ผศ.ดร. เดชฤทธิ์ มณีธรรม

คัมภีร์การใช้งาน ไมโครคอนโทรลเลอร์ MICROCONTROLLER MCS-51

ซีเอ็ด

คัมภีร์การใช้งาน ไมโครคอนโทรลเลอร์ MCS-51

โดย ผศ.ดร เดชฤทธิ์ มณีธรรม

สงวนลิขสิทธิ์ในประเทศไทยตาม พ.ร.บ. ลิขสิทธิ์ © พ.ศ. 2559 โดย ผศ.ดร. เดชฤทธิ์ มณีธรรม
ห้ามคัดลอก ลอกเลียน ดัดแปลง ทำซ้ำ จัดพิมพ์ หรือกระทำอื่นใด โดยวิธีการใดๆ ในรูปแบบใดๆ
ไม่ว่าส่วนหนึ่งส่วนใดของหนังสือเล่มนี้ เพื่อเผยแพร่ในสื่อทุกประเภท หรือเพื่อวัตถุประสงค์ใดๆ
นอกจากจะได้รับอนุญาต

ข้อมูลทางบรรณานุกรมของหอสมุดแห่งชาติ
เดชฤทธิ์ มณีธรรม, ผศ.ดร.
คัมภีร์การใช้งาน ไมโครคอนโทรลเลอร์ MCS-51. —กรุงเทพฯ : ซีเอ็ดยูเคชั่น, 2559.
1. ไมโครคอนโทรลเลอร์.
I. ชื่อเรื่อง.
004.64

ISBN (e-book) : 978-616-08-2717-6

ผลิตและจัดจำหน่ายโดย

บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน)
SE-EDUCATION PUBLIC COMPANY LIMITED

1858/87-90 ถนนบางนา-ตราด แขวงบางนา
เขตบางนา กรุงเทพฯ 10260 โทรศัพท์ 0-2739-8000

[หากมีคำแนะนำหรือติชม สามารถติดต่อได้ที่ comment@se-ed.com]

คำนำ

ปัจจุบันไมโครคอนโทรลเลอร์ (Microcontroller) ได้ถูกนำมาใช้อย่างกว้างขวาง ในงานอุตสาหกรรม เพื่อควบคุมเครื่องจักรกลและระบบการผลิตต่างๆ เช่น ควบคุมระบบในรถยนต์ (ECU) ควบคุมหุ่นยนต์อุตสาหกรรม หรือควบคุมระบบขนถ่ายวัสดุ เป็นต้น โดยทั่วไปวัตถุประสงค์ของการใช้ไมโครคอนโทรลเลอร์ในการควบคุม ก็เพื่อความรวดเร็ว ถูกต้อง และแม่นยำ ของระบบการผลิต ตลอดจนทำให้เครื่องจักรทำงานได้ประสิทธิภาพสูง และสามารถลดจำนวนคนงานได้อีกมากทีเดียว

หนังสือ “**คัมภีร์การใช้งาน ไมโครคอนโทรลเลอร์** MCS-51 (Microcontroller MCS-51)” เล่มนี้ มีจุดมุ่งหมายที่จะพัฒนาระบบควบคุมอัตโนมัติ (Automation Control System) ตลอดจนหุ่นยนต์อุตสาหกรรม (Industrial Robot) ให้มีการแข่งขันและพัฒนาให้ทันกับโลกปัจจุบัน ที่มีการแข่งขันทางด้านการตลาด และระบบเศรษฐกิจที่สูง โดยจะเริ่มต้นอธิบายเกี่ยวกับไมโครคอนโทรลเลอร์ (MCS-51) ภาษาซี (C Language) การออกแบบควบคุมการทำงานของ Input / Output Port การควบคุม Stepping Motor การควบคุม DC Motor การควบคุม DC Servo Motor และจบด้วยการควบคุมหุ่นยนต์อุตสาหกรรม โดยหนังสือเล่มนี้เหมาะสำหรับนักศึกษาได้ใช้เรียน และค้นคว้าเพิ่มเติม รวมทั้งช่างไฟฟ้า ช่างอิเล็กทรอนิกส์ วิศวกรโรงงาน ตลอดจนผู้สนใจในงานควบคุมด้วยไมโครคอนโทรลเลอร์

ผู้เขียนมีความปรารถนาที่จะขอขอบพระคุณ คุณพรชัย โอภาสยล ที่ได้ให้ความช่วยเหลือด้านข้อมูลเอกสาร การจัดทำ ตลอดจนคำแนะนำต่าง ๆ เพื่อให้หนังสือเล่มนี้ มีความสมบูรณ์ยิ่งขึ้นจนสำเร็จเป็นรูปเล่มได้

หากหนังสือเล่มนี้มีความผิดพลาด ผู้เขียนยินดีน้อมรับ คำแนะนำ คำติชม เพื่อนำไปพัฒนาและปรับปรุงให้ดียิ่งขึ้นต่อไป ด้วยความเคารพยิ่ง

ผศ.ดร. เดชฤทธิ์ มณีธรรม
dech_m@ rmut.ac.th

SE-ED


SE-ED

สารบัญ

SE-ED

SE-ED

SE-ED

SE-ED

สารบัญรูป

SE-ED

SEED

SE-ED

สารบัญตาราง

SE-ED


SE-ED

1 ไมโครคอนโทรลเลอร์ (Microcontroller)




SE-ED

- ไมโครคอนโทรลเลอร์ MCS-51
- โครงสร้างของไมโครคอนโทรลเลอร์ (MCS-51)
- การจัดหน่วยความจำของไมโครคอนโทรลเลอร์ตระกูล MCS-51 แบบแฟลช
- อินเตอร์รัปต์ (Interrupt)


SE-ED

ปัจจุบันเครื่องใช้ไฟฟ้าอุปกรณ์อิเล็กทรอนิกส์ตลอดจนระบบโรงงานอุตสาหกรรมจะใช้ไมโครคอนโทรลเลอร์ซึ่งติดตั้งอยู่ภายในเป็นตัวควบคุมเกือบทั้งหมด ทำให้ไมโครคอนโทรลเลอร์กลายเป็นอีกหนึ่งอุปกรณ์ที่ผู้ผลิตสารกึ่งตัวนำหลายๆ บริษัทให้ความสนใจ และมีการแข่งขันสูงมาก อุปกรณ์ที่ใช้ไมโครคอนโทรลเลอร์ควบคุมมีมากมายหลายชนิด เช่น อุปกรณ์เครื่องใช้ภายในบ้าน สัญญาณไฟจราจร รถยนต์ ตลอดจนระบบอุตสาหกรรม PLC, CNC, Robot เป็นต้น

ปัจจุบัน ไมโครคอนโทรลเลอร์ (MCS-51) ซึ่งเป็นไมโครคอมพิวเตอร์แบบชิปเดี่ยว (Single Chip Microcontroller) ถูกใช้งานอย่างแพร่หลาย การศึกษาเกี่ยวกับไมโครคอนโทรลเลอร์จึงมีองค์ประกอบหลายอย่าง เช่น สถาปัตยกรรมของไมโครคอนโทรลเลอร์ วงจรอิเล็กทรอนิกส์ ตลอดจนภาษาที่จะใช้ในการเขียนซึ่งจะต้องศึกษาควบคู่กันไป โดยจะนำเสนอลำดับการทำงานดังต่อไปนี้

