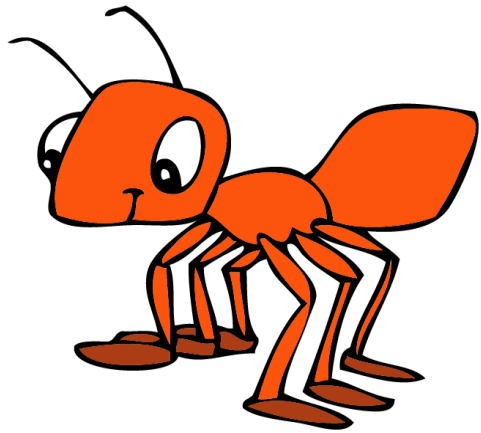
Embedded system development using Opensource tools

(AVR -Base Microcontroller)



สารบัญ

ความรู้เบื้องต้นเกี่ยวกับระบบสมองกลฝังตัว..... 5
ขั้นตอนการพัฒนาระบบสมองกลฝังตัว.....6
สถาปัตยกรรมของ E-block.....7
เครื่องมือ Opensource สำหรับระบบสมองกลฝังตัว..... 8
การติดตั้งระบบ Opensource development.....9
การติดตั้งระบบเพื่อการใช้งาน..... 10
พื้นฐานการรับส่งข้อมูล.....11
อินเตอร์รัปท์ (Interrupt)..... 12
การตั้งเวลา (Timer couter).....13
ระบบการสื่อสาร (Communication)..... 14
การแปลงสัญญาณ Analog to Digital 15
การรวมโครงงานเข้าด้วยกัน (Project integration).....16
สิ่งผิดพลาดที่มักเกิดเป็นประจำในการพัฒนา..... 17

Concept Introduce concept in each chapter with simple lab (library explanation). Focus in detail by using application explanation (Learning through Project)
About 200 pages
Simple and basic start up advance in CDROM and Web Main

Prerequisite

Books on C programming
Embedded Technology (TPA)

1

ความรู้เบื้องต้นเกี่ยวกับระบบสมองกลฝังตัว

1. อะไรคือสมองกลฝังตัว

คำว่า Embedded System แปลเป็นไทยได้ว่าสมองกลฝังตัวนั้นหมายถึง ระบบซึ่งถูกออกแบบมาเพื่อทำงานใดงานหนึ่งเฉพาะ โดยใช้ไมโครโปรเซสเซอร์หรือ ไมโครคอนทรอเลอร์ในการควบคุม เช่น กล้องถ่ายภาพ, กล้องดิจิตอล, หรือแม้แต่เครื่องซักผ้าในปัจจุบันก็ใช้คอมพิวเตอร์ในการควบคุม คำว่า สมองกลฝังตัวนั้นอาจทำจาก PC ก็ได้ ถ้าเป็น PC ที่ออกแบบมาเพื่อใช้งานเฉพาะด้านใดด้านหนึ่ง เช่น เครื่อง Point of Sales ในห้างสรรพสินค้า แต่โดยส่วนใหญ่แล้วระบบสมองกลฝังตัวมักจะมีคุณลักษณะดังต่อไปนี้

a. มีการตอบสนองแบบทันเวลา (Real time) หมายถึงการตอบสนองได้ในเวลาที่กำหนด เช่น ภายใน 1 ms, 1 sec แล้วแต่

ลักษณะงาน เช่น เบรก ABS ต้องตอบสนองทันที, หม้อหุงข้าวอาจจะหุงเสร็จใน 1 ชั่วโมงก็ได้ แต่โดยรวมแล้วต้องสามารถกำหนดเวลาในการตอบสนองได้อย่างถูกต้อง

b. ออกแบบมาเพื่อใช้งานใดงานหนึ่งโดยเฉพาะ เนื่องจากระบบสมองกลจะเกี่ยวกับการควบคุมอุปกรณ์เฉพาะ เช่น กล้องดิจิตอล, โทรศัพท์มือถือ ทำให้ระบบถูกยึดติดกับฮาร์ดแวร์ ซึ่งต่างจาก PC ที่สามารถต่ออุปกรณ์ได้หลายอย่างและประยุกต์ใช้งานได้หลายอย่าง

c. มีระบบตรวจสอบตนเองและจัดการเมื่อมีการทำงานผิดพลาด เนื่องจากระบบสมองกลฝังตัวจะเป็นโปรแกรมที่ต้องเสถียร เพราะถ้ามีการผิดพลาดอาจจะทำให้เสียชีวิตได้ เช่น ระบบควบคุมเครื่องบิน, เรดาร์

ส่วนลักษณะอื่นๆ ซึ่งอาจจะมีไม่มีก็ได้ คือ การใช้กำลังงานน้อย, ตัวโปรแกรมมักจะเก็บอยู่ในตัวหน่วยความจำแบบถาวรในเครื่อง แต่เนื่องจากวิทยาการในปัจจุบันก้าวหน้าทำให้สมองกลฝังตัวเริ่มมีความสามารถมากขึ้น ทำให้ใช้กำลังงานมากขึ้นและอาจมีการโหลดโปรแกรมจากหน่วยความจำภายนอกได้

2. ศาสตร์ที่เกี่ยวข้องกับการพัฒนาระบบสมองกลฝังตัว

ในการพัฒนาระบบสมองกลฝังตัวนั้นจำเป็นต้องมีความรู้ในศาสตร์หลายๆ

แขนง โดยแบ่งเป็น 3 แขนงใหญ่ๆ คือ

a. คอมพิวเตอร์ เนื่องจากปัจจุบันการพัฒนา Firmware จะใช้ภาษา C/C++ เป็นหลัก ดังนั้น ผู้พัฒนาจะต้องมีความรู้ในด้าน Programming และ Software Engineering ในการพัฒนาซอฟท์แวร์ในปัจจุบันระบบสมองกลฝังตัวอาจมีโปรแกรมได้ถึง 1 ล้านบรรทัด ดังนั้นจึงต้องมีระบบการจัดการการเขียนซอฟท์แวร์และการจัดการที่ดี
b. อิเลคทรอนิคส์ เนื่องจากระบบสมองกลจะทำหน้าที่ควบคุมติดต่อกับอุปกรณ์อิเล็คทรอนิคส์เป็นส่วนใหญ่ ดังนั้น จึงต้องมีความรู้ในวงจรอิเล็คทรอนิคส์ รวมถึง การออกแบบ PCB Analog / Digital Design แต่ในบริษัทส่วนใหญ่จะมีการแยกเป็น Hardware / Software Engineer แต่ถ้าเป็นบริษัทเล็กๆ หรือทำเป็นงานอดิเรกก็จำเป็นต้องมีความรู้ทั้ง 2 สาขา
c. ความรู้ในศาสตร์เฉพาะที่เกี่ยวข้อง ทั้งนี้ขึ้นอยู่กับระบบงานที่พัฒนา เช่น พัฒนาหุ่นยนต์ ก็ต้องมีความรู้เรื่องเครื่องกล พัฒนาระบบ ควบคุมเครื่องยนต์ ก็ต้องรู้เรื่องระบบรถยนต์, การสันดาป ฯลฯ

3. การเตรียมตัวเป็นนักพัฒนาระบบสมองกลฝังตัว

การเริ่มต้นในการเป็นนักพัฒนาระบบสมองกลฝังตัวนั้นถ้ายังไม่มีความรู้อะไร

เลย ควรจะเริ่มจาก

a. การเขียนโปรแกรมลงบนบอร์ดทดลองที่มีขายในตลาด ทั้งนี้เพื่อประหยัดเวลาในการประกอบวงจรเอง และเห็นผลได้เร็วที่สุด โดยหนังสือที่ควรศึกษา คือ การเขียนโปรแกรมภาษา C, คุณลักษณะเฉพาะของตัวไอซีไมโครคอนทรอเลอร์ที่ใช้, การออกแบบโปรแกรม
b. เริ่มทำตัวบอร์ดฮาร์ดแวร์เอง โดยออกแบบวงจรที่ต้องการเอง หนังสือที่ควรศึกษาคือ Analog Design, Digital Design, PCB Design การประกอบวงจรโดยใช้หัวแร้งบัคกรี, การทำ PCB
c. การออกแบบระบบทั้งหมด โดยเริ่มจากกำหนดหัวข้อที่สนใจ เช่น ระบบ GPS, ระบบควบคุมหุ่นยนต์ โดยจำเป็นต้องศึกษาในรายละเอียดของหัวข้อที่สนใจ เช่น GPS ทำงานอย่างไร, ฟอร์แมทของรหัสที่ได้ หรือการควบคุมหุ่นยนต์ โดยใช้ DID, Fuzzy Logic จากหัวข้อที่กำหนดจึงหันมาแยกออกแบบเป็นส่วนฮาร์ดแวร์และซอฟท์แวร์

