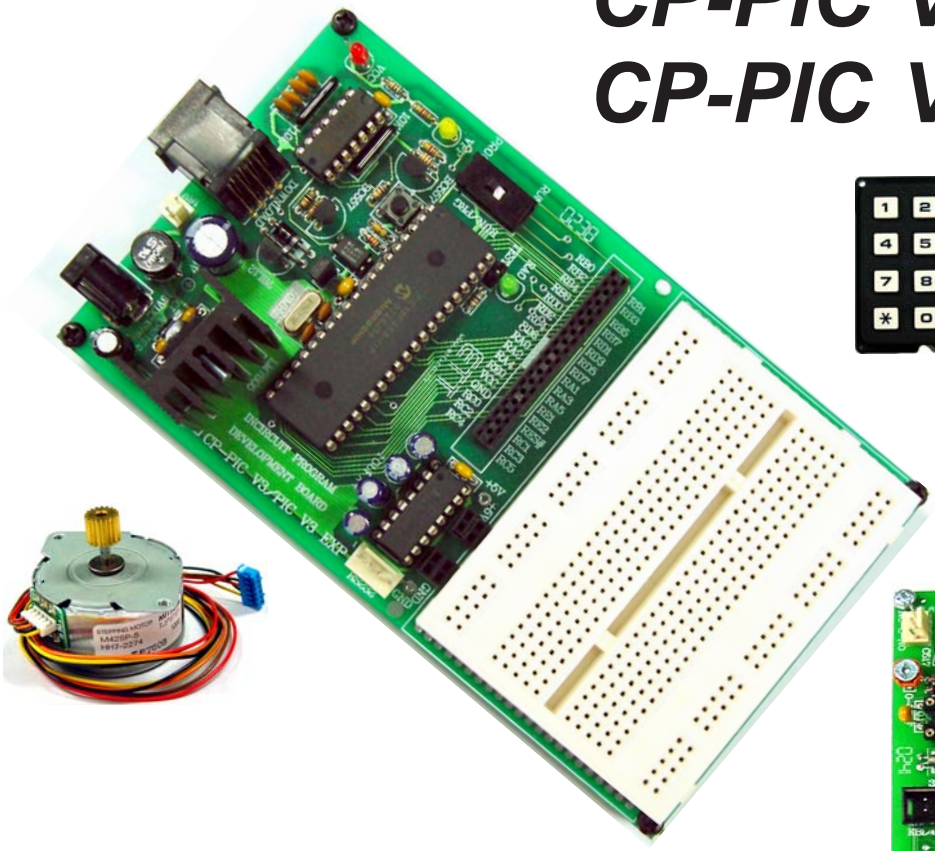
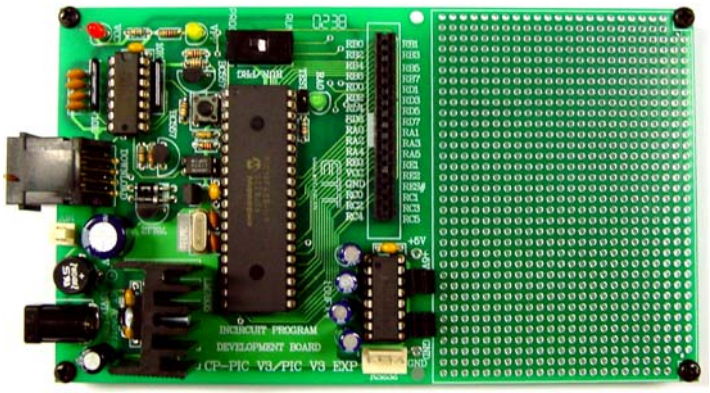
คู่มือการใช้งานบอร์ดไมโครคอนโทรลเลอร์

CP-PIC V3.0
CP-PIC V3.0 EXP
CP-PIC V4.0





ETT
www.ett.co.th

บริษัท อีทีที จำกัด

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

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

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

ชื่อหนังสือ “คู่มือการใช้งานบอร์ดไมโครคอนโทรลเลอร์ CP-AVR V3 & V4”

ISBN

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)

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

คำนำ

คู่มือเล่มนี้จำทำขึ้นเพื่อประกอบการใช้งานบอร์ด CP-PIC V3.0&V4.0 เนื้อหาภายในจะประกอบไปด้วยส่วนต่างๆ เช่น ส่วนประกอบของบอร์ด การใช้งาน การดาวน์โหลด และ การพัฒนาโปรแกรมบอร์ดที่ได้ออกแบบนี้ จะแบ่งเป็น 3 รุ่น คือ CP-PICV3.0, CP-PIC V3.0 EXPANSION และ CP-PICV4.0 ซึ่งทั้งสามรุ่นจะมีคุณสมบัติแตกต่างกันในเรื่องของทรัพยากรณ์ ต่างๆ บนบอร์ดเพื่อให้สามารถเลือกใช้ได้ตามความเหมาะสมกับงาน ในส่วนของ I/O Port ต่างๆได้พยายามจัดสรร ให้มีความอ่อนตัวมากที่สุด เพื่อให้สะดวกต่อการนำไปใช้งาน และในส่วนของการพัฒนาโปรแกรม ซึ่งเราได้ออกแบบวงจรสำหรับโปรแกรมข้อมูลลง CPU ไว้ภายในบอร์ด จึงทำให้ง่ายต่อการพัฒนาโปรแกรม ทั้งนี้หวังว่าคู่มือการใช้งานเล่มนี้จะช่วยให้ท่านสามารถใช้งาน บอร์ด CP-PIC V3.0&V4.0 ได้อย่างถูกต้องและมี ประสิทธิภาพ

ทีมงานอีทีที

สารบัญ

ลักษณะโดยทั่วไป

บอร์ดไมโครคอนโทรลเลอร์ CP-PIC V3.0 & V4.0 ที่ได้ออกแบบนี้ เป็นบอร์ดที่ออกแบบไว้ใช้งานกับ ไมโครคอนโทรลเลอร์ในตระกูล PIC โดยจะสามารถใช้ได้ กับเบอร์ 16F877-20P,18F442 และ 18F458 หรือเบอร์อื่นๆ ที่มีโครงสร้างและตำแหน่งขาสัญญาณเหมือนกันโดย CPU แต่ละเบอร์ก็จะมีคุณสมบัติที่แตกต่างกันซึ่งสามารถสรุปคุณสมบัติของ CPU แต่ละเบอร์อย่างคร่าวๆ ดังตารางด้านล่าง

DEVICE	Program Memory	Data Memory		CAN Module	I/O (Bit)	OSC max (MHz)	Timers	PLL
	Flash	RAM (Bytes)	EEPROM (Byte)					
PIC 16F877	8K (14-Bit Words)	368	256	NO	33	20MHz	3	NO
PIC 18F442	16 Kbyte	768	256	NO	34	40MHz	4	YES
PIC 18F458	32 Kbyte	1536	256	YES	34	40MHz	4	YES

CPU ดังกล่าวจะบรรจุอยู่ภายในตัวถังแบบ DIP ขนาด 40 ขา และมีทรัพยากรต่างๆบรรจุไว้ภายในตัว CPU อย่างครบถ้วน ไม่ว่าจะเป็น ADC/TIMER/COUNTER/PWM หรือ PORT I/O ต่างๆ ซึ่งมีความเหมาะสมในการนำไปประยุกต์ใช้งานในลักษณะต่างๆได้เป็นอย่างดี

สำหรับอุปกรณ์ I/O ต่างๆ ซึ่งไม่ได้มีบรรจุไว้ในตัวCPU ด้วยทางทีมงานอีทีที ก็ได้จัดหาและทำการออกแบบวงจรสำหรับเชื่อมต่อกับอุปกรณ์ต่างๆที่มีความจำเป็นไว้ให้ด้วยแล้ว ไม่ว่าจะเป็นจอแสดงผลแบบ LCD ระบบฐานเวลา RTC วงจร Line Driver สำหรับการสื่อสารข้อมูลอนุกรมแบบ RS232 และ RS422/485 และยังสามารถให้ผู้ใช้ทำการเพิ่มเติมอุปกรณ์ I/O อื่นๆเข้าไปได้อีกตามความจำเป็นในการใช้งาน

โดยลักษณะของบอร์ดไมโครคอนโทรลเลอร์ที่ได้ออกแบบนี้ จะแบ่งออกเป็น 3 รุ่น แต่ละรุ่นก็จะมีทรัพยากรณ์บนบอร์ดแตกต่างกันไป เพื่อ ให้สามารถเลือกใช้งานกันตามความเหมาะสมกับงานดังนี้คือ

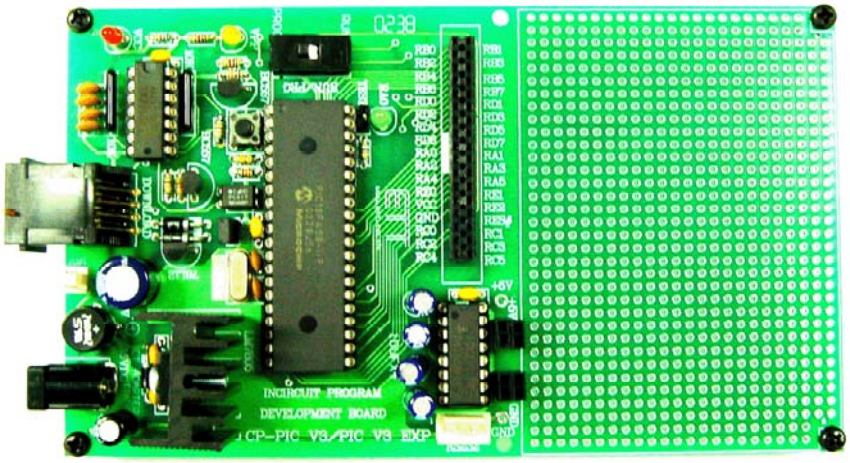
- **CP-PIC V3.0** เป็นบอร์ดไมโครคอนโทรลเลอร์ ซึ่งออกแบบวงจรเฉพาะส่วนพื้นฐานที่จำเป็น เช่น แหล่งจ่ายไฟ วงจรรีเซ็ต วงจรกำเนิดความถี่สัญญาณนาฬิกา วงจรสำหรับ Download โปรแกรม และวงจรสื่อสารอนุกรม ส่วนวงจร I/O ภายนอกนั้น จะไม่ได้จัดเตรียมไว้ให้ด้วย แต่จะทำการต่อสัญญาณ I/O ต่างๆจาก CPU มาไว้ยังขั้วต่อ Connector สำหรับให้ผู้ใช้นำไปเชื่อมต่อกับอุปกรณ์ I/O ภายนอกได้โดยง่าย และยังมีพื้นที่เอนกประสงค์สำหรับให้ผู้ใช้ออกแบบวงจร I/O และต่อวงจร I/O เพิ่มเติมได้เอง เหมาะสำหรับผู้ใช้ที่ต้องการนำบอร์ดไปใช้พัฒนางานต้นแบบโดยการสร้าง I/O ต่างๆขึ้นมาใช้งานเอง ซึ่งในบอร์ดจะประกอบด้วย
  - RS – 232 1 แชนเนล
  - ETT CON 34PIN (ET BUS I/O 34PIN)
  - 5 Volt Regulator On Board
  - วงจรโปรแกรมแบบ High Voltage ภายในบอร์ด

- **CP-PIC V3.0 EXPANSION** จะมีลักษณะเดียวกันกับบอร์ด CP-PIC V3.0 แต่จะมีแผง Photo Board สำหรับให้ผู้ใช้ต่อทดลองวงจร I/O อย่างง่ายๆได้เอง เหมาะสำหรับผู้ใช้ที่ต้องการศึกษาเรียนรู้และต้องการทดลองวงจร I/O ต่างๆ ร่วมกับ CPU อย่างง่ายๆ

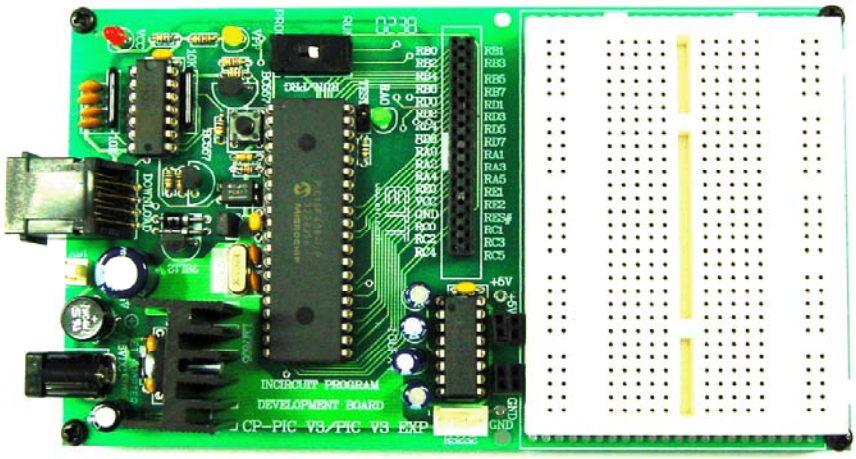
- **CP-PIC V4.0** เป็นบอร์ดไมโครคอนโทรลเลอร์ที่มีการออกแบบวงจรสำหรับเชื่อมต่อกับอุปกรณ์ I/O ภายนอกอื่นๆที่มีความจำเป็นไว้รองรับการใช้งานในลักษณะต่างๆ เพื่อให้ผู้ใช้งานสามารถนำบอร์ดไปใช้งานในลักษณะงานที่แตกต่างกันได้ โดยไม่ต้องดัดแปลงวงจร หรือ อาจดัดแปลงวงจรเพียงเล็กน้อยสำหรับงานบางอย่าง ซึ่งบอร์ดรุ่นนี้เหมาะสำหรับกลุ่มผู้ที่ต้องการนำ บอร์ดไมโครคอนโทรลเลอร์ไปใช้งานจริงๆแต่ไม่สะดวกที่จะสร้างบอร์ดเอง ประกอบด้วยส่วนต่างๆ ดังนี้

- RS – 232 1 แชนเนล
- RS-422/458 1 แชนเนล (IC 75176 เป็น Option)
- ETT CON 34PIN (ET BUS I/O 34PIN)
- 5 Volt Regulator On Board
- วงจรโปรแกรมแบบ High Voltage ภายในบอร์ด
- ใช้ Adaptor 16VDC (Option)
- ADC/IO(CPU) พอร์ตสำหรับต่อ อินพุตอนาลอก 8 Channel
- CLCD 14PIN พอร์ตสำหรับต่อ LCD (4 Bit Data)
- RTC #PCF8583P (Option)
- EEPROM ตั้งแต่เบอร์ #2432 ขึ้นไป (Option)
- I^2C IN/OUT เป็นIC ขยายพอร์ต I/O #PCF8574AP (Option)
- KBI/IO 10 Pin สำหรับต่อกับ Keyboard หรือ ใช้เป็น Input /Output Port
- Relay Onboard 5V 1ตัว (Option)
- MCRB02TTL ขั้วต่อ Macnetic Card Reader
- Mini Speaker/Buzzer
- I^2C BUS(EXPAND)
- PWM1 ขั้วต่อสำหรับใช้งาน Capture/Compare/PWM ตัวที่หนึ่ง
- PWM2 ขั้วต่อสำหรับใช้งาน Capture/Compare/PWM ตัวที่สอง

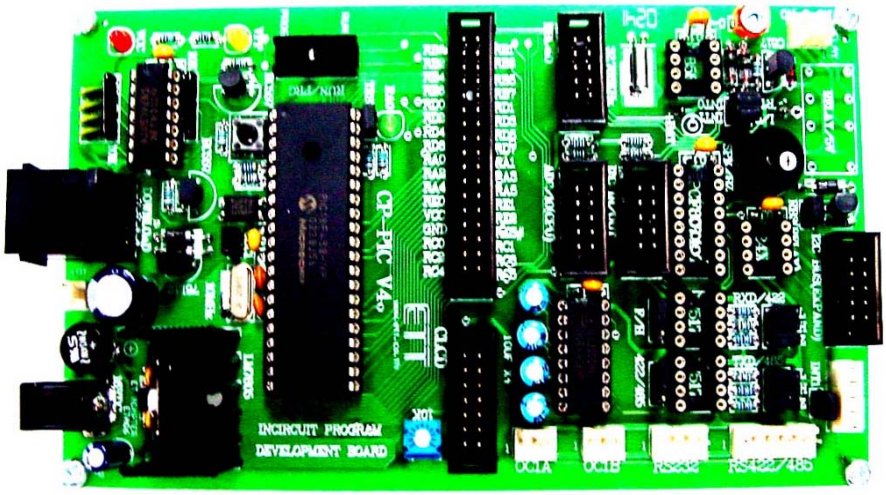
#หมายเหตุ Option คือ ส่วนที่ออกแบบไว้ให้เป็น Socket เปล่าๆหากต้องการใช้งานต้องหาซื้อเพิ่มเอง



