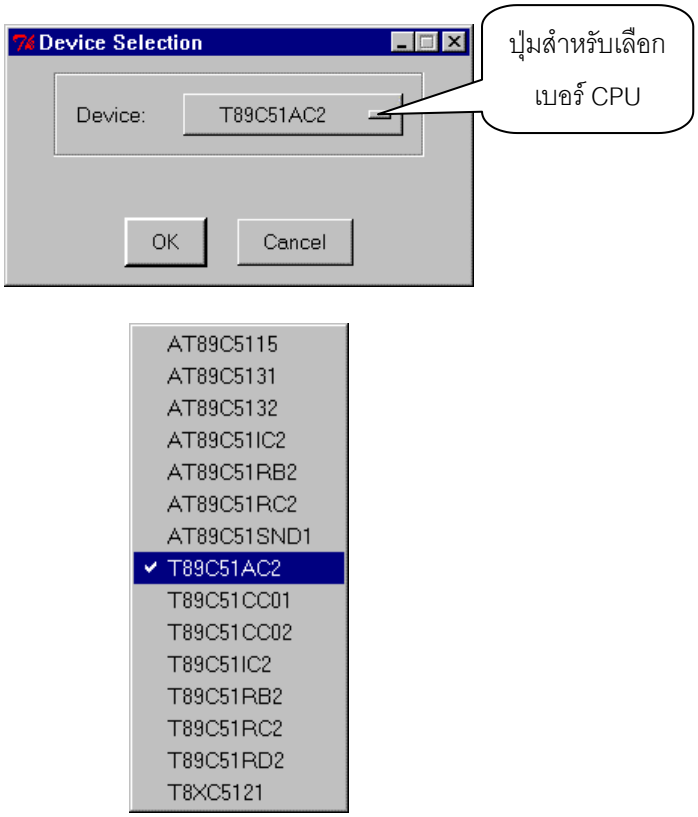
- Edit ใช้สำหรับสั่งแก้ไขข้อมูลใน Buffer ซึ่งเมื่อเรียกใช้คำสั่งนี้จะปรากฏหน้าต่างการทำงานของ Buffer พร้อมทั้งการแสดงค่าของข้อมูลในตำแหน่งแอดเดรสต่างๆใน Buffer ให้ทราบ ซึ่งผู้ใช้สามารถเข้าไปทำการแก้ไขค่าข้อมูลในตำแหน่งแอดเดรสต่างๆได้ตามต้องการ
- Options ใช้สำหรับกำหนดคุณสมบัติของ Buffer ซึ่งเมื่อเรียกใช้งานคำสั่งนี้จะปรากฏหน้าต่างสำหรับให้กำหนดค่า Option ต่างๆของ Buffer ซึ่งในกรณีที่ใช้งานกับบอร์ด CP-JR51AC2 นั้นขอแนะนำให้ทำการกำหนดตัวเลือกต่างๆดังรูป



เมนู Device ใช้สำหรับสั่งจัดการกับ CPU ไม่ว่าจะเป็นการกำหนดหรือเลือกเบอร์ CPU รวมทั้งการสั่งจัดการหน่วยความจำของ CPU ไม่ว่าจะเป็นการสั่งลบ (Erase) สั่งตรวจสอบข้อมูลในตัว CPU (Blank Check) สั่งอ่านข้อมูลภายในหน่วยความจำของ CPU มาไว้ใน Buffer (Read) สั่งเขียนข้อมูลจาก Buffer ให้กับหน่วยความจำภายในตัว CPU (Program) หรือสั่งเปรียบเทียบข้อมูลในตัว CPU กับข้อมูลใน Buffer (Verify) เป็นต้น



- **Select** ใช้สำหรับสั่งเลือกกำหนดเบอร์ CPU ที่จะนำมาใช้กับโปรแกรม FLIP ซึ่งเมื่อเรียกใช้คำสั่งนี้จะปรากฏหน้าต่างสำหรับเลือกกำหนดเบอร์ของ CPU ให้เห็นพร้อมกับแสดงเบอร์ของ CPU ที่กำหนดไว้แล้วในช่อง Device ซึ่งถ้าเบอร์ที่ต้องการถูกต้องแล้วให้เลือก OK แต่ถ้ายังไม่ตรงกับความต้องการให้คลิกเมาส์ที่ปุ่มเบอร์ของ CPU ที่แสดงไว้ ซึ่งโปรแกรมจะแสดงเมนูสำหรับเลือกเบอร์ขึ้นมาให้เลือก ซึ่งในขั้นตอนนี้ให้ทำการเลื่อนเมาส์ให้แถบสว่างเลื่อนไปยังเบอร์ CPU ที่ต้องการแล้วคลิกเมาส์เพื่อเลือกเบอร์ดังรูป

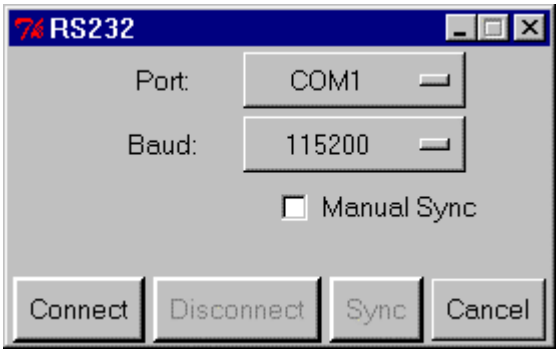


รูปแสดง การเลือกกำหนดเบอร์ CPU สำหรับใช้กับบอร์ด CP-JR51AC2

- **Erase** ใช้สำหรับสั่งลบข้อมูลจากหน่วยความจำของ CPU
- **Blank Check** ใช้สำหรับสั่งตรวจสอบหน่วยความจำโปรแกรมของ CPU ว่ามีข้อมูลอยู่หรือไม่
- **Read** ใช้สำหรับสั่งอ่านข้อมูลจากหน่วยความจำของ CPU มาพักรอไว้ใน Buffer
- **Program** ใช้สำหรับสั่งโปรแกรมข้อมูลใน Buffer ให้กับหน่วยความจำของ CPU
- **Verify** ใช้สำหรับสั่งเปรียบเทียบข้อมูลใน Buffer และหน่วยความจำของ CPU ว่าตรงกันหรือไม่

เมนู Setting ใช้สำหรับกำหนดรูปแบบการเชื่อมต่อของโปรแกรม FLIP กับ CPU

- **Communication** ใช้สำหรับกำหนดรูปแบบการสื่อสารของโปรแกรม FLIP กับ CPU ซึ่งในกรณีที่ใช้กับบอร์ด CP-JR51AC2 นั้นจะสามารถเลือกกำหนดรูปแบบการสื่อสารได้เฉพาะแบบ RS232 เท่านั้นไม่สามารถเลือกแบบอื่นๆได้ โดยในส่วนของ Port ให้เลือกกำหนดตามความเป็นจริง ว่าต่อสายสัญญาณจากบอร์ด CP-JR51AC2 เข้ากับพอร์ตอนุกรมใดของเครื่องคอมพิวเตอร์ PC ส่วนค่า Baud ขอแนะนำให้เลือกกำหนดเป็น 115200 จะดีที่สุด ซึ่งในกรณีที่เลือกค่านี้แล้วเกิดปัญหาในการสื่อสารขึ้นอาจเปลี่ยนเป็นค่าอื่นๆที่เหมาะสม เช่น 19200 หรือ 9600 เป็นต้น



- **Command Window** ใช้สำหรับสั่งเปิดหน้าต่างการแสดงผลการทำงานของคำสั่งต่างๆที่สั่งงานผ่านเมนูคำสั่งหรือปุ่มคำสั่งต่างๆ ซึ่งไม่มีความจำเป็นต้องใช้งาน

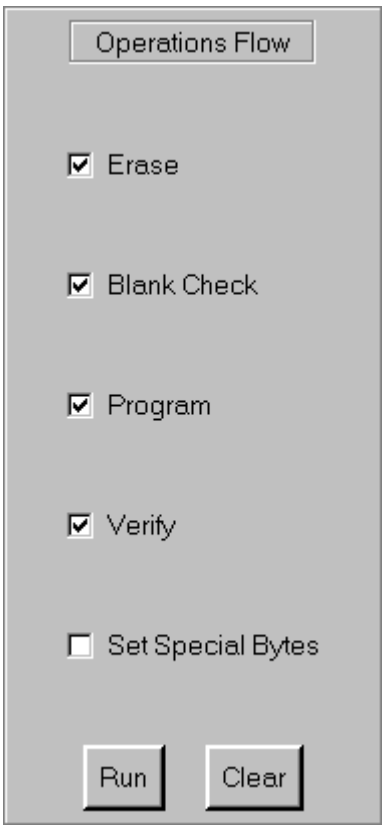
เมนู Help ใช้สำหรับเรียก Help File เพื่อดูคำอธิบายการใช้งานโปรแกรม

- Contents เป็นการเรียกดูรายละเอียดการใช้งานโปรแกรม FLIP โดยสามารถเลือกหัวข้อที่ต้องการจะดูรายละเอียดต่างๆได้ตามต้องการ ซึ่งเมื่อเลือกใช้คำสั่งนี้จะเกิดหน้าต่าง Help พร้อมกับแสดงหัวข้อต่างๆให้เลือกตามต้องการ
- About Flip ใช้สำหรับแสดง รุ่น หรือ Version ของโปรแกรม FLIP
- Third Party Vendor ใช้สำหรับแสดงรายชื่อบริษัทที่ร่วมกันพัฒนา Driver เพื่อใช้งานร่วมกับโปรแกรม FLIP

การสั่งงานโปรแกรม FLIP แบบอัตโนมัติ

สำหรับในกรณีที่ต้องการสั่งให้โปรแกรม FLIP จัดการโปรแกรมข้อมูลให้กับหน่วยความจำของ CPU นั้น ถ้าใช้วิธีการสั่งงานทีละคำสั่งนั้น อาจเกิดความล่าช้ามาก และต้องเสียเวลาคลิกเมาส์เพื่อสั่งงานแต่ละคำสั่งหลายๆครั้ง ซึ่งโปรแกรม FLIP ได้ออกแบบคำสั่ง Operation Flow เพื่อให้สามารถเลือกกำหนดการทำงานของคำสั่งต่างๆตามลำดับขั้นและขั้นตอน ซึ่งตามปรกติแล้วเมื่อจะสั่งโปรแกรมข้อมูล ให้กับ CPU จะต้องสั่งงาน CPU เป็นลำดับขั้นคือ

- Erase เพื่อลบข้อมูลเดิมที่อยู่ในตัว CPU ออกเสียก่อน
- Blank Check เพื่อตรวจสอบค่าข้อมูลในหน่วยความจำว่าลบหมดแล้วหรือยัง
- Program เพื่อเขียนข้อมูลจาก Buffer ไปยังหน่วยความจำของ CPU
- Verify เพื่อตรวจสอบดูว่าข้อมูลที่เขียนให้กับหน่วยความจำของ CPU นั้นถูกต้องเหมือนใน Buffer หรือไม่