ทั้งนี้เป็นการแนะนำเท่านั้น ในบางบริษัทที่ใหญ่ Hardware Engineer อาจไม่เคยเขียน Software ก็ได้ แต่การมีความรู้ทั้งระบบในการพัฒนาจะทำให้สามารถทำงานร่วมกันได้เข้าใจมากขึ้น

4. ไมโครโปรเซสเซอร์กับไมโครคอนทรอเลอร์ต่างกันอย่างไร

โดยปกติใน PC เราจะได้ยินคำว่าไมโครโปรเซสเซอร์ เช่น Pentium AMD แต่ในโลกของระบบสมองกลฝังตัวเราจะใช้ไมโครคอนทรอเลอร์มากกว่า เช่น MCS-SI, PIC, AVR, ARM ข้อแตกต่างหลักๆ ระหว่างสองตัวนี้แสดงได้ดังตารางต่อไปนี้

ไมโครโปรเซสเซอร์	ไมโครคอนทรอเลอร์
ใช้หน่วยความจำภายนอก	มีหน่วยความจำภายใน
หน่วยความจำมาก	หน่วยความจำน้อย
ต้องต่อกับอุปกรณ์ภายนอก	มีอุปกรณ์บางตัวต่ออยู่ภายใน เช่น Timer, IC, A/D
กินกระแสมาก (มีระบบระบายความร้อน)	กินกระแสต่ำ (สามารถใช้แบตเตอรี่ได้)
รันโปรแกรมจากความจำภายนอก	รันโปรแกรมจากความจำในตัว Chip

จะเห็นว่าจากคุณสมบัติที่ต่างกันทำให้ไมโครคอนโทรเลอร์ถูกเลือกใช้ในระบบสมองกลฝังตัวมากกว่า เนื่องจากขนาดเล็กกว่า, กินไฟน้อยกว่า และโดยส่วนใหญ่จะมีราคาถูกกว่าไมโครโปรเซสเซอร์มาก

5. C สำหรับไมโครคอนทรอเลอร์ กับ C บน PC ต่างกันอย่างไร

การที่ C ถูกเลือกใช้ในการพัฒนาระบบสมองกลฝังตัวเพราะว่าโปรแกรมที่คอมไพล์ได้มีขนาดเล็กกว่าภาษาอื่น และการมี Pointer ทำให้การอ้างอิงถึงตำแหน่งหน่วยความจำใน μc ทำได้ง่ายแต่เนื่องจากโปรแกรมระบบ ES มีขนาด

ใหญ่และซับซ้อนขึ้น ประกอบกับความเร็วและหน่วยความจำของ µc มีมากขึ้นทำให้ภาษา C จึงมักถูกเลือกใช้มากกว่าภาษา Assembly
อย่างไรก็ตาม ภาษา C บน µc กับ C ที่เขียนบน PC นั้นมีข้อแตกต่างกันบ้าง เนื่องจาก µc มีหน่วยความจำจำกัด ดังนั้นการเขียน C บน µc มีข้อควรระวัง, แตกต่างกับบน PC ดังนี้คือ

- หลีกเลี่ยงการใช้หน่วยความจำมากๆ โดยไม่จำเป็น เช่น Array การใช้ค่าตัวแปรหลายตัว เช่น Temporary Variable ควรใช้ร่วมกันถ้าทำได้
- หลีกเลี่ยงการใช้ฟังก์ชั่นที่ซับซ้อนและกินหน่วยความจำมาก เช่น printf หันมาใช้ส่งค่าออกเอาท์พุทโดยตรงแทน
- หลีกเลี่ยงการคำนวณ คูณ, หาร ถ้าไม่จำเป็น เช่น สามารถใช้ operation shift แทนการคูณ, หาร ด้วย 2^x
- ระวังเรื่องโครงสร้างตัวแปรในแต่ละไมโครคอนทรอเลอร์ เช่น int = 16 bit ใน AVR แต่ = 32 bit ใน ARM บน PC จะเป็นมาตรฐานมากกว่า
- การเขียนให้โค้ดกระชับ, สั้นและทำงานเร็วที่สุด เช่น ถ้าใช้ลูปเพียง 2 รอบอาจจะเขียนโค้ดทำงาน 2 ครั้งแทน (ในปัจจุบัน Code Optimizer จะทำหน้าที่ตรวจสอบให้)
- ไลบรารีแต่ละไมโครคอนทรอเลอร์ไม่เหมือนกัน แม้แต่ไมโครคอนทรอเลอร์ตัวเดียวกันแต่คอมไพเลอร์คนละเจ้าก็จะเรียกใช้ไม่เหมือนกัน
- ลดการเรียก function call บ่อยๆ เพราะจะทำให้การทำงานช้าและ

เปลืองหน่วยความจำในการเก็บ stack

6. คำศัพท์ที่ควรรู้ในระบบสมองกลฝังตัว

ในการเรียนรู้ระบบสมองกลฝังตัวมีคำศัพท์ใหม่ๆ มากมาย หัวข้อนี้จะอธิบายคำศัพท์ที่ใช้บ่อยในระบบ ES

- **Register** คือหน่วยความจำที่อยู่ในตัว µc เป็นหน่วยความจำที่มีความเร็วสูง และตัว µc ติดต่อได้โดยตรง ส่วนใหญ่จะใช้ในการควบคุมการทำงานของตัว µc โดยการเขียน, อ่านค่า register ที่กำหนด
- **EEPROM** เป็นหน่วยความจำชนิดหนึ่งซึ่งจะเก็บค่าไว้ได้แม้ดับไฟไปแล้ว แต่โดยมากจะมีขนาดหน่วยความจำไม่มาก
- **Flash Memory** เป็นหน่วยความจำชนิดเดียวกับที่ใช้ใน Thumb Drive ดังนั้นจะเก็บค่าได้แม้ไฟดับ มีขนาดใหญ่กว่า EEPROM แต่ข้อแตกต่างคือ Flash Memory ต้องเขียนข้อมูลทีละ Block ไม่สามารถเขียน ลบ ทีละ Byte ได้
- **API (Application Programming Interface)** คือ ชื่อของ library ที่สามารถเรียกใช้ได้โดยไม่จำเป็นต้องรู้ขั้นตอนการทำงานภายในของโปรแกรม เช่น API ของ RFID เราสามารถอ่านค่า RFID ได้โดยไม่ต้องรู้ว่าจะควบคุมตัว RFID Reader อย่างไร
- **PIEZO** เป็นอุปกรณ์อิเล็คทรอนิคส์ตัวหนึ่งซึ่งจะสั่นเมื่อป้อนกระแสไฟฟ้าเข้าไป ทำให้สามารถนำมาทำเป็นอุปกรณ์กำเนิดเสียงได้
- **TIMER** วงจรจับเวลาโดยปกติงานของ µc จะยุ่งเกี่ยวกับเวลา Timer

จะเป็นวงจรในการกำหนด / จับเวลา เช่น การนับจำนวนคนที่ผ่านมา, การวัดความกว้างของสัญญาณ, การตั้งเวลา

- **JTAG** คือ พอร์ตของ µc ที่ใช้ในการ Debug โปรแกรมที่อยู่ในตัว µc แต่ JTAG จะไม่สามารถ debug อุปกรณ์ภายนอกได้ เช่น Timer A to D ถึงแม้ว่าอุปกรณ์พวกนี้จะรวมอยู่ในตัว µc
- **PCB (Print Circuit Board)** คือ แผงวงจรที่ใช้เป็นที่ติดตั้งอุปกรณ์อิเล็คทรอนิคส์โดยจะมีลายทองแดงเชื่อมระหว่างอุปกรณ์

7. ข้อแนะนำในการอ่านหนังสือเล่มนี้

เนื่องจากหนังสือนี้มีจุดประสงค์เพื่อให้เข้าใจการพัฒนาระบบ ES และ µc 8 bit ขั้นพื้นฐาน ดังนั้นผู้อ่านจึงควรศึกษาเพิ่มเติมจากหนังสืออื่นในส่วนที่เกี่ยวกับ

