ชุดทดลองไมโครคอนโทรลเลอร์ขนาด 32 บิต

NUVOTON MICROCONTROLLER 32 BIT LAB

นายทิวากร เคี่ยมขาว

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมไฟฟ้าและอิเล็กทรอนิกส์ประยุกต์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีปทุม

ปีการศึกษา 2557

56EE210

หัวข้อโครงงาน	ชุดทดลองไมโครคอนโทรลเลอร์ขนาด 32 บิต
โดย	นายทิวากร เคี่ยมขาว
ภาควิชา	วิศวกรรมไฟฟ้าและอิเล็กทรอนิกส์ประยุกต์
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ เพชร นันทิวัฒนา

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีปทุม อนุมัติให้นับโครงงานวิศวกรรมฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

..... หัวหน้าภาควิชาวิศวกรรมไฟฟ้า
(ดร.ภรชัย จูอนุวัฒนกุล) และอิเล็กทรอนิกส์ประยุกต์

..... อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์ เพชร นันทิวัฒนา)

(วันที่ 2 เดือน ธันวาคม พ.ศ.2557)

รหัสโครงงาน 56EE210

ชุดทดลองไมโครคอนโทรลเลอร์ขนาด 32 บิต
NUVOTON MICROCONTROLLER 32 BIT LAB

บทคัดย่อ

โครงงานนี้ได้ทำการศึกษาเกี่ยวกับชุดทดลอง NUVOTON NUC140 เป็นชุดทดลองไมโครคอนโทรลเลอร์ขนาด 32 บิต เพื่อที่จะศึกษาการทำการงานและเขียนคู่มือการใช้งานในบางส่วนของชุดลอง เพื่อที่จะเป็นตัวอย่างในการศึกษาชุดทดลอง NUVOTON NUC140 และสถาปัตยกรรมของไมโครโพรเซสเซอร์ ARM Cortex ™ - M0 คือ ARM ที่มีหน่วยประมวลผลที่มีขนาดเล็กที่สุดที่มีพื้นที่ซิลิคอนขนาดเล็กพิเศษใช้พลังงานต่ำและใช้การประมวลผลน้อยที่สุดช่วยให้นักพัฒนาทำงานได้อย่างมีประสิทธิภาพมากขึ้น การศึกษาชุดทดลองนี้ก็จะมีด้วยกัน 8 การทดลองซึ่งจะศึกษาการใช้งานของการทดลองการแสดงผลออกทาง LED การทดลองการแสดงผลออกทาง 7-Segment การทดลองการแสดงผลออกทาง Graphic LCD การทดลองการรับค่าจาก Matrix Switch การทดลองการแสดงผลออกทาง Buzzer การทดลองการรับค่าจาก Variable Resistance การทดลองร่วมกับชุดทดลอง ETT ควบคุม DC Motor การทดลองร่วมกับชุดทดลอง ETT ควบคุม Stepper Motor จากการศึกษาชุดทดลองเบื้องต้นทั้งหมด จึงได้จัดทำคู่มือการใช้งานขึ้นมาและตัวอย่างโปรแกรมการใช้งานเบื้องต้นเพื่อที่จะให้สะดวกแก่ผู้ที่จะนำไปใช้งานและนำไปพัฒนาต่อยอดจากเดิมและเพิ่มเติมจากที่มีอยู่ให้ง่ายขึ้นเหมาะแก่การนำไปใช้งานในภายภาคหน้า

กิตติกรรมประกาศ

โครงงานนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี เนื่องจากได้รับความกรุณาจากอาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ เพชร นันทิวัฒนา อาจารย์ที่ปรึกษาโครงงานที่คอยให้ความช่วยเหลือ และคอยให้คำแนะนำในโครงงานเมื่อเกิดปัญหาต่างๆมาโดยตลอดตั้งแต่เริ่มต้นทำโครงงานจนกระทั่งโครงงานนี้สำเร็จลุล่วงตามวัตถุประสงค์ที่ตั้งไว้จึงขอกราบขอบพระคุณอาจารย์เป็นอย่างสูง

ขอกราบขอบพระคุณอาจารย์ประจำภาควิชาวิศวกรรมไฟฟ้าและอิเล็กทรอนิกส์ประยุกต์ทุกท่านและอาจารย์ที่เป็นผู้ประสิทธิ์ประสาทวิชาความรู้ให้แก่ข้าพเจ้าต้องขอขอบคุณมหาวิทยาลัยศรีปทุมที่ให้สถานที่และห้องปฏิบัติการในการจัดทำโครงงานรวมถึงสนับสนุนทดลอง NUVOTON NUC140ในการทำโครงงานจนสำฤทธิ์ผล

ขอขอบคุณรุ่นพี่และเพื่อนๆนักศึกษาสาขาวิชาวิศวกรรมไฟฟ้าและอิเล็กทรอนิกส์ประยุกต์ทุกคนที่เป็นกำลังใจและช่วยเหลือมาโดยตลอด

สุดท้ายขอกราบขอบพระคุณบิดา มารดา และครอบครัวของข้าพเจ้ารวมทั้งผู้มีพระคุณทุกท่านที่คอยให้กำลังใจที่ดีเสมอมา คอยห่วงใย และให้การสนับสนุนทางการศึกษามาโดยตลอด คุณความดีอันใดที่เกิดจากโครงงานฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

สารบัญภาพ(ต่อ)

หน้า

บทที่ 1
บทนำ

สถาปัตยกรรมของ ARM ไมโครโพรเซสเซอร์ ARM Cortex ™ - M0 คือ ARM หน่วยประมวลผล ที่เล็กที่สุดที่มีพื้นที่ซิลิคอนขนาดเล็กพิเศษใช้พลังงานต่ำและใช้การประมวลผลน้อยที่สุดช่วยให้นักพัฒนาทำงานได้อย่างมีประสิทธิภาพมากที่สุด หน่วยประมวลผล Cortex - M0 ยังช่วยให้การใช้งานในการผสมสัญญาณอนาล็อกเข้ากับอุปกรณ์สัญญาณดิจิตอล Cortex - M0เป็นอุปกรณ์ที่พัฒนามาจากARM926EJ-Sและสามารถแยกประเภทออกไป แบบ Cortex – R Series และ Cortex – A Series เป็นรุ่นที่นิยมใช้กันในจำพวกอุปกรณ์มือถือ แท็บเล็ต ที่ใช้ในด้านการประมวลผลแอปพลิเคชั่นซึ่งมีหลากหลายรุ่นในปัจจุบัน

1.1 ความสำคัญของปัญหา

ในปัจจุบันเทคโนโลยีด้านระบบสมองกลฝังตัว (Embedded System) เข้ามามีบทบาทอย่างมากในชีวิตประจำวัน และมีการพัฒนาเทคโนโลยีด้านตัวประมวลผลขนาดเล็กที่ใช้พื้นที่ในการประมวลผลน้อย ปัจจุบันตัวประมวลผลที่นิยมใช้กันมากในในโทรศัพท์มือถือ Tablet และระบบที่มีการคำนวณหันก็หันมาใช้ไมโครคอนโทรลเลอร์และไมโครคอนโทรลเลอร์ตระกูล ARM ที่มีอย่างแพร่หลายเนื่องจากมีหลายบริษัทซื้อลิขสิทธิ์สถาปัตยกรรมภายในไปพัฒนาต่อยอดใช้งานในด้านต่างๆ ที่บริษัทนั้นได้ออกแบบอุปกรณ์ระบบสมองกลฝังตัวไว้ จากที่สาขาวิชาวิศวกรรมไฟฟ้าและอิเล็กประยุกต์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีปทุม ได้รับความร่วมมือสนับสนุนชุดทดลอง NUVOTON รุ่น Nu-LB-NUC140 ชิปตระกูล ARM Cortex™-M0 จากประเทศไต้หวัน จำนวน12 ชุด เพื่อไว้ใช้ในการเรียนการสอนวิชาไมโครโปรเซสเซอร์ วิชาไมโครคอนโทรลเลอร์และระบบสมองกลฝังตัว

ดังนั้นเพื่อให้สามารถนำชุดการทดลองที่ได้รับการสนับสนุนมาใช้ประโยชน์สูงสุดในการประกอบการเรียนการสอน ทางผู้จัดทำโครงงานนี้จึงได้เสนอการออกแบบชุดทดลองไมโครคอนโทรลเลอร์ขนาด 32 บิต NUVOTON พร้อมคู่มือการใช้งาน ใบงานการทดลอง และเฉลยใบงานการทดลอง เพื่อง่ายที่จะศึกษาและนำไปพัฒนาต่อยอดจากที่ผู้เขียนได้เริ่มต้นไว้ในภายภาคหน้า

1.2 วัตถุประสงค์

1 เพื่อที่จะศึกษาไมโครคอนโทรลเลอร์ตระกูล ARM ของบริษัท NUVOTON NUC 140

2 เพื่อที่จะนำผลการทดลองของบอร์ด NUVOTON ไปใช้ร่วมกับบอร์ด ETT LAB3A ในการเรียนการสอน

1.3 ประโยชน์ของโครงงาน

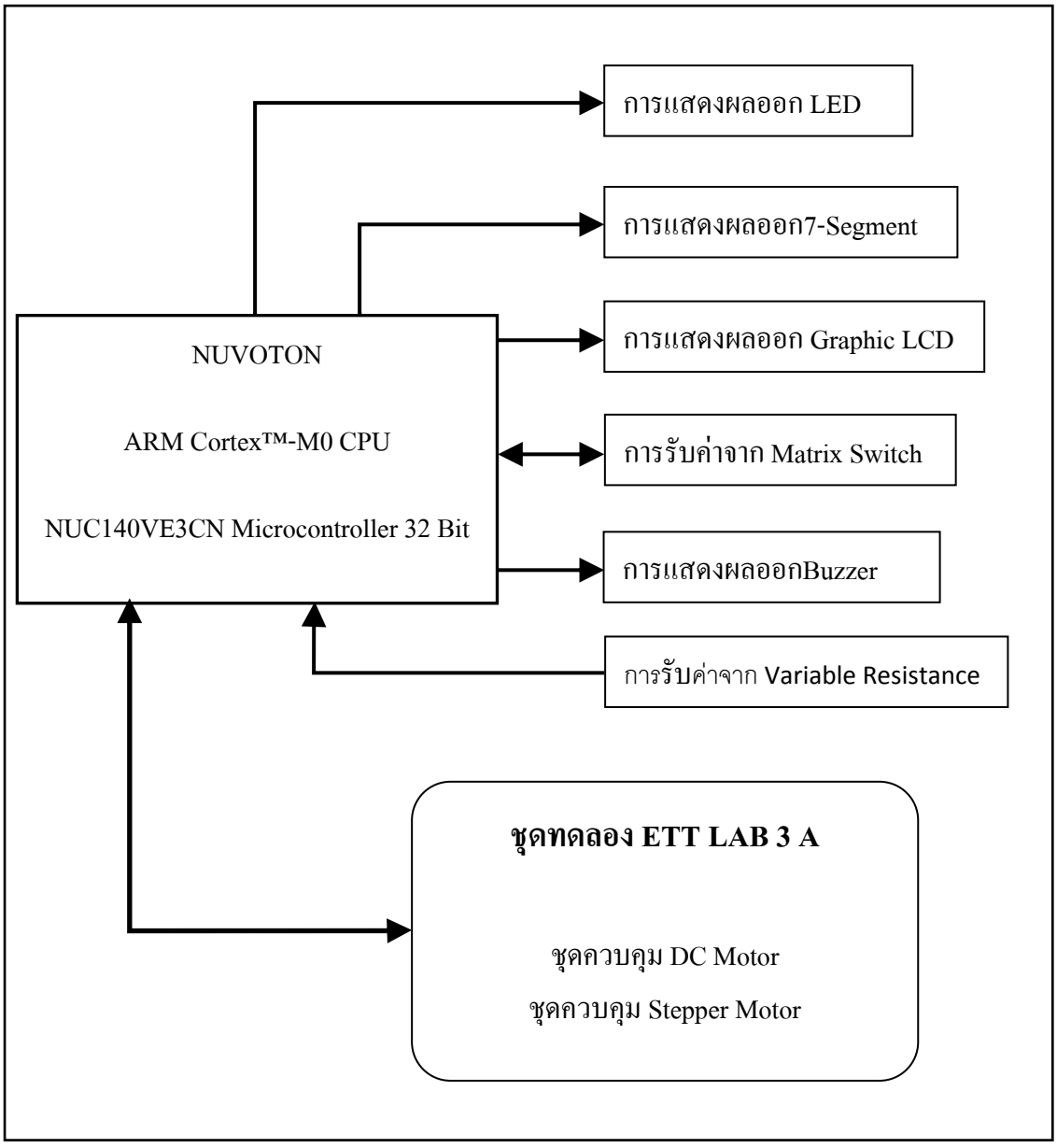
1 มีความรู้เกี่ยวกับไมโครคอนโทรลเลอร์ขนาด32บิต

2 ใช้เป็นตัวอย่างในการสอนไมโครคอนโทรลเลอร์

3 ใช้เป็นการทดลองปฏิบัติไมโครคอนโทรลเลอร์ต่อยอดจากห้องปฏิบัติการที่เรียน

4 สามารถนำข้อมูลที่มีไปควบคุมระบบต่างๆได้

1.4 โครงสร้างโครงงาน



ภาพที่ 1.1 โครงสร้างของโครงงาน

1.5 ขอบเขตของโครงงาน

ชุดทดลองไมโครคอนโทรลเลอร์ขนาด 32 บิต ของ NuMicro ARM Cortex™-M0 NUC 140 ประกอบด้วยหัวข้อต่อไปนี้

- การกำหนดค่าการใช้งาน NuMicro ARM Cortex™-M0 NUC 140
- การทดลองการแสดงผลออกทาง LED
- การทดลองการแสดงผลออกทาง 7-Segment
- การทดลองการแสดงผลออกทาง Graphic LCD
- การทดลองการแสดงผลออกทาง Buzzer
- การทดลองการรับค่าจาก Variable Resistance
- การทดลองการรับค่าจาก Matrix Switch
- การทดลองร่วมกับชุดทดลอง ETT
    - DC Motor
    - Stepper Motor

1.6 แผนการดำเนินงาน

ตารางที่ 1.1 แผนการดำเนินงาน

ลำดับ	ขั้นตอนการดำเนินงาน	EEG491				EEG492			
		ก.พ.	มี.ค.	เม.ย.	พ.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.
1.	ศึกษาและหาข้อมูลที่เกี่ยวข้อง - ศึกษาหาข้อมูลเกี่ยวกับบอร์ดไมโครคอนโทรลเลอร์ - ศึกษารายละเอียดบอร์ดไมโครคอนโทรเลอร์ - เรียบเรียงรูปเล่มคู่มือการใช้งานบอร์ด	▬	▬	▬	▬				
2.	ทำตัวอย่างปฏิบัติการเกี่ยวกับบอร์ด ไมโครคอนโทรเลอร์				▬	▬			
3.	พัฒนาต่อยอดจากการทำปฏิบัติการทดลองบอร์ด ไมโครคอนโทรลเลอร์					▬	▬		
4.	รวบรวมรายละเอียดการปฏิบัติและพัฒนาเพิ่มเติม พร้อมทำรูปเล่มปฏิบัติการของการทดลอง							▬	▬

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 แนะนำ ARM Cortex-M0[1]

ARM Cortex-M0 เป็นไมโครโปรเซสเซอร์ขนาด32บิตที่พัฒนาภายใต้สถาปัตยกรรม ARM926EJ-S หรือ ARMv6 M ที่ใช้หน่วยประมวลผล Cortex M0® ARMv6 M โดยพัฒนาเพิ่มประสิทธิภาพขึ้นและได้ ARM Cortex-M โดยสามารถแบ่งออกเป็น ARM Cortex-M0, ARM Cortex-M1, ARM Cortex-M3 และ SC300 โดยสถาปัตยกรรมภายในมีคุณสมบัติที่น่าสนใจดังนี้

- เป็นโปรเซสเซอร์แบบไฮเออราชิคอล(Hierarchical)ที่รวมเอาซีพียูคอร์เข้ากับระบบเพอริเฟอรัลขั้นสูง
- ซีพียูคอร์เป็นสถาปัตยกรรมแบบฮาวาร์ด (Harvard Architecture) ไปป์ไลน์ 3 ระดับพร้อมรองรับการกระโดดไปทำงานแบบทันทีทันใดรองรับคำสั่ง Thumb และ Thumb-2
- หน่วยประมวลผลทางคณิตศาสตร์และลอจิก (ALU) มีโมดูลหารแบบฮาร์ดแวร์
- สามารถคูณเลขขนาด 16 และ 32 บิตได้อย่างรวดเร็ว
- มีตัวควบคุมเวกเตอร์อินเตอร์รัปต์ซับซ้อน (NVIC : Nested Vector Interrupt Controller)
- ใช้ระบบบัสแบบเมตริกซ์
- ประสิทธิภาพความเร็วในการประมวลผล 1.21 DMIPS / MHz

สถาปัตยกรรม ARMv6 M ที่ใช้หน่วยประมวลผล Cortex M0® ARMv6 M มีชุดคำสั่งที่ใช้ง่าย รวมถึงเทคโนโลยีการใช้งานที่ง่ายในระดับ 2 คาดการว่าสถาปัตยกรรมไมโครโปรเซสเซอร์ 32 บิตรุ่นใหม่แบบทันสมัย มีการเข้าที่รหัสที่สูงและประสิทธิภาพยอดเยี่ยม มีความจุของข้อมูลมากกว่าไมโครคอนโทรลเลอร์ขนาด 8 บิต และ 16 บิต ARM Cortex-M มีการประมวลผลที่มีขนาดข้อมูลที่มากกว่าขนาด 8 บิตและ 16 บิตมีข้อดีสำคัญในแง่ของการลดความต้องการการใช้หน่วยความจำและการเพิ่มประสิทธิภาพการใช้งานของชิปหน่วยความจำแบบแฟลช

2.1 สถาปัตยกรรมของ ARM Cortex-M0 [3]

ไมโครโพรเซสเซอร์ ARM Cortex-M0 คือ ARM หน่วยประมวลผล ที่เล็กที่สุดที่มีพื้นที่ซิลิคอนขนาดเล็กพิเศษใช้พลังงานต่ำและใช้การประมวลผลน้อยที่สุดช่วยให้นักพัฒนาทำงานได้อย่างมีประสิทธิภาพมากที่สุด หน่วยประมวลผล Cortex-M0 ยังช่วยให้การใช้งานในการผสมสัญญาณอนาล็อกเข้ากับอุปกรณ์สัญญาณดิจิตอล Cortex-M0 การประมวลผลประมาณ 16μW/MHz พื้นที่หน่วยประมวลผลต่ำกว่า 12 กิโลไบต์ โดยที่ไม่มีใครเทียบ ARM ในฐานะที่เป็นผู้นำในด้านเทคโนโลยีพลังงานต่ำและยังเป็นอุปกรณ์ที่ใช้พลังงานต่ำเป็นพิเศษ ยังสามารถที่จะแยกประเภทของ ARM Cortex ได้อีก 3 ประเภทคือ

2.1.1 ARM Cortex A Series เป็นซีพียูที่เน้นให้นำไปใช้กับการประมวลผลแอปพลิเคชั่นสำหรับงานที่มีความซับซ้อนสูงมากเช่น ระบบปฏิบัติการ(OS : Operrating System)เช่น Symbian,Linux หรือ Window Embedded โดยซีพียูสำหรับงานลักษณะนี้ต้องมีความสามารถในการประมวลผลข้อมูลสูงสุด ระบบหน่วยความจำเสมือนหรือ Virtual Memory System ต้องสามารถรองรับหน่วยบริหารหน่วยความจำหรือ MMU (Memory Management Unit) ได้เป็นอย่างดีรองรับการทำงานกับภาษาจาวาตัวอย่างของผลิตภัณฑ์ที่ใช้ซีพียูระดับนี้คือ โทรศัพท์เคลื่อนที่ สมาร์ทโฟน และพ็อกเก็ตพีซีเป็นต้น

2.1.2 ARM Cortex R Series ใช้กับการประมวลผลแบบ Real-Time ความเร็วสูง ซีพียูต้องมีความเสถียรภาพและความน่าเชื่อถือในการทำงานระดับสูง ตัวอย่างการพัฒนาชิปไปสู่การใช้งานจริงของซีพียูในรุ่นนี้คือชิปควบคุมฮาร์ดดิส เนื่องจากฮาร์ดดิสในปัจจุบันมีความจุสูงดังนั้นการควบคุมระบบหยุดและเคลื่อนที่ของหัวอ่านต้องมีความเร็วและแม่นยำสูง เพื่อให้สามารถเข้าข้อมูลได้อย่างถูกต้องและรวดเร็ว

2.1.3 ARM Cortex M Series เป็นซีพียูที่ออกแบบมาเพื่อต้องการนำไปพัฒนาผลิตภัณฑ์ที่ต้องการความสามารถในการประผลสูง งบประมาณต่ำ กินพลังงานต่ำ มีวัตถุในแต่ละแอปพลิเคชั่นเจาะจงอย่างชัดเจนซึ่งนั้นก็คือวัตถุประสงค์ในการทำงานของไมโครคอลโทรลเลอร์นั่นเองดังนั้นจากชื่อรุ่นของซีพียู ARMv6 M ตัวอักษร M จึงอาจตีความได้ว่าซีพียูนี้เหมาะสำหรับการไปพัฒนาเป็ฯชิปไมโครคอลโทรลเลอร์ความสามารถสูงต่อไป

2.2 ชุดคำสั่ง Thumb-2 กับการพัฒนาไมโครโปรเซสเซอร์ ARM Cortex-M0[4]

ARM ได้พัฒนาชุดคำสั่ง Thumb-2 ขึ้นมาตั้งแต่ปี 2003 ซึ่งเป็นชุดคำสั่งที่สามารถรองรับการทำงานกับคำสั่ง Thumb ทั้งแบบ 16 และ 32 บิต เนื่องจากปกติแล้วคำสั่งของ Thumb จะเป็นคำสั่ง16 บิตที่เป็นส่วนหลักในการทำงาน

สำคัญของการพัฒนาชุดคำสั่งนี้คือเป็นคำสั่งที่ใช้งานง่ายมีความต้องการหน่วยความจำน้อยและเป็นคำสั่งที่มีประสิทธิภาพสูง

เหตุผลหลักประการหนึ่งของการพัฒนาชุดคำสั่ง Thumb-2 คือต้องการให้ซีพียู ARMv6 M สามารถทำงานกับคำสั่ง 32บิตได้ โดยใช้หน่วยความจำที่น้อยและคล่องตัวกว่า หากไม่มีการพัฒนาชุดคำสั่ง Thumb-2 การที่โปรเซสเซอร์ ARMv6 M จะทำงานกับคำสั่ง 32 บิตก็ต้องพัฒนาไปใช้คำสั่ง ARM ซึ่งทำให้ต้องการหน่วยความจำในการประมวลผลมาก ต้นทุนในการพัฒนาชิปสูง

Cortex-M0 เป็นโปรเซสเซอร์ที่มีความพร้อมในการรองรับชุดคำสั่ง Thumb-2 ได้อย่างสมบูรณ์ นั่นจึงเป็นสาเหตุที่ทำให้ Cortex-M0 โปรเซสเซอร์มีความแตกต่างจาก ARM โปรเซสเซอร์ดั้งเดิมเนื่องจากชุดคำสั่ง Thumb-2 รองรับการทำงานทั้งคำสั่ง 16 และ 32 บิต ทำให้ Cortex-M0 ไม่ต้องสลับการประมวลผลคำสั่งไปมาระหว่างชุดคำสั่ง Thumb มาตรฐาน (16บิต) กับ ARM (32บิต) ดังรุ่น ARMv4T (ซึ่งก็คือARM926EJ-S) ส่งผลให้การทำงานเร็วขึ้นมีค่าการหน่วงเวลาประมวลผลลดลงสามารถสร้างโปรแกรมที่ซับซ้อนด้วยขนาดของหน่วยความจำที่น้อย

2.3 ไปป์ไลน์ของ Cortex-M0

Cortex-M0 คือโปรเซสเซอร์ที่มีไปป์ไลน์ 3 ช่วง ประกอบด้วยช่วงอ่านหรือเฟตช์ (Fetch) คำสั่ง, ช่วงถอดรหัสคำสั่ง (Decode) และช่วงเอ็กซิคิวต์ (Execute) หรือการทำคำสั่งอย่างไรก็ตาม หาก Cortex-M0 โปรเซสเซอร์ทำงานกับคำสั่ง 16 บิต ซีพียูไม่มีความจำเป็นต้องเฟตช์หรืออ่านคำสั่งทุกๆไซเกิล เนื่องจากซีพียูสามารถรองรับคำสั่งขนาด 32 บิตได้ ดังนั้นจึงสามารถอ่านคำสั่ง 16 บิตได้ 2 คำสั่งในคราวเดียวโดยคำสั่ง 16 บิตอีกคำสั่งหนึ่งจะถูกพักได้ในบัฟเฟอร์เพื่อรอการประมวลผลต่อไปทำให้ซีพียูกระทำตามคำสั่งได้เร็วขึ้น

2.4 ส่วนประกอบในระบบโปรเซสเซอร์ของ Cortex-M0

ส่วนประกอบต่างๆในระบบโปรเซสเซอร์ มีดังนี้

2.4.1 Cortex-M0 : ประกอบด้วยรีจีสเตอร์ หน่วยคำนวณทางคณิตศาสตร์และลอจิก บัสข้อมูลบัสคำสั่งและส่วนเชื่อมต่อระบบบัส

2.4.2 ส่วนควบคุมการอินเตอร์รัปต์ : (NVIC : Nested Vectored Interrupt Controller) : เป็นส่วนที่จัดการอินเตอร์รัปต์ทั้งหมด สามารถเปลี่ยนแปลงไปตามผู้ผลิตไมโครคอนโทรลเลอร์

2.4.3 SYSTICK ไทเมอร์ของระบบ : เป็นไทเมอร์นับถอยหลังใช้ในการสร้างสัญญาณการอินเตอร์รัปต์ตามช่วงเวลาที่กำหนด มีลักษณะการทำงานคล้ายกับการทำงานของวอทช์ด็อกไทเมอร์ในไมโครคอลโทรเลอร์ 8 บิต โดยไทเมอร์ SYSTICK เป็นส่วนหนึ่งของ NVIC

2.4.4 ส่วนป้องกันหน่วยความจำ(MPU : Memory Protection Unit) : ใช้ในการป้องส่วนของข้อมูลในหน่วยความจำภายในที่สำคัญของซีพียูเป็นส่วนเสริมที่มีมาให้ในไมโครคอนโทรลเลอร์ Cortex-M0

2.4.5 บัสเมตริกซ์(Bus Matrix) : เป็นหัวใจของระบบบัสภายในของ Cortex-M0 เนื่องจากเป็นส่วนจัดการเชื่อมต่อทั้งหมดของบัสซีพียู (AHB : Advanced High-Performance Bus) ในการถ่ายทอดข้อมูลจากส่วนต่างๆของโปรเซสเซอร์

2.4.6 ส่วนเชื่อต่อบัสซีพียูกับบัสเพอริเฟอรัล (AHB-to-APB Bus Bridge) : ในการเชื่อมต่ออุปกรณ์เพอริเฟอรัลเข้ากับบัสซีพียู

2.5 NUC140VE3CN ไมโครคอนโทรลเลอร์ Cortex-M0 ของ NUVOTON[6]

2.5.1 ซีพียูคอร์ : ARM Cortex-M0 สามารถอัพความไวการประมวลผลได้ถึง 50 MHz ประสิทธิภาพของความเร็วในการทำงาน1.21DMIPS/MHz หรือเท่ากับ Dhyston 2.1 เมื่อไม่มีสภาวะการรอคอยเพื่อเข้าถึงหน่วยความจำ มีตัวหารเลขแบบฮาร์ดแวร์และหน่วยประมวลผลทางคณิตศาสตร์สามารถคูนเลข 32 บิตได้ในเวลา 1 ไซเกิลสัญญาณนาฬิกา

2.5.2 หน่วยความจำ : โปรแกรมหน่วยความจำภายในขนาด 32/64/128 กิโลไบต์

– โหลดหน่วยความจำแบบ SRAM ฝังตัว (LDROM) ขนาด 4 กิโลไบต์

– การตั้งค่าการใช้งานข้อมูล แฟรช มีที่อยู่และขนาด 128 กิโลไบต์ ระบบ แฟรช ข้อมูล แบบฝังตัวขนาด 4 กิโลไบต์ 32 กิโลไบต์ และ 64 กิโลไบต์ของหน่วยความจำภายในระบบ

– เลือกรูปแบบการใช้งานขนาด 4 กิโลไบต์ 8 กิโลไบต์และ 16 กิโลไบต์

– มีโหมดการใช้งาน PDMA โหมด

2.5.3 ระบบสัญญาณนาฬิกา : สามารถเลือกใช้งานนาฬิกาแบบยืดหยุ่นได้

- มีออสซิลเลเตอร์ภายในขนาด 22.1184 MHz (ตัด 1% ที่ใช้สำหรับระบบการดำเนินงาน) และ 10 KHz ของการใช้งานในโหมดประหยัดพลังงาน

– PLL 1 ถึง 50 MHz การดำเนินการของระบบประสิทธิภาพสูงสุดที่อุปกรณ์จะรับได้

–การเข้าใช้งานจากคริสตัลภายนอกสำหรับดำเนินการที่ต้องการเวลาที่แม่นยำโดยใช้ที่ความถี่ 4 ~24 เมกะเฮิร์ต

