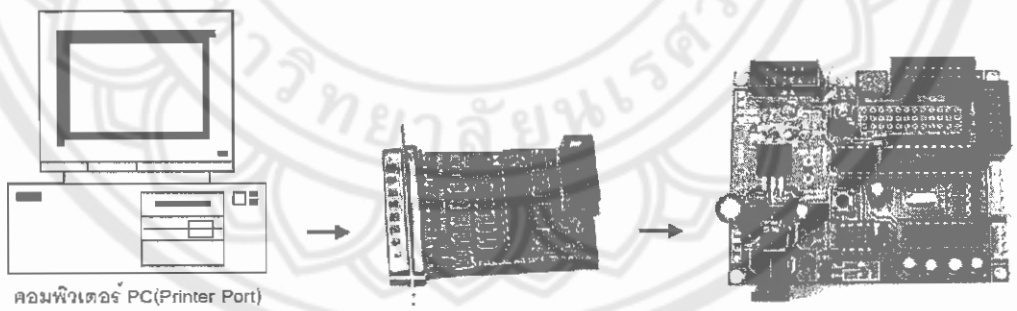
ภาคผนวก (ก)

คู่มือการติดตั้งและการใช้ Board

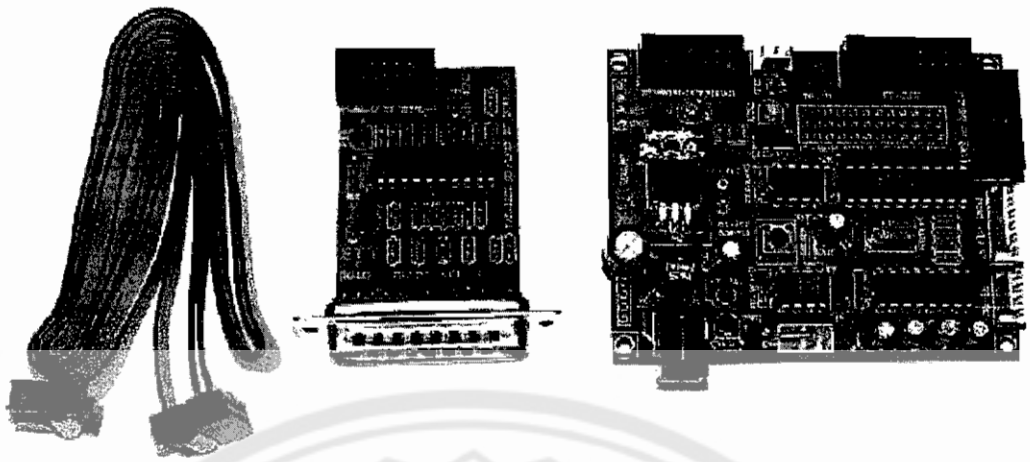
การพัฒนาโปรแกรมของบอร์ด ET-BASE LP4052 V1.0

ในการพัฒนาโปรแกรมของบอร์ด ET-BASE LP4052 V1.0 นั้น ตามปรกติจะแบ่งออกเป็น2 ส่วน คือ ส่วนของการเขียนโปรแกรมควบคุมการทำงานหรือ Monitor Program และการนำ Code ที่ได้จากการแปลคำสั่งของ Monitor Program ไปทำการ Download ให้กับหน่วยความจำ Flash ของ MCU โดยในส่วนของการเขียนโปรแกรมควบคุมการทำงานของบอร์ดนั้นจะมีวิธีการเหมือนกันกับการพัฒนาโปรแกรมของไมโครคอนโทรลเลอร์ MCS51 มาตรฐานทั่วๆไป เนื่องจาก AT89S52 นั้นจะใช้ชุดคำสั่งสำหรับสั่งงาน MCU เช่น เดียวกันกับ MCS51 มาตรฐานทุกประการ ซึ่งรายละเอียดส่วนนี้สามารถศึกษาได้จากคู่มือการใช้งานโปรแกรม Assembler หรือ Complier ที่จะใช้ในการพัฒนาโปรแกรม

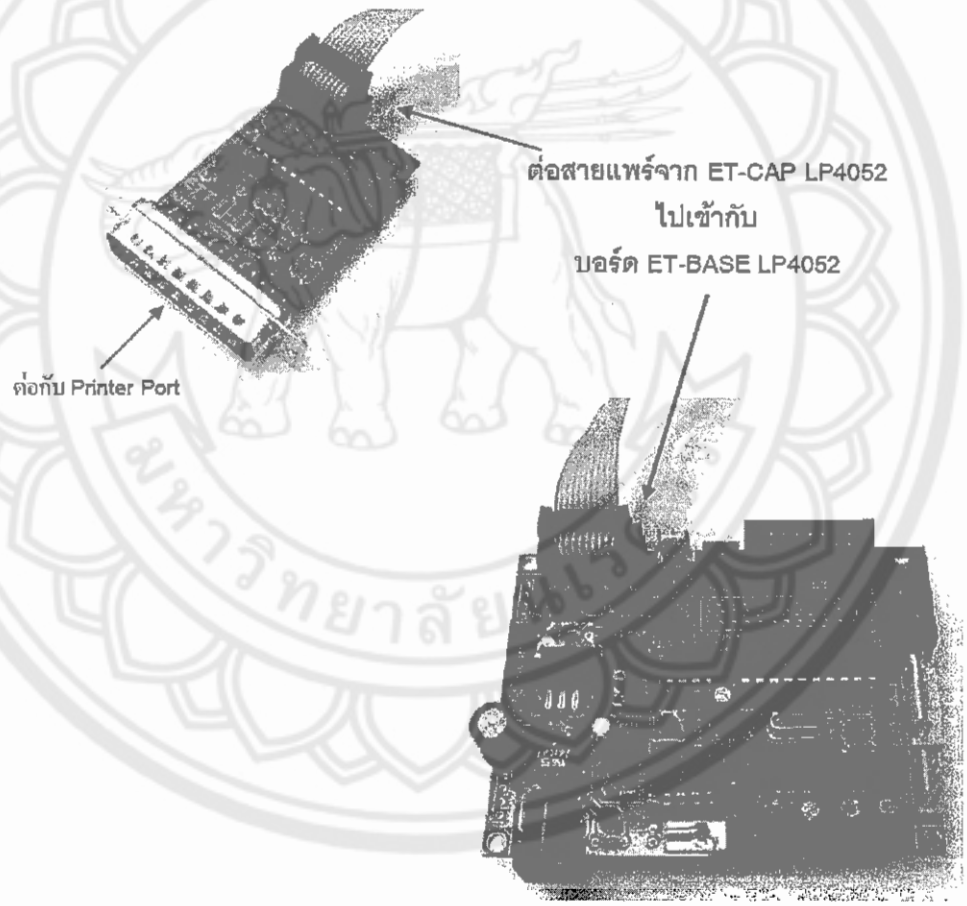
โดยในที่นี้จะขอกล่าวถึงเพียงวิธีการ Download Code โปรแกรมให้กับหน่วยความจำ Flash ของ MCU เท่านั้น ซึ่งบอร์ด ET-BASE LP4052 V1.0 นั้นได้จัดเตรียมขั้ว IDE-10PIN สำหรับทำการเชื่อมต่อสัญญาณเพื่อใช้ในการ Download Code ให้กับ MCU ไว้อยู่แล้ว ซึ่งในการDownload Code ให้กับหน่วยความจำ Flash ของ MCU นั้น สามารถกระทำได้ในขณะที่ MCUฝังตัวอยู่ในบอร์ดได้ทันที โดยไม่จำเป็นต้องถอด MCU ออกจากบอร์ด ให้เสียเวลา เพียงแต่มีข้อจำกัดว่าในขณะที่จะทำการ Download Code ให้กับ MCU นั้น จะต้องไม่นำขาสัญญาณ P1.4 ถึงP1.7 ของMCU ไปเชื่อมต่อกับอุปกรณ์อื่นๆไว้ เช่น LCD หรืออุปกรณ์ภายนอกอื่นๆที่เชื่อมต่อมายังขาสัญญาณทั้ง 4 เส้น ดังกล่าวข้างต้น ซึ่งถ้ามีจะต้องปลดออกให้เรียบร้อยเสียก่อนจึงจะทำการDownload แบบ In-System Serial Programming ได้โดยไม่เกิดปัญหา



รูปที่ ก.1 แผนผังการ Download Program ให้กับบอร์ด ET-Base LP4052 V1.0



รูปที่ ก.2 แสดงบอร์ด ET-CAP LP4052 และบอร์ด ET-BASE LP4052 V1.0



รูปที่ ก.3 แสดง การเตรียมการ Download โปรแกรมให้บอร์ด ET-BASE LP4052 V1.0

การ Download Code ให้ MCU ด้วย ET-CAP LP4052

ในการ Download Code (Hex File) ให้กับหน่วยความจำ Flash เบอร์ AT89S52 ของบอร์ด ET-BASE LP4052 V1.0 นั้น โดยมีลำดับขั้นตอนดังนี้