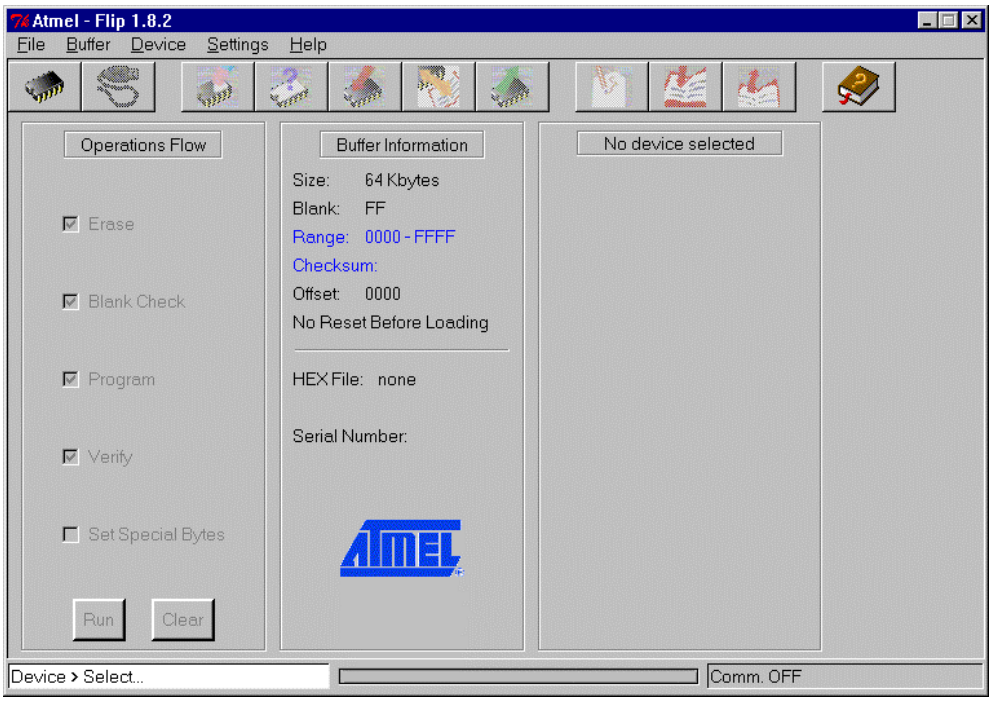


แสดง ลักษณะของหน้าต่าง Operations Flow สำหรับสั่งงานโปรแกรม FLIP แบบต่อเนื่อง

การ SETUP โปรแกรม FLIP เพื่อใช้งานกับบอร์ด CP-JR51AC2

เมื่อต้องการจะสั่ง Download ข้อมูลแบบ HEX File ให้กับ CPU เบอร์ T89C51AC2 ซึ่งอยู่ในบอร์ดรุ่น CP-JR51AC2 นั้น ก่อนอื่นจะต้องเตรียม HEX File ที่ต้องการ Download ให้กับหน่วยความจำของ CPU ให้เรียบร้อยเสียก่อน ซึ่งจะไม่ขอกล่าวถึงการสร้าง HEX File ในที่นี้ จากนั้นจึงทำตามขั้นตอนดังต่อไปนี้

1. ทำการต่อสายสัญญาณ RS232 ระหว่างบอร์ด CP-JR51AC2 (ขั้ว 4PIN RS232) กับพอร์ตสื่อสารอนุกรมของเครื่องคอมพิวเตอร์ PC (พอร์ตอนุกรม หรือ Comport เช่น Com1 หรือ Com2)
2. จ่ายไฟเลี้ยงวงจรให้กับบอร์ด CP-JR51AC2 ซึ่งในขณะนี้จะสังเกตุเห็นหลอดแสดงผล LED สีแดงของ “PWR” ติดสว่างให้เห็นเพื่อแสดงว่ามีไฟเลี้ยงวงจรของบอร์ดแล้ว
3. ทำการเรียกใช้งานโปรแกรม FLIP ซึ่งในครั้งแรกเมื่อยังไม่กำหนดตัวเลือกใดๆให้กับโปรแกรมจะเป็นดังรูป

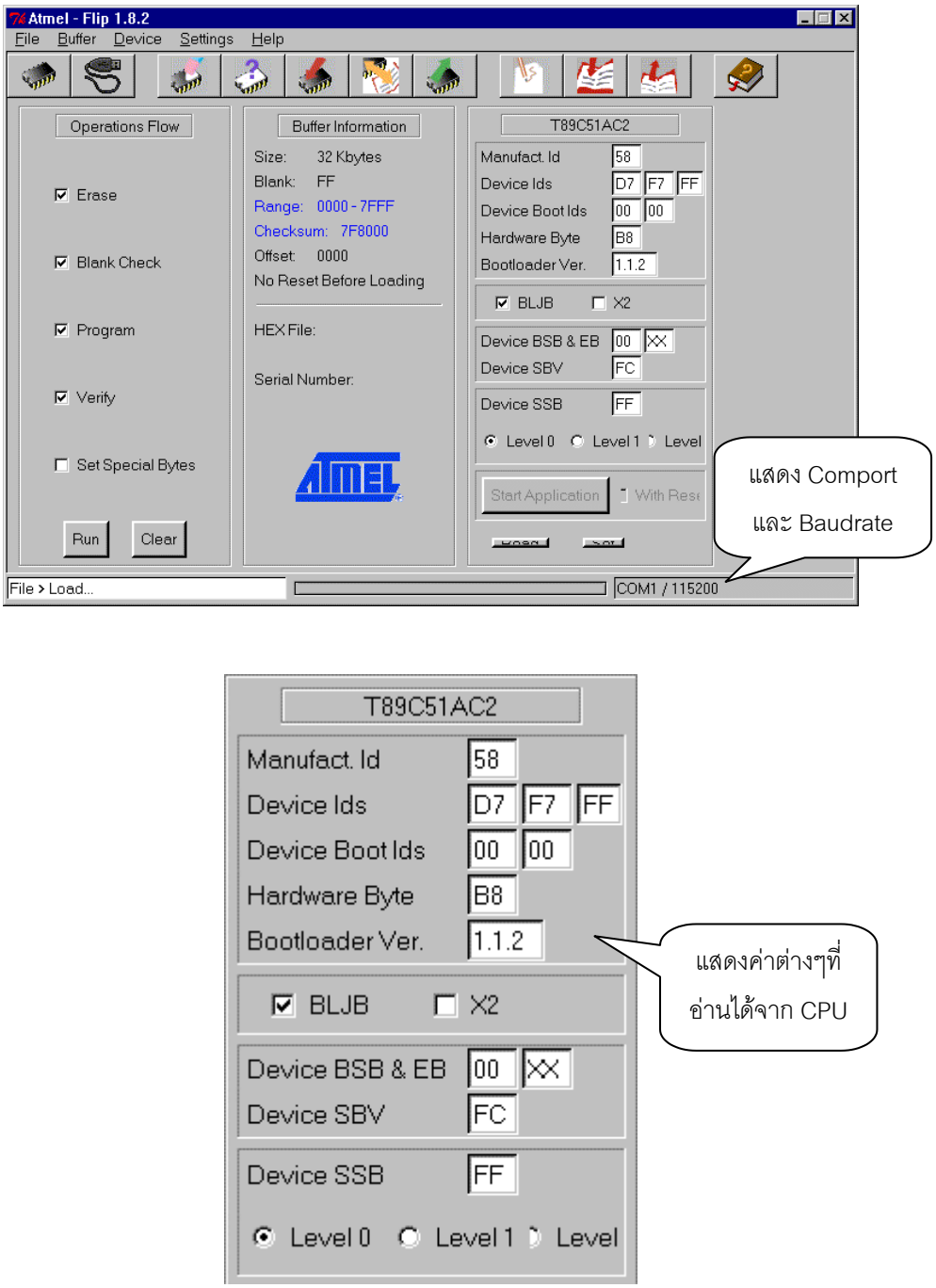


แสดงลักษณะของโปรแกรม FLIP (Version 1.8.2) เมื่อเริ่มต้นเปิดโปรแกรมในครั้งแรก

4. ทำการเลือกกำหนดเบอร์ CPU เป็น T89C51AC2
5. ทำการรีเซ็ตบอร์ด CP-JR51AC2 เพื่อให้ CPU เข้าทำงานใน Monitor Mode ดังนี้

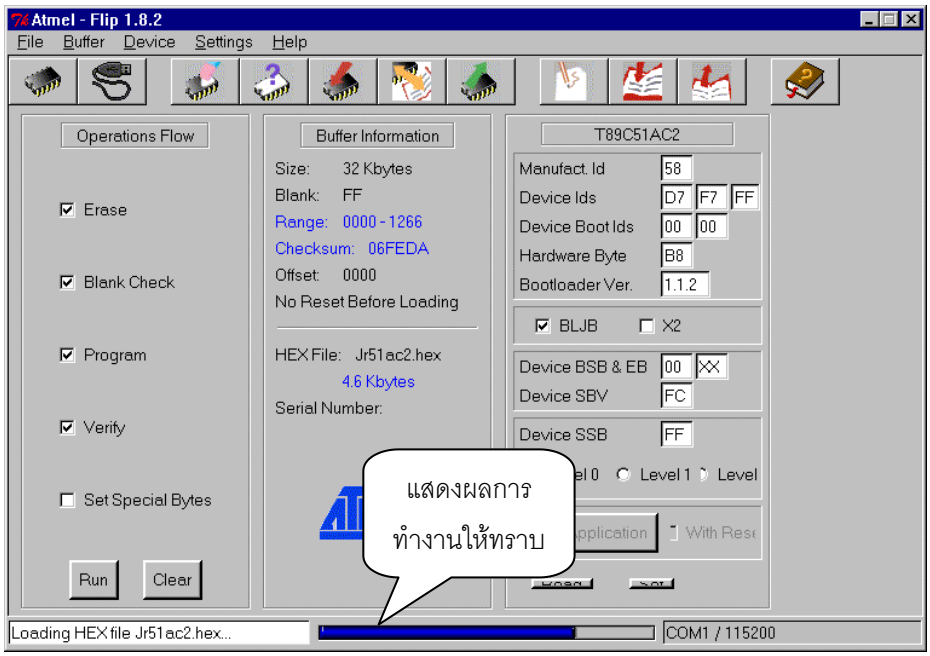
   - กดสวิตช์ PSEN ค้างไว้
   - กดสวิตช์ RESET โดย PSEN ยังกดค้างอยู่
   - ปล่อยสวิตช์ RESET โดย PSEN ยังกดค้างอยู่
   - ปล่อยสวิตช์ PSEN เป็นลำดับสุดท้าย