–การเข้าใช้งานคริสตัลภายนอกสำหรับฟังก์ชัน RTC และการเข้าโหมดประหยัดพลังงานที่ความถี่32.768 กิโลเฮิร์ต

2.5.4 ไทม์เมอร์ : มี 4 ชุด รองรับการทำงานตั้งแต่ขนาด 4,8,16 และ 32 บิตมีนาฬิกาแยกการทำงานอย่างอิสระสำหรับจับเวลาแต่ละชุดคำสั่งการทำงาน โดยเลือกการทำงานแบบครั้งเดียว แบบประจำ แบบสลับโหมดการทำงานอย่างต่อเนื่องนับการดำเนินงานโหมด

2.5.5 PWM : การกำหนดจังหวะนาฬิกา

- เป็นตัวกำเนิดจังหวะนาฬิกา PWM 16 บิต และมีสัญญาณ PWM เอาท์พุตที่ 4 กับ 8 บิต
- แสดงผลแบบเดี่ยวหรือเสริมคู่การแสดงผลของ PWM
- เป็นตัวกำเนิดจังหวะนาฬิกา PWM แต่ละตัวพร้อมกับตัวเลือกนาฬิกาการทำงาน
- สามารถเลือกการทำงานตั้งแต่ 8 บิต จนถึง 16 บิต ได้และสามารถแยกการทำงานกันระหว่าง 8 บิต กับ 16 บิต ที่ทำงานในคำสั่งเดียวกันได้
- มีการจับสัญญาณการขัดจังหวะโดยที่ไม่มีการเสียเวลาการทำงานของระบบ

2.5.6 ADC : การแปลงสัญญาณจากอนาล็อกเป็นดิจิตอล

- ป้อนข้อมูลต่างกัน 4 ช่องทางรวมถึงสามารถป้อนอินพุตเดียวสิ้นสุดที่ 8 ช่องทาง
- มีการสแกนข้อมูลทั้งแบบต่อเนื่องและสแกนรอบเดียว
- มีรีจิสเตอร์แสดงผลแต่ละช่องทาง
- ใช้แรงดันไฟฟ้าในการตรวจขีดจำกัดของความไวการประมวลผล
- การเรียกใช้ซอฟต์แวร์โปรแกรมจากภายในหรือภายนอก

2.5.7 การเชื่อมต่ออินเตอร์เฟส

- สามารถเลือกการทำงานในโหมดหลักหรือโหมดรองได้
- สารถถ่ายโอนข้อมูลระหว่างโหมดได้แบบ 2 ทิศทาง
- ใช้สัญญาณนาฬิกาในการควบคุมการส่งข้อมูล
- สามารถส่งข้อมูลผ่านพอร์ต RS485

2.5.8 I2S

- การอินเตอร์เฟสกับตัวแปลงสัญญาณเสียงจากภายนอก
- ทำงานเป็นโหมดหลักหรือโหมดรอง
- สามารถเลือกขนาดของข้อมูล 8, 16, 24 และ 32 บิต ใช้ในการคำนวน
- ใช้ในการฟังเสียงทั้งแบบโมโนและสเตอริโอ

2.5.9 USB 2.0

- ชุดของอุปกรณ์ USB 2.0 FS ความไวในการประมวลผล 12 Mbps
- มีตัวรับส่งสัญญาณ USB ภายในชิป
- การอินเตอร์รัปต์ 1 ช่องทางสามารถใช้ได้พร้อมกัน 4 อินเตอร์รัปต์
- ควบคุมการส่งข้อมูลเสียงผ่านอินเตอร์รัปต์
- สามารถส่งโปรแกรมได้ 6 โปรแกรม
- ภายใน SRAM เป็นบัฟเฟอร์ USB ขนาด 512 ไบต์
- สามารถควบคุมการทำงานด้วยระยะไกลโดยการทำ Remote ควบคุม

2.5.10 การคอมพาราเตอร์แบบอนาล็อก

- พาราเตอร์แบบอนาล็อกอัพได้ถึงขั้น2
- อินพุตภายนอกหรือภายในมีแถบช่องว่างของแรงดันที่แรงดันลบ
- อินเตอร์รัปต์เมื่อเปรียบเทียบกับผลการเปลี่ยนแปลง
- เลือกการปลุกการทำงานได้

2.5.11 RTC การควบคุมการสร้างฐานเวลาจริง

- การใช้ซอฟแวร์เลือกค่าในการตั้งความถี่หรือ FCR
- การนับเวลา (วินาที นาที ชั่วโมง) และสามารถตั้งปฏิทิน (วัน เดือน ปี)
- สามารถตั้งเวลาการปลุก (วินาที นาที ชั่วโมง วัน เดือน ปี)

2.5.12 บัส EBI เฉพาะ100 ขาและ 64 ขา เท่านั้น

- สามารถเข้าถึงข้อมูลขาด 64 กิโลไบต์ในโหมด 8 บิตหรือ 128 กิโลไบต์ในโหมด 16 บิต
- ความกว้างของการรับข้อมูล 8-16 บิต
- ในโหมด 16 บิตสามารถเขียนข้อมูลลงได้ถึงระดับไบต์

2.5.13 Brown-out Detector ; ตัวจับแรงดันลดลง

- มี 4 ระดับเลือกใช้: 4.5V / 3.8V / 2.7V / 2.2V
- สามารถเลือกการใช้งานและรีเซ็ตค่าใหม่ได้

2.5.14 GPIOs พอร์ตต่ออินเตอร์เฟซใช้งานภายนอก

- ขาใช้งานทั่วไปของ I/O (GPIO) 80 ขา
- โหมดการใช้งาน I/O มี4โหมด
    - Quasi Bi-Direction
    - Push-Pull Output
    - Open-Drain Output
    - Input Only With High Impendence
- เลือกป้อนทริกเกอร์ TTL/Schmitt
- ขา GPIO ทั้งหมดสามารถถูกกำหนดให้เกิดการอินเตอร์รัปต์การใช้งานในขอบเขตที่กำหนดไว้ได้

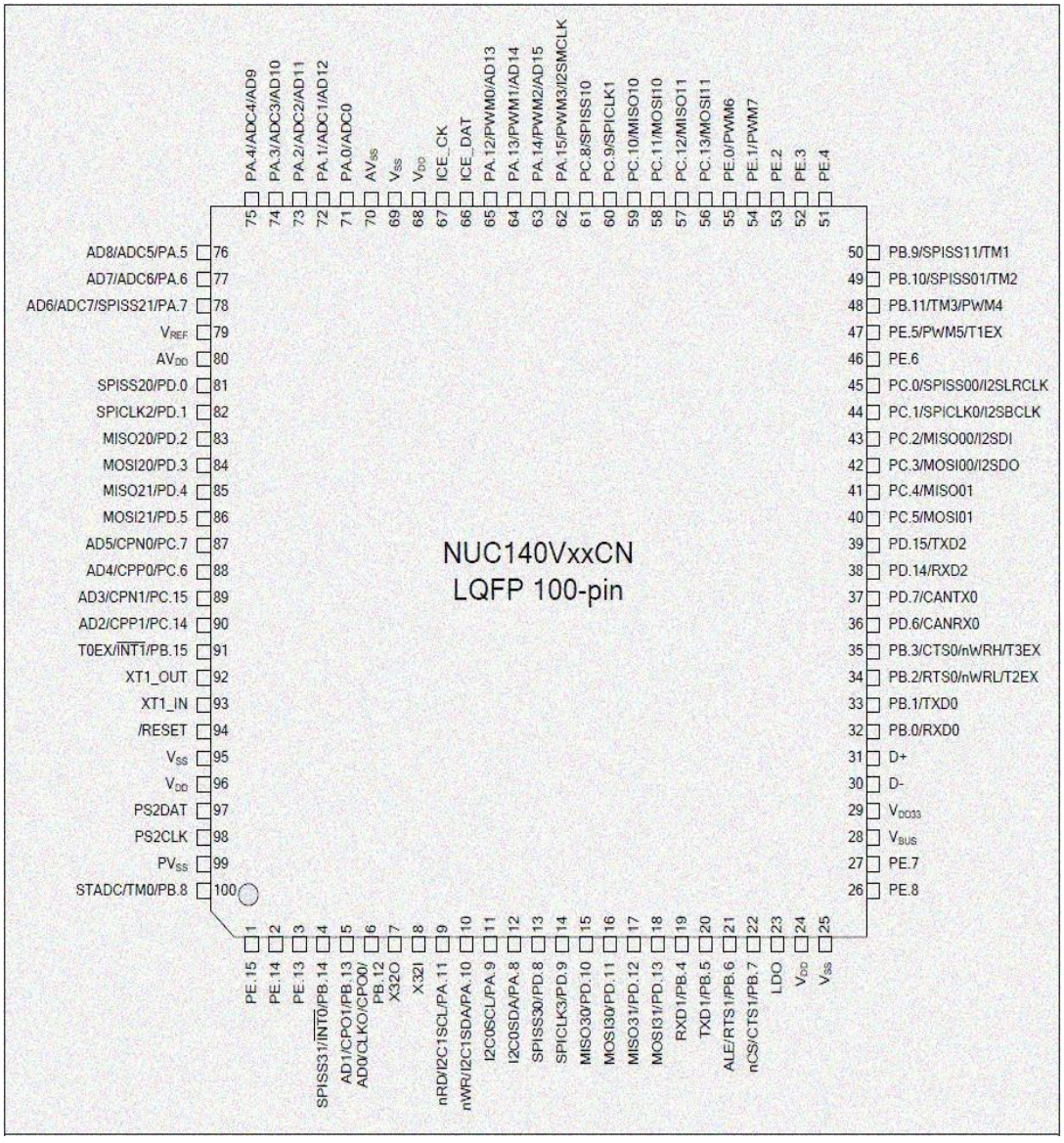
2.5.15 แรงดันไฟฟ้าที่ใช้

- 2.5V ถึง 5.5V

2.5.16 อุณหภูมิการใช้งาน

- ที่ -40° C ~ 85° C

2.6 ขาการใช้งาน [1]



ภาพที่ 2.1 ขาการใช้งาน NUC140VE3CN [1]

2.7 เริ่มต้นสร้างโปรเจค[7]

2.7.1 การลงโปรแกรม Keil uVision 4

เปิดโปรแกรมจากแผ่นซีดีข้อมูลที่ทางบริษัท NUVOTON ที่ให้มาในกล่องไมโครคอนโทรลเลอร์หรือทำการดาวน์โหลดซอฟต์แวร์ของบริษัท Keil โดยตรงที่ www.keil.com ให้พิจารณาในส่วน Software Download ทางด้านซ้ายมือคลิกเลือกหัวข้อ Evaluation Software เลือกหัวข้อย่อย ARM ในที่นี้จะใช้โปรแกรมทางที่บริษัท NUVOTON ให้มา ทำการเปิดโปรแกรมจะแสดงหน้าต่างดังภาพที่ 2.2

ภาพที่ 2.2 หน้าต่างโปรแกรมในแผ่นซีดีข้อมูลของบริษัท NUVOTON

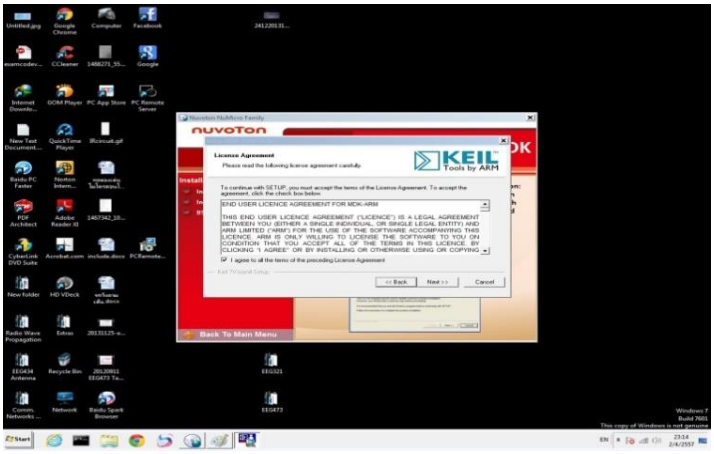
คลิกเลือกโปรแกรม Keil จะปรากฏหน้าต่างดังภาพที่ 2.3

ภาพที่ 2.3 หน้าต่างเลือกโปรแกรม Keil

ให้เลือก Install Keil RVMDK EV Version และ Install Nu-Link Keil Driver

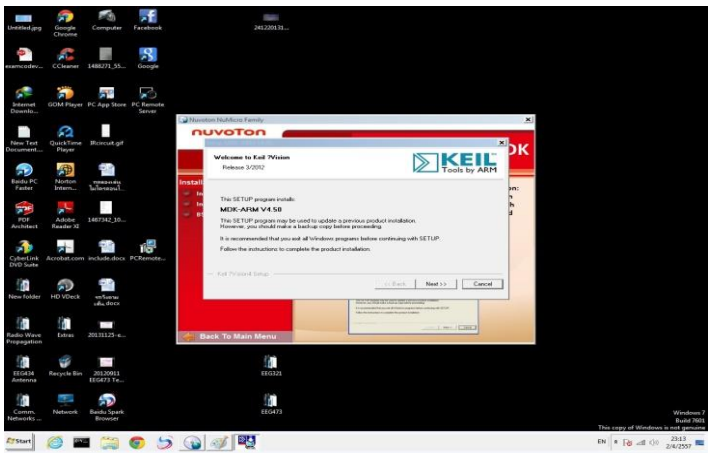
ภาพที่ 2.4 โปรแกรม Keil MDK-ARM V4.50

จะปรากฏหน้าต่างหน้านี้ขึ้นมา



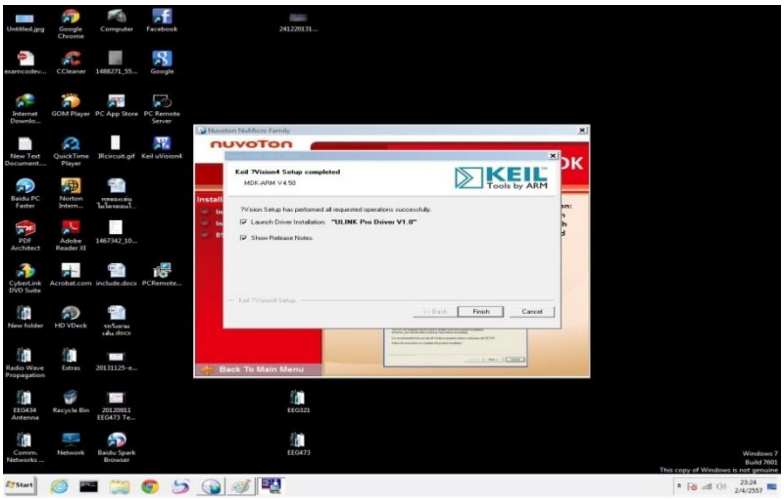
ภาพที่ 2.5 หน้าต่างหน้าต่างให้เลือกยอมรับ License

คลิกปุ่ม Next ก็จะเข้าสู่หน้าต่าง License Agreement ให้คลิกเครื่องหมายถูกที่หน้าช่องข้อความ I Agree to All Terms Of The Proceeding License Agreement จากนั้นคลิกปุ่ม Next



ภาพที่ 2.6 เริ่มต้นการติดตั้งโปรแกรม Keil

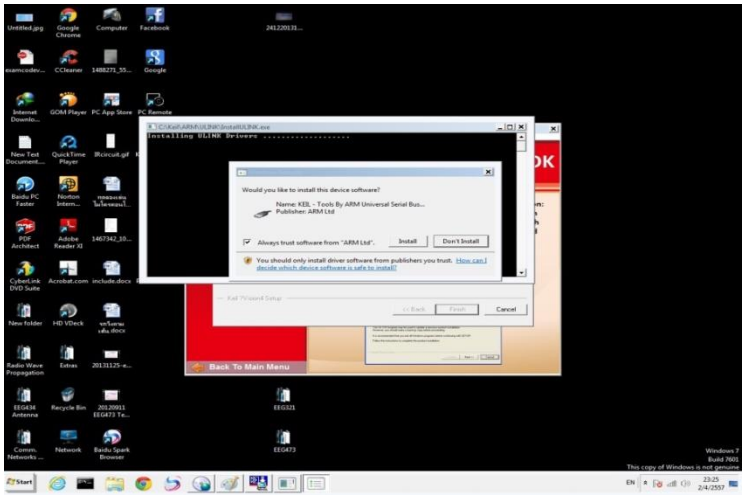
หลังจากนั้นจะมีหน้าต่าง Folder Selection เพื่อเลือกโฟลเดอร์สำหรับติดตั้งโปรแกรม ในที่นี้ให้เลือกเป็น C:\Keil ตามที่เครื่องกำหนดให้ เมื่อคลิก Next หน้าต่าง Customer Information จะปรากฏขึ้น ให้กรอกข้อมูลตามความเป็นจริง โดยเฉพาะช่อง e-mail ถ้าไม่ป้อนข้อมูล จะไม่สามารถคลิกปุ่ม Next ต่อไปเรื่อยๆจนติดตั้งโปรแกรมสำเร็จ



ภาพที่ 2.7 ติดตั้งโปรแกรมเรียบร้อยแล้ว

หลังจากติดตั้งโปรแกรมเรียบร้อยแล้ว จะแสดงหน้าต่าง Setup Complete ดังภาพที่ 2.7 ให้คลิกปุ่ม Finish เป็นอันเสร็จขั้นตอนการติดตั้ง

และหลังจากที่กดปุ่ม Finish จะมีหน้าต่างขึ้นมาถาม Install ULINK Driver ให้ติดตั้งลงไปด้วยเพื่อสะดวกในการคอมไพเลอร์กับชุดลอง NUVOTON NUC140

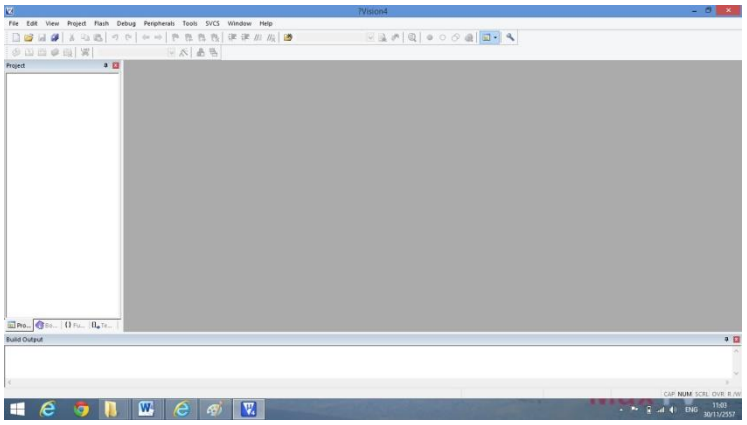


ภาพที่ 2.8 การติดตั้ง Install ULINK Driver

หลังจากการติดตั้งโปรแกรมเสร็จสมบูรณ์ ก็เริ่มต้นการใช้งานโปรแกรม Keil Uvision4 ได้เลย

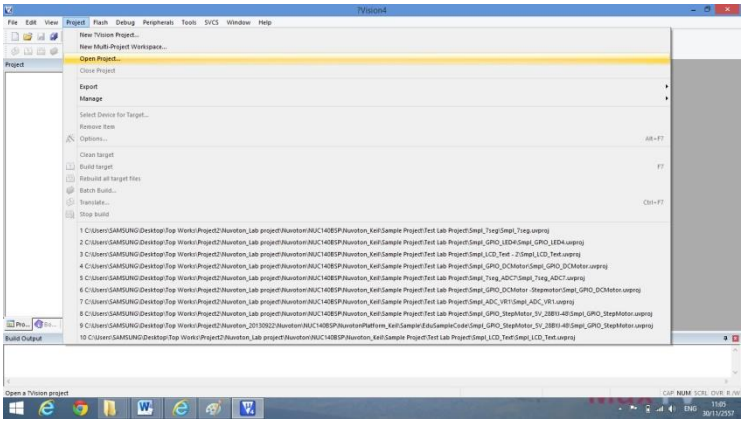
2.7.2 การใช้งานโปรแกรม Keil Uvision4

เปิดโปรแกรม Keil Uvision4 จะปรากฏหน้าต่างดังภาพที่ 2.9



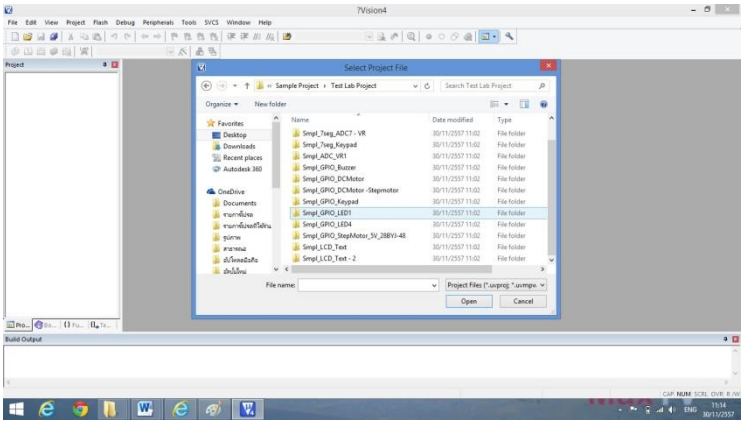
ภาพที่ 2.9 หน้าต่างโปรแกรม Keil Uvision4

ให้คลิกเลือก Project จากนั้นเลือก Open Project…



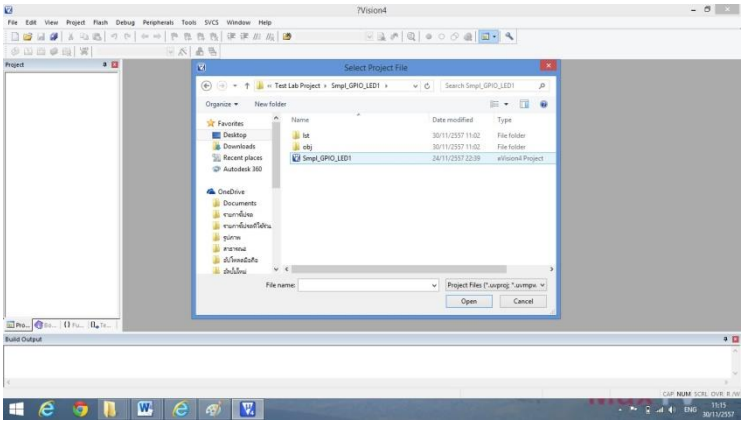
ภาพที่2.10 การเลือกใช้งาน Open Project…จากตัวอย่าง

ให้เลือกโฟลเดอร์ Nuvoton_Lab project จากนั้นเลือก Nuvoton >> NUC140BSP >> Nuvoton_Keil >> Sample Project>> Test Lab Project และเลือก Smpl_GPIO_LED1 ดังภาพ



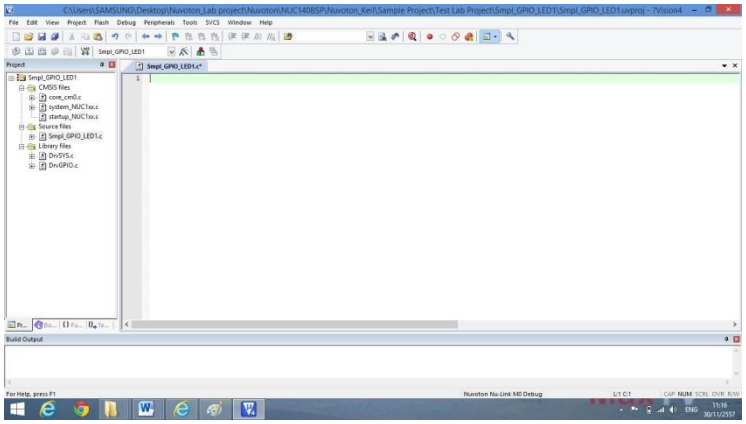
ภาพที่ 2.11 เลือกโฟลเดอร์ Smpl_GPIO_LED1

เมื่อเลือกไฟล์ Smpl_GPIO_LED1 แล้วเปิดไฟล์ Keil Uvision4 ดังภาพที่ 2.12



ภาพที่ 2.12 คลิกเปิดโปรแกรม Keil Uvision4 Smpl_GPIO_LED1

ปรากฏหน้าต่างดังภาพที่ 2.13



ภาพที่ 2.13 หน้าต่างตัวอย่างโปรแกรมโปรแกรม

บทที่ 3
ใบงานการทดลอง

ใบงานที่ 1
การทดลองการแสดงผลออกทาง LED

วัตถุประสงค์

1. เพื่อที่จะให้นักศึกษาสามารถเขียนโปรแกรมควบคุมการแสดงผลด้วย LED
2. เพื่อที่จะให้ LED แสดงผลตามที่ต้องการได้
3. สามารถนำไปประยุกต์ใช้กับ LED รูปแบบอื่นๆได้

อุปกรณ์การทดลอง

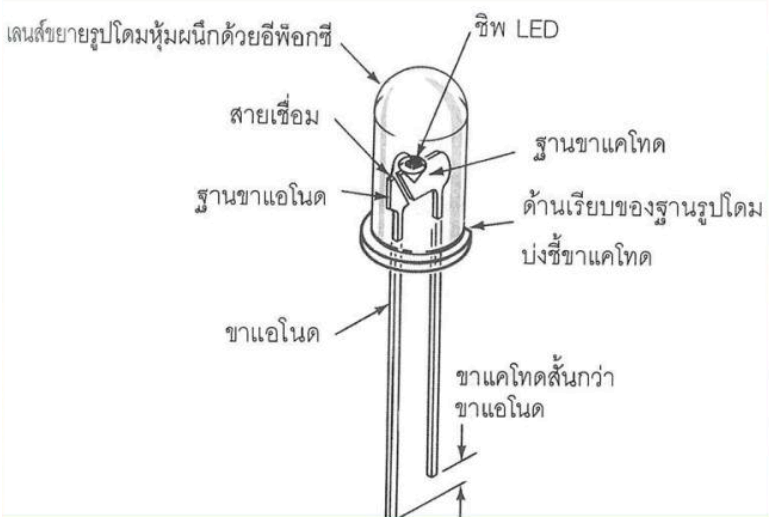
1. เครื่องคอมพิวเตอร์
2. ชุดทดลอง NOVOTON NUC140
3. สาย micro USB

การทำงานของหลอดแสดงผล LED (Light Emitting Diodes)[8]

แอลอีดี (LED) ซึ่งย่อมาจากคำว่า LIGHT EMITTING DIODE เป็นไดโอดชนิดพิเศษที่ทำมาจากสารกึ่งตัวนำหัวต่อ พี-เอ็น (P-N Junction) เมื่อมีการป้อนไบแอสเป็นแบบฟอร์เวิร์ดไบแอสจะทำให้ไดโอดส่องแสงสว่างออกมาแสงที่ส่องสว่างออกมามีหลายช่วงคลื่น แล้วแต่ชนิดของสารที่ทำ เช่น แสงสีแดง แสงสีเขียว แสงสีเหลือง หรือ แสงที่ตาเรามองไม่เห็น เช่น แสงอินฟาเรด (Infrared)

โครงสร้างของแอลอีดีนั้น ประกอบด้วยชั้นซับสเตรท (Substrate Layer) ทำด้วย GaP Of GaAs ชั้นถัดมาจะทำด้วยสาร GaAsP ทั้งสองชั้น แล้วจะทำการโด๊ป (Doped) ให้เป็นสารเอ็น ชั้นที่

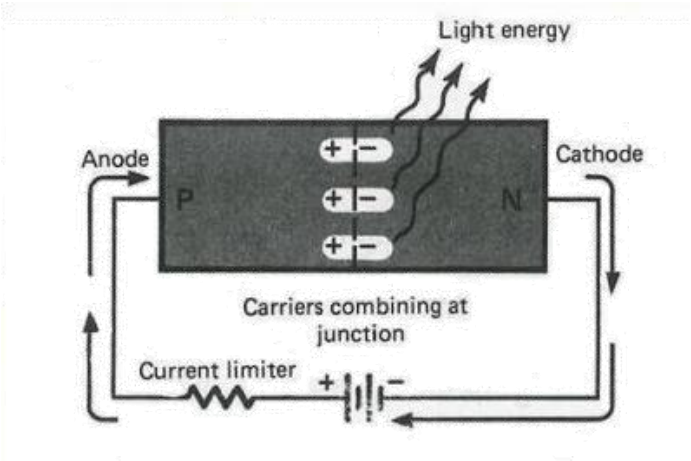
สามจะเป็นชั้นของ ซิลิคอนไพไตต์ (Si3A4) แล้วทำการแพร่สารพีลงไปให้ติดกับสารเอ็น เกิดเป็นรอยต่อพี-เอ็น (P-N Junction)



ภาพที่3.1 แสดงรูปร่าง ของไดโอดเปล่งแสง [8]