- C Programming โดยตัวอย่างในหนังสือนี้จะอิงกับ C และ library arr_gcc ดูรายละเอียดของ arr_gcc ได้จาก
- อิเล็คทรอนกส์ขั้นพื้นฐาน ผู้อ่านควรเข้าใจการทำงานของอุปกรณ์พื้นฐาน เช่น R,C, LED, SW, TR และสัญลักษณ์ที่เกี่ยวข้อง
- คณิตศาสตร์ที่เกี่ยวกับ Boolean เช่น การ AND, OR, XOR
- ขั้นตอนการบัคกรีและประกอบวงจร, การอ่านสัญลักษณ์ของวงจร ถ้าต้องการประกอบเอง

สำหรับในระดับ Advance ก็อาจจะศึกษาเพิ่มเติมเกี่ยวกับ

- Software Engineering ในการบริหารการจัดการ การเขียน Software
- UML, State Transition ในการออกแบบระบบ

- PCB Design การออกแบบแผงวงจร PCB เอง
- Digital Design การออกแบบวงจรดิจิตอล

8. โครงงาน Environment Logger and Control (ELC)

เพื่อให้เห็นประโยชน์การใช้งานระบบสมองกลฝังตัวในหนังสือเล่มนี้จะใช้ตัวโครงงาน ELC เป็นตัวอย่างในการพัฒนา โดยเริ่มจาก Phase 1 ถึง Phase สุดท้าย โดยในแต่ละบทจะแบ่งเป็นการทดลองของบทนั้นๆ และส่วนของ Project ELC ที่เกี่ยวข้องกับบทนั้นๆ โดยจะพัฒนาในลักษณะโมดูลเพิ่มเข้าไปเรื่อยๆ ในบทที่ 5 จะอธิบายถึงขั้นตอนตั้งแต่การออกแบบ Requirement จนถึงการออกแบบ และบทที่ 6 เป็นต้นไปจะเป็นการเริ่มการพัฒนาระบบ

2

ขั้นตอนการพัฒนาระบบสมองกลฝังตัว

กระบวนการพัฒนาระบบ ES

ขั้นตอนการพัฒนา ES จะแตกต่างกับการพัฒนาโปรแกรมบนคอมพิวเตอร์ เนื่องจากจะมีส่วนของการพัฒนาฮาร์ดแวร์เข้ามาเกี่ยวข้อง แต่โดยทั่วไปเราสามารถแบ่งเป็นขั้นตอนได้ ดังนี้

a. ขั้นตอนการกำหนดคุณสมบัติของระบบ (Product Specification) หรือการทำ Product Requirement คือ การกำหนดคุณสมบัติหลักๆ ที่ต้องการในระบบว่ามีอะไรบ้าง Product Specification ในขั้นนี้จะเป็น high level อยู่ คือ กำหนดว่าต้องการอะไร แต่ไม่ได้ระบุว่าทำอย่างไร ด้วยอะไร เช่น ต้องการเก็บค่าอุณหภูมิภายนอก แต่ไม่ได้ระบุว่าจะใช้อุปกรณ์

อะไร หรือ ความละเอียดของอุณหภูมิเป็นเท่าไร

b. Detail Specification คือการนำ Requirement ข้างต้นมาแตกรายละเอียดย่อยลงไปในด้านเทคนิค เช่น ความละเอียดของอุณหภูมิที่ต้องการมีการเก็บข้อมูลที่ไหน ช่วงเวลาเท่าไร

c. Hardware / Software Partition คือการแบ่งส่วนว่าส่วนใดจะทำโดย Hardware ส่วนใดจะทำโดย Software ตัวอย่าง เราอาจจะใช้วงจร A to D ที่อยู่ภายใน µc เพื่อแปลงข้อมูลอนาล็อกเป็นดิจิตอล หรืออาจจะใช้ IC ภายนอกซึ่งมีความละเอียดมากกว่าก็ได้ อีกตัวอย่าง เช่นเราอาจใช้เขียน SW เพื่อการจัดการกับโพรโตคอล TCP/IP หรือใช้ IC ที่มีขายในท้องตลาดก็ได้ โดยทั่วไปการแยกว่าจะพัฒนาโดยใช้ HW ภายนอกหรือพัฒนาโดยใช้ Algorithm ขึ้นอยู่กับเงื่อนไขดังต่อไปนี้

   i. ความเร็วที่ต้องการใช้ IC เฉพาะย่อมมีความเร็วสูงกว่า
   ii. ความละเอียด เช่น A to D ภายนอกจะสามารถให้ความละเอียดสูงกว่า A to D ใน µc
   iii. ขนาด / ต้นทุน การใช้ SW ย่อมลดจำนวนอุปกรณ์ที่ใช้และต้นทุนที่ต่ำลง
   iv. ความสามารถของ µc ถ้ามีความสามารถสูง หน่วยความจำมาก ก็อาจจะพัฒนาโดยใช้ S/W ทำแทนก็ได้
   v. ความยืดหยุ่นของระบบ การใช้ S/W ทำให้เราสามารถ

เปลี่ยนแปลงตัว Algorithm ได้ง่ายกว่า ขั้นตอนนี้รวมถึงการเลือก CPU และอุปกรณ์ข้างเคียงที่เหมาะสมกับ Project

d. Hardware / Software Design ในขั้นนี้จะทำการพัฒนาแยกกัน คือ Hardware Engineer จะทำการออกแบบวงจร, PCB (แผงวงจร) Software Engineer ก็จะออกแบบโดยสร้างโปรแกรม, โครงสร้างข้อมูล, State Diagram ในขั้นนี้ Hardware และ Software Engineer จะต้องมีการกำหนดรูปแบบมาตรฐานในการต่อเชื่อมกัน เช่น ขนาดรูปร่างสัญญาณไฟฟ้า, Timing Diagram ทั้งฝั่ง Input และ Output เพื่อให้ S/W สามารถตีความและประมวลผลได้ถูกต้อง

e. Hardware / Software Implement ขั้นตอนการพัฒนาระบบโดย Hardware จะพัฒนาวงจรต้นแบบ ออกแบบ PCB Software จะพัฒนาโปรแกรมตามข้อกำหนดที่ได้มา เนื่องจากการพัฒนา Software จะทำควบคู่กับ Hardware ดังนั้นจะไม่มีตัว Hardware ที่สมบูรณ์ในการทดสอบ จึงจำเป็นต้องมีตัว H/W Simulator ช่วยในการทดสอบโปรแกรมในขั้นตอนการพัฒนา อาจจะเป็นสวิทช์, Voltage Divider หรือ Signal Connector เพื่อจำลองสัญญาณ I/P

f. Integration Testing และ Debugging คือขั้นตอนการรวม S/W และ H/W เข้าด้วยกันเพื่อทดสอบการทำงานขั้นสุดท้าย โดยจะมี

การออกแบบ Test Case โดยอ้างอิงจาก Requirement Specification ใน b เพื่อทดสอบผลิตภัณฑ์ในขั้นตอนนี้จะมีการ Debug โดยใช้ Tools ต่างๆ เช่น JTAG ICE (Incircuit Emulator) ในการหาและดูค่าต่างๆ ใน CPU เพื่อหาจุดผิดพลาดของโปรแกรม

g. Project Maintenance คือขั้นตอนการรวบรวมปรับปรุงและแก้ไข Project Document เช่น Final Specification, Source Code, Schematic ให้ตรงกับการแก้ไขในขั้นสุดท้าย ซึ่งแม้ว่าเราจะมีการสร้างเอกสารต่างๆ ก่อนหน้านี้แล้วแต่โดยส่วนใหญ่เมื่อจบ Project เอกสารจะมีความคลาดเคลื่อนกับระบบสุดท้ายที่พัฒนา ดังนั้นจึงต้องมีการตรวจสอบแก้ไขอีกครั้ง รวมถึงแผนการปรับปรุงผลิตภัณฑ์ในรุ่นต่อๆ ไป

3

สถาปัตยกรรมของ E-block

สถาปัตยกรรมของ E-Block™

1) อะไรคือ E-Block

E-Block ย่อมาจาก Embedded System Block คือชุดพัฒนาระบบสมองกลฝังตัวพัฒนาโดยบริษัทเกียร์ทรอนิคส์ โดยออกแบบให้การพัฒนา Embedded System สะดวกและรวดเร็วขึ้น โดยสามารถนำงานที่เคยออกแบบแล้วมาใช้ได้อีกคล้ายตัวต่อ Lego E-Block นั้นไม่ได้กำหนดเฉพาะส่วนของ Hardware / Software แต่ยังวางเป้าหมายครอบคลุมทั้งการพัฒนา ES โดยครอบคลุมส่วนต่างๆ ดังต่อไปนี้

