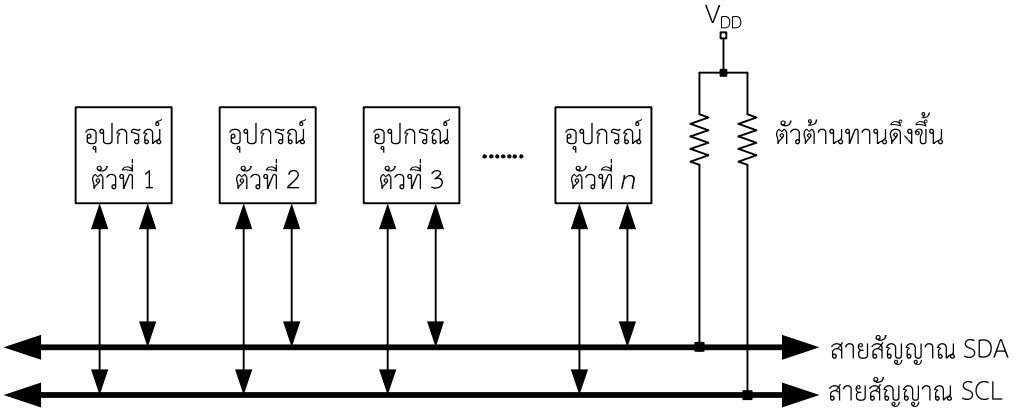
240-319 Embedded System Developer Module
LAB 13 : I2C and PCF-8574

อุปกรณ์ที่ใช้ในการทดลอง

- บอร์ด Arduino UNO พร้อมบอร์ด CoE PSU Arduino Shield
- มอดูล PCF-8574
- คีย์แพ็ดแบบเมมเบรน 4x4
- บอร์ด CoE LED 8 Bit
- บอร์ด CoE SW and Buzzer

6.1 การเชื่อมต่อแบบไอสแควร์ซี

การเชื่อมต่อแบบไอสแควร์ซี (I^2C) หรือเรียกอีกอย่างหนึ่งว่าระบบบัสแบบทู-ไวร์ (Two-Wire Interface: TWI) เป็นการสื่อสารอนุกรมอย่างง่ายที่มีความซับซ้อนต่ำ โพรโตคอลกำหนดให้มีสายสัญญาณสองเส้น ได้แก่ เส้นสำหรับส่งสัญญาณนาฬิกาซึ่งมีชื่อว่า SCL (ย่อมาจาก Serial Clock) และอีกเส้นหนึ่งสำหรับส่งข้อมูลซึ่งมีชื่อว่า SDA (ย่อมาจาก Serial Data) สัญญาณแต่ละเส้นจะต้องถูกเชื่อมต่อด้วยตัวต้านทานดึงขึ้น ดังรูปที่ 13.1 ระบบบัสถูกออกแบบมาให้รองรับอุปกรณ์บนสายสัญญาณทั้งสองได้มากสุดถึง 128 ตัว แต่ละอุปกรณ์บนบัสจะต้องมีค่าเลขที่อยู่ (Address) เฉพาะตัวของมันไม่ซ้ำกับอุปกรณ์อื่นที่อยู่บนบัสเดียวกัน อุปกรณ์ทุกตัวจะต้องทำงานตามข้อกำหนดของโพรโตคอลอย่างเคร่งครัด จึงจะสามารถทำงานร่วมกันบนบัสได้โดยไม่รบกวนซึ่งกันและกัน



รูปที่ 13.1 ระบบบัสไอสแควร์ซี

จากรูปที่ 13.1 แสดงอุปกรณ์ที่ต่ออยู่กับบัส I^2C จำนวน *n* ตัว แต่ละอุปกรณ์ที่จะขับสัญญาณลงบัสจะต้องมีเอาต์พุตแบบเดรนเปิด (Open-drain) หรือคอลเล็กเตอร์เปิด (Open-Collector) ซึ่งช่วยให้สามารถทำกระบวนการไวร์แอนด์ (Wired-AND) บนเส้นสัญญาณของบัสได้ ดังนั้นการที่สายสัญญาณบนบัสจะมีค่าตรรกะต่ำได้ก็ต่อเมื่อมีอุปกรณ์อย่างน้อยหนึ่งตัวส่งค่าตรรกะต่ำลงบนสายสัญญาณเส้นนั้น แต่หากไม่มีอุปกรณ์ใดบนบัสส่งค่าตรรกะต่ำลงบนบัสเลยก็จะส่งผลให้สายสัญญาณเส้นนั้นมีค่าตรรกะสูง หรือในกรณีที่อุปกรณ์ทุกตัวบนบัสมิได้ส่งค่าทางตรรกะ (ไม่ว่าจะสูงหรือต่ำ) ลงบนบัสเลย หรือพูดอีกอย่างคือ เอาต์พุตของอุปกรณ์ทุกตัวอยู่ในสภาวะอิมพีแดนซ์สูงก็จะส่งผลให้สายสัญญาณเส้นนั้นเป็นตรรกะสูงเนื่องจากถูกเชื่อมต่อด้วยตัวต้านทานดึงขึ้นนั่นเอง

การเชื่อมต่อแบบ I^2C ถูกคิดค้นขึ้นในปี ค.ศ. 1982 โดยบริษัทฟิลลิปส์เซมิคอนดักเตอร์ (ปัจจุบัน คือ บริษัทเอ็นเอ็กซ์พีเซมิคอนดักเตอร์ : NXP Semiconductor) แต่การที่บางบริษัท (รวมทั้งบริษัท Atmel ผู้คิดค้นสถาปัตยกรรมเอวีอาร์) เลี่ยงที่จะใช้คำว่า I^2C บนผลิตภัณฑ์ของตนแต่เปลี่ยนไปเรียกว่า Two-Wire Interface หรือ TWI แทนเนื่องจากต้องการหลีกเลี่ยงปัญหาเรื่องข้อเรียกร้องในสิทธิบัตรและข้ออนุญาตการใช้งาน [1], [2] ปัจจุบันมีอุปกรณ์ต่าง ๆ เป็นจำนวนมากสนับสนุนการเชื่อมต่อแบบนี้ ยกตัวอย่างเช่น ไอซีสำหรับเพิ่มขยายพอร์ต ไอซีหน่วยความจำเอฟอีแรม (FeRAM) [3] ไอซีแปลงสัญญาณแอนะล็อกเป็นดิจิทัล ไอซีแปลงสัญญาณดิจิทัลเป็นแอนะล็อก ไอซีสำหรับขับแอลอีดีชนิด 7 ส่วน ตัวตรวจรู้ความใกล้ชิด (Proximity Sensor) และ ไอซีนาฬิกาเวลาจริง (Real-time Clock) เป็นต้น

13.1.1 การรับส่งข้อมูลแบบใช้เลขที่อยู่ขนาด 7 บิต

เนื่องจากตามมาตรฐานของการเชื่อมต่อ I^2C ในตอนแรกเริ่มถูกกำหนดให้ใช้ขนาดของเลขที่อยู่จำนวน 7 บิต จึงส่งผลให้มีอุปกรณ์บนบัสได้สูงสุดไม่เกิน 128 ตัว (2^7=128) อย่างไรก็ดี หากค่าความจุไฟฟ้า (Capacitance) โดยของบัสมีค่าเกิน 400 พิโคฟารัดก็จะส่งผลให้จำนวนอุปกรณ์ที่ต่ออยู่กับบัสอาจต้องมีการลดจำนวนลง (ต่ำกว่า 128) ความเร็วในการรับส่งข้อมูลบนบัสมีอยู่หลายแบบวิธี แต่ที่นิยมใช้งานโดยทั่วไปมีอยู่ 2 ค่า ได้แก่ แบบวิธีปรกติซึ่งทำงานที่ความเร็ว 100 กิโลเฮิรทซ์ และแบบวิธีเร็ว (Fast Mode) ซึ่งทำงานที่ความเร็ว 400 กิโลเฮิรทซ์