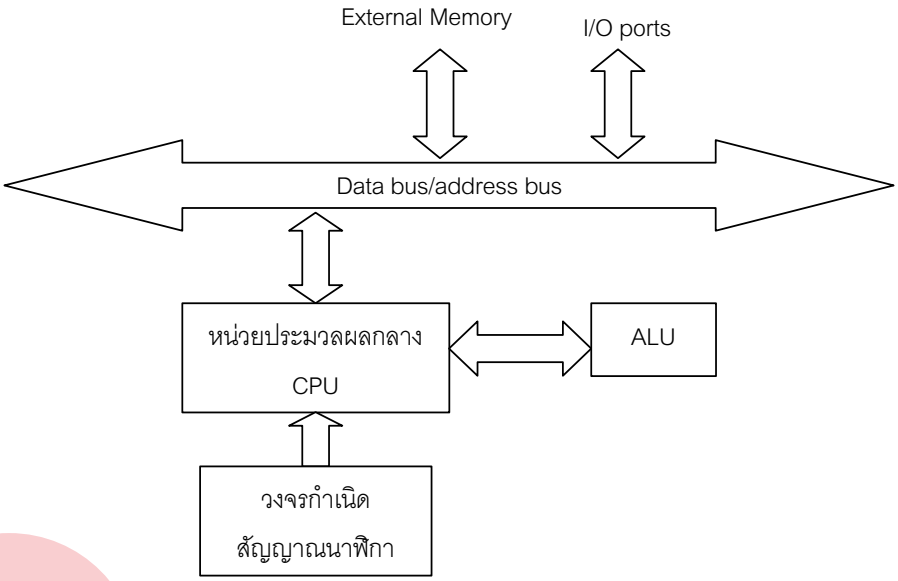
1.1 ไมโครคอนโทรลเลอร์ MCS-51

1.1.1 ความหมายของไมโครคอนโทรลเลอร์

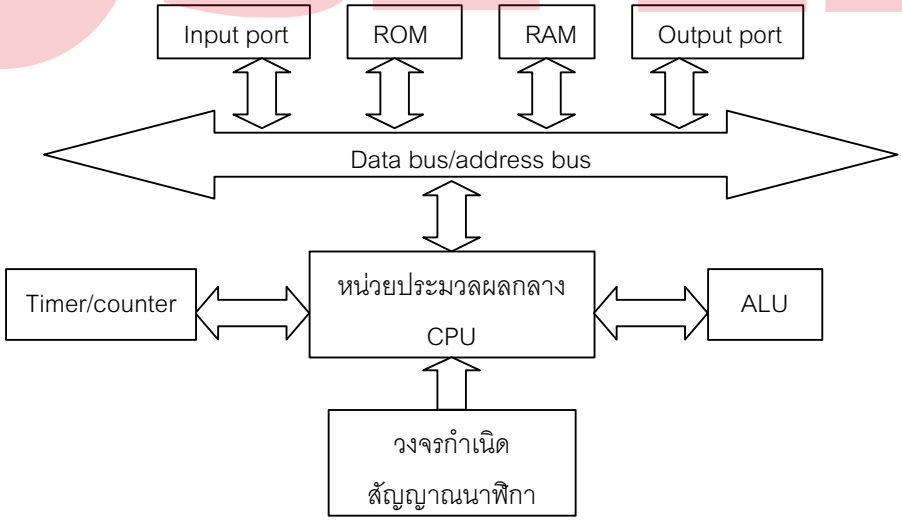
ไมโครคอนโทรลเลอร์ คือ อุปกรณ์อิเล็กทรอนิกส์อย่างหนึ่งซึ่งภายในประกอบด้วยวงจรอื่นๆ หลายวงจรและทำงานร่วมกัน เช่น หน่วยประมวลผลกลาง (CPU: Central Processing Unit) หน่วยคำนวณทางคณิตศาสตร์และลอจิก (ALU: Arithmatic Logic Unit) วงจรออสซิลเลเตอร์ หน่วยความจำ (Memory: ROM, RAM) วงจรรับสัญญาณอินพุทและขับสัญญาณเอาต์พุท (I/O port) เป็นต้น ด้วยเหตุนี้ไมโครคอนโทรลเลอร์จึงสามารถนำไปประยุกต์ใช้งานควบคุมต่างๆ ได้ดี เนื่องจากสามารถเขียนโปรแกรมควบคุมได้อย่างอิสระ แล้วแต่เราต้องการควบคุมอะไร

1.1.2 ข้อแตกต่างระหว่างไมโครโปรเซสเซอร์กับไมโครคอนโทรลเลอร์

ไมโครโปรเซสเซอร์ที่ใช้อยู่ในปัจจุบัน เช่น ซีพียูเบอร์ Z80 เป็นต้น จะไม่มีหน่วยความจำ RAM, ROM และ Port อยู่ในตัวชิป ทำให้ต้องต่อหน่วยความจำโปรแกรมภายนอกเพิ่มและต้องใช้ ICs ขยายพอร์ตเพิ่มเติม ข้อดีคือ สามารถเพิ่มหน่วยความจำได้ตลอด ส่วนไมโครคอนโทรลเลอร์จะมีวงจรพื้นฐานประกอบอยู่ภายในชิป เช่น หน่วยความจำ RAM, ROM และ I/O Port ดังนั้น ในระบบไมโครคอนโทรลเลอร์จึงมีขนาดเล็กกว่าและราคาต่ำกว่าระบบไมโครโปรเซสเซอร์



รูปที่ 1.1 โครงสร้างพื้นฐานของไมโครโปรเซสเซอร์



รูปที่ 1.2 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์

1.2 โครงสร้างของไมโครคอนโทรลเลอร์ (MCS-51)

Microcontroller MCS-51 มีโครงสร้างหลายลักษณะ ทั้ง 20 pin และ 40 pin โดยมีหน่วยความจำแบบ ROM หรือ EPROM ภายในมีขนาดไม่เกิน 4K byte และมีหน่วยความจำแบบ RAM 256 Byte โดย Microcontroller MCS-51 แบบ Flash มีพอร์ตให้ใช้งานทั้งสิ้น 4 พอร์ต คือ P0, P1, P2 และ P3 แต่ละพอร์ตจะมีขนาด 8 บิท เป็นพอร์ตแบบมี 2 ทิศทาง คือ สามารถเป็นได้ทั้งอินพุทและเอาต์พุท โดยมีคุณลักษณะดังต่อไปนี้

- มีพอร์ต I/O ขนาด 8 บิท 4 พอร์ต
- มีหน่วยความจำ ROM 4K byte
- มีหน่วยความจำ RAM 128 byte
- อ้างตำแหน่งหน่วยความจำโปรแกรมภายนอก 64 K byte
- อ้างตำแหน่งหน่วยความจำข้อมูลภายนอก 64 K byte
- Timer / Counter 16 บิท 2 ตัว
- วงจรสื่อสารแบบอนุกรมแบบฟูลดูเพล็กซ์ (Full Dulplex)
- วงจรควบคุมการอินเตอร์รัปต์จากแหล่งกำเนิดสัญญาณ 6 ประเภท
- วงจรออสซิสเลเตอร์ภายใน
- มีหน่วยความจำโปรแกรมแบบ Flash สามารถเขียนและลบได้พันครั้ง