- Open Concept
- Open H/W
- Open S/W

- Open Process (กำลังพัฒนา)
- Open Courseware (กำลังพัฒนา)

Open Concept คือแนวคิดในการเปิดเผยผลงานที่ได้สู่สาธารณะชน โดยผู้ใช้สามารถเลือก Block ที่ตัวเองต้องการนำมาประกอบกันเพื่อสร้างผลิตภัณฑ์ใหม่ โดยผู้ใช้อาจจะทำการเผยแพร่ Block ที่ตัวเองถนัด เช่น Block เกี่ยวกับ sensor ในขณะเดียวกันก็สามารถนำ Block จากคนอื่นมาใช้กับผลิตภัณฑ์ตัวเองได้ โดยที่ส่วน Integration ระหว่าง Block ก็เป็นสิทธิ์ของคนสร้างในการสร้างผลิตภัณฑ์ที่หลากหลายและเป็นความลับของบุคคลนั้น

Open Hardware คือการเปิดเผยวงจรและการออกแบบ PCB เพื่อให้ผู้อื่นสามารถนำไปใช้ได้ ES จะแตกต่างกับการพัฒนา SW ทั่วไป เนื่องจากจะอิงติดกับ H/W ด้วย ดังนั้นการเปิดเผยวงจรจึงมีความจำเป็นในระบบเปิด รวมถึงการกำหนด Interface ในลักษณะของ Guideline เพื่อให้แต่ละ Block สามารถต่อเชื่อมร่วมกันได้

Open Software คือการเปิดเผย Source Code ของโปรแกรม ทั้งนี้ยังรวมถึงการใช้ Development Environment ในระบบเปิด เพื่อให้ทุกคนสามารถมีสิ่งแวดล้อมในการพัฒนาที่เหมือนกัน คือการใช้เครื่องมือร่วมกัน เนื่องจาก library ของ ES จะขึ้นกับ Compiler เป็นหลัก การกำหนดสิ่งแวดล้อมในการพัฒนาจะช่วยให้การ Compile Code Portable มากขึ้น นอกจากนี้ยังรวมถึง Open API ที่มีการเรียกใช้งานที่เหมือนกันไม่ว่าจะใช้ µc ตัวใด รวมถึงการ

เปิดสำหรับเอกสารที่ใช้ในการออกแบบโปรแกรม เช่น UML, State Diagram สิ่งเหล่านี้จะช่วยให้การศึกษา Source Code ทำได้ง่ายขึ้นและเร็วขึ้น

Open Process (กำลังพัฒนา) คือการเปิดเผยและการกำหนดมาตรฐานในการพัฒนาระบบ ES เริ่มต้นแต่การออกแบบถึงการทดสอบระบบ โดยมีการสร้าง template มาตรฐานเป็น guideline เพื่อให้สามารถเข้าใจตรงกันระหว่างโครงงาน คำแนะนำในการออกแบบระบบโดยใช้สถาปัตยกรรม E-Block คำแนะนำในการเขียนโค้ดที่ดี, การออกแบบวงจร, การควบคุมจัดการระบบ โดยใช้ software versioning และการควบคุมรายงาน Bug Report

Open Courseware (กำลังพัฒนา) คือการเปิดเผยเ อกสารประกอบการเรียนการสอน, รูปแบบการสอน, วิดีโอประกอบการสอน (ถ้ามี), ประสบการณ์ในการพัฒนาจากผู้อื่น ทั้งนี้มีวัตถุประสงค์เพื่อให้ผู้เรียนรู้สามารถเรียนรู้ได้ในเวลาอันสั้น เนื่องจากมีรูปแบบการสอนที่เหมือนกันทำให้ง่ายต่อการเรียนรู้

2) ทำไมต้องมี E-Block

ในการพัฒนา ES โดยทั่วๆ ไปเรามักจะประสบปัญหาดังต่อไปนี้ คือ

1) เมื่อมีการเปลี่ยน µc ในการพัฒนาจะไม่สามารถใช้อุปกรณ์บนบอร์ดเดิมได้ เช่น LCD, LED, SW มักจะต้องซื้อใหม่พร้อมกับ µc
2) ไม่สามารถนำโปรแกรมของคนอื่นมาใช้ได้โดยตรงเนื่องจากการใช้สิ่งแวดล้อมในการพัฒนาที่ต่างกัน ฮาร์ดแวร์และตัวเชื่อม (connection) ที่ต่างกัน

3) ไม่สามารถนำบางส่วนของโครงงานอื่นมาใช้ได้ เนื่องจากโครงงานถูกออกแบบเป็นชิ้นงานเดียว
4) ขยะอิเลคทรอนิคส์จากชิ้นส่วนที่ล้าสมัยและการไม่สามารถนำบางส่วนกลับมาใช้ใหม่ เนื่องจากออกแบบเป็นชิ้นงานเดียว
5) การต่อสายยุ่งยาก บางครั้งต้องการเพียง 3 เส้นเพื่อใช้งาน ถ้าจำเป็นต้องใช้ connection 10 pkin แทนที่จะใช้ แค่ 3 pin และการต่อในลักษณะลูกโซ่ทำได้ยาก
6) การพัฒนาผลิตภัณฑ์ที่ซับซ้อน ต้องใช้งานและงบประมาณมากในการพัฒนา ถ้าต้องพัฒนาทุกๆ ส่วนเองทำให้ SME ไม่สามารถพัฒนาผลิตภัณฑ์ที่ซับซ้อนแข่งกับบริษัทใหญ่ได้
7) การออกแบบผลิตภัณฑ์ชิ้นเดียวทำให้ยากต่อการพัฒนาร่วมกันหลายคน เพราะต้องใช้บอร์ดร่วมกัน การทดสอบและการหาปัญหาที่เกิดก็ทำได้ยาก เพราะไม่สามารถยกโมดูลมาทดสอบได้

3) ส่วนประกอบทางฮาร์ดแวร์และซอฟท์แวร์ของ E-Block

เนื่องจาก ES จะต้องมีทั้งส่วนประกอบทางฮาร์ดแวร์, ซอฟท์แวร์อยู่ด้วยกันเสมอ ซึ่งต่างจากโปรแกรมคอมพิวเตอร์ที่ทำงานบนเครื่อง PC เครื่องใดก็ได้ (บน OS เดียวกัน) ดังนั้นถ้าต้องการให้ ES สามารถที่จะทำงานร่วมกันได้ก็จะต้องกำหนดมาตรฐานทาง H/W เพื่อที่จะนำไปต่อกับ Block อื่นได้

3.1 E-Block Open H/W คือการกำหนดการอินเตอร์เฟสระหว่าง block ต่างๆ

กัน รวมถึงการเปิดเผยวงจรอุปกรณ์ที่ใช้ ลายเส้นวงจร (PCB) ดังนั้น E-Block Open H/W จะเป็นการกำหนดข้อตกลงร่วมกันในการต่อทาง H/W อันได้แก่

- ขนาดของแผงวงจรที่ใช้
- รูปแบบและตำแหน่งของตัวต่อ (connector)
- การกำหนดสัญญาณต่างๆ บนแต่ละตัวต่อ
- ข้อตกลงร่วมในการสร้างแผงวงจร E-Block (E-Block Compliance)

ข้อกำหนดต่างๆ ที่เป็นแนวทางในการพัฒนา อย่างไรก็ตาม ผู้ใช้สามารถนำหลักการของ E-Block ไปใช้ได้โดยไม่จำเป็นต้องอิงกับคำแนะนำ เนื่องจากในการพัมนาเป็นผลิตภัณฑ์อาจต้องการให้ผลิตภัณฑ์มีขนาดเล็ก จึงจำต้องลดขนาดของตัวต่อ แต่ก็จะทำให้เราไม่สามารถใช้ Block ที่มีผู้พัฒนาไว้ได้ จริงๆ แล้ว E-Block ถูกออกแบบเพื่อใช้ในการทำ Prototype ได้เร็วขึ้น เมื่อได้ Prototype ที่ทำงานได้แล้วก็อาจจะทำ Product Optimize เพื่อลดขนาดของแผงวงจรลงเพื่อใช้ในการผลิตจริง

- ขนาดของแผงวงจรที่ใช้ หลักการของ E-block คือการออกแบบ H/W เป็นโมดูลโดยแยกเป็นโมดูลที่สามารถทำงานอิสระได้ เพื่อให้ E-Block สามารถต่อร่วมกันได้อย่างสวยงามจึงมีการกำหนดขนาดของแผงวงจรที่ใช้เป็น 2 ขนาด
    - ขนาด A มีขนาด □ x □ รูขนาดเส้นผ่านศูนย์กลาง _ ที่ 4 มุม