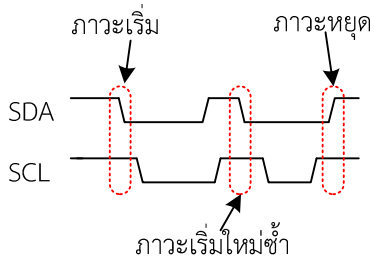
อุปกรณ์บนบัส I^2C ตัวใดก็ตามที่เป็นผู้เริ่มต้นการรับส่งข้อมูลบนบัสจะถูกเรียกว่า มาสเตอร์ (Master) โดยโพรโตคอลกำหนดให้อุปกรณ์ที่ทำหน้าที่มาสเตอร์ทำหน้าที่ส่งสัญญาณนาฬิกาบนสายสัญญาณ SCL ด้วย นอกจากนี้ เมื่อการรับส่งข้อมูลสิ้นสุดลงก็จะตัวมาสเตอร์จะต้องที่ทำหน้าที่เป็นผู้สั่งการให้เกิดภาวะหยุดด้วยตนเอง

อุปกรณ์บนบัสตัวที่ถูกเรียกใช้โดยมาสเตอร์จะถูกเรียกว่า สเลฟ (Slave) โดยที่ตัวสเลฟจะทำงานเป็นผู้ส่งข้อมูล (Transmitter) เพื่อส่งข้อมูลให้กับมาสเตอร์ หรืออาจทำหน้าที่เป็นผู้รับข้อมูล (Receiver) จากมาสเตอร์ด้วยก็ได้ นอกจากนี้ตัวมาสเตอร์ยังสามารถทำหน้าที่เป็นได้ทั้งผู้รับข้อมูลและผู้ส่งข้อมูล ขึ้นอยู่กับลักษณะการทำงานร่วมกันระหว่างมาสเตอร์กับสเลฟว่าจะเป็นรูปแบบใด บนบัสหนึ่งสามารถมีอุปกรณ์มากกว่าหนึ่งตัวเป็นมาสเตอร์ได้ แต่ในขณะใดขณะหนึ่งจะมีอุปกรณ์ที่เป็นมาสเตอร์ได้เพียงตัวเดียวเท่านั้น

การติดต่อบนบัสจะเกิดขึ้นเมื่อมาสเตอร์ทำการส่งภาวะเริ่ม (START condition) ลงบนบัส และจะถือว่าบัสอยู่ในสถานะไม่ว่างไปจนกว่าตัวมาสเตอร์จะส่งภาวะหยุด (STOP condition) ในระหว่างที่บัสไม่ว่างจะต้องไม่มีอุปกรณ์อื่นใดบนบัสพยายามทำตัวเป็นมาสเตอร์ อย่างไรก็ดีโพรโตคอลของ I2C อนุญาตให้มาสเตอร์ตัวที่กำลังครอบครองบัสสามารถร้องขอภาวะเริ่มใหม่ได้โดยที่การรับส่งบนบัสยังไม่สิ้นสุด ซึ่งเราเรียกภาวะนี้ว่า ภาวะเริ่มใหม่ซ้ำ (REPEATED START condition) ดังรูปที่ 13.2 จะสังเกตเห็นว่าทั้งการส่งการร้องขอภาวะเริ่ม และภาวะเริ่มใหม่ซ้ำสามารถทำได้โดยการเปลี่ยนค่าระดับค่าตรรกะของสายสัญญาณ SDA จากสูงไปต่ำในระหว่างที่สายสัญญาณ SCL มีค่าตรรกะสูง ส่วนการร้องขอภาวะหยุดนั้นทำได้โดยการเปลี่ยนค่าระดับตรรกะของสายสัญญาณ SDA จากต่ำไปสูงในระหว่างที่สายสัญญาณ SCL มีค่าตรรกะสูง

ข้อพึงสังเกต คือ การส่งภาวะเริ่ม ภาวะเริ่มใหม่ซ้ำ และ ภาวะหยุดของมาสเตอร์ เป็นความพยายามของมาสเตอร์ที่จะเปลี่ยนแปลงค่าระดับตรรกะของสายสัญญาณ SDA ซึ่งต้องทำในขณะที่ระดับตรรกะของสายสัญญาณ SCL เป็นค่าตรรกะคงที่อยู่ที่ตรรกะสูงเท่านั้น

เมื่ออุปกรณ์บนบัสตัวใดต้องการเป็นมาสเตอร์ มันจะตรวจสอบสถานะบนบัสเสียก่อนว่าบัสว่างหรือไม่ หากสายสัญญาณ SDA และ SCL เป็นตรรกะสูงทั้งคู่แสดงว่าในขณะนั้นบัสอยู่ในสถานะว่าง จึงส่งผลให้อุปกรณ์ที่ต้องการเป็นมาสเตอร์ทำการส่งสถานะเริ่มต้นลงบนบัส ในกรณีที่มีอุปกรณ์มากกว่าหนึ่งตัวส่งการร้องขอสถานะเริ่มต้นพร้อมกัน จะมีกลไกการตัดสินว่าอุปกรณ์ใดได้สิทธิการเป็นมาสเตอร์ หรือเป็นผู้ได้สิทธิการครอบครองบัส

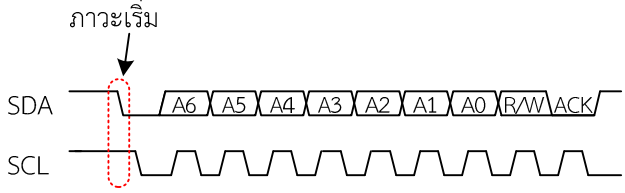


รูปที่ 13.2 การร้องขอภาวะเริ่ม ภาวะเริ่มซ้ำใหม่ และภาวะหยุดของมาสเตอร์ในบัส I^2C

13.1.2 แพ็กเก็ตเลขที่อยู่และแพ็กเก็ตข้อมูล

เมื่อมาสเตอร์ได้รับการครอบครองบัสแล้ว จะมีการส่งแพ็กเก็ตเลขที่อยู่ตามด้วยแพ็กเก็ตข้อมูล แต่ละแพ็กเก็ตมีความยาว 9 บิต แพ็กเก็ตเลขที่อยู่มีโครงสร้างดังรูปที่ 13.3 ประกอบด้วยบิตเลขที่อยู่ขนาด 7 บิต และบิตอ่าน/เขียน (R/W) ตามด้วยบิตตอบรับ หรือเรียกอีกอย่างหนึ่งว่าบิต ACK (Acknowledge) หากบิตอ่าน/เขียนมีค่าตรรกะสูงหมายถึงการอ่าน แต่หากมีตรรกะต่ำหมายถึงการเขียน

บิตเลขที่อยู่ขนาด 7 บิต ประกอบด้วย บิต A0-A2 ซึ่งเป็นบิตที่ผู้ใช้สามารถตั้งค่าเองได้ ส่วนอีกสี่บิตถัดมาเป็นเลขประจำตัวของผู้ผลิตรายนั้น ๆ ซึ่งไม่สามารถเปลี่ยนแปลงโดยผู้ใช้ได้ การส่งเลขที่อยู่บนบัสจะส่งบิตที่มีนัยสำคัญสูงสุด (บิต A6) ออกมาก่อนตามด้วยบิตที่มีนัยสำคัญรองลงมาและปิดท้ายด้วยบิตที่มีนัยสำคัญต่ำสุด (บิต A0) ตามด้วยบิตอ่าน/เขียน

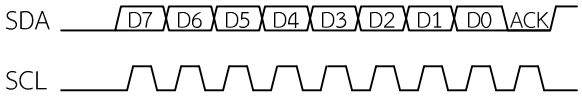


รูปที่ 13.3 แพ็กเก็ตเลขที่อยู่ในระบบบัส I^2C

เมื่อมาสเตอร์ส่งบิตเลขที่อยู่และบิตอ่าน/เขียนออกไปจนครบแล้วมันจะส่งค่าตรรกะสูงในช่วงเวลาของบิต ACK อุปกรณ์ที่เป็นสเลฟบนบัสตัวที่ได้รับการติดต่อจะทำการตอบรับด้วยการดึงสายสัญญาณ SDA ให้เป็นตรรกะต่ำในช่วงเวลาของบิต ACK ในขณะนี้ตัวมาสเตอร์จะคอยเฝ้าดูสถานะทางตรรกะที่ปรากฏบนบัสว่ามีการตอบรับจากสเลฟหรือไม่ หากไม่มีการตอบรับจากสเลฟไม่ว่าด้วยเหตุผลใดก็ตาม มาสเตอร์สามารถเลือกได้ว่าจะร้องขอภาวะหยุด หรือภาวะเริ่มซ้ำใหม่

