รายงานสหกิจศึกษาฉบับสมบูรณ์

บอร์ดทดสอบการทำงานของไอซี
Application Bench Board

นางสาวภัทรภร ศรีบูรพา

ภาควิชาอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2560

รายงานสหกิจศึกษาฉบับสมบูรณ์

บอร์ดทดสอบการทำงานของไอซี

Application Bench Board

นางสาวภัทรภร ศรีบูรพา

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2560

ชื่อโครงงานสหกิจศึกษา บอร์ดทดสอบการทำงานของไอซี

ชื่อ-สกุล นักศึกษา นางสาวภัทรภร ศรีบูรพา

คณะ วิศวกรรมศาสตร์ **ภาควิชา** อิเล็กทรอนิกส์

ชื่อ-สกุล อาจารย์นิเทศ อาจารย์ชินภัทร นันทจิวากรชัย

ชื่อ-สกุล ผู้นิเทศงาน นายธนวัช เจริญชาศรี

สถานประกอบการ บริษัท เอ็นเอ็กซ์พี แมนูแฟคเจอริ่ง (ไทยแลนด์) จำกัด

บทคัดย่อ

โครงงานนี้จัดทำขึ้นเพื่อลดปัญหาความยุ่งยาก ซับซ้อนและพัฒนาวิธีการทำ bench test ให้มีความสะดวกและใช้งานง่ายมากขึ้น โดยส่วนประกอบที่สำคัญหลักๆจะประกอบไปด้วยส่วนของฮาร์ดแวร์คือตัวบอร์ดทดสอบการทำงานของไอซี (application bench board) และส่วนของซอฟต์แวร์คือหน้าต่างแสดงการใช้งาน (user interface) โดยการทำงานนี้จะครอบคลุมทั้งหมด 4 ผลิตภัณฑ์ได้แก่ LED driver , voltage translator , GPIO และ multiplexer ซึ่งในส่วนของตัวบอร์ดนั้นมีซ็อกเก็ตที่รองรับขาของไอซีได้มากที่สุด 56 ขา มีบัสไว้สำหรับต่อไฟเลี้ยง กราวด์และจุดวัดสัญญาณต่างๆ รวมถึงมีการต่อหลอดไฟไว้สำหรับในส่วนของการแสดงผลอีกด้วย นอกจากนี้ยังมีโมดูลของวงจรรักษาระดับแรงดัน วงจรดึงกระแสและวงจรแรมป์ภายในบอร์ดเพื่ออำนวยความสะดวกในการใช้งาน ส่วนในส่วนของซอฟต์แวร์นั้นจะใช้การเขียนคำสั่งลงในบอร์ด Arduino ในการทดสอบการทำงานของไอซี โดยผู้ใช้งานสามารถสั่งการทำงานได้จากโปรแกรม WINI2C โดยตรงหรือจาก UI ก็ได้ โดยฟังก์ชันสำคัญที่ได้ทำขึ้นนั้นคือการอ่านรีจิสเตอร์ทั้งหมดและตารางเปรียบเทียบรีจิสเตอร์ภายในไอซีระหว่างตัวดีและตัวที่มีปัญหา โดยอาศัยหลักการทำงานของ I2C (Inter-Integrated Circuit)

คำสำคัญ : บอร์ดทดสอบการทำงานของไอซี , หน้าต่างแสดงการใช้งาน , ตารางเปรียบเทียบรีจิสเตอร์อ่านรีจิสเตอร์ทั้งหมด , I2C

Cooperative Title : Application Bench Board

Student intern name : Miss Pataraporn Sriburapa

Faculty : Engineering **Department :** Electronics

Advisor name : Mr. Chinnapat Nantajiwakornchai

Mentor name : Mr. Thanawat Jaroenchasri

Company : NXP MANUFACTURING (THAILAND)

ABSTRACT

This project is invented to decrease the troubles and develop method of bench test for convenience and easier to use. By compose of hardware is application bench board and software is user interface (UI). The operation of board includes 4 products such as LED driver, voltage translator , GPIO and multiplexer. Application bench board has the maximum socket's pins are 56 pins. Bus for supply voltage , ground , signals and LEDs for outputs. In addition , the board has voltage regulator , active load and ramp module for facility. Software is programed by using Arduino for test operation of products. By user can command the operation from WINI2C program or user interface. The important function is invented as read all registers and compare table registers. By using operation of I2C (Inter-Integrated Circuit)

Keywords : application bench board , user interface , compare table registers , read all registers , I2C

กิตติกรรมประกาศ

โครงงานนี้ประสบผลสำเร็จไปได้ด้วยดีทั้งนี้เนื่องจากคำปรึกษาของอาจารย์ชินภัทร นันทจิวากรชัย และอาจารย์เกรียงไกร สุขสุด อาจารย์นิเทศภาควิชาวิศวกรรมอิเล็กทรอนิกส์ และคำแนะนำของผู้นิเทศงาน นายธนวัช เจริญชาศรี ที่ได้ให้คำเสนอแนะ แนวคิด ตลอดจนแก้ไขข้อบกพร่องต่างๆมาโดยตลอดจนทำให้โครงงานสำเร็จลุล่วงไปได้ รวมถึงพี่ๆในแผนกทุกคนที่คอยให้คำแนะนำ สอนงาน และให้การต้อนรับเป็นอย่างดี ทำให้ผู้จัดทำได้รับความรู้และประสบการณ์เพิ่มมากขึ้น ขอบคุณพี่ๆแผนกทำพีซีบีและพี่ฝ่าย HR ที่ช่วยสละเวลามาติดต่อประสานงาน ให้คำแนะนำและให้การสนับสนุนมาโดยตลอด

สุดท้ายนี้ขอขอบพระคุณ คุณพ่อ คุณแม่และครอบครัว ที่ให้คำปรึกษาในเรื่องต่างๆ รวมทั้งเป็นกำลังใจที่ดีเสมอมา รวมถึงเพื่อนๆทุกคนที่ให้คำแนะนำดีๆเกี่ยวกับโครงงานนี้ด้วย

ภัทรภร ศรีบูรพา

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญ

เนื่องจากไอซีทุกตัวที่ได้ผลิตออกมานั้นต้องได้รับการทดสอบการทำงานและคุณสมบัติต่างๆ ที่ได้ว่าเป็นไปตามที่กำหนดหรือไม่ก่อนที่จะส่งออกไปให้ลูกค้า โดยจะทำการทดสอบด้วยเครื่อง tester ต่างๆที่ได้โปรแกรมคำสั่งเอาไว้ซึ่งการเขียนโปรแกรมบนเครื่อง tester จำเป็นต้องอ่านได้ผลเช่นเดียวกับค่าที่ได้จากการทดลองทดสอบในห้องแล็ปหรือที่เรียกว่า “bench test” โดยการทดสอบนี้จะทำโดยการต่อวงจรตาม application ของไอซีตัวนั้นๆ แล้วใช้เครื่องมือวัดสัญญาณและค่าต่างๆว่าถูกต้องตรงตามสเปคหรือไม่ ซึ่งวิธีการนี้จะทำให้สามารถทราบถึงปัญหาของไอซีได้หรือได้ผลการทดสอบคุณสมบัติตามสเปคที่มีความน่าเชื่อถือสำหรับใช้เป็นค่าอ้างอิงในการ calibrate โปรแกรมได้ และในกรณีที่สินค้าที่ส่งออกไปถูก complain กลับมาก็จะต้องทำการทดสอบในห้องทดลองเพื่อหาสาเหตุของปัญหาที่เกิดขึ้นโดยใช้วิธีเดียวกันนี้อีกด้วย

แต่วิธีการนี้จะใช้เวลามากและอาจมีข้อผิดพลาดเกิดขึ้นได้เนื่องจากต้องต่อวงจรเอง ซึ่งถ้ามีไอซีหลายตัวที่มีการทำงานไม่เหมือนกันก็ต้องต่อวงจรใหม่ทุกครั้งก่อนที่จะสามารถทำการทดสอบได้ หรือถ้าไอซีมีจำนวนขาที่มากก็จะทำให้เกิดความซับซ้อน ยุ่งยากและใช้เวลามากขึ้น จึงมีความคิดที่ว่าจะออกแบบบอร์ดที่สามารถรองรับไอซีได้หลายชนิดและหลายขาไว้ภายในบอร์ดเดียว โดยภายในบอร์ดนั้นจะมีอุปกรณ์ที่ต่อไว้พร้อมใช้งานได้เลย ทำให้มีความสะดวกสบายในการใช้งาน ประหยัดเวลาที่ใช้ในการทดลอง bench test และเวลาที่จะใช้ในการพัฒนาโปรแกรมสำหรับเครื่อง tester ได้ด้วย

1.2 วัตถุประสงค์ของการวิจัย

ศึกษาวิธีการทดสอบการทำงานของไอซีแล้วนำปัญหาที่เกิดขึ้นมาแก้ไขและพัฒนาให้ดีขึ้น โดยนำความรู้พื้นฐานทางด้านเซอร์กิตมาประยุกต์ใช้ในบอร์ดในส่วนของวงจรรักษาระดับแรงดัน วงจรดึงกระแสและวงจรแรมป์ ซึ่งเป็นวงจรย่อยที่ช่วยให้บอร์ดหลักทำงานได้ง่ายขึ้นและยังมีส่วนแสดงผลโดยใช้หลอดไฟในการอ่านค่าทำให้ง่ายขึ้นต่อการใช้งาน โดยใช้โปรแกรม designspark ในการออกแบบลายวงจร อีกทั้งยังมีส่วนของซอฟต์แวร์ที่ต้องพัฒนาอีกด้วย ซึ่ง Application bench board นี้จัดทำขึ้นเพื่อลดระยะเวลาและความยุ่งยากในการต่อวงจร อีกทั้งยังง่ายต่อการใช้งานอีกด้วย

1.3 ขอบเขตของการวิจัย

ศึกษาและออกแบบวงจรที่ประกอบด้วยอุปกรณ์ต่างๆ เช่น ตัวต้านทาน จัมเปอร์ และหลอดไฟไว้สำหรับแสดงผล อีกทั้งยังมีส่วนโมดูลของวงจรรักษาระดับแรงดัน วงจรดึงกระแสและวงจรแรมป์ไว้สำหรับช่วยในการทำงานของบอร์ดหลัก อีกทั้งในส่วนของซอฟต์แวร์ที่นำโปรแกรมคำสั่ง I2C มาใช้ประกอบกับการทดสอบการทำงานของไอซี และซ็อกเกตไว้สำหรับใส่ไอซีที่ต้องการทดสอบด้วย

1.4 วิธีการดำเนินการวิจัย

ศึกษาวิธีการ bench test และสังเกตถึงปัญหาที่เกิดขึ้น จากนั้นจึงทำการรวบรวมข้อมูลของตัวผลิตภัณฑ์และกำหนดขอบเขตของโครงงาน ศึกษาหาข้อมูลและเรียนรู้การออกแบบวงจรต่างๆ เพื่อที่จะนำมาประยุกต์ใช้กับบอร์ดหลัก ออกแบบวงจรและอุปกรณ์บางตัวในโปรแกรม designspark แก้ไขและตรวจสอบความเรียบร้อยแล้วนำไปกัดลายวงจร จากนั้นเริ่มศึกษาวิธีการใช้โปรแกรม visual studio และภาษา C++ เบื้องต้น เพื่อที่จะนำมาพัฒนาในส่วนของซอฟต์แวร์ที่ใช้ในการทดสอบการทำงานของไอซี เมื่อได้บอร์ดมาแล้วจึงนำมาทดสอบการทำงานร่วมกับคำสั่งที่ได้เขียนเอาไว้ว่าได้ประสิทธิภาพตามที่ได้ตั้งเป้าหมายเอาไว้หรือไม่ จากนั้นนำผลการทดลองที่ได้มาสรุปผลและทำการนำเสนอผลงาน

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ได้เรียนรู้การออกแบบวงจรจากการนำอุปกรณ์แต่ละตัวมาประกอบกัน และไม่ต้องใช้เครื่องมือทางอิเล็กทรอนิกส์หลายอย่างในการทดสอบวงจร เนื่องจากมีวงจรรักษาระดับแรงดันภายในบอร์ดไว้สำหรับแบ่งจ่ายไฟเลี้ยงให้ส่วนต่างๆ ทำให้ลดต้นทุนในการซื้อเครื่องมือ อีกทั้งบอร์ดนี้ทำไว้สำหรับรองรับได้หลายผลิตภัณฑ์ทำให้ลดต้นทุนการผลิต ง่ายต่อการใช้งานและประหยัดเวลาอีกด้วย

บทที่ 2

แนวคิด ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

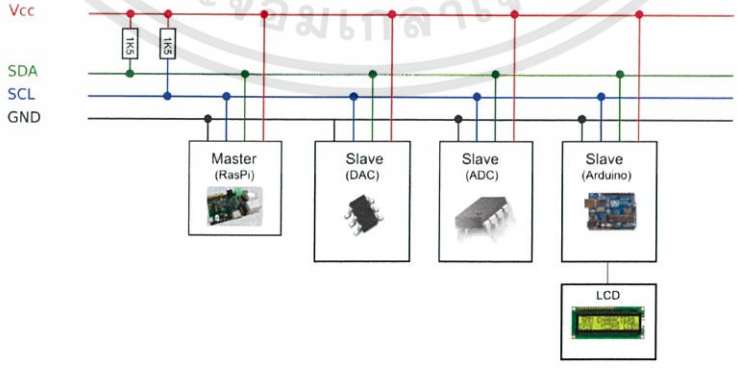
2.1 ทฤษฎีที่เกี่ยวข้อง

Application bench board เป็นบอร์ดที่ไว้ใช้ทดสอบการทำงานของไอซีระหว่าง Arduino กับไอซีโดยใช้การติดต่อสื่อสารด้วยบัส I2C และมีวงจรรักษาระดับแรงดัน วงจรดึงกระแส และวงจรแรมป์ภายในบอร์ดด้วยเพื่ออำนวยความสะดวกในการใช้งาน โดยมีหลักการการทำงานดังนี้

2.1.1 Inter-Integrated Circuit (I2C)

I2C ย่อมาจาก Inter-Integrated Circuit หมายถึงการติดต่อสื่อสารระหว่างไอซีโดยบัส I2C ได้รับการพัฒนาโดยฟิลิปส์ (Philips) ด้วยจุดมุ่งหมายหลักคือต้องการให้ไอซีหรือโมดูลสามารถติดต่อสั่งงาน และควบคุมภายใต้สายสัญญาณเพียง 2 เส้น เส้นหนึ่งคือสายสัญญาณนาฬิกา(SCL) ที่ใช้กำหนดจังหวะการทำงาน และอีกเส้นหนึ่งคือสายข้อมูล(SDA) การต่อร่วมกันของอุปกรณ์บนบัส I2C ทำได้โดยการต่อสายข้อมูลและสายสัญญาณนาฬิกาของอุปกรณ์แต่ละตัวขนานหรือพ่วงกันไป ส่วนการกำหนดแอดเดรสหรือตำแหน่งสำหรับติดต่ออุปกรณ์แต่ละตัว จะใช้รหัสข้อมูลและการกำหนดสภาวะโลจิกที่ขาแอดเดรสของอุปกรณ์แต่ละตัว สายข้อมูลบนบัส I2C มีชื่อเรียกอย่างเป็นทางการว่าสายข้อมูลอนุกรม หรือ SDA (Serial Data line) ส่วนสายสัญญาณนาฬิกามีชื่อเรียกว่าสายสัญญาณนาฬิกาอนุกรม หรือ SCL (Serial Clock line)

สาย SDA และ SCL เป็นสายสัญญาณ 2 ทิศทาง (bi-directional line) ต้องมีการต่อตัวต้านทาน pull-up กับแรงดัน +5V ไว้ตลอดเวลา เพื่อให้สายมีสถานะลอจิกสูงในขณะที่ไม่มีการติดต่อใช้งาน ทั้งยังช่วยป้องกันสัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสอง วงจรเอาต์พุตของอุปกรณ์ที่ต่ออยู่บนบัส I2C ต้องมีลักษณะเป็นวงจรเดรนเปิด(Open-drain) หรือคอลเล็กเตอร์เปิด(Open-collector)



ภาพที่ 2.1 ลักษณะการการเชื่อมต่ออุปกรณ์แบบ I2C BUS

อุปกรณ์บนบัส I2C สามารถเป็นได้ทั้งตัวรับและส่ง บางอุปกรณ์ทำหน้าที่เป็นตัวรับอย่างเดียวจะไม่มีอุปกรณ์ใดบนบัส I2C ที่ทำหน้าที่เป็นตัวส่งอย่างเดียว

- อุปกรณ์ที่เป็นผู้สร้างข้อมูลหรือส่งข้อมูลเรียกว่า ตัวส่ง (transmitter)
- อุปกรณ์ที่เป็นผู้รับข้อมูลเรียกว่า ตัวรับ (receiver)
- อุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการติดต่อบนบัส I2C เรียกว่า มาสเตอร์ (master)
- อุปกรณ์ที่ถูกควบคุมหรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส I2C เรียกว่า สเลฟ (slave)

การสื่อสารแบบ I2C มีรูปแบบและมีขั้นตอนการรับส่งข้อมูลเป็นลำดับ ดังนี้

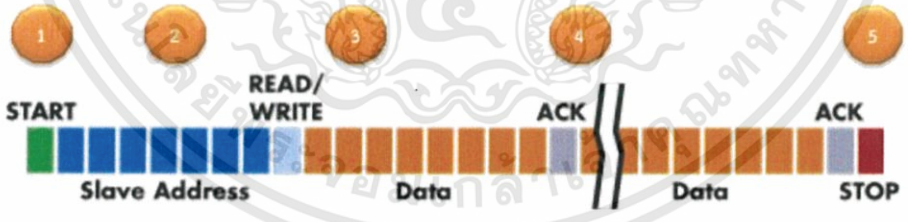
1. เพื่อเริ่มต้นสถานะการสื่อสารแบบ I2C อุปกรณ์ Master เริ่มจะส่งคำสั่ง START ซึ่งจะทำให้อุปกรณ์ Slave ทุกตัวที่อยู่ใน Bus เตรียมพร้อมรับข้อมูลจากสายส่งข้อมูล (SDA)

2. อุปกรณ์ Master ส่ง Address ขนาด 7 บิตและตามด้วยคำสั่งอ่านหรือเขียน (Read/Write) อีก 1 บิต

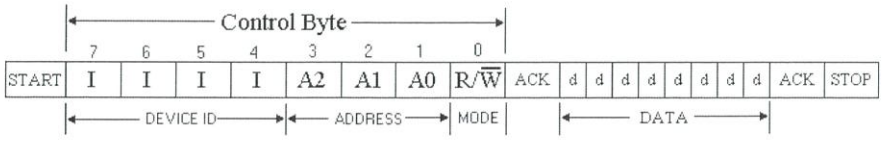
3. อุปกรณ์ Slave ทุกตัวใน Bus จะตรวจสอบ Address ในสายส่งข้อมูลว่าตรงกับ Address ของตนหรือไม่ หากตรงกันจะส่งสัญญาณ ACKNOWLEDGE (ACK) ขนาด 1 บิต กลับไปยังอุปกรณ์ Master เพื่อเตรียมพร้อมดำเนินการต่อไป

4. การรับส่งข้อมูลระหว่าง Master และ Slave ที่ระบุ Address จะดำเนินการอย่างต่อเนื่อง ในกรณีที่ Master ส่งคำสั่ง Read อุปกรณ์ Slave จะส่งข้อมูลเป็นชุดๆ ชุดละ 8 บิต (1 ไบต์) เมื่ออุปกรณ์ Master ส่งสัญญาณ ACK เมื่อได้รับทุกๆ ไบต์

5. อุปกรณ์ Master จะส่งคำสั่ง STOP เพื่อสิ้นสุดสถานะการสื่อสารแบบ I2C



ภาพที่ 2.2 รูปแบบการสื่อสารแบบ I2C

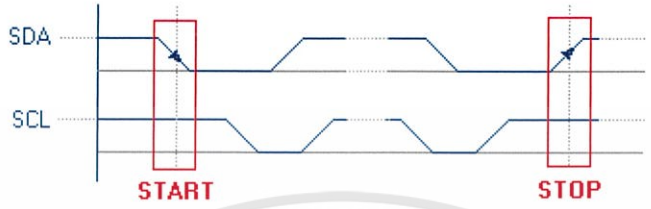


ภาพที่ 2.3 รูปแบบการเขียน-อ่านข้อมูลแบบ I2C BUS

สภาวะที่เกิดขึ้นบนบัส I2C มีดังนี้

1. สถานะบัสว่าง คือเมื่อบัสไม่ได้ถูกใช้งานทั้ง SCL และ SDA จะเป็น 1 ทั้งคู่

2. การกำหนดสถานะเริ่มต้นและสถานะสิ้นสุดของ I2C BUS (START and STOP Conditions)



ภาพที่ 2.4 I2C BUS START and STOP Conditions

ลักษณะการกำหนดสถานะเริ่มต้นและสถานะสิ้นสุดของ **I2C BUS**

- เมื่อต้องการส่งข้อมูลจะต้องส่งสถานะเริ่มต้น (START Conditions) คือให้ SDA เปลี่ยนจาก 1 มาเป็น 0 ในขณะที่ SCL มีค่าเป็น 1

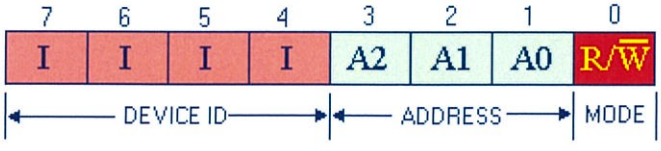
- เมื่อสิ้นสุดการใช้บัสจะต้องส่งสถานะสิ้นสุด (STOP Conditions) คือให้ SDA เปลี่ยนจาก 0 มาเป็น 1 ในขณะที่ SCL มีค่าเป็น 1

3. รหัสควบคุมของ I2C BUS (Control Byte)

รหัสควบคุมของ I2C BUS ประกอบด้วยรหัสประจำตัวของอุปกรณ์ (Device ID) ประกอบด้วยบิต 1-7 และบิต 0 เป็นบิตควบคุมการเขียน-อ่าน

- รหัสประจำตัวของอุปกรณ์ประกอบด้วยรหัสประจำตัวจากผู้ผลิต Product ID 4 บิต (บิต 4-7) ที่เปลี่ยนแปลงแก้ไขไม่ได้ และ Device Address 3 บิต (บิต 1-3) ซึ่งผู้ใช้สามารถกำหนดเองได้ รวมแล้วเป็นรหัส 7 บิต ใช้ระบุตัวอุปกรณ์ที่ต่ออยู่บนบัสจะมีค่าซ้ำกันไม่ได้

- บิตควบคุมการเขียน-อ่าน (Mode) บิต 0 เมื่อต้องการเขียนข้อมูลไปยังอุปกรณ์ก็กำหนดให้บิตนี้เป็น 0 และเมื่อต้องการอ่านข้อมูลจากอุปกรณ์ก็กำหนดให้บิตนี้เป็น 1

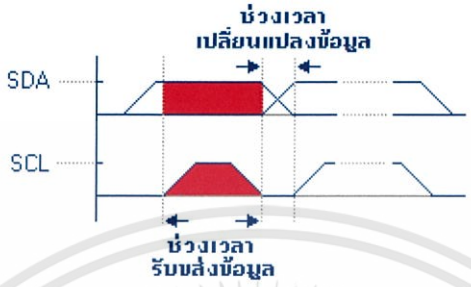


ภาพที่ 2.5 I2C BUS (Control Byte)

4. ช่วงเวลารับส่งบิตข้อมูลของ I2C BUS

- สภาวะการรับ-ส่งข้อมูล จะกระทำในขณะที่ขา SCL เป็น 1

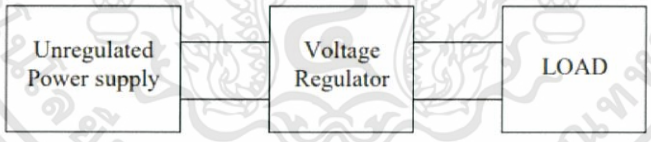
- สภาวะการเปลี่ยนแปลงข้อมูล จะกระทำในขณะที่ขา SCL เป็น 0



ภาพที่ 2.6 การรับส่งบิตข้อมูลของ I2C BUS

2.1.2 วงจรรักษาระดับแรงดัน (Voltage regulator)

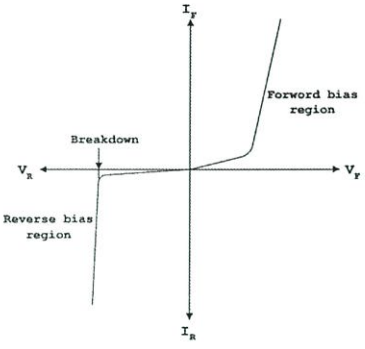
วงจรรักษาระดับแรงดัน คือวงจรที่ต่อระหว่างแหล่งจ่ายไฟตรงไมคงค่า (Unregulator Power Supply) กับโหลด มีหน้าที่จ่ายไฟตรงให้กับโหลดและสามารถรักษาแรงดันให้คงตัว การสร้างวงจรรักษาระดับแรงดันจะมีอยู่ 2 ลักษณะคือการสร้างแบบอนุกรมกับโหลด และแบบขนานกับโหลดโดยใช้ซีเนอร์ไดโอด ทรานซิสเตอร์และไอซี โดยในที่นี้จะอธิบายถึงหลักการทำงานของวงจรรักษาระดับแรงดันที่ใช้ซีเนอร์ไดโอดและไอซี TL431



ภาพที่ 2.7 บล็อกไดอะแกรมของแหล่งจ่ายไฟตรงคงค่าแรงดัน

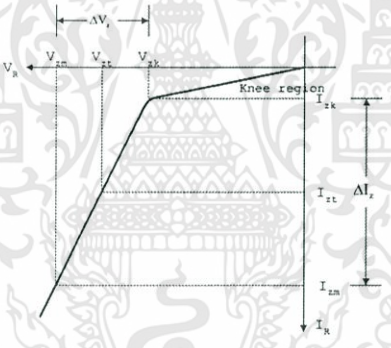
2.1.2.1 วงจรรักษาระดับแรงดันโดยใช้ซีเนอร์ไดโอด (Zener Voltage Regulator)