- ขนาด 2A มีขนาดกว้างเป็น 2 เท่าของขนาด A

ขนาดแผงวงจรนี้ออกแบบเพื่อให้สามารถวางบนตัวต่อ Lego เพื่อให้สามารถสร้างกล่องหรือขยายฐานของแผงวงจรได้ง่ายโดยใช้ตัวต่อ Lego ถ้าวงจรมีขนาดใหญ่กว่านี้ก็สามารถทำได้ แต่ขนาดของแผงวงจรควรจะเป็นสัดส่วนจำนวนเท่าของ A เช่น สูงเป็น 2 เท่าของ A กว้างเป็น 3 เท่าของ A ทั้งนี้เพื่อสามารถประกอบกับ Lego ได้โดยง่าย

<< รูปตัวต่อบน Lego>>

- รูปแบบและตำแหน่งของตัวต่อเป็นการกำหนดขนาดของตัวต่อ รูปแบบและระยะบนแผงวงจร E-Block มีรูปแบบการต่อได้ 3 แบบ คือ
  - แผงวงจรสำหรับ CPU จะประกอบด้วยตัวต่อขนาด 10 pin ตัวผู้ข้างละ 2 ตัว แผงวงจรนี้จะเป็นจุดเริ่มต้นของ Project หมายถึงทุกๆ Block จะต่อออกจาก CPU นี้
  - แผงวงจรสำหรับ I/O จะประกอบด้วยตัวต่อขนาด 10 pin ตัวผู้ 2 ตัวต่อทางซ้ายมือ ตัวเมีย 2 ตัวต่อทางด้านขวามือ จะเป็นแผงวงจรสำหรับวงจรที่เป็น I/O และเป็น SPI Port <<รูป>>
  - แผงวงจรสำหรับวงจร A/O และ communicator (IC, VART) จะประกอบด้วยหมู่ตัวต่อขนาด 10 pin ตัวผู้ 2 ตัวอยู่ทางด้านขวามือ ตัวเมีย 2 ตัวต่อทางซ้ายมือ จะเป็นแผงวงจรสำหรับวงจรที่เป็น connection และ A/O

- การกำหนดสัญญาณต่างๆ บนตัวต่อจะคล้ายกับ BUS ที่ใช้ในคอมพิวเตอร์ โดยมีการแยกเป็น 4 กลุ่มขอสัญญาณ คือ
    - กลุ่มที่ 1 ด้านบนซ้าย คือกลุ่มสัญญาณ Output
    - กลุ่มที่ 2 ด้านล่างซ้าย คือกลุ่มสัญญาณ Input
    - กลุ่มที่ 3 ด้านบนขวา คือกลุ่มสัญญาณ Communicator
    - กลุ่มที่ 4 ด้านล่างขวา คือกลุ่มสัญญาณ A to D
- ข้อตกลงร่วมในการสร้างแผงวงจรสำหรับ E-Block คือข้อแนะนำในการสร้างวงจรให้สามารถต่อเชื่อมกับ Block อื่นๆ ได้

1) ออกแบบให้ CPU และ I/O แยกส่วนจากกันเพื่อให้สามารถเปลี่ยน CPU ได้ในอนาคต
2) ออกแบบ Block แต่ละ Block ทำงานอย่างเดียว และแยกเป็น 4 กลุ่มหลักๆ คือ Input, Output, Connector และ A to D ตัวอย่าง เช่น VGA Controller, SD Card Reader, TCP/IP ใน 1 Block ไม่ควรมีหลายๆ หน้าที่ เช่น การอ่าน SD Card และมีจอ LCD ด้วย
3) สำหรับ I/O บอร์ด สัญญาณใดที่ไม่ใช้ให้ bypasss สัญญาณนั้นไปยังขั้วต่ออีกด้านที่ตรงกันเพื่อให้ Block อื่นๆ สามารถใช้สัญญาณได้ ส่วนสัญญาณที่ถูกใช้จะถูกบล็อกไม่ให้ไปยังขั้วต่อที่อยู่อีกด้าน ทั้งนี้เพื่อป้องการการใช้ซ้ำของสัญญาณ จะเห็นว่าการต่อแบบนี้มีลักษณะดี คือ สามารถเลือกใช้เฉพาะสัญญาณที่ต้องการได้ และป้องการการใช้สัญญาณเส้นเดียวกัน Block ในแต่ละ Block ก็สามารถสลับที่กันได้

(ภายในข้างเดียวกัน) การต่อแบบนี้ทำให้เราสามารถพัฒนาและหาจุดบกพร่องของแต่ละโมดูลได้ง่าย ทั้งการทดสอบแต่ละโมดูลและการทดสอบรวมของระบบ ถ้ามีปัญหาในโมดูลใดก็สามารถถอดโมดูลนั้นออกไปโมดูลอื่นก็ยังสามารถทำงานต่อไปได้

3.2 E-Block Open Software คือการเปิดเผย Source Code สำหรับ E-Block Module ทั้งนี้เพื่อให้ผู้อื่นสามารถนำไปพัฒนาต่อได้ ในหนึ่ง 2-Block Hardware อาจจะมี E-Block S/W Module มากกว่าหนึ่งก็ได้ เช่น E-IO H/W จะมี E-Block S/W Module สำหรับขับ 7 SEGMENT, Ingrared ใน E-Block Open SW จะประกอบด้วยไฟล์ *.h เพื่อกำหนดชื่อฟังก์ชั่นที่ใช้ได้ในโมดูล *.c เช่นเดียวกับ H/W เราก็มีข้อกำนดสำหรับการเขียน E-Block S/W โมดูลเช่นกัน ดังต่อไปนี้

1) ฟังก์ชั่นที่เรียกใช้ได้และค่าตัวแปรตางๆ จะอยู่ใน *.h ไฟล์เท่านั้น ฟังก์ชั่นที่ไม่ได้อยู่ใน *.h ต้องเป็นฟังก์ชั่นภายในและไม่ควรเรียกใช้
2) การเขียนตามฟอร์แมทของ Template เพื่อให้การอ่านโค้ดทำได้ง่ายและมีความเข้าใจตรงกัน
3) โครงสร้างไดเรคตอรี่จะเป็นดังนี้

   E-Block\SW\Architecture\Module_name ตัวอย่าง เช่น E-Block\SW\ATMEGA32\E_IR
4) การกำหนดชื่อตัวแปร, ฟังก์ชั่น เพื่อให้เข้าใจตรงกัน เช่น g_

สำหรับ global variables ชื่อฟังก์ชั่นนำหน้าด้วย H/W Module และ S/W โมดูล เช่น 10_IR_XXX_

นอกจากในส่วนของ Source Code แล้ว Open SW ยังรวมถึงส่วนของเอกสารและ Algorithm, Pseudo Code ที่ใช้ในการเขียน ทั้งนี้เพื่อให้การพอร์ทไปแพลตฟอร์มอื่นทำได้ง่าย ดูตัวอย่างการเขียนได้จากเอกสารใน S/W โมดูล

E-Block ที่ใช้คู่กับหนังสือเล่มนี้

การทดลองในหนังสือเล่มนี้อิงอยู่กับ E-Block H/W โมดูลสองโมดูล คือ

- E-Mega32 ใช้ CPU MEGA 32 เป็น µc ที่ความถี่ 16 MHz มี SOCKET สำหรับเสียบ RTC เบอร์และ IC EEPROM Serial Port เพื่อต่อกับ PC ผ่าน IC MAX232 เพื่อการปรับแรงดันให้เข้ากับ A232 Signal Std. มี INMPER ด้วยกันทั้งหมด 8 ตัว ดังต่อไปนี้

  J1
  J2
  J3
  J4
  J5
  J6
  J7
  J8
  รูป

- E-IO คือ I/O โมดลที่ออกแบบมาเพื่อทดสอบการทำงานด้าน I/O ของ µc ประกอบด้วย I/O ดังต่อไปนี้
    - PIEZO สำหรับ output ทางเดียว
    - 7Segment สองตัว เพื่อการแสดงผล
    - Switch สองตัว เพื่อการรับค่า
    - IR Sendor รับค่า IR Remote

