

241-302 Computer Engineering Lab II ภาคเรียนที่ 2 ปีการศึกษา 2554

Lab 3HB06 Hardware and Software co-design and Debugging

ผู้สอน ดร. ปัญญยศ ไชยกาฬ

1. วัตถุประสงค์

- เพื่อให้นักศึกษาได้เรียนรู้เทคนิคการออกแบบฮาร์ดแวร์และซอฟต์แวร์ร่วมกันเพื่อให้ง่ายในการทดสอบความถูกต้อง
- เพื่อให้นักศึกษาได้ฝึกการเขียนซอฟต์แวร์สำหรับการทดสอบความถูกต้องของฮาร์ดแวร์
- เพื่อให้นักศึกษาได้ฝึกเทคนิคการ Debug โปรแกรม

2. เป้าหมาย

- นักศึกษาสามารถใช้โปรแกรม Proteus ในการทดสอบการทำงานร่วมกันของฮาร์ดแวร์และซอฟต์แวร์ได้อย่างมีประสิทธิภาพ ก่อนที่จะนำซอฟต์แวร์ไปทดสอบการทำงานบนฮาร์ดแวร์จริง

3. กำหนดส่งงานและวิธีการส่งงาน

- คำถามก่อนการทดลอง
ให้นักศึกษาทำมาให้เสร็จและส่งที่อาจารย์ผู้สอนก่อนเริ่มคาบเวลาปฏิบัติการ โดยโค้ดส่วนหนึ่งจะเป็นส่วนประกอบของโค้ดโปรแกรมใน Checkpoint 1 และ Checkpoint 2 ด้วย
- วิธีการตรวจ Checkpoint
 - ใน Checkpoint 1 นั้นให้นักศึกษาทำ Checkpoint 1.1-1.2 ให้เสร็จก่อน แล้วจึงค่อยเรียกผู้คุมแลบตรวจทั้ง 2 ข้อย่อยในคราวเดียว
 - ใน Checkpoint 2 ให้ทำ Checkpoint 2.1 เสร็จแล้วจึงเรียกผู้คุมแลบตรวจ จากนั้นจึงทำ Checkpoint 2.2 ให้เสร็จแล้วจึงเรียกผู้คุมแลบมาตรวจอีกครั้งหนึ่ง
- การส่ง Logbook
ให้ส่งหลังจากทำแลบ 3HB08 เสร็จแล้ว
- การส่งคำถามท้ายการทดลอง
ให้ส่งใน LMS ภายในวันที่อาจารย์ประกาศ (จะแจ้งให้ทราบในวันทำแลบ)
- การสอบ
กำหนดสอบ หลังจากทำแลบเสร็จแล้ว 2 สัปดาห์ (จะแจ้งวันเวลาสอบให้ทราบในภายหลัง)

4. คำถามก่อนการทดลอง

ให้เขียนฟังก์ชันสำหรับหน่วยเวลาในภาษาซีของซีพียู AVR โดยผู้ใช้สามารถป้อนค่าพารามิเตอร์จำนวนวินาทีที่ต้องการหน่วง (รับเลขจำนวนเต็มขนาด 8 บิต) กำหนดให้โค้ดดังกล่าวรันบนซีพียูความเร็ว 16 MHz (คำแนะนำ: ให้ใช้ Timer 1 ของซีพียู AVR ให้เป็นประโยชน์)

5. การให้สัดส่วนคะแนน

คะแนนของการทดลองนี้แบ่งออกเป็น 4 ส่วน ได้แก่

- คำถามก่อนการทดลอง	5	%
- Checkpoint	30	%
- Logbook	10	%
- คำถามท้ายการทดลอง	15	%
- สอบปฏิบัติการ	40	%

6. Hardware and Software Co-Design

ในการออกแบบระบบสมองกลฝังตัว (Embedded System) ประกอบไปด้วย 2 ส่วนคือส่วนของฮาร์ดแวร์และส่วนของซอฟต์แวร์ การออกแบบระบบดังกล่าวในอดีตมักเริ่มจากการออกแบบฮาร์ดแวร์ก่อน เมื่อออกแบบฮาร์ดแวร์เสร็จจึงเข้าสู่กระบวนการออกแบบและพัฒนาซอฟต์แวร์ และตามด้วยการนำทั้งสองส่วนมาทำงานร่วมกันบนฮาร์ดแวร์จริง เนื่องจากฮาร์ดแวร์และซอฟต์แวร์ไม่ได้ถูกออกแบบในคราวเดียวกัน หากต้องการเพิ่มฟังก์ชันการทำงานของทางซอฟต์แวร์อาจทำได้ลำบากในขณะเดียวกันการออกแบบฮาร์ดแวร์ที่ไม่มีการวางแผนที่ดีพอจะทำให้การอัปเดตฮาร์ดแวร์ทำได้ด้วยความยากลำบาก ดังนั้นแนวโน้มการออกแบบระบบสมองกลฝังตัวในปัจจุบันจึงเน้นไปที่การออกแบบร่วมกันของฮาร์ดแวร์และซอฟต์แวร์ โดยมีวัตถุประสงค์เพื่อให้ได้ระบบที่..