ซีเนอร์ไดโอด (Zener Diode) เป็นไดโอดชนิดพิเศษที่สร้างให้มีการทำงานแตกต่างจากไดโอดเรียงกระแสทั่วไป คือเมื่อให้ไบอัสตรงกับซีเนอร์ไดโอดการทำงานจะเหมือนกับไดโอดเรียงกระแสคือนำกระแสได้ และมีแรงดันตกคร่อมซีเนอร์ไดโอดขณะไดรับไบอัสเท่ากับ V_B แต่เมื่อซีเนอร์ไดโอดไดรับไบอัสกลับถึงค่าแรงดันที่กำหนด ซีเนอร์ไดโอดนำกระแสไดและจะเกิดแรงดันตกคร่อมตัวเองคงที่เท่ากับค่าแรงดันที่กำหนดจาก Datasheet



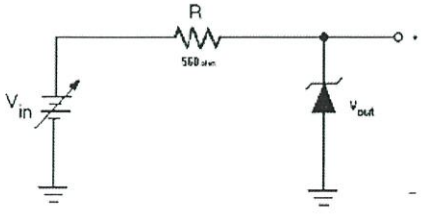
ภาพที่ 2.8 กราฟลักษณะสมบัติทางกระแสและแรงดันของซีเนอรไดโอด

คุณลักษณะของการพังทลาย (Breakdown Characteristics) พิจารณาจากกราฟลักษณะสมบัติโดยเฉพาะการพังทลายของซีเนอรไดโอดเมื่อไดรับไบอัสกลับดังภาพที่ 2.9 เมื่อเพิ่มแรงดันไบอัสกลับจนถึงค่าแรงดันซีเนอร (VZ) จะเกิดกระแสไหลผ่านซีเนอรไดโอดมากขึ้นที่จุดเอียงของกราฟ (Knee Point) จะมีกระแสไหลผ่านซีเนอรไดโอดเท่ากับ IZK และถ้าซีเนอรไดโอดไดรับแรงดันสูงสุดขึ้นอีกกระแสจะเพิ่มขึ้นแต่แรงดันซีเนอรจะคงที่ แต่ถ้าเพิ่มกระแสเกินกว่าค่ากระแสซีเนอรสูงสุด (IZM) แรงดันซีเนอรจะไมคงที่



ภาพที่ 2.9 กราฟแสดงลักษณะสมบัติของกระแสและแรงดันของซีเนอร์ไดโอดเมื่อได้รับไบอัสกลับ

ดังนั้นการนำซีเนอรไดโอดไปใช้ในการควบคุมให้แรงดันไฟตรงคงที่โดยใช้ค่าแรงดันซีเนอรนั้น จึงต้องออกแบบวงจรควบคุมให้มีกระแสไหลผ่านซีเนอรไดโอดย่านระหว่างค่ากระแส IZK ถึงค่า IZM สำหรับกระแส IZT หมายถึงค่ากระแสทดสอบค่าแรงดันซีเนอร (Zener Test Current) ซึ่งเป็นค่ากระแสที่พิกัดของแรงดันซีเนอรตามค่าที่กำหนดไวใน Datasheet

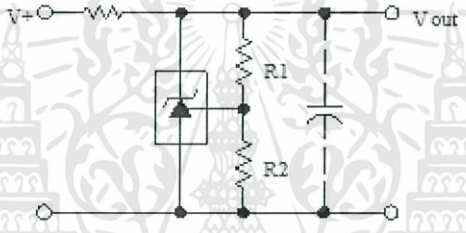


ภาพที่ 2.10 วงจรรักษาระดับแรงดันโดยใช้ซีเนอร์ไดโอด

2.1.2.2 วงจรรักษาระดับแรงดันโดยใช้ไอซี

วงจรแหล่งจ่ายไฟสมัยใหม่ที่นิยมกันมากที่สุด คือวงจรรักษาระดับแรงดันโดยใช้ไอซีเพราะว่าออกแบบวงจรง่าย ราคาถูก ขนาดเล็กและให้คุณภาพสูง ในการออกแบบวงจรรักษาระดับแรงดันโดยใช้ไอซีมีลักษณะต่างๆ ดังนี้คือ ใช้ไอซี 3 ขาแบบแรงดันเอาตพุตคงที่บวกและลบ และใช้ไอซี 3 ขาแบบปรับค่าแรงดันเอาตพุตได้ วงจรรักษาระดับแรงดันโดยใช้ไอซีจะไดสัญญาณแรงดันอินพุตมาจากวงจรเรกติไฟเออรที่ผ่านการกรองแรงดันมาแลว โดยวงจรรักษาระดับแรงดันจะทำหน้าที่ปรับแต่งแรงดันให้เรียบขึ้น และรักษาระดับแรงดันให้คงที่ตลอดการใช้งาน โดยในที่นี้จะกล่าวถึงวงจรรักษาระดับแรงดันโดยใช้ไอซี 3 ขาแบบปรับค่าแรงดันเอาตพุตได้คือ TL431

TL431 เป็นไอซีที่หาได้ง่าย ราคาถูกและมีขนาดเล็ก สามารถปรับแรงดันเอาต์พุตได้ตั้งแต่ V_{ref} ถึง 36V สามารถจ่ายกระแสได้ตั้งแต่ 1mA ถึง 100mA และมีสัญญาณรบกวนที่เอาต์พุตต่ำจึงนำมาออกแบบวงจรรักษาระดับแรงดัน โดยต่อวงจรแบบ Shunt Regulator



ภาพที่ 2.11 Shunt Regulator

โดยค่า R1 และ R2 ส่วนมากจะใช้ค่า 1kΩ และสามารถคำนวณหาค่าแรงดันเอาต์พุตได้จากสมการ

$$v_{out} + 1 = \frac{R1}{R2}(v_{ref})$$

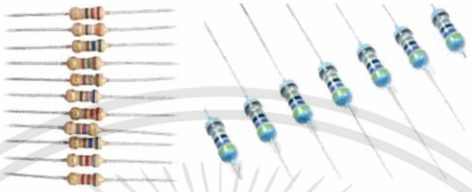
2.1.3 วงจรป้องกันกระแสเกิน (protection)

การป้องกันกระแสเกินถูกประยุกต์ใช้งานในวงจรไฟฟ้าและวงจรอิเล็กทรอนิกส์ต่างๆ เพื่อป้องกันกระแสเกิน ช่วยป้องกันอุปกรณ์ขับกำลังให้กับวงจรนั้นๆ ปลอดภัยและให้อยู่ในช่วงการทำงานที่เหมาะสม และโดยทั่วไปการป้องกันกระแสเกินจะถูกออกแบบให้สามารถกลับคืนการทำงานด้วยตัวเองได้ ในกรณีที่วงจรมีการทำงานผิดปกติ ในกรณีที่ทำให้เกิดการจ่ายกระแสเกินของวงจรนั้นเป็นไปได้หลายกรณี เช่น การชอร์ตเซอร์กิต การชำรุดเสียหายของอุปกรณ์ที่เกี่ยวข้องการขับกำลัง เป็นต้น และผลที่เกิดขึ้นนี้ก็จะทำให้เกิดความร้อนที่อุปกรณ์ขับกำลัง การระเบิดเสียหายของอุปกรณ์อื่นๆ ใกล้เคียงหรือลายวงจรบนแผ่นทองแดงอีกด้วย อุปกรณ์ที่มีหน้าที่ในการควบคุมการไหลของกระแสไฟฟ้าที่นำมาประยุกต์ใช้ในวงจรมีดังนี้

2.1.3.1 ตัวต้านทาน (Resistor)

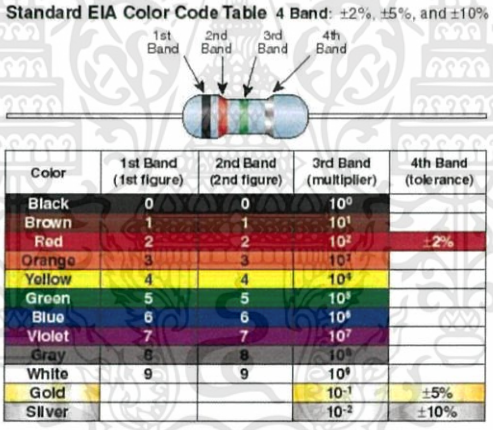
ตัวต้านทานเป็นอุปกรณ์อิเล็กทรอนิกส์ชนิดหนึ่งที่มีคุณสมบัติในการต้านการไหลของกระแสไฟฟ้า ในที่นี้นำมาใช้ในวงจร 2 ประเภท ได้แก่

1) ตัวต้านทานคงที่ (Fixed Value Resistor) เป็นตัวต้านทานที่มีค่าความต้านทานของการไหลของกระแสไฟฟ้าคงที่ สามารถอ่านค่าความต้านทานได้จากแถบสีที่คาดอยู่บนตัวความต้านทาน มีหน่วยเป็นโอห์ม Ω



ภาพที่ 2.12 ตัวต้านทานคงที่แบบ 4 แถบสีและ 5 แถบสี

การอ่านค่าตัวต้านทานแบบคงที่



Color	1st Band (1st figure)	2nd Band (2nd figure)	3rd Band (multiplier)	4th Band (tolerance)
Black	0	0	10^0	
Brown	1	1	10^1	
Red	2	2	10^2	±2%
Orange	3	3	10^3	
Yellow	4	4	10^4	
Green	5	5	10^5	
Blue	6	6	10^6	
Violet	7	7	10^7	
Gray	8	8	10^8	
White	9	9	10^9	
Gold			10^{-1}	±5%
Silver			10^{-2}	±10%

ภาพที่ 2.13 ตารางการอ่านค่าตัวต้านทานแบบคงที่

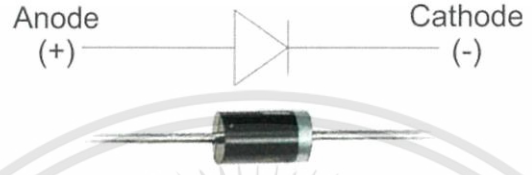
2) ตัวต้านทานเปลี่ยนค่าได้ (Variable Value Resistor) เป็นตัวต้านทานที่เมื่อหมุนแกนของตัวต้านทานแล้วค่าความต้านทานจะเปลี่ยนแปลงไป นิยมใช้ในการควบคุมค่าแรงดันไฟฟ้าในวงจรอิเล็กทรอนิกส์

ภาพที่ 2.14 ตัวต้านทานเปลี่ยนค่าได้

2.1.3.2 ไดโอด (Diode)

ไดโอดเป็นอุปกรณ์อิเล็กทรอนิกส์ที่ยอมให้กระแสไฟฟ้าไหลได้ทางเดียว ทำจากสารกึ่งตัวนำ แบ่งเป็น 2 ชนิด ดังนี้

1) ไดโอดธรรมดา (Normal Diode) ทำหน้าที่ควบคุมการไหลของกระแสไฟฟ้าให้ไหลทางเดียว ทำจากสารกึ่งตัวนำมีขั้วคือขั้วแอโนดและแคโทด



ภาพที่ 2.15 สัญลักษณ์และภาพไดโอดธรรมดา

2) ไดโอดเปล่งแสง (Light Emitting Diode; LED) สามารถเปล่งแสงออกมาได้เมื่อได้รับกระแสไฟฟ้า ทำหน้าที่เปลี่ยนพลังงานไฟฟ้าให้เป็นพลังงานแสง แสงที่เปล่งออกมามีหลายสีขึ้นกับชนิดของสารกึ่งตัวนำที่ผลิต

ภาพที่ 2.16 ไดโอดเปล่งแสง

ประโยชน์ของไดโอด

1) ไดโอดธรรมดา ทำหน้าที่ควบคุมการไหลของกระแสไฟฟ้าให้ไหลไปทิศทางเดียว หากต่อวงจรผิดกระแสไฟฟ้าจะไม่สามารถไหลได้ จึงช่วยป้องกันอุปกรณ์อิเล็กทรอนิกส์ไม่ให้ถูกทำลายจากกระแสไฟฟ้า
2) ไดโอดเปล่งแสง ใช้ในเครื่องใช้ไฟฟ้าต่างๆ ที่มีตัวเลขและตัวหนังสือเรืองแสง เช่น วิทยุเทป หน้าปัดนาฬิกา เครื่องคิดเลข และจอคอมพิวเตอร์

2.1.3.3 ทรานซิสเตอร์ (Transistor)

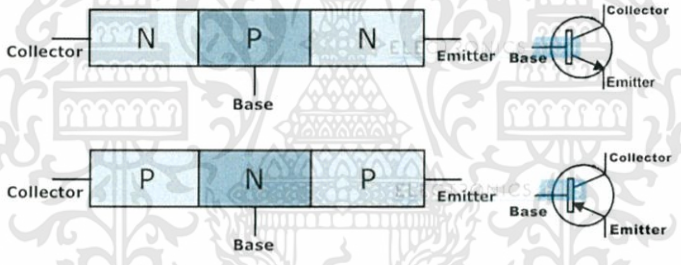
ทรานซิสเตอร์ มีหน้าที่ในการขยายกระแสไฟฟ้าและควบคุมการไหลของกระแสไฟฟ้า (ทั้งอนุญาตให้ไหลและบล็อคไม่ให้ไหลผ่าน) ซึ่งคล้ายๆกับไดโอด แต่ทรานซิสเตอร์สามารถทำอะไรได้มากกว่า เพราะนอกจากจะควบคุมทิศทางการไหลได้แล้ว ยังสามารถควบคุมปริมาณกระแสไฟฟ้าได้ด้วยความสามารถดังกล่าวเกิดขึ้นได้เพราะสารกึ่งตัวนำภายในตัวทรานซิสเตอร์ ทรานซิสเตอร์แต่ละชนิดจะมี 3 ขา ได้แก่

1. ขาเบส (Base : B)
2. ขาอิมิตเตอร์ (Emitter : E)
3. ขาคอลเล็กเตอร์ (Collector : C)

หากแบ่งประเภทของทรานซิสเตอร์ตามโครงสร้างของสารที่นำมาใช้จะแบ่งได้ 2 แบบ คือ

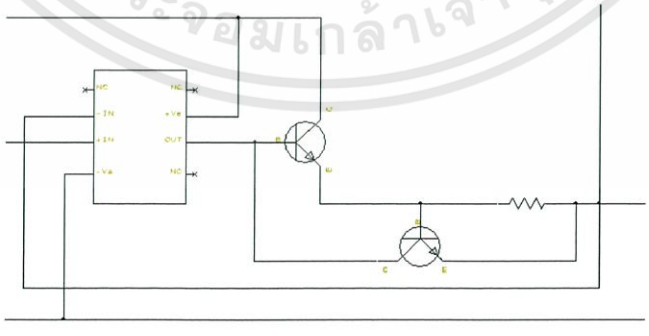
1) ทรานซิสเตอร์ชนิด พีเอ็นพี (PNP) มีหลักการทำงานคือขา C จะทำหน้าที่เป็นกราวด์เมื่อมีกระแสไฟฟ้าเพียงเล็กน้อยที่ขา B มันจะทำการบล็อคไม่ให้กระแสไฟฟ้าไหลผ่านจากขา E ไป C ได้ แต่เมื่อไม่มีกระแสไฟฟ้าที่ขา B เลยหรือกระแสไฟฟ้าติดลบ มันก็จะยอมให้กระแสไฟฟ้าที่มากกว่าไหลผ่านจากขา E ไปขา C

2) ทรานซิสเตอร์ชนิด เอ็นพีเอ็น (NPN) มีหลักการทำงานคือเมื่อมีกระแสไฟฟ้าเพียงเล็กน้อยที่ขา B ทรานซิสเตอร์ก็จะอยู่ในสภาวะทำงาน มันก็จะยอมให้กระแสไฟฟ้าที่มากกว่าหลายเท่าไหลผ่านขา C ไปยังขา E ได้ แต่ในทางตรงกันข้ามถ้าไม่มีกระแสไฟฟ้าที่ขา B เลย ทรานซิสเตอร์จะอยู่ในสภาวะ cut-off คือมันจะบล็อกไม่ให้กระแสไฟฟ้าไหลผ่านขา C ไป E ได้ โดยที่ขา E ทำหน้าที่เป็นกราวด์



ภาพที่ 2.17 ทรานซิสเตอร์ชนิด PNP และ NPN

โดยได้นำอุปกรณ์ที่ควบคุมการไหลของกระแสมาประยุกต์ใช้ในการออกแบบวงจรป้องกันกระแสเกินดังรูป

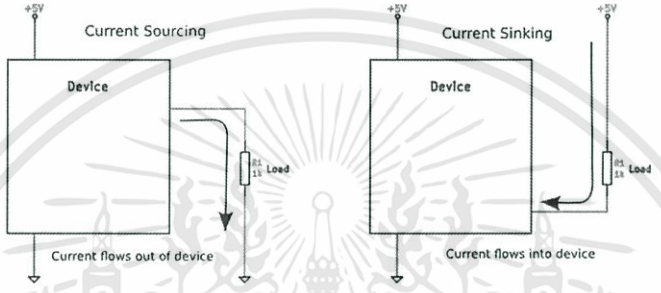


ภาพที่ 2.18 วงจรป้องกันกระแสเกิน

2.1.4 วงจรดึงกระแส (Active load)

วงจรดึงกระแสเป็นวงจรกระแสคงที่ที่เปลี่ยนไปตามค่าแรงดันที่ตกคร่อมตัวต้านทาน ทำหน้าที่ในการทดสอบการดึงกระแสในพาวเวอร์ซัพพลายหรืออุปกรณ์ต่างๆ ที่ทำหน้าที่เป็นโหลด โดยหลักการออกแบบวงจรคล้ายๆ กับวงจรป้องกันกระแสเกิน คือนำอุปกรณ์ที่ควบคุมการไหลของกระแสมาประยุกต์ใช้ในการออกแบบ สามารถออกแบบได้ 2 ลักษณะคือ

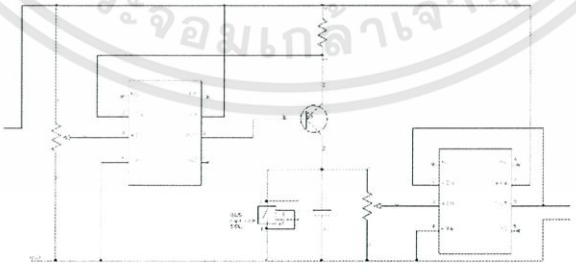
1) Current sink : กระแสจะไหลผ่านโหลดแล้วเข้าขาอุปกรณ์
2) Current source : กระแสจะไหลออกจากขาอุปกรณ์ผ่านโหลดแล้วลงกราวนด์



ภาพที่ 2.19 ลักษณะของวงจร Current source และ Current sink

2.1.5 วงจรแรมป์ (Ramp circuit)

วงจรแรมป์คือวงจรที่เมื่อควบคุมแรงดันและกระแสที่ไหลผ่านตัวเก็บประจุแล้ว จะสามารถปรับค่าเวลาในการแรมป์ได้ขึ้นอยู่กับค่าตัวต้านทานและตัวเก็บประจุที่ใส่เข้าไป โดยที่เมื่อเราจ่ายแรงดันเข้าไปตัวเก็บประจุจะทำหน้าที่ชาร์ตประจุให้เต็มทำให้สัญญาณที่เอาต์พุตเป็นรูปสามเหลี่ยม(ramp) แต่เมื่อหยุดจ่ายไฟตัวเก็บประจุจะคายประจุออกจนหมดทำให้สัญญาณที่ได้เป็นเส้นตรง สามารถออกแบบวงจรได้ดังรูป



ภาพที่ 2.20 วงจรแรมป์

สามารถคำนวณหาค่าเวลาได้จากสมการ

$$T = \frac{1}{RC}$$

2.1.6 การสื่อสารข้อมูลระหว่างบอร์ด Arduino กับไอซีด้วย Wire.h

Wire.h คือไลบรารีของ Arduino ที่ใช้ในการสื่อสารข้อมูลกันระหว่าง Arduino กับไอซีด้วยระบบบัส I2C โดยโปรแกรมให้ Arduino ทำหน้าที่เป็นอุปกรณ์ที่เรียกว่า I2C Master และให้ไอซีทำหน้าที่เป็น I2C Slave ซึ่งอาจมีได้หลายชุด

ฟังก์ชันที่ใช้ในที่นี้คือ

- begin()
- write()
- request.From()
- available()
- begin.Transmission()
- read()
- end.Transmission()

การสื่อสารผ่านบัส I2C เป็นการสื่อสารแบบ Synchronous & Serial คือการส่งข้อมูลทีละบิตและใช้สัญญาณ Clock ในการกำหนดจังหวะการส่งข้อมูล ข้อดีของการสื่อสารข้อมูลแบบบัส I2C คือ ใช้สายสัญญาณเพียง 2 เส้น คือ SCL (สายสัญญาณ Serial Clock) และ SDA (สายสัญญาณข้อมูล Serial Data) และเป็นสัญญาณแบบ 2 ทิศทาง (Bidirectional) มีวงจรภายในสำหรับ I/O แบบ Open-Drain / Open-Collector เวลาใช้งานต้องมีตัวต้านทานแบบ pull-up resistors ต่ออยู่ด้วย บัส I2C สามารถพ่วงอุปกรณ์ได้หลายอุปกรณ์แต่ละอุปกรณ์จะมีหมายเลขที่อยู่ (Device Address) ที่ต้องไม่ซ้ำกัน โดยทั่วไปจะใช้หมายเลขอยู่ที่ขนาด 7 บิต (7-bit Device Address) ซึ่งระบุได้ถึง 128 อุปกรณ์ และในไบต์ดังกล่าวจะมีบิตที่เรียกว่า Read/Write (R/W) Bit สำหรับระบุว่าจะเป็นการเขียนหรืออ่านข้อมูลต่อจากนั้น โดยใน wire library นั้นในการใส่ Device Address จะใส่เพียงแค่ 7 บิต โดยถ้าต้องการเขียนข้อมูลจะใช้คำสั่ง wire.write() ในการส่งไบต์ควบคุม (Control Byte) และไบต์ข้อมูล (Data Byte) ตามลำดับ ส่วนในการอ่านข้อมูลนั้นจะใช้คำสั่ง request.From() ในการกำหนด Device Address และจำนวนไบต์ที่ต้องการจะอ่าน ตามด้วย wire.read() ในการอ่านข้อมูล

```
Serial.begin(9600);
Wire.begin();
Wire.beginTransmission(196);
Wire.write(0x02);
Wire.write(0xFF);
Wire.endTransmission();
```

```
Wire.beginTransmission(AddrInt);
Wire.write(i);
Wire.endTransmission();

Wire.requestFrom(AddrInt,1);//addr,byte
 ch0 = Wire.read(); // read first channel
Serial.println(ch0,HEX);
Wire.endTransmission();
```

ภาพที่ 2.21 ตัวอย่างโค้ดคำสั่งโดยใช้ wire.h library

2.1.7 Serial Communication

Port Communication คือช่องทางการติดต่อสื่อสารระหว่างระบบ System Unit และอุปกรณ์รอบข้าง (Peripheral) เช่น จอภาพ โมเด็ม เครื่องพิมพ์ ซึ่งอุปกรณ์เหล่านี้จะทำการเชื่อมต่อโดยผ่านสายเคเบิลเข้าสู่ Port ของอุปกรณ์ต่างๆ ซึ่งถูกกำหนดขึ้นให้เป็นมาตรฐานสำหรับการติดต่อสื่อสาร Port ที่ใช้การเชื่อมต่อมีอยู่หลายรูปแบบ หลายหัวเสียบบางทีเรียกว่าตัว Connectors ซึ่ง Connectors ก็จะมีอยู่ 2 อย่างด้วยกันคือ Male Type และ Female Type หรือที่รู้จักกันคือปลั๊กตัวผู้และปลั๊กตัวเมีย แบ่งได้ดังนี้

1) Parallel Port

ภาพที่ 2.22 Parallel Port

2) Serial Port

ภาพที่ 2.23 Serial Port

3) USB (Universal Serial Bus)

ภาพที่ 2.24 USB (Universal Serial Bus)

ซึ่งในที่นี้จะกล่าวถึง Serial Port

Serial Port คือพอร์ตอนุกรม ในการสื่อสารข้อมูลนั้นพอร์ตอนุกรมจะมีความเร็วในการสื่อสารที่ช้ากว่าแบบขนาน เพราะการเคลื่อนย้ายข้อมูลแบบอนุกรมนั้นเป็นการส่งข้อมูลครั้งละ 1 บิต แต่พอร์ตขนานนั้นสามารถส่งข้อมูลทีละหลายๆ บิทพร้อมๆ กันได้ แต่ข้อดีของการสื่อสารข้อมูลแบบอนุกรมคือสามารถส่งข้อมูลได้ในระยะทางที่ไกลกว่าแบบขนาน และใช้สายสัญญาณที่น้อยกว่าการสื่อสารข้อมูลแบบขนาน

ประเภทของการสื่อสารแบบอนุกรมแบ่งตามลักษณะสัญญาณในการส่งแบ่งได้ 2 แบบ คือ

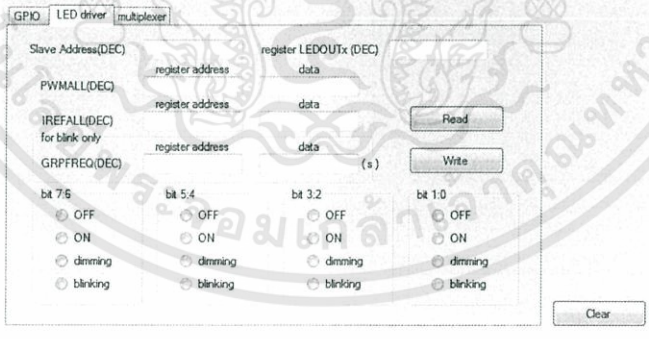
1. การสื่อสารแบบซิงโครนัส (Synchronous) เป็นการสื่อสารข้อมูลโดยใช้สัญญาณนาฬิกาในการควบคุมจังหวะของการรับส่งสัญญาณ ซึ่งสำหรับการติดต่อสื่อสารแบบบัส I2C จะใช้การรับส่งข้อมูลแบบ Synchronous

