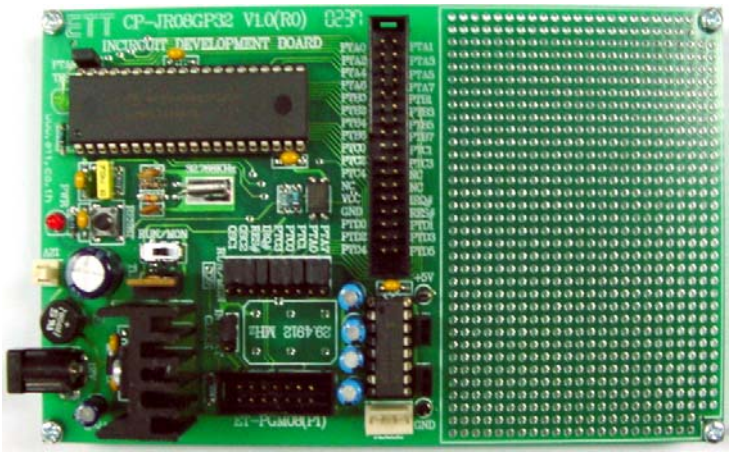
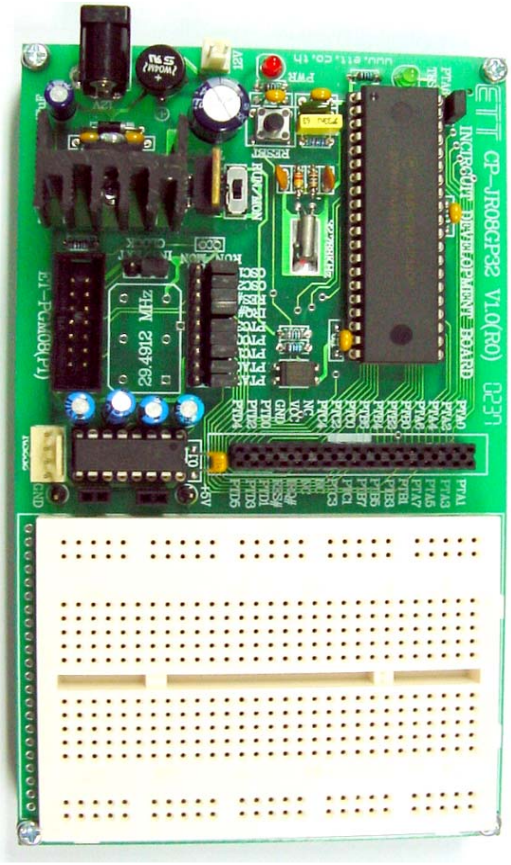
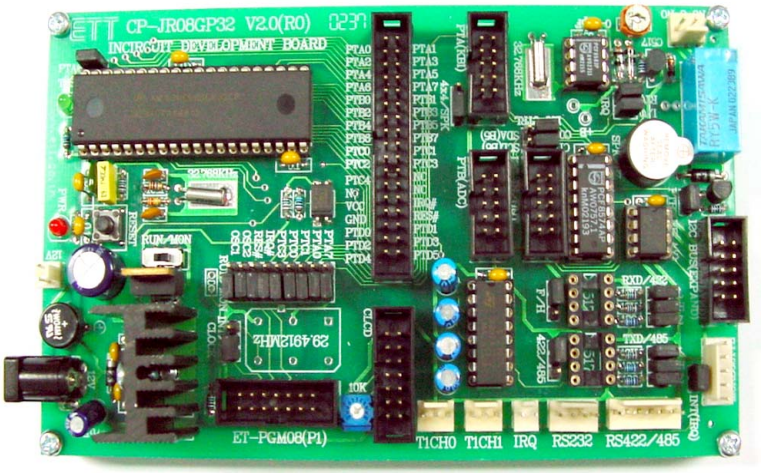
คู่มือการใช้งานบอร์ดอีทีที

MC68HC908GP32

CP-JR08GP32 V1.0
CP-JR08GP32 V1.0 EXP
CP-JR08GP32 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 “คู่มือการใช้งานบอร์ดอีทีที MC68HC908GP32”

ISBN 974-90816-9-2

ผู้เขียน นายเอกชัย มะการ

พิมพ์ครั้งที่ 1

18 พฤศจิกายน 2545

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.

คำนำ

บอร์ด CP-JR08GP32 นั้น ถูกพัฒนาขึ้น โดยได้แบ่งแยก รุ่นของบอร์ด ออกเป็น 3 รุ่น สำหรับตอบสนองต่อความต้องการของกลุ่มผู้ใช้ที่มีลักษณะของความต้องการที่แตกต่างกัน โดยทุกบอร์ดต่างก็อาศัยพื้นฐานทางโครงสร้างของวงจรที่เหมือนกัน แต่จะมีความแตกต่างกันในรายละเอียดและส่วนประกอบย่อยๆเพื่อให้คุณสมบัติของบอร์ดสามารถตอบสนองต่อความต้องการและตรงกับจุดประสงค์ของการนำไปใช้งานของผู้ใช้แต่ละกลุ่มมากที่สุด โดยลักษณะของบอร์ดนั้น จะออกแบบโครงสร้างของบอร์ดเพื่อ 3 จุดประสงค์หลัก คือ

- รุ่นแรก CP-JR08GP32 V1.0 EXPANSION สำหรับกลุ่มลูกค้าที่ต้องการนำบอร์ดไปใช้ศึกษาทดลองและเรียนรู้ทำความเข้าใจกับ CPU อย่างง่ายๆโดยมุ่งเน้นที่จะต่อทดลองวงจรส่วนของ I/O ต่างๆขึ้นมาทดลองเอง โดยใช้วิธีการต่อวงจรด้วยแผงทดลอง Photo Board เป็นหลัก
- รุ่นที่สอง คือ CP-JR08GP32 V1.0 นั้น จุดประสงค์ของการออกแบบบอร์ดเพื่อใช้ตอบสนองกลุ่มลูกค้าที่ต้องการนำบอร์ดไปใช้พัฒนางานต้นแบบ โดยมุ่งเน้นการออกแบบวงจรในส่วน I/O เอง ซึ่งภายในบอร์ดจะมีพื้นที่สำหรับต่อวงจร I/O ไว้สำหรับบัดกรีต่อวงจรเพิ่มเติม
- รุ่นที่สาม คือ CP-JR08GP32 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

พฤศจิกายน 2545

สารบัญ

****

CP-JR08GP32

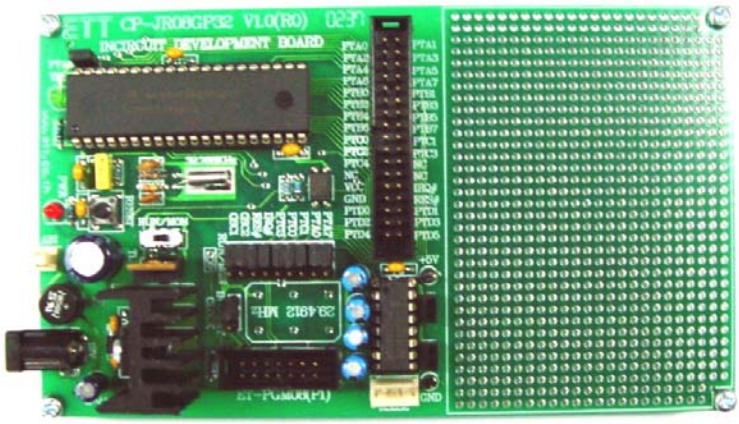
ลักษณะโดยทั่วไป

บอร์ดไมโครคอนโทรลเลอร์ในกลุ่ม CP-JR08GP32 เป็นบอร์ดไมโครคอนโทรลเลอร์ขนาดกลาง โดยเลือกใช้ CPU ไมโครคอนโทรลเลอร์ขนาด 8บิต ของ Motorola ในตระกูล MC68HC08 เบอร์ MC68HC908GP32 เป็น CPU ประจำบอร์ด ซึ่ง CPU ตัวนี้บรรจุอยู่ภายในตัวถังแบบ DIP ขนาด 40 ขา และมีทรัพยากรต่างๆบรรจุไว้ภายในตัว CPU อย่างครบถ้วน ไม่ว่าจะเป็น ADC/TIMER/COUNTER/PWM หรือ PORT I/O ต่างๆ ซึ่งมีความเหมาะสมในการนำไปประยุกต์ใช้งานในลักษณะต่างๆได้เป็นอย่างดี เนื่องจากสถาปัตยกรรมทางด้านฮาร์ดแวร์ของ CPU เบอร์นี้ จะมีความอ่อนตัวในการใช้งานได้ค่อนข้างดี กล่าวคือ ฟังก์ชั่นการทำงานต่างๆของฮาร์ดแวร์สามารถปรับเปลี่ยนการทำงานได้ด้วยโปรแกรม ดังนั้นผู้ใช้งานจึงสามารถนำระบบฮาร์ดแวร์แบบเดียวกันไปประยุกต์ใช้งานในลักษณะต่างๆกันได้โดยไม่ยากนัก โดยการปรับเปลี่ยนโปรแกรมสำหรับควบคุมการทำงานของบอร์ดเพียงเล็กน้อยเท่านั้น

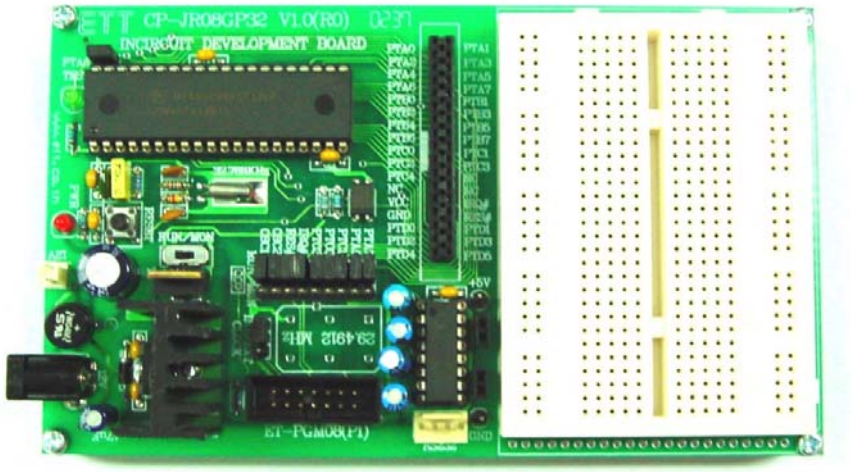
สำหรับอุปกรณ์ I/O ต่างๆ ซึ่งไม่ได้มีบรรจุไว้ในตัว CPU ด้วย ทางทีมงานอีทีที ก็ได้จัดหาและทำการออกแบบวงจรสำหรับเชื่อมต่อกับอุปกรณ์ต่างๆที่มีความจำเป็นไว้ให้ด้วยแล้ว ไม่ว่าจะเป็นจอแสดงผลแบบ LCD ระบบฐานเวลา RTC วงจร Line Driver สำหรับการสื่อสารข้อมูลอนุกรมแบบ RS232 และ RS422/485 และยังสามารถให้ผู้ใช้ทำการเพิ่มเติมอุปกรณ์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

โดยลักษณะของบอร์ดไมโครคอนโทรลเลอร์ในกลุ่ม CP-JR08GP32 นี้ จะแบ่งออกเป็น 3 รุ่น ให้ผู้ใช้ได้เลือกใช้งานกันตามความเหมาะสมดังนี้คือ

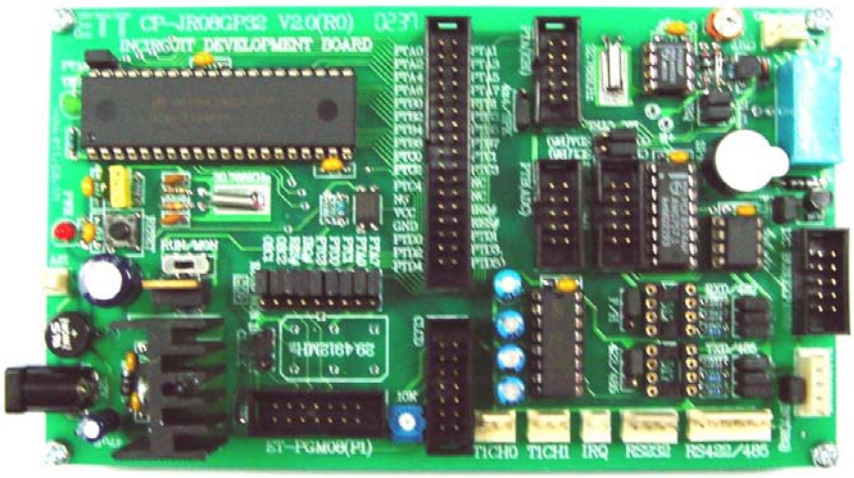
- CP-JR08GP32 V1.0 เป็นบอร์ดไมโครคอนโทรลเลอร์ ซึ่งออกแบบวงจรเฉพาะส่วนพื้นฐานที่จำเป็น เช่น แหล่งจ่ายไฟ วงจรรีเซ็ต วงจรกำเนิดความถี่สัญญาณนาฬิกา วงจรสำหรับ Download โปรแกรม และวงจรสื่อสารอนุกรม ส่วนวงจร I/O ภายนอกนั้น จะไม่ได้จัดเตรียมไว้ให้ด้วย แต่จะทำการต่อสัญญาณ I/O ต่างๆจาก CPU มาไว้ยังขั้วต่อ Connector สำหรับให้ผู้ใช้นำไปเชื่อมต่อกับอุปกรณ์ I/O ภายนอกได้โดยง่าย และยังมีพื้นที่เอนกประสงค์สำหรับให้ผู้ใช้ออกแบบวงจร I/O และต่อวงจร I/O เพิ่มเติมได้เอง เหมาะสำหรับผู้ใช้ที่ต้องการนำบอร์ดไปใช้พัฒนางานต้นแบบโดยการสร้าง I/O ต่างๆขึ้นมาใช้งานเอง
- CP-JR08GP32 V1.0 EXPANSION จะมีลักษณะเดียวกันกับบอร์ด CP-JR08GP32 V1.0 แต่จะมีแผง Photo Board สำหรับให้ผู้ใช้ต่อทดลองวงจร I/O อย่างง่ายๆได้เอง เหมาะสำหรับผู้ใช้ที่ต้องการศึกษาเรียนรู้และต้องการทดลองวงจร I/O ต่างๆ ร่วมกับ CPU อย่างง่ายๆ
- CP-JR08GP32 V2.0 เป็นบอร์ดไมโครคอนโทรลเลอร์ที่มีการออกแบบวงจรสำหรับเชื่อมต่อกับอุปกรณ์ I/O ภายนอกอื่นๆที่มีความจำเป็นไว้รองรับการใช้งานในลักษณะต่างๆ เพื่อให้ผู้ใช้งานสามารถนำบอร์ดไปใช้งานในลักษณะงานที่แตกต่างกันได้ โดยไม่ต้องดัดแปลงวงจร หรือ อาจดัดแปลงวงจรเพียงเล็กน้อยสำหรับงานบางอย่าง ซึ่งบอร์ดรุ่นนี้เหมาะสำหรับกลุ่มผู้ที่ต้องการนำบอร์ดไมโครคอนโทรลเลอร์ไปใช้งานจริงๆแต่ไม่สะดวกที่จะสร้างบอร์ดเอง



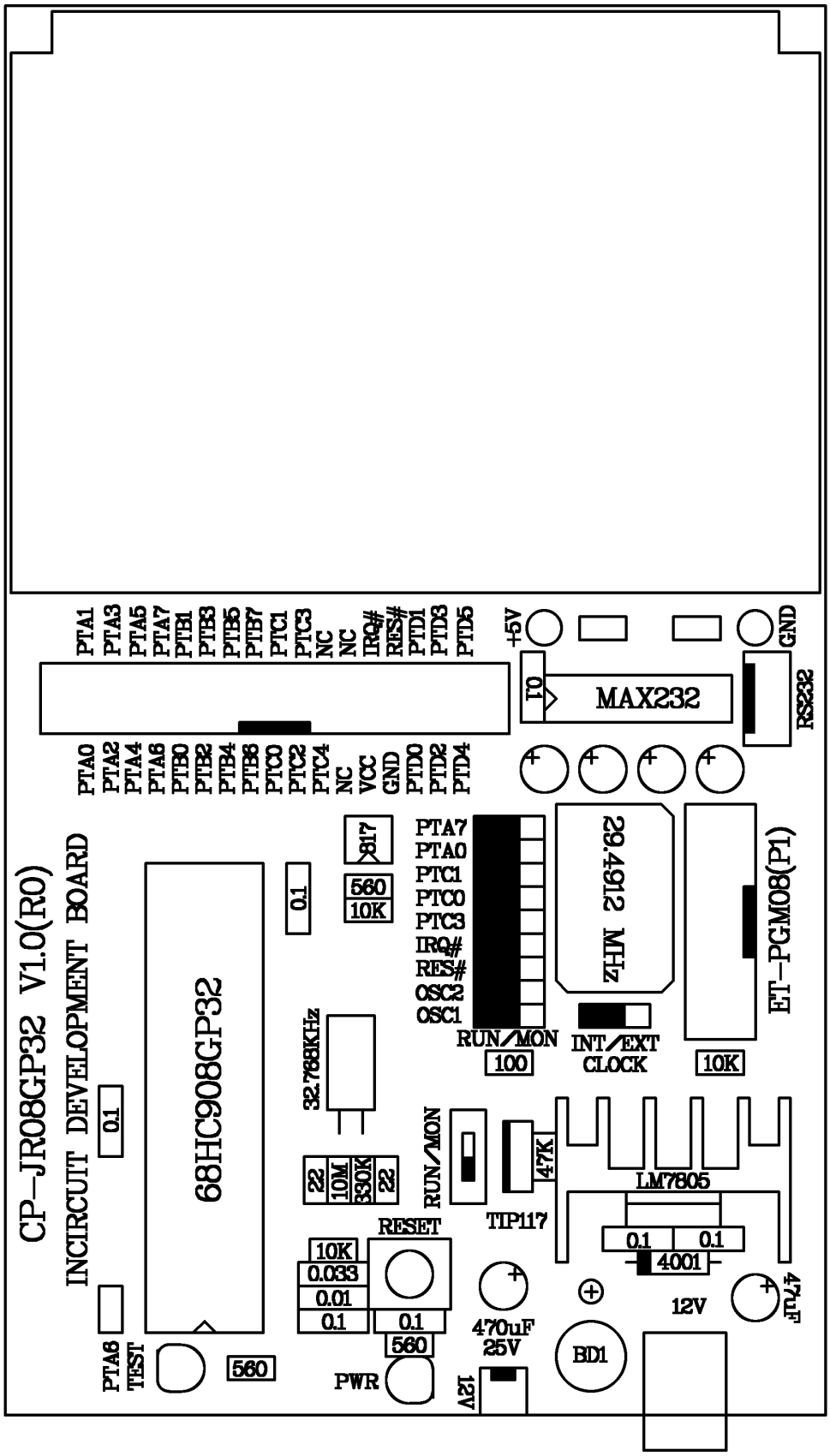
รูปแสดง ลักษณะของบอร์ด CP-JR08GP32 V1.0



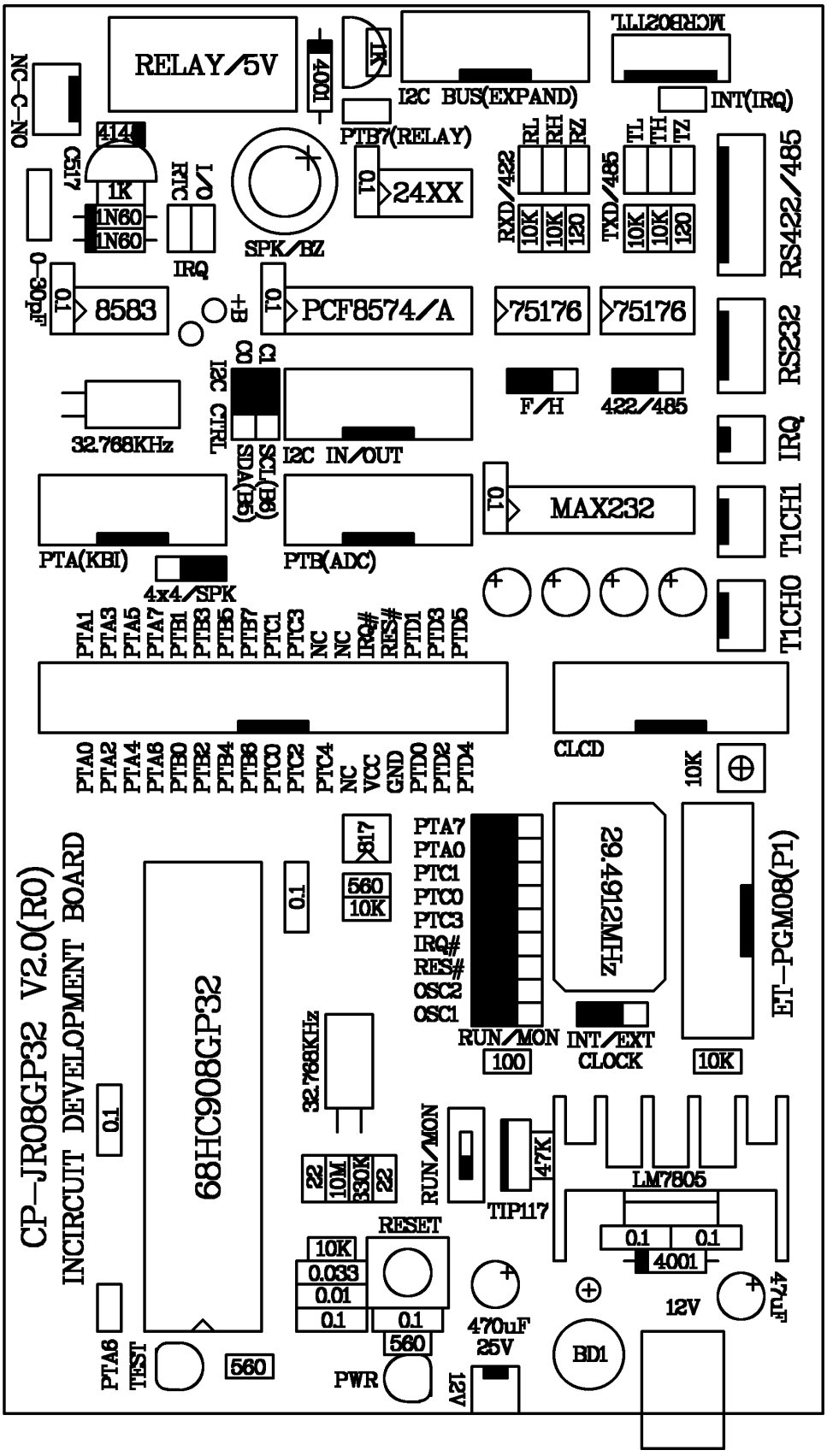
รูปแสดง ลักษณะของบอร์ด CP-JR08GP32 V1.0 EXPANSION



รูปแสดง ลักษณะของบอร์ด CP-JR08GP32 V2.0



รูปแสดง ลักษณะโครงสร้างของบอร์ด CP-JR08GP32 V1.0 และ V1.0 EXPANSION



รูปแสดง โครงสร้างของบอร์ด CP-JR08GP32 V2.0

แหล่งจ่ายไฟ (POWER SUPPLY)

สำหรับแหล่งจ่ายไฟของบอร์ดในกลุ่ม CP-JR08GP32 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

โดยแรงดันไฟจากวงจรของแหล่งจ่ายไฟที่จะป้อนให้กับวงจรต่างๆภายในบอร์ด CP-JR08GP32 นั้น จะมี SLIDE SWITCH (MON/RUN) สำหรับควบคุมการทำงานของแหล่งจ่ายไฟรวมอยู่ด้วย โดยหน้าที่การทำงานของ SLIDE SWITCH ดังกล่าว จะทำหน้าที่เลือกกำหนดการทำงานของแหล่งจ่ายไฟได้ 2 โหมดคือ