  ใน E-IO ที่จะประกอบด้วยซอฟท์แวร์โมดูลดังต่อไปนี้

  io_ir.c
  io_7segment.c
  io_sw.c
  io_piezo.c

4

เครื่องมือ Opensource สำหรับระบบสมองกลฝังตัว

- Overall process for develop Embedded systemWinAVR
- E_block library
- Compiler GNU C
- Program Editor
- Programmer
- avrdude ใช้ในการติดต่อกับ ตัวโปรแกรมเมอร์ แบบต่างๆที่ใช้กับ ปอร์ด AVR เช่น ISP, STK500, Ponyprog โดยจะต้องเลือก avrdude เวอร์ชั่นที่สนับสนุนตัวโปรแกรมเมอร์รุ่นต่างๆ
- Bootloader
- Debugging
-

5

การติดตั้งระบบ Opensource development

การติดตั้งซอฟท์แวร์

ในการพัฒนา Embeded System จะมีการติดตั้งซอฟท์แวร์ทั้งหมด 2 ตัว

a. WinAVR เป็นคอมไพเลอร์ในภาษา C ในชุด WinAVR จะมีโปรแรมหลักดังนี้ (อยู่ใน c:\winavr-20070525\bin

1. AVR_GCC เป็นคอมไพเลอร์ C สำหรับ AVR คือมีการติดตั้งไลบรารี่สำหรับ AVR ลงมาด้วย
2. Program Note Pad เป็นโปรแกรม Editor ใช้ในการเขียน, คอมไพล์ภาษา C
3. MFile โปรแกรมการสร้าง / แก้ไข Make files
4. Avrdude สำหรับโปรแกรมลงบนบอร์ด AVR

5. Avrice สำหรับโปรแกรม InCircuit Emulator รายละเอียดการใช้งานจะอยู่ในเอกสารภายใต้ c:\winavr_20070525.doc

b. AVRSTUDIO 4 จาก ATMEL ตัวนี้ไม่ใช่ Opensource แต่ก็แจกฟรีเราสามารถเขียนโปรแกรมบน Avrstudio แล้ว compile ด้วย avr_gcc ก็ได้ เราติดตั้งตัวนี้เพื่อใช้ในการทำ debug และ simulator

1.1 การติดตั้งซอฟท์แวร์ WINAVR จะต้องติดตั้งก่อน AVRSTUDIO เสมอ (เพื่อให้ AVR STUDIO สามารถหา avr_gcc ได้เจอตอนติดตั้ง) ทำโดยการ click เลือก winavr_20070525-install.exe เลือก directory ที่ต้องการติดตั้ง แล้วกด Next โปรแกรมจะติดตั้งให้โดยอัตโนมัติภายใต้ directory ที่เลือก โดยมี directory ย่อยที่น่าสนใจคือ

\bin	เก็บโปรแกรมทั้งหมดของ WINAVR
\avr	ไลบรารี่, include file สำหรับภาษา C ของ AVR
\examples	ไฟล์ตัวอย่าง
\doc	เอกสารการใช้งานของโปรแกรมต่างๆ
\lib	library ของตัว gcc
\pn	โปรแกรม programmer notepad
winavr_user_manual.html	เอกสารประกอบ WinAVR

ถ้าต้องการดูรายละเอียดของตัว WinAVR สามารถดูได้จากที่นี่

1.2 การติดตั้ง AVR STUDIO เนื่องจากโปรแกรมนี้มีฟังก์ชั่นการจำลองการทำงานของ CPU ช่วยทำให้ค้นหาข้อผิดพลาดได้ง่ายขึ้น การติดตั้งก็สามารถคลิกที่โปรแกรม Studio4b528.exe สามารถดาวน์โหลดได้จาก www.atmet.com ในอนาคตถ้าต้องการเวอร์ชั่นใหม่ๆ

1.3 การตั้งค่าเริ่มต้นเพื่อใช้งาน WinAVR

สำหรับ WinAVR นั้นเราสามารถใช้ตัวสร้างโปรแกรม (Editor) ได้ 2 ตัวต่อ Programmer Notepad และ AVRStudio ในที่นี้จะสอนเฉพาะการใช้ Programmer Notepad เพราะเป็นโปรแกรม Opensource และสามารถแก้ไข Config ค่าต่างๆได้ง่ายกว่า

ก่อนเริ่มต้นการใช้งานขออธิบายขั้นตอนการสร้างโปรแกรมและการคอมไพล์โดยย่อๆ ดังนี้

- เริ่มต้นด้วยการสร้าง Project File คือ แฟ้มที่เก็บงานของทั้งหมด ตัวอย่างเช่น โปรเจคทำเครื่องวัดอุณหภูมิก็จะมีไฟล์โปรแกรมอ่านค่าอุณหภูมิแสดงผลอุณหภูมิ ซึ่งโดยปกติเราจะนิยมเขียนใส่ 1 ไฟล์ภาษา C ทำงานหลักๆ 1 อย่าง ทั้งนี้เพื่อให้สามารถนำไปใช้งานกับโปรแกรมอื่นได้
- การเพิ่มไฟล์เข้าไปในโปรเจคต์ทั้ง *.c, *.h เพื่อบอกว่าโปรเจคนี้มี

ไฟล์อะไรบ้าง

- การสร้าง Makefile Makefile คือไฟล์ที่เป็นตัวบอก Compiler โปรเจคนี้มีไฟล์อะไรบ้าง, CPU เป็นเบอร์อะไร speed เท่าไร Makefile จะช่วย Compiler รู้ว่าจะต้อง Compile ไฟล์ไหน อย่างไร และไฟล์อยู่ที่ใดบ้าง โดยปกติ makefile จะอยู่ที่ไดเรคตอรี่ของโปรเจคนั้นและชื่อว่า Makefile
- การคอมไพล์โปรแกรมตามค่า option ต่างๆ ที่ระบุไว้ใน Makefile
- การโหลดโปรแกรมจากคอมพิวเตอร์ไปยัง AVR หรือที่เราเรียกว่า Target Board

2. **การติดตั้ง Hardware**

E_block Module ที่ให้มากับหนังสือมีด้วยกัน 2 โมดูล คือ

- E_Mega32 เป็น CPU Board ที่มี AVR_mega32 เป็น CPU มีขนาดโปรแกรม 32 K Byte RAM 22 K EEPROM 1 K มีตัว Socket สำหรับ IC RTC (สัญญาณนาฬิกา) และ External EEPROM หรืออุปกรณ์ IC ตัวอื่นๆ พอร์ต Serial สำหรับต่อกับ Computer และ IEP สำหรับถ่ายโปรแกรมจาก PC สู่ Board Power Port สำหรับไฟเลี่ยง 5 V.

<<<<picture>>>>>>

การ Plug E_IO Board ทางด้านซ้ายมือและเสียบสาย Power ที่ต่อจาก USB

ลงบนที่ Board จะเห็น

- E_IO เป็นบอร์ด I/O ประกอบด้วยตัวรับ IR, PIZZO, LED 7 Segments 2 หลัก, สวิทช์กดติด-ปล่อยดับ 2 ตัว ใช้เพื่อทดลองการต่อ I/O เข้ากับตัว AVR โดย EIO จะเข้ากับ E_mega 32 ทางด้านซ้ายมือ โดยตัว PIZZO จะอยู่ด้านบน

ข้อควรระวัง ควรต่อ E_IO และ E_Mega32 ให้ถูกข้าง มิฉะนั้นอาจจะเสียหายได้

3. ทดลองลงโปรแกรม hello.c เป็นโปรแกรมในการแสดงค่า Hello ลงบนจอเพื่อทดลองการทำงานของ Hardware และความถูกต้องของการลง Software
4. รูปแบบของการทดลองในบทต่อไป

ในบทต่อๆ ไป จะเป็นการศึกษาถึงคุณสมบัติต่างๆ ของตัว AVR โดยจะมีรูปแบบการเขียนที่เหมือนกันเพื่อให้ง่ายต่อการเรียนรู้ โดยจะมีส่วนต่างๆ ดังต่อไปนี้

- ทฤษฎีการทำงาน จะพูดถึงการทำงานโดยรวมของตัว AVR หรือแบบที่เกี่ยวข้อง
- รีจิสเตอร์ ที่เกี่ยวข้อง
- E_block ที่เกี่ยวข้อง (ถ้ามี) รวมถึง API ที่ต้องใช้
- โปรแกรมแลปการทดลองและคำอธิบาย
- ส่วนของ Project Environment Monitoring ที่เกี่ยวข้อง (ถ้ามี)