โครงสร้างทางไมโครคอนโทรลเลอร์ MCS-51 แบบ Flash จะประกอบไปด้วยส่วนต่างๆ ดังต่อไปนี้

1.2.1 หน่วยประมวลผลกลาง (CPU: Central Processing Unit)

CPU เปรียบได้กับสมองของคนเรานั่นเอง เพราะการคำนวณต่างๆ เกิดขึ้นที่นี่ CPU ประกอบด้วยวงจรต่างๆ หลายวงจร เช่น Decoder, Register, Counter, Adder, Subtraction, Buffer, Oscillator เป็นต้น

1.2.2 หน่วยความจำ (Memory)

ในการเขียนโปรแกรมด้วยภาษา C ให้กับไมโครคอนโทรลเลอร์นั้นต้องคำนึงถึงชนิดของหน่วยความจำและวิธีการเข้าถึงด้วย ซึ่งต่างจากการเขียนบน PC ที่สนใจเพียงชนิดของตัวแปรว่าจะใช้เก็บข้อมูลประเภทใด สำหรับหน่วยความจำในระบบไมโครคอนโทรลเลอร์นั้นแบ่งเป็น 2 ประเภท ดังนี้

- **หน่วยความจำข้อมูลภายใน (Internal RAM)**

หน่วยความจำส่วนนี้มีไว้ใช้เก็บข้อมูลขณะประมวลผลโปรแกรม สามารถอ่านและเขียนข้อมูลได้ขณะมีไฟเลี้ยง แต่เมื่อไม่จ่ายไฟเลี้ยงข้อมูลต่างๆ จะสลายไป หากหน่วยความจำส่วนนี้ไม่พอใช้งานจะต้องต่อหน่วยความจำแรมภายนอกเพิ่ม (External RAM หรือ Data Memory) ปัจจุบันเทคโนโลยีก้าวหน้าขึ้นมากชิปบางตัวจะมีการบรรจุหน่วยความจำประเภท Data Memory รวมเข้าไปในชิปเลย

- **หน่วยความจำโปรแกรม (Program Memory หรือ ROM)**

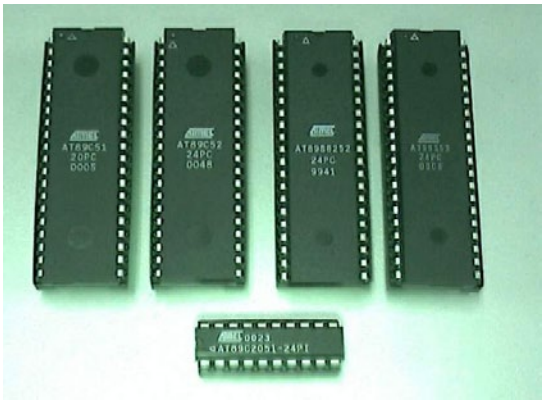
หน่วยความจำส่วนนี้ใช้เก็บโปรแกรมที่เราเขียน สามารถอ่านได้อย่างเดียวขณะประมวลผล ถ้าจะเขียนข้อมูลลง ROM จะต้องใช้เครื่องโปรแกรม เมื่อก่อนจะเก็บไว้ใน EPROM (เขียนและลบด้วย UV) แต่ปัจจุบันจะใช้ flash ROM ที่สามารถเขียนและลบได้ด้วยกระแสไฟฟ้า ซึ่งสะดวกและรวดเร็วมาก

1.2.3 พอร์ตอินพุท/เอาต์พุท (I/O port)

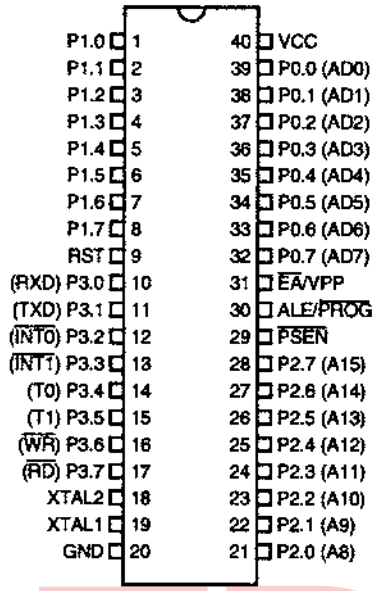
พอร์ตมีหน้าที่ทำให้ระบบไมโครคอนโทรลเลอร์สามารถติดต่อสื่อสารกับอุปกรณ์ภายนอกได้ แล้วแต่วัตถุประสงค์ในการใช้งานและคุณสมบัติของพอร์ต เช่น สามารถติดต่อสื่อสารกับอุปกรณ์ Pushbutton, Keypad, Sensor, LCD เป็นต้น

ตารางที่ 1.1 Microcontroller MCS-51

เบอร์	หน่วยความจำโปรแกรม	หน่วยความจำข้อมูล	Timer/Counter
AT89C1051	1K byte	64 byte	1
AT89C2051	2K byte	128 byte	2
AT89C51	4K byte	128 byte	2
AT89C52	8K byte	256 byte	3
AT89C55	20K byte	256 byte	3
AT89S53	12K byte	256 byte	3



ก) Microcontroller



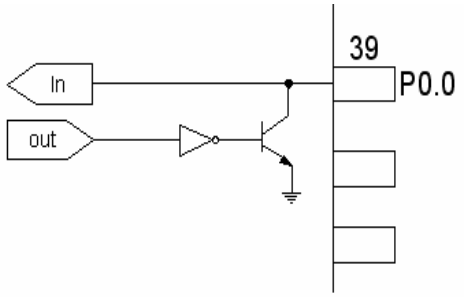
ข) การจัดขาของไมโครคอนโทรลเลอร์

รูปที่ 1.3 **ไมโครคอนโทรลเลอร์ตระกูล** MCS-51

SE-ED

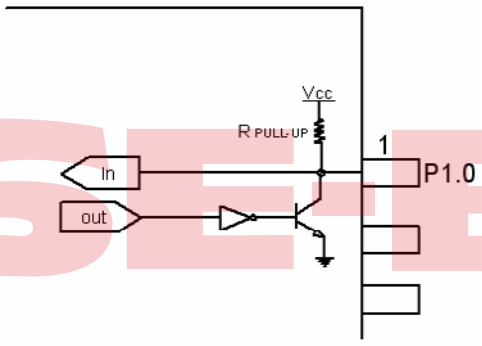
- **ขา 40 "Vcc" :** ขานี้ต้องต่อไฟเลี้ยง +5VDC
- **ขา 20 "GND" :** ต่อลงกราวด์ของระบบ
- **ขา 32-39 "Port 0" :** ขาทั้ง 8 ขานี้มีหน้าที่ใช้งานได้ทั้งอินพุทและเอาต์พุทสำหรับใช้ควบคุมอุปกรณ์ภายนอก โดยถ้าต้องการใช้ขาเป็นอินพุทต้องเขียนลอจิก "1" ไปที่ขาที่ต้องการ (ที่ขา port 0 มีสถานะ High Impedance เพราะไม่มีการต่อ R pull-up ไว้ภายใน) นอกจากนี้ยังใช้ติดต่อกับขา address ไบต์ต่ำของหน่วยความจำภายนอก (A0-A7) และขาข้อมูล (D0-D7) โดยวิธีการมัลติเพล็กซ์เพื่อสลับหน้าที่การทำงาน