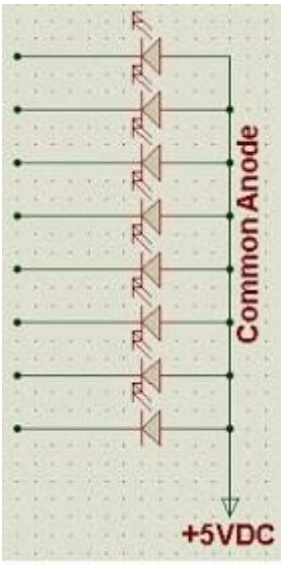
เมื่อแอลอีดีได้รับฟอร์เวิร์ดไบแอสอิเล็กตรอนในสารกึ่งตัวนำชนิดเอ็น มีพลังงานสูงขึ้นจนสามารถวิ่งข้ามรอยต่อไปรวมกับโฮลในสารพี ทำให้เกิดแสงที่เรียกว่าพลังงานโฟตอน (Photon)เปล่งแสงออกมา ปรากฏการณ์ที่เกิดขึ้นกับไดโอดเปล่งแสงนั้น เกิดจากกระบวนการผลิตที่เรียกว่า“อิเล็กโตรลูมิเนสเซนส์” (Electro Luminescence) คือ การสร้างให้หน้าสัมผัสของสารพีมีขนาดเล็กกว่าสารเอ็น เติมสารเจือบางชนิดลงไปเช่น แกลเลี่ยมอาร์เซไนด์ฟอสไฟด์ (Gallium Arsenird Phosphide : GaAsP) หรือ แกลเลี่ยม ฟอสไฟด์ (Gallium Phosphide : GaP) เพื่อให้พลังงานโฟตอนมีจำนวนมากที่สุด และเรืองแสงสว่างสุกใสขึ้น ในปัจจุบันการพัฒนาทางด้านแสดงของอุปกรณ์ อิเล็กทรอนิกส์ได้ก้าวหน้าไปอย่างรวดเร็ว จนมีอุปกรณ์อิเล็กทรอนิกส์ทางแสงขึ้นมากเช่นไฟเบอร์ออปติค ไอซีประเภทออปติคอล รวมไปถึงวงจรที่มีการเชื่อมโยงทางแสง หรือออปโต้ไอโซเลต (Opto Isolate)

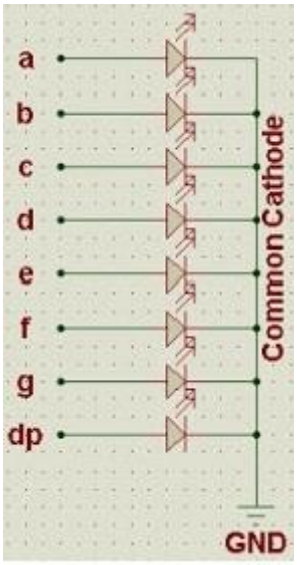
ไดโอดเปล่งแสงมีข้อดีก็คือการใช้งานจะใช้แรงดันไฟฟ้าน้อยกระแสไฟฟ้าต่ำ โดยทั่วไปจะใช้แรงดันไฟฟ้าตั้งแต่ 1.5 ถึง 3.5V ใช้กระแสอยู่ในช่วง 5 – 20 mA ซึ่งจะทำงานได้ดีที่สุด ไดโอดเปล่งแสงทั่ว ๆ ไปจะมีสีเดียว แต่ปัจจุบันถูกสร้างให้มีสองสีอยู่ในตัวเดียวกัน ซึ่งเรียกว่า ทูลีด (Two LED) มีทั้งแบบที่เป็น 3 ขา เมื่อป้อนไบแอสแต่ละครั้งก็จะมีสีแตกต่างกันออกไป



ภาพที่ 3.2 การ Forward Bias [8]

แสงจากแอลอีดีที่เป็นพลังงานโฟตอนนั้นมีอยู่ด้วยกัน 2 แบบ คือ พลังงานแสงที่สามารถมองเห็นด้วยตาเปล่า และ พลังงานแสงที่ไม่สามารถมองเห็นด้วยตาเปล่า ซึ่งแอลอีดีที่ให้พลังงานแสงที่ไม่สามารถมองเห็นด้วยตาเปล่าได้แก่ อินฟราเรดแอลอีดี (Infrared LED) ซึ่งจะมีรูปร่างและลักษณะงานเช่นเดียวกันแอลอีดีธรรมดา เพียงแต่ไม่มีแสงให้เห็นด้วยตาเปล่าในเวลาทำงาน

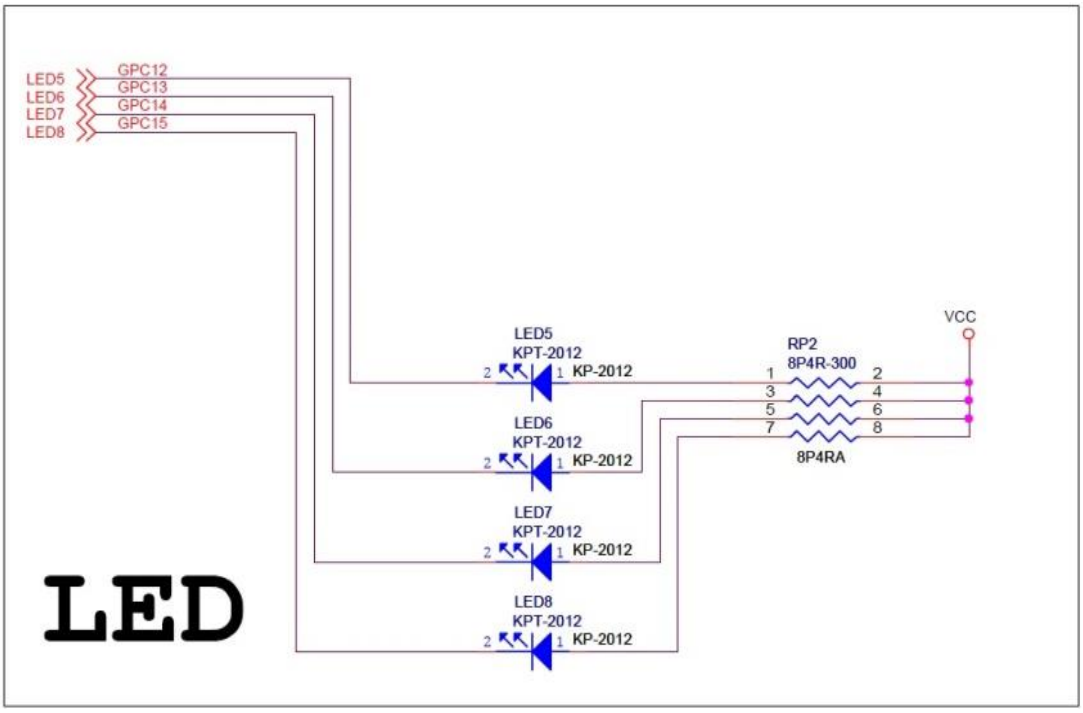




ภาพที่ 3.3 ก.การต่อวงจรแบบ Active Low ข.การต่อวงจร Active High

การเชื่อมต่อ LED ในไมโครคอนโทรลเลอร์ของ NUVOTON

คุณสมบัติของไมโครคอนโทรลเลอร์ในกรณีที่ส่งข้อมูลออกพอร์ท แอคทีปที่ลอจิก“1”จะจ่ายกระแสซอร์สประมาณ 10 mA. ดังนั้นเมื่อนำ LED มาต่อ อาจทำให้ไมโครคอนโทรลเลอร์เสียหายได้ เพราะ LED สว่างเต็มที่จะกินกระแสถึง 15 mA หรือต่อ LED ในลักษณะเป็นการซิงก์กระแส คือให้ไฟที่พอร์ต แอคทีปที่ลอจิก 0 แล้วทำให้ LED ติดดังภาพที่ 3.4



ภาพที่ 3.4 วงจร LED ในชุดทดลอง NUVOTON [1]

ขั้นตอนการทดลอง

1 เปิดโปรแกรม Keil Uvision4

2. เปิดไฟล์ตามหัวข้อ 2.7.2 การใช้งานโปรแกรม

ตัวอย่างโปรแกรมการติด - ดับของ LED 1 หลอด

```
#include "NUC1xx.h"
#include "Driver\DrvSYS.h"
#include "Driver\DrvGPIO.h"

void Init_LED();

int main (void)
{
	UNLOCKREG();
	DrvSYS_Open(48000000);
	LOCKREG();
	Init_LED();
	while (1)
	{
		DrvGPIO_ClrBit(E_GPC, 12); // output Low to turn on LED
		DrvSYS_Delay(300000);   // delay
		DrvGPIO_SetBit(E_GPC, 12); // output Hi to turn off LED
		DrvSYS_Delay(300000);   // delay
	}
}
void Init_LED()
{
	DrvGPIO_Open(E_GPC, 12, E_IO_OUTPUT);
	DrvGPIO_SetBit(E_GPC, 12);
}
```

3. ทำการ Compile โปรแกรม

4. Download Program ลงชุดทดลอง

สังเกตผลการทดลองและบันทึกผลที่เกิดขึ้น

ให้นักศึกษาอธิบายโปรแกรม

ให้นักศึกษาเขียน Flow Chart ของโปรแกรม

ให้นักศึกษาทำการแก้ไขโปรแกรม โดยให้ LED ติด-ดับ 4หลอด จากซ้ายไปขวา

ใบงานที่ 2

การแสดงผลออกทาง 7 Segments

วัตถุประสงค์

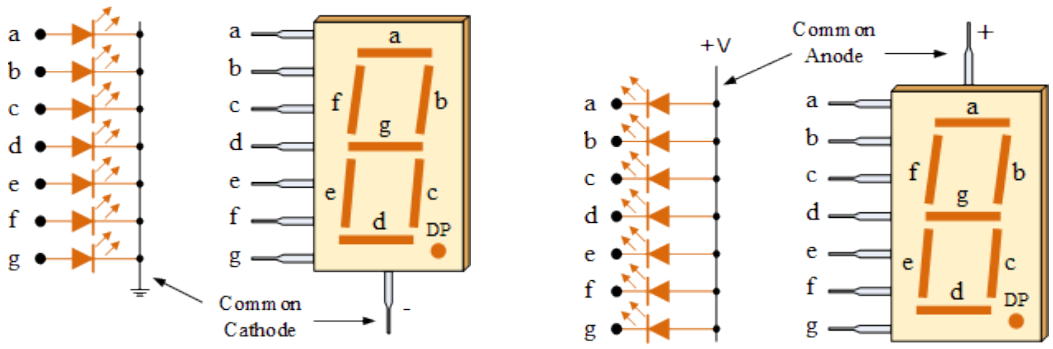
1. เพื่อให้นักศึกษาเข้าใจ โปรแกรมไมโครคอนโทรลเลอร์เพื่อควบคุม 7 Segments
2. สามารถนำไปประยุกต์ใช้กับ 7 Segments ในรูปแบบอื่นๆได้

อุปกรณ์การทดลอง

1. เครื่องคอมพิวเตอร์
2. ชุดทดลอง NOVOTON NUC140
3. สาย micro USB

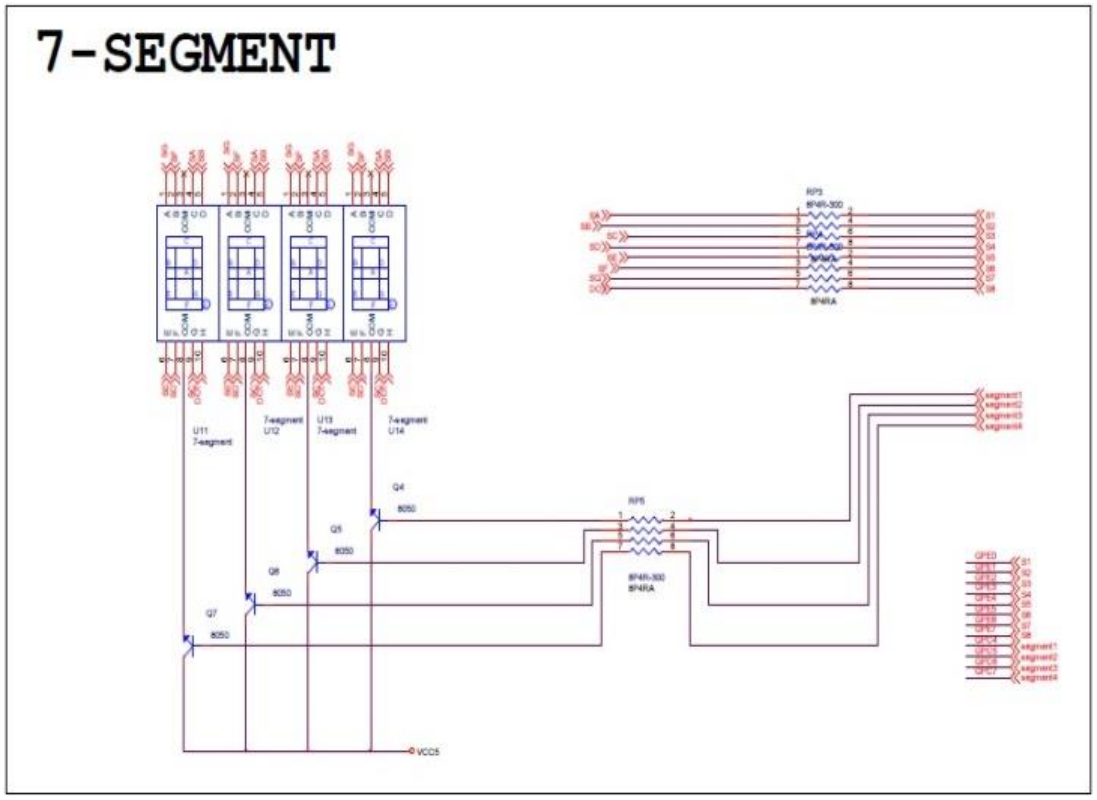
7 Segments [8]

7 Segments หรือ LED แสดงผลตัวเลขชนิด 7 ส่วน นั้นถือเป็นส่วนแสดงผลชนิดหนึ่งที่ถูกนำ มาใช้งานอย่างแพร่หลาย เนื่องจากคุณสมบัติเด่นทางด้านความเด่นชัดในการแสดงผล มีโครงสร้างที่ไม่ซับซ้อนใช้งานได้ง่าย และสามารถประยุกต์ใช้ได้หลากหลายรูปแบบตามความต้องการ สาเหตุที่หน่วยแสดงผล ชนิดนี้ถูกเรียกว่า LED 7 ส่วนนั้น เนื่องมาจากโครงสร้างที่มีลักษณะเป็นการต่อ LED 7 ตัวร่วมกันใน ตำแหน่งต่างๆ 7 Segment ประกอบด้วย LED 7 ดวง รวมจุดอีก 1 ดวง รวมเป็น 8 ดวง 7 Segment มี 2 แบบ คือ Common Cathode และ Common Anode ในที่นี้จะกล่าวเฉพาะ Common Cathode



ภาพที่ 3.5 การต่อ 7 Segment แบบ Common Cathode และ Common Anode

การเชื่อมต่อ 7 Segment ในไมโครคอนโทรลเลอร์ของ NUVOTON



ภาพที่ 3.6 วงจร 7 Segment ในชุดทดลอง NUVOTON [1]

จากวงจร 7 Segment ในชุดทดลอง NUVOTON ด้านบน แสดงการต่อ 7 Segment ที่อยู่ภายในชุดทดลองของ NUVOTON โดยจะต่อแบบ Common Cathode จะทำการแสดงผลโดยใช้พอร์ต GPIO E0-7และใช้พอร์ต GPIO C4-7เป็นตัวเลือกในการแสดงหลักของ 7 Segment

ขั้นตอนการทดลอง

1. เปิดโปรแกรม Keil Uvision4
2. เปิดไฟล์ตามหัวข้อ 2.7.2 การใช้งานโปรแกรม

ตัวอย่างโปรแกรม LED แสดงผลตัวเลขชนิด 7 ส่วน

```
#include "NUC1xx.h"
#include "Driver\DrvSYS.h"
#include "Seven_Segment.h"

void seg_display(int16_t value);

int32_t main (void)
{
	char num[16];
	int number=0,count=0;
	UNLOCKREG();
	DrvSYS_Open(48000000);
	LOCKREG();
	close_seven_segment();
	while(1)
	{
		if(number>9) number=0;
```

```
                while(count<10)
                {
                        seg_display(number);
                        count++;
                }

                count=0;
                number++;
        }
}
void seg_display(int16_t value)
{
int8_t digit;
        digit = value / 1000;
        close_seven_segment();
        show_seven_segment(3,digit);
        DrvSYS_Delay(5000);
        close_seven_segment();

        value = value - digit * 1000;
        digit = value / 100;
        close_seven_segment();
        show_seven_segment(2,digit);
        DrvSYS_Delay(5000);

        value = value - digit * 100;
        digit = value / 10;
        close_seven_segment();
```

```
        show_seven_segment(1,digit);
        DrvSYS_Delay(5000);

        value = value - digit * 10;
        digit = value;
        close_seven_segment();
        show_seven_segment(0,digit);
        DrvSYS_Delay(5000);
}
```

3. ทำการ Compile โปรแกรม

4. Download Program ลงชุดทดลอง

สังเกตผลการทดลองและบันทึกผลที่เกิดขึ้น

ให้นักศึกษาอธิบายโปรแกรม

ให้นักศึกษาเขียน Flow Chart ของโปรแกรม

ให้นักศึกษาทำการแก้ไขโปรแกรม โดยให้ 7 Segments นับขึ้นจนถึงรหัสนักศึกษา 2 หลักสุดท้าย

ใบงานที่ 3

การใช้งาน LCD (Graphic LCD)

วัตถุประสงค์

1. เพื่อที่จะให้ศึกษาการเขียนโปรแกรมการแสดงผลบนจอ LCD
2. เพื่อที่จะสามารถนำโปรแกรมที่มีอยู่มาแก้ไขให้แสดงผลตามที่ต้องการ

อุปกรณ์การทดลอง

1. เครื่องคอมพิวเตอร์
2. ชุดทดลอง NOVOTON NUC140
3. สาย micro USB

LCD (Liquid Crystal Display) [9]

หรือที่เราเรียกกันโดยทั่วไปว่า จอ LCD นั้น เป็นจอแสดงผลชนิดหนึ่งที่ได้รับความนิยมอย่างมาก สามารถพบเห็นได้โดยทั่วไป เช่น เครื่องคิดเลข นาฬิกา จอโทรศัพท์สาธารณะ โทรศัพท์มือถือ เครื่องเล่นเพลง MP3 หรือแม้แต่บนเครื่องมือวัดต่างๆ โดยทั่วไปจอ LCD จะถูกแบ่งออกเป็น 3 ประเภทหลักๆ คือ ประเภทที่ใช้ในการแสดงตัวอักษร (Character Type LCD) และประเภทที่ใช้ในการแสดงภาพ (Graphic Type LCD) และจอ LCD ที่แสดงผลในรูปแบบอื่นๆ เฉพาะทางโดยมักมีลักษณะการแสดงผลเป็นส่วนๆ (Segment Display Type LCD) เช่นจอ LCD แสดงตัวเลขแบบ 7 ส่วนของเครื่องมือวัด หรือจอ LCD ของเครื่องเล่นเกมในอดีต จอ LCD เป็นอุปกรณ์แสดงผลแบบผลึกเหลว หรือ “จอแสดงผลแบบ LCD” (LCD: Liquid Crystal Display) นั้นเป็นจอแสดงผลแบบ Dot-Matrix ขนาด 4x16 ตัวอักษร 4 บรรทัด บรรทัดละ 16 ตัวอักษร ความละเอียดในตัวอักษรเท่ากับ 128x64 Pixels จอแสดงผลแบบ LCD ประกอบด้วย 2 ส่วนหลัก คือ ส่วนแสดงผล (หน้าจอ) และส่วนควบคุม ในการเขียนโปรแกรมแสดงผลทาง LCD จะเป็นการเข้าถึง IC LCD Controller โดยตัวไมโครคอนโทรลเลอร์จะเป็นตัวควบคุมจังหวะการทำงานทั้งหมด โครงสร้างทั่วไปของ LCD Module

<table>
<tr><th>Pin No</th><th>Symbol</th><th>I/O</th><th>Function</th></tr>
<tr><td>1~4</td><td>NC</td><td>-</td><td>No Connection.</td></tr>
<tr><td>5</td><td>V_{LCD}</td><td>PWR</td><td>Main LCD power supply.</td></tr>
<tr><td>6</td><td>V_{B0+}</td><td>PWR</td><td rowspan="4">LCD Bias Voltages. These are the voltage sources to provide SEG driving currents. These voltages are generated internally. Connect capacitors of C_{BX} value between V_{BX+} and V_{BX-}.</td></tr>
<tr><td>7</td><td>V_{B0-}</td><td>PWR</td></tr>
<tr><td>8</td><td>V_{B1-}</td><td>PWR</td></tr>
<tr><td>9</td><td>V_{B1+}</td><td>PWR</td></tr>
<tr><td>10</td><td>VSS</td><td>PWR</td><td>Power Ground.</td></tr>
<tr><td>11</td><td>VDD</td><td>PWR</td><td>Power supply terminal VCC.</td></tr>
<tr><td>12</td><td>BM1</td><td>I</td><td>Bus mode: “HL”: 8080 “HH”: 6800</td></tr>
<tr><td>13</td><td>BM0</td><td>I</td><td>BM[1:0] “LH”: S9 “LL”: S8</td></tr>
<tr><td>14</td><td>DB7</td><td>I/O</td><td rowspan="8">Bi-directional bus for both serial and parallel host interfaces. In serial modes, connect DB0 to SCK, DB3 to SDA.
<table><tr><th></th><th>BM=1x (Parallel)</th><th>BM=0x (Serial)</th></tr><tr><td>D0</td><td>D0</td><td>SCK</td></tr><tr><td>D1</td><td>D1</td><td></td></tr><tr><td>D2</td><td>D2</td><td></td></tr><tr><td>D3</td><td>D3</td><td>SDA</td></tr><tr><td>D4</td><td>D4</td><td></td></tr><tr><td>D5</td><td>D5</td><td></td></tr><tr><td>D6</td><td>D6</td><td>-</td></tr><tr><td>D7</td><td>D7</td><td>-</td></tr></table></td></tr>
<tr><td>15</td><td>DB6</td><td>I/O</td></tr>
<tr><td>16</td><td>DB5</td><td>I/O</td></tr>
<tr><td>17</td><td>DB4</td><td>I/O</td></tr>
<tr><td>18</td><td>DB3/SDA</td><td>I/O</td></tr>
<tr><td>19</td><td>DB2</td><td>I/O</td></tr>
<tr><td>20</td><td>DB1</td><td>I/O</td></tr>
<tr><td>21</td><td>DB0/SCK</td><td>I/O</td></tr>
<tr><td>22</td><td>WR1</td><td>I</td><td rowspan="2">WR [1:0] controls the read/write operation of the host interface. See Host Interface section for details. The meaning of WR [1:0] depends on whether the interface is in the 6800 mode, or the 8080 mode. In serial modes, these two pins are not used and can be connected to Vss.</td></tr>
<tr><td>23</td><td>WR0</td><td>I</td></tr>
<tr><td>24</td><td>CD</td><td>I</td><td>Select the incoming command if it is a control instruction or for display data. CD pin is not used in S9 mode, connect it to Vdd or Vss.
” L”: control instruction ”H”: display data</td></tr>
<tr><td>25</td><td>RST</td><td>I</td><td>When RST=”L”, all control registers are re-initialized by their default states.</td></tr>
<tr><td>26</td><td>/CS0</td><td>I</td><td>Chip Select or Chip Address. In parallel mode and S8 mode, chip is selected when /CS0=”L”. When the chip is not selected, DB[7:0] will be high impedance.</td></tr>
<tr><td>27~30</td><td>NC</td><td>-</td><td>No Connection.</td></tr>
</table>

ตารางที่ 3.1 การทำงานของแต่ละขาที่อยู่ภายในจอ LDC [1]

รีจิสเตอร์และหน่วยความจำของ LCD Controller

- รีจิสเตอร์ IR (Instructor Register) : ทำหน้าที่เก็บข้อมูลที่เป็นรหัสคำสั่ง เช่น การเคลียร์การแสดงผลการควบคุมการเคลื่อนที่ของ Cursor นอกจากนี้แล้วยังทำหน้าที่เป็นตัวเก็บรายละเอียดต่างๆของการแสดงผล ณ ขณะนั้นด้วย

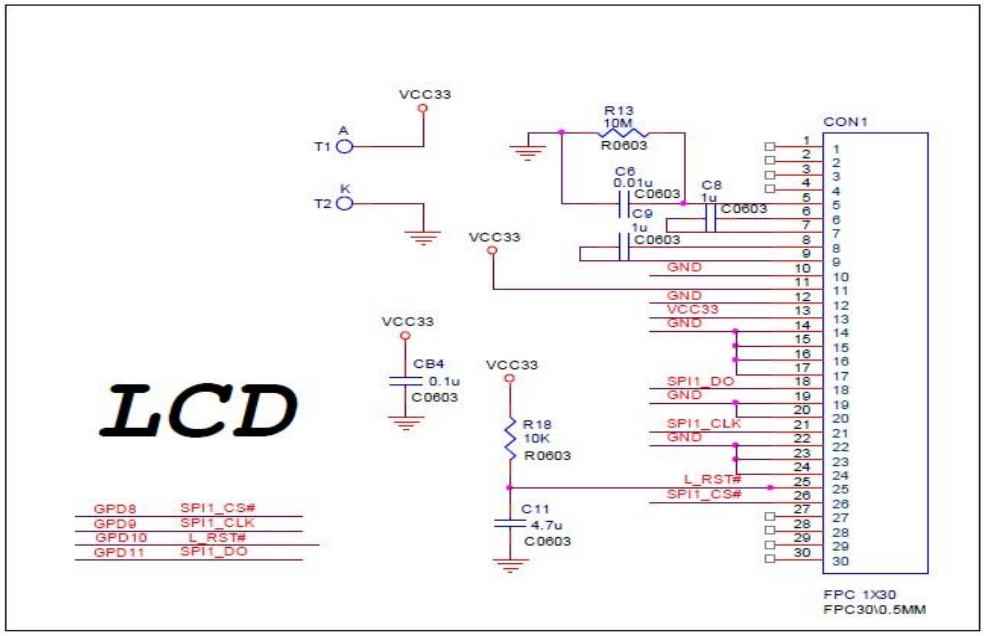
- รีจิสเตอร์ DR (Data Register) : ทำหน้าที่เก็บข้อมูลชั่วคราวสำหรับการทำงานภายในลักษณะต่างๆ

Bus Type		8080	6800	SPI (S8)	SPI (S9)
Control & Data Pins	BM[1:0]	10b	11b	00b	01b
	CS[1:0]	Chip Select			
	CD	Control/Data			–
	WR0	$\overline{WR}$	R/$\overline{W}$	–	
	WR1	$\overline{RD}$	EN	–	
	Access	Read/Write		Write Only	
	D[7:0]	8-bit bus (Tri-state)		D0=SCK, D3=SDA	

* Connect unused control pins and data bus pins to V_{DD} or V_{SS}

ตารางที่3.2 บิตตั้งค่าจอ LCD [1]

การเชื่อมต่อ LCD ในไมโครคอนโทรลเลอร์ของ NUVOTON



ภาพที่ 3.7 การต่อใช้งาน LCD [1]

จากวงจรการเชื่อมต่อจอแสดงผล LCD ในไมโครคอนโทรลเลอร์ของ NUVOTON จะถูกฟิกพอร์ตการใช้งานเนื่องจากจะถูกติดตั้งไว้บนชุดทดลอง ผ่านทางพอร์ต GPIO D 9-11 โดยที่พอร์ต GPD 9-11 เป็นพอร์ตการต่อแบบอินเตอร์เฟสภายในวงจรชุดทดลอง

ขั้นตอนการทดลอง

1. เปิดโปรแกรม Keil Uvision4
2. เปิดไฟล์ตามหัวข้อ 2.7.2 การใช้งานโปรแกรม

ตัวอย่างโปรแกรม LCD (Graphic LCD)

```
#include "NUC1xx.h"
#include "Driver\DrvSYS.h"
#include "Driver\DrvGPIO.h"
#include "NUC1xx-LB_002\LCD_Driver.h"

int main(void)
{
	UNLOCKREG();
	DrvSYS_Open(48000000); // set to 48MHz
	LOCKREG();
	Initial_panel();
	clr_all_panel();

	print_lcd(0, "Hello Welcom to ");
	print_lcd(1, "Microtroller lab");
}
```

3. ทำการ Compile โปรแกรม

4. Download Program ลงชุดทดลอง

สังเกตผลการทดลองและบันทึกผลที่เกิดขึ้น

ให้นักศึกษาอธิบายโปรแกรม

ให้นักศึกษาเขียน Flow Chart ของโปรแกรม

ให้นักศึกษาทำการแก้ไขโปรแกรม ให้แสดงชื่อ รหัสนักศึกษา สาขาที่เรียน

ใบงานที่ 4
การใช้งาน Key Pad Switch

วัตถุประสงค์

1. เพื่อที่ศึกษาการเขียนโปรแกรมการใช้งาน Key Pad Switch
2. เพื่อที่จะสามารถนำโปรแกรมที่มีอยู่มาแก้ไขรับคำสั่งจาก Key Pad Switch แสดงผลตามที่ต้องการได้

