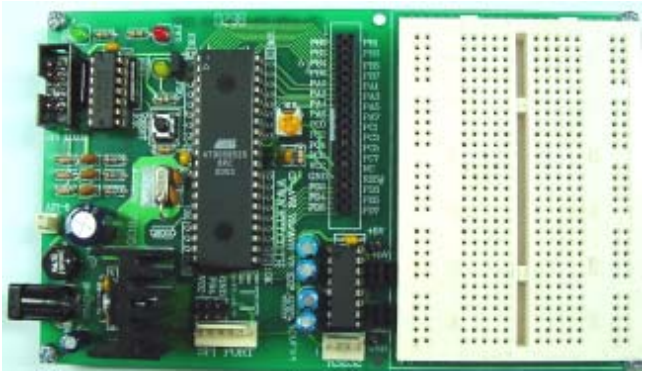
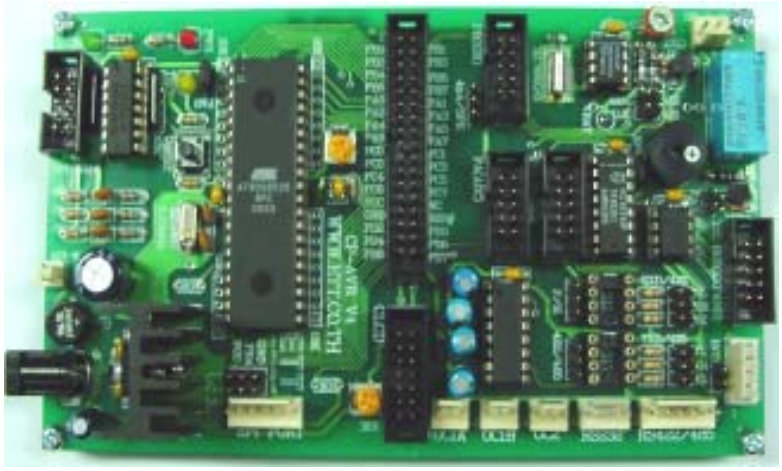
คู่มือการใช้งานบอร์ด ไมโครคอนโทรลเลอร์

CP-AVR V3.0
CP-AVR V3.0 EXP
CP-AVR V4.0





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

www.ett.co.th

ชื่อหนังสือ “คู่มือการใช้งานบอร์ดไมโครคอนโทรลเลอร์ CP-AVR V3 & V4”

ISBN 974-90821-6-8

ผู้เขียน นายกิตติพงษ์ กาพาด

พิมพ์ครั้งที่ 1

25 พฤศจิกายน 2545

จำนวน 40 หน้า

ราคา 50 บาท

พิมพ์จำนวน 1000 เล่ม

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

สงวนลิขสิทธิ์ตามพระราชบัญญัติลิขสิทธิ์ พ.ศ. 2537
ห้ามลอกเลียนไม่ว่าส่วนหนึ่งส่วนใดของหนังสือเล่มนี้
ไม่ว่าในรูปแบบใดนอกจากจะได้รับอนุญาตเป็นลาย
ลักษณ์อักษรจากผู้จัดพิมพ์

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

www.ett.co.th

บทนำ

ในปัจจุบัน อุปกรณ์ไมโครคอนโทรลเลอร์ได้พัฒนาไปเร็วมากดังจะเห็นได้จากการที่มีไมโครคอนโทรลเลอร์หลายตระกูลออกมาวางขายในท้องตลาด และผู้ผลิตไมโครคอนโทรลเลอร์เหล่านี้ต่างก็ได้พยายามพัฒนาขีดความสามารถในการทำงานของไมโครคอนโทรลเลอร์ของตนเองให้มีมากขึ้น ทั้งในด้านการเพิ่มความเร็วของ CPU การเพิ่มฟังก์ชั่นการทำงานภายในให้มีมากขึ้น ในขณะที่ไมโครคอนโทรลเลอร์บางเบอร์จะถูกออกแบบมาเพื่อให้ผู้ใช้งานสามารถเขียนโปรแกรมได้ง่ายขึ้นด้วย ซึ่ง AVR ไมโครคอนโทรลเลอร์ก็เป็น CPU อีกตระกูลหนึ่งที่เป็นที่นิยมนำมาใช้งานในปัจจุบัน เพราะมันเป็นไมโครคอนโทรลเลอร์ที่มีทั้งความเร็วและการออกแบบโครงสร้างภายในที่เป็นระเบียบทำให้ผู้ใช้งานสามารถเขียนโปรแกรมได้ง่าย

ทางบริษัท อีทีที จำกัด ได้ออกแบบบอร์ด AVR ไมโครคอนโทรลเลอร์ขึ้นมาใหม่ 3 รุ่นด้วยกันซึ่งประกอบไปด้วย CP-AVR V3.0, CP-AVR V3.0 EXP และ CP-AVR V4.0 ซึ่งรายละเอียดของบอร์ดไมโครคอนโทรลเลอร์แต่ละรุ่นท่านสามารถศึกษาได้จากเนื้อหาภายในของคู่มือเล่มนี้ โดยบอร์ดไมโครคอนโทรลเลอร์ทั้ง 3 รุ่นนี้สามารถใช้ได้กับ AVR ไมโครคอนโทรลเลอร์เบอร์ AT90S8535 หรือ ATmega163 หรือเบอร์อื่น ที่มีตำแหน่งขาของอุปกรณ์ที่ตรงกันเช่นเบอร์ ATmega323 เป็นต้น

ในการพัฒนาโปรแกรมสำหรับผู้เริ่มต้นนั้น ท่านสามารถศึกษาโปรแกรมตัวอย่างได้จากแผ่น CD-ROM ที่ท่านจะได้รับเมื่อท่านซื้อบอร์ด AVR ไมโครคอนโทรลเลอร์ ซึ่งโปรแกรมตัวอย่างเหล่านี้จะถูกเขียนขึ้นด้วยภาษาแอสเซมบลีโดยใช้โปรแกรม Astudio4 และเขียนด้วยภาษาเบสิกโดยใช้โปรแกรม BASCOM นอกจากนั้นแล้วในแผ่น CD-ROM นี้ยังบรรจุโปรแกรม ASSEMBLER โปรแกรมสำหรับดาวน์โหลดข้อมูล Datasheet และโปรแกรมอื่นๆ ที่จำเป็นต่อการพัฒนางานของท่านไว้ด้วย

พฤศจิกายน 2002

ทีมงาน อีทีที

สารบัญ

เรื่อง	หน้า
- ลักษณะโดยทั่วไป	1
- แหล่งจ่ายไฟ	1
- การใช้งานพอร์ตของ CPU เป็น I/O Port (Input/Output Port)	5
- การใช้งานขั้วต่อ PB(KBI)	6
- การใช้งานขั้วต่อ PA(ADC)	7
- การใช้งานขั้วต่อ I^2C IN/OUT	7
- การใช้งานขั้วต่อ I^2C BUS EXPAND	8
- การใช้งานเครื่องอ่านบัตรแถบแม่เหล็ก (MAGNETIC-CARD READER)	8
- การใช้งาน OUTPUT RELAY	9
- การใช้งานลำโพงขนาดเล็ก หรือ BUZZER	10
- การใช้งานจอแสดงผลแบบ LCD (Dot-Matrix Character LCD)	10
- การเชื่อมต่อกับอุปกรณ์ I^2C BUS	12
- การใช้งาน Interrupt ของอุปกรณ์ I^2C	13
- แอดเดรสของอุปกรณ์ I^2C	13
- การใช้งาน I^2C RTC เบอร์ PCF8583	14
- การใช้งานหน่วยความจำ EEPROM (24XX)	16
- การใช้งาน I/O Port แบบ I^2C (PCF8574/A)	17
- การใช้งานพอร์ตสื่อสารอนุกรม SPI/RS232/RS422/RS485	19
- การสื่อสารอนุกรม SPI	19
- การสื่อสารอนุกรมแบบ RS232	20
- การสื่อสารอนุกรมแบบ RS422	22
- การสื่อสารอนุกรมแบบ RS485	23
- การกำหนด Jumper สำหรับการสื่อสารแบบ RS422/485	24
- การใช้งานโปรแกรม w95s8535v5 และ w95mega163v1 สำหรับดาวน์โหลดข้อมูล	26
- แนะนำการใช้โปรแกรม Astudio4 (AVR assembler)	30
- วงจรของบอร์ด CP-AVR V3	36
- วงจรของบอร์ด CP-AVR V4	37
- วงจรของบอร์ด CP-AVR V4 ต่อ	38

CP-AVR

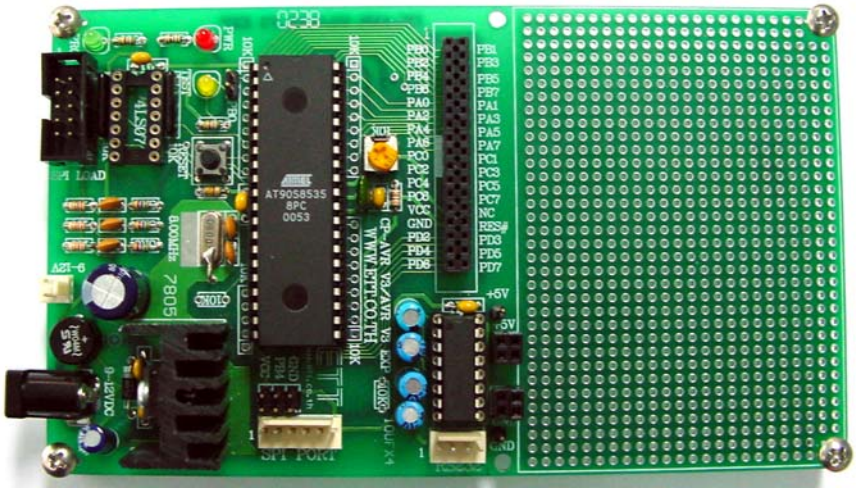
ลักษณะโดยทั่วไป

บอร์ดไมโครคอนโทรลเลอร์ CP-AVR รุ่น CP-AVR V3.0, CP-AVR V3.0 EXP และ CP-AVR V4.0 สามารถใช้ได้กับไมโครคอนโทรลเลอร์ตระกูล AVR เบอร์ AT90S8535 และ ATmega163 โดยจะทำงานที่ความถี่สูงสุด 8 MHz ลักษณะของบอร์ดไมโครคอนโทรลเลอร์ในกลุ่ม CP-AVR นี้ จะแบ่งออกเป็น 3 รุ่น ให้ผู้ใช้ได้เลือกใช้งานกันตามความเหมาะสมดังนี้คือ

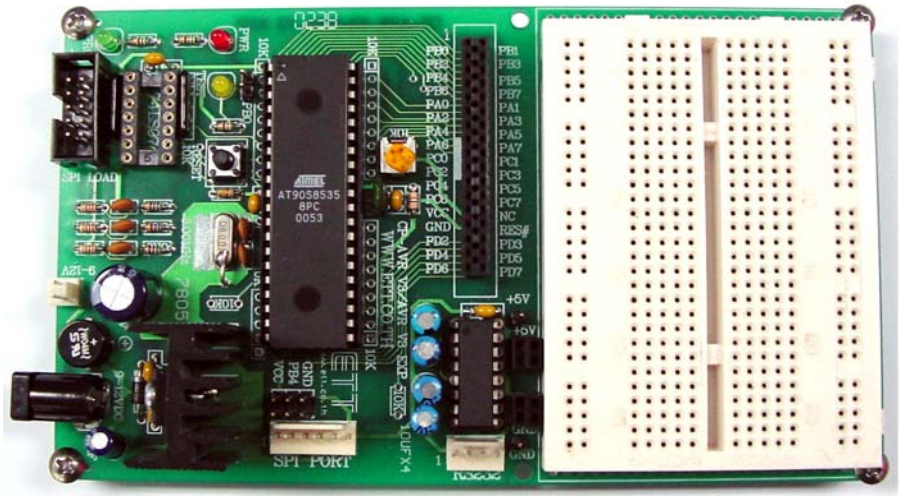
- CP-AVR V3.0 เป็นบอร์ดไมโครคอนโทรลเลอร์ ซึ่งออกแบบวงจรเฉพาะส่วนพื้นฐานที่จำเป็น เช่น แหล่งจ่ายไฟ วงจรรีเซ็ต วงจรกำเนิดความถี่สัญญาณนาฬิกา วงจรสำหรับ Download โปรแกรม และวงจรสื่อสารอนุกรม ส่วนวงจร I/O ภายนอกนั้น จะไม่ได้จัดเตรียมไว้ให้ด้วย แต่จะทำการต่อสัญญาณ I/O ต่างๆจาก CPU มาไว้ยังขั้วต่อ Connector สำหรับให้ผู้ใช้นำไปเชื่อมต่อกับอุปกรณ์ I/O ภายนอกได้โดยง่าย และยังมีพื้นที่เอนกประสงค์สำหรับให้ผู้ใช้ออกแบบวงจร I/O และต่อวงจร I/O เพิ่มเติมได้เอง เหมาะสำหรับผู้ใช้ที่ต้องการนำบอร์ดไปใช้พัฒนางานต้นแบบโดยการสร้าง I/O ต่างๆขึ้นมาใช้งานเอง
- CP-AVR V3.0 EXPANSION จะมีลักษณะเดียวกันกับบอร์ด CP-AVR V3.0 แต่จะมีแผง Photo Board สำหรับให้ผู้ใช้ต่อทดลองวงจร I/O อย่างง่ายๆได้เอง เหมาะสำหรับผู้ใช้ที่ต้องการศึกษาเรียนรู้และต้องการทดลองวงจร I/O ต่างๆ ร่วมกับ CPU อย่างง่ายๆ
- CP-AVR V4.0 เป็นบอร์ดไมโครคอนโทรลเลอร์ที่มีการออกแบบวงจรสำหรับเชื่อมต่อกับอุปกรณ์ I/O ภายนอกอื่นๆที่มีความจำเป็นไว้รองรับการใช้งานในลักษณะต่างๆ เพื่อให้ผู้ใช้งานสามารถนำบอร์ดไปใช้งานในลักษณะงานที่แตกต่างกันได้ โดยไม่ต้องดัดแปลงวงจร หรือ อาจดัดแปลงวงจรเพียงเล็กน้อยสำหรับงานบางอย่าง ซึ่งบอร์ดรุ่นนี้เหมาะสำหรับกลุ่มผู้ที่ต้องการนำ บอร์ดไมโครคอนโทรลเลอร์ไปใช้งานจริงๆแต่ไม่สะดวกที่จะสร้างบอร์ดเอง

แหล่งจ่ายไฟ (POWER SUPPLY)

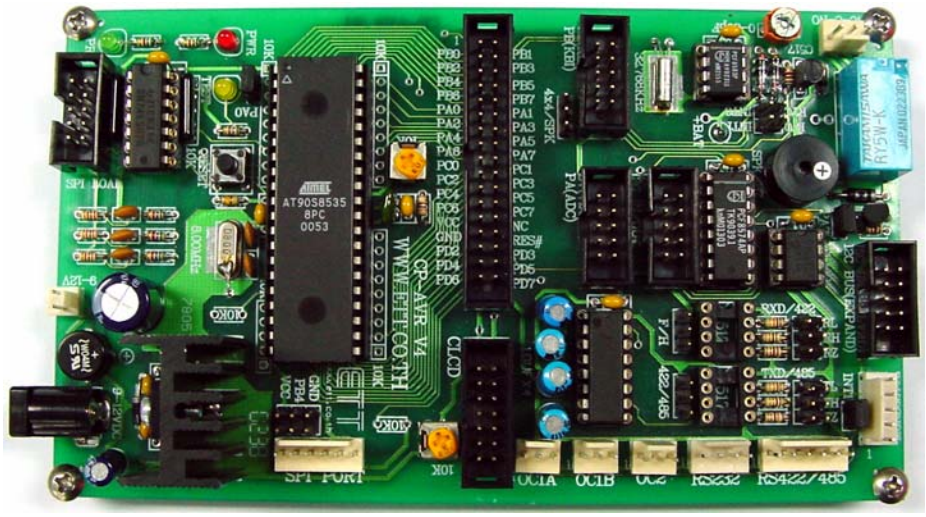
สำหรับแหล่งจ่ายไฟของบอร์ดในกลุ่ม CP-AVR นั้น จะสามารถต่อใช้งานได้ทั้งกับไฟกระแสตรงและกระแสสลับ เนื่องจากในบอร์ดได้จัดเตรียมวงจร RECTIFIER แบบ BRIDGE พร้อมวงจร FILTER และ REGULATOR ขนาด +5V ไว้ให้อย่างครบถ้วนอยู่แล้ว โดยผู้ใช้สามารถป้อนแรงดันไฟตรงหรือไฟสลับที่มีระดับแรงดันประมาณ 9-12V ให้กับบอร์ด โดยสามารถเลือกต่อกับขั้ว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