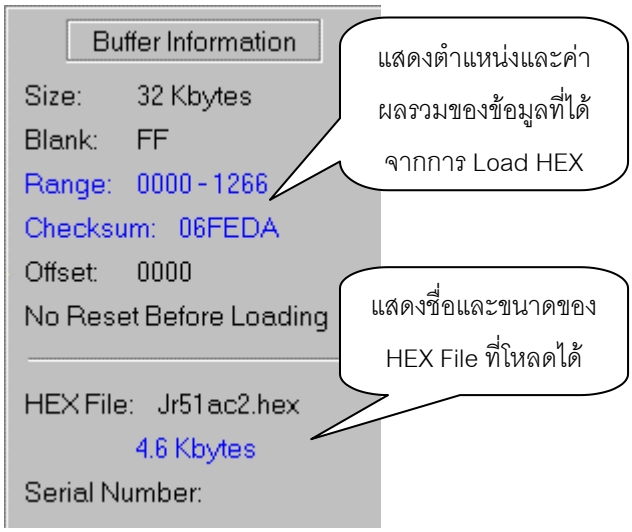
6. กำหนดการสื่อสารของโปรแกรม FLIP กับบอร์ด โดยเลือก Setting → Communication → RS232 → กำหนด Comport ตามที่ต่อสายไว้ (เช่น Com1) → กำหนด Baudrate เป็น 115200 → Connect ซึ่งถ้าทุกอย่างถูกต้อง โปรแกรมจะเริ่มต้นทำการติดต่อสื่อสารกับ CPU และสั่งอ่านค่าพารามิเตอร์ต่างๆจากตัว CPU แล้วนำมาแสดงผลให้ทราบทางหน้าจอของโปรแกรม FLIP ซึ่งเมื่อเสร็จเรียบร้อยจะได้ผลดังรูป



รูปแสดง ลักษณะการแสดงผลของโปรแกรม FLIP เมื่อติดต่อสื่อสารกับ CPU ได้

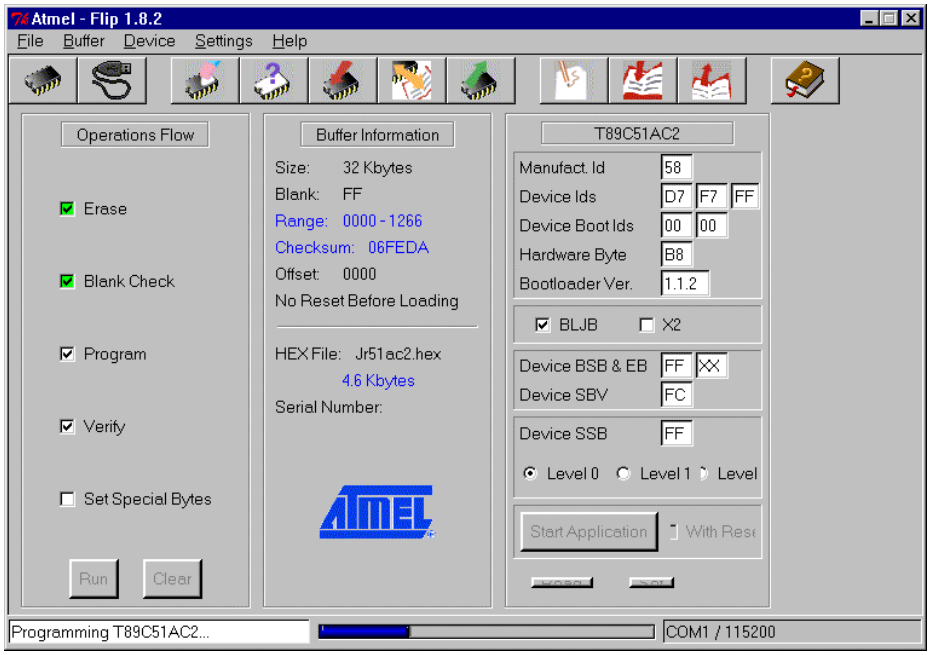
7. ทำการโหลดข้อมูล HEX File มารอไว้ยัง Buffer ของโปรแกรม โดยเลือก File → Load HEX → กำหนดชื่อและที่อยู่ของ HEX File เพื่อสั่งโหลด HEX File มารอไว้ใน Buffer



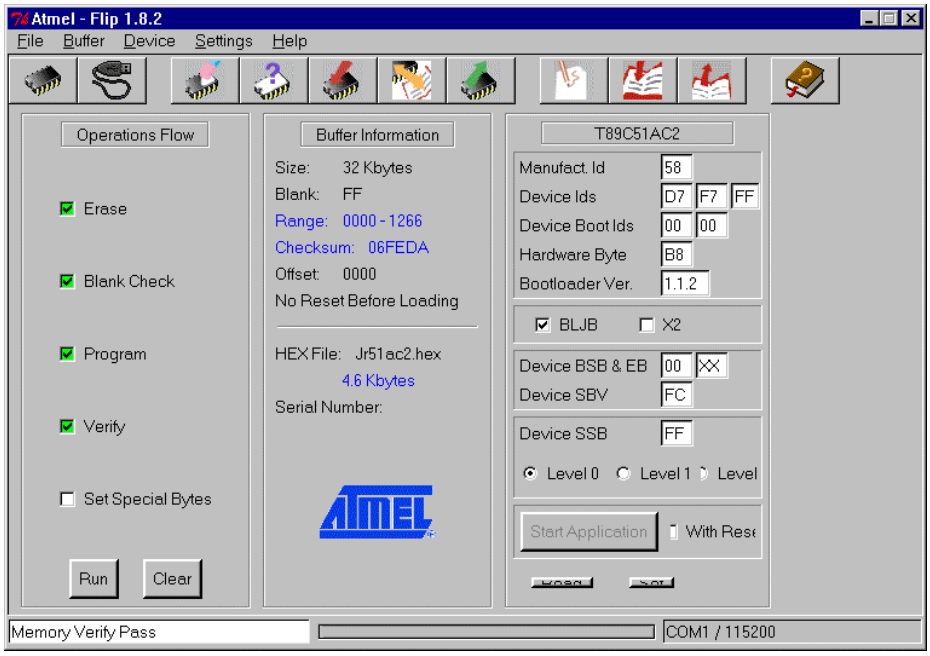


รูปแสดง ลักษณะการแสดงผลของโปรแกรมเมื่อ Load HEX ได้สำเร็จ

8. เลือกกำหนดตัวเลือกใน Operations Flow ที่ต้องการให้โปรแกรมทำงาน เช่น Erase,Blank Check,Program,Verify แล้วเลือก Run ซึ่งโปรแกรมจะเริ่มต้นทำงานทีละคำสั่งจากคำสั่งที่อยู่ด้านบนสุดไปยังคำสั่งที่อยู่ล่างสุด โดยในขณะที่กำลังทำงานตามคำสั่งใด ก็จะแสดงผลการทำงานนั้นๆให้ทราบตลอด และเมื่อทำงานตามคำสั่งใดเสร็จแล้วที่ช่องตัวเลือกการทำงานของคำสั่งนั้นๆก็จะเปลี่ยนเป็นสีเขียวให้ทราบทีละคำสั่งจนครบทุกคำสั่ง ดังรูป



รูปแสดง ลักษณะของโปรแกรม FLIP ในขณะกำลังทำงาน



รูปแสดง ลักษณะของโปรแกรม FLIP เมื่อทำงานตามคำสั่งเสร็จเรียบร้อยแล้ว

9. รอจนการทำงานของคำสั่งต่างๆที่เลือกไว้เสร็จเรียบร้อย
10. กดสวิตช์ RESET เพื่อให้ CPU เริ่มทำงานตามโปรแกรมที่ทำการ Download ไปให้

การสร้าง Configuration File

ตามปรกติแล้วเมื่อสั่งเปิดโปรแกรม FLIP ขึ้นมาในแต่ละครั้งนั้น ในครั้งแรกผู้ใช้จะต้องเริ่มต้นด้วยการกำหนดสิ่งต่างๆให้กับโปรแกรม เช่น สั่งเลือกเบอร์ CPU ที่จะใช้งาน(Select Device) กำหนดรูปแบบการสื่อสารให้กับโปรแกรมกำหนดชื่อและที่อยู่ของ HEX File ที่จะทำการ Download และสั่งโหลด HEX File มารอไว้ใน Buffer (Load HEX) จากนั้นก็สั่งจัดการ Download ข้อมูลให้กับหน่วยความจำของ CPU ซึ่งก็อาจต้องทำอีกหลายขั้นตอน เช่น สั่งลบข้อมูลออกจากหน่วยความจำของ CPU (Erase) สั่งตรวจสอบว่าข้อมูลถูกลบหมดหรือยัง (Blank Check) สั่งเขียนข้อมูลให้กับหน่วยความจำของ CPU (Program) สั่งเปรียบเทียบค่าของข้อมูลในหน่วยความจำของ CPU กับ Buffer (Verify) เป็นต้น ซึ่งจะเห็นว่าในการเรียกใช้งานโปรแกรม FLIP ในแต่ละครั้งนั้น จะต้องกระทำกระบวนการที่ซ้ำๆกันเสมอ ยิ่งในกรณีที่ผู้ใช้อยู่ในระหว่างการพัฒนาโปรแกรมด้วยแล้ว ก็อาจมีความจำเป็นต้องสั่งงานโปรแกรมในลักษณะเช่นนี้อยู่หลายๆครั้ง โดยไม่ได้เปลี่ยนแปลงข้อกำหนดใดๆของโปรแกรมเลยแม้แต่ชื่อ HEX File ที่จะทำการ Download ซึ่งจากปัญหาดังกล่าวนี้ โปรแกรม FLIP จึงได้สร้างไฟล์แบบ Configuration File เพื่อทำการบันทึกการกระทำต่างๆดังที่กล่าวไว้แล้วนี้ไว้ และเมื่อสั่งเปิดโปรแกรม FLIP ในครั้งต่อไปผู้ใช้ก็ไม่จำเป็นต้องเสียเวลามาสั่งงานโปรแกรม FLIP หลายๆขั้นตอนดังกล่าวนั้นอีกแล้ว เพียงแต่ใช้วิธีการสั่งอ่าน Configuration File ที่บันทึกไว้ขึ้นมา ซึ่งโปรแกรม FLIP ก็จะทำการกำหนดการ Setup ต่างๆให้เองโดยอัตโนมัติ สำหรับวิธีการสร้าง Configuration File นั้นสามารถกระทำได้ดังนี้

ขั้นตอนในการสร้าง Configuration File

สำหรับวิธีการสร้าง Configuration File นั้นสามารถทำได้ 2 แบบ คือการสร้างด้วยวิธีการเขียนเป็น Text File ตามรูปแบบคำสั่งที่โปรแกรม FLIP กำหนดไว้ให้ และวิธีการสร้างด้วยการสั่งบันทึกจากเมนูคำสั่งของโปรแกรม FLIP เอง ซึ่งในที่นี้จะขอแนะนำวิธีการสร้าง Configuration File จากการสั่งบันทึกด้วยคำสั่งในเมนูคำสั่งของโปรแกรม FLIP เอง ซึ่งจะทำได้ง่ายและสะดวกกว่าการเขียนเป็น Text File มาก โดยมีขั้นตอนดังนี้

1. เตรียมการให้โปรแกรม FLIP สื่อสารกับ CPU ใน Monitor Mode โดยเริ่มต้นจากการเชื่อมต่อสายสื่อสารข้อมูล RS232 ระหว่างบอร์ด CP-JR51AC2 กับพอรต์อนุกรม RS232 ของคอมพิวเตอร์ PC และจ่ายไฟเลี้ยงวงจรให้บอร์ดพร้อมทำงาน จากนั้นจึงทำการรีเซ็ตการทำงานของ CPU ในบอร์ด ให้เริ่มต้นเข้าทำงานใน Monitor Mode รอไว้
2. สั่งเปิดโปรแกรม FLIP ให้ทำงาน
3. สั่งเลือกเบอร์ CPU (Select Device) โดยกำหนดเป็น T89C51AC2
4. สั่งกำหนดรูปแบบการสื่อสารของบอร์ดเป็น RS232 โดยใช้ Baudrate เป็น19200 ส่วน Port ให้กำหนดตามความเป็นจริง เช่น Com1 หรือ Com2 เป็นต้น
5. สั่งโหลด HEX File มายัง Buffer โดยใช้คำสั่ง Load HEX
6. สั่งลบข้อมูลเก่าออกจากตัว CPU โดยใช้คำสั่ง Erase
7. สั่งตรวจสอบข้อมูลว่าลบหมดหรือยังโดยใช้คำสั่ง Blank Check
8. สั่งเขียนข้อมูลจาก Buffer ไปยังหน่วยความจำของ CPU โดยใช้คำสั่ง Program
9. สั่งเปรียบเทียบค่าข้อมูลในหน่วยความจำของ CPU กับ Buffer โดยใช้คำสั่ง Verify