Note !	กระแสซิงค์ (I_{sink}) ≈ 20 mA และ กระแสซอร์ส (I_{source}) ≈ 5mA



รูปที่ 1.4 **โครงสร้างอย่างง่ายภายใน** Port 0

- **ขา 1-8 "Port 1" :** ขานี้มีหน้าที่ใช้งานได้ทั้งอินพุทและเอาต์พุทสำหรับใช้ควบคุมอุปกรณ์ภายนอก ภายในมีการต่อ R pull-up อยู่แล้ว



รูปที่ 1.5 **โครงสร้างอย่างง่ายภายใน** Port 1

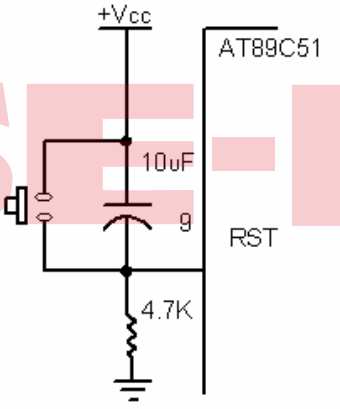
- **ขา 21-28 "Port 2" :** ขาทั้ง 8 ขานี้มีหน้าที่ใช้งานได้ทั้งอินพุทและเอาต์พุทสำหรับใช้ควบคุมอุปกรณ์ภายนอก นอกจากนี้ยังใช้ติดต่อกับขา Address ไบต์สูงหน่วยความจำภายนอก (A8-A15) โครงสร้างพอร์ตภายในเหมือนพอร์ต 1

- **ขา 10-17 "Port 3" :** ขาทั้ง 8 ขานี้มีหน้าที่ใช้งานได้ทั้งอินพุทและเอาต์พุทสำหรับใช้ควบคุมอุปกรณ์ภายนอก นอกจากนี้ยังมีหน้าที่พิเศษในการติดต่อรับสัญญาณควบคุมการทำงานของไมโครคอนโทรลเลอร์ดังต่อไปนี้

  - P3.0 (RXD) ใช้รับสัญญาณจาก Serial port หรือ RS232
  - P3.1 (TXD) ใช้ส่งสัญญาณทาง Serial port

- P3.2 ($\overline{INT0}$) ใช้รับสัญญาณ interrupt หมายเลข 0
- P3.3 ($\overline{INT1}$) ใช้รับสัญญาณ interrupt หมายเลข 1
- P3.4 (T0) เป็นขารับสัญญาณ pulse หมายเลข 0 เข้าวงจร Counter
- P3.5 (T1) เป็นขารับสัญญาณ pulse หมายเลข 1 เข้าวงจร Counter
- P3.6 ($\overline{WR}$) เป็นขาสัญญาณเขียน RAM
- P3.7 ($\overline{RD}$) เป็นขาสัญญาณอ่าน RAM

- **ขา 9 "Reset" :** ขานี้มีไว้สำหรับรีเซ็ตไมโครคอนโทรลเลอร์ โดยระบบจะรีเซ็ตเมื่อขานี้ได้รับสัญญาณพัลส์ (Pulse) บวกเป็นเวลาอย่างน้อย 2 แมชีนไซเคิล (Machine Cycle) ดังนั้นจะต้องสร้างวงจรรีเซ็ตให้กับขานี้ โดยใช้วงจร RC หรือใช้ IC reset โดยเฉพาะก็ได้ ในที่นี้จะใช้วงจร RC

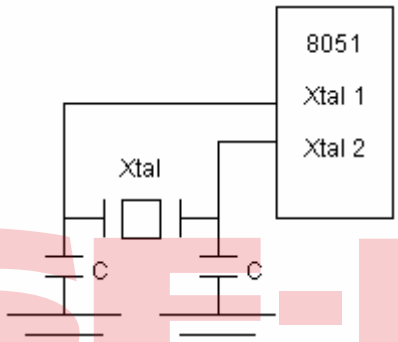


รูปที่ 1.6 **วงจร** Reset

เมื่อเริ่มจ่ายไฟ C จะเริ่มเก็บประจุ ขณะนี้ C เปรียบเสมือน short circuit แรงดันตกคร่อม C เป็น 0 โวลต์ แต่แรงดันตกคร่อม R = 5V เมื่อเวลาผ่านไป C ก็เก็บประจุมากขึ้นๆ ตรงกันข้ามแรงดันคร่อม R จะลดลงตามกฎการแบ่งแรงดัน จนในที่สุดแรงดันคร่อม C = 5Vc และแรงดันคร่อม R = 0V ด้วยเหตุนี้เองทำให้เกิดสัญญาณพัลส์ที่ขา 9 ส่วนความกว้างของพัลส์นั้นขึ้นอยู่กับค่า R, C ซึ่งเป็นไปตามหลักการของวงจร RC ส่วนสวิตช์ที่คร่อม C นั้นมีหน้าที่ discharge ให้กับ C หรือเป็นการ reset MCS-51

Note !	R ที่เหมาะสมมีค่าประมาณ 1K-10K และ C ที่เหมาะสมมีค่าประมาณ 1uF-10uF

- **ขา 18, 9 "X ' tal 1, X ' tal 2" :** ขาทั้งสองข้างนี้มีไว้ต่อกับวงจรสร้างสัญญาณนาฬิกาให้กับ MCS-51 ค่าความถี่ให้ใช้ตามสเปคของแต่ละเบอร์ หากต้องการสื่อสารแบบอนุกรมด้วย ค่าที่เหมาะสมคือ 11.0592 MHz เพราะค่าความถี่นี้จะตรงกับมาตรฐาน RS232 ดังรูปที่ 1.7 ส่วนค่า C ใช้ได้ตั้งแต่ 20-35 pF กรณีที่งานที่จะออกแบบไม่มีการสื่อสารอนุกรมก็สามารถใช้ X'tal ค่าอื่นก็ได้ที่อยู่ในช่วงที่ชิปรับได้ เช่น 4-24 MHz



รูปที่ 1.7 การต่อวงจรสร้างสัญญาณนาฬิกา

- **ขา 30 "ALE/PROG" :** เป็นขาที่บอกว่ามีสัญญาณแอดเดรส A0-A7 ออกมาจากพอร์ต 0 byte ต่ำ เพื่อให้ไอซีแลตช์ (IC Latch) ทำการเก็บค่าแอดเดรสดังกล่าวไว้ก่อนที่จะหายไป เนื่องจากการมัลติเพล็กซ์เป็นดาต้าบัส (Data Bus)

  ในการใช้หน่วยความจำภายนอก หรือในชิปบางรุ่นที่มี EEPROM ขานี้ยังใช้รับพัลส์ของการโปรแกรมเพื่อโปรแกรมข้อมูลลงชิปอีกด้วย