อุปกรณ์การทดลอง

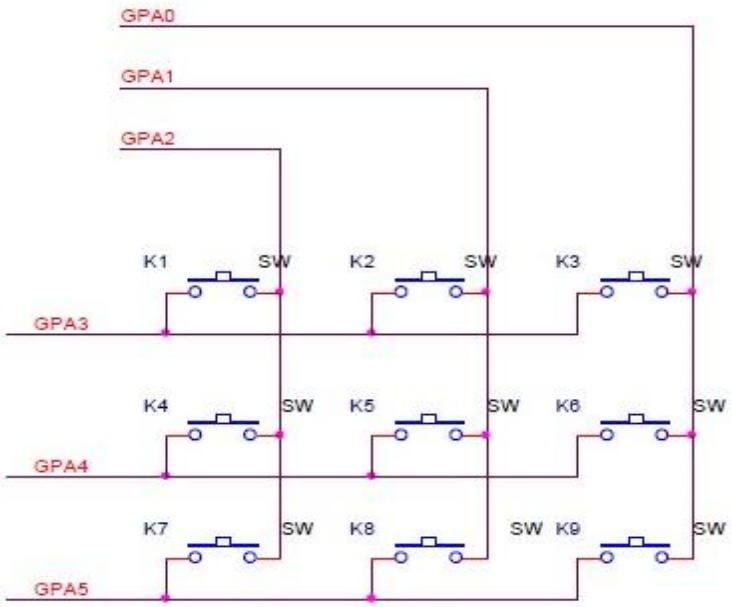
1. เครื่องคอมพิวเตอร์
2. ชุดทดลอง NOVOTON NUC140
3. สาย micro USB

การต่อวงจรสวิตซ์แบบเมตริก [10]

โดยวิธีการนี้เหมาะกับระบบที่มีความจำเป็นต้องใช้งานสวิตซ์มากๆ เช่น วงจรแป้นพิมพ์ คีย์บอร์ด ที่ใช้สำหรับป้อนค่าตัวเลข ตัวอักษร และข้อความต่างๆ ซึ่ง จะเกิดความไม่สะดวกสำหรับผู้ใช้งานเป็นอย่างมาก ถ้าออกแบบให้มีจำนวนสวิตซ์คีย์บอร์ดน้อยๆ เพราะจะยากแก่การค้นหาตำแหน่งของตัวเลข ตัวอักษรที่อยู่ซ้อนกันอยู่ในคีย์เดียวหลายๆ ชั้น โดยวิธีการ นี้จะต้องใช้พอร์ต 2 ส่วน คือ พอร์ตสำหรับอ่านค่าสถานะของ คีย์สวิตซ์ จากทางแถว (Row) และ พอร์ตสำหรับทำหน้าที่ส่งค่าออกไป สแกนคีย์ ในแต่ละหลัก (Column) ของวงจรโดยจำนวนของสวิตซ์จะขึ้นอยู่กับขนาดของแถวและหลักที่ใช้ เช่น ถ้าเป็นขนาด 4x4 ก็จะได้ทั้งหมด 16 ตำแหน่ง เป็นต้นซึ่งวิธีการสแกนคีย์แบบเมตริกนี้จะทำทีละหลัก (Column) โดยเริ่มจากหลักแรกไปหาหลัก สุดท้ายตามลำดับ สำหรับลักษณะของการต่อวงจรโดยทั่วไปของวิธีการนี้จะนิยมคงสถานะของ สัญญาณด้านที่เป็น Input ให้มีค่าเป็น “1” รอไว้ก่อนเสมอโดยการต่อตัวต้านทาน Pull-Up เข้ากับ Port- Input รอไว้ก่อน โดยในการสแกนคีย์จะทำทางด้านหลัก (Column) โดยส่งค่าออกไปทางด้าน Port- Input ให้มีค่าเป็น

“0” ครั้งละ 1 บิต แล้วก็อ่านค่าจาก Port-Input เข้ามาตรวจสอบว่าทุกบิตยังคงเป็น “1” อยู่หรือไม่ ซึ่งถ้าพบว่ามีบิตใดเป็น “0” (Column Active = “0”) ก็สามารถทราบได้ทันทีว่ามีการกดคีย์ ขึ้นที่ตำแหน่ง Row และ Column นั้นๆ แต่ถ้าทุกบิตยังคงมีค่าเป็น “1” ก็ให้เปลี่ยนการ Scan ไปยัง Column ถัดไปอีกโดยทำเหมือนกันกับ Column แรกจนครบทุก Column

การเชื่อมต่อ Key Pad Switch ในไมโครคอนโทรลเลอร์ของ NUVOTON



ภาพที่ 3.8 ขาใช้งานของ Key Pad Switch [1]

จากบล็อกไดอะแกรมด้านบนเป็น Key Pad Switch แบบ 3x3 ที่อยู่ในชุดทดลอง NUVOTON NUC140 โดยจะมีพอร์ต GPIO A 3-5 เป็น ROW เพื่อสแกนในการจ่ายลอจิกอินพุต และพอร์ต GPIO A 0-3 เป็น Column เพื่อที่จะจ่ายเอาต์พุตไปยังไมโครคอนโทรลเลอร์ ทำการประมวลผลและแสดงค่าออกไปยังพอร์ตที่ต้องการ

ขั้นตอนการทดลอง

1. เปิดโปรแกรม Keil Uvision4
2. เปิดไฟล์ตามหัวข้อ 2.7.2 การใช้งานโปรแกรม

ตัวอย่างโปรแกรม Key Pad Switch

```
#include "NUC1xx.h"
#include "DrvSYS.h"
#include "Seven_Segment.h"
#include "scankey.h"

void seg_display(int16_t value);

int32_t main (void)
{
        int8_t number,get_number,click=0,set=0;
        UNLOCKREG();
        DrvSYS_Open(48000000);
        LOCKREG();
        OpenKeyPad();
        while(1)
        {
                get_number = Scankey();
                if(get_number == 0) set=1;
                else if(get_number != 0&&set==1);
                seg_display(number);
                close_seven_segment();
```

```
        }
}
void seg_display(int16_t value)
{
        int8_t digit;
                digit = value / 1000;
                close_seven_segment();
                show_seven_segment(3,digit);
                DrvSYS_Delay(5000);
                close_seven_segment();
                value = value - digit * 1000;
                digit = value / 100;
                close_seven_segment();
                show_seven_segment(2,digit);
                DrvSYS_Delay(5000);
                value = value - digit * 100;
                digit = value / 10;
                close_seven_segment();
                show_seven_segment(1,digit);
                DrvSYS_Delay(5000);
                value = value - digit * 10;
                digit = value;
                close_seven_segment();
                show_seven_segment(0,digit);
                DrvSYS_Delay(5000);
}
```

3. ทำการ Compile โปรแกรม

4. Download Program ลงชุดทดลอง

สังเกตผลการทดลองและบันทึกผลที่เกิดขึ้น

ให้นักศึกษาอธิบายโปรแกรม

ให้นักศึกษาเขียน Flow Chart ของโปรแกรม

ให้นักศึกษาทำการแก้ไขโปรแกรม โดยให้กด Key Pad Switch แสดงตัวเลขบน 7-Segments 2-3 หลัก

ใบงานที่ 5

การแสดงผลออกทาง Buzzer

วัตถุประสงค์

1. เพื่อที่จะศึกษาการเขียนโปรแกรมการแสดงผลออกทาง Buzzer
2. เพื่อที่จะสามารถแก้ไขโปรแกรมการหน่วงเวลาส่งเสียงออกทาง Buzzer

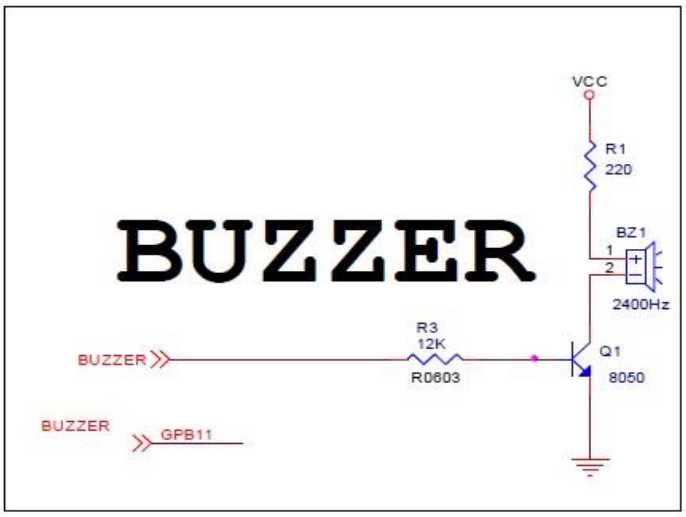
อุปกรณ์การทดลอง

1. เครื่องคอมพิวเตอร์
2. ชุดทดลอง NOVOTON NUC140
3. สาย micro USB

BUZZER

เป็นอุปกรณ์ไฟฟ้าที่นำผลของแม่เหล็กไฟฟ้ามาดึงดูดให้แกนอามาเจอร์(Armature) เคลื่อนที่มาเคาะกับกระดิ่ง (Bell) ทำให้เกิดเสียงดังได้ โครงสร้างภายในประกอบด้วยแท่งเหล็กรูปตัวยู (U-Shaped) พันขดลวดรอบๆ แท่งเหล็กนี้ต่ออนุกรมกับหน้าสัมผัสซึ่งเปิดปิดได้โดยการเคลื่อนที่ของก้านอามาเจอร์การใช้งานต้องต่อกระดิ่งไฟฟ้าอนุกรมกับสวิทช์กดปุ่ม (Push Button) และแหล่งจ่ายไฟฟ้า เช่น แบตเตอรี่ เมื่อกดสวิตช์กระแสไฟฟ้าจะผ่านหน้าสัมผัสและขดลวด ทำให้เกิดการดึงดูดอามาเจอร์ให้เคลื่อนที่มาเคาะกระดิ่งทำให้เกิดเสียงดัง ในขณะที่อามาเจอร์เคลื่อนที่ก็จะตัดวงจรไฟฟ้าออกไปด้วย ดังนั้นเมื่อก้านอามาเจอร์เคาะกระดิ่ง แล้วก็จะดีดไปตำแหน่งเดิมทันทีและต่อวงจรไฟฟ้าอีกครั้ง เมื่อใดที่ปล่อยมือจากสวิตช์กระบวนการที่เกิดขึ้นก็จะหยุดลง

การเชื่อมต่อ Buzzer ในไมโครคอนโทรลเลอร์ของ NUVOTON



ภาพที่ 3.9 ขาการต่อใช้ของ Buzzer [1]

จากบล็อกไดอะแกรมด้านบน แสดงการต่อใช้งานของ Buzzer ในชุดทดลอง NUVOTON NUC140 โดยต่อการใช้งานจากพอร์ต GPIO B 11 เป็นขารับคำสั่งจากไมโครคอนโทรลเลอร์เป็นลอจิก 0 เพื่อให้สามารถใช้งานได้

ขั้นตอนการทดลอง

1. เปิดโปรแกรม Keil Uvision4
2. เปิดไฟล์ตามหัวข้อ 2.7.2 การใช้งานโปรแกรม

ตัวอย่างโปรแกรม การแสดงผลออกทาง Buzzer

```
#include "NUC1xx.h"
#include "Driver\DrvSYS.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvADC.h"
```

```
int main (void)
{
        UNLOCKREG();
        DrvSYS_Open(48000000);
        LOCKREG();
        DrvGPIO_Open(E_GPB, 11, E_IO_OUTPUT);

        while(1)
        {
        DrvGPIO_ClrBit(E_GPB,11);
        DrvSYS_Delay(100000);
        DrvGPIO_SetBit(E_GPB,11);
        DrvSYS_Delay(100000);
        }
}
```

3. ทำการ Compile โปรแกรม

4. Download Program ลงชุดทดลอง

สังเกตผลการทดลองและบันทึกผลที่เกิดขึ้น

ให้นักศึกษาอธิบายโปรแกรม

ให้นักศึกษาเขียน Flow Chart ของโปรแกรม

ให้นักศึกษาทำการแก้ไขโปรแกรม โดยทำการหน่วงเวลาเพื่อให้ BUZZER ส่งเสียงเตือนช้าลง

ใบงานที่ 6
การใช้งาน Variable Resistance

วัตถุประสงค์

1. เพื่อที่จะศึกษาการเขียนโปรแกรมการแสดงค่าจาก Variable Resistance
2. เพื่อที่จะสามารถแก้ไขโปรแกรมใช้งานการรับค่าจาก Variable Resistance ได้

อุปกรณ์การทดลอง

1. เครื่องคอมพิวเตอร์
2. ชุดทดลอง NOVOTON NUC140
3. สาย micro USB

ADC (Analog-To-Digital Converters) [11]

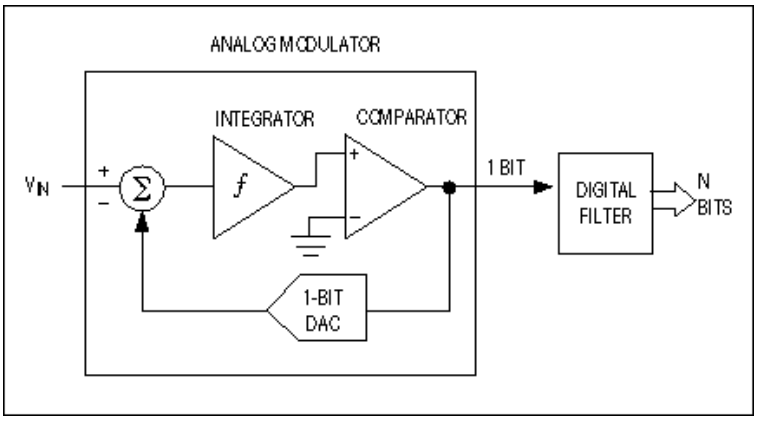
ADC แต่ละชนิดจะมีจุดเด่นจุดด้อยแตกต่างกันไปผู้ออกแบบจำเป็นต้องเลือกใช้ ADC ให้เหมาะสมกับลักษณะงาน

แปลอย่างตรงตัวคือ ตัวแปลงสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอล การแปลงสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอลมีด้วยกันหลายวิธี ดังต่อไปนี้

1. Delta-Sigma

Delta-Sigma เป็นการแปลงสัญญาณที่มีความละเอียดสูง เป็นการแปลงที่ถือว่าเป็นอุดมคติ (Ideal) และทำงานได้หลายย่านความถี่ ตั้งแต่สัญญาณ DC ไปจนถึงหลัก MHz การทำงานของ Delta-Sigma ADC สัญญาณอินพุตจะถูก Oversample โดยตัว Modulator หลังจากนั้นจะนำสัญญาณมากรองอีกทีเพื่อให้ได้ค่าที่ถูกต้องมากขึ้นโดย Digital Filter จึงทำให้ได้ค่า ADC ที่มีความละเอียดสูง ที่เรทแซมปลิ้งต่ำเพราะว่า Delta-Sigma ทำการ Oversample สัญญาณอินพุตจึงทำให้ได้สัญญาณเรียบขึ้น(Anti-Aliasing) และในวงจรส่วน Digital Filter จะมีต้นทุนที่ต่ำกว่า Analog Filter โดยปกติแล้วความละเอียดสูงๆจากการ

แปลงแบบ Delta-Sigma จะใช้ในงานด้านเสียง (Audio), งานควบคุมในอุตสาหกรรม และงานเครื่องมือวัดโดยปกติแล้ว Delta-sigma จะรับสัญญาณความแตกต่างระหว่าง 2 อินพุต แทนที่จะเป็นการวัดโวลต์เทียบกราวด์ การวัดสัญญาณความแตกต่าง (Differential) ของอินพุตสามารถนำไปวัดเซ็นเซอร์แบบบริดจ์ เช่นเทอร์โมคัปเปิ้ลได้ Delta-Sigma เป็นการแปลงสัญญาณที่แตกต่างกับ SAR การแปลงสัญญาณแบบ SAR จะเหมือนกับการวัดสัญญาณ ณ ตอนนั้น ส่วน Delta-Sigma จะเหมือนกับค่าเฉลี่ยของสัญญาณไฟฟ้าใน 1 ช่วงเวลาโดยส่วนมาก Delta-Sigma จะมีบัฟเฟอร์และตัวขยาย (Programmable Gain Amplifiers = PGA) อยู่ในตัว บัฟเฟอร์จะมีอิมพิแดนส์สูงเพื่อให้ต่อตรงกับขาสัญญาณได้โดยไม่มีทำให้วงจรมีค่าอิมพิแดนส์มีค่าผิดไปจากเดิม ดังนั้น Delta-Sigma จึงสามารถใช้วัดสัญญาณที่มีขนาดเล็กได้ดี เช่นสัญญาณจากเทอร์โมคัปเปิ้ล เพราะมี PGA อยู่ในตัวสามารถปรับค่า Gain ได้ตามความเหมาะสม

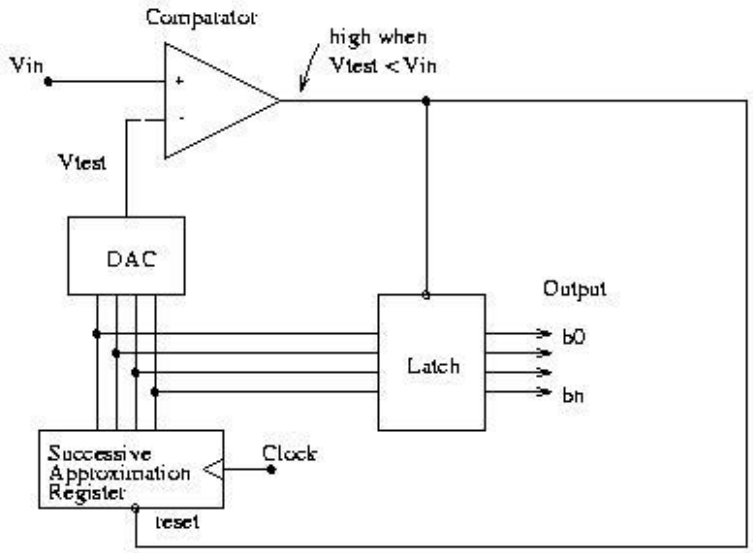


ภาพที่ 3.10 โครงสร้างภายในของ Delta-Sigma ADC [1]

2. SAR ADCs

Successive-Approximation-Register (SAR) Analog-To-Digital Converters (ADCs) เป็น ADC ที่มีขายในตลาดมากที่สุด ความละเอียดจะอยู่ในระดับกลางถึงถึงความละเอียดสูง SAR ADCs ให้อัตราการ Sampling ถึง 5 Msps ที่ความละเอียด 8-16 บิต โครงสร้างแบบ SAR จะให้ประสิทธิภาพสูง กินไฟน้อย และมีขนาดเล็กหลักการของ SAR จะเหมือนตาชั่งแบบ Balance คือจะมีน้ำหนักที่ไม่ทราบค่าอยู่ด้านหนึ่ง และน้ำหนักที่ทราบ

ค่าอยู่อีกด้านหนึ่ง เราจะเปลี่ยนน้ำหนักที่ทราบค่าไปเรื่อยๆจนกระทั่งตาชั่ง Balance เมื่อมีสัญญาณอินพุตเข้ามาสัญญาณจะถูก Sample เข้ามาและถูก Hold ไว้เพื่อเปรียบเทียบกับแรงดันที่ทราบค่า และจะส่งผลลัพธ์ไปที่เอาต์พุต

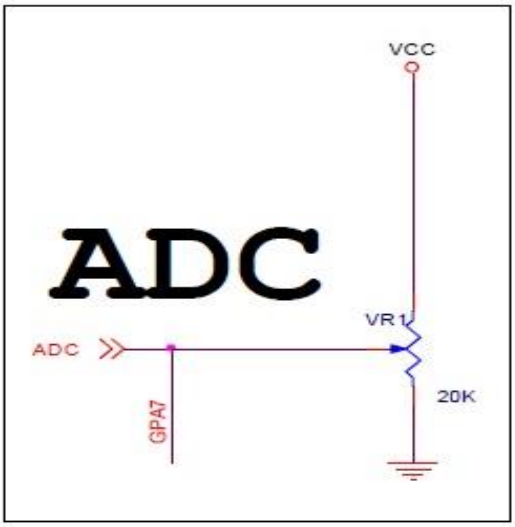


ภาพที่ 3.11 โครงสร้างภายในของ SAR ADC [1]

3. Flash ADC

Flash ADCs เป็นที่รู้กันในอีกชื่อว่า Parallel ADCs หรือแปลเป็นไทยได้ว่า การแปลงสัญญาณอนาลอกเป็นดิจิตอลแบบขนาน ADC ชนิดนี้เป็น ADC ที่เร็วที่สุด Flash ADCs จะใช้ในงานที่มีแบนวิธกว้าง อย่างไรก็ตามโดยทั่วไปแล้วจะกินกระแสมากกว่า ADC ชนิดอื่น มีราคาแพง และโดยทั่วไปจะจำกัดความละเอียดไว้ที่ความละเอียด 8 บิต Flash ADCs สร้างมาจากวงจร Comparator มาต่อเรียงกัน โดย Comparator แต่ละตัวจะมีค่าเท่ากับ 1 บิต ซึ่งเป็นผลลัพธ์มาจากการเปรียบเทียบค่าของวงจร Comparator ตัวอย่างการใช้งานได้แก่การสื่อสารผ่านดาวเทียม, ออสซิสโลสโคป, ระบบประมวลผลเรดาร์

การเชื่อมต่อ ADCในไมโครคอนโทรลเลอร์ของ NUVOTON



ภาพที่ 3.12 ขาการใช้งาน ADC [1]

จากวงจรด้านบนรับค่าจาก VR ในชุดทดลองของ NUVOTON NUC140 โดย VR จะปรับค่าตั้งแต่ 0-5V โดยไมโครคอนโทรลเลอร์จะแปลงเป็นสัญญาณดิจิตอลได้ 4095 ระดับ

ขั้นตอนการทดลอง

1. เปิดโปรแกรม Keil Uvision4
2. เปิดไฟล์ตามหัวข้อ 2.7.2 การใช้งานโปรแกรม

ตัวอย่างโปรแกรม Variable Resistance

```
#include "NUC1xx.h"
#include "Driver\DrvSYS.h"
#include "Seven_Segment.h"
```

```
void InitADC(void);
void seg_display(int16_t value);

int32_t main (void)
{
	int32_t adc_value;
	UNLOCKREG();
	SYSCLK->PWRCON.XTL12M_EN = 1;
	SYSCLK->CLKSEL0.HCLK_S = 0;
	LOCKREG();

	InitADC();
	while(1)
	{
		while(ADC->ADSR.ADF==0);
		ADC->ADSR.ADF=1;
		adc_value=ADC->ADDR[7].RSLT;
		seg_display(adc_value);
		ADC->ADCR.ADST=1;
	}
}
void InitADC(void)
{
	GPIOA->OFFD|=0x00800000;
	SYS->GPAMFP.ADC7_SS21_AD6=1;
	SYSCLK->CLKSEL1.ADC_S = 2;
	SYSCLK->CLKDIV.ADC_N = 1;
	SYSCLK->APBCLK.ADC_EN = 1;
```

```
    ADC->ADCR.ADEN = 1;
    ADC->ADCR.DIFFEN = 0;
    ADC->ADCR.ADMD   = 0;
    ADC->ADCHER.CHEN = 0x80;
    ADC->ADSR.ADF =1;
    ADC->ADCR.ADIE = 1;
    ADC->ADCR.ADST=1;
}

void seg_display(int16_t value)
{
    int8_t digit;
    digit = value / 1000;
    close_seven_segment();
    show_seven_segment(3,digit);
    DrvSYS_Delay(5000);
    value = value - digit * 1000;
    digit = value / 100;
    close_seven_segment();
    show_seven_segment(2,digit);
    DrvSYS_Delay(5000);
    value = value - digit * 100;
    digit = value / 10;
    close_seven_segment();
    show_seven_segment(1,digit);
    DrvSYS_Delay(5000);
    value = value - digit * 10;
    digit = value;
```

```
        close_seven_segment();
        show_seven_segment(0,digit);
        DrvSYS_Delay(5000);
}
```

3. ทำการ Compile โปรแกรม

4. Download Program ลงชุดทดลอง

สังเกตผลการทดลองและบันทึกผลที่เกิดขึ้น

ให้นักศึกษาอธิบายโปรแกรม

ให้นักศึกษาเขียน Flow Chart ของโปรแกรม

ให้นักศึกษาทำการแก้ไขโปรแกรม โดยให้แสดงค่าออกทางจอกราฟฟิก LCD

ใบงานที่ 7
การต่อใช้งานร่วมกับชุดทดลอง ETT LAB3A โดยการควบคุม DC Motor

วัตถุประสงค์

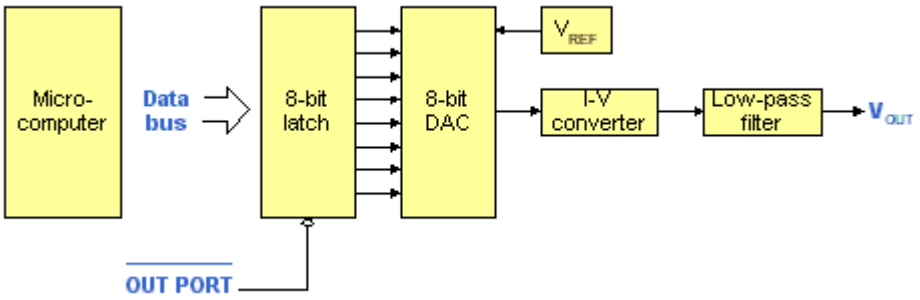
1. เพื่อที่จะศึกษาการเขียนโปรแกรมควบคุม DC Motor
2. เพื่อที่จะสามารถแก้ไขโปรแกรมใช้งานควบคุม DC Motor รูปแบบต่างๆได้

อุปกรณ์การทดลอง

1. เครื่องคอมพิวเตอร์
2. ชุดทดลอง NOVOTON NUC140
3. สาย micro USB
4. สายจั๊มเปอร์เมีย เมีย

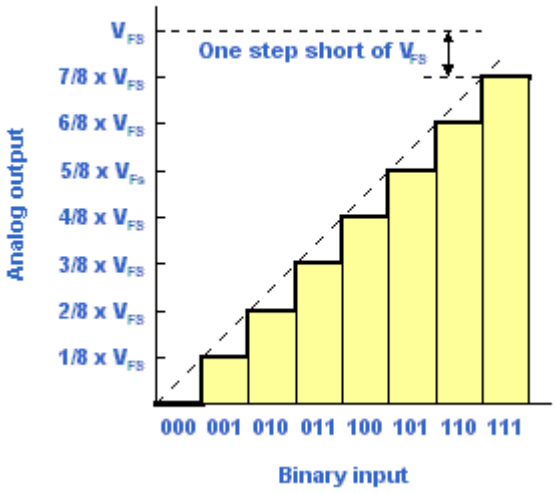
คุณสมบัติของ Digital To Analog Converter [12]

อุปกรณ์ทางไฟฟ้า/อิเล็กทรอนิกส์ โดยทั่วไปที่เป็นอนาล็อก สามารถควบคุมการทำงานโดยการให้อินพุตเป็นระดับ แรงดันที่แตกต่างกัน ตัวอย่างเช่น มอเตอร์กระแสตรง ซึ่งควบคุมความเร็วโดยเปลี่ยนระดับแรงดัน (หรือกระแส) ของขดลวดสนามเมื่อนำระบบดิจิตอลหรือไมโครคอนโทรลเลอร์มาใช้ควบคุมอุปกรณ์ทางอนาล็อกเหล่านี้ จึงต้องมีวงจรซึ่ง สามารถแปลงสัญญาณทางดิจิตอลเป็นระดับแรงดันต่อเนื่อง แบบอนาล็อก ตั้งแต่ศูนย์โวลต์จนถึงระดับสูงสุดที่กำหนดไว้ เรียกว่าวงจร Digital To Analog Converter (DAC)



ภาพที่ 3.13 ระบบการแปลงสัญญาณดิจิตอลเป็นอนาล็อก

จากภาพแสดงถึงส่วนประกอบหลักของระบบ DAC โดยทั่วไป ไมโครคอมพิวเตอร์จะมีเอาต์พุตเป็นค่าไบนารีวงจรแลทช์รับค่าไบนารีเข้ามาเพื่อส่งไปยัง DAC ในวงจรจะใช้แหล่งกำเนิดแรงดันหรือกระแสคงที่เพื่ออ้างอิงในการแปลงข้อมูล ไบนารีเป็นระดับกระแส ต่อมาจะมีวงจรแปลงจากกระแสเป็นระดับแรงดัน (Current-To-Voltage Converter) ซึ่งปกติจะใช้ออปแอมป์ ท้ายสุดสัญญาณอนาล็อกที่ได้จะผ่านวงจร Low-Pass Filter เพื่อกำจัดสัญญาณความถี่สูงที่แฝงอยู่ในสัญญาณที่ถูกสร้างขึ้นมา



ภาพที่ 3.14 Transfer Curve ในอุดมคติของ DAC 3 บิต

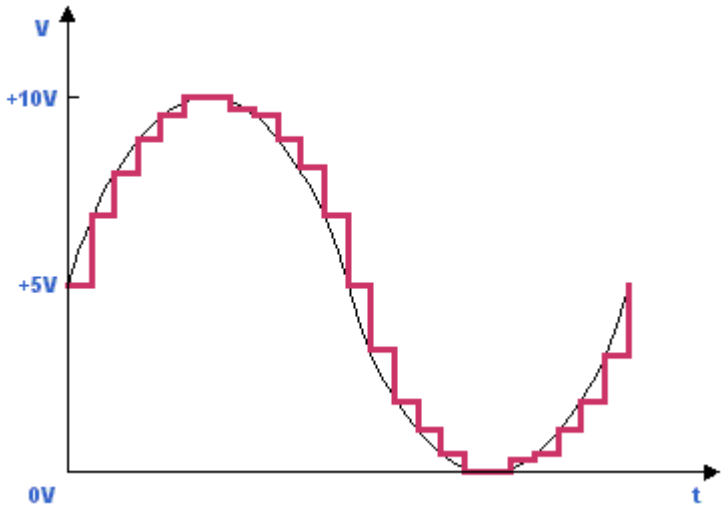
จากภาพเป็นกราฟแสดงถึงความสัมพันธ์ระหว่างเอาต์พุตที่เป็นอนาล็อกกับอินพุตที่เป็นดิจิตอล ขนาด 3 บิตเรียกว่า Transfer Curve สังเกตว่าเมื่ออินพุตไบนารีเพิ่มขึ้น เอาต์พุตอนาล็อกจะเพิ่มในลักษณะขั้นบันได ขนาดของแต่ละขั้นจะหาได้จาก

$$\text{stepsize} = V_{FS}/2^n$$

เมื่อให้ V_{FS} คือระดับแรงดันเอาต์พุตสูงสุด

n คือจำนวนบิตของอินพุต

เนื่องจากเอาต์พุตของ DAC จะเพิ่มเป็นขั้นๆ รูปคลื่นสัญญาณ ที่ได้จาก DAC จึงมีลักษณะไม่เรียบ ดังตัวอย่างในภาพแสดงถึงสัญญาณไซน์ ที่สร้างจาก DAC



