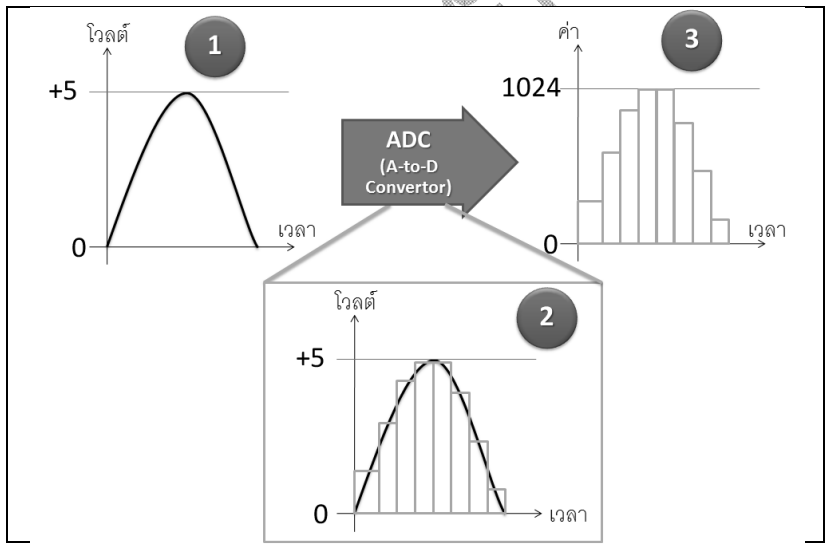
การแปลงแอนะล็อกเป็นดิจิทัล

1. บทนำ

โมดูลเอดีซีมีหน้าที่แปลงสัญญาณแอนะล็อกซึ่งเป็นสัญญาณแบบต่อเนื่อง (Continuous Signal) จากขานำสัญญาณเข้าเป็นสัญญาณแบบไม่ต่อเนื่อง (Discrete Signal) ที่เรียกว่าดิจิทัลในรูปแบบของตัวเลขดิจิทัล ดังรูปที่ 13.1

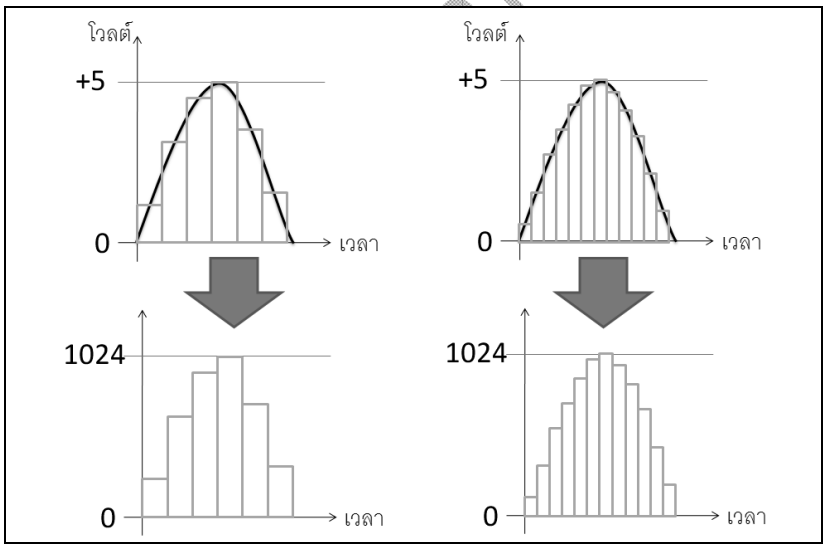
รูปที่ 13.1 ขั้นตอนแปลงสัญญาณแอนะล็อกเป็นดิจิทัล



จากรูปที่ 13.1 แสดงให้เห็นขั้นตอนของการแปลงสัญญาณแอนะล็อกเป็นสัญญาณดิจิทัล คือ (1) เมื่อมีสัญญาณแบบแอนะล็อกส่งเข้ามาส่วนตัวแปลงที่เรียกว่าเอดีซี (2) ตัวแปลงจะเปรียบเทียบแรงดันเป็นค่าตัวเลขจำนวนเต็มแบบดิจิทัล และ (3) ตัวแปลงสัญญาณจะส่งข้อมูลที่แปลงเป็นค่าตัวเลขดิจิทัลกลับออกมา

ข้อสังเกตจากการทำงานของเอดีซีจะพบว่าโมดูลเอดีซีมีความละเอียดที่มากจะทำให้ได้ค่าตัวเลขดิจิทัลที่ละเอียดขึ้น เช่น ถ้าโมดูลเอดีซีทำงานแบบ 8 บิต จะได้ค่าดิจิทัลอยู่ในช่วงค่า 0 ถึง 255 ถ้าโมดูลเอดีซีทำงานแบบ 10 บิต จะได้ค่าดิจิทัลอยู่ในช่วงค่า 0 ถึง 1024 เป็นต้น นอกจากนี้ความเร็วในการทำงานของโมดูลเอดีซีมีผลต่อความละเอียดของการแปลงค่า เช่น ถ้าโมดูลเอดีซีตัวหนึ่งทำงานที่ความเร็ว 10 มิลลิวินาที และอีกโมดูลหนึ่งทำงานด้วยความเร็ว 5 มิลลิวินาทีจะทำให้โมดูลเอดีซีตัวที่สองนั้นมีความละเอียดในการทำงานสูงกว่า และได้จำนวนข้อมูลที่มากกว่าโมดูลเอดีซีตัวแรกถึง 2 เท่า ดังรูปที่ 13.2 ดังนั้น โมดูลเอดีซีที่ทำงานได้ดีจะต้องมีความละเอียดของการแปลงที่มาก และมีความเร็วในการแปลงค่าที่เร็ว

รูปที่ 13.2 เปรียบเทียบโมดูลเอดีซีที่มีความเร็วที่แตกต่างกัน



ในแต่ละรุ่นของไมโครคอนโทรลเลอร์ PIC มีคุณสมบัติพิเศษเกี่ยวกับการแปลงสัญญาณแอนะล็อกเป็นดิจิทัลมีคุณสมบัติของโมดูลเอดีซี ดังตารางที่ 13.1

ตารางที่ 13.1 เปรียบเทียบคุณสมบัติด้านโมดูลเอดีซีของไมโครคอนโทรลเลอร์ PIC

ไมโครคอนโทรลเลอร์	ขนาด (บิต)	จำนวน ช่องสัญญาณ	พอร์ตนำสัญญาณเข้า (ชื่อขา:ชื่อช่องสัญญาณ)
PIC16F877	10	8	RA0:AN0 RA1:AN1 RA2:AN2 RA3:AN3 RA5:AN4 RE0:AN5 RE1:AN6 RE2:AN7
PIC18F458	10	8	
PIC18F8722	10	16	RA0:AN0 RA1:AN1 RA2:AN2 RA3:AN3 RA5:AN4 RF0:AN5 RF1:AN6 RF2:AN7 RF3:AN8 RF4:AN9 RF5:AN10 RF6:AN11 RH4:AN12 RH5:AN13 RH6:AN14 RH7:AN15
PIC18F8628	12	16	

2. โมดูลเอดีซี

การทำงานของโมดูลเอดีซีของไมโครคอนโทรลเลอร์ PIC ใช้เรจิสเตอร์ 4 ตัว คือ ADRESH ADRESL ADCON0 และ ADCON1 โดยแต่ละตัวทำหน้าที่ดังต่อไปนี้

1. เรจิสเตอร์ ADRESH เก็บค่าไบต์สูงของผลลัพธ์จากการแปลงค่าจากโมดูลเอดีซี
2. เรจิสเตอร์ ADRESL เก็บค่าไบต์ต่ำของผลลัพธ์จากการแปลงค่าจากโมดูลเอดีซี
3. เรจิสเตอร์ ADCON0 ทำหน้าที่กำหนดค่าการทำงานและเปิดหรือปิดการทำงานของโมดูลเอดีซี
4. เรจิสเตอร์ ADCON1 ทำหน้าที่กำหนดรูปแบบของการเก็บข้อมูลจากการแปลงค่าแอนะล็อกเป็นดิจิทัล และกำหนดบทบาทของขาสัญญาณที่เกี่ยวข้องกับโมดูลเอดีซี

หมายเหตุ

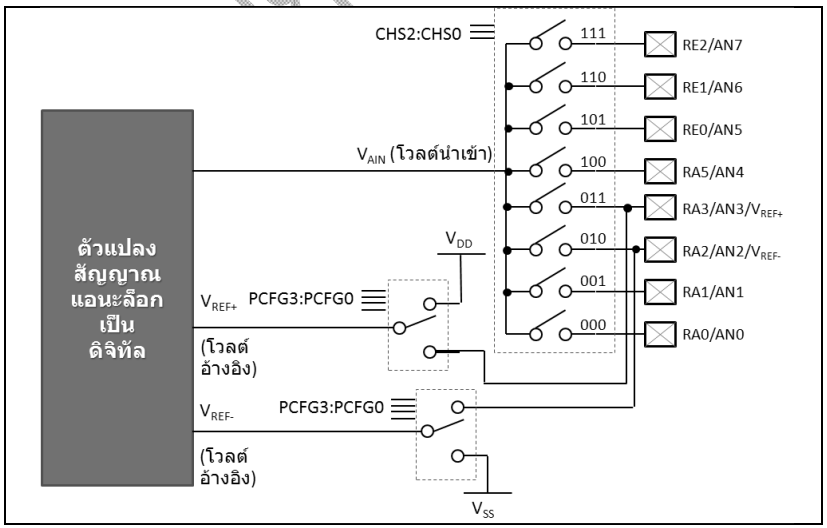
รายละเอียดของเรจิสเตอร์ ADCON0 ADCON1 และรูปแบบการจัดเก็บข้อมูลจากการทำงานของโมดูลเอดีซีอ่านได้จากบทที่ 7

ในกระบวนการทำงานของโมดูลเอดีซีจะจัดเก็บข้อมูลขนาด 10 บิตไว้ในเรจิสเตอร์ ADRESH:ADRESL ซึ่งการสั่งให้เริ่มต้นให้ทำงานนั้นจะต้องกำหนดให้บิต GO/DONE' ของเรจิสเตอร์ ADCON0 มีค่าเป็น 1 หลังจากนั้นต้องรอจนกว่าบิตนี้จะเปลี่ยนสถานะไปเป็น 0 ซึ่งหมายถึงโมดูลเอดีซีดำเนินการแปลงค่าสำเร็จ หลังจากนั้นจึงสามารถนำค่าที่เก็บในเรจิสเตอร์ทั้งสองไปใช้งานได้