- **ขา 29 "PSEN" :** Program Store Enable ขานี้จะใช้ติดต่อกับหน่วยความจำโปรแกรมภายนอก โดยจะส่งสัญญาณมาที่ขานี้ 2 ครั้ง ในแต่ละแมชีนไซเคิล (Machine Cycle)

- **ขา 31 "EA / Vpp" :** เป็นขาเลือกว่าจะใช้งานหน่วยความจำโปรแกรมภายในหรือภายนอก โดยถ้าขานี้ได้รับลอจิก "1" จะเป็นการใช้หน่วยความจำโปรแกรมภายใน

SE-ED

1.3 การจัดหน่วยความจำของไมโครคอนโทรลเลอร์ ตระกูล MCS-51 แบบแฟลช

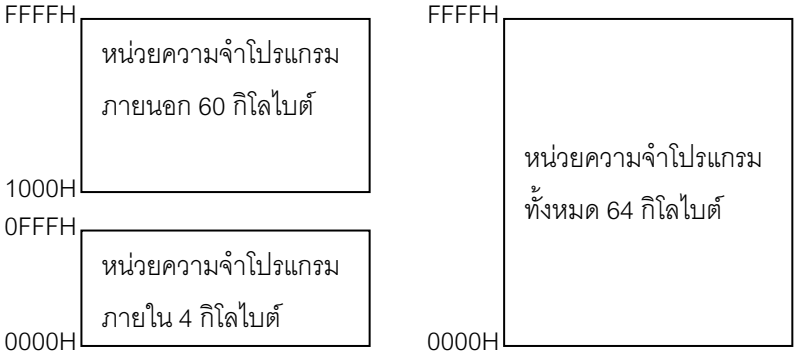
ในไมโครคอนโทรลเลอร์ตระกูล MCS-51 แบบแฟลช มีหน่วยความจำอยู่ 2 ส่วนคือ หน่วยความจำโปรแกรม (Program Memory) และหน่วยความจำข้อมูล (Data Memory) ซึ่งชิปแต่ละเบอร์ก็จะมีขนาดและราคาไม่เท่ากัน ทั้งนี้ขึ้นอยู่กับความต้องการของผู้ใช้

1.3.1 หน่วยความจำโปรแกรม (Program Memory)

หน่วยความจำชนิดนี้ใช้เก็บโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์หรือโปรแกรมมอนิเตอร์ (Monitor Program) อ่านได้อย่างเดียว ซึ่งสามารถติดต่อหน่วยความจำโปรแกรมได้สูงสุด 64 กิโลไบต์ ถ้าเป็นเบอร์ AT89C51 จะมีหน่วยความจำโปรแกรม 4 กิโลไบต์ หากไม่เพียงพอก็อาจต่อหน่วยความจำโปรแกรมภายนอกเพิ่ม หรือใช้เบอร์ที่มีขนาดหน่วยความจำโปรแกรมใหญ่ขึ้น เช่น AT89C52 ในบางเบอร์ก็มีหน่วยความจำโปรแกรมภายในถึง 64 กิโลไบต์ แต่ราคาจะสูงขึ้นมาก

SE-ED

Note ! CPU จะไปอ่านคำสั่งในหน่วยความจำโปรแกรมที่ Address 0000H เสมอ เมื่อเกิดขบวนการ Reset

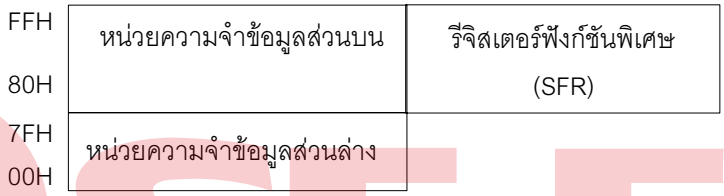


รูปที่ 1.8 แสดงการจัดพื้นที่หน่วยความจำโปรแกรมของชิปเบอร์ AT89C51

1.3.2 หน่วยความจำข้อมูล (Data Memory)

หรือเรียกอีกอย่างว่า หน่วยความจำแรม หน่วยความจำชนิดนี้สามารถอ่านและเขียนได้ ซึ่งมีด้วยกัน 2 แบบ คือ หน่วยความจำข้อมูลภายใน (Internal RAM) และหน่วยความจำข้อมูลภายนอก (External RAM) ขนาดก็แตกต่างกันในแต่ละเบอร์

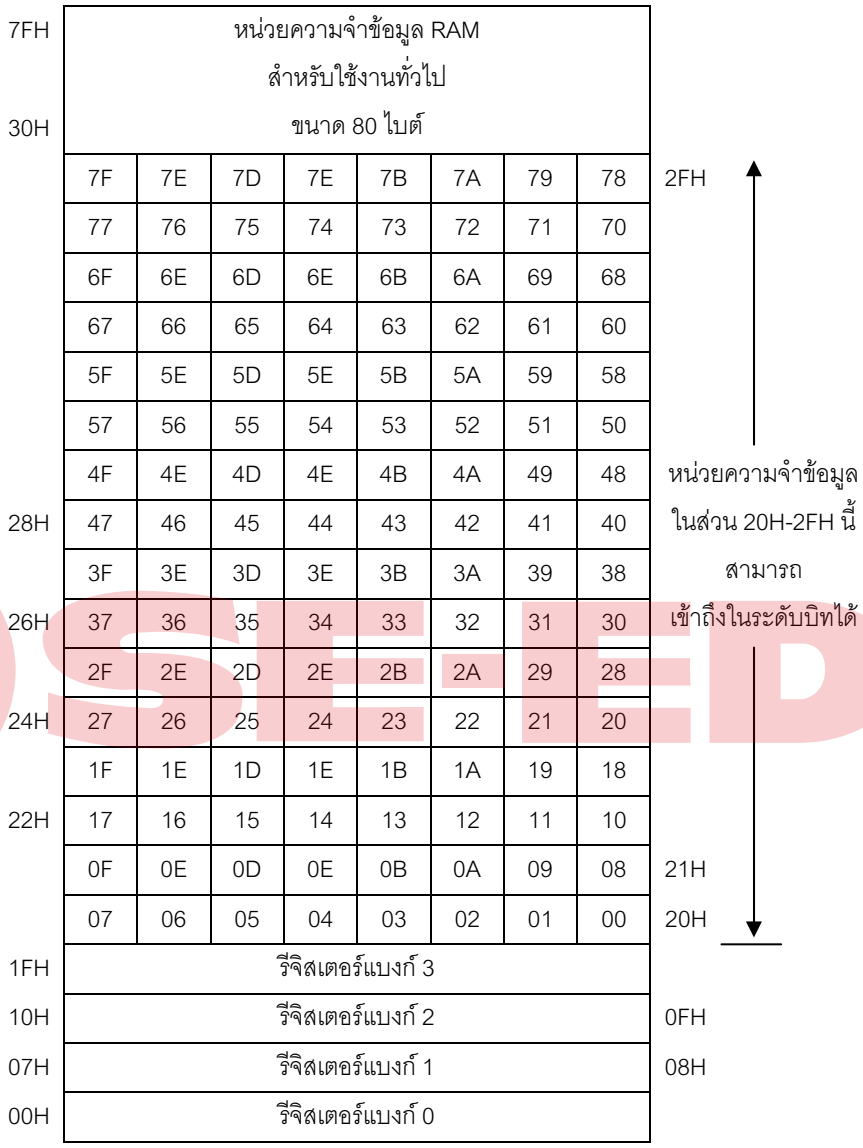
Note !	สมัยก่อนเรียกแรมว่า Data Memory แต่ปัจจุบันเทคโนโลยีก้าวหน้าขึ้น ผู้ผลิตชิปหลายรายสามารถใส่หน่วยความจำส่วนนี้เข้าไปในชิปได้เลย ดังนั้นจึงไม่เรียก Data Memory ว่าเป็นแรมภายนอก