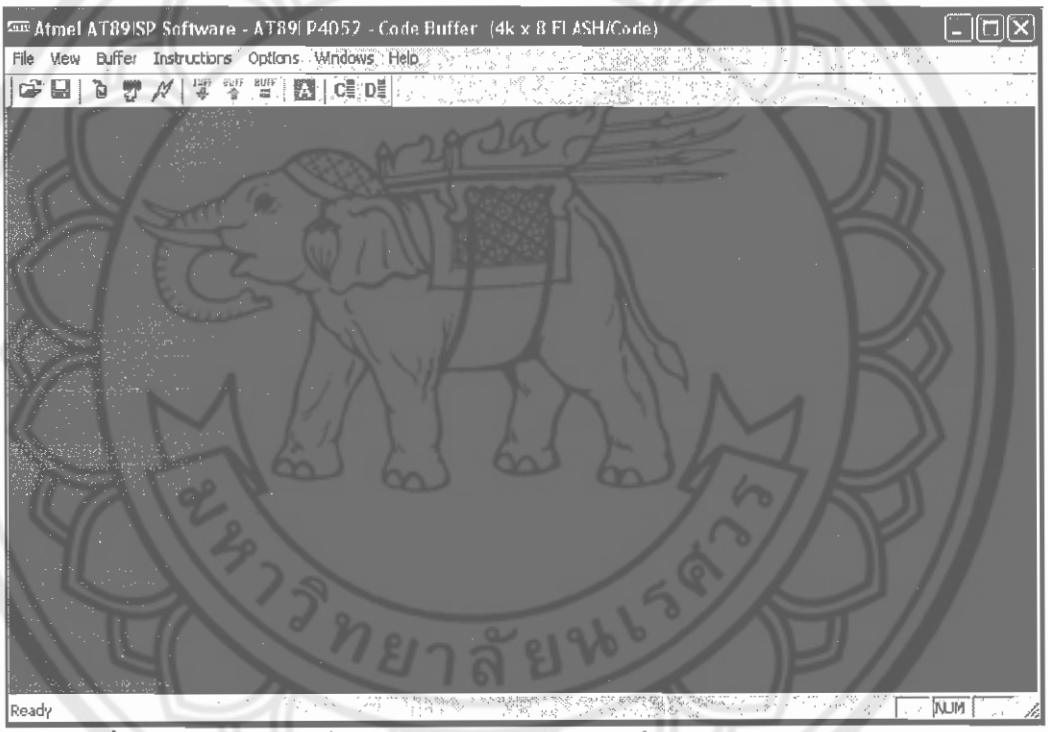
1. เชื่อมต่อชุด ET-CAP LP4052 เข้ากับพอร์ตขนาน (Printer Port) ของเครื่องคอมพิวเตอร์

2. ต่อสายแพร 10PIN จากชุด ET-CAP LP4052 เข้ากับขั้วต่อสำหรับ Download Codeของบอร์ด ET-BASE LP4052 (ขั้วต่อ ET-89LP-DOWNLOAD)

3. ปลดสายสัญญาณที่ทำการเชื่อมต่อ P1.4 ถึง P1.7 ของ MCU กับ LCD ทางขั้วต่อ ETCLCDหรืออุปกรณ์ภายนอกอื่นๆ ที่ต่อไว้ทางขั้วต่อ PORT-P1[0..7] ออกจากบอร์ดให้เรียบร้อย

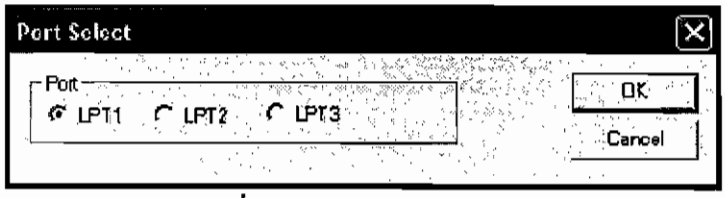
4. จ่ายไฟเลี้ยงวงจรให้กับบอร์ด ET-BASE LP4052 V1.0

5. สั่ง Run โปรแกรม AT89ISP เพื่อเตรียมทำการ Download Code ให้กับ MCU ดังรูป



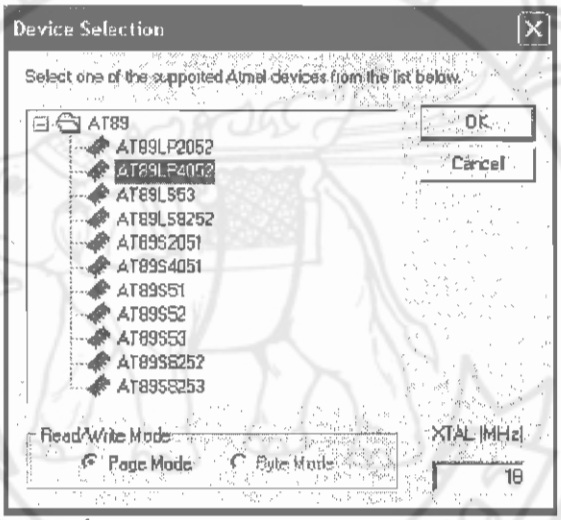
รูปที่ ก.4 แสดงหน้าการสั่ง Run โปรแกรม AT89ISP เพื่อเตรียมทำการ Download Code

5.1 คลิกเมาส์ที่เมนูคำสั่ง Options / Select Port เพื่อเลือกกำหนดหมายเลขของพอร์ตขนานที่ใช้ในการเชื่อมต่อชุด ET-CAP LP4052 ไว้ ซึ่งปรกติจะเป็น LPT1 เสมอ แต่ถ้าหมายเลขของพอร์ตขนานที่ใช้งานอยู่เป็น LPT อื่นๆ ก็ให้เลือกเปลี่ยนแปลงตามความเป็นจริง และเลือก OK ดังรูป



รูปที่ ก.5 แสดงหน้า Port Select

5.2 คลิกเมาส์ที่เมนูคำสั่ง Options / Select Device เพื่อเลือกกำหนดเบอร์ของ MCUที่ติดตั้ง ไว้ภายในบอร์ด ET-BASE LP4052 V1.0 ซึ่งปรกติจะเป็นเบอร์ AT89LP4052 แต่ถ้าเป็นเบอร์อื่นก็ให้เลือกกำหนดให้ถูกต้องตามความเป็นจริง พร้อมทั้งกำหนดค่า XTALเป็นเลขจำนวนเต็ม ซึ่งในกรณีของบอร์ด ET-BASE LP4052 V1.0 นั้นจะใช้ XTAL ค่า11.932 MHz ดังนั้นให้กำหนดค่า XTAL เป็น 12 และเลือก OK ดังรูปที่ ก.6

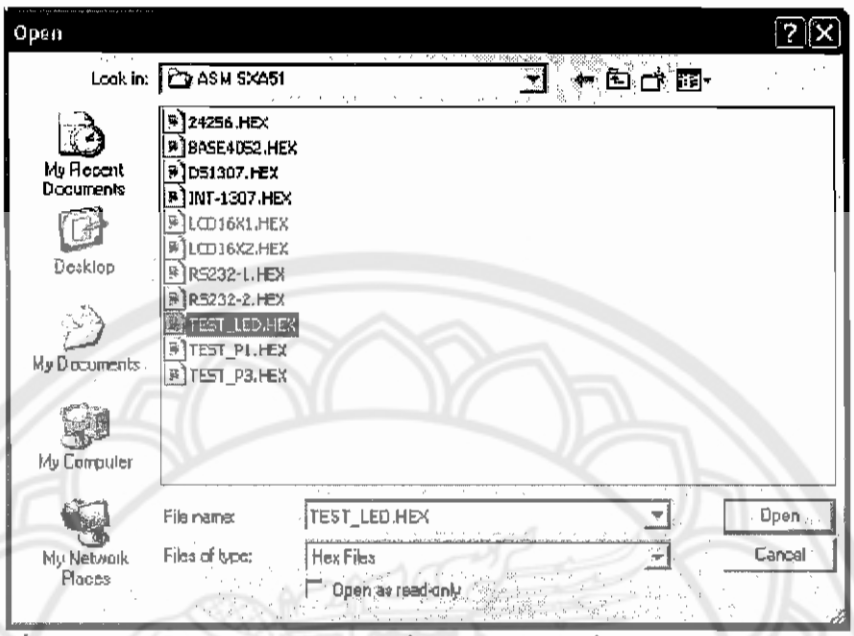


รูปที่ ก.6 แสดงการเลือกไมโครคอนโทรลเลอร์

5.3 คลิกเมาส์ที่เมนูคำสั่ง Options / Initialize Target เพื่อสั่งให้โปรแกรมส่งสัญญาณไปควบคุมการทำงานของ MCU ให้ทำงานในโหมด ISP Program และเลือก OK ดังรูปที่ ก.7