เลขที่อยู่ขนาด 7 บิตค่า “0000000” จะถูกสงวนเอาไว้ใช้สำหรับการทำการเรียกทั่วไป (general call) ดังนั้นหากมาสเตอร์ส่งการเรียกทั่วไปออกมาแล้ว อุปกรณ์ที่เป็นสเลฟทุกตัวในระบบจะต้องดึงค่าในสายสัญญาณ SDA ให้เป็นตรรกะต่ำในช่วงเวลาของบิต ACK หลังจากนั้นแพ็กเก็ตของข้อมูลที่จะตามมาจะถูกอุปกรณ์ที่เป็นสเลฟทุกตัวในระบบรับเอาไปใช้ การทำการเรียกทั่วไปจะต้องมีค่าของบิตอ่าน/เขียนเป็นตรรกะต่ำเสมอ นั่นหมายถึงการที่มาสเตอร์ต้องการเขียนข้อมูลอะไรบางอย่างให้กับสเลฟทุก ๆ ตัว หากทำการเรียกทั่วไปด้วยบิตอ่าน/เขียนเป็นตรรกะสูงจะเกิดภาวะการชนกันของข้อมูลขึ้น เนื่องจากอุปกรณ์ทุกตัวเข้าใจว่ามาสเตอร์ต้องการอ่านข้อมูลจากตัวมันเองส่งผลให้สเลฟแต่ละตัวพยายามส่งข้อมูลลงบนบัสพร้อมกันที่ขา SDA ส่งผลให้มีการชนกันของข้อมูลเกิดขึ้น ดังนั้นผู้เขียนโปรแกรมต้องพึงระวังมิให้เกิดภาวะดังกล่าว

แพ็กเก็ตข้อมูลมีขนาด 9 บิต โดย 8 บิตแรกที่ส่งเป็นข้อมูลที่จะส่งจริงจากผู้ส่งไปยังผู้รับและบิตที่ 9 จะเป็นบิต ACK ซึ่งจะถูกดึงให้เป็นตรรกะต่ำโดยผู้รับ หากผู้ส่งพบว่าบิต ACK บนสายสัญญาณ SDA มีค่าเป็นตรรกะสูงแสดงว่าไม่มีผู้รับข้อมูลแพ็กเก็ตนั้น ข้อมูลที่ส่งบนบัสจะเริ่มตั้งแต่บิตที่มีนัยสำคัญสูงสุด (บิต D7) ออกไปก่อนตามด้วยบิตที่มีนัยสำคัญรองลงมา และปิดท้ายด้วยบิตที่มีนัยสำคัญต่ำสุด (บิต D0) โดยในระหว่างที่มีการรับส่งแพ็กเก็ตข้อมูล มาสเตอร์จะเป็นผู้ทำหน้าที่สร้างสัญญาณนาฬิกาและภาวะเริ่มรวมทั้งภาวะหยุด



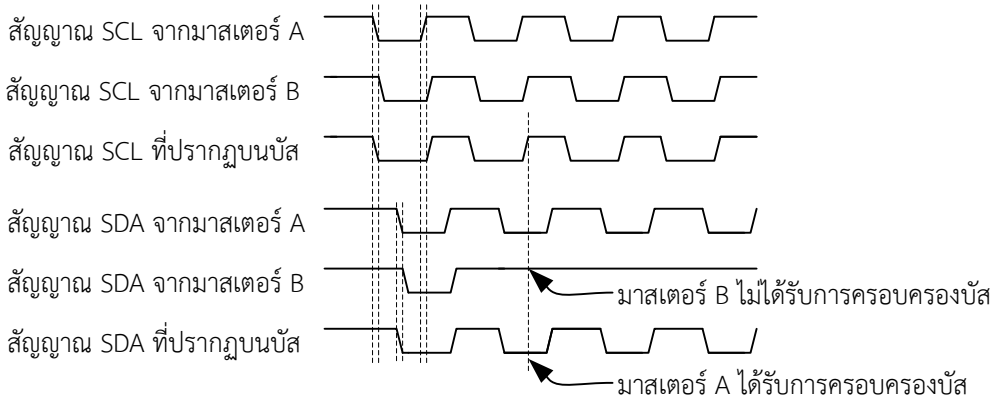
รูปที่ 13.4 แพ็กเก็ตข้อมูลในระบบบัส I^2C

13.1.3 การตัดสินชี้ขาดการร้องขอการครอบครองบัส

การเชื่อมต่อแบบทู-ไวร์อนุญาตให้มีอุปกรณ์ทำหน้าที่มาสเตอร์มากกว่าหนึ่งตัวอยู่บนบัสเดียวกันได้ หากมาสเตอร์มากกว่าหนึ่งตัวเห็นว่าบัสว่างอยู่ก็จะเกิดสถานการณ์ที่มาสเตอร์หลายตัวพยายามใช้บัสพร้อม ๆ กัน หากเกิดเหตุการณ์ดังกล่าว ระบบบัส I^2C จะมีกลไกการตัดสินชี้ขาดว่ามาสเตอร์ตัวใดจะได้รับการครอบครองบัส ซึ่งทำได้โดยใช้การดำเนินการไวร์แอนด์ (Wired-AND) บนเส้นสัญญาณของบัส รูปที่ 13.5 แสดงตัวอย่างการพยายามครอบครองบัสของมาสเตอร์ A และมาสเตอร์ B โดยสัญญาณ SCL ที่ปรากฏบนบัสจะถูกใช้เป็นสัญญาณนาฬิการ่วมกันระหว่างมาสเตอร์ทั้งสองตัว การตัดสินการครอบครองบัสจะอยู่ที่ภาวะสัญญาณ SDA ที่ปรากฏบนบัสว่าตรงกับสัญญาณที่ส่งจากมาสเตอร์ตัวใด จะเห็นว่าในการส่งสัญญาณรอบแรกลงบนเส้น SDA นั้น สัญญาณที่ส่งออกมาจากทั้งมาสเตอร์ทั้งสองมีค่าตรรกะต่ำทั้งคู่ ยังผลให้มาสเตอร์ทั้งสองตรวจพบว่าสัญญาณบนเส้น SDA มีค่าตรงกับค่าที่ตนเองส่งออกมาทำให้ในเวลาดังกล่าวมาสเตอร์ทั้งสองยังไม่ถือว่าถูกตัดสิทธิ์การครอบครองบัสและทำการส่งข้อมูลที่สองในสัญญาณนาฬิการอบถัดมา ต่อมามาสเตอร์ B ส่งสัญญาณค่าตรรกะสูงออกที่เส้นสัญญาณ SDA ในขณะที่มาสเตอร์ A ส่งค่าตรรกะต่ำออกมา ยังผลให้สัญญาณที่ปรากฏบนเส้นสัญญาณ

SDA มีค่าตรรกะต่ำตามมาสเตอร์ A ในขณะเดียวกันมาสเตอร์ B ก็จะสามารถตรวจจับสภาวะนี้ได้ที่ขอบขาขึ้นของสัญญาณนาฬิการอบที่สาม ยังผลให้มาสเตอร์ B ทราบว่าตนเองเสียการครอบครองบัสไปให้ผู้อื่นแล้ว

อุปกรณ์มาสเตอร์ใดที่มิได้รับการครอบครองบัสจากกระบวนการตัดสินชี้ขาด จะต้องผันตัวเองไปเป็นสเลฟเป็นการชั่วคราวไปพลางก่อน รอจนกว่าบัสจะว่างแล้วจึงดำเนินการร้องขอใช้งานบัสใหม่



รูปที่ 13.5 ตัวอย่างการตัดสินชี้ขาดการร้องขอการครอบครองบัสของการเชื่อมต่อแบบไอสแควร์ซี

13.2 อุปกรณ์ที่สนับสนุนการเชื่อมต่อแบบไอสแควร์ซี

ปัจจุบันมีอุปกรณ์ต่าง ๆ เป็นจำนวนมากสนับสนุนการเชื่อมต่อแบบนี้ ยกตัวอย่างเช่น ไอซีสำหรับเพิ่มขยายพอร์ต ไอซีหน่วยความจำเอฟอีแรม (FeRAM) ไอซีหน่วยความจำชนิดอีอีพร็อม ไอซีแปลงสัญญาณแอนะล็อกเป็นดิจิทัลไอซีแปลงสัญญาณดิจิทัลเป็นแอนะล็อก ไอซีสำหรับขับแอลอีดีแบบ 7 ส่วน ตัวตรวจรู้ความใกล้ชิด (Proximity Sensor) ไอซีนาฬิกาเวลาจริง (RTC: Real-time Clock) เป็นต้น ตารางที่ 13.1 แสดงตัวอย่างหมายเลขอุปกรณ์ที่สนับสนุนการเชื่อมต่อแบบทูไวร์หรือไอสแควร์ซี