รูปที่ 1.9 พื้นที่หน่วยความจำข้อมูลภายใน (Internal RAM)

จากรูปที่ 1.9 จะเห็นว่ามีการแบ่งพื้นที่หน่วยความจำข้อมูลภายใน (Internal RAM) ออกเป็น 3 ส่วนคือ หน่วยความจำข้อมูลส่วนล่างซึ่งมีขนาด 128 ไบต์ (00H-7FH), ส่วนบนขนาด 128 ไบต์ (80H-FFH) และรีจิสเตอร์ฟังก์ชั่นพิเศษ (SFR : Special Function Register) ซึ่งพื้นที่ส่วนนี้จะซ้อนทับกับหน่วยความจำข้อมูลส่วนบน แต่มีวิธีการเข้าถึงแตกต่างกัน

SE-ED



รูปที่ 1.10 แสดงพื้นที่หน่วยความจำข้อมูลภายใน (Internal RAM) ขนาด 128 ไบต์

B7	B6	B5	B4	B3	B2	B1	B0	FOH รีจิสเตอร์ B
A7	A6	A5	A4	A3	A2	A1	A0	EOH รีจิสเตอร์ ACC
D7	D6	D5	D4	D3	D2	D1	D0	DOH รีจิสเตอร์
			D4	D3	D2	D1	D0	BBH รีจิสเตอร์ IP
3.7	3.6	3.5	3.4	3.3	3.2	3.1	3.0	BOH รีจิสเตอร์ P3
D7			D4	D3	D2	D1	D0	ABH รีจิสเตอร์ IE
2.7	2.6	2.5	2.4	2.3	2.2	2.1	2.0	AOH รีจิสเตอร์ P2
S7	S6	S5	S4	S3	S2	S1	S0	99H รีจิสเตอร์
1.7	1.6	1.5	1.4	1.3	1.2	1.1	1.0	90H รีจิสเตอร์ P1
เข้าถึงระดับบิทไม่ได้								D8H รีจิสเตอร์
เข้าถึงระดับบิทไม่ได้								8CH รีจิสเตอร์ THO
เข้าถึงระดับบิทไม่ได้								8BH รีจิสเตอร์ TL1
เข้าถึงระดับบิทไม่ได้								8AH รีจิสเตอร์
เข้าถึงระดับบิทไม่ได้								89H รีจิสเตอร์ TMOD
T7	T6	T5	T4	T3	T2	T1	T0	88H รีจิสเตอร์ TCON
เข้าถึงระดับบิทไม่ได้								87H รีจิสเตอร์ PCON
เข้าถึงระดับบิทไม่ได้								83H รีจิสเตอร์ DPH
เข้าถึงระดับบิทไม่ได้								82H รีจิสเตอร์ DPL
เข้าถึงระดับบิทไม่ได้								81H รีจิสเตอร์ SP
0.7	0.6	0.5	0.4	0.3	0.2	0.1	0	80H รีจิสเตอร์ PO

รูปที่ 1.11 การจัดพื้นที่จองรีจิสเตอร์ฟังก์ชันพิเศษ (SFR)

1.4 อินเตอร์รัปต์ (Interrupt)

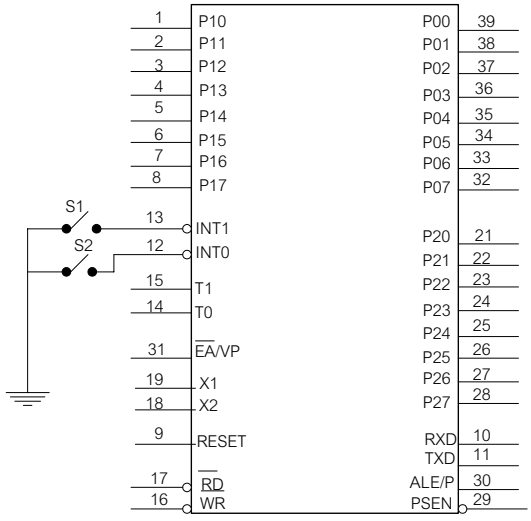
ในหน่วยนี้จะศึกษาเกี่ยวกับการอินเตอร์รัปต์ (Interrupt) ว่าเกิดขึ้นได้อย่างไร มีหลักการทำงานอย่างไร มีความสำคัญอย่างไรและจะเกิดอะไรขึ้นเมื่อมีการอินเตอร์รัปต์ สำหรับไมโครคอนโทรลเลอร์ MCS-51 จะมีการเกิดสัญญาณอินเตอร์รัปต์ ได้หลายลักษณะคือ

1. สัญญาณการเกิดอินเตอร์รัปต์จากภายนอก (External Interrupt)
2. สัญญาณการเกิดอินเตอร์รัปต์จากภายใน (Internal Interrupt)
3. สัญญาณการเกิดอินเตอร์รัปต์จากวงจรนับเวลา (Timer Interrupt)
4. สัญญาณการเกิดอินเตอร์รัปต์จากการรับส่งข้อมูลอนุกรม (Serial Interrupt)

โดยในหน่วยนี้จะกล่าวถึงการเกิดอินเตอร์รัปต์จากภายนอก จากวงจรนับเวลา และจากการรับส่งข้อมูลอนุกรม ดังต่อไปนี้

1.4.1 อินเตอร์รัปต์ภายนอก (External Interrupt)

อินเตอร์รัปต์ภายนอก หมายถึง การเกิดอินเตอร์รัปต์ภายนอกของไมโครคอนโทรลเลอร์ ไม่ได้เกิดจากภายใน โดยจะเกิดขึ้นที่ขาสัญญาณ INT0 (P3.2) และขาสัญญาณ INT1 (P3.3) ของพอร์ต 3 นั่นเอง



รูปที่ 1.12 **การอินเตอร์รัปต์ภายนอก**

จากรูปสามารถนำเอา Switch S1, S2 หรือ Sensor มาต่อเข้ากับขาสัญญาณ INT0 หรือ INT1 ได้โดยตรง ส่วนอีกด้านหนึ่งก็จะต่อลงกราวด์ ทั้ง 2 ขานี้ จะทำงานก็ต่อเมื่อเราป้อนลอจิก “0” ให้กับตัวมัน เพราะขานี้จะทำงานที่ลอจิก “0” ซึ่งที่ขานี้ไม่จำเป็นต้องต่อ R pull-up เพราะภายในพอร์ตมี R pull-up ภายในอยู่แล้ว