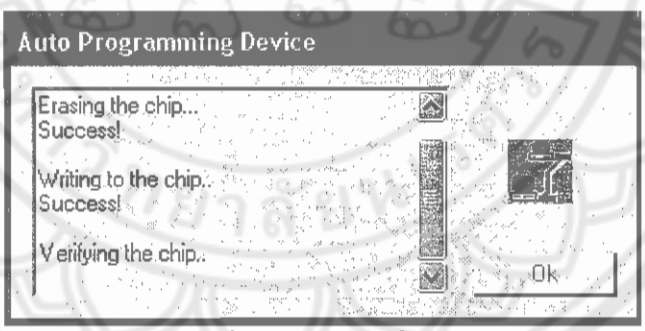
รูปที่ ก.7 การสั่งให้โปรแกรมส่งสัญญาณไปควบคุมการทำงานในโหมด ISP Program

5.4 คลิกเมาส์ที่เมนูคำสั่ง File / Load Buffer... เพื่อกำหนดตำแหน่งและชื่อของ HEX File ที่จะทำการ Download Code ไปให้กับ MCU ดังรูปที่ ก.8.



รูปที่ ก.8 แสดงการกำหนดตำแหน่งและชื่อของ HEX Fileที่จะทำการ Download Code

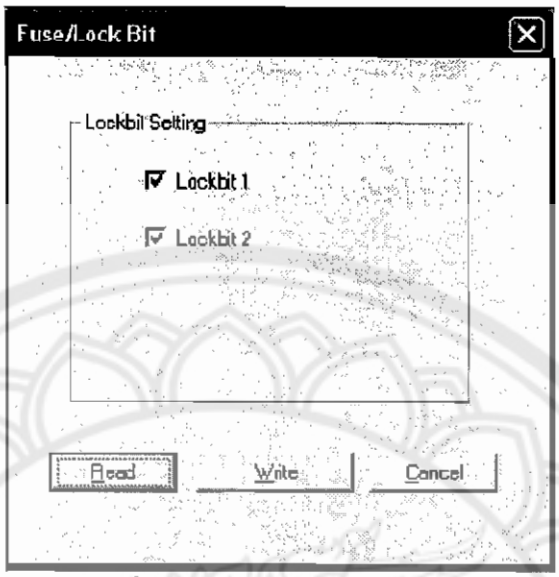
5.5 คลิกเมาส์ที่เมนูคำสั่ง Instructions / Auto program เพื่อสั่งให้โปรแกรมเริ่มต้นทำการ Download Code ให้กับ MCU ซึ่งจะได้ผลดังรูปที่ ก.9



รูปที่ ก.9 แสดงหน้าในการสั่งให้โปรแกรมเริ่มต้นทำการDownload Code

ในขั้นตอนนี้โปรแกรมจะทำการสั่ง Erase เพื่อทำการลบข้อมูลเดิมในหน่วยความจำของ MCU ออกพร้อมกับตรวจสอบข้อมูลในหน่วยความจำว่าว่างพร้อมที่จะทำการ Download Codeใหม่ลงไปหรือไม่ ซึ่งถ้าถูกต้องก็จะทำการ Download Code ที่อยู่ใน Buffer ให้กับ MCU พร้อมกับสั่ง Verify เพื่อตรวจสอบความถูกต้องของการ Download ซึ่งถ้ากระบวนการทั้งหมดเสร็จสมบูรณ์

โดยไม่เกิดความผิดพลาดใดๆ โปรแกรมจะรอให้ผู้ใช้ทำการสั่งกำหนดระดับการ Lock Bit เพื่อกำหนดระดับการป้องกันการอ่านข้อมูลจากตัว MCU ดังรูปที่ ก.10

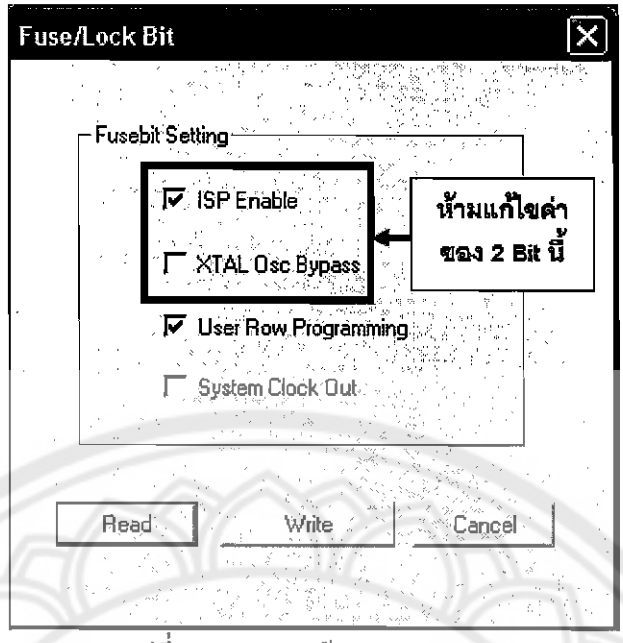


รูปที่ ก.10 แสดงหน้า Fuse/Lock Bit

โดยในขั้นตอนนี้ให้ผู้ใช้ทำการเลือกกำหนดระดับการ Lock Bit ได้ตามต้องการ โดยถ้าต้องการเลือก Lock Bit ใดก็ให้คลิกเมาส์ ที่หน้าตัวเลือกนั้นจนได้เครื่องหมายถูกที่หน้าตัวเลือกนั้นๆ โดยในกรณีที่ใช้ MCU เบอร์ AT89LP4052 จะสามารถเลือกการ Lock Bit ได้ 2 ระดับคือ

- Lock Bit1 ซึ่งจะทำให้ไม่สามารถสั่ง Download Code ให้หน่วยความจำ Flash ซ้ำใหม่ได้อีกจนกว่าจะสั่ง Erase เสียก่อน แต่การ Lock Bit1 ยังสามารถสั่งอ่าน Code จากหน่วยความจำของ MCU กลับออกมาได้อยู่

- Lock Bit2 จะทำให้ไม่สามารถเขียนและอ่าน Code จากหน่วยความจำ Flash ของMCU ได้อีก จนกว่าจะสั่ง Erase เสียก่อน ซึ่งควรเลือกระดับการ Lock Bit เป็น Lock Bit2 เสมอเพื่อป้องกันไม่ให้ผู้อื่นนำ MCU ของเราไปสั่งอ่าน Code มาใช้งานได้อีกต่อไป โดยในกรณีที่ต้องการสั่ง Lock Bit2 นั้น จะต้องเลือกเครื่องหมายถูกที่หน้าตัวเลือก Lock Bit1 และ Lock Bit2 ด้วยถ้าเลือกเครื่องหมายถูกที่หน้าตัวเลือก Lock Bit2 เพียงอย่างเดียวจะไม่เป็นผลใดๆต่อ Lock Bit เลย



รูปที่ ก.11 แสดงหน้า Fusebit Setting

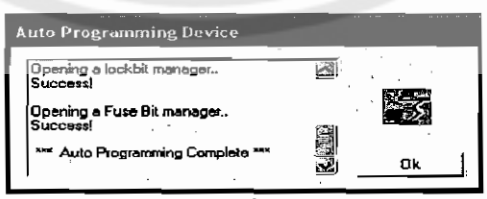
ในขั้นตอนนี้จะเป็นการกำหนดค่าของ Fuse Bit ซึ่งสามารถ เลือกกำหนดค่า Fuse Bit ได้ดามต้องการ โดยถ้าเลือกเครื่องหมายถูก (/) หน้าตัวเลือกใดจะเป็นการเลือก Enable ค่า Fuse Bit นั้นๆ แต่ถ้าปล่อยว่างไว้จะเป็นการยกเลิกค่าตัวเลือกของ Fuse Bit นั้นๆ โดยค่าของ Fuse Bit ที่เลือกนี้จะมีผลก็ต่อเมื่อสั่ง Write แล้วเท่านั้น และเมื่อสั่ง Write ไปแล้วค่าของ Fuse Bit นั้นๆ จะไม่สามารถ แก้ไขได้อีกจนกว่าจะสั่ง Erase เสียก่อนโดยการสั่งเปลี่ยนแปลงค่า Fuse Bit ต่างๆเหล่านี้จะกระทำได้ก็ต่อเมื่อยังไม่ได้สั่ง Lock Bit ใดๆ เลย เท่านั้น ถ้ามีการสั่ง Lock Bit ไปแล้วจะไม่สามารถสั่งเปลี่ยน แปลงแก้ไขค่า Fuse Bit ใดๆได้ โดยในกรณีที่ใช้ MCU เบอร์ AT89LP4052 นั้นจะมี Fuse Bit ให้เลือกจำนวน 4 บิต คือ