2.1 การเลือกช่องสัญญาณนำเข้าข้อมูลแอนะล็อก

จากผังการกำหนดการทำงานของขาเพื่อใช้กับโมดูลเอดีซี (รูปที่ 13.3) จะพบว่าการเลือกขาหรือช่องสัญญาณนำเข้าข้อมูลแอนะล็อกจะถูกกำหนดโดยบิต CHS0 CHS1 และ CHS2 ของ ADCON0 การกำหนดโวลต์อ้างอิงทั้ง V_{REF+} และ V_{REF-} ถูกควบคุมด้วยบิต PCFG0 PCFG1 PCFG2 และ PCFG3 ของเรจิสเตอร์ ADCON1 ซึ่งใช้ขา RA3 หรือ RA2 เป็น V_{REF+} และ V_{REF-} จะทำให้ช่องสัญญาณสำหรับนำสัญญาณเข้าแบบแอนะล็อกน้อยกว่า 8 ช่องสัญญาณ นอกจากนี้ถ้าต้องการนำเข้าหลายช่องสัญญาณจะต้องทยอยอ่านทีละช่องสัญญาณสลับกันไป เนื่องจากการเลือกช่องสัญญาณกระทำได้ครั้งละ 1 ช่องสัญญาณเท่านั้น

รูปที่ 13.3 ผังการกำหนดการทำงานของขาเพื่อใช้กับโมดูลเอดีซี

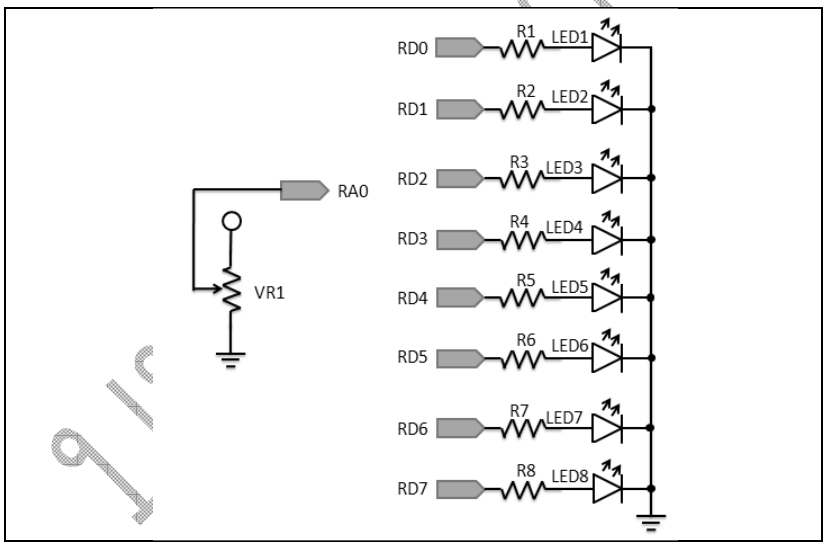


หมายเหตุ

เมื่อเลือกขาใดให้ทำหน้าที่นำสัญญาณเข้าแบบแอนะล็อกจะต้องกำหนดให้ขานั้นเป็นขานำเข้าสัญญาณด้วย เช่น กำหนดให้ขา RA0 เป็นขานำสัญญาณเข้าแบบแอนะล็อก จะต้องกำหนดให้ TRISA มีค่าเป็น 00000001_2

วงจรสำหรับทดสอบการทำงานของโมดูลเอดีซีเป็นดังรูปที่ 13.4 ซึ่งเป็นการต่อตัวต้านทานปรับค่าได้เข้ากับขา RA0 หรือช่องอ่านสัญญาณแอนะล็อกช่องที่ 0 (AN0) และต่อเชื่อมขาของพอร์ตดีเข้ากับวงจรขับแอลอีดี 8 ดวง เพื่อใช้เป็นส่วนแสดงผลลัพธ์

รูปที่ 13.4 วงจรทดลองการทำงานของโมดูลเอดีซี



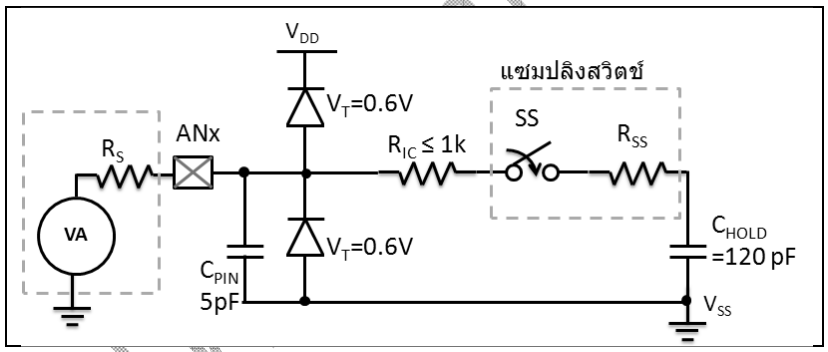
จากวงจรรูปที่ 13.4 มีรายการอุปกรณ์ที่ต้องใช้ดังนี้

1. ตัวต้านทานปรับค่าได้ VR1 ความต้านทาน 10 กิโลโอมห์
2. ตัวต้านทาน R1 ถึง R8 ความต้านทาน 270 โอมห์
3. แอลอีดี LED1 ถึง LED8 สีแดง

2.2 การคำนวณค่าเวลาที่ต้องใช้ก่อนเริ่มการแปลงค่า

การนำเข้าสัญญาณแอนะล็อกของโมดูลเอดีซีมีต้องเสียเวลากับการแปลงค่าแรงดันที่นำเข้าเพื่อให้เป็นค่าแบบดิจิทัลขนาด 10 บิต (12 บิต สำหรับบางรุ่น) โดยวงจรของการทำงานเป็นดังรูปที่ 13.5 นั่นคือ เมื่อนำเข้าสัญญาณแอนะล็อก VA ที่ต่อพ่วงกับตัวต้านทาน R_S ผ่านทางขา ANx มาทางสวิตช์ชักตัวอย่างหรือแซมปลิงสวิตช์ (Sampling Switch) R_{SS} เพื่อชาร์จประจุเข้าตัวเก็บประจุ C_{HOLD} ที่มีความจุ 120 พิโกฟารัดจนเต็มจึงทำให้การแปลงค่ามีความเที่ยงตรง ซึ่งแซมปลิงสวิตช์ R_{SS} ทำหน้าที่วัดความต้านทานต่อการแปรปรวนของกระแสที่มาจากแหล่งกำเนิดกระแสของอุปกรณ์ V_{DD}

รูปที่ 13.5 ตัวแบบการนำเข้าสัญญาณแอนะล็อก



โดยที่ C_{PIN} คือ ตัวเก็บประจุกระแสที่นำเข้า

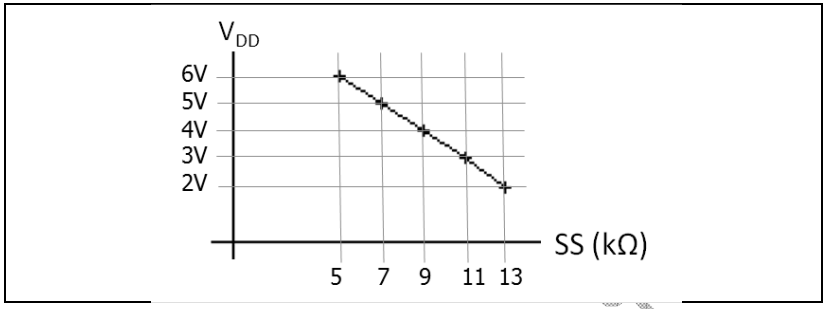
V_T คือ ค่าโวลต์ที่เป็นตัวขีดแบ่ง (Threshold Voltage)

R_{IC} คือ ความต้านของการเชื่อมโยง (Interconnect Resistance) กับแซมปลิงสวิตช์

C_{HOLD} คือ ตัวเก็บประจุที่เก็บประจุที่แบ่งมาจาก V_{DD}

SS คือ แซมปลิงสวิตช์ ซึ่งมีค่าความต้านทานที่สัมพันธ์กับค่าของ V_{DD} ดังรูปที่ 13.6

รูปที่ 13.6 ความสัมพันธ์ระหว่าง VDD กับ ค่าความต้านทานของแซมปลิงสวิตช์



หมายเหตุ

1. ค่าอิมพีแดนซ์สูงสุดของแหล่งนำเข้าสัญญาณเข้าแบบแอนะล็อก R_S ที่แนะนำ คือมีความต้านทาน 10 กิโลโอมห์

2. เนื่องจากใช้ V_{DD} 5 โวลต์ ค่าของ R_{SS} จึงเป็น 7 กิโลโอมห์

ก่อนที่จะเริ่มต้นการแปลงนำเข้าสัญญาณแอนะล็อกจากขาสัญญาณที่ถูกเลือกให้เป็นค่าตัวเลขดิจิทัล จะต้องรอเป็นช่วงระยะเวลาหนึ่งเรียกว่า T_{ACQ} ซึ่งคำนวณได้จากสมการต่อไปนี้

T_{ACQ} = เวลาเริ่มต้นของตัวขยายสัญญาณ +
เวลาที่ใช้ในการชาร์จตัวเก็บประจุ +
สัมประสิทพธ์อุณหภูมิ

= $T_{AMP} + T_C + T_{COFF}$

= 2 ไมโครวินาที + T_C +
[(อุณหภูมิ-25 องศาเซลเซียส) ×
(0.05ไมโครวินาที/องศาเซลเซียส)]

ซึ่ง T_C = $C_{HOLD} \times (R_{IC} + R_{SS} + R_S) \times \ln(\frac{1}{2047})$

= -120 ไมโตรฟารัด $\times$ (1กิโลโอมห์ + 7 กิโลโอมห์ + 10 กิโลโอมห์) $\times$
$\ln(\frac{1}{2047})$

= 16.47 ไมโครวินาที

ดังนั้น T_{ACQ} = 2 ไมโครวินาที + 16.47 ไมโครวินาที +
[(50 – 25 องศาเซลเซียส) × (0.05ไมโครวินาที/องศาเซลเซียส)]

= 19.72 ไมโครวินาที

ด้วยเหตุนี้ ก่อนเริ่มมีการแปลงค่าจากสัญญาณแอนะล็อกให้เป็นค่าตัวเลขดิจิทัลจะต้องหน่วงเวลาเป็นอย่างน้อย 19.72 ไมโครวินาที เสมอ