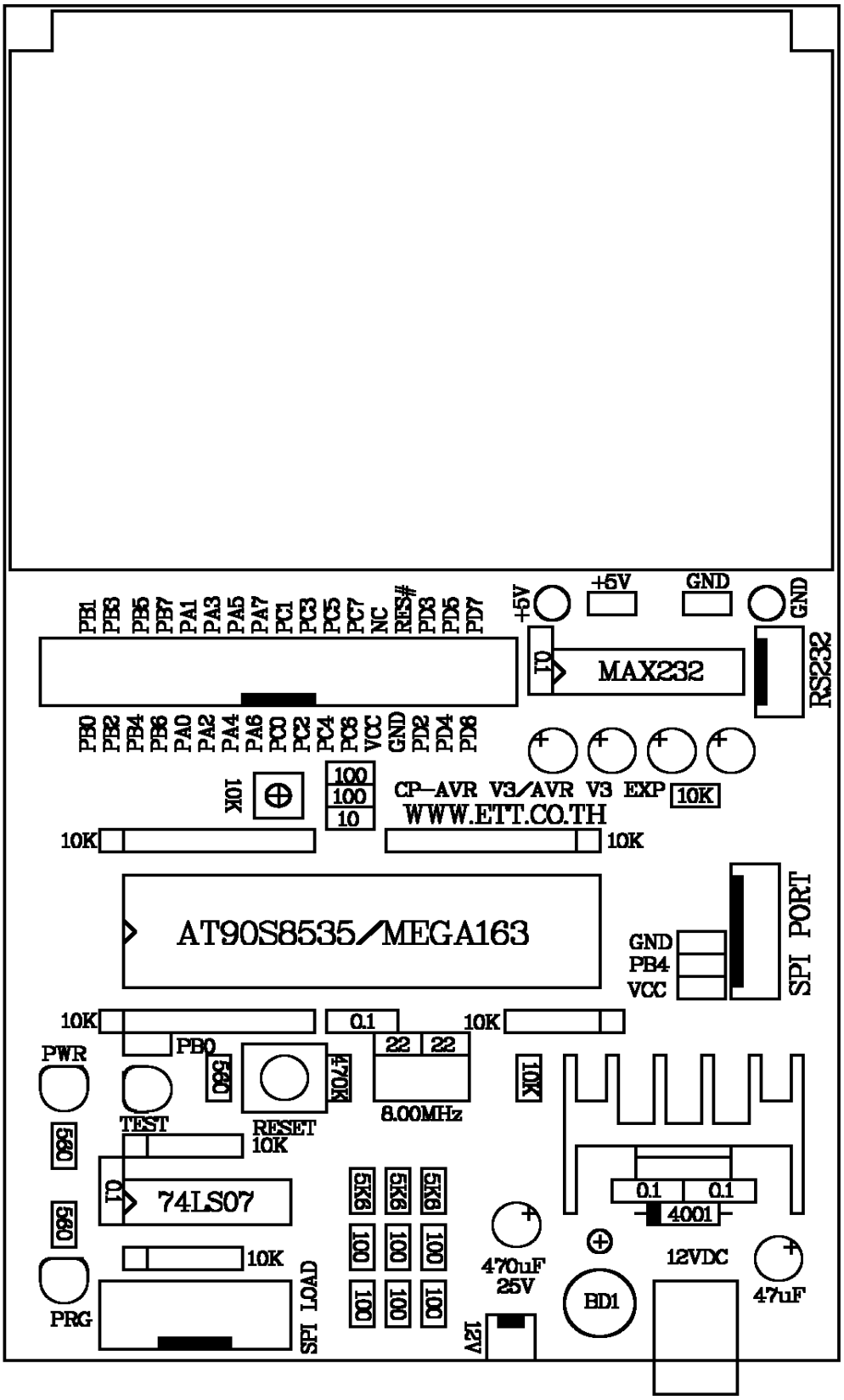
รูปแสดง บอร์ดไมโครคอนโทรลเลอร์รุ่น CP-AVR V3



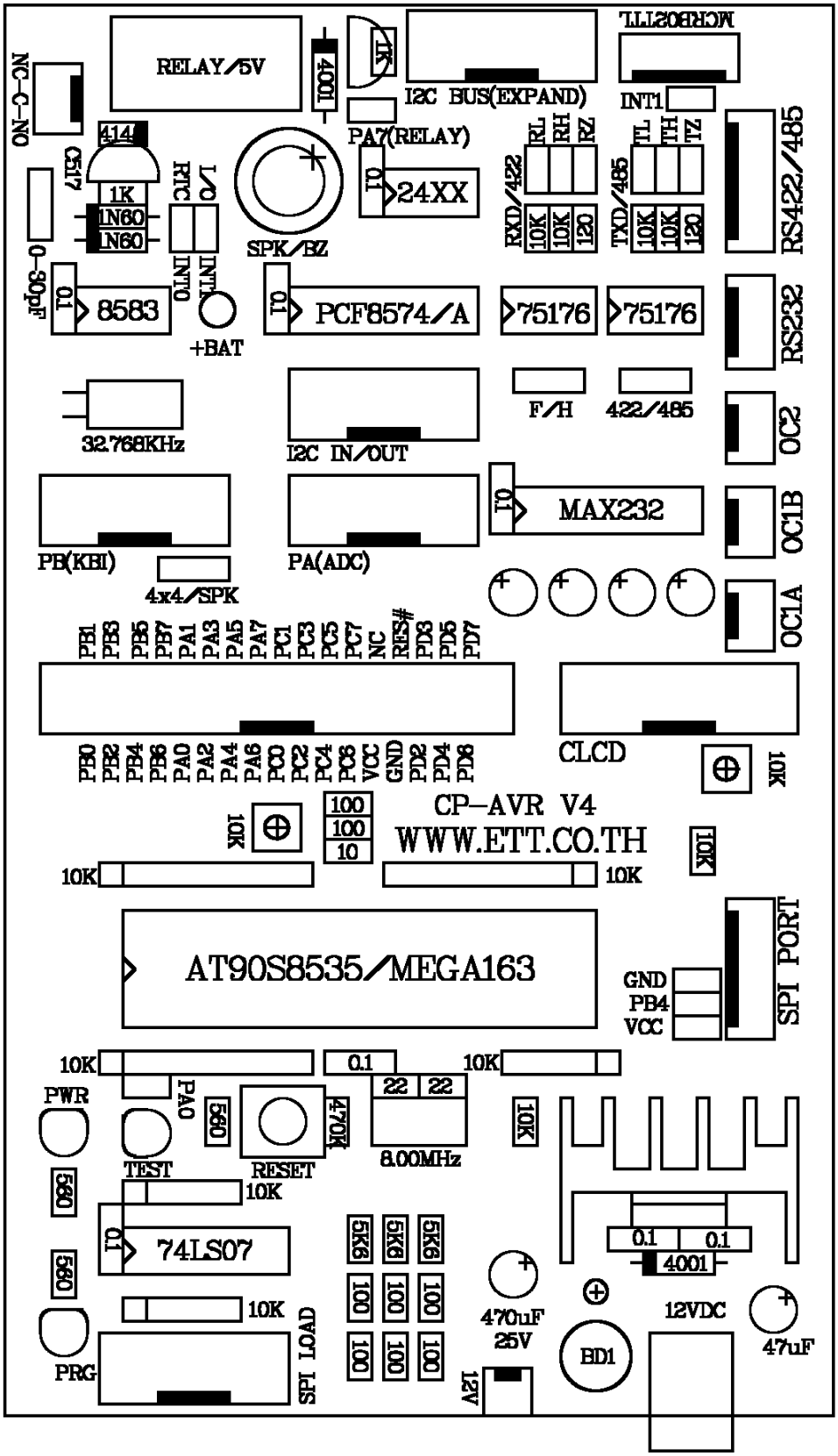
รูปแสดง บอร์ดไมโครคอนโทรลเลอร์รุ่น CP-AVR V3 EXP



รูปแสดง บอร์ดไมโครคอนโทรลเลอร์รุ่น CP-AVR V4



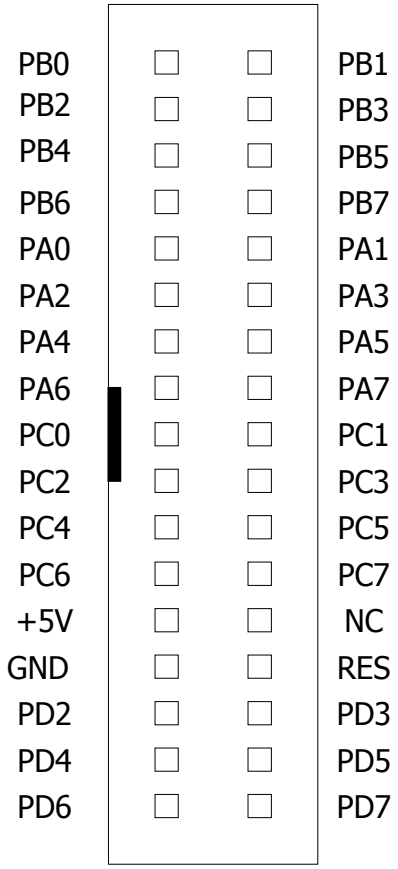
รูปแสดง ลักษณะของบอร์ด CP-AVR V3 / AVR V3 EXP



รูปแสดง ลักษณะของบอร์ด CP-AVR V4

การใช้งานพอร์ตของ CPU **เป็น** I/O Port (Input/Output Port)

ลักษณะการต่อขาพอร์ตของไมโครคอนโทรลเลอร์ (MCU) เข้ากับพอร์ต I/O ขนาด 34 ขา(72IOZ80)สามารถแสดงได้ดังรูปต่อไปนี้ ซึ่งบอร์ดทั้งสามรุ่นจะต่อในรูปแบบเดียวกัน



รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้ว 34PIN

โดยปกติพอร์ต I/O ของ AVR สามารถรับกระแส sink ได้ประมาณ 20 mA ดังนั้นเราจึงสามารถนำพอร์ตเหล่านี้ไปขับ LED ให้ทำงานได้ ในการใช้งานพอร์ตของ AVR เป็น Input/Output Port เราจะต้องทำการกำหนดทิศทางของพอร์ตว่าจะให้เป็นอินพุตหรือเป็นเอาพุตโดยการเซตหรือเคลียร์ค่าในรีจิสเตอร์ DDRx และถ้าเราต้องการกำหนดระดับของสัญญาณ(high/low)ในขณะที่ PORTx ทำงานในโหมด output เราก็สามารถทำได้โดยการไปset หรือ clear ค่าในรีจิสเตอร์ PORTx ส่วนในกรณีที่พอร์ตทำงานในโหมด input นั้นถ้าหากเรามีการต่อ R pull low ภายนอกไว้ และทำการ Set พอร์ตที่ต่อ R Pull Low ให้เป็น 1 (Enable Internal Pull Up) พอร์ตนั้นจะสามารถจ่ายกระแสออกมาให้กับR ที่ต่อ Pull Low ภายนอกได้ด้วยกระแสประมาณ 20 mA

ในการอ่านค่าของสัญญานจากภายนอกนั้น ถ้าเราอ่านจากรีจิสเตอร์ PORTx เราจะได้ค่าที่ถูก Latch ค้างไว้ ซึ่งเป็นค่าที่เกิดก่อนหน้าที่เราจะทำการอ่าน แต่ถ้าเราอ่านจากรีจิสเตอร์ PINx เราจะได้ค่าที่เป็นจริงที่เกิดขึ้นที่ขาพอร์ตขณะที่เราทำการอ่าน ตารางต่อไปนี้จะเป็นรูปแบบการกำหนดค่าเพื่อให้พอร์ตทำงานในโหมด I/O

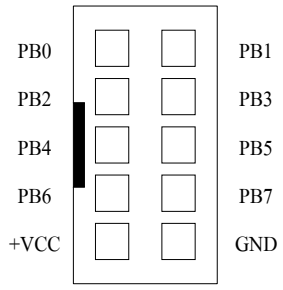
DDxn	PORTxn	I/O	Pull up	Comment
0	0	Input	No	Tri-state (Hi-Z)
0	1	Input	Yes	Pxn will source current if ext. pulled low
1	0	Output	No	Push-Pull Zero output
1	1	Output	No	Push-Pull One output

หมายเหตุ : x คือพอร์ต A – D, n คือบิต 0 – 7

ตารางแสดง การกำหนดค่าให้กับพอร์ตของไมโครคอนโทรลเลอร์ตระกูล AVR เพื่อเลือกโหมดการทำงาน

การใช้งานขั้วต่อ PB(KBI)

พอร์ต PB(KBI) ถูกจัดไว้ที่ขั้ว Connector ขนาด 10 PIN แบบ IDE ซึ่งขั้วต่อนี้จะมีอยู่เฉพาะ ในบอร์ดรุ่น CP-AVR V4.0 เท่านั้น โดยขั้วต่อนี้จะเชื่อมต่อสัญญาณมาจาก PORTB ของ MCU ทั้ง 8 เส้น โดย PB0-PB6 จะถูกนำมาจัดเรียงไว้โดยตรงอยู่แล้ว ส่วน PB7 จะต้องเลือกจาก Jumper (4x4/SPK) อีกครั้งหนึ่ง โดยลักษณะของขาสัญญาณที่จัดเรียงไว้ที่ขั้ว PB(KBI) จะเป็นดังรูป



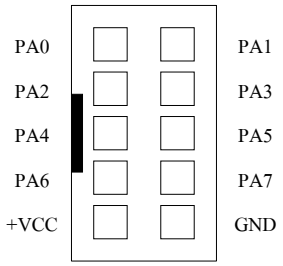
รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้ว PB(KBI)

โดยจุดประสงค์ในการออกแบบของบอร์ด CP-AVR V4.0 นั้น ขั้วต่อ PB(KBI) จะใช้สำหรับให้ผู้ใช้เชื่อมต่อกับวงจรคีย์บอร์ดแบบ Matrix ซึ่งสามารถจะใช้ได้กับคีย์บอร์ดแบบ Matrix ขนาด 4x3 หรือ 4x4 ก็ได้ ซึ่งในกรณีที่ใช้กับคีย์บอร์ดขนาด 4x3 จะเหลือสัญญาณไว้ 1 เส้นคือ PB7 ซึ่งสามารถนำไปใช้ควบคุมการกำเนิดเสียงของลำโพงขนาดเล็กหรือ BUZZER เพื่อกำเนิดเสียงได้

สำหรับในกรณีที่ไม่มีความจำเป็นต้องใช้งานคีย์บอร์ดแล้ว ขั้วต่อ PB(KBI) นี้ก็ยังสามารถนำไปต่อใช้งานเป็น Input หรือ Output ทั่วๆไปได้อีกด้วย โดยจากวงจรท่านจะพบว่าเราได้ต่อ LED เข้ากับขา PB0 ดังนั้นหากท่านไม่ใช้งานคีย์บอร์ดท่านสามารถใช้งาน LED TEST ตัวนี้ได้โดยการ Short Jumper PB0 สำหรับบอร์ดไมโครคอนโทรลเลอร์ CP-AVR V3.0/V3.0 EXP หรือ Short Jumper PA0 สำหรับบอร์ดไมโครคอนโทรลเลอร์ CP-AVR V4.0

การใช้งานขั้วต่อ PA(ADC)

พอร์ต PA(ADC) นี้จะถูกเชื่อมต่อออกมารอไว้ยังขั้ว Connector ขนาด 10 PIN แบบ IDE ซึ่งขั้วต่อนี้จะมีอยู่เฉพาะบอร์ดในรุ่น CP-AVR V4.0 เท่านั้น โดยขั้วต่อ PA(ADC) นี้ สามารถโปรแกรมหน้าที่การใช้งานได้หลายหน้าที่ เช่น โปรแกรมให้ทำหน้าที่เป็น ADC ขนาด 10 บิต 8 ช่อง เพื่อรับค่าสัญญาณ Analog ขนาด 0-5VDC จากภายนอกเข้ามาและเปลี่ยนเป็นค่าข้อมูลดิจิตอล (000H-3FFH) ให้กับ CPU นำไปประมวลผลตามต้องการ หรือถ้าไม่ต้องการใช้งานเป็น ADC พอร์ต PA(ADC) นี้ก็ยังสามารถโปรแกรมหน้าที่การทำงานสำหรับใช้งานเป็น Input / Output ทั่วไปได้อีกด้วย โดยลักษณะของขาสัญญาณที่จัดเรียงไว้ที่ขั้ว PA (ADC) จะเป็นดังรูป

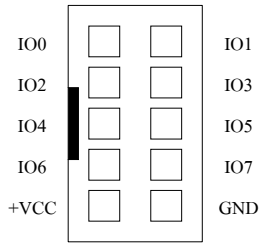


รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้ว PA(ADC)

นอกจากนี้แล้วขา PA7 ในกรณีที่ไม่ต้องการใช้งานเป็น ADC สามารถนำไปใช้งานเป็น Output สำหรับควบคุมการทำงานของ RELAY ได้อีกด้วย โดยการ Short Jumper PA7(RELAY) ไว้

การใช้งานขั้วต่อ I^2C IN/OUT

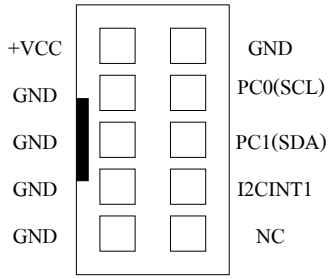
พอร์ต I^2C IN/OUT นี้จะถูกเชื่อมต่อออกมารอไว้ยังขั้ว Connector ขนาด 10 PIN แบบ IDE ซึ่งขั้วต่อนี้จะมีอยู่เฉพาะบอร์ดในรุ่น CP-AVR V4.0 เท่านั้น โดยขั้วต่อ I^2C IN/OUT นี้ จะเชื่อมต่อมาจากขาสัญญาณ I/O Port ของ PCF8574/A ซึ่งสามารถโปรแกรมหน้าที่การใช้งาน ให้เป็น Input หรือ Output ก็ได้ตามต้องการจากโปรแกรม 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