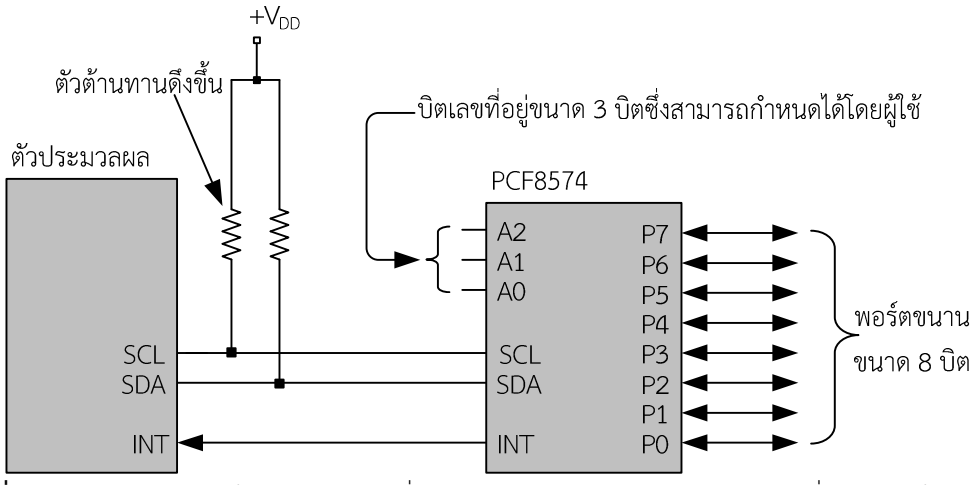
ตารางที่ 13.1 ตัวอย่างไอซีที่สนับสนุนการเชื่อมต่อแบบทู-ไวร์หรือไอสแควร์ซี

เบอร์ไอซี	หน้าที่ใช้งาน
DS1307	ไอซีนาฬิกาเวลาจริง
PCF-8574A	ไอซีเพิ่มขยายพอร์ตขนาด 8 บิต
PCF-8575	ไอซีเพิ่มขยายพอร์ตขนาด 16 บิต
SX9310	ตัวตรวจรู้ความใกล้ชิด
PCF8591	ไอซีแปลงแอนะล็อกเป็นดิจิทัลและแปลงดิจิทัลเป็นแอนะล็อก ความละเอียดขนาด 8 บิต
ADS1000-Q1	ไอซีแปลงแอนะล็อกเป็นดิจิทัล ความละเอียดขนาด 12 บิต
DAC7571	ไอซีแปลงดิจิทัลเป็นแอนะล็อก ความละเอียดขนาด 12 บิต
PCD3311C	ไอซีกำเนิดสัญญาณดีทีเอ็มเอฟ (DTMF : Dual-tone multi frequency)
PCF-8582	ไอซีหน่วยความจำอีอีพร็อมขนาด 2 กิโลบิต
24C32	ไอซีหน่วยความจำอีอีพร็อมขนาด 32 กิโลบิต
MR44V064B	ไอซีหน่วยความจำเอฟอีแรม (Ferroelectric RAM) ขนาด 64 กิโลบิต
PCF8576	ไอซีขับหน่วยแสดงผลแอลซีดี
MAX6955	ไอซีขับแอลอีดีชนิด 7 ส่วน
AN32181B	ไอซีขับแอลอีดีชนิดดอตเมทริกซ์ขนาด 12x12 จุด

ในการใช้งานอุปกรณ์ที่รองรับมาตรฐานการเชื่อมต่อแบบทู-ไวร์หรือไอสแควร์ซี ผู้เขียนโปรแกรมจำเป็นต้องทราบข้อมูลหมายเลขบิตที่อยู่ขนาด 7 บิตของอุปกรณ์นั้น ๆ โดยเฉพาะข้อมูลบิต A[6:3] ซึ่งเป็นเลขประจำตัวของผู้ผลิตไอซี ส่วนบิต A[2:0] นั้นผู้ผลิตส่วนใหญ่ออกแบบมาให้ผู้ใช้สามารถตั้งค่าเองได้โดยสามารถเลือกที่จะต่อขา A[2:0] ของตัวไอซีเข้ากับขั้วบวกของแหล่งจ่าย (ขา V_{DD}) และขากราวด์ได้ตามที่ผู้ใช้ต้องการ และผู้ใช้งานต้องศึกษารายละเอียดของโพรโทคอลการรับส่งข้อมูลของไอซีตัวนั้น ๆ จากแผ่นข้อมูล (Datasheet) ของผู้ผลิตให้เข้าใจก่อนใช้งาน

13.3 ไอซีขยายพอร์ตอินพุตเอาต์พุต PCF8574

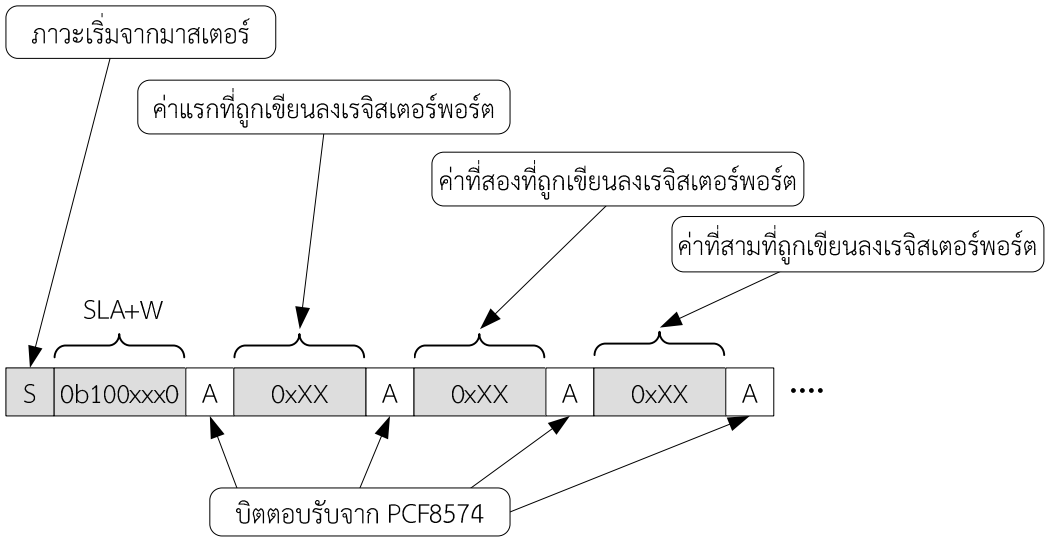
PCF8574 [7] เป็นไอซีที่ถูกออกแบบมาสำหรับเพิ่มขยายพอร์ตอินพุตเอาต์พุต มีพอร์ตขนาด 8 บิต ให้ใช้งานจำนวน 1 พอร์ต แต่ละบิตสามารถใช้เป็นอินพุตหรือเอาต์พุตได้อย่างอิสระ ตัวไอซีถูกออกแบบมาให้รองรับการเชื่อมต่อแบบทู-ไวร์ หรือไอสแควร์ซีได้ โดยมีค่าหมายเลขที่อยู่ A[6:3] เท่ากับ 0100_2 ส่วนบิต A[2:0] สามารถกำหนดเองโดยผู้ใช้งานได้โดยผ่านการป้อนค่าตรรกะเข้าที่ขาใช้งาน A[2:0] ของตัวไอซีเอง รูปที่ 13.13 แสดงวงจรใช้งานของไอซี PCF8574 และการต่อประสานกับตัวประมวลผลผ่านการเชื่อมต่อแบบทู-ไวร์



รูปที่ 13.13 วงจรใช้งานของไอซี PCF8574 เมื่อทำการติดต่อกับตัวประมวลผลผ่านการเชื่อมต่อแบบไอสแควร์ซี