ภาพที่ 3.15 คลื่นไซน์ที่สร้างจาก DAC

ถ้าเพิ่มจำนวนบิต ความละเอียดของ DAC จะเพิ่มขึ้น เช่น เมื่อ ใช้ DAC 12 บิต และ V_{FS} = 5.0 V ความละเอียดคือ 5.0 V / 4096 = 1.22 mV ซึ่งจะ ละเอียดกว่า DAC 8 บิตถึง 16 เท่าความถูกต้องของ DAC ขึ้นอยู่กับหลายส่วน

1. Quantization Error

DAC บิต V_{FS} = 5.0 V เอาต์พุตจะมีความละเอียด 19.53 mV ถ้าต้องการเอาต์พุต 4.00 V DAC จะให้เอาต์พุตได้ใกล้เคียง ที่สุดคือ 4.04 V (19.53 mV x 205) ผิดพลาด 4 mV โดยทั่วไป ค่าผิด พลาดจะเท่ากับ +/- 0.5 LSB (Least sSignificant Bit) ตัวอย่างเช่น DAC 8 บิต ความผิดพลาดจะเป็น 1 ใน 512 หรือ +/- 0.195 %

2. Offset And Gain Errors

เมื่ออินพุตไบนารีเท่ากับ 0 แต่เอาต์พุตของ DAC ไม่เป็น 0 เรียก ว่า Offset Error และอาจเกิดร่วมกับ Gain Error ความผิดพลาดเหล่า นี้จะทำให้ Transfer Curve โค้งขึ้นหรือลง ขึ้นอยู่กับความไม่สมดุลภายใน DAC อย่างไรก็ตาม Offset Error และ Gain Error จะแก้ได้โดยใช้ความต้านทานปรับค่าได้ต่อไว้ภายนอก

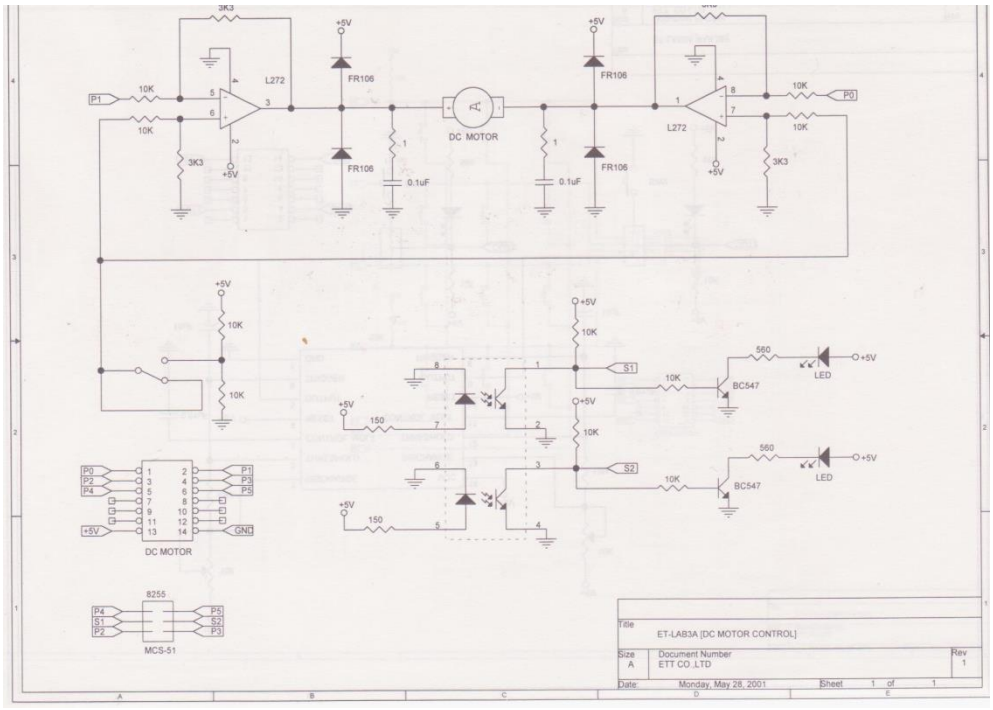
3. Nonlinearity

คือค่าความคลาดเคลื่อนสูงสุดของ Transfer Curve เทียบกับเส้นตรงจากจุดศูนย์และจุดสูงสุด ซึ่งจะขึ้นอยู่กับความผิดพลาดของส่วนประกอบภาย ใน DAC ใน Data Sheet ของ DAC จะระบุเป็นเปอร์เซ็นต์เทียบกับค่าสูงสุด หรือระบุเป็นเศษส่วนของ LSB (โดยทั่วไปคือ +/- 0.5 LSB)

4. Settling time

คือช่วงเวลานับแต่ให้อินพุตจนกระทั่ง DAC ให้เอาต์พุต วัดเมื่อเอาต์พุตที่ได้ผิดพลาดจากค่าจริงน้อยกว่า 0.5 LSB ค่าเวลานี้อาจน้อยกว่า 100 ns สำหรับ DAC ความเร็วสูง และอาจมากกว่า 100 us สำหรับ DAC ราคาถูก

บล็อกไดอะแกรมชุดทดลองของ ETT-LAB3A



ภาพที่3.16 บล็อกไดอะแกรมชุดทดลองของ ETT-LAB3A

จากบล็อกไดอะแกรมด้านบนเป็นการต่อวงจรชุดขับมอเตอร์ในชุดทดลองของ ETT-LAB3A สามารถต่อผ่านพอร์ตที่มีอยู่ในชุดทดลองและต่อกับพอร์ต GPIO A 0-3 ของชุดทดลอง NUVOTON NUC140 โดยทำการต่อกับพอร์ต P0-3 ในวงจรขับมอเตอร์ของชุดทดลอง ETT-LAB3A และต่อไฟพอร์ต P10 และต่อกราวด์พอร์ต P9

ขั้นตอนการทดลอง

1. เปิดโปรแกรม Keil Uvision4
2. เปิดไฟล์ตามหัวข้อ 2.7.2 การใช้งานโปรแกรม

ตัวอย่างโปรแกรมการควบคุม DC Motor

```
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvSYS.h"

void InitGPIO();
void Forward();
void Backward();
void Stop();

int main (void)
{
	UNLOCKREG();
	DrvSYS_Open(48000000);
	LOCKREG();
	Initial_panel();
	clr_all_panel();
	InitGPIO();
	Forward();
	DrvSYS_Delay(45000000);
	Backward();
	DrvSYS_Delay(45000000);
	Stop();
	while(1)
	{
		Forward();
```

```
                DrvSYS_Delay(45000000);
                Stop();
                Backward();
                DrvSYS_Delay(45000000);
        }
}
void InitGPIO()
{
        DrvGPIO_Open(E_GPA,0,E_IO_OUTPUT);
        DrvGPIO_Open(E_GPA,1,E_IO_OUTPUT);
        DrvGPIO_ClrBit(E_GPA,0);
        DrvGPIO_ClrBit(E_GPA,1);
}
void Forward()
{
        DrvGPIO_SetBit(E_GPA,0);
        DrvGPIO_ClrBit(E_GPA,1);
}
void Backward()
{
        DrvGPIO_ClrBit(E_GPA,0);
        DrvGPIO_SetBit(E_GPA,1);
}
void Stop()
{
        DrvGPIO_ClrBit(E_GPA,0);
        DrvGPIO_ClrBit(E_GPA,1);
}
```

3. ทำการ Compile โปรแกรม

4. Download Program ลงชุดทดลอง

สังเกตผลการทดลองและบันทึกผลที่เกิดขึ้น

ให้นักศึกษาอธิบายโปรแกรม

ให้นักศึกษาเขียน Flow Chart ของโปรแกรม

ให้นักศึกษาทำการแก้ไขโปรแกรมโดยให้ DC Motor หมุนไปทางด้านซ้าย 3 ครั้งและทางด้านขวา 2 ครั้ง

ใบงานที่ 8

การต่อใช้งานร่วมกับชุดทดลอง ETT LAB3A โดยการควบคุม Stepping Motor

วัตถุประสงค์

1. เพื่อที่จะศึกษาการเขียนโปรแกรมควบคุม Stepping Motor
2. เพื่อที่จะสามารถแก้ไขโปรแกรมใช้งานควบคุม Stepping Motor รูปแบบต่างๆได้

อุปกรณ์การทดลอง

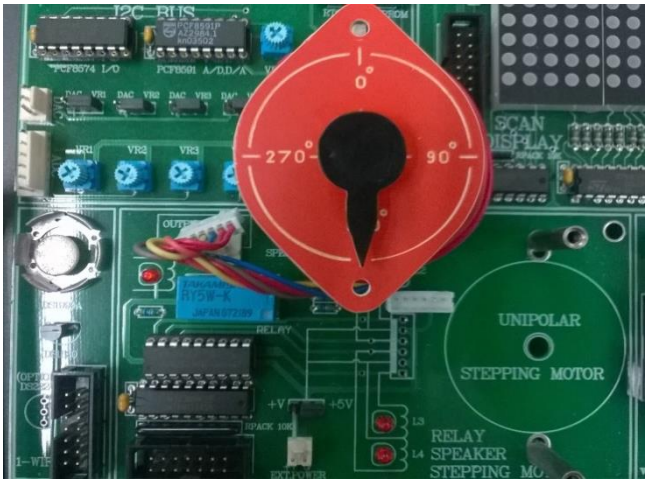
1. เครื่องคอมพิวเตอร์
2. ชุดทดลอง NOVOTON NUC140
3. สาย micro USB
4. สายจั๊มเปอร์เมีย เมีย

สเตปปิ้งมอเตอร์ (Stepping Motor) [13]

สเต็ปปิ้งมอเตอร์เป็นอุปกรณ์เอาต์พุตอย่างหนึ่ง ซึ่งสามารถนำไอซีไมโครคอนโทรลเลอร์มาทำการควบคุมได้สะดวก และเป็นมอเตอร์ที่เหมาะสมสำหรับใช้ในงานควบคุมการหมุน ที่ต้องการตำแหน่ง และทิศทางที่แน่นอน การทำงานของ สเต็ปปิ้งมอเตอร์จะขับเคลื่อนทีละขั้นๆ ละ (Step) 0.9, 1.8, 5, 7.5, 15 หรือ 50 องศา ซึ่งขึ้นอยู่กับคุณสมบัติแต่ละชนิดของสเต็ปปิ้งมอเตอร์ตัวนั้นๆ สเต็ปปิ้งมอเตอร์จะแตกต่างจากมอเตอร์กระแสตรงทั่วไป (DC MOTOR) โดยการทำงานของมอเตอร์กระแสตรงจะหมุนไปแบบต่อเนื่อง ไม่สามารถหมุนเป็นแบบสเต็ปๆ ได้ดังนั้นในการนำไปกำหนดตำแหน่งจึงควบคุมได้ยากกว่า แต่ในส่วนใหญ่เราจะใช้สเต็ปปิ้งมอเตอร์มาทำการการควบคุมโดยใช้วิธีในระบบดิจิตอล เช่น พรินเตอร์ (Printer) พล็อตเตอร์ (X-Y Plotter) ดิสก์ไดร์ฟ (Disk drive) ฯลฯ

ข้อดีของสเต็ปปิ้งมอเตอร์เมื่อเปรียบกับมอเตอร์กระแสตรง (DC MOTOR)

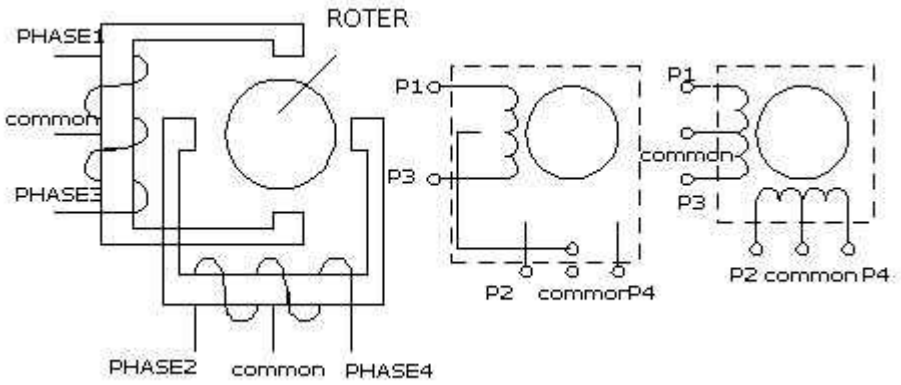
1. การควบคุมไม่ต้องอาศัยตัวตรวจจับการหมุน
2. ไม่ต้องใช้แปรงถ่าน ดังนั้นจึงทำให้ไม่มีส่วนที่จะต้องสึกหรอ และปัญหาของการสปาร์ค (ที่เกิดจากหน้าสัมผัสของแปรงถ่านแหวนตัวนำในโรเตอร์) ที่ทำให้เกิดสัญญาณรบกวน
3. การควบคุมโดยทางวงจรดิจิตอลหรือไมโครโพรเซสเซอร์ ทำได้ง่าย และสะดวก



ภาพที่ 3.17 สเต็ปปิ้งมอเตอร์ในชุดทดลอง ETT-LAB3A

สเต็ปปิ้งมอเตอร์ที่จะนำมาใช้ในการทดลองนี้ จะใช้สเต็ปปิ้งแบบยูนิโพล่าร์ (Uni.-Polar Stepper Motor) ซึ่งโครงสร้างของสเต็ปปิ้งมอเตอร์แบบนี้จะมีส่วนประกอบที่สำคัญ 2 ส่วนด้วยกันคือ

1. ส่วนที่ทำการหมุน (Rotor) จะเป็นแม่เหล็กถาวรหรืออื่นๆ
2. ส่วนที่อยู่กับที่ (Stator) เป็นขดลวดที่พันไว้จำนวนหลายๆขด



ภาพที่ 3.18 สเต็ปปิ้งมอเตอร์ 4 เฟส แบบยูนิโพล่าร์ (Uni-Polar Stepper Motor)

วิธีการขับสเต็ปปิ้งมอเตอร์ให้หมุนโดยการกระตุ้นเฟส

ในการควบคุมสเต็ปปิ้งมอเตอร์เพื่อที่จะให้ทำการหมุน มีวิธีการควบคุมกระแสไฟที่จ่ายให้กับขดลวดสเตเตอร์ (Stator) ในแต่ละเฟสของสเต็ปปิ้งมอเตอร์ อย่างเป็นลำดับที่แน่นอน โดยถ้าหากเราต้องการให้กระแสไหลในเฟสใดๆ ก็จะทำให้สถานะของเฟสนั้นๆเป็นสถานะลอจิก "1" และในการกระตุ้นเฟสของของสเต็ปปิ้งมีอยู่ด้วยกัน 2 แบบคือ

1. การกระตุ้นเฟส แบบฟูลสเต็ปมอเตอร์ (Full Step Motor)

ยังสามารถแบ่งการกระตุ้นเฟสออกได้เป็นอีก 2 วิธีด้วยกันคือ

1.1 การกระตุ้นเฟสแบบฟูลสเต็ป 1 เฟส (Single-Phase Driver)

หรือแบบเวฟ แสดงดังตารางที่ 1 จะเป็นการป้อนกระแสไฟให้กับขดลวด ของสเต็ปปิ้งมอเตอร์ทีละขด โดยจะป้อนกระแสเรียงตามลำดับกันไป ดังนั้นกระแส ที่ไหลในขดลวด จะทำการไหลในทิศทางเดียวกันทุกขด ลักษณะเช่นนี้จึงทำให้แรงขับของสเต็ปปิ้งมอเตอร์มีน้อย

1.2 การกระตุ้นเฟสแบบฟูลสเต็ป 2 เฟส (Two-Phase Driver)

ตารางที่ 2 เป็นการป้อนกระแสให้กับขดลวด 2 ขด ของสเต็ปปิ้งมอเตอร์พร้อมๆ กันไป และจะกระตุ้นเรียงถัดกับไปเช่นเดียวกับแบบหนึ่งเฟส ดังนั้นการกระตุ้นแบบนี้จึงต้องใช้กำลังไฟมากขึ้น และจะทำให้มีแรงบิดของมอเตอร์มากกว่าการกระตุ้นแบบ 1 เฟส

2.การกระตุ้นเฟส แบบฮาล์ฟสเต็ป (Half Step Motor)

หรือ One-Two Phase Driver คือการกระตุ้นเฟสแบบฟลูสเต็ป 1 เฟส และ 2 เฟส เรียงลำดับกันไป แสดงดังตารางที่ 3 แรงบิดที่ได้จากการกระตุ้นเฟสแบบนี้จะมีเพิ่มมากขึ้น เพราะช่วงของสเต็ปมีระยะสั้นลง ในการกระตุ้นแบบนี้ เราจะต้องมีการกระตุ้นที่เฟสถึง 2 ครั้ง จึงจะได้ระยะของ สเต็ปเท่ากับการกระตุ้นเพียงครั้งเดียว ของแบบฟูลสเต็ป 2 แบบแรก ความละเอียดของการหมุนตำแหน่งองศาต่อสเต็ป ก็เป็นสองเท่าของแบบแรก ความถูกต้องของตำแหน่งที่กำหนดจึงมีมากขึ้น

ตางรางที่ 3.3 แบบฟูลสเต็ป 1 เฟส

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

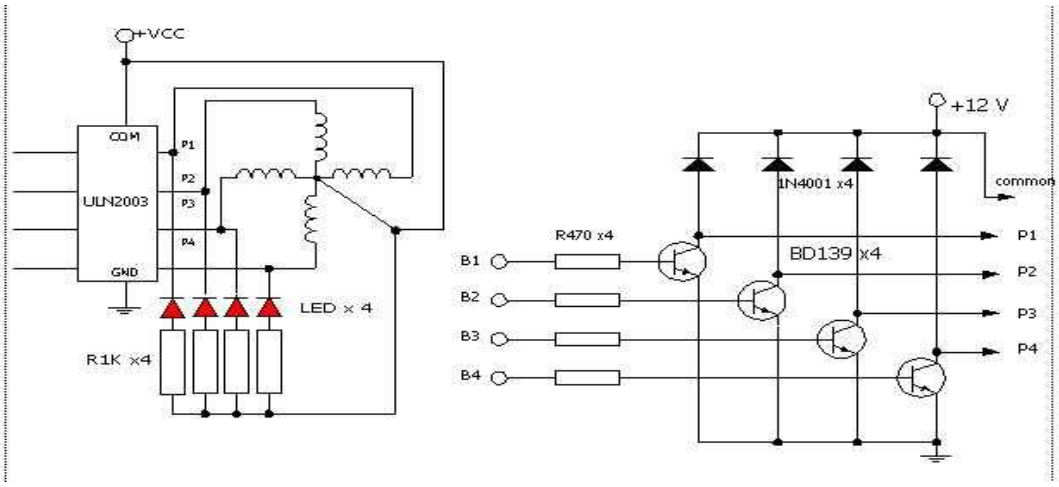
ตารางที่ 3.4 แบบฟูลสเต็ป 2 เฟส

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

ตารางที่ 3.5 แบบฮาล์ฟสเต็ป 2 เฟส

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

วงจรที่ใช้ในการขับสเต็ปมิ้งมอเตอร์โดยใช้ไอซีสำเร็จรูป และวงจรจากทรานซิสเตอร์ โดยใช้ไอซีสำเร็จรูปเบอร์ ULN2003 จะมีคุณสมบัติเป็นไอซีไดเวอร์กระแสสูงแบบคอลเล็คเตอร์เปิด สามารถเลือกแรงดันได้กว้าง 5-30 V จ่ายกระแสได้สูงถึง 500 mA ต่อขา และมีไดโอดที่ป้องกันกระแสย้อนกลับอยู่ภายในไอซี ส่วนแอลอีดีที่ต่อในวงจรเราจะต่อไว้เพื่อแสดงการกระตุ้นแต่ละเฟสของแต่ละแบบ



ภาพที่ 3.19 แสดงการต่อวงจรขับสเต็ปมิ้งมอเตอร์โดยใช้ไอซีสำเร็จรูป และวงจรทรานซิสเตอร์

วิธีการตรวจสอบหาเฟสของขดลวด สเต็ปปิ้งมอเตอร์

ในขั้นตอนที่ 1

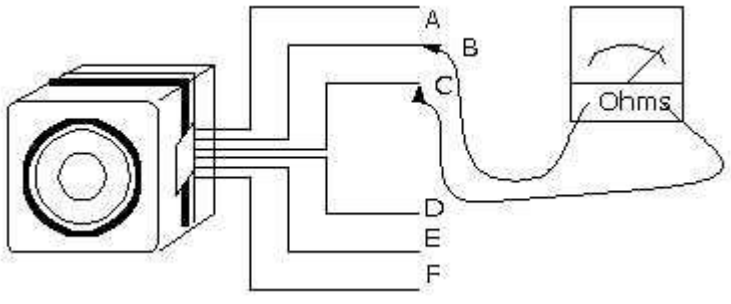
ให้สังเกตว่า สเต็ปปิ้งมอเตอร์ที่นำมาทดลองที่เป็นแบบยูนิโพล่าร์ (Uni-Polar Stepper Motor) จะมีจำนวนสาย 5 เส้นหรือ 6 เส้น (ถอดจากดิสก์ไดร์ฟเก่าขนาด 5 นิ้วนำมาใช้งานได้)

ในขั้นตอนที่ 2

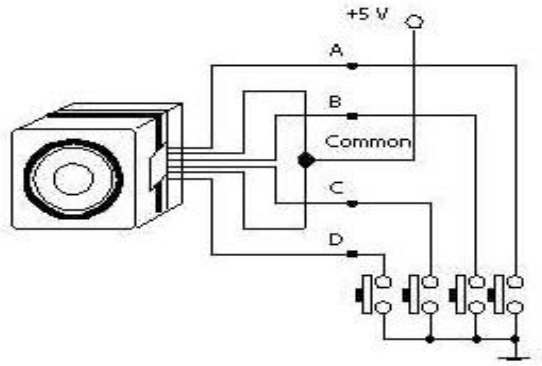
ใช้มิเตอร์วัดค่าความต้านทานของเส้นลวดในแต่ละขดขั้นตอนการวัด ให้หาสายที่ต่อเป็นจุดร่วมเสียก่อน(Common) โดยให้ใช้ มัลติมิเตอร์ตั้งค่าไว้สำหรับการวัดค่าความต้านทาน แต่ละเส้นสังเกตที่ค่าความต้านทาน ถ้าหากเราไม่ได้วัดระหว่าง จุดต่อร่วม(Common) กับสายแต่ละเส้น ค่าความต้านทานจะมีค่าเป็น 2 เท่าของการวัดระหว่างจุดต่อร่วมกับสายที่ใช้งาน ตัวอย่างเช่น ถ้าให้จุด B เป็นจุดร่วม หากวัดระหว่างที่จุด A กับจุด B จะมีค่าเท่ากับ 60 Ohm แต่ถ้าวัดระหว่างที่จุด A และจุด C ซึ่งไม่ใช่จุดร่วมก็จะได้ค่าเท่ากับ 120 Ohm หากเป็นแบบที่มีสาย 6 เส้นก็จะมีจุดร่วมสองจุด เพราะมีขดลวดคนละชุดกัน และสายที่เป็นจุดร่วมส่วนใหญ่จะมีสีเหมือนกัน ทำนองเดียวกันหากเป็นแบบที่มีสาย 5 เส้นก็จะมีจุดร่วมเพียงจุดเดียวเท่านั้น

ในขั้นตอนที่ 3

หากเป็นแบบที่มีสาย 6 เส้นก็ให้ทำการต่อจุดร่วมเข้าด้วยกันจะได้เป็น 5 เส้น แล้วต่อวงจรตาม รูปหลังจากนั้นให้ทดลองกดสวิตซ์ ที่ต่อเข้ากับแต่ละจุดโดยเริ่มที่ จุด A จุด B จุด C และจุด D แล้วให้สังเกตการหมุนของสเต็ปปิ้งมอเตอร์ว่าหมุนได้ต่อเนื่องหรือไม่ หากมีการกระโดดข้ามสเต็ปก็ให้ทดลองโดยเรียงลำดับการกดสวิตซ์ใหม่ จนหาลำดับของสายได้ถูกต้องคือมอเตอร์เดินตามที่ละสเต็ป อย่างเป็นลำดับ

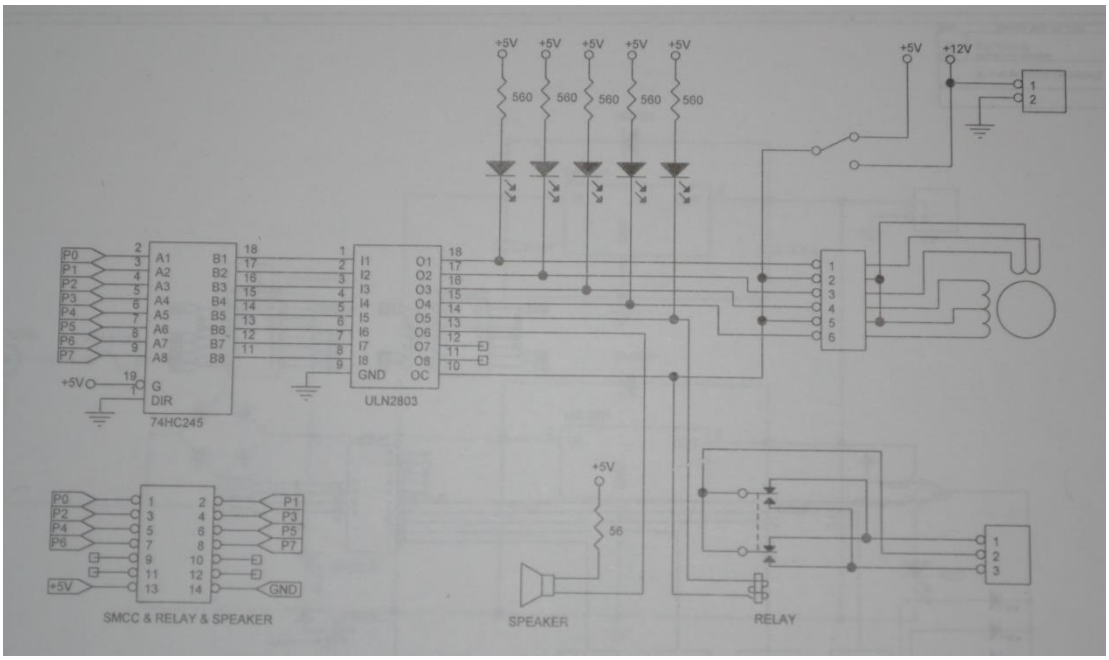


ภาพที่ 3.20 การใช้มิเตอร์วัดค่าความต้านทาน



ภาพที่ 3.21 แสดงการต่อวงจรเพื่อทดสอบโดยการสวิตซ์เพื่อหาลำดับ

บล็อกไดอะแกรมที่ใช้ขับ สเต็ปปิ้งมอเตอร์



ภาพที่ 3.22 วงจรขับสเต็ปปิ้งมอเตอร์ ETT-LAB3A

จากภาพด้านบนแสดงวงจรควบคุมสเต็ปปิ้งมอเตอร์ของบอร์ดทดลอง ETT-LAB3A ซึ่งใช้มอเตอร์สเต็ปปิ้ง ชนิดยูนิโพล่าร์ 48 สเต็ป นั่นคือมีการเคลื่อนที่ไปสเต็ปละ 7.5 องศาเมื่อควบคุม

แบบฟลูสเต็ป โดย สัญญาณลอจิกที่เข้ามาควบคุมจะถูกส่งผ่าน IC 74HC245 ซึ่งทำหน้าที่เป็นบัฟเฟอร์ให้กับไมโคร คอนโทรลเลอร์ และใช้ IC ULN2803 ในการขับกระแสเพื่อควบคุมมอเตอร์ โดยสายสัญญาณของเฟสที่ 1-4 จะถูกต่อกับสายพอร์ตขาที่ P0, 1, 2 และ 3 โดยต่อไปยังพอร์ต GPIO A 0-3 และจ่ายไฟเลี้ยงจากชุดทดลอง NUVOTON NUC140 ไปยังพอร์ต P10 และต่อกราวด์ไปยังพอร์ต P9 เพื่อจ่ายไฟไปยังชุดทดลอง ETT-LAB3A