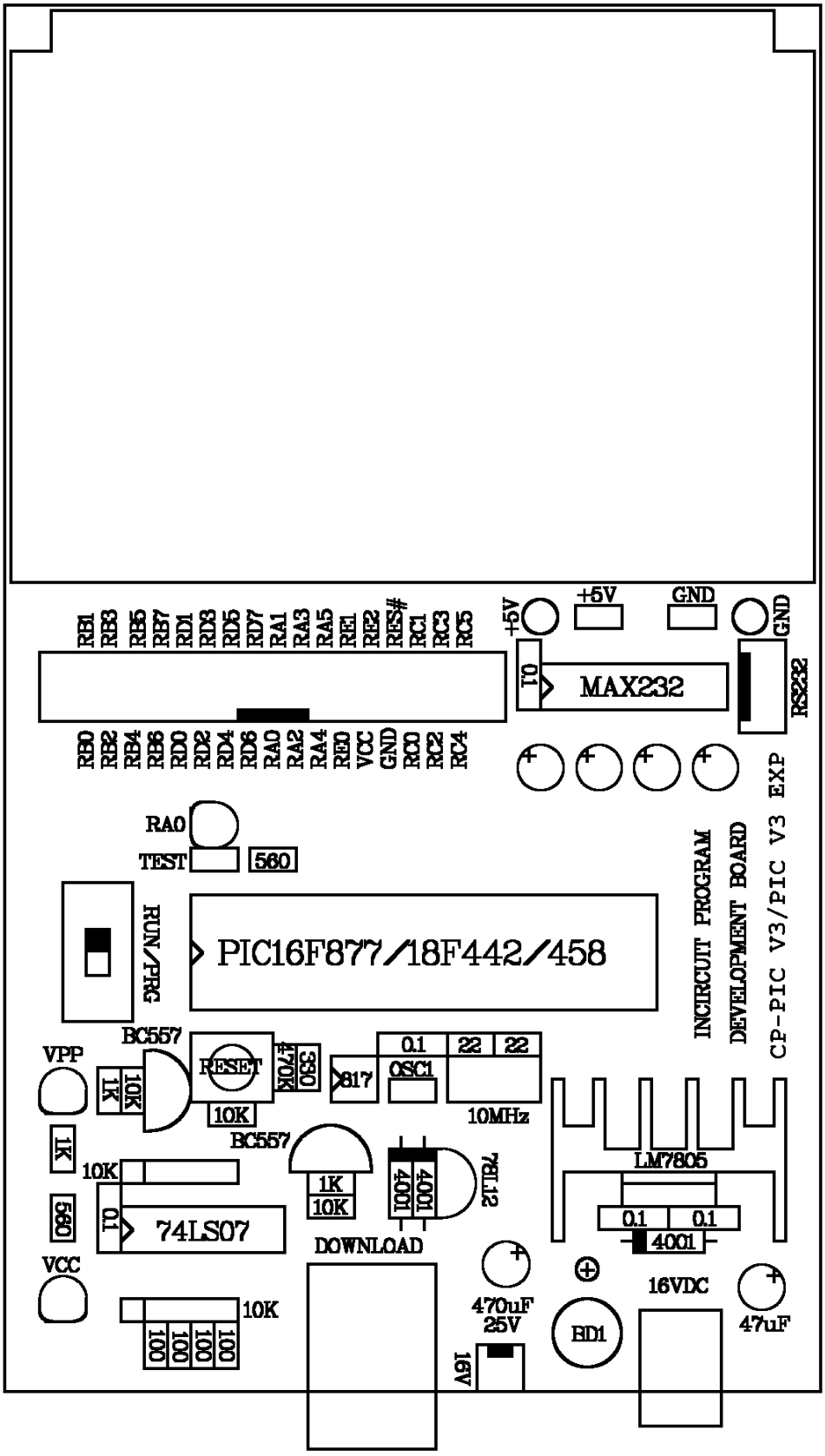
รูปแสดง ลักษณะของบอร์ด CP-PIC V3.0



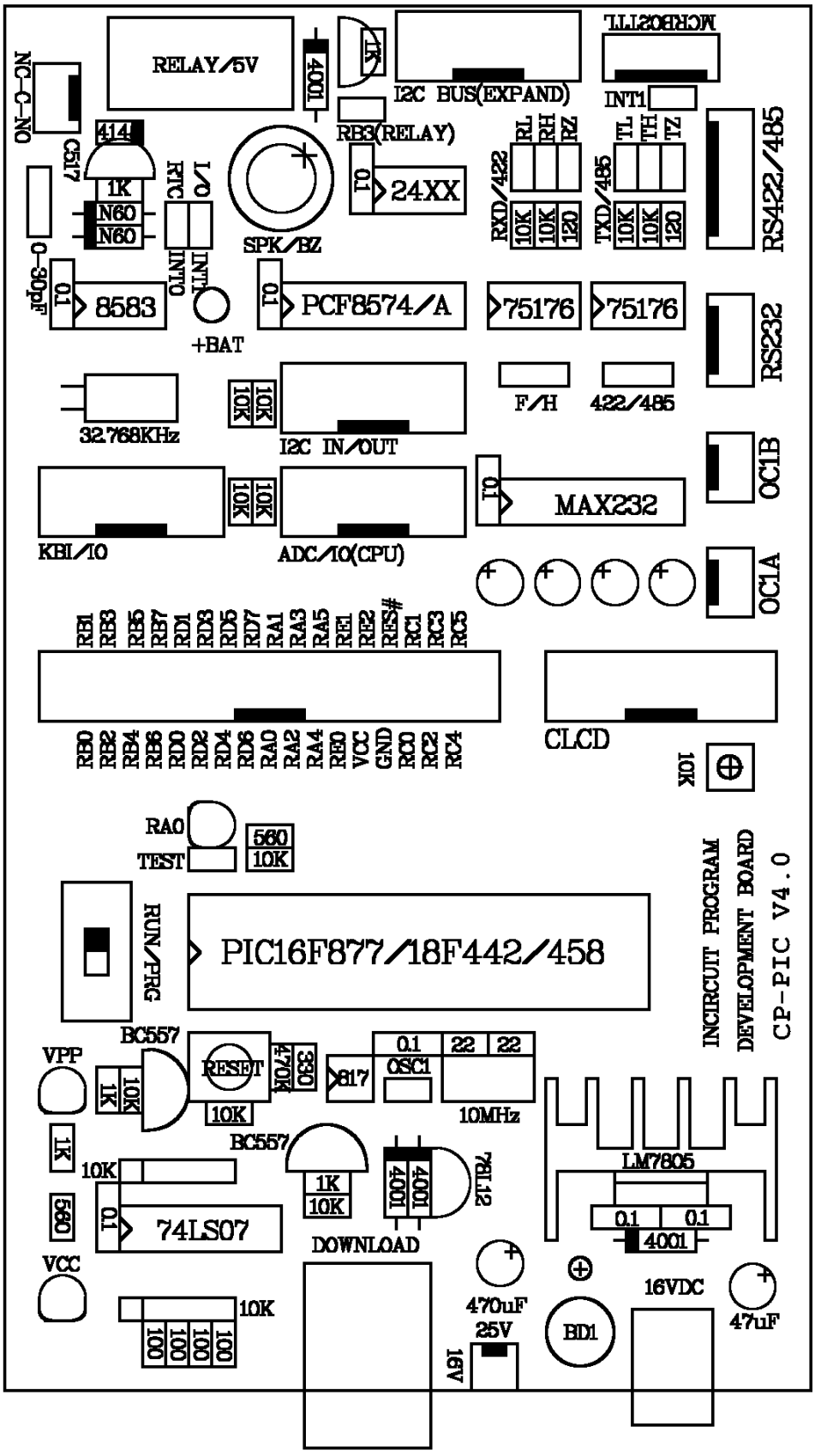
รูปแสดงลักษณะของบอร์ด CP-PIC V3.0 EXPANSION



รูปแสดงลักษณะของบอร์ด CP-PIC V4.0



รูปแสดง ลักษณะโครงสร้างของบอร์ด CP-PIC V3.0 และ V3.0 EXPANSION



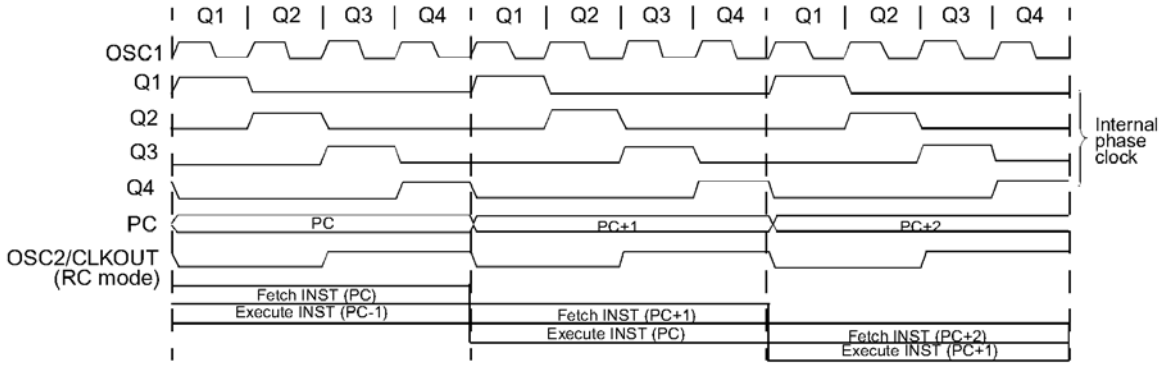
รูปแสดง โครงสร้างของบอร์ด CP-PIC V4.0

แหล่งจ่ายไฟ (POWER SUPPLY)

สำหรับแหล่งจ่ายไฟของบอร์ด CP-PIC ทั้ง V3.0 และ V4.0 นั้น จะสามารถต่อใช้งานได้ทั้งกับไฟกระแสตรงและกระแสสลับ เนื่องจากในบอร์ดได้จัดเตรียมวงจร RECTIFIER แบบ BRIDGE พร้อมวงจร FILTER และ REGULATOR ขนาด +5V ไว้ให้อย่างครบถ้วนอยู่แล้ว โดยผู้ใช้สามารถป้อนแรงดันไฟตรงหรือไฟสลับที่มีระดับแรงดันไม่ต่ำกว่า 13V เนื่องจากบอร์ดถูกออกแบบให้โปรแกรมข้อมูลด้วย ไฟสูง คือ 13 โวลท์ จึงต้องใช้แหล่งจ่ายไฟที่มากกว่า 13 โวลท์ อยู่ในช่วงประมาณ 13V ถึง 16V โดยสามารถเลือกต่อกับขั้ว CONNECTOR แบบ CPA ขนาด 2 ขา หรือจะต่อผ่านขั้ว CONNECTOR สำหรับ ADAPTER จ่ายไฟก็ได้เช่นกัน โดยการทำงานของแหล่งจ่ายไฟจะมีหลอดแสดงผล LED “VCC” สำหรับแสดงผลการทำงานให้ทราบด้วย

สัญญาณนาฬิกา CLOCK

ไมโครคอนโทรลเลอร์จะมีการทำงานตามจังหวะสัญญาณนาฬิกาของระบบ ซึ่งใน 1 Cycle (Clock Bus) ของCPU จะประกอบไปด้วย สัญญาณนาฬิกาจากภายนอกจำนวณ 4 cycle คือ Q1,Q2,Q3 และ Q4 ดังรูปด้านล่าง ฉะนั้นความถี่ที่ CPU ประมวลผลต่อคำสั่งจะเท่ากับ ความถี่จากคริสตอลภายนอก หารด้วย 4 หรือ อาจกล่าวได้ว่าความเร็วในการทำงานของ ไมโครคอนโทรลเลอร์ตระกูล PIC ต่อคำสั่งจะมีค่าเป็น ¼ เท่าของความถี่คริสตอลออสซิลเลเตอร์ภายนอก



รูปแสดงสัญญาณนาฬิกาของไมโครคอนโทรลเลอร์ PIC

ความถี่ของสัญญาณนาฬิกาที่จะป้อนให้กับ CPU เบอร์ 18F442 และ 18F458 นั้น สามารถเลือกกำหนดแหล่งกำเนิดสัญญาณนาฬิกาได้ 2 แบบ คือ

- ใช้ค่าความถี่จากวงจรกำเนิดความถี่แบบ Oscillator จากภายนอก โดยสามารถใช้ได้กับ Oscillator ที่มีค่าความถี่ระหว่าง 0-40MHz ในเบอร์ 18F442 กับ 18F458 ส่วน 16F877-20P จะอยู่ในช่วง 0-20MHz
- ใช้ค่าความถี่จากวงจร PLL(Phase-Lock-Loop) ที่บรรจุไว้ภายในตัว CPU แล้ว โดยวิธีการนี้จะต้องป้อนค่าความถี่ Crystal ขนาด 0-10MHz ให้กับขา OSC1 และ OSC2 ของ CPU ด้วย แล้วจึงทำการกำหนดค่า Configuration ในการโปรแกรมให้อยู่ในโหมดของ เฟสล็อกลูป (วิธีการ และ รายละเอียดจะกล่าวต่อไปในหัวข้อการใช้งาน EPICwin) ซึ่งเมื่อโปรแกรมการทำงานในโหมดนี้ CPU จะใช้วงจร เฟสล็อกลูปภายใน คูณสัญญาณนาฬิกาที่เข้ามาทางขา OSC1 และ OSC2 ด้วยสี่ เช่น ใช้คริสตอลออสซิลเลเตอร์ขนาดความถี่ 10 MHz เมื่อผ่านวงจรเฟสล็อกลูปภายใน สัญญาณที่ได้ออกมาจะมีความถี่ 40 MHz ซึ่ง

เป็นสี่เท่าของสัญญาณนาฬิกาที่เข้ามา คุณสมบัตินี้จะไม่มีอยู่ใน CPU PIC16F877 สามารถทำได้เฉพาะเบอร์ 18F442 และ 18F458 เท่านั้น

ซึ่งการทำงานของ CPU จะอาศัยสัญญาณนาฬิการะบบ หรือ BUS CLOCK เป็นจุดอ้างอิงการทำงานให้สัมพันธ์และสอดคล้องกับวงจรภายในอื่นๆ 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 0-40MHz (เฉพาะ 18F442 และ 18F458 ส่วน 16F877 จะอยู่ในช่วง 0-20MHz) หรือในกรณีที่จะใช้ค่าความถี่ของสัญญาณนาฬิกาจาก Crystal ภายนอกร่วมกับวงจร PLL นั้น ค่าคริสตอลที่นำมาต่อจะต้องมีค่าไม่เกิน 10MHz

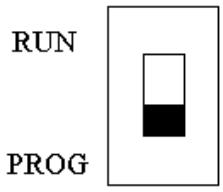
โหมดการทำงานของบอร์ด

การทำงานของบอร์ด CP-PIC V3.0&V4.0 นั้น สามารถกำหนดโหมดการทำงานของบอร์ดได้ 2 โหมดการทำงานด้วยกัน คือ โหมดการโปรแกรม (PROG) และโหมดการทำงานปรกติ (RUN)

การทำงานในโหมดการโปรแกรม (PROG)

ในโหมดนี้จะใช้สำหรับในกรณีที่ต้องการโปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์ ในบอร์ด CP-PIC V3.0&V4.0 นี้ ได้ออกแบบในส่วนของการโปรแกรมเป็นแบบ High Voltage Programing ใช้แรงดันในการโปรแกรม 13 โวลท์ ข้อดีของการโปรแกรมแบบนี้คือสามารถใช้งาน I/O Port ได้ครบทุกขา และในการออกแบบได้ใช้ SLIDE SWITCH เพื่อตัดต่อขาสัญญาณที่ใช้ในการโปรแกรม ดังนั้นเมื่อทำงานใน โหมดปกติขาสัญญาณต่างๆ ก็จะถูกแยกออกจากวงจรส่วนของการโปรแกรม ดังนั้นจึงสามารถใช้งานขาสัญญาณต่างๆ ได้ครบทั้งหมด

การเข้าสู่โหมดของการโปรแกรมทำได้โดยการเลือกตำแหน่ง SLIDE SWITCH (PROG/RUN) มาที่ตำแหน่ง PROG ส่วนซอฟ์แวร์ ที่ใช้ในการดาวน์โหลดโปรแกรมจะใช้ โปรแกรม EPICwin ซึ่งในรายล่ะเอียดการใช้งานจะกล่าวภายหลัง ในหัวข้อ การใช้งาน EPICwin



สวิตช์เลือกโหมด RUN/PROG

การทำงานใน USER MODE หรือ RUN MODE

การทำงานในโหมดนี้ คือ การทำให้ CPU กระทำตามคำสั่งต่างๆ ตามโปรแกรมที่เราได้ออกแบบไว้ซึ่งการเข้าสู่โหมดนี้ ทำได้โดยการเลือกตำแหน่ง SLIDE SWITCH (PROG/RUN) มาที่ตำแหน่ง RUN สวิตช์ Slide ก็จะทำการแยกขาสัญญาณต่างๆ ออกจากวงจรในส่วนของการโปรแกรม ฉนั้นในการใช้งาน I/O Port จึงสามารถนำมาใช้งานได้ครบทั้งหมด

การจัดสรร I/O ของบอร์ด CP-PIC V3.0 & V4.0

บอร์ด CP-PIC V3.0&V4.0 จะใช้ได้กับ CPU เบอร์ 16F877,18F442 และ 18F458 ทั้งนี้ขึ้นอยู่กับการเลือกใช้งาน โดยตัว CPU เหล่านี้จะมีขาสัญญาณที่สามารถนำมาใช้งานเป็น I/O Port ซึ่งในเบอร์ 18F442 และ 18F458 จะมี I/O รวมทั้งสิ้น 34 เส้น ส่วน 16F877 มี 33 เส้นประกอบด้วย

- RA0-RA6 จำนวน 7 เส้นสัญญาณ (ส่วน 16F877 จะมีเพียง RA0-RA5 คือ 6 เส้นเท่านั้น)
- RB0-RB7 จำนวน 8 เส้นสัญญาณ
- RC0-RC7 จำนวน 8 เส้นสัญญาณ
- RD0-RD7 จำนวน 8 เส้นสัญญาณ
- RE0-RE2 จำนวน 3 เส้นสัญญาณ

โดยการออกแบบวงจรของบอร์ด CP-PIC V3.0&V4.0 นั้น ได้พยายามออกแบบวงจรโดยวางโครงสร้างของบอร์ด ให้มีความอ่อนตัวในการใช้งานมากที่สุด เพื่อให้ผู้ใช้งานสามารถนำบอร์ดไปประยุกต์ใช้งานในหลายๆลักษณะได้โดยไม่ต้องดัดแปลงโครงสร้างวงจรของบอร์ดไปจากเดิมมากนัก ดังนั้นจึงได้มีการจัดสรรขาสัญญาณ Port I/O ของ CPU ให้สามารถทำงานได้หลายหน้าที่ โดยให้ผู้ใช้สามารถเลือกได้ตามต้องการ ซึ่งหน้าที่การใช้งาน Port I/O ของ CPU ในบอร์ด CP-PIC V3.0&V4.0 นั้น สามารถสรุปได้ดังต่อไปนี้

RA0-RA3 และ RA5 ขาสัญญาณเหล่านี้นอกจากจะใช้งานเป็น I/O ปกติได้แล้วยังทำหน้าที่เป็นขาอินพุตของสัญญาณอนาล็อก (AN0-AN4) อีกด้วยดังนั้นเราจึงต่อสายสัญญาณเหล่านี้เข้ากับขั้วต่อ ADC/IO(CPU) เพื่อให้สะดวกต่อการนำไปใช้งาน

RA4 จะใช้งานในส่วนของแอลซีดี ซึ่งจะต่อเข้าที่ขา 6 ของคอนเนคเตอร์ CLCD โดยทำหน้าที่เป็นขา Enable ให้กับแอลซีดี

RA6/OSC2/CLKO เป็นขาสัญญาณที่ทำหน้าที่ในหลายส่วน คือ เป็นขา OSC2 และ CLKO จะนำมาใช้เป็นขาสัญญาณ I/O ได้ก็ต่อเมื่อเราใช้คริสตอลออสซิลเลเตอร์แบบที่เป็นโมดูลสำเร็จสามารถต่อเข้ากับขา OSC1/CLKIN ได้เลยโดยที่ไม่ต้องต่อกับขา RA6/OSC2 ทำให้ ขา RA6 ว่างและนำไปใช้เป็น I/O ได้ แต่ในบอร์ดที่เราออกแบบจะใช้งานขา RA6/OSC2 ร่วมกับ OSC1 ในการรับสัญญาณนาฬิกาจากภายนอกดังนั้น ขา RA6 นี้จึงไม่สามารถต่อออกไปใช้งานได้

RB0-RB7 สำหรับขาสัญญาณเหล่านี้จะสามารถใช้งานเป็น I/O ได้ปกติ แต่จะมีคุณสมบัติพิเศษคือจะมีวงจรพูลอัพ (Pull-Up) ภายในและยังเป็นแหล่งกำเนิดสัญญาณอินเตอรัพท์ต่างๆ ดังนี้