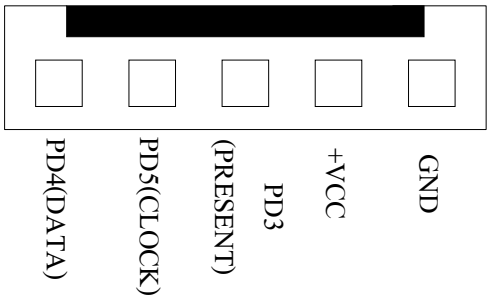
พอร์ต I^2C BUS EXPAND นี้จะถูกเชื่อมต่อออกมารอไว้ยังขั้ว Connector ขนาด 10 PIN แบบ IDE ซึ่งขั้วต่อนี้จะมีอยู่เฉพาะบอร์ดในรุ่น CP-AVR V4.0 เท่านั้น โดยขั้วต่อ I^2C BUS EXPAND นี้ จะใช้สำหรับทำการขยายหรือเพิ่มเติมจำนวนอุปกรณ์ที่ใช้การติดต่อสื่อสารแบบ I2C ให้กับบอร์ด



รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้ว I^2C BUS EXPAND

การใช้งานเครื่องอ่านบัตรแถบแม่เหล็ก (MAGNETIC-CARD READER)

สำหรับบอร์ด CP-AVR V4.0 นั้น จะออกแบบให้สามารถเชื่อมต่อกับเครื่องอ่านบัตรแถบแม่เหล็กรุ่น “MCR-B02TTL” ได้โดยตรง โดยไม่ต้องดัดแปลงวงจรใดๆทั้งสิ้น โดยในบอร์ดจะจัดเตรียมขั้วแบบ CPA ขนาด 5 PIN ไว้รองรับอยู่แล้ว ผู้ใช้สามารถนำขั้วต่อของเครื่องอ่านบัตรแถบแม่เหล็ก รุ่น “MCR-B02TTL” ของบริษัท อีทีที ต่อเข้าไปได้ทันที สำหรับในการเขียนโปรแกรมเพื่อเชื่อมต่อระหว่างบอร์ด CP-AVR V4.0 กับเครื่องอ่านบัตรแถบแม่เหล็กนั้น จะสามารถทำได้ 2 แบบ คือ ใช้วิธีการวนรอบตรวจสอบสัญญาณจากเครื่องอ่านบัตรแถบแม่เหล็กเอง ซึ่งวิธีการนี้จะใช้สัญญาณจาก CPU เพียง 2 เส้นสัญญาณคือ PD4 ทำหน้าที่เป็น DATA และ PD5 ทำหน้าที่เป็น CLOCK โดยต้องกำหนดคุณสมบัติของสัญญาณทั้ง 2 เส้นให้มีทิศทางเป็น Input และควร ENABLE การ PULL-UP ของสัญญาณทั้ง 2 นี้ด้วยเสมอ ส่วนอีกวิธีหนึ่งคือการใช้วิธีการ Interrupt ซึ่งสามารถทำได้โดยการ SHORT JUMPER “INT1” ที่อยู่ใกล้ๆกับขั้วต่อเครื่องอ่านบัตรแถบแม่เหล็ก ซึ่งการ SHORT JUMPER จะเป็นการเชื่อมต่อสัญญาณ INT1 ของ CPU เข้ากับ สัญญาณ PRESENT ของเครื่องอ่านบัตรแถบแม่เหล็ก ดังนั้นเมื่อมีการนำบัตรแถบแม่เหล็กไปรูดผ่านเครื่องอ่านบัตรแถบแม่เหล็กก็จะมีการส่งสัญญาณ Interrupt ออกมายัง CPU ซึ่งผู้ใช้ก็เพียงแต่เขียนโปรแกรมสำหรับบริการการ Interrupt ของ INT1 ไว้ก็สามารถใช้งานได้แล้ว



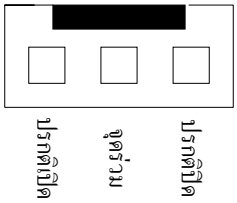
รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้วต่อ
เครื่องอ่านบัตรแถบแม่เหล็ก MCR-B02TTL

<u>**หมายเหตุ**</u> เนื่องจากการเลือกใช้ Interrupt จากสัญญาณ INT1 ของ CPU นั้น ภายในบอร์ด รุ่น CP-AVR V4.0 สัญญาณ INT1 จะถูกออกแบบให้สามารถเลือกกำหนดใช้งานร่วมกับอุปกรณ์หลายๆตัว เช่น การ Interrupt จากไอซีขยาย PORT I/O เบอร์ PCF8574/A และการ Interrupt จาก เครื่องอ่านบัตรแถบแม่เหล็ก MCR-B02TTL ดังนั้นในกรณีที่ต้องการเลือกใช้วิธีการ Interrupt จาก INT1 นั้น จะต้องทำการ OPEN JUMPER สำหรับเลือกการ Interrupt จากอุปกรณ์อื่นๆที่ไม่เกี่ยวข้องออกเสียก่อน ให้เหลือการเชื่อมต่อสัญญาณ INT1 จาก CPU กับอุปกรณ์เพียงตัวใดตัวหนึ่งเท่านั้น เพื่อให้แน่ใจว่าสัญญาณ Interrupt ที่เกิดขึ้นจะถูกส่งมาออกมาจากเครื่องอ่านบัตรแถบแม่เหล็กเท่านั้น ไม่เช่นนั้นแล้วอาจเกิดความผิดพลาดขึ้นได้

การใช้งาน OUTPUT RELAY

ภายในบอร์ด CP-AVR V4.0 นั้น จะออกแบบวงจรควบคุม RELAY ไว้ให้ผู้ใช้งานสามารถนำไปประยุกต์ใช้งานทั่วไปได้ด้วย จำนวน 1 ชุด โดยวงจร RELAY ดังกล่าวสามารถใช้งานได้ ทั้งหน้าสัมผัสแบบปรกติเปิด (Normal Open : NO) และแบบหน้าสัมผัสปรกติปิด (Normal Close : NC)

สำหรับสัญญาณ Output ในการควบคุมการทำงานของ RELAY นั้น จะแบ่งมาจาก PA7 ของ CPU ซึ่งเมื่อต้องการใช้งาน RELAY จะต้องทำการ SHORT JUMPER PA7(RELAY) ไว้ด้วยเพื่อเชื่อมต่อสัญญาณ PA7 มาทำการควบคุมการทำงานของ RELAY และต้องแน่ใจว่าไม่ได้ต่อสัญญาณ PA7 จากขั้วต่ออื่นๆออกไปใช้งานกับอุปกรณ์ใดๆนอกเหนือจาก RELAY เนื่องจากสัญญาณ PA7 นั้น นอกจากจะต่อมาใช้ควบคุมการทำงานของ RELAY แล้วยังต่อไปยังขั้วต่อ 34PIN และขั้วต่อ PA(ADC) ด้วย โดยการทำงานของ RELAY จะถูกควบคุมการทำงานจากขาสัญญาณ PA7 ผู้ใช้ต้องกำหนดคุณสมบัติของสัญญาณ PA7 ให้ทำหน้าที่เป็น OUTPUT ซึ่งเมื่อขาสัญญาณ PA7 มีสภาวะเป็น OUTPUT และให้สถานะเป็น “1” จะทำให้ RELAY ทำงาน แต่ถ้าสถานะของสัญญาณ PA7 มีค่าเป็น “0” จะทำให้ RELAY หยุดทำงาน



รูปแสดง ลักษณะขั้วต่อสัญญาณจากหน้าสัมผัสของ RELAY

หมายเหตุ เนื่องจากสัญญาณ PA7 ที่นำมาใช้ควบคุมการทำงานของ RELAY จะเป็นสัญญาณเส้นเดียวกับ PA7 ที่ต่อไว้ยังขั้ว 34PIN และขั้วต่อ 10PIN แบบ IDE ของพอร์ต PA(ADC) ดังนั้น เมื่อต้องการใช้งาน RELAY โดยการควบคุมจาก PA7 แล้ว ต้องแน่ใจว่าไม่ได้ต่อใช้งานสัญญาณ PA7 ในจุดอื่นๆด้วย แต่ถ้าหากมีความจำเป็นต้องใช้งาน PA7 พร้อมกับการใช้งาน RELAY ด้วยในเวลาเดียวกัน ก็อาจดัดแปลงวงจรได้โดยการ OPEN JUMPER PA7(RELAY) ออก แล้วใช้วิธีการเชื่อมสายสัญญาณเส้นอื่นๆจาก PORT I/Oของ CPU ที่ไม่ได้ใช้งานมาเข้ากับวงจรควบคุม RELAY แทนก็ได้เช่นเดียวกัน โดยให้เชื่อมต่อสายสัญญาณที่ต้องการไปยัง Jumper PA7(RELAY) ด้านที่ต่อกับตัวต้านทานค่า 1KOhm แต่การดัดแปลงวิธีนี้ควรถอดตัว JUMPER

PA7(RELAY) ออกจากบอร์ดเสียก่อน เพื่อจะได้ไม่หลงลืมทำการ SHORT JUMPER นี้ซ้ำอีกในภายหลัง เนื่องจากจะเป็นการ SHORT สัญญาณ PA7 เข้ากับสัญญาณเส้นใหม่ที่บัดกรีมายังวงจร RELAY นี้อีก

การใช้งานลำโพงขนาดเล็ก หรือ BUZZER

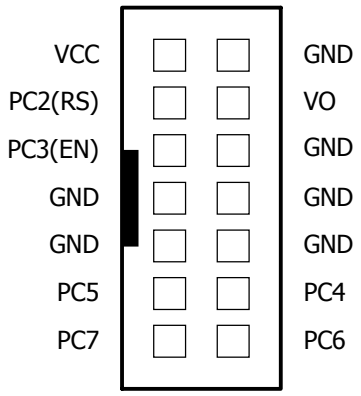
ภายในบอร์ด CP-AVR V4.0 จะมีวงจรกำเนิดเสียงรวมอยู่ด้วย 1 จุด ซึ่งในตำแหน่งนี้สามารถเลือกใส่อุปกรณ์กำเนิดเสียงแบบลำโพงขนาดเล็ก หรือ จะเลือกใส่ BUZZER แทนก็ได้เช่นเดียวกัน โดยในกรณีที่เลือกใช้ลำโพงจะมีข้อดีคือ สามารถสร้างความถี่เสียงได้หลากหลายความถี่ตามต้องการแต่การเขียนโปรแกรมจะยุ่งยากกว่า BUZZER เนื่องจากต้องสร้างเป็นสัญญาณความถี่จึงจะสามารถทำให้ลำโพงกำเนิดเสียงให้ได้ ส่วนในกรณีที่เลือกใช้ BUZZER นั้น จะมีข้อดีคือ เขียนโปรแกรมควบคุมการกำเนิดเสียงได้ง่ายกว่าลำโพง เนื่องจากใช้วิธีการ ON หรือ OFF เท่านั้น โดยการ ON บิตควบคุม BUZZER ให้มีสภาวะเป็น “1” เท่านั้น BUZZER ก็จะกำเนิดเสียงให้แล้ว แต่ความถี่เสียงของ BUZZER จะไม่สามรถเลือกได้ เหมือนกับลำโพง

สำหรับสัญญาณ Output ในการควบคุมการทำงานของ ลำโพง หรือ BUZZER นั้น จะแบ่งมาจาก PB7 ของ CPU ซึ่งเมื่อต้องการใช้งาน ลำโพง หรือ BUZZER จะต้องทำการเลือก JUMPER 4x4/SPK มารอไว้ยังตำแหน่ง SPK ด้วย เพื่อให้สามารถนำสัญญาณ PB7 มาทำการควบคุมการทำงานของลำโพง หรือ BUZZER ได้ โดยผู้ใช้ต้องทำการกำหนดคุณสมบัติของสัญญาณ PB7 ให้ทำหน้าที่เป็น OUTPUT ไว้ด้วย ซึ่งเมื่อขาสัญญาณ PB7 มีสภาวะเป็น OUTPUT และให้สถานะเป็น “1” จะทำให้ ลำโพง หรือ BUZZER ทำงาน แต่ถ้าสถานะของสัญญาณ PB7 มีค่าเป็น “0” จะทำให้ ลำโพง หรือ BUZZER หยุดทำงาน

*****หมายเหตุ***** เนื่องจากสัญญาณ PB7 ที่นำมาใช้ควบคุมการทำงานของ ลำโพง หรือ BUZZER นั้น จะเป็นสัญญาณเส้นเดียวกับ PB7 ที่ต่อไว้ยังขั้ว 34PIN และขั้วต่อ 10PIN แบบ IDE ของพอร์ต PB(KBI) ด้วย ดังนั้นเมื่อต้องการใช้งานลำโพง หรือ BUZZER โดยการควบคุมจาก PB7 แล้ว ต้องแน่ใจว่าไม่ได้ต่อใช้งานสัญญาณ PB7 ในจุดอื่นๆทั้งสอง ดังกล่าวด้วย แต่ถ้าหากมีความจำเป็นต้องใช้งาน PB7 พร้อมกับการใช้งานลำโพง หรือ BUZZER ด้วยในเวลาเดียวกัน ก็อาจดัดแปลงวงจรได้ โดยการใช้วิธีการบัดกรีเชื่อมสายสัญญาณเส้นอื่นๆจาก PORT I/Oของ CPU ที่ไม่ได้ใช้งานมาเข้ากับวงจรควบคุมลำโพงหรือ BUZZER แทนก็ได้เช่นเดียวกัน โดยให้เชื่อมต่อสายสัญญาณที่ต้องการไปยัง Jumper 4x4/SPK ด้าน SPK แต่การดัดแปลงวิธีนี้ควรถอดตัว JUMPER 4x4/SPK ออกจากบอร์ดเสียก่อนแล้วทำการเชื่อมต่อขาสัญญาณ Jumper 4x4/SPK ด้าน ตำแหน่ง 4x4 ไว้ด้วยกันเลย เพื่อจะได้ไม่หลงลืมทำการเลือก JUMPER กลับมายังด้าน SPK นี้อีกในภายหลัง เนื่องจากจะเป็นการ SHORT สัญญาณ PB7 เข้ากับสัญญาณเส้นใหม่ที่บัดกรีมายังวงจรควบคุมลำโพงหรือ BUZZER นี้อีก

การใช้งานจอแสดงผลแบบ LCD (Dot-Matrix Character LCD)

บอร์ด CP-AVR V4.0 สามารถใช้เชื่อมต่อกับจอแสดงผล LCD แบบ Dot-Matrix โดยเชื่อมต่อผ่านทาง Connector ขนาด 14 PIN และใช้ตัวต้านทานปรับค่าได้แบบเกือกม้าขนาด 10K สำหรับปรับระดับความสว่างของหน้าจอ LCD โดยวงจรในการเชื่อมต่อ LCD ของบอร์ดนี้จะออกแบบวงจรให้ใช้วิธีการควบคุมการทำงานแบบ "DATA 4-BIT"

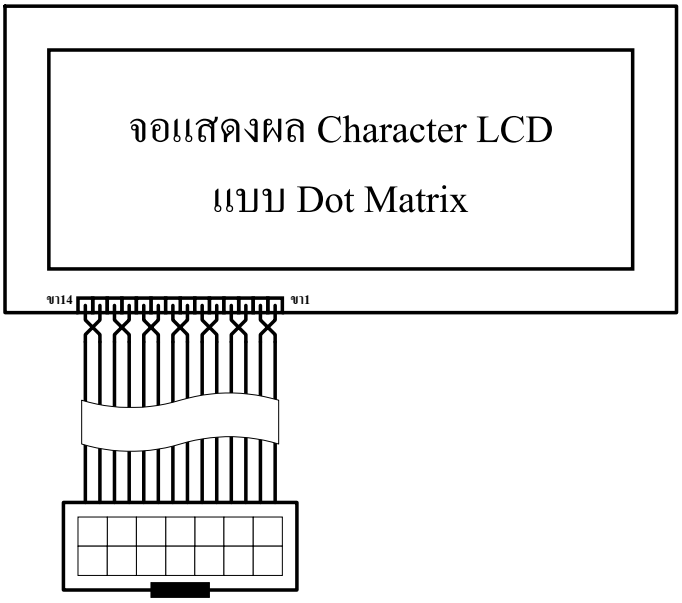


รูปแสดง ขาสัญญาณของขั้วต่อ CLCD

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

- แบบที่เป็นขั้วต่อแบบแถวเดี่ยว ขนาด 14PIN (HEADER 14X1) โดยการต่อสายของ LCD แบบนี้ จะใช้สายแพร์ขนาด 14 เส้น แบบเข้าหัว CONNECTOR ไว้ด้านเดียว สำหรับเสียบกับขั้วต่อ CLCD ภายในบอร์ด CP-AVR V4.0 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 ของขั้วต่อแต่ละด้านอยู่ในตำแหน่งที่ตรงกัน ก็สามารถใช้งานได้แล้ว