- สะดวกในการนำโมดูลซอฟต์แวร์ที่มีอยู่แล้วกลับมาใช้ใหม่ เนื่องจากผู้พัฒนาที่ใช้แนวทางการออกแบบนี้มักจะออกแบบซอฟต์แวร์แบ่งออกเป็น 2 ส่วนอย่างชัดเจนคือ ส่วนแรกเป็นส่วนที่ขึ้นต่อฮาร์ดแวร์ และส่วนที่สอง เป็นส่วนที่ไม่ขึ้นต่อฮาร์ดแวร์ ดังนั้น หากมีความจำเป็นที่จะต้องเปลี่ยนตัวฮาร์ดแวร์ ก็จะต้องเปลี่ยนเฉพาะส่วนที่ขึ้นต่อฮาร์ดแวร์เท่านั้น ส่วนที่ไม่ขึ้นต่อฮาร์ดแวร์ก็ยังสามารถนำกลับมาใช้ใหม่ได้เรื่อยๆ นอกจากนี้ยังสะดวกต่อการเพิ่มโมดูลของซอฟต์แวร์อื่นๆ เข้าไปในภายหลังอีกด้วย
- สามารถอัปเดตไปใช้กับฮาร์ดแวร์รุ่นใหม่ๆ ได้โดยไม่ต้องดัดแปลงมาก (Portability) การออกแบบระบบที่ดีจะต้องเผื่อไว้สำหรับการอัปเดตวงจรเอาไว้อย่างดี มี 2 ปัจจัยที่ทำให้ต้องเปลี่ยนฮาร์ดแวร์คือ ประการแรกคือผู้พัฒนาต้องการฮาร์ดแวร์ที่ดีกว่าเดิมเพื่อรองรับความสามารถที่เพิ่มขึ้น ประการที่สองคือเมื่อเวลาผ่านไปฮาร์ดแวร์รุ่นปัจจุบันอาจจะล้าสมัยและหาซื้ออะไหล่ได้ยากขึ้น การวางแผนที่ดี จะช่วยให้การอัปเดตฮาร์ดแวร์ทำได้ง่ายและไม่เพิ่มภาระแก่ผู้พัฒนามากนัก
- สามารถดีบั๊กโปรแกรมได้ง่าย การออกแบบฮาร์ดแวร์พร้อมๆ กับซอฟต์แวร์ ช่วยให้ผู้ออกแบบสามารถที่จะเพิ่มโมดูลฮาร์ดแวร์สำหรับทดสอบการทำงานของซอฟต์แวร์เข้าไปได้ (Testing H/W module) ซึ่งช่วยให้การดีบั๊กโปรแกรมทำได้ง่ายยิ่งขึ้น โดยโมดูลทดสอบนี้มักจะใช้ในตอนพัฒนาวงจรต้นแบบ (Prototype) เท่านั้น และเมื่อทดสอบการทำงานของซอฟต์แวร์จนถูกต้องแล้วตัวโมดูลฮาร์ดแวร์ส่วนนี้อาจจะถูกถอดออกไปจากวงจรซึ่งจะต้องส่งมอบให้ลูกค้าเพื่อประหยัดค่าใช้จ่ายในการผลิตก็ได้
- ลดขั้นตอนในการตรวจสอบฮาร์ดแวร์ ในระหว่างที่มีการพัฒนาซอฟต์แวร์ต้นแบบเพื่อทดสอบระบบนั้น อาจมีความจำเป็นที่จะต้องเคลื่อนย้ายตัวฮาร์ดแวร์ไปมาระหว่างสภาพแวดล้อมในการทดสอบ และสภาพแวดล้อมของการพัฒนาซอฟต์แวร์ เมื่อมีการเคลื่อนย้ายตัวฮาร์ดแวร์ไปมา ความจำเป็นในการถอดฮาร์ดแวร์บางส่วนเข้าออกจากแผงวงจรบ่อยๆ อาจทำให้เกิดอาการหลวมของสายไฟ หรือการหลวมของซ็อกเก็ตรวมถึง Connector เชื่อมต่อ ทำให้โอกาสที่ฮาร์ดแวร์จะทำงานผิดพลาดใน