- RB0/INT0 เป็นขาสัญญาณอินเตอร์รัพท์ภายนอก 0
- RB1/INT1 เป็นขาสัญญาณอินเตอร์รัพท์ภายนอก 1
- RB2/INT2 เป็นขาสัญญาณอินเตอร์รัพท์ภายนอก 2
- RB3/INT3 เป็นขาสัญญาณอินเตอร์รัพท์ภายนอก 3 (เฉพาะเบอร์ 18F442)
- RB4-RB7 เป็นขาที่สามารถกำเนิดสัญญาณอินเตอร์รัพท์ได้หากมีการเปลี่ยนแปลงในขาสัญญาณดังกล่าวและมีการ Enable อินเตอร์รัพท์ประเภทนี้ไว้ จึงเหมาะกับการนำไปใช้งานในส่วนของ สวิตช์คีย์บอร์ด เนื่องจากมีทั้ง อินเตอร์รัพท์และวงจรพูลอัพในตัว

จากคุณสมบัติดังกล่าวเราจึงจัดสรรค์การใช้งานดังนี้

RB0/INT0 จะต่อกับขาสัญญาณอินเตอร์รัพท์ของ RTC เบอร์ PCF8583 โดยจะต่อผ่านจัมเปอร์จึงสามารถเลือกที่จะต่อหรือไม่ก็ได้ สามารถเลือกได้โดยการ Shot หรือ Open จัมเปอร์ INT0

RB1/INT1 จะต่อเข้ากับขาสัญญาณอินเตอร์รัพท์ของ PCF8574A และขาสัญญาณอินเตอร์รัพท์ของ Macnetic Card Reader (MCRB02TTL) ทั้งสองส่วนนี้จะต่อผ่านจัมเปอร์ (INT1) ทำให้สามารถเลือกที่จะต่อหรือไม่ต่อก็ได้ การใช้งานจะต้องเลือกใช้งานอย่างใดอย่างหนึ่งเท่านั้น ไม่สามารถใช้งานทั้งสองตัวพร้อมกันได้

RB2 เป็นขาสัญญาณที่ต่อเข้ากับ SPK/BUZZER เพื่อควมคุมการทำงานของ Speaker หรือ Buzzer

RB3 เป็นขาสัญญาณที่ต่อกับวงจรที่ควบคุมการทำงานของรีเลย์(Relay) โดยจะต่อผ่านจัมเปอร์ดังนั้นจึงสามารถเลือกใช้งานหรือไม่ก็ได้โดยการ Shot หรือ Open จัมเปอร์ RB3(RELAY)

RB4-RB7 จะต่อเข้ากับขั้วต่อ KBI/IO สามารถนำไปต่อกับ คีย์บอร์ดประเภท Matrix แบบ 4x4,4x3 หรือ จะใช้เป็น I/O ธรรมดาก็ได้ ในขา RB6 และ RB7 นั้นนอกจากจะต่อกับขั้วต่อ KBI/IO แล้วยังต่อกับสวิตช์ PROG/RUN เพื่อใช้เป็นสัญญาณในการโปรแกรมเมื่ออยู่ในโหมดของการโปรแกรม แต่เมื่ออยู่ในโหมด RUN สามารถนำมาใช้งานเป็น I/O ได้ปกติ

RC0 ขาสัญญาณนี้จะต่อเข้ากับขั้วต่อ แอลซีดี (CLCD) โดยจะต่อเข้าที่ขา 4 ของคอนเนคเตอร์ เพื่อทำหน้าที่เป็นขาสัญญาณ RS เพื่อควบคุมการทำงานของ LCD

RC1 เป็นขาสัญญาณที่ต่อเข้ากับขั้วต่อ OC1B เพื่อใช้งานในส่วนของ ขาสัญญาณอินพุตของ Timer 1 หรือใช้เป็นขาสัญญาณในส่วนของ Capture2 input /Compare2 Output/PWM2

RC2 เป็นขาสัญญาณที่ต่อเข้ากับขั้วต่อ OC1A เพื่อใช้เป็นขาสัญญาณในส่วนของ Capture1 input /Compare1 Output/PWM1

RC3 สำหรับขาสัญญาณ RC3 จะใช้ทำหน้าที่เป็นขาสัญญาณ SCL ในการติดต่อกับอุปกรณ์ I^2C Bus และจะต่อเข้ากับ ขั้วต่อ I^2C EXPAND เพื่อขยายพอร์ต I^2C BUS

RC4 สำหรับขาสัญญาณ RC4 จะใช้ทำหน้าที่เป็นขาสัญญาณ SDA ในการติดต่อกับอุปกรณ์ I^2C Bus และจะต่อเข้ากับ ขั้วต่อ I^2C EXPAND เพื่อขยายพอร์ต I^2C BUS

RC5 จะใช้เป็นสัญญาณควบคุมการรับส่งข้อมูลในการใช้งาน RS485 โดยจะควบคุมการทำงานของอุปกรณ์ที่เป็น Line Driver ก็คือ IC 75176

RC6 เป็นขาสัญญาณที่ทำหน้าที่ในการส่งข้อมูล (Tx) ในโหมดการสื่อสารอนุกรม RS232,RS422 และ RS485 โดยจะต่อเข้ากับ IC ที่เป็น Line Driver คือ Max 232 และ 75176

RC7 เป็นขาสัญญาณที่ทำหน้าที่ในการรับข้อมูล (Rx) ในโหมดการสื่อสารอนุกรม RS232,RS422 และ RS485 โดยจะต่อเข้ากับ IC ที่เป็น Line Driver คือ Max 232 และ 75176

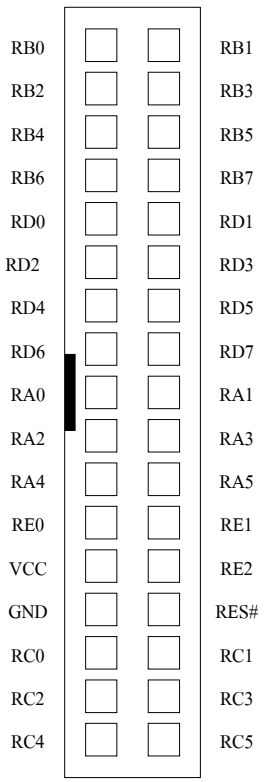
RD0-RD3 สำหรับขาสัญญาณเหล่านี้จะต่อเข้ากับขั้วต่อ KBI/IO เพื่อใช้งานสำหรับการต่อ คีย์สวิตช์ 4x4 หรือ 4x3 ซึ่งเมื่อใช้งานเป็นคีย์บอร์ดดังกล่าวจะทำงานร่วมกับ พอร์ต RB4-RB7 หรือจะใช้งานเป็น I/O ก็ได้

RD4-RD7 ขาสัญญาณเหล่านี้จะทำหน้าที่เป็นขาสัญญาณ Data ที่ใช้ติดต่อกับ LCD โดยจะถูกต่อไปที่คอนเน็คเตอร์ CLCD ซึ่งขั้วต่อ LCD ที่ได้ออกแบบนี้จะเป็นแบบ 4 Bit Data ฉนั้นในการรับส่งข้อมูลจะทำผ่านสายสัญญาณทั้ง 4 เส้น คือ RD4-RD7

RE0-RE2 ขาสัญญาณเหล่านี้สามารถใช้งานเป็น I/O ได้ตามปกติ แต่จะมีคุณสมบัติพิเศษคือขาสัญญาณดังกล่าวจะทำหน้าที่เป็นขาอินพุตอนาลอก (AN5-AN7) เมื่ออยู่ในโหมดของ Analog to Digital โดยเราจะนำไปต่อกับขั้วต่อ ADC/IO(CPU) ทำให้สามารถต่อออกไปใช้งานได้สะดวก

การใช้งานขั้วต่อ 34PIN (72IOZ80)

สำหรับขั้วต่อ Connector ขนาด 34 PIN ของบอร์ด CP-PIC V3.0&V4.0 นั้น จะมีอยู่ด้วยกัน 2 แบบ คือ ถ้าเป็นบอร์ด CP-PIC V3.0 และ CP-PIC V4.0 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-PIC V3.0 EXPANSION นั้น ขั้วต่อ 34 PIN จะเป็นแบบ IDE ตัวเมีย เพื่อให้ผู้ใช้สามารถใช้สาย Jumper ต่อสัญญาณต่างๆจากขั้วต่อนี้ไปยังแผงทดลอง Photo Board เพื่อต่อร่วมกับวงจรต่างๆได้โดยง่าย โดยลักษณะการจัดเรียงสัญญาณเป็นดังนี้

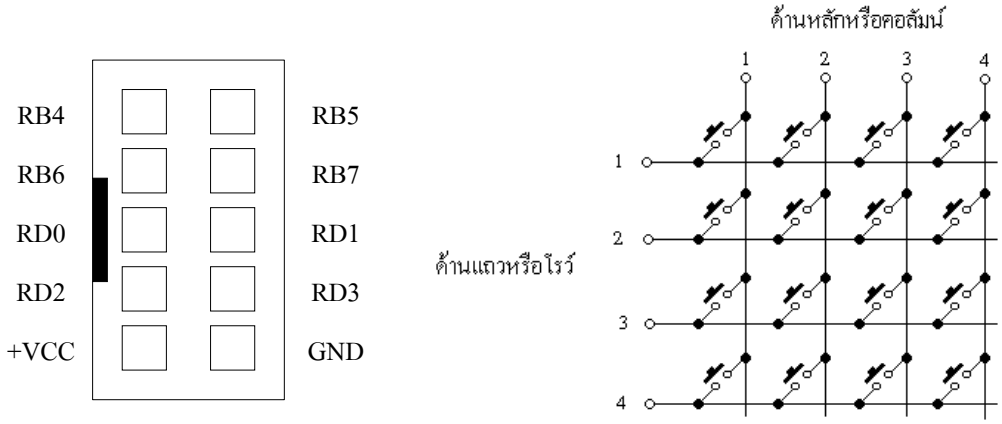


รูปแสดงลักษณะของการจัดเรียงสัญญาณของขั้ว 34PIN

การใช้งานขั้วต่อ KBI/IO

พอร์ต KBI/IO ถูกจัดไว้ที่ขั้ว Connector ขนาด 10 PIN แบบ IDE ซึ่งขั้วต่อนี้จะมีอยู่เฉพาะ ในบอร์ด รุ่น CP-PIC V4.0 เท่านั้น โดยขั้วต่อนี้จะเชื่อมต่อสัญญาณมาจาก PORTB(RB4-RB7) และ PORTD(RD0-RD3) ของ CPU ทั้งหมด 8 เส้น ซึ่งเหมาะกับการนำไปประยุกต์ใช้งานในส่วนของการเชื่อมต่อกับวงจรคีย์บอร์ดแบบ Matrix ซึ่งสามารถจะใช้ได้กับคีย์บอร์ดแบบ Matrix ขนาด 4x3 หรือ 4x4 ก็ได้

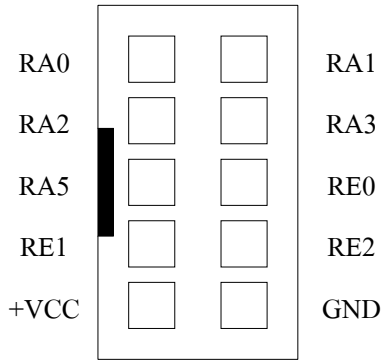
สำหรับในกรณีที่ไม่มีความจำเป็นต้องใช้งานคีย์บอร์ดแล้ว ขั้วต่อ KBI/IO นี้ก็ยังสามารถนำไปต่อใช้งานเป็น Input หรือ Output ทั่วๆไปได้อีกด้วย



รูปแสดงลักษณะของการจัดเรียงสัญญาณของขั้ว KBI/IO

การใช้งานขั้วต่อ ADC/IO(CPU)

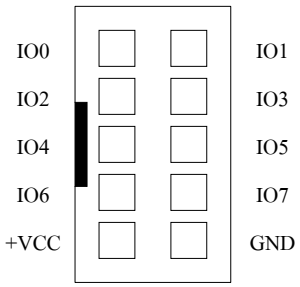
พอร์ต ADC/IO(CPU) นี้จะถูกเชื่อมต่อออกมารอไว้ยังขั้ว Connector ขนาด 10 PIN แบบ IDE ซึ่งขั้วต่อนี้จะมีอยู่เฉพาะบอร์ดในรุ่น CP-PIC V4.0 เท่านั้น โดยขั้วต่อนี้ สามารถนำไปต่อใช้งานเป็นอินพุตของสัญญาณอนาลอกทั้ง 8 เชนแนล (AN0-AN7) ซึ่งหากไม่ใช้งานในส่วนนี้สามารถใช้งานเป็นอินพุตเอาต์พุตพอร์ตได้ตามปกติ ซึ่งขาสัญญาณต่างๆ ที่นำมาต่อจะเป็นดังรูปด้านล่าง



รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้ว ADC/IO(CPU)

การใช้งานขั้วต่อ I²C IN/OUT

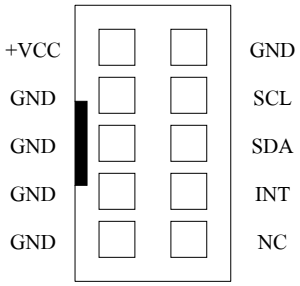
พอร์ต I²C IN/OUT นี้จะถูกเชื่อมต่อออกมารอไว้ยังขั้ว Connector ขนาด 10 PIN แบบ IDE ซึ่งขั้วต่อนี้จะมีอยู่เฉพาะบอร์ดในรุ่น CP-PIC V4.0 เท่านั้น โดยขั้วต่อ I²C IN/OUT นี้ จะเชื่อมต่อมาจากขาสัญญาณ I/O Port ของ PCF8574/A ซึ่งสามารถโปรแกรมหน้าที่การใช้งาน ให้เป็น Input หรือ Output ก็ได้ตามต้องการจากโปรแกรม แต่ต้องกำหนดหน้าที่ให้เป็น Input หรือ Output อย่างใดอย่างหนึ่งเท่านั้น ไม่สามารถใช้งานทั้งสองหน้าที่พร้อมๆกันได้ และในการกำหนดหน้าที่ให้เป็น Input หรือ Output ก็ต้องกำหนดให้เหมือนกันทั้ง 8 บิต ด้วย โดยลักษณะของขาสัญญาณที่จัดเรียงไว้ที่ขั้ว I²C IN/OUT จะเป็นดังรูป



รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้ว I²C IN/OUT

การใช้งานขั้วต่อ I²C BUS EXPAND

พอร์ต I²C BUS EXPAND นี้จะถูกเชื่อมต่อออกมารอไว้ยังขั้ว Connector ขนาด 10 PIN แบบ IDE ซึ่งขั้วต่อนี้จะมีอยู่เฉพาะบอร์ดในรุ่น CP-PIC V4.0 เท่านั้น โดยขั้วต่อ I²C BUS EXPAND นี้ จะใช้สำหรับทำการขยายหรือเพิ่มเติมจำนวนอุปกรณ์ที่ใช้การติดต่อสื่อสารแบบ I²C ให้กับบอร์ด

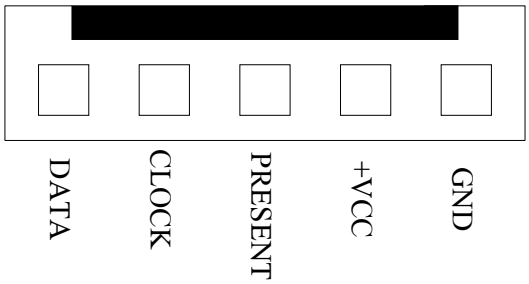


รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้ว I²C BUS EXPAND

การใช้งานเครื่องอ่านบัตรแถบแม่เหล็ก (MAGNETIC-CARD READER)

สำหรับบอร์ด CP-PIC V4.0 นั้น จะออกแบบให้สามารถเชื่อมต่อกับเครื่องอ่านบัตรแถบแม่เหล็กรุ่น “MCR-B02TTL” ได้โดยตรง โดยไม่ต้องดัดแปลงวงจรใดๆทั้งสิ้น โดยในบอร์ดจะจัดเตรียมขั้วแบบ CPA ขนาด 5 PIN ไว้รองรับอยู่แล้ว ผู้ใช้สามารถนำขั้วต่อของเครื่องอ่านบัตรแถบแม่เหล็ก รุ่น “MCR-B02TTL” ของบริษัท อีทีที ต่อเข้าไปได้ทันที สำหรับในการเขียนโปรแกรมเพื่อเชื่อมต่อระหว่างบอร์ด CP-PIC V4.0 กับเครื่องอ่านบัตรแถบแม่เหล็กนั้น จะสามารถทำได้ 2 แบบ คือ ใช้วิธีการวนรอบตรวจสอบสัญญาณจากเครื่องอ่านบัตรแถบแม่เหล็กเอง ซึ่งวิธีการนี้จะใช้สัญญาณ

จาก CPU เพียง 2 เส้นสัญญาณคือ RC1 ทำหน้าที่เป็น DATA และ RC2 ทำหน้าที่เป็น CLOCK โดยต้องกำหนดคุณสมบัติของสัญญาณทั้ง 2 เส้นให้มีทิศทางเป็น Input ส่วนอีกวิธีหนึ่งคือการใช้วิธีการ Interrupt ซึ่งสามารถทำได้โดยการ SHORT JUMPER "INT1" ที่อยู่ใกล้ๆกับขั้วต่อเครื่องอ่านบัตรแถบแม่เหล็ก ซึ่งการ SHORT JUMPER จะเป็นการเชื่อมต่อสัญญาณอินเตอร์รัพท์ของ CPU เข้ากับ สัญญาณ PRESENT ของเครื่องอ่านบัตรแถบแม่เหล็ก ดังนั้นเมื่อมีการนำบัตรแถบแม่เหล็กไปรูดผ่านเครื่องอ่านบัตรแถบแม่เหล็กก็จะมีการส่งสัญญาณ Interrupt ออกมายัง CPU ซึ่งผู้ใช้ก็เพียงแต่เขียนโปรแกรมสำหรับบริการการ Interrupt ของ INT1 ไว้ก็สามารถใช้งานได้แล้ว



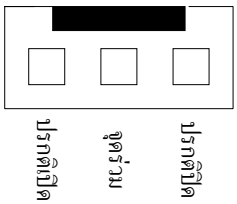
รูปแสดง ลักษณะของการจัดเรียงสัญญาณของขั้วต่อเครื่องอ่านบัตรแถบแม่เหล็ก MCR-B02TTL

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

การใช้งาน OUTPUT RELAY

ภายในบอร์ด CP-PIC V4.0 นั้น จะออกแบบวงจรควบคุม RELAY ไว้ให้ผู้ใช้งานสามารถนำไปประยุกต์ใช้งานทั่วไปได้ด้วยจำนวน1ชุดโดยวงจรRELAYดังกล่าวสามารถใช้งานได้ ทั้งหน้าสัมผัสแบบปกติเปิด (Normal Open : NO) และแบบหน้าสัมผัสปกติปิด (Normal Close : NC)