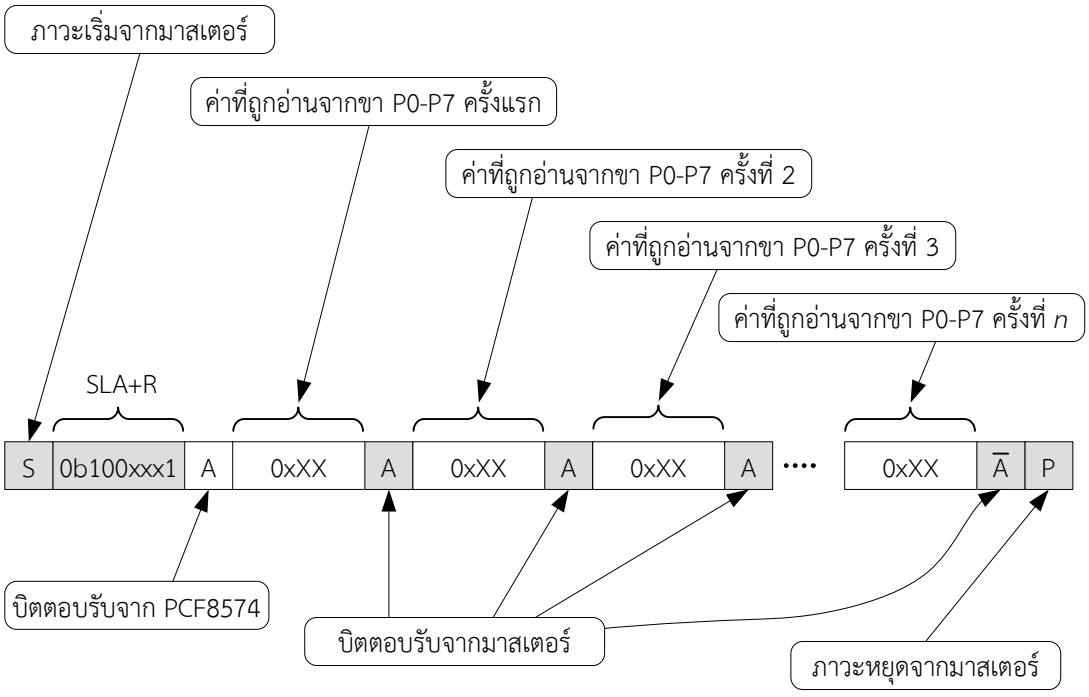
พอร์ตใช้งานขนาด 8 บิตของไอซี PCF8574 มีชื่อเรียกใช้งานว่า P0-P7 แต่ละขาถูกสร้างขึ้นมาให้มีการทำงานแบบกึ่งสองทิศทาง (Quasi-bidirectional) กล่าวคือ จะมีเรจิสเตอร์ขนาด 1 บิตต่ออยู่กับแต่ละขาโดยไม่มีเรจิสเตอร์ควบคุมทิศทางของพอร์ตเหมือนอย่างที่สถาปัตยกรรมเอวีอาร์ใช้ การใช้งานขาของพอร์ตใดเป็นอินพุตสามารถทำได้โดยการเขียนค่าตรรกะสูงมายังเรจิสเตอร์ของขาพอร์ตนั้น ๆ เสียก่อน และแม้ว่าแต่ละขาของพอร์ต P0-P7 จะสามารถใช้งานเป็นอินพุตหรือเอาต์พุตได้อย่างอิสระแยกจากกัน อย่างไรก็ดี การอ่านและเขียนค่าจากพอร์ตจะต้องกระทำครั้งละ 8 บิต หรือในหน่วยของไบต์ ยกตัวอย่างเช่น สมมุติว่าขา P0-P3 ถูกใช้งานเป็นอินพุตในขณะที่ P4-P7 ถูกใช้งานเป็นเอาต์พุต แต่การเขียนค่ามายัง P4-P7 ยังต้องกระทำในหน่วยของไบต์ นั่นหมายถึงจะต้องมีการเขียนค่าตรรกะที่ต้องการลงใน P4-P7 และเขียนค่าตรรกะสูงลงสู่ P0-P3 พร้อม ๆ กัน (การเขียนค่าตรรกะสูงลงสู่ P0-P3 ก็เพื่อให้สี่บิตล่างของพอร์ตสามารถทำงานเป็นพอร์ตรับเข้าได้อย่างถูกต้อง) และหากต้องมีการอ่านค่าจากขาพอร์ต P0-P3 ก็จะต้องทำในระดับไบต์ นั่นหมายถึงการอ่านค่าจากพอร์ตจะได้ค่าขนาด 8 บิต ซึ่งเป็นหน้าที่ของผู้เขียนโปรแกรมที่จะต้องนำค่าที่ได้ไปพรางค่าในสี่บิตบนของข้อมูลที่อ่านได้ทิ้งไปให้คงเหลือเพียงข้อมูลบิต P0-P3 ที่ใช้งานเป็นขาอินพุต

ในการเขียนข้อมูลลงสู่พอร์ตของไอซี PCF8574 ตัวประมวลผลจะต้องทำงานในแบบวิธีมาสเตอร์ทรานสมิตเตอร์ ในขณะที่ไอซี PCF8574 ต้องทำงานในแบบวิธีสเลฟรีซีฟเวอร์ โดยเริ่มจากการที่ตัวประมวลผลส่งภาวะเริ่ม ตามด้วยหมายเลขที่อยู่พร้อมบิตเขียน (SLA+W) หากมีการตอบรับจากไอซี PCF8574 ตัวประมวลผลก็สามารถส่งข้อมูลไบต์แรกให้กับสเลฟได้ เมื่อมีการตอบรับข้อมูลไบต์แรกจากสเลฟแล้ว ตัวประมวลผลก็สามารถส่งข้อมูลเอาต์พุตไบต์ถัดไปให้กับพอร์ตของไอซี PCF8574 ได้ดังแสดงในรูปที่ 13.14

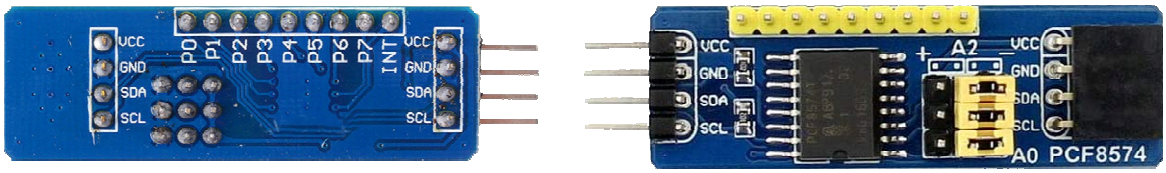


รูปที่ 13.14 การทำงานของไอซี PCF8574 ในแบบวิธีสเลฟรีซีฟเวอร์

ในการอ่านข้อมูลจากพอร์ตของไอซี PCF8574 ตัวประมวลผลจะต้องทำงานในแบบวิธีมาสเตอร์รีซีฟเวอร์ ในขณะที่ไอซี PCF8574 ต้องทำงานในแบบวิธีสเลฟทรานสมิตเตอร์ โดยเริ่มจากการที่ตัวประมวลผลส่งภาวะเริ่ม ตามด้วยหมายเลขที่อยู่พร้อมบิตอ่าน (SLA+R) หากมีการตอบรับจากไอซี PCF8574 แล้ว ตัวไอซีจะอ่านสภาวะทางตรรกะที่ขาของพอร์ตส่งกลับไปยังตัวประมวลผล เมื่อมีการตอบรับจากมาสเตอร์ ตัวไอซี PCF8574 ก็จะอ่านค่าสภาวะทางตรรกะที่ขาของพอร์ตส่งไปยังตัวประมวลผลอีกครั้ง และจะเป็นเช่นนี้ไปเรื่อย ๆ ตราบเท่าที่ตัวประมวลผลยังตอบรับข้อมูลจากสเลฟอยู่ ในกรณีที่ตัวประมวลผลต้องการหยุดการอ่านค่าจากพอร์ตของไอซี PCF8574 ก็สามารถทำได้โดยการไม่ตอบรับข้อมูลจากสเลฟและส่งภาวะหยุดลงสู่บัส ดังแสดงในรูปที่ 13.15