2. การสื่อสารแบบอะซิงโครนัส (Asynchronous) เป็นการสื่อสารที่ใช้สายข้อมูลเพียงตัวเดียวจะใช้รูปแบบของการส่งข้อมูล (Bit Pattern) เป็นตัวกำหนดว่าส่วนไหนเป็นส่วนเริ่มต้นข้อมูล ส่วนไหนเป็นตัวข้อมูล ส่วนไหนจะเป็นตัวตรวจสอบความถูกต้องของข้อมูลและส่วนไหนเป็นส่วนปิดท้ายของข้อมูล โดยต้องกำหนดให้สัญญาณนาฬิกาเท่ากันทั้งภาคส่งและภาครับ

2.1.8 User Interface (UI)

User Interface หมายถึงส่วนติดต่อระหว่างผู้ใช้กับระบบเพื่อรองรับการนำข้อมูลหรือคำสั่งเข้าไปสู่ระบบ ตลอดจนนำเสนอสารสนเทศกลับมายังผู้ใช้ สามารถแบ่งออกได้เป็น 3 รูปแบบหลักๆ คือ การโต้ตอบด้วยภาษามนุษย์ การโต้ตอบด้วยคำสั่งและการโต้ตอบด้วยกราฟิก ซึ่งในโครงงานนี้ในส่วนของ UI ทำแบบการโต้ตอบด้วยคำสั่งโดยส่งคำสั่งจากคอมพิวเตอร์ไปยัง Arduino

การโต้ตอบด้วยคำสั่ง (Command Language Interaction: CLI) หรือ Command เป็นการโต้ตอบโดยผู้ใช้งานจะต้องใช้ คำสั่งที่เป็นตัวอักษร (String) ผ่านคีย์บอร์ดสำหรับเป็น input เข้าสู่ระบบและระบบจะแสดง output ผ่านหน้าจอ มีข้อดีคือประมวลผลเร็ว ใช้งานง่ายและไม่ซับซ้อน แต่มีข้อเสียคือต้องใช้คำสั่งเฉพาะและภาพลักษณ์ไม่ดึงดูดผู้ใช้งาน



ภาพที่ 2.25 ตัวอย่างของ UI

2.2 งานวิจัยที่เกี่ยวข้อง

ในที่นี้จะกล่าวถึงกลุ่มผลิตภัณฑ์หลักที่ Application bench board รองรับการใช้งาน แบ่งออกเป็น 4 กลุ่มหลักๆ ได้แก่

2.2.1 LED driver

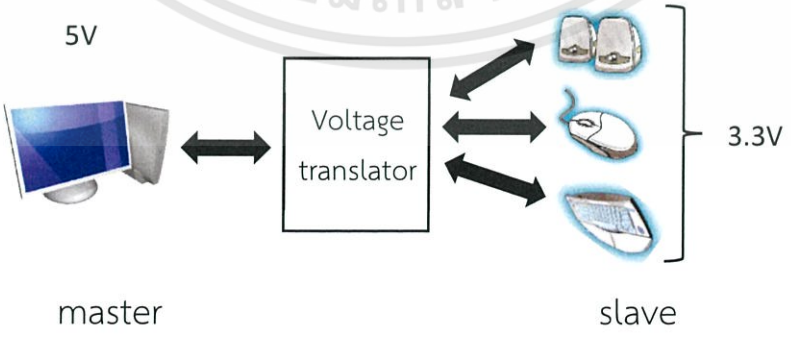
LED driver เป็นไอซีที่ควบคุมสถานะของหลอดไฟมักนำไปใช้สำหรับอุปกรณ์ที่มีหลอดไฟเป็นส่วนประกอบเช่น จอแสดงผล ป้ายไฟ ใช้ในอุปกรณ์มือถือเช่นปุ่มกด แป้นพิมพ์ หรือใช้ในหน้าปัดรถยนต์ มีคุณสมบัติหลักๆ ได้แก่

- โปรแกรมคำสั่งให้หลอดไฟติด-ดับ
- ควบคุมความสว่างของหลอดไฟด้วย PWM (Pulse Width Modulation) และกระแส (IREF)
- โปรแกรมคำสั่งให้หลอดไฟแต่ละดวงกระพริบได้ด้วยความถี่
- กระแสเอาท์พุทแต่ละดวงขึ้นอยู่กับตัวต้านทานที่ต่อจากภายนอก (REXT input)
- ใช้บัส I2C ในการติดต่อสื่อสาร

ภาพที่ 2.26 ตัวอย่างอุปกรณ์ที่ใช้ LED driver

2.2.2 Voltage translator

Voltage translator ทำหน้าที่ในการแปลงระดับแรงดันค่าหนึ่งให้เป็นอีกค่าหนึ่งเพื่อให้อุปกรณ์ที่มีระดับแรงดันในการทำงานที่ต่างกัน สามารถติดต่อสื่อสารกันได้



ภาพที่ 2.27 แผนภาพการทำงานของ Voltage translator

มีคุณสมบัติหลักๆ ได้แก่

- เป็นบัฟเฟอร์ 2 ทิศทาง
- สามารถติดต่อสื่อสารได้ด้วยบัส I2C และบัส SM
- ในบัส I2C สามารถทำงานได้ทั้งในโหมด stand mode และ fast mode
- สามารถต่อตัว master ได้หลายตัวภายในบัส

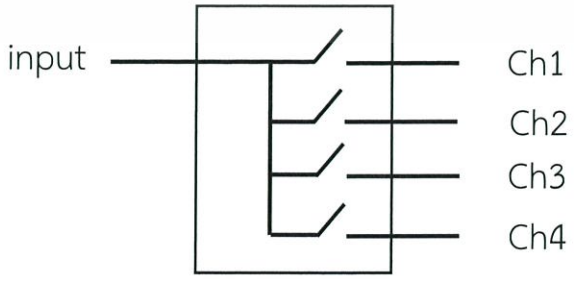
2.2.3 GPIO (General Purpose Input/Output)

GPIO คือพอร์ตเอนกประสงค์ที่เราสามารถควบคุมแต่ละขาให้เป็น “0” หรือ “1” ได้ตามที่ต้องการเรียกได้ว่าเป็นพอร์ตตัวหนึ่งที่ใช้ไปเชื่อมกับอุปกรณ์ที่ทำให้งานเกิดขึ้นได้จริงๆ สามารถนำไปต่อยอดใช้ได้กับอุปกรณ์หลายอย่างเช่น อุปกรณ์ทางการแพทย์ โทรศัพท์มือถือ เครื่องเล่นเกม อุปกรณ์การวัด หรือใช้สำหรับอุตสาหกรรม เป็นต้น

ภาพที่ 2.28 แผนภาพการทำงานของ GPIO

2.2.4 Multiplexer

Multiplexer เป็นไอซีที่ทำหน้าที่เป็นสวิตช์คือมีอินพุทเข้ามาทางเดียวแต่สามารถเลือกทางออกเอาท์พุทได้หลายทาง ติดต่อสื่อสารกันด้วยบัส I2C ไอซีชนิดนี้ไว้ใช้สำหรับเมื่อต้องใช้ไอซีที่มี address เหมือนกันในบัสเดียวกัน หรือใช้สำหรับอุปกรณ์ที่มีระดับแรงดันต่างกันในบัสเดียวกัน ไอซีชนิดนี้มักใช้ในคอมพิวเตอร์ เซิฟเวอร์ การโทรคมนาคม แหล่งจ่ายไฟ หรืออะไรก็ตามที่ต้องการใช้บัส I2C แยกกัน



ภาพที่ 2.29 ลักษณะการทำงานของ Multiplexer

บทที่ 3

วิธีดำเนินการวิจัย

Application Bench Board แบ่งการทำงานออกเป็น 2 ส่วนคือส่วน Hardware และส่วน Software

3.1 ส่วน Hardware

ออกแบบวงจรต่างๆ ได้แก่ บอร์ดหลัก โมดูลวงจรรักษาระดับแรงดัน วงจรดึงกระแสและวงจรแรมป์ มีรายละเอียดดังต่อไปนี้

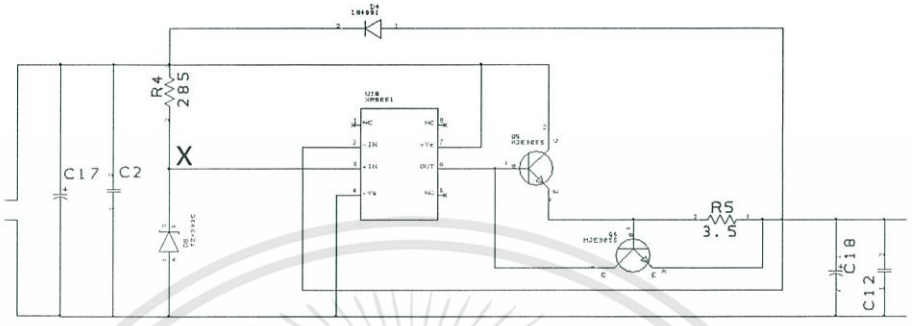
3.1.1 ศึกษาการทำงานของไอซี 4 กลุ่มหลักๆ ได้แก่ LED driver , voltage translator , GPIO และ multiplexer กำหนดระยะเวลาในการทำโครงงานและกำหนดขอบเขตของโครงงาน

ตารางที่ 3.1 ระยะเวลาในการทำโครงงาน

งานที่ทำ	สิงหาคม				กันยายน				ตุลาคม				พฤศจิกายน			
1.กำหนดขอบเขตของโครงงาน	↔															
2.ศึกษาการทำงานของไอซีและเรียนรู้การออกแบบวงจรต่างๆ	↔	↔														
3.คำนวณค่าต่างๆและออกแบบวงจรโมดูลและบอร์ดหลัก		↔	↔													
4.ลากลายวงจรและสั่งซื้อของ			↔	↔	↔	↔	↔	↔								
5.ศึกษาวิธีการติดต่อสื่อสารของ Arduino และภาษา C#								↔	↔							
6.โปรแกรมคำสั่งให้ไอซีทำงานและออกแบบ User Interface									↔	↔	↔	↔				
7.ทดสอบการทำงานของส่วน software ทั้งหมด											↔	↔				
8.ทดสอบการทำงานของบอร์ดกับโปรแกรมที่เขียน												↔	↔	↔	↔	
9.เก็บผลการทดลอง													↔	↔	↔	
10.ทำpresentationและรายงาน															↔	↔

3.1.2 ศึกษาการออกแบบวงจรรักษาระดับแรงดัน และคำนวณค่าต่างๆ แบ่งออกเป็น 3 วงจร ได้แก่ วงจรรักษาระดับแรงดันคงที่ 3.3 โวลต์ , 5 โวลต์ และแรงดันปรับค่าได้ 0 ถึง 7 โวลต์ โดยใช้ซีเนอร์ไดโอดและไอซี TL431 ในการรักษาระดับแรงดัน ยกตัวอย่างเช่น

ที่แรงดันคงที่ 3.3 โวลต์ โดยใช้ซีเนอร์ไดโอดในการรักษาระดับแรงดัน



ภาพที่ 3.1 วงจรรักษาระดับแรงดันคงที่ 3.3 โวลต์

จากภาพ สามารถคำนวณค่าต่างๆได้ดังนี้

- ค่าตัวเก็บประจุที่อินพุทและเอาท์พุทมีไว้เพื่อกรองความถี่และสัญญาณรบกวน
- คำนวณหาค่า R4 เมื่อป้อนอินพุท 9V และใช้ซีเนอร์ 3.3V โดยที่กระแสที่ไหลผ่านซีเนอร์ไดโอด (I_z) มีค่าเท่ากับ 20 mA (ดูจาก datasheet) โดยที่จุด X คือค่าแรงดันตกคร่อมของซีเนอร์ไดโอด (V_z)

  คำนวณได้จากสมการ $R = \frac{V}{I}$

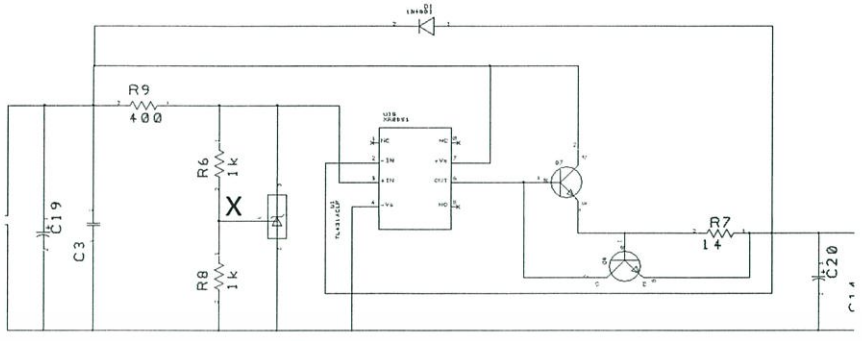
  จะได้ $R = \frac{9-3.3}{20mA} = 285\ \Omega$

- ใช้ออปแอมป์แบบ rail-to-rail ค่ากระแส 100mA
- ใช้ทรานซิสเตอร์เบอร์ MJE3055 ตัวแรกในการขยายกระแส สามารถทนกระแสได้สูงสุด 10A และตัวที่สองใช้เพื่อดูดกระแสที่ขาเบสของตัวแรกไม่ให้มีค่ามากเกินไป
- คำนวณหาค่า R5 โดยที่แรงดันที่ทำให้ทรานซิสเตอร์ทำงานอยู่ที่ 0.7V และต้องการกระแสเอาท์พุท 200 mA จะได้

  $R = \frac{0.7}{200mA} = 3.5\ \Omega$

- ไดโอดเบอร์ 1N4001 ต่อไว้เพื่อป้องกันไม่ให้กระแสไหลย้อนกลับไปในวงจร ป้องกันความเสียหายที่จะเกิดขึ้นกับวงจร

ที่แรงดันคงที่ 5 โวลต์ โดยใช้ไอซีเบอร์ TL431 ในการรักษาระดับแรงดัน



ภาพที่ 3.2 วงจรรักษาระดับแรงดันคงที่ 5 โวลต์

จากภาพ คำนวณเหมือนภาพที่ 3.1 แต่ต่างกันที่ส่วนรักษาระดับแรงดัน

- คำนวณหาค่า R6 และ R8 โดยใช้หลักการของ voltage divider โดยที่จุด X คือ V_{ref} ของ TL431 มีค่าเท่ากับ 2.5V และให้ R6 = 1KΩ หาค่า R8

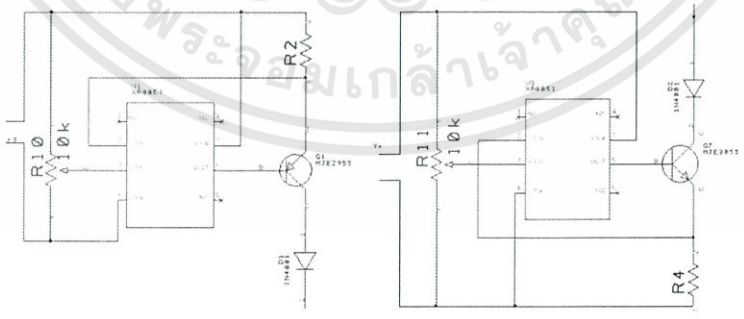
  จะได้ $V_2 = \frac{R8}{R6+R8} \times V_t$; V_t = แรงดันที่ตกคร่อม TL431 = 5V

  $2.5V = \frac{R8}{1K+R8} \times 5V$ ➡ R8 = 1KΩ

- คำนวณหาค่า R9 โดยที่กระแสที่ไหลผ่าน TL431 มีค่าเท่ากับ 10mA

  จะได้ $R = \frac{9-5}{10mA} = 400\ \Omega$

3.1.3 ศึกษาการออกแบบวงจรดึงกระแส และคำนวณค่าต่างๆ แบ่งออกเป็น 2 วงจร ได้แก่ วงจรดึงกระแสแบบ current source และ current sink ดังนี้



ภาพที่ 3.3 วงจรดึงกระแสแบบ current source และ current sink

จากภาพ สามารถคำนวณค่าต่างๆได้ดังนี้

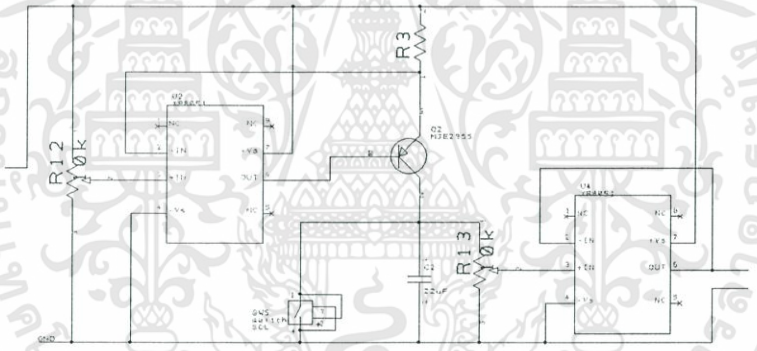
- R10 และ R11 เป็นตัวต้านทานปรับค่าได้ ไว้ควบคุมค่าแรงดันที่ขา V+และV- ของออปแอมป์
- R2 และ R4 ไว้สำหรับควบคุมกระแสเอาท์พุท ในบอร์ดทำไว้เป็น connector pin สามารถใส่ตัวต้านทานตามค่ากระแสที่ต้องการได้ ถ้าคิดในกรณี worst case คือปรับตัวต้านทานปรับค่าได้ให้เท่ากับ 0V จะทำให้กระแสเอาท์พุทไหลได้สูงสุด

  คำนวณได้จากสมการ $R = \frac{V}{I} = \frac{\text{Vin}-\text{V}_{\text{ที่ปรับค่าได้}}}{\text{Ioutput}}$

  ถ้าป้อนอินพุท 9V และต้องการกระแสเอาท์พุท 200mA

  จะได้ $R = \frac{9-0}{200mA} = 45\ \Omega$
- ไดโอดเบอร์ 1N4001 ต่อไว้เพื่อป้องกันไม่ให้กระแสไหลย้อนกลับไปในวงจร ป้องกันความเสียหายที่จะเกิดขึ้นกับวงจร

3.1.4 ศึกษาการออกแบบวงจรแรมป์ และคำนวณค่าต่างๆ

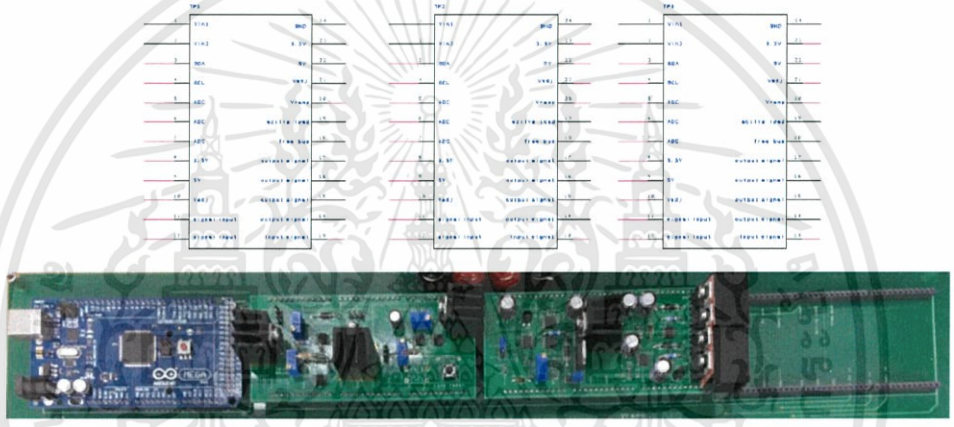


ภาพที่ 3.4 วงจรแรมป์

- R12 ตัวต้านทานปรับค่าได้ ไว้ปรับช่วงเวลาของสัญญาณที่แรมป์ขึ้น
- R3 ไว้สำหรับควบคุมกระแส ในบอร์ดทำไว้เป็น connector pin สามารถใส่ตัวต้านทานตามค่ากระแสที่ต้องการได้
- R13 ตัวต้านทานปรับค่าได้ ไว้ปรับค่าแรงดันเอาท์พุท
- SW5 สวิตช์ไว้สำหรับ reset สัญญาณแรมป์

3.1.5 ออกแบบบอร์ดหลักโดยการจัดวางอุปกรณ์ต่างๆ แบ่งออกเป็น 3 ส่วนหลักๆ ได้แก่ ส่วนของโมดูล ส่วนการต่อวงจร และส่วนแสดงผล จากนั้นวาด schematic ด้วยโปรแกรม designspark

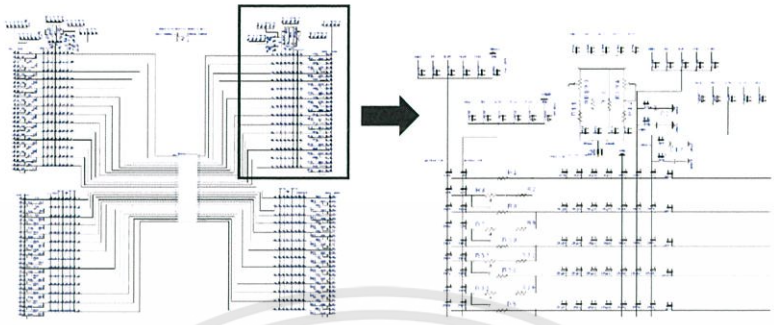
- ส่วนของโมดูล มี 4 โมดูล ประกอบด้วยโมดูล Arduino โมดูลวงจรรักษาระดับแรงดัน โมดูลวงจรดึงกระแสและวงจรแรมป์ และที่ว่างสำหรับโมดูลอื่นๆ ที่สามารถนำมาใช้ได้ในบอร์ดซึ่งอยู่ที่บริเวณด้านบนของบอร์ด มีหน้าที่การทำงานต่างๆกัน ดังนี้
  - โมดูล Arduino : รับคำสั่งจากคอมพิวเตอร์แล้วส่งไปให้ตัวงานทำงานตามที่ต้องการ
  - โมดูลวงจรรักษาระดับแรงดัน : จ่ายแรงดันระดับต่างๆ ให้ตัวงานและอุปกรณ์ภายในบอร์ด
  - โมดูลวงจรดึงกระแส : ใช้ทดสอบเรื่อง IOL , IOH ,VOL และ VOH
  - โมดูลวงจรแรมป์ : ใช้ทดสอบเรื่อง POR
  - ที่ว่างสำหรับโมดูลอื่นๆ : สำหรับโมดูลที่สร้างขึ้นมาในอนาคตเพื่อนำมาใช้ร่วมกับบอร์ด

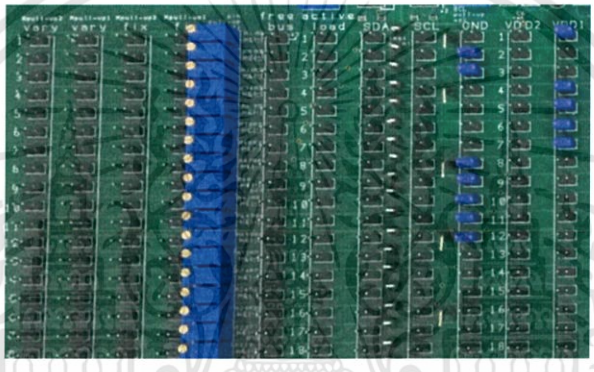


ภาพที่ 3.5 ส่วนของโมดูล

- ส่วนการต่อวงจร ที่ตรงกลางของบอร์ดเป็นซ็อกเกตไว้สำหรับใส่ตัวงานที่ต้องการทดลองรองรับตัวงานที่มีขามากที่สุดได้ 56 ขา ซึ่งในแต่ละขาจะมีบัสต่างๆ และตัวจัมเปอร์ต่ออยู่เพื่อใช้ในการต่อวงจร ดังนี้
  - บัส VDD : ไว้สำหรับเชื่อมต่อขางานที่ต้องการไฟเลี้ยง สามารถเลือกระดับแรงดันได้ 5 ระดับ คือ แรงดันจากภายนอกได้ 2 ระดับด้วยพาวเวอร์ซัพพลาย แรงดันคงที่ 3.3V , 5V และแรงดันปรับค่าได้ 0-7V ได้จากโมดูลรักษาระดับแรงดัน ซึ่งในบัส VDD นี้จะต่อค่าตัวเก็บประจุไว้ด้วยเพื่อกรองความถี่และลดสัญญาณรบกวน
  - บัส GND : ไว้สำหรับเชื่อมต่อขางานที่เป็น GND
  - บัส I2C : แบ่งออกเป็น 2 บัสได้แก่ บัส SDA และบัส SCL ซึ่งในแต่ละบัสสามารถเลือกระดับแรงดัน ได้ 5 ระดับเช่นเดียวกัน
  - บัส active load : ไว้สำหรับเชื่อมต่อขางานที่ต้องการทดสอบเรื่องการดึงกระแส
  - บัส free bus : สามารถนำสัญญาณต่างๆจากภายนอกเข้ามายังขาตัวงานได้

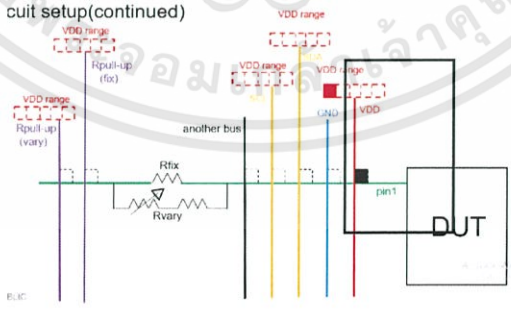
- บัส pull-up resistors : ไว้สำหรับขางานที่จำเป็นต้องต่อ Rpull-up ค่าความต้านทานสามารถเลือกได้ 2 แบบคือ ค่าความต้านทานคงที่ 10KΩ และค่าความต้านทานปรับค่าได้ 1KΩ-10KΩ โดยในบัสนี้สามารถเลือกระดับแรงดันได้ 5 ระดับเช่นเดียวกัน