สำหรับสัญญาณ Output ในการควบคุมการทำงานของ REALY นั้น จะมาจากขา RB3 ของ CPU ซึ่งเมื่อต้องการใช้งาน RELAY จะต้องทำการ SHORT JUMPER RB3(RELAY) ไว้ด้วยเพื่อเชื่อมต่อสัญญาณRB3 มาทำการควบคุมการทำงานของ RELAY และต้องแน่ใจว่าไม่ได้ต่อสัญญาณ RB3 จากขั้วต่ออื่นๆออกไปใช้งานกับอุปกรณ์ใดๆนอกเหนือจาก RELAY เนื่องจากสัญญาณ RB3 นั้น นอกจากจะต่อมาใช้ควบคุมการทำงานของ RELAY แล้วยังต่อไปยังขั้วต่อ 34PIN อีกด้วย โดยการทำงานของ RELAY จะถูกควบคุมการทำงานจากขาสัญญาณ RB3 โดยผู้ใช้ต้องกำหนดคุณสมบัติของสัญญาณ RB3 ให้ทำหน้าที่เป็น OUTPUT ไว้ด้วย ซึ่งเมื่อขาสัญญาณ RB3 มีสภาวะเป็น OUTPUT และให้สถานะเป็น "1" จะทำให้ RELAY ทำงาน แต่ถ้าสถานะของสัญญาณ RB3 มีค่าเป็น "0" จะทำให้ RELAY หยุดทำงาน



รูปแสดง ลักษณะขั้วต่อสัญญาณจากหน้าสัมผัสของ RELAY

*****หมายเหตุ***** เนื่องจากสัญญาณ RB3 ที่นำมาใช้ควบคุมการทำงานของ RELAY จะเป็นสัญญาณเส้นเดียวกับ RB3 ที่ต่อไว้ยังขั้ว 34PIN ดังนั้น เมื่อต้องการใช้งาน RELAY โดยการควบคุมจาก RB3 แล้ว ต้องแน่ใจว่าไม่ได้ต่อใช้งานสัญญาณ RB3 ในจุดอื่นๆอีก แต่ถ้าหากมีความจำเป็นต้องใช้งาน RB3 พร้อมกับการใช้งาน RELAY ด้วยในเวลาเดียวกัน ก็อาจดัดแปลงวงจรได้โดยการ OPEN JUMPER RB3(RELAY) ออก แล้วใช้วิธีการเชื่อมสายสัญญาณเส้นอื่นๆจาก PORT I/Oของ CPU ที่ไม่ได้ใช้งานมาเข้ากับวงจรควบคุม RELAY แทนก็ได้เช่นเดียวกัน โดยให้เชื่อมต่อสายสัญญาณที่ต้องการไปยัง Jumper RB3(RELAY) ด้านที่ต่อกับตัวต้านทานค่า 1KOhm แต่การดัดแปลงวิธีนี้ควรถอดตัว JUMPER RB3(RELAY) ออกจากบอร์ดเสียก่อน เพื่อจะได้ไม่หลงลืมทำการ SHORT JUMPER นี้ซ้ำอีกในภายหลัง เนื่องจากจะเป็นการ SHORT สัญญาณ RB3 เข้ากับสัญญาณเส้นใหม่ที่บัดกรีมายังวงจร RELAY นี้อีก

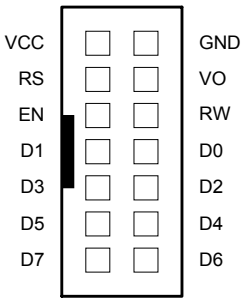
การใช้งานลำโพงขนาดเล็ก หรือ BUZZER

ภายในบอร์ด CP-PIC 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 จะมาจากขา RB2 ของ CPU ซึ่งการควบคุมการทำงานของลำโพง หรือ BUZZER ผู้ใช้ต้องทำการกำหนดคุณสมบัติของสัญญาณ RB2 ให้ทำหน้าที่เป็น OUTPUT ไว้ด้วย ซึ่งเมื่อขาสัญญาณ RB2 มีสภาวะเป็น OUTPUT และให้สถานะเป็น “1” จะทำให้ ลำโพง หรือ BUZZER ทำงาน แต่ถ้าสถานะของสัญญาณ RB2 มีค่าเป็น “0” จะทำให้ ลำโพง หรือ BUZZER หยุดทำงาน

การใช้งานจอแสดงผลแบบ LCD (Dot-Matrix Character LCD)

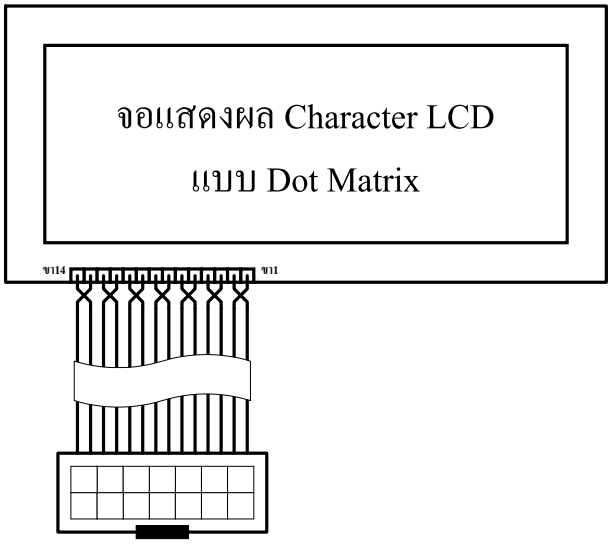
บอร์ด CP-PIC V4.0 สามารถใช้เชื่อมต่อกับจอแสดงผล LCD แบบ Dot-Matrix โดยเชื่อมต่อผ่านทาง Connector ขนาด 14 PIN และใช้ตัวต้านทานปรับค่าได้แบบเกือกม้าขนาด 10K สำหรับปรับระดับความสว่างของหน้าจอ LCD โดยวงจรในการเชื่อมต่อ LCD ของบอร์ดนี้จะออกแบบวงจรให้ใช้วิธีการควบคุมการทำงานแบบ "DATA 4-BIT"



รูปแสดง ขาสัญญาณของขั้วต่อ CLCD

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

- แบบที่เป็นขั้วต่อแบบแถวเดี่ยว ขนาด 14PIN (HEADER 14X1) โดยการต่อสายของ LCD แบบนี้ จะใช้สายแพร์ขนาด 14 เส้น แบบเข้าหัว CONNECTOR ไว้ด้านเดียว สำหรับเสียบกับขั้วต่อ CLCD ภายในบอร์ด CP-PIC 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 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 เข้ากับขา 2 ของ LCD ส่วนขา (K) ก็ให้ต่อเข้ากับขา 1 ของ LCD ก็ได้เช่นกัน

การเชื่อมต่อกับอุปกรณ์ I^2C BUS

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

- I^2C RTC เบอร์ PCF8583 ของ PHILIPS
- EEPROM ในตระกูล 24XX ซึ่งสามารถเลือกใช้ได้หลายเบอร์หลายผู้ผลิต ขึ้นอยู่กับขนาดความจุของหน่วยความจำที่ต้องการจะใช้ ซึ่งในบอร์ด CP-PIC 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 RC3 เป็นขาสัญญาณ SCL และใช้สัญญาณ RC4 เป็นสัญญาณ SDA ในการควบคุมบัส ซึ่ง CPU จะทำหน้าที่เป็นตัวแม่ในการควบคุมบัส นอกจากนี้แล้วยังสามารถขยายอุปกรณ์จำพวก I^2C นี้ได้อีก แต่ต้องเป็นอุปกรณ์ที่มีรหัสควบคุม Control Word ไม่ซ้ำกันกับอุปกรณ์ที่มีอยู่แล้วภายในบอร์ดด้วยโดยอาจเชื่อมต่อผ่านทางขั้วต่อ “I^2C EXPANSION” ที่บอร์ดจัดเตรียมไว้ให้แล้วก็ได้

การใช้งาน Interrupt ของอุปกรณ์ I^2C

สำหรับสัญญาณ Interrupt จากอุปกรณ์ I^2C นั้น เนื่องจาก CPU มีสัญญาณ Input ที่ใช้สำหรับรับการ Interrupt จากภายนอก 2 สัญญาณ คือ INT0 และ INT1 ดังนั้น เราจึงได้ออกแบบการต่อใช้งานสัญญาณ Interrupt ดังนี้ คือ

- ต่อสัญญาณการ Interrupt INT1ให้กับเครื่องอ่านบัตรแถบแม่เหล็ก
- ต่อสัญญาณการ Interrupt INT0ให้กับ RTC PCF8583
- ต่อสัญญาณการ Interrupt INT1ให้กับ PCF8574/A

จะเห็นได้ว่าสัญญาณ Interrupt INT1 จะถูกต่อใช้งานถึงสองส่วนคือ เชื่อมต่อกับสัญญาณ Interrupt จากเครื่องอ่านแถบแม่เหล็ก และ ต่อกับสัญญาณ Interrupt ของ PCF8574/A ดังนั้นในการใช้งานจะต้องเลือกใช้งานอย่างใดอย่างหนึ่งเท่านั้น ซึ่งสามารถทำได้โดยการกำหนดการทำงานให้กับ จัมเปอร์ INT1 เช่น หากใช้งาน Interrupt ของ PCF8574/A ก็ให้ Short Jumper INT1 ของ PCF8574/A และ Open Jumper INT1 ของเครื่องอ่านบัตรแม่เหล็ก MCRB02TTL

แอดเดรสของอุปกรณ์ I^2C

เนื่องจากคุณสมบัติของ BUS แบบ I^2C นั้น สามารถเชื่อมต่ออุปกรณ์ต่างๆที่ใช้วิธีการสื่อสารแบบ I^2C ได้มากมากหลายตัวภายในบัสเดียวกันได้ เพียงแต่มีข้อแม้ว่า อุปกรณ์ที่จะนำมาต่อร่วมกันภายในบัสเดียวกันนั้น จะต้องมีรหัสตำแหน่งในการติดต่อสื่อสาร (Control Byte) ที่ไม่ซ้ำกัน ซึ่งอุปกรณ์บางตัวนั้น ผู้ผลิตได้มีการออกแบบให้สามารถกำหนดค่ารหัสตำแหน่ง Control Byte ได้มากกว่า 1 ค่าเพื่อให้สามารถเชื่อมต่ออุปกรณ์ประเภทเดียวกันร่วมกันภายในบัสเดียวกันได้มากกว่า 1 ตัว โดยใช้วิธีการกำหนดค่าโลจิกให้กับขาสัญญาณสำหรับใช้ระบุตำแหน่ง (Address) ของอุปกรณ์เบอร์นั้นๆได้เอง เช่น I/O Port เบอร์ PCF8574 นั้น สามารถต่อร่วมกันภายในบัสเดียวกันได้มากถึง 8 ตัว และยังสามารถเชื่อมต่ออุปกรณ์ I/O Port ที่มีคุณสมบัติเหมือนกันแต่มีรหัสตำแหน่งที่แตกต่างกันคือ PCF8574A เพิ่มเติมได้อีก 8 ตัว ซึ่งจะเห็นได้ว่าอุปกรณ์ I/O Port นั้นสามารถทำการเพิ่มเติมเข้าไปให้ระบบบัสเดียวกันได้มากถึง 16 ตัว และในทำนองเดียวกัน หน่วยความจำ E^2PROM เบอร์ 24LC256 ก็สามารถต่อร่วมกันภายในบัสเดียวกันได้มากถึง 8 ตัว จากตัวอย่างข้างต้นจะเห็นได้ว่าภายในบัสเดียวกันของ I^2C นั้น อุปกรณ์เพียง 2 ประเภท คือ I/O และ E^2PROM สามารถต่อร่วมกันภายในบัสเดียวกันได้มากถึง 24 ตัว คือ I/O PCF8574 8 ตัว ,PCF8574A 8ตัว และ 24LC256 อีก 8 ตัว และยังสามารถนำอุปกรณ์ I^2C อื่นๆที่มีรหัสตำแหน่งของ Control Byte ไม่ซ้ำกันมาต่อเพิ่มเติมได้อีก แต่อย่างไรก็ตามถึงแม้ว่าอุปกรณ์แบบ I^2C นี้ยอมให้มีการเชื่อมต่อร่วมกันภายในบัสเดียวกันได้หลายตัวภายในระบบบัสเดียวกันก็ตาม แต่ในทางปฏิบัติแล้วอาจเกิดข้อจำกัดในเรื่องของโหลด (FAN-IN/FAN-OUT) เนื่องจากคุณสมบัติของ Port I/O ของ CPU เอง ก็มีข้อจำกัดในการขับกระแสให้กับโหลดได้ประมาณ 25mA เท่านั้น ซึ่งคงไม่สามารถต่ออุปกรณ์ร่วมกันในบัสได้โดยไม่จำกัดจำนวนเหมือนในทฤษฎีบอกไว้ ซึ่งในความเป็นจริงอาจต้องพิจารณาตามความเหมาะสมและความจำเป็นในการใช้งานจริงๆด้วยว่าในระบบบัสหนึ่งๆของ I^2C นั้นควรต่ออุปกรณ์ในบัสจำนวนเท่าใด

หน้าที่และเบอร์ของอุปกรณ์ I^2C	**รหัสตำแหน่งมาตรฐานในการอ่าน/เขียน**	**รหัสตำแหน่งของบอร์ด CP-PIC 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-PIC V4.0

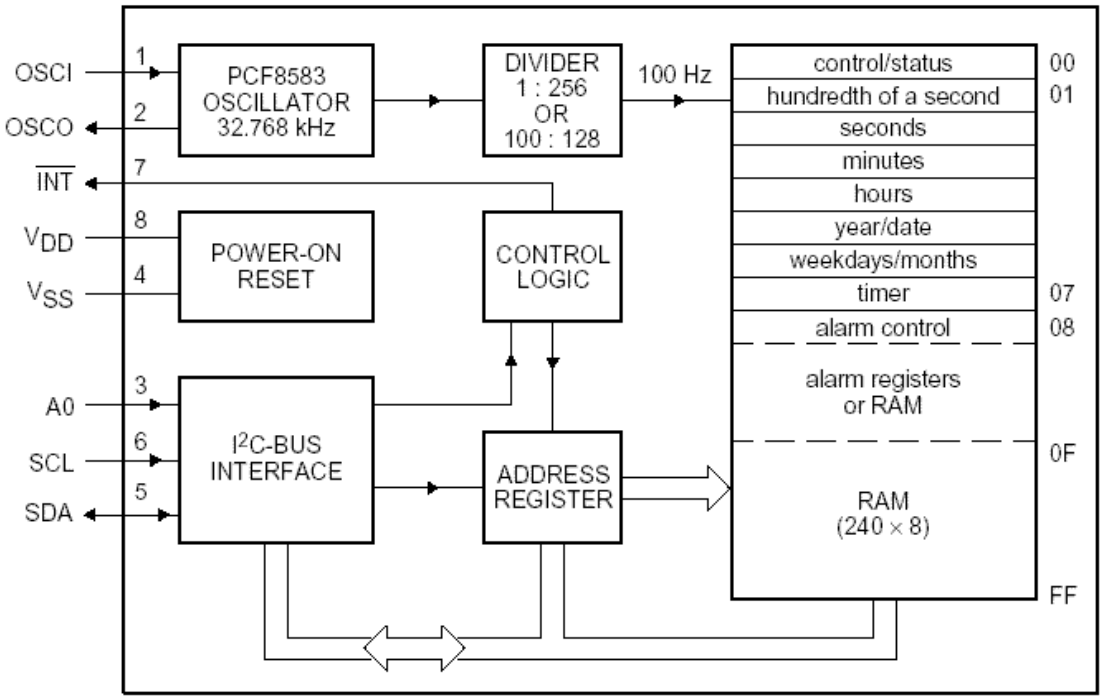
*****หมายเหตุ***** 1. ค่า X หมายถึงค่าโลจิกของขาสัญญาณ Address ของอุปกรณ์ ที่กำหนดในวงจร

2. ค่า ? หมายถึงบิตสำหรับกำหนดว่าต้องการเขียน หรือ อ่าน ข้อมูลกับอุปกรณ์

3. เนื่องจาก RTC เบอร์ PCF8583 และ E^2PROM เบอร์ในกลุ่ม 24XX นั้นมีรหัสตำแหน่ง 4บิตแรกซ้ำกัน หรือ อยู่ในกลุ่มเดียวกัน ซึ่งในบอร์ด CP-PIC V4.0 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-PIC 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



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-PIC 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 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-PIC V4.0 นั้น จะกำหนดให้ขาสัญญาณ A0 ของ PCF8583 มีสภาวะทางโลจิกเป็น “1” คงที่ไว้เลย ดังนั้น RTC เบอร์ PCF8583 ในบอร์ด CP-PIC V4.0 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) ในกลุ่มที่สามารถกำหนดตำแหน่งรหัส 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	$R/\overline{W}$

รูปแสดงรหัส 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 โดยกำหนดสภาวะของขา สัญญาณ ลอจิก แอดเดรสที่แตกต่างกันออกไป โดยสามารถสรุปให้เห็นได้ดังตารางต่อไปนี้

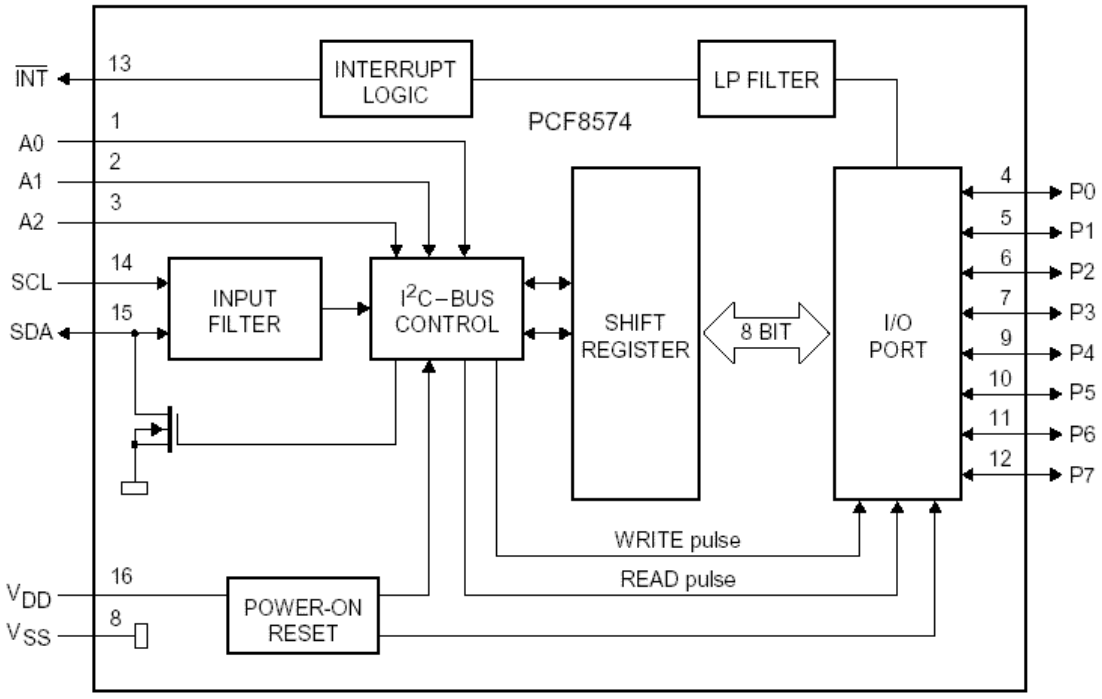
เบอร์(ความจุ)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
24XX32 (4Kx8)	1	0	1	0	A2	A1	A0	R/W
24XX64 (8Kx8)	1	0	1	0	A2	A1	A0	R/W
24XX128 (16Kx8)	1	0	1	0	A2	A1	A0	R/W
24XX256 (32Kx8)	1	0	1	0	A2	A1	A0	R/W