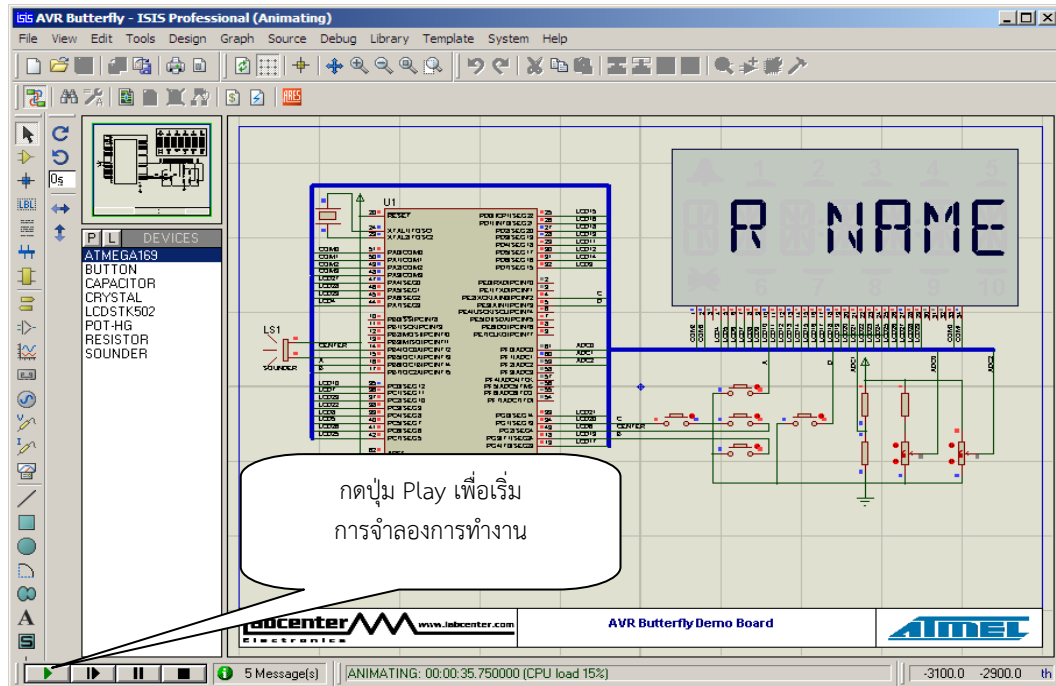
ระหว่างการพัฒนาซอฟต์แวร์มีสูง ดังนั้นเมื่อออกแบบฮาร์ดแวร์และซอฟต์แวร์ร่วมกันตั้งแต่แรก ผู้พัฒนาสามารถเพิ่มส่วนของซอฟต์แวร์สำหรับทดสอบการทำงานของฮาร์ดแวร์เข้าไปได้ (H/W Self-test) โดยโมดูลของซอฟต์แวร์ดังกล่าวจะใช้ในการตรวจสอบความถูกต้องของฮาร์ดแวร์ต้นแบบ เท่านั้น ซึ่งผู้ผลิตสามารถตัดส่วนโมดูลซอฟต์แวร์เหล่านี้ออกไปจากผลิตภัณฑ์ที่ส่งมอบให้ลูกค้าจริง เพื่อจะประหยัดหน่วยความจำของระบบก็ได้

การพัฒนาโปรแกรมเพื่อควบคุมฮาร์ดแวร์ในอดีต จะใช้วิธีการถอดไมโครคอนโทรลเลอร์จากบอร์ดฮาร์ดแวร์เพื่อบันทึกโปรแกรมภาษาเครื่องลงบนตัวไมโครคอนโทรลเลอร์ด้วยเครื่องบันทึกโปรแกรม และนำไมโครคอนโทรลเลอร์ไปเสียบกับบอร์ดวงจรเพื่อดูว่าโปรแกรมที่พัฒนาขึ้นนั้นสามารถทำงานถูกต้องหรือไม่ หากไม่ถูกต้องก็ต้องจะต้องทำการดีบั๊กโปรแกรมใหม่และนำโปรแกรมที่แก้ไขแล้วมาบันทึกลงบนตัวชิพอีกครั้ง ส่งผลให้วงจรการพัฒนาซอฟต์แวร์เพื่อควบคุมฮาร์ดแวร์มีความยุ่งยากเป็นอย่างมาก การใช้อีพรอมอีมิูเลเตอร์ (EPROM Emulator) ช่วยลดความยุ่งยากในการพัฒนาโปรแกรมลงได้ระดับหนึ่งเนื่องจากผู้พัฒนาไม่ต้องเสียเวลาในการย้ายตัวชิพไปมาระหว่างบอร์ดทดลองกับเครื่องบันทึกโปรแกรม ในเวลาต่อมาผู้พัฒนาไมโคร-คอนโทรลเลอร์ได้ออกแบบให้ภายในตัวชิพมีโปรแกรม Bootloader ซึ่งช่วยให้ผู้พัฒนาสามารถโหลดโปรแกรมลงสู่ตัวไมโครคอนโทรลเลอร์ได้ผ่านพอร์ตอนุกรมของเครื่องคอมพิวเตอร์โดยไม่ต้องถอดตัวชิพจากบอร์ดวงจรมาเสียบที่เครื่องบันทึกโปรแกรมอีกต่อไปซึ่งช่วยลดภาระแก่ผู้พัฒนาได้เป็นอย่างดี ปัจจุบันซอฟต์แวร์ช่วยดีบั๊กโปรแกรมได้ถูกพัฒนาความสามารถให้สูงขึ้น ประกอบกับความสามารถของฮาร์ดแวร์ของเครื่อง PC ที่เพิ่มสูงขึ้น ช่วยให้ผู้พัฒนาสามารถใช้ซอฟต์แวร์บน PC ในการจำลองการทำงานของฮาร์ดแวร์ได้โดยตรงโดยไม่ต้องดาวน์โหลดโปรแกรมที่ต้องการทดสอบลงสู่ฮาร์ดแวร์จริงซึ่งช่วยลดระยะเวลาการพัฒนาโปรแกรมควบคุมลงได้มาก ซึ่งซอฟต์แวร์ที่จะใช้ในการทดลองนี้คือซอฟต์แวร์ชุด Proteus ซึ่งจะได้กล่าวรายละเอียดในหัวข้อที่ 6 ต่อไป