- **MONITOR MODE (MON)** เมื่อทำการเลือกตำแหน่งของ SLIDE SWITCH มาไว้ยังตำแหน่งนี้ การทำงานของแหล่งจ่ายไฟจะถูกควบคุมจากสัญญาณ "PGM-VCP" ที่ส่งมาจากเครื่องโปรแกรมรุ่น ET-PGM08 เพื่อให้แหล่งจ่ายไฟสามารถสั่งเปิดและปิดการทำงานได้อย่างถูกต้องและสัมพันธ์กับความต้องการของโปรแกรมควบคุมที่จะใช้ใน MONITOR MODE ได้อย่างถูกต้อง
- **USER MODE (RUN)** เมื่อทำการเลือกตำแหน่งของ SLIDE SWITCH มาไว้ยังตำแหน่งนี้ การทำงานของแหล่งจ่ายไฟจะไม่ผ่านวงจรควบคุมใดๆ แต่จะทำงานตลอดเวลาที่มีการจ่ายไฟให้กับวงจรซึ่งตามปรกติแล้วเมื่อจะนำบอร์ด CP-JR08GP32 ไปใช้งานจะต้องเลือกตำแหน่งการทำงานของ SLIDE SWITCH นี้มาไว้ยังตำแหน่ง RUN นี้เสมอ

สัญญาณนาฬิกา CLOCK

ความถี่ของสัญญาณนาฬิกาที่จะป้อนให้กับ CPU เบอร์ MC68HC908GP32 นั้น สามารถเลือกกำหนดแหล่งกำเนิดสัญญาณนาฬิกาได้ 2 แบบ คือ

- ใช้ค่าความถี่จากวงจรกำเนิดความถี่แบบ Oscillator จากภายนอก โดยสามารถใช้ได้กับ Oscillator ที่มีค่าความถี่ระหว่าง 0-32MHz
- ใช้ค่าความถี่จากวงจร PLL(Phase-Lock-Loop) ที่บรรจุไว้ภายในตัว CPU แล้ว โดยวิธีการนี้จะต้องป้อนค่าความถี่ Crystal 32.768KHz ให้กับขา OSC1 และ OSC2 ของ CPU ด้วย แล้วจึงทำการกำหนดค่าการคูณความถี่ให้กับวงจร PLL เพื่อสร้างความถี่ของสัญญาณนาฬิกาให้กับระบบ BUS หรือ BUS CLOCK ของ CPU จากโปรแกรมอีกต่อหนึ่ง

ซึ่งการทำงานของ CPU จะอาศัยสัญญาณนาฬิการะบบ หรือ BUS CLOCK เป็นจุดอ้างอิงการทำงานให้สัมพันธ์และสอดคล้องกับวงจรภายในอื่นๆ ซึ่ง CPU สามารถทำงานได้กับความถี่ของสัญญาณนาฬิกา BUS CLOCK ที่มีค่าความเร็วสูงสุด 8MHz โดยที่ค่าความเร็วของสัญญาณนาฬิการะบบ หรือ BUS CLOCK ของ CPU นั้น จะมีค่าเป็น ¼ ของสัญญาณนาฬิกาจาก Oscillator ภายนอก หรือในกรณีที่มีการใช้งานวงจร PLL ด้วย ค่าความถี่ของ BUS CLOCK ก็จะมีค่าเป็น ¼ ของสัญญาณนาฬิกา Output ที่ได้จากวงจร PLL เช่นกัน

ดังนั้นในกรณีที่จะใช้สัญญาณนาฬิกาจาก Oscillator ภายนอก จะต้องเลือกใช้ Oscillator ที่มีค่าความถี่อยู่ระหว่าง 0-32MHz หรือในกรณีที่จะใช้ค่าความถี่ของสัญญาณนาฬิกาจากวงจร Crystal 32.768KHz

ร่วมกับวงจร PLL นั้น ก็จะต้องทำการโปรแกรมค่าการคูณความถี่ของวงจร PLL ให้ได้ความถี่ของสัญญาณนาฬิกา Output ไม่เกิน 32MHz ด้วย

แต่สำหรับสัญญาณนาฬิกาของบอร์ด CP-JR08GP32 นั้น ในส่วนของวงจรกำเนิดความถี่ของสัญญาณนาฬิกานั้น จะสามารถเลือกกำหนด สัญญาณนาฬิกาที่จะป้อนให้กับ CPU ได้จากวงจะกำเนิดความถี่ 2 แหล่ง คือ

- ใช้สัญญาณนาฬิกาจาก Crystal 32.768KHz ภายนอก ร่วมกับ วงจร PLL(Phase-Lock-Loop) ที่บรรจุไว้ภายในตัว CPU เรียบร้อยแล้ว โดยสัญญาณนาฬิกาส่วนนี้จะใช้งานได้เมื่อเลือกโหมดการทำงานของ CPU ไว้ในโหมด RUN และเลือก JUMPER ของสัญญาณ OSC1 และ OSC2 ไว้ทางด้านตำแหน่ง “RUN” ซึ่งในการเลือกกำหนดให้บอร์ด CP-JR08GP32 ใช้สัญญาณนาฬิกา จากวงจร Crystal 32.768KHz นี้จะมีข้อดีเป็นอย่างมาก กล่าวคือ ค่าความถี่ของสัญญานาฬิกาของระบบหรือ BUS CLOCK จะสามารถปรับเปลี่ยนค่าได้จากโปรแกรม ตั้งแต่ 0-32MHz ตามความต้องการโดยไม่ต้องปรับเปลี่ยนวงจรกำเนิดความถี่จากภายนอก แต่จะใช้วิธีการโปรแกรมค่าการคูณความถี่ให้กับวงจร PLL (Phase-Lock-Loop) ให้ได้ค่าความถี่ตามต้องการ ซึ่งจะทำให้การประยุกต์ใช้งานบอร์ดมีความอ่อนตัวมากยิ่งขึ้น และข้อดีอีกประการหนึ่งก็คือ สัญญาณรบกวนที่เกิดจากวงจรกำเนิดความถี่แบบนี้จะมีค่าต่ำมากๆ แต่อย่างไรก็ตาม สำหรับในกรณีที่ต้องการให้ CPU ทำงานใน MONITOR MODE นั้น จะไม่สามารถเลือกใช้สัญญาณนาฬิกาจากวงจรกำเนิดความถี่ 32.768KHz นี้ได้ จะต้องป้อนสัญญาณนาฬิกาจากวงจร Oscillator จากภายนอกให้กับขา OSC1 ของ CPU แทน
- สัญญาณนาฬิกาจาก Oscillator โดยสัญญาณนาฬิกาส่วนนี้จะสามารถใช้งานได้ทั้งในขณะที่ CPU ทำงานใน MONITOR MODE และ RUN MODE แต่ใน MONITOR MODE นั้น จะต้องใช้ค่าความถี่ของสัญญาณนาฬิกาจาก Oscillator ที่เป็นค่ามาตรฐานตามที่ Motorola กำหนดไว้ด้วย เพื่อให้ CPU ทำการหารค่าความถี่ Baud rate ของการสื่อสารอนุกรม SCI ได้ลงตัวด้วย โดยค่าความถี่ของสัญญาณนาฬิกาจาก Oscillator นั้น จะต้องกำหนดให้มีค่าอยู่ระหว่าง 0-32MHz โดยค่าความถี่ที่เหมาะสมและสามารถหารค่าความถี่ Baudrate ของพอร์ตสื่อสารอนุกรมได้ลงตัวมากที่สุดได้แก่ 4.9152MHz,9.8304MHz,19.6608MHz หรือ 29.4912MHz เป็นต้น แต่สำหรับใน RUN MODE นั้นค่าความถี่ของสัญญาณนาฬิกาจะใช้ค่าเท่าใดก็ได้ โดยควรมีค่าความถี่อยู่ระหว่าง 0-32MHz แต่ถ้าหากว่าต้องใช้งานพอร์ตสื่อสารอนุกรมด้วยแล้ว ก็ต้องพิจารณาค่าความถี่ของสัญญาณนาฬิกา Oscillator ให้เหมาะสมด้วยเช่นกัน โดยค่าความถี่จาก Oscillator นี้ยังสามารถเลือกกำหนดได้จาก 2 แหล่ง โดยเลือกจาก JUMPER CLOCK (INT/EXT) คือ ใช้ Oscillator จากภายในบอร์ดเอง (CLOCK = INT) โดยการเลือกติดตั้ง Oscillator ไว้ภายในบอร์ด หรือเลือกใช้ Oscillator จากภายนอก (CLOCK = EXT) ซึ่งในกรณีนี้จะเป็นการเลือกใช้สัญญาณนาฬิกาที่ส่งมาจากเครื่องโปรแกรมรุ่น ET-PGM08 ซึ่งมีค่าความถี่เป็น 4.9152MHz

การโปรแกรมค่าความถี่ของ Phase Lock Loop

ในกรณีที่ผู้ใช้เลือกสัญญาณนาฬิกาของบอร์ดจากตัวกำเนิดความถี่แบบ Oscillator ภายนอกนั้น คงไม่มีปัญหาอะไร สามารถใช้งานได้ทันที โดยค่าความถี่ของสัญญาณนาฬิกาของบอร์ดใน User Mode หรือค่าความถี่ของสัญญาณนาฬิกาที่จะป้อนให้กับระบบบัสของ CPU ก็จะมีค่าเป็น ¼ ของความถี่จาก Oscillator ที่ป้อนให้กับบอร์ดโดยอัตโนมัติอยู่แล้ว แต่สำหรับในกรณีที่ผู้ใช้เลือกใช้ความถี่จากสัญญาณนาฬิกา Crystal ค่า 32.768KHz ร่วมกับวงจร Phase Lock Loop ภายในตัว CPU นั้น จะต้องมีการเขียนโปรแกรมสำหรับกำหนดค่าให้กับรีจิสเตอร์ต่างๆของวงจร Phase Lock Loop ด้วย เพื่อให้วงจร Phase Lock Loop สร้างสัญญาณความถี่ของสัญญาณนาฬิกาที่จะป้อนให้กับระบบบัสของ CPU ตามต้องการ โดยค่าความถี่ของสัญญาณนาฬิกาของระบบบัสจะมีค่าเป็น ¼ ของความถี่ที่ได้จากวงจร Phase Lock Loop ด้วย โดยในการโปรแกรมค่าความถี่ให้กับ Phase Lock Loop นั้น จะต้องกำหนดค่าให้กับรีจิสเตอร์ต่างๆดังนี้

- กำหนดค่าให้กับบิต PRE1:PRE0 ของรีจิสเตอร์ PCTL(Phase Lock Loop Control Register)
- กำหนดค่าให้กับบิต VPR1:VPR0 ของรีจิสเตอร์ PCTL(Phase Lock Loop Control Register)
- กำหนดค่าการคูณความถี่ของ PLL ให้กับรีจิสเตอร์ PMSH:PMSL
- กำหนดย่านความถี่ให้กับรีจิสเตอร์ PMRS (PLL VCO Range Select Register)
- กำหนดค่าการหารให้กับรีจิสเตอร์ PMDS (PLL Reference Divide Select Register)

โดยค่าที่จะกำหนดให้กับบิตต่างๆของรีจิสเตอร์นั้น ตามปรกติแล้วจะต้องทำการคำนวณเอาเองทั้งสิ้น โดยรายละเอียดต่างๆสามารถศึกษาเพิ่มเติมได้จาก Data Sheet ของ CPU แต่ในที่นี้จะขอสรุปเป็นค่ามาตรฐานที่มีการใช้งานกันบ่อยๆโดยอ้างอิงจากการคำนวณโดยใช้ค่า Crystal 32.768KHz เป็นหลัก

ค่าความถี่ของสัญญาณนาฬิกา (F_{BUS})	ค่าพารามิเตอร์สำหรับกำหนดความถี่ของ Phase Lock Loop				
	R (PMDS)	N (PMSH:SL)	P (บิต PRE)	E (บิต VPR)	L (PLL VCO)
2.00 MHz	1	00F5H	0	0	D1H
2.4576 MHz	1	012CH	0	1	80H
2.50 MHz	1	0132H	0	1	83H
4.00 MHz	1	01E9H	0	1	D1H
4.9152 MHz	1	0258H	0	2	80H
5.00 MHz	1	0263H	0	2	82H
7.3728 MHz	1	0384H	0	2	C0H
8.00 MHz	1	03D1H	0	2	D0H

ตารางแสดงการกำหนดค่าความถี่สำหรับวงจร Phase Lock Loop โดยใช้ Crystal 32.768KHz

สำหรับการเขียนโปรแกรมเพื่อกำหนดค่าพารามิเตอร์ต่างๆให้กับรีจิสเตอร์ที่เกี่ยวข้องให้กับวงจร Phase Lock Loop เพื่อให้ได้ค่าความถี่ของสัญญาณนาฬิกาตามต้องการนั้นสามารถแสดงให้เห็นได้ดังตัวอย่างโปรแกรมต่อไปนี้ โดยโปรแกรมส่วนนี้ควรกำหนดไว้ในส่วนเริ่มต้นการทำงานของโปรแกรมเสมอ เพื่อให้วงจร Phase Lock Loop เริ่มทำงานทันทีเมื่อ CPU พ้นจากสภาวะรีเซ็ตในแต่ละครั้ง

```
;/*******************************/;
;/* Initial  PLL  For Clock Bus */;
;/* External X-TAL = 32.768 KHz */;
;/* Clock Bus Freq = 7.3728 MHz */;
;/*******************************/;
;
BCLR      BCS,PCTL        ; Select External Clock
BCLR      PLLON,PCTL      ; Turn-off PLL
BCLR      PRE1,PCTL       ; P = 0
BCLR      PRE0,PCTL
BSET      VPR1,PCTL       ; E = 2
BCLR      VPR0,PCTL
MOV       #$01,PMDS       ; R = $01
MOV       #$03,PMSH       ; N = $0384
MOV       #$84,PMSL
MOV       #$C0,PMRS       ; L = $C0
LDA       PCTL            ; Clear Status
BSET      AUTO,PBWC       ; Auto Bandwidth Control
BSET      PLLON,PCTL      ; Turn-on PLL
BRCLR     LOCK,PBWC,*     ; Wait PLL Lock Complete
BSET      BCS,PCTL        ; Used PLL O/P = Base Clock
;
ส่วนอื่นๆ ของโปรแกรมตามปรกติ
```

แสดง ตัวอย่างโปรแกรม สำหรับกำหนดค่าความถี่ของ Phase Lock Loop **เป็น** 7.3728MHz

ในกรณีที่ผู้ใช้ต้องการเปลี่ยนค่าความถี่เป็นค่าอื่นๆ ก็สามารถทำการแทนค่าต่างๆที่เกี่ยวข้องทั้ง 5 ค่า คือ P,E,N,R และ L ซึ่งแสดงให้เห็นในตารางที่ผ่านมาแล้วได้ หรือในกรณีที่เลือกใช้ค่าความถี่ของสัญญาณนาฬิกาจากภายนอกเพียงอย่างเดียวก็สามารถตัดโปรแกรมส่วนนี้ออกไปได้ ซึ่งตามปรกติแล้วค่าความถี่ของสัญญาณนาฬิกาของระบบบัสนั้น หลังการรีเซ็ตทุกๆครั้งระบบฮาร์ดแวร์ของ CPU จะทำการกำหนดให้สัญญาณนาฬิกาของระบบส่งตรงมาจากความถี่ของสัญญาณนาฬิกาจากภายนอกเสมอ ซึ่งถ้าใช้ Crystal 32.768KHz แล้วไม่มีการเขียนโปรแกรมสำหรับสั่งให้วงจร Phase Lock Loop ทำงานดังกล่าวแล้ว ค่าความถี่ของสัญญาณนาฬิการะบบหรือ Bus Clock ก็จะมีค่าเป็น 8.192KHz (¼ ของ 32.768KHz) เท่านั้น

หมายเหตุ เมื่อเลือกโหมดการทำงานของบอร์ดเป็น Monitor Mode นั้น จะไม่สามารถใช้งานวงจร Phase Lock Loop ในการสร้างความถี่ของสัญญาณนาฬิกาได้ ซึ่งใน Monitor Mode จะต้องใช้ความถี่ของสัญญาณนาฬิกาจากภายนอกเพียงอย่างเดียวเท่านั้น

โหมดการทำงานของบอร์ด

การทำงานของบอร์ด CP-JR08GP32 นั้น สามารถกำหนดโหมดการทำงานของบอร์ดได้ 2 โหมดการทำงานด้วยกัน คือ MONITOR MODE (MON) และโหมดการทำงานปรกติ USER MODE (RUN)

การทำงานใน MONITOR MODE

ในโหมดนี้จะใช้สำหรับในกรณีที่ต้องการพัฒนาโปรแกรมของบอร์ด โดยต่อร่วมกับเครื่องพัฒนาโปรแกรมรุ่น ET-PGM08 ซึ่งในโหมดนี้ผู้ใช้สามารถสั่งจัดการหน่วยความจำ FLASH ภายในตัวของ CPU ได้โดยตรงไม่ว่าจะเป็นการสั่งลบข้อมูล หรือเขียนข้อมูลใหม่ให้กับหน่วยความจำ FLASH ของ CPU นอกจากนี้แล้วยังสามารถสั่งทำการตรวจสอบการทำงานของโปรแกรมที่เขียนขึ้น ในลักษณะต่างๆได้หลายรูปแบบ ไม่ว่าจะเป็น In-circuit Simulator หรือ In-circuit Debugger โดยในบอร์ด CP-JR08GP32 นั้น จะใช้วิธีการเลือกโหมดการทำงานของ CPU ให้เข้าทำงานใน MONITOR MODE ด้วยวิธีการป้อนแรงดันไฟสูงให้กับขา IRQ เพื่อเลือกโหมดการทำงานของ CPU ให้เข้าทำงานใน MONITOR MODE โดยไม่มีเงื่อนไขและข้อจำกัดเหมือนวิธีอื่นๆ

ข้อจำกัดของการใช้งานบอร์ดใน MONITOR MODE

1. สัญญาณ PTA0,PTA7,PTC0,PTC1,PTC3,IRQ และ RESET จะถูกใช้งานสำหรับกำหนดเงื่อนไขในการทำงานของ CPU ใน MONITOR โหมด ดังนั้นผู้ใช้จึงไม่สามารถใช้งานขาสัญญาณดังกล่าวรวมทั้งฟังก์ชั่นพิเศษอื่นๆที่เกี่ยวข้องกับขาสัญญาณดังกล่าวข้างต้นได้ เช่น ไม่สามารถใช้การ Interrupt จากขาสัญญาณ IRQ ได้ ไม่สามารถสั่งงานขาสัญญาณ PTA0,PTA7,PTC0,PTC1 และ PTC3 รวมทั้งฟังก์ชั่นอื่นๆที่เกี่ยวข้อง เช่น การสั่งเปลี่ยนแปลงทิศทาง(Direction) ของสัญญาณ เป็นต้น
2. วงจรกำเนิดสัญญาณนาฬิกาภายในตัวของ CPU จะไม่ทำงาน ดังนั้น ในการทำงานใน MONITOR MODE นี้จะต้องใช้วงจรกำเนิดสัญญาณนาฬิกาจากภายนอกป้อนให้กับขา OSC1 ของ CPU ด้วย โดยในบอร์ด CP-JR08GP32 จะสามารถเลือกใช้สัญญาณนาฬิกาสำหรับการทำงานใน MONITOR MODE ได้ 2 แหล่ง คือ เลือกใช้สัญญาณนาฬิกาจากภายนอก (CLOCK = EXT) โดยส่งผ่านมาจากตัวเครื่อง ET-PGM08 ซึ่งมีความถี่ 4.9152MHz หรือเลือกใช้สัญญาณนาฬิกาจากภายในตัวบอร์ดเอง (CLOCK = INT) ซึ่งผู้ใช้จะต้องทำการติดตั้งตัวกำเนิดความถี่แบบ Oscillator ให้กับบอร์ดเพิ่มเติมด้วย โดยถ้าต้องการความเร็วในการทำงานให้มีประสิทธิภาพสูงสุดควรใช้ Oscillator ค่าความถี่ 29.4912MHz จะดีที่สุด
3. ความเร็วในการทำงานของบอร์ดจะไม่ใช่ความเร็วจริงเหมือนการทำงานใน RUN MODE ถ้าใช้งานบอร์ดแบบ In-circuit Simulator แต่ถ้าหากต้องการความเร็วในการทำงานเท่ากับความเร็วจริง จะต้องเลือกใช้งานบอร์ดแบบ In-circuit Debugger แทน

สำหรับวิธีการกำหนดโหมดการทำงานของบอร์ด CP-JR08GP32 ให้พร้อมทำงานใน MONITOR MODE นั้นสามารถทำได้ดังนี้

- เลือกตำแหน่ง SLIDE SWITCH สำหรับควบคุมการทำงานของแหล่งจ่ายไฟไว้ทางด้าน MON
- เลือก JUMPER ของสัญญาณต่างๆที่เกี่ยวข้องกับการเลือกกำหนดโหมดการทำงานของบอร์ดซึ่งได้แก่ PTA0,PTA7,PTC0,PTC1,PTC3,IRQ,RES,OSC2,OSC1 มาไว้ทางด้านตำแหน่ง MON
- จ่ายไฟเลี้ยงวงจรให้กับบอร์ด CP-JR08GP32 แต่วงจร POWER SUPPLY อาจจะยังไม่สามารถทำงานได้ในตอนนี้เนื่องจากการทำงานของวงจรจะต้องถูกควบคุมจากโปรแกรมผ่านทางเครื่องโปรแกรมรุ่น ET-PGM08 อีกต่อหนึ่ง
- ต่อสายแพร์ขนาด 14PIN ระหว่างขั้วต่อ ET-PGM08(P1) ของบอร์ด CP-JR08GP32 และขั้วต่อ P1 ของเครื่อง ET-PGM08 เข้าด้วยกัน
- ต่อสายสัญญาณ RS232 จากเครื่องคอมพิวเตอร์ PC เข้ากับเครื่อง ET-PGM08
- จ่ายไฟเลี้ยงวงจรให้กับเครื่องโปรแกรม ET-PGM08
- สั่ง RUN โปรแกรมสำหรับสั่งงาน CPU ใน MONITOR MODE เช่น PROG08SZ.EXE หรือ DATABLAZE.EXE หรือ ICS08GPZ.EXE หรือ ICD08SZ.EXE เป็นต้น

การทำงานใน USER MODE หรือ RUN MODE

การทำงานในโหมดนี้เป็นโหมดการทำงานปรกติของบอร์ด โดยจะใช้สำหรับในกรณีที่ผู้ใช้ทำการโปรแกรมข้อมูลให้กับหน่วยความจำ FLASH ของ CPU เรียบร้อยแล้ว ซึ่งในโหมดการทำงานนี้ สามารถจะใช้งานทรัพยากรต่างๆของ CPU ได้อย่างครบถ้วนโดยไม่มีข้อจำกัดใดๆ โดยวิธีการกำหนดโหมดการทำงานของบอร์ด CP-JR08GP32 เป็น RUN MODE นี้สามารถทำได้โดย

- เลือกตำแหน่ง SLIDE SWITCH สำหรับควบคุมการทำงานของแหล่งจ่ายไฟไว้ทางด้าน RUN
- เลือกกำหนด JUMPER ของสัญญาณต่างๆ ที่ใช้สำหรับเลือกกำหนดโหมดการทำงานของบอร์ด เช่น PTA0,PTA7,PTC0,PTC1,PTC3,IRQ,RES,OSC2,OSC1 มาไว้ทางด้านตำแหน่ง RUN
- จ่ายไฟเลี้ยงวงจรให้กับบอร์ด CP-JR08GP32 ซึ่งจะสังเกตุเห็น LED แสดงผล PWR สีเดงติดสว่างให้เห็นตลอดเวลา

ซึ่งหลังจากทำการเปลี่ยนโหมดการทำงานของบอร์ดมายังโหมดนี้แล้ว เมื่อทำการจ่ายไฟให้กับบอร์ดหรือทำการรีเซ็ตบอร์ด CPU ก็จะเริ่มทำงานตามโปรแกรมที่บรรจุไว้ในหน่วยความจำของ CPU ในทันที

การจัดสรร I/O ของบอร์ด CP-JR08GP32 V2.0

บอร์ด CP-JR08GP32 V2.0 จะใช้ CPU เบอร์ MC68HC908GP32 เป็น CPU ประจำบอร์ด โดยตัว CPU เบอร์นี้จะมีขาสัญญาณที่สามารถนำมาใช้งานเป็น I/O Port ได้ทั้งหมด 30 เส้นสัญญาณ ประกอบด้วย

- PTA0-PTA7 จำนวน 8 เส้นสัญญาณ
- PTB0-PTB7 จำนวน 8 เส้นสัญญาณ
- PTC0-PTC4 จำนวน 5 เส้นสัญญาณ
- PTD0-PTD5 จำนวน 6 เส้นสัญญาณ
- PTE0-PTE1 จำนวน 2 เส้นสัญญาณ
- IRQ จำนวน 1 เส้นสัญญาณ

โดยการออกแบบวงจรของบอร์ด CP-JR08GP32 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-JR08GP32 V2.0 นั้น สามารถสรุปได้ดังต่อไปนี้