หมายเหตุ

เนื่องการเขียนด้วยภาษาเบสิกมีเรียกใช้คำสั่งหลายคำสั่งที่ผู้เขียนโปรแกรมมองไม่เห็น ดังนั้น จึงอาจจะไม่จำเป็นต้องหน่วงเวลาก่อนสั่งให้โมดูลเอดีซีเริ่มทำงาน

2.3 การเลือกสัญญาณนาฬิกาสำหรับโมดูลเอดีซี

โมดูลเอดีซีของไมโครคอนโทรลเลอร์ PIC มีค่าเวลาของการแปลงต่อบิต (conversion time per bit) ที่เรียกกว่า T_{AD} ซึ่งไมโครคอนโทรลเลอร์ต้องการอย่างน้อย $12T_{AD}$ ต่อการแปลงข้อมูล 10 บิต โดยที่ขั้นตอนการแปลงนี้ใช้สัญญาณนาฬิกาสำหรับใช้ควบคุมการแปลงค่าตามที่กำหนดไว้โดยผู้เขียนโปรแกรมด้วยการกำหนดค่าแก่บิต ADCS0 และ ADCS1 ของเรจิสเตอร์ ADCON0 อันเป็นรายการใดรายการหนึ่งจากรายการต่อไปนี้

$2T_{OSC}$ ซึ่งมีค่าเท่ากับ $\frac{F_{OSC}}{2}$

$8T_{OSC}$ ซึ่งมีค่าเท่ากับ $\frac{F_{OSC}}{8}$

$32T_{OSC}$ ซึ่งมีค่าเท่ากับ $\frac{F_{OSC}}{32}$

หรือจากวงจรอาร์ซีที่อยู่ภายในโมดูลเอดีซี ซึ่งใช้เวลาประมาณ 2-6 ไมโครวินาที

ถ้าใช้ความถี่สัญญาณนาฬิกา F_{OSC} ที่ 10 เมกะเฮิรตซ์จะได้ค่าเวลาของ $2T_{OSC}$ $8T_{OSC}$ และ $32T_{OSC}$ ดังตารางที่ 13.2 ซึ่งคำนวณมาจากสมการต่อไปนี้

คาบเวลาของการแปลงค่า $= \frac{n}{10{,}000{,}000} \times 1{,}000{,}000$ ไมโครวินาที

$= \frac{n}{10}$ ไมโครวินาที

โดยที่ n คือ ค่านำหน้า T_{OSC} ซึ่งมีค่าเป็น 2 8 หรือ 32

ตารางที่ 13.2 ค่า T_{AD} เทียบกับคาบเวลาที่ใช้แปลงค่า

แหล่งสัญญาณนากาสำหรับโมดูลเอดีซี (T_{AD}) รายการทำงาน	ADCS1:ADCS0	คาบเวลาของการแปลงค่า (ไมโครวินาที)
$2T_{OSC}$	00	0.2
$8T_{OSC}$	01	0.8
$32T_{OSC}$	10	3.2
RC	11	2-6

หมายเหตุ

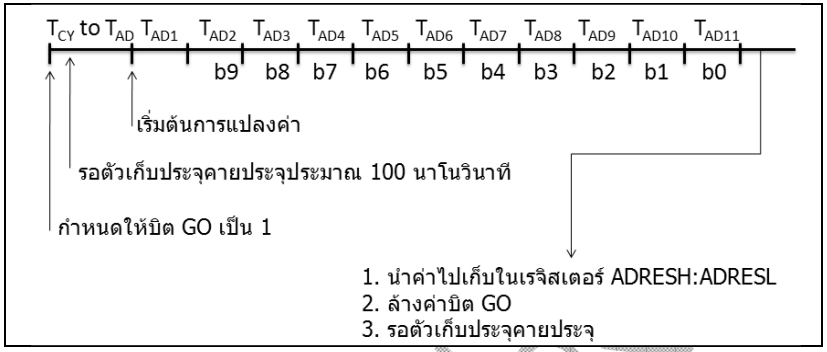
คาบเวลาของการแปลงค่าต้องมากกว่า 1.6 ไมโครวินาที ดังนั้น ถ้าใช้ F_{OSC} 10 เมกะเฮิรตซ์ T_{AD} ที่ใช้ได้มีเพียง 2 แบบ คือ $32T_{OSC}$ หรือ RC

2.4 การแปลงจากแอนะล็อกเป็นดิจิทัล

เมื่อการทำงานของโมดูลเอดีซีถูกยกเลิกด้วยการล้างค่าบิต GO/DONE' ของเรจิสเตอร์ ADCON0 ให้เป็น 0 ขณะที่ทำการแปลงสัญญาณแอนะล็อกให้เป็นข้อมูลดิจิทัลอยู่นั้น จะทำให้เรจิสเตอร์ ADRESL และ ADRESH เก็บค่าที่ไม่สมบูรณ์ เนื่องจากมีการแปลงค่าเพียงบางส่วนแล้วถูกยกเลิกก่อนการทำงานจบกระบวนการ

จากที่ได้กล่าวในหัวข้อของการเลือกสัญญาณนาฬิกาสำหรับโมดูลเอดีซีที่ระบุไว้ว่าการแปลงค่าทั้ง 10 บิตจะต้องใช้เวลาทั้งสิ้น $12T_{AD}$ เมื่อมีการยกเลิกการทำงานระหว่างการแปลงค่า จะต้องรอเป็นเวลา $2T_{AD}$ จึงจะเริ่มต้นอ่านค่าในรอบถัดไปได้ ซึ่งคาบเวลาของการแปลงค่าทั้ง 10 บิต เป็นดังรูปที่ 13.7

รูปที่ 13.7 รอบของ T_{AD} ในการแปลงสัญญาณแอนะล็อกเป็นค่าดิจิทัล



จากรูปที่ 13.7 จะพบว่าหลังจากที่บิต GO ถูกกำหนดให้เป็น 1 จะต้องใช้เวลาประมาณ T_{CY} ถึง T_{AD} เพื่อเตรียมความพร้อมของการแปลงค่า และจะได้รับค่าบิตที่ 9 หรือ b9 หลังสิ้นสุดเวลา T_{AD2} ไปจนได้รับค่าบิตที่ 0 หรือ b0 หลังจากสิ้นสุดเวลา T_{AD11} เมื่อแปลงค่าเสร็จสิ้นสมบูรณ์โมดูลเอดีซีจะแบ่งผลลัพธ์ขนาด 10 บิตเป็นสองส่วน เพื่อนำไปเก็บในเรจิสเตอร์ ADRESH และ ADRESL ตามรูปแบบของการจัดเรียงที่กำหนดในบิต ADFM ของเรจิสเตอร์ ADCON1

ดังนั้น ถ้าเลือกใช้สัญญาณนาฬิกาจากวงจรอาร์ซีภายในโมดูลเอดีซีจะต้องใช้เวลาในการแปลงค่าเท่ากับ 12 x 6 ไมโครวินาที = 72 ไมโครวินาที

หมายเหตุ

ห้ามกำหนดให้บิต GO/DONE' ของเรจิสเตอร์ ADCON0 มีค่าเป็น 1 ในชุดคำสั่งเดียวกับการเปิดให้โมดูลเอดีซีทำงาน

2.5 การทำงานเมื่ออยู่ในโหมดประหยัดพลังงาน

โมดูลเอดีซีทำงานได้แม้อยู่ในโหมดประหยัดพลังงานด้วยการใช้สัญญาณนาฬิกาประเภทอาร์ซีภายในโมดูลเอง โดยที่โมดูลเอดีซีต้องรอการทำงานหนึ่งรอบก่อนเริ่มต้นแปลงค่าเพื่อยอมให้คำสั่ง SLEEP ถูกเรียกขึ้นมาทำงาน เมื่อการแปลงค่าเสร็จสิ้นค่าของบิต GO/DONE' จะถูกล้างค่าแล้วผลลัพธ์จากการแปลงจะถูกนำไปเก็บในเรจิสเตอร์ ADRESH และ ADRESL ตามปกติแล้วโมดูล

เอดีซีจะหยุดทำงาน (ถึงแม้บิต ADON จะมีค่าเป็น 1 ก็ตาม) จนกว่าจะกลับเข้าสู่สภาวะปกติหรือเวกอัพ

ถ้าเลือกใช้แหล่งกำเนิดสัญญาณนาฬิกาเป็นประเภทอื่นๆ การแปลงค่าจะถูกยกเลิกทันทีที่คำสั่ง SLEEP ถูกเรียกให้ทำงาน

2.6 ผลกระทบจากการรีเซ็ต

เมื่อเกิดการรีเซ็ตโมดูลเอดีซีจะปิดการทำงานและกระบวนการแปลงค่าจะถูกยกเลิกการทำงาน ขานำเข้าสัญญาณที่ถูกกำหนดให้เป็นขานำเข้าสำหรับโมดูลเอดีซียังคงถูกกำหนดให้นำสัญญาณเข้าแบบแอนะล็อกต่อไปโดยที่ไม่ถูกเปลี่ยนหน้าที่เป็นขานำเข้าสัญญาณแบบดิจิทัล และค่าที่เก็บในเรจิสเตอร์ ADRESH และ ADRESL จะไม่ถูกแก้ไขใดๆ ถ้าเป็นการเร็ตเซ็ตแบบพาวเวอร์ออน

2.7 การอ่านค่าจากโมดูลเอดีซี

ขั้นตอนการแปลงค่าจากแอนะล็อกให้เป็นดิจิทัลมีทั้งสิ้น 6 ขั้นตอน ดังนี้

ขั้นตอนที่ 1 กำหนดค่าให้แก่โมดูลเอดีซีดังนี้

- 1.1 กำหนดขาสำหรับนำสัญญาณเข้าในเรจิสเตอร์ ADCON1
- 1.2 เลือกช่องสัญญาณนำเข้าในเรจิสเตอร์ ADCON0
- 1.3 เลือกสัญญาณนาฬิกาที่ใช้ในการแปลงค่าด้วยการกำหนดค่าในเรจิสเตอร์ ADCON0
- 1.4 เปิดการทำงานของโมดูลเอดีซีด้วยการกำหนดให้บิต ADON ของเรจิสเตอร์ ADCON0 มีค่าเป็น 1

ขั้นตอนที่ 2 รอช่วงเวลาเตรียมการแปลงค่าอย่างน้อย 19.72 ไมโครวินาที

ขั้นตอนที่ 3 เริ่มต้นการแปลงค่าด้วยการกำหนดให้บิต GO ของเรจิสเตอร์ ADCON0 มีค่าเป็น 1