6

พื้นฐานการรับส่งข้อมูล

ในบทต่อไปนี้จะพูดถึงสถาปัตยกรรมโครงสร้างของ µc ATMGGA และ Peripheral ที่อยู่ในตัว µc เช่น TIMER, A/O, VART โดยจะมีรูปแบบของหัวข้อในบทดังต่อไปนี้

- ประโยชน์การใช้งาน พูดถึงประโยชน์และการนำไปใช้งานของเนื้อหาที่เรียนในบทนี้
- ทฤษฎีการทำงาน อธิบายการทำงานของอุปกรณ์ต่างๆ และร้อสเตอร์ที่เกี่ยวข้อง
- ตัวอย่างโปรแกรมพร้อมการอธิบายการทำงานเพื่อเข้าใจในเนื้อหา
- โครงงาน คือ การออกแบบโปรแกรมเพิ่มเติมสำหรับโครงงานหลัก

ประโยชน์การใช้งาน การอ่านเขียนข้อมูลดิจิตอลจากตัว µc ไปยังข้างนอกถือ

เป็นการทำงานพื้นฐานของ µc ทุกตัว เพราะ µc มักต้องติดต่อกับโลกภายนอกเสมอ โดยพอร์ท I/O นี้จะรับสัญญาณดิจิตอลเท่านั้น การใช้งาน เช่น เราใช้เป็นอินพุตอ่านค่าจากสวิทช์ภายนอกหรือการเขียนค่าเพื่อไปขับวงจร LED, Relay ต่างๆ

ทฤษฎีการทำงาน โดยปกติแต่ละขาของ µc จะเป็นพอร์ตๆ หนึ่งซึ่งสามารถตั้งค่าได้ว่าจะใช้รับข้อมูลหรือส่งข้อมูล โดยตั้งค่าในรีจิสเตอร์ที่ชื่อว่า DDR (Direction Register) DDR จะมีทั้งหมด 4 ตัวสำหรับแต่ละพอร์ต โดยมีชื่อดังต่อไปนี้

DDRA สำหรับ Port A

DDRB สำหรับ Port B

DDRC สำหรับ Port C

DDRD สำหรับ Port D

โดยแต่ละ PIN ในพอร์ทสามารถตั้งค่าให้เป็นอินพุทหรือเอาท์พุทโดยตั้งค่าให้บิทนั้นเป็น 0 สำหรับการอ่านค่าและ 1 สำหรับการเขียนค่า โดยแต่ละบิทสามารถแยกตั้งค่าอิสระจากกันได้

การใช้งานพอร์ตเป็นอินพุท

การใช้พอร์ตเป็นอินพุทสามารถทำได้ด้วยการตั้งค่า DDRx (x – A, B, C, D) เป็น 0 ในบิทของพอร์ตที่ต้องการตั้ง เช่น พอร์ต A ที่ขา A2 เป็น output

DDRA = xxxx x0xx x เป็น 1 หรือ 0 ก็ได้

สำหรับการใช้พอร์ทเป็นอินพุทจะมีการตั้งค่า R Pull Up ได้ด้วยการตั้งค่า R

Pull Up ทำให้ไม่ต้องมีการใช้ R Pull Up ภายนอก เหตุผลที่ต้องมีการตั้งค่า R Pull Up ทำให้ไม่ต้องมีการใช้ภายนอก เหตุผลที่ต้องมีการตั้งค่า R Pull Up เพื่อป้องกันสัญญาณรบกวนเข้ามาทำให้การอ่านค่าอินพุทผิดไป เพราะสัญญาณรบกวนอาจทำให้อินพุทเป็น 1 ได้ การตั้งค่า R Pull Up จะทำให้อินพุทเป็น 1 เสมอจนกว่าจะมีการกดสวิทช์ที่ต่อลง ground เพื่อดึงอินพุทเป็น 0

<<รูป>>

การตั้งค่า Rpullup จะต้องมีการส่งค่า 1 ให้แก่ PORTxn ที่ต้องการตั้งค่า และ Enable Rpullup ใน SFIOR รีจิสเตอร์บิทที่ชื่อว่า PUD ให้เป็น 0 ถ้า PUD ถูกตั้งค่าเป็น 1 พอร์ทจะถูกตั้งค่าเอง High Independence คือมีลักษณะเหมือนขาลอยออกจากวงจร (ไม่ใช่ 0 หรือ 1) (ดูตัวอย่างในการทดลอง) ค่าที่อ่านได้จะถูกเก็บอยู่ในค่ารีจิสเตอร์ PINX (x = A, B, C, D) สำหรับการอ่านค่าจากพอร์ทเพื่อประกันการอ่านค่าได้ถูกต้องจะต้องมีการใส่คำสั่ง nop หรือ delay 1 ลูก clock เพื่อให้ข้อมูลที่อ่านค่าบาง sync ได้ถูกต้อง สำหรับขาที่ไม่ถูกใช้ควรจะ Enable pull up ไว้เพื่อป้องกันสัญญาณรบกวน

การใช้งานพอร์เป็นเอาท์พุททำได้โดยการตั้งค่าบิทใน DDRx รีจิสเตอร์เป็น 1 สำหรับเอาท์พุทและตั้งค่าที่ต้องการไปยังพอร์ท PORT x

โปรแกรมตัวอย่าง การอ่านค่าอินพุทจากสวิทช์และแสดงไปยัง LED

การทำงานของโปรแกรมกดปุ่มทางด้านบนเพิ่ม 7SEGMENT จาก 0-9 และกดปุ่มด้านลงจะลดค่าจากค่าปัจจุบันจนถึง 0

7

อินเตอร์รัปท์ (Interrupt)

Chapter 7 Interrupt

- Theory of operation
- Internal interrupt.
  - Timer interrupt
  - lab : Timer watch
  - project : Timer for logging data
  -
- External interrupt.
  - Interrupt from Switch
  - lab : Intruder alarm
  - Interrupt from Infrared Rx
  - lab Remote control one channel
  - project start/stop logging

การอินเตอร์รัปต์เป็นหัวใจสำหคัญของตัว µc และระบบ Embedded System เนื่องจากเป็นกลไกที่สามารถทำให้ µc ตอบสนองแบบ Realtime ได้ ตัวอินเตอร์รัปต์จะอยู่ในคอร์ของ µc และการจัดการอินเตอร์รัปต์ของแต่ละ µc ก็จะแตกต่างกันไป µc ความสามารถสูงก็จะสามารถจัดลำดับความสำคัญได้ เปลี่ยนแปลงความสำคัญได้และอินเตอร์รัปต์ซ้อนอินเตอร์รัปต์ก็ได้

ประโยชน์การใช้งาน

อินเตอร์รัปต์จะนำมาใช้ในงานที่ต้องการการตอบสนองทันทีเมื่อเกิดเหตุการณ์ต่างๆขึ้น เช่น เมื่อมีข้อมูลเข้ามายัง Serial Port เราต้องการให้โปรแกรมมารับค่าทันทีเพื่อป้องกันการสูญหายของข้อมูล อีกตัวอย่างการใช้งานอินเตอร์รัปต์ คือ การนำมาใช้ตรวจสอบอินพุท เช่น สวิทช์ โดยแบบเดิมเราจะต้องวนลูปเพื่อตรวจสอบการกดสวิทช์เสมอ ถ้าเราต่อสวิทช์ไว้ที่ขาอินเตอร์รัปต์ เมื่อมีการกดสวิทช์ขึ้นขาอินเตอร์รัปต์ก็จะถูกกระตุ้นและ µc จะกระโดดไปทำงานตามโปรแกรมที่ตั้งไว้ใน Interrupt Vector Table

ในบทที่ 6 จะเห็นว่าเราจะต้องวนลูปโปรแกรมเพื่อแสดงตัวเลข 2 ตัวสลับกันไป ในกรณีนี้ถ้าโปรแกรมทำงานนานเกินไปก่อนจะถึงโปรแกรมแสดงตัวเลข ตัวเลขก็อาจจะกระพริบได้ จะเห็นว่าวิธีนี้การแสดงตัวเลขจะขึ้นอยู่กับการทำงานส่วนอื่น เราสามารถใช้อินเตอร์รัปต์เพื่อทำให้เวลาที่ใช้ในการแสดงผลมีค่าแน่นอนได้ โดยใช้อินเตอร์รัปต์ร่วมกับ Timer ในการกำหนดเวลาในการรีเฟรชข้อมูล