- ISP Enable ใช้สำหรับเลือกว่าต้องการให้ MCU สามารถใช้วิธีการ Downloadหรือ Programข้อมูลด้วยวิธีการแบบ ISP ผ่านทางพอร์ต SPI ได้หรือไม่ ซึ่งตามปรกติแล้ว FuseBit นี้จะถูกสั่ง Enable มาจากโรงงานให้อยู่แล้ว เพื่อให้สามารถทำการ Download หรือ Programข้อมูลให้กับ MCU ด้วยวิธีการแบบ ISP โดยใช้ระบบ SPI ของ MCU ในการติดต่อสื่อสารกับ MCUได้ โดยบอร์ด ET-BASE LP4052 V1.0 ก็จะใช้การ Download ด้วยวิธีการนี้เช่นกัน ซึ่งถ้าสั่งยกเลิก Fuse Bit นี้ออก จะทำให้ไม่สามารถสั่ง Download ข้อมูลให้กับ MCU ด้วยวิธีการนี้ได้อีกในครั้งต่อไป จนกว่าจะนำ MCU ไปสั่ง Enable ค่า Fuse Bit นี้กลับคืนมาใหม่ ซึ่งต้องใช้เครื่องโปรแกรม MCU แบบ Parallel เป็นตัวจัดการ ดังนั้นขอแนะนำไม่ให้สั่งยกเลิกค่าของ Fuse Bit นี้ออกเป็นอันขาด มิฉะนั้นแล้วจะไม่สามารถ Download ข้อมูลได้อีก

- XTAL Osc Bypass ใช้สำหรับเลือกกำหนดแหล่งสัญญาณนาฬิกาที่จะป้อนให้กับ MCU โดยถ้าเลือก Enable บิตนี้จะเป็นการสั่งยกเลิกการทำงานของวงจรกำเนิดสัญญาณนาฬิกาภายในตัว MCU ออก เพื่อใช้งานกับแหล่งกำเนิดสัญญาณแบบ Oscillator จากภายนอก โดยต้องป้อนสัญญาณนาฬิกาจากแหล่งกำเนิดภายนอกให้กับขา XTAL1 ส่วนขา XTAL ให้ปล่อยลอยไว้ ซึ่งถ้าสั่งเลือก Fuse Bit นี้จะทำให้ MCU ไม่สามารถใช้งานกับวงจรกำเนิดสัญญาณนาฬิกาที่ใช้กับXTALได้ ซึ่งในกรณีของบอร์ด ET-BASE LP4052 V1.0 นั้นจะออกแบบวงจรให้ MCU ใช้แหล่งกำหนดสัญญาณนาฬิกาจากภายในตัว MCU โดยต่อ XTAL จากภายนอกให้กับขาสัญญาณ XTAL1 และXTAL2 ของ MCU ดังนั้นจึงห้ามไม่ให้เลือก Fuse Bit นี้เป็นอันขาด มิฉะนั้นแล้วจะทำให้ MCU ไม่สามารถทำงานได้อีก เนื่องจากวงจรกำเนิดสัญญาณนาฬิกาหยุดทำงาน ซึ่งจะต้องนำสัญญาณนาฬิกามาป้อนให้กับขา XTAL ของ MCU แทน หรือนำ MCU ไปทำการสั่งยกเลิก Fuse Bit นี้ออกด้วยเครื่องโปรแกรมแบบ Parallel เสียก่อน

- User Row Programming ถ้าสั่ง Enable ค่า Fuse Bit นี้จะใช้สำหรับเลือก Enable การเขียนข้อมูลให้กับหน่วยความจำพิเศษ ซึ่งเรียกว่า "User Signature Row"ภายในตัว MCU โดยถ้าเลือก Fuse Bit นี้จะทำให้สามารถสั่งเขียนข้อมูลให้กับหน่วยความจำส่วนนี้ได้ด้วย เพียงแต่ว่าในโปรแกรม ATMEL AT89ISP V2.2 ยังไม่รองรับฟังก์ชั่นการเขียนข้อมูลให้กับหน่วยความจำส่วนนี้ปัจจุบันทำได้เฉพาะการอ่านข้อมูลจาก "User Signature Row" เท่านั้น โดยเลือกจากคำสั่ง Instructions /Check Signature / Read user Signature Row

- System Clock Out ใช้สำหรับสั่งให้ MCU เชื่อมต่อสัญญาณนาฬิกาของระบบ ออกไปทางขาสัญญาณ P3.7 (SYSCLK) ด้วย โดยถ้าเลือก Enable ค่าของ Fuse Bit นี้จะทำให้สัญญาณนาฬิกา ของระบบถูกเชื่อมต่อเข้ากับขาสัญญาณ P3.7 โดยค่าความถี่ของสัญญาณนาฬิกาจะมีค่าความถี่เหมือนกับที่ป้อนให้กับ MCU ทุกประการ เพียงแต่สัญญาณนาฬิกาจะผ่านวงจร Buffer ด้วยเพื่อให้สามารถนำไปใช้ขับวงจรอื่นๆได้ โดยผู้ใช้จะต้องเขียนโปรแกรมเพื่อสั่งกำหนดสถานของ P3.7ให้เป็นแบบ Push-Pull Output Mode ด้วยดังนั้นในขั้นตอนนี้เมื่อค่าตัวเลือกของ "ISP Enable" ถูกเลือกไว้แล้ว พร้อมกับค่า FuseBit ของ "XTAL Osc Bypass" ถูกยกเลิกไว้แล้ว ขอแนะนำให้เลือก Cancel เพื่อข้ามไปยังขั้นตอนถัดไปเลย โดยโปรแกรมจะแสดงผลการทำงานของคำสั่งเป็น Auto Programming Complete จากนั้นให้เลือก OK ดังรูปที่ ก.12



รูปที่ ก.12 แสดงผลการทำงานของคำสั่งเป็น Auto Programming Complete

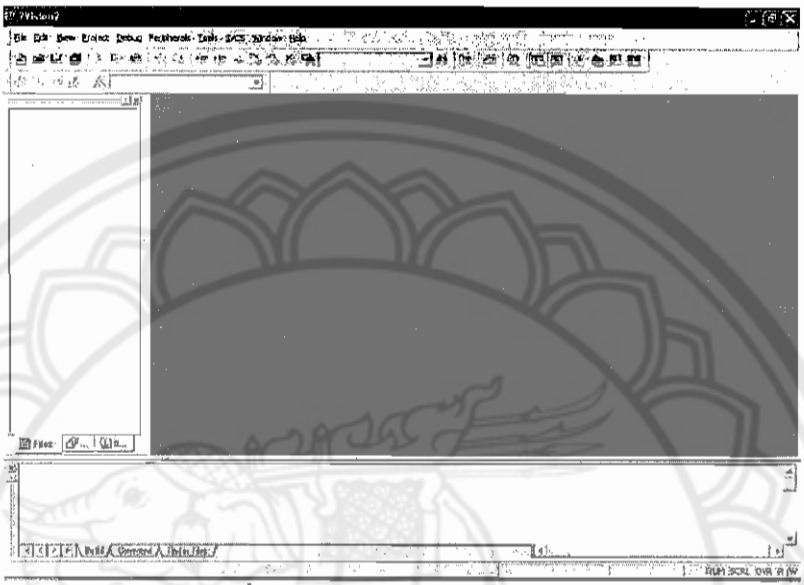
5.6 คลิกเมาส์ที่เมนูคำสั่ง Instructions / Run Target เพื่อสั่งให้ MCU เริ่มต้นทำงานตามคำสั่งใน Code ที่สั่ง Download ไป แล้วเลือก OKซึ่งหลังจากเลือก OK แล้วจะเห็นว่า MCU ในบอร์ดจะเริ่มต้นทำงานตาม Code คำสั่ง ที่ทำการสั่ง Download ไปแล้วทันที ซึ่งจากตัวอย่างจะเป็นการสั่งโหลดโปรแกรมตัวอย่างสำหรับทดสอบการทำงานของบอร์ด ชื่อ TEST_LED.HEX ซึ่งผลการทำงานของโปรแกรมนี้จะเป็นการสั่งให้ LED ที่ต่อไว้กับพอร์ต P3.7 ของ MCU ติด และ ดับสลับกันไปในลักษณะของไฟกระพริบ แต่จะต้องเลือกกำหนด Jumper ของ LED/P3.7 ไว้ทางด้าน LED ด้วยจึงจะเห็นการกระพริบของ LED ดังกล่าว

ภาคผนวก (ข)