PTA0 สำหรับขาสัญญาณนี้เมื่อเลือกการทำงานของบอร์ดไว้ใน Monitor Mode ขาสัญญาณ PTA0 นี้จะถูกใช้ในการติดต่อสื่อสารกับเครื่องโปรแกรม ET-PGM08 ดังนั้นใน Monitor Mode ขาสัญญาณ PTA0 และฟังก์ชั่นต่างๆที่เกี่ยวข้องกับ PTA0 จะไม่สามารถใช้งานได้ และในคำสั่งของโปรแกรมที่จะนำมา RUN ใน Monitor Mode ก็ต้องไม่ไปเปลี่ยนแปลงค่าต่างๆที่เกี่ยวข้องกับสัญญาณ PTA0 ด้วย เช่น

- บิต PTA0 ในรีจิสเตอร์ PTA
- บิต DDRA0 ในรีจิสเตอร์ DDRA
- บิต PTAPUE0 ในรีจิสเตอร์ PTAPUE
- บิต KBIE0 ในรีจิสเตอร์ INTKBIER

ส่วนใน User Mode นั้น จะสามารถใช้งานสัญญาณ PTA0 ได้ตามต้องการ โดยสัญญาณ PTA0 ใน User Mode จะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN และขั้วต่อ PTA(KBI) ไว้ให้ผู้ใช้เลือกใช้งานตามต้องการ

PTA1-PTA6 สำหรับขาสัญญาณเหล่านี้จะสามารถใช้งานได้ทั้งใน Monitor Mode และ User Mode โดยขาสัญญาณจะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN และขั้วต่อ PTA(KBI) ไว้ แต่ขา PTA6 นั้นจะถูกเชื่อมต่อไปยัง Jumper (PTA6) สำหรับควบคุมการแสดงผลของหลอดแสดงผล TEST ด้วย

PTA7 สำหรับขาสัญญาณนี้เมื่อเลือกการทำงานของบอร์ดไว้ใน Monitor Mode ขาสัญญาณ PTA7 นี้จะถูกใช้ในการกำหนดโหมดการทำงานของ CPU ดังนั้นใน Monitor Mode ขาสัญญาณ PTA7 และฟังก์ชั่นต่างๆที่เกี่ยวข้องกับ PTA7 จะไม่สามารถใช้งานได้ และในคำสั่งของโปรแกรมที่จะนำมา RUN ใน Monitor Mode ก็ต้องไม่ไปเปลี่ยนแปลงค่าต่างๆที่เกี่ยวข้องกับสัญญาณ PTA7 ด้วย เช่น

- บิต PTA7 ในรีจิสเตอร์ PTA

- บิต DDRA7 ในรีจิสเตอร์ DDRA
- บิต PTAPUE7 ในรีจิสเตอร์ PTAPUE
- บิต KBIE7 ในรีจิสเตอร์ INTKBIER

ส่วนใน User Mode นั้น จะสามารถใช้งานสัญญาณ PTA7 ได้ตามต้องการ โดยสัญญาณ PTA7 ใน User Mode จะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN และ Jumper (4x4 / SPK) เพื่อใช้เลือกว่าจะนำสัญญาณ PTA7 ไปต่อไว้ยังขั้วต่อ PTA(KBI) สำหรับ Scan Keyboard ขนาด 4x4 หรือจะนำไปควบคุมการทำงานของลำโพงโดยถ้าเลือก Jumper (4x4 / SPK) ไว้ทางด้าน SPK ขาสัญญาณ PTA7 จะถูกเชื่อมต่อไปควบคุมการกำเนิดเสียงของลำโพงแต่ถ้าเลือก Jumper (4x4 / SPK) ไว้ทางด้าน 4x4 ขาสัญญาณ PTA7 จะถูกเชื่อมต่อไปรอไว้ยังขั้วต่อ PTA(KBI) สำหรับนำไปใช้เป็น I/O หรือ Scan Keyboard ขนาด 4x4 ได้

PTB0-PTB4 สำหรับขาสัญญาณเหล่านี้จะสามารถใช้งานได้ทั้งใน Monitor Mode และ User Mode โดยขาสัญญาณจะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN และขั้วต่อ PTB(ADC) ไว้ให้เลือกใช้งานตามต้องการ

PTB5 สำหรับขาสัญญาณ PTB5 จะสามารถใช้งานได้ทั้งใน Monitor Mode และ User Mode โดยขาสัญญาณจะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN และขั้วต่อ PTB(ADC) ไว้ และนอกจากนี้แล้วยังถูกจัดสรรไปต่อยัง Jumper (SDA) เพื่อเลือกไปทำหน้าที่เป็นสัญญาณ SDA ในการติดต่อกับอุปกรณ์ I^2C Bus ต่างๆ อีกด้วย

PTB6 สำหรับขาสัญญาณ PTB6 จะสามารถใช้งานได้ทั้งใน Monitor Mode และ User Mode โดยขาสัญญาณจะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN และขั้วต่อ PTB(ADC) ไว้ และนอกจากนี้แล้วยังถูกจัดสรรไปต่อยัง Jumper (SCL) เพื่อเลือกไปทำหน้าที่เป็นสัญญาณ SCL ในการติดต่อกับอุปกรณ์ I^2C Bus ต่างๆ อีกด้วย

PTB7 สำหรับขาสัญญาณ PTB7 จะสามารถใช้งานได้ทั้งใน Monitor Mode และ User Mode โดยขาสัญญาณจะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN และขั้วต่อ PTB(ADC) ไว้ และนอกจากนี้แล้วยังถูกจัดสรรไปต่อยัง Jumper (PTB7) สำหรับใช้ควบคุมการทำงานของ RELAY อีกด้วย

PTC0 สำหรับขาสัญญาณนี้เมื่อเลือกการทำงานของบอร์ดไว้ใน Monitor Mode ขาสัญญาณ PTC0 นี้จะถูกใช้ในการกำหนดโหมดการทำงานของ CPU ดังนั้นใน Monitor Mode ขาสัญญาณ PTC0 และฟังก์ชั่นต่างๆที่เกี่ยวข้องกับ PTC0 จะไม่สามารถใช้งานได้ และในคำสั่งของโปรแกรมที่จะนำมา RUN ใน Monitor Mode ก็ต้องไม่ไปเปลี่ยนแปลงค่าต่างๆที่เกี่ยวข้องกับสัญญาณ PTC0 ด้วย เช่น

- บิต PTC0 ในรีจิสเตอร์ PTC
- บิต DDRC0 ในรีจิสเตอร์ DDRC
- บิต PTCPUE0 ในรีจิสเตอร์ PTCPUE

ส่วนใน User Mode นั้น จะสามารถใช้งานสัญญาณ PTC0 ได้ตามต้องการ โดยสัญญาณ PTC0 ใน User Mode จะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN ไว้ นอกจากนี้แล้ว PTC0 ยังถูกจัดสรรไปต่อไว้ยัง Jumper (SDA) เพื่อเลือกไปทำหน้าที่เป็นสัญญาณ SDA ในการติดต่อกับอุปกรณ์ I^2C Bus ต่างๆ อีกด้วย

PTC1 สำหรับขาสัญญาณนี้เมื่อเลือกการทำงานของบอร์ดไว้ใน Monitor Mode ขาสัญญาณ PTC1 นี้จะถูกใช้ในการกำหนดโหมดการทำงานของ CPU ดังนั้นใน Monitor Mode ขาสัญญาณ PTC1 และฟังก์ชั่น

ต่างๆที่เกี่ยวข้องกับ PTC1 จะไม่สามารถใช้งานได้ และในคำสั่งของโปรแกรมที่จะนำมา RUN ใน Monitor Mode ก็ต้องไม่ไปเปลี่ยนแปลงค่าต่างๆที่เกี่ยวข้องกับสัญญาณ PTC1 ด้วย เช่น

- บิต PTC1 ในรีจิสเตอร์ PTC
- บิต DDRC1 ในรีจิสเตอร์ DDRC
- บิต PTCPUE1 ในรีจิสเตอร์ PTCPUE

ส่วนใน User Mode นั้น จะสามารถใช้งานสัญญาณ PTC1 ได้ตามต้องการ โดยสัญญาณ PTC1 ใน User Mode จะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN ไว้ นอกจากนี้แล้ว PTC1 ยังถูกจัดสรรไปต่อไว้ยัง Jumper (SCL) เพื่อเลือกไปทำหน้าที่เป็นสัญญาณ SCL ในการติดต่อกับอุปกรณ์ I^2C Bus ต่างๆ อีกด้วย

PTC2 สำหรับขาสัญญาณ PTC2 จะสามารถใช้งานได้ทั้งใน Monitor Mode และ User Mode โดยขาสัญญาณจะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN ไว้ นอกจากนี้แล้วยังถูกจัดสรรไปต่อยังขั้วต่อของ จอแสดงผล LCD เพื่อทำหน้าที่เป็นขาสัญญาณ EN ในการควบคุมการทำงานของ LCD ด้วย

PTC3 สำหรับขาสัญญาณนี้เมื่อเลือกการทำงานของบอร์ดไว้ใน Monitor Mode ขาสัญญาณ PTC3 นี้จะถูกใช้ในการกำหนดโหมดการทำงานของ CPU ดังนั้นใน Monitor Mode ขาสัญญาณ PTC3 และฟังก์ชั่นต่างๆที่เกี่ยวข้องกับ PTC3 จะไม่สามารถใช้งานได้ และในคำสั่งของโปรแกรมที่จะนำมา RUN ใน Monitor Mode ก็ต้องไม่ไปเปลี่ยนแปลงค่าต่างๆที่เกี่ยวข้องกับสัญญาณ PTC3 ด้วย เช่น

- บิต PTC3 ในรีจิสเตอร์ PTC
- บิต DDRC3 ในรีจิสเตอร์ DDRC
- บิต PTCPUE3 ในรีจิสเตอร์ PTCPUE

ส่วนใน User Mode นั้น จะสามารถใช้งานสัญญาณ PTC3 ได้ตามต้องการ โดยสัญญาณ PTC3 ใน User Mode จะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN ไว้ให้ใช้งาน นอกจากนี้แล้ว PTC3 ยังถูกจัดสรรออกไปต่อไว้ยัง Jumper (422/485) เพื่อเลือกไปทำหน้าที่เป็นสัญญาณ ควบคุมทิศทางการรับส่งข้อมูลของ RS485 ด้วย

PTC4 สำหรับขาสัญญาณ PTC4 จะสามารถใช้งานได้ทั้งใน Monitor Mode และ User Mode โดยขาสัญญาณจะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN ไว้ นอกจากนี้แล้วยังถูกจัดสรรไปต่อยังขั้วต่อของ จอแสดงผล LCD เพื่อทำหน้าที่เป็นขาสัญญาณ RS ในการควบคุมการทำงานของ LCD ด้วย

PTD0-PTD3 สำหรับขาสัญญาณเหล่านี้ จะสามารถใช้งานได้ทั้งใน Monitor Mode และ User Mode โดยขาสัญญาณจะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN ไว้ นอกจากนี้แล้วยังถูกจัดสรรไปต่อยังขั้วต่อของ จอแสดงผล LCD เพื่อทำหน้าที่เป็นขาสัญญาณ DATA ในการควบคุมการทำงานของ LCD ด้วย

PTD4 สำหรับขาสัญญาณ PTD4 จะสามารถใช้งานได้ทั้งใน Monitor Mode และ User Mode โดยขาสัญญาณจะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN ไว้ นอกจากนี้แล้วยังถูกจัดสรรไปต่อยังขั้วต่อของ T1CH0 สำหรับสร้างสัญญาณ PWM หรือใช้เป็น Input/Out ของ Timer1 ช่อง0 และนอกจากนี้แล้ว PTD4 นี้ยังถูกจัดสรรไปยังขั้วต่อสำหรับเชื่อมต่อกับเครื่องอ่านบัตรแถบแม่เหล็กรุ่น “MCR-B02TTL” โดยทำหน้าที่เป็นสัญญาณ DATA ในการอ่านข้อมูลจากบัตรแถบแม่เหล็กอีกด้วย

PTD5 สำหรับขาสัญญาณ PTD5 จะสามารถใช้งานได้ทั้งใน Monitor Mode และ User Mode โดยขาสัญญาณจะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN ไว้ นอกจากนี้แล้วยังถูกจัดสรรไปต่อยังขั้วต่อของ T1CH1 สำหรับสร้างสัญญาณ PWM หรือใช้เป็น Input/Out ของ Timer1 ช่อง1 และนอกจากนี้แล้ว PTD5 นี้ยังถูกจัดสรรไปยังขั้วต่อสำหรับเชื่อมต่อกับเครื่องอ่านบัตรแถบแม่เหล็กรุ่น “MCR-B02TTL” โดยทำหน้าที่เป็นสัญญาณ CLOCK ในการอ่านข้อมูลจากบัตรแถบแม่เหล็กอีกด้วย

PTE0 สำหรับขาสัญญาณ PTE0 จะสามารถใช้งานได้ทั้งใน Monitor Mode และ User Mode โดยขาสัญญาณนี้จะถูกใช้งานเป็นขา TXD ของพอร์ตสื่อสารอนุกรมทั้งแบบ RS232 และ RS422/485 ด้วย

PTE1 สำหรับขาสัญญาณ PTE1 จะสามารถใช้งานได้ทั้งใน Monitor Mode และ User Mode โดยขาสัญญาณนี้จะถูกใช้งานเป็นขา RXD ของพอร์ตสื่อสารอนุกรมทั้งแบบ RS232 และ RS422/485 ด้วย

IRQ สำหรับขาสัญญาณนี้เมื่อเลือกการทำงานของบอร์ดไว้ใน Monitor Mode ขาสัญญาณ IRQ นี้จะถูกใช้ในการเลือกโหมดการทำงานของ CPU ดังนั้นใน Monitor Mode ขาสัญญาณ IRQ และฟังก์ชั่นต่างๆที่เกี่ยวข้องกับ IRQ จะไม่สามารถใช้งานได้ และในคำสั่งของโปรแกรมที่จะนำมา RUN ใน Monitor Mode ก็ต้องไม่ไปกำหนดการใช้งาน Interrupt จาก IRQ นี้ด้วย เช่น

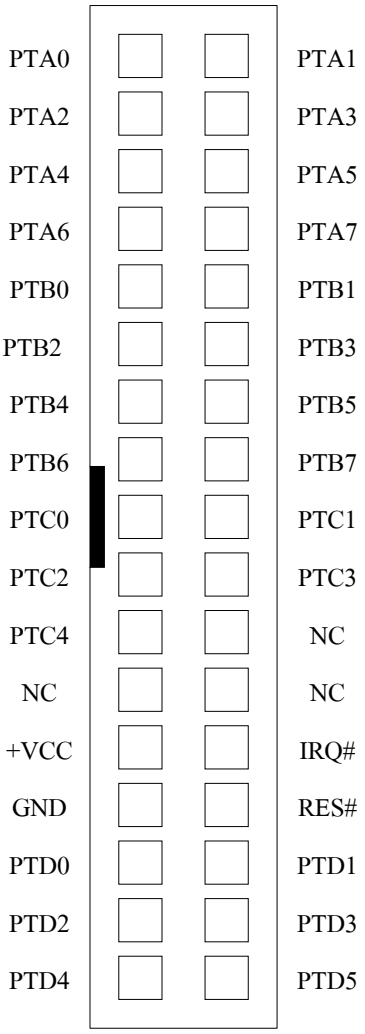
- การใช้งาน Interrupt ของ IRQ
- บิต IRQF ในรีจิสเตอร์ INTSCR
- บิต ACK ในรีจิสเตอร์ INTSCR
- บิต IMASK ในรีจิสเตอร์ INTSCR
- บิต MODE ในรีจิสเตอรื INTSCR
- การเปลี่ยนแปลงการ Pull-Up ของขาสัญญาณ IRQ
- คำสั่งตรวจสอบสถานะทางโลจิกของขาสัญญาณ IRQ

ส่วนใน User Mode นั้น จะสามารถใช้งานสัญญาณ IRQ1 ได้ตามต้องการ โดยสัญญาณ IRQ1 ใน User Mode จะถูกเชื่อมต่อไปยังขั้วต่อ 34 PIN และขั้วต่อ IRQ ไว้ และนอกจากนี้แล้วสัญญาณ IRQ นี้ยังถูกจัดสรรไปยัง Jumper สำหรับกำหนดการ Interrupt จากอุปกรณ์ต่างๆ เช่น RTC เบอร์ PCF8583 หรือ I^2C I/O เบอร์ PCF8574 หรือ เครื่องอ่านบัตรแถบแม่เหล็กรุ่น “MCR-B02TTL” รวมทั้งขั้วต่อ 2 PIN CPA (IRQ) อีกด้วย

ซึ่งตามปรกติแล้ว ขาสัญญาณ IRQ นี้จะทำหน้าที่สำหรับรอรับสัญญาณการ Interrupt จากภายนอกให้ CPU ได้รับรู้ แต่อย่างไรก็ตามขาสัญญาณ IRQ นี้ถ้าไม่ได้เปิดการใช้งานการ Interrupt ไว้ก็ยังสามารถนำมาใช้งานเป็น Input Port ได้ เนื่องจาก CPU เองมีคำสั่งสำหรับทำหน้าที่ตรวจสอบสถานะทางโลจิกของขาสัญญาณ IRQ ไว้ให้ใช้งานด้วย

การใช้งานขั้วต่อ 34PIN (72IOZ80)

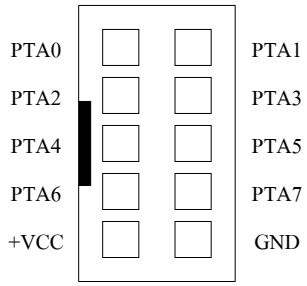
สำหรับขั้วต่อ Connector ขนาด 34 PIN ของบอร์ด CP-JR08GP32 นั้น จะมีอยู่ด้วยกัน 2 แบบ คือ ถ้าเป็นบอร์ด CP-JR08GP32 V1.0 และ CP-JR08GP32 V2.0 นั้น จะเป็นขั้วแบบ IDE ขนาด 34 PIN ตัวผู้ ซึ่งขั้วต่อนี้ออกแบบไว้สำหรับให้ผู้ใช้เชื่อมต่อสัญญาณต่างๆของ CPU ออกไปใช้งานกับบอร์ดอื่นๆ ซึ่งอาจเป็นบอร์ดที่ผู้ใช้ออกแบบและสร้างขึ้นเอง หรืออาจใช้บอร์ด I/O ต่างๆที่ ทาง บริษัท อีทีที จำกัด สร้างขึ้นไว้สนับสนุนการใช้งานก็ได้ โดยวิธีการเชื่อมต่อนั้นขอแนะนำให้ใช้สายแพร์ขนาด 34 PIN จะสะดวกที่สุดเพราะสามารถทำการเชื่อมต่อหรือแยกบอร์ดออกจากกันได้ง่าย ส่วนในกรณีที่ใช้บอร์ดรุ่น CP-JR08GP32 V1.0 EXPANSION นั้น ขั้วต่อ 34 PIN จะเป็นแบบ IDE ตัวเมีย เพื่อให้ผู้ใช้สามารถใช้สาย Jumper ต่อสัญญาณต่างๆจากขั้วต่อนี้ไปยังแผงทดลอง Photo Board เพื่อต่อร่วมกับวงจรต่างๆได้โดยง่าย โดยลักษณะการจัดเรียงสัญญาณเป็นดังนี้



รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้ว 34PIN

การใช้งานขั้วต่อ PTA(KBI)

พอร์ต PTA(KBI) ถูกจัดไว้ที่ขั้ว Connector ขนาด 10 PIN แบบ IDE ซึ่งขั้วต่อนี้จะมีอยู่เฉพาะ ในบอร์ดรุ่น CP-JR08GP32 V2.0 เท่านั้น โดยขั้วต่อนี้จะเชื่อมต่อสัญญาณมาจาก PORTA ของ CPU ทั้ง 8 เส้น โดย PTA0-PTA6 จะถูกนำมาจัดเรียงไว้โดยตรงอยู่แล้ว ส่วน PTA7 จะต้องเลือกจาก Jumper (4x4 / SPK) อีกครั้งหนึ่ง โดยลักษณะของขาสัญญาณที่จัดเรียงไว้ที่ขั้ว PTA(KBI) จะเป็นดังรูป



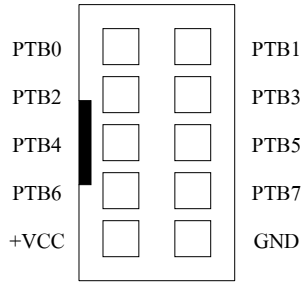
รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้ว PTA(KBI)

โดยจุดประสงค์ในการออกแบบของบอร์ด CP-JR08GP32 V2.0 นั้น ขั้วต่อ PTA(KBI) จะใช้สำหรับให้ผู้ใช้เชื่อมต่อกับวงจรคีย์บอร์ดแบบ Matrix ซึ่งสามารถจะใช้ได้กับคีย์บอร์ดแบบ Matrix ขนาด 4x3 หรือ 4x4 ก็ได้ ซึ่งในกรณีที่ใช้กับคีย์บอร์ดขนาด 4x3 จะเหลือสัญญาณไว้ 1 เส้นคือ PTA7 ซึ่งสามารถนำไปใช้ควบคุมการกำเนิดเสียงของลำโพงขนาดเล็กหรือ BUZZER เพื่อกำเนิดเสียงได้

สำหรับในกรณีที่ไม่มีความจำเป็นต้องใช้งานคีย์บอร์ดแล้ว ขั้วต่อ PTA(KBI) นี้ก็ยังสามารถนำไปต่อใช้งานเป็น Input หรือ Output ทั่วๆไปได้อีกด้วย

การใช้งานขั้วต่อ PTB(ADC)

พอร์ต PTB(ADC) นี้จะถูกเชื่อมต่อออกมารอไว้ยังขั้ว Connector ขนาด 10 PIN แบบ IDE ซึ่งขั้วต่อนี้จะมีอยู่เฉพาะบอร์ดในรุ่น CP-JR08GP32 V2.0 เท่านั้น โดยขั้วต่อ PTB(ADC) นี้ สามารถโปรแกรมหน้าที่การใช้งานได้ หลายหน้าที่ เช่น โปรแกรมให้ทำหน้าที่เป็น ADC ขนาด 8 บิต 8 ช่อง เพื่อรับค่าสัญญาณ Analog ขนาด 0-5VDC จากภายนอกเข้ามาและเปลี่ยนเป็นค่าข้อมูล (00H-FFH) ให้กับ CPU นำไปประมวลผลตามต้องการ หรือถ้าไม่ต้องการใช้งานเป็น ADC พอร์ต PTB(ADC) นี้ก็ยังสามารถโปรแกรมหน้าที่การทำงานสำหรับใช้งานเป็น Input / Output ทั่วไปได้อีกด้วย โดยลักษณะของขาสัญญาณที่จัดเรียงไว้ที่ขั้ว PTB (ADC) จะเป็นดังรูป



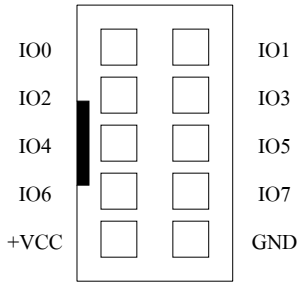
รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้ว PTB(ADC)

นอกจากนี้แล้วสัญญาณ PTB5,PTB6 และ PTB7 ของ พอร์ต PTB นั้นยังจัดสรรเป็นทางเลือกสำหรับนำไปใช้ประโยชน์อื่นๆได้อีกเช่น

- PTB5 ในกรณีที่ไม่ต้องการใช้งานเป็น ADC สามารถนำไปใช้งานเป็นขาสัญญาณ SDA สำหรับเชื่อมต่อกับอุปกรณ์ I2C ได้อีกด้วย โดยการเลือก Jumper SDA มาไว้ยังด้าน PTB5
- PTB6 ในกรณีที่ไม่ต้องการใช้งานเป็น ADC สามารถนำไปใช้งานเป็นขาสัญญาณ SCL สำหรับเชื่อมต่อกับอุปกรณ์ I2C ได้อีกด้วย โดยการเลือก Jumper SCL มาไว้ยังด้าน PTB6
- PTB7 ในกรณีที่ไม่ต้องการใช้งานเป็น ADC สามารถนำไปใช้งานเป็น Output สำหรับควบคุมการทำงานของ RELAY ได้อีกด้วย โดยการ Short Jumper PTB7(RELAY) ไว้