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 A0-A2 ดังนั้นเมื่อทำการติดตั้งใช้งานหน่วยความจำเบอร์เหล่านี้กับบอร์ด CP-PIC 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)

ตามปรกติแล้ว CPU เบอร์ PIC 16F877,18F442 และ 18F458 นั้นจะมีพอร์ต I/O สำหรับให้ผู้ใช้สามารถนำไปใช้งานได้มากถึง 5 พอร์ต อยู่แล้ว ซึ่งในส่วนของบอร์ด CP-PIC V3.0 นั้น พอร์ต I/O ทั้งหมดของ CPU จะปล่อยว่างไว้ให้ผู้ใช้เลือกใช้งานอย่างอิสระตามต้องการ แต่สำหรับบอร์ดรุ่น CP-PIC V4.0 นั้น พอร์ต I/O ต่างๆของ CPU จะถูกจัดสรรออกไปใช้งานในวงจรต่างๆ ดังได้กล่าวอธิบายมาแล้วในข้างต้น แต่ถ้าหากว่าผู้ใช้งานมีความจำเป็นต้องใช้งานพอร์ต I/O จำนวนมาก และจำนวนพอร์ต I/O ของ CPU ที่มีอยู่ไม่เพียงพอต่อการใช้งานแล้ว ผู้ใช้ก็สามารถทำการเพิ่มเติมพอร์ต I/O ได้อีก โดยในบอร์ด CP-PIC 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

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	$R/\overline{W}$

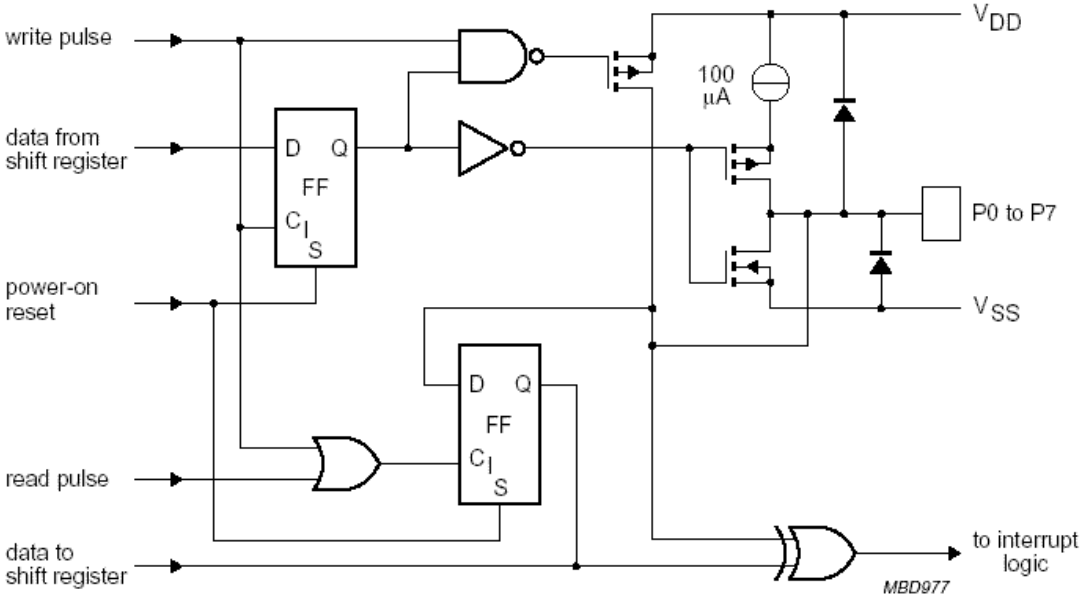
รูปแสดง รหัส Control Byte **ของ** PCF8574

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 PIC 16F877,18F442 และ 18F458 ที่ใช้กับบอร์ด CP-PIC V3.0&V4.0 นั้น จะมีวงจรสื่อสารแบบอนุกรม (Universal Synchronous Asynchronous Receiver Transmitter : USART) บรรจุรวมไว้ด้วยแล้ว ซึ่งวงจรส่วนนี้ผู้ใช้งานสามารถทำการเขียนโปรแกรมควบคุมการสื่อสารข้อมูลของ CPU กับอุปกรณ์อื่นๆได้ตามต้องการ โดยในส่วนของโปรแกรมนั้น ผู้ใช้สามารถกำหนดรูปแบบของการสื่อสารข้อมูลได้เองจากโปรแกรมที่เขียนขึ้น ไม่ว่าจะเป็นความเร็วในการสื่อสาร (Baudrate) จำนวนบิตข้อมูลในการรับส่ง (Data Bit) การกำหนดบิตรวจสอบความถูกต้องข้อมูล (Parity) และคุณสมบัติอื่นๆ ซึ่งในรายละเอียดส่วนนี้จะไม่ขอกล่าวถึงขอให้ผู้ใช้ศึกษาจากคู่มือสถาปัตยกรรมทางฮาร์ดแวร์หรือ Data Sheet ของ CPU เบอร์ ต่างๆ เหล่านี้เอง

ซึ่งปกติแล้วขาสัญญาณสำหรับ รับ-ส่ง ข้อมูลของ CPU นั้น สามารถนำไปเชื่อมต่อกับขาสัญญาณ รับ-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-PIC V3.0&V4.0 จึงได้ออกแบบวงจร Line Driver สำหรับทำหน้าที่เป็น Buffer เพื่อเปลี่ยนแปลงระดับสัญญาณทางไฟฟ้าของขาสัญญาณ รับ-ส่ง ข้อมูลของ CPU ที่เป็น แบบ TTL ให้สามารถรับส่งข้อมูลกันได้ในระยะทางที่ไกลมากขึ้น (สามารถอ่านรายละเอียดเพิ่มเติมได้จากหัวข้อ “ความรู้ทั่วไปเกี่ยวกับการสื่อสารอนุกรม” ในภาคผนวกท้ายเล่มของคู่มือนี้) โดยบอร์ด CP-PIC V4.0 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

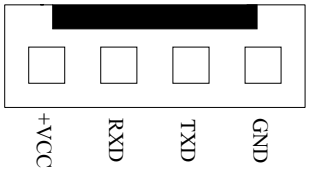
สำหรับสายสัญญาณที่จะนำมาใช้สำหรับทำการสื่อสารแบบ RS232 นั้น จะใช้สัญญาณเพียง 2-3 เส้น เท่านั้น ทั้งนี้ขึ้นอยู่กับความต้องการในการสื่อสารว่าต้องการสื่อสารแบบทิศทางเดียวหรือสองทิศทาง

- **การสื่อสาร RS232 แบบสองทิศทาง** ซึ่งจะมีทั้งการรับข้อมูลและส่งข้อมูลไปมา ระหว่างด้านรับและด้านส่ง โดยในกรณีนี้จะต้องใช้สายสัญญาณจำนวน 3 เส้น สัญญาณรับข้อมูล (RXD) สัญญาณส่งข้อมูล(TXD) และสัญญาณอ้างอิง (GND) โดยในการเชื่อมต่อสายนั้นจะต้องทำการสลับสัญญาณกับอุปกรณ์ปลายทางด้วย คือ สัญญาณส่ง (TXD) จากบอร์ด CP-PIC V3.0&V4.0 จะต้องต่อเข้ากับ

สัญญาณรับ (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-PIC V3.0&V4.0 นั้น จะเป็นจุดเชื่อมต่อของสัญญาณ รับ-ส่ง ข้อมูล ที่เปลี่ยนระดับสัญญาณเป็นแบบ RS232 แล้ว ซึ่งจะมีลักษณะเป็นแบบขั้ว CPA ขนาด 4 PIN สำหรับใช้เป็นจุดเชื่อมต่อสัญญาณ รับ-ส่ง ข้อมูลกับอุปกรณ์ภายนอก โดยมีลักษณะการจัดเรียงสัญญาณดังนี้



แสดงขั้วต่อสัญญาณ RS232 ของบอร์ด CP-PIC V3.0 & 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-PIC V3.0&V4.0 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

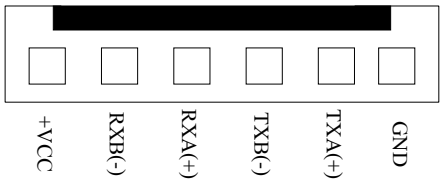
การสื่อสารอนุกรมแบบ RS422

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 และตัว

แปลงสัญญาณด้านส่งอีก 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 คือสามารถทำการสื่อสารกันได้ในระยะทางที่ไกลขึ้นกว่าแบบ 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-PIC V4.0 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-PIC V4.0 เมื่อเลือกเป็น RS422

การสื่อสารอนุกรมแบบ RS485

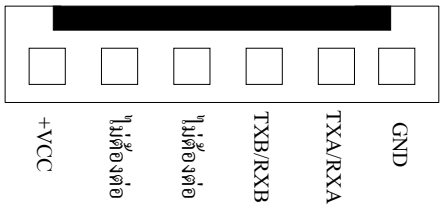
ในการสื่อสารแบบ RS485 นี้จะมีคุณสมบัติของสัญญาณทางไฟฟ้าเหมือนกับ RS422 ทุกประการ เพียงแต่ว่าในการสื่อสารแบบ RS485 นี้จะใช้สายสัญญาณในการรับส่งข้อมูลกันเพียง 2 เส้น เท่านั้น แต่จะมีความพิเศษกว่าแบบ RS422 ตรงที่ ทิศทางของสัญญาณจะสามารถปรับเปลี่ยนได้จากโปรแกรม กล่าวคือ สัญญาณทั้ง 2 เส้น นี้สามารถจะสลับหน้าที่เป็นด้านส่ง และ เป็นด้านรับได้ ตามต้องการ โดยการควบคุมจาก CPU โดยจากบอร์ด CP-PIC V4.0 นั้น จะกำหนดให้สัญญาณ RC5 ทำหน้าที่สำหรับควบคุมทิศทางของข้อมูลว่าจะให้เป็นรับหรือส่ง โดยถ้าควบคุมให้ RC5 มี

สภาวะเป็น “1” จะเป็นการกำหนดทิศทางให้เป็นฝ่ายส่งข้อมูล แต่ถ้าสภาวะของ RC5 เป็น “0” จะเป็นการกำหนดทิศทางให้เป็นฝ่ายรับข้อมูล ซึ่งจากคุณสมบัติข้อนี้จะทำให้การสื่อสารแบบ RS485 สามารถทำการต่อขนานอุปกรณ์ร่วมกันในสายส่งเดียวกันได้จำนวนหลายๆจุด โดยถ้าใช้ไอซี Line Driver เบอร์ 75176 จะสามารถต่อขนานอุปกรณ์กันได้จำนวน 32 จุด แต่ถ้าเลือกใช้ไอซี Line Driver เบอร์ MAX3088 แล้วจะสามารถต่อขนานอุปกรณ์ในสายคู่เดียวกันได้มากถึง 256 จุด เลยทีเดียว แต่มีข้อแม้ว่า เมื่อมีการต่ออุปกรณ์ขนานกันในสายสัญญาณคู่เดียวกันมากกว่า 2 จุดแล้วจะต้องเขียนโปรแกรมควบคุมให้มีการส่งข้อมูลออกมาในสายครั้งละ 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-PIC 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” ทำหน้าที่เป็นฝ่ายส่งข้อมูล

เพียงอย่างเดียว แต่ถ้าต้องการใช้งานแบบ RS485 จะต้องกำหนด Jumper ไว้ทางด้าน “485” เพื่อกำหนดให้ไอซี Line Driver ในตำแหน่ง “TXD/485” ทำหน้าที่เป็นทั้งฝ่ายรับและฝ่ายส่ง ตามการควบคุมของสัญญาณ RC5

- **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 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 ดังนั้นจึงต้องทำการต่อค่าความต้านทานจากภายนอกไปชดเชยค่าความต้านทานของสายสัญญาณด้วย 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 ซึ่งถ้าหากว่ามีการต่อสายสัญญาณระยะทางไกลๆหรือมีการต่อสายระยะทางไกล้ๆแต่ไม่ได้มีการส่งข้อมูลออกมาในสายตลอดเวลาแล้ว ควรที่จะทำการ Short Jumper นี้ไว้ด้วยเสมอ โดยเฉพาะอย่างยิ่งเมื่อใช้งานเป็นแบบ RS485 หรือใช้งานเป็นตัวอุปกรณ์ที่อยู่ในตำแหน่ง

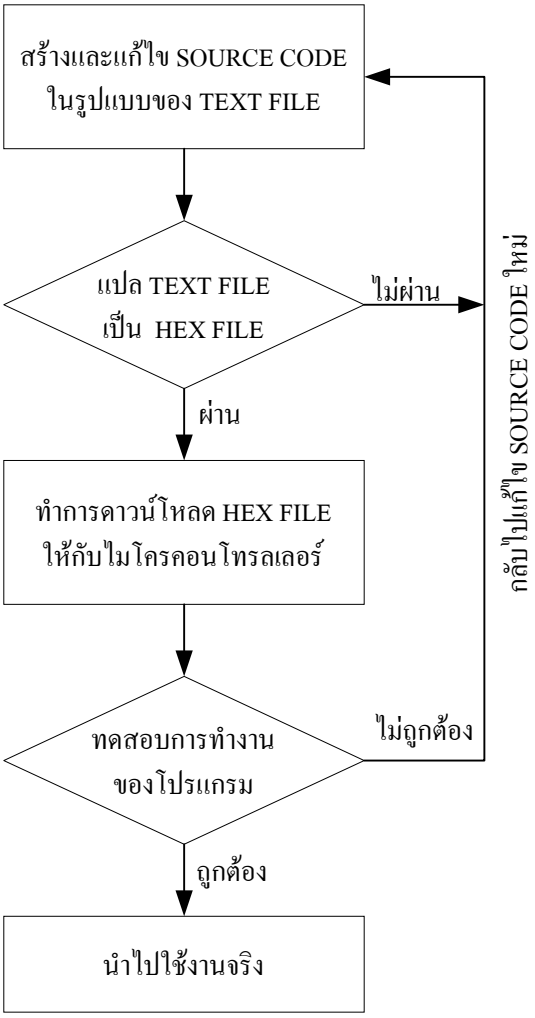
ต้นสายและปลายสายควรทำการ Short Jumper นี้ไว้เสมอ ส่วนอุปกรณ์ที่อยู่ในตำแหน่งอื่นๆที่มีระยะไม่ไกลจากจุดต้นสายและปลายสายมากนักก็อาจ Open Jumper นี้ออกก็ได้ แต่อย่างน้อยที่สุด ควรมีการ Short Jumper นี้ให้กับอุปกรณ์ที่ต่อร่วมอยู่ในสายสัญญาณจำนวน 1 จุดเสมอ

- **Jumper TH** เป็น Jumper ใช้สำหรับเลือกกำหนดการเชื่อมต่อ ตัวต้านทานสำหรับทำหน้าที่คงสถานะของสัญญาณ TXA (TX+) หรือ Fail Safe Resister เพื่อให้สัญญาณ TXA (TX+) มีสภาวะแน่นอนเมื่อไม่มีการส่งสัญญาณใดๆออกมาในสายเลย ซึ่งถ้าหากว่ามีการต่อสายสัญญาณระยะทางไกลๆหรือมีการต่อสายระยะทางไกล้ๆแต่ไม่ได้มีการส่งข้อมูลออกมาในสายตลอดเวลาแล้ว ควรที่จะทำการ 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 นั้นจะมีความใกล้เคียง กันมาก แต่มีข้อแตกต่างอย่างหนึ่งที่เห็นได้ชัดเจนที่สุด คือ ถ้าเป็นแบบ RS422 จะไม่สามารถสั่งเปลี่ยน ทิศทางการรับส่งข้อมูลด้วยโปรแกรมได้ ซึ่งทิศทางการรับส่งจะกำหนดตายตัวจากวงจร แต่ถ้าเป็นแบบ RS485 นั้น จะสามารถสั่งควบคุมทิศทางการรับส่งจากโปรแกรมได้ว่าจะให้ทำหน้าที่เป็นฝ่ายรับ หรือฝ่ายส่ง อย่างใดอย่างหนึ่งได้ตามต้องการได้

การพัฒนาโปรแกรมของบอร์ด CP-PIC V3.0&V4.0

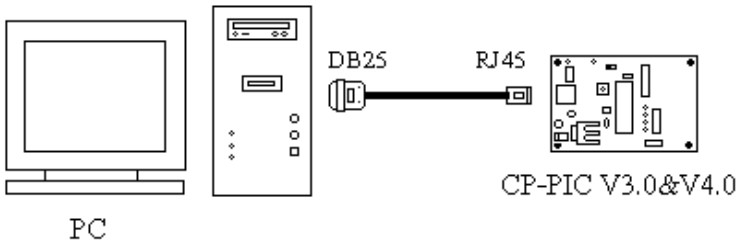
สำหรับการพัฒนาโปรแกรมนั้น ผู้ใช้งานสามารถเลือกที่จะใช้ภาษาใดก็ได้ในการพัฒนา เช่น ภาษาAssembly ,Basic หรือ ภาษา C ทั้งนี้ขึ้นอยู่กับความถนัดของผู้ใช้งาน แต่สุดท้ายแล้วจะต้องได้ไฟล์ที่จะโปรแกรมให้กับ CPU นั่นก็คือ HEX FILE ดังนั้นในการพัฒนาโปรแกรมจึงต้องมี Compiler สำหรับแปลภาษาที่เราเขียน (TEXT FILE) ให้เป็นภาษาที่ไมโครคอนโทรลเลอร์เข้าใจ (HEX FILE) สำหรับในที่นี้จะขอกล่าวถึงเฉพาะวิธีการ Download Hex File ให้กับบอร์ดเท่านั้น ส่วนวิธีการเขียนโปรแกรมและการสั่งแปลคำสั่งให้ได้เป็น Hex File นั้น ขอให้ผู้ใช้ศึกษาจากข้อกำหนดของโปรแกรมแปลภาษาที่จะนำมาใช้ในการเขียนโปรแกรมเอง ซึ่งบอร์ด CP-PIC V3.0&V4.0 จะออกแบบวงจรให้สามารถทำการโปรแกรมข้อมูลลง CPU ได้ภายในบอร์ดโดยในขั้นตอนการพัฒนาโปรแกรมนั้นสามารถสรุปเป็น โฟลว์ชาร์ต ดังนี้