ขั้นตอนที่ 4 รอจนกว่าบิต GO จะเปลี่ยนสถานะจาก 1 เป็น 0

ขั้นตอนที่ 5 อ่านผลลัพธ์จากการแปลงค่าที่เก็บในเรจิสเตอร์ ADRESH และ ADRESL

ขั้นตอนที่ 6 กลับไปขั้นตอนที่ 1 หรือ 2 อีกครั้งเมื่อต้องอ่านค่าครั้งต่อไป

เมื่อนำ 6 ขั้นตอนการอ่านค่าจากโมดูลเอดีซีเขียนเป็นโปรแกรมย่อยสำหรับอ่านค่าสัญญาณแอนะล็อกผ่านช่องสัญญาณ RA0 และบันทึกผลลัพธ์การแปลงค่าในตัวแปร dataADC ได้ดังตัวอย่างที่ 13.1

ตัวอย่างที่ 13.1 โปรแกรมย่อยอ่านผลลัพธ์ทำงานของโมดูลเอดีซีจากขา RA0

```
บรรทัด  โค้ด
1       dim dataADC as word
2       sub procedure ReadRA0()
3         ADCON1 = %10000010
4         ADCON0 = %11000001
5         ADCON0.GO = 1
6         while ADCON0.GO = 1
7         wend
8         dataADC = ADRESH
9         dataADC = (dataADC<<8)+ADRESL
10      end sub
```

โปรแกรมย่อย ReadRA0() ของตัวอย่างที่ 13.1 มีรายละเอียดการทำงานดังนี้

บรรทัดที่ 1 ประกาศตัวแปรชื่อ dataADC ให้เป็นตัวแปรชนิด word เพื่อใช้เก็บผลลัพธ์จากการแปลงค่าแอนะล็อกเป็นตัวเลขดิจิทัล

บรรทัดที่ 3 กำหนดค่าให้เรจิสเตอร์ ADCON1 โดยให้บิตต่างๆ มีความหมายดังนี้

- ADFM มีค่าเป็น 12 เพื่อเลือกรูปแบบการเก็บข้อมูลให้จัดเรียง แบบชิดขวา
- PCFG3:PCFG2:PCFG1:PCFG0 มีค่าเป็น 00102 เพื่อให้ขา RE2 RE1 RE0 เป็นขานำสัญญาณเข้าแบบดิจิทัล โดยขา RA5 RA3 RA2 RA1 และ RA0 เป็นขาสำหรับนำสัญญาณเข้าแบบแอนะล็อก ใช้ V_{REF+} จาก V_{DD} และ V_{REF-} จาก V_{SS}

บรรทัดที่ 4 กำหนดค่าให้แก่เรจิสเตอร์ ADCON0 โดยให้บิตต่างๆ มีความหมายดังนี้

- ADCS1:ADCS0 มีค่าเป็น 11_2 เพื่อเลือกแหล่งสัญญาณนาฬิกาสำหรับการแปลงค่า เป็นสัญญาณนาฬิกาจากวงจรอาร์ซีที่อยู่ภายในโมดูลเอดีซี
- CHS2:CHS1:CHS0 มีค่าเป็น 000_2 เพื่อเลือกช่องสัญญาณที่ 0 เป็นช่องสัญญาณนำเข้าข้อมูลแอนะล็อก

GO/DONE' มีค่าเป็น 0_2

ADON มีค่าเป็น 1_2 เพื่อเปิดให้โมดูลเอดีซีทำงาน

บรรทัดที่ 5 กำหนดให้บิต GO ของเรจิสเตอร์ ADCON0 มีค่าเป็น 1 เพื่อให้โมดูลเอดีซีเริ่มต้นทำงานตามรอบของ T_{AD}

บรรทัดที่ 6-7 ทำซ้ำจนกว่าบิต GO ของเรจิสเตอร์ ADCON0 เปลี่ยนเป็น 0

บรรทัดที่ 8 นำค่าที่เก็บในเรจิสเตอร์ ADRESH ไปเก็บในตัวแปร dataADC

บรรทัดที่ 9 นำค่าที่เก็บในตัวแปร dataADC เลื่อนบิตไปทางซ้าย 8 บิต หลังจากนั้นนำมาบวกกับค่าที่เก็บในเรจิสเตอร์ ADRESL แล้วนำผลลัพธ์จากการบวกไปเก็บในตัวแปร dataADC

ขั้นตอนการแปลงค่าจากแอนะล็อกให้เป็นดิจิทัลโดยใช้การรับสัญญาณขัดจังหวะมีทั้งสิ้น 4 ขั้นตอน ดังนี้

ขั้นตอนที่ 1 กำหนดค่าให้แก่โมดูลเอดีซีดังนี้

1.1 กำหนดขาสำหรับนำสัญญาณเข้าในเรจิสเตอร์ ADCON1

1.2 เลือกช่องสัญญาณนำเข้าในเรจิสเตอร์ ADCON0

1.3 เลือกสัญญาณนาฬิกาที่ใช้ในการแปลงค่าด้วยการกำหนดค่าในเรจิสเตอร์ ADCON0

1.4 เปิดการทำงานของโมดูลเอดีซีด้วยการกำหนดให้บิต ADON ของเรจิสเตอร์ ADCON0 มีค่าเป็น 1

ขั้นตอนที่ 2 เปิดให้รองรับการขัดจังหวะจากแหล่งกำเนิดสัญญาณขัดจังหวะชนิดโมดูลเอดีซีดังนี้

2.1 กำหนดให้บิต ADIF ของเรจิสเตอร์ PIR1 มีค่าเป็น 0

2.2 เปิดให้รองรับแหล่งกำเนิดสัญญาณจากโมดูลเอดีซีด้วยการกำหนดให้บิต ADIE ของเรจิสเตอร์ PIE1 มีค่าเป็น 1

2.3 เปิดให้การขัดจังหวะจากอุปกรณ์รอบข้างทำงาน ด้วยการกำหนดให้บิต PEIE ของเรจิสเตอร์ INTCON มีค่าเป็น 1

2.4 เปิดให้การขัดจังหวะทำงาน ด้วยการกำหนดให้บิต GIE ของเรจิสเตอร์ INTCON มีค่าเป็น 1

ขั้นตอนที่ 3 เริ่มต้นการแปลงค่าด้วยการกำหนดให้บิต GO ของเรจิสเตอร์ ADCON0 มีค่าเป็น 1

ขั้นตอนที่ 4 ตักการสถานะบิต ADIF ของเรจิสเตอร์ PIR1 ในโปรแกรมย่อย interrupt() ซึ่งเมื่อเงื่อนไขเป็นจริงให้ทำขั้นตอนดังต่อไปนี้

4.1 ล้างค่าบิต ADIF ของเรจิสเตอร์ PIR1

4.2 อ่านผลลัพธ์จากการแปลงค่าที่เก็บในเรจิสเตอร์ ADRESH และ ADRESL

4.3 หน่วงเวลา เพื่อรอการเกิดการขัดจังหวะครั้งต่อไป

เมื่อนำ 4 ขั้นตอนการอ่านค่าจากโมดูลเอดีซีเขียนเป็นโปรแกรมย่อยสำหรับอ่านค่าสัญญาณแอนะล็อกผ่านช่องสัญญาณ RA0 และบันทึกผลลัพธ์การแปลงค่าในตัวแปร dataADC ได้ดังตัวอย่างที่ 13.2

ตัวอย่างที่ 13.2 อ่านผลลัพธ์ทำงานของโมดูลเอดีซีจากขา RA0 ด้วยวิธีการขัดจังหวะ

บรรทัด	โค้ด
1	program Sample13_01
2	dim dataADC as word
3	sub procedure interrupt()
4	if (PIR1.ADIF = 1) then
5	PIR1.ADIF = 0
6	dataADC = ADRESH
7	dataADC = (dataADC<<8)+ADRESL
8	Delay_us(100)
9	end if
10	end sub
11	main:
12	TRISA = $FF
13	TRISD = 0
14	PORTD = 0
15	ADCON1 = %10000010
16	ADCON0 = %11000001
17	PIR1.ADIF = 0
18	PIE1.ADIE = 1
19	INTCON.PEIE = 1
20	INTCON.GIE = 1
21	do_again:
22	ADCON0.GO = 1
23	goto do_again
24	end.

จากตัวอย่างที่ 13.2 มีรายละเอียดการทำงานดังนี้

บรรทัดที่ 1 ประกาศตัวแปรชื่อ dataADC ให้เป็นตัวแปรชนิด word เพื่อใช้เก็บผลลัพธ์จากการแปลงค่าแอนะล็อกเป็นตัวเลขดิจิทัล

บรรทัดที่ 3-10 โปรแกรมย่อย interrupt()

บรรทัดที่ 4-9 ถ้าบิต ADIF ของเรจิสเตอร์ PIR1 มีค่าเป็น 1 ซึ่งหมายถึงเกิดการขัดจังหวะจากการทำงานของโมดูลเอดีซี ให้ทำตามถ้อยแถลงบรรทัดที่ 5-8

บรรทัดที่ 5 กำหนดให้บิต ADIF ของเรจิสเตอร์ PIR1 มีค่าเป็น 0

บรรทัดที่ 6 นำค่าที่เก็บในเรจิสเตอร์ ADRESH ไปเก็บในตัวแปร dataADC

บรรทัดที่ 7 นำค่าที่เก็บในตัวแปร dataADC เลื่อนบิตไปทางซ้าย 8 บิต หลังจากนั้นนำมาบวกกับค่าที่เก็บในเรจิสเตอร์ ADRESL แล้วนำผลลัพธ์จากการบวกไปเก็บในตัวแปร dataADC

บรรทัดที่ 8 หน่วงเวลา 100 ไมโครวินาที เพื่อรอการเกิดการขัดจังหวะจากโมดูลเอดีซีครั้งต่อไป

บรรทัดที่ 12 กำหนดให้ทุกขาของพอร์ตเอเป็นขานำสัญญาณเข้า

บรรทัดที่ 13 กำหนดให้ทุกขาของพอร์ตดีเป็นขานำสัญญาณออก

บรรทัดที่ 14 นำค่า 0 ส่งไปที่พอร์ตดี

บรรทัดที่ 15 กำหนดค่าให้เรจิสเตอร์ ADCON1 โดยให้บิตต่างๆ มีความหมายดังนี้

ADFM มีค่าเป็น 12 เพื่อเลือกรูปแบบการเก็บข้อมูลให้จัดเรียง แบบชิดขวา

PCFG3:PCFG2:PCFG1:PCFG0 มีค่าเป็น 00102 เพื่อให้ขา RE2 RE1 RE0 เป็นขานำสัญญาณเข้าแบบดิจิทัล โดยขา RA5 RA3 RA2 RA1 และ RA0 เป็นขาสำหรับนำสัญญาณเข้าแบบแอนะล็อก ใช้ V_{REF+} จาก V_{DD} และ V_{REF-} จาก V_{SS}