*****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



รูปแสดง วิธีการต่อสาย LCD แบบใช้ขั้วแถวเดี่ยว

การเชื่อมต่อกับอุปกรณ์ I^2C BUS

สำหรับอุปกรณ์แบบ I^2C Bus ที่ใช้ในบอร์ด CP-AVR V4.0 นั้น จะออกแบบให้สามารถติดตั้งใช้งานอุปกรณ์ I^2C ได้พร้อมกันในบอร์ดทั้งหมดด้วยกัน 3 ตัว คือ

- I^2C RTC เบอร์ PCF8583 ของ PHILIPS
- EEPROM ในตระกูล 24XX ซึ่งสามารถเลือกใช้ได้หลายเบอร์หลายผู้ผลิต ขึ้นอยู่กับขนาดความจุของหน่วยความจำที่ต้องการจะใช้ ซึ่งในบอร์ด CP-AVR V4.0 นั้น สามารถติดตั้งใช้งานหน่วยความจำ EEPROM แบบ I^2C นี้ได้ตั้งแต่ เบอร์ 24XX32,24XX64,24XX128,24XX256 หรือ 24XX512 เป็นต้น
- I^2C I/O เบอร์ PCF8574 หรือ PCF8574A ของ Phillips

โดยอุปกรณ์ I^2C ทั้ง 3 ตัวนี้จะต่อร่วมกันอยู่ภายในบัสเดียวกัน และใช้สัญญาณ PC1 เป็นขาสัญญาณ SDA และใช้สัญญาณ PC0 เป็นสัญญาณ SCL ในการควบคุมบัส ซึ่ง CPU จะทำหน้าที่เป็นตัวแม่ในการควบคุมบัส นอกจากนี้แล้วยังสามารถขยายอุปกรณ์จำพวก I^2C นี้ได้อีก แต่ต้องเป็นอุปกรณ์ที่มีรหัสควบคุม Control Word ไม่ซ้ำกันกับอุปกรณ์ที่มีอยู่แล้วภายในบอร์ดด้วย โดยอาจเชื่อมต่อผ่านทางขั้วต่อ “I^2C EXPANSION” ที่บอร์ดจัดเตรียมไว้ให้แล้วก็ได้

สำหรับในการใช้งานอุปกรณ์ I^2C นั้น เนื่องจาก CPU เบอร์ AT90S8535 ไม่มีส่วนของฮาร์ดแวร์ที่ทำหน้าที่ติดต่อสื่อสารแบบ I^2C บรรจุไว้ในตัว CPU ด้วย ดังนั้นจึงต้องใช้วิธีการนำ I/O Port ของ CPU มาใช้ติดต่อกับอุปกรณ์แบบ I^2C แทน โดยกำหนดให้

- **SDA** ต่อกับขา **PC1** ทำหน้าที่เป็นสัญญาณข้อมูลแบบ 2 ทิศทาง ใช้สำหรับรับส่งข้อมูลจาก CPU กับอุปกรณ์ I^2C โดยเมื่อ CPU ต้องการเขียนข้อมูลไปยังอุปกรณ์ ผู้ใช้จะต้องกำหนดให้สัญญาณนี้ทำหน้าที่เป็น Output แต่เมื่อ CPU ต้องการอ่านหรือรับข้อมูลจากอุปกรณ์ ผู้ใช้ก็ต้องกำหนดให้

สัญญาณนี้ทำหน้าที่เป็น Input แทนและต้องกำหนดการ Pull-Up ให้กับสัญญาณที่ทำหน้าที่เป็น SDA นี้ด้วยเสมอ

- **SCL** ต่อกับขา **PC0** ทำหน้าที่เป็นสัญญาณนาฬิกา Clock สำหรับใช้ติดต่อสื่อสารกับอุปกรณ์ I^2C สัญญาณ SCL นี้จะเป็นสัญญาณ Output จาก CPU เพื่อใช้ควบคุมการรับส่งข้อมูลระหว่าง CPU กับอุปกรณ์

สำหรับการใช้งานฟังก์ชั่น I^2C ใน CPU เบอร์ ATmega163 นั้นเราสามารถเลือกวิธีการติดต่อสื่อสารข้อมูลแบบการกำหนดให้พอร์ตทำงานเป็น I/O ดังที่ได้อธิบายมาแล้วในข้างต้นหรือจะเลือกใช้ฟังก์ชั่น 2-wire Serial Interface ที่มีอยู่ในตัวไอซีเบอร์นี้ก็ได้ ซึ่งฟังก์ชั่นนี้จะเป็นฟังก์ชั่นที่ใช้สำหรับสื่อสารข้อมูลในมาตรฐาน I^2C โดยเฉพาะ ทำให้เราสามารถเขียนโปรแกรมได้ง่ายและมีประสิทธิภาพมากขึ้น

การใช้งาน Interrupt ของอุปกรณ์ I^2C

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 ถ้าผู้ใช้เลือก Short Jumper I/O INT1 จะเป็นการต่อขา INT(ขา 13) ของ PCF8574/A เข้ากับขา INT1 ของ CPU และในขณะเดียวกันสัญญาณ INT1 นี้ก็จะถูกต่อผ่านไปยังขั้ว I^2C BUS EXPAND ด้วย และถ้าผู้ใช้เลือก Short Jumper RTC INT0 จะเป็นการต่อขา INT(ขา 7) ของ PCF8583 เข้ากับขา INT0 ของ CPU อย่างไรก็ตามทั้งสัญญาณ INT0 และ INT1 ของ CPU นี้จะถูกต่อโดยตรงเข้ากับขั้ว 34 Pin

*****หมายเหตุ***** ไม่ควรทำการ Short Jumper ของ INT0 หรือ INT1 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 EEPROM เบอร์ 24LC256 ก็สามารถต่อร่วมกันภายในบัสเดียวกันได้มากถึง 8 ตัว จากตัวอย่างข้างต้นจะเห็นได้ว่าภายในบัสเดียวกันของ I^2C นั้น อุปกรณ์เพียง 2 ประเภท คือ I/O และ EEPROM 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 20mA เท่านั้น ซึ่งคงไม่สามารถต่ออุปกรณ์ร่วมกันในบัสได้โดยไม่จำกัดจำนวนเหมือนในทฤษฎีบอกไว้ ซึ่งในความเป็นจริงอาจต้องพิจารณาตามความเหมาะสมและความจำเป็นในการใช้งานจริงๆด้วยว่าในระบบบัสหนึ่งๆของ I^2C นั้นควรต่ออุปกรณ์ในบัสจำนวนเท่าใด

หน้าที่และเบอร์ของอุปกรณ์ I^2C	รหัสตำแหน่งมาตรฐาน **ในการอ่าน/เขียน**	**รหัสตำแหน่งของบอร์ด** CP-AVR V4.0	
		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-AVR V4.0

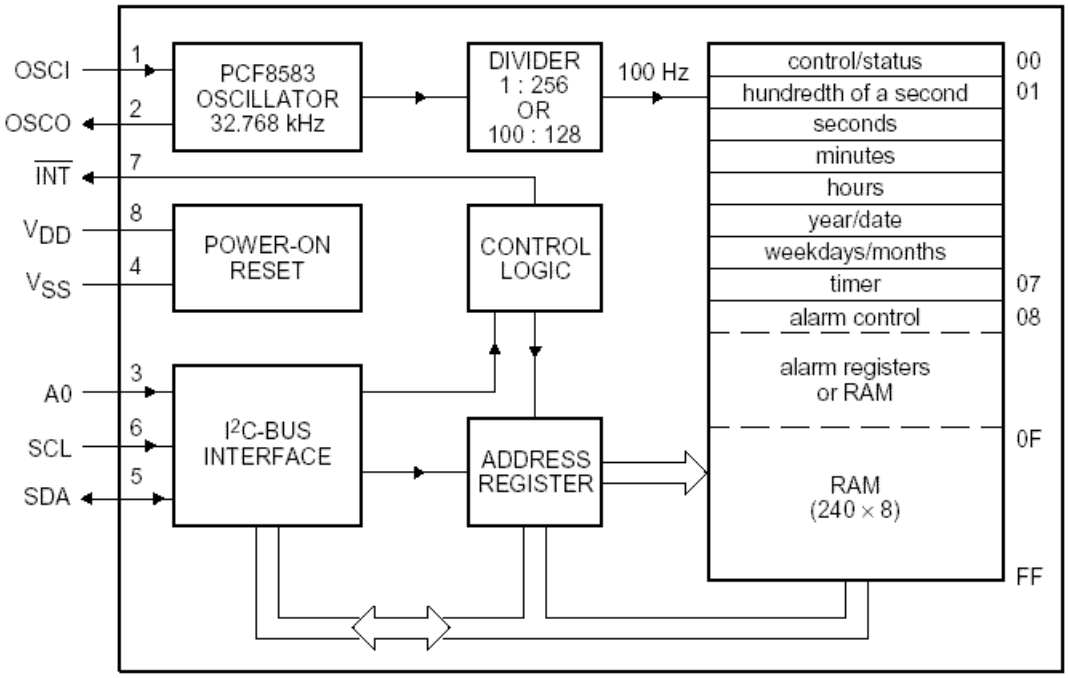
*****หมายเหตุ***** 1. ค่า X หมายถึงค่าโลจิกของขาสัญญาณ Address ของอุปกรณ์ ที่กำหนดในวงจร

2. ค่า ? หมายถึงบิตสำหรับกำหนดว่าต้องการเขียน หรือ อ่าน ข้อมูลกับอุปกรณ์

3. เนื่องจาก RTC เบอร์ PCF8583 และ EEPROM เบอร์ในกลุ่ม 24XX นั้นมีรหัสตำแหน่ง 4 บิตแรก ซ้ำกัน หรือ อยู่ในกลุ่มเดียวกัน ซึ่งในบอร์ด CP-AVR V4.0 นั้นออกแบบให้ RTC เบอร์ PCF8583 มีรหัสตำแหน่งของ Control Byte คงที่เป็น 1010001X ไว้ ส่วน EEPROM ก็กำหนดรหัสตำแหน่ง Control Byte ไว้ที่ 1010100X ดังนั้นถ้าต้องการเพิ่มเติมอุปกรณ์ใดๆเข้าไปอีกต้องกำหนดให้ค่า Control Byte ของอุปกรณ์ที่จะต่อเพิ่มเข้าไปไม่ซ้ำกับค่า Control Byte ทั้งสองดังกล่าวนี้ด้วย

การใช้งาน I^2C RTC เบอร์ PCF8583

สำหรับวงจรฐานเวลา RTC นั้น ในบอร์ด CP-AVR V4.0 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



รูปแสดง โครงสร้างภายในของ 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-AVR V4.0 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 ก็ตาม 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

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-AVR V4.0 นั้น จะกำหนดให้ขาสัญญาณ A0 ของ PCF8583 มีสภาวะทางโลจิกเป็น “1” คงที่ไว้เลย ดังนั้น RTC เบอร์ PCF8583 ในบอร์ด CP-AVR V4.0 นั้นจึงมีรหัส Control Byte คงที่เป็น “1010001X” เสมอ

*****หมายเหตุ***** ค่า X หรือ บิต0(R/W) 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 EEPROM (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 EEPROMในตระกูล 24XX (I^2C Bus) ในกลุ่มที่สามารถกำหนดรหัส Control Byte ของหน่วยความจำจากฮาร์ดแวร์ (ขาสัญญาณ A2,A1 และ A0) ได้ เช่น เบอร์ 24XX32,64,128 และ 24XX256 ของ Microchip เป็นต้น

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 ซึ่งจากคุณสมบัติดังกล่าวจะทำให้

สามารถทำการต่อหน่วยความจำดังกล่าวได้มากถึง 8 ตัวภายในระบบบัสเดียวกัน โดยกำหนดสภาวะของขาสัญญาณลอจิกแอดเดรสที่แตกต่างกันออกไป โดยสามารถสรุปให้เห็นได้ดังตารางต่อไปนี้

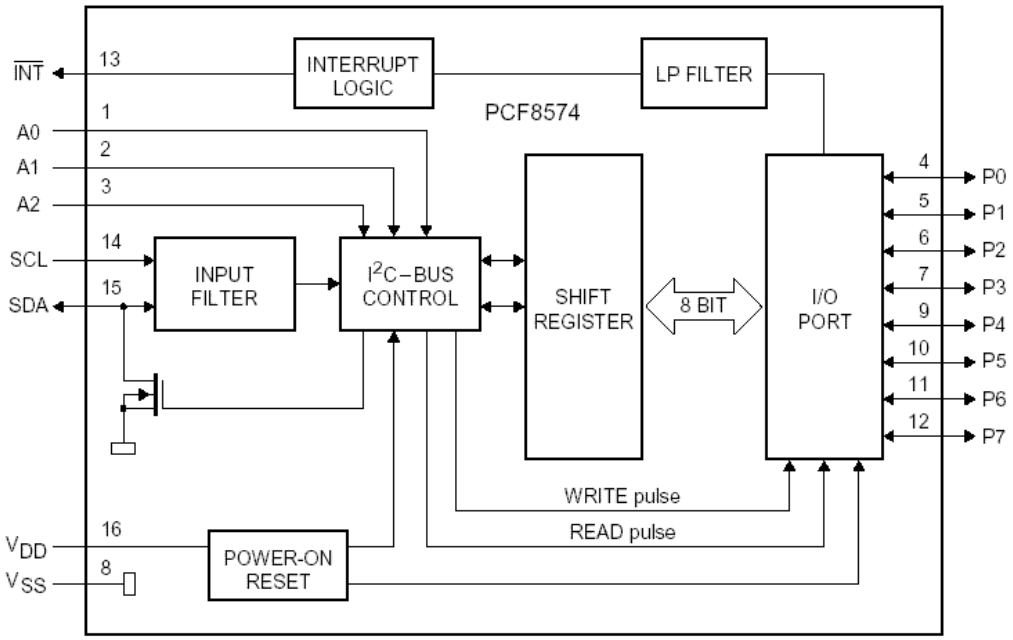
เบอร์(ความจุ)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
24XX32 (4Kx8)	1	0	1	0	A2	A1	A0	R/W
24XX64 (8Kx8)	1	0	1	0	A2	A1	A0	R/W
24XX128 (16Kx8)	1	0	1	0	A2	A1	A0	R/W
24XX256 (32Kx8)	1	0	1	0	A2	A1	A0	R/W

ตารางแสดง รหัส Control Byte **ของหน่วยความจำแบบ** I^2C Bus **ของ** Microchip

จากตารางจะเห็นได้ว่า หน่วยความจำ EEPROM แบบ I^2C-BUS นั้น 24XX32/64/128/256 ของ Microchip นั้นจะมีรหัส Control Code ที่เหมือนกันทุกเบอร์ แต่จะมีความแตกต่างกันที่ขนาดของหน่วยความจำ ดังนั้นเมื่อทำการติดตั้งใช้งานหน่วยความจำเบอร์เหล่านี้กับบอร์ด CP-AVR V4.0 แล้วจะมีรหัส Control Byte เป็น "1010100X" คงที่ตลอด แต่ถ้ามีการต่อหน่วยความจำเหล่านี้เพิ่มเติมจากภายนอกบอร์ดแล้วรหัส Control Byte ก็จะขึ้นอยู่กับการกำหนดสภาวะทางลอจิกให้กับขาสัญญาณ A2,A1 และ A0 ของหน่วยความจำที่ต่อไว้

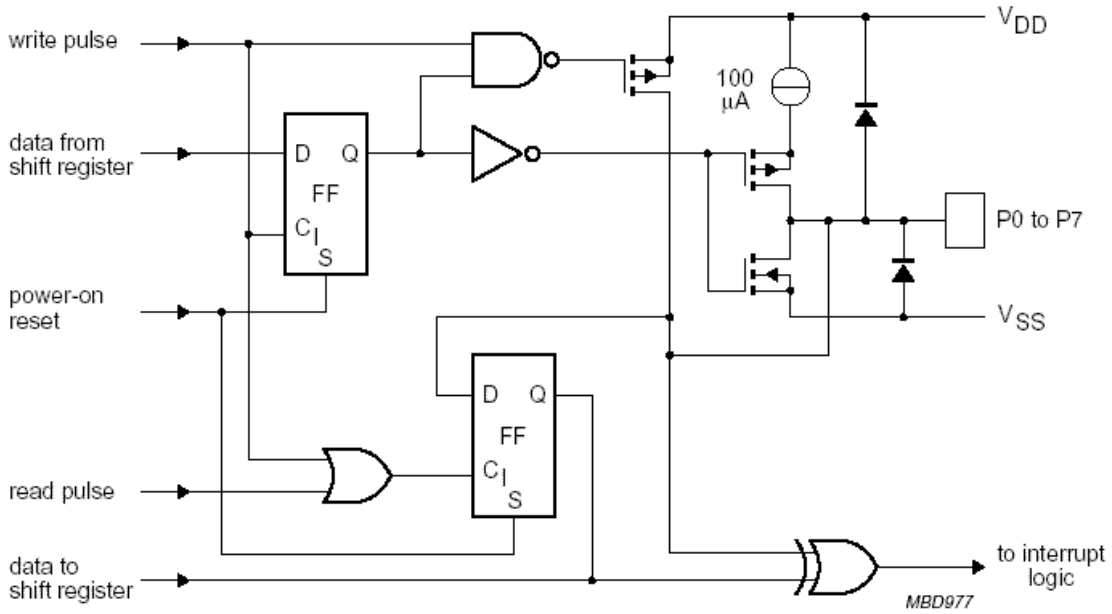
การใช้งาน I/O Port แบบ I^2C (PCF8574/A)