การพัฒนาโปรแกรมภาษาซีของบอร์ด ET-BASE LP4052 ด้วย KEIL

การพัฒนาโปรแกรมภาษาซีของบอร์ดET-BASE LP4052 ด้วย KEIL

1. เปิดโปรแกรม Keil uVision ซึ่งเป็นโปรแกรม Text Editor ของ KEIL-C8051 ใช้สำหรับเขียน Source Codeภาษาซี ซึ่งถ้าใช้ V7.50A จะมีลักษณะดังรูปที่ ข.1

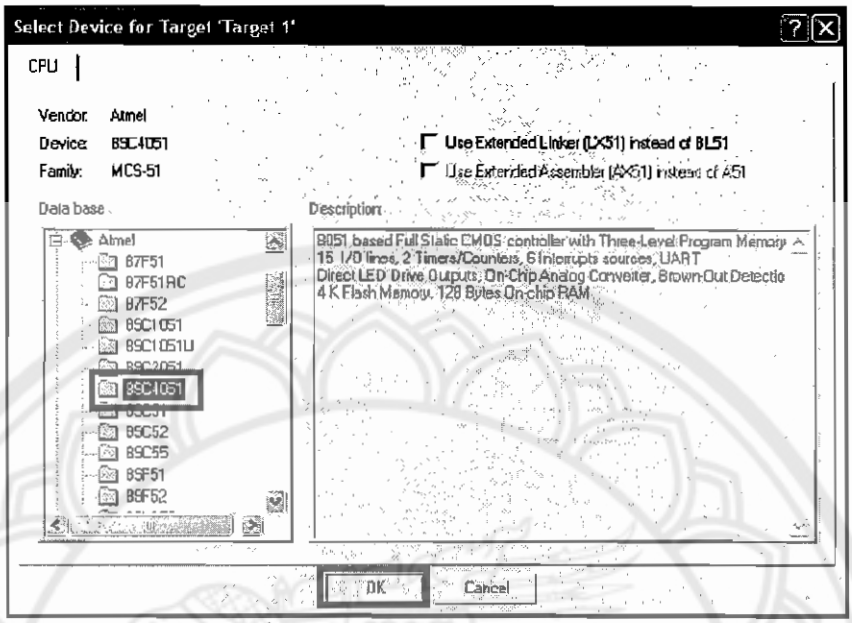


รูปที่ ข.1 หน้าโปรแกรม Keil uVision

2. ทำการสร้าง Project File ขึ้นมาใหม่ โดยเรียกเมนูคำสั่ง Project / New Project จากนั้นให้เลือกกำหนดหรือสร้างตำแหน่ง Folder ที่จะบันทึก Project File พร้อมกับกำหนดชื่อ Project File ตามต้องการ เช่น ถ้าต้องการสร้าง Project File ชื่อ EX1 โดยเก็บไว้ในFolder ชื่อ EX1 ก็สามารถกำหนดตำแหน่ง Folder และชื่อ Project File ได้เอง โดยเมื่อกำหนดชื่อในช่อง File name แล้วให้เลือก Save เพื่อบันทึก Project File ไว้ ดังรูปที่ ข.2

รูปที่ ข.2 ทำการสร้าง Project File ขึ้นมาใหม่

3. ให้ทำการกำหนด เบอร์ MCU ที่จะใช้งาน โดยเลือกจากเมนูคำสั่ง Project / Select Device For Target 'Target1'



รูปที่ ข.3 ให้ทำการกำหนด เบอร์ MCU

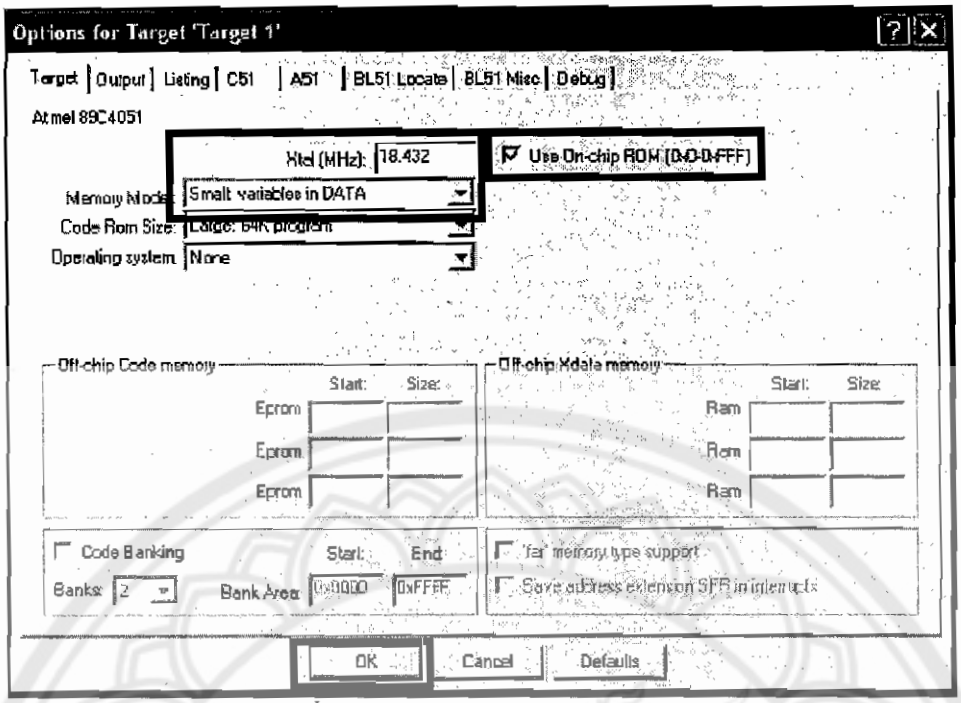
4. ให้ทำการกำหนดค่า Option ของ Project File โดยเลือกเมนูคำสั่ง Project / Option for Target 'Target 1'จากนั้นเลือกที่ Tab ของ Target เพื่อกำหนดค่าของ MCU Target โดยให้กำหนดดังนี้

- XTAL ให้กำหนดเป็น 11.0592 MHz

- ขนาด ROM ให้เลือกใช้จากหน่วยความจำโปรแกรมภายในตัวชิพ โดยให้เลือกสั่ง Enable หัวข้อ Used On-Chip ROM (0x0-0xFFF) เพื่อกำหนดให้โปรแกรมทำการกำหนดขอบเขตของพื้นที่ในการแปล Code คำสั่งในช่วงหน่วยความจำดังกล่าว

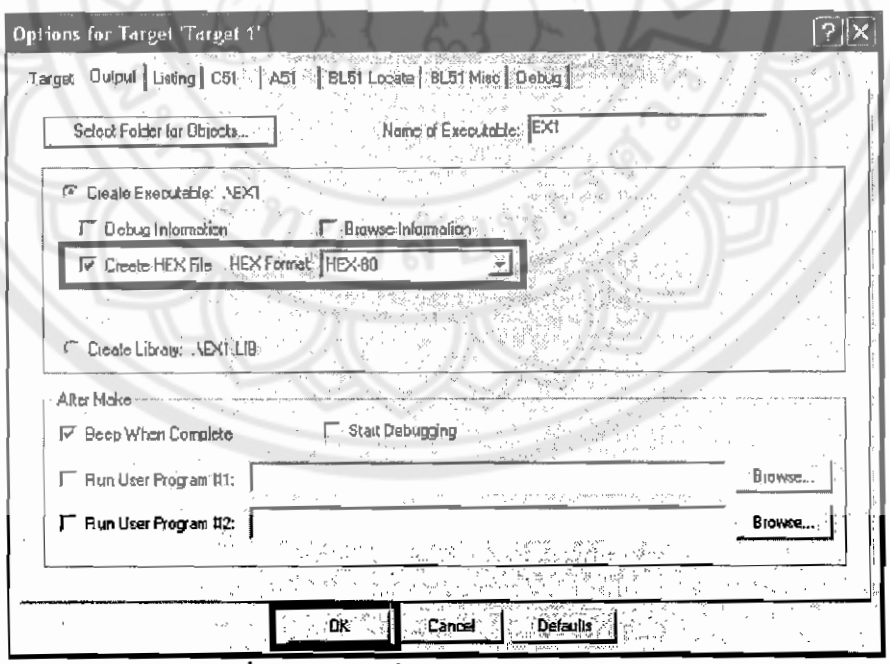
- Memory Model ให้เลือกเป็น Small: variables in DATA

- Code Rom Size: ไม่ต้องสนใจ ซึ่งในกรณีที่ใช้ Complier รุ่นที่เป็น Demo นั้นจะสามารถสั่งแปลโปรแกรมในขนาดที่ไม่เกิน 2KByte อยู่แล้ว