บรรทัดที่ 16 กำหนดค่าให้แก่เรจิสเตอร์ ADCON0 โดยให้บิตต่างๆ มีความหมายดังนี้

ADCS1:ADCS0 มีค่าเป็น 11_2 เพื่อเลือกแหล่งสัญญาณนาฬิกาสำหรับการแปลงค่า เป็นสัญญาณนาฬิกาจากวงจรอาร์ซีที่อยู่ภายในโมดูลเอดีซี

CHS2:CHS1:CHS0 มีค่าเป็น 000_2 เพื่อเลือกช่องสัญญาณที่ 0 เป็นช่องสัญญาณนำเข้าข้อมูลแอนะล็อก

GO/DONE' มีค่าเป็น 0_2

ADON มีค่าเป็น 1_2 เพื่อเปิดให้โมดูลเอดีซีทำงาน

บรรทัดที่ 17 กำหนดให้บิต ADIF ของเรจิสเตอร์ PIR1 มีค่าเป็น 0

บรรทัดที่ 18 เปิดให้รองรับแหล่งกำเนิดสัญญาณจากโมดูลเอดีซีด้วยการกำหนดให้บิต ADIE ของเรจิสเตอร์ PIE1 มีค่าเป็น 1

บรรทัดที่ 19 เปิดให้การขัดจังหวะจากอุปกรณ์รอบข้างทำงาน ด้วยการกำหนดให้บิต PEIE ของเรจิสเตอร์ INTCON มีค่าเป็น 1

บรรทัดที่ 20 เปิดให้การขัดจังหวะทำงาน ด้วยการกำหนดให้บิต GIE ของเรจิสเตอร์ INTCON มีค่าเป็น 1

บรรทัดที่ 22 กำหนดให้บิต GO ของเรจิสเตอร์ ADCON0 มีค่าเป็น 1 เพื่อให้โมดูลเอดีซีเริ่มต้นทำงานตามรอบของ T_{AD}

บรรทัดที่ 23 กระโดดไปทำถ้อยแถลงที่อยู่ถัดจากเลเบล do_again

2.8 การแปลงค่าดิจิทัลเป็นระดับของแรงดันเพื่อแสดงผล

เพื่อนำผลลัพธ์จากการแปลงค่าไปแสดงผลกับวงจรแอลอีดี 8 ดวงดังรูปที่ 13.4 จึงต้องมีกระบวนการแปลงผลลัพธ์ที่เป็นค่าตัวเลขเป็นระดับแรงดันก่อนที่จะนำผลลัพธ์ที่เป็นระดับแรงดันส่งไปให้วงจรแอลอีดีแสดงผลดังรูปที่ 13.8

รูปที่ 13.8 ผลลัพธ์การแสดงระดับแรงดันที่อ่านจากโมดูลเอดีซี

ค่า	ผลลัพธ์	หมายเหตุ
น้อยที่สุด		ดับทุกดวง
ปานกลาง		สว่าง 4 ดวงแรก
มากที่สุด		สว่างทุกดวง

จากรูปที่ 13.8 ถ้าค่าจากการแปลงเป็น 0 หลอดแอลอีดีจะดับทุกดวง เมื่อค่าจากการแปลงมีค่ามากขึ้นเป็นลำดับหลอดแอลอีดีแต่ละดวงจะทยอยสว่างไล่กันไปจนเมื่อค่าจากการแปลงเป็น 1023 หลอดแอลอีดีสว่างครบทุกดวง

เนื่องจากช่วงค่าของผลลัพธ์จากการแปลงค่า คือ [0, 1023] ซึ่งมีทั้งหมด 1024 ระดับ แต่หลอดแอลอีดีมีเพียง 8 ดวง จึงได้อัตราส่วนของผลลัพธ์จากโมดูลเอดีซีต่อแอลอีดี 1 ดวง ดังนี้

$$\text{อัตราส่วน} = \frac{1024}{8}$$
$$= 128$$

นั่นหมายความว่า เมื่อมีการเปลี่ยนแปลงค่า 128 ค่า จะทำให้หลอดแอลอีดีสว่างเพิ่มขึ้น 1 ดวง จึงนำมาเขียนเป็นโปรแกรมย่อยได้ดังตัวอย่างที่ 13.3

ตัวอย่างที่ 13.3 โปรแกรมย่อยแสดงระดับแรงดันจากขาที่แอลอีดี 8 ดวง

บรรทัด	โค้ด
1	`sub procedure barLED(dim ByRef led_port as byte, dim value as word)`
2	`  if (value = 0) then`
3	`    led_port = %00000000`
4	`  end if`
5	`  if (value >0) and (value < 128) then`
6	`    led_port = %00000001`
7	`  end if`
8	`  if (value >=128) and (value < 256) then`
9	`    led_port = %00000011`
10	`  end if`
11	`  if (value >=256) and (value < 384) then`
12	`    led_port = %00000111`
13	`  end if`
14	`  if (value >=384) and (value < 512) then`
15	`    led_port = %00001111`
16	`  end if`
17	`  if (value >=512) and (value < 640) then`
18	`    led_port = %00011111`
19	`  end if`
20	`  if (value >=640) and (value < 768) then`
21	`    led_port = %00111111`
22	`  end if`
23	`  if (value >=768) and (value < 896) then`

ตัวอย่างที่ 13.3 โปรแกรมย่อยแสดงระดับแรงดันจากขาที่แอลอีดี 8 ดวง (ต่อ)

บรรทัด	โค้ด
24	`    led_port = %01111111`
25	`  end if`
26	`  if (value >=896) and (value < 1024) then`
27	`    led_port = %11111111`
28	`  end if`
29	`end sub`

โปรแกรมย่อย barLED() ของตัวอย่างที่ 13.3 ต้องการพารามิเตอร์ 2 ตัว คือ led_port และ value โดยที่พารามิเตอร์ led_port เป็นพารามิเตอร์แบบอ้างอิงไปยังข้อมูลแบบไบต์ และพารามิเตอร์ value เป็นพารามเตอร์ชนิด word รายละเอียดการทำงานของโปรแกรมย่อยเป็นดังนี้

บรรทัดที่ 2-4 ถ้าค่าที่เก็บในพารามิเตอร์ value มีค่าเป็น 0 จะทำตามถ้อยแถลงบรรทัดที่ 3

บรรทัดที่ 3 นำค่า 00000000_2 ส่งไปที่พารามิเตอร์ led_port

บรรทัดที่ 5-7 ถ้าค่าที่เก็บในพารามิเตอร์ value มีค่ามากกว่า 0 และน้อยกว่า 128 จะทำตามถ้อยแถลงบรรทัดที่ 6

บรรทัดที่ 6 นำค่า 00000001_2 ส่งไปที่พารามิเตอร์ led_port

บรรทัดที่ 8-10 ถ้าค่าที่เก็บในพารามิเตอร์ value มีค่ามากกว่าหรือเท่ากับ 128 และน้อยกว่า 256 จะทำตามถ้อยแถลงบรรทัดที่ 9

บรรทัดที่ 9 นำค่า 00000011_2 ส่งไปที่พารามิเตอร์ led_port

บรรทัดที่ 11-13 ถ้าค่าที่เก็บในพารามิเตอร์ value มีค่ามากกว่าหรือเท่ากับ 256 และน้อยกว่า 384 จะทำตามถ้อยแถลงบรรทัดที่ 12

บรรทัดที่ 12 นำค่า 00000111_2 ส่งไปที่พารามิเตอร์ led_port

บรรทัดที่ 14-16 ถ้าค่าที่เก็บในพารามิเตอร์ value มีค่ามากกว่าหรือเท่ากับ 384 และน้อยกว่า 512 จะทำตามถ้อยแถลงบรรทัดที่ 15

บรรทัดที่ 15 นำค่า 00001111_2 ส่งไปที่พารามิเตอร์ led_port

บรรทัดที่ 17-19 ถ้าค่าที่เก็บในพารามิเตอร์ value มีค่ามากกว่าหรือเท่ากับ 512 และน้อยกว่า 640 จะทำตามถ้อยแถลงบรรทัดที่ 18

บรรทัดที่ 18 นำค่า 00011111_2 ส่งไปที่พารามิเตอร์ led_port

บรรทัดที่ 20-22 ถ้าค่าที่เก็บในพารามิเตอร์ value มีค่ามากกว่าหรือเท่ากับ 640 และน้อยกว่า 768 จะทำตามถ้อยแถลงบรรทัดที่ 21

บรรทัดที่ 21 นำค่า 00111111_2 ส่งไปที่พารามิเตอร์ led_port

บรรทัดที่ 23-25 ถ้าค่าที่เก็บในพารามิเตอร์ value มีค่ามากกว่าหรือเท่ากับ 768 และน้อยกว่า 896 จะทำตามถ้อยแถลงบรรทัดที่ 24

บรรทัดที่ 24 นำค่า 01111111_2 ส่งไปที่พารามิเตอร์ led_port

บรรทัดที่ 26-28 ถ้าค่าที่เก็บในพารามิเตอร์ value มีค่ามากกว่าหรือเท่ากับ 896 และน้อยกว่า 1024 จะทำตามถ้อยแถลงบรรทัดที่ 27

บรรทัดที่ 27 นำค่า 11111111_2 ส่งไปที่พารามิเตอร์ led_port

เมื่อนำโปรแกรมย่อย ReadRA0() ที่ทำหน้าที่อ่านค่าผลลัพธ์จากการแปลงสัญญาณแอนะล็อกเป็นค่าตัวเลขดิจิทัล และส่งผลลัพธ์ไปให้โปรแกรมย่อย barLED() เพื่อแสดงผลลัพธ์ในรูปแบบระดับแรงดัน จะได้โปรแกรมดังตัวอย่างที่ 13.4

ตัวอย่างที่ 13.4 แสดงระดับแรงดันจากขา RA0 ที่แอลอีดี 8 ดวง

บรรทัด	โค้ด
1	program Sample13_02
2	dim dataADC as word
3	sub procedure ReadRA0()
4	ADCON1 = %10000010
5	ADCON0 = %11000001
6	ADCON0.GO = 1
7	while ADCON0.GO = 1
8	wend
9	dataADC = ADRESH
10	dataADC = (dataADC<<8)+ADRESL
11	end sub
12	sub procedure barLED(dim ByRef led_port as byte, dim value as word)
13	if (value = 0) then
14	led_port = %00000000
15	end if
16	if (value >0) and (value < 128) then
17	led_port = %00000001
18	end if
19	if (value >=128) and (value < 256) then
20	led_port = %00000011
21	end if
22	if (value >=256) and (value < 384) then
23	led_port = %00000111
24	end if
25	if (value >=384) and (value < 512) then
26	led_port = %00001111
27	end if

ตัวอย่างที่ 13.4 แสดงระดับแรงดันจากขา RA0 ที่แอลอีดี 8 ดวง (ต่อ)

บรรทัด	โค้ด
28	if (value >=512) and (value < 640) then
29	led_port = %00011111
30	end if
31	if (value >=640) and (value < 768) then
32	led_port = %00111111
33	end if
34	if (value >=768) and (value < 896) then
35	led_port = %01111111
36	end if
37	if (value >=896) and (value < 1024) then
38	led_port = %11111111
39	end if
40	end sub
41	main:
42	TRISA = $FF
43	TRISD = 0
44	PORTD = 0
45	do_again:
46	ReadRA0()
47	barLED(PORTD, dataADC)
48	goto do_again
49	end.