เนื่องจากเวลาในการทำแล็บของนักศึกษามีจำกัด การทดลองนี้ถูกออกแบบมาให้ให้นักศึกษาได้ทำการทดลองการออกแบบร่วมกันระหว่างฮาร์ดแวร์และซอฟต์แวร์ที่มีความซับซ้อนไม่มากนัก และง่ายต่อการทำความเข้าใจสำหรับผู้เริ่มต้นศึกษา ซึ่งเมื่อนักศึกษาได้เข้าใจปรัชญาการออกแบบร่วมกันของฮาร์ดแวร์และซอฟต์แวร์ดังกล่าวแล้วก็จะจะเป็นพื้นฐานนำไปสู่การออกแบบระบบสมองกลฝังตัวที่มีความซับซ้อนสูงยิ่งขึ้นในโอกาสต่อไป

7. แนะนำการใช้งานซอฟต์แวร์ชุด Proteus

โปรแกรมชุด Proteus เป็นซอฟต์แวร์สำหรับจำลองการทำงานของวงจรไฟฟ้า ซึ่งสามารถจำลองการทำงานของวงจรได้ตั้งแต่วงจรอนาล็อก วงจรดิจิทัลตลอดจนถึงไมโครโปรเซสเซอร์และไมโครคอนโทรลเลอร์ โดยผู้ทดลองสามารถนำไฟล์นามสกุล .HEX ซึ่งได้จากการแอสเซมเบลอร์โปรแกรมที่เขียนขึ้นด้วยภาษาแอสเซมบลีมาทำการจำลองบนสภาพแวดล้อมการเชื่อมต่อไมโครคอนโทรลเลอร์กับอุปกรณ์ภายนอกเสมือนจริงได้ ตัวชุดโปรแกรม Proteus ประกอบไปด้วยซอฟต์แวร์ย่อยหลายตัว ในแล็บนี้จะใช้โปรแกรมย่อยชื่อ ISIS ในการจำลองการทำงานของไมโครคอนโทรลเลอร์ AVR