แผนผัง แสดงขั้นตอนในการพัฒนาโปรแกรม

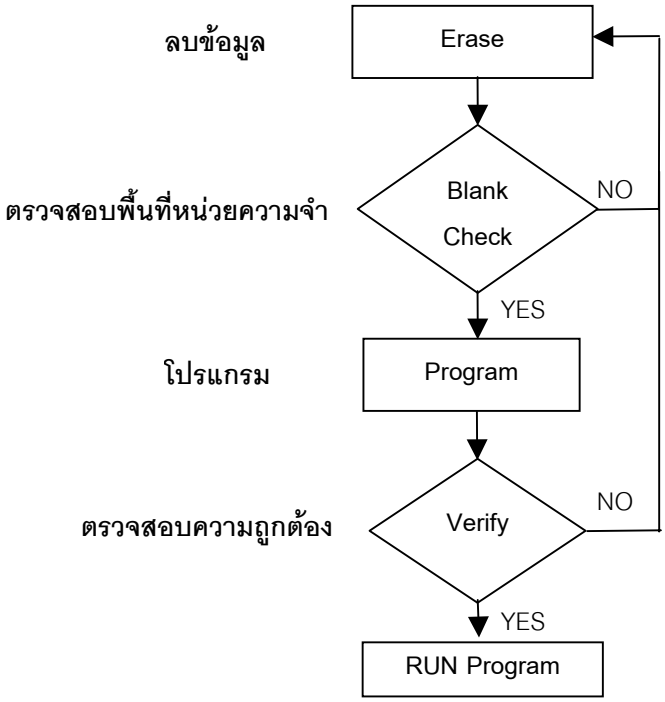
ขั้นตอนการดาว์นโหลดโปรแกรม

1. ให้ต่อสายดาว์นโหลดระหว่างบอร์ด CP-PIC กับ เครื่อง PC โดยปลายด้านที่ต่อกับ บอร์ด CP-PIC จะเป็นคอนเนคเตอร์ RJ-45 ส่วนปลายอีกด้านที่ต่อกับ PC จะเป็น DB 25 (PRINTER PORT) ดังรูป



2. จ่ายไฟให้กับบอร์ด CP-PIC โดยจะต้องจ่ายไฟ 13 ถึง 16 V
3. เลื่อนสวิตช์ RUN/PROG มาที่ตำแหน่ง PROG เพื่อปรับให้อยู่ในโหมดของการโปรแกรม
4. เปิดโปรแกรม EPICwin โดยการดับเบิลคลิกที่ ICON 

5. เปิดไฟล์ที่จะดาวน์โหลดโดยไฟล์ที่จะดาวน์โหลดจะต้องมีนามสกุลเป็น .HEX เท่านั้น
6. กำหนดเบอร์อุปกรณ์ (Device) และ ค่า Configuration ต่างๆ ตามการใช้งานให้ถูกต้อง ซึ่งสามารถดูรายละเอียดเพิ่มเติมได้ในหัวข้อ การใช้งาน EPICWin
7. ทำการโปรแกรมข้อมูลลง CPU ซึ่งโดยทั่วไปแล้วขั้นตอนการโปรแกรมจะเรียงลำดับดังนี้คือ



ซึ่งเราอาจข้ามขั้นตอนบางขั้นตอนได้เพื่อเป็นการประหยัดเวลาในการโปรแกรม เช่น อาจจะ Erase แล้ว ทำการ Program เลยก็ได้ ซึ่งหน้าที่การทำงานของขั้นตอนต่างๆ สามารถดูรายละเอียดได้ในหัวข้อการใช้งาน EPICWin และเมื่อต้องการกลับสู่โหมดการ RUN โปรแกรมให้เลื่อนสวิตช์ RUN/PROG มาที่ตำแหน่ง RUN

การใช้งาน EPICWin

EPICWin เป็นโปรแกรมที่ทำหน้าที่ในการโปรแกรมข้อมูลลง CPU ตระกูล PIC ได้หลายสิบเบอร์ ตามรายการที่แสดงไว้ในช่อง Device นอกจากนี้ ยังสามารถอ่านข้อมูล กลับขึ้นมาได้อีกด้วยซึ่งการโปรแกรมจะเป็นแบบ High Voltage ICSP Program หรือก็คือ การโปรแกรมแบบไฟสูงนั่นเองโดยหน้าที่การทำงานของเมนูต่างๆ มีรายละเอียดดังนี้

รูปแสดงหน้าต่างโปรแกรม EPICWIN

หน้าที่การทำงานของเมนูต่าง ๆ

Open : ทำหน้าที่ในการเปิด File สำหรับดาวน์โหลด

Save : ทำหน้าที่บันทึกโปรแกรม

Program : ใช้โปรแกรมข้อมูลลง CPU

Verify : ใช้ในการตรวจสอบความถูกต้องของข้อมูลที่ได้ทำการโปรแกรมไปแล้ว ซึ่งจะเป็นการอ่านข้อมูลจาก CPU มาเปรียบเทียบกับข้อมูลที่เราดาวน์โหลดว่าตรงกันหรือไม่ ถ้าไม่ตรงแสดงว่าการดาวน์โหลดผิดพลาด

Read : ใช้ในการอ่านข้อมูลจากหน่วยความจำโปรแกรมของ CPU จะใช้ได้ก็ต่อเมื่อ CPU ไม่ได้มีการ Lock หรือ Protect ไว้เท่านั้น

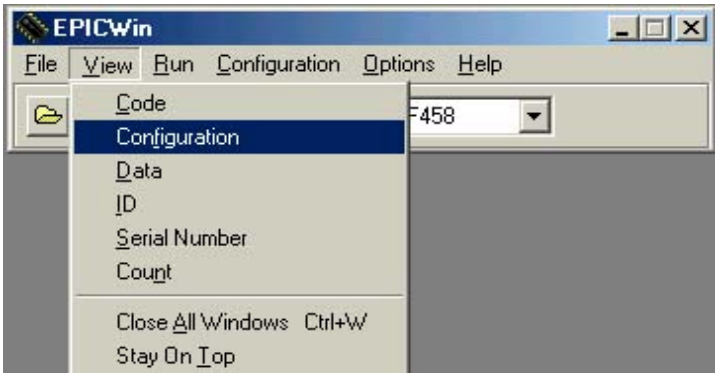
Bank Check : ทำหน้าที่ตรวจสอบพื้นที่ของหน่วยความจำใน CPU ว่าว่างหรือไม่ ซึ่งในการโปรแกรม พื้นที่ของหน่วยความจำใน CPU จะต้องว่างจึงจะสามารถโปรแกรมได้

Erase : ทำหน้าที่ ลบข้อมูลในหน่วยความจำของ CPU

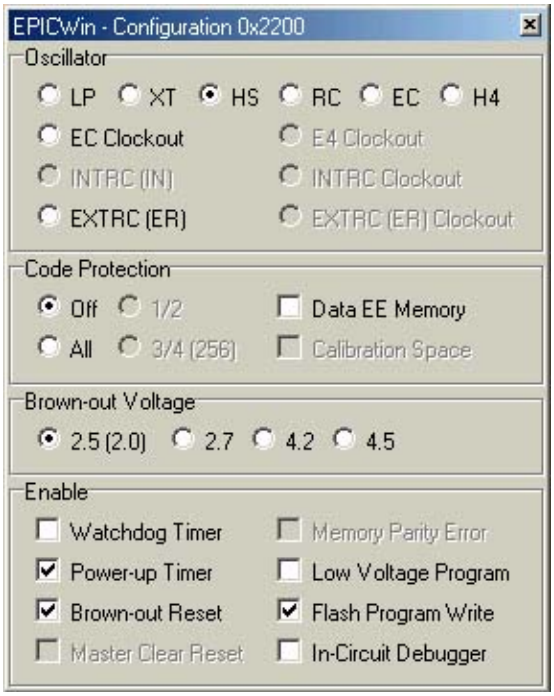
Device : ทำหน้าที่กำหนดเบอร์ CPU ที่ต้องการโปรแกรม

การกำหนดค่า Configuration

ในการโปรแกรมข้อมูลในแต่ละครั้งเราจะต้องกำหนด Configuration ให้ถูกต้องตามการใช้งาน โดยเข้าไปที่ เมนู View -> Configuration ตามรูป



รูปแสดงการเปิดเมนู Configuration



แสดงหน้าต่างของ Configuration

ประกอบไปด้วยส่วนต่างๆ ดังนี้

Oscillator : เป็นการกำหนดโหมดของความถี่ในการทำงานประกอบด้วยส่วนต่างๆ ต่อไปนี้

- **LP** (Low Power Crystal) : โหมด คริสตอลพลังงานต่ำ
- **XT** (Crystal/Resonator) : คริสตอล หรือ เรโซเนเตอร์
- **HS** (High Speed Crystal/Resonator) : คริสตอล หรือ เรโซเนเตอร์ความเร็วสูง
- **RC** (External Resistor/Capacitor) : วงจร RC ภายนอก

- **EC** (External Clock) : สัญญาณนาฬิกาจากภายนอก

- **H4** (HS + PLL : High Speed Crystal/Resonator with PLL enabled)

: คูณ 4 PLL คือ จะทำการคูณสัญญาณนาฬิกาที่เข้ามาเช่น ใช้ OSC ความถี่ 10 MHz เมื่อผ่านกระบวนการนี้จะได้ความถี่เท่ากับ 40MHz (คุณสมบัตินี้จะมีใน PIC บางตัวเท่านั้น)

- **EC** Clockout : ใช้คริสตอลภายนอก และ กำหนดให้ขา OSC2 เป็นเอาต์พุตออสซิลเลเตอร์

- **INTRC**(IN) : วงจร RC ภายในไมโครคอนโทรลเลอร์

- **INTRC** Clockout : วงจร RC ภายในไมโครคอนโทรลเลอร์ กำหนดให้ OSC2 เป็นเอาต์พุต

- **EXTRC** (ER) : วงจรตัวต้านทานภายนอก กำหนดค่าความถี่จากค่าความต้านทานของตัวต้านทานที่นำมาต่อ

- **EXTRC** Clockout : วงจรตัวต้านทานภายนอก กำหนดค่าความถี่จากค่าความต้านทานของตัวต้านทานที่นำมาต่อ และ กำหนดให้ OSC2 เป็น Clockout

ซึ่งในการเลือกใช้ สามารถพิจารณาได้จากตารางต่อไปนี้

ตารางแสดงค่าความถี่ คริสตอล ในโหมดต่าง ๆ

OSC TYPE	Crystal Freq	Cap , C1	Cap , C2
LP	32 KHz	33pF	33pF
	200 KHz	15pF	15pF
XT	200 KHz	47-68 pF	47-68 pF
	1.0 MHz	15 pF	15 pF
	4.0 MHz	15 pF	15 pF
HS	4.0 MHz	15 pF	15 pF
	8.0 MHz	15-33pF	15-33pF
	10.0 MHz	15-33pF	15-33pF
	20.0 MHz	15-33pF	15-33pF
HS+PLL	4.0 MHz	15 pF	15 pF
	8.0 MHz	15-33pF	15-33pF
	10.0 MHz	TBD	TBD

Code Protection : ใช้กำหนดการปกป้องข้อมูลให้ไม่สามารถอ่านกลับได้ ซึ่งสามารถเลือกได้หลายระดับ หรือ ถ้าไม่ต้องการ Protect ก็ให้เลือก Off ก็จะสามารถอ่านข้อมูลได้ปกติ

Brown-Out Voltage : ใช้กำหนดขนาดของแรงดันในการโปรแกรมโดยทั่วไปแล้วโปรแกรมจะ Default ค่าไว้ให้แล้ว

Enable : เป็นการกำหนดการทำงานของฟังก์ชันต่างๆ ซึ่งเราสามารถเลือกได้ว่าจะให้ทำงาน(Enable) หรือไม่ทำงาน (Disable) โดยประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

- Watchdog Timer
- Memory Parity Error
- Power-up Timer
- Low Voltage Program
- Brown-out Reset
- Flash Program Write
- Master Clear Reset
- In-Circuit Debugger

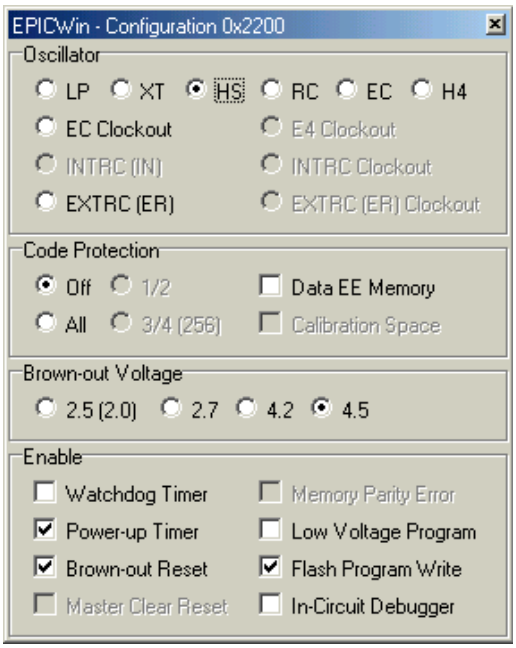
ข้อควรระวัง โดยทั่วไปแล้ว ค่า Configurator ต่างๆ เหล่านี้ จะถูกกำหนดเอาไว้แล้วในขณะที่เราเลือกเบอร์ CPU แต่สิ่งที่ต้องพิจารณาก็คือโหมดของ

 # Oscillator เนื่องจากโปรแกรม EPICWIN ไม่สามารถตรวจสอบได้ว่าเราใช้ คริสตอล ขนาดความถี่ประเภทใด เราจึงต้องตรวจสอบในส่วนนี้ให้ถูกต้องด้วยตัวเอง ซึ่งหากกำหนดผิด CPU ก็จะไม่ทำงาน เราสามารถพิจารณาการเลือกค่าความถี่ ได้ตามตาราง

 # Low Voltage Program เราจะต้อง ไม่เลือกการทำงานในส่วนนี้เพราะถ้าเลือกการทำงานในส่วนนี้ จะทำให้ขา PGM (RB5 เบอร์ 18F458) ทำงานในโหมด Low Voltage Program เราจะไม่สามารถนำขา RB5(PGM) นี้ ไปใช้งานเป็น I/O ได้ เพราะหากขา RB5(PGM) ได้รับ Logic ‘1’ CPU จะเข้าสู่โหมดการโปรแกรมแบบ Low Voltage Program ทันที

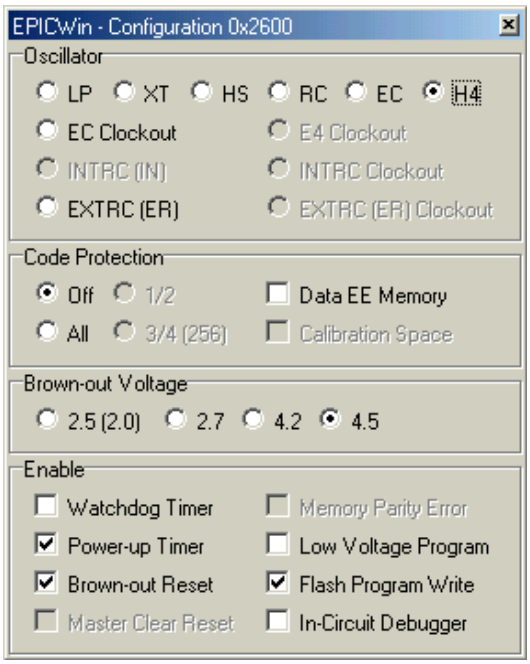
 # Watchdog Timer ไม่ควรเลือกให้ทำงานในโหมดนี้ ถ้าหากไม่ได้ใช้งานในส่วนของ วอตช์ด็อกไท-เมอร์ เพราะหาก คลิกเลือกให้วอตช์ด็อกทำงาน เมื่อ CPU ทำงาน หากเราไม่เขียนโปรแกรมไปเคลียร์ค่า วอตช์ด็อก ก็จะเกิดการรีเซ็ท ตลอดตามค่าของวอตช์ด็อกที่ตั้งไว้

* ตัวอย่างการกำหนดค่า Configuration ให้ทำงานที่ความถี่ 10 MHz ของบอร์ด CP-PIC V3.0&V4.0



รูปแสดงการกำหนดค่า Configuration เมื่อใช้งานความถี่ 10 MHz

* ตัวอย่างการกำหนดค่า Configuration ของ CP-PIC V3.0&V4.0 กรณีการใช้ เฟสล็อคลูป(x4 PLL) เพื่อคูณสัญญาณนาฬิกาเป็น 4 เท่า คือ ค่าคริสตอล 10 MHz เมื่อผ่านกระบวนการเฟสล็อกลูป ผลลัพธ์จะได้ เท่ากับ 40 MHz โดยกำหนดค่าต่างๆดังรูป



รูปแสดงการกำหนดค่า Configuration เมื่อใช้งานความถี่ 40MHz (PLL)

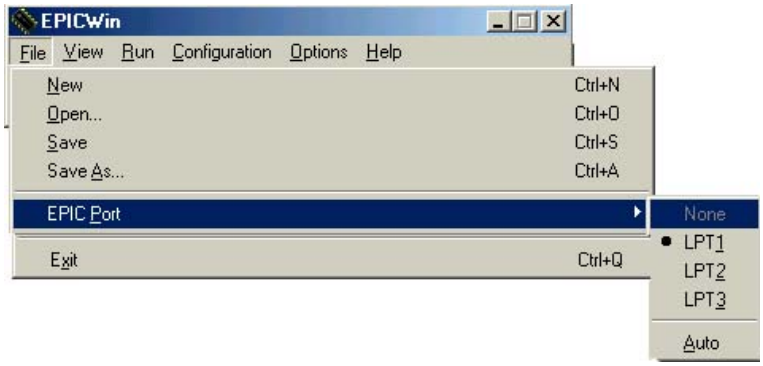
 # หมายเหตุ ในบอร์ด CP-PIC V3.0&V4.0 นี้ถ้าใช้กับ CPU PIC เบอร์ 18F442 หรือ 18F458 ซึ่งมีฟังก์ชันของเฟสล็อกลูปภายในจึง สามารถเลือกทำงานได้ 2 ความถี่คือ

- 10 MHz ความถี่เท่ากับคริสตอลออสซิลเลเตอร์ภายนอก
- 40 MHz ความถี่เท่ากับคริสตอลออสซิลเลเตอร์ภายนอกคูณด้วย 4 (ใช้วงจรเฟสล็อคลูปภายใน)

* ในการเปลี่ยนฟังก์ชันการทำงานของความถี่ในแต่ละครั้งจะต้องทำการปลดไฟเลี้ยง CPU ออกก่อนทุกครั้ง เช่น ถ้าเราใช้งานในโหมดความถี่ 10 MHz อยู่แล้วเราจะโปรแกรมใหม่เป็น 40 MHz เราต้องปลดไฟเลี้ยงบอร์ดออกแล้วจึงทำการโปรแกรมเพื่อเปลี่ยนโหมด เป็น 40MHz หากไม่ทำเช่นนี้ CPU ก็จะยังทำงานในความถี่เดิม

การกำหนดพอร์ตสำหรับดาว์นโหลด

การดาว์นโหลดจะทำผ่านพอร์ตขนาน (Parallel Port) ซึ่งจะต้องเลือกให้ตรงกับพอร์ตที่เราต่ออยู่สามารถทำได้โดย คลิกที่ เมนู File -> EPIC Ports แล้วทำการเลือกให้ถูกต้อง หรือ ถ้าเลือกเป็น Auto โปรแกรมก็จะค้นหาพอร์ตให้โดยอัตโนมัติ