จากตัวอย่างที่ 13.4 มีรายละเอียดการทำงานดังนี้

บรรทัดที่ 2 ประกาศตัวแปรชื่อ dataADC ให้เป็นตัวแปรชนิด word เพื่อใช้เก็บผลลัพธ์จากการทำงานของโมดูลเอดีซี

บรรทัดที่ 3-11 โปรแกรมย่อย ReadRA0() มีหลักการทำงานดังที่อธิบายในตัวอย่างที่ 13.1

บรรทัดที่ 12-40 โปรแกรมย่อย barLED() มีหลักการทำงานดังที่อธิบายในตัวอย่างที่ 13.2

บรรทัดที่ 42 กำหนดให้ทุกขาของพอร์ตเอเป็นขานำสัญญาณเข้า

บรรทัดที่ 43 กำหนดให้ทุกขาของพอร์ตดีเป็นขานำสัญญาณออก

บรรทัดที่ 44 นำค่า 0 ส่งไปที่พอร์ตดี

บรรทัดที่ 46 เรียกใช้โปรแกรมย่อย ReadRAD0() เพื่ออ่านผลลัพธ์จากโมดูลเอดีซีมาเก็บในตัวแปร dataADC

บรรทัดที่ 47 เรียกใช้โปรแกรมย่อย barLED() โดยส่งพารามิเตอร์เป็น PORTD และตัวแปร dataADC

บรรทัดที่ 48 กระโดดไปทำถ้อยแถลงที่อยู่ถัดจากเลเบล do_again

ข้อจำกัดของโปรแกรมย่อย ReadRA0() คืออ่านค่าจากขา RA0 และนำผลลัพธ์ไปเก็บในตัวแปร dataADC เท่านั้น จึงไม่สะดวกต่อการนำกลับมาใช้ ดังนั้น เพื่อให้โปรแกรมย่อยมีความยืดหยุ่นที่สูงขึ้น คือ รองรับการเลือกช่องสัญญาณนำเข้า และคืนค่าจากการแปลงค่ากลับมาได้ จึงแปลงโปรแกรมย่อย ReadRA0() ให้เป็นฟังก์ชัน และต้องการพารามิเตอร์เป็นค่าช่องสัญญาณนำเข้าดังตัวอย่างที่ 13.4

ตัวอย่างที่ 13.5 โปรแกรมย่อยอ่านค่าจากโมดูลเอดีซี

บรรทัด	โค้ด
1	sub function ReadADC(dim adc_port as byte) as word
2	dim temp as word
3	dim oldADCON0 as byte
4	oldADCON0 = ADCON0
5	select case adc_port
6	case 0
7	ADCON0 = %11000001
8	case 1
9	ADCON0 = %11001001
10	case 2
11	ADCON0 = %11010001
12	case 3
13	ADCON0 = %11011001
14	case 4
15	ADCON0 = %11100001
16	case 5
17	ADCON0 = %11101001
18	case 6
19	ADCON0 = %11110001
20	case 7
21	ADCON0 = %11111001
22	end select
23	ADCON0.GO = 1
24	while ADCON0.GO = 1
25	Wend
26	temp = ADRESH
27	ADCON0 = oldADCON0
28	Result = (temp<<8)+ADRESL
29	end sub

ฟังก์ชัน ReadADC() ต้องการพารามิเตอร์ชื่อ adc_port เป็นข้อมูลแบบไบต์ เพื่อใช้สำหรับระบุช่องสัญญาณนำเข้า และคืนค่ากลับเป็นข้อมูลชนิด word ซึ่งมีรายละเอียดของการทำงานดังนี้

บรรทัดที่ 2 ประกาศตัวแปรชื่อ temp ให้เป็นตัวแปรชนิด word เพื่อใช้เก็บค่าของเรจิสเตอร์ADRESH

บรรทัดที่ 3 ประกาศตัวแปรชื่อ oldADCON0 ให้เป็นตัวแปรชนิด byte เพื่อเก็บค่าของเรจิสเตอร์ ADCON0 ก่อนที่จะเปลี่ยนค่า

บรรทัดที่ 4 นำค่าที่เก็บในเรจิสเตอร์ ADCON0 ไปเก็บในตัวแปร oldADCON0

บรรทัดที่ 5-22 ตรวจสอบค่าของพารามิเตอร์ adc_port

บรรทัดที่ 6-7 ถ้ามีค่าเป็น 0 ให้ทำตามถ้อยแถลงบรรทัดที่ 7

บรรทัดที่ 7 กำหนดให้เรจิสเตอร์ ADCON0 มีค่าเป็น 11000001_2 ซึ่งเป็นการเลือกช่องสัญญาณ 0 หรืออ่านสัญญาณแบบแอนะล็อกผ่านทางขา RA0

บรรทัดที่ 8-9 ถ้ามีค่าเป็น 1 ให้ทำตามถ้อยแถลงบรรทัดที่ 8

บรรทัดที่ 9 กำหนดให้เรจิสเตอร์ ADCON0 มีค่าเป็น 11001001_2 ซึ่งเป็นการเลือกช่องสัญญาณ 1 หรืออ่านสัญญาณแบบแอนะล็อกผ่านทางขา RA1

บรรทัดที่ 10-11 ถ้ามีค่าเป็น 2 ให้ทำตามถ้อยแถลงบรรทัดที่ 11

บรรทัดที่ 11 กำหนดให้เรจิสเตอร์ ADCON0 มีค่าเป็น 11010001_2 ซึ่งเป็นการเลือกช่องสัญญาณ 2 หรืออ่านสัญญาณแบบแอนะล็อกผ่านทางขา RA2

บรรทัดที่ 12-13 ถ้ามีค่าเป็น 3 ให้ทำตามถ้อยแถลงบรรทัดที่ 13

บรรทัดที่ 13 กำหนดให้เรจิสเตอร์ ADCON0 มีค่าเป็น 11011001_2 ซึ่งเป็นการเลือกช่องสัญญาณ 3 หรืออ่านสัญญาณแบบแอนะล็อกผ่านทางขา RA3

บรรทัดที่ 14-15 ถ้ามีค่าเป็น 4 ให้ทำตามถ้อยแถลงบรรทัดที่ 15

บรรทัดที่ 15 กำหนดให้เรจิสเตอร์ ADCON0 มีค่าเป็น 11100001_2 ซึ่งเป็นการเลือกช่องสัญญาณ 4 หรืออ่านสัญญาณแบบแอนะล็อกผ่านทางขา RA5

บรรทัดที่ 16-17 ถ้ามีค่าเป็น 5 ให้ทำตามถ้อยแถลงบรรทัดที่ 17

บรรทัดที่ 17 กำหนดให้เรจิสเตอร์ ADCON0 มีค่าเป็น 11101001_2 ซึ่งเป็นการเลือกช่องสัญญาณ 5 หรืออ่านสัญญาณแบบแอนะล็อกผ่านทางขา RE0

บรรทัดที่ 18-19 ถ้ามีค่าเป็น 6 ให้ทำตามถ้อยแถลงบรรทัดที่ 19

บรรทัดที่ 19 กำหนดให้เรจิสเตอร์ ADCON0 มีค่าเป็น 11110001_2 ซึ่งเป็นการเลือกช่องสัญญาณ 6 หรืออ่านสัญญาณแบบแอนะล็อกผ่านทางขา RE1

บรรทัดที่ 20-21 ถ้ามีค่าเป็น 6 ให้ทำตามถ้อยแถลงบรรทัดที่ 21

บรรทัดที่ 21 กำหนดให้เรจิสเตอร์ ADCON0 มีค่าเป็น 11111001_2 ซึ่งเป็นการเลือกช่องสัญญาณ 7 หรืออ่านสัญญาณแบบแอนะล็อกผ่านทางขา RE2

บรรทัดที่ 23 กำหนดให้บิต GO ของเรจิสเตอร์ ADCON0 มีค่าเป็น 1

บรรทัดที่ 24-25 ทำซ้ำถ้าบิต GO ของเรจิสเตอร์ ADCON0 ยังคงมีค่าเป็น 1

บรรทัดที่ 26 นำค่าที่เก็บในเรจิสเตอร์ ADRESH ไปเก็บในตัวแปร temp

บรรทัดที่ 27 นำค่าที่เก็บในตัวแปร oldADCON0 ไปเก็บในเรจิสเตอร์ ADCON0

บรรทัดที่ 28 นำค่าที่เก็บในตัวแปร temp เลื่อนไปทางซ้าย 8 บิต แล้วไปบวกกับค่าที่เก็บในเรจิสเตอร์ ADRESL หลังจากนั้นนำผลลัพธ์จากการบวกไปเก็บตัวแปร Result เพื่อส่งค่ากลับ

เมื่อนำใช้ฟังก์ชัน ReadADC() มาแทนโปรแกรมย่อย ReadRA0() จะได้โค้ดดังตัวอย่างที่ 13.6

ตัวอย่างที่ 13.6 โปรแกรมค่าจากโมดูลเอดีซีและแสดงผลที่แอลอีดี 8 ดวง

บรรทัด	โค้ด
1	program Sample13_03
2	dim dataADC as word
3	sub function ReadADC(dim adc_port as byte) as word
4	dim temp as word
5	dim oldADCON0 as byte
6	oldADCON0 = ADCON0
7	select case adc_port
8	case 0
9	ADCON0 = %11000001
10	case 1
11	ADCON0 = %11001001
12	case 2
13	ADCON0 = %11010001
14	case 3
15	ADCON0 = %11011001
16	case 4
17	ADCON0 = %11100001
18	case 5
19	ADCON0 = %11101001
20	case 6
21	ADCON0 = %11110001
22	case 7
23	ADCON0 = %11111001
24	end select
25	ADCON0.GO = 1
26	while ADCON0.GO = 1
27	Wend
28	temp = ADRESH
29	ADCON0 = oldADCON0
30	Result = (temp<<8)+ADRESL
31	end sub