หากในการใช้งานมีความจำเป็นต้องใช้งานพอร์ต I/O จำนวนมาก และจำนวนพอร์ต I/O ของ CPU ที่มีอยู่ไม่เพียงพอต่อการใช้งานแล้ว ผู้ใช้สามารถทำการเพิ่มเติมพอร์ต I/O ได้อีก โดยในบอร์ด CP-AVR V4.0 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

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

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

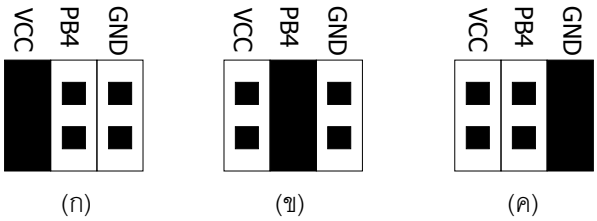
สำหรับรหัส Control Byte ของพอร์ต I/O เบอร์ PCF8574 หรือ PCF8574A ของบอร์ดนั้น จะถูกกำหนดไว้ตายตัว โดยขาสัญญาณแอดเดรส A0,A1 และ A2 จะถูกกำหนดสภาวะโลจิกเป็น “0” ไว้ทั้งหมด ซึ่งในกรณีที่ผู้ใช้ทำการติดตั้งพอร์ต I/O เบอร์ PCF8574 จะมีรหัส Control Byte เป็น “0100000X” แต่สำหรับกรณีที่ผู้ใช้ทำการติดตั้งพอร์ต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

การใช้งานพอร์ตสื่อสารอนุกรม SPI/RS232/RS422/RS485

พอร์ตที่ใช้ในการสื่อสารข้อมูลแบบอนุกรมใน CPU เบอร์ 90S8535 จะมีอยู่ 2 พอร์ตด้วยกันนั่นคือ SPI Port กับ Serial Port (RS232) ส่วน ATmega163 นั้น จะมีพอร์ตที่ใช้ในการสื่อสารข้อมูลอนุกรมอยู่ 3 พอร์ตด้วยกันได้แก่พอร์ต SPI พอร์ตอนุกรม(RS232) และพอร์ต 2-wire Serial Interface ซึ่งพอร์ตนี้จะต่อออกมาใช้งานภายนอกผ่านทางขั้ว I2C EXPAND และ ขั้วต่อ I/O 34 Pin สำหรับบอร์ด CP-AVR V4.0 เราได้ต่อวงจร RS422 และ RS485 เพิ่มเติม โดยใช้ไอซีเบอร์ 75176 ช่วยปรับระดับของสัญญาณ ต่อไปนี้เราจะกล่าวถึงพอร์ตสื่อสารอนุกรมแบบต่างๆ ที่อยู่บนบอร์ดทั้ง 3 รุ่น

การสื่อสารอนุกรม SPI

สำหรับพอร์ต SPI (Serial Peripheral Interface Port) นั้นปรกติเราจะใช้ในการดาวน์โหลดโปรแกรม แต่เราสามารถนำมันมาใช้ในการรับส่งข้อมูลแบบอนุกรมได้เช่นเดียวกันกับพอร์ตอนุกรมอื่นๆ โดยในการใช้งานนั้นเราจะต้องมี CPU ที่ทำหน้าที่เป็นตัวมาสเตอร์และ CPU ที่ทำหน้าที่เป็นตัวสเลฟ นั่นแสดงว่าเราจะต้องมี CPU อย่างน้อย 2 ตัวขึ้นไป และเราสามารถกำหนดให้ CPU ตัวที่จะมาต่อกับพอร์ตนี้เป็นตัวมาสเตอร์หรือสเลฟก็ได้โดยการ Short Jumper VCC/PB4/GND ดังรูปต่อไปนี้



(ก) ทำงานในโหมด Master

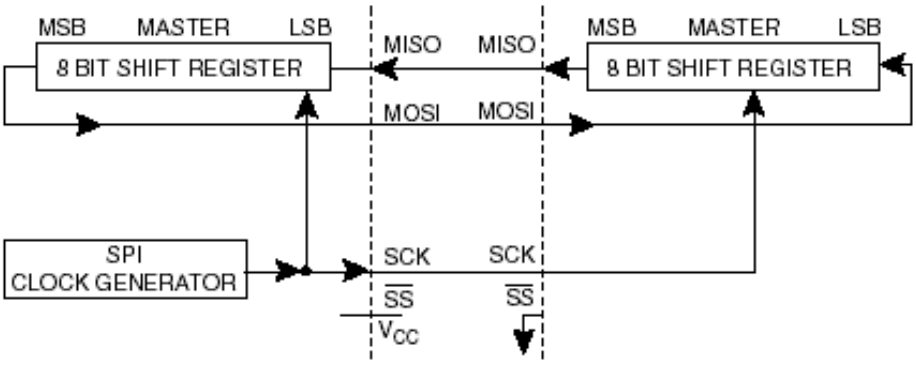
(ข) กำหนดโหมดการทำงานได้จากโปรแกรม

(ค) ทำงานในโหมด Slave

รูปแสดง การกำหนด jumper ให้กับพอร์ต SPI เพื่อเลือกโหมดการทำงาน

จากรูปข้างบนอธิบายได้ว่า หากเราต้องการให้ CPU ที่มาต่อกับขั้วนี้ทำงานในโหมด Master ก็ให้ทำการเลือก Jumper ดังรูป (ก) และถ้าเราเลือก Jumper ดังรูป (ข) จะทำให้เราสามารถเขียนโปรแกรมควบคุม CPU ได้ว่าเราต้องการให้มันทำงานอยู่ในโหมดSlave หรือ Master แต่ถ้าเราเสียบ jumper ดังรูป (ค) จะเป็นการกำหนดให้ CPU ทำงานในโหมด Slave

รูปต่อไปนี้จะเป็นการอธิบายถึงการต่อ CPU 2 ตัวเข้าด้วยกันโดยการสื่อสารข้อมูลจะทำผ่านพอร์ต SPI ซึ่งจากรูปท่านจะพบว่าขา MOSI (Master Out Slave In) ของตัวมาสเตอร์ต่ออยู่กับขา MOSI ของอุปกรณ์สเลฟ และขา MISO (Master In Slave Out) ของตัวมาสเตอร์จะต่ออยู่กับ MISO ของตัว Slave ส่วนขา SCK ของ CPU ทั้งสองตัว จะต่อเข้ากับ clock ตัวเดียวกันเพื่อให้เกิดการซิงโครนัสในการส่งข้อมูล ขา $\overline{SS}$ ของตัวมาสเตอร์จะต่ออยู่กับ VCC ส่วนขา $\overline{SS}$ ของตัวสเลฟจะต่ออยู่กับ GND 8 Bit Shift Register เป็นรีจิสเตอร์ที่อยู่ใน CPU ทำหน้าที่ในการเก็บข้อมูลและเลื่อนข้อมูลเข้าออก ดังนั้นเมื่อเราต่อ CPU ดังรูป การทำงานของมันจะเป็นเหมือนการนำรีจิสเตอร์ 8 บิต 2 ตัวมาต่อกันเป็นรีจิสเตอร์ขนาด 16 บิต 1 ตัว นั่นเอง



รูปแสดง การต่อ CPU 2 ตัวเข้าด้วยกันผ่าน SPI Port

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

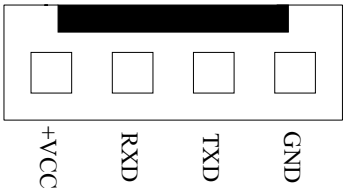
สำหรับสายสัญญาณที่จะนำมาใช้สำหรับทำการสื่อสารแบบ RS232 นั้น จะใช้สัญญาณเพียง 2-3 เส้นเท่านั้น ทั้งนี้ขึ้นอยู่กับความต้องการในการสื่อสารว่าต้องการสื่อสารแบบทิศทางเดียวหรือสองทิศทาง

- **การสื่อสาร RS232 แบบสองทิศทาง** ซึ่งจะมีทั้งการรับข้อมูลและส่งข้อมูลไปมา ระหว่างด้านรับและด้านส่ง โดยในกรณีนี้จะต้องใช้สายสัญญาณจำนวน 3 เส้น สัญญาณรับข้อมูล (RXD)

สัญญาณส่งข้อมูล(TXD) และสัญญาณอ้างอิง (GND) โดยในการเชื่อมต่อสายนั้นจะต้องทำการสลับสัญญาณกับอุปกรณ์ปลายทางด้วย คือ สัญญาณส่ง (TXD) จากบอร์ด CP-AVR 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-AVR ทั้ง 3 รุ่น นั้น จะเป็นจุดเชื่อมต่อของสัญญาณ รับ-ส่ง ข้อมูล ที่เปลี่ยนระดับสัญญาณเป็นแบบ RS232 แล้ว ซึ่งจะมีลักษณะเป็นแบบขั้ว CPA ขนาด 4 PIN สำหรับใช้เป็นจุดเชื่อมต่อสัญญาณ รับ-ส่ง ข้อมูลกับอุปกรณ์ภายนอก โดยมีลักษณะการจัดเรียงสัญญาณดังนี้



รูปแสดง ขั้วต่อสัญญาณ RS232 **ของบอร์ด** CP-AVR V3.0/V3.0 EXP/V4.0

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

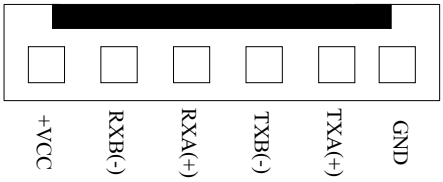
****หมายเหตุ*** สำหรับไอซี Line Drive แบบ RS232 นั้น จะจัดเป็นอุปกรณ์มาตรฐานของบอร์ดในตระกูล CP-AVR ซึ่งจะมีติดตั้งให้ไปกับบอร์ดอยู่แล้ว ผู้ใช้ไม่ต้องจัดหาเพิ่มเติม แต่พึงระลึกไว้เสมอว่า จะต้องทำการติดตั้งไอซี Line Driver สำหรับเลือกชนิดสัญญาณทางไฟฟ้าของการสื่อสารอนุกรม ได้เพียงอย่างใดอย่างหนึ่งเท่านั้น 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 เพื่อทำหน้าที่เปลี่ยนระดับสัญญาณไฟฟ้าในการ 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 เลยทีเดียว 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-AVR V4.0 เท่านั้น

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



รูปแสดง ขั้วต่อสัญญาณ RS422/485 **ของบอร์ด** CP-AVR V4.0 **เมื่อเลือกเป็น** RS422

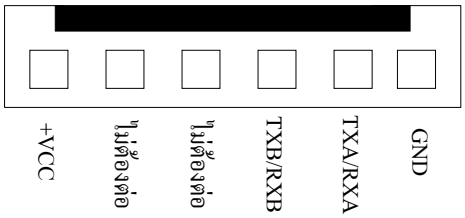
การสื่อสารอนุกรมแบบ RS485

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-AVR V4.0 นั้น จะกำหนดให้สัญญาณ PD7 ทำหน้าที่สำหรับควบคุมทิศทางของข้อมูลว่าจะให้เป็นรับหรือส่ง โดยถ้าควบคุมให้ PD7 มีสภาวะเป็น “1” จะเป็นการกำหนดทิศทางให้เป็นฝ่ายส่งข้อมูล แต่ถ้าสภาวะของ PD7 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



รูปแสดง ขั้วต่อสัญญาณ RS422/485 **ของบอร์ด** CP-AVR V4.0 **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 PD7
- **Jumper F/H** (Full/Half) เป็น Jumper ใช้สำหรับเลือกกำหนดรูปแบบการสื่อสารให้เป็นแบบ Full Duplex (F) หรือ Half Duplex (H) โดยถ้าต้องการใช้งานแบบ RS422 จะต้องเลือกกำหนด Jumper นี้ไว้ทางด้าน 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 ซึ่งถ้าหากว่ามีการต่อสายสัญญาณระยะทางไกลๆหรือมีการต่อสายระยะทางไกล้ๆแต่ไม่ได้มีการส่งข้อมูลออกมาในสายตลอดเวลาแล้ว ควรที่จะทำการ 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

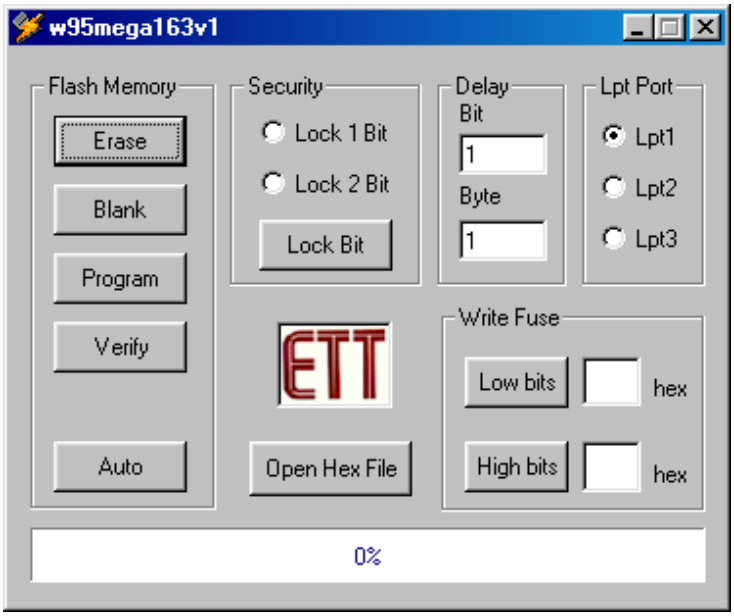
การใช้งานโปรแกรม w95s8535v5 และ w95mega163v1 สำหรับดาวน์โหลดข้อมูล

ก่อนการ Download โปรแกรมทุกครั้งเราควรจะตรวจสอบให้แน่ใจก่อนว่า ได้จ่ายไฟเลี้ยงให้กับบอร์ดถูกต้องแล้ว ซึ่งจะสังเกตได้จาก LED ตัวสีแดงจะสว่างตลอดเวลาที่เราจ่ายไฟให้กับบอร์ดทั้ง 3 รุ่น สำหรับไฟเลี้ยงนั้นจะเป็นไฟ AC หรือ DC ก็ได้แต่ควรจะมีระดับของแรงดันอยู่ระหว่าง 9 -12 V และกระแส 800 –1000 mA ต่อไปให้เราเชื่อมต่อพอร์ตปริ๊นเตอร์(Printer Port)เข้ากับบอร์ดทดลองด้วยสายดาวน์โหลด ET-PSPI โดยในการเลือกพอร์ตปริ๊นเตอร์สำหรับการดาวน์โหลดข้อมูลนั้น หากคอมฯของท่านมีพอร์ตปริ๊นเตอร์มากกว่า 1 พอร์ต ท่านก็สามารถเลือกพอร์ตปริ๊นเตอร์พอร์ตใดก็ได้สำหรับต่อสายดาวน์โหลด เมื่อต่อสายดาวน์โหลดข้อมูลเสร็จแล้ว ให้ท่านทำการเปิดโปรแกรม w95s8535v5 ถ้าท่านใช้ CPU เบอร์ 90S8535 แต่ถ้าหากท่านใช้เบอร์ ATmega163 ก็ให้ท่านเลือกเปิดโปรแกรม w95mega163v1 ขึ้นมาแทน ซึ่งโปรแกรมทั้งสองตัวนี้เป็นโปรแกรมที่จะใช้ในการดาวน์โหลดข้อมูลลงไปที่ตัว CPU นั่นเอง โปรแกรม w95s8535v5 และ w95mega163v1 จะมีฟังก์ชั่นที่ท่านต้องทำการกำหนดก่อนเริ่มทำการดาวน์โหลดโปรแกรมดังนี้

ฟังก์ชั่น Lpt Port

ก่อนการใช้งานโปรแกรม w95s8535v5 และ w95mega163v1 ในครั้งแรกสุด เราจะต้องทำการเลือกพอร์ตปริ๊นเตอร์ที่เราจะใช้ในการดาวน์โหลดโปรแกรมก่อน โดยท่านสามารถคลิกเลือกพอร์ตปริ๊นเตอร์ได้ในกรอบ Lpt Port ซึ่งในกรอบนี้จะมีตัวเลือก 3 ตัวด้วยกันนั่นคือ Lpt1, Lpt2 และ Lpt3 ซึ่งถ้าหากเราเลือกพอร์ตไม่ตรงและพยายามที่จะทำการโปรแกรมข้อมูล โปรแกรมจะแสดงกล่องข้อความผิดพลาดขึ้นมา วิธีแก้ก็คือให้เรากดปุ่ม Ignore และโปรแกรมจะแสดงกล่องข้อความ “Error! Please check your CPU, power supply and printer port.” ให้เรากดปุ่ม OK จากนั้นก็เข้าไปเลือกพอร์ตปริ๊นเตอร์ใหม่ให้ถูกต้อง โปรแกรมก็จะสามารถทำงานได้