ขั้นตอนการทดลอง

1 เปิดโปรแกรม Keil Uvision4

2. เปิดไฟล์ตามหัวข้อ 2.7.2 การใช้งานโปรแกรม

ตัวอย่างโปรแกรมการควบคุม Stepping Motor

```
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvSYS.h"

#define d360 12
#define d180 12/2
#define d90  12/4
#define d45  12/8

unsigned char CW[8] ={0x09,0x01,0x03,0x02,0x06,0x04,0x0c,0x08};
unsigned char CCW[8]={0x08,0x0c,0x04,0x06,0x02,0x03,0x01,0x09};
void CW_MOTOR(uint16_t deg);
void CCW_MOTOR(uint16_t deg);

int main (void)
```

```
{
        CW_MOTOR(d360);
        CCW_MOTOR(d180);
}
void CW_MOTOR(uint16_t deg)
{
        int i=0,j=0;
        for(j=0;j<(deg);j++)
        {
                for(i=0;i<8;i++)
                {
                        GPIOA->DOUT=CW[i];
                        DrvSYS_Delay(50000);//delay 2000us = 2ms
                }
        }
}
void CCW_MOTOR(uint16_t deg)
{
        int i=0,j=0;
        for(j=0;j<(deg);j++)
        {
                for(i=0;i<8;i++)
                {
                        GPIOA->DOUT=CCW[i];
                        DrvSYS_Delay(50000);
                }
        }
}
```

3. ทำการ Compile โปรแกรม

4. Download Program ลงชุดทดลอง

สังเกตผลการทดลองและบันทึกผลที่เกิดขึ้น

ให้นักศึกษาอธิบายโปรแกรม

ให้นักศึกษาเขียน Flow Chart ของโปรแกรม

ให้นักศึกษาทำการแก้ไขโปรแกรม โดยให้ Stepping Motor หมุนเดินหน้า 90 องศาถอยหลัง 180 องศาและเดินหน้า 360 องศา

บทที่ 4
เฉลยใบงานการทดลอง

เฉลยใบงานที่ 1 การทดลองการแสดงผลออกทาง LED

สังเกตผลการทดลองและบันทึกผลที่เกิดขึ้น

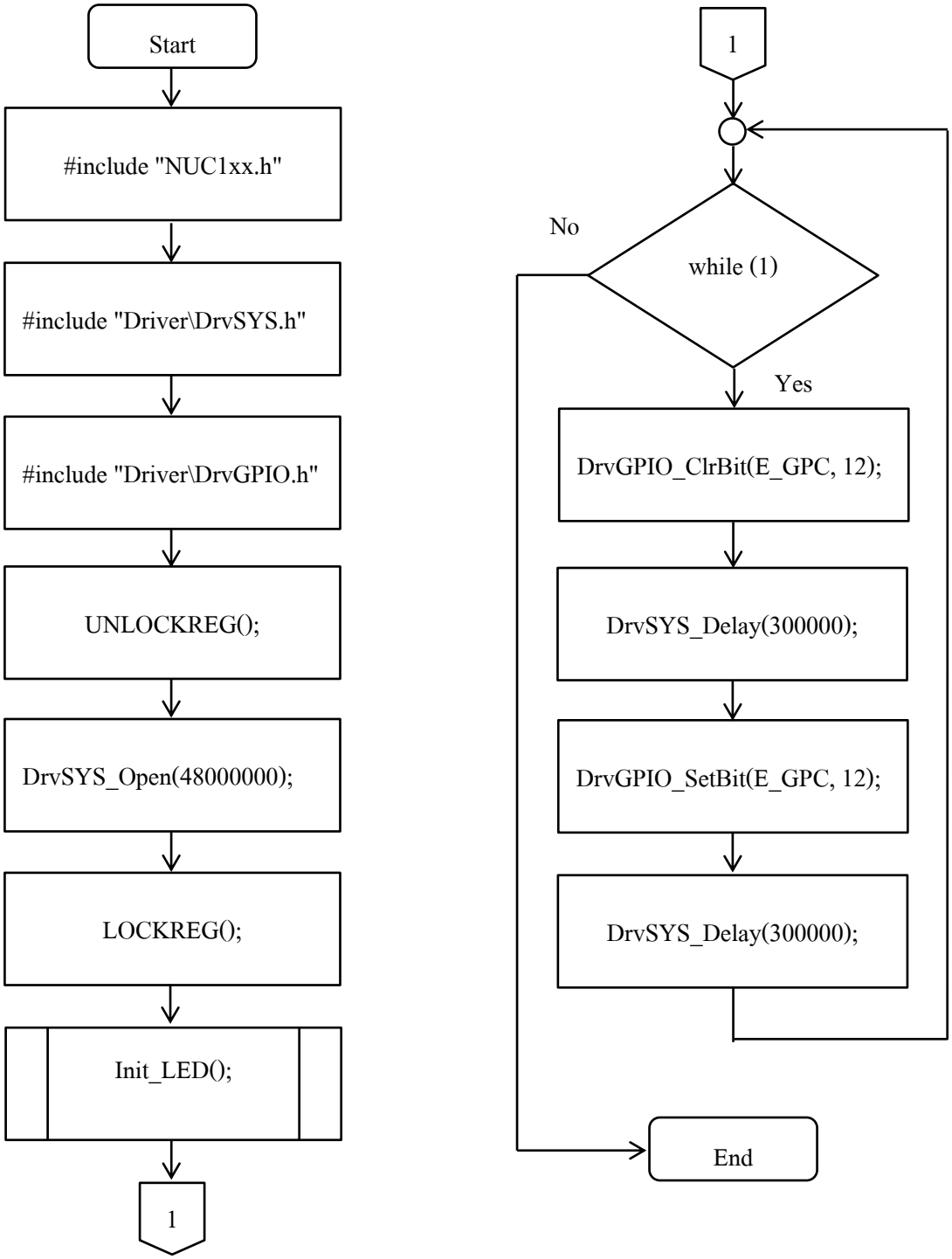
จากการทดลองการแสดงผลออกทาง LED ในชุดทดลองของ NUVOTON NUC140 จากโปรแกรมตัวอย่างที่ให้มา ทำให้เห็นว่า LED ติดและดับเป็นจังหวะตามการสั่งงานของโปรแกรม โดย LED จะติดดวงแรกทางซ้ายมือจาก LED ทั้งหมด 4ดวง

ให้นักศึกษาอธิบายโปรแกรม

โปรแกรมการติดดับของ LED เริ่มต้นการ #include "NUC1xx.h" เป็นการดึงไดเวอร์ของชิป NUVOTON มาใช้งานเป็นค่าเริ่มต้มจากนั้น #include "Driver\DrvSYS.h" ดึงค่าการใช้งานของซิสเท็มไดเวอร์ออกมาเพื่อตั้งค่าส่วนต่างๆภายในไอซี #include "Driver\DrvGPIO.h"เป็นเซ็ตค่าขาการใช้งานที่อยู่บนชุดทดลอง NUVOTON

ในชุดคำสั่ง main เริ่มต้น UNLOCKREG(); เป็นการปลดล็อกรีจิสเตอร์เพื่อเข้าไปเซ็ตค่าความเร็วของ CPU โดยเซ็ตค่าความเร็วในการทำงานไว้ที่ 48 MHz จากนั้นก็ทำการปิดรีจิสเตอร์ LOCKREG(); เริ่มคำสั่ง LED ทำงานโดยการเขียนโปรแกรมสั่งงานกำหนดอินพุตให้กับ DrvGPIO_Open(E_GPC, 12, E_IO_OUTPUT);เปิดการทำงานโดยใช้พอร์ตขา E 12 เป็นขาเอาต์พุต DrvGPIO_ClrBit(E_GPC, 12); เป็นการเซ็ตค่าให้บิตเป็น 0 เพื่อให้หลอด LED ทำงาน จากนั้นหน่วงเวลาให้ LED แสดงผล และทำการเซ็ตบิตอีกครั้ง DrvGPIO_SetBit(E_GPC, 12); เพื่อให้ LED ดับ และทำงานต่อไปในโหมด while จนครบตามที่เขียนโปรแกรมไว้

เฉลย Flow chart ของโปรแกรม



เฉลยใบงานแก้ไขโปรแกรม โดยให้ LED ติดดับ 4 ดวง

```
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvUART.h"
#include "Driver\DrvSYS.h"

void Init_LED();
int main (void)
{
	UNLOCKREG();
	DrvSYS_Open(48000000);
	LOCKREG();

Init_LED();

	while (1)
	{
		DrvGPIO_ClrBit(E_GPC, 12);
		DrvSYS_Delay(300000);
		DrvGPIO_SetBit(E_GPC, 12);
		DrvSYS_Delay(300000);
		DrvGPIO_ClrBit(E_GPC, 13);
		DrvSYS_Delay(300000);
		DrvGPIO_SetBit(E_GPC, 13);
		DrvSYS_Delay(300000);
		DrvGPIO_ClrBit(E_GPC, 14);
		DrvSYS_Delay(300000);
```

```
                DrvGPIO_SetBit(E_GPC, 14);
                DrvSYS_Delay(300000);
                DrvGPIO_ClrBit(E_GPC, 15);
                DrvSYS_Delay(300000);
                DrvGPIO_SetBit(E_GPC, 15);
                DrvSYS_Delay(300000);
        }
}
void Init_LED()
{
        DrvGPIO_Open(E_GPC, 12, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPC, 13, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPC, 14, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPC, 15, E_IO_OUTPUT);
        DrvGPIO_SetBit(E_GPC, 12);
        DrvGPIO_SetBit(E_GPC, 13);
        DrvGPIO_SetBit(E_GPC, 14);
        DrvGPIO_SetBit(E_GPC, 15);
}
```

เฉลยใบงานที่ 2 การทดลองการแสดงผลออกทาง 7-Segment
สังเกตผลการทดลองและบันทึกผลที่เกิดขึ้น

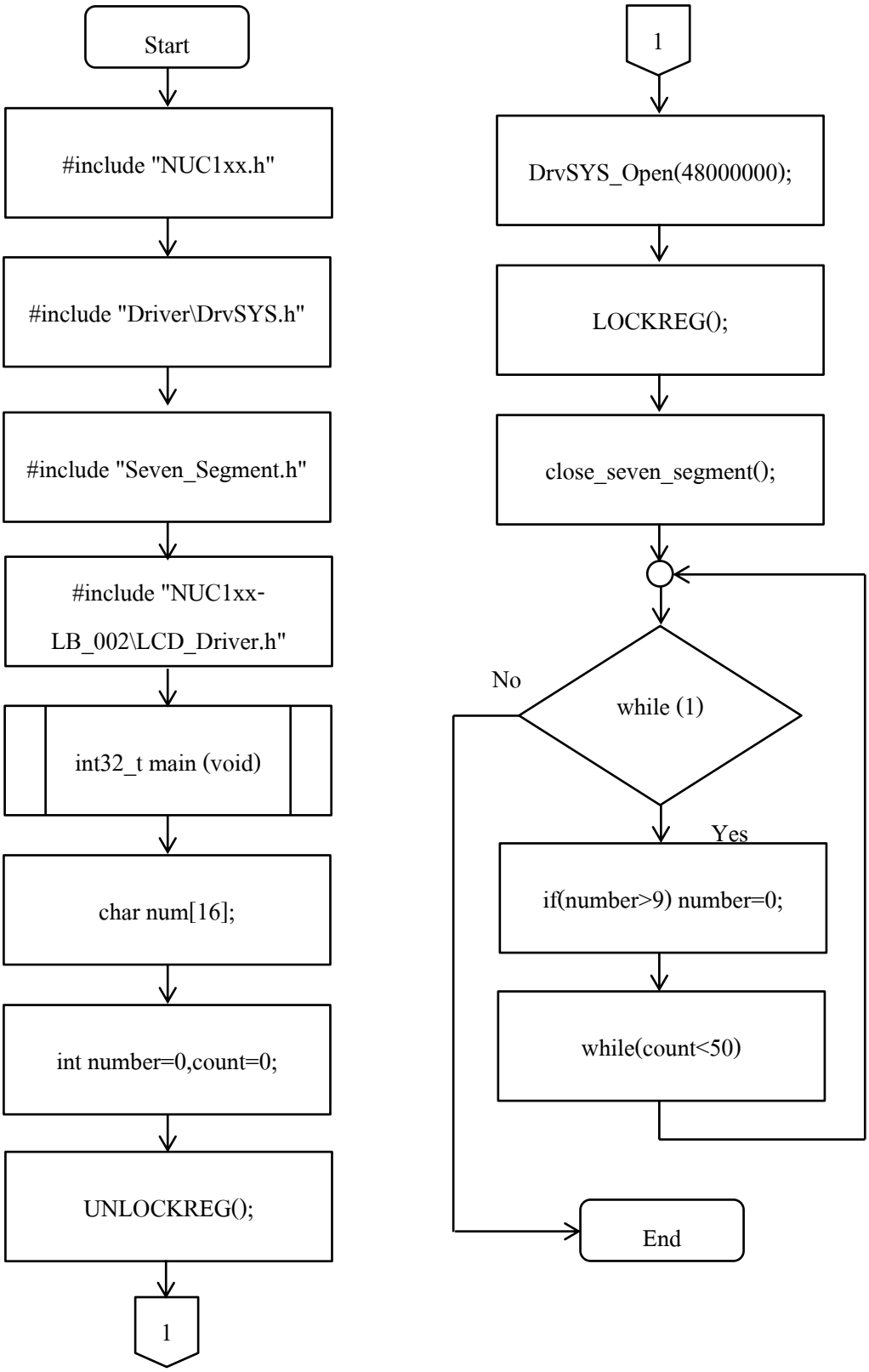
จากการทดลองการแสดงผลออกทาง 7-Segment ในชุดลอง NUVOTON NUC140 จากโปรแกรมที่ให้มาแสดงถึง 7-Segment ที่มีการทำงาน โดยการการนับขึ้นจาก 0 ถึง 9 และเมื่อครบรอบแล้วก็จะมีการรีเซ็ตค่ากลับมาเริ่มนับ 0 ใหม่และนับขึ้นไปแบบนี้เรื่อยๆจบครบคำสั่งของโปรแกรมที่ให้มาในตัวอย่างชุดการทดลอง

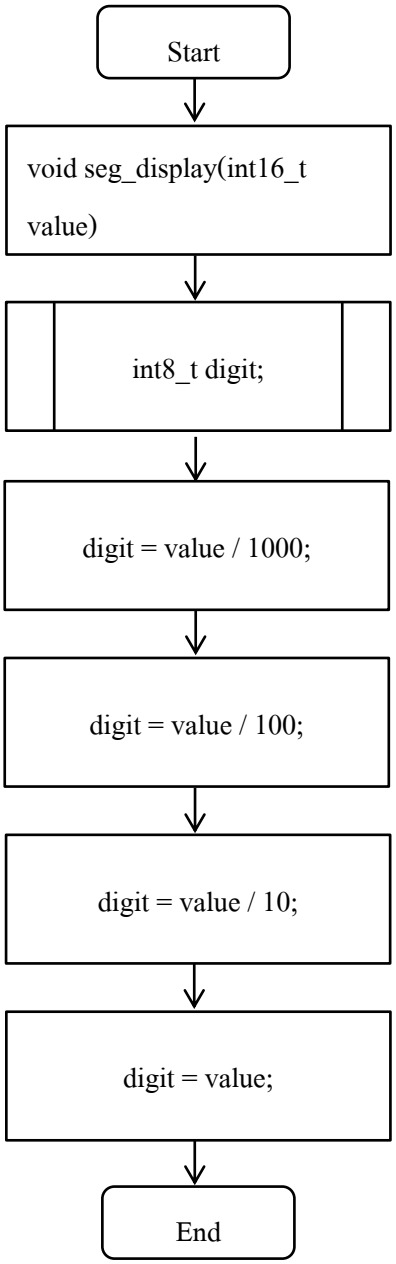
ให้นักศึกษาอธิบายโปรแกรม

โปรแกรมการแสดงผลออกทาง 7-Segment เริ่มต้นการ #include "NUC1xx.h" เป็นการดึงไดเวอร์ของชิป NUVOTON มาใช้งานเป็นค่าเริ่มต้มจากนั้น #include "Driver\DrvSYS.h" ดึงค่าการใช้งานของซิสเท็มไดเวอร์ออกมาเพื่อตั้งค่าส่วนต่างๆภายในไอซี #include "Seven_Segment.h" เป็นการเซ็ตค่าขา 7-Segment ที่อยู่บนชุดทดลอง NUVOTON

ในชุดคำสั่ง main เริ่มต้น UNLOCKREG(); เป็นการปลดล็อกรีจิสเตอร์เพื่อเข้าไปเซ็ตค่าความเร็วของ CPU โดยเซ็ตค่าความเร็วในการทำงานไว้ที่ 48 MHz จากนั้นก็ทำการปิดรีจิสเตอร์ LOCKREG(); จากนั้นประกาศให้ 7-Segment บิตสุดท้ายทำงานโดยใช้คำสั่ง digit = value เพื่อเลือกแสดงหลัก 7-Segment จากนั้นทำการเคลีย 7-Segment โดยใช้คำสั่ง close_seven_segment();เพื่อเป็นการเซ็ตค่าเริ่มต้น คือ 0 จากนั้นใช้คำสั่ง show_seven_segment เพื่อแสดงตัวเลขตามที่โปรแกรมไว้ และทำการหน่วงเวลาการแสดงผลของ 7-Segment โดยใช้คำสั่ง if(number>9) number=0; เป็นคำสั่งนับขึ้นจาก 0 ถึง 9 และวนกลับมาที่ 0 ดังเดิม

เฉลยใบงานที่ 2 Flow chart ของโปรแกรม




Start
void seg_display(int16_t value)
int8_t digit;
digit = value / 1000;
digit = value / 100;
digit = value / 10;
digit = value;
End

เฉลยใบงานที่ 2 โปรแกรม

```
#include "NUC1xx.h"
#include "Driver\DrvSYS.h"
#include "Seven_Segment.h"
#include "NUC1xx-LB_002\LCD_Driver.h"

void seg_display(int16_t value);
int32_t main (void)
{
	char num[16];
	int number=0,count=0;
	UNLOCKREG();
	DrvSYS_Open(48000000);
	LOCKREG();
	close_seven_segment();
	while(1)
	{

		{
			seg_display(number);
			count++;
		}
			count=0;
			number++;
			if(number>99) number=0;
			while(count<50)
		{
```

```
                    seg_display(number);
                    count++;
                }
                count=0;
                number++;
        }
}
void seg_display(int16_t value)
{
        int8_t digit;
                digit = value / 1000;
                close_seven_segment();
                show_seven_segment(3,digit);
                DrvSYS_Delay(5000);
                close_seven_segment();
                value = value - digit * 1000;
                digit = value / 100;
                close_seven_segment();
                show_seven_segment(2,digit);
                DrvSYS_Delay(5000);
                value = value - digit * 100;
                digit = value / 10;
                close_seven_segment();
                show_seven_segment(1,digit);
                DrvSYS_Delay(5000);
                value = value - digit * 10;
                digit = value;
                close_seven_segment();
```

```
                show_seven_segment(0,digit);
                DrvSYS_Delay(5000);
}
```

เฉลยใบงานที่ 3 การทดลองการแสดงผลออกทาง Graphic LCD
สังเกตผลการทดลองและบันทึกผลที่เกิดขึ้น

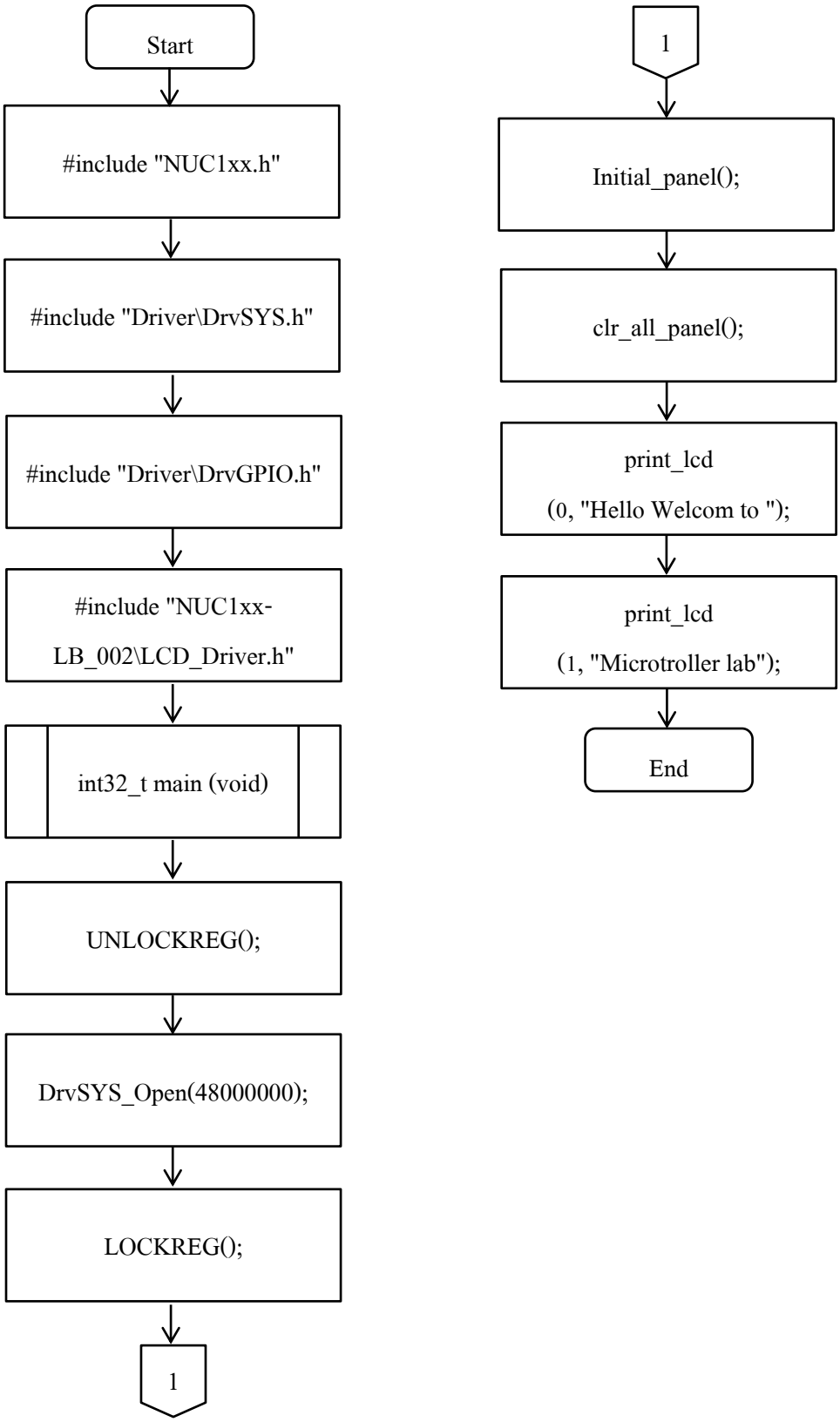
จากการทดลองการแสดงผลออกทาง Graphic LCD บนชุดทดลอง NUVOTON NUC140 ที่มีจอแสดงผล Graphic LCD ที่ขนาด 128x64 pixel ซึ่งแสดงตัวอักษรได้ 16 ตัวอักษรต่อหนึ่งบรรทัดและมีบรรทัดทั้งหมด 4 บรรทัด ให้แสดงค่าด้วยกันจากโปรแกรมที่ให้มาเมื่อทำการ compile เสร็จแล้วก็จะแสดงคำว่า Hello Welcom to ที่อยู่บนบรรทัดแรกและบรรทัดที่สองจะแสดงคำว่า Microtroller lab จากการสังเกตทำให้เห็นว่าจะมีบรรทัดว่างอยู่อีกสองบรรทัดเพื่อที่จะใส่โปรแกรมลงไปแสดงผลได้อีก

ให้นักศึกษาอธิบายโปรแกรม

โปรแกรมการแสดงผลออกทาง Graphic LCD เริ่มต้นการ #include "NUC1xx.h" เป็นการดึงไดเวอร์ของชิป NUVOTON มาใช้งานเป็นค่าเริ่มต้มจากนั้น #include "Driver\DrvSYS.h" ดึงค่าการใช้งานของซิสเท็มไดเวอร์ออกมาเพื่อตั้งค่าส่วนต่างๆภายในไอซี #include "Driver\DrvGPIO.h"เป็นเซ็ตค่าขาการใช้งานที่อยู่บนชุดทดลอง NUVOTON จากนั้นเลือกเซ็ตค่าการใช้งานจอ LCD #include "NUC1xx-LB_002\LCD_Driver.h" เพื่อดึงไลบราลี่ของการแก้ไขจอ LCD

ในชุดคำสั่ง main เริ่มต้น UNLOCKREG(); เป็นการปลดล็อกรีจิสเตอร์เพื่อเข้าไปเซ็ตค่าความเร็วของ CPU โดยเซ็ตค่าความเร็วในการทำงานไว้ที่ 48 MHz จากนั้นก็ทำการปิดรีจิสเตอร์ LOCKREG(); จากให้ทำการแสดงผลจอ LCD โดยใช้ชุดคำสั่ง print_lcd(0, "Hello Welcom to "); เป็นคำสั่งโดยให้แสดงค่าที่บรรทัดแรก จากนั้น print_lcd(1, "Microtroller lab");เพื่อให้แสดงค่าบรรทัดที่สอง

เฉลยใบงานที่ 3 Flow chart ของโปรแกรม



เฉลยใบงานที่ 3 โปรแกรม

```
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvSYS.h"
#include "Driver\DrvGPIO.h"
#include "NUC1xx-LB_002\LCD_Driver.h"

int main(void)
{
        UNLOCKREG();
        DrvSYS_Open(48000000);
        LOCKREG();

        Initial_panel();
        clr_all_panel();

        print_lcd(0, "Hello Welcom to ");
        print_lcd(1, "Microtroller lab");
        print_lcd(2, "SRIPATUM UNI.   ");
        print_lcd(3, "BY SARGMET GROUP");
}
```

เฉลยใบงานที่ 4 การทดลองการรับค่าจาก Matrix Switch
สังเกตผลการทดลองและบันทึกผลที่เกิดขึ้น

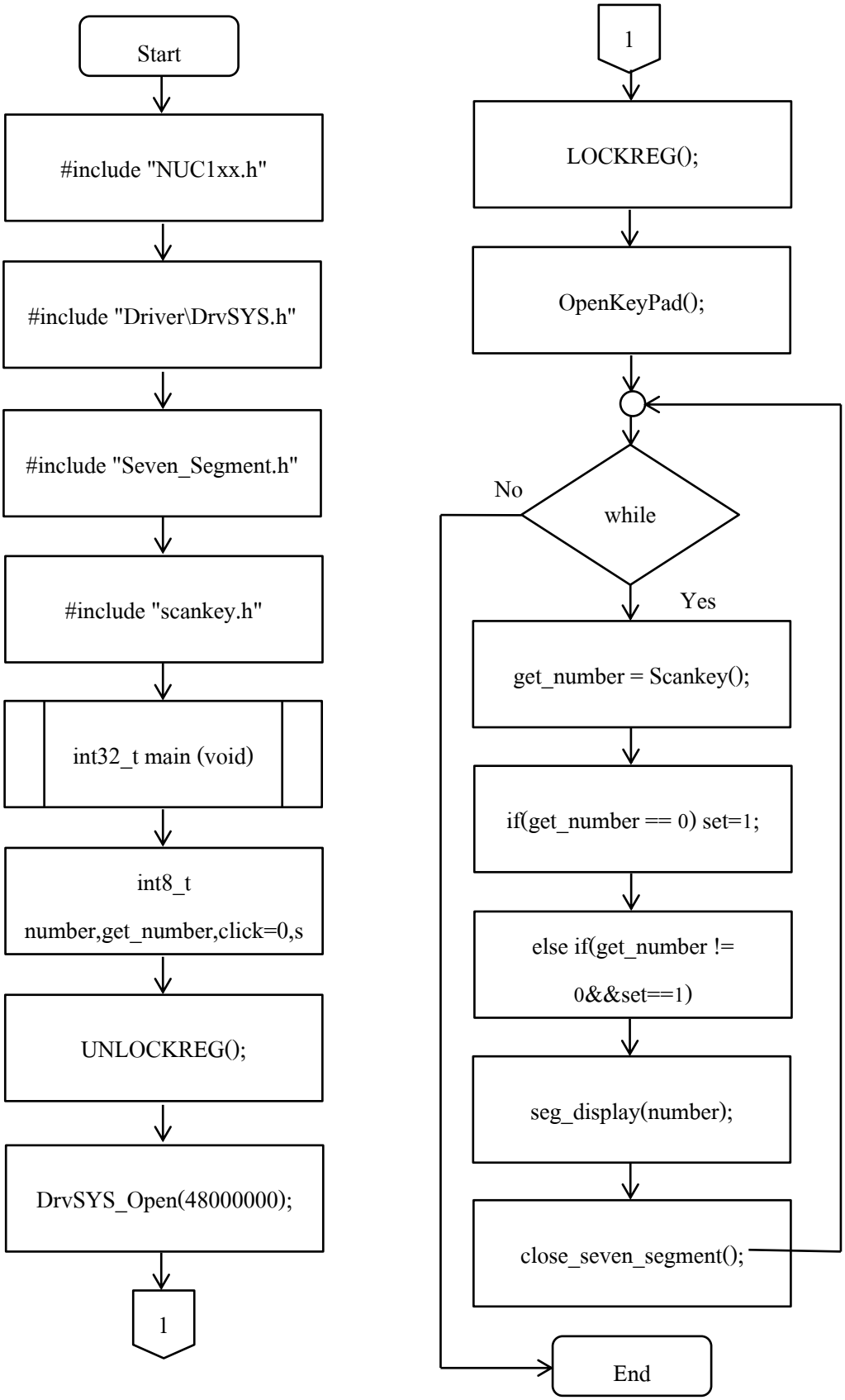
การทดลองการรับค่าจาก Matrix Switch ที่อยู่บนชุดทดลองนั้น จะมีขนาด 3x3 และมีปุ่มกดอยู่ 9 ปุ่มด้วยกันจากที่เขียนโปรแกรมจากตัวอย่างเสร็จแล้วก็เห็น 7-segment สว่างขึ้นมา จากนั้นลองกดปุ่มดูก็เห็นตัวเลขไปแสดงบน 7-Segment ที่อยู่บนชุดลอง ตามค่าค่าตัวเลขที่กดไป เช่นถ้ากดปุ่มตรงกลางของปุ่มทั้งหมดก็จะแสดงตัวเลข 5 และเมื่อลองกดปุ่มสุดท้ายก็จะแสดงตัวเลข 9 แทนตัวเลข 5 ที่เห็นอยู่ก่อนหน้านี้