ตัวอย่างที่ 13.6 โปรแกรมค่าจากโมดูลเอดีซีและแสดงผลที่แอลอีดี 8 ดวง (ต่อ)

บรรทัด	โค้ด
32	sub procedure barLED(dim ByRef led_port as byte, dim value as word)
33	if (value = 0) then
34	led_port = %00000000
35	end if
36	if (value >0) and (value < 128) then
37	led_port = %00000001
38	end if
39	if (value >=128) and (value < 256) then
40	led_port = %00000011
41	end if
42	if (value >=256) and (value < 384) then
43	led_port = %00000111
44	end if
45	if (value >=384) and (value < 512) then
46	led_port = %00001111
47	end if
48	if (value >=512) and (value < 640) then
49	led_port = %00011111
50	end if
51	if (value >=640) and (value < 768) then
52	led_port = %00111111
53	end if
54	if (value >=768) and (value < 896) then
55	led_port = %01111111
56	end if
57	if (value >=896) and (value < 1024) then
58	led_port = %11111111
59	end if
60	end sub
61	main:
62	TRISA = $FF
63	TRISD = 0
64	PORTD = 0
65	ADCON1 = %10000010
66	do_again:
67	dataADC = ReadADC(0)
68	barLED(PORTD, dataADC)
69	goto do_again
70	end.

จากตัวอย่างที่ 13.6 มีรายละเอียดการทำงานดังนี้

บรรทัดที่ 2 ประกาศตัวแปรชื่อ dataADC ให้เป็นตัวแปรชนิด word เพื่อใช้เก็บผลลัพธ์จากการทำงานของโมดูลเอดีซี

บรรทัดที่ 3-31 โปรแกรมย่อย ReadADC() มีหลักการทำงานดังที่อธิบายในตัวอย่างที่ 13.4

บรรทัดที่ 32-60 โปรแกรมย่อย barLED() มีหลักการทำงานดังที่อธิบายในตัวอย่างที่ 13.2

บรรทัดที่ 62 กำหนดให้ทุกขาของพอร์ตเอเป็นขานำสัญญาณเข้า

บรรทัดที่ 63 กำหนดให้ทุกขาของพอร์ตดีเป็นขานำสัญญาณออก

บรรทัดที่ 64 นำค่า 0 ส่งไปที่พอร์ตดี

บรรทัดที่ 65 กำหนดค่าให้เรจิสเตอร์ ADCON1 โดยให้บิตต่างๆ มีความหมายดังนี้

ADFM มีค่าเป็น 12 เพื่อเลือกรูปแบบการเก็บข้อมูลให้จัดเรียง แบบชิดขวา

PCFG3:PCFG2:PCFG1:PCFG0 มีค่าเป็น 00102 เพื่อให้ขา RE2 RE1 RE0 เป็นขานำสัญญาณเข้าแบบดิจิทัล โดยขา RA5 RA3 A2 RA1 และ RA0 เป็นขานำสัญญาณเข้าแบบแอนะล็อก ใช้ V_{REF+} จาก V_{DD} และ V_{REF-} จาก V_{SS}

บรรทัดที่ 67 เรียกใช้โปรแกรมย่อย ReadADC() เพื่ออ่านสัญญาณช่อง 0 และนำผลลัพธ์จากการทำงานของโมดูลเอดีซีไปเก็บในตัวแปร dataADC

บรรทัดที่ 68 เรียกใช้โปรแกรมย่อย barLED() โดยส่งพารามิเตอร์เป็น PORTD และตัวแปร dataADC

บรรทัดที่ 69 กระโดดไปทำถ้อยแถลงที่อยู่ถัดจากเลเบล do_again

3. คำสั่ง ADC_Read

คำสั่ง ADC_Read เป็นคำสั่งสำหรับอ่านค่าผลลัพธ์จากการทำงานของโมดูลเอดีซี โดยรูปแบบของคำสั่งเป็นดังนี้

```
sub function ADC_Read(dim channel as byte) as word
```

จากรูปแบบของคำสั่งจะพบว่า ADC_Read ต้องการพารามิเตอร์ 1 ตัว คือ channel ที่เป็นพารามิเตอร์ชนิด byte ใช้สำหรับระบุหมายเลขช่องสัญญาณที่ต้องการอ่านสัญญาณแอนะล็อก โดย

ค่าของพารามิเตอร์นี้จะต้องไม่เกินจำนวนช่องสัญญาณที่รุ่นของไมโครคอนโทรลเลอร์สนับสนุนดังตาราง 13.1 ซึ่งเริ่มนับช่องสัญญาณแรกเป็นช่องสัญญาณที่ 0 ด้วยเหตุนี้ ถ้าสนับสนุน 8 ช่องสัญญาณ ค่าของพารามิเตอร์ channel จะมีค่าตั้งแต่ 0 ถึง 7

ตัวอย่างที่ 13.7 โปรแกรมย่อยอ่านค่าจากโมดูลเอดีซีโดยใช้คำสั่ง ADC_Read

บรรทัด	โค้ด
1	program Sample13_04
2	dim dataADC as word
3	sub procedure barLED(dim ByRef led_port as byte, dim value as word)
4	if (value = 0) then
5	led_port = %00000000
6	end if
7	if (value >0) and (value < 128) then
8	led_port = %00000001
9	end if
10	if (value >=128) and (value < 256) then
11	led_port = %00000011
12	end if
13	if (value >=256) and (value < 384) then
14	led_port = %00000111
15	end if
16	if (value >=384) and (value < 512) then
17	led_port = %00001111
18	end if
19	if (value >=512) and (value < 640) then
20	led_port = %00011111
21	end if
22	if (value >=640) and (value < 768) then
23	led_port = %00111111
24	end if
25	if (value >=768) and (value < 896) then
26	led_port = %01111111
27	end if
28	if (value >=896) and (value < 1024) then
29	led_port = %11111111
30	end if
31	end sub
32	main:
33	TRISA = $FF
34	TRISD = 0
35	PORTD = 0
36	ADCON1 = %10000010
37	do_again:
38	dataADC = ADC_Read(0)
39	barLED(PORTD, dataADC)
40	goto do_again
41	end.

จากตัวอย่างที่ 13.7 มีการทำงานเหมือนกับตัวอย่างที่ 13.6 แต่เรียกใช้คำสั่ง ADC_Read() แทนเรียกใช้ฟังก์ชันที่เขียนขึ้นเอง ซึ่งตัวอย่างมีรายละเอียดการทำงานดังนี้

บรรทัดที่ 2 ประกาศตัวแปรชื่อ dataADC ให้เป็นตัวแปรชนิด word เพื่อใช้เก็บผลลัพธ์จากการทำงานของโมดูลเอดีซี

บรรทัดที่ 3-31 โปรแกรมย่อย barLED() มีหลักการทำงานดังที่อธิบายในตัวอย่างที่ 13.2

บรรทัดที่ 33 กำหนดให้ทุกขาของพอร์ตเอเป็นขานำสัญญาณเข้า

บรรทัดที่ 34 กำหนดให้ทุกขาของพอร์ตดีเป็นขานำสัญญาณออก

บรรทัดที่ 35 นำค่า 0 ส่งไปที่พอร์ตดี

บรรทัดที่ 36 กำหนดค่าให้เรจิสเตอร์ ADCON1 โดยให้บิตต่างๆ มีความหมายดังนี้

ADFM มีค่าเป็น 12 เพื่อเลือกรูปแบบการเก็บข้อมูลให้จัดเรียงแบบชิดขวา

PCFG3:PCFG2:PCFG1:PCFG0 มีค่าเป็น 00102 เพื่อให้ขา RE2 RE1 RE0 เป็นขานำสัญญาณเข้าแบบดิจิทัล โดยขา RA5 RA3 A2 RA1 และ RA0 เป็นขานำสัญญาณเข้าแบบแอนะล็อก ใช้ V_{REF+} จาก V_{DD} และ V_{REF-} จาก V_{SS}

บรรทัดที่ 38 เรียกใช้คำสั่ง ADC_Read() เพื่ออ่านสัญญาณช่อง 0 และนำผลลัพธ์จากการทำงานของโมดูลเอดีซีไปเก็บในตัวแปร dataADC

บรรทัดที่ 39 เรียกใช้โปรแกรมย่อย barLED() โดยส่งพารามิเตอร์เป็น PORTD และตัวแปร dataADC

บรรทัดที่ 40 กระโดดไปทำถ้อยแถลงที่อยู่ถัดจากเลเบล do_again

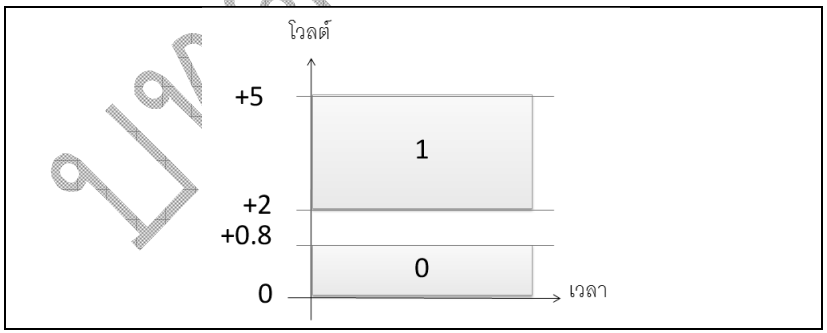
4. ลอจิกโพรบ

ลอจิกโพรบ (Logic Probe) เป็นอุปกรณ์ที่คล้ายปากกา (รูปที่ 13.9) ใช้สำหรับวิเคราะห์หรือแก้ไขปัญหาที่เกี่ยวข้องกับสถานะลอจิกของวงจรดิจิทัล ซึ่งใช้ได้กับอุปกรณ์ที่เป็นแบบทีทีแอล (TTL: Transistor-transistor logic) หรือซีมอส (CMOS: Complementary Metallic Oxide Seminconductor)