รูปแสดง โปรแกรมดาวน์โหลด w95s8535v5



รูปแสดง โปรแกรมดาวน์โหลด w95mega163v1

ฟังก์ชั่น Delay

ในการดาวน์โหลดข้อมูลลงไปที่ไมโครคอนโทรลเลอร์นั้นเนื่องจากความเร็ว CPU ของคอมพิวเตอร์ของผู้ใช้งานมีความเร็วแตกต่างกันไป ดังนั้นเพื่อให้การดาวโหลดข้อมูลอยู่ในมาตรฐานความเร็วที่สามารถสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์ได้ เราจึงต้องเพิ่มฟังก์ชั่น Delay ขึ้นมาเพื่อให้ผู้ใช้ทำการปรับความเร็วในการส่งข้อมูลผ่านพอร์ตปริ๊นเตอร์ให้ช้าลง เราขอแนะนำให้ท่านทดลองดาวน์โหลดข้อมูลก่อน หากการดาวน์โหลดข้อมูลไม่สามารถทำได้ ท่านจึงค่อยๆเพิ่มค่าในช่อง Bit กับ Byte ขึ้นไปเรื่อยๆจนมันสามารถดาวน์โหลดข้อมูลได้โดยไม่เกิดการผิดพลาด

ฟังก์ชั่น Write Fuse(เฉพาะ w95mega163v1)

ฟังก์ชั่นนี้จะใช้ในการโปรแกรม Fuse bit ไบต์สูงและไบต์ต่ำ สำหรับข้อมูลรายละเอียดของ Fuse bit ทั้งสองไบต์นี้ท่านสามารถศึกษาได้จาก datasheet ของ CPU ฟังก์ชั่นนี้จะใช้ได้แต่เฉพาะกับ ATmega163 เท่านั้น เนื่องจาก CPU เบอร์นี้มี Fuse bit หลายตัวที่จำเป็นต่อการทำงาน

เราควรจะทำการโปรแกรม Fuse Bits ก่อนการโปรแกรม Lock Bit และเมื่อเราทำการ Erase CPU จะไม่มีผลกระทบใดๆ ต่อ Fuse Bits ดังนั้นหากเราต้องการเปลี่ยนสถานะของมันเราจะทำโดยการโปรแกรมค่าใหม่ลงไปให้กับมันแทน

ข้อสังเกตุ สำหรับการใช้ CPU เบอร์ ATmega163 ในครั้งแรกกับบอร์ดทั้ง 3 รุ่น เราขอแนะนำให้ท่านทำการโปรแกรม Fuse Bits Low ก่อน โดยการพิมพ์ FA ลงไปที่ช่อง Bits Low และกดที่ปุ่ม Bits Low การกระทำเช่นนี้ก็เพื่อให้เราสามารถใช้คริสตอล 8 MHz ที่ต่ออยู่ภายนอกได้ แต่หากท่านไม่ทำการโปรแกรม Fuse Bits Low ก่อน แต่นำมันไปใช้งานเลย CPU จะใช้สัญญาณนาฬิกาที่อยู่ภายในตัวมันซึ่งจะมีค่าประมาณ 1 MHz

ฟังก์ชั่น Security

เราจะใช้ฟังก์ชั่นนี้เพื่อการโปรแกรม Lock Bit1 และ 2 (เมื่อLock Bit ใดถูกโปรแกรม Lock Bit นั้นจะมีค่าเป็น 0) ซึ่งการโปรแกรม Lock Bit1 จะทำให้เราไม่สามารถทำการโปรแกรมข้อมูลลงบนหน่วยความจำแบบแฟลชและEEPROMได้ ส่วนการโปรแกรม Lock Bit2 จะให้ผลเหมือนกับการโปรแกรม Lock Bit1 และนอกจากนั้นแล้วเรายังจะไม่สามารถทำการ Verify หรือ อ่านข้อมูลได้ด้วย

อย่างไรก็ตามสถานะของ Lock bit1 และ 2 จะเปลี่ยนเป็น 1(1 ในที่นี้หมายถึงการไม่ได้ถูกโปรแกรม) ทุกครั้งเมื่อเราทำการ Erase ข้อมูล

ปุ่ม Open Hex File

ก่อนเราจะทำการเขียนโปรแกรมลงไปที่หน่วยความจำแฟลชของไมโครคอนโทรลเลอร์ทุกครั้ง แน่นอนที่สุดเราจะต้องมีข้อมูลก่อน ดังนั้นปุ่มนี้จะใช้สำหรับนำข้อมูลที่อยู่ในหน่วยความจำถาวรเช่น ฮาร์ดดิสก์ มาเก็บไว้ในหน่วยความจำชั่วคราวของเครื่องเพื่อรอที่จะถูกเขียนลงไปยังหน่วยความจำแฟลชของไมโครคอนโทรลเลอร์ โดยข้อมูลที่เราจะทำการอ่านเข้ามาเก็บในหน่วยความจำชั่วคราวนี้จะต้องอยู่ในรูปแบบภาษาเครื่องหรือที่เราเรียกว่า Hex File(*.hex) เท่านั้น สำหรับในคู่มือการใช้งานเล่มนี้จะแนะนำการใช้โปรแกรม AStudio4 AVR Assembler เพื่อทำการคอมไพล์ไฟล์แอสเซมบลี (*.asm) ให้เป็น Hex File อีกทีหนึ่ง

ปุ่ม Erase

ปุ่มนี้จะใช้ในการเคลียร์หน่วยความจำแบบแฟลชของ CPU ให้พร้อมสำหรับการรับข้อมูลใหม่ การ Erase ทุกครั้งจะทำให้หน่วยความจำแบบแฟลชมีข้อมูลเป็น FFh ทั้งหมด และเราจะต้องทำการ Erase ข้อมูลทุกครั้งก่อนการโปรแกรมข้อมูลชุดใหม่ลงไป

ปุ่ม Blank

ปุ่มนี้จะใช้ในการตรวจสอบข้อมูลที่อยู่ในหน่วยความจำแบบแฟลชว่ามีข้อมูลเป็น FF หรือหน่วยความจำถูกเคลียร์ทั้งหมดหรือไม่ หากพบว่าข้อมูลในหน่วยความจำไม่เท่ากับ FF มันจะแสดงกล่องข้อความเล็กๆ ว่า Not Blank ในการใช้งานปกติเราไม่มีความจำเป็นต้องใช้ปุ่มนี้

ปุ่ม Program

ปุ่มนี้จะใช้ในการโปรแกรมข้อมูลลงไปเก็บยังหน่วยความจำแฟลชของ CPU

ปุ่ม Verify

ปุ่มนี้จะใช้ในการตรวจสอบว่าข้อมูลที่อยู่ในหน่วยความจำแบบแฟลชของ CPU นั้นเป็นข้อมูลตัวเดียวกันกับที่อยู่ในหน่วยความจำชั่วความที่อยู่บนเครื่องคอมพิวเตอร์หรือไม่ ถ้าไม่เท่ากันมันจะแสดงกล่องข้อความบอกว่า Verify Error ดังนั้นปุ่มนี้จึงมีไว้สำหรับการตรวจสอบข้อมูลที่เราได้โปรแกรมลงไปว่ามันได้ถูกโปรแกรมลงไปอย่างถูกต้องหรือไม่นั่นเอง

ปุ่ม Auto

ปุ่มนี้จะทำการโปรแกรมข้อมูลแบบอัตโนมัติ โดยมันจะเริ่มทำตั้งแต่การ Erase, Blank, Program และสิ้นสุดที่ Verify ดังนั้นก่อนที่เราจะกดปุ่มนี้ เราควรจะมีข้อมูลอยู่ในหน่วยความจำชั่วคราวก่อนโดยทำการ Open Hex File เพื่อเลือกโปรแกรมที่เราต้องการทำการโปรแกรมจากนั้นจึงกดปุ่ม Auto

ในการกดปุ่ม Erase, Blank, Program, Verify หรือ Auto หากการทำงานไม่เกิดการผิดพลาด ท่านจะสังเกตเห็น LED สีเขียวกระพริบ ซึ่งจะเป็นตัวแสดงให้เราทราบว่าขณะนี้ข้อมูลกำลังถูกส่งผ่านสาย MOSI

สำหรับโปรแกรม w95s8535v5 และ w95mega163v1 นี้สามารถทำงานบนระบบปฏิบัติการ Windows 95/98/ME ได้ทันที ส่วนบนระบบปฏิบัติการ Windows NT/2000/XP นั้นผู้ใช้งานจะต้องเปิดการทำงานของพอร์ตปริ๊นเตอร์ก่อนท่านจึงจะสามารถทำการดาวน์โหลดข้อมูลได้ ในการเปิดการทำงานของพอร์ตปริ๊นเตอร์บน Windows NT/2000/XP นั้น เพื่อความสะดวกเราขอแนะนำให้ท่านใช้โปรแกรม User Port ซึ่งอยู่ในแผ่น CD-ROM

แนะนำการใช้โปรแกรม Astudio4 (AVR assembler)

สำหรับในหัวข้อนี้เราจะอธิบายถึงวิธีการใช้งานโปรแกรม Astudio4 เวอร์ชั่น 4.05 อย่างง่าย นั่นคือเราจะอธิบายเฉพาะการใช้ text editer และวิธีการคอมไพล์โปรแกรมเท่านั้น แต่ในส่วนอื่นเช่นการ debug program, chip simulator หากท่านใดสนใจก็สามารถศึกษาได้จากเมนู Help ของ Astudio4

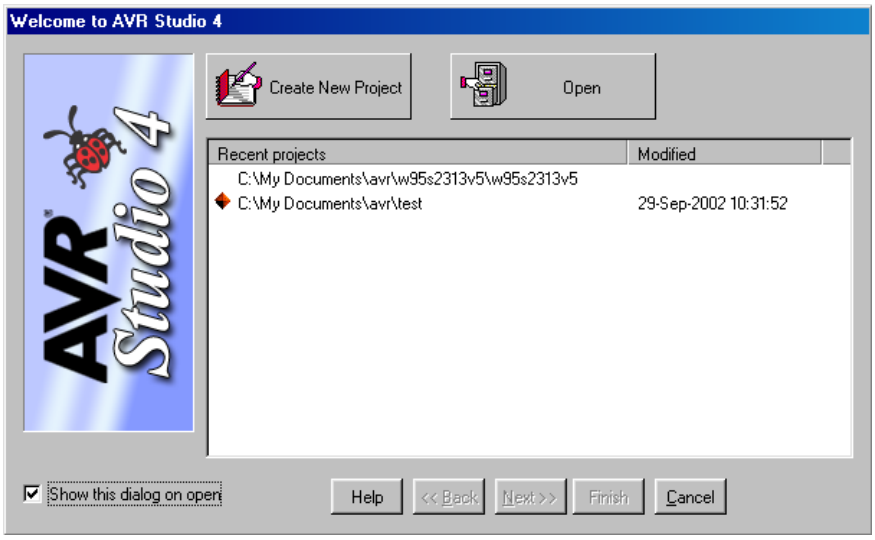
สำหรับโปรแกรมนี้มันสามารถทำงานได้บนวินโดวส์ 9x/ME/NT/2000/XP ดังนั้นจึงไม่มีปัญหาใดๆ สำหรับผู้ใช้ระบบปฏิบัติการที่ต่างชนิดกัน การติดตั้งโปรแกรมก็ทำได้ไม่ยาก โดยให้ท่านคลายไฟล์ Astudio4(zip file)ออกมา จากนั้นก็รันไฟล์ setup.exe และก็ทำตามขั้นตอนที่แนะนำไปเรื่อยๆ จนการติดตั้งเสร็จสมบูรณ์ ซึ่งหลังการติดตั้งโปรแกรมนี้มันก็พร้อมที่จะทำงานทันทีโดยเราไม่ต้องเสียเวลารีสตาร์ตเครื่องใหม่

System requirement

- โปรแกรมนี้รันบนระบบปฏิบัติการ windows 9x/ME/NT/2000/XP
- ความเร็วของ CPU Intel Pentium 200 MHz
- ความละเอียดหน้าจอไม่ควรต่ำกว่า 800x600 pixels
- แรม 64 MB
- ที่ว่างฮาร์ดดิสก์ 15 MB ขึ้นไป

การสร้างไฟล์โปรเจ็กต์(*.aps)และไฟล์แอสเซมบลี(*.asm)

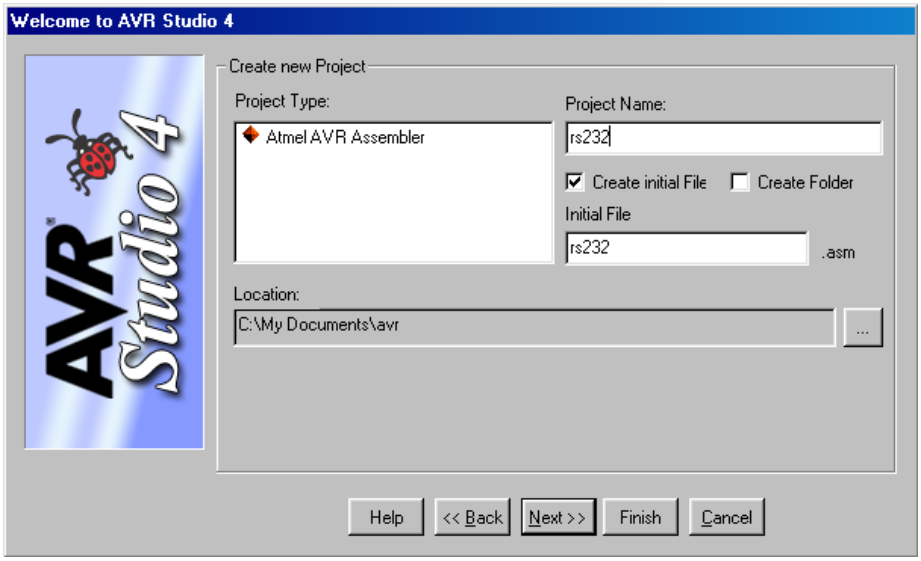
เมื่อการติดตั้งเสร็จสมบูรณ์แล้ว และเราเปิดโปรแกรมนี้ขึ้นมา เราจะพบหน้าต่าง Welcome to AVR Studio4 ดังนี้



รูปแสดง หน้าต่าง Welcome to AVR Studio 4

หน้าต่างนี้จะมีไว้สำหรับให้เราสามารถทำการสร้างโปรเจ็กต์ไฟล์ใหม่หรือสามารถเปิดโปรเจ็กต์ไฟล์ที่มีอยู่แล้วได้ โปรเจ็กต์ไฟล์จะมีนามสกุลเป็น .aps สำหรับหน้าต่างนี้มันจะแสดงเป็นอันดับแรกของการเปิดโปรแกรมทุกครั้ง หากเราไม่ต้องการให้มันแสดงหน้าต่างนี้ เราสามารถทำได้โดยการคลิกเมาส์ 1 ครั้งที่เช็คบ็อก Show this dialog on open ที่อยู่ทางด้านล่างซ้ายมือของหน้าต่างนี้ เพื่อเอาเครื่องหมายถูกที่กาอยู่ในเช็คบ็อกออกไป ซึ่งในการเปิดโปรแกรมครั้งต่อไปท่านจะไม่พบหน้าต่างนี้อีก และการเปิดหรือสร้างโปรเจ็กต์ใหม่เรา

สามารถทำได้ผ่านทางออปชั่น Project ที่อยู่ในเมนูบาร์แทน แต่สำหรับการอธิบายในที่นี้เราจะใช้หน้าต่างนี้ในการสร้างโปรเจ็กต์ใหม่ โดยให้ท่านคลิกที่ปุ่ม Create New Project ซึ่งท่านจะพบกับหน้าต่างใหม่ดังรูปต่อไปนี้



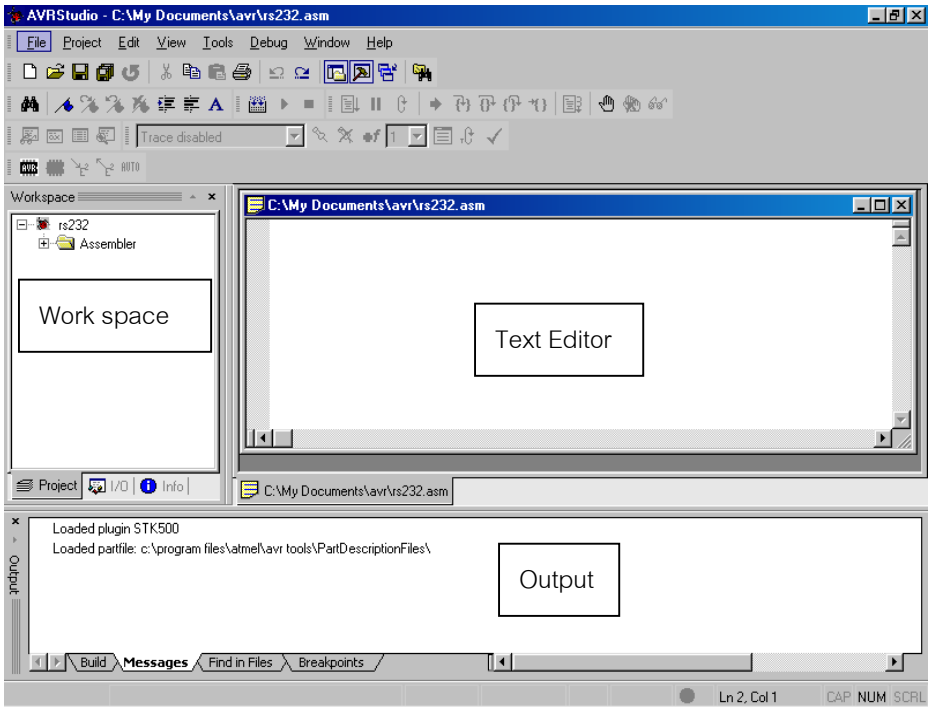
รูปแสดง หน้าต่างที่ใช้ในการสร้างโปรเจ็กต์ไฟล์