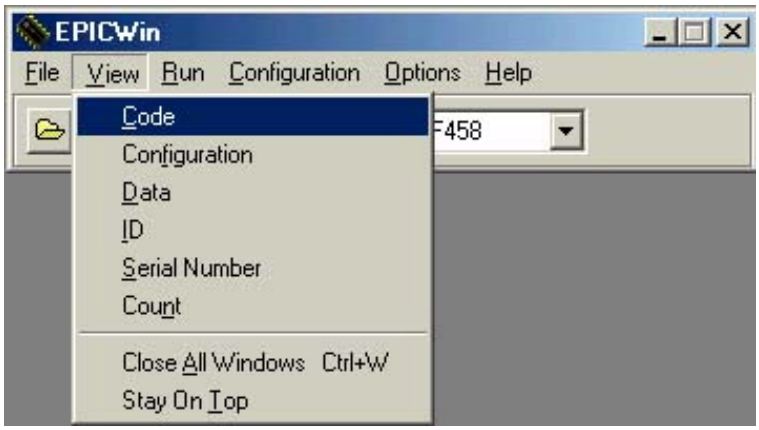


รูปแสดงการกำหนดพอร์ตสำหรับดาวน์โหลด

การอ่านข้อมูลจาก CPU

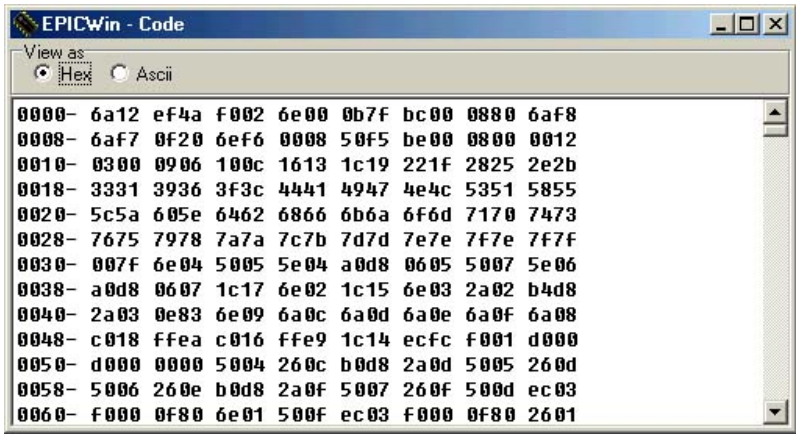
จากที่กล่าวไว้แล้วข้างต้น คือ EPICWin นี้สามารถทำการอ่านข้อมูลจากหน่วยความจำของ CPUได้ (เฉพาะตัวที่ไม่ถูก Protect ไว้) ซึ่งสามารถทำได้ตามขั้นตอนต่างๆ ดังนี้

- ทำการอ่านข้อมูลโดยใช้ เมนู READ (Run -> Read หรือ Ctrl R) หรือ คลิกที่ 
- เปิดเมนู View -> Code เพื่อดูข้อมูลที่อ่านได้ ดังรูป



รูปแสดงการเปิดดูข้อมูลที่อ่านได้ หรือ ข้อมูลที่อยู่ใน บัฟเฟอร์

- จะเกิดหน้าต่างที่แสดงซอร์สโค้ดต่างๆ ที่อ่านได้จาก CPU ดังรูป ซึ่ง Code เหล่านี้เราสามารถ Saveเก็บไว้ได้



รูปแสดง Code ต่างๆที่อ่านได้จาก CPU

ปัญหาที่อาจเกิดขึ้น และแนวทางการแก้ไข

CPU ไม่ทำงานหรือทำงานผิดพลาด อาจเกิดขึ้นจากหลายสาเหตุดังนี้

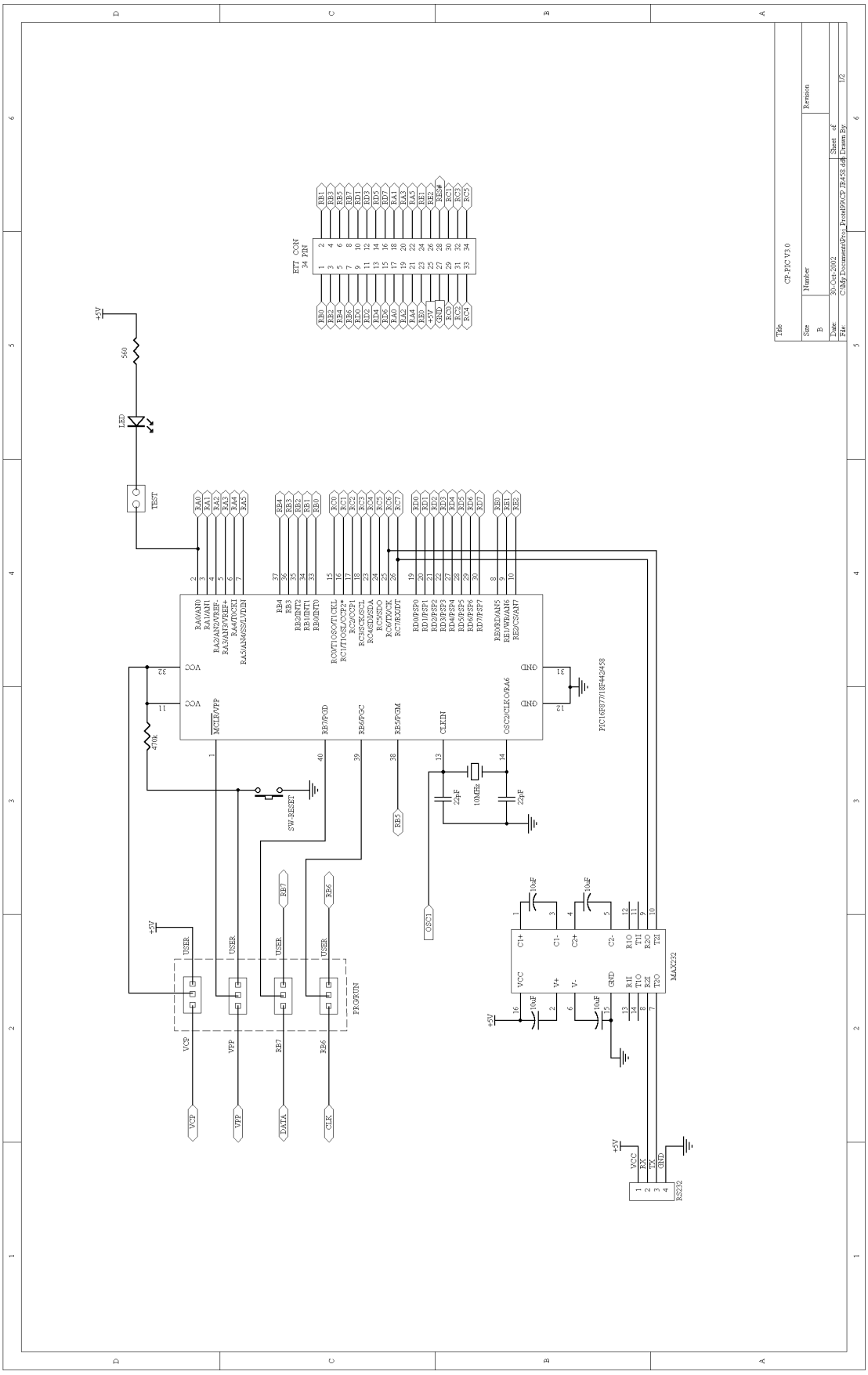
- สวิตช์ PROG/RUN อยู่ในตำแหน่ง PROG ทำให้ CPU อยู่ในโหมดโปรแกรมจึงไม่สามารถทำงานตามโปรแกรมได้ การแก้ไขให้เลือกสวิตช์มาที่โหมด RUN เมื่อต้องการให้ CPU ทำงานในโหมด Run Program
- Jumper OSC1 ยังเสียบอยู่ (Short) ให้ดึง (Open) Jumper OSC1 ออกแต่หากในขณะที่เสียบไว้แล้ว CPU ยังสามารถทำงานได้ ก็ไม่จำเป็นต้องถอดออก โดย Jumper OSC1 นี้จะทำหน้าที่กดสัญญาณ CLK ในระหว่างการดาวน์โหลด ในกรณีที่เกิดปัญหาในการดาว์นโหลดจึงต้อง Short จัมเปอร์นี้ แต่เมื่อเราทำการทดลองโปรแกรมจนแน่นอนพร้อมที่จะนำไปใช้งานจริงๆแล้ว แนะนำให้ถอด Jumper OSC1 ออก เพื่อให้ OSC ทำงานได้อย่างอิสระ
- การตั้งค่า Configuration โดยเฉพาะใน โหมดของ OSC มีผลมาก เพราะ PIC 18F458 สามารถทำงานได้ 2 ความถี่ คือ ความถี่จาก OSC ภายนอก(HS :ความถี่ 10 MHz) และ โหมด PLL (H4 : OSC ภายนอก x 4 ก็คือ 40MHz) ให้ตรวจสอบให้ถูกต้องว่าใช้งานในโหมดไหน
- Watchdog Timer หากเลือกการทำงานนี้ ใน Configuration แล้วไม่เขียนโปรแกรมไป Clear ค่า Watchdog จะทำให้เกิดการรีเซต ตลอดเวลาตามค่า Watchdog Timer ดังนั้นจึงควรตรวจสอบก่อนว่ามีการใช้งาน Watchdog Timer หรือไม่ ถ้าไม่ก็ไม่ควรเลือกการทำงานในส่วนนี้
- การ Set Jumper ต่างๆไม่ถูกต้อง ให้ตรวจสอบให้ถูกต้อง ซึ่งสามารถดูรายละเอียดได้จากหัวข้อ รายละเอียดการใช้งาน Jumper
- ในกรณีที่เขียนด้วยภาษา BASIC (ใช้ PIC Basic Pro) จะต้องมีการกำหนดค่าความถี่ที่ใช้งานโดยใช้คำสั่ง DEFINE OSC _ _ ซึ่ง หากไม่มีการ กำหนดความถี่ให้กับ โปรแกรม มันจะตีว่าเป็นความถี่ 4 MHz ซึ่งจะทำให้โปรแกรมทำงานผิดพลาดได้ดังนั้นเราจึงควรกำหนดให้ถูกต้อง

ตัวอย่าง

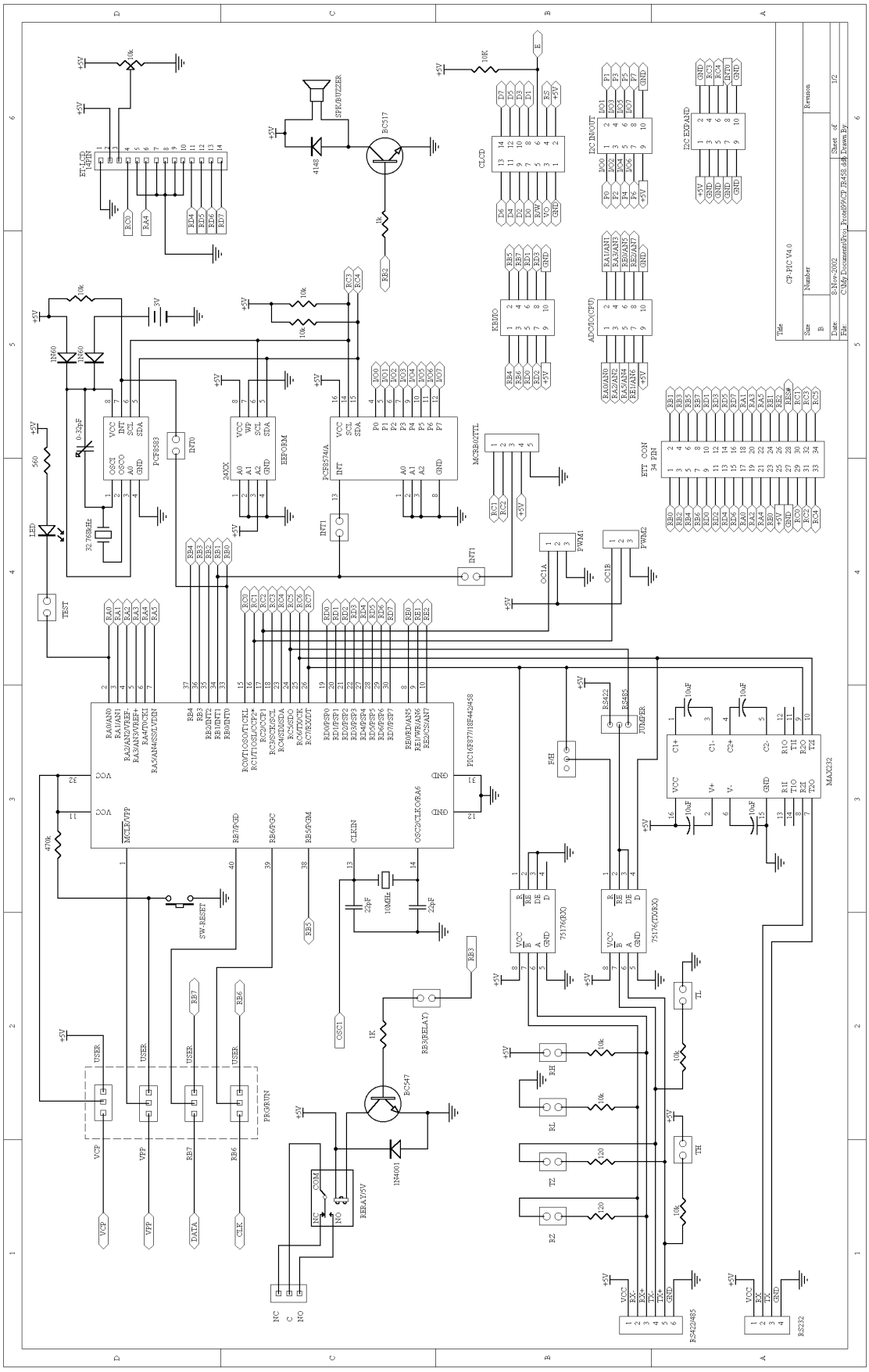
```
DEFINE    OSC    10    ‘ การกำหนดความถี่ 10 MHz
DEFINE    OSC    40    ‘ การกำหนดความถี่ 40 MHz
```

- การเปลี่ยนโหมดความถี่ เนื่องจาก PIC 18F458 ในบอร์ด CP-PIC V3.0&V4.0 สามารถทำงานได้ทั้ง 10MHz(HS) และ 40 MHz(H4) ฉะนั้นในการเปลี่ยนโหมดการทำงานจาก 10 MHz ไปเป็น 40MHz หรือจาก 40MHz เป็น 10MHz เมื่อทำการโปรแกรมเปลี่ยนโหมดเสร็จ CPU จะยังคงทำงานในโหมดความถี่เดิมอยู่ จนกว่าจะมีการปลดไฟเลี้ยง แล้วจ่ายไฟเข้ามาใหม่ CPU จึงจะเปลี่ยนมาทำงานในโหมดความถี่ใหม่ที่โปรแกรมให้ครั้งสุดท้าย
- ปัญหาอีกอย่างที่อาจเจอในการใช้งานโปรแกรม EPICWin คือ การตั้งค่า Power Option ของเครื่องคอมพิวเตอร์ ซึ่งจะอยู่ใน Control Panel โดยเมื่อเราตั้งค่า Power Option เป็นค่าต่างๆ อาจทำให้โปรแกรม EPICWin ไม่สามารถทำงานได้ทำให้ต้อง Restart เครื่องใหม่ ดังนั้นจึงควรตั้งค่า ต่างๆของ Power Option เป็นดังนี้

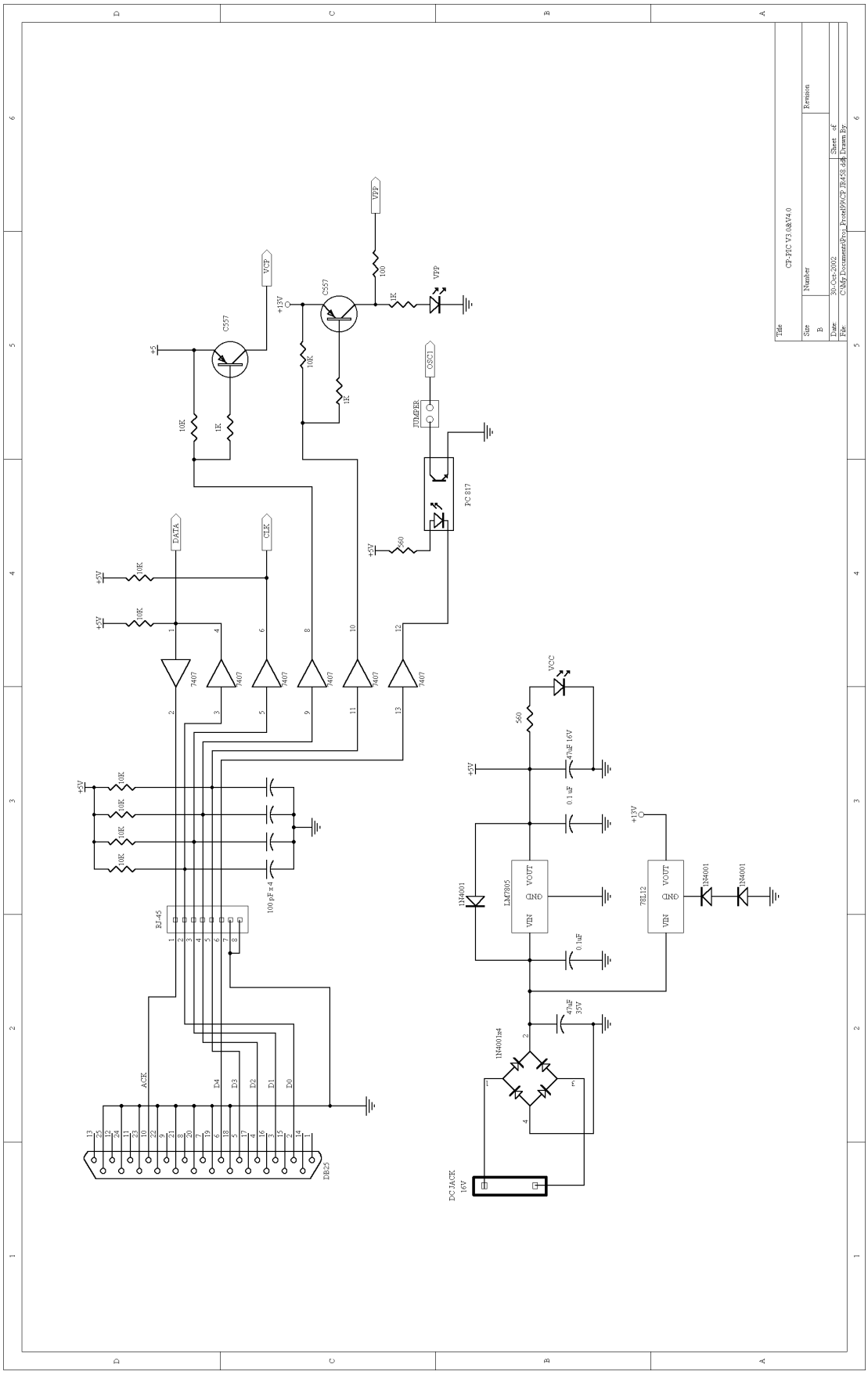
```
Turn off monitor      = Never
Turn off hard disks   = Never
System Stand by       = Never
```



รูป แสดงวงจรของบอร์ด CP-PIC V3.0



รูป แสดงวงจรของบอร์ด CP-PIC V4.0



รูป แสดงวงจรของบอร์ด CP-PIC V3.0&V4.0

ตารางคำสั่งของ PIC 16F877

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff		1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00 0000 0000 1000		
SLEEP	-	Go into standby mode	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is ’1’ for a pin configured as input and is driven low by an external device, the data will be written back with a ’0’.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 module.