รูปที่ ข.4 การกำหนดค่า Option ของ Project File

จากนั้นให้เลือก Tab Output เพื่อกำหนดเงื่อนไขการสร้าง Output File โดยให้เลือก Name เป็น EX1 ตามชื่อ Source File พร้อมกับเลือกการ Create HEX File เป็น HEX-80 ส่วน Option อื่นๆ นอกจากนี้ไม่ต้องสนใจก็ได้ให้ใช้ค่าตาม Default ไปก่อน จากนั้นให้เลือก OK ดังรูปที่ ข.5

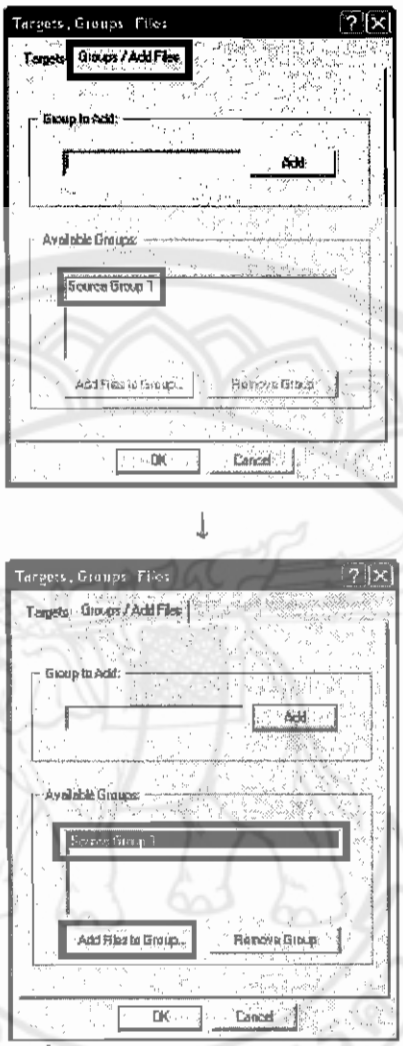


รูปที่ ข.5 กำหนดเงื่อนไขการสร้าง Output File

5. ให้สั่งสร้างไฟล์ใหม่ โดยเลือกที่เมนูคำสั่ง File / New... ซึ่งจะได้เป็นหน้ากระดาษเปล่าสำหรับเขียน Source Code โดยในครั้งแรกจะมีชื่อเป็น TEXTI ให้ทำการพิมพ์คำสั่งโปรแกรมภาษาซีตามต้องการในพื้นที่หน้ากระดาษสำหรับเขียนโปรแกรม ดังรูปที่ ข.6

รูปที่ ข.6 ทำการพิมพ์คำสั่งโปรแกรมภาษาซี

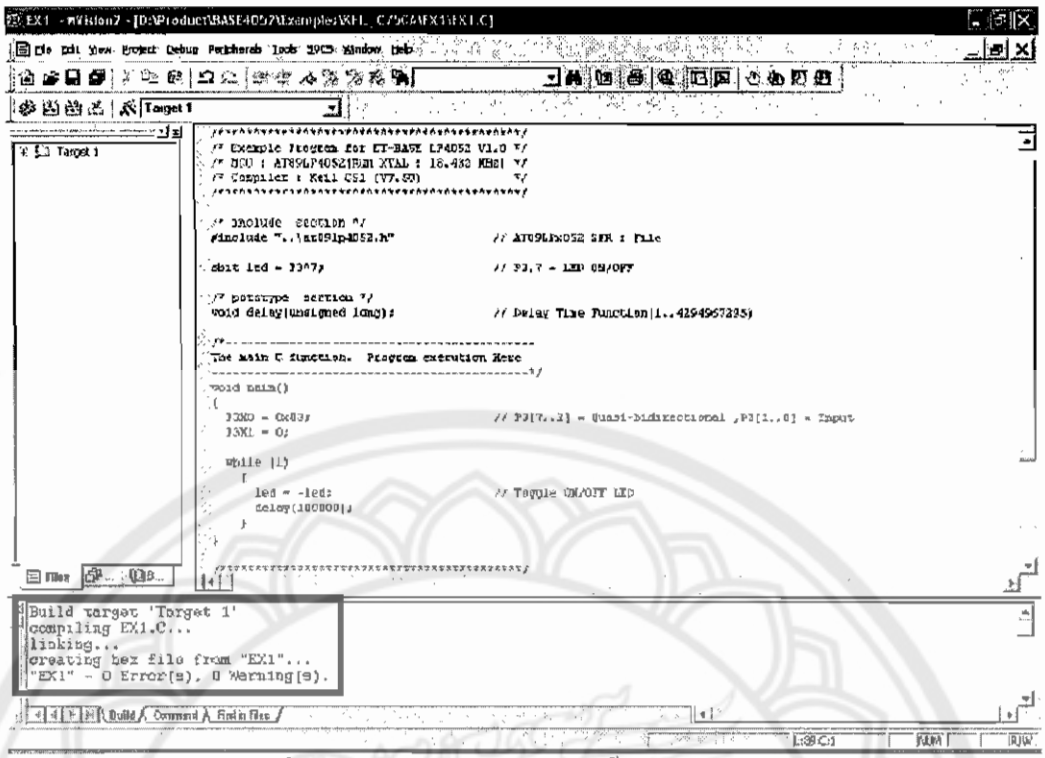
6. ทำการกำหนด Group File โดยคลิกเมาส์ที่คำสั่ง Project / targets, Groups, Files จากนั้นให้เลือกที่ Tabของ Groups/ Add Files แล้วคลิกที่ Source Group 1 ใน Available Groups: ดังรูป



รูปที่ ข.7 แสดงการกำหนด Group File

จากนั้นให้เลือกคลิกเมาส์ที่ Add Files to Group... แล้วเลือกคลิกเมาส์ที่ไอคอนของไฟล์ EX1.C เพื่อทำการสั่ง Add ไฟล์ดังกล่าวให้กับ Projcct File แล้วเลือก Close เพื่อจบการทำงาน

7. สั่งแปลโปรแกรมที่เขียนขึ้นให้เป็น Hex File โดยคลิกเมาส์ที่คำสั่ง Project/ Rebuild all target file ซึ่งถ้าไม่เกิดข้อผิดพลาดใดๆจะได้ผลการแปลเป็น 0 Error(s) และ 0 Warning(s) พร้อมกับได้ไฟล์ชื่อ EX1.HEX อยู่ใน Folder เดียวกันกับ EX1.C ซึ่งสามารถนำไปสั่ง Download ให้ MCU ด้วยโปรแกรม ATMELISP ได้ทันที ดังรูปที่ ข.9



รูปที่ ข.8 สั่งแปลโปรแกรมที่เขียนขึ้นให้เป็น Hex File

ภาคผนวก ค

Source Code โปรแกรมเครื่องควบคุมอุปกรณ์ด้วยรีโมทความถี่วิทยุ

Code ของตัวรับสัญญาณ TRW

```
#include <reg52.h>
#include <absacc.h>
#include <ctype.h>
#include <intrins.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
/*************************/
/***** Define I / O Port  *****/
/*************************/
#define Port_Test  P3
Sbit   Data       = P1^0;
Shit   CLK1       = P1^1;
Sbit   DR1        = P1^2;
Sbit   Dout2      = P1^3;
Sbit   CLK2       = P1^6;
Sbit   DR2        = P1^5;
Sbit   CS         = P1^4;
Sbit   CE         = P1^7;
Sbit   TX_LED   = P1^2;
Sbit   led        = P3^3;
```

```
/*************************/
/*Subrutine for Delay time */
/*************************/
void dmsec (unsigned int count)
{
   unsigned int i ;
    while (count) {
      i = 225; while (i>0) i--;
      count--;
  }
}
                        //=======================================//
void Wait (unsigned int x)
{
   unsinged int i ;
    for (i = 0 ; i<x; i++)
      {}
}
                        //=======================================//
void CLK_TRW24(void)
{
     CLK1 = 0;
      dmsec(1);
     CLK1 = 1;
      dmsec(1);
}
                        //=======================================//
```

```
void Write_TRW24 (unsigned char Dat)
{
    unsigned char i;
     bit Out;
     for (i = 0 ; i<8 ; i++)
     {
          Out  = Dat & 0x80;
          Data = Out ;
          CLK_TRW24() ;
          Data = Dat << 1 ;
     }
}
                              //=============================//
void SetMode_TRW24 (unsigned char Mode)
{
      Wait (500) ;
        CE = 0;
        CS = 1 ;
      Write_TRW24 (0x8E) ;
      Write_TRW24 (0x08) ;
      Write_TRW24 (0x1C) ;
      Write_TRW24 (0x08) ;
      Write_TRW24 (0xC0) ;
      Write_TRW24 (0xAA) ;
      Write_TRW24 (0x55) ;
      Write_TRW24 (0xAA) ;
      Write_TRW24 (0x55) ;
      Write_TRW24 (0x00) ;
      Write_TRW24 (0x00) ;
      Write_TRW24 (0x00) ;
      Write_TRW24 (0x00) ;
```

```
        Write_TRW24 (0x00) ;
        Write_TRW24 (0xA3) ;
        Write_TRW24 (0x6F) ;
        if (MOde == 1)
        {
           Write_TRW24 (0x14) ;
        }
        else
        {
           Write_TRW24 (0x15) ;
           Data=1 ; DR1=1; CE=1;
        }
           CS = 0 ;
           Wait (500) ; }
                        //=============================================//
unsigned char Read_TRW24 (void)
{
    unsigned char i , Temp ;
    bit Out ;
     Data = 1 ;
    for ( i = 0 ; i < 8 ; i++)
    {
        Temp = Temp << 1 ;
        CLK = 1 ;
        Wait (50) ;
        Out = Data ;
        if (Out) { Temp = Temp + 0x01;}
                        CLK = 0 ;
                        Wait (50);
    }
}
```

```
/*******************************/
/******** Main Program *********/
/*******************************/
void main (void)
{
    unsigned char DATA ;
      TX_LED = 0;
        dmsec (750) ;
      TX_LED = 1 ;
        dmsec (750) ;
      TX_LED = 0 ;
        dmsec (750);
      TX_LED = 1 ;
        Data = 0 ;
        Dout2 = 0 ;
        DR1 = 0 ;
        DR2 = 0 ;
      CLK2 = 0 ;
      CLK1 = 0 ;
        CE = 0 ;
        CS = 0 ;
        SetMode_TRW24 (0) ;
        while (1)  {
            while (DR == 0);
            P3 = DATA;
                    }
}
```