ภาพที่ 3.6 ส่วนการต่อวงจร

วิธีในการต่อวงจรยกตัวอย่างเช่น ถ้าขา1ของตัวงานเป็นขาไฟเลี้ยงให้ใส่ตัวจัมเปอร์ไว้ที่ขา1ของตัวงานที่บัส VDD และใส่ตัวจัมเปอร์เพื่อเลือกระดับแรงดันที่ต้องการบนบัส VDD ด้วย ดังรูป

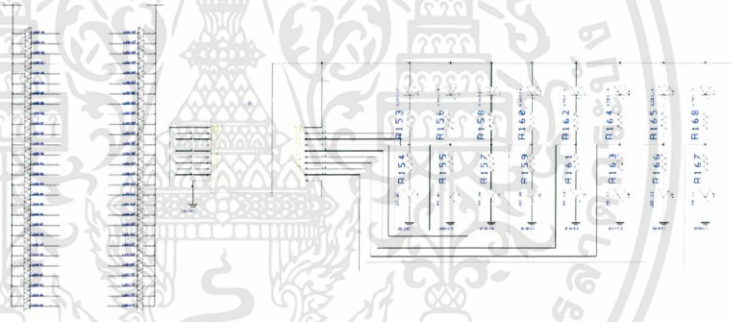


ภาพที่ 3.7 ตัวอย่างการใช้งานของบอร์ด

นอกจากจะมีบัสเหล่านี้แล้ว ยังมีจุด test point ไว้สำหรับวัดสัญญาณที่แต่ละขาของตัวงานและ test point สำหรับ GND เพื่ออำนวยความสะดวกในการใช้งานอีกด้วย

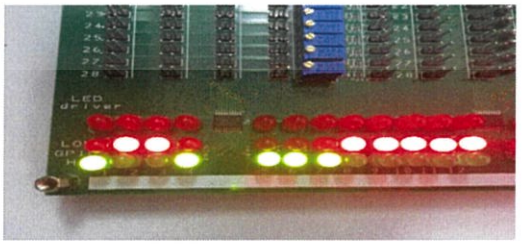
ภาพที่ 3.8 test point สำหรับวัดสัญญาณ และ GND

- ส่วนแสดงผล แบ่งออกเป็น 2 ส่วนได้แก่ ส่วนแสดงผลสำหรับ LED driver และส่วนแสดงผลสำหรับ GPIO ในส่วนนี้จะเป็นการต่อหลอดไฟเพื่อให้ทราบถึงสถานะของขาตัวงาน มีรายละเอียดดังนี้
  - LED driver : จะต่อหลอดไฟโดยตรงเข้ากับขาตัวงานเลย โดยต่อขั้วลบเข้าขาตัวงาน
  - GPIO : จะต่อบัฟเฟอร์จากขาตัวงานก่อนที่จะเข้าหลอดไฟเพื่อช่วยในการขับกระแส มีหลอดไฟ 2 ชุดคือ ชุดแรกหลอดไฟจะติดเมื่อขางานมีสถานะเป็น “HIGH” และชุดที่สองหลอดไฟจะติดเมื่อขางานมีสถานะเป็น “LOW”



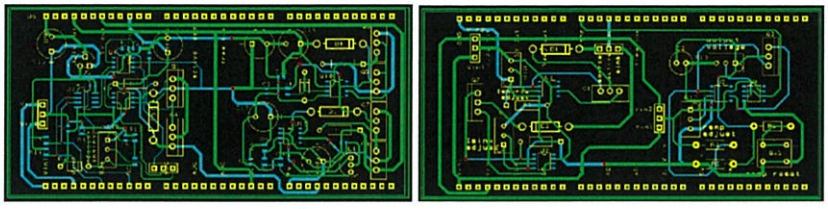
ภาพที่ 3.9 schematic ส่วนแสดงผลสำหรับ LED driver และ GPIO

- แถวบนสำหรับ LED driver
- 2 แถวล่างสำหรับ GPIO ซึ่งถ้าขาตัวงานมีสถานะเป็น “LOW” หลอดไฟสีแดงจะติด แต่ถ้าขาตัวงานมีสถานะเป็น “HIGH” หลอดไฟสีเขียวจะติด

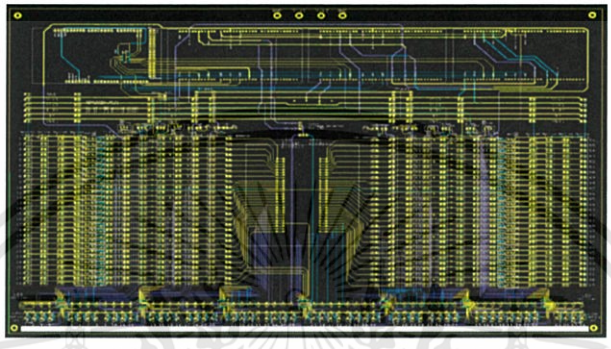


ภาพที่ 3.10 PCB ส่วนแสดงผลสำหรับ LED driver และ GPIO

3.1.6 จากนั้นลากลายพีซีบี 2 เลเยอร์ของโมดูลต่างๆ และพีซีบี 4 เลเยอร์ของบอร์ดหลัก ดังรูป



ภาพที่ 3.11 พีซีบีของโมดูลต่างๆ

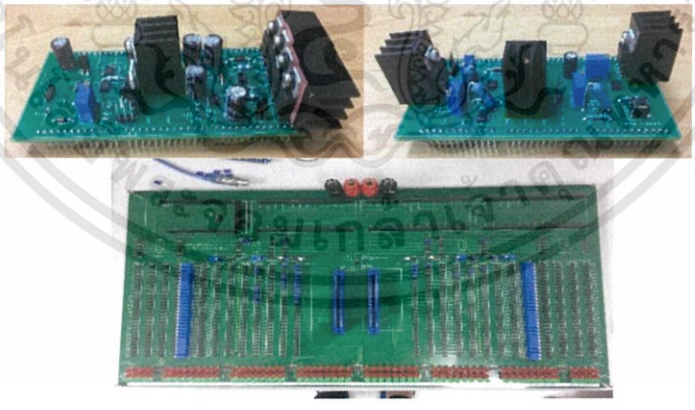


ภาพที่ 3.12 พีซีบีของบอร์ดหลัก

3.1.7 ตรวจสอบความถูกต้องของลายวงจรในพีซีบีทั้งหมดแล้วส่งไปให้โรงงานผลิตพีซีบี

3.1.8 ค้นหาอุปกรณ์ในเว็บไซต์ RS components online และสั่งซื้อ

3.1.9 ลงอุปกรณ์ในแผ่นพีซีบี ดังรูป



ภาพที่ 3.13 บอร์ดที่ลงอุปกรณ์

3.2 ส่วน Software

ในส่วนนี้จะกล่าวถึงการเขียนโค้ดใน Arduino เพื่อควบคุมการทำงานของไอซีกลุ่มผลิตภัณฑ์หลัก 3 กลุ่มที่มีการติดต่อสื่อสารด้วยบัส I2C ได้แก่ LED driver , GPIO และ multiplexer และออกแบบหน้าต่างผู้ใช้งาน (User Interface) โดยใช้ภาษา C# ด้วยโปรแกรม visual basic เพื่อไว้ใช้โปรแกรมคำสั่งการทำงานร่วมกับ Application bench board มีรายละเอียดดังต่อไปนี้

3.2.1 ศึกษาและทำความเข้าใจเกี่ยวกับพื้นฐานการเขียนโปรแกรมเบื้องต้น การทำงานของ I2C และวิธีการติดต่อสื่อสารระหว่าง Arduino กับไอซีด้วย wire library

3.2.2 เขียนโค้ดควบคุมการทำงานของไอซีโดยแบ่งออกเป็น 2 ส่วนคือ เขียนรีจิสเตอร์เพื่อสั่งให้ไอซีทำงาน และอ่านรีจิสเตอร์ภายในตัวไอซีทั้งหมด

3.2.2.1 การเขียนรีจิสเตอร์ เพื่อสั่งให้ไอซีทำงานด้วยการติดต่อสื่อสารบนบัส I2C จะเริ่มต้นด้วยการใส่แอดเดรสของตัวงานด้วยคำสั่ง Wire.beginTransmission() ใส่คำสั่งควบคุมและคำสั่งข้อมูลด้วยคำสั่ง Wire.write() ดังรูป

```
Wire.begin();
Wire.beginTransmission(AddrInt);//slave addr
Wire.write(CntrlInt);  //register addr
Wire.write(DataInt); //data
Wire.endTransmission();
```

ภาพที่ 3.14 ตัวอย่างโค้ดการเขียนรีจิสเตอร์

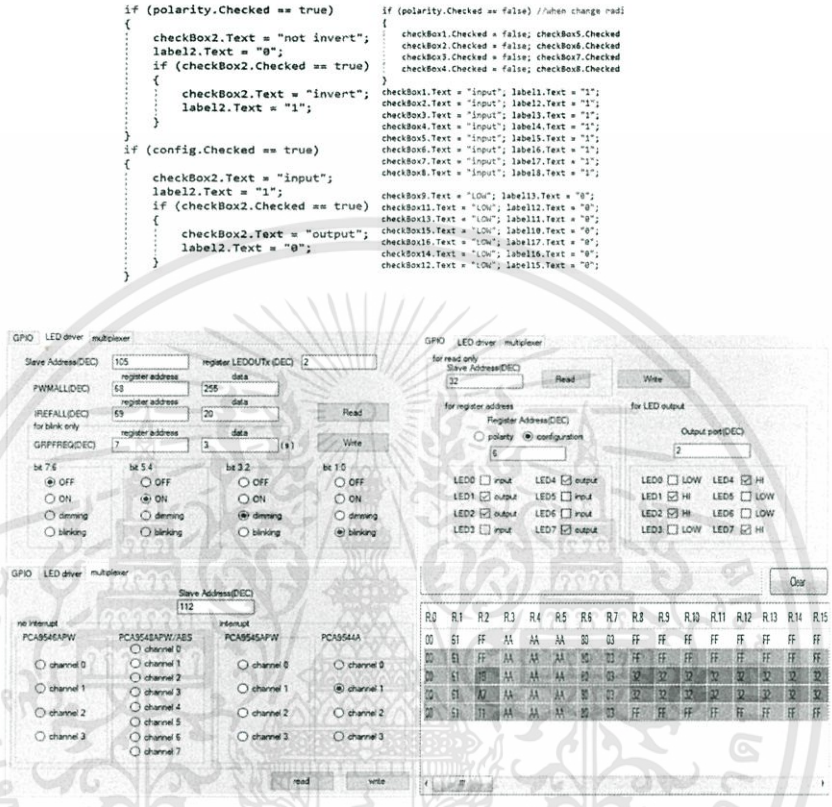
3.2.2.2 การอ่านรีจิสเตอร์ จะเริ่มต้นด้วยการใส่แอดเดรสของตัวงานตามด้วยจำนวนไบต์ที่ต้องการอ่านด้วยคำสั่ง Wire.requestFrom() และคำสั่งอ่านรีจิสเตอร์ด้วยคำสั่ง Wire.read() ดังรูป

```
Wire.requestFrom(AddrInt,1);//addr,byte
 ch0 = Wire.read(); // read first channel
Serial.println(ch0,HEX);
Wire.endTransmission();
```

ภาพที่ 3.15 ตัวอย่างโค้ดการอ่านรีจิสเตอร์

3.2.3 ศึกษาและทำความเข้าใจเกี่ยวกับวิธีการติดต่อสื่อสารระหว่างคอมพิวเตอร์และ Arduino ด้วย serial port และวิธีการออกแบบหน้าต่างผู้ใช้งาน (UI) ด้วยภาษา C#

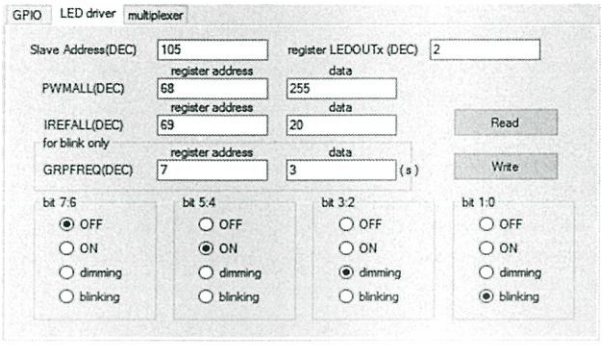
3.2.4 ออกแบบหน้าต่างผู้ใช้งาน โดยการเขียนโค้ดภาษา C# ในโปรแกรม visual basic ดังรูป



ภาพที่ 3.16 ตัวอย่างโค้ดการเขียน UI และตัวอย่าง UI

โดยในส่วนนี้ถูกสร้างขึ้นมาเพื่อรับคำสั่งจากผู้ใช้งานแล้วส่งไปยัง Arduino เพื่อทำการโปรแกรมคำสั่งให้ตัวงานทำงาน มี 3 รูปแบบดังนี้

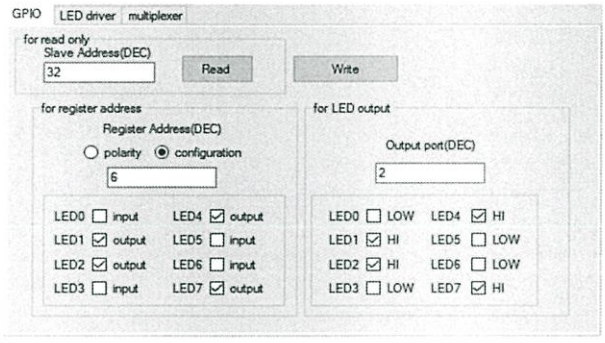
1) LED driver มีการทำงานดังนี้



ภาพที่ 3.17 UI ของ LED driver

- ใส่แอดเดรสของตัวงานที่ slave address
- เลือกรีจิสเตอร์ของเอาท์พุทที่ต้องการทดลองที่ register LEDOUTx โดยในแต่ละเอาท์พุทจะมีหลอดไฟ 4 ดวง
- ถ้าต้องการควบคุมความสว่างของหลอดไฟสามารถเลือกรีจิสเตอร์ได้ 2 แบบได้แก่
  - เลือกรีจิสเตอร์ที่ควบคุมระดับความสว่างของหลอดไฟและระดับความสว่างที่ PWMALL ระดับสว่างน้อยสุดไปถึงสว่างมากสุด (0-255) ในส่วนนี้จะใช้ Pulse Width Modulation (PWM) ในการควบคุมความสว่าง สัญญาณที่วัดได้จากขาตัวงานที่โปรแกรมคำสั่งจะเป็นรูปสัญญาณพัลส์
  - เลือกรีจิสเตอร์ที่ควบคุมระดับความสว่างของหลอดไฟและระดับความสว่างที่ IREFALL ระดับสว่างน้อยสุดไปถึงสว่างมากสุด (0-255) แต่ในส่วนนี้จะใช้กระแสในการควบคุมความสว่าง โดยค่ากระแสที่ได้จะขึ้นอยู่กับค่าความต้านทานที่ต่อภายนอกวงจร (REXT) สัญญาณที่วัดได้จากขาตัวงานที่โปรแกรมคำสั่งจะเป็นกราฟเส้นตรง
- ถ้าต้องการควบคุมการกระพริบของหลอดไฟให้เลือกรีจิสเตอร์ที่ GRPFREQ และช่วงเวลาในการกระพริบด้วย โดยสามารถคำนวณช่วงเวลาได้จากสูตรใน datasheet
- ลำดับสุดท้ายให้เลือกสถานะของหลอดไฟแต่ละดวงโดยสามารถเลือกได้ 4 สถานะ ได้แก่
  - หลอดไฟดับ
  - หลอดไฟติด (สว่างเต็มที่)
  - หลอดไฟที่ถูกควบคุมความสว่าง (หลอดจาง)
  - หลอดไฟกระพริบ
- ส่งคำสั่งด้วยการกดปุ่ม “Write” หรืออ่านรีจิสเตอร์ทั้งหมดด้วยการกดปุ่ม “Read”

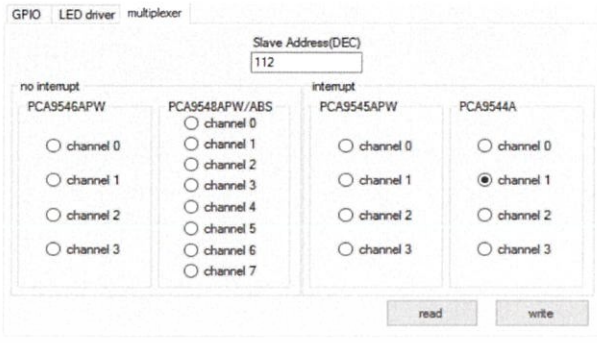
2) GPIO มีการทำงานดังนี้



ภาพที่ 3.18 UI ของ GPIO

- ใส่แอดเดรสของตัวงานที่ slave address
- เลือกรีจิสเตอร์โหมดการทำงาน แบ่งออกเป็น 2 โหมด ได้แก่
  - โหมด polarity : กำหนดสถานะของขางานว่าให้เป็น “inverted” หรือ “not inverted”ถ้าขางานถูกกำหนดให้เป็น inverted ค่ารีจิสเตอร์ที่อ่านได้จะตรงข้ามกับค่าอินพุทที่ใส่เข้าไป ยกตัวอย่างเช่น ป้อนอินพุทที่ขางานเป็น “10010110” อ่านค่าได้เป็น “01101001”
  - โหมด configuration : กำหนดสถานะของขางานว่าให้เป็นเอาท์พุทหรืออินพุท ถ้าขางานถูกกำหนดให้เป็นเอาท์พุท หลอดไฟดวงที่ถูกโปรแกรมคำสั่งจะทำงาน
- ในกรณีที่เลือกโหมด polarity ให้เลือกสถานะของขางานว่าให้เป็น “inverted” หรือ “not inverted” แต่ถ้าในกรณีที่เลือกโหมด configuration ให้เลือกสถานะของขางานว่าให้เป็น “input” หรือ “output”
- ในกรณีที่ต้องการทดสอบการทำงานของเอาท์พุทให้เลือกรีจิสเตอร์ของเอาท์พุทที่ Output port โดยขางานที่ถูกกำหนดให้เป็นเอาท์พุทจะแสดงผลตามที่ถูกโปรแกรมคำสั่ง แต่ขางานที่ถูกกำหนดให้เป็นอินพุทจะไม่แสดงผล
- เลือกสถานะของหลอดไฟแต่ละดวง โดยเลือกได้ 2 สถานะ ได้แก่
  - LOW คือหลอดดับ
  - HI คือหลอดติด
- ส่งคำสั่งด้วยการกดปุ่ม “Write” หรืออ่านรีจิสเตอร์ทั้งหมดด้วยการกดปุ่ม “Read”

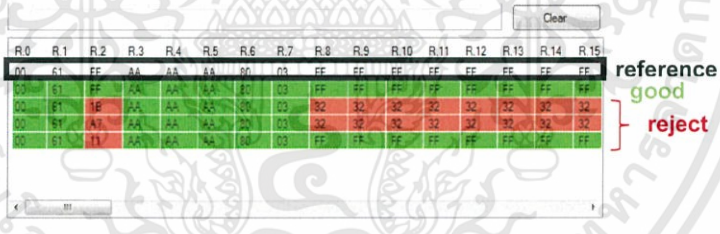
3) multiplexer มีการทำงานดังนี้



ภาพที่ 3.19 UI ของ multiplexer

- ใส่แอดเดรสของตัวงานที่ slave address
- เลือกตัวงานที่ต้องการทดลอง โดยในที่นี้จะแบ่งตัวงานออกเป็น 2 กลุ่ม คือตัวงานที่ไม่มีโหมด interrupt และตัวงานที่มีโหมด interrupt
- ลำดับสุดท้ายเลือกแชนเนลที่ต้องการให้เอาท์พุทแสดงผล
- ส่งคำสั่งด้วยการกดปุ่ม “Write” หรืออ่านรีจิสเตอร์แชนเนลที่ถูกเลือกด้วยการกดปุ่ม “Read”

3.2.5 ออกแบบตารางเปรียบเทียบรีจิสเตอร์ภายในตัวไอซีทั้งหมด



ภาพที่ 3.20 ตารางเปรียบเทียบรีจิสเตอร์

- บรรทัดแรกคือตัวงานที่ให้เป็นตัวอ้างอิงว่าเป็นตัวงานที่ดี
- บรรทัดถัดไปคือตัวงานอื่นๆ ที่นำมาทดลองว่าเป็นตัวงานที่ดีหรือเสียโดยการเปรียบเทียบรีจิสเตอร์ โดยที่สีเขียวคือรีจิสเตอร์ที่ถูกต้องและสีแดงคือรีจิสเตอร์ที่ผิดเพี้ยนไป
- เมื่อกดปุ่ม “Clear” รีจิสเตอร์ในตารางจะหายไปทั้งหมด

3.2.6 ทดสอบการติดต่อสื่อสารของโปรแกรมระหว่างคอมพิวเตอร์กับไอซี

3.2.7 ทดสอบโปรแกรมที่เขียนกับตัวบอร์ดจริง เก็บรายละเอียดและแก้ไขปัญหาที่เกิดขึ้น

3.2.8 ทดลองใช้งานจริงและจดบันทึกผลการทดลอง

บทที่ 4

ผลการวิจัย

4.1 โมดูลวงจรรักษาระดับแรงดัน

เมื่อป้อนอินพุท 9V ได้แรงดันเอาท์พุทค่าต่างๆ ดังนี้

ภาพที่ 4.1 แรงดันเอาท์พุท 3.3V , 5V และ 0-7V

4.2 โมดูลวงจรดึงกระแส

เมื่อป้อนอินพุท 9V วัดค่ากระแสที่ค่าความต้านทานต่างๆกันได้ ดังนี้

- ที่วงจรดึงกระแสแบบ current source

ตารางที่ 4.1 ผลการทดลองที่วงจรดึงกระแสแบบ current source

resistors (Ω)	Isource (mA)	Vadj (V)
680	0.48	8.660
	7.62	0.703
920	0.62	8.420
	5.67	0.687

- ที่วงจรดึงกระแสแบบ current sink

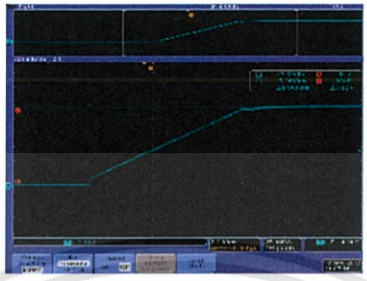
ตารางที่ 4.2 ผลการทดลองที่วงจรดึงกระแสแบบ current sink

resistors (Ω)	Isource (mA)	Vadj (V)
680	0	0.001
	9.19	8.280
1K	0	0.002
	6.23	8.300

4.3 โมดูลวงจรแรมป์

เมื่อวัดสัญญาณแรมป์และคาบเวลาที่ค่าความต้านทานต่างๆกันได้ผลการทดลอง ดังนี้

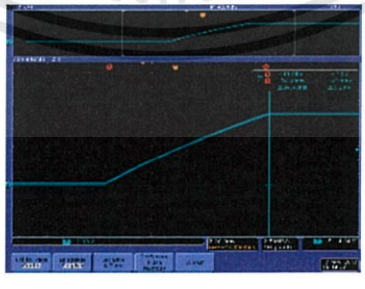
- ที่ power supply ทั่วไป ➡ T = 8.9 ms



ภาพที่ 4.2 สัญญาณแรมป์ที่ power supply ทั่วไป

- ramp adjust (R=25Ω) ➡ T = 6.5 ms
- ramp adjust (R=100Ω) ➡ T = 27.3 ms
- ramp adjust (R=294Ω) ➡ T = 94.4 ms

ภาพที่ 4.3 สัญญาณแรมป์เมื่อ R=25Ω และ R=100Ω



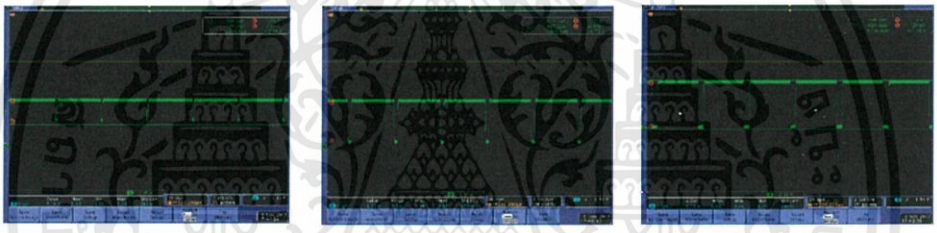
ภาพที่ 4.4 สัญญาณแรมป์เมื่อ R=294Ω

4.4 ผลการทดลองของไอซีชนิด LED driver

เมื่อวัดค่ากระแสและสัญญาณที่ขาของตัวงานที่ระดับความสว่างต่างกันได้ผลการทดลองดังนี้

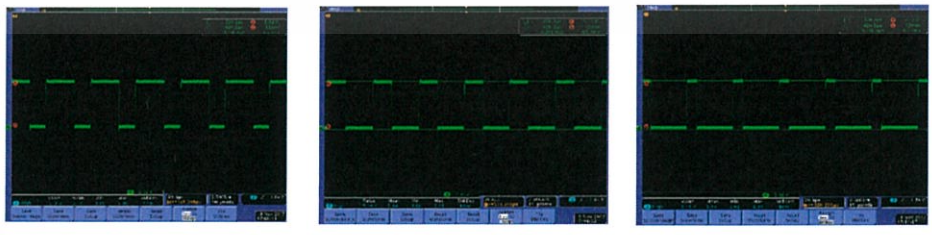
ตารางที่ 4.3 ค่ากระแสที่ขาของตัวงานที่ระดับความสว่างต่างๆกัน

ระดับความสว่าง	ค่ากระแส (mA)	ระดับความสว่าง	ค่ากระแส (mA)
10	0.08	70	1.32
15	0.19	80	1.52
20	0.35	90	1.71
30	0.55	100	1.90
40	0.75	150	2.86
50	0.94	200	3.81
60	1.13	240	4.58