■ Interrupt มีหลักการอย่างไร

Interrupt ก็คือ การขัดจังหวะของการทำงานในช่วงเวลาหนึ่ง แล้วให้ไปทำงานอีกอย่างหนึ่ง เมื่อทำเสร็จแล้วก็จะกลับไปทำงานเดิม ตัวอย่างเช่น สมมติว่าขณะที่เรากำลังดูทีวีอยู่นั้น เกิดมีเสียงโทรศัพท์ดังขึ้น เราก็จำเป็นที่จะต้องหยุดดูทีวีเพื่อมารับโทรศัพท์ชั่วขณะก่อน เมื่อคุยเสร็จแล้วจึงกลับไปดูทีวีต่อ ลักษณะนี้โทรศัพท์เป็นตัว Interrupt

ตัวเรา คือ CPU
การดูทีวี คือ โปรแกรมหลัก
โทรศัพท์ คือ การ Interrupt

■ ชนิดของการ Interrupt

การเกิดอินเตอร์รัปต์ก็สามารถแบ่งออกตามลักษณะการทำงานได้ 2 ลักษณะ ดังนี้

- Software Interrupt หมายถึง การ Interrupt จากคำสั่งที่เราเขียน เช่น คำสั่ง CALL ในภาษาแอสเซมบลี ส่วนภาษาซีก็ใช้การเขียนฟังก์ชัน Interrupt ขึ้นมา
- Hardware Interrupt หมายถึง การ Interrupt จากวงจรภายนอกหรือวงจรภายใน เช่น External Interrupt, Timer Interrupt, Serial Interrupt เป็นต้น

■ การเขียนโปรแกรมสำหรับการ Interrupt

ทำได้โดยการเซ็ตบิทของรีจิสเตอร์ Interrupt รีจิสเตอร์ตัวนี้มีชื่อว่า IE (Interrupt Enable Register)

ตารางที่ 1.2 IE (Interrupt Enable Register) ตำแหน่งที่ A8h

IE.7	IE.6	IE.5	IE.4	IE.3	IE.2	IE.1	IE.0
EA	-	ET2	ES	ET1	EX1	ET0	EX0

SE-ED

บิท	ความหมาย
EA	ทำหน้าที่เปิด/ปิด การ Interrupt ของ Interrupt ทุกตัว (Active High)
ET2	ทำหน้าที่เปิด/ปิด การ Interrupt ของ Timer/Counter#2 ซึ่งจะมีอยู่ในเบอร์ AT89C52 ขึ้นไป
ES	ทำหน้าที่เปิด-ปิด การ Interrupt ของ Serial Port
ET1	ทำหน้าที่เปิด/ปิด การ Interrupt ของ Timer/Counter#1
EX1	ทำหน้าที่เปิด/ปิด การ Interrupt ของขา INT1
ET0	ทำหน้าที่เปิด/ปิด การ Interrupt ของ Timer/Counter#0
EX0	ทำหน้าที่เปิด/ปิด การ Interrupt ของขา INT0

ตัวอย่าง การเซ็ตค่ารีจิสเตอร์ IE ให้สามารถใช้งานอินเตอร์รัปต์ได้

ตัวอย่างที่ 1.1 สมมติว่าต้องการใช้งาน INT0 มีขั้นตอนดังนี้

1. กำหนดค่าบิทที่ 7 ในรีจิสเตอร์ IE หรือบิท EA = 1 : เพื่อเปิดการใช้งาน Interrupt
2. กำหนดค่าบิทที่ 0 ในรีจิสเตอร์ IE หรือบิท EX0 = 1 : เพื่อใช้งานขา INT0 ดังนั้น ค่าที่จะกำหนดให้กับรีจิสเตอร์ IE = 1000 0001 หรือ 81H

ตัวอย่างที่ 1.2 สมมติว่าต้องการใช้งาน Timer#1 มีขั้นตอนดังนี้

1. กำหนดค่าบิทที่ 7 ในรีจิสเตอร์ IE หรือบิท EA = 1 : เพื่อเปิดการใช้งาน Interrupt
2. กำหนดค่าบิทที่ 3 ในรีจิสเตอร์ IE หรือบิท ET0 = 1 : เพื่อใช้งานขา Timer#1

ดังนั้น ค่าที่จะกำหนดให้กับรีจิสเตอร์ IE = 1000 1000 หรือ 88H

ตัวอย่างที่ 1.3 การเขียนโปรแกรมใช้งาน Interrupt 0 โดยต่อวงจรตามรูปที่ 1.12 และต่อ LED เข้าที่พอร์ต 1 เมื่อไม่มีการกด S1 ที่พอร์ต 1 จะส่งค่า 0xFA ออกมา แต่เมื่อมีการกด S1 ที่พอร์ต 1 จะส่งค่า 0x55 ออกมาแทน

```
#include<reg51.h>
void main (void)
{
  IE = 0x81;
```

```
  While (1)
  {
   P1 = 0xFA;
  }
}
void check (void) interrupt 0     //  บิทที่ 0 หรือ EX0
{
P1 = 0x55;
}
```

จากตัวอย่าง เราทำการกดสวิตช์ S1 ขาสัญญาณ INT0 มีลอจิก 0 เข้ามา โปรแกรมจะกระโดดไปทำคำสั่งที่ Address 0003H ทันที หรือจะกระโดดมาทำโปรแกรมย่อยทันที และจะกระโดดกลับไปที่โปรแกรมหลักอีกครั้ง (กลับไปทำที่ Address ที่กระโดดมา) เมื่อทำงานโปรแกรมย่อยเสร็จ เหตุที่ MCS-51 ถูก Interrupt จาก External Interrupt#0 (INT0) แล้วกระโดดไปทำคำสั่งที่ Address 0003H เพราะว่า MCS-51 จะถูกกำหนดค่าในการ Interrupt โดยจะกำหนดให้ไปทำคำสั่งที่ Address ต่างๆ เมื่อเกิดการ Interrupt ดังตารางที่ 1.3

ตารางที่ 1.3 ตำแหน่งการเกิด Interrupt

บิท Interrupt	Interrupt	บิท Flag	Address
EX0	External Interrupt#0	IE0	0003H
ET0	Timer/Counter#0	TF0	000BH
EX1	External Interrupt#1	IE1	0013H
ET1	Timer/Counter#1	TF1	001BH
ES	Serial Port	RI&TI	0023H
ET2	Timer/Counter#2	TF2 & EXF2	002BH

จากตารางเป็นตำแหน่ง Address ของการ Interrupt ที่โปรแกรมจะกระโดดไป โดยสังเกตว่ามีบิท Flag ที่เกิดจาก Interrupt ของแต่ละตัว ซึ่ง Interrupt แต่ละตัวจะมีบิท Flag ประจำตัวอยู่ และจะทำหน้าที่เป็นตัวบอก MCS-51 ว่าเกิด Interrupt จากตัวไหน สมมติว่าเกิดการ Interrupt จาก INT0 โดยการกดสวิตช์ S1 ขา INT0 จะเป็นลอจิก “0” MCS-51 จะทำการเซ็ตบิท IE0 ให้เป็น “1” ทันที จากนั้นเมื่อ MCS-51 รู้ว่าบิท IE0 ถูกเซ็ตเป็น “1” มันก็จะทำการตรวจสอบว่าบิท EA และบิท EX0 เป็น “1” หรือเปล่า ถ้าตรวจสอบว่าเป็น “1” MCS-51 จะกระโดดไปทำคำสั่งที่ Address 0003H ทันที จากนั้นมันจะทำคำสั่งไปเรื่อยๆ จนกว่าจะพบคำสั่ง RETI (Return Interrupt) ในภาษาแอสเซมบลี มันจึงจะกลับมาทำคำสั่งเดิมที่มันกระโดดมา บิท Flag เหล่านี้จะอยู่ที่รีจิสเตอร์ TCON ดังแสดงตารางที่ 1.4