รูปที่ 1 โปรแกรม ISIS 7.0 สำหรับจำลองการทำงานของไมโครคอนโทรลเลอร์

8. ข้อมูลสำหรับการลงปฏิบัติการ

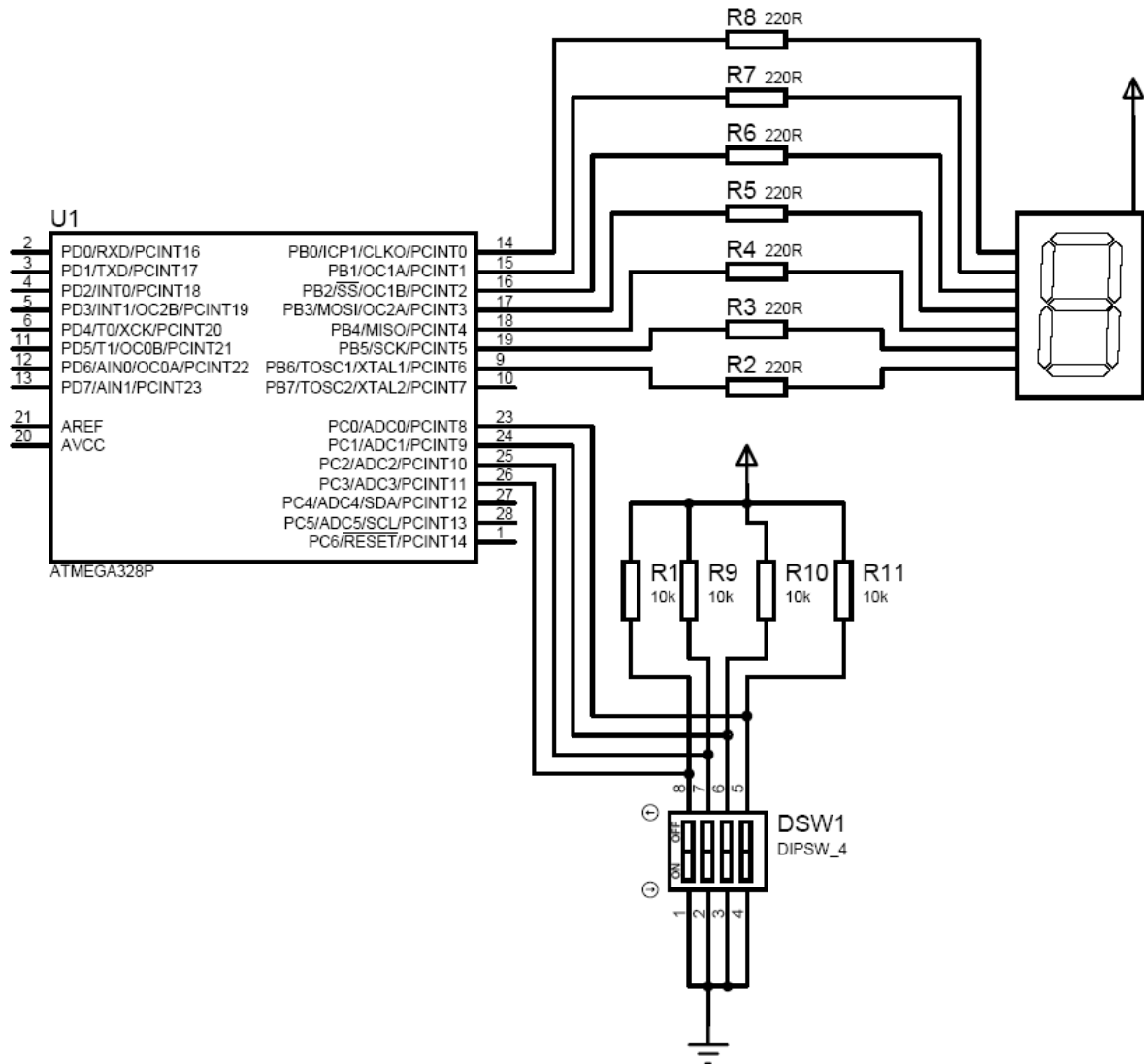
ให้นักศึกษาศึกษาข้อมูลทฤษฎีการออกแบบร่วมกันของฮาร์ดแวร์และซอฟต์แวร์เพิ่มเติมจากเว็บไซต์ <http://www.tik.ee.ethz.ch/education/lectures/hswcd/> ก่อนที่จะมาเข้าลงปฏิบัติการ

9. การทดลอง

9.1 ทดลองใช้โปรแกรม Proteus ในการดีบั๊กและตรวจสอบความถูกต้องของโปรแกรมควบคุม AVR

ให้นักศึกษาเปิดโปรแกรม ISIS ในชุดซอฟต์แวร์ Proteus คลิกเลือก New Design เลือกขนาดของกระดาษเป็น Landscape A4 ทำการวาดวงจรเพื่อทำการอ่านค่าจากสวิตช์ดังรูปที่ 1 กดปุ่ม  จากนั้นดับเบิลคลิกที่  เพื่อเลือกใช้อุปกรณ์จากไลบรารีต่อไปนี้

Category	Sub-Category	Device	คำอธิบาย
Microprocessor ICs	AVR Family	ATMEGA328P	ไมโครคอนโทรลเลอร์ AVR
Optoelectronics	7-Segment Displays	7-SEG-COM-CAT	ชนิด Common Cathode
Switches & Relays	Switches	DIPSW-4	Dip Switch 4 ตัวใน 1 package
Resistors	Generic	RES	ตัวต้านทาน