การใช้งานขั้วต่อ I^2C IN/OUT

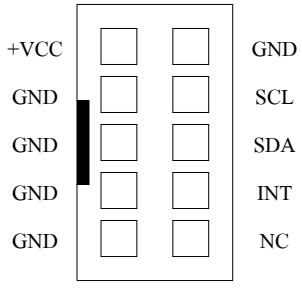
พอร์ต I^2C IN/OUT นี้จะถูกเชื่อมต่อออกมารอไว้ยังขั้ว Connector ขนาด 10 PIN แบบ IDE ซึ่งขั้วต่อนี้จะมีอยู่เฉพาะบอร์ดในรุ่น CP-JR08GP32 V2.0 เท่านั้น โดยขั้วต่อ I^2C IN/OUT นี้ จะเชื่อมต่อมาจากขาสัญญาณ I/O Port ของ PCF8574/A ซึ่งสามารถโปรแกรมหน้าที่การใช้งาน ให้เป็น Input หรือ Output ก็ได้ตามต้องการจากโปรแกรม แต่ต้องกำหนดหน้าที่ให้เป็น Input หรือ Output อย่างใดอย่างหนึ่งเท่านั้น ไม่สามารถใช้งานทั้งสองหน้าที่พร้อมๆกันได้ และในการกำหนดหน้าที่ให้เป็น Input หรือ Output ก็ต้องกำหนดให้เหมือนกันทั้ง 8 บิต ด้วย โดยลักษณะของขาสัญญาณที่จัดเรียงไว้ที่ขั้ว I^2C IN/OUT จะเป็นดังรูป



รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้ว I^2C IN/OUT

การใช้งานขั้วต่อ I^2C BUS EXPAND

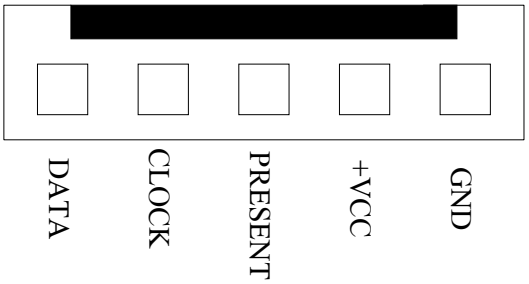
พอร์ต I^2C BUS EXPAND นี้จะถูกเชื่อมต่อออกมารอไว้ยังขั้ว Connector ขนาด 10 PIN แบบ IDE ซึ่งขั้วต่อนี้จะมีอยู่เฉพาะบอร์ดในรุ่น CP-JR08GP32 V2.0 เท่านั้น โดยขั้วต่อ I^2C BUS EXPAND นี้ จะใช้สำหรับทำการขยายหรือเพิ่มเติมจำนวนอุปกรณ์ที่ใช้การติดต่อสื่อสารแบบ I2C ให้กับบอร์ด



รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้ว I^2C BUS EXPAND

การใช้งานเครื่องอ่านบัตรแถบแม่เหล็ก (MAGNETIC-CARD READER)

สำหรับบอร์ด CP-JR08GP32 V2.0 นั้น จะออกแบบให้สามารถเชื่อมต่อกับเครื่องอ่านบัตรแถบแม่เหล็กรุ่น "MCR-B02TTL" ได้โดยตรง โดยไม่ต้องดัดแปลงวงจรใดๆทั้งสิ้น โดยในบอร์ดจะจัดเตรียมขั้วแบบ CPA ขนาด 5 PIN ไว้รองรับอยู่แล้ว ผู้ใช้สามารถนำขั้วต่อของเครื่องอ่านบัตรแถบแม่เหล็ก รุ่น "MCR-B02TTL" ของบริษัท อีทีที ต่อเข้าไปได้ทันที สำหรับในการเขียนโปรแกรมเพื่อเชื่อมต่อระหว่างบอร์ด CP-JR08GP32 V2.0 กับเครื่องอ่านบัตรแถบแม่เหล็กนั้น จะสามารถทำได้ 2 แบบ คือ ใช้วิธีการวนรอบตรวจสอบสัญญาณจากเครื่องอ่านบัตรแถบแม่เหล็กเอง ซึ่งวิธีการนี้จะใช้สัญญาณจาก CPU เพียง 2 เส้นสัญญาณคือ PTD4 ทำหน้าที่เป็น DATA และ PTD5 ทำหน้าที่เป็น CLOCK โดยต้องกำหนดคุณสมบัติของสัญญาณทั้ง 2 เส้นให้มีทิศทางเป็น Input และควร ENABLE การ PULL-UP ของสัญญาณทั้ง 2 นี้ด้วยเสมอ ส่วนอีกวิธีหนึ่งคือการใช้วิธีการ Interrupt ซึ่งสามารถทำได้โดยการ SHORT JUMPER "INT(IRQ)" ที่อยู่ใกล้ๆกับขั้วต่อเครื่องอ่านบัตรแถบแม่เหล็ก ซึ่งการ SHORT JUMPER จะเป็นการเชื่อมต่อสัญญาณ IRQ1 ของ CPU เข้ากับ สัญญาณ PRESENT ของเครื่องอ่านบัตรแถบแม่เหล็ก ดังนั้นเมื่อมีการนำบัตรแถบแม่เหล็กไปรูดผ่านเครื่องอ่านบัตรแถบแม่เหล็กก็จะมีการส่งสัญญาณ Interrupt ออกมายัง CPU ซึ่งผู้ใช้ก็เพียงแต่เขียนโปรแกรมสำหรับบริการการ Interrupt ของ IRQ1 ไว้ก็สามารถใช้งานได้แล้ว



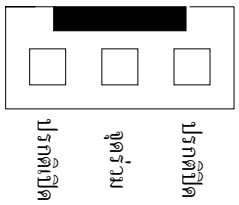
รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้วต่อเครื่องอ่านบัตรแถบแม่เหล็ก MCR-B02TTL

<u>หมายเหตุ</u> เนื่องจากการเลือกใช้ Interrupt จากสัญญาณ IRQ ของ CPU นั้น ภายในบอร์ด รุ่น CP-JR08GP32 V2.0 สัญญาณ IRQ จะถูกออกแบบให้สามารถเลือกกำหนดใช้งานร่วมกับอุปกรณ์หลายๆตัว เช่น การ Interrupt จาก RTC เบอร์ PCF8583 หรือ การ Interrupt จาก PORT I/O เบอร์ PCF8574/A รวมทั้งการ Interrupt จาก เครื่องอ่านบัตรแถบแม่เหล็ก MCR-B02TTL นี้ด้วย ดังนั้น ในกรณีที่ต้องการเลือกใช้วิธีการ Interrupt จาก IRQ นั้น จะต้องทำการ OPEN JUMPER สำหรับเลือกการ Interrupt จากอุปกรณ์อื่นๆที่ไม่เกี่ยวข้องออกเสียก่อน ให้เหลือการเชื่อมต่อสัญญาณ IRQ จาก CPU กับอุปกรณ์เพียงตัวใดตัวหนึ่งเท่านั้น เพื่อให้แน่ใจว่าสัญญาณ Interrupt ที่เกิดขึ้นจะถูกส่งมาออกมาจากเครื่องอ่านบัตรแถบแม่เหล็กเท่านั้น ไม่เช่นนั้นแล้วอาจเกิดความผิดพลาดขึ้นได้

การใช้งาน OUTPUT RELAY

ภายในบอร์ด CP-JR08GP32 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 PTB7 ของ CPU ซึ่งเมื่อต้องการใช้งาน RELAY จะต้องทำการ SHORT JUMPER PTB7(RELAY) ไว้ด้วยเพื่อเชื่อมต่อสัญญาณ PTB7 มาทำการควบคุมการทำงานของ RELAY และต้องแน่ใจว่าไม่ได้ต่อสัญญาณ PTB7 จากขั้วต่ออื่นๆออกไปใช้งานกับอุปกรณ์ใดๆนอกเหนือจาก RELAY เนื่องจากสัญญาณ PTB7 นั้น นอกจากจะต่อมาใช้ควบคุมการทำงานของ RELAY แล้วยังต่อไปยังขั้วต่อ 34PIN และขั้วต่อ PTB(ADC) อีกด้วย โดยการทำงานของ RELAY จะถูกควบคุมการทำงานจากขาสัญญาณ PTB7 โดยผู้ใช้ต้องกำหนดคุณสมบัติของสัญญาณ PTB7 ให้ทำหน้าที่เป็น OUTPUT ไว้ด้วย ซึ่งเมื่อขาสัญญาณ PTB7 มีสภาวะเป็น OUTPUT และให้สถานะเป็น "1" จะทำให้ RELAY ทำงาน แต่ถ้าสถานะของสัญญาณ PTB7 มีค่าเป็น "0" จะทำให้ RELAY หยุดทำงาน



รูปแสดง ลักษณะขั้วต่อสัญญาณจากหน้าสัมผัสของ RELAY

*****หมายเหตุ***** เนื่องจากสัญญาณ PTB7 ที่นำมาใช้ควบคุมการทำงานของ RELAY จะเป็นสัญญาณเส้นเดียวกับ PTB7 ที่ต่อไว้ยังขั้ว 34PIN และขั้วต่อ 10PIN แบบ IDE ของพอร์ต PTB (ADC) ดังนั้นเมื่อต้องการใช้งาน RELAY โดยการควบคุมจาก PTB7 แล้ว ต้องแน่ใจว่าไม่ได้ต่อใช้งานสัญญาณ PTB7 ในจุดอื่นๆทั้งสอง ดังกล่าวไปใช้งานด้วย แต่ถ้าหากมีความจำเป็นต้องใช้งาน PTB7 พร้อมกับการใช้งาน RELAY ด้วยในเวลาเดียวกัน ก็อาจดัดแปลงวงจรได้โดยการ OPEN JUMPER PTB7(RELAY) ออก แล้วใช้วิธีการเชื่อมสายสัญญาณเส้นอื่นๆจาก PORT I/Oของ CPU ที่ไม่ได้ใช้งานมาเข้ากับวงจรควบคุม RELAY แทนก็ได้เช่นเดียวกัน โดยให้เชื่อมต่อสายสัญญาณที่ต้องการไปยัง Jumper PTB7(RELAY) ด้านที่ต่อกับตัวต้านทานค่า 1KOhm แต่การดัดแปลงวิธีนี้ควรถอดตัว JUMPER PTB7(RELAY) ออกจากบอร์ดเสียก่อน เพื่อจะได้ไม่หลงลืมทำการ SHORT JUMPER นี้ซ้ำอีกในภายหลัง เนื่องจากจะเป็นการ SHORT สัญญาณ PTB7 เข้ากับสัญญาณเส้นใหม่ที่บัดกรีมายังวงจร RELAY นี้อีก

การใช้งานลำโพงขนาดเล็ก หรือ BUZZER

ภายในบอร์ด CP-JR08GP32 V2.0 จะมีวงจรกำเนิดเสียงรวมอยู่ด้วย 1 จุด ซึ่งในตำแหน่งนี้สามารถเลือกใส่อุปกรณ์กำเนิดเสียงแบบลำโพงขนาดเล็ก หรือ จะเลือกใส่ BUZZER แทนก็ได้เช่นเดียวกัน โดยในกรณีที่เลือกใช้ลำโพงจะมีข้อดีคือ สามารถสร้างความถี่เสียงได้หลากหลายความถี่ตามต้องการแต่การเขียนโปรแกรมจะยุ่งยากกว่า BUZZER เนื่องจากต้องสร้างเป็นสัญญาณความถี่จึงจะสามารถทำให้ลำโพงกำเนิดเสียงให้ได้ ส่วนในกรณีที่เลือกใช้ BUZZER นั้น จะมีข้อดีคือ เขียนโปรแกรมควบคุมการกำเนิดเสียงได้ง่ายกว่าลำโพง เนื่อง

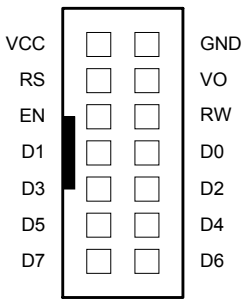
จากใช้วิธีการ ON หรือ OFF เท่านั้น โดยการ ON บิตควบคุม BUZZER ให้มีสภาวะเป็น “1” เท่านั้น BUZZER ก็จะกำเนิดเสียงให้แล้ว แต่ความถี่เสียงของ BUZZER จะไม่สามรถเลือกได้ เหมือนกับลำโพง

สำหรับสัญญาณ Output ในการควบคุมการทำงานของ ลำโพง หรือ BUZZER นั้น จะแบ่งมาจาก PTA7 ของ CPU ซึ่งเมื่อต้องการใช้งาน ลำโพง หรือ BUZZER จะต้องทำการเลือก JUMPER 4x4/SPK มารอไว้ยังตำแหน่ง SPK ด้วย เพื่อให้สามารนำสัญญาณ PTA7 มาทำการควบคุมการทำงานของลำโพง หรือ BUZZER ได้ โดยผู้ใช้ต้องทำการกำหนดคุณสมบัติของสัญญาณ PTA7 ให้ทำหน้าที่เป็น OUTPUT ไว้ด้วย ซึ่งเมื่อขาสัญญาณ PTA7 มีสภาวะเป็น OUTPUT และให้สถานะเป็น “1” จะทำให้ ลำโพง หรือ BUZZER ทำงาน แต่ถ้าสถานะของสัญญาณ PTA7 มีค่าเป็น “0” จะทำให้ ลำโพง หรือ BUZZER หยุดทำงาน

*****หมายเหตุ***** เนื่องจากสัญญาณ PTA7 ที่นำมาใช้ควบคุมการทำงานของ ลำโพง หรือ BUZZER นั้นจะเป็นสัญญาณเส้นเดียวกับ PTA7 ที่ต่อไว้ยังขั้ว 34PIN และขั้วต่อ 10PIN แบบ IDE ของพอร์ต PTA (KBI) ด้วย ดังนั้นเมื่อต้องการใช้งาน ลำโพง หรือ BUZZER โดยการควบคุมจาก PTA7 แล้ว ต้องแน่ใจว่าไม่ได้ต่อใช้งานสัญญาณ PTA7 ในจุดอื่นๆทั้งสอง ดังกล่าวด้วย แต่ถ้าหากมีความจำเป็นต้องใช้งาน PTA7 พร้อมกับการใช้งาน ลำโพง 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 ด้านตำแหน่ง 4x4 ไว้ด้วยกันเลย เพื่อจะได้ไม่หลงลืมทำการเลือก JUMPER กลับมายังด้าน SPK นี้อีกในภายหลัง เนื่องจากจะเป็นการ SHORT สัญญาณ PTA7 เข้ากับสัญญาณเส้นใหม่ที่บัดกรีมายังวงจรควบคุมลำโพงหรือ BUZZER นี้อีก

การใช้งานจอแสดงผลแบบ LCD (Dot-Matrix Character LCD)

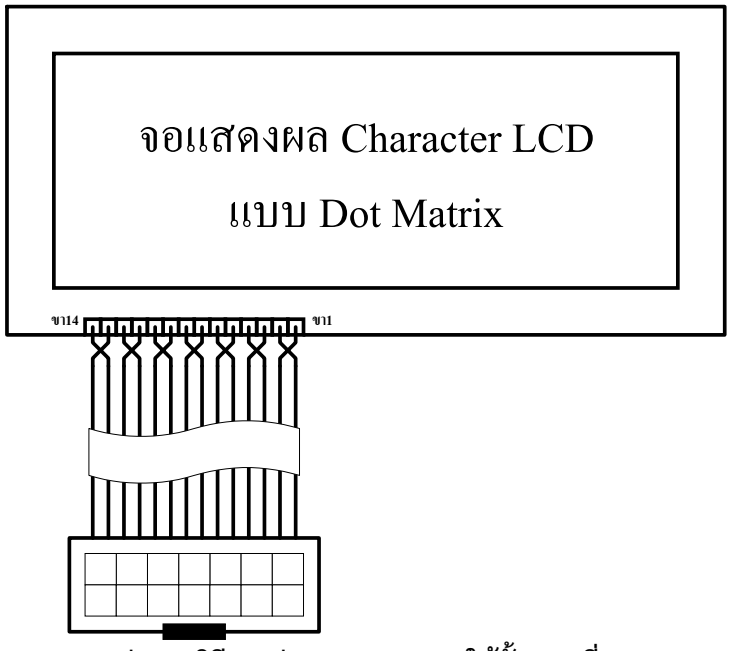
บอร์ด CP-JR08GP32 V2.0 สามารถใช้เชื่อมต่อกับจอแสดงผล LCD แบบ Dot-Matrix โดยเชื่อมต่อผ่านทาง Connector ขนาด 14 PIN และใช้ตัวต้านทานปรับค่าได้แบบเกือกม้าขนาด 10K สำหรับปรับระดับความสว่างของหน้าจอ LCD โดยวงจรในการเชื่อมต่อ LCD ของบอร์ดนี้จะออกแบบวงจรให้ใช้วิธีการควบคุมการทำงานแบบ "DATA 4-BIT"



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-JR08GP32 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-JR08GP32 V2.0 นั้น จะออกแบบให้สามารถติดตั้งใช้งานอุปกรณ์ I²C ได้พร้อมกันในบอร์ดทั้งหมดด้วยกัน 3 ตัว คือ

- I²C RTC เบอร์ PCF8583 ของ PHILIPS
- EEPROM ในตระกูล 24XX ซึ่งสามารถเลือกใช้ได้หลายเบอร์หลายผู้ผลิต ขึ้นอยู่กับขนาดความจุของหน่วยความจำที่ต้องการจะใช้ ซึ่งในบอร์ด CP-JR08GP32 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 PTC0 หรือ PTB5 เป็นขาสัญญาณ SDA และใช้สัญญาณ PTC1 หรือ PTB6 เป็นสัญญาณ SCL ในการควบคุมบัส ซึ่ง CPU จะทำหน้าที่เป็นตัวแม่ในการควบคุมบัส นอกจากนี้แล้วยังสามารถขยายอุปกรณ์จำพวก I²C นี้ได้อีก แต่ต้องเป็นอุปกรณ์ที่มีรหัสควบคุม Control Word ไม่ซ้ำกันกับอุปกรณ์ที่มีอยู่แล้วภายในบอร์ดด้วย โดยอาจเชื่อมต่อผ่านทางขั้วต่อ “I²C EXPANSION” ที่บอร์ดจัดเตรียมไว้ให้แล้วก็ได้

สำหรับในการใช้งานอุปกรณ์ I²C นั้น เนื่องจาก CPU เบอร์ MC68HC908GP32 ไม่มีส่วนของฮาร์ดแวร์ที่ทำหน้าที่ติดต่อสื่อสารแบบ I²C บรรจุไว้ในตัว CPU ด้วย ดังนั้นจึงต้องใช้วิธีการนำ I/O Port ของ CPU มาใช้ติดต่อกับอุปกรณ์แบบ I²C แทน ซึ่งอุปกรณ์ I²C นั้นต้องการสัญญาณในการติดต่อสื่อสารกันจำนวน 2 เส้น สัญญาณ คือ SCL(Clock) และ SDA(Data) โดยสัญญาณของ I/O Port ที่ใช้ในการติดต่อสื่อสารกับอุปกรณ์แบบ I²C ของบอร์ด CP-JR08GP32 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 แทนและต้องกำหนดการ Pull-Up ให้กับสัญญาณที่ทำหน้าที่เป็น SDA นี้ด้วยเสมอ โดยสัญญาณของ CPU ที่จะสามารถใช้ทำหน้าที่เป็นสัญญาณ SDA สำหรับติดต่อสื่อสารกับอุปกรณ์ I²C นี้ สามารถเลือกได้ 2 สัญญาณ โดยการเลือกจาก Jumper SDA (C0/B5) ซึ่งถ้าเลือก Jumper SDA ไว้ทางด้าน C0 จะหมายถึง การกำหนดให้สัญญาณ PTC0 ของ CPU ทำหน้าที่เป็น SDA แต่ถ้าเลือก Jumper SDA ไว้ทางด้าน B5 จะหมายถึงกำหนดให้สัญญาณ PTB5 ทำหน้าที่เป็น SDA สำหรับข้อพิจารณาในการเลือกสัญญาณที่จะนำมาใช้ทำหน้าที่เป็น SDA นั้น ถ้าผู้ใช้ต้องการเขียนโปรแกรมทดลองใน Monitor Mode จะต้องเลือกสัญญาณ PTB5 ให้ทำหน้าที่เป็น SDA เนื่องจากใน Monitor Mode สัญญาณ PTC0 จะไม่สามารถนำมาใช้งานได้ ส่วนในโหมดการทำงานปรกติ หรือ User Mode จะสามารถเลือกได้ทั้ง 2 แบบ ซึ่งตามปรกติควรเลือก Jumper SDA ไว้ทางด้าน C0 จะดีกว่า
- **SCL** ทำหน้าที่เป็นสัญญาณนาฬิกา Clock สำหรับใช้ติดต่อสื่อสารกับอุปกรณ์ I²C โดยสัญญาณที่ทำหน้าที่เป็น SCL นี้จะเป็นสัญญาณ Output จาก CPU เพื่อใช้ควบคุมการรับส่งข้อมูลระหว่าง CPU กับอุปกรณ์ โดยสัญญาณ I/O Port ของ CPU ที่จะสามารถใช้ทำหน้าที่เป็นสัญญาณ SCL

สำหรับติดต่อสื่อสารกับอุปกรณ์ I^2C นี้ สามารถเลือกได้ 2 สัญญาณ โดยการเลือกจาก Jumper SCL (C1/B6) ซึ่งถ้าเลือก Jumper SCL ไว้ทางด้าน C1 จะหมายถึง การกำหนดให้สัญญาณ PTC1 ของ CPU ทำหน้าที่เป็น SCL แต่ถ้าเลือก Jumper SCL ไว้ทางด้าน B6 จะหมายถึงกำหนดให้สัญญาณ PTB6 ทำหน้าที่เป็น SCL สำหรับข้อพิจารณาในการเลือกสัญญาณที่จะนำมาใช้ทำหน้าที่เป็น SCL นั้น ถ้าผู้ใช้ต้องการเขียนโปรแกรมทดลองใน Monitor Mode จะต้องเลือกสัญญาณ PTB6 ให้ทำหน้าที่เป็น SCL เนื่องจากใน Monitor Mode สัญญาณ PTC1 จะไม่สามารถนำมาใช้งานได้ ส่วนในโหมดการทำงานปรกติ หรือ User Mode จะสามารถเลือกได้ทั้ง 2 แบบ ซึ่งตามปรกติควรเลือก Jumper SCL ไว้ทางด้าน C1 จะดีกว่า

การใช้งาน Interrupt ของอุปกรณ์ I^2C

สำหรับสัญญาณ Interrupt จากอุปกรณ์ I^2C นั้น เนื่องจาก CPU มีสัญญาณ Input ที่ใช้สำหรับรับการ Interrupt จากภายนอก ได้เพียง 1 สัญญาณ คือ IRQ ดังนั้น ผู้ใช้จำเป็นต้องจัดสรรและเลือกว่า จะให้อุปกรณ์ตัวใดใช้สัญญาณ Interrupt ดังกล่าว โดยบอร์ด CP-JR08GP32 V2.0 นั้น จะออกแบบให้ผู้ใช้สามารถเลือกการใช้สัญญาณ Interrupt จาก IRQ ได้ว่า จะทำการเชื่อมต่อสัญญาณ IRQ เข้ากับอุปกรณ์ตัวใด โดยมีจุดเลือกการต่อสัญญาณ Interrupt ให้กับ IRQ ได้ทั้งหมด 5 แหล่ง คือ

- ต่อสัญญาณการ Interrupt ให้กับ IRQ ทางขั้วต่อ 34PIN
- ต่อสัญญาณการ Interrupt ให้กับ IRQ ทางขั้วต่อ 2PIN CPA (IRQ)
- ต่อสัญญาณการ Interrupt ให้กับ IRQ จากเครื่องอ่านบัตรแถบแม่เหล็ก
- ต่อสัญญาณการ Interrupt ให้กับ IRQ จาก RTC PCF8583
- ต่อสัญญาณการ Interrupt ให้กับ IRQ จาก I/O Port PCF8574/A

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

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

แอดเดรสของอุปกรณ์ I^2C

เนื่องจากคุณสมบัติของ BUS แบบ I^2C นั้น สามารถเชื่อมต่ออุปกรณ์ต่างๆที่ใช้วิธีการสื่อสารแบบ I^2C ได้มากมากหลายตัวภายในบัสเดียวกันได้ เพียงแต่มีข้อแม้ว่า อุปกรณ์ที่จะนำมาต่อร่วมกันภายในบัสเดียวกันนั้นจะต้องมีรหัสตำแหน่งในการติดต่อสื่อสาร (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-JR08GP32 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-JR08GP32 V2.0