10. สั่งบันทึกการกระทำต่างๆเป็น Configuration File ไว้ โดยใช้คำสั่ง File → Save Configuration → ชื่อไฟล์ โดยกำหนดสกุลเป็น cfg (.cfg)

ซึ่งหลังจากกระทำตามขั้นตอนต่างๆข้างต้นจนเสร็จเรียบร้อยแล้ว การทำงานตามขั้นตอนต่างๆที่เกิดขึ้นหลังจากเริ่มเปิดโปรแกรมนั้นจะถูกบันทึกไว้ในไฟล์ Configuration ตามชื่อที่สั่งบันทึกไว้ โดยจะมีสกุลเป็น cfg (.cfg) โดยไฟล์ดังกล่าวจะถูกบันทึกเก็บไว้ในลักษณะของไฟล์แบบข้อมูล (Text File) ซึ่งถ้าใช้โปรแกรม Text Editor สั่งเปิดดูจะเป็นดังนี้

```
selectDevice T89C51AC2
set port COM1
set baud 115200
initProtocol RS232Standard
connectRS232 Standard
parseHexFile "C:/PRODUCT/jr-51ac2/examples/Jr51ac2.hex"
setupFullEraseDevice
setupBlankCheckDevice
set gui(blankCheckMin) 0000
set gui(blankCheckMax) 7FFF
setupProgramDevice
setupVerifyDevice
```

การเรียกใช้งาน Configuration File

ซึ่งหลังจากทำการบันทึก Configuration File ไว้แล้ว ในครั้งต่อไปเมื่อต้องการกลับเข้ามาทำงานโดยใช้วิธีการและขั้นตอนต่างๆเหมือนกันนี้อีกก็จะสามารถลดขั้นตอนการสั่งงานไปได้มากทีเดียว โดยถ้าต้องการทำงานซ้ำใหม่ ซึ่งอาจกลับไปแก้ไข HEX File ใหม่แต่ยังบันทึกไว้ในชื่อเดิมและเก็บ HEX File นั้นไว้ในตำแหน่งที่อยู่เดิมก็สามารถทำได้ดังต่อไปนี้

1. เตรียมการให้โปรแกรม FLIP สื่อสารกับ CPU ใน Monitor Mode โดยเริ่มต้นจากการเชื่อมต่อสายสื่อสารข้อมูล RS232 ระหว่างบอร์ด CP-JR51AC2 กับพอรต์อนุกรม RS232 ของคอมพิวเตอร์ PC และจ่ายไฟเลี้ยงวงจรให้บอร์ดพร้อมทำงาน จากนั้นจึงทำการรีเซ็ตการทำงานของ CPU ในบอร์ด ให้เริ่มต้นเข้าทำงานใน Monitor Mode รอไว้
2. สั่งเปิดโปรแกรม FLIP ให้ทำงาน
3. สั่งอ่าน Configuration File โดยใช้คำสั่ง File → Read Configuration → ชื่อที่สั่งบันทึกไว้
4. สั่งให้โปรแกรมทำงานตามคำสั่ง โดยใช้คำสั่ง File → Execute Configuration File

ซึ่งจะเห็นได้ว่าโปรแกรมจะทำงานตามขั้นตอนต่างๆจนครบทุกขั้นตอน โดยเริ่มต้นตั้งแต่ การเลือกกำหนดเบอร์ CPU การกำหนดรูปแบบในการสื่อสาร RS232 การโหลด HEX File มายัง Buffer การสั่งลบข้อมูลออกจากหน่วยความจำของ CPU (Erase) การสั่งตรวจสอบผลการลบข้อมูล (Blank Check) การสั่งเขียนข้อมูลให้กับหน่วยความจำของ CPU (Program) และสั่งตรวจสอบผลการ Program (Verify) โดยสั่งงานโปรแกรมเพียงแค่ 2 คำสั่ง คือ Read Configuration File และ Execute Configuration File ทำให้สะดวกในการใช้งานมาก

ความหมายของค่าพารามิเตอร์ต่าง ๆที่แสดงในโปรแกรม

จะเห็นได้ว่าเมื่อทำการกำหนดเลือกเบอร์ CPU และทำการกำหนดรูปแบบในการติดต่อสื่อสารระหว่างโปรแกรม FLIP และ CPU ได้สำเร็จแล้ว โปรแกรม FLIP จะทำการอ่านค่าพารามิเตอร์ต่างๆจากตัว CPU ออกมาแสดงผลให้ทราบทางหน้าจอของโปรแกรมด้วย ซึ่งค่าต่างๆเหล่านี้จะเป็นค่าที่อ่านได้จริงจากตัว CPU โดยค่าบางอย่างเป็นค่าที่สามารถอ่านออกมาแสดงผลให้ทราบได้อย่างเดียว แต่ค่าบางอย่างก็สามารถเปลี่ยนแปลงแก้ไขและเขียนกลับเข้าไปในตัว CPU ได้ด้วย โดยเมื่อสั่งเขียนด้วยคำสั่ง Set Device Special Byte นั้น เมื่อโปรแกรมทำการเขียนค่าต่างๆไปให้ CPU เสร็จแล้วโปรแกรม FLIP จะอ่านค่าต่างๆย้อนกลับออกมาเพื่อแสดงให้ทราบด้วย ซึ่งค่าที่แสดงให้เห็นจึงเป็นค่าที่อ่านได้จริงจากตัว CPU ไม่ใช่ค่าที่กำหนดจากโปรแกรมโดยค่าของพารามิเตอร์ต่างๆมีหน้าที่และการทำงานที่ควรทราบดังต่อไปนี้

- **Manufact. Id** เป็นค่ารหัสผู้ผลิตของ CPU ซึ่งถ้าเป็นของ ATMEL จะมีค่าเป็น **58H** โดยค่านี้จะกำหนดไว้ตายตัวในขั้นตอนของการผลิต CPU สามารถอ่านได้อย่างเดียว
- **Device ids** เป็นค่ารหัส ID Code ของตัว CPU ซึ่งในกรณีที่เป็นเบอร์ T89C51AC2 ของ ATEML นั้นจะมีรหัส ID Code ขนาด 3 ไบท์ โดยมีค่าคงที่เป็น **D7H,F7H,FFH** ตามลำดับ โดยค่านี้จะกำหนดไว้ตายตัวในขั้นตอนของการผลิต CPU สามารถอ่านได้อย่างเดียว
- Device Boot ids เป็นค่า Device Boot ID Code ของ CPU สามารถอ่านได้อย่างเดียว
- **Hardware Byte** เป็นค่าของ Hardware Security โดยค่าของไบท์ข้อมูลนี้ ในส่วนของโปรแกรม FLIP จะยอมให้ผู้ใช้สามารถอ่านค่าออกมาได้อย่างเดียวไม่อนุญาติให้ผู้ใช้ทำการเข้าไปแก้ไขค่าของข้อมูลของไบท์นี้ด้วยวิธีการป้อนค่าเป็นตัวเลขแบบไบท์ เพื่อป้องกันความผิดพลาด โดยถ้าต้องการแก้ไขค่าของบิตต่างๆของ Hardware Security ไบท์นี้ ต้องเลือกกำหนดจากตัวเลือกของบิตต่างๆที่เกี่ยวข้องเอง เช่น X2,BLJB หรือค่า Security Level0,Level1หรือ Level2 ของ Device SSB แทน ซึ่งเมื่อแก้ไขค่าต่างๆดังกล่าวแล้วสั่ง Set Device Special Byte แล้วค่าของ Hardware ไบท์นี้จะเปลียนแปลงตามค่าที่กำหนดใหม่ด้วย
- **Bootloader Ver.** ใช้แสดง Version ของโปรแกรม Boot Loader ที่อยู่ในตัว CPU ซึ่งอ่านได้อย่างเดียว
- **BLJB** เป็นค่าของ **Boot Loader Jump Bit** เป็นบิตใช้สำหรับกำหนดการทำงานของ CPU หลังการรีเซ็ต ว่าต้องการให้ CPU กระโดดไปทำงานยังตำแหน่งแอดเดรส 0000H ซึ่งเป็นตำแหน่งการทำงานปรกติ หรือจะให้กระโดดไปทำงานยังตำแหน่ง F800H ซึ่งเป็นตำแหน่งการทำงานของ Boot Loader ใน Monitor Mode โดยถ้าเลือกเครื่องหมายถูก (√) ที่บิตนี้จะเป็นการกำหนดให้บิตนี้มีค่าเป็น “0” ซึ่งหมายถึง กำหนดให้ CPU กระโดดไปทำงานใน Boot Loader หรือ Monitor Mode แต่ถ้าไม่เลือกจะเป็นการกำหนดให้บิตนี้เป็น “1” ซึ่งหมายถึง CPU จะกระโดดไปทำงานยังตำแหน่งแอดเดรส 0000 ซึ่งเป็นโปรแกรมทำงานตามปรกติของผู้ใช้ โดยบิตนี้ผู้ใช้สามารถสั่งเปลี่ยนแปลงหรือแก้ไขได้
- **X2** เป็นค่า **X2 Fuse Bit** ซึ่งเป็นบิต **X2B** หรือ บิต7 ของรีจิสเตอร์ Hardware Security Byte ซึ่งเป็นบิตใช้สำหรับกำหนดโหมดการทำงานเริ่มต้นหลังการรีเซ็ตของ Oscillator ว่าจะให้ CPU เริ่มต้นทำงานใน Standard Mode (12 Clock / Machine Cycle) หรือจะให้ CPU เริ่มต้นทำงานแบบ X2 Mode (6 Clock / Machine Cycle) โดยถ้าเลือกเครื่องหมายถูก (√) หน้าบิตนี้จะเป็นการกำหนดให้บิตนี้มีค่าเป็น “0” ซึ่งหมายถึงให้ CPU เริ่มต้นทำงานใน X2 Mode (6 Clock / Machine Cycle) แต่ถ้าไม่เลือกจะเป็นการกำหนดให้บิตนี้เป็น “1” ซึ่งหมายถึง ให้ CPU เริ่มต้นทำงานแบบ Standard Mode (12 Clock / Machine Cycle)