ภาพที่ 4.5 สัญญาณ PWM ที่ระดับความสว่าง 10 , 30 และ 50

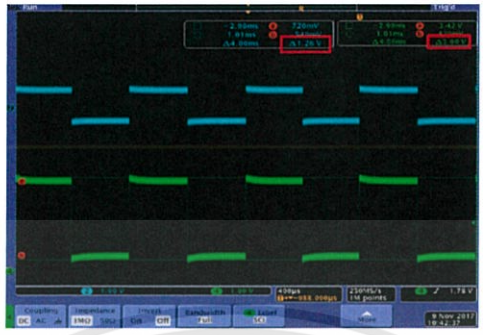
ภาพที่ 4.6 สัญญาณ PWM ที่ระดับความสว่าง 60 , 80 และ 100



ภาพที่ 4.7 สัญญาณ PWM ที่ระดับความสว่าง 150 , 200 และ 240

4.5 ผลการทดลองของไอซีชนิด voltage translator

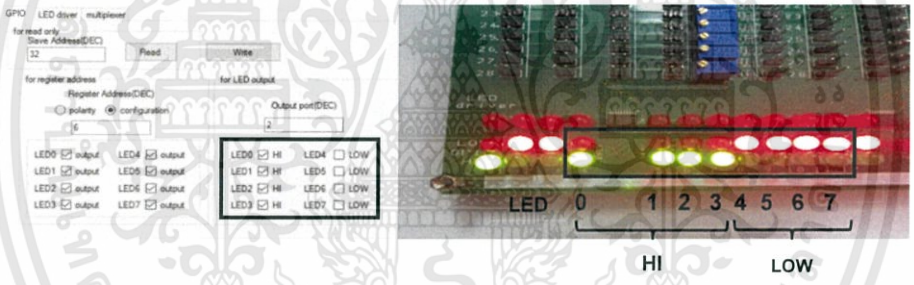
เมื่อแปลงระดับแรงดันจาก 1.2V เป็น 3.3V แล้ววัดสัญญาณด้วยออสซิลโลสโคป



ภาพที่ 4.8 ระดับแรงดันที่ 1.2V และ 3.3V

4.6 ผลการทดลองของไอซีชนิด GPIO

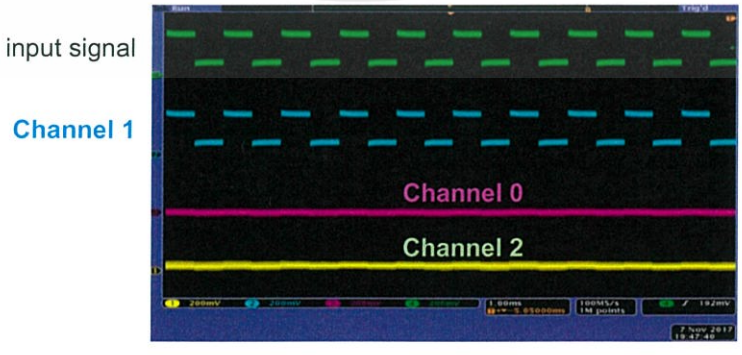
เมื่อทดสอบการทำงานของเอาท์พุทด้วยการเช็คสถานะของหลอดไฟ



ภาพที่ 4.9 ตัวอย่างการทำงานของ GPIO

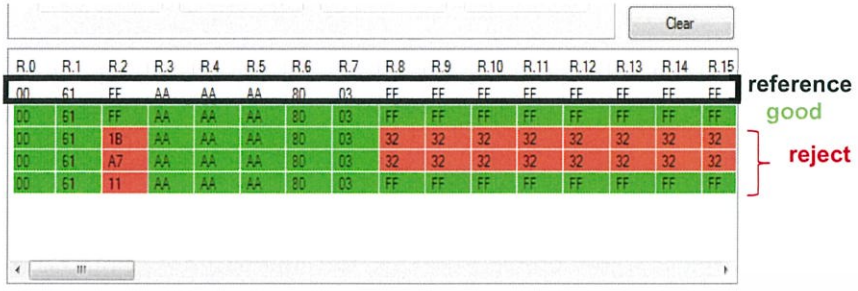
4.7 ผลการทดลองของไอซีชนิด multiplexer

เมื่อจ่ายสัญญาณรูปสี่เหลี่ยมเข้าไปที่บัส I2C แล้วเลือกให้เอาท์พุทออกที่แชนแนลที่ 1



ภาพที่ 4.10 ตัวอย่างการทำงานของ multiplexer

4.8 ผลการทดลองของตารางเปรียบเทียบรีจิสเตอร์



ภาพที่ 4.11 ตัวอย่างการทำงานของ multiplexer

จากภาพ จะเห็นได้ว่าตารางเปรียบเทียบรีจิสเตอร์นี้สามารถอ่านได้ทุกรีจิสเตอร์ในตัวงาน และสามารถอ่านตัวงานได้หลายตัวไม่จำกัด โดยจากการทดลองนั้นพบว่าค่ารีจิสเตอร์ที่อ่านได้มีความถูกต้องแม่นยำและเนื่องจากใช้สีในการเปรียบเทียบทำให้อ่านค่าได้ง่ายและใช้เวลาในการอ่านค่าเร็วกว่าวิธีการเดิม

บทที่ 5

สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

โครงงานนี้จัดทำขึ้นเพื่อพัฒนาวิธีการทดสอบการทำงานของไอซีให้ง่ายต่อการใช้งาน มีความถูกต้องแม่นยำรวมถึงแก้ไขปัญหาที่เกิดขึ้นด้วย โดยการออกแบบบอร์ดทดสอบการทำงานของไอซีที่เรียกว่า “Application Bench Board” และหน้าต่างผู้ใช้งาน (User Interface) เพื่ออำนวยความสะดวกในการใช้งาน โดยแบ่งส่วนการทำงานออกเป็น 2 ส่วน คือส่วนฮาร์ดแวร์และส่วนซอฟต์แวร์ ซึ่งทั้ง 2 ส่วนนี้จะทำงานควบคู่ไปด้วยกัน ในส่วนของฮาร์ดแวร์จะเริ่มต้นตั้งแต่การทำความเข้าใจเกี่ยวกับการทำงานของไอซีแต่ละชนิด การวางแผนระยะเวลาในการทำงาน กำหนดขอบเขตของโครงงาน การออกแบบลายวงจรและลงอุปกรณ์ ส่วนในส่วนของซอฟต์แวร์จะเริ่มจากการทำความเข้าใจเกี่ยวกับภาษา C เบื้องต้น วิธีการติดต่อสื่อสารระหว่างคอมพิวเตอร์กับ Arduino การติดต่อสื่อสารระหว่าง Arduino กับตัวงาน และวิธีการออกแบบหน้าต่างผู้ใช้งานด้วยภาษา C# จากนั้นนำทั้ง 2 ส่วนนี้มาทดสอบการทำงานเข้าด้วยกัน แก้ไขปัญหาที่เกิดขึ้นและบันทึกผลการทดลอง

จากผลการทดลองพบว่าในส่วนของโมดูลต่างๆ มีผลการทดลองที่ถูกต้องดังนี้ โมดูลวงจรรักษาระดับแรงดันมีค่าแรงดันเอาท์พุทตรงตามที่ออกแบบไว้ทั้งแบบแรงดันเอาท์พุทคงที่และแบบปรับค่าได้ โมดูลวงจรดึงกระแสทั้ง 2 แบบเมื่อเพิ่มค่าความต้านทานขึ้นค่ากระแสเอาท์พุทก็จะเพิ่มขึ้นด้วย และวงจรแรมป์สำหรับในเครื่องพาวเวอร์ซัพพลายทั่วไปคาบเวลาที่สัญญาณแรมป์ขึ้นจะอยู่ที่ประมาณ 9 ms ซึ่งสำหรับในวงจรนี้เมื่อเพิ่มค่าความต้านทานในส่วนของการควบคุมคาบเวลาของสัญญาณจะพบว่าคาบเวลาจะเพิ่มขึ้นด้วย ส่วนการทำงานของบอร์ดหลักเมื่อนำมาทดสอบการทำงานจริงกับไอซีแล้วพบว่าผลการทดลองถูกต้องดังนี้ เมื่อนำมาทดลองกับไอซีชนิด LED driver โดยวัดค่ากระแสและสัญญาณที่ระดับความสว่างต่างๆกันพบว่าเมื่อปรับระดับความสว่างให้อยู่ที่ระดับต่ำๆ จะทำให้กระแสที่ถูกดึงลงกราวนด์มีค่าน้อยมากจนทำให้ขางานมีสภาวะเป็น “HIGH” มากกว่า แต่ในทางกลับกันถ้าปรับระดับความสว่างให้อยู่ที่ระดับสูงๆ จะทำให้กระแสที่ถูกดึงลงกราวนด์มีค่ามากและขางานมีสภาวะเป็น “LOW” มากกว่า เมื่อนำมาทดลองกับไอซีชนิด voltage translator พบว่าการแปลงระดับแรงดันถูกต้องตามที่ต้องการ และเนื่องจากไอซีชนิดนี้มีหน้าที่ในการแปลงระดับแรงดันจึงไม่จำเป็นต้องมี UI ในการโปรแกรมคำสั่งการใช้งาน เมื่อนำมาทดลองกับไอซีชนิด GPIO พบว่าผลการทดลองเป็นไปตามคำสั่งที่โปรแกรมเข้าไป ผลลัพธ์ถูกต้องแม่นยำและง่ายต่อการบันทึกผล เมื่อนำมาทดลองกับไอซีชนิด multiplexer พบว่าค่าสัญญาณที่ออกมาตรงกับแชนแนลเอาท์พุทที่ถูกเลือก และสุดท้ายตารางเปรียบเทียบรีจิสเตอร์สามารถอ่านค่ารีจิสเตอร์ได้ทั้งหมดภายในตัวงาน และนำรีจิสเตอร์มาเปรียบเทียบกันระหว่างตัวดีและตัวเสียซึ่งมีค่าแม่นยำ ถูกต้องและเร็วมากกว่าวิธีการเดิมที่เคยทดสอบ

5.2 ข้อเสนอแนะ

เนื่องจากการจัดทำโครงงานนี้ในส่วนของฮาร์ดแวร์ได้ใช้โปรแกรมออกแบบลายวงจรที่ชื่อว่า designspark ซึ่งเป็นโปรแกรมที่ผู้จัดทำไม่เคยใช้มาก่อน จึงทำให้ใช้เวลาในการศึกษาการใช้โปรแกรมนี้ค่อนข้างมากและไม่ชำนาญในการใช้งานมากนัก ทำให้ระยะเวลาในการออกแบบลายวงจรที่วางแผนไว้ล่าช้าไปประมาณ 1-2 อาทิตย์ และเนื่องจากก่อนส่งพีซีบีไปให้ทางโรงงานกัดปรินท์นั้นทางผู้จัดทำต้องทำการตรวจทานรายละเอียดความถูกต้องอย่างรอบคอบและหลายครั้ง เนื่องจากค่าใช้จ่ายในส่วนนี้ค่อนข้างมากจึงไม่ควรปล่อยให้มีความผิดพลาดเกิดขึ้นได้ และช่องทางการติดต่อสื่อสารระหว่างผู้จัดทำกับทางโรงงานไม่ค่อยสะดวกมากนักจึงทำให้เวลาในส่วนนี้ล่าช้าไปด้วย ดังนั้นเวลาที่ผู้จัดทำจะเริ่มทำโครงงานแต่ละครั้งควรศึกษาหาข้อมูลเกี่ยวกับเรื่องที่จะจัดทำหรือศึกษาการใช้โปรแกรมที่จำเป็นต้องใช้มาก่อนเพื่อที่จะได้ลดระยะเวลาในการทำโครงงาน และควรคำนวณเวลาให้โครงงานเสร็จก่อนเวลาที่กำหนดเผื่อไว้สำหรับปัญหาต่างๆที่ผู้จัดทำไม่สามารถคาดการณ์ได้ล่วงหน้าไว้ด้วย

ส่วนในส่วนการทำงานของซอฟต์แวร์นั้นผู้ใช้งานมีความไม่ชำนาญเป็นอย่างมาก ทำให้ต้องใช้เวลาศึกษาและทำความเข้าใจในเรื่องของภาษา C เบื้องต้นรวมถึงวิธีการใช้งานของ Arduino ร่วมกับตัวงานด้วย อีกทั้งยังมีเรื่องของการติดต่อสื่อสารและโปรแกรมคำสั่งระหว่างคอมพิวเตอร์กับตัวงาน ซึ่งเป็นเรื่องใหม่ที่ผู้จัดทำไม่เคยศึกษามาก่อนและไม่มีในเนื้อหาที่เรียนจึงทำให้ใช้เวลาไปกับส่วนนี้พอสมควร ซึ่งทำให้ส่วนนี้ผู้จัดทำได้รับความรู้เป็นอย่างมากและสามารถนำเอาพื้นฐานวิชาไมโครโปรเซสเซอร์มาประยุกต์ใช้ในการจัดทำโครงงานนี้ได้ และทำให้ผู้จัดทำได้ทราบว่าในส่วนของซอฟต์แวร์นั้นมีส่วนสำคัญเป็นอย่างมากในการใช้ชีวิตประจำวันและการนำไปประยุกต์ใช้ในการทำงานในอนาคต ดังนั้นถึงแม้ว่าบางเรื่องจะไม่มีสอนภายในเนื้อหาที่เรียนทั้งหมดแต่เราก็ควรจะศึกษาหาความรู้เพิ่มเติมจากภายนอกเองด้วย

เอกสารอ้างอิง

1. admin. ไม่ปรากฏปีที่แต่ง. การป้องกันกระแสเกินสำหรับแหล่งจ่ายไฟฟ้า. 21 ตุลาคม 2560 , http://www.semi-journal.com

2. Aimagin Blog. (2014). การใช้งานพอร์ตสื่อสาร I2C. 15 ตุลาคม 2560 , http://aimagin.com/blog

3. Education. (2015). ส่วนจัดการสื่อประสานผู้ใช้(User Interface-management). 11 พฤศจิกายน 2560 , https://www.slideshare.net/tumetr/user-interfacemanagement

4. Jatuchai. (2009). ความรู้เรื่อง I2C ระบบบัสที่มีสัญญาณเพียง 2 เส้น. 15 ตุลาคม 2560 , www.eclubthai.com/board/index.php?topic=16140.0;wap2

5. krupattaraporn. ไม่ปรากฏปีที่แต่ง. อิเล็กทรอนิกส์เบื้องต้น. 25 ตุลาคม 2560 , https://sites.google.com/site/krupattaraporn/bth-thi-2-xilekthrxniks-beuxng-tn

6. Mountain A. (2017). I2C Communications. 15 ตุลาคม 2560 , www.arduitronics.com/article/47/i2c-communication-case-study-of-gy-30-ambient-light-sensor

7. Ratchanon. (2013). การส่งผ่านข้อมูลดิจิตอล. 3 พฤศจิกายน 2560 , https://sites.google.com/site/chaywu03/kar-sng-phan-khxmul-dicitxl

8. Thaimicrotron.com. (2015). การเชื่อมต่ออุปกรณ์แบบ I2C. 15 ตุลาคม 2560 , www.thaimicrotron.com/CCS-628/Referrence/I2CBUS.htm

9. VERANUCH. (2015). Serial Communication. 11 พฤศจิกายน 2560 , https://riverplusblog.com/2011/06/18/serial-communication/

10. วิกิพีเดีย. (2017). ทรานซิสเตอร์. 27 ตุลาคม 2560 , https://th.wikipedia.org/wiki/

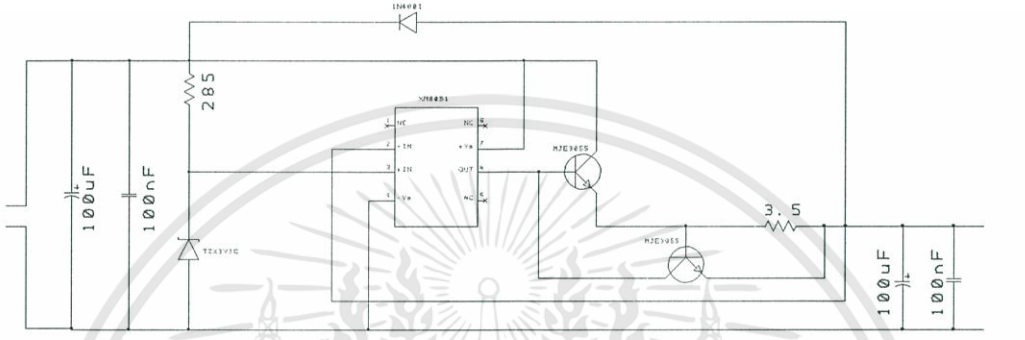
11. ไม่ปรากฏผู้แต่ง. (2014). วงจรรักษาระดับแรงดัน. 21 ตุลาคม 2560 , http://elecpnt.rmutl.ac.th/attachments/014_el_lesson3

ภาคผนวก ก

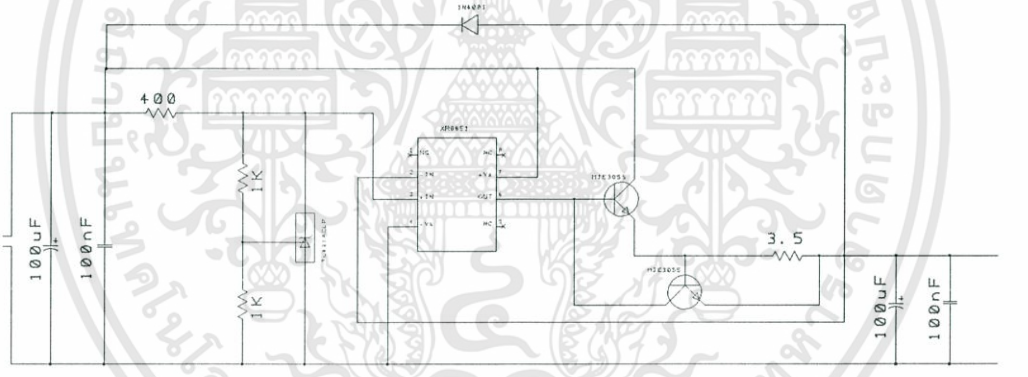
ส่วนของฮาร์ดแวร์ ประกอบด้วย schematic ลายวงจรและตัวบอร์ด