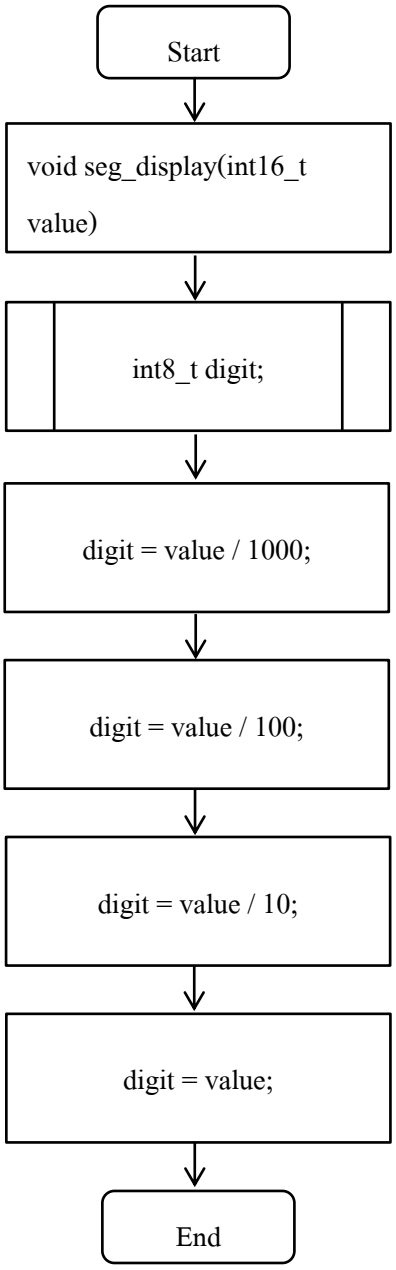
ให้นักศึกษาอธิบายโปรแกรม

โปรแกรมการรับค่าจาก Matrix Switch เริ่มต้นการ #include "NUC1xx.h" เป็นการดึงไดเวอร์ของชิป NUVOTON มาใช้งานเป็นค่าเริ่มต้มจากนั้น #include "Driver\DrvSYS.h" ดึงค่าการใช้งานของซิสเท็มไดเวอร์ออกมาเพื่อตั้งค่าส่วนต่างๆภายในไอซี ไอซี #include "Seven_Segment.h" เป็นการเซ็ตค่าขา 7-Segment ที่อยู่บนชุดทดลอง NUVOTON #include "scankey.h"เลือกรับค่าจาก Key Pad Switch ขนาด 3x3

ในชุดคำสั่ง main เริ่มต้น UNLOCKREG(); เป็นการปลดล็อกรีจิสเตอร์เพื่อเข้าไปเซ็ตค่าความเร็วของ CPU โดยเซ็ตค่าความเร็วในการทำงานไว้ที่ 48 MHz จากนั้นก็ทำการปิดรีจิสเตอร์ LOCKREG();เริ่มต้น get_number = Scankey(); เป็นการรอรับค่ามาจาก Key pad ถ้ากดปุ่ม Key pad ปุ่มใดปุ่มนึง seg_display(number); โปรแกรมจะดูว่าเรากดปุ่มไหนตัวเลขอะไรและจะทำการแสดงผลไปยัง 7-Segment ที่อยู่บนชุดทดลองโดยจะติดอยู่ตรงที่ 7-Segment บิตที่ 0 หรือบิตสุดท้าย

เฉลยใบงานที่ 4 Flow chart ของโปรแกรม




Start
void seg_display(int16_t value)
int8_t digit;
digit = value / 1000;
digit = value / 100;
digit = value / 10;
digit = value;
End

เฉลยใบงานที่ 4 โปรแกรม

```
#include "NUC1xx.h"
#include "DrvSYS.h"
#include "Seven_Segment.h"
#include "scankey.h"

void seg_display(int16_t value);
int32_t main (void)
{
	int8_t number,get_number,click=0,set=0;
	UNLOCKREG();
	DrvSYS_Open(48000000);
	LOCKREG();
	OpenKeyPad();
	while(1)
	{
		get_number = Scankey();
		if(get_number == 0) set=1;
		else if(get_number != 0&&set==1)
			{
				set=0;
				if(click==0)
					{
						number = 0;
						number = get_number;
						click++;
					}
```

```
                                else if(click==1)
                                        {
                                                number = (number*10)+get_number;
                                                click=0;
                                        }
                                }
                        seg_display(number);
                        close_seven_segment();
                }
}
void seg_display(int16_t value)
{
        int8_t digit;
                digit = value / 1000;
                close_seven_segment();
                show_seven_segment(3,digit);
                DrvSYS_Delay(5000);
                close_seven_segment();
                value = value - digit * 1000;
                digit = value / 100;
                close_seven_segment();
                show_seven_segment(2,digit);
                DrvSYS_Delay(5000);
                value = value - digit * 100;
                digit = value / 10;
                close_seven_segment();
                show_seven_segment(1,digit);
                DrvSYS_Delay(5000);
```

```
                value = value - digit * 10;
                digit = value;
                close_seven_segment();
                show_seven_segment(0,digit);
                DrvSYS_Delay(5000);
}
```

เฉลยใบงานที่ 5 การทดลองการแสดงผลออกทาง Buzzer
สังเกตผลการทดลองและบันทึกผลที่เกิดขึ้น

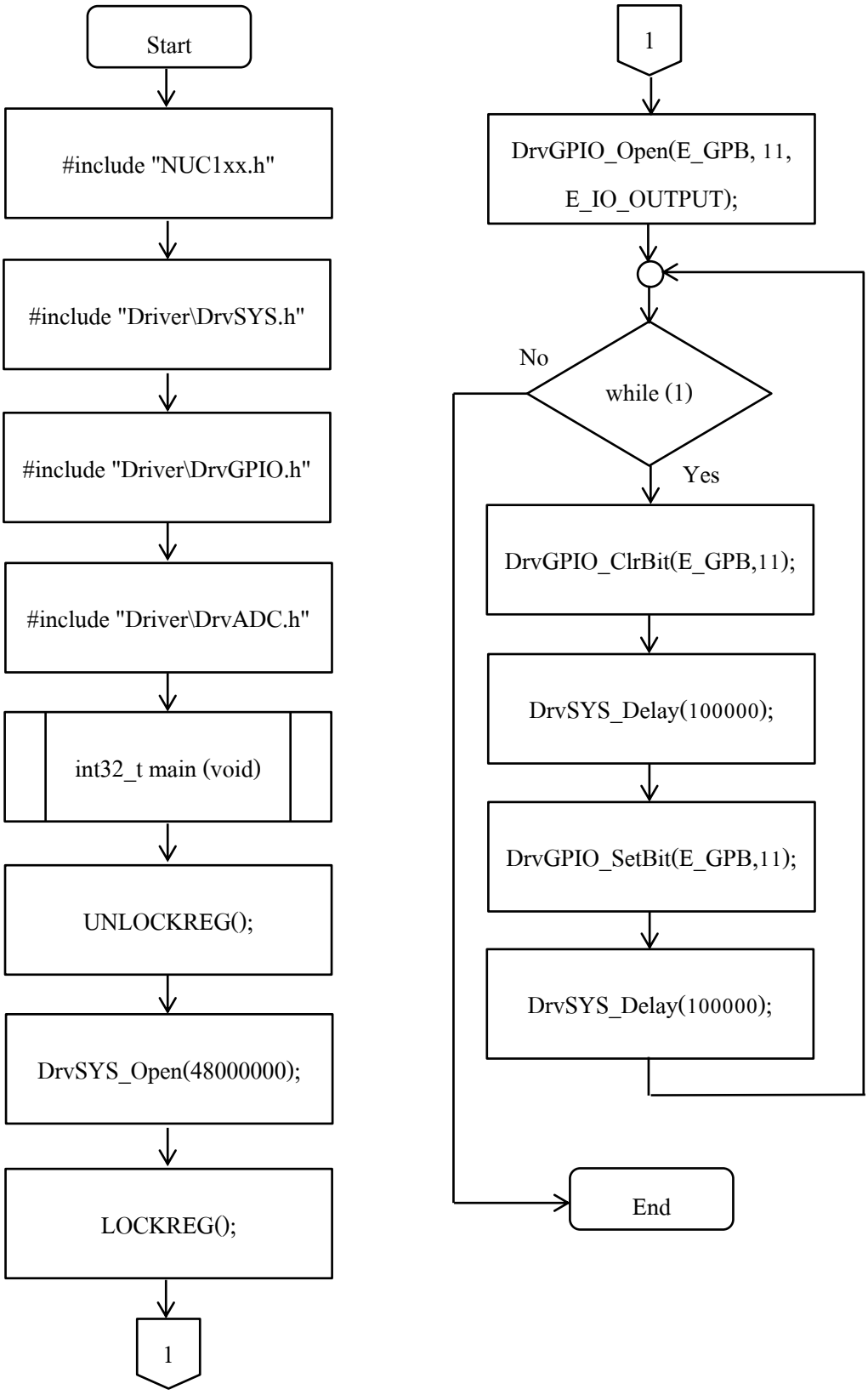
จากการทดลองการแสดงผลออกทาง Buzzer ในชุดลองจะมีตัว Buzzer ขนาดเล็กอยู่บนชุดทดลองด้วยเพื่อที่ง่ายและสะดวกในการส่งเสียงเตือน นากโปรแกรมที่ให้มานั้นแสดงให้เห็นการส่งเสียงของ Buzzer ที่อยู่บนชุดทดลอง จะมีการส่งเสียงบี๊บเป็นจังหวะตามที่โปรแกรมไว้และก็ส่งเสียงไปแบบนั้นไม่มีหยุดจนกว่าจะปิดแหล่งจ่ายไฟให้กับชุดลอง

ให้นักศึกษาอธิบายโปรแกรม

โปรแกรมการแสดงผลออกทาง Buzzer เริ่มต้นการ #include "NUC1xx.h" เป็นการดึงไดเวอร์ของชิป NUVOTON มาใช้งานเป็นค่าเริ่มต้มจากนั้น #include "Driver\DrvSYS.h" ดึงค่าการใช้งานของซิสเท็มไดเวอร์ออกมาเพื่อตั้งค่าส่วนต่างๆภายในไอซี #include "Driver\DrvGPIO.h"เป็นเซ็ตค่าขาการใช้งานที่อยู่บนชุดทดลอง NUVOTON จากนั้นเลือกเซ็ตค่าการใช้งาน #include "Driver\DrvADC.h" เพื่อดึงไลบราลี่ของการแปลงอนาล็อกเป็นดิจิตอล

ในชุดคำสั่ง main เริ่มต้น UNLOCKREG(); เป็นการปลดล็อกรีจิสเตอร์เพื่อเข้าไปเซ็ตค่าความเร็วของ CPU โดยเซ็ตค่าความเร็วในการทำงานไว้ที่ 48 MHz จากนั้นก็ทำการปิดรีจิสเตอร์ LOCKREG();เซ็ตลอจิก DrvGPIO_ClrBit(E_GPB,11); เลือกขาใช้งาน GPB 11 เป็นบิต 0 เพื่อให้ BUZZER ทำงานและหน่วงเวลาการส่งเสียงไว้ 1 วินาที จากนั้นก็เลือกเคลียบิตขานั้น DrvGPIO_SetBit(E_GPB,11); เพื่อเลือกปิดการทำงานส่งเสียงของ BUZZER และหน่วงเวลา 1 วินาทีจากนั้นก็จะทำงานวนต่อไปในลูป while

เฉลยใบงานที่ 5 Flow chart ของโปรแกรม



เฉลยใบงานที่ 5 โปรแกรม

```
#include "NUC1xx.h"
#include "Driver\DrvSYS.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvADC.h"

int main (void)
{
        UNLOCKREG();
        DrvSYS_Open(48000000);
        LOCKREG();
        DrvGPIO_Open(E_GPB, 11, E_IO_OUTPUT);
        while(1)
        {
        DrvGPIO_ClrBit(E_GPB,11);
        DrvSYS_Delay(42000000);
        DrvGPIO_SetBit(E_GPB,11);
        DrvSYS_Delay(42000000);
        }
}
```

เฉลยใบงานที่ 6 การทดลองการรับค่าจาก Variable Resistance
สังเกตผลการทดลองและบันทึกผลที่เกิดขึ้น

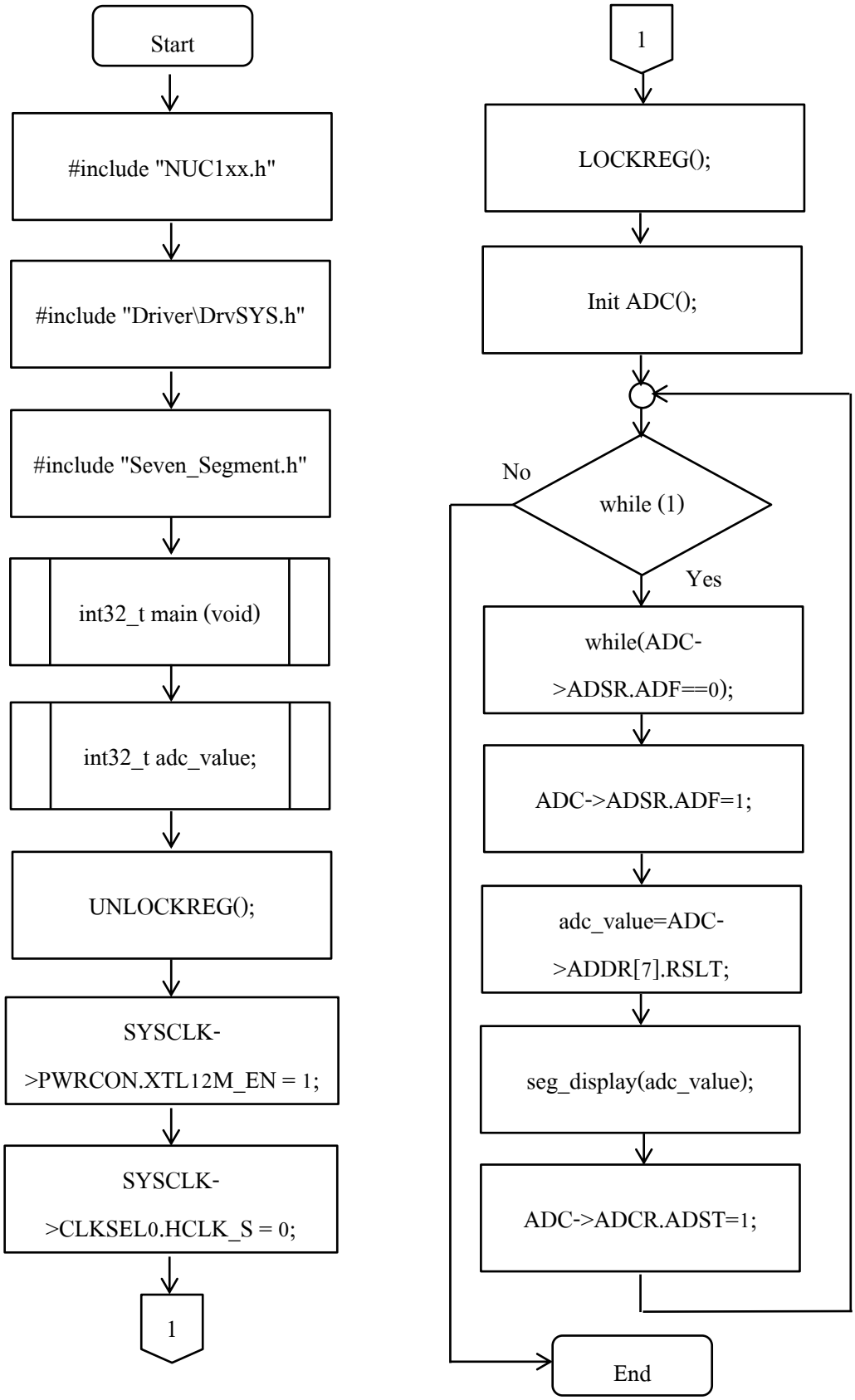
จากการทดลองการรับค่าจาก Variable Resistance ที่มีอยู่บนชุดทดลองนั้น ก็จะเป็นตัวต้านทานแบบปรับค่าได้ ที่จะแปลงค่าอนาล็อกเป็นดิจิตอลและจากโปรแกรมที่ให้มาแสดงให้เห็นการปรับค่าเพื่อไปแสดงผลบน 7-Segment จากที่ลองหมุนปรับค่าเริ่มต้นจาก 0 และปรับขึ้นไปเรื่อยๆจนสุดจะแสดงค่า 4095 จะเห็นว่าเป็นค่าสุดท้ายที่แสดงผล

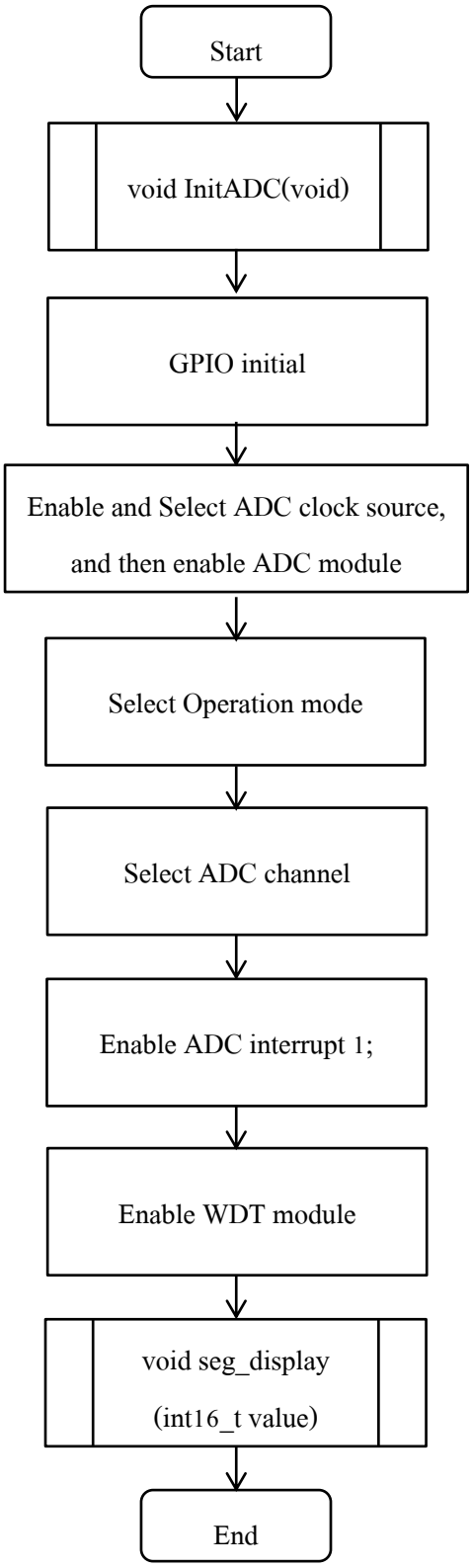
ให้นักศึกษาอธิบายโปรแกรม

โปรแกรมการรับค่าจาก Variable Resistance เริ่มต้นการ #include "NUC1xx.h" เป็นการดึงไดเวอร์ของชิป NUVOTON มาใช้งานเป็นค่าเริ่มต้มจากนั้น #include "Driver\DrvSYS.h" ดึงค่าการใช้งานของซิสเท็มไดเวอร์ออกมาเพื่อตั้งค่าส่วนต่างๆภายในไอซี #include "Seven_Segment.h" เป็นการเซ็ตค่าขา 7-Segment ที่อยู่บนชุดทดลอง NUVOTON

ในชุดคำสั่ง main เริ่มต้น UNLOCKREG(); เป็นการปลดล็อกรีจิสเตอร์เพื่อเข้าไปเซ็ตค่าความเร็วของ CPU โดยเซ็ตค่าความเร็วในการทำงานไว้ที่ 48 MHz จากนั้นก็ทำการปิดรีจิสเตอร์ LOCKREG(); รับค่า ADC จากขา GPA 7 มาเก็บไว้ที่ adc_value แล้วนำมาแสดงผลที่ 7-Segment ซึ่งค่า ADC จะมีขนาด 10 บิตจะแสดงค่าได้จาก 0 ถึง 4095

เฉลยใบงานที่ 6 Flow chart ของโปรแกรม




Start
void InitADC(void)
GPIO initial
Enable and Select ADC clock source,
and then enable ADC module
Select Operation mode
Select ADC channel
Enable ADC interrupt 1;
Enable WDT module
void seg_display
(int16_t value)
End

เฉลยใบงานที่ 6 โปรแกรม

```
#include "NUC1xx.h"
#include "DrvSYS.h"
#include "NUC1xx-LB_002\LCD_Driver.h"

void InitADC(void);

int32_t main (void)
{
        char TEXT1[16]="ADC Value:      ";
        UNLOCKREG();
        SYSCLK->PWRCON.XTL12M_EN = 1;
        SYSCLK->CLKSEL0.HCLK_S = 0;
        LOCKREG();
        InitADC();
        Initial_panel();
        clr_all_panel();
        print_lcd(0, "Smpl_ADC_VR1");

        while(1)
        {
                while(ADC->ADSR.ADF==0);
                ADC->ADSR.ADF=1;
                sprintf(TEXT1+10,"%4d",ADC->ADDR[7].RSLT);
                print_lcd(1, TEXT1);
                DrvSYS_Delay(20000);
                ADC->ADCR.ADST=1;
```

```
        }
}

void InitADC(void)
{
        GPIOA->OFFD|=0x00800000;
        SYS->GPAMFP.ADC7_SS21_AD6=1;
        SYSCLK->CLKSEL1.ADC_S = 2;
        SYSCLK->CLKDIV.ADC_N = 1;
        SYSCLK->APBCLK.ADC_EN = 1;
        ADC->ADCR.ADEN = 1;
        ADC->ADCR.DIFFEN = 0;
        ADC->ADCR.ADMD   = 0;
        ADC->ADCHER.CHEN = 0x80;
        ADC->ADSR.ADF =1;
        ADC->ADCR.ADIE = 1;
        ADC->ADCR.ADST=1;
}
```

เฉลยใบงานที่ 7 การทดลองร่วมกับชุดทดลอง ETT ควบคุม DC Motor
สังเกตผลการทดลองและบันทึกผลที่เกิดขึ้น

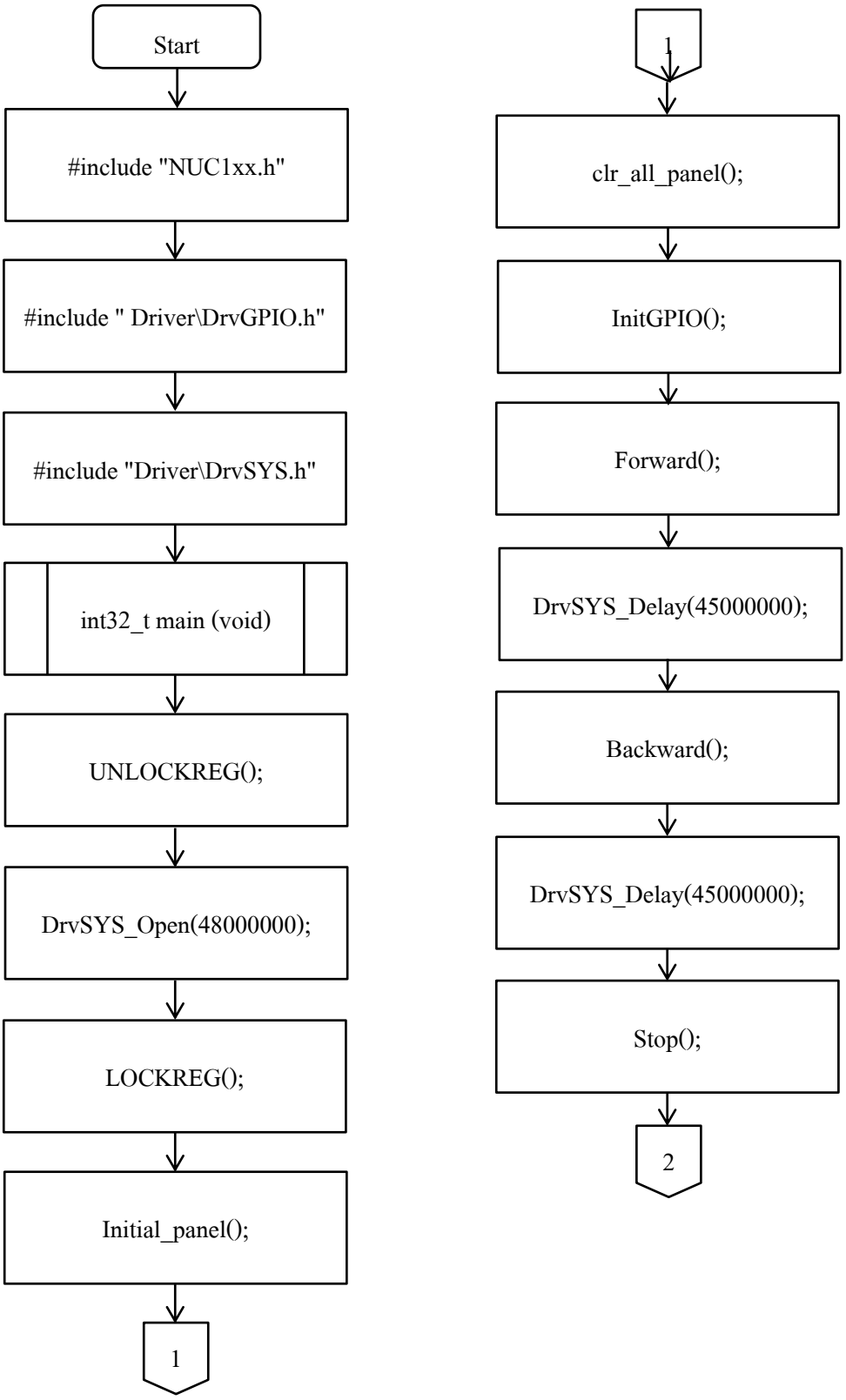
จากการทดลองร่วมกับชุดทดลอง ETT ควบคุม DC Motor โดยสั่งโปรแกรมจากชุดทดลอง NUVOTON NUC140 ไปยังชุดลอง ETT เพื่อควบคุมมอเตอร์ จากโปรแกรมตัวอย่างแสดงให้เห็นถึงการหมุนของมอเตอร์ที่หมุนไปทางขวาและทางซ้ายสลับกันไปเร็วมากจนตาเปล่ามองไม่ทันแต่ยังมีเซ็นเซอร์ที่อยู่บนชุดทดลองช่วยให้เห็นทิศทางของการหมุนในตอนเริ่มต้นอยู่