ซึ่งการกำหนดบิต X2 จากโปรแกรม FLIP นี้เป็นเพียงการกำหนดการทำงานของ Oscillator ของ CPU หลังการรีเซ็ตเท่านั้น ซึ่งการทำงานของ Oscillator ของ CPU นี้ยังสามารถแก้ไขได้จากโปรแกรมอีกครั้งหนึ่ง โดยการควบคุมจากบิต X2 (บิต0) ของรีจิสเตอร์ CKCON ในโปรแกรมของผู้ใช้ แต่ถ้าในโปรแกรมของผู้ใช้ไม่ได้ไปสั่งเปลี่ยนแปลงค่าของ บิต X2 (บิต0) ของรีจิสเตอร์ CKCON แล้ว การทำงานของ Oscillator ของ CPU ก็จะมีคุณสมบัติเหมือนกับที่กำหนดไว้จากบิต X2 ของโปรแกรม FLIP นี้ไปตลอด

- **Device BSB & EB** เป็นค่าของ Boot Status Byte และ Extra Byte โดย BSB หรือ Boot Status Byte นั้น ถ้ากำหนดให้เป็น 00H ไว้ เมื่อถูกรีเซ็ตแล้วเงื่อนไขทางฮาร์ดแวร์ของ Monitor Mode ไม่ถูกต้องจะทำให้ CPU เริ่มต้นทำงานตามโปรแกรมในตำแหน่ง 0000H ในทันทีโดยไม่สนใจเงื่อนไขของบิต BLJB แต่ถ้ากำหนดให้ค่าของ BSB มีค่าอื่นที่ไม่ใช่ 00H แล้ว CPU จะตรวจสอบเงื่อนไขของบิต BLJB ด้วย ส่วน EB หรือ Extra Byte นั้นเป็นไบท์ข้อมูลพิเศษสำหรับให้ผู้ใช้กำหนดรหัสต่างๆของข้อมูลให้กับ CPU ได้อย่างอิสระตามต้องการ ซึ่งสามารถสั่งเปลี่ยนแปลงหรือแก้ไขได้ตามต้องการจากโปรแกรมของผู้ใช้ แต่ในส่วนของโปรแกรม FLIP นั้นค่าของ EB จะอ่านออกมาจาก CPU เพื่อแสดงผลเพียงอย่างเดียว ไม่สามารถให้ทำการแก้ไขค่าข้อมูลของไบท์นี้จากโปรแกรม FLIP
- **Device SBV** เป็นค่าของ **Software Boot Vector** หรือ Boot Vector Byte ซึ่งเป็นค่าตำแหน่งไบท์สูง ของตำแหน่งการทำงาน Boot Loader ในโปรแกรมส่วนของผู้ใช้ (ไม่ใช่ของ Monitor Mode) ส่วนค่าของตำแหน่งแอดเดรสไบท์ต่ำจะถูกกำหนดคงที่ด้วยค่า 00 เช่น เมื่อกำหนดค่าของ SBV ไว้เป็น F8 จะหมายถึงตำแหน่ง F800
- **Device SSB** เป็นค่าข้อมูลของ Software Security Byte ซึ่งใช้สำหรับกำหนดระดับการป้องกันข้อมูลของ CPU โดยค่าของ Device SSB นี้เป็นค่าที่สามารถอ่านออกมาได้อย่างเดียวไม่อนุญาติให้ผู้ใช้ทำการแก้ไขค่าของ ไบท์นี้ได้โดยตรง ซึ่งถ้าต้องการแก้ไขให้เลือกจากตัวเลือก Level0 หรือ Level1 หรือ Level2 อย่างใดอย่างหนึ่งแล้วจึงสั่ง Set Device Special Byte ซึ่งจะทำให้ค่าของ Device SSB เปลี่ยนค่าตามไปด้วยโดยอัตโนมัติ
- **Level0** เป็นการกำหนดให้ยกเลิกระบบการป้องกันการคัดลอกข้อมูล ซึ่งสามารถสั่งอ่านข้อมูลออกจากหน่วยความจำข้อมูลของ CPU ที่สั่งโปรแกรมเข้าไปแล้วได้ โดยเมื่อเลือกตัวเลือกนี้แล้ว ยังสามารถเปลี่ยนไปยัง Level1 หรือ Level2 แล้วสั่ง Set Device Special Byte เพื่อเพิ่มระดับการป้องกันข้อมูลให้สูงขึ้นกว่าระดับเดิมนี้ได้
- **Level1** เป็นการกำหนดระดับการป้องกันข้อมูลโดยไม่สามารถเขียนข้อมูลเข้าไปยังหน่วยความจำโปรแกรมของ CPU ได้อีกถ้าต้องการเขียนจะต้องสั่งลบข้อมูลเก่าออกก่อน ส่วนการอ่านข้อมูลยังสามารถทำได้ โดยเมื่อเลือกระดับการป้องกันข้อมูลไว้ที่ Level1 นี้พร้อมกับสั่ง Set Device Special Byte ไปแล้ว จะไม่สามารถแก้ไขระดับการป้องกันกลับเป็น Level0 ได้ แต่สามารถเปลี่ยนระดับการป้องกันเป็น Level2 ได้
- **Level2** เป็นการกำหนดระดับการป้องกันข้อมูลระดับสูงสุด ซึ่งจะไม่สามารถอ่านหรือเขียนข้อมูลไปยังหน่วยความจำของ CPU ได้อีก โดยเมื่อเลือกระดับการป้องกันข้อมูลไว้ที่ Level2 นี้พร้อมกับสั่ง Set Device Special Byte ไปแล้ว จะไม่สามารถแก้ไขระดับการป้องกันกลับเป็น Level1 หรือ Level0 ได้อีก

สิ่งที่ควรรู้เกี่ยวกับโปรแกรม FLIP

1. เมื่อสั่งลบข้อมูล (Erase Device)
   - ค่าของ Device BSB (Boot Status Byte) จะมีค่าเป็น FFH
   - ค่าของ Device SBV (Software Boot Vector)จะมีค่าเป็น FCH
   - ค่าของ Device SSB (Software Security Byte) จะมีค่าเป็น FFH (การป้องกันข้อมูลเป็น Level0)
2. เมื่อสั่งโปรแกรมข้อมูล (Program Device)
   - ค่าของ Device BSB (Boot Status Byte) จะมีค่าเป็น 00H

BLJB	BSB	SBV	**โหมดการทำงานหลังการทำงานแบบปรกติ**
( ) ไม่เลือก	XX	XX	User Mode (เริ่ม Run ตำแหน่ง 0000H)
(√) เลือก	00H	XX	User Mode (เริ่ม Run ตำแหน่ง 0000H)
(√) เลือก	ไม่ใช่ 00H	ไม่ใช่ FCH	User Mode (เริ่ม Run จากตำแหน่ง [SBV:00H])
(√) เลือก	ไม่ใช่ 00H	FCH	Monitor Mode (เริ่ม Run ตำแหน่ง F800H)

ตาราง แสดงโหมดการทำงานของ CPU หลังการรีเซ็ตแบบปรกติ

3. การกำหนดระดับการป้องกันข้อมูลของ CPU ด้วยโปรแกรม FLIP นั้นจะสามารถทำได้ 3 ระดับเท่านั้น ซึ่งตามปรกติแล้วระดับการป้องกันข้อมูลของ CPU จะมีถึง 4 ระดับ โดยการกำหนดจากบิต **LB2:0 หรือบิต 2,1 และ 0** ของ Hardware Security Byte ซึ่งเป็นบิต **Lock Bit** มีด้วยกันทั้งหมด 3บิต โดยจะใช้ร่วมกันสำหรับกำหนดระดับการป้องการข้อมูลของ CPU ซึ่งมีความหมายดังนี้

LB0	LB1	LB2	**ลักษณะการป้องกัน**
1	1	1	ไม่มีการป้องกันใดๆ สามารถอ่านเขียนข้อมูลได้ตามต้องการ
0	1	1	ไม่สามารถใช้คำสั่ง MOVC ที่ Run จากหน่วยความจำที่อยู่ภายนอกตัว CPU ได้ สถานะของขา EA จะถูก Latch ในช่วงของการ Reset เพียงครั้งเดียว หลังการรีเซ็ตแล้วการเปลี่ยนแปลงสถานะของ EA จะไม่มีผลต่อการทำงานของ CPU และไม่สามารถสั่งเขียนข้อมูลให้กับ CPU ได้อีกจนกว่าจะสั่งลบ ส่วนการอ่านข้อมูลจาก CPU ยังทำได้ตามปรกติ
1	0	1	มีลักษณะเช่นเดียวกับการ Lock บิต LB0 แต่แบบนี้จะไม่สามารถสั่งอ่านข้อมูลจากหน่วยความจำของ CPU จากภายนอกจากเครื่องโปรแกรม (Verify) ได้อีก
1	1	0	มีลักษณะเช่นเดียวกับการ Lock บิต LB1 แต่แบบนี้จะไม่สามารถต่อหน่วยความจำภายนอกให้กับ CPU ได้อีก ถึงแม้ว่าจะอ้างตำแหน่งเกิน 7FFFH แล้วก็ตาม

ซึ่งจะเห็นได้ว่าการกำหนดระดับการป้องกันข้อมูลของ CPU โดยใช้โปรแกรม FLIP นั้น จะสามารถกำหนดได้ 3 ระดับ คือ

- ไม่มีการ Lock บิตใดๆ (Level0 หรือ LB2,LB1 และ LB0 เป็น "1" ทั้งหมด) ซึ่งจะทำให้สามารถสั่งอ่านเขียนข้อมูลให้กับหน่วยความจำของ CPU ได้ตามต้องการ นอกจากนี้แล้วยังสามารถกำหนดให้ CPU ทำงานจากคำสั่งในโปรแกรมที่บรรจุอยู่ในหน่วยความจำภายนอกของ CPU ได้อีกด้วย โดยกำหนดจากขา EA ตามคุณสมบัติของ CPU ในตระกูล MCS51 ทั่ว
- สั่ง Lock บิต LB0 เพียงบิตเดียว (Level1) ซึ่งในกรณีนี้ยังจะไม่สามารถสั่งเขียนข้อมูลใดๆให้กับหน่วยความจำโปรแกรมของ CPU ได้อีกแล้ว ถึงแม้ว่าหน่วยความจำจะยังว่างอยู่ก็ตามที ส่วนการอ่านข้อมูลจากหน่วยความจำจากภายนอกยังสามารถทำได้ตามปรกติ และการทำงานของ CPU จะเริ่มต้นทำงานในตำแหน่งแอดเดรส 0000H ของหน่วยความจำที่อยู่ภายในตัว CPU เท่านั้น ส่วนการทำงานตามโปรแกรมที่อยู่ในหน่วยความจำภายนอกนั้นจะกระทำได้เมื่อค่าตำแหน่งแอดเดรสของหน่วยความจำที่อ้างถึงมีค่าเกินกว่าตำแหน่งหน่วยความจำภายในตัวของ CPU ทีมีอยู่จริงแล้ว
- สั่ง Lock บิต LB1 ซึ่งคุณสมบัติของ CPU เมื่อถูก Lock บิตนี้แล้วจะมีลักษณะเหมือนกับการ Lock บิต LB0 ทุกประการ แต่สิ่งที่เพิ่มเติมเข้ามานั้นก็คือ จะไม่สามารถสั่งอ่านค่าข้อมูลจากหน่วยความจำของ CPU จากภายนอกได้อีกแล้ว ซึ่งเป็นการป้องกันการคัดลอกข้อมูลภายในตัว CPU ด้วย
- สั่ง Lock บิต LB2 จะมีคุณสมบัติเหมือนกับการ Lock บิต LB1 แต่สิ่งที่เพิ่มเติมคือ CPU จะไม่สามารถใช้งานกับหน่วยความจำที่อยู่ภายนอกตัว CPU ได้ ถึงแม้ว่าตำแหน่งแอดเดรสที่อ้างถึงจะไม่มีอยู่ในตัว CPU แล้วก็ตาม แต่อย่างไรก็ตามการทำงานของ CPU ในบอร์ด CP-JR51AC2 นั้นจะทำงานใน Single Chips Mode ซึ่งไม่ได้ออกแบบให้ CPU ต่อกับหน่วยความจำภายนอกอยู่แล้วดังนั้นจึงไม่มีผลต่อบอร์ด