ในขณะที่ทำการสร้างไฟล์โปรเจ็กต์ โปรแกรมสามารถที่จะสร้างไฟล์ .asm ซึ่งเป็นไฟล์โปรแกรมภาษาแอสเซมบลีได้พร้อมกันไปในตัวโดยการกาเครื่องหมายถูกที่เช็คบ็อก Create Initial File จากนั้นให้เราพิมพ์ชื่อไฟล์โปรเจ็กต์ที่ช่องข้อความ Project Name ในตัวอย่างเราจะใช้ชื่อไฟล์โปรเจ็กต์ rs232 และในระหว่างที่เราพิมพ์ชื่อไฟล์โปรเจ็กต์ เราจะสังเกตุเห็นว่าโปรแกรมจะพิมพ์ชื่อไฟล์โปรเจ็กต์ลงที่ช่องข้อความ Initial File ไปด้วย ซึ่งจะเป็นการสร้างไฟล์โปรแกรม .asm (ไฟล์แอสเซมบลี) และหากเราต้องการให้โปรแกรมสร้างไฟล์ .asm ที่มีชื่อต่างจากชื่อไฟล์โปรเจ็กต์ เราก็สามารถแก้ไขชื่อไฟล์ .asm ได้ที่ช่องข้อความ Initial File นี้

ข้อควรระวังในการตั้งชื่อไฟล์แอสเซมบลีต่างจากชื่อไฟล์โปรเจ็กต์นั่นก็คือ เมื่อเราทำการคอมไพล์แล้วไฟล์ต่างๆ ที่โปรแกรมทำการสร้างขึ้นมามันจะตั้งชื่อตามชื่อไฟล์โปรเจ็กต์เช่น ถ้าเราตั้งชื่อไฟล์โปรเจ็กต์เป็น Testproject และเราตั้งชื่อไฟล์แอสเซมบลีชื่อ rs232 หลังการคอมไพล์เราจะได้ไฟล์ Testproject.hex ดังนี้เป็นต้น

หากเราไม่ต้องการให้มันสร้างไฟล์โปรแกรม .asm ขึ้นในขณะที่เราสร้างไฟล์โปรเจ็กต์ เราสามารถทำได้โดยการคลิกที่เช็คบ็อก Create Initial File เพื่อเอาเครื่องหมายถูกที่กาอยู่ออก ส่วนที่เช็คบ็อก Create Folder จะเป็นการสร้างโฟลเดอร์ขึ้นมาเพื่อเก็บไฟล์โปรเจ็คไว้นั่นเอง ถัดลงมาที่ช่องข้อความ Location จะเป็นตำแหน่งของไฟล์โปรเจ็คที่จะถูกนำไปเก็บ ซึ่งเราสามารถเลือกที่จะเก็บไฟล์ที่ใหม่ได้โดยการคลิกที่ปุ่ม ... ที่อยู่ท้ายช่องรับความหรือพิมพ์ชื่อ path ลงไปเลยก็ได้เช่นเดียวกัน สำหรับในตัวอย่างนี้เราจะเก็บไฟล์โปรเจ็กต์ไว้ที่ C:\My Documents\avr และทำการสร้างไฟล์ .asm ขึ้นพร้อมกับการสร้างไฟล์โปรเจ็กต์ด้วย

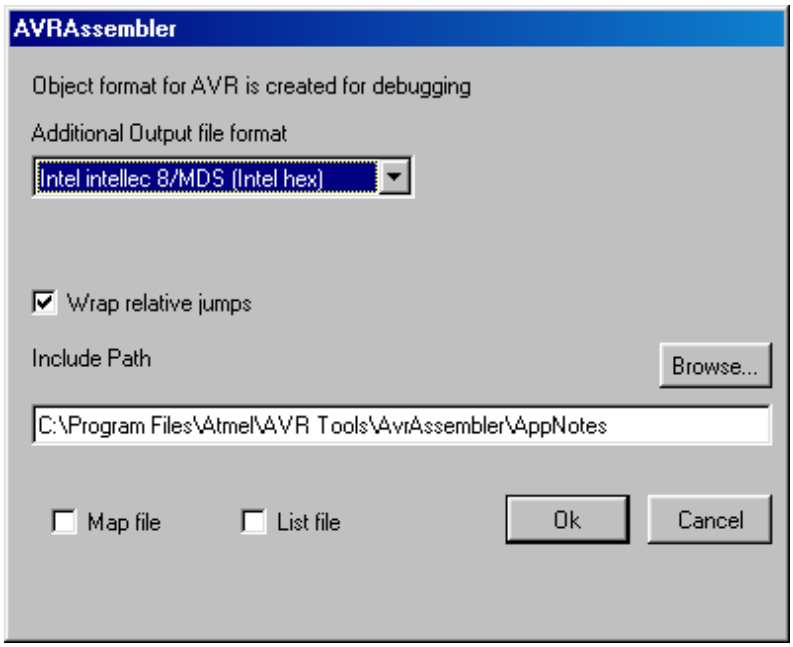
เมื่อทุกอย่างถูกต้องดีแล้วก็ให้เราทำการคลิกที่ปุ่ม Finish เพื่อจะเข้าหน้าจอ text editor สำหรับเขียนโปรแกรมต่อไป หน้าจอที่เกิดขึ้นใหม่หลังการกดปุ่ม Finish แสดงได้ดังรูปต่อไปนี้ (ถ้าเรากดปุ่ม Next โปรแกรมจะให้เราเลือก paltform และ MCU ที่เราจะใช้ในการซิมมูเลตและ debug โปรแกรม ต่อไป)



รูปแสดง หน้าจอที่เกิดขึ้นหลังการกดปุ่ม Finish

จากรูปข้างบน จะประกอบไปด้วยหน้าต่าง Work space ซึ่งมีไว้สำหรับจัดการกับไฟล์ในโปรเจ็กต์เช่น การ Add existing file, Remove File หรือ การ Set as Entry File เป็นต้น ส่วนหน้าต่างถัดมาคือ Text Editor มีไว้สำหรับให้เราเขียนโปรแกรมลงไป และสุดท้ายเป็นหน้าต่าง Output ที่ทำหน้าที่แสดงข้อความบอกให้เราทราบถึงผลจากการคอมไฟล์โปรแกรม ซึ่งหากท่านไม่พบหน้าต่างทั้งสามนี้ท่านสามารถเปิดมันขึ้นมาได้เอง โดยการเข้าไปเลือกที่เมนู View

เมื่อเราทำการเขียนโปรแกรมลงไปที่ text editor เรียบร้อยแล้วก็ให้ทำการบันทึกไฟล์โดยการเลือกไปที่เมนู File>save และทำการคอมไพล์โปรแกรมโดยการกดปุ่ม F7 หรือเลือกที่เมนู Project>Build หากเราเขียนโปรแกรมถูกต้อง โปรแกรมจะไม่ฟ้อง Error ที่หน้าต่าง Output แต่หากเขียนโปรแกรมไม่ถูกต้องมันจะฟ้อง Error หรือ Warning ก็ให้เราแก้ไขมันให้ถูกต้องและทำการคอมไพล์ใหม่ซึ่งจะทำให้ได้ไฟล์ rs232.aps, rs232.hex, rs232.lst, rs232.map, rs232.asm และ rs232.obj ขึ้นมา หากเราไม่ต้องการไฟล์ .map และ .lst ให้เราเข้าไปที่เมนู Project>AVR Asembler Setup จากนั้นจะพบหน้าต่าง AVRAssembler ให้เราคลิกเอาเครื่องหมายถูกที่ช่อง Map File กับ Lst File ออกและจากนั้นก็กดปุ่ม Ok ก็เป็นอันเสร็จสิ้น สำหรับไฟล์นามสกุลต่างๆ ดังที่เรากล่าวมาข้างต้นท่านสามารถเปิดเข้าไปดูเนื้อหาข้างในไฟล์ได้ด้วยการเลือกไปที่เมนู File>Open File จากนั้นก็เลือกไฟล์ที่ท่านต้องการเปิดและกดปุ่ม Open



รูปแสดง หน้าต่าง AVRAssembler

การเปิดไฟล์โปรเจ็กต์และไฟล์แอสเซมบลี

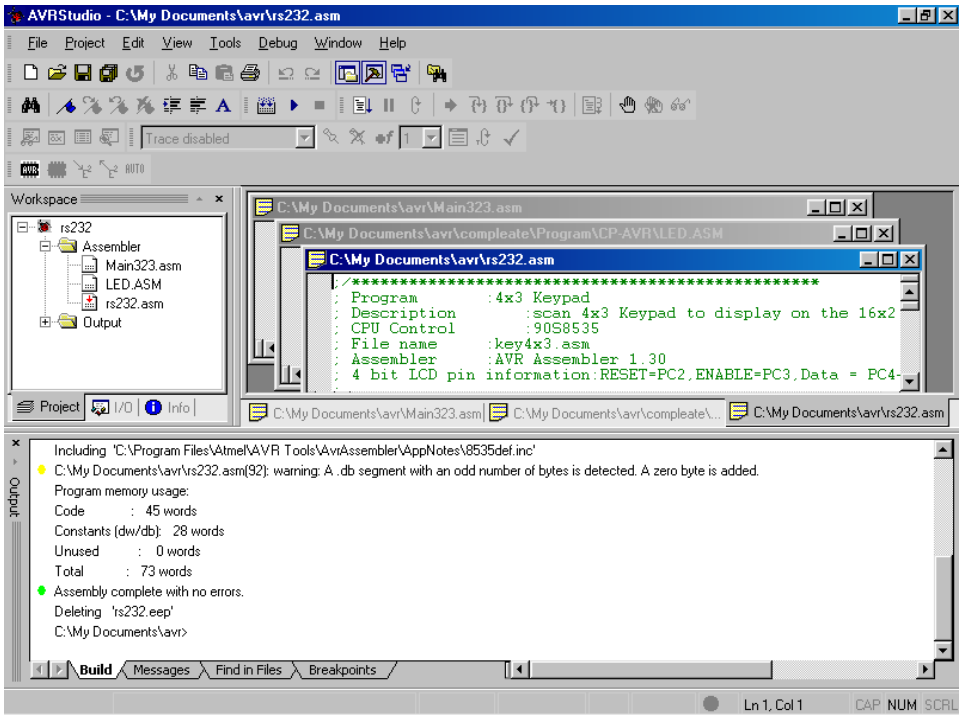
เมื่อเราต้องการเปิดโปรแกรมที่มีอยู่แล้วขึ้นมาแก้ไขเราสามารถทำได้โดยการคลิกเมนู Project>Open Project เพื่อเป็นการเปิดไฟล์โปรเจ็กต์(*.aps) หลังจากที่เราเปิดไฟล์โปรเจ็กต์แล้วจากนั้นก็คลิกเลือกไปที่โฟลเดอร์ Assembler ที่อยู่ในหน้าต่าง Work space เราจะพบไฟล์ .asm ที่เราต้องการแก้ไขอยู่ในนั้น แต่ถ้าหากเราเปิดไฟล์โดยใช้เมนู File>Open File เราก็สามารถที่จะเปิดไฟล์ .asm(หรือไฟล์อื่นๆ ที่ text editor รองรับ) ได้เช่นเดียวกัน แต่เราจะไม่สามารถทำการคอมไพล์มันได้ เป็นได้แต่เพียงการแก้ไขและบันทึกเท่านั้น อย่างไรก็ตาม การเปิดไฟล์โปรเจ็กต์(*.aps) ด้วยเมนู File>Open File นี้ก็สามารถทำได้และให้ผลการทำงานเช่นเดียวกับการเปิดมันด้วยเมนู Project>Open Project

การ Add existing File, Remove File และ Set as Entry File

Add existing File ก็คือการเพิ่มไฟล์ .asm เข้ามาในโปรเจ็กต์ โดยเราสามารถทำได้ 2 วิธีด้วยกันคือ การเลือกไปที่เมนู Project>Add existing File จากนั้นก็ให้เราเลือกไฟล์ .asm ที่เราต้องการนำเข้ามา ส่วนอีกวิธีหนึ่งทำได้โดยการคลิกเมาส์ปุ่มขวาไปที่โฟลเดอร์ Assembler ที่อยู่ในหน้าต่าง Work space จากนั้นเราจะเห็นหน้าต่างออปชั่นเล็กๆ ปรากฏขึ้นมาก็ให้เราเลื่อนเมาส์ไปคลิกเลือกที่ Add existing File จะทำให้เราสามารเลือกไฟล์ .asm ที่เราต้องการได้

Remove File จะทำงานตรงกันข้ามกับ Add existing File นั่นคือมันจะลบไฟล์ .asm ที่เราไม่ต้องการออกจากโปรเจ็กต์ไฟล์ของเรา เราสามารถ Removeไฟล์ได้โดยการคลิกเมาส์ปุ่มขวาเลือกไปที่ไฟล์ที่เราต้องการลบ จากนั้นจะปรากฏหน้าต่างขึ้นมาให้เราเลือกไปที่ Remove File form Project เพียงเท่านี้ไฟล์ที่เราเลือกมันก็จะถูกลบออกไปจากโปรเจ็กต์ที่เรากำลังทำงานอยู่

Set as Entry File จะเป็นการเลือกไฟล์ .asm ที่อยู่ในโปรเจ็กต์ไฟล์ขณะนั้นให้มันพร้อมที่จะถูกคอมไพล์ ฟังก์ชั่นนี้จะมีความจำเป็นมากเมื่อเรามีไฟล์ .asm หลายไฟล์อยู่ในโปรเจ็กต์เดียวกัน ซึ่งโปรแกรม Astudio4 นี้จะไม่ทำการคอมไพล์ไฟล์ .asm ทั้งหมดพร้อมกัน แต่จะเลือกคอมไพล์เฉพาะไฟล์ที่โดน Set as Entry File เท่านั้น เราสามารถรู้ได้ว่าไฟล์ .asm ตัวใดที่ถูก set ไว้โดยการสังเกตไอคอนที่อยู่หน้าชื่อไฟล์จะมีรูปลูกศรสีแดงหันหัวลงดังนี้ หรือดูได้จากรูปนี้



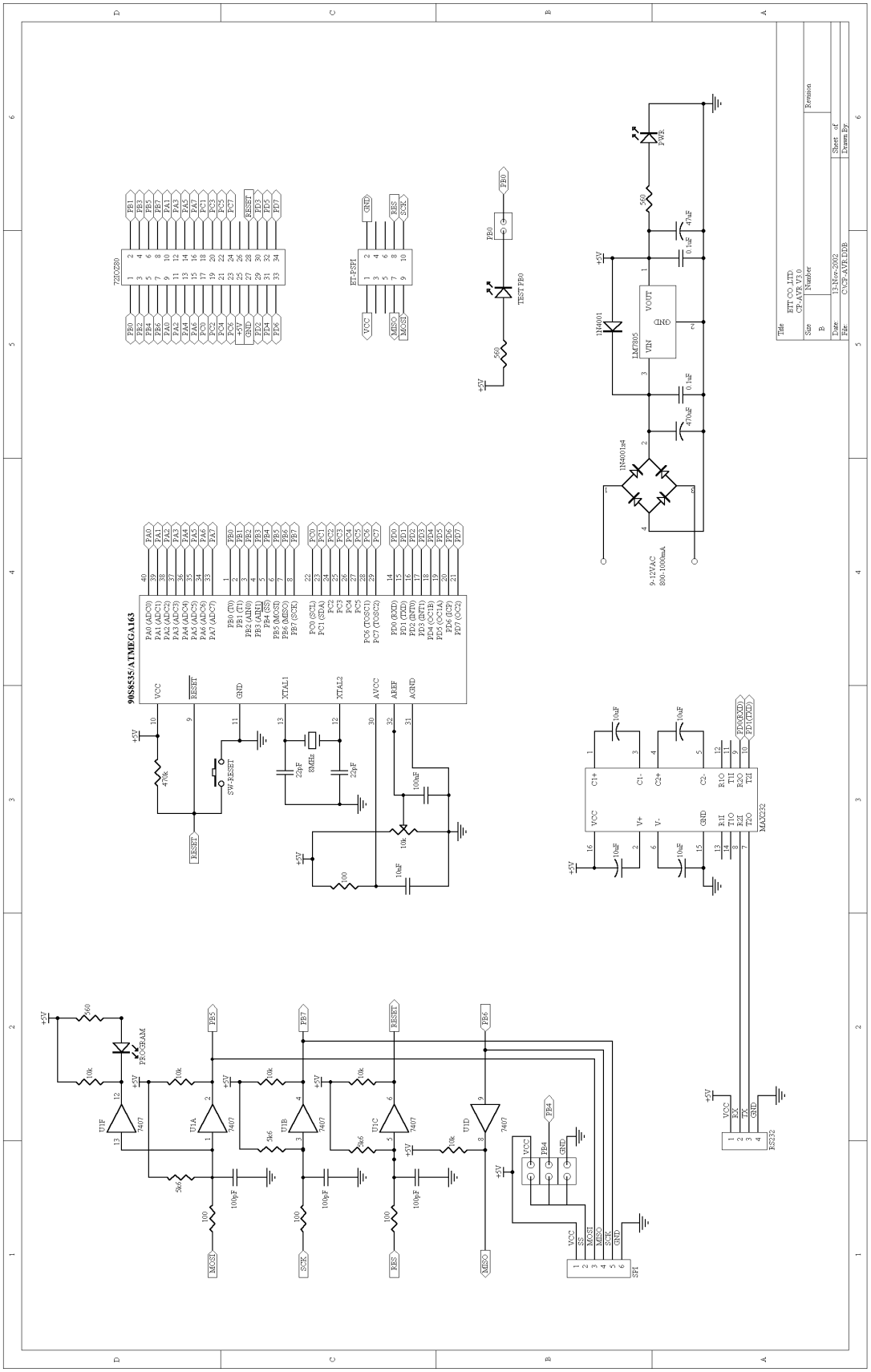
รูปแสดง หน้าจอที่เกิดจากการ Add existing File และหน้าต่าง Output ที่รายงานผลการคอมไพล์โปรดสังเกตการ Warning (วงกลมสีเหลือง) ต่าง ๆด้วย ถึงแม้การคอมไพล์ไม่ผิดพลาดแต่อาจจะมีผลต่อการทำงานของ CPU ได้