รูปที่ 2 วงจรอ่านค่าจากสวิตช์แสดงผลทาง 7-segment LED

ทำการเขียนโปรแกรมภาษาซีเพื่อใช้ควบคุมไมโครคอนโทรลเลอร์ของวงจรในรูปที่ 2 โดยตัวโปรแกรมแสดงให้เห็นในรูปที่ 3 การทำงานของตัวโปรแกรม จะทำการอ่านค่าจากดิปสวิตช์ขนาด 4 บิตเข้ามาทางพอร์ต C บิตที่ 0-3 และทำการแปลงค่าไบนารีที่ได้ ซึ่งมีค่า 0-15 ไปแสดงผลทางแอลอีดี 7 เซกเมนต์ ค่า 0-F ให้นักศึกษาเขียนโปรแกรมภาษาซีด้วย AVR Studio จากนั้นใช้คำสั่ง Build เพื่อคอมไพล์ให้เป็นภาษาเครื่องของ AVR ซึ่งจะได้ไฟล์เอาต์พุตนามสกุล .HEX


```

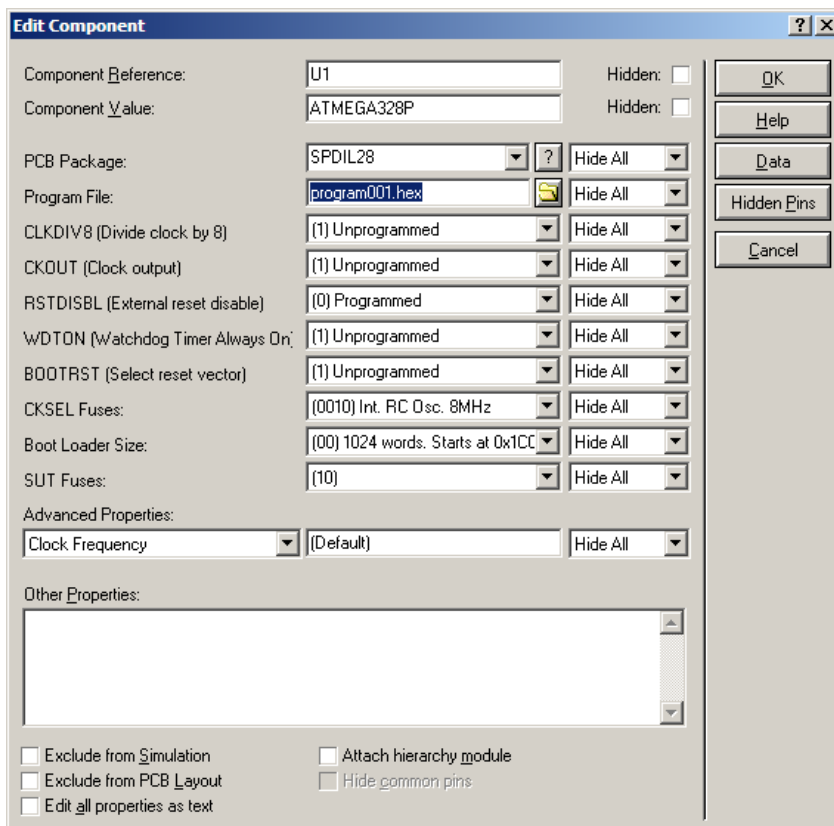
#include <avr/io.h>
int main(void)
{
    DDRC = 0x00;    // set port C as input
    DDRB = 0xFF;    // set port B as output

    unsigned char SWITCH_V, DISP;
    unsigned char LOOKUPTB[] = { 0b00111111,
                                0b00000110,
                                0b01011011,
                                0b01001111,
                                0b01100110,
                                0b01101101,
                                0b01111101,
                                0b00000111,
                                0b01111111,
                                0b01101111,
                                0b01110111,
                                0b01111100,
                                0b00111001,
                                0b01011110,
                                0b01111001,
                                0b01110001 };

    while (1)
    {
        //---read input switch via portB
        SWITCH_V = PINC;
        SWITCH_V &= 0x0F;    //upper 4 bits are masked
        DISP = LOOKUPTB[SWITCH_V];
        PORTB =  DISP ;
    }
}
    
```