****หมายเหตุ**** หลังจากสั่งลบข้อมูลของ CPU ด้วยคำสั่ง Erase แล้วค่าสถานะของบิต LB2:0 จะมีค่าเป็น "1" ทั้งหมด การโปรแกรม Lock Bit ทำได้โดยการเขียนค่า Lock Bit ให้เป็น "0"

สำหรับบิต LB2 นั้นจะไม่สามารถสั่งเปลี่ยนแปลงด้วยโปรแกรม FLIP ได้ ต้องใช้กับเครื่องมือโปรแกรม CPU ที่ใช้วิธีการโปรแกรมแบบขนาด (Parallel Programming) เท่านั้น

ส่วนวิธีการยกเลิกระดับการป้องกันข้อมูลหรือ Security Level หลังจากสั่ง Set Device Special Byte ไปแล้วจะสามารถทำได้วิธีเดียวด้วยการสั่งลบข้อมูลออกจากตัว CPU ทั้งหมดด้วยคำสั่ง Erase เท่านั้น ซึ่งการกระทำดังกล่าวจะส่งผลให้ข้อมูลในหน่วยความจำของ CPU รวมทั้งบิตสำหรับกำหนดระดับการป้องกันข้อมูลของ CPU กลับมาอยู่ที่ Level0 (ไม่ป้องกันการอ่านหรือเขียนใดๆ)

แต่สำหรับในกรณีที่เลือกระดับการป้องกัน จาก Level0 ไปเป็น Level1 หรือ Level2 นั้น ถ้ายังไม่ได้สั่ง Set Device Special Byte ค่าที่เลือกไว้จะยังไม่มีผลต่อการทำงานใดๆของ CPU ทั้งสิ้น

ปัญหาต่าง ๆในขณะใช้งานโปรแกรม FLIP และแนวทางการแก้ไข

ในบางครั้งเมื่อเรียกใช้คำสั่งต่างๆของโปรแกรม FLIP แล้ว อาจเกิดความผิดพลาดบางประการขึ้น ซึ่งอาจไม่ใช่ปัญหาที่เกิดจากความบกพร่องของระบบฮาร์ดแวร์ แต่อาจเกิดการการกำหนดพารามิเตอร์บางอย่างในโปรแกรมไม่ถูกต้องหรือข้ามขั้นตอนบางประการไป ซึ่งเมื่อโปรแกรม FLIP ไม่สามารถปฏิบัติตามคำสั่งที่ผู้ใช้งานสั่งไปได้สำเร็จจะแสดงอาการ Error ต่างๆให้ทราบ ซึ่งพอสรุปได้ดังนี้

1. **Time Out Error** เป็นความผิดพลาดที่เกิดจากการที่โปรแกรม FLIP ไม่สามารถทำการสื่อสารกับ CPU ใน Monitor Mode ได้ ซึ่งอาจเกิด หลายสาเหตุ เช่น
   - การต่อสายสัญญาณระหว่างขั้วต่อ RS232 (4PIN) ของบอร์ด CP-JR51AC2 กับขั้วต่อพอร์ตสื่อสารอนุกรม RS232 ของคอมพิวเตอร์ยังไม่เรียบร้อยหรือต่อไม่ตรงกับที่กำหนดตัวเลือกไว้ในโปรแกรม หรือการกำหนดรูปแบบและตัวเลือกต่างๆในการสื่อสารไม่ถูกต้อง เมื่อพบปัญหานี้ให้ลองทำการตรวจสอบค่าต่างๆในการสื่อสารใน Setting → Communication → RS232
   - ยังไม่ได้รีเซ็ตให้ CPU เข้าทำงานใน Monitor Mode รอไว้ก่อนที่จะสั่งงานโปรแกรม หรือบอร์ดยังไม่พร้อมทำงาน เช่น ยังไม่ได้จ่ายไฟเลี้ยงให้บอร์ด
   - กำหนดค่า Baudrate เร็วเกินไป ซึ่งในกรณีที่ใช้งานกับเครื่องคอมพิวเตอร์ที่มีความเร็วมากๆนั้น ควรกำหนดค่า Baudrate ในการสื่อสารให้ช้าลง ซึ่งอาจใช้ค่า 19200 หรือ 9600 ก็พอ เพราะถ้ากำหนดให้ความเร็วมากเกินไป เมื่อโปรแกรม FLIP ส่งข้อมูลให้กับ CPU แบบต่อเนื่องนั้น อาจทำให้ CPU ไม่สามารถประมวลผลคำสั่งหรือข้อมูลต่างๆที่ส่งไปให้ทันก็จะทำให้เกิดความผิดพลาดบ่อยครั้งขึ้น
2. **Software Security Bit Set. Cannot access device Data** เป็นความผิดพลาดที่เกิดจากการนำ CPU ที่มีการสั่ง Lock Bit ของ Security Bit ไว้ก่อนแล้ว จึงมาสั่ง Program หรือ Verify หรือ Read ในภายหลังโดยยังไม่ได้สั่งลบข้อมูลเก่าออกเสียก่อน ซึ่งให้แก้ปัญหาด้วยการสั่งลบข้อมูล (Erase) เสียก่อนแล้วจึงสั่งเขียนข้อมูลใหม่อีกครั้งหนึ่ง
3. **The board reply is not correct** เป็นความผิดพลาดที่เกิดจากการสื่อสารข้อมูลระหว่างโปรแกรม FLIP กับไมโครคอนโทรลเลอร์ เกิดความผิดพลาดในลักษณะของ Framing Error ขึ้น ซึ่งปัญหาอาจเกิดจากการกำหนดค่า Baudrate ไม่ถูกต้องกับค่าความถี่ของ Crystal ที่ใช้กับบอร์ด
4. **The RS232 port could not be opened** เป็นความผิดพลาดที่เกิดจากโปรแกรม FLIP ไม่สามารถสั่งเปิดการทำงานของพอร์ตสื่อสารอนุกรม RS232 ของเครื่องคอมพิวเตอร์ PC ได้ ซึ่งอาจเกิดจากการกำหนดหมายเลข Comport ในโปรแกรมที่เลือกไว้ไม่มีอยู่จริง หรือมีโปรแกรมอื่นเรียกใช้งาน Comport นั้นค้างอยู่หรือเรียกใช้งานโปรแกรม FLIP ในขณะที่กำลังสั่งปิดโปรแกรมอื่นๆที่มีการใช้งาน Comport อยู่ด้วย ซึ่งให้ลองปิดโปรแกรม FLIP แล้วสั่งเปิดโปรแกรมใหม่ดู ถ้ายังเกิดปัญหาเดิมอยู่อีกอาจลองตรวจสอบสาเหตุอื่นๆที่เกี่ยวข้องและทำการแก้ไข
5. **Check sum error** เป็นความผิดพลาดที่เกิดจากการที่ CPU รับข้อมูลที่ส่งไปจากคอมพิวเตอร์ PC ไม่ครบถูกต้องทั้งหมด ซึ่งปัญหาอาจเกิดจากการกำหนดความเร็วในการสื่อสาร Baudrate เร็วเกินไป หรือกำหนดไว้ไม่เหมาะสมกับค่าความถี่ Crystal ค่า 18.432MHz ให้ลองเปลี่ยนค่า Baudrate ให้ช้าลงกว่าเดิม ซึ่งค่าที่เหมาะสมได้แก่ 9600,19200 และ 38400 แต่ถ้าคอมพิวเตอร์ไม่เร็วมากนักก็อาจกำหนดเป็น 57600 หรือ 115200 ก็ได้ แต่ถ้ากำหนดค่าสูงๆแล้วเกิด Error ควรลดค่า Baudrate ให้ช้าลงกว่าเดิม

6. การสั่ง Load HEX ไม่ได้ เป็นความผิดพลาดที่เกิดจากการที่โปรแกรม FLIP ไม่สามารถอ่านข้อมูลใน HEX File ออกมาได้ ซึ่งอาจเกิดจากไฟล์ที่สั่งโหลดนั้น ไม่ใช่ไฟล์แบบ Intel HEX เนื่องจากโปรแกรม FLIP สามารถใช้งานกับไฟล์แบบ Intel HEX เท่านั้น ส่วนไฟล์ในรูปแบบอื่นๆจะไม่สามารถนำมาใช้งานกับโปรแกรมนี้ได้ ส่วนปัญหาอีกประการหนึ่งที่มักพบอยู่บ่อยๆ คือโปรแกรม FLIP ไม่สามารถอ่าน HEX File ได้ทั้งๆที่ไฟล์ที่สั่งให้อ่านนั้นเป็นไฟล์แบบ Intel HEX อยู่แล้ว ซึ่งที่พบอยู่บ่อยๆก็ได้แก่ HEX File ที่สั่งแปลโดยใช้โปรแกรม Assembler ของ SXA51.EXE เนื่องจาก HEX File ที่ได้จากการแปลของโปรแกรมตัวนี้จะเกิดบรรทัดว่างอยู่ในไฟล์ในส่วนเริ่มต้นบรรทัดแรกด้วย 1 บรรทัด ซึ่งตามรูปแบบของ HEX File แล้ว ในแต่ละบรรทัดของไฟล์จะต้องเริ่มต้นด้วยเครื่องหมายโคลอน (:) แล้วตามด้วยข้อมูลต่างๆในบรรทัดนั้น แต่เมื่อบรรทัดแรกเป็นบรรทัดว่างโปรแกรมจึงแสดง Error ว่าไม่ใช่ HEX File โดยโปรแกรม FLIP จะแสดง Error . ให้ทราบดังนี้

สำหรับวิธีการแก้ไขปัญหานี้ให้ใช้โปรแกรม Text Editor เปิด HEX File ที่ได้จากการแปลของ SXA51.EXE แล้วตัดบรรทัดว่างในไฟล์นั้นทิ้งไปแล้วสั่งบันทึกใหม่ก็จะสามารถนำไปใช้ได้แล้ว

```
:10 0000075892175985 0D2BCC2ACC2A9C2AB74FB31
:100 0F58DF58BD28CD28ED2AF1200880D0A0AE4
:1 บรรทัดว่างที่ได้จากการแปล 2054455342043502D4A52359A
:1 ของโปรแกรม SXA51 2056312026205632202831329 9
:10 636B290D0A50726573732041E9
:100050006E79204B657920466F7220546573742049
:00000001FF
```