ทฤษฎีการทำงาน

การอินเตอร์รัปต์มีหลักการทำงานง่ายๆ คือ เมื่อมีสัญญาณอินเตอร์รัปต์เกิดขึ้น โปรแกรมก็จะกระโดดไปทำงานยังโปรแกรมที่กำหนดไว้ให้รองรับการอินเตอร์รัปต์ตรงที่ใช้ในการบอกว่าอินเตอร์รัปต์ใดควรไปที่โปรแกรมส่วนใด เรียกว่า Interrupt Vector Table หรืออาจเรียกว่า Interrupt Handling

ใน AVR จะมี Vector Table ทั้งหมด 21 ตัว ดังตาราง โดยอินเตอร์รัปต์ที่อยู่บนสุดจะมีลำดับความสำคัญมากสุด

<<Copy จาก manual จาก 42>>

โดยแหล่งที่มาของสัญญาณอินเตอร์รัปต์สามารถแยกได้เป็นจากภายนอก µc คือ ขา Reset, INT0, INT1, INT2 และจาก Peripheral ภายในตัว µc เช่น Timer, พอร์ตสื่อสาร (USART, SPI, IC), ADC EEPROM ภายในตัว µc

โดยวิธีการใช้งานตัว IVT เราสามารถทำได้โดย

1) ตั้งค่าใน IVT ให้ชี้ไปยังส่วนของโปรแกรมที่ต้องการทำงานโดยใน Source Code ขึ้นบรรทัดของ function ด้วยคำสั่ง SIGNAL (SIG_INTERRVTx) x = 0, 1, 2 ใช้ SIGNAL แทนการกำหนดชื่อของฟังก์ชั่น
2) ตั้งค่าพารามิเตอร์ในการกำหนดคุณสมบัติของอินเตอร์รัปต์สำหรับอินเตอร์รัปต์จากขาภายนอก สามารถตั้งได้ว่าให้เกิด

สัญญาณอินเตอร์รัปต์เมื่อมีการเปลี่ยนแปลงสัญญาณที่ขาอย่างไร โดยการตั้งค่า MCUCR รีจิสเตอร์ ?? (ISCO1, ISCO0) สำหรับ INT0 และบิทที่ 2, 3 (ISCI0, ISCI1) สำหรับ INT1 ส่วน INT2 จะถูกตั้งค่าให้เกิดสัญญาณเมื่อมีการเปลี่ยนแปลงสัญญาณเท่านั้น ส่วน INT0, 1 จะสามารถตั้งค่าได้มากกว่าตามตารางดังต่อไปนี้

บิท ISCx1 (x = 0, 1)	บิท ISCx0	การเกิดอินเตอร์รัปต์
0	0	เกิดสัญญาณ Low ที่ขา อินเตอร์รัปต์ ใช้ใน sleep mode ได้
0	1	มีการเปลี่ยนแปลงสัญญาณจาก 1 → 0 0 → 1
1	0	เกิดการเปลี่ยนแปลงจาก 0 → 1 (Full Edge)
1	1	เกิดการเปลี่ยนแปลงจาก 0 → 1
หมายเหตุ: โหมด 2 – 4 ไม่สามารถรันโปรแกรมอินเตอร์รัปต์จาก sleep mode ได้		

ตัวอย่างการตั้งค่า

MCUCR 1 = 0x01 ใช้ OR เพื่อไม่ให้บิทอื่นถูกเปลี่ยน

ตั้งค่า ISCO0 = 1
ISCO1 = 0
ดังนั้น จะเกิดการอินเตอร์รัปต์เมื่อมีการเปลี่ยนแปลงจาก 0 → 1 หรือ 1 → 0

3) อีนาเบิ้ลสัญญาณอินเตอร์รัปต์จากภายนอกเพื่อให้อินเตอร์รัปต์สามารถทำงานได้ โดยการตั้งค่าให้เป็น 1 ในรีจิสเตอร์ GICR บิทที่ 5 สำหรับ INT2 บิทที่ 6 สำหรับ INT0 และบิทที่ 7 สำหรับ INT1
4) อีนาเบิ้ลบิท 7 7 bit ใน SREG เพื่อให้อินเตอร์รัปต์สามารถทำงานได้
5) ถ้าต้องการตรวจสอบการเกิดอินเตอร์รัปต์สามารถตรวจสอบได้จากรีจิสเตอร์ GIFR โดยขาที่ 5 INTF2 จะถูกตั้งเป็น 1 ถ้ามีการเกิดอินเตอร์รัปต์ที่ขา INT2 บิทที่ 6 INFO สำหรับ INT0 และบิทที่ 7 INTF1 สำหรับ INT1

Lab 7-1

สำหรับการอินเตอร์รัปต์อื่นๆ สามารถดูตัวอย่างเพิ่มเติมในบทถัดไปในเรื่องของ TIMER, COUNTER

8

การตั้งเวลา (Timer couter)

ประโยชน์การใช้งาน ตัวนับและตัวจับเวลาเป็นอุปกรณ์ที่ใช้มากสุดตัวหนึ่งใน µc โดย µc เกือบทุกตัวจะมี TIMER, COUNTER มาให้เสมอ เราสามารถประยุกต์การใช้งาน TIMER, COUNTER ได้หลายอย่าง เช่น การนับจำนวนสินค้าในการผลิต, การนับจำนวนลูกคลื่นสัญญาณ, วัดความยาวของคลื่น, การสร้างเวลาอ้างอิงต่างๆ เช่น จับเวลาแบบคร่าวๆ นอกจากนี้เรายังสามารถสร้างคลื่นความถี่ได้จากตัว COUNTER ได้ (สัญญาณเสียงต่างๆ)

9

ระบบการสื่อสาร (Communication)

Chapter 9 Communication

- UART
  - Communication with Terminal
  - UART Library drivers
  - lab Master mind on terminal
    - Games , Show picture
    - Piano keyboard
  - project
    - Send log data to serial / text file
- I2C (extra IC need or give away)
  - READ time /Set time from RTC using terminal
    - lab digital clock

  project
    - Get RTC for time stamp

10

การแปลงสัญญาณ Analog to Digital

Chapter 10 Analog to digital

- Analog to digital convertor.
    - Read LDR light value
        - lab on when dark

      Project light sensor
    - Read temperature using thermister
        - lab warn when temperature sensor
        - project temperature sensor

11

การรวมโครงงานเข้าด้วยกัน (Project integration)

Chapter 11 Project intergration

- Light & temperature monitoring and control
  - integrate to relay
  - integrat to IR tx for remote control
- Game via terminal (KBD)
- Strobscope

12

สิ่งผิดพลาดที่มักเกิดเป็นประจำในการพัฒนา

Chapter 12 Common pitfall in Embedded system/ E-block

- Use same I/O
- Use same register
- User same clock/ prescale
- Use same interrupt
- Timing/ Response consider when use interrupt. Is it enough time to process next cycle Timer /Counter use different mode may use same register , Clock source / Prescale Ex Input capture and Output compare use the same TCNT when output compare match TCNT is reset to 0. This make ICP did not get correct value.
- Timer0/Timer 1 use same prescale
- Need to design for timing, state diagram, interrupt time

13

Appendix A Advance Application (Need extra components (Detail In CDROM)

Extra components need

- Remote control
  - I feel control using remote
  - Timer to Switch TV off
  - Wireless piano using IR remoter control
  - Weather station / Display on terminal
  - Game using remote control
  - Temperature station via Web / SMS
  - TV interface

13

Appendix B Resource weblink, books for further study

Books

Embedded C programming and the ATMEL AVR
เทคโนโลยี่สมองกลฝังตัว
สำนักพิมพ์ ส.ส.ท โดย JASA

Information Websites

www,avrfreaks.net
Community for AVR
www.atmel.com
AVR website

http://instruct1.cit.cornell.edu/courses/ee476/

Sample AVR projects from Cornell

Download website

Bootloader

www.chip45.com

winavr

winavr

Template manual

Set/Modify page style

- Press F11 or menu Format --> Styles and Formatting
- Select page icon Modify it

Insert different page style

- Insert page break
- Select different page style by Press F11 or menu Format --> Styles and Formatting

Define chapter outline

- Choose tools --> Outline numbering
- This menu can set order hierarchy of paragraph and automatic insert number or alphabet numbering

Style in these template

Chapterno number of chapter

Chaptertitle title chapter

Tips to use template

- Always end chapter with page break
-

How to enter index
How to set//enter chapter

- Goto edit paragraph/styles (click right mouse button)
- select organizer tab
- set next style for style use next
- set style category

How to set/enter toc

- Goto edit index/table
- At index/table tab check outline box and press ... to edit
- Map desired style to TOC level

บรรณานุกรม

Embedded..... **10**