Code ของตัวส่งสัญญาณ TRW

```
#pragma code
#include <reg51.h>
#include <absacc.h>
#include <ctype.h>
#include <intrins.h>
#include <math.b>
#include <stdarg.b>
#include <stdio.h>
#include <stdlib.b>
/*******************************/
/********  Define I/O Port  *******/
/*******************************/
#define Port_Test P3
    sbit Data   = P1^6;          // Port send data to module 2.4G chanel 1
    sbit CLK1   = P1^5;          // Port Clock befor send data chanel 1
    sbit DR1    = P1^7;          // port select data to cbanel1
    sbit CS     = P1^4;          // Port chip select
    sbit CE     = P1^3;          // Port Enable Shift
sbit TX_LED = P3^0;
sbit    sw1 = P1^2;
sbit    sw2 = P1^1;
sbit    sw3 = P1^0;
sbit    sw4 = P3^7;
sbit    sw5 = P3^5;
sbit    sw6 = P3^4;
sbit    sw7 = P3^3;
sbit    sw8 = P3^2;
```

```
/**********************************/
/*       Subrutine for Delay time         */
/**********************************/
void dmsec (unsigned int count) {              // mSec Delay 11.0592 Mhz
  unsigned int i;                              // Keil v7.08
    while (count) {
       i = 225; while (i>0) i--;
    count--;
  }
}
                          //═══════════════════════════════//
void Wait(unsigned int x)
{
  unsigned int i;
    for (i=0;i<x;i++)
       {}
}
                          //═══════════════════════════════//
void CLK_TRW24(void)
{
    CLK1 = 0;
       dmsee(1);
    CLK1 = 1;
       dmsec(1);
}
                          //═══════════════════════════════//
```

```
void Write_TRW24(unsigned char Dat)
{
    unsigned char i;
      bit Out;
        for (i=0;i<8;i++)
        {
                Out = Dat & 0x80;
                Data = Out;
                CLK_TRW24();
                Dat = Dat << 1;
        }
}
                        //=====================================//
void SetMode_TRW24( unsigned char Mode)
{
  Wait(500);
     CE = 0;
     CS = 1;
        Write_TRW24(0x8E);   /* Reserved for testing */
        Write_TRW24(0x08);   /* Reserved for testing */
        Write_TRW24(0x1C);   /* Reserved for testing */
        Write_TRW24(0x08);   /* Length of Bit Ch 2 */
        Write_TRW24(0x08);   /* Length of Bit Ch 1 */
        Write_TRW24(0xC0);   /* Address 5 Byte Ch 2 */
        Write_TRW24(0xAA);
        Write_TRW24(0x55);
        Write_TRW24(0xAA);
        Write_TRW24(0x55);
        Write_TRW24(0xAA);   /* Address 5 Byte Ch 1 */
        Write_TRW24(0x55);
        Write_TRW24(0xAA);
```

```
        Write_TRW24(0x55);
        Write_TRW24(0xAA);
        Write_TRW24(0xA3);  /* Number of Address bit + CRC */
        Write_TRW24(0x4F);  /* RF Programming */
            if (Mode == 1)        /* Tx Mode */
        {
            Write_TRW24(0x14);     // Data=1; DR1=1; CE=1;
        }
                    else            /* Rx_Mode */
        {
            Write_TRW24(0x15);
              Data=1; DR1=1; CE=1;
        }
            CS = 0;
                Wait(200);
}
                        //=============================================//

void send_TRW24(unsigned char dat)
{
  Wait(500);
    CS = 0;
    CE = 1;
      Write_TRW24(0xAA);   // address 5 byte send
      Write_TRW24(0x55);   // address 5 byte send
      Write_TRW24(0xAA);   // address 5 byte send
      Write_TRW24(0x55);   // address 5 byte send
      Write_TRW24(0xAA);   // address 5 byte send
      Write_TRW24(dat);    // data
```

```
  Wait(250);
     CLK1 = 0;
     CE =0;
        Wait(250);
}
/*******************************/
/******** Main Program ********/
/*******************************/
void main (void)
{
  dmsee(500);
     TX_LED = 0;
   dmsec(500);
     TX_LED = 1;
        sw1=1;
        sw2=1;
        sw3=1;
        sw4=1;
        sw5=1;
        sw6=1;
        sw7=1;
        sw8=1;
        Data  = 0;
        DR1 = 0;
        CLK1= 0;
        CE   = 0;
        CS   = 0;
    dmsec(250);
      SetMode_TRW24(1);        // mode 1 to send mode
```

```
dmsec(250);
  while(1)
   {
      if(sw2==0)
       { dmsec(50);
            if(sw2==0)
       { TX_LED = 0;
            send_TRW24(0x01);
            while(sw2==0);
            TX_LED = 1;}
       }
       if(sw1==0)
       { dmsec(50);
            if(sw1==0)
       {  TX_LED = 0;
            send_TRW24(0x02);
            while(sw1==0);
            TX_LED = 1;}
       }
       if(sw4==0)
       { dmsec(50);
            if(sw4==0)
       { TX_LED = 0;
            send_TRW24(0x03);
            while(sw4==0);
            TX_LED = 1;} }
       if(sw3==0)
       { dmsec(50);
            if(sw3==0)
       {TX_LED = 0;
```

```
        send_TRW24(0x04);
        while(sw3==0);
        TX_LED = 1;}
}
if(sw5==0)
{ dmsec(50);
        if(sw5==0)
{TX_LED = 0;
        send_TRW24(0x08);
        while(sw5==0);
        TX_LED = 1;}
}
if(sw6==0)
{ dmsec(50);
        if(sw6==0)
{ TX_LED = 0;
        send_TRW24(0x07);
        while(sw6==0);
        TX_LED = 1;}
}
if(sw7==0)
{ dmsec(50);
        if(sw7==0)
{ TX_LED = 0;
        send_TRW24(0x06);
        while(sw7==0);
        TX_LED = 1;}
}
if(sw8==0)
{ dmsec(50);
      if(sw8==0)
```

```
        { TX_LED = 0;
            send_TRW24(0x05);
            while(sw8==0);
            TX_LED = 1;}   }
        }
}
```