รูปแสดง ลักษณะของ HEX File ที่ได้จากการแปลของ SXA51 ซึ่งจะเกิดบรรทัดว่างอยู่ 1 บรรทัด

```
:10000000758921759850D2BCC2ACC2A9C2AB74FB31
:10001000F58DF58BD28CD28ED2AF1200880D0A0AE4
:1000200044454D4F2054455342043502D4A52359A
:1000300031414332205631202620562202831329 9
:100040002D436C6F636B290D0A50726573732041E9
:100050006E79204B657920466F7220546573742049
:00000001FF
```

รูปแสดง ลักษณะของ HEX File ที่สามารถใช้กับโปรแกรม FLIP ได้หลังตัดบรรทัดว่างทิ้งไปแล้ว

7. เมื่อสั่งโปรแกรมข้อมูลให้กับ CPU เรียบร้อยแล้วหลังจากรีเซ็ตบอร์ดแล้วไม่ทำงาน ซึ่งปัญหานี้อาจเกิดจากสาเหตุความผิดพลาดหลายประการ ซึ่งพอสรุปได้ 2 กรณี คือ
   - โปรแกรมที่เขียนขึ้นไม่ถูกต้องยังไม่สามารถทำงานได้เอง ซึ่งปัญหานี้ผู้ใช้ต้องหาทางตรวจสอบและแก้ไขความผิดพลาดที่เกิดขึ้นเอง
   - ยังไม่ได้มีการสั่ง Load HEX เข้ามารอไว้ยัง Buffer แล้วสั่งโปรแกรม (Program Device) ซึ่งโปรแกรม FLIP จะนำข้อมูลที่อยู่ใน Buffer เขียนไปยังหน่วยความจำของโปรแกรม
   - สวิตช์ PSEN อาจถูกกดค้างอยู่ จึงทำให้การรีเซ็ตบอร์ดทุกๆครั้งนั้น CPU จะเข้าไปทำงานใน Monitor Mode เสมอ ซึ่งปัญหานี้สามารถตรวจสอบได้โดยการวัดระดับลอจิกที่ขาสัญญาณ PSEN ของ CPU ซึ่งอยู่ที่ขา 38 (PLCC-44) ซึ่งควรมีสภาวะเป็น “1” ถ้าไม่มีการกดสวิตช์ PSEN ไว้ และควรมีสภาวะเป็น “0” ถ้ามีการกดสวิตช์ PSEN ไว้
   - สวิตช์ RESET อาจถูกกดค้างอยู่ จึงทำให้ CPU ไม่สามารถหลุดพ้นจากสภาวะการรีเซ็ตได้ ซึ่งปัญหานี้สามารถตรวจสอบได้โดยการวัดระดับลอจิกที่ขาสัญญาณ RESET ของ CPU ซึ่งอยู่ที่ขา 44 (PLCC44) ซึ่งควรมีสภาวะเป็น “0” ถ้าไม่มีการกดสวิตช์ RESET ไว้ และควรมีสภาวะเป็น “1” ถ้ามีการกดสวิตช์ RESET ไว้
   - มีการเลือกบิต BLJB ของ CPU ไว้ด้วย ซึ่งในกรณีนี้ ถ้ามีการเลือกเครื่องหมายถูก (√) หน้าตัวเลือกของบิต BLJB (Boot Loader Jump Bit) ไว้ด้วย ซึ่งจะเป็นการกำหนดให้ CPU ต้องกระโดดไปทำงานยังตำแหน่ง F800H หลังจากการรีเซ็ต โดย CPU จะตรวจสอบเงื่อนไขอื่นๆร่วมด้วยดังตาราง

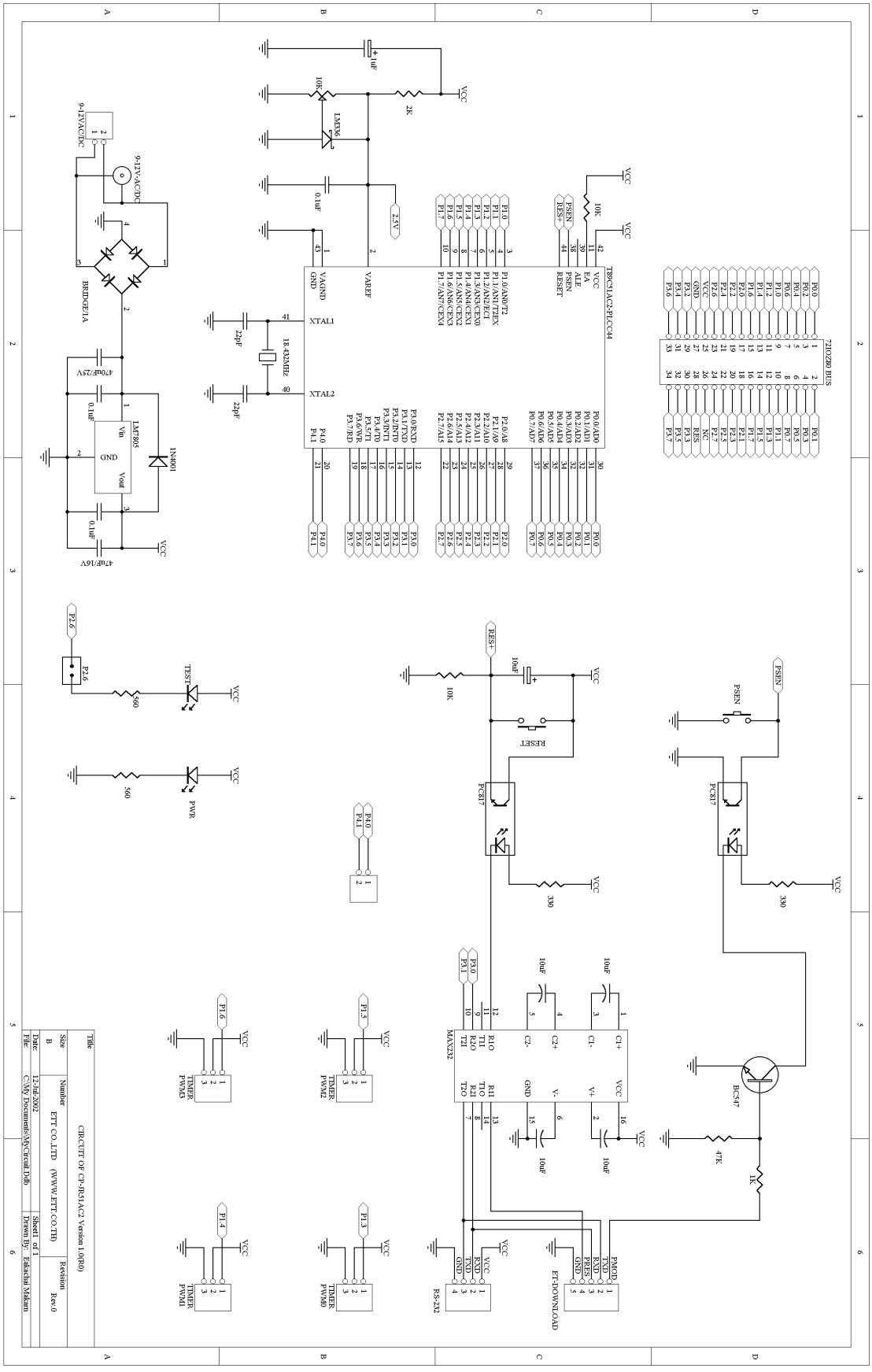
BLJB	BSB	SBV	**โหมดการทำงานหลังการทำงานแบบปรกติ**
( ) ไม่เลือก	XX	XX	User Mode (เริ่ม Run ตำแหน่ง 0000H)
(√) เลือก	00H	XX	User Mode (เริ่ม Run ตำแหน่ง 0000H)
(√) เลือก	ไม่ใช่ 00H	ไม่ใช่ FCH	User Mode (เริ่ม Run จากตำแหน่ง [SBV:00H])
(√) เลือก	ไม่ใช่ 00H	FCH	Monitor Mode (เริ่ม Run ตำแหน่ง F800H)

ตาราง แสดงโหมดการทำงานของ CPU หลังการรีเซ็ตแบบปรกติ

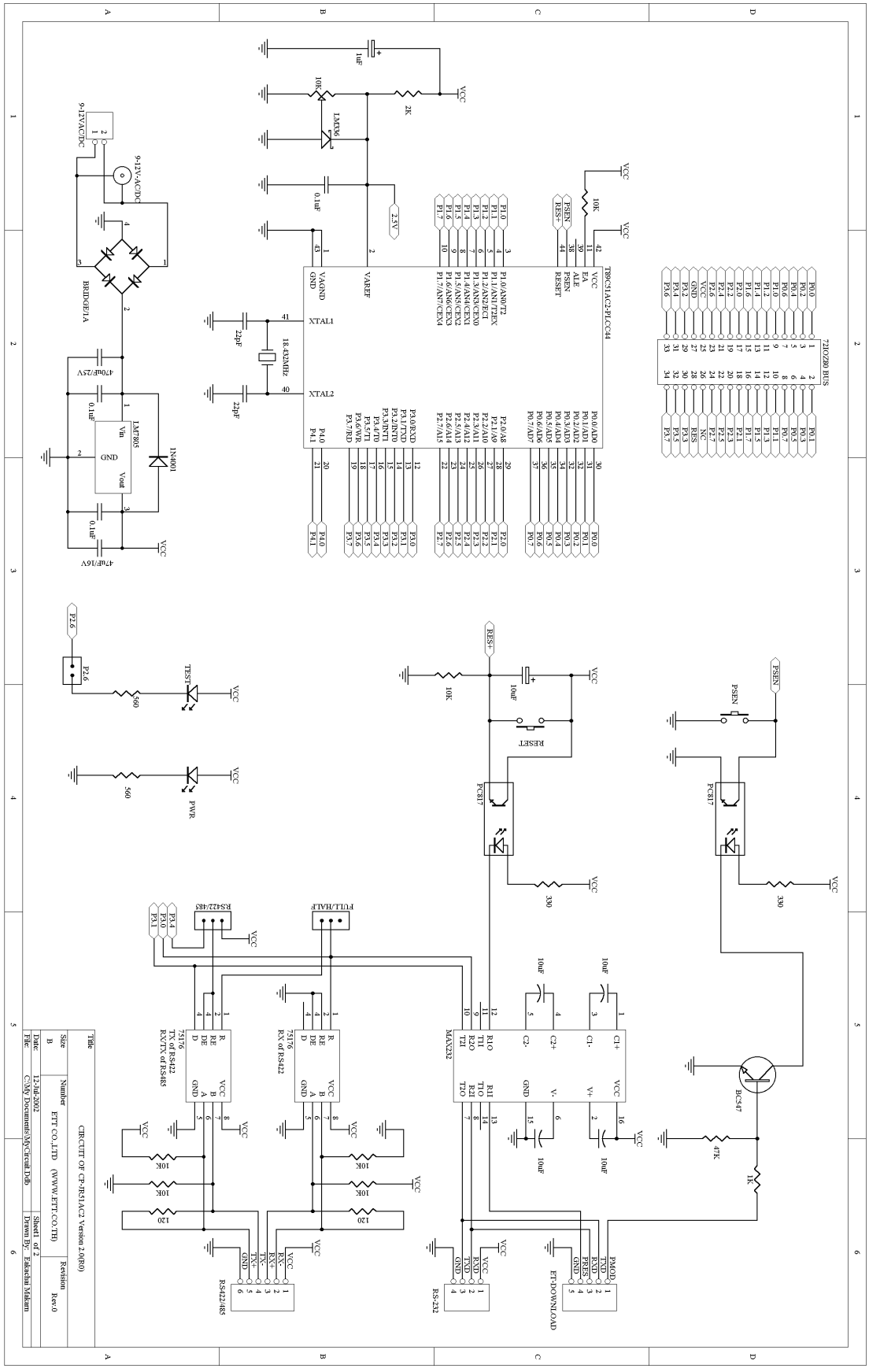
*****หมายเหตุ*****

- BLJB **หมายถึง** Boot Loader Jump Bit
- BSB **หมายถึง** Boot Status Byte
- SBV **หมายถึง** Software Boot Vector