สำหรับการ Set as Entry File สามารถทำได้โดยการคลิกเมาส์ปุ่มขวาเลือกไปที่ชื่อไฟล์ที่เราต้องการคอมไพล์จากนั้นจะปรากฏเมนูเล็กๆ ขึ้นมาให้เราเลือกไปที่ Set as Entry File ก็เป็นอันเสร็จ

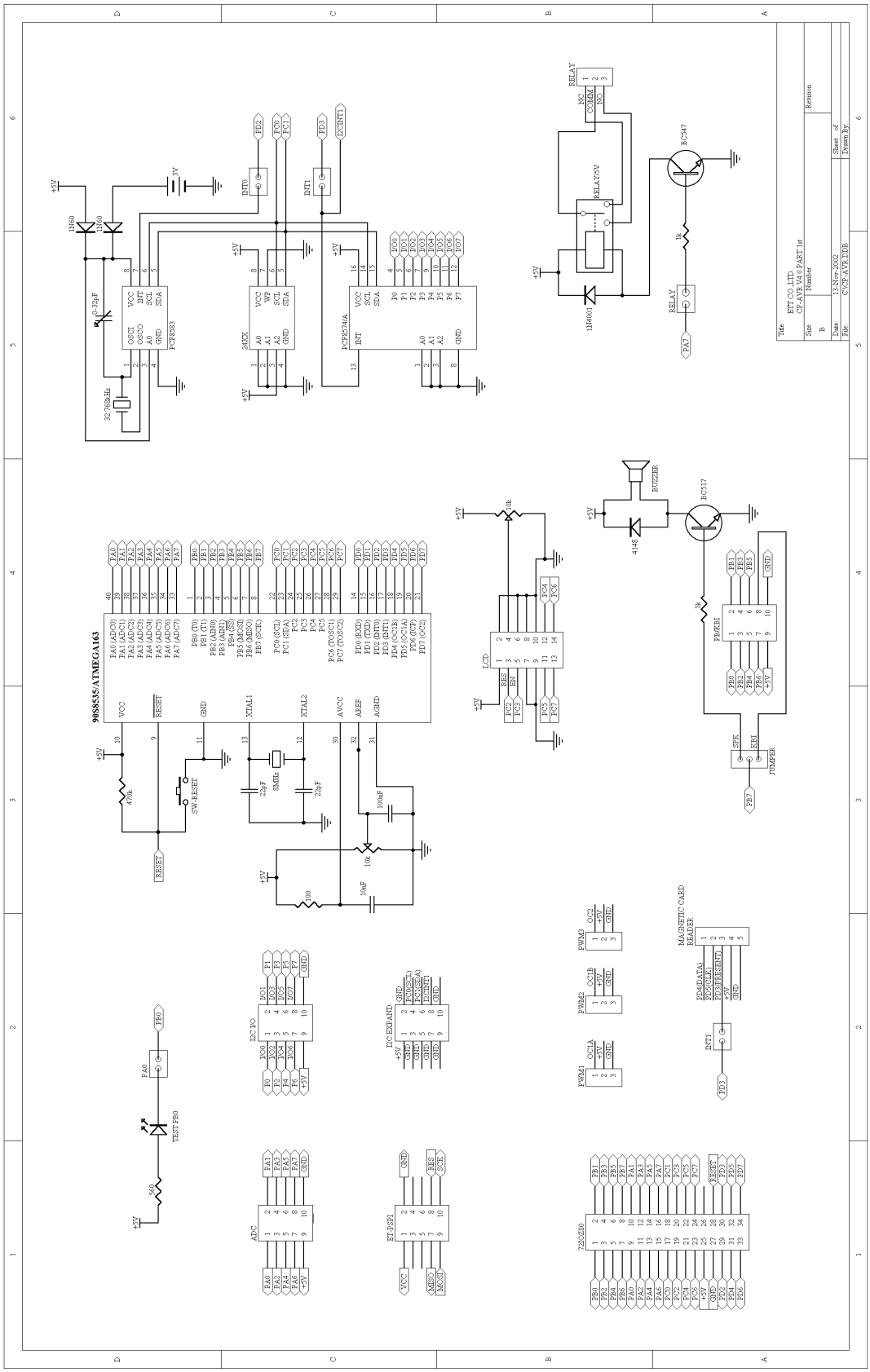
เราควรจำไว้อย่างหนึ่งว่าหลังการคอมไพล์แล้วไม่ว่าเราจะทำการคอมไพล์ไฟล์ .asm ตัวใดก็ตามในโปรเจ็กต์นั้น ไฟล์เอาพุตที่โปรแกรมสร้างขึ้นมานั้นจะมีชื่อตามชื่อไฟล์โปรเจ็กต์นั้นเสมอ มันจะไม่มีชื่อไฟล์ตามไฟล์ .asm ที่ถูกคอมไพล์ ดังนั้นเพื่อป้องกันการสับสนท่านอาจจะให้มีเพียงไฟล์ .asm เพียงไฟล์เดียวอยู่ในไฟล์โปรเจ็กต์แต่ละโปรเจ็กต์ก็ได้และตั้งชื่อให้ทั้งสองไฟล์นี้ตรงกัน

อีกอย่างหนึ่งที่เราอยากจะให้ท่านสังเกต นั่นก็คือผลที่ได้จากการคอมไพล์ในแต่ละครั้งว่าเป็นเช่นไรบ้าง เช่น ขนาดของไฟล์ที่คอมไพล์ได้มีขนาดเท่าไรซึ่งบางทีเราเขียนโปรแกรมเพลินมันอาจจะมีขนาดโตกว่าหน่วยความจำที่จะเก็บมันก็ได้ หรือที่สำคัญอีกอย่างหนึ่งก็คือข้อความ Warning ที่โปรแกรมเตือนดังนี้ “C:\My Documents\avr\rs232.asm(92): warning: A .db segment with an odd number of bytes is detected. A zero byte is added.”

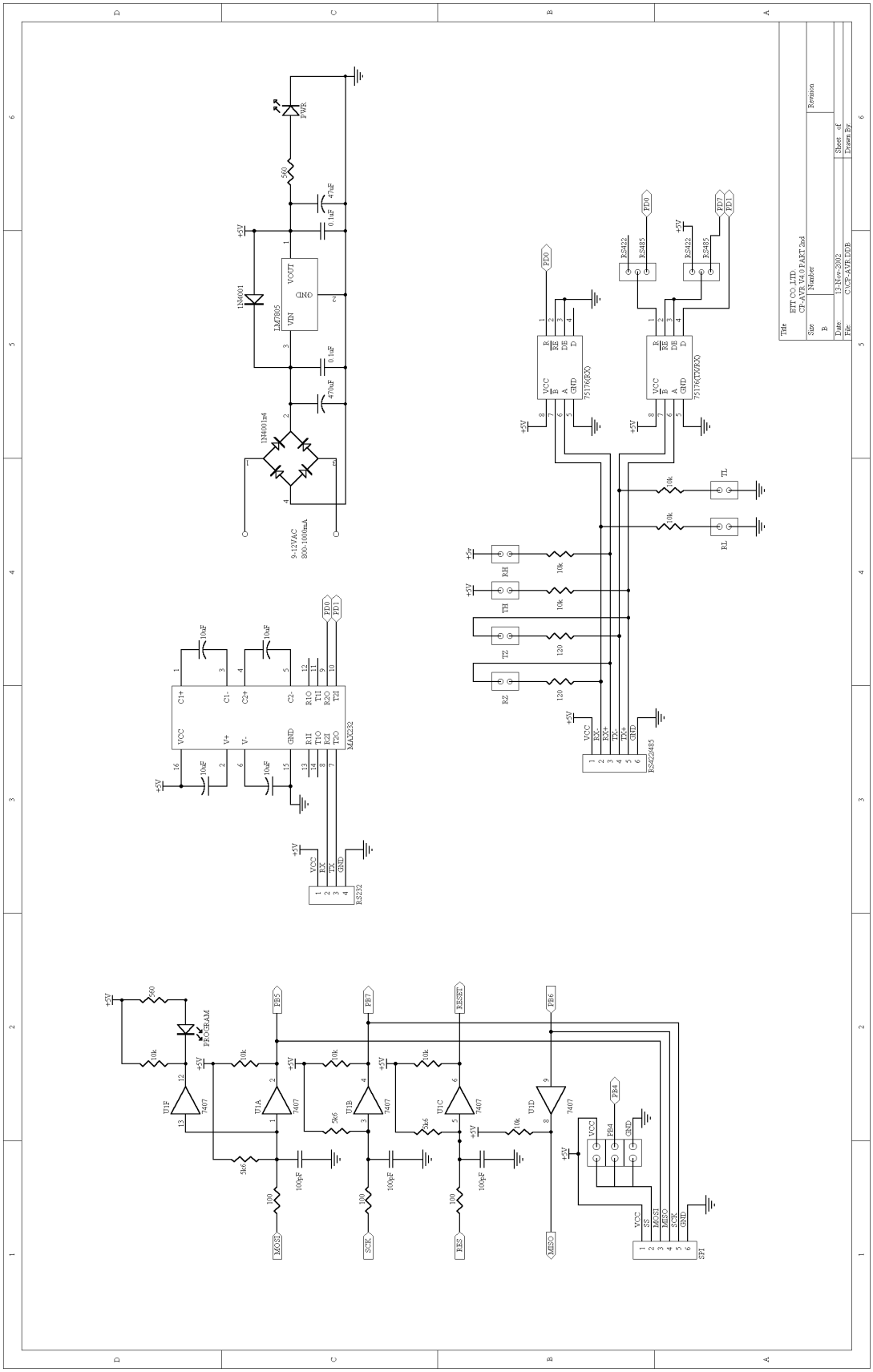
หากท่านใดเคยเล่น AVR และใช้ฟังก์ชั่นที่เกี่ยวกับพอร์ตอนุกรม RS232 คงพอจะทราบดีว่าบางครั้ง AVR มันก็ไม่สามารถแสดงข้อความที่อยู่หลังบรรทัดนี้(บรรทัดที่แสดงข้อความ Warning) ต่อไปได้ ทั้งที่ข้อความยังไม่สิ้นสุด นั่นเป็นเพราะว่ามันจะทำการเขียนข้อมูลลงหน่วยความจำโปรแกรมเป็นทีละ word(2 ไบต์) ดังนั้นเมื่อข้อความที่เราต้องการจะเขียนลงบนหน่วยความจำมันมีจำนวนเป็นเลขคี่ มันก็จะเพิ่มเลข 0 เข้าไปท้ายข้อความในบรรทัดนั้นๆ เพื่อให้ครบจำนวนเป็นเลขคู่เสมอ ทีนี้เมื่อเราเขียนโปรแกรมให้ตรวจสอบข้อความว่าถ้าพบเลข 0 เมื่อใดก็ให้ถือว่าสิ้นสุดข้อความทันที ดังนั้นเมื่อโปรแกรมทำการอ่านและตรวจสอบข้อมูลมาถึงไบต์ที่ AVR เพิ่มเลข 0 ขึ้นมาเอง โปรแกรมก็เข้าใจว่าเราต้องการสิ้นสุดข้อความที่บรรทัดนี้ดังนั้นจึงทำให้ข้อความที่เหลือไม่ถูกตรวจพบนั่นเอง และวิธีแก้ง่ายๆ นั่นก็คือให้เราเพิ่มอักขระอะไรก็ได้เข้าไปอีกตัวหนึ่งในบรรทัดนี้ยกเว้นเลข 0 เท่านี้ก็สามารถแก้การ warining ตรงนี้ได้(เลข 0 ที่เรากำลังพูดถึงกันอยู่ขณะนี้ไม่ใช่เลข “0” ที่เป็นรหัส ASCII นะครับหากแต่เป็นเลข 0 ที่เป็นเซตของเลขจำนวนเต็ม)



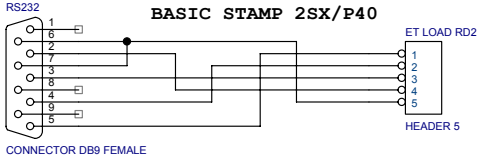
รูปวงจร CP-AVR V3.0/V3.0EXP



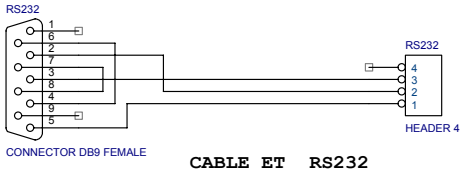
รูปวงจร CP-AVR V4.0



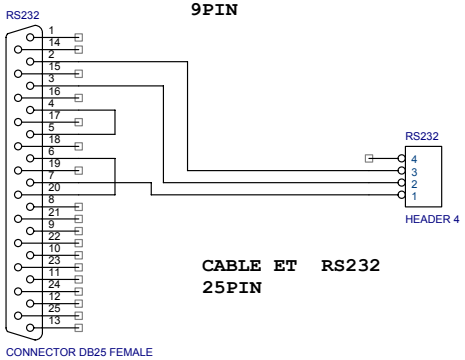
รูปวงจร CP-AVR V4.0 (ต่อ)



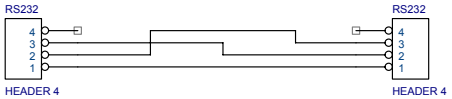
CABLE DOWNLOAD
ET LOAD RD2 /PROGRAM
BASIC STAMP 2SX/P40



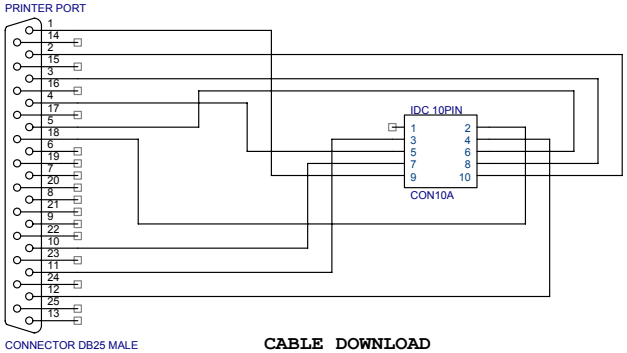
CABLE ET RS232
9PIN



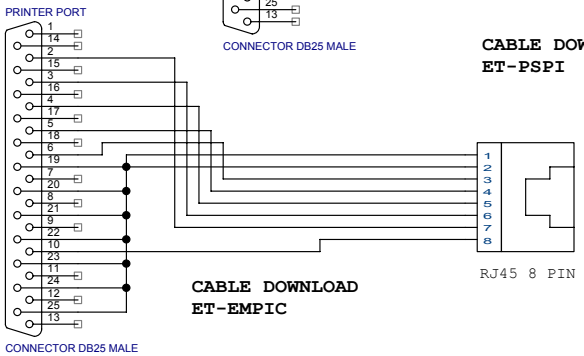
CABLE ET RS232
25PIN



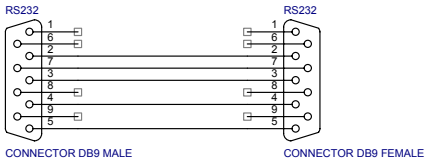
CABLE RS232 ET 4PIN TO ET 4PIN



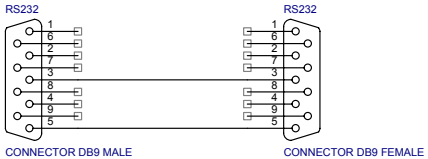
CABLE DOWNLOAD
ET-PSPI



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	