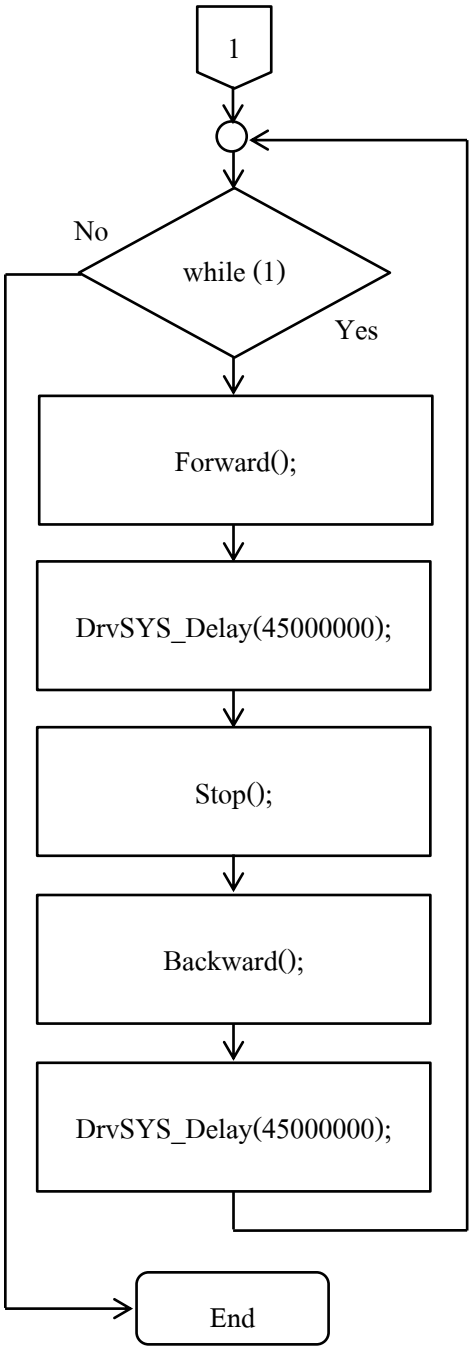
ให้นักศึกษาอธิบายโปรแกรม

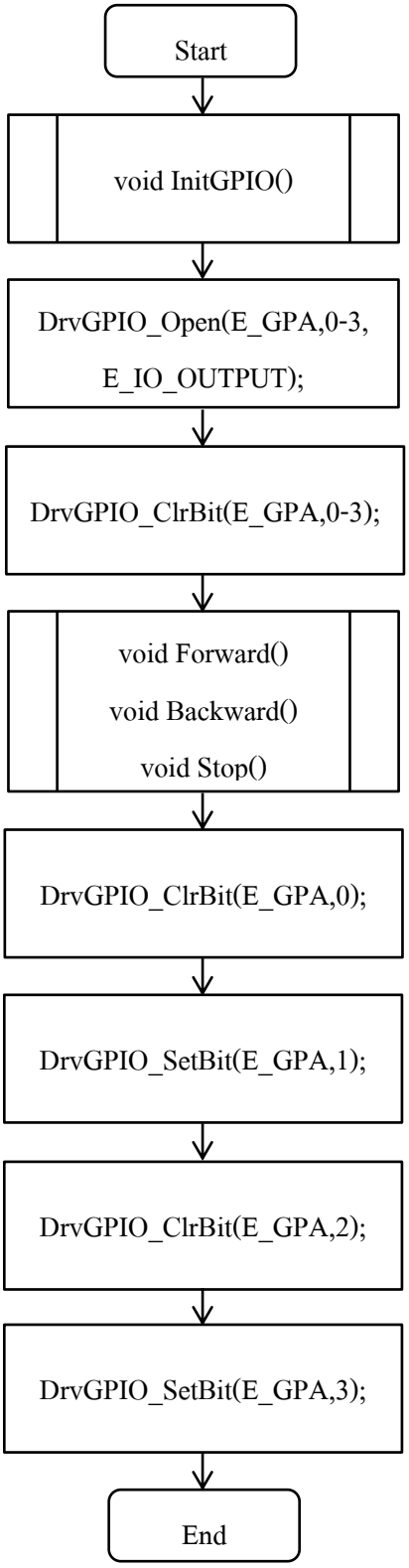
โปรแกรมการทดลองร่วมกับชุดทดลอง ETT ควบคุม DC Motor เริ่มต้นการ #include "NUC1xx.h" เป็นการดึงไดเวอร์ของชิป NUVOTON มาใช้งานเป็นค่าเริ่มต้มจากนั้น #include "Driver\DrvSYS.h" ดึงค่าการใช้งานของซิสเท็มไดเวอร์ออกมาเพื่อตั้งค่าส่วนต่างๆภายในไอซี #include "Driver\DrvGPIO.h"เป็นเซ็ตค่าขาการใช้งานที่อยู่บนชุดทดลอง NUVOTON

ในชุดคำสั่ง main เริ่มต้น UNLOCKREG(); เป็นการปลดล็อกรีจิสเตอร์เพื่อเข้าไปเซ็ตค่าความเร็วของ CPU โดยเซ็ตค่าความเร็วในการทำงานไว้ที่ 48 MHz จากนั้นก็ทำการปิดรีจิสเตอร์ LOCKREG(); จากนั้นสั่งให้มอเตอร์เดินหน้า DrvGPIO_SetBit(E_GPA,0); เป็นการเคลียบิต โดยให้ GPA 0 ทำงานโดยให้มอเตอร์หมุนไปทางด้านหน้า DrvGPIO_ClrBit(E_GPA,1);เป็นการเซ็ตค่าให้มอเตอร์หยุดการทำงาน DrvGPIO_ClrBit(E_GPA,0); เป็นการสั่งให้มอเตอร์ทำงานในหารหมุนถอยหลัง DrvGPIO_SetBit(E_GPA,1); สั่งให้เคลียบิตการทำงานให้มอเตอร์หยุด DrvGPIO_ClrBit(E_GPA,0); DrvGPIO_ClrBit(E_GPA,1); เป็นคำสั่งให้มอเตอร์หยุดการทำงานทั้งโปรแกรม

เฉลยใบงานที่ 7 Flow chart




1
No
while (1)
Yes
Forward();
DrvSYS_Delay(45000000);
Stop();
Backward();
DrvSYS_Delay(45000000);
End


Start
void InitGPIO()
DrvGPIO_Open(E_GPA,0-3,
E_IO_OUTPUT);
DrvGPIO_ClrBit(E_GPA,0-3);
void Forward()
void Backward()
void Stop()
DrvGPIO_ClrBit(E_GPA,0);
DrvGPIO_SetBit(E_GPA,1);
DrvGPIO_ClrBit(E_GPA,2);
DrvGPIO_SetBit(E_GPA,3);
End

เฉลยใบงานที่ 7 โปรแกรม

```
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvSYS.h"
#include "NUC1xx-LB_002\LCD_Driver.h"

void InitGPIO();
void Forward();
void Backward();
void Stop();

int main (void)
{
	UNLOCKREG();
	DrvSYS_Open(48000000);
	LOCKREG();
	Initial_panel();
	clr_all_panel();
	InitGPIO();
	Forward();
	DrvSYS_Delay(45000000);
	Backward();
	DrvSYS_Delay(45000000);
	Stop();
	while(1)
	{
		Forward();
```

```
                DrvSYS_Delay(45000000);
                Forward();
                DrvSYS_Delay(45000000);
                Forward();
                DrvSYS_Delay(45000000);
                Stop();
                DrvSYS_Delay(45000000);
                Stop();
                DrvSYS_Delay(45000000);
                Backward();
                DrvSYS_Delay(45000000);
                Backward();
                DrvSYS_Delay(45000000);
                Stop();
                DrvSYS_Delay(45000000);
                Stop();
                DrvSYS_Delay(45000000);
        }
}
void InitGPIO()
{
        DrvGPIO_Open(E_GPA,0,E_IO_OUTPUT);
        DrvGPIO_Open(E_GPA,1,E_IO_OUTPUT);
        DrvGPIO_Open(E_GPA,2,E_IO_OUTPUT);
        DrvGPIO_Open(E_GPA,3,E_IO_OUTPUT);
        DrvGPIO_ClrBit(E_GPA,0);
        DrvGPIO_ClrBit(E_GPA,1);
        DrvGPIO_ClrBit(E_GPA,2);
```

```
        DrvGPIO_ClrBit(E_GPA,3);
}
void Forward()
{
        DrvGPIO_SetBit(E_GPA,0);
        DrvGPIO_ClrBit(E_GPA,1);
        DrvGPIO_SetBit(E_GPA,2);
        DrvGPIO_ClrBit(E_GPA,3);
}
void Backward()
{
        DrvGPIO_ClrBit(E_GPA,0);
        DrvGPIO_SetBit(E_GPA,1);
        DrvGPIO_ClrBit(E_GPA,2);
        DrvGPIO_SetBit(E_GPA,3);
}
void Stop()
{
        DrvGPIO_ClrBit(E_GPA,0);
        DrvGPIO_ClrBit(E_GPA,1);
        DrvGPIO_ClrBit(E_GPA,2);
        DrvGPIO_ClrBit(E_GPA,3);
}
```

เฉลยใบงานที่ 8 การทดลองร่วมกับชุดทดลอง ETT ควบคุม Stepper Motor
สังเกตผลการทดลองและบันทึกผลที่เกิดขึ้น

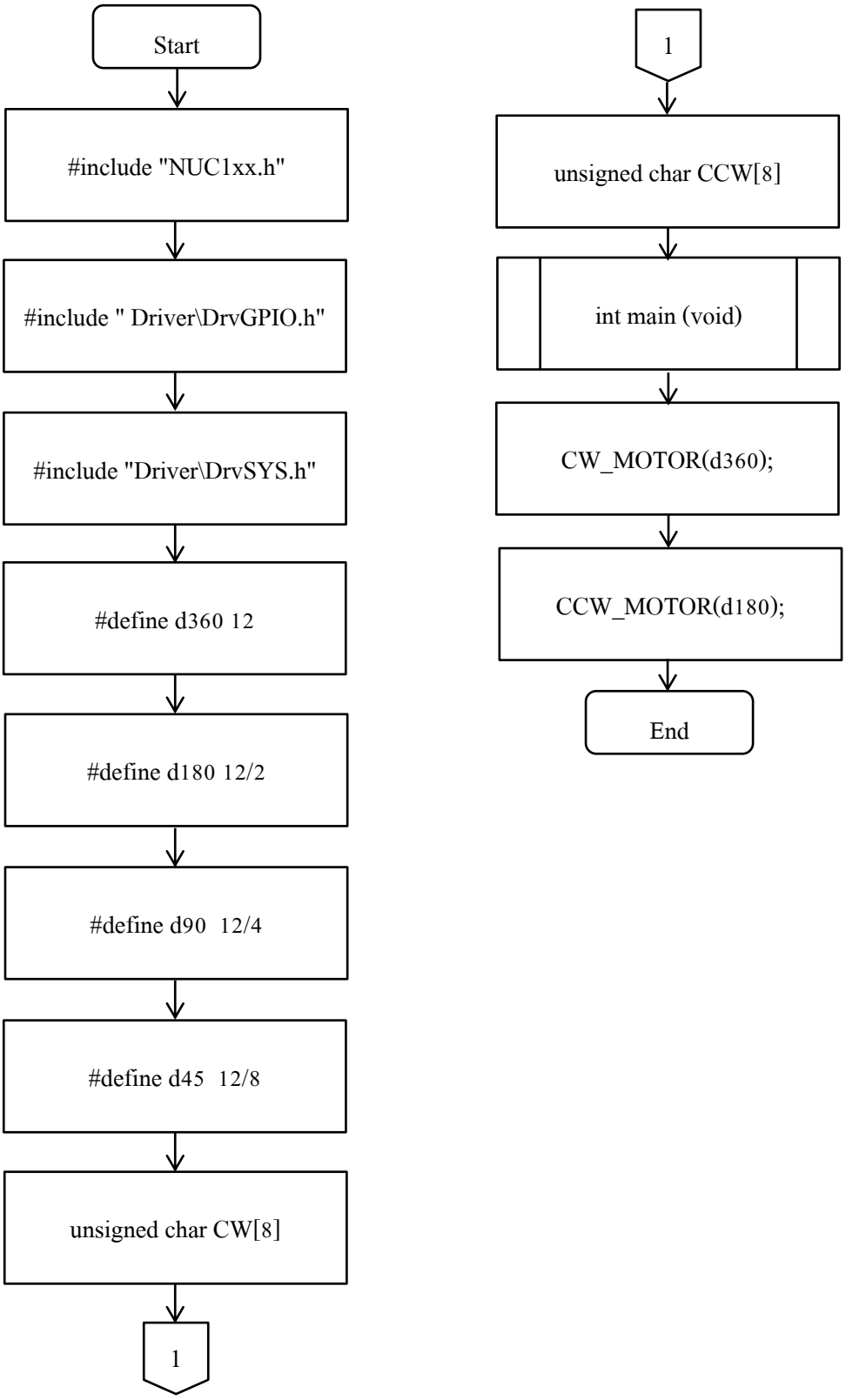
จากการทดลองร่วมกับชุดทดลอง ETT ควบคุม Stepper Motor โดยการสั่งโปรแกรมผ่านทางชุดทดลอง NUVOTON NUC140 เพื่อไปควบคุมการทำงานของ Stepper Motor โดยให้ Stepper Motor ทำงานตามโปรแกรมที่สั่งไว้จากตัวอย่างโปรแกรมที่ให้มานั้นจะสังเกตได้ว่า Stepper Motor จะหมุนตามเข็มนาฬิกาไปทีละสเต็ปจนครบรอบ 360 องศาตามที่ตั้งโปรแกรมไว้จากนั้น Stepper Motor จะหมุนกลับทางเป็นการทวนเข็มนาฬิกาไปทีละสเต็ปดังเดิมจนครบ 180 องศาตามที่ตั้งไว้ในโปรแกรมและจะหยุดการทำงานลง

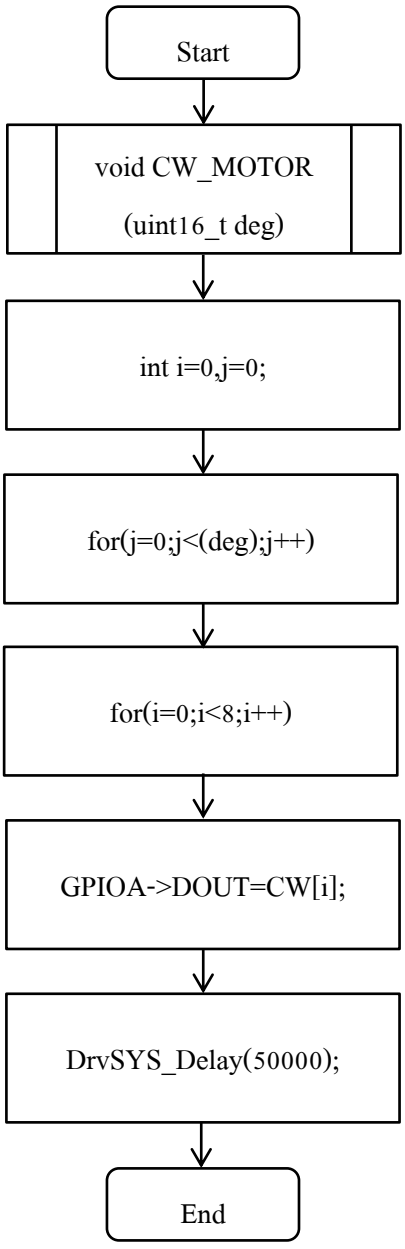
ให้นักศึกษาอธิบายโปรแกรม

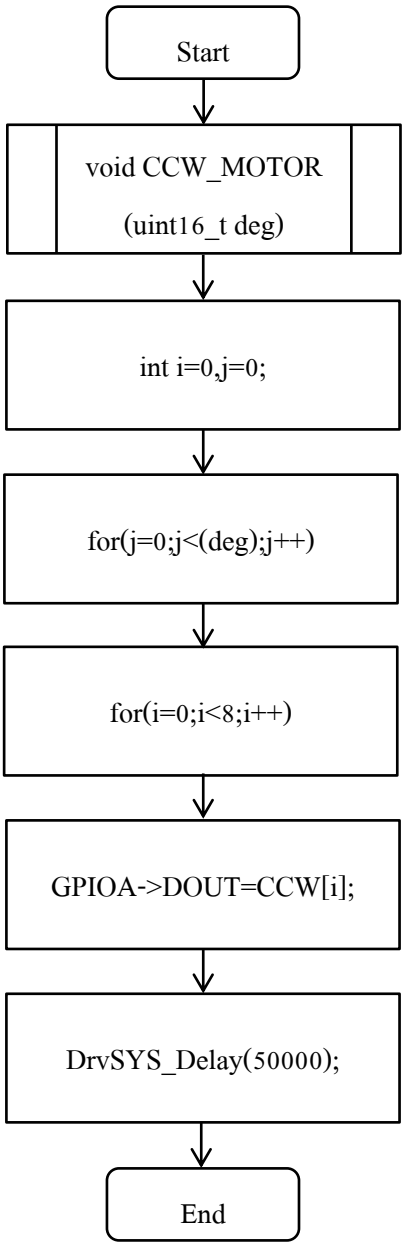
โปรแกรมการทดลองร่วมกับชุดทดลอง ETT ควบคุม Stepper Motor เริ่มต้นการ #include "NUC1xx.h" เป็นการดึงไดเวอร์ของชิป NUVOTON มาใช้งานเป็นค่าเริ่มต้มจากนั้น #include "Driver\DrvSYS.h" ดึงค่าการใช้งานของซิสเท็มไดเวอร์ออกมาเพื่อตั้งค่าส่วนต่างๆภายในไอซี #include "Driver\DrvGPIO.h"เป็นเซ็ตค่าขาการใช้งานที่อยู่บนชุดทดลอง NUVOTON

ในชุดคำสั่ง main เริ่มต้น UNLOCKREG(); เป็นการปลดล็อกรีจิสเตอร์เพื่อเข้าไปเซ็ตค่าความเร็วของ CPU โดยเซ็ตค่าความเร็วในการทำงานไว้ที่ 48 MHz จากนั้นก็ทำการปิดรีจิสเตอร์ LOCKREG(); CW_MOTOR(d360); เป็นการสั่งให้ Step Motor เดินหน้าตามเข็มนาฬิกาหนึ่งรอบ 360 องศา CCW_MOTOR(d180); เป็นคำสั่งให้ Step Motor ถอยหลังทวนเข็มนาฬิกามา 180 องศา โดยที่ Step Motor เดินหน้ามีความละเอียดในการหมุนทีละ Step ดังนี้ CW[8] = {0x09,0x01,0x03,0x02,0x06,0x04,0x0c,0x08}; และ Step การเดินถอยหลัง CCW[8]= {0x08,0x0c,0x04,0x06,0x02,0x03,0x01,0x09}; เมื่อเริ่มทำงานโปรแกรมจะสั่งให้หมุนไปทีละ Step

เฉลยใบงานที่ 8 Flow chart




Start
void CW_MOTOR
(uint16_t deg)
int i=0,j=0;
for(j=0;j<(deg);j++)
for(i=0;i<8;i++)
GPIOA->DOUT=CW[i];
DrvSYS_Delay(50000);
End


Start
void CCW_MOTOR
(uint16_t deg)
int i=0,j=0;
for(j=0;j<(deg);j++)
for(i=0;i<8;i++)
GPIOA->DOUT=CCW[i];
DrvSYS_Delay(50000);
End

เฉลยใบงานที่ 8 โปรแกรม

```
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvSYS.h"

#define d360 12
#define d180 12/2
#define d90  12/4
#define d45  12/8

unsigned char CW[8] ={0x09,0x01,0x03,0x02,0x06,0x04,0x0c,0x08};
unsigned char CCW[8]={0x08,0x0c,0x04,0x06,0x02,0x03,0x01,0x09};

void CW_MOTOR(uint16_t deg);
void CCW_MOTOR(uint16_t deg);

int main (void)
{
        CW_MOTOR(d90);
        CCW_MOTOR(d180);
        CW_MOTOR(d360);
}

void CW_MOTOR(uint16_t deg)
{
        int i=0,j=0;
        for(j=0;j<(deg);j++)
```

```
        {
        for(i=0;i<8;i++)
                {
                GPIOA->DOUT=CW[i];
                DrvSYS_Delay(50000);
                }
        }
}
void CCW_MOTOR(uint16_t deg)
{
        int i=0,j=0;
        for(j=0;j<(deg);j++)
        {
                for(i=0;i<8;i++)
                        {
                        GPIOA->DOUT=CCW[i];
                        DrvSYS_Delay(50000);
                        }
        }
}
```

บทที่ 5
บทสรุปและข้อเสนอแนะ

จากการศึกษาการทดลองที่ 1 ทำให้ทราบถึงการแสดงผลบนหลอด LED ในชุดทดลองและขาการใช้งาน LED ที่อยู่บนชุดลอง ว่าจะสั่งงานแบบไหนได้บ้าง และอยากให้นำไปพัฒนาต่อยอดไปใช้กับชุดขับ LED ที่อยู่บนชุดทดลอง ETT LAB3A และตัวอย่าง LED รูปแบบประเภทอื่นๆและควรนำข้อมูลที่ได้มาเผยแพร่ต่อไป

การทดลองที่ 2 การศึกษา 7 Segment หรือ LED แสดงผลตัวเลขแบบ 7 ส่วน ทำให้ทราบถึงการที่จะนำไปใช้งานประยุกต์เข้าอุปกรณ์แสดงผลแบบดิจิตอลได้ที่แสดงผลในรูปแบบตัวเลข ทำให้ง่ายที่จะนำไปพัฒนาต่อยอดการใช้งานในการแสดงผลแบบหลายหลักพร้อมๆกันได้

การทดลองที่ 3 การใช้งาน LCD (Graphic LCD) ทำให้ทราบถึงข้อมูลของจอ LCD ที่อยู่บนชุดทดลอง ว่ามีขนาดกี่ตัวอักษร กี่บรรทัดและสามารถแสดงผลเป็นแอนิเมชั่นได้เพียงแค่นำไปพัฒนาต่อยอดว่าจอแสดงผลการทำงานอย่างไรกี่ดอท และแสดงผลแบบไหนจากข้อมูลที่อยู่แล้วเพื่อที่จะได้มีตัวอย่างที่หลากหลายขึ้นจากคู่มือเล่มนี้

การทดลองที่ 4 การใช้งาน Key Pad Switch เป็นศึกษาการรับค่าจาก Key Pad Switch ที่อยู่บนชุดทดลองที่มีขนาด 3x3 โดยในการทดลองให้ทำการแสดงตัวเลข 1-9 เมื่อรับค่าจาก Key Pad Switch และไปแสดงผลบนชุดทดลอง 7 Segment และอยากให้นำไปศึกษาต่อเพื่อแสดงผลบนจอ LCD หรือการทำฟังก์ชั่นอื่นที่รับค่าจาก Key Pad Switch

การทดลองที่ 5 การแสดงผลออกทาง BUZZER ในตัวอย่างการทดลองจะแสดงให้เป็นถึงการทำงานของ BUZZER แบบเบื้องต้นเพียงแค่ส่งเสียงเตือนเมื่อโปรแกรมทำงาน เนื่องจากการศึกษาในโปรแกรมชุดนี้มีเวลาที่จำกัดจากหลายๆอย่างจึงอยากให้ผู้ที่สนในนำไปพัฒนาต่อยอดเพื่อเป็นอุปกรณ์เพื่อนภัยหรืออุปกรณ์อื่นที่ใช้ BUZZER เข้ามาแสดงผล

การทดลองที่ 6 การใช้งาน Variable Resistance เป็นการทดลองรับค่าอนาล็อกเข้ามาแสดงผลยังชุดทลองโดยที่ชุดทดลองจะรับค่าจาก 0 Vถึง 5 Vและแปลงค่าแสดงได้ละเอียดถึง 10 บิต 4095 ค่า เหมาะที่จะนำไปทำการเก็บค่าข้อมูลรูปแบบต่างๆเพื่อเปรียบเทียบตารางแสดงผลการทดสอบในชุดโปรแกรมที่จะพัฒนาในภายภาคหน้า

การทดลองที่ 7 การต่อใช้งานร่วมกับชุดทดลอง ETT LAB3A ในการทดลองนี้เป็นการทดลองควบคุม DC motor ที่อยู่ในชุดทดลองของ ETT ที่มีอยู่ในห้องปฏิบัติการไมโครคอลโทรล

เลอร์เพื่อทดสอบโปรแกรมคุม DC motor สั่งการได้หรือไม่โดยให้ทำการเดินหน้าและถอยหลังจากโปรแกรมเบื้องต้นสามารถนำไปประยุกต์ใช้กับรถเดินตามเส้นโดยใช้ไมโครคอนโทรลเลอร์ควบคุมได้อีกด้วย

การทดลองที่ 8 การต่อใช้งานร่วมกับชุดทดลอง ETT LAB3A เพื่อใช้ในการควบคุม Stepping Motor จากการทดลองควบคุม Stepping Motor ทำได้ทราบว่าเมื่อสั่งให้เริ่มทำงานตามจังหวะเดินหน้า 360 องศาและถอยหลัง 180 องศานั้น Stepping Motor จะทำงานขาดไปจังหวะนึงซึ่งอาจจะมีปัญหาหากนำไปใช้กับงานที่ต้องการความละเอียดอย่างมาก แต่เพื่อการศึกษาโปรแกรมการควบคุมก็จะไม่เป็นปัญหาในหารใช้งาน Stepping Motor

จากการศึกษาชุดทดลอง NUVOTON NUC140 ได้ทราบข้อมูลต่างๆที่มีอยู่ในชุดทดลองที่นอกเหนือจากที่เคยทราบมาจากการศึกษาไมโครคอนโทรลเลอร์ขนาด 8 บิต ที่มีอยู่ในแลปการทดลอง ได้ศึกษาโปรแกรม Keil Uvision4 ที่ใช้กับชุดทดลอง NUVOTON ตั้งแต่การตั้งค่าไปจนถึงการแอดไลบรารี่เข้าไปในโปรแกรมซึ่งการแอดไลบรารี่นี้มีความยุ่งยากและซับซ้อน ทำให้มีปัญหาบ้างในการทำแลปการทดลองขึ้นมา เนื่องจากไลบรารี่บางส่วนติดดีบัค ในการเขียนโปรแกรมก็จะไม่ค่อยมีปัญหาสามารถนำโปรแกรมมาประยุกต์ใช้งานกันได้

เอกสารอ้างอิง

[1] Cortex-M0 Processor

http://www.arm.com/products/processors/cortex-m/cortex-m0.php

[2] เอกสารคำสอนวิชาสถาปัตยกรรมคอมพิวเตอร์ ผศ.ดร.สุรินทร์ กิตติธรกุล

www.ce.kmitl.ac.th/download.php?DOWNLOAD_ID=2990

[3] Nuvoton Wikipedia

http://www.nuvoton.com/NuvotonMOSS/Community/ProductInfo

[4] นคร ภักดีชาติ. ปฏิบัติการไมโครคอนโทรลเลอร์ ARM Cortex-M3 กับ STM 32. กรุงเทพฯ : บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด

[5] ผศ. ชาญวิทย์. ตั้งสิริวรกุล. (2555). การศึกษาและทดลอง ARM Fujitsu ด้วยภาษาซี. กรุงเทพฯ : ห้างหุ้นส่วนจำกัด วี เจ พริ้นติ้ง

[6] โอภาส ศิริครรชิตถาวร. (2553). เปิดโลก 32 บิตกับ STM32-Discovery. กรุงเทพฯ. ห้างหุ้นส่วนจำกัด วี เจ พริ้นติ้ง

[7] Keil uVision4

www.keil.com

[8] การทำงานของหลอดแสดงผล LED (Light Emitting Diodes)

www.technicchan.ac.th/~web/planteach/14022552aUE.doc

[9] GFG128064A-FPFA liquid crystal module

http://www.datamate-j.com/wp-content/uploads/2013/06/GFG128064A-FPFAB001SPEC.pdf

[10] ปฏิบัติการไมโครโปรเซสเซอร์และการออกแบบไมโครคอมพิวเตอร์ การเชื่อมต่อ Matrix Keyboard & Switch
http://course.eau.ac.th/course/Download/00137619/LabSheet_%E0%B8%9B%E0%B8%8F%E0%B8%B4%E0%B8%9A%E0%B8%B1%E0%B8%95%E0%B8%B4%E0%B9%84%E0%B8%A1%E0%B9%82%E0%B8%84%E0%B8%A3%E0%B9%82%E0%B8%9B%E0%B8%A3%E0%B9%80%E0%B8%8B%E0%B8%AA%E0%B9%80%E0%B8%8B%E0%B8%AD%E0%B8%A3%E0%B9%8C04.pdf

[11] เรียนรู้ ADC Architecture แบบต่างๆ
http://www.thaiembedded.com/blog/?cat=38

[12] ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์
http://www.cpe.ku.ac.th/~yuen/204471/conversion/dac/

[13] สเตปปิ้งมอเตอร์ (Stepping Motor)โดย Adisak chinawong
http://www.adisak51.com/page22.html

ประวัติส่วนตัว

ชื่อ นายทิวากร เคี่ยมขาว

ข้อมูลส่วนตัว

วัน เดือน ปีเกิด 8 ตุลาคม 2534 **อายุ** 22ปี **ส่วนสูง** 185

เชื้อชาติ ไทย **สัญชาติ** ไทย **ศาสนา** พุทธ

ที่อยู่ 93/194 อาคารลุมพินีวิลล์ หมู่ 9 ตำบลบางเขน อำเภอ เมืองนนทบุรี จังหวัด นนทบุรี 11000

โทรศัพท์ 0899862167

E-mail. Thiwagon_02@hotmail.com thiwagon02@gmail.com

ประวัติการศึกษา

ระดับมัธยมศึกษา โรงเรียนทุ่งสง **ปีที่จบ** 2550

ระดับประกาศนียบัตรวิชาชีพ วิทยาลัยเทคนิคทุ่งสง **สาขา** อิเล็กทรอนิกส์ **ปีที่จบ** 2553

ระดับปริญญาตรี มหาวิทยาลัยศรีปทุม **คณะ**วิศวกรรมศาสตร์ **สาขา** ไฟฟ้าและอิเล็กทรอนิกส์ประยุกต์

ปีที่จบ กำลังศึกษา

กิจกรรมที่เคยเข้าร่วม

ได้เข้าร่วมการขันทักษะวิชาชีพ ประเภท ช่างอิเล็กทรอนิกส์ หุ่นยนต์เล็ก ปีที่เข้าร่วม 2551

เข้าร่วมการแข่งขันหุ่นยนต์อาชีวศึกษาชิงชนะเลิศระดับภาค(ABU Robocon)ปีที่เข้าร่วม 2552

ได้เข้าร่วมการขันทักษะวิชาชีพ ประเภท ช่างอิเล็กทรอนิกส์ หุ่นยนต์เล็ก ปีที่เข้าร่วม 2552

เข้าร่วมการแข่งขันหุ่นยนต์อาชีวศึกษาชิงชนะเลิศระดับภาค(ABU Robocon)ปีที่เข้าร่วม 2553

ได้เข้าร่วมโครงการ Go Green In The City 2013 ของบริษัท Schneider ได้รับรางวัลชมเชย ปีที่เข้าร่วม 2556