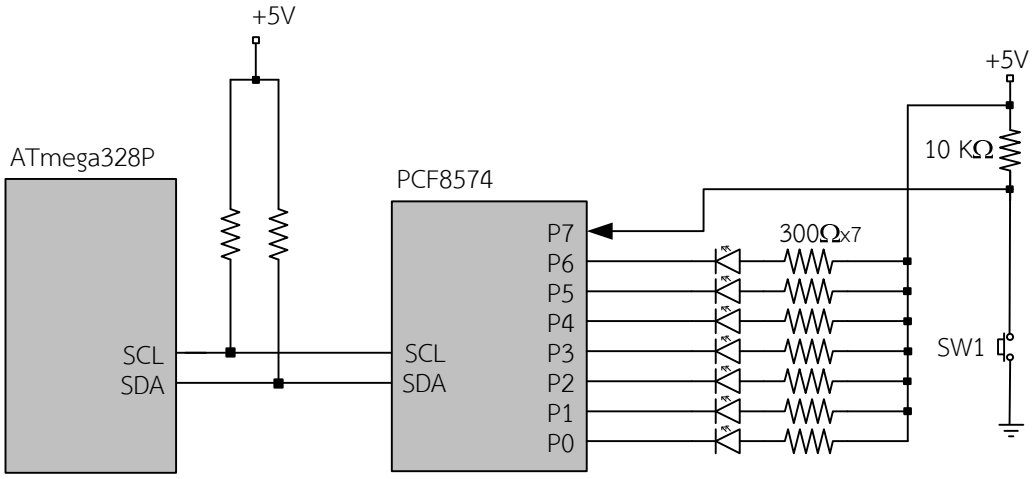
รูปที่ 13.15 การทำงานของไอซี PCF8574 ในแบบวิธีสเลฟทรานสมิตเตอร์



รูปที่ 13.16 มอดูล PCF-8574

13.4 การทดลองที่ 13.1

พิจารณาผังวงจรในรูปที่ 13.6 จากนั้นต่อวงจรตามรูปที่ 13.7 และทดลองนำโค้ดในรูปที่ 13.8 มาคอมไพล์ด้วยโปรแกรม Arduino IDE (ให้ดาวน์โหลดไฟล์ my_twi.h และ my_pcf8574.h จาก LMS มาใส่ในโฟลเดอร์เดียวกับโฟลเดอร์ที่โปรเจ็กต์ถูกจัดเก็บอยู่) จากนั้นอัพโหลดโปรแกรมลงบอร์ด Arduino UNO เมื่ออัพโหลดเสร็จให้เปิดโปรแกรม Serial Monitor จากนั้นตั้งค่าความเร็วของการรับส่งในโปรแกรม Serial Monitor เป็น 38400 bps ทดลองกดปุ่มบนสวิตช์กดติดปล่อยดับ สังเกตข้อความที่ขึ้นบนโปรแกรม Serial Monitor

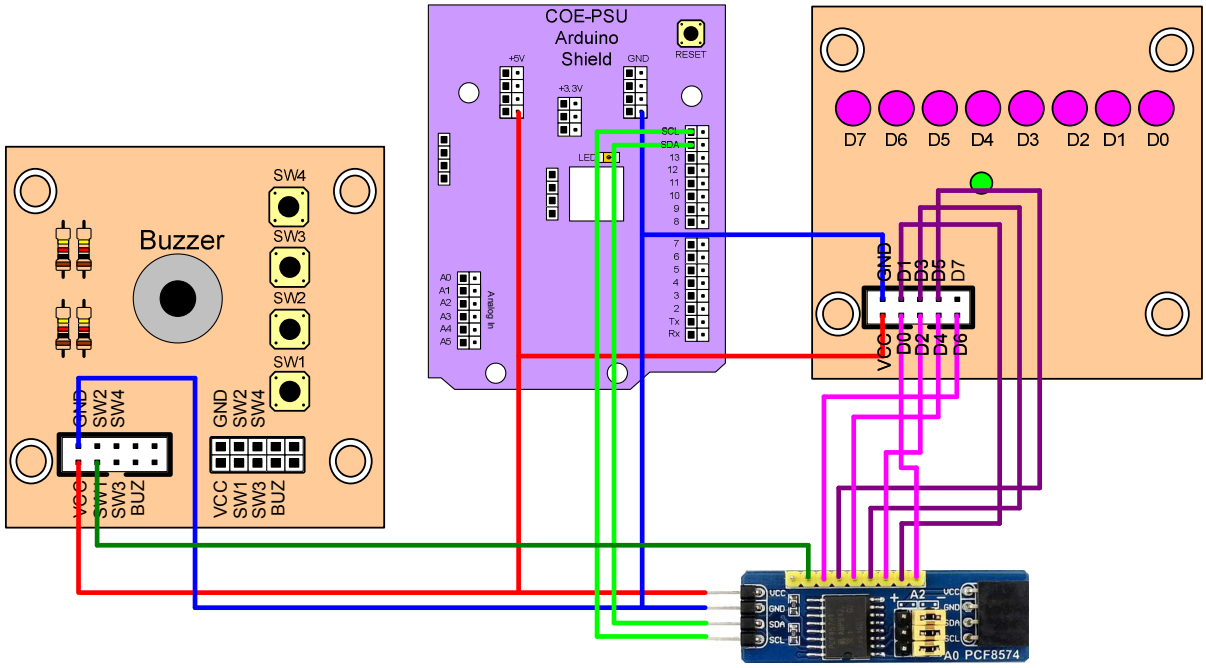


รูปที่ 13.6 ผังวงจรของการทดลองที่ 13.1-13.2 และ Checkpoint 13.1-13.2

```
#include <avr/io.h>                 //เรียกใช้คลังโปรแกรม io.h
#include <avr/interrupt.h>          //เรียกใช้คลังโปรแกรมสำหรับขัดจังหวะ
#include "my_twi.h"                 //เรียกใช้คลังโปรแกรมจัดการมอดูลทีดับเบิลยูไอ
#include "my_pcf8574.h"             //เรียกใช้คลังโปรแกรมจัดการไอซี PCF8574
uint8_t count;                      //ตั้งค่าตัวแปรส่วนกลาง (global variable)
unsigned long last_time;

uint8_t prepare_data(uint8_t d)
{
  uint8_t tmp;
  tmp = ~d;                 //กลับทุกบิตเป็นตรงข้ามเนื่องจาก LED เป็น Common A
  tmp |= 0x80;              //เขียนค่ารหัสขับแอลอีดีโดยมิให้มีผล
  return tmp;               //--กระทบต่อบิต MSB ที่ใช้เป็นอินพุต
}

void setup()
{
```

```
  last_time =0;
  Serial.begin(38400);
  uint8_t tmp, i;              //ประกาศตัวแปร tmp และตัวแปร i ขนาด 8 บิต
  init_twi_module();           //สั่งให้มอดูลทีดับเบิลยูไอเริ่มทำงาน
  count = 0;                   //สั่งให้ตัวแปร count มีค่าเริ่มต้นเท่ากับศูนย์
  tmp = prepare_data(count);
  PCF8574_write(0, tmp);
}                              //จบการทำงานของฟังก์ชันหลัก

void loop() {
  uint8_t tmp;

  tmp = PCF8574_read(0);
  tmp &= 0x80;
  if(!tmp)
  {
    if( millis() - last_time > 20)
    {
      last_time = millis();
      tmp = PCF8574_read(0);
      tmp &= 0x80;
      if (!tmp)
      {
        Serial.println("Keypressed detected");
      }
      do
      {
        tmp = PCF8574_read(0);
        tmp &= 0x80;
      }
      while(!tmp);
    }
  }
}
```

รูปที่ 13.8 โค้ดโปรแกรมสำหรับการทดลองที่ 13.1



รูปที่ 13.7 การต่อวงจรในการทดลองที่ 13.1-13.2 และ Checkpoint 13.1-13.2

13.5 การทดลองที่ 13.2

สร้างโปรเจ็กต์ใหม่ใน Arduino IDE จากนั้นนำโปรแกรมโค้ดในรูปที่ 13.9 มาวางลงในหน้าต่าง Editor ของโปรแกรม Arduino IDE คอมไพล์โปรแกรมและอัพโหลดโปรแกรมลงสู่บอร์ด Arduino UNO สังเกตการทำงานของวงจรและการเปลี่ยนแปลงของแอลอีดีบนบอร์ด

13.6 Checkpoints

การทดลองนี้ประกอบด้วย 3 Checkpoint โดยคิดคะแนนดังนี้

Checkpoint 13.1	35 %
Checkpoint 13.2	35 %
Checkpoint 13.3 (bonus)	10 %
รายงานสรุปผลและวิเคราะห์ผลการทดลองแต่ละ Checkpoint (รวม 3 Checkpoint)	30%