1. โมดูลวงจรรักษาระดับแรงดัน

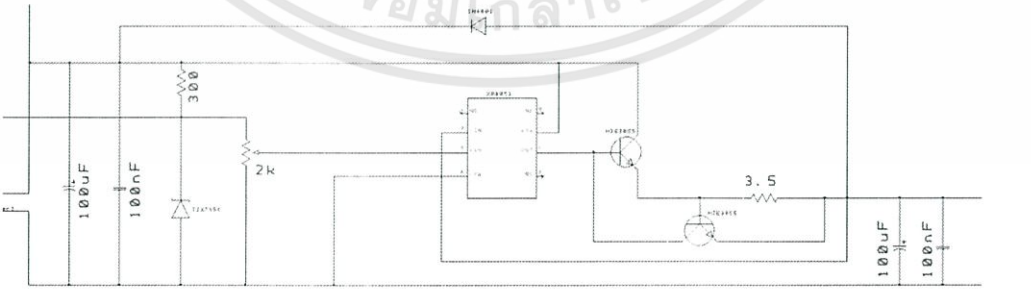
แรงดันคงที่ 3.3V

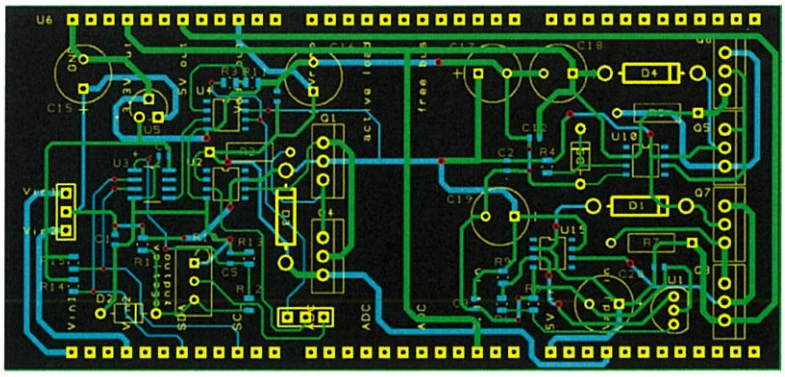


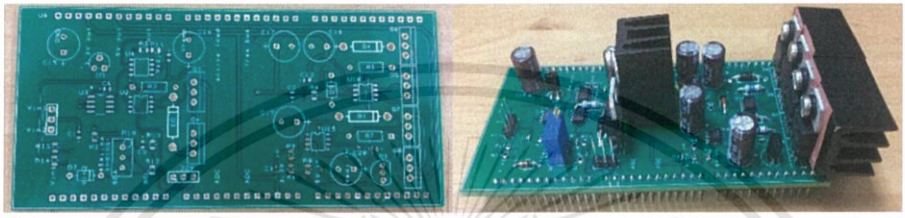
แรงดันคงที่ 5V



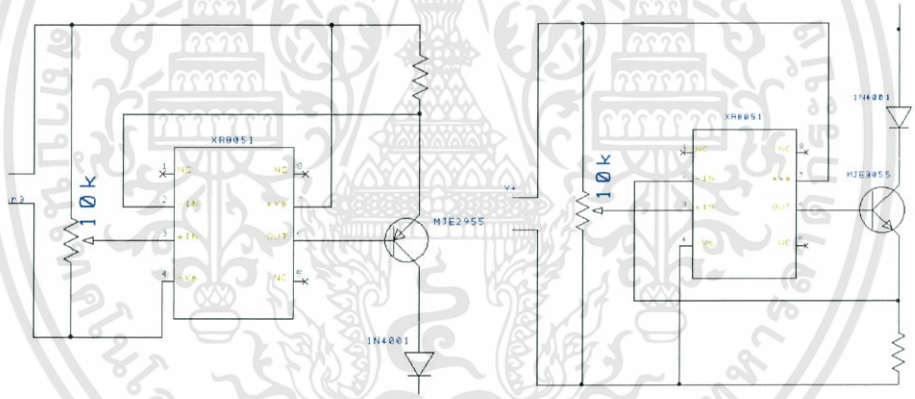
แรงดันปรับค่าได้ 0-7V

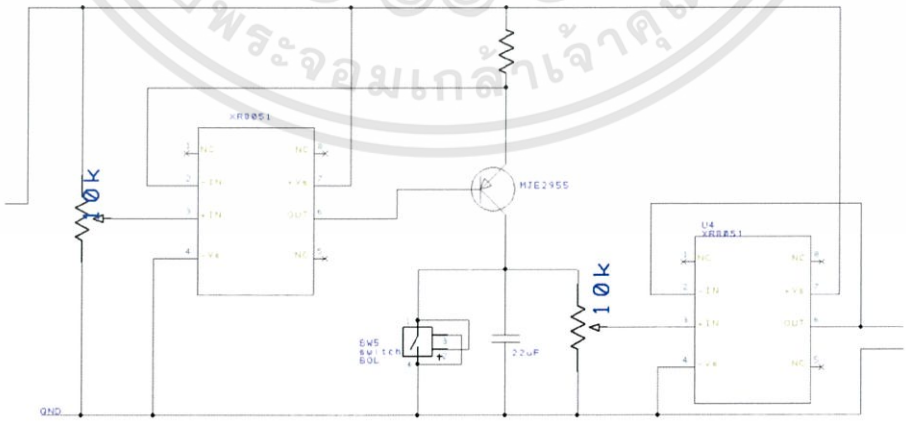


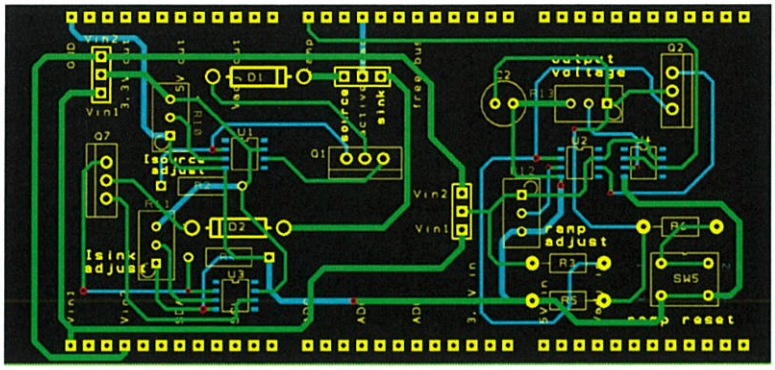


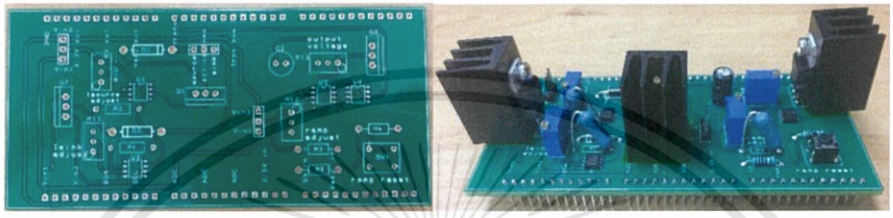


2. โมดูลวงจรดึงกระแสและวงจรแรมป์

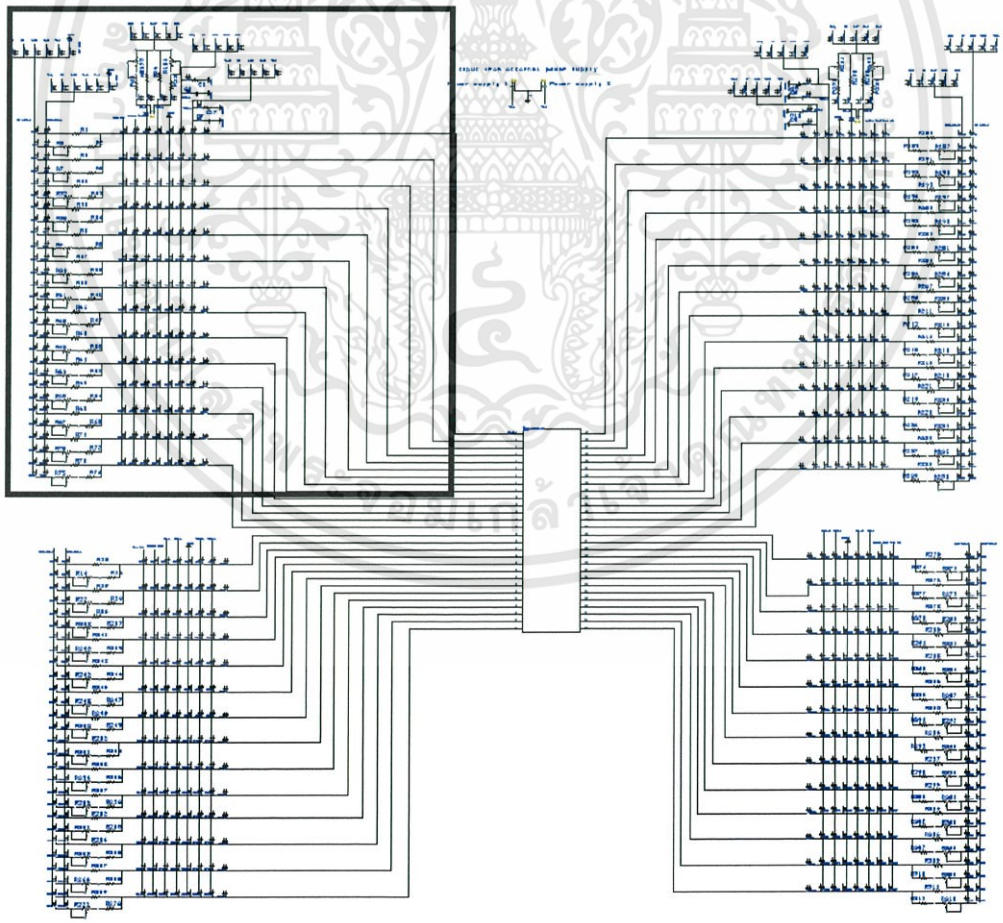




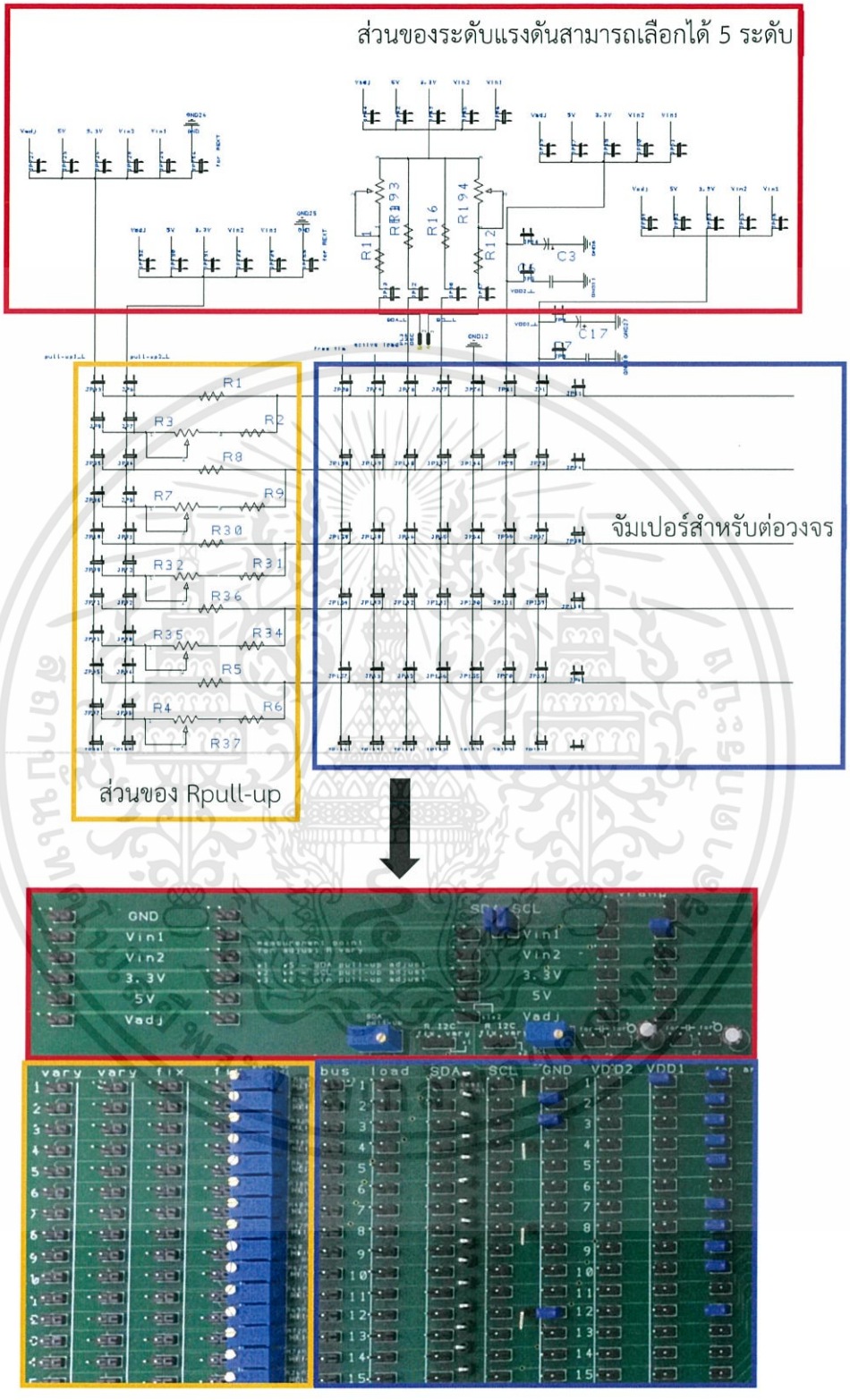




3. บอร์ดหลักในส่วนของการต่อวงจร

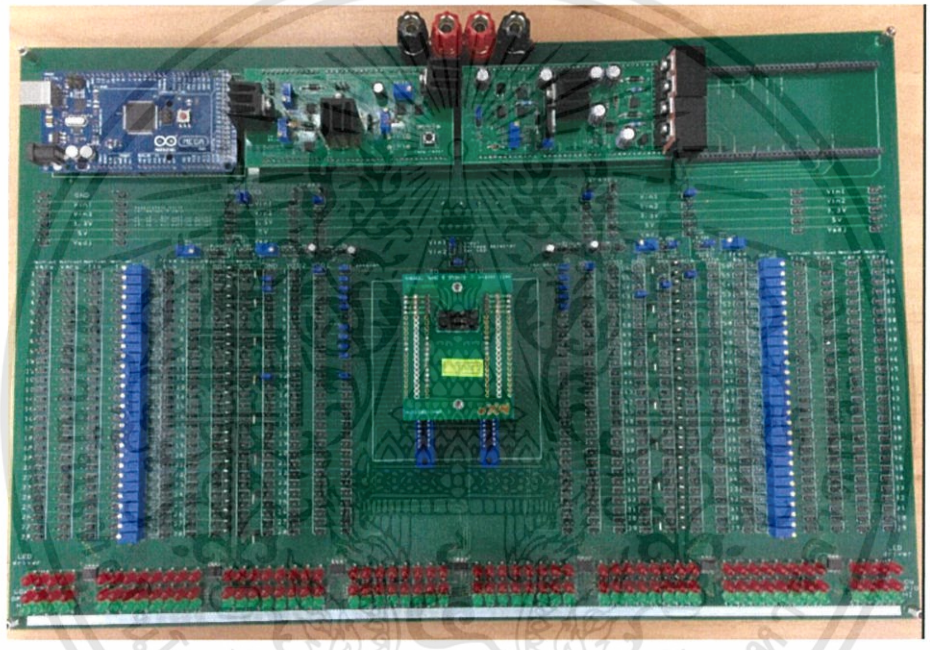


ภาพขยายในกรอบสี่เหลี่ยมจากภาพด้านบน



4. บอร์ดหลักในส่วนของส่วนแสดงผล

- แถวบนสำหรับ LED driver
- 2 แถวล่างสำหรับ GPIO
- แถบข้างล่างสีขาวสำหรับมาร์กจุดสังเกตหลอดไฟที่ต้องการทดลอง



บอร์ดสำเร็จรูป

ภาคผนวก ข

ส่วนของซอฟต์แวร์ ประกอบด้วยโค้ดการทำงานและหน้าต่างผู้ใช้งาน

- โค้ดการติดสื่อสารระหว่างคอมพิวเตอร์กับ Arduino

```
private SerialPort myport;
public Form1()
{
    InitializeComponent();
    myport = new SerialPort();

    string portName = "";
    foreach (string s in SerialPort.GetPortNames())
    {
        portName = s;
    }
    if (portName != "")
    {
        myport.PortName = portName;
        myport.BaudRate = 9600;
        myport.ReadTimeout = 2000;
        myport.WriteTimeout = 2000;
    }
    else
    {
        MessageBox.Show("No Arduino connected." + Er
        //Application.Exit();
    }
```

- ส่วนการรับข้อมูลจากคอมพิวเตอร์มายัง Arduino และแปลงข้อมูลจากสตริงมาเป็นตัวเลข

```
void loop()
{
  if (Serial.available())
  {
    myStr = Serial.readStringUntil('\n');
    String str = myStr.substring(4,1);
    String addr = myStr.substring(13,10);
    addr.toCharArray(SerChArr,addr.length()+1);
    AddrInt = atoi(SerChArr);
    String cntrl = myStr.substring(21,18);
    cntrl.toCharArray(SerChArr,cntrl.length()+1)
    CntrlInt = atoi(SerChArr);
    String data = myStr.substring(30,27);
    data.toCharArray(SerChArr,data.length()+1);
    DataInt = atoi(SerChArr);
```

```
if(str == "xxx")
{
  write_device();
  read_registers();
}
else if(str == "mux")
{
  write_mux();
  read_mux();
}
```

- โค้ดคำสั่งการสร้างตารางเปรียบเทียบรีจิสเตอร์

```
for (int j = 1; j < 100; j++)
{
    lvi.SubItems.Add(myport.ReadLine().PadLeft(3, '0'));
}
for (int i = 1; i < listView1.Items.Count; i++)
{
    for (int k = 0; k < 100; k++)
    {
        lvi.UseItemStyleForSubItems = false;
        if (listView1.Items[0].SubItems[k].Text == listView1.Items[i].SubItems[k].Text)
        {
            listView1.Items[i].SubItems[k].BackColor = Color.Chartreuse;
        }
        else if (listView1.Items[0].SubItems[k].Text != listView1.Items[i].SubItems[k].Text)
        {
            listView1.Items[i].SubItems[k].BackColor = Color.Tomato;
        }
    }
}
```

- โค้ดคำสั่งการทำงานของ LED driver และ GPIO

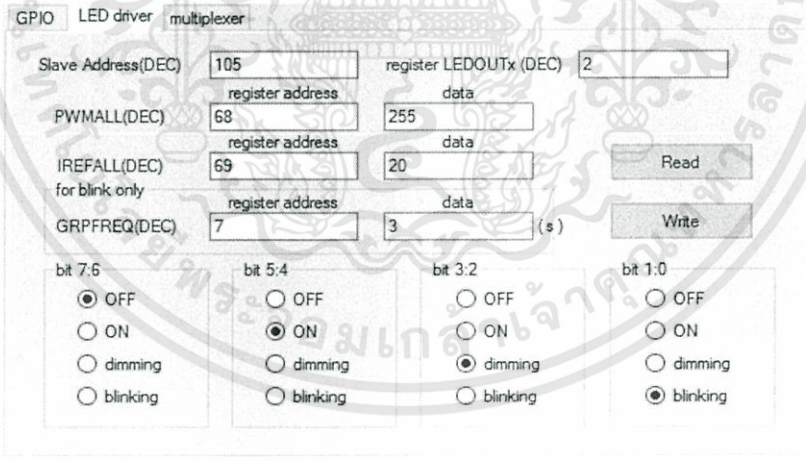
```
void read_registers()
{
  for(int i=0;i<100;i++)
    {
    Wire.beginTransmission(AddrInt);
    Wire.write(i);
    Wire.endTransmission();

    Wire.requestFrom(AddrInt,1);//addr,byte
     ch0 = Wire.read(); // read first channel
    Serial.println(ch0,HEX);
    Wire.endTransmission();
    }
}
```

```
void write_device()
{
    Wire.begin();
    Wire.beginTransmission(AddrInt);
    Wire.write(CntrlInt);  //registe
    Wire.write(DataInt); //data
    Wire.endTransmission();
}
```

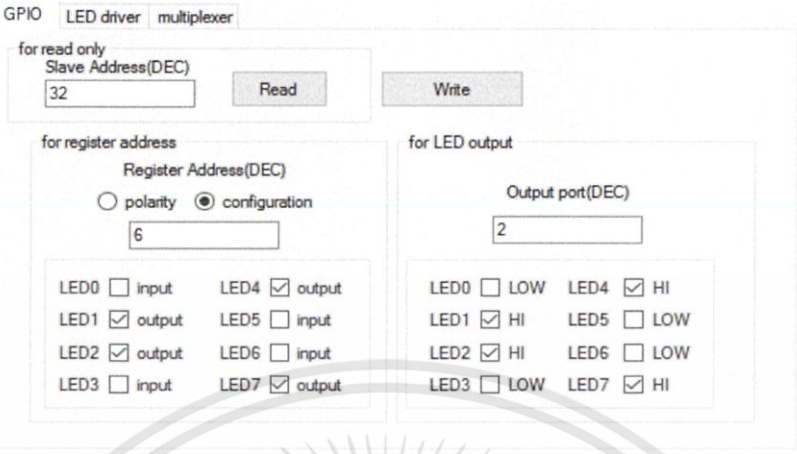
- โค้ดคำสั่งการทำงานของ multiplexer

```
void read_mux()
{
    Wire.requestFrom(AddrInt,1);//addr,byte
     ch0 = Wire.read(); // read first channel
    Serial.println(ch0,HEX);
    Wire.endTransmission();
}
void write_mux()
{
    Wire.begin();
    Wire.beginTransmission(AddrInt);
    Wire.write(CntrlInt);
    Wire.endTransmission();
}
```

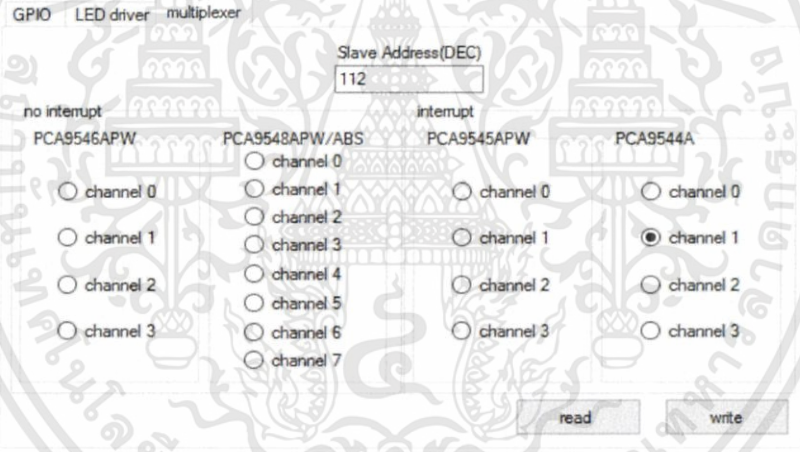
- หน้าต่างผู้ใช้งานของ LED driver



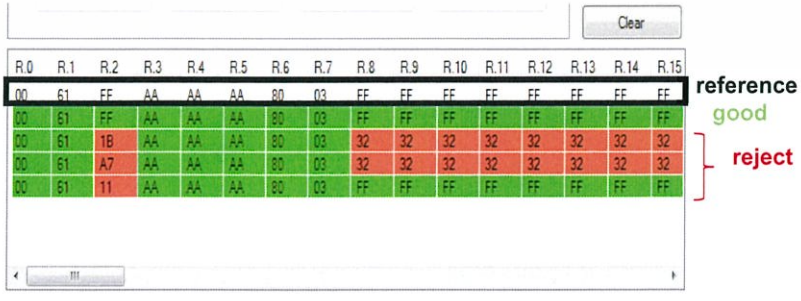
- หน้าต่างผู้ใช้งานของ GPIO



- หน้าต่างผู้ใช้งานของ multiplexer



- ตารางเปรียบเทียบรีจิสเตอร์



ภาคผนวก ค

- ตัวอย่างข้อมูลใน datasheet ที่จำเป็นต้องศึกษาของไอซีชนิด LED driver

PCA9955B

16-channel Fm+ I²C-bus 57 mA/20 V constant current LED driver

Rev. 2.1 — 2 May 2017 Product data sheet

2. Features and benefits

- 16 LED drivers. Each output programmable at:
  - Off
  - On
  - Programmable LED brightness
  - Programmable group dimming/blinking mixed with individual LED brightness
  - Programmable LED output delay to reduce EMI and surge currents
- Gradation control for all channels
  - Each channel can assign to one of four gradation control groups
  - Programmable gradation time and rate for ramp-up and/or ramp-down operations
  - Programmable step time (6-bit) from 0.5 ms (minimum) to 512 ms (maximum)
  - Programmable hold-on time after ramp-up and hold-off time after ramp-down (3-bit) from 0 s to 6 s
  - Programmable final ramp-up and hold-on current
  - Programmable brightness current output adjustment, either linear or exponential curve
- 16 constant current output channels can sink up to 57 mA, tolerate up to 20 V when OFF
- Output current adjusted through an external resistor (REXT input)
- Output current accuracy
  - ±4 % between output channels
  - ±6 % between PCA9955B devices
- Open/short load/overtemperature detection mode to detect individual LED errors
- 1 MHz Fast-mode Plus compatible I²C-bus interface with 30 mA high drive capability on SDA output for driving high capacitive buses
- 256-step (8-bit) linear programmable brightness per LED output varying from fully off (default) to maximum brightness fully ON using a 31.25 kHz PWM signal
- 256-step group brightness control allows general dimming (using a 122 Hz PWM signal) from fully off to maximum brightness (default)
- 256-step group blinking with frequency programmable from 15 Hz to 16.8 s and duty cycle from 0 % to 99.6 %
- Output state change programmable on the Acknowledge or the STOP condition to update outputs byte-by-byte or all at the same time (default to 'Change on STOP').
- Active LOW Output Enable ($\overline{OE}$) input pin allows for hardware blinking and dimming of the LEDs

- Three quinary hardware address pins allow 125 PCA9955B devices to be connected to the same I²C-bus and to be individually programmed
- 4 software programmable I²C-bus addresses (one LED Group Call address and three LED Sub Call addresses) allow groups of devices to be addressed at the same time in any combination (for example, one register used for 'All Call' so that all the PCA9955Bs on the I²C-bus can be addressed at the same time and the second register used for three different addresses so that 1/3 of all devices on the bus can be addressed at the same time in a group). Software enable and disable for each programmable I²C-bus address.
- Unique power-up default Sub Call address allows mixing of devices with different channel widths
- Software Reset feature (SWRST Call) allows the device to be reset through the I²C-bus
- 8 MHz internal oscillator requires no external components
- Internal power-on reset
- Noise filter on SDA/SCL inputs
- No glitch on LEDn outputs on power-up
- Low standby current
- Operating power supply voltage (V_{DD}) range of 3 V to 5.5 V
- 5.5 V tolerant inputs on non-LED pins
- –40 °C to +105 °C operation
- ESD protection exceeds 4000 V HBM per JESD22-A114
- Latch-up testing is done to JEDEC Standard JESD78 which exceeds 100 mA
- Packages offered: HTSSOP28

3. Applications

- Amusement products
- RGB or RGBA LED drivers
- LED status information
- LED displays
- LCD backlights
- Keypad backlights for cellular phones or handheld devices
- Fade-in and fade-out for breathlight control
- Automotive lighting (PCA9955BTW/Q900)

6.2 Pin description

Table 3. Pin description

Symbol	Pin	Type	Description
REXT	1	I	current set resistor input; resistor to ground
AD0	2	I	address input 0
AD1	3	I	address input 1
AD2	4	I	address input 2
$\overline{OE}$	5	I	active LOW output enable for LEDs
LED0	6	O	LED driver 0
LED1	7	O	LED driver 1
LED2	8	O	LED driver 2
LED3	9	O	LED driver 3
LED4	11	O	LED driver 4
LED5	12	O	LED driver 5
LED6	13	O	LED driver 6
LED7	14	O	LED driver 7
LED8	15	O	LED driver 8
LED9	16	O	LED driver 9
LED10	17	O	LED driver 10
LED11	18	O	LED driver 11
LED12	20	O	LED driver 12
LED13	21	O	LED driver 13
LED14	22	O	LED driver 14
LED15	23	O	LED driver 15
$\overline{RESET}$	25	I	active LOW reset input with external 10 kΩ pull-up resistor
SCL	26	I	serial clock line
SDA	27	I/O	serial data line
V_{SS}	10, 19, 24 [1]	ground	supply ground
V_{DD}	28	power supply	supply voltage

[1] HTSSOP28 package supply ground is connected to both V_{SS} pins and exposed center pad. V_{SS} pins must be connected to supply ground for proper device operation. For enhanced thermal, electrical, and board level performance, the exposed pad must be soldered to the board using a corresponding thermal pad on the board and for proper heat conduction through the board, thermal vias must be incorporated in the printed-circuit board in the thermal pad region.

7. Functional description

Refer to Figure 1 "Block diagram of PCA9955B".

7.1 Device addresses

Following a START condition, the bus master must output the address of the slave it is accessing.

For PCA9955B there are a maximum of 125 possible programmable addresses using the three quinary hardware address pins.

7.1.1 Regular I²C-bus slave address

The I²C-bus slave address of the PCA9955B is shown in Figure 3. The 7-bit slave address is determined by the quinary input pads AD0, AD1 and AD2. Each pad can have one of five states (GND, pull-up, floating, pull-down, and V_{DD}) based on how the input pad is connected on the board. At power-up or hardware/software reset, the quinary input pads are sampled and set the slave address of the device internally. To conserve power, once the slave address is determined, the quinary input pads are turned off and will not be sampled until the next time the device is power cycled. Table 4 lists the five possible connections for the quinary input pads along with the external resistor values that must be used.

Table 4. Quinary input pad connection

Pad connection (pins AD2, AD1, AD0)[1]	Mnemonic	External resistor (kΩ)	
		Min.	Max.
tie to ground	GND	0	17.9
resistor pull-down to ground	PD	34.8	270
open (floating)	FLT	503	∞
resistor pull-up to V_{DD}	PU	31.7	340
tie to V_{DD}	V_{DD}	0	22.1

[1] These AD[2:0] inputs must be stable before the supply V_{DD} to the chip.

Table 5 lists all 125 possible slave addresses of the device based on all combinations of the five states connected to three address input pins AD0, AD1 and AD2.