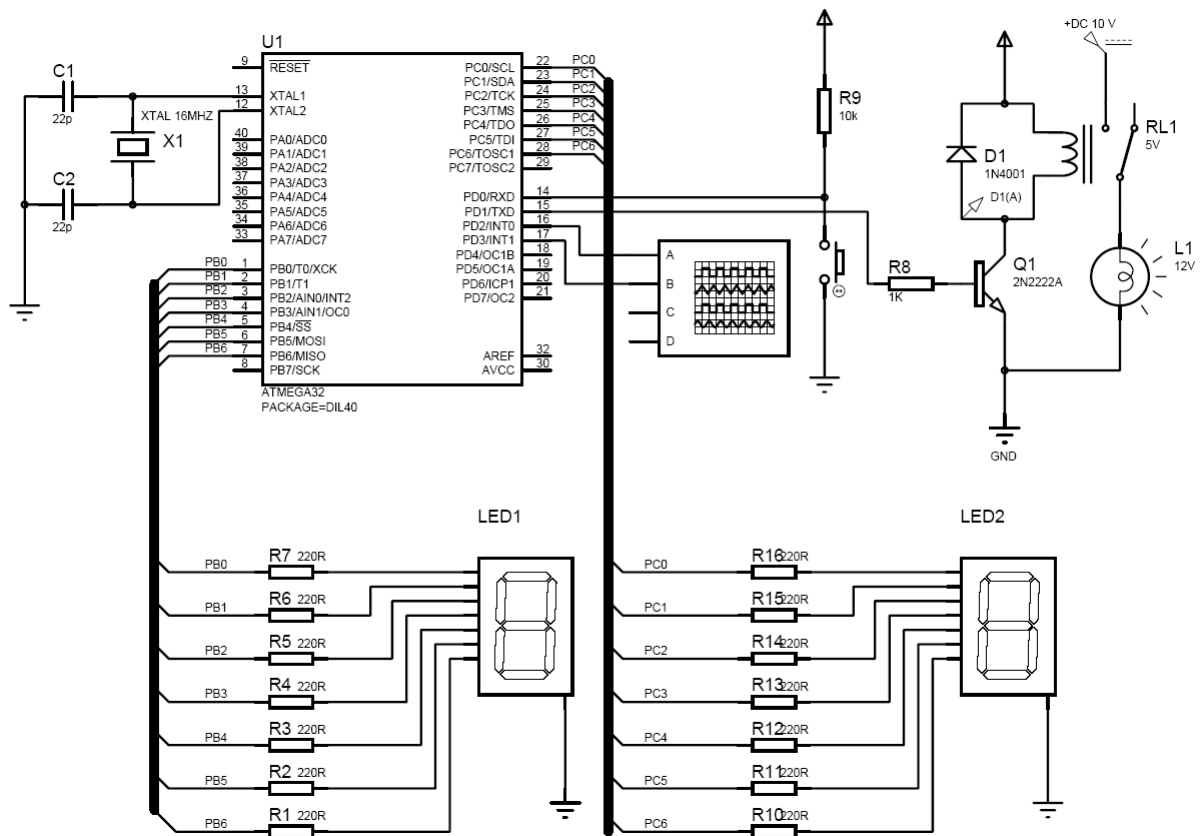
รูปที่ 3 โปรแกรมภาษาซีสำหรับอ่านค่าจากสวิตช์และแสดงผลทางแอลอีดี 7 เซกเมนต์

คลิกขวาที่ตัวชิพ ATMEGA ในโปรแกรม Proteus เพื่อระบุที่อยู่ของไฟล์นามสกุล .HEX ซึ่งชิพนี้จะโหลดโปรแกรมขึ้นมาจำลองการทำงาน ดังรูปที่ 4 ให้นักศึกษาใส่ไฟล์ .HEX ที่ได้จากการคอมไพล์โปรแกรมภาษาซีในรูปที่ 3 จากนั้นกดปุ่ม  ซึ่งอยู่ด้านล่างซ้ายมือในโปรแกรม ISIS เพื่อเริ่มต้นจำลองการทำงานของวงจร



รูปที่ 4 การระบุที่อยู่ของไฟล์ .HEX สำหรับไมโครคอนโทรลเลอร์ในโปรแกรม ISIS




Checkpoint 1	ลายเซ็น	วัน-เดือน-ปี
#checkpoint 1.1 ทดลองจำลองการทำงานของโปรแกรมที่เขียนขึ้นด้วยภาษาซีกับวงจรไมโครคอนโทรลเลอร์ในโปรแกรม ISIS 7.0 บันทึกผลการทดลองที่ได้		
#checkpoint 1.2 ทำการแก้ไขข้อผิดพลาดของโปรแกรมภาษาซีในรูปที่ 3 แล้วดีบั๊กพร้อมทั้งทดสอบการทำงานด้วยโปรแกรม ISIS 7.0		



รูปที่ 5 วงจรควบคุมการเปิดปิดเครื่องใช้ไฟฟ้าด้วยสวิทช์สัมผัสพร้อมระบบหน่วงเวลา

9.2 ออกแบบและทดสอบระบบสวิทช์สัมผัสสำหรับเปิดปิดเครื่องใช้ไฟฟ้า

รูปที่ 5 แสดงวงจรสวิทช์สัมผัสสำหรับเปิดปิดเครื่องใช้ไฟฟ้าซึ่งควบคุมโดยไมโครคอนโทรลเลอร์ AVR รีเลย์ซึ่งทำหน้าที่ควบคุมการตัดต่อวงจรไฟฟ้าถูกควบคุมผ่านพอร์ต D บิตที่ 1 (PD1) ผู้ใช้สามารถควบคุมการเปิดปิดวงจรได้ด้วยการกดสวิทช์(แบบกดติดปล่อยดับ) ซึ่งต่ออยู่กับพอร์ต D บิตที่ 0 (PD0) รายละเอียดการทำงานของวงจรเป็นดังนี้