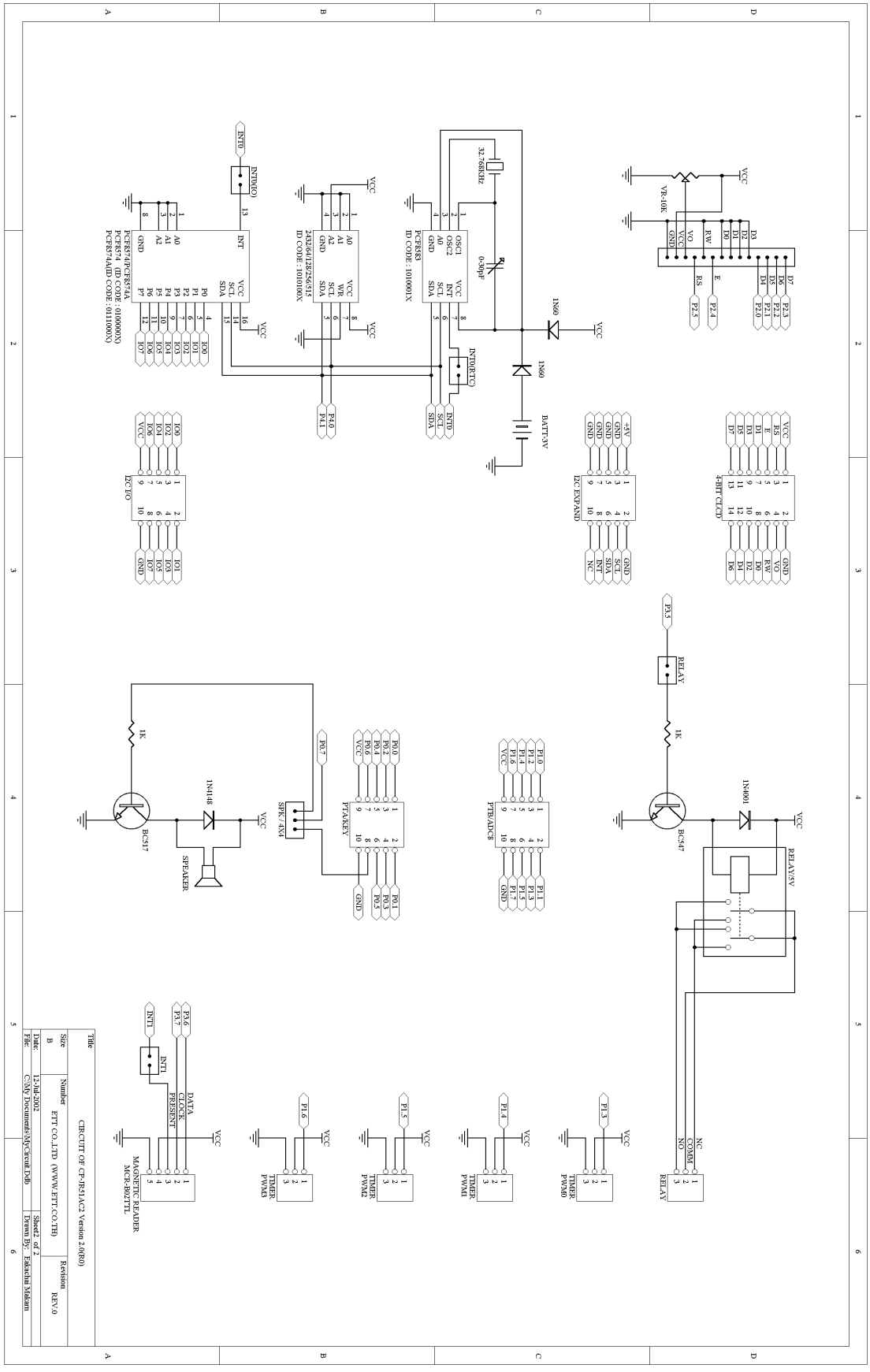
ซึ่งจะเห็นได้ว่าถ้ามีการเลือกบิต BLJB ไว้ แล้วค่าของ BSB ไม่ได้ถูกกำหนดให้มีค่าเป็น “00H” ไว้ด้วยแล้ว จะทำให้ CPU กระโดดไปทำงานที่ตำแหน่งอื่นๆที่ไม่ใช่ 0000H ซึ่งจะขึ้นอยู่กับค่าของ SBV โดยถ้าค่าของ SBV มีค่าเป็น FCH จะทำให้ CPU กลับเข้าไปทำงานใน Monitor Mode ที่ตำแหน่ง F800H แต่ถ้าค่าของ SBV เป็นค่าอื่นๆที่ไม่ใช่ FCH จะทำให้ CPU กระโดดไปทำงานยังตำแหน่งที่ชี้โดย SBV โดยค่าที่กำหนดให้ SBV จะเป็นค่าแอดเดรสไบต์สูงส่วนค่าแอดเดรสไบท์ต่ำจะมีค่าเป็น 00H เสมอ



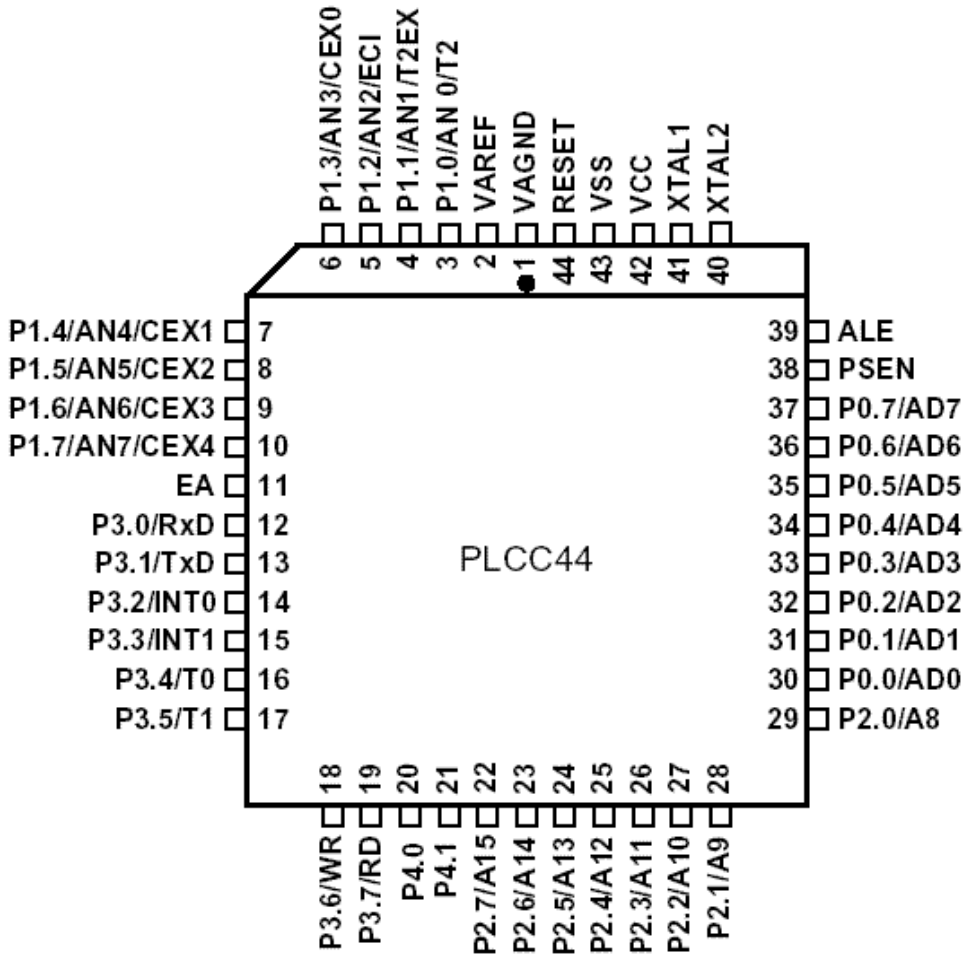
รูป แสดงวงจรของบอร์ด CP-JR51AC2 V1.0 & V1.0 EXPANSION



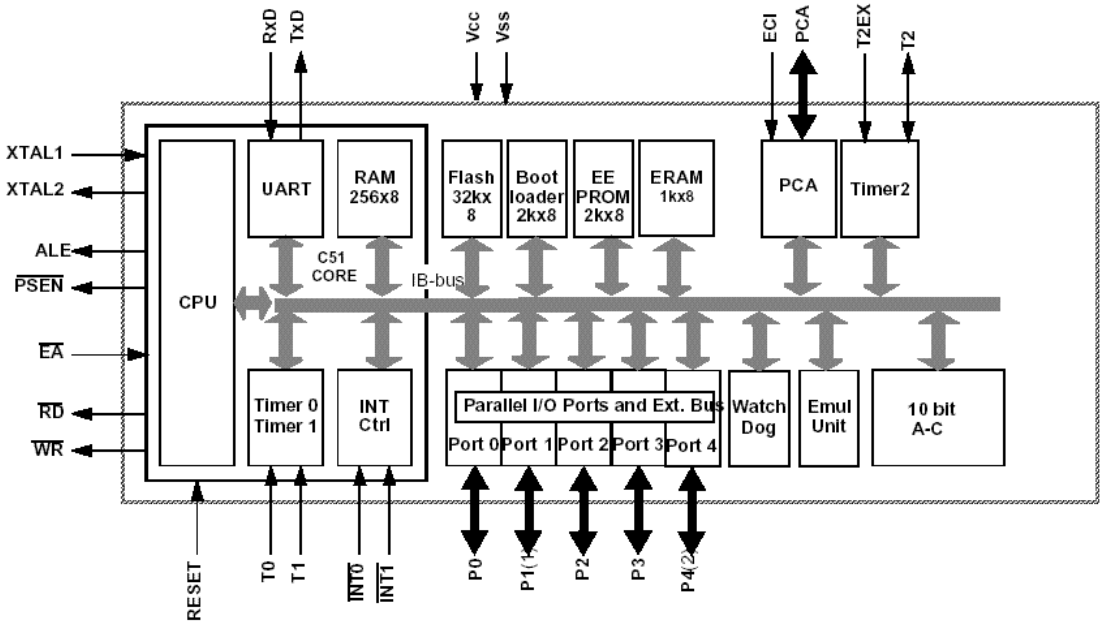
รูป แสดงวงจรของบอร์ด CP-JR51AC2 V2.0 (1/2)



รูป แสดงวงจรของบอร์ด CP-JR51AC2 V2.0 (2/2)



การจัดขาสัญญาณของ T89C51AC2 (PLCC-44)



Block Diagram ของ T89C51AC2