ในส่วนของรายงานนั้น ให้ทำการสรุปผลการทดลองของแต่ละ Checkpoint และอธิบายการทำงานของโค้ดที่เขียนขึ้นอย่างละเอียด เฉพาะการทดลองใน Checkpoint 13.1-13.2 ให้ใช้วงจรในรูปที่ 13.6 และ 13.7

6.6.1 Checkpoint 13.1

จงเขียนโปรแกรมควบคุมวงจรในรูปที่ 13.6 และ 13.7 (โดยนำโค้ดในรูปที่ 13.8 และ 13.9 มาดัดแปลง) กำหนดให้แอลอีดีดับทุกดวงเมื่อวงจรเริ่มทำงาน จากนั้นเมื่อมีการกดสวิตช์จะมีการเพิ่มของค่าที่แสดงผลของแอลอีดีขึ้นหนึ่งค่า โดยแอลอีดีทั้ง 7 ดวงจะทำหน้าที่เป็นวงจรนับขนาด 7 บิต และมีการเพิ่มค่าขึ้นหนึ่งค่าทุกครั้งที่มีการกดสวิตช์ หากแอลอีดีติดทุกดวง (แสดงค่า 127_{10}) แล้วหากพบการกดอีกหนึ่งครั้งก็จะส่งผลให้แอลอีดีกลับมาแสดงผลค่า 0 (ดับทุกดวง) ใหม่อีกครั้ง

```
#include <avr/io.h>            //เรียกใช้คลังโปรแกรม io.h
#include <avr/interrupt.h>     //เรียกใช้คลังโปรแกรมสำหรับขัดจังหวะ
#include "my_twi.h"            //เรียกใช้คลังโปรแกรมจัดการมอดูลทีดับเบิลยูไอ
#include "my_pcf8574.h"        //เรียกใช้คลังโปรแกรมจัดการไอซี PCF8574
                               //---ส่วนบิตอื่นที่เหลือเป็นเอาต์พุต
```

```
uint8_t count;              //ตั้งค่าตัวแปรส่วนกลาง (global variable)
unsigned long last_time;

uint8_t prepare_data(uint8_t d)
{
  uint8_t tmp;
  tmp = ~d;                 //กลับทุกบิตเป็นตรงข้ามเนื่องจาก LED เป็น Common A
  tmp |= 0x80;              //เขียนค่ารหัสขับแอลอีดีโดยมิให้มีผล
  return tmp;               //--กระทบต่อบิต MSB ที่ใช้เป็นอินพุต
}

void setup()
{
  last_time =0;
  Serial.begin(38400);
  uint8_t tmp, i;           //ประกาศตัวแปร tmp และตัวแปร i ขนาด 8 บิต
  init_twi_module();        //สั่งให้มอดูลทีดับเบิลยูไอเริ่มทำงาน
  count = 0;                //สั่งให้ตัวแปร count มีค่าเริ่มต้นเท่ากับศูนย์
  tmp = prepare_data(count);
  PCF8574_write(0, tmp);
}                           //จบการทำงานของฟังก์ชันหลัก

void loop(){

  uint8_t tmp;
  delay(400);
  count = 0;
  tmp = prepare_data(count);
  PCF8574_write(0, tmp);
  delay(400);
  count = 127;
  tmp = prepare_data(count);
  PCF8574_write(0, tmp);
}
```

รูปที่ 13.9 โค้ดโปรแกรมสำหรับการทดลองที่ 13.2

6.6.2 Checkpoint 13.2

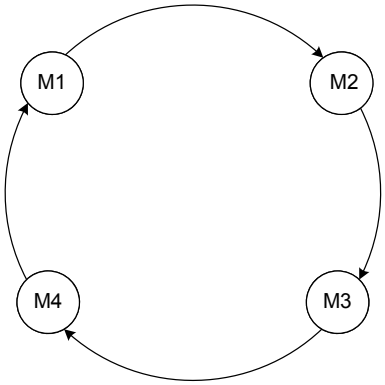
จงเขียนโปรแกรมควบคุมวงจรในรูปที่ 13.6 และ 13.7 โดยวงจรมีการทำงาน 4 โหมด ได้แก่ โหมด M1, M2, M3 และ M4 โดยแต่ละโหมดมีการทำงานดังนี้

- M1 แอลอีดีเพิ่มค่าขึ้นทีละ 1 ค่า (นับขึ้นแบบไบนารี) โดยนับตั้งแต่ 0-127 และวกกลับเป็น 0 ใหม่เมื่อค่านับจนถึงค่า 127
- M2 แอลอีดีสว่างทีละดวง และเลื่อนไปทางขวาทุก ๆ วินาที เมื่อเลื่อนไปขวาสุดแล้วก็เริ่มสว่างที่หลอดซ้ายสุดใหม่

- M3 แอลอีดีสว่างทีละดวง และเลื่อนไปทางซ้ายทุก ๆ วินาที เมื่อเลื่อนไปซ้ายสุดแล้วก็เริ่มสว่างที่หลอดขวาสุดใหม่

- M4 แอลอีดีติดทุกดวง สลับกับดับทุกดวง ทุก ๆ หนึ่งวินาที

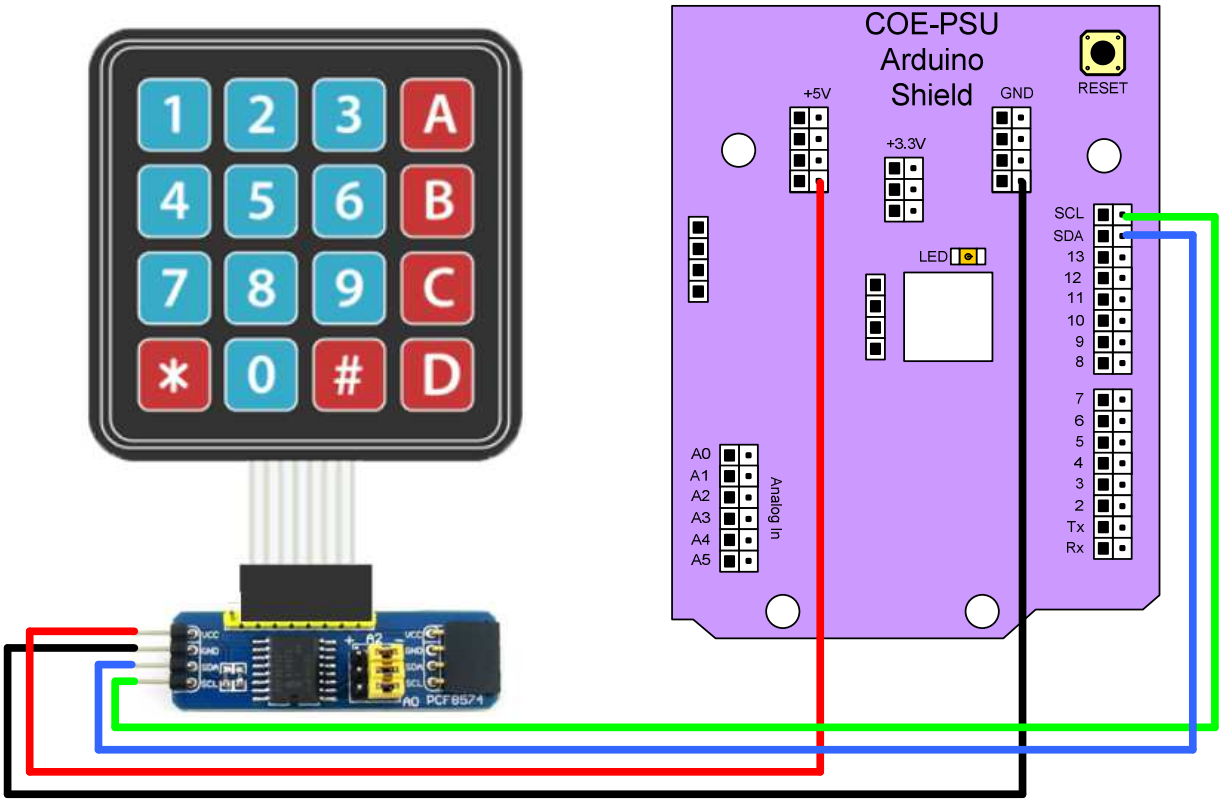
กำหนดให้เมื่อเริ่มโปรแกรม ให้วงจรทำงานที่โหมด M1 เมื่อมีการกดสวิตช์แต่ละครั้งจะมีการเลื่อนโหมดดังรูปที่ 13.10