Table 5. I²C-bus slave address

Hardware selectable input pins			I²C-bus slave address for PCA9955B			
AD2	AD1	AD0	Decimal	Hexadecimal	Binary (A[6:0])	Address (R/W = 0)
GND	GND	GND	1	01	0000001[1]	02h
GND	GND	PD	2	02	0000010[1]	04h
GND	GND	FLT	3	03	0000011[1]	06h
GND	GND	PU	4	04	0000100[1]	08h
GND	GND	V_{DD}	5	05	0000101[1]	0Ah
GND	PD	GND	6	06	0000110[1]	0Ch
GND	PD	PD	7	07	0000111[1]	0Eh

Table 5. I²C-bus slave address ...*continued*

Hardware selectable input pins			I²C-bus slave address for PCA9955B			
AD2	AD1	AD0	Decimal	Hexadecimal	Binary (A[6:0])	Address (R/$\overline{W}$ = 0)
GND	PD	FLT	8	08	0001000	10h
GND	PD	PU	9	09	0001001	12h
GND	PD	V_{DD}	10	0A	0001010	14h
GND	FLT	GND	11	0B	0001011	16h
GND	FLT	PD	12	0C	0001100	18h
GND	FLT	FLT	13	0D	0001101	1Ah
GND	FLT	PU	14	0E	0001110	1Ch
GND	FLT	V_{DD}	15	0F	0001111	1Eh
GND	PU	GND	16	10	0010000	20h
GND	PU	PD	17	11	0010001	22h
GND	PU	FLT	18	12	0010010	24h
GND	PU	PU	19	13	0010011	26h
GND	PU	V_{DD}	20	14	0010100	28h
GND	V_{DD}	GND	21	15	0010101	2Ah
GND	V_{DD}	PD	22	16	0010110	2Ch
GND	V_{DD}	FLT	23	17	0010111	2Eh
GND	V_{DD}	PU	24	18	0011000	30h
GND	V_{DD}	V_{DD}	25	19	0011001	32h
PD	GND	GND	26	1A	0011010	34h
PD	GND	PD	27	1B	0011011	36h
PD	GND	FLT	28	1C	0011100	38h
PD	GND	PU	29	1D	0011101	3Ah
PD	GND	V_{DD}	30	1E	0011110	3Ch
PD	PD	GND	31	1F	0011111	3Eh
PD	PD	PD	32	20	0100000	40h
PD	PD	FLT	33	21	0100001	42h
PD	PD	PU	34	22	0100010	44h
PD	PD	V_{DD}	35	23	0100011	46h
PD	FLT	GND	36	24	0100100	48h
PD	FLT	PD	37	25	0100101	4Ah
PD	FLT	FLT	38	26	0100110	4Ch
PD	FLT	PU	39	27	0100111	4Eh
PD	FLT	V_{DD}	40	28	0101000	50h
PD	PU	GND	41	29	0101001	52h
PD	PU	PD	42	2A	0101010	54h
PD	PU	FLT	43	2B	0101011	56h
PD	PU	PU	44	2C	0101100	58h
PD	PU	V_{DD}	45	2D	0101101	5Ah
PD	V_{DD}	GND	46	2E	0101110	5Ch
PD	V_{DD}	PD	47	2F	0101111	5Eh

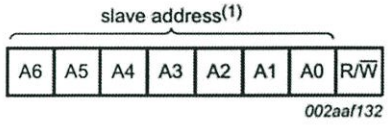
Table 5. I²C-bus slave address ...*continued*

Hardware selectable input pins			I²C-bus slave address for PCA9955B			
AD2	AD1	AD0	Decimal	Hexadecimal	Binary (A[6:0])	Address ($\overline{R/W}$ = 0)
PD	V_{DD}	FLT	48	30	0110000	60h
PD	V_{DD}	PU	49	31	0110001	62h
PD	V_{DD}	V_{DD}	50	32	0110010	64h
FLT	GND	GND	51	33	0110011	66h
FLT	GND	PD	52	34	0110100	68h
FLT	GND	FLT	53	35	0110101	6Ah
FLT	GND	PU	54	36	0110110	6Ch
FLT	GND	V_{DD}	55	37	0110111	6Eh
FLT	PD	GND	56	38	0111000	70h
FLT	PD	PD	57	39	0111001	72h
FLT	PD	FLT	58	3A	0111010	74h
FLT	PD	PU	59	3B	0111011	76h
FLT	PD	V_{DD}	60	3C	0111100	78h
FLT	FLT	GND	61	3D	0111101	7Ah
FLT	FLT	PD	62	3E	0111110	7Ch
FLT	FLT	FLT	63	3F	0111111	7Eh
FLT	FLT	PU	64	40	1000000	80h
FLT	FLT	V_{DD}	65	41	1000001	82h
FLT	PU	GND	66	42	1000010	84h
FLT	PU	PD	67	43	1000011	86h
FLT	PU	FLT	68	44	1000100	88h
FLT	PU	PU	69	45	1000101	8Ah
FLT	PU	V_{DD}	70	46	1000110	8Ch
FLT	V_{DD}	GND	71	47	1000111	8Eh
FLT	V_{DD}	PD	72	48	1001000	90h
FLT	V_{DD}	FLT	73	49	1001001	92h
FLT	V_{DD}	PU	74	4A	1001010	94h
FLT	V_{DD}	V_{DD}	75	4B	1001011	96h
PU	GND	GND	76	4C	1001100	98h
PU	GND	PD	77	4D	1001101	9Ah
PU	GND	FLT	78	4E	1001110	9Ch
PU	GND	PU	79	4F	1001111	9Eh
PU	GND	V_{DD}	80	50	1010000	A0h
PU	PD	GND	81	51	1010001	A2h
PU	PD	PD	82	52	1010010	A4h
PU	PD	FLT	83	53	1010011	A6h
PU	PD	PU	84	54	1010100	A8h
PU	PD	V_{DD}	85	55	1010101	AAh
PU	FLT	GND	86	56	1010110	ACh
PU	FLT	PD	87	57	1010111	AEh

Table 5. I²C-bus slave address ...continued

Hardware selectable input pins			I²C-bus slave address for PCA9955B			
AD2	AD1	AD0	Decimal	Hexadecimal	Binary (A[6:0])	Address (R/$\overline{W}$ = 0)
PU	FLT	FLT	88	58	1011000	B0h
PU	FLT	PU	89	59	1011001	B2h
PU	FLT	V_{DD}	90	5A	1011010	B4h
PU	PU	GND	91	5B	1011011	B6h
PU	PU	PD	92	5C	1011100	B8h
PU	PU	FLT	93	5D	1011101	BAh
PU	PU	PU	94	5E	1011110	BCh
PU	PU	V_{DD}	95	5F	1011111	BEh
PU	V_{DD}	GND	96	60	1100000	C0h
PU	V_{DD}	PD	97	61	1100001	C2h
PU	V_{DD}	FLT	98	62	1100010	C4h
PU	V_{DD}	PU	99	63	1100011	C6h
PU	V_{DD}	V_{DD}	100	64	1100100	C8h
V_{DD}	GND	GND	101	65	1100101	CAh
V_{DD}	GND	PD	102	66	1100110	CCh
V_{DD}	GND	FLT	103	67	1100111	CEh
V_{DD}	GND	PU	104	68	1101000	D0h
V_{DD}	GND	V_{DD}	105	69	1101001	D2h
V_{DD}	PD	GND	106	6A	1101010	D4h
V_{DD}	PD	PD	107	6B	1101011	D6h
V_{DD}	PD	FLT	108	6C	1101100	D8h
V_{DD}	PD	PU	109	6D	1101101	DAh
V_{DD}	PD	V_{DD}	110	6E	1101110	DCh
V_{DD}	FLT	GND	111	6F	1101111	DEh
V_{DD}	FLT	PD	112	70	1110000	E0h
V_{DD}	FLT	FLT	113	71	1110001	E2h
V_{DD}	FLT	PU	114	72	1110010	E4h
V_{DD}	FLT	V_{DD}	115	73	1110011	E6h
V_{DD}	PU	GND	116	74	1110100	E8h
V_{DD}	PU	PD	117	75	1110101	EAh
V_{DD}	PU	FLT	118	76	1110110	ECh
V_{DD}	PU	PU	119	77	1110111	EEh
V_{DD}	PU	V_{DD}	120	78	1111000[1]	F0h
V_{DD}	V_{DD}	GND	121	79	1111001[1]	F2h
V_{DD}	V_{DD}	PD	122	7A	1111010[1]	F4h
V_{DD}	V_{DD}	FLT	123	7B	1111011[1]	F6h
V_{DD}	V_{DD}	PU	124	7C	1111100[1]	F8h
V_{DD}	V_{DD}	V_{DD}	125	7D	1111101[1]	FAh

[1] See 'Remark' below.



(1) This slave address must match one of the 125 internal addresses as shown in Table 5

Fig 3. **PCA9955B slave address**

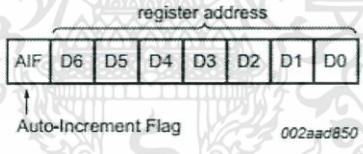
The last bit of the address byte defines the operation to be performed. When set to logic 1 a read is selected, while a logic 0 selects a write operation.

7.2 Control register

Following the successful acknowledgement of the slave address, LED All Call address or LED Sub Call address, the bus master sends a byte to the PCA9955B, which is stored in the Control register.

The lowest 7 bits are used as a pointer to determine which register is accessed (D[6:0]). The highest bit is used as Auto-Increment Flag (AIF).

This bit along with the MODE1 register bit 5 and bit 6 provide the Auto-Increment feature.



reset state = 80h

Remark: The Control register does not apply to the Software Reset I²C-bus address

Fig 4. **Control register**

When the Auto-Increment Flag is set (AIF = logic 1), the seven low-order bits of the Control register are automatically incremented after a read or write. This allows the user to program the registers sequentially. Four different types of Auto-Increment are possible, depending on AI1 and AI0 values of MODE1 register.

Table 6. **Auto-Increment options**

AIF	AI1[1]	AI0[1]	Function
0	0	0	no Auto-Increment
1	0	0	Auto-Increment for registers (00h to 43h). D[6:0] roll over to 00h after the last register 43h is accessed.
1	0	1	Auto-Increment for individual brightness registers only (08h to 17h). D[6:0] roll over to 08h after the last register (17h) is accessed.
1	1	0	Auto-Increment for MODE1 to IREF15 control registers (00h to 27h). D[6:0] roll over to 00h after the last register (27h) is accessed.
1	1	1	Auto-Increment for global control registers and individual brightness registers (06h to 17h). D[6:0] roll over to 06h after the last register (17h) is accessed.

[1] AI1 and AI0 come from MODE1 register.

7.3 Register definitions

Table 7. Register summary

Register number (hex)	D6	D5	D4	D3	D2	D1	D0	Name	Type	Function
00h	0	0	0	0	0	0	0	MODE1	read/write	Mode register 1
01h	0	0	0	0	0	0	1	MODE2	read/write	Mode register 2
02h	0	0	0	0	0	1	0	LEDOUT0	read/write	LED output state 0
03h	0	0	0	0	0	1	1	LEDOUT1	read/write	LED output state 1
04h	0	0	0	0	1	0	0	LEDOUT2	read/write	LED output state 2
05h	0	0	0	0	1	0	1	LEDOUT3	read/write	LED output state 3
06h	0	0	0	0	1	1	0	GRPPWM	read/write	group duty cycle control
07h	0	0	0	0	1	1	1	GRPFREQ	read/write	group frequency
08h	0	0	0	1	0	0	0	PWM0	read/write	brightness control LED0
09h	0	0	0	1	0	0	1	PWM1	read/write	brightness control LED1
0Ah	0	0	0	1	0	1	0	PWM2	read/write	brightness control LED2
0Bh	0	0	0	1	0	1	1	PWM3	read/write	brightness control LED3
0Ch	0	0	0	1	1	0	0	PWM4	read/write	brightness control LED4
0Dh	0	0	0	1	1	0	1	PWM5	read/write	brightness control LED5
0Eh	0	0	0	1	1	1	0	PWM6	read/write	brightness control LED6
0Fh	0	0	0	1	1	1	1	PWM7	read/write	brightness control LED7
10h	0	0	1	0	0	0	0	PWM8	read/write	brightness control LED8
11h	0	0	1	0	0	0	1	PWM9	read/write	brightness control LED9
12h	0	0	1	0	0	1	0	PWM10	read/write	brightness control LED10
13h	0	0	1	0	0	1	1	PWM11	read/write	brightness control LED11
14h	0	0	1	0	1	0	0	PWM12	read/write	brightness control LED12
15h	0	0	1	0	1	0	1	PWM13	read/write	brightness control LED13
16h	0	0	1	0	1	1	0	PWM14	read/write	brightness control LED14
17h	0	0	1	0	1	1	1	PWM15	read/write	brightness control LED15
18h	0	0	1	1	0	0	0	IREF0	read/write	output gain control register 0
19h	0	0	1	1	0	0	1	IREF1	read/write	output gain control register 1
1Ah	0	0	1	1	0	1	0	IREF2	read/write	output gain control register 2
1Bh	0	0	1	1	0	1	1	IREF3	read/write	output gain control register 3
1Ch	0	0	1	1	1	0	0	IREF4	read/write	output gain control register 4
1Dh	0	0	1	1	1	0	1	IREF5	read/write	output gain control register 5
1Eh	0	0	1	1	1	1	0	IREF6	read/write	output gain control register 6
1Fh	0	0	1	1	1	1	1	IREF7	read/write	output gain control register 7
20h	0	1	0	0	0	0	0	IREF8	read/write	output gain control register 8
21h	0	1	0	0	0	0	1	IREF9	read/write	output gain control register 9
22h	0	1	0	0	0	1	0	IREF10	read/write	output gain control register 10
23h	0	1	0	0	0	1	1	IREF11	read/write	output gain control register 11
24h	0	1	0	0	1	0	0	IREF12	read/write	output gain control register 12
25h	0	1	0	0	1	0	1	IREF13	read/write	output gain control register 13
26h	0	1	0	0	1	1	0	IREF14	read/write	output gain control register 14
27h	0	1	0	0	1	1	1	IREF15	read/write	output gain control register 15
28h	0	1	0	1	0	0	0	RAMP_RATE_GRP0	read/write	ramp enable and rate control for group 0
29h	0	1	0	1	0	0	1	STEP_TIME_GRP0	read/write	step time control for group 0
2Ah	0	1	0	1	0	1	0	HOLD_CNTL_GRP0	read/write	hold ON/OFF time control for group 0
2Bh	0	1	0	1	0	1	1	IREF_GRP0	read/write	output gain control for group 0
2Ch	0	1	0	1	1	0	0	RAMP_RATE_GRP1	read/write	ramp enable and rate control for group 1
2Dh	0	1	0	1	1	0	1	STEP_TIME_GRP1	read/write	step time control for group 1
2Eh	0	1	0	1	1	1	0	HOLD_CNTL_GRP1	read/write	hold ON/OFF time control for group 1
2Fh	0	1	0	1	1	1	1	IREF_GRP1	read/write	output gain control for group 1
30h	0	1	1	0	0	0	0	RAMP_RATE_GRP2	read/write	ramp enable and rate control for group 2
31h	0	1	1	0	0	0	1	STEP_TIME_GRP2	read/write	step time control for group 2
32h	0	1	1	0	0	1	0	HOLD_CNTL_GRP2	read/write	hold ON/OFF time control for group 2

Table 7. Register summary ...continued

Register number (hex)	D6	D5	D4	D3	D2	D1	D0	Name	Type	Function
33h	0	1	1	0	0	1	1	IREF_GRP2	read/write	output gain control for group 2
34h	0	1	1	0	1	0	0	RAMP_RATE_GRP3	read/write	ramp enable and rate control for group 3
35h	0	1	1	0	1	0	1	STEP_TIME_GRP3	read/write	step time control for group 3
36h	0	1	1	0	1	1	0	HOLD_CNTL_GRP3	read/write	hold ON/OFF time control for group 3
37h	0	1	1	0	1	1	1	IREF_GRP3	read/write	output gain control for group 3
38h	0	1	1	1	0	0	0	GRAD_MODE_SEL0	read/write	gradation mode select register for channel 7 to channel 0
39h	0	1	1	1	0	0	1	GRAD_MODE_SEL1	read/write	gradation mode select register for channel 15 to channel 8
3Ah	0	1	1	1	0	1	0	GRAD_GRP_SEL0	read/write	gradation group select for channel 3 to channel 0
3Bh	0	1	1	1	0	1	1	GRAD_GRP_SEL1	read/write	gradation group select for channel 7 to channel 4
3Ch	0	1	1	1	1	0	0	GRAD_GRP_SEL2	read/write	gradation group select for channel 11 to channel 8
3Dh	0	1	1	1	1	0	1	GRAD_GRP_SEL3	read/write	gradation group select for channel 15 to channel 12
3Eh	0	1	1	1	1	1	0	GRAD_CNTL	read/write	gradation control register for all four groups
3Fh	0	1	1	1	1	1	1	OFFSET	read/write	Offset/delay on LEDn outputs
40h	1	0	0	0	0	0	0	SUBADR1	read/write	I²C-bus subaddress 1
41h	1	0	0	0	0	0	1	SUBADR2	read/write	I²C-bus subaddress 2
42h	1	0	0	0	0	1	0	SUBADR3	read/write	I²C-bus subaddress 3
43h	1	0	0	0	0	1	1	ALLCALLADR	read/write	All Call I²C-bus address
44h	1	0	0	0	1	0	0	PWMALL	write only	brightness control for all LEDn
45h	1	0	0	0	1	0	1	IREFALL	write only	output gain control for all registers IREF0 to IREF15
46h	1	0	0	0	1	1	0	EFLAG0	read only	output error flag 0
47h	1	0	0	0	1	1	1	EFLAG1	read only	output error flag 1
48h	1	0	0	1	0	0	0	EFLAG2	read only	output error flag 2
49h	1	0	0	1	0	0	1	EFLAG3	read only	output error flag 3
4Ah to 7Fh	-	-	-	-	-	-	-	reserved	read only	not used[1]

[1] Reserved registers should not be written to and will always read back as zeros.

7.3.1 MODE1 — Mode register 1

Table 8. MODE1 - Mode register 1 (address 00h) bit description
*Legend: * default value.*

Bit	Symbol	Access	Value	Description
7	AIF	read only	0	Register Auto-Increment disabled.
			1*	Register Auto-Increment enabled.
6	AI1	R/W	0*	Auto-Increment bit 1 = 0. Auto-increment range as defined in Table 6.
			1	Auto-Increment bit 1 = 1. Auto-increment range as defined in Table 6.
5	AI0	R/W	0*	Auto-Increment bit 0 = 0. Auto-increment range as defined in Table 6.
			1	Auto-Increment bit 0 = 1. Auto-increment range as defined in Table 6.
4	SLEEP	R/W	0*	Normal mode[1].
			1	Low power mode. Oscillator off[2][3].
3	SUB1	R/W	0	PCA9955B does not respond to I²C-bus subaddress 1.
			1*	PCA9955B responds to I²C-bus subaddress 1.
2	SUB2	R/W	0*	PCA9955B does not respond to I²C-bus subaddress 2.
			1	PCA9955B responds to I²C-bus subaddress 2.
1	SUB3	R/W	0*	PCA9955B does not respond to I²C-bus subaddress 3.
			1	PCA9955B responds to I²C-bus subaddress 3.
0	ALLCALL	R/W	0	PCA9955B does not respond to LED All Call I²C-bus address.
			1*	PCA9955B responds to LED All Call I²C-bus address.

[1] It takes 500 µs max. for the oscillator to be up and running once SLEEP bit has been set to logic 0. Timings on LEDn outputs are not guaranteed if PWMx, GRPPWM or GRPFREQ registers are accessed within the 500 µs window.

[2] No blinking, dimming or gradation control is possible when the oscillator is off.

[3] The device must be reset if the LED driver output state is set to LDRx=11 after the device is set back to Normal mode.

7.3.2 MODE2 — Mode register 2

Table 9. MODE2 - Mode register 2 (address 01h) bit description
*Legend: * default value.*

Bit	Symbol	Access	Value	Description
7	OVERTEMP	read only	0*	O.K.
			1	overtemperature condition
6	ERROR	read only	0*	no error at LED outputs
			1	any open or short-circuit detected in error flag registers (EFLAGn)
5	DMBLNK	R/W	0*	group control = dimming
			1	group control = blinking
4	CLRERR	write only	0*	self clear after write '1'
			1	Write '1' to clear all error status bits in EFLAGn register and ERROR (bit 6). The EFLAGn and ERROR bit sets to '1' if open or short-circuit is detected again.
3	OCH	R/W	0*	outputs change on STOP condition
			1	outputs change on ACK
2	EXP_EN	R/W	0*	linear adjustment for gradation control
			1	exponential adjustment for gradation control
1	-	read only	0*	reserved
0	-	read only	1*	reserved

7.3.3 LEDOUT0 to LEDOUT3, LED driver output state

Table 10. LEDOUT0 to LEDOUT3 - LED driver output state registers (address 02h to 05h) bit description

*Legend: * default value.*

Address	Register	Bit	Symbol	Access	Value	Description
02h	LEDOUT0	7:6	LDR3	R/W	10*	LED3 output state control
		5:4	LDR2	R/W	10*	LED2 output state control
		3:2	LDR1	R/W	10*	LED1 output state control
		1:0	LDR0	R/W	10*	LED0 output state control
03h	LEDOUT1	7:6	LDR7	R/W	10*	LED7 output state control
		5:4	LDR6	R/W	10*	LED6 output state control
		3:2	LDR5	R/W	10*	LED5 output state control
		1:0	LDR4	R/W	10*	LED4 output state control
04h	LEDOUT2	7:6	LDR11	R/W	10*	LED11 output state control
		5:4	LDR10	R/W	10*	LED10 output state control
		3:2	LDR9	R/W	10*	LED9 output state control
		1:0	LDR8	R/W	10*	LED8 output state control
05h	LEDOUT3	7:6	LDR15	R/W	10*	LED15 output state control
		5:4	LDR14	R/W	10*	LED14 output state control
		3:2	LDR13	R/W	10*	LED13 output state control
		1:0	LDR12	R/W	10*	LED12 output state control

LDRx = 00 — LED driver x is off (x = 0 to 15).

LDRx = 01 — LED driver x is fully on (individual brightness and group dimming/blinking not controlled). The $\overline{OE}$ pin can be used as external dimming/blinking control in this state.

LDRx = 10 — LED driver x individual brightness can be controlled through its PWMx register (default power-up state) or PWMALL register for all LEDn outputs.

LDRx = 11 — LED driver x individual brightness and group dimming/blinking can be controlled through its PWMx register and the GRPPWM registers.

Remark: Setting the device in low power mode while being on group dimming/blinking mode may cause the LED output state to be in an unknown state after the device is set back to normal mode. The device must be reset and all register values reprogrammed.

7.3.4 GRPPWM, group duty cycle control

Table 11. GRPPWM - Group brightness control register (address 06h) bit description

*Legend: * default value*

Address	Register	Bit	Symbol	Access	Value	Description
06h	GRPPWM	7:0	GDC[7:0]	R/W	1111 1111*	GRPPWM register

When DMBLNK bit (MODE2 register) is programmed with logic 0, a 122 Hz fixed frequency signal is superimposed with the 31.25 kHz individual brightness control signal. GRPPWM is then used as a global brightness control allowing the LED outputs to be dimmed with the same value. The value in GRPFREQ is then a 'Don't care'.

7.3.5 GRPFREQ, group frequency

Table 12. GRPFREQ - Group frequency register (address 07h) bit description
*Legend: * default value.*

Address	Register	Bit	Symbol	Access	Value	Description
07h	GRPFREQ	7:0	GFRQ[7:0]	R/W	0000 0000*	GRPFREQ register

GRPFREQ is used to program the global blinking period when DMBLNK bit (MODE2 register) is equal to 1. Value in this register is a 'Don't care' when DMBLNK = 0. Applicable to LED outputs programmed with LDRx = 11 (LEDOUT0 to LEDOUT3 registers).

Blinking period is controlled through 256 linear steps from 00h (67 ms, frequency 15 Hz) to FFh (16.8 s).

$$global\ blinking\ period = \frac{GFRQ[7:0]+1}{15.26}(s) \quad (2)$$

7.3.6 PWM0 to PWM15, individual brightness control

Table 13. PWM0 to PWM15 - PWM registers 0 to 15 (address 08h to 17h) bit description
*Legend: * default value.*

Address	Register	Bit	Symbol	Access	Value	Description
08h	PWM0	7:0	IDC0[7:0]	R/W	0000 0000*	PWM0 Individual Duty Cycle
09h	PWM1	7:0	IDC1[7:0]	R/W	0000 0000*	PWM1 Individual Duty Cycle
0Ah	PWM2	7:0	IDC2[7:0]	R/W	0000 0000*	PWM2 Individual Duty Cycle
0Bh	PWM3	7:0	IDC3[7:0]	R/W	0000 0000*	PWM3 Individual Duty Cycle
0Ch	PWM4	7:0	IDC4[7:0]	R/W	0000 0000*	PWM4 Individual Duty Cycle
0Dh	PWM5	7:0	IDC5[7:0]	R/W	0000 0000*	PWM5 Individual Duty Cycle
0Eh	PWM6	7:0	IDC6[7:0]	R/W	0000 0000*	PWM6 Individual Duty Cycle
0Fh	PWM7	7:0	IDC7[7:0]	R/W	0000 0000*	PWM7 Individual Duty Cycle
10h	PWM8	7:0	IDC8[7:0]	R/W	0000 0000*	PWM8 Individual Duty Cycle
11h	PWM9	7:0	IDC9[7:0]	R/W	0000 0000*	PWM9 Individual Duty Cycle
12h	PWM10	7:0	IDC10[7:0]	R/W	0000 0000*	PWM10 Individual Duty Cycle
13h	PWM11	7:0	IDC11[7:0]	R/W	0000 0000*	PWM11 Individual Duty Cycle
14h	PWM12	7:0	IDC12[7:0]	R/W	0000 0000*	PWM12 Individual Duty Cycle
15h	PWM13	7:0	IDC13[7:0]	R/W	0000 0000*	PWM13 Individual Duty Cycle
16h	PWM14	7:0	IDC14[7:0]	R/W	0000 0000*	PWM14 Individual Duty Cycle
17h	PWM15	7:0	IDC15[7:0]	R/W	0000 0000*	PWM15 Individual Duty Cycle

A 31.25 kHz fixed frequency signal is used for each output. Duty cycle is controlled through 255 linear steps from 00h (0 % duty cycle = LED output off) to FEh (99.2 % duty cycle = LED output at maximum brightness) and FFh (100 % duty cycle = LED output completed ON). Applicable to LED outputs programmed with LDRx = 10 or 11 (LEDOUT0 to LEDOUT3 registers).

$$duty\ cycle = \frac{IDCx[7:0]}{256} \quad (3)$$

Remark: The first lower end 8 steps of PWM and the last (higher end) steps of PWM will not have effective brightness control of LEDs due to edge rate control of LED output pins.

7.3.7 IREF0 to IREF15, LED output current value registers

These registers reflect the gain settings for output current for LED0 to LED15.

Table 14. IREF0 to IREF15 - LED output gain control registers (address 18h to 27h) bit description

*Legend: * default value.*

Address	Register	Bit	Access	Value	Description
18h	IREF0	7:0	R/W	00h*	LED0 output current setting
19h	IREF1	7:0	R/W	00h*	LED1 output current setting
1Ah	IREF2	7:0	R/W	00h*	LED2 output current setting
1Bh	IREF3	7:0	R/W	00h*	LED3 output current setting
1Ch	IREF4	7:0	R/W	00h*	LED4 output current setting
1Dh	IREF5	7:0	R/W	00h*	LED5 output current setting
1Eh	IREF6	7:0	R/W	00h*	LED6 output current setting
1Fh	IREF7	7:0	R/W	00h*	LED7 output current setting
20h	IREF8	7:0	R/W	00h*	LED8 output current setting
21h	IREF9	7:0	R/W	00h*	LED9 output current setting
22h	IREF10	7:0	R/W	00h*	LED10 output current setting
23h	IREF11	7:0	R/W	00h*	LED11 output current setting
24h	IREF12	7:0	R/W	00h*	LED12 output current setting
25h	IREF13	7:0	R/W	00h*	LED13 output current setting
26h	IREF14	7:0	R/W	00h*	LED14 output current setting
27h	IREF15	7:0	R/W	00h*	LED15 output current setting

7.3.12 PWMALL — brightness control for all LEDn outputs

When programmed, the value in this register will be used for PWM duty cycle for all the LEDn outputs and will be reflected in PWM0 through PWM15 registers.

Table 25. PWMALL - brightness control for all LEDn outputs register (address 44h) bit description

*Legend: * default value.*

Address	Register	Bit	Access	Value	Description
44h	PWMALL	7:0	write only	0000 0000*	duty cycle for all LEDn outputs

Remark: Write to any of the PWM0 to PWM15 registers will overwrite the value in corresponding PWMn register programmed by PWMALL.

7.3.13 IREFALL register: output current value for all LED outputs

The output current setting for all outputs is held in this register. When this register is written to or updated, all LED outputs will be set to a current corresponding to this register value.