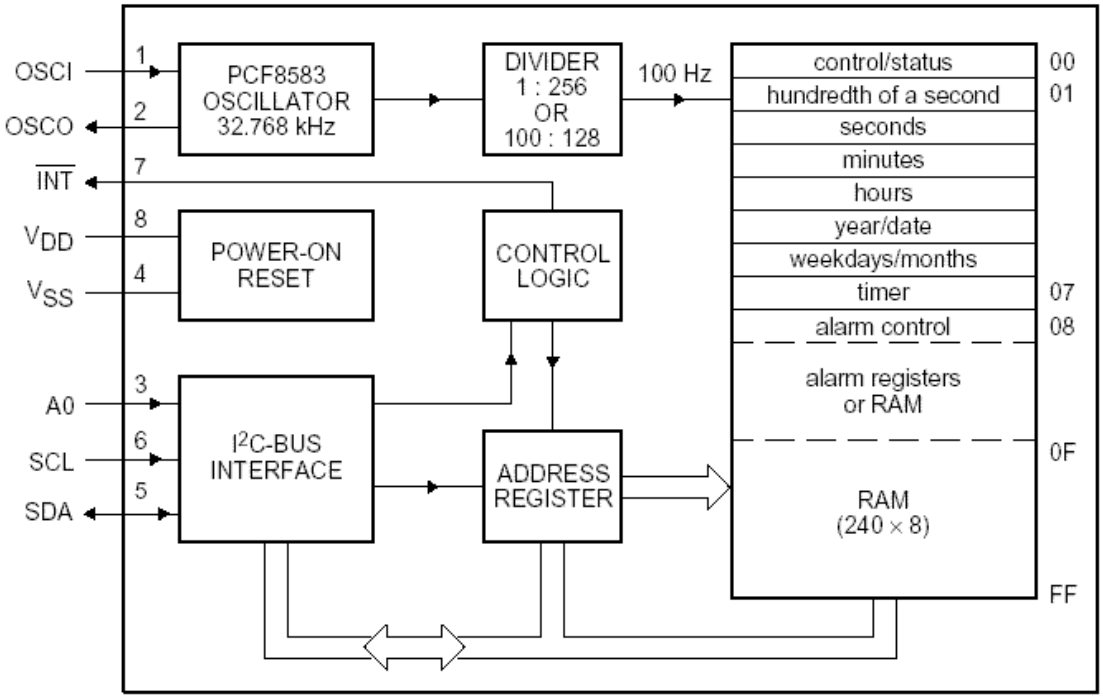
*****หมายเหตุ***** 1. ค่า X หมายถึงค่าโลจิกของขาสัญญาณ Address ของอุปกรณ์ ที่กำหนดในวงจร

2. ค่า ? หมายถึงบิตสำหรับกำหนดว่าต้องการเขียน หรือ อ่าน ข้อมูลกับอุปกรณ์

3. เนื่องจาก RTC เบอร์ PCF8583 และ E^2PROM เบอร์ในกลุ่ม 24XX นั้นมีรหัสตำแหน่ง 4 บิตแรก ซ้ำกัน หรือ อยู่ในกลุ่มเดียวกัน ซึ่งในบอร์ด CP-JR08GP32 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-JR08GP32 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-JR08GP32 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-JR08GP32 V2.0 นั้น จะกำหนดให้ขาสัญญาณ A0 ของ PCF8583 มีสภาวะทางโลจิกเป็น "1" คงที่ไว้เลย ดังนั้น RTC เบอร์ PCF8583 ในบอร์ด CP-JR08GP32 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

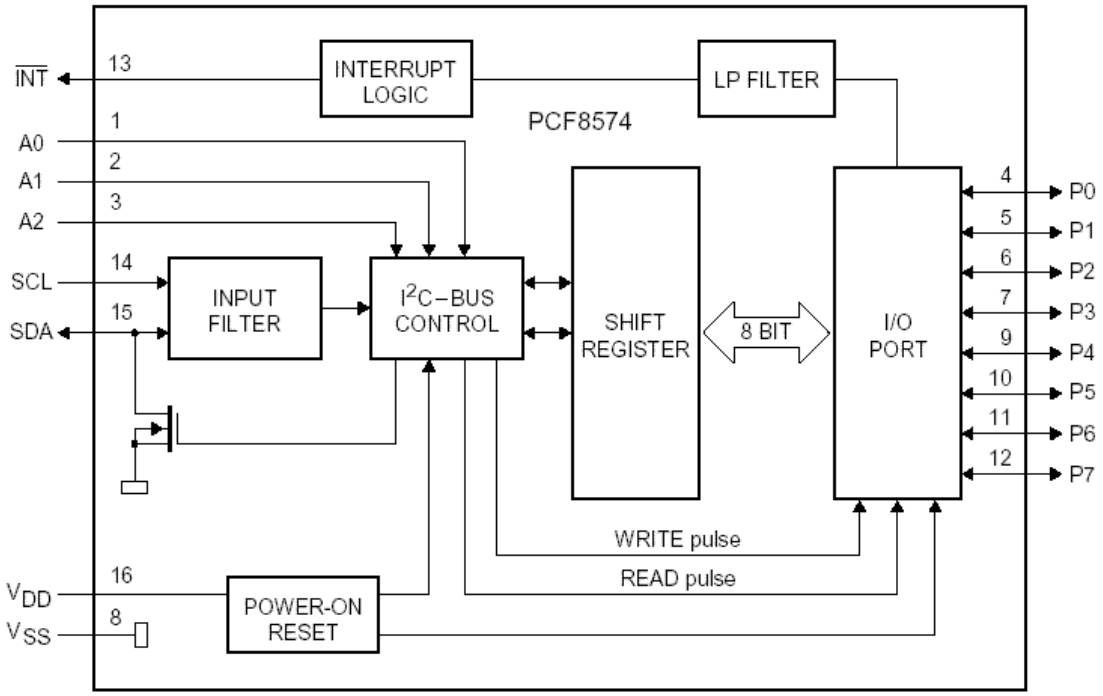
เบอร์(ความจุ)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
24XX32 (4Kx8)	1	0	1	0	A2	A1	A0	R/W
24XX64 (8Kx8)	1	0	1	0	A2	A1	A0	R/W
24XX128 (16Kx8)	1	0	1	0	A2	A1	A0	R/W
24XX256 (32Kx8)	1	0	1	0	A2	A1	A0	R/W

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-JR08GP32 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 MC68HC908GP32 นั้นจะมีพอร์ต I/O สำหรับให้ผู้ใช้สามารถนำไปใช้งานได้มากถึง 5 พอร์ต อยู่แล้ว ซึ่งในส่วนของบอร์ด CP-JR08GP32 V1.0 นั้น พอร์ต I/O ทั้งหมดของ CPU จะปล่อยว่างไว้ให้ผู้ใช้เลือกใช้งานอย่างอิสระตามต้องการ แต่สำหรับบอร์ดรุ่น CP-JR08GP32 V2.0 นั้น พอร์ต I/O ต่างๆ ของ CPU จะถูกจัดสรรออกไปใช้งานในวงจรต่างๆ ดังได้กล่าวอธิบายมาแล้วในข้างต้น แต่ถ้าหากว่าผู้ใช้งานมีความจำเป็นต้องใช้งานพอร์ต I/O จำนวนมาก และจำนวนพอร์ต I/O ของ CPU ที่มีอยู่ไม่เพียงพอต่อการใช้งานแล้ว ผู้ใช้ก็สามารถทำการเพิ่มเติมพอร์ต I/O ได้อีก โดยในบอร์ด CP-JR08GP32 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	$R/\overline{W}$

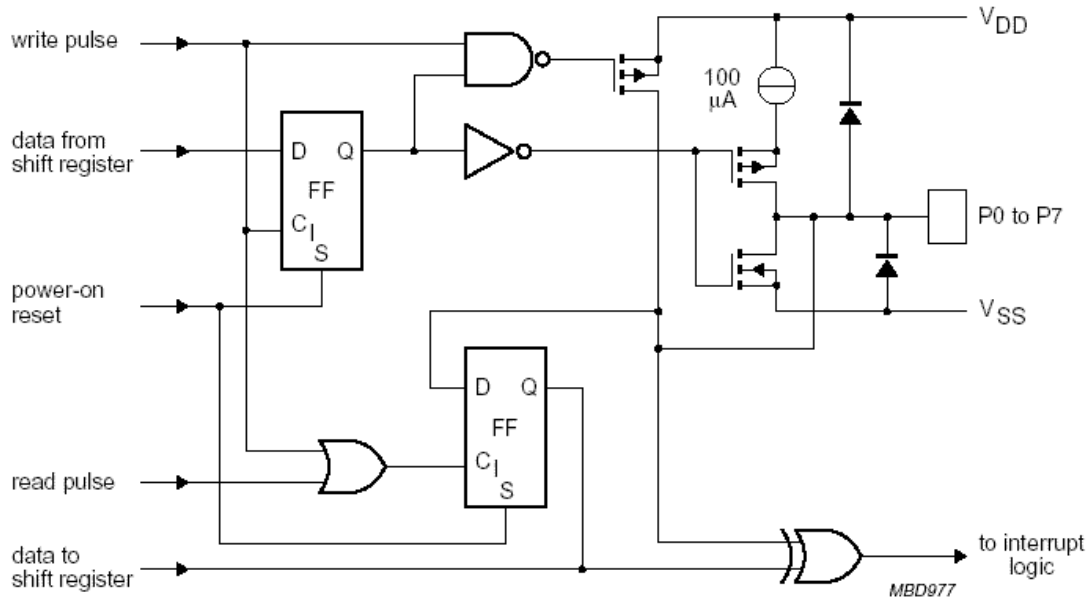
รูปแสดง รหัส Control Byte **ของ** PCF8574

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 MC68HC908GP32 ที่ใช้กับบอร์ด CP-JR08GP32 นั้น จะมีวงจรสื่อสารแบบอนุกรม (Serial Communication Interface : SCI) บรรจุรวมไว้ด้วยแล้ว ซึ่งวงจรส่วนนี้ผู้ใช้งานสามารถทำการเขียนโปรแกรมควบคุมการสื่อสารข้อมูลของ CPU กับอุปกรณ์อื่นๆได้ตามต้องการ โดยในส่วนของโปรแกรมนั้นผู้ใช้สามารถกำหนดรูปแบบของการสื่อสารข้อมูลได้เองจากโปรแกรมที่เขียนขึ้น ไม่ว่าจะเป็นความเร็วในการสื่อสาร (Baudrate) จำนวนบิตข้อมูลในการรับส่ง (Data Bit) การกำหนดบิตรวจสอบความถูกต้องข้อมูล (Parity) และคุณสมบัติอื่นๆ ซึ่งในรายละเอียดส่วนนี้จะไม่ขอกล่าวถึง ขอให้ผู้ใช้ศึกษาจากคู่มือสถาปัตยกรรมทางฮาร์ดแวร์ หรือ Data Sheet ของ CPU เบอร์ MC68HC908GP32 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-JR08GP32 จึงได้ออกแบบวงจร Line Driver สำหรับทำหน้าที่เป็น Buffer เพื่อเปลี่ยนแปลงระดับสัญญาณทางไฟฟ้าของขาสัญญาณ รับ-ส่ง ข้อมูลของ CPU ที่เป็น แบบ TTL ให้สามารถรับส่งข้อมูลกันได้ในระยะทางที่ไกลมากขึ้น (สามารถอ่านรายละเอียดเพิ่มเติมได้จากหัวข้อ “ความรู้ทั่วไปเกี่ยวกับการสื่อสารอนุกรม” ในภาคผนวกท้ายเล่มของคู่มือนี้) โดยบอร์ด CP-JR08GP32 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

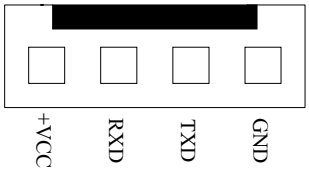
สำหรับสายสัญญาณที่จะนำมาใช้สำหรับทำการสื่อสารแบบ RS232 นั้น จะใช้สัญญาณเพียง 2-3 เส้นเท่านั้น ทั้งนี้ขึ้นอยู่กับความต้องการในการสื่อสารว่าต้องการสื่อสารแบบทิศทางเดียวหรือสองทิศทาง

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

สลับสัญญาณกับอุปกรณ์ปลายทางด้วย คือ สัญญาณส่ง (TXD) จากบอร์ด CP-JR08GP32 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-JR08GP32 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-JR08GP32 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

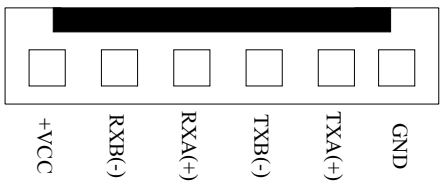
****หมายเหตุ*** สำหรับไอซี Line Drive แบบ RS232 นั้น จะจัดเป็นอุปกรณ์มาตรฐานของบอร์ดในตระกูล CP-JR08GP32 ซึ่งจะมีติดตั้งให้ไปกับบอร์ดอยู่แล้ว ผู้ใช้ไม่ต้องจัดหาเพิ่มเติม แต่พึงระลึกไว้เสมอว่า จะต้องทำการติดตั้งไอซี Line Driver สำหรับเลือกชนิดสัญญาณทางไฟฟ้าของการสื่อสารอนุกรม ได้เพียงอย่างใดอย่างหนึ่งเท่านั้น 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-JR08GP32 V2.0 เท่านั้น

สำหรับลักษณะของขั้วต่อของสัญญาณ RS422 นั้น จะเป็นแบบ CPA ขนาด 6 PIN ดังรูป โดยในการสื่อสารกันนั้น จะใช้สายสัญญาณในการ รับ-ส่ง ข้อมูลกัน จำนวน 4 เส้น สัญญาณ คือ สัญญาณในการรับข้อมูล จำนวน 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-JR08GP32 V2.0 **เมื่อเลือกเป็น** RS422

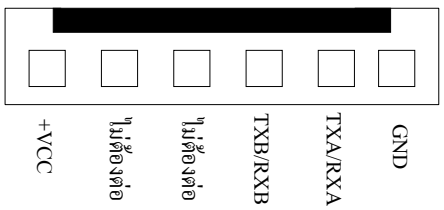
การสื่อสารอนุกรมแบบ RS485

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-JR08GP32 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-JR08GP32 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

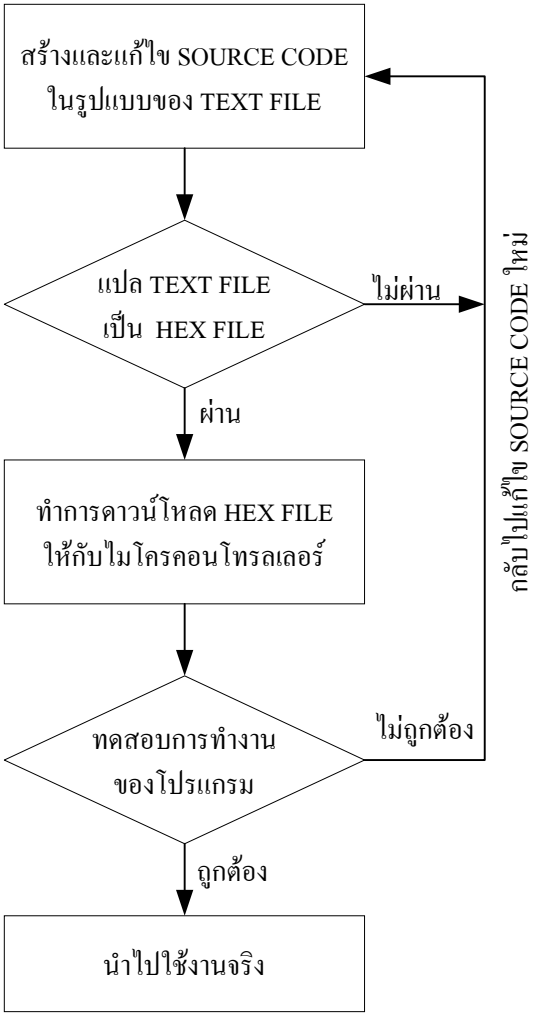
ไปชดเชยค่าความต้านทานของสายสัญญาณด้วย โดยเมื่อทำการ Short Jumper ตำแหน่ง RZ นี้ไว้ก็จะเป็นการต่อตัวต้านทานคร่อมระหว่าง RXA (RX+) และ RXB (RX-) ไว้ แต่ถ้าหากว่าต่อสายสัญญาณในระยะทางที่ไม่ไกลมากนัก ก็ให้ทำการ Open Jumper นี้ออกก็ได้

- **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-JR08GP32

สำหรับการพัฒนาโปรแกรมนั้น บอร์ด CP-JR08GP32 จะออกแบบวงจรให้สามารถทำการพัฒนาโปรแกรมของบอร์ดได้ โดยเลือกโหมดการทำงานของบอร์ดเป็น Monitor Mode โดยในการพัฒนาโปรแกรมของบอร์ดนั้นต้องต่อใช้งานร่วมกับเครื่องโปรแกรม CPU รุ่น “ET-PGM08” ของ บริษัท อีทีที จำกัดซึ่งในการพัฒนาโปรแกรมนั้นจะมีลำดับขั้นตอนในการพัฒนาโปรแกรมดังแผนผังต่อไปนี้



แผนผัง แสดงขั้นตอนในการพัฒนาโปรแกรมของบอร์ด CP-JR08GP32

สำหรับในที่นี้จะขอกล่าวถึงเฉพาะวิธีการ Download Hex File ให้กับบอร์ดเท่านั้น ส่วนวิธีการเขียนโปรแกรมและการสั่งแปลคำสั่งให้ได้เป็น Hex File นั้น ขอให้ผู้ใช้ศึกษาจากข้อกำหนดของโปรแกรมแปลภาษาที่จะนำมาใช้ในการเขียนโปรแกรมเอง

โดยเมื่อกำหนดโหมดการทำงานของบอร์ดไว้ใน Monitor Mode แล้วจะทำให้สามารถเลือกวิธีการพัฒนาโปรแกรมของบอร์ดในโหมดการทำงานนี้ได้มากถึง 3 รูปแบบ คือ การพัฒนาแบบ In-circuit Simulator การพัฒนาแบบ In-circuit Debugger และการพัฒนาโปรแกรมแบบ In-circuit Programmer ซึ่งวิธีการพัฒนาโปรแกรมแต่ละแบบ ต่างก็มีข้อดี และข้อจำกัด ที่แตกต่างกันไปดังนี้

1. **การพัฒนาโปรแกรมแบบ** In-circuit Simulator (ICS08GPZ.EXE)

การพัฒนาโปรแกรมแบบนี้ จะเหมาะสำหรับผู้เริ่มต้น ซึ่งยังขาดความรู้ความชำนาญในการเขียนโปรแกรม เนื่องจากวิธีนี้เป็นการพัฒนาโปรแกรมที่มีความสะดวกมากอีกวิธีหนึ่งเพราะผู้ใช้สามารถทำการสั่งตรวจสอบข้อผิดพลาดหรือผลการทำงานของ CPU ได้อย่างง่ายดาย โดยการพัฒนาโปรแกรมแบบนี้จะต้องใช้งานร่วมกับโปรแกรม ICS08GPZ.EXE ของ P&E ซึ่งมีคุณสมบัติดังนี้

ข้อดีของการพัฒนาโปรแกรมด้วย In-circuit Simulator

- สามารถเขียนโปรแกรมในตำแหน่งการทำงานจริงเพื่อทดสอบการทำงานกับบอร์ดพร้อมทั้งดูผลการทำงานของโปรแกรมที่เขียนขึ้นผ่านทางหน้าจอของโปรแกรม In-circuit Simulator ได้
- สามารถตรวจสอบการทำงานของโปรแกรมในขณะที่กำลัง RUN อยู่ได้ และยังสามารถติดตามตรวจสอบหรือดูการเปลี่ยนแปลงค่าของรีจิสเตอร์หรือหน่วยความจำหรืออุปกรณ์ I/O ต่างๆในขณะที่บอร์ดทำงานอยู่จริงๆได้ (Run โปรแกรมแบบ Multi Step)
- สามารถหยุดการทำงานของโปรแกรม (BREAK) ในตำแหน่งแอดเดรสต่างๆของโปรแกรมเพื่อตรวจสอบค่าของรีจิสเตอร์หรือหน่วยความจำต่างๆได้ตามต้องการ
- สามารถใช้วิธีการ RUN แบบทีละคำสั่ง (Single Step) เพื่อศึกษาผลการทำงานของคำสั่งที่มีผลต่อ รีจิสเตอร์ หรือหน่วยความจำ หรืออุปกรณ์ I/O ต่างๆได้จริง
- ง่ายต่อการศึกษาและทำความเข้าใจการทำงานของคำสั่งต่างๆของ CPU

ข้อจำกัดของการพัฒนาโปรแกรมด้วย In-circuit Simulator

- ความเร็วในการทำงานของบอร์ดจะไม่ใช่ความเร็วจริงเหมือนการทำงานใน RUN MODE แต่ถ้าหากต้องการความเร็วในการทำงานเท่ากับความเร็วจริงเหมือนใน RUN MODE ต้องใช้วิธีการพัฒนาโปรแกรมแบบ In-circuit Debugger แทน

2. **การพัฒนาโปรแกรมแบบ** In-circuit Debugger (ICD08SZ.EXE)

การพัฒนาโปรแกรมด้วยวิธีนี้จะมีลักษณะและวิธีการคล้ายกันกับแบบ In-circuit Simulator แต่มีข้อดีกว่าคือสามารถทำงานที่ความเร็วเท่าปรกติได้ โดยวิธีการนี้จะต้องใช้งานร่วมกับโปรแกรม ICD08SZ.EXE ของ P&E ซึ่งมีคุณสมบัติที่น่าสนใจดังต่อไปนี้

ข้อดีของการพัฒนาโปรแกรมด้วย In-circuit Debugger

- การทำงานของ CPU สามารถทำงานได้เท่าความเร็วปรกติเหมือนใน MODE RUN
- สามารถสั่งให้ CPU ทำงานแบบทีละคำสั่ง (Single Step) ได้ทั้งโปรแกรมที่เขียนเพื่อให้ CPU ทำงานทั้งในพื้นที่ของ RAM และ ROM (Flash)
- สามารถสั่งหยุดการทำงานของโปรแกรมได้ทั้งตำแหน่งที่เป็นตำแหน่งของหน่วยความจำใน ROM (Flash) หรือ RAM ได้
- สามารถสั่งเปลี่ยนแปลงแก้ไขค่ารีจิสเตอร์ต่างๆของ CPU ได้
- สามารถเขียนโปรแกรมเพื่อทดลองในพื้นที่ของหน่วยความจำ RAM ซึ่งจะทำให้สามารถทำการเปลี่ยนแปลงแก้ไขค่าต่างๆของโปรแกรมได้โดยง่าย ซึ่งจะเหมาะสำหรับการพัฒนาโปรแกรมที่

ประเภทโปรแกรมย่อยต่างๆ หรือโปรแกรมที่ต้องมีการปรับแต่งค่าบ่อยๆ เช่น โปรแกรม Delay เป็นต้น หรือถ้าต้องการเขียนโปรแกรมในตำแหน่งที่เป็นหน่วยความจำ Flash ก็สามารถทำได้เช่นกัน แต่จะต้องใช้วิธีการ Download โปรแกรมด้วยวิธีการ In-circuit Programmer มาก่อนแล้วจึงมาสั่ง Run ด้วย Debugger นี้ในภายหลัง

ข้อจำกัดของการพัฒนาโปรแกรมด้วย In-circuit Debugger

- ไม่สามารถสั่งทำการ Step คำสั่งที่กระโดดอยู่กับที่ได้
- ไม่สามารถสั่งทำการ Step คำสั่ง SWI ในโปรแกรมได้
- ไม่สามารถสั่งแก้ไขค่าในหน่วยความจำที่เป็น ROM หรือ Flash ได้ ถ้าต้องการแก้ไขค่าของหน่วยความจำที่เป็น Flash จะต้องใช้วิธีการ In-circuit Programmer แทน
- ไม่สามารถใช้งานวงจร Break ในตัว CPU ได้เนื่องจากโปรแกรม ICD08SZ จะสงวนไว้ใช้งานในการสั่ง Break ไว้ก่อนแล้ว ดังนั้นผู้ใช้จึงไม่สามารถเขียนโปรแกรมเพื่อสั่งงานการ Break เองได้
- ถ้ามีการสั่ง Refresh หน้าจอของการแสดงผลต่างๆของหน่วยความจำหรือรีจิสเตอร์ ที่แสดงไว้ในหน้าต่าง Memory Windows หรือ Variable Windows ของโปรแกรม ICD08SZ แล้วค่าของหน่วยความจำ RAM และค่าของรีจิสเตอร์บางส่วนอาจเสียหายหรือเกิดการเปลี่ยนแปลงขึ้นได้เนื่องจากโปรแกรม ICD08SZ จะต้องสั่งให้ CPU อ่านค่าหน่วยความจำหรือรีจิสเตอร์ดังกล่าวด้วย
- โปรแกรม ICD08SZ ต้องการใช้งานพื้นที่ของสแต็กจำนวน 13Byte ดังนั้นผู้ใช้ต้องไม่สั่งเขียนข้อมูลใดๆไปยัง RAM ในตำแหน่งที่ชี้โดย SP (SP-13 ถึง SP+0)
- ถ้าเกิดการ Interrupt ในขณะกำลังสั่งทำการ Single Step อยู่ จะไม่สามารถเกิดการ Interrupt ได้
- ห้ามสั่งหยุดการทำงานหรือกำหนดตำแหน่งการ BREAK ในตำแหน่งที่เป็นพื้นที่เก็บโปรแกรมของ Monitor ROM (MON) ในตัว CPU