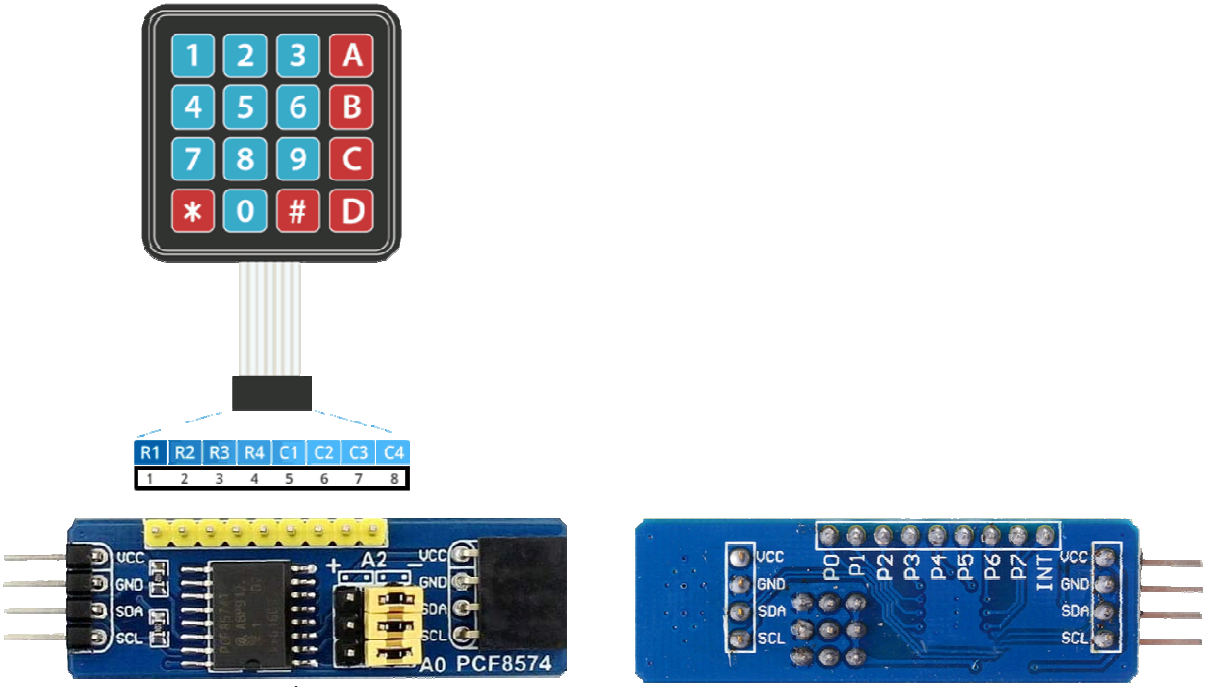
รูปที่ 13.10 การเปลี่ยนโหมดของวงจรเมื่อมีการกดสวิตช์แต่ละครั้ง

6.6.3 Checkpoint 13.3

จงต่อวงจรดังรูปที่ 13.17 และ 13.18 และนำโปรแกรมในรูปที่ 13.19 คอมไพล์บน Arduino IDE จากนั้นกดปุ่มบนเมมเบรนคีย์แพ็ด จากนั้นสังเกตข้อความที่ขึ้นบนโปรแกรม Serial Monitor



รูปที่ 13.17 การอินเตอร์เฟสคีย์แพ็ดกับ Arduino UNO ผ่านมอดูล PCF-8574



รูปที่ 13.18 การแมปขาของคีย์แพ็ดกับขา P0-P7 ของมอดูล PCF-8574

```
1  #include <Keypad_I2C.h>
2  #include <Keypad.h>
3  #include <Wire.h>
4
5  #define I2CADDR 0x20
6
7  const byte ROWS = 4;
8  const byte COLS = 4;
9
10 char hexaKeys[ROWS][COLS] = {
11   {'1','2','3','A'},
12   {'4','5','6','B'},
13   {'7','8','9','C'},
14   {'*','0','#','D'}
15 };
16 byte rowPins[ROWS] = {7, 6, 5, 4};
17 byte colPins[COLS] = {3, 2, 1, 0};
18
19 Keypad_I2C keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS, I2CADDR);
20
21 void I2C_bus_scan(void);
22
```

รูปที่ 13.19 ตัวอย่างการเขียนโปรแกรมเพื่ออ่านค่าจากคีย์แพดซึ่งเชื่อมต่ออยู่กับมอดูล PCF-8574

```
23  void setup(){
24    Wire.begin( );
25    keypad.begin( );
26    Serial.begin(38400);
27    I2C_bus_scan();
28  }
29
30  void loop(){
31    char key = keypad.getKey();
32    if (key != NO_KEY){
33      Serial.print(key);
34      Serial.println(" is pressed");
35    }
36  }
37
38  void I2C_bus_scan(void)
39  {
40    Serial.println ();
41
42    Serial.println ("www.9arduino.com ...");
      Serial.println ("I2C scanner. Scanning ...");
43    byte count = 0;
44
45    Wire.begin();
46    for (byte i = 8; i < 120; i++)  // Loop ค้นหา Address
47    {
48      Wire.beginTransmission (i);
49      if (Wire.endTransmission () == 0)
50        {
51        Serial.print ("Found address: ");
52        Serial.print (i, DEC);
53        Serial.print (" (0x");
54        Serial.print (i, HEX);
55        Serial.println (")");
56        count++;
57        delay (1);
58        }
59    }
60    Serial.println ("Done.");
61    Serial.print ("Found ");
62    Serial.print (count, DEC);
63    Serial.println (" device(s).");
64  }
```

รูปที่ 13.19 ตัวอย่างการเขียนโปรแกรมเพื่ออ่านค่าจากคีย์แพดซึ่งเชื่อมต่ออยู่กับมอดูล PCF-8574 (ต่อ)

จากโค้ดบรรทัดที่ 17 ในรูปที่ 13.19 เป็นการระบุให้ขา Row1, Row2, Row3, Row4 ถูกแม็ปกับขา P7, P6, P5, P4 ของ PCF8574 ตามลำดับ

ส่วนโค้ดบรรทัดที่ 18 ในรูปที่ 13.19 เป็นการระบุให้ Column1, Column2, Column3, Column4 ถูกแม็บเข้ากับขา P3, P2, P1, P0 ของ PCF-8574 ตามลำดับ ส่วนรูปแบบการเรียงขาของคีย์แพ็ดเป็นไปตามที่แสดงในรูปที่ 13.18

เอกสารอ้างอิง

[1] Atmel Corporation, “AVR Libc Reference Manual: Example using the two-wire interface,” Jan. 4, 2018. [Online]. Available: https://www.microchip.com/webdoc/AVRLibcReferenceManual/group__twi__demo_1twi_demo_project.html [Accessed Dec. 12, 2018].

[2] Wikipedia: The Free Encyclopedia, “I^2C,” Oct, 30, 2018. [Online]. Available: https://en.wikipedia.org/wiki/I%C2%B2C [Accessed Dec. 15, 2018]

[3] Lapis Semiconductor, “MR44V064B: 64k(8,192-Word × 8-Bit) FeRAM (Ferroelectric Random Access Memory) I2C,” MR44V064B datasheet, January, 2016.

[4] Atmel Corporation, “8-bit Microcontroller with 4/8/16/32KBytes In-System Programmable Flash,” ATmega48A/48PA/88A/88PA/168A/168PA/328/328 datasheet, August, 2010, p. 222.

[5] Atmel Corporation, “8-bit Microcontroller with 4/8/16/32KBytes In-System Programmable Flash,” ATmega48A/48PA/88A/88PA/168A/168PA/328/328 datasheet, August, 2010, p. 224.

[6] Maxim Integrated Products Inc., “DS1307: 64 x 8, Serial, I2C Real-Time Clock,” DS1307 datasheet, March, 2015.

[7] NXP Semiconductors Inc., “PCF8574; PCF8574A Remote 8-bit I/O expander for I2C-bus with interrupt,” PCF8574 datasheet, May, 2013.