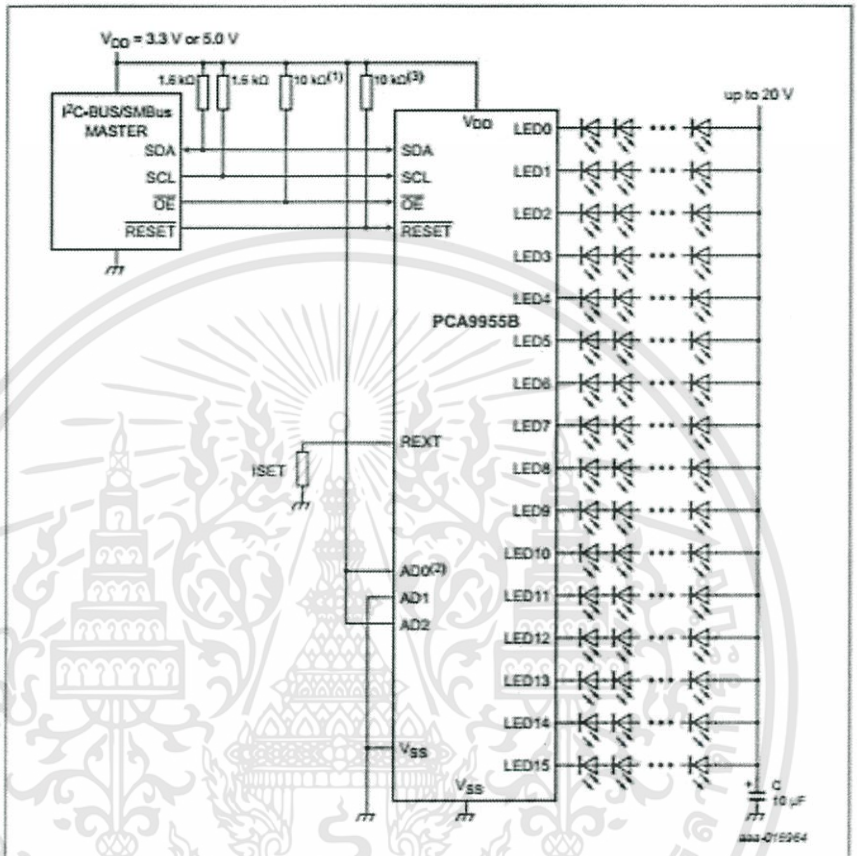
Writes to IREF0 to IREF15 will overwrite the output current settings.

Table 26. IREFALL - Output gain control for all LED outputs (address 45h) bit description

*Legend: * default value.*

Address	Register	Bit	Access	Value	Description
45h	IREFALL	7:0	write only	00h*	Current gain setting for all LED outputs.

10. Application design-in information



(1) $\overline{OE}$ requires pull-up resistor if control signal from the master is open-drain

(2) I²C-bus address = 1101001 when AD0, AD2 tied to V_{DD} and AD1 tied to V_{SS} (see Table 5)

(3) $\overline{RESET}$ requires a pull-up resistor of <100 kΩ if not used or connected to open-drain output

Fig 25. Typical application

- ตัวอย่างข้อมูลใน datasheet ที่จำเป็นต้องศึกษาของไอซีชนิด voltage translator

PCA9617A

Level translating Fm+ I²C-bus repeater

Rev. 1 — 20 March 2013 Product data sheet

2. Features and benefits

- 2 channel, bidirectional buffer isolates capacitance and allows 540 pF on either side of the device at 1 MHz and up to 4000 pF at lower speeds
- Voltage level translation from 0.8 V to 5.5 V and from 2.2 V to 5.5 V
- Footprint and functional replacement for PCA9517A at Fast-mode speeds
- Port A operating supply voltage range of 0.8 V to 5.5 V with normal levels
- Port B operating supply voltage range of 2.2 V to 5.5 V with static offset level
- 5 V tolerant I²C-bus and enable pins
- 0 Hz to 1000 kHz clock frequency (the maximum system operating frequency may be less than 1000 kHz because of the delays added by the repeater)
- Active HIGH repeater enable input referenced to $V_{CC(B)}$
- Open-drain input/outputs
- Latching free operation
- Supports arbitration and clock stretching across the repeater
- Accommodates Standard-mode, Fast-mode and Fast-mode Plus I²C-bus devices, SMBus (standard and high power mode), PMBus and multiple masters
- Powered-off high-impedance I²C-bus pins
- ESD protection exceeds 5500 V HBM per JESD22-A114 and 1000 V CDM per JESD22-C101
- Latch-up testing is done to JEDEC Standard JESD78 which exceeds 100 mA
- Packages offered: TSSOP8 and HWSON8

5.2 Pin description

Table 3. Pin description

Symbol	Pin		Description
	TSSOP8	HWSON8	
$V_{CC(A)}$	1	7	port A supply voltage (0.8 V to 5.5 V)
SCLA	2	8	serial clock port A bus
SDAA	3	1	serial data port A bus
GND	4	2[1]	supply ground (0 V)
EN	5	3	active HIGH repeater enable input
SDAB	6	4	serial data port B bus
SCLB	7	5	serial clock port B bus
$V_{CC(B)}$	8	6	port B supply voltage (2.2 V to 5.5 V)

[1] HWSON8 package die supply ground is connected to both GND pin and exposed center pad. GND pin must be connected to supply ground for proper device operation. For enhanced thermal, electrical, and board level performance, the exposed pad needs to be soldered to the board using a corresponding thermal pad on the board and for proper head conduction through the board, thermal vias need to be incorporated in the printed-circuit board in the thermal pad region.

7. Application design-in information

A typical application is shown in Figure 4. In this example, the system master is running on a 3.3 V I²C-bus while the slave is connected to a 1.2 V bus. Both buses run at 1000 kHz. Master devices can be placed on either bus.

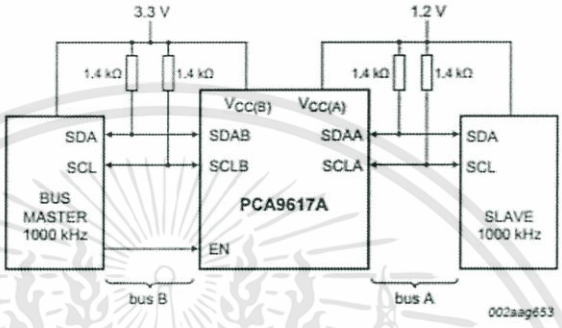


Fig 4. Typical application

8. Limiting values

Table 4. Limiting values

In accordance with the Absolute Maximum Rating System (IEC 60134).

Symbol	Parameter	Conditions	Min	Max	Unit
$V_{CC(B)}$	supply voltage port B		−0.5	+7	V
$V_{CC(A)}$	supply voltage port A	adjustable	−0.5	+7	V
$V_{I/O}$	voltage on an input/output pin	port A and port B; enable pin (EN)	−0.5	+7	V
$I_{I/O}$	input/output current	port A; port B	-	50	mA
I_I	input current	EN, $V_{CC(A)}$, $V_{CC(B)}$, GND	-	50	mA
P_{tot}	total power dissipation		-	100	mW
T_{stg}	storage temperature		−55	+125	°C
T_{amb}	ambient temperature	operating in free air	−40	+85	°C
T_j	junction temperature		-	+125	°C

- ตัวอย่างข้อมูลใน datasheet ที่จำเป็นต้องศึกษาของไอซีชนิด GPIO

PCA9535; PCA9535C

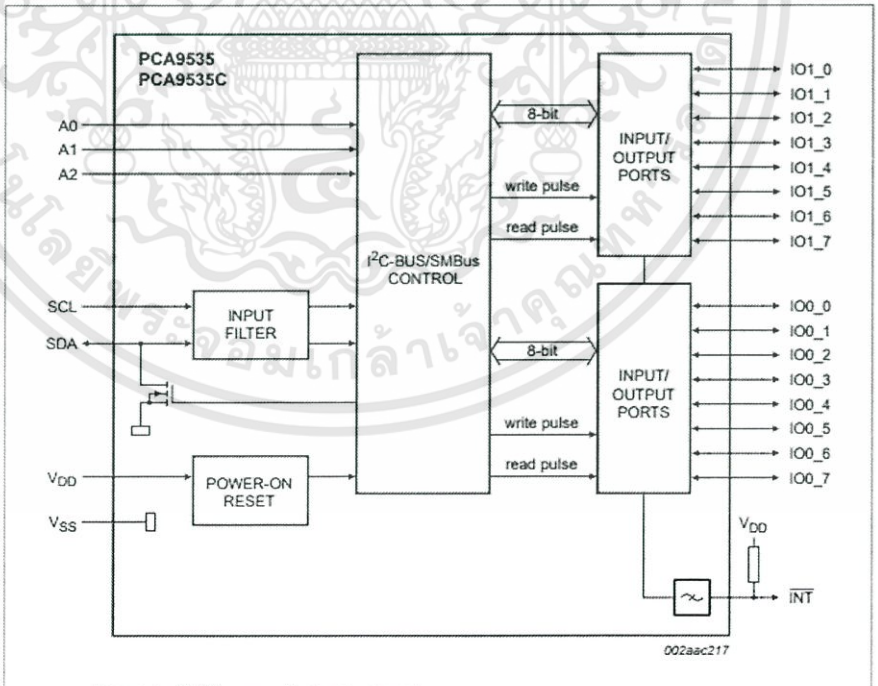
16-bit I²C-bus and SMBus, low power I/O port with interrupt

Rev. 6 — 7 November 2017 Product data sheet

2. Features and benefits

- Operating power supply voltage range of 2.3 V to 5.5 V
- 5 V tolerant I/Os
- Polarity Inversion register
- Active LOW interrupt output
- Low standby current
- Noise filter on SCL/SDA inputs
- No glitch on power-up
- Internal power-on reset
- 16 I/O pins which default to 16 inputs
- 0 Hz to 400 kHz clock frequency
- ESD protection exceeds 2000 V HBM per JESD22-A114, 200 V MM per JESD22-A115, and 1000 V CDM per JESD22-C101
- Latch-up testing is done to JEDEC Standard JESD78 which exceeds 100 mA
- Offered in four different packages: SO24, TSSOP24, HVQFN24 and HWQFN24

4. Block diagram



Remark: All I/Os are set to inputs at reset.

Fig 1. Block diagram of PCA9535; PCA9535C

5.2 Pin description

Table 3. Pin description

Symbol	Pin		Description
	SO24, TSSOP24	HVQFN24, HWQFN24	
$\overline{INT}$	1	22	interrupt output (open-drain)
A1	2	23	address input 1
A2	3	24	address input 2
IO0_0	4	1	port 0 input/output[1]
IO0_1	5	2	
IO0_2	6	3	
IO0_3	7	4	
IO0_4	8	5	
IO0_5	9	6	
IO0_6	10	7	
IO0_7	11	8	
V_{SS}	12	9[2]	supply ground
IO1_0	13	10	port 1 input/output[1]
IO1_1	14	11	
IO1_2	15	12	
IO1_3	16	13	
IO1_4	17	14	
IO1_5	18	15	
IO1_6	19	16	
IO1_7	20	17	
A0	21	18	address input 0
SCL	22	19	serial clock line
SDA	23	20	serial data line
V_{DD}	24	21	supply voltage

[1] PCA9535 I/Os are totem pole, whereas the I/Os on PCA9535C are open-drain.

[2] HVQFN24 and HWQFN24 package die supply ground is connected to both the V_{SS} pin and the exposed center pad. The V_{SS} pin must be connected to supply ground for proper device operation. For enhanced thermal, electrical, and board-level performance, the exposed pad needs to be soldered to the board using a corresponding thermal pad on the board, and for proper heat conduction through the board thermal vias need to be incorporated in the PCB in the thermal pad region.

6. Functional description

Refer to Figure 1 "Block diagram of PCA9535; PCA9535C".

6.1 Device address

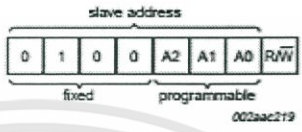


Fig 6. PCA9535; PCA9535C device address

6.2 Registers

6.2.1 Command byte

The command byte is the first byte to follow the address byte during a write transmission. It is used as a pointer to determine which of the following registers will be written or read.

Table 4. Command byte

Command	Register
0	Input port 0
1	Input port 1
2	Output port 0
3	Output port 1
4	Polarity Inversion port 0
5	Polarity Inversion port 1
6	Configuration port 0
7	Configuration port 1

6.2.2 Registers 0 and 1: Input port registers

This register is an input-only port. It reflects the incoming logic levels of the pins, regardless of whether the pin is defined as an input or an output by Register 3. Writes to this register have no effect.

The default value 'X' is determined by the externally applied logic level.

Table 5. Input port 0 Register

Bit	7	6	5	4	3	2	1	0
Symbol	I0.7	I0.6	I0.5	I0.4	I0.3	I0.2	I0.1	I0.0
Default	X	X	X	X	X	X	X	X

Table 6. Input port 1 register

Bit	7	6	5	4	3	2	1	0
Symbol	I1.7	I1.6	I1.5	I1.4	I1.3	I1.2	I1.1	I1.0
Default	X	X	X	X	X	X	X	X

6.2.3 Registers 2 and 3: Output port registers

This register is an output-only port. It reflects the outgoing logic levels of the pins defined as outputs by Registers 6 and 7. Bit values in this register have no effect on pins defined as inputs. In turn, reads from this register reflect the value that is in the flip-flop controlling the output selection, not the actual pin value.

Table 7. Output port 0 register

Bit	7	6	5	4	3	2	1	0
Symbol	O0.7	O0.6	O0.5	O0.4	O0.3	O0.2	O0.1	O0.0
Default	1	1	1	1	1	1	1	1

Table 8. Output port 1 register

Bit	7	6	5	4	3	2	1	0
Symbol	O1.7	O1.6	O1.5	O1.4	O1.3	O1.2	O1.1	O1.0
Default	1	1	1	1	1	1	1	1

6.2.4 Registers 4 and 5: Polarity Inversion registers

This register allows the user to invert the polarity of the Input port register data. If a bit in this register is set (written with '1'), the Input port data polarity is inverted. If a bit in this register is cleared (written with a '0'), the Input port data polarity is retained.

Table 9. Polarity Inversion port 0 register

Bit	7	6	5	4	3	2	1	0
Symbol	N0.7	N0.6	N0.5	N0.4	N0.3	N0.2	N0.1	N0.0
Default	0	0	0	0	0	0	0	0

Table 10. Polarity Inversion port 1 register

Bit	7	6	5	4	3	2	1	0
Symbol	N1.7	N1.6	N1.5	N1.4	N1.3	N1.2	N1.1	N1.0
Default	0	0	0	0	0	0	0	0

6.2.5 Registers 6 and 7: Configuration registers

This register configures the directions of the I/O pins. If a bit in this register is set (written with '1'), the corresponding port pin is enabled as an input with high-impedance output driver. If a bit in this register is cleared (written with '0'), the corresponding port pin is enabled as an output. At reset, the device's ports are inputs.

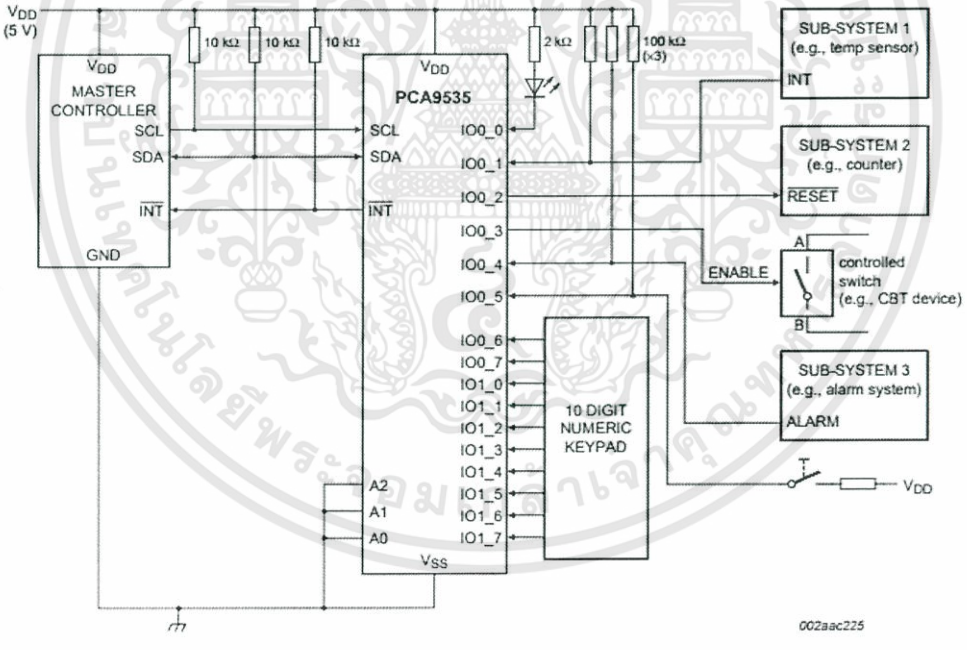
Table 11. Configuration port 0 register

Bit	7	6	5	4	3	2	1	0
Symbol	C0.7	C0.6	C0.5	C0.4	C0.3	C0.2	C0.1	C0.0
Default	1	1	1	1	1	1	1	1

Table 12. Configuration port 1 register

Bit	7	6	5	4	3	2	1	0
Symbol	C1.7	C1.6	C1.5	C1.4	C1.3	C1.2	C1.1	C1.0
Default	1	1	1	1	1	1	1	1

8. Application design-in information



Device address configured as 1110 100Xb for this example.

IO0_0, IO0_2, IO0_3 configured as outputs.

IO0_1, IO0_4, IO0_5 configured as inputs.

IO0_6, IO0_7, and IO1_0 to IO1_7 configured as inputs.

Fig 17. Typical application

- ตัวอย่างข้อมูลใน datasheet ที่จำเป็นต้องศึกษาของไอซีชนิด multiplexer

PCA9544A

4-channel I²C-bus multiplexer with interrupt logic

Rev. 5 — 23 April 2014 Product data sheet

2. Features and benefits

- 1-of-4 bidirectional translating multiplexer
- I²C-bus interface logic; compatible with SMBus
- 4 active LOW interrupt inputs
- Active LOW interrupt output
- 3 address pins allowing up to 8 devices on the I²C-bus
- Channel selection via I²C-bus
- Power-up with all multiplexer channels deselected
- Low R_{on} switches
- Allows voltage level translation between 1.8 V, 2.5 V, 3.3 V and 5 V buses
- No glitch on power-up
- Supports hot insertion
- Low standby current
- Operating power supply voltage range of 2.3 V to 5.5 V
- 5 V tolerant Inputs
- 0 Hz to 400 kHz clock frequency
- ESD protection exceeds 2000 V HBM per JESD22-A114 and 1000 V CDM per JESD22-C101
- Latch-up testing is done to JEDEC Standard JESD78 which exceeds 100 mA
- Three packages offered: SO20, TSSOP20 and HVQFN20

5.2 Pin description

Table 3. Pin description

Symbol	Pin		Description
	SO20, TSSOP20	HVQFN20	
A0	1	19	address input 0
A1	2	20	address input 1
A2	3	1	address input 2
$\overline{INT0}$	4	2	active LOW interrupt input 0
SD0	5	3	serial data 0
SC0	6	4	serial clock 0
$\overline{INT1}$	7	5	active LOW interrupt input 1
SD1	8	6	serial data 1
SC1	9	7	serial clock 1
V_{SS}	10	8[1]	supply ground
$\overline{INT2}$	11	9	active LOW interrupt input 2
SD2	12	10	serial data 2
SC2	13	11	serial clock 2
$\overline{INT3}$	14	12	active LOW interrupt input 3
SD3	15	13	serial data 3
SC3	16	14	serial clock 3
$\overline{INT}$	17	15	active LOW interrupt output
SCL	18	16	serial clock line
SDA	19	17	serial data line
V_{DD}	20	18	supply voltage

6. Functional description

Refer to Figure 1 "Block diagram".

6.1 Device addressing

Following a START condition the bus master must output the address of the slave it is accessing. The address of the PCA9544A is shown in Figure 5. To conserve power, no internal pull-up resistors are incorporated on the hardware selectable address pins and they must be pulled HIGH or LOW.

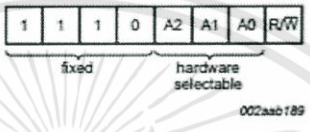


Fig 5. Slave address

The last bit of the slave address defines the operation to be performed. When set to logic 1 a read is selected, while a logic 0 selects a write operation.

6.2 Control register

Following the successful acknowledgement of the slave address, the bus master sends a byte to the PCA9544A which is stored in the Control register. If the PCA9544A receives multiple bytes, it saves the last byte received. This register can be written and read via the I²C-bus.

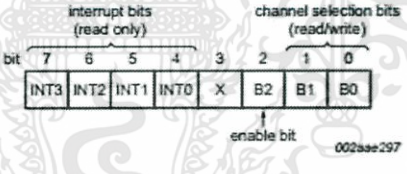


Fig 6. Control register

Table 4. Control register: Write — channel selection; Read — channel status

INT3	INT2	INT1	INT0	D3	B2	B1	B0	Command
X	X	X	X	X	0	X	X	no channel selected
X	X	X	X	X	1	0	0	channel 0 enabled
X	X	X	X	X	1	0	1	channel 1 enabled
X	X	X	X	X	1	1	0	channel 2 enabled
X	X	X	X	X	1	1	1	channel 3 enabled
0	0	0	0	0	0	0	0	no channel selected; power-up default state

6.3 Interrupt handling

Table 5. Control register read — interrupt

INT3	INT2	INT1	INT0	D3	B2	B1	B0	Command
X	X	X	0	X	X	X	X	no interrupt on channel 0
			1					interrupt on channel 0
X	X	0	X	X	X	X	X	no interrupt on channel 1
		1						interrupt on channel 1
X	0	X	X	X	X	X	X	no interrupt on channel 2
	1							interrupt on channel 2
0	X	X	X	X	X	X	X	no interrupt on channel 3
1								interrupt on channel 3

Remark: Several interrupts can be active at the same time. For example: INT3 = 0, INT2 = 1, INT1 = 1, INT0 = 0, means that there is no interrupt on channel 0 and channel 3, and there is an interrupt on channel 1 and on channel 2.

7.5 Bus transactions

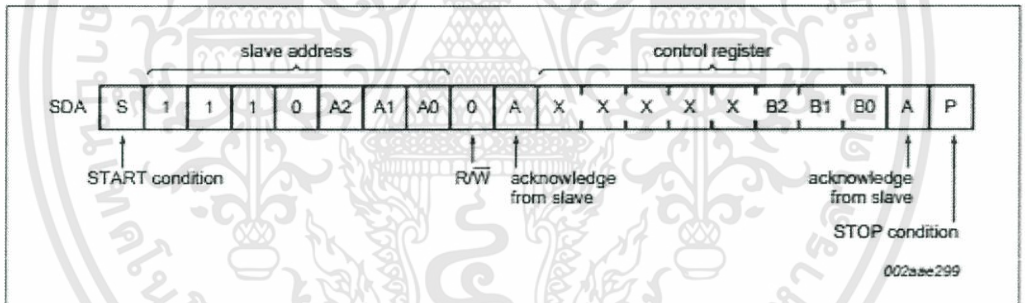


Fig 12. Write control register

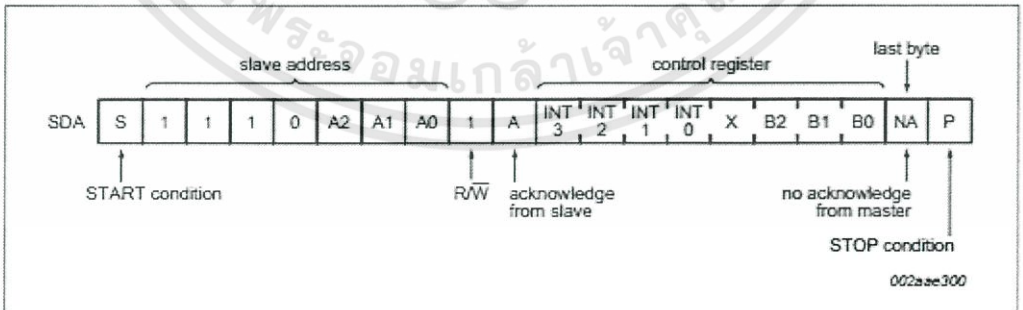
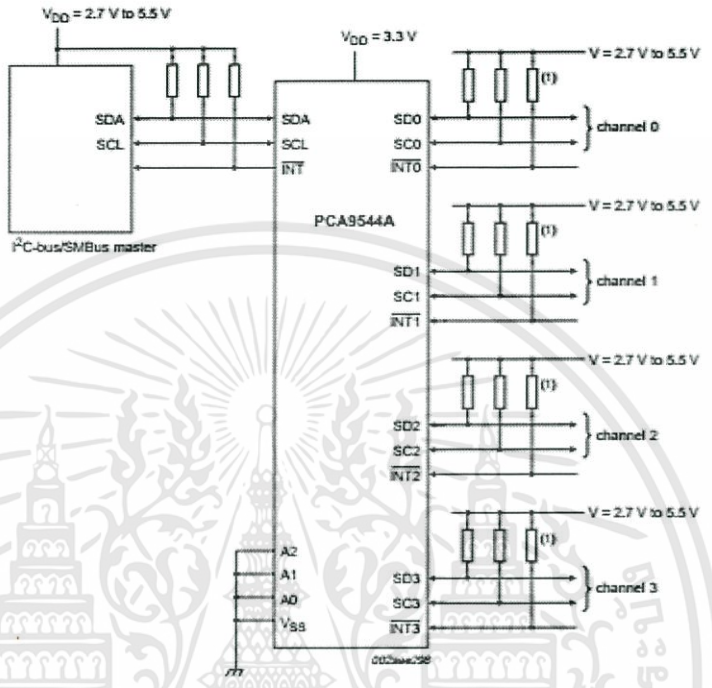


Fig 13. Read control register

8. Application design-in information



(1) If the device generating the interrupt has an open-drain output structure or can be 3-stated, a pull-up resistor is required.

If the device generating the interrupt has a totem pole output structure and cannot be 3-stated, a pull-up resistor is not required.

The interrupt inputs should not be left floating.

Fig 14. Typical application