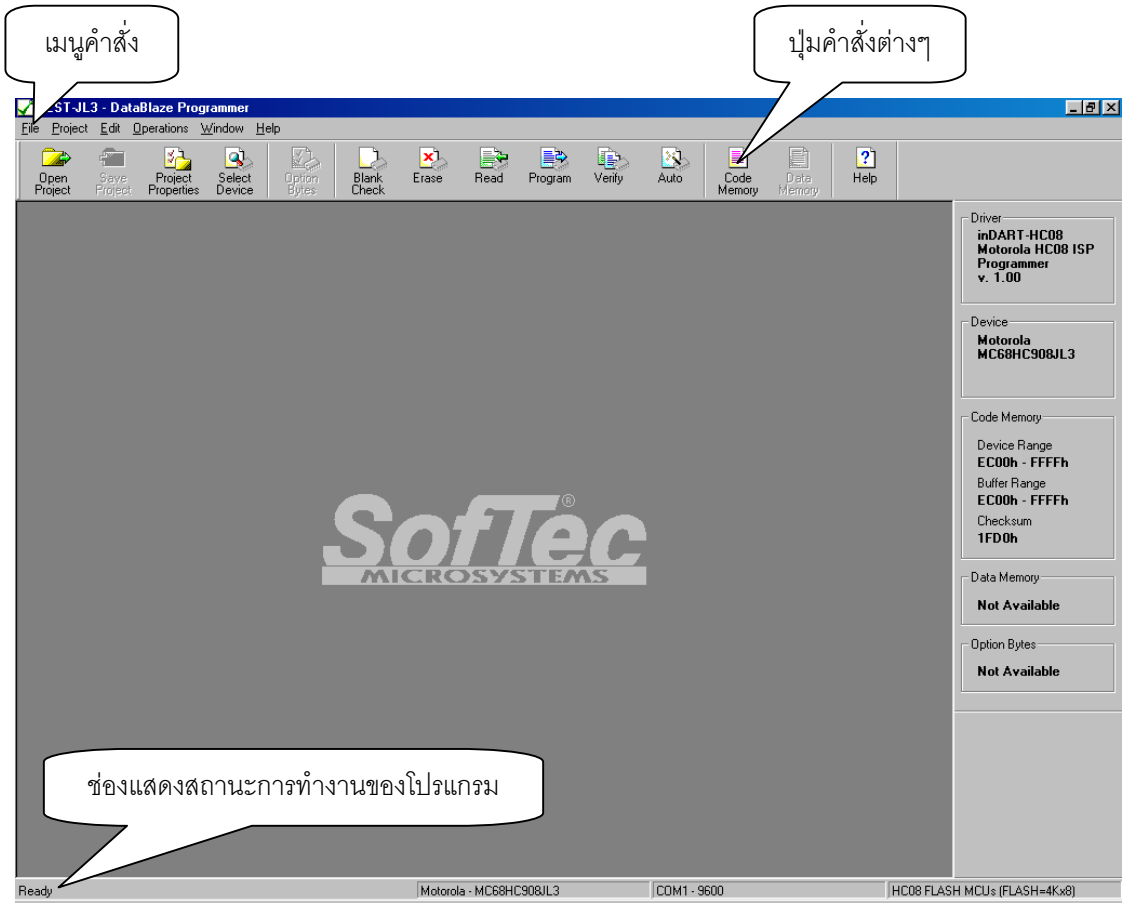
3. การพัฒนาโปรแกรมแบบ In-circuit Programmer (PROG08SZ.EXE หรือ DATABLAZE.EXE)

การพัฒนาโปรแกรมด้วยวิธีนี้ โดยมากแล้วจะใช้ในกรณีที่ทำการทดลองโปรแกรมเสร็จสมบูรณ์แล้ว ซึ่งจะเป็นการสั่งจัดการหน่วยความจำของ CPU ได้ตามต้องการ ไม่ว่าจะเป็นการสั่งลบข้อมูลในหน่วยความจำ หรือสั่งเขียนข้อมูลหรือโปรแกรมข้อมูลให้กับหน่วยความจำ ซึ่งถ้าหากไฟล์ที่นำมาโปรแกรมให้กับหน่วยความจำของ CPU นั้น มีความสมบูรณ์ถูกต้อง เมื่อเลือกโหมดการทำงานของบอร์ดกลับไปทางด้าน RUN แล้ว ก็สามารถนำบอร์ดไปใช้งานจริงได้ทันที

หรือในกรณีที่ต้องการทดสอบการทำงานของโปรแกรมด้วยวิธีการ In-circuit Debugger แต่ต้องการเขียนโปรแกรมในตำแหน่งแอดเดรสการทำงานของหน่วยความจำ Flash จริงๆ ก็จำเป็นต้องใช้วิธีการ Download โปรแกรมหรือสั่งเขียนข้อมูลให้กับหน่วยความจำ Flash ด้วยวิธีการ In-Circuit Programmer นี้ให้เสร็จเรียบร้อยเสียก่อนแล้วจึงจะสามารถกลับไปใช้งานโปรแกรม In-Circuit Debugger เพื่อสั่ง Run โปรแกรม หรือตรวจสอบการทำงานของโปรแกรมด้วย Debugger ได้

การใช้งานโปรแกรม DATABLAZE.EXE สำหรับ Download โปรแกรม

โปรแกรม DATABLAZE.EXE เป็นโปรแกรมสำหรับโปรแกรมข้อมูลให้กับหน่วยความจำของไมโครคอนโทรลเลอร์ตระกูล MC68HC08 ซึ่งตามปรกติแล้วจะใช้คู่กับเครื่องโปรแกรมของ “Softec Microsystem” ซึ่งจะใช้สำหรับทำหน้าที่เกี่ยวกับการติดต่อสื่อสารกับ CPU ใน Monitor Mode เพื่อให้ผู้ใช้สั่งจัดการกับหน่วยความจำภายในตัว CPU ไม่ว่าจะเป็นการ สั่งลบข้อมูล(Erase) สั่งตรวจสอบข้อมูลในหน่วยความจำ(Blank Check) สั่งโปรแกรมข้อมูลให้กับหน่วยความจำโปรแกรมของ CPU (Program) สั่งเปรียบเทียบข้อมูลจาก Buffer กับหน่วยความจำในตัว CPU (Verify) หรือสั่งอ่านข้อมูลจากหน่วยความจำของ CPU (Read) เป็นต้น โดยลักษณะของโปรแกรม DATABLAZE.EXE เป็นดังรูป



รูปแสดง ลักษณะหน้าต่างโปรแกรมของ DATABLAZE.EXE

ซึ่งจะเห็นได้ว่า โปรแกรม DATABLAZE.EXE นี้จะมีคำสั่งอยู่มากมายหลายคำสั่ง เพื่ออำนวยความสะดวกต่อผู้ใช้ โดยผู้ใช้สามารถเลือกสั่งงานคำสั่งต่างๆได้ตามต้องการ ซึ่งการเรียกใช้งานคำสั่งต่างๆนั้นอาจเลือกจากเมนูคำสั่งโดยตรง หรืออาจเลือกจากปุ่มคำสั่งโดยตรงก็ได้ เนื่องจากคำสั่งที่มีความจำเป็นต้องใช้งานบ่อยๆนั้นถ้าใช้วิธีการสั่งงานจากเมนูคำสั่ง อาจทำให้เกิดความไม่สะดวกต่อการใช้งานเนื่องจากต้องผ่านขั้นตอนหลายขั้นตอน ซึ่งโปรแกรม DATABLAZE.EXE เองก็ได้สร้างปุ่มคำสั่งพิเศษขึ้นมาให้เลือกใช้งาน นอกจากนั้นแล้วยังสามารถเรียกใช้งานคำสั่งด้วยการใช้คีย์ลัด (Short Key) ได้ด้วย เช่น ถ้าต้องการจะสั่งลบข้อมูล (Erase) นั้นแทนที่จะต้องเลือกจากคำสั่งในเมนูคำสั่ง Operation → Erase ก็อาจใช้วิธีการกดคีย์ “Ctrl+E” หรือเลือกคลิก

เมาส์ที่ปุ่มคำสั่ง Erase เพียงปุ่มเดียวก็ได้เช่นกัน โดยวิธีการใช้งานต่างๆโดยละเอียดนั้นสามารถดูได้จาก คู่มือของโปรแกรมเอง หรืออาจศึกษาจาก Help ในเมนูคำสั่ง Help ก็ได้ ซึ่งในที่นี้จะขอกล่าวและอธิบายถึงคำสั่งที่มีความจำเป็นต่อการใช้งานเพื่อเป็นแนวทางให้ทราบดังต่อไปนี้

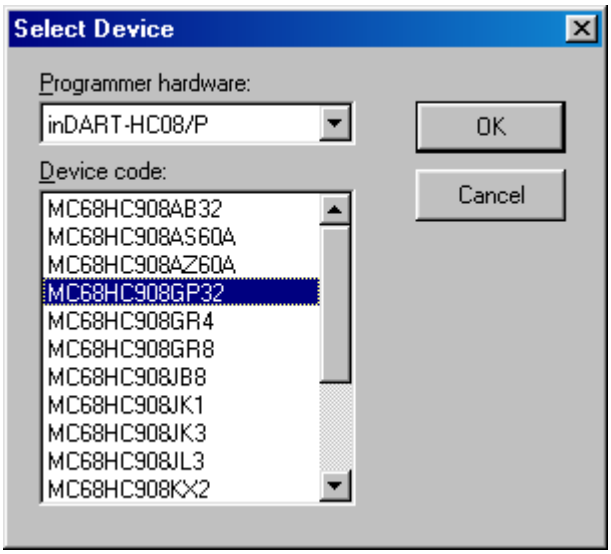
- **Open Project** ใช้สำหรับสั่งเปิด Project File ที่สร้างไว้แล้วขึ้นมาใช้งาน
- **Save Project** ใช้สำหรับสั่งบันทึกคุณสมบัติต่างๆที่เลือกไว้แล้วเป็น Project File ไว้
- **Project Properties** ใช้สำหรับเปลี่ยนแปลงแก้ไขและกำหนดคุณสมบัติต่างๆของ Project File
- **Select Device** ใช้สำหรับเลือกกำหนดเบอร์ CPU ที่จะใช้กับเครื่อง
- **Blank Check** ใช้สำหรับสั่งตรวจสอบว่าหน่วยความจำของ CPU ว่างอยู่หรือไม่
- **Erase** ใช้สำหรับสั่งลบข้อมูลเก่าในหน่วยความจำเพื่อรอการโปรแกรมซ้ำใหม่
- **Read** ใช้สำหรับสั่งอ่านข้อมูลจากหน่วยความจำมารอไว้ใน Buffer
- **Program** ใช้สำหรับสั่งโปรแกรมข้อมูลใน Buffer ไปยังหน่วยความจำของ CPU
- **Verify** ใช้สำหรับเปรียบเทียบข้อมูลใน Buffer กับข้อมูลในหน่วยความจำของ CPU
- **Auto** ใช้สำหรับสั่งโปรแกรมข้อมูลจาก Buffer ให้กับหน่วยความจำ CPU แต่คำสั่งนี้มีความพิเศษตรงที่การทำงานของคำสั่ง สามารถเลือกกำหนดขั้นตอนที่จำเป็นๆสำหรับการโปรแกรมข้อมูลให้กับหน่วยความจำของ CPU โดยการนำคำสั่งหลายๆคำสั่งมาเรียงลำดับกันไว้ เช่น Erase →Blank → Program →Verify แต่ไม่จำเป็นต้องกำหนดให้ทำงานทุกขั้นตอนก็ได้ เช่น อาจกำหนดให้โปรแกรมทำงานเฉพาะคำสั่งที่จำเป็นคือ Erase → Program เพียง 2 คำสั่งเพื่อให้เกิดความรวดเร็วในการทำงานมากยิ่งขึ้นก็ได้
- **Code Memory** ใช้สำหรับดูหรือแก้ไขข้อมูลใน Buffer

การ SETUP โปรแกรม DATABLAZE.EXE เพื่อใช้งานกับเครื่องโปรแกรม ET-PGM08

เนื่องจากเครื่องโปรแกรม ET-PGM08 ได้รับการออกแบบวงจรให้มีคุณสมบัติการทำงานตรงตามข้อกำหนดมาตรฐานของ Motorola ทุกประการ และในส่วนของโปรแกรม DATABLAZE.EXE เองก็สามารถใช้งานกับเครื่องมือต่างๆที่ทำงานอยู่ภายใต้ข้อกำหนดมาตรฐานของ Motorola ได้ด้วยเช่นกัน ดังนั้นระบบการทำงานของเครื่องโปรแกรม ET-PGM08 จึงสามารถใช้งานร่วมกันกับโปรแกรม DATABLAZE.EXE ได้โดยไม่มีปัญหา โดยในการกำหนดคุณสมบัติการทำงานของโปรแกรม DATABLAZE.EXE ให้สามารถใช้ควบคุมการทำงานเครื่อง ET-PGM08 สำหรับโปรแกรมข้อมูลให้กับ CPU ในบอร์ด CP-JR08GP32 V1.0 และ V2.0 ได้ดังนี้

1. ต่อสายสัญญาณ RS232 จากเครื่องคอมพิวเตอร์ PC เข้ากับเครื่องโปรแกรม ET-PGM08
2. ต่อสายแพร์ 14 PIN จากขั้วต่อ P1 จากเครื่อง ET-PGM08 เข้ากับขั้ว Download Program ของบอร์ด CP-JR08GP32 พร้อมกับเลือก Jumper และ Slide Switch สำหรับเลือกโหมดการทำงานของบอร์ดไว้ทางด้าน Monitor Mode (MON)
3. จ่ายไฟเลี้ยงวงจรให้กับเครื่อง ET-PGM08 และบอร์ด CP-JR08GP32
4. ทำการเรียกใช้งานโปรแกรม DATABLAZE.EXE เพื่อสั่งงานเครื่องโปรแกรม ET-PGM08

5. ให้ทำการกำหนดเบอร์ CPU ที่ต้องการจะโปรแกรม โดยอาจทำการคลิกเมาส์ที่เมนูคำสั่ง Operation → Select Device หรืออาจทำการคลิกเมาส์ที่ปุ่ม Select Device โดยตรงก็ได้เช่นกัน ซึ่งจะปรากฏหน้าต่างสำหรับแสดงรายการเบอร์ของ CPU ที่สามารถใช้งานร่วมกับเครื่องโปรแกรมได้ ให้ทำการคลิกเมาส์สำหรับเลือกกำหนดเบอร์ CPU เป็น MC68HC908GP32 ดังรูป



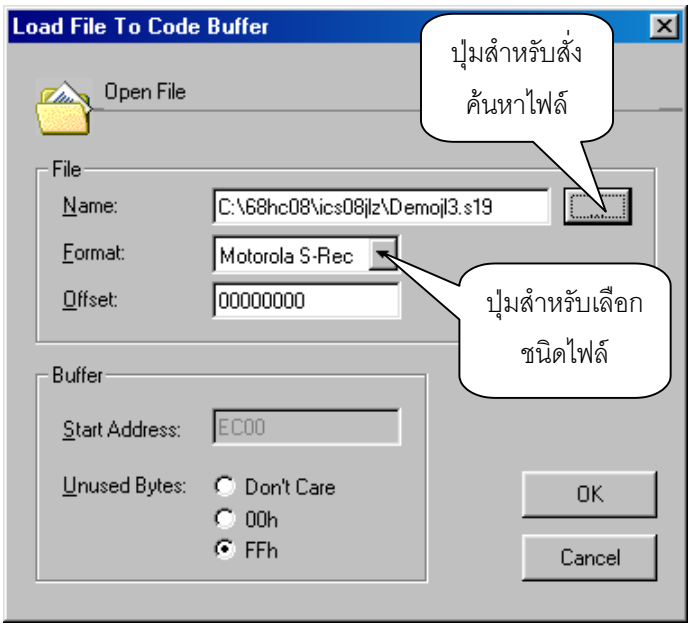
รูปแสดง หน้าต่างของโปรแกรม DATABLAZE.EXE สำหรับเลือกเบอร์ CPU

ให้ทำการคลิกเมาส์ไปยังเบอร์ CPU ที่ต้องการในช่อง Device Code จนเป็นแถบสว่างแล้วเลือกคลิกเมาส์ที่ปุ่ม OK เพื่อทำการเลือกเบอร์ CPU ตามต้องการ หรืออาจใช้วิธีการ Double คลิกเมาส์ที่เบอร์ CPU ที่ต้องการในช่อง Device Code ก็ได้เช่นเดียวกัน

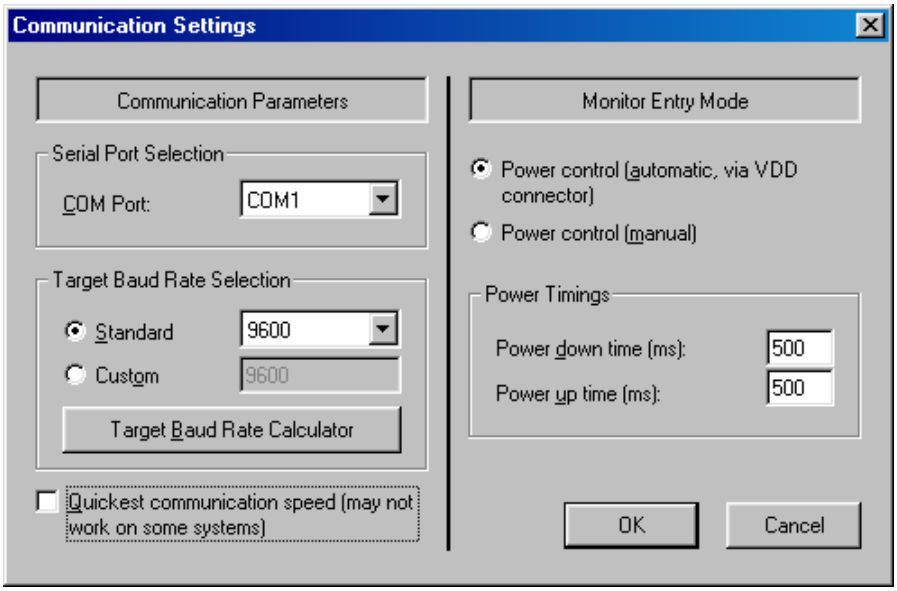
6. ให้ทำการเปิดไฟล์ที่ต้องการจะนำมาโปรแกรมข้อมูลให้กับ CPU โดยให้ทำการคลิกเมาส์ที่เมนูคำสั่ง File → Load → Code Buffer ซึ่งจะปรากฏหน้าต่างสำหรับเลือกกำหนด ชนิดของไฟล์และชื่อของไฟล์ที่จะนำมาโปรแกรมข้อมูลให้กับ CPU ซึ่งไฟล์ดังกล่าวจะต้องเป็นไฟล์ที่ทำการแปลให้อยู่ในรูปในรูปแบบมาตรฐานที่พร้อมทำการโปรแกรมข้อมูลให้กับ CPU ได้ โดยโปรแกรม DATABLAZE.EXE เอง สามารถรับไฟล์มาตรฐานได้ 3 ชนิดด้วยกัน คือ

   - Intel HEX Format (HEX)
   - Motorola HEX Format (S19)
   - Binary Format (BIN)

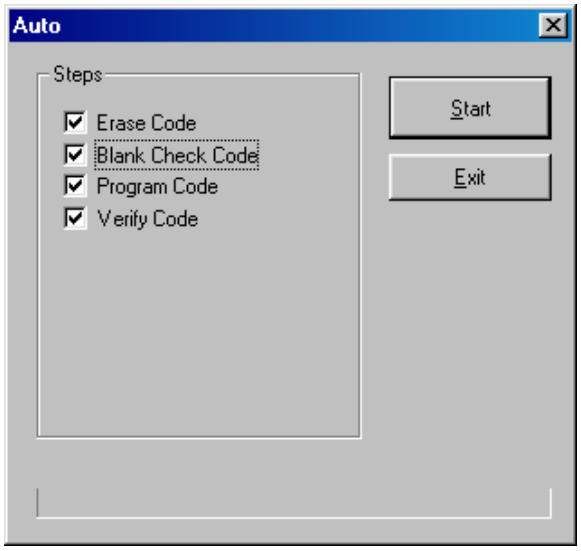
   โดยให้ทำการเลือกกำหนดชนิดของไฟล์ โดยการคลิกเมาส์เลือกที่ช่อง Format แล้วเลือกรายการชนิดของไฟล์ตามต้องการ สำหรับโปรแกรม HEX ที่ได้จากการแปลของโปรแกรมแอสแซมเบอร์ CASM08Z จะเป็นแบบ Motorola HEX ให้เลือกชนิดของไฟล์เป็น Motorola S-Rec ส่วนชื่อไฟล์นั้นสามารถระบุโดยการพิมพ์ตำแหน่งที่อยู่พร้อมชื่อของไฟล์ในช่อง Name ได้ทันที หรืออาจทำการคลิกเมาส์ที่ปุ่มหลังช่อง Name เพื่อทำการค้นหาตำแหน่งและชื่อของไฟล์เองก็ได้ตามต้องการ



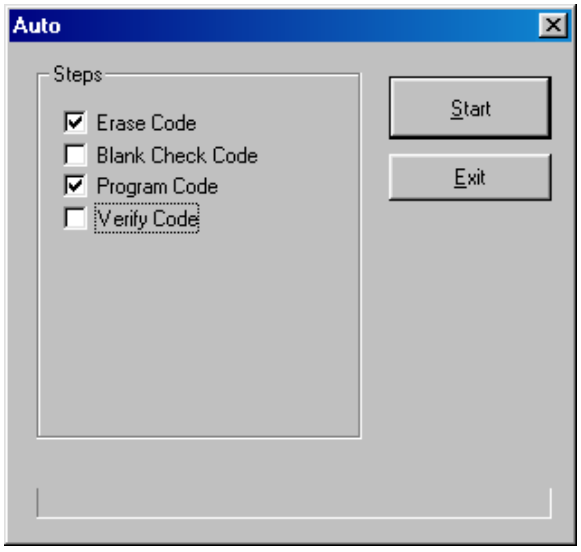
7. ให้ทำการกำหนดการเชื่อมต่อสัญญาณ โดยการคลิกเมาส์ที่เมนูคำสั่ง Operation → Communication Settings ซึ่งจะปรากฏหน้าต่างสำหรับกำหนดรูปแบบการติดต่อสื่อสารระหว่างเครื่องโปรแกรม ET-PGM08 กับโปรแกรม DATABLAZE.EXE ดังตัวอย่าง
   - COM Port ให้เลือกตามความเป็นจริงที่ทำการต่อสายสัญญาณไว้ก่อนหน้านี้แล้ว
   - Target Baudrate Selection ให้เลือกเป็น Standard : 9600
   - Monitor Entry Mode ให้เลือกเป็น Power Control (Automatic via VDD Connector)
   - Power Down Time ให้กำหนดเป็น 500mS
   - Power Up Time ให้กำหนดค่าเป็น 500mS



8. ให้ทำการสั่งงานเครื่องโปรแกรมให้ทำงานตามต้องการ แต่เพื่อความสะดวกขอแนะนำให้เลือกใช้คำสั่ง Auto จะดีที่สุด โดยอาจเลือกคลิกเมาส์ที่เมนูคำสั่ง Operation → Auto หรืออาจทำการคลิกเมาส์ที่ปุ่มคำสั่ง Auto โดยตรงก็ได้เช่นกัน ซึ่งเมื่อเลือกคำสั่งนี้แล้วจะปรากฏหน้าต่างคำสั่งของการ Auto Program ขึ้นมา โดยในเมนูคำสั่งดังกล่าวจะมีคำสั่งย่อยให้เลือกกำหนดเงื่อนไขการทำงานหลายคำสั่ง ให้ทำการคลิกเมาส์หน้าคำสั่งที่ต้องการให้เครื่องโปรแกรมทำงาน โดยถ้าต้องการให้คำสั่งใดทำงานให้คลิกเมาส์จนเกิดเครื่องหมายถูก(√) หน้าคำสั่งนั้นๆ จากนั้นจึงคลิกเมาส์ที่ปุ่มคำสั่ง Start เพื่อสั่งงานโปรแกรม และรอจนเสร็จทุกขั้นตอน