Code ของ Micro Controller

```
#include <REGX51.H>
#include<i2c_1.h>
#include<lcd.h>
#define DS1307_ID 0xD0
sbit A1 = P2^4;
sbit A2 = P2^5;
sbit A3 = P2^6;
sbit   relay1 = P0^0;
sbit   relay2 = P0^1;
sbit   relay3 = P0^2;
sbit   relay4 = P0^3;
sbit   relay5 = P0^4;
sbit   relay6 = P0^5;
sbit   relay7 = P0^6;
sbit   relay8 = P0^7;
sbit   remote_in = P3^4;
sbit   watch_dog = P3^5;
 bit   st_relay0;
 bit   st_relay1;
 bit   st_relay2;
 bit   st_relay3;
 bit   st_relay4;
 bit   st_relay5;
 bit   st_relay6;
 bit   st_relay7;
 int time = 0;

void  service(void)  interrupt 1
{
      TH0 = 0xDC;
```

```
        TL0 = 0x00;
        TF0=0;
            time++;
                if(time>2)
                    {    time=0;
            watch_dog=~watch_dog; }
}
 void delay_db(int time)
{
        do
       {  time--;
        }while(time>0);
}
char scan_key(void)
{
        char ret=0xFF;
        char out = 0xFF;
           P2 = 0xFE;
                if(A1==0)
        {
           delay_db(20000);
                 ret=0x01;
                    goto end;
                                }
       if(A2==0)
          {
           delay_db(20000);
                 ret=0x02;
                      goto end;
                                }
        if(A3==0)
```

```
    {
    delay_db(20000);
          ret=0x03;
               goto end;
                    }
P2 = 0xFD;
if(A1==0)
  {
  delay_db(20000);
  ret=0x04;
       goto end;
                    }
if(A2==0)
  {
   delay_db(20000);
        ret=0x05;
             goto end;
                    }
 if(A3==0)
  {
   delay_db(20000);
         ret=0x06;
              goto end;
                    }
P2 = 0xFB;
 if(A1==0)
 {
  delay_db(20000);
          ret=0x07;
               goto end;
                    }
```

```
if(A2==0)
  {
   delay_db(20000);
        ret=0x08;
              goto end;   }
 if(A3==0)
  {
  delay_db(20000);
        ret=0x09;
            goto end;
                  }
P2 = 0xF7;
 if(A1==0)
  {
   delay_db(20000);
        ret=0x0A;
            goto end;
                  }
if(A2==0)
  {
   delay_db(20000);
        ret=0x00;
             goto end;
                  }
 if(A3==0)
  {
   delay_db(20000);
        ret=0x0B;
             goto end;
                  }
end:
```

```
        return(ret);
    }
Unsigned char DS1307_rd(unsigned char addr)
    {
        unsigned char ret=0;            // For keep return value
        i2c_start();                    // i2c start condition
        i2c_write(DS1307_ID);           // Write DS1307 ID for connect
        i2c_write(addr);                // Write RAM address on DS1307 for conn
        i2c_start();                    // i2c start condition
        i2c_write(DS1307_ID+1)          // Write DS1307 ID for Read Mode connect
        ret = i2c_read();               // Read data and keep to ret
        i2c_stop();                     // i2c stop condition
        return(ret);                    // return value
    }
    /************************************************/
    /***** Function write hour/min/sec on chip DS1307 ******/
    /************************************************/
    void DS1307_wr (unsigned char addr,unsigned char sw)
    {
        i2c_start();                    // i2c start condition
        i2c_write(DS1307_ID);           // Write DS1307 ID for connect
        i2c_write(addr);                // Write control byte to access RAM address 00H
        i2c_write(sw);                  // Write sec on RAM address 00H
        i2c_stop();                     // i2c stop condition
    }
    void rd_logic(void)
        { unsigned  char buff[8];
    buff[0] = DS1307_rd(0x08);
    buff[1] = DS1307_rd(0x09);
    buff[2] = DS1307_rd(0x0A);
    buff[3] = DS1307_rd(0x0B);
```

```
buff[4] = DS1307_rd(0x0C);
buff[5] = DS1307_rd(0x0D);
buff[6] = DS1307_rd(0x0E);
buff[7] = DS1307_rd(0x0F);
    if (buff[0] == 0x0A) {  st_relay0 = 0;
lcd_put(0x84,"ON ");
     }
   else { st_relay0 = 1;
       lcd_put(0x84,"OFF");
            }
    if (buff[1] == 0x0A) {  st_relay1 = 0;
       lcd_put(0xC4,"ON ");
    }
    else { st_relay1 = 1;
       lcd_put(0xC4,"OFF");
     }
    if (buff[2] == 0x0A) {  st_relay2 = 0;
       lcd_put(0x94,"ON ");
     }
    else { st_relay2 = 1;
       lcd_put(0x94,"OFF");
    }
    if (buff[3] == 0x0A) {  st_relay3 = 0;
        lcd_put(0xD4,"ON");
     }
    else { st_relay3 = 1;
        lcd_put(0xD4,"OFF");
     }
    if (buff[4] == 0x0A) {  st_relay4 = 0;
        lcd_put(0x8D,"ON ");
     }
```

```
        else { st_relay4 = 1;
lcd_put(0x8D,"OFF");
          }
        if (buff[5] == 0x0A) {  st_relay5 = 0;
           lcd_put(0xCD,"ON ");
         }
        else { st_relay5 = 1;
          lcd_put(0xCD,"OFF");
         }
       if (buff[6] == 0x0A) {  st_relay6 = 0;
          lcd_put(0x9D,"ON ");
         }
       else { st_relay6 = 1;
          lcd_put(0x9D,"OFF");
         }
        if (buff[7] == 0x0A) {  st_relay7 = 0;
           lcd_put(0xDD,"ON ");
         }
        else { st_relay7 = 1;
           lcd_put(0xDD,"OFF");
         }
        }
    void set_relay(void) {    relay1 = st_relay0;
        relay2 = st_relay1;
        relay3 = st_relay2;
        relay4 = st_relay3;
        relay5 = st_relay4;
        relay6 = st_relay5;
        relay7 = st_relay6;
        relay8 = st_relay7;
                              }
```

```
void set_relay_turn( char key)
{
switch (key) {    case 0x01 : {st_relay0 =  ~st_relay0;
          if(st_relay0 == 0) {
            DS1307_wr(0x08,0x0A);
                lcd_put(0x84,"ON ");   }
         else{ DS1307_wr(0x08,0x0F);
                lcd_put(0x84,"OFF");   }
          relay1 = st_relay0;
     }break;
        ease 0x02 : {st_relay1 =  ~st_relay1;
        if(st_relay1 == 0) {
                DS1307_wr(0x09,0x0A);
                  lcd_put(0xC4,"ON ");  }
         else{ DS1307_wr(0x09,0x0F);
                lcd_put(0xC4,"OFF");  }
          relay2 = st_relay1;  }break;
      case 0x03 : {st_relay2 =  ~st_relay2;
        if(st_relay2 == 0) {
           DS1307_wr(0x0A,0x0A);
                lcd_put(0x94,"ON ");  }
          else{ DS1307_wr(0x0A,0x0F);
                lcd_put(0x94,"OFF");  }
          relay3 = st_relay2;  }break;
     case 0x04 : {st_relay3=  ~st_relay3;
          if(st_relay3 == 0) {
             DS1307_wr(0x0B,0x0A);
                lcd_put(0xD4,"ON "); }
          else{ DS1307_wr(0x0B,0x0F);
                lcd_put(0xD4,"OFF");  }
         relay4= st_relay3;  }break;
```

```
case 0x05 : {st_relay4 = ~st_relay4;
    if(st_relay4 == 0) {
        DS1307_wr(0x0C,0x0A);
            lcd_put(0x8D,"ON "); }
    else{ DS1307_wr(0x0C,0x0F);
            lcd_put(0x8D,"OFF"); }
    relay5= st_relay4; }break;
case 0x06 : {st_relay5= ~st_relay5;
    if(st_relay5 == 0) {
        DS1307_wr(0x0D,0x0A);
            lcd_put(0xCD,"ON "); }
    else{ DS1307_wr(0x0D,0x0F);
            lcd_put(0xCD,"OFF"); }
    relay6= st_relay5; }break;
case 0x07 : {st_relay6= ~st_relay6;
    if(st_relay6== 0) {
        DS1307_wr(0x0E,0x0A);
            lcd_put(0x9D,"ON "); }
    else{ DS1307_wr(0x0E,0x0F);
            lcd_put(0x9D,"OFF"); }
    relay7 = st_relay6; }break;
case 0x08 : {st_relay7= ~st_relay7;
    if(st_relay7== 0) {
        DS1307_wr(0x0F,0x0A);
            lcd_put(0xDD,"ON "); }
    else{ DS1307_wr(0x0F,0x0F);
            lcd_put(0xDD,"OFF"); }
    relay8 = st_relay7; }break; }
void main(void) {
    char key = 0xFF;
        EA = 1;
```

```
            ET0 =1;
        remote_in  = 1;
            P3 = 0x1F;
            TMOD = 0x11;
           TH0 = 0xDC;
           TL0 = 0x00;
            TF0=0;
            TR0=1;

     lcd_init();
    lcd_put(0x80,"CH1     CH5");
     lcd_put(0xC0,"CH2     CH6");
     lcd_put(0x90,"CH3     CH7");
     lcd_put(0xD0,"CH4     CH8");
           rd_logic();
          rd_logic();
           set_relay();
               while(1) {
                   key = scan_key();
                       if(key <= 0x09)
                   { set_relay_turn(key); }
                     if (remote_in == 0 )
                   { key = P3 & 0x0F;
                       while(remote_in == 0);
                   set_relay_turn(key);
           }                    // end while(1)
}                               // end program
```