- เมื่อเริ่มจ่ายไฟให้กับเครื่อง เครื่องใช้ไฟฟ้าจะดับอยู่ และแอลอีดีแสดงสถานะ 
- เมื่อกดสวิทช์ 1 ครั้งจะเป็นการกดเปิดเครื่องใช้ไฟฟ้าโดยมีแอลอีดี 7 เซกเมนต์ 2 ตัวแสดงสถานะของวงจรเป็น 
- เมื่อกดสวิทช์อีก 1 ครั้งจะเป็นการสั่งปิดเครื่องใช้ไฟฟ้า แต่เครื่องใช้ไฟฟ้าจะไม่ถูกตัดไฟในทันที แต่จะมีการหน่วงเวลาออกไป 20 วินาที พร้อมทั้งแสดงการนับลงที่แอลอีดีทั้งสอง เมื่อครบ 20 วินาทีจึงดับเครื่องใช้ไฟฟ้าแล้วแสดงสถานะที่แอลอีดีเป็น 

ในการทดลองนี้ ให้นักศึกษาเขียนโปรแกรม AVR ด้วยภาษาซีเพื่อควบคุมเครื่องใช้ไฟฟ้า และทดสอบการทำงานกับโปรแกรม ISIS 7.0 โดยมีขั้นตอนในการทดลองดังนี้

- ทำการโหลดไฟล์จำลองการทำงานของวงจรในรูปที่ 5 จาก lms เปิดไฟล์ด้วยโปรแกรม ISIS 7.0
- เขียนโปรแกรมควบคุมด้วยภาษาซี โดยเริ่มจากการสร้างส่วนของการหน่วงเวลา เป็นเวลา 1 วินาที ขึ้นมาก่อน (แนะนำให้ใช้ Timer1 ของ AVR ให้เป็นประโยชน์) ทำการทดสอบความถูกต้องของการหน่วงเวลาการกลับลอจิกของพอร์ต D บิตที่ 2 (PD2) เป็นตรงกันข้ามทุกๆ 1 วินาทีโดยใช้ออสซิลอ-สโคปในโปรแกรม ISIS วัดสัญญาณพัลส์รูปคลื่นสี่เหลี่ยมที่ออกมาจากพอร์ต PD2
- เขียนโปรแกรมควบคุมด้วยภาษาซีสำหรับการควบคุมการเปิดปิดเครื่องใช้ไฟฟ้าและพร้อมทั้งระบบหน่วงเวลา ทำการดีบั๊กและทดสอบความถูกต้องด้วยการจำลองการทำงานในโปรแกรม ISIS

Checkpoint 2	ลายเซ็น	วัน-เดือน-ปี
#checkpoint 2.1 ทดสอบความถูกต้องของการหน่วงเวลา 1 วินาที สัญญาณพัลส์ที่ออกมาจากขา PD2 มีค่าความถี่เท่ากับ.....Hz		
#checkpoint 2.2 ทดสอบความถูกต้องของโปรแกรมควบคุมการเปิดปิดเครื่องใช้ไฟฟ้าพร้อมทั้งระบบหน่วงเวลา		

10. คำถามท้ายการทดลอง

- วงจรในรูปที่ 5 หากต้องการออกแบบซอฟต์แวร์ควบคุมเพิ่มเติมเพื่อให้มีความสามารถในการทดสอบฮาร์ดแวร์ (Hardware self-test) ก่อนที่จะใช้งานวงจร (เมื่อเริ่มจ่ายไฟฟ้าให้กับวงจร โปรแกรมจะทำการทดสอบความพร้อมของฮาร์ดแวร์เสียก่อน และรายงานสถานะของระบบให้ผู้ใช้ทราบ) จะต้องทำอย่างไร จงอธิบายแนวคิดในการออกแบบโปรแกรม โดยไม่ต้องมีส่วนของฮาร์ดแวร์เพิ่มเติม

11. เอกสารอ้างอิง

- ปัญญุต ไชยกาพ, 2553, เอกสารประกอบการสอนรายวิชา 241-210 สถาปัตยกรรมไมโครโพรเซสเซอร์และภาษาแอสเซมบลี. <http://lms.psu.ac.th/course/view.php?id=2999>
- Steven F. Barrett, Daniel J. Pack, “Atmel AVR microcontroller primer: programming and interfacing,” Morgan and Claypool, 2008.
- Richard H. Barnett, Larry O’Cull, Sarah Cox, “Embedded C programming and the Atmel AVR,” Thomson Delmar Learning, 2006.