ซึ่งในขั้นตอนนี้อาจไม่จำเป็นต้องสั่งให้โปรแกรมทำงานทุกขั้นตอนให้เสียเวลาก็ได้ เนื่องจากสามารถตัดขั้นตอนบางอย่างที่ไม่จำเป็นออกไปเพื่อให้การทำงานมีความสะดวกรวดเร็วมากยิ่งขึ้น เช่น กำหนดให้โปรแกรมทำการลบ (Erase) และโปรแกรม (Program Code) เพียง 2 ขั้นตอนก็ได้ก็สามารถใช้งานได้เช่นกัน

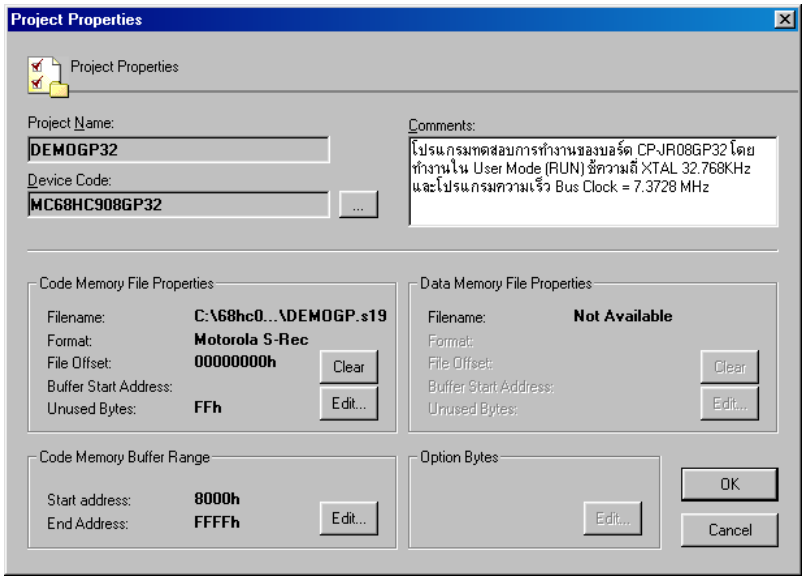


9. หลังจากโปรแกรมข้อมูลให้กับ CPU เสร็จเรียบร้อยแล้วสามารถเลือกกำหนด Jumper และ Slide Switch สำหรับเลือกโหมดการทำงานของบอร์ดไว้ทางด้าน RUN เพื่อให้ CPU ทำงานตามโปรแกรมได้ทันที

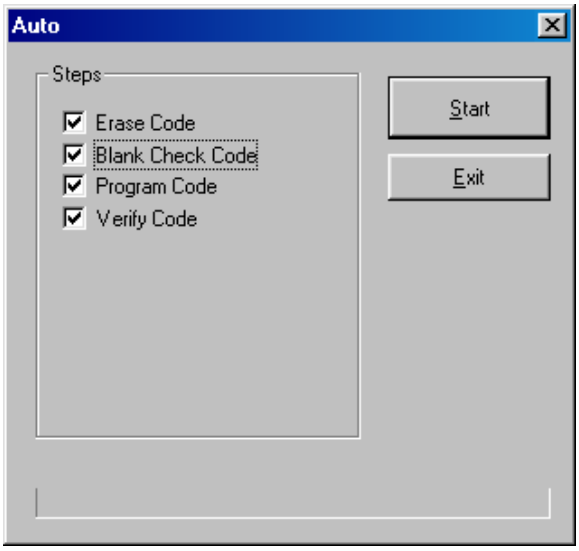
การสร้าง Project File

สำหรับในกรณีที่ต้องการใช้งานโปรแกรม DATABLAZE.EXE ร่วมกับเครื่อง ET-PGM08 สำหรับพัฒนาโปรแกรมของบอร์ด CP-JR08GP32 นั้น ขอแนะนำให้ใช้วิธีการสร้างเป็น Project File เก็บไว้ จะสะดวกที่สุด เนื่องจากในคุณสมบัติของ Project File นั้นสามารถเก็บค่าคุณสมบัติต่างๆที่ต้องใช้ในการทำงานไว้ได้ทั้งหมด ไม่ว่าจะเป็นเบอร์ของ CPU ที่จะใช้โปรแกรม ชื่อไฟล์ที่จะทำการโปรแกรม รวมทั้งคุณสมบัติในการติดต่อสื่อสารต่างๆที่จำเป็น ซึ่งเมื่อผู้ใช้เริ่มทำการสั่งให้เปิดโปรแกรม DATABLAZE.EXE ขึ้นมาในแต่ละครั้งก็ไม่จำเป็นต้องเสียเวลาไปทำการกำหนดคุณสมบัติต่างๆของเครื่องใหม่ให้เสียเวลา เพียงแต่ใช้วิธีการสั่งเปิด Project File ที่สร้างเก็บไว้ขึ้นมาเท่านั้น ค่าคุณสมบัติต่างๆจะถูกกำหนดให้เองโดยอัตโนมัติ ซึ่งวิธีการแบบนี้เหมาะสำหรับการพัฒนาโปรแกรมซึ่งมีความจำเป็นต้องทำการสั่งโปรแกรมข้อมูลให้กับ CPU ด้วยชื่อไฟล์เดิมๆหลายๆครั้ง เนื่องจากอาจต้องย้อนกลับไปแก้ไขโปรแกรมพร้อมกับสั่งแปลโปรแกรมใหม่แล้วจึงกลับมาสั่งโปรแกรม CPU อีกจนกว่าจะได้การทำงานที่ถูกต้อง ซึ่งมีวิธีการดังนี้

1. ต่อสายสัญญาณ RS232 จากเครื่องคอมพิวเตอร์ PC เข้ากับเครื่องโปรแกรม ET-PGM08
2. ต่อสายแพร์ 14 PIN จากขั้วต่อ P1 จากเครื่อง ET-PGM08 เข้ากับขั้ว Download Program ของบอร์ด CP-JR08GP32 พร้อมกับเลือก Jumper และ Slide Switch สำหรับเลือกโหมดการทำงานของบอร์ดไว้ทางด้าน Monitor Mode (MON)
3. จ่ายไฟเลี้ยงวงจรให้กับเครื่อง ET-PGM08 และบอร์ด CP-JR08GP32
4. ทำการเรียกใช้งานโปรแกรม DATABLAZE.EXE เพื่อสั่งงานเครื่องโปรแกรม ET-PGM08
5. ทำการสร้าง Project File โดยการคลิกเมาส์ที่เมนูคำสั่ง Project → New → พร้อมกับกำหนดชื่อของ Project File ตามต้องการซึ่งจะปรากฏหน้าต่างสำหรับให้กำหนดคุณสมบัติต่างๆของ Project File ดังรูป



6. ให้ทำการสั่งงานเครื่องโปรแกรมให้ทำงานตามต้องการ แต่เพื่อความสะดวกขอแนะนำให้เลือกใช้คำสั่ง Auto จะดีที่สุด โดยอาจเลือกคลิกเมาส์ที่เมนูคำสั่ง Operation → Auto หรืออาจทำการคลิกเมาส์ที่ปุ่มคำสั่ง Auto โดยตรงก็ได้เช่นกัน ซึ่งเมื่อเลือกคำสั่งนี้แล้วจะปรากฏหน้าต่างคำสั่งของการ Auto Program ขึ้นมา โดยในเมนูคำสั่งดังกล่าวจะมีคำสั่งย่อยให้เลือกกำหนดเงื่อนไขการทำงานหลายคำสั่ง ให้ทำการคลิกเมาส์หน้าคำสั่งที่ต้องการให้เครื่องโปรแกรมทำงาน โดยถ้าต้องการให้คำสั่งใดทำงานให้คลิกเมาส์จนเกิดเครื่องหมายถูก(√) หน้าคำสั่งนั้นๆ จากนั้นจึงคลิกเมาส์ที่ปุ่มคำสั่ง Start เพื่อสั่งงานโปรแกรม และรอจนเสร็จทุกขั้นตอน



7. หลังจากโปรแกรมข้อมูลให้กับ CPU เสร็จเรียบร้อยแล้วสามารถเลือกกำหนด Jumper และ Slide Switch สำหรับเลือกโหมดการทำงานของบอร์ดไว้ทางด้าน RUN เพื่อให้ CPU ทำงานตามโปรแกรมได้ทันที

*****หมายเหตุ***** สำหรับในกรณีที่เปิดโปรแกรม DATABLAZE.EXE สำหรับสั่งโปรแกรม CPU ดังกล่าวข้างต้นแล้วนั้น อาจไม่จำเป็นต้องสั่งปิดโปรแกรมทุกๆครั้งที่สั่งโปรแกรมเสร็จแล้วก็ได้ แต่อาจใช้วิธีการสลับการทำงานไปยังโปรแกรมอื่นๆก่อนแล้วจึงย้อนกลับมาใช้งานโปรแกรมนี้ในภายหลังอีกก็ได้ ซึ่งเมื่อสลับการทำงานจากโปรแกรมอื่นๆกลับมายังโปรแกรม DATABLAZE.EXE นี้อีกในครั้งใด โปรแกรมจะทำการตรวจสอบคุณสมบัติต่างๆของไฟล์ที่กำหนดการเชื่อมโยงกับ Project File ไว้ ซึ่งถ้าหากว่าไฟล์นั้น เกิดการเปลี่ยนแปลงก็จะแสดงคำเตือนให้ทราบเพื่อให้ผู้ใช้ยืนยันเสมอ

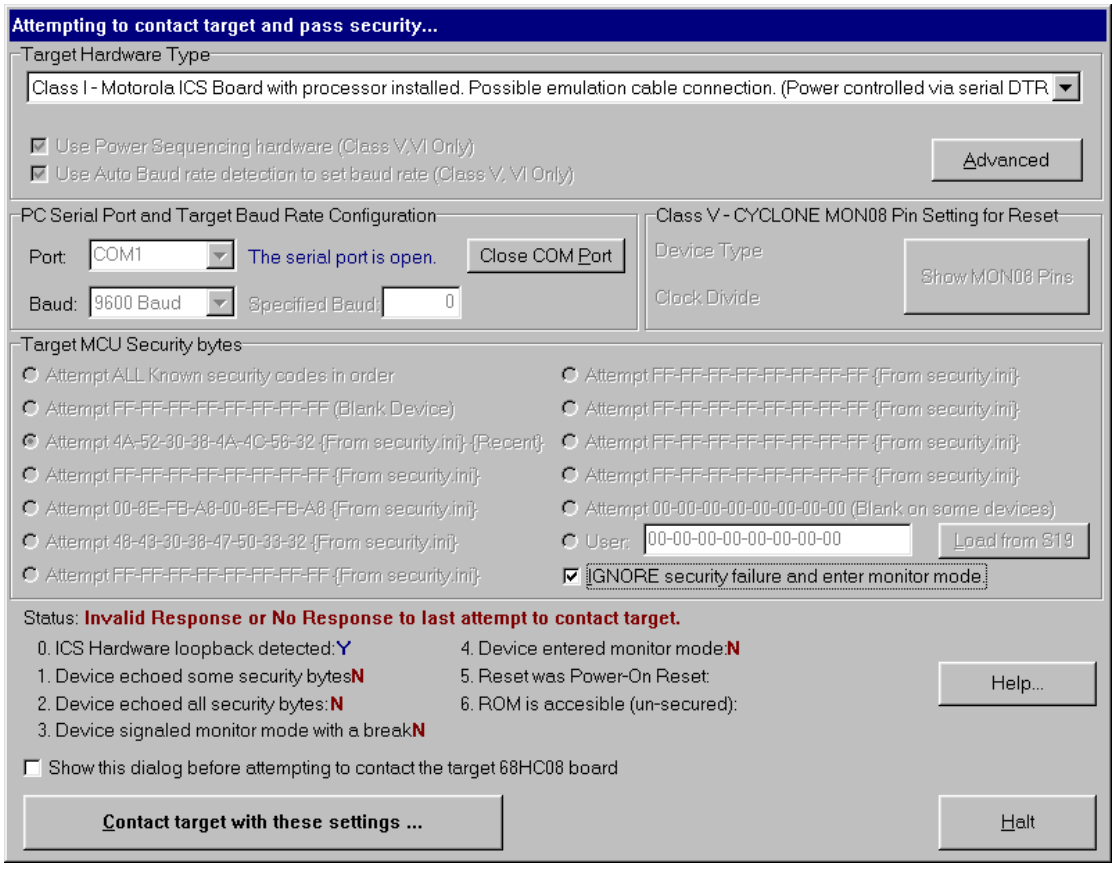
การใช้งานโปรแกรม PROG08SZ สำหรับ Download โปรแกรม

โปรแกรม PROG08SZ เป็นโปรแกรมสำหรับสั่ง โปรแกรมข้อมูลแบบ Motorola HEX ให้กับ CPU ตระกูล MC68HC08 (Flash Programmer) ซึ่งตามปรกติต้องใช้คู่กับเครื่องพัฒนาโปรแกรมของต่างประเทศ (P&E) แต่ก็สามารถนำมาใช้กับเครื่องโปรแกรมของ บริษัท อีทีที จำกัด ได้เช่นกัน แต่การใช้งานโปรแกรมจะดูซับซ้อนไปบ้าง อาจยากสำหรับผู้ใช้ที่เริ่มต้นใหม่และขาดทักษะในการพัฒนาโปรแกรมด้านไมโครคอนโทรลเลอร์อยู่บ้าง เพราะลักษณะการสั่งงานเครื่องต่างๆ ผู้ใช้ต้องค้อนข้างรู้จักโครงสร้างและคุณสมบัติของ CPU ดีพอสมควร แต่ก็สามารถใช้งานได้ดี

การใช้งานโปรแกรม PROG08SZ กับเครื่องโปรแกรม ET-PGM08

ชุดพัฒนา ET-PGM08 เอง นอกจากจะสามารถสั่งงานผ่านโปรแกรม ET-PGM08 แล้ว ยังสามารถสั่งงานผ่านโปรแกรม PROG08SZ ได้อีกด้วย โดยมีวิธีการดังนี้

1. ต่อสายสัญญาณ RS232 จากเครื่องคอมพิวเตอร์ PC เข้ากับเครื่องโปรแกรม ET-PGM08
2. ต่อสายแพร์ขนาด 14 PIN จากขั้วต่อ P1 จากเครื่อง ET-PGM08 เข้ากับขั้ว Download Program ของบอร์ด CP-JR08GP32 พร้อมกับเลือก Jumper และ Slide Switch สำหรับเลือกโหมดการทำงานของบอร์ดไว้ทางด้าน Monitor Mode (MON)
3. จ่ายไฟเลี้ยงวงจรให้กับเครื่อง ET-PGM08 และบอร์ด CP-JR08GP32
4. ทำการเรียกใช้งานโปรแกรม PROG08SZ.EXE เพื่อสั่งงานเครื่องโปรแกรม ET-PGM08 ซึ่งจะได้ผลดังรูป

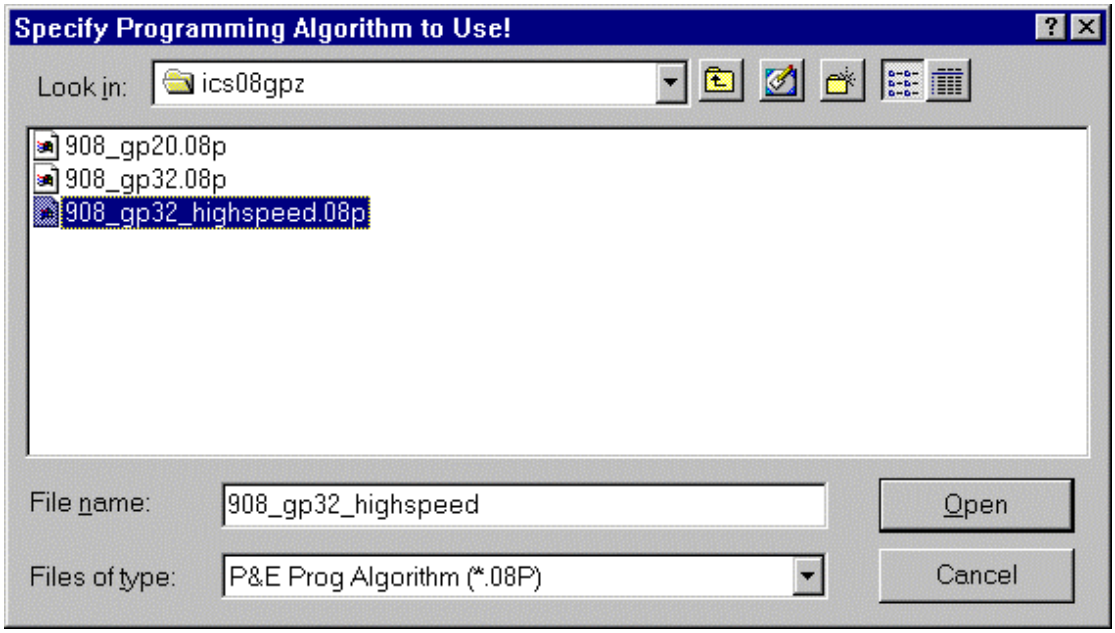


ให้เลือกกำหนดคุณสมบัติต่างๆให้กับโปรแกรมสำหรับทำการติดต่อกับบอร์ด CP-JR08GP32 ดังนี้

- Target Hardware Type ให้เลือกเป็น Class I – Motorola ICS Board with processor installed Possible emulation cable connection…(Power Controlled via serial DTR line)
- Target MCU Security byte ให้เลือกกำหนดเป็น IGNORE Security failure and enter monitor mode
- PC Serial Port and Target Baudrate Configuration ในหน้าต่างของ Port ให้เลือกกำหนดตามที่ต่อสายไว้จริง เช่น Com1 หรือ Com2 ส่วน Baud นั้นให้เลือกเป็น 9600 Baud ซึ่งถ้าต้องแก้ไขค่าทั้งสองให้เลือกที่ Close Com Port เสียก่อนจึงจะสามารถเข้าไปแก้ไขค่าได้

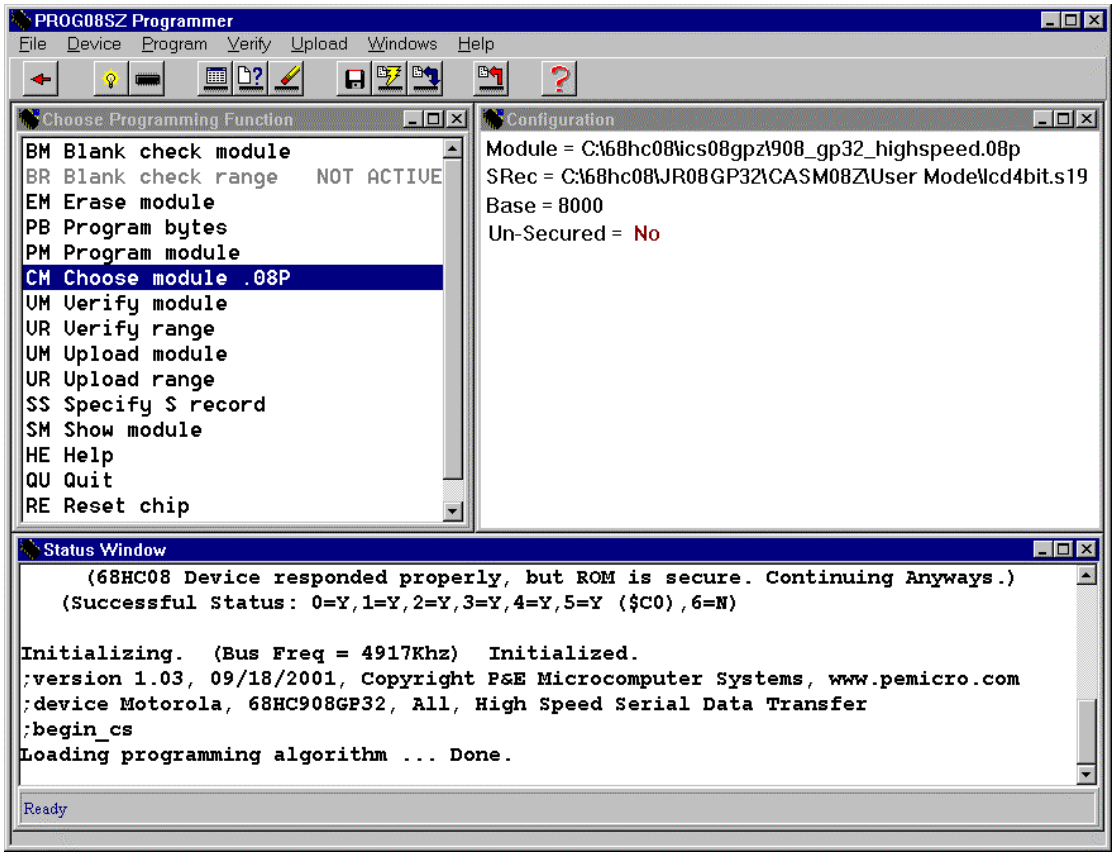
จากนั้นให้เลือก Contact target with these setting… ซึ่งถ้าทุกอย่างถูกต้องโปรแกรมจะข้ามไปทำงานยังขั้นตอนถัดไป แต่ถ้ามีข้อผิดพลาดเกิดขึ้น คือโปรแกรมยังไม่สามารถสั่งงานเครื่อง ET-PGM08 เพื่อบังคับให้ CPU เข้าสู่การทำงานใน Monitor โหมดได้ โปรแกรมจะยังคงค้างอยู่หน้าต่างนี้ต่อไปอีก ถ้าต้องการยกเลิกการทำงานของโปรแกรมให้เลือก Halt แต่ถ้าต้องการทำงานต่อไปให้ตรวจสอบหาข้อผิดพลาดพร้อมกับแก้ไขแล้วจึงเลือก Contact target with these setting อีกครั้งหนึ่ง

5. เมื่อผ่านมาถึงขั้นตอนนี้ โปรแกรมจะให้เลือกกำหนด Algorithm ของ CPU ที่ต้องการจะ COPY ดังรูป



ให้ทำการเลือก Algorithm ในการโปรแกรมสำหรับเบอร์ CPU ที่ต้องการจะโปรแกรม โดยไฟล์สำหรับใช้กำหนด Algorithm สำหรับ CPU แต่ละเบอร์นั้นจะมีสกุลเป็น 08P ซึ่งบอร์ด CP-JR08GP32 นั้น จะใช้ CPU เบอร์ MC68HC908GP32 แบบ Flash ดังนั้นให้เลือกเป็น 908_gp32_highspeed.08p แล้วเลือก Open

6. เมื่อมาถึงขั้นตอนนี้ก็สามารถสั่งงานตัวเครื่อง ET-PGM08 ให้ทำงานตามเมนูคำสั่งต่างๆได้ตามต้องการ โดยลักษณะของหน้าต่างโปรแกรมจะเป็นดังรูป



ซึ่งจะเห็นได้ว่าหน้าต่างของโปรแกรม PROG08SZ นั้นจะถูกแบ่งออกเป็น 3 ส่วนด้วยกัน โดยเป็นหน้าต่างสำหรับสั่งงาน 1 หน้าต่าง และหน้าต่างสำหรับแสดงผลอีก 2 หน้าต่างดังนี้

1. **หน้าต่าง** Configuration จะใช้สำหรับแสดงค่า Configuration ต่างๆของโปรแกรมที่เลือกไว้

2. **หน้าต่าง** Status Windows ใช้สำหรับแสดงสถานะการทำงานของคำสั่งต่างๆของโปรแกรม

3. **หน้าต่าง** Choose Programming Function ใช้สำหรับเลือกคำสั่งที่ต้องการสั่งงานตัวเครื่อง โดยต้องการเลือกใช้คำสั่งใดให้ทำการ Double Click ที่คำสั่งนั้นๆตามต้องการ โดยมีคำสั่งต่างๆดังนี้