3: If Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

Note: Additional information on the mid-range instruction set is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

ตารางคำสั่งของ PIC 18F442 และ 18F458

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb	LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS							
ADDWF	f, d, a	Add WREG and f	1	0010 01da	ffff ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and Carry bit to f	1	0010 00da	ffff ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	AND WREG with f	1	0001 01da	ffff ffff	Z, N	1,2
CLRF	f, a	Clear f	1	0110 101a	ffff ffff	Z	2
COMF	f, d, a	Complement f	1	0001 11da	ffff ffff	Z, N	1, 2
CPFSEQ	f, a	Compare f with WREG, skip =	1 (2 or 3)	0110 001a	ffff ffff	None	4
CPFSGT	f, a	Compare f with WREG, skip >	1 (2 or 3)	0110 010a	ffff ffff	None	4
CPFSLT	f, a	Compare f with WREG, skip <	1 (2 or 3)	0110 000a	ffff ffff	None	1, 2
DECF	f, d, a	Decrement f	1	0000 01da	ffff ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010 11da	ffff ffff	None	1, 2, 3, 4
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100 11da	ffff ffff	None	1, 2
INCF	f, d, a	Increment f	1	0010 10da	ffff ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011 11da	ffff ffff	None	4
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100 10da	ffff ffff	None	1, 2
IORWF	f, d, a	Inclusive OR WREG with f	1	0001 00da	ffff ffff	Z, N	1, 2
MOVF	f, d, a	Move f	1	0101 00da	ffff ffff	Z, N	1
MOVFF	f_s, f_d	Move f_s (source) to 1st word f_d (destination) 2nd word	2	1100 ffff 1111 ffff	ffff ffff ffff ffff	None	
MOVWF	f, a	Move WREG to f	1	0110 111a	ffff ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000 001a	ffff ffff	None	
NEGF	f, a	Negate f	1	0110 110a	ffff ffff	C, DC, Z, OV, N	1, 2
RLCF	f, d, a	Rotate Left f through Carry	1	0011 01da	ffff ffff	C, Z, N	
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100 01da	ffff ffff	Z, N	1, 2
RRCF	f, d, a	Rotate Right f through Carry	1	0011 00da	ffff ffff	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100 00da	ffff ffff	Z, N	
SETF	f, a	Set f	1	0110 100a	ffff ffff	None	
SUBFWB	f, d, a	Subtract f from WREG with borrow	1	0101 01da	ffff ffff	C, DC, Z, OV, N	1, 2
SUBWF	f, d, a	Subtract WREG from f	1	0101 11da	ffff ffff	C, DC, Z, OV, N	
SUBWFB	f, d, a	Subtract WREG from f with borrow	1	0101 10da	ffff ffff	C, DC, Z, OV, N	1, 2
SWAPF	f, d, a	Swap nibbles in f	1	0011 10da	ffff ffff	None	4
TSTFSZ	f, a	Test f, skip if 0	1 (2 or 3)	0110 011a	ffff ffff	None	1, 2
XORWF	f, d, a	Exclusive OR WREG with f	1	0001 10da	ffff ffff	Z, N	
BIT-ORIENTED FILE REGISTER OPERATIONS							
BCF	f, b, a	Bit Clear f	1	1001 bbba	ffff ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000 bbba	ffff ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011 bbba	ffff ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010 bbba	ffff ffff	None	3, 4
BTG	f, d, a	Bit Toggle f	1	0111 bbba	ffff ffff	None	1, 2

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is ’1’ for a pin configured as input and is driven low by an external device, the data will be written back with a ’0’.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are 2-word instructions. The second word of these instructions will be executed as a `NOP`, unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

5: If the Table Write starts the write cycle to internal memory, the write will continue until terminated.

ตารางคำสั่งของ PIC 18F442 และ 18F458 (ต่อ)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb LSb	Status Affected	Notes
CONTROL OPERATIONS						
BC	n	Branch if Carry	1 (2)	1110 0010 nnnn nnnn	None	
BN	n	Branch if Negative	1 (2)	1110 0110 nnnn nnnn	None	
BNC	n	Branch if Not Carry	1 (2)	1110 0011 nnnn nnnn	None	
BNN	n	Branch if Not Negative	1 (2)	1110 0111 nnnn nnnn	None	
BNOV	n	Branch if Not Overflow	1 (2)	1110 0101 nnnn nnnn	None	
BNZ	n	Branch if Not Zero	2	1110 0001 nnnn nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110 0100 nnnn nnnn	None	
BRA	n	Branch Unconditionally	1 (2)	1101 0nnn nnnn nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110 0000 nnnn nnnn	None	
CALL	n, s	Call subroutine 1st word	2	1110 110s kkkk kkkk	None	
		2nd word		1111 kkkk kkkk kkkk		
CLRWDT	—	Clear Watchdog Timer	1	0000 0000 0000 0100	$\overline{TO}$, $\overline{PD}$	
DAW	—	Decimal Adjust WREG	1	0000 0000 0000 0111	C	
GOTO	n	Go to address 1st word	2	1110 1111 kkkk kkkk	None	
		2nd word		1111 kkkk kkkk kkkk		
NOP	—	No Operation	1	0000 0000 0000 0000	None	
NOP	—	No Operation **(Note 4)**	1	1111 xxxx xxxx xxxx	None	
POP	—	Pop top of return stack (TOS)	1	0000 0000 0000 0110	None	
PUSH	—	Push top of return stack (TOS)	1	0000 0000 0000 0101	None	
RCALL	n	Relative Call	2	1101 1nnn nnnn nnnn	None	
RESET		Software device RESET	1	0000 0000 1111 1111	All	
RETFIE	s	Return from interrupt enable	2	0000 0000 0001 000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	Return with literal in WREG	2	0000 1100 kkkk kkkk	None	
RETURN	s	Return from Subroutine	2	0000 0000 0001 001s	None	
SLEEP	—	Go into Standby mode	1	0000 0000 0000 0011	$\overline{TO}$, $\overline{PD}$	
LITERAL OPERATIONS						
ADDLW	k	Add literal and WREG	1	0000 1111 kkkk kkkk	C, DC, Z, OV, N	
ANDLW	k	AND literal with WREG	1	0000 1011 kkkk kkkk	Z, N	
IORLW	k	Inclusive OR literal with WREG	1	0000 1001 kkkk kkkk	Z, N	
LFSR	f, k	Move literal (12-bit) 2nd word	2	1110 1110 00ff kkkk	None	
		to FSRx 1st word		1111 0000 kkkk kkkk		
MOVLB	k	Move literal to BSR<3:0>	1	0000 0001 0000 kkkk	None	
MOVLW	k	Move literal to WREG	1	0000 1110 kkkk kkkk	None	
MULLW	k	Multiply literal with WREG	1	0000 1101 kkkk kkkk	None	
RETLW	k	Return with literal in WREG	2	0000 1100 kkkk kkkk	None	
SUBLW	k	Subtract WREG from literal	1	0000 1000 kkkk kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR literal with WREG	1	0000 1010 kkkk kkkk	Z, N	
DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS						
TBLRD*		Table Read	2	0000 0000 0000 1000	None	
TBLRD*+		Table Read with post-increment		0000 0000 0000 1001	None	
TBLRD*-		Table Read with post-decrement		0000 0000 0000 1010	None	
TBLRD+*		Table Read with pre-increment		0000 0000 0000 1011	None	
TBLWT*		Table Write	2 (5)	0000 0000 0000 1100	None	
TBLWT*+		Table Write with post-increment		0000 0000 0000 1101	None	
TBLWT*-		Table Write with post-decrement		0000 0000 0000 1110	None	
TBLWT+*		Table Write with pre-increment		0000 0000 0000 1111	None	

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is ’1’ for a pin configured as input and is driven low by an external device, the data will be written back with a ’0’.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are 2-word instructions. The second word of these instructions will be executed as a `NOP`, unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

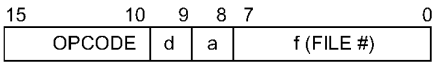
5: If the Table Write starts the write cycle to internal memory, the write will continue until terminated.

Field	Description
a	RAM access bit a = 0: RAM location in Access RAM (BSR register is ignored) a = 1: RAM bank is specified by BSR register
bbb	Bit address within an 8-bit file register (0 to 7)
BSR	Bank Select Register. Used to select the current RAM bank.
d	Destination select bit; d = 0: store result in WREG, d = 1: store result in file register f.
dest	Destination either the WREG register or the specified register file location
f	8-bit Register file address (0x00 to 0xFF)
fs	12-bit Register file address (0x000 to 0xFFF). This is the source address.
fd	12-bit Register file address (0x000 to 0xFFF). This is the destination address.
k	Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value)
label	Label name
mm * *+ *- +*	The mode of the TBLPTR register for the Table Read and Table Write instructions. Only used with Table Read and Table Write instructions: No change to register (such as TBLPTR with Table Reads and Writes) Post-Increment register (such as TBLPTR with Table Reads and Writes) Post-Decrement register (such as TBLPTR with Table Reads and Writes) Pre-Increment register (such as TBLPTR with Table Reads and Writes)
n	The relative address (2’s complement number) for relative branch instructions, or the direct address for Call/Branch and Return instructions
PRODH	Product of Multiply high byte
PRODL	Product of Multiply low byte
s	Fast Call/Return mode select bit; s = 0: do not update into/from shadow registers s = 1: certain registers loaded into/from shadow registers (Fast mode)
u	Unused or Unchanged
WREG	Working register (accumulator)
x	Don't care (0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
TBLPTR	21-bit Table Pointer (points to a Program Memory location)
TABLAT	8-bit Table Latch
TOS	Top-of-Stack
PC	Program Counter
PCL	Program Counter Low Byte
PCH	Program Counter High Byte
PCLATH	Program Counter High Byte Latch
PCLATU	Program Counter Upper Byte Latch
GIE	Global Interrupt Enable bit
WDT	Watchdog Timer
$\overline{TO}$	Time-out bit
$\overline{PD}$	Power-down bit
C, DC, Z, OV, N	ALU status bits Carry, Digit Carry, Zero, Overflow, Negative
[ ]	Optional
( )	Contents
→	Assigned to
< >	Register bit field
∈	In the set of
italics	User defined term (font is courier)

GENERAL FORMAT FOR INSTRUCTIONS ของ PIC 18F442 และ 18F458

Byte-oriented file register operations

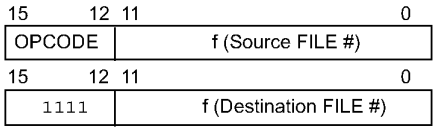
Example Instruction




```
ADDWF MYREG, W, B
```

d = 0 for result destination to be WREG register
d = 1 for result destination to be file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

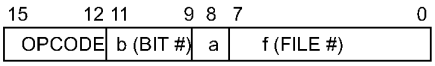
Byte to Byte move operations (2-word)




```
MOVFF MYREG1, MYREG2
```

f = 12-bit file register address

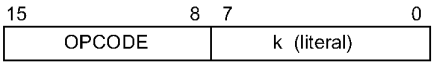
Bit-oriented file register operations




```
BSF MYREG, bit, B
```

b = 3-bit position of bit in file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

Literal operations

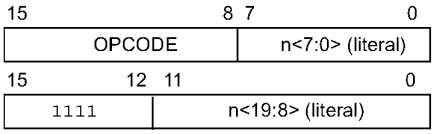



```
MOVLW 0x7F
```

k = 8-bit immediate value

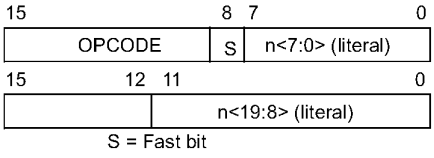
Control operations

CALL, GOTO and Branch operations



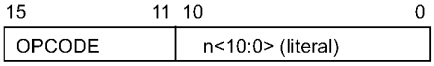

```
GOTO Label
```

n = 20-bit immediate value

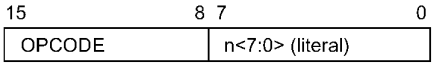



```
CALL MYFUNC
```

S = Fast bit




```
BRA MYFUNC
```




```
BC MYFUNC
```

2.0 OSCILLATOR CONFIGURATIONS

2.1 Oscillator Types

The PIC18FXX8 can be operated in one of eight Oscillator modes, programmable by three configuration bits (FOSC2, FOSC1, and FOSC0).

1. LP Low Power Crystal
2. XT Crystal/Resonator
3. HS High Speed Crystal/Resonator
4. HS4 High Speed Crystal/Resonator with PLL enabled
5. RC External Resistor/Capacitor
6. RCIO External Resistor/Capacitor with I/O pin enabled
7. EC External Clock
8. ECIO External Clock with I/O pin enabled

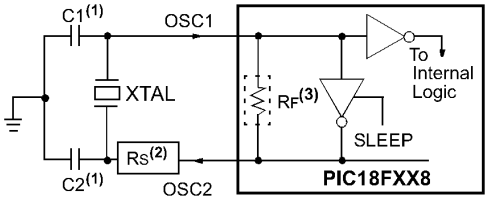
2.2 Crystal Oscillator/Ceramic Resonators

In XT, LP, HS or HS4 (PLL) Oscillator modes, a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation. Figure 2-1 shows the pin connections. An external clock source may also be connected to the OSC1 pin, as shown in Figure 2-3 and Figure 2-4.

The PIC18FXX8 oscillator design requires the use of a parallel cut crystal.

Note: Use of a series cut crystal may give a frequency out of the crystal manufacturer's specifications.

FIGURE 2-1: CRYSTAL/CERAMIC RESONATOR OPERATION (HS, XT OR LP OSC CONFIGURATION)



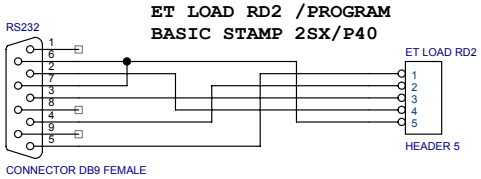
Note 1: See Table 2-1 and Table 2-2 for recommended values of C1 and C2.

2: A series resistor (RS) may be required for AT strip cut crystals.

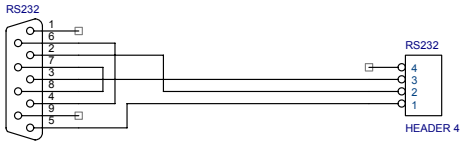
3: RF varies with the crystal chosen.

TABLE 2-1: CERAMIC RESONATORS

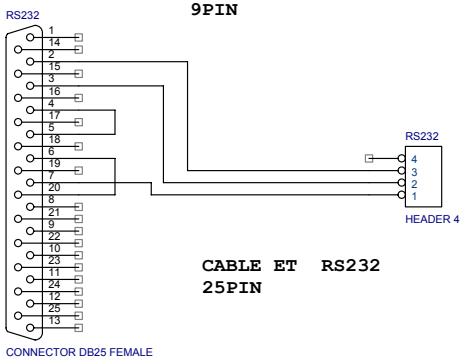
Ranges Tested:			
Mode	**Freq**	**OSC1**	**OSC2**
XT	455 kHz	68 - 100 pF	68 - 100 pF
	2.0 MHz	15 - 68 pF	15 - 68 pF
	4.0 MHz	15 - 68 pF	15 - 68 pF
HS	8.0 MHz	10 - 68 pF	10 - 68 pF
	16.0 MHz	10 - 22 pF	10 - 22 pF
	20.0 MHz	TBD	TBD
	25.0 MHz	TBD	TBD
HS+PLL	4.0 MHz	TBD	TBD
	8.0 MHz	10 - 68 pF	10 - 68 pF
	10.0 MHz	TBD	TBD
These values are for design guidance only. See notes following Table 2-2.			
Resonators Used:			
455 kHz	Panasonic EFO-A455K04B	±0.3%	
2.0 MHz	Murata Erie CSA2.00MG	±0.5%	
4.0 MHz	Murata Erie CSA4.00MG	±0.5%	
8.0 MHz	Murata Erie CSA8.00MT	±0.5%	
16.0 MHz	Murata Erie CSA16.00MX	±0.5%	
All resonators used did not have built-in capacitors.			



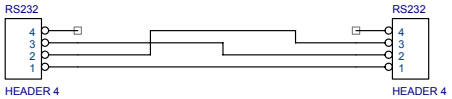
CABLE DOWNLOAD
ET LOAD RD2 /PROGRAM
BASIC STAMP 2SX/P40



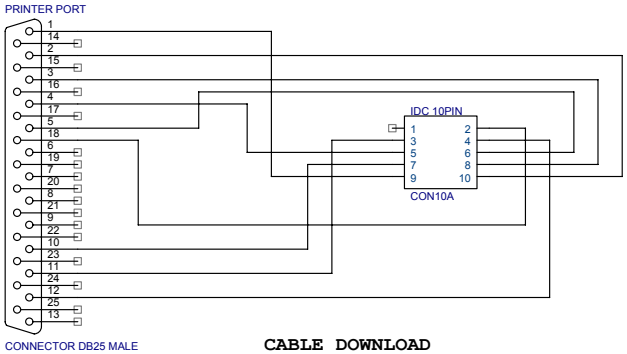
CABLE ET RS232
9PIN



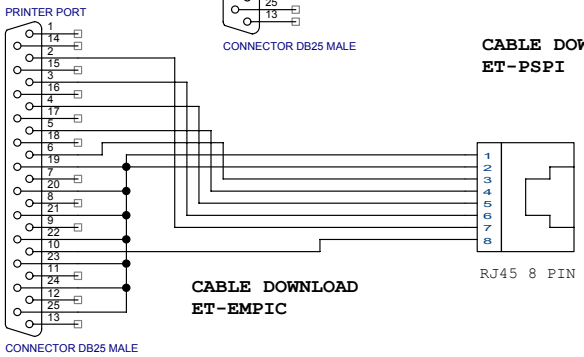
CABLE ET RS232
25PIN



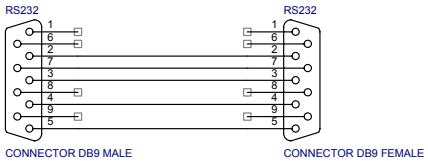
CABLE RS232 ET 4PIN TO ET 4PIN



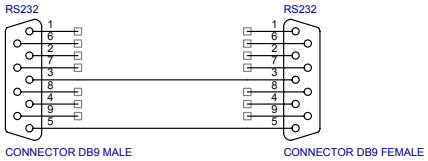
CABLE DOWNLOAD
ET-PSPI



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	