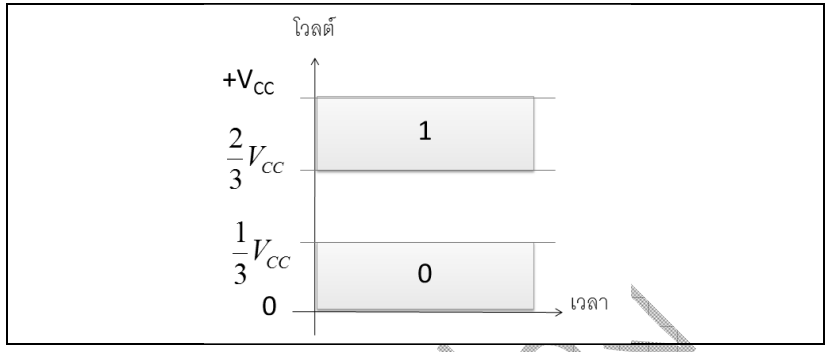
รูปที่ 13.9 ลอจิกโพรบ

ความแตกต่างของอุปกรณ์ที่เป็นทีทีแอลกับซีมอสคือ ระดับแรงดันของสถานะที่เป็น 1 และ 0 มีความแตกต่างกันดังรูปที่ 13.10 และ 13.11

รูปที่ 13.10 ระดับแรงดันของไอซีดิจิทัลแบบทีทีแอล

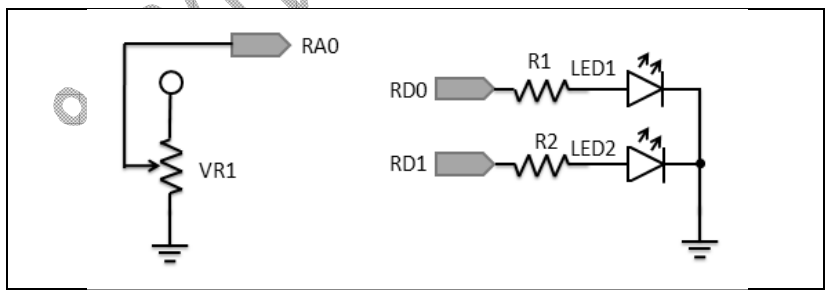


รูปที่ 13.11 ระดับแรงดันของไอซีดิจิทัลแบบซีมอส



เมื่อนำหลักการทำงานของโมดูลเอดีซีมาประยุกต์เพื่อทดลองสร้างการทำงานที่เหมือนกับลอจิกโพรบที่ใช้กับอุปกรณ์ประเภททีทีแอล จึงได้วงจรทดลองดังรูปที่ 13.12 โดยอ่านค่าจากช่องสัญญาณที่ 0 หรือผ่านทางขา RA0 แล้วตรวจสอบค่าของแรงดันที่อ่านว่าอยู่ในช่วงของสถานะใดระหว่าง 0 กับ 1 แล้วแสดงผลสถานะของระดับแรงดันที่ LED1 แต่ถ้าแรงดันที่อ่านได้อยู่ในช่วงที่เป็น 0 หรือ 1 จะให้ LED2 สว่าง เพื่อบ่งบอกให้ทราบว่าระดับแรงดันอยู่ในช่วงที่ไม่สามารถระบุระดับของแรงดันได้

รูปที่ 13.12 วงจรทดลองการทำงานของลอจิกโพรบ



จากวงจรรูปที่ 13.12 มีรายการอุปกรณ์ที่ต้องใช้ดังนี้

1. ตัวต้านทานปรับค่าได้ VR1 ความต้านทาน 10 กิโลโอมห์
2. ตัวต้านทาน R1 และ R2 ความต้านทาน 270 โอมห์
3. แอลอีดี LED1 และ LED8 สีแดง

เนื่องจากค่าที่อ่านจากโมดูลเอดีซีมีทั้งสิ้น 1024 ค่า จึงต้องคำนวณเพื่อหาค่าที่เป็นแรงดัน 0.8 โวลต์ และ 2.0 โวลต์ ซึ่งค่าสูงสุดของแรงดันที่อ่านเข้ามานั้นเป็น 5 โวลต์ ทำให้ได้อัตราส่วนของค่าจากการแปลงต่อแรงดันดังสมการต่อไปนี้

$$\text{อัตราส่วนต่อ 1 โวลต์} = \frac{1024}{5}$$

$$\text{แรงดัน 0.8 โวลต์} = 0.8 \times \frac{1024}{5}$$

$$= 163.84$$

$$\text{หรือประมาณ} \quad 164$$

$$\text{แรงดัน 2 โวลต์} = 2 \times \frac{1024}{5}$$

$$= 409.6$$

$$\text{หรือประมาณ} \quad 410$$

ตัวอย่างที่ 13.8 โปรแกรมวัดระดับแรงดันสัญญาณดิจิทัลหรือลอจิกโพรบ

บรรทัด	โค้ด
1	program Sample13_05
2	dim dataADC as word
3	dim hi_cutoff as word
4	dim lo_cutoff as word
5	main:
6	TRISA = $FF
7	TRISD = 0
8	PORTD = 0
9	ADCON1 = %10000010
10	hi_cutoff = 164
11	lo_cutoff = 410
12	do_again:
13	dataADC = ADC_Read(0)
14	if (dataADC >= hi_cutoff) then
15	PORTD = %00000001
16	Else
17	if (dataADC <= lo_cutoff) then
18	PORTD = %00000000
19	Else
20	PORTD = %00000010
21	end if
22	end if
23	goto do_again
24	end.

จากตัวอย่างที่ 13.8 มีรายละเอียดการทำงานดังนี้

บรรทัดที่ 2 ประกาศตัวแปรชื่อ dataADC ให้เป็นตัวแปรชนิด word เพื่อใช้เก็บผลลัพธ์จากการทำงานของโมดูลเอดีซี

บรรทัดที่ 3 ประกาศตัวแปรชื่อ hi_curoff ให้เป็นตัวแปรชนิด word เพื่อใช้เก็บค่าของตัวเลขของแรงดัน 2 โวลต์

บรรทัดที่ 4 ประกาศตัวแปรชื่อ low_curoff ให้เป็นตัวแปรชนิด word เพื่อใช้เก็บค่าของตัวเลขของแรงดัน 0.8 โวลต์

บรรทัดที่ 6 กำหนดให้ทุกขาของพอร์ตเอเป็นขานำสัญญาณเข้า

บรรทัดที่ 7 กำหนดให้ทุกขาของพอร์ตดีเป็นขานำสัญญาณออก

บรรทัดที่ 8 นำค่า 0 ส่งไปที่พอร์ตดี

บรรทัดที่ 9 กำหนดค่าให้เรจิสเตอร์ ADCON1 โดยให้บิตต่างๆ มีความหมายดังนี้

ADFM มีค่าเป็น 12 เพื่อเลือกรูปแบบการเก็บข้อมูลให้จัดเรียงแบบชิดขวา

PCFG3:PCFG2:PCFG1:PCFG0 มีค่าเป็น 00102 เพื่อให้ขา RE2 RE1 RE0 เป็นขานำสัญญาณเข้าแบบดิจิทัล โดยขา RA5 RA3 A2 RA1 และ RA0 เป็นขานำสัญญาณเข้าแบบแอนะล็อก ใช้ V_{REF+} จาก V_{DD} และ V_{REF-} จาก V_{SS}

บรรทัดที่ 10 กำหนดให้ตัวแปร hi_cutoff มีค่าเป็น 164

บรรทัดที่ 11 กำหนดให้ตัวแปร hi_cutoff มีค่าเป็น 410

บรรทัดที่ 13 เรียกใช้คำสั่ง ADC_Read() เพื่ออ่านสัญญาณช่อง 0 และนำผลลัพธ์จากการทำงานของโมดูลเอดีซีไปเก็บในตัวแปร dataADC

บรรทัดที่ 14-22 ถ้าค่าที่เก็บในตัวแปร dataADC มากกว่าหรือเท่ากับ hi_cutoff เป็นจริงให้ทำตามถ้อยแถลงบรรทัดที่ 15

บรรทัดที่ 15 กำหนดให้พอร์ตมีค่าเป็น 00000001_2 ซึ่งเป็นการทำให้หลอด LED1 สว่าง

บรรทัดที่ 16-22 ถ้าเงื่อนไขจากบรรทัดที่ 14 ไม่เป็นจริงให้ทำตามถ้อยแถลงบรรทัดที่ 17 ถึง 21

บรรทัดที่ 17-21 ถ้าค่าที่เก็บในตัวแปร dataADC น้อยกว่าหรือเท่ากับ low_cutoff เป็นจริงให้ทำตามถ้อยแถลงบรรทัดที่ 18

บรรทัดที่ 18 กำหนดให้พอร์ตมีค่าเป็น 00000000_2 ซึ่งเป็นการทำให้หลอด LED1 ดับ

บรรทัดที่ 19-21 ถ้าเงื่อนไขบรรทัดที่ 17 ไม่เป็นจริงให้ทำตามถ้อยแถลงบรรทัดที่ 20

บรรทัดที่ 20 กำหนดให้พอร์ตมีค่าเป็น 00000010_2 ซึ่งเป็นการทำให้หลอด LED2 สว่าง เนื่องจากค่าแรงดันที่อ่านเข้ามานั้นอยู่ในช่วงไม่สามารถบอกสถานะของระดับแรงได้

บรรทัดที่ 23 กระโดดไปทำถ้อยแถลงที่อยู่ถัดจากเลเบล do_again

5. สรุป

ในบทนี้ได้เรียนรู้ถึงหลักของการแปลงสัญญาณแบบแอนะล็อกซึ่งเป็นสัญญาณแบบต่อเนื่องให้เป็นสัญญาณแบบดิจิทัลที่ไม่ต่อเนื่อง และได้เข้าใจถึงความสัมพันธ์ระหว่างค่าความละเอียดของการแปลงค่าว่า ถ้าไมโครคอนโทรลเลอร์มีความสามารถในการแปลงที่สูงจะทำให้ได้ข้อมูลที่มีความละเอียดและใกล้เคียงกับสัญญาณที่ต่อเนื่องมากขึ้นเท่านั้น

นอกจากนี้ได้เรียนรู้วิธีการควบคุมการทำงานของโมดูลเอดีซีของไมโครคอนโทรลเลอร์ PIC เพื่อนำเข้าสัญญาณแบบแอนะล็อก โดยเขียนโปรแกรมขึ้นมาเอง และใช้ชุดคำสั่งที่ mikroBasic PIC เตรียมไว้ให้ พร้อมทั้งมีตัวอย่างที่ประยุกต์การใช้งานโมดูลเอดีซีมาเป็นลอจิกโพรบ ซึ่งนำไปพัฒนาให้ระบบมีความสมบูรณ์และรองรับแรงดันระดับซีมอสได้ต่อไป