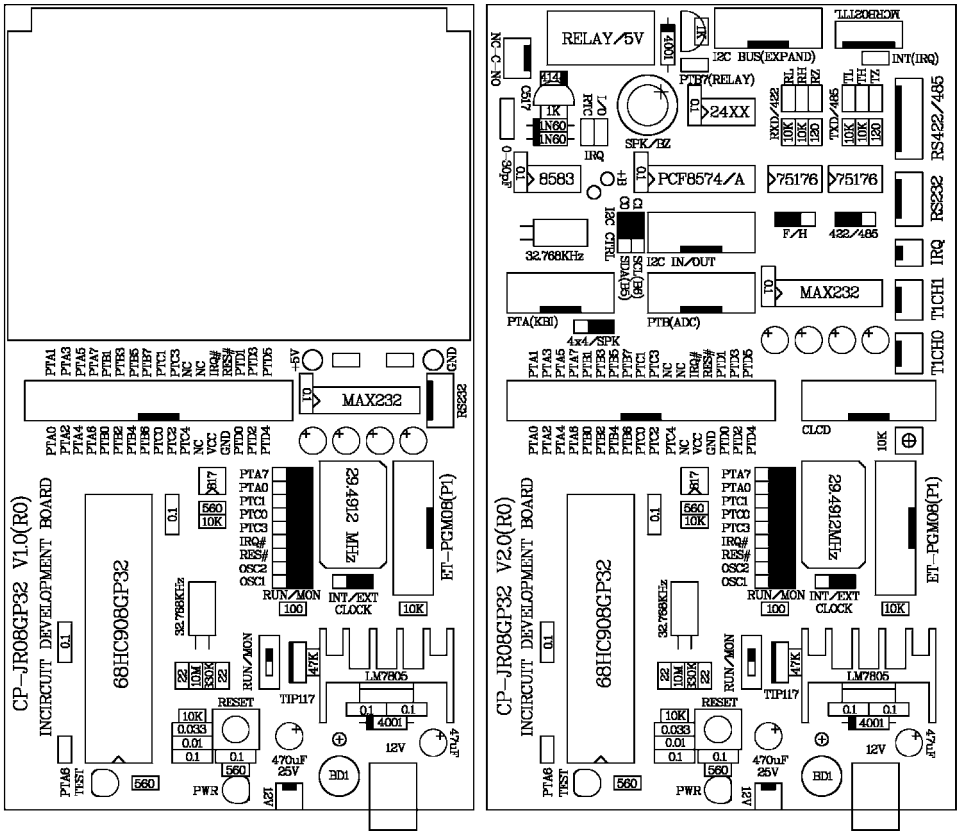
- BM : Blank check module ใช้สำหรับตรวจสอบหน่วยความจำโปรแกรมทั้งหมดของ CPU ว่ามีข้อมูลอยู่หรือไม่
- BR : Blank check range ใช้สำหรับตรวจสอบหน่วยความจำโปรแกรมในช่วงที่กำหนดว่ามีข้อมูลอยู่หรือไม่
- EM : Erase module ใช้สำหรับสั่งลบข้อมูลในหน่วยความจำของโปรแกรมของ CPU ทั้งหมด
- PB : Program byte ใช้สำหรับสั่งโปรแกรม CPU ตามตำแหน่งที่กำหนด
- PM : Program module ใช้สำหรับสั่งโปรแกรม CPU ทุกตำแหน่ง
- CM : Choose module เลือกกำหนด Algorithm ให้กับ CPU แต่ละเบอร์
- VM : Verify module ใช้สำหรับเปรียบเทียบข้อมูลในหน่วยความจำโปรแกรมของ CPU ทั้งหมดกับข้อมูลใน Buffer ที่โหลดไว้แล้ว

- VR : **Verify range** ใช้สำหรับเปรียบเทียบข้อมูลในหน่วยความจำโปรแกรมของ CPU ในช่วงที่กำหนดกับข้อมูลใน Buffer ที่โหลดไว้แล้ว
- UM : **Upload module** ใช้สำหรับอ่านข้อมูลในหน่วยความจำโปรแกรมของ CPU ทั้งหมดไปบันทึกเป็นไฟล์ไว้
- UR : **Upload Range** ใช้สำหรับอ่านข้อมูลในหน่วยความจำโปรแกรมของ CPU ในช่วงที่กำหนดไปบันทึกเป็นไฟล์ไว้
- SS : **Specify S Record** ใช้สำหรับโหลดข้อมูลที่เป็น Motorola HEX Format (S19) มาเก็บไว้ใน Buffer ของโปรแกรม PROG08SZ
- SM : **Show module** ใช้สำหรับแสดงค่าข้อมูลในหน่วยความจำโปรแกรมของ CPU ตามตำแหน่งแอดเดรสที่กำหนดทางหน้าจอ
- HE : **Help** ใช้สำหรับเปิดไฟล์ Help ต่างๆของโปรแกรม
- QU : **Quit** จบการทำงานของโปรแกรม
- RE : **Reset chip** สั่ง Reset CPU เพื่อเริ่มต้นการทำงานใน Monitor โหมดใหม่
- VC : **Verify checksum** เปรียบเทียบค่า Checksum ของ CPU กับ Buffer

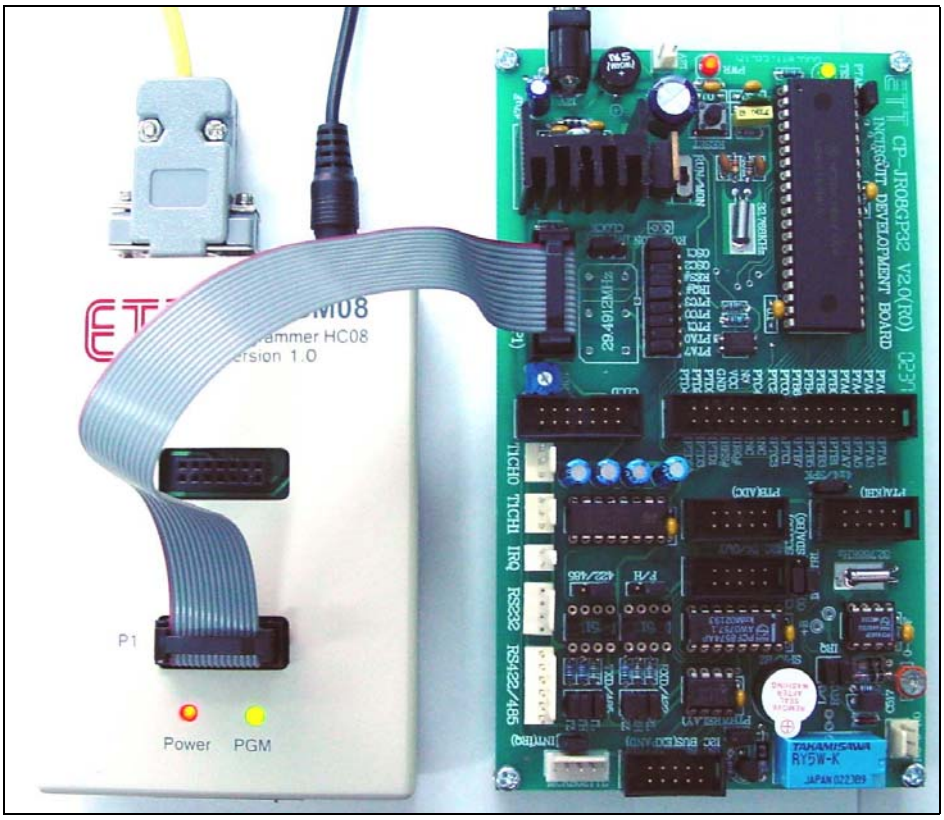
โดยสำหรับในกรณีที่ต้องการจะทำการ Download โปรแกรมให้กับ CPU เบอร์ MC68HC908GP32 ของบอร์ด CP-JR08GP32 นั้น ให้ทำดังนี้

- เลือก Choose module เป็น 908_gp32_highspeed.08P
- เลือก **SS : Specify S Record** พร้อมกับกำหนดชื่อไฟล์ S19 ที่ต้องการนำมาโปรแกรมให้กับ CPU มารอไว้ใน buffer
- **เลือก EM : Erase module** เพื่อลบข้อมูลเก่าในตัว CPU ทิ้งก่อน และรอจนกว่า Status Windows จะรายงานผลเป็น Erasing…Module has been erased
- เลือก **PM : Program module** เพื่อสั่งโปรแกรมข้อมูลใน Buffer ให้กับ CPU โดยที่หน้าต่าง Status Windows จะแสดง Programming Address $xxxx…Programmed
- **เลือก VM : Verify module** เพื่อสั่งเปรียบเทียบข้อมูลใน Buffer กับข้อมูลใน CPU โดยที่หน้าต่าง Status Windows จะแสดง Verifying Address $xxxx…Verified

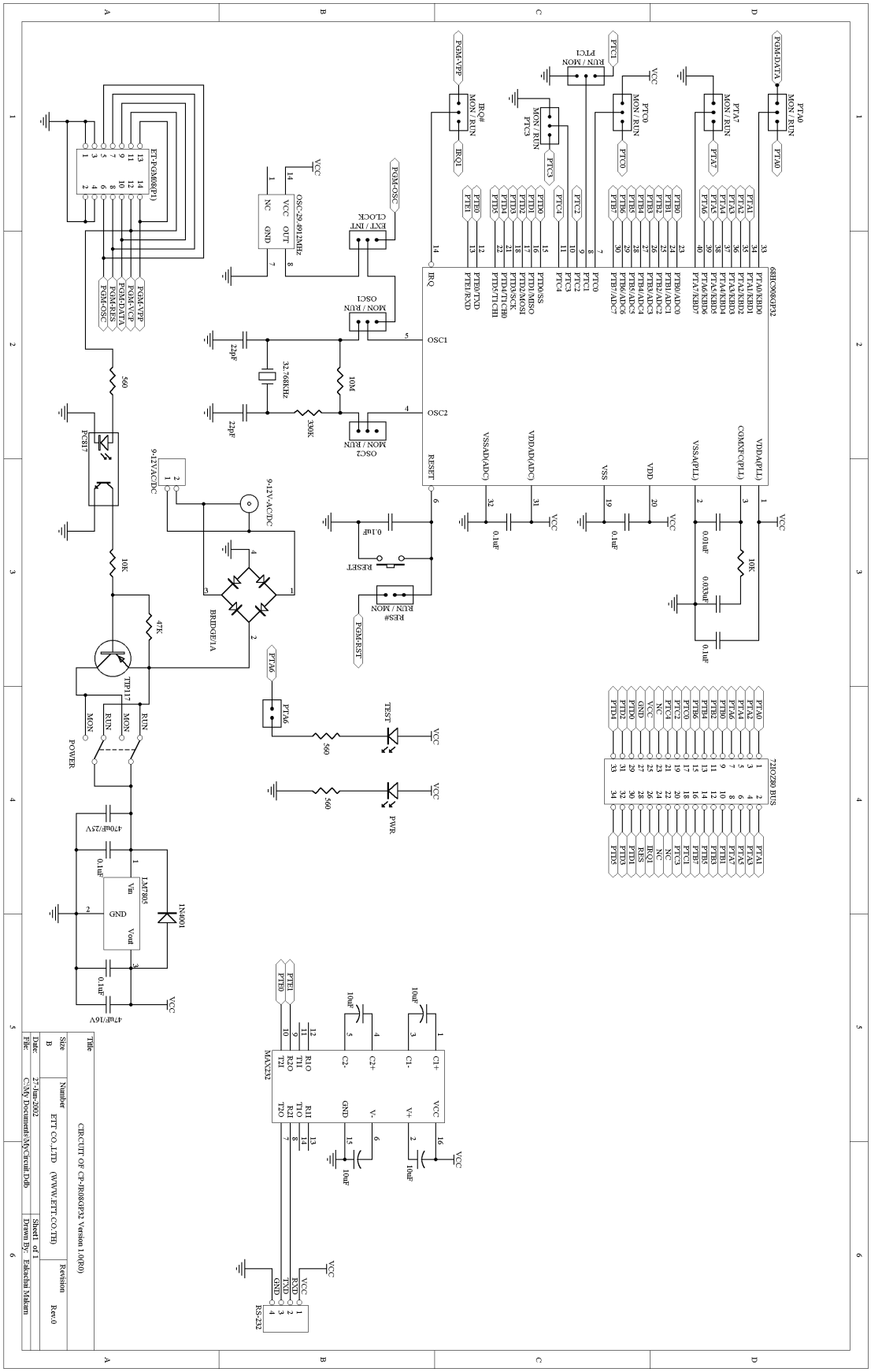
ถ้าทุกขั้นตอนถูกต้องก็สามารถเปลี่ยนโหมดการทำงานของบอร์ดไปยัง RUN Mode เพื่อให้ CPU ทำงานตามคำสั่งของโปรแกรมที่ถูกโปรแกรมไปแล้วทันที



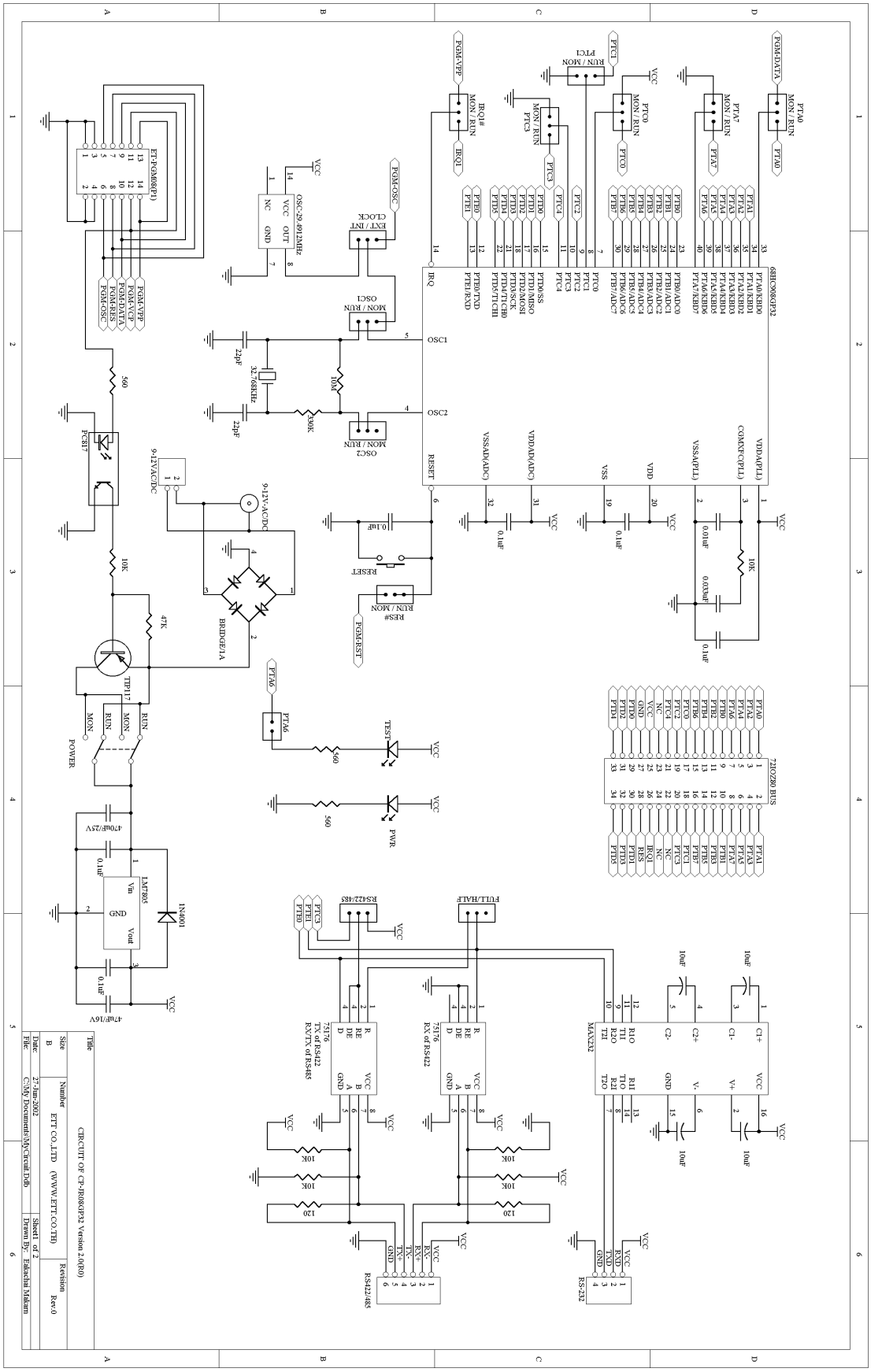
รูปแสดง การกำหนด Jumper ของบอร์ด CP-JR08GP32 สำหรับพัฒนาโปรแกรมของบอร์ด



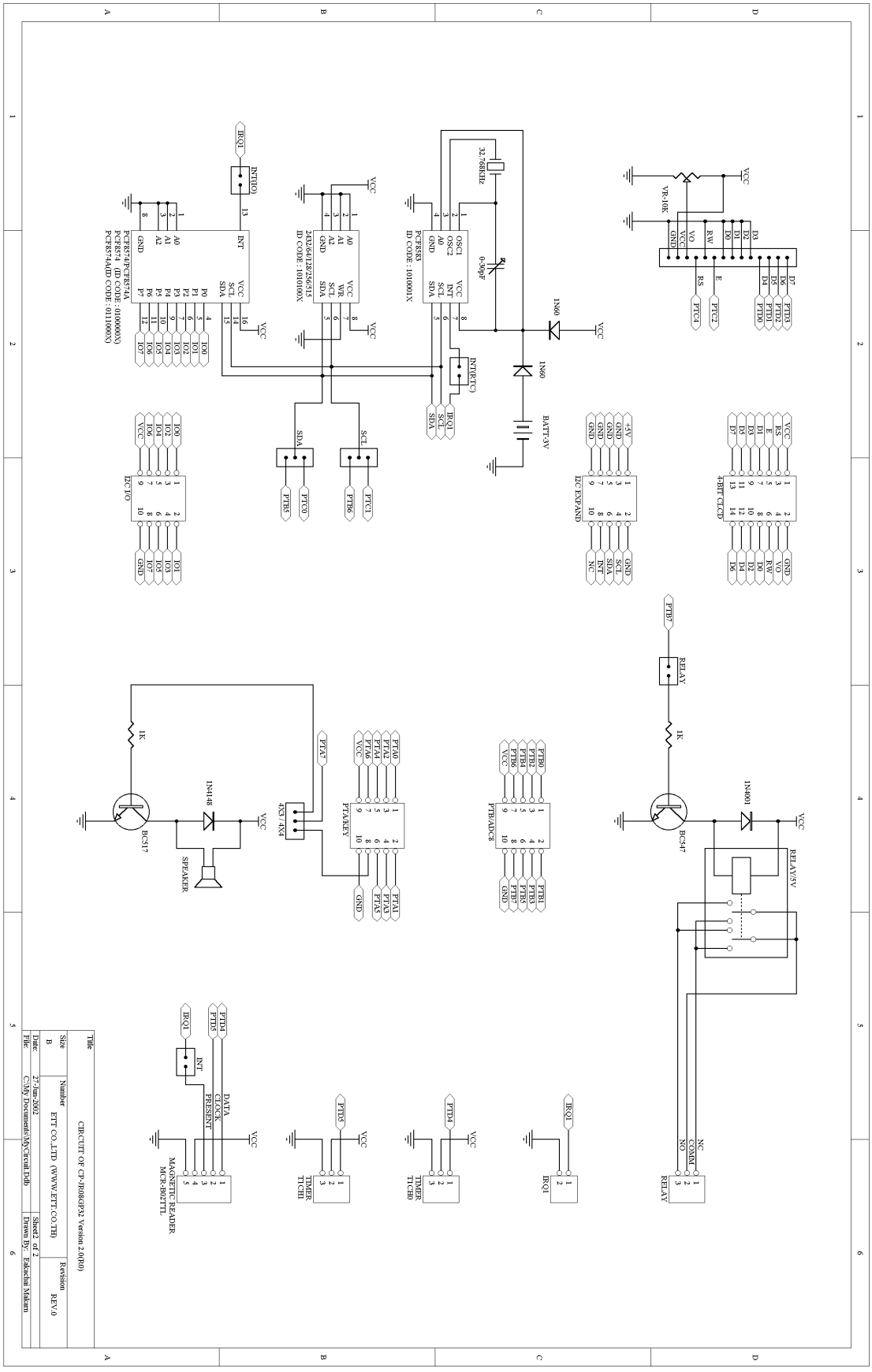
รูปแสดงวิธีการต่อสายบอร์ด CP-JR08GP32 ร่วมกับเครื่อง ET-PGM08 สำหรับพัฒนาบอร์ด



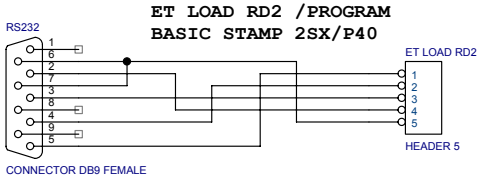
รูป แสดงวงจรของบอร์ด CP-JR08GP32 V1.0 & V1.0 EXPANSION



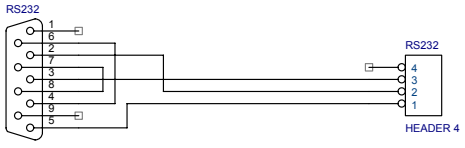
รูป แสดงวงจรของบอร์ด CP-JR08GP32 V2.0 (1/2)



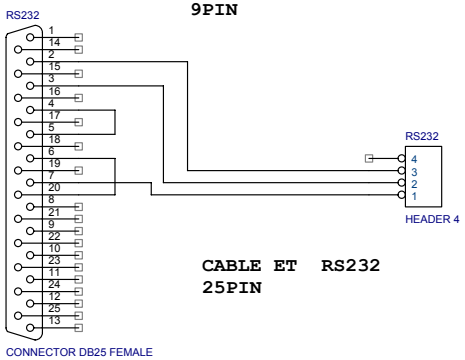
รูป แสดงวงจรของบอร์ด CP-JR08GP32 V2.0 (2/2)



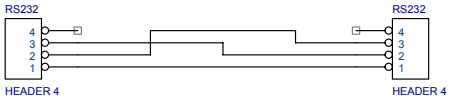
CABLE DOWNLOAD
ET LOAD RD2 /PROGRAM
BASIC STAMP 2SX/P40



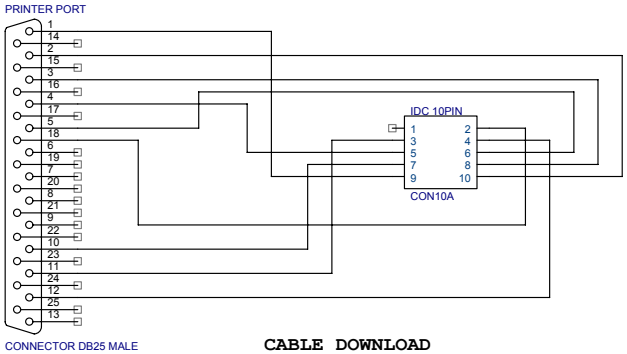
CABLE ET RS232
9PIN



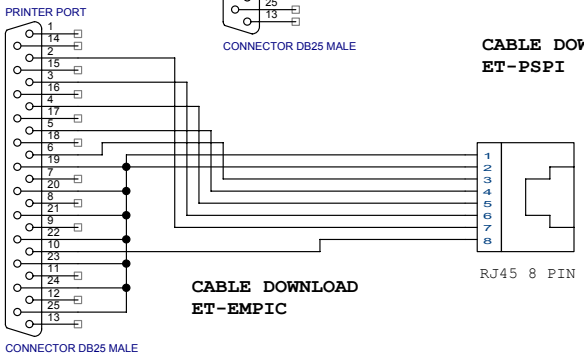
CABLE ET RS232
25PIN



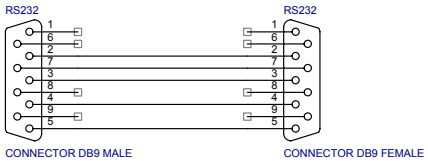
CABLE RS232 ET 4PIN TO ET 4PIN



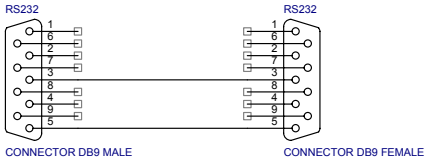
CABLE DOWNLOAD
ET-PSPI



CABLE DOWNLOAD
ET-EMPIC



CABLE ET-PGM08 V1



CABLE RS232
ET-CNT6/P3

Title	ETT CO.,LTD.	
Size B	Document Number CABLE ETT	Rev {RevCode}
Date: Saturday, June 15, 2002	Sheet 1 of 1	