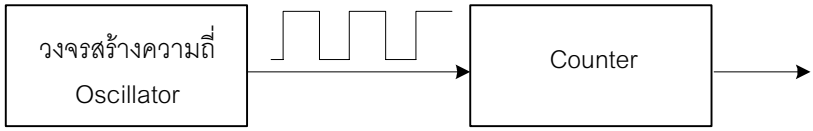
ตารางที่ 1.4 TCON (Timer/Counter Control Register) ตำแหน่งที่ 88H

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

บิท	ความหมาย
TF1	จะถูกเซ็ต เมื่อ Timer/Counter#1 Over Flow ถูกเคลียร์โดย Hardware
TR1	เปิดปิดการทำงานของ Timer/Counter#1
TF0	จะถูกเซ็ต เมื่อ Timer/Counter#0 Over Flow ถูกเคลียร์โดย Hardware
TR0	เปิดปิดการทำงานของ Timer/Counter#0
IE1	จะถูกเซ็ต เมื่อมีการ Interrupt จากขา INT1 ถูกเคลียร์โดย Hardware
IT1	จะถูกเซ็ต เมื่อมีสัญญาณเข้าที่ขา T1
IE0	จะถูกเซ็ต เมื่อมีการ Interrupt จากขา INT0 ถูกเคลียร์โดย Hardware
IT0	จะถูกเซ็ต เมื่อมีสัญญาณเข้าที่ขา T0

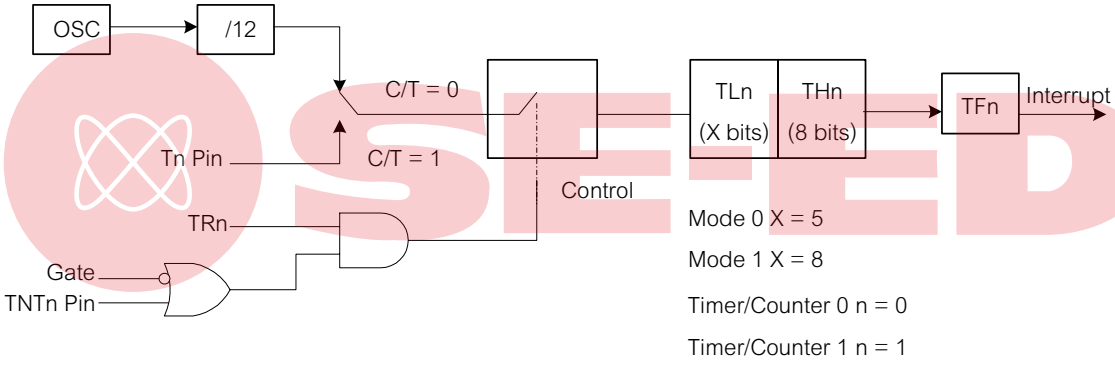
1.4.2 Timer Interrupt

ในหน่วยนี้จะกล่าวถึงการทำงานพื้นฐานของวงจร Timer ว่าทำงานอย่างไร และมีรีจิสเตอร์อะไรบ้างที่เกี่ยวข้องกับ Timer อีกทั้งวิธีคำนวณหาค่าความถี่ของ Timer ที่ต้องการ



รูปที่ 1.13 แสดงวงจร Timer พื้นฐาน

จากรูปข้างบนเป็นวงจรพื้นฐานของวงจร Timer ใน MCS-51 สามารถแบ่งออกเป็น 2 ส่วนหลักๆ คือ วงจร Oscillator ซึ่งทำหน้าที่กำเนิดสัญญาณนาฬิกาขึ้นมา จากนั้นก็จะส่งสัญญาณนาฬิกาไปให้วงจร Counter เพื่อทำหน้าที่หารความถี่ ซึ่งขึ้นอยู่กับค่าหารของวงจร Counter ว่ามีค่าเท่าไร สมมติให้วงจรหารด้วย 50 หมายความว่า ต้องมีสัญญาณนาฬิกาเข้าที่วงจร Counter จำนวน 50 ลูก จึงจะมีสัญญาณนาฬิกาออกที่เอาต์พุทของวงจร Counter 1 ลูก



รูปที่ 1.14 แสดงวงจรการทำงานของวงจร Timer

ตารางที่ 1.5 TMOD (Timer/Counter Mode Control Register) ตำแหน่งที่ 89H

Gate1	C/T1	M1	M0	Gate0	C/T0	M1	M0

บิท	ความหมาย
Gate1	Timer/Counter จะทำงานเมื่อบิท TR ใน TCON = 1 และสถานะลอจิกที่ขา INT0, INT1 เป็น “1” เป็นการควบคุมทาง Hardware (อ้างอิงจากแหล่งกำเนิดภายนอก)
C/T1	เลือกให้ทำงานเป็น Counter จะรับอินพุทจากภายนอกเข้าทางขา T0 หรือขา T1
M1	เป็นบิทบนสำหรับเลือก Mode การทำงานของ Timer 1
M0	เป็นบิทล่างสำหรับเลือก Mode การทำงานของ Timer 1

บิท	ความหมาย
Gate0	Timer/Counter จะทำงานเมื่อบิท TR ใน TCON = 1 เป็นการควบคุมทาง Software (อ้างอิง X'tal ภายใน)
C/T0	เลือกให้ทำงานเป็น Timer0 (อ้างอิง X'tal ภายใน)
M1	เป็นบิทบนสำหรับเลือก Mode การทำงานของ Timer 0
M0	เป็นบิทล่างสำหรับเลือก Mode การทำงานของ Timer 0

ตารางที่ 1.6 การเลือก Mode Timer/Counter

M1	M0	Mode	การทำงาน
0	0	0	ใช้งาน Timer/Counter แบบ 13 Bit
0	1	1	ใช้งาน Timer/Counter แบบ 16 Bit
1	0	2	ใช้งานแบบ 8 Bit Auto Reload
1	1	3	ใช้งานแบบแยกบิทโดยแบ่งเป็นแบบ 8 บิท 2 ตัว

ขั้นตอนในการหาค่า TH0, TL0 ของ Timer 0

(Mode 1, 16 Bit/Counter)

1. กำหนดเวลาความถี่ที่ต้องการให้ Timer Interrupt : f (ในกรณีที่เป็นเวลาให้เปลี่ยนเป็นความถี่ โดยใช้สูตร f = 1/T)
2. เอาค่าความถี่ของ X-TAL มาหารด้วย 12 = f_{12}
3. นำค่า f_{12} / f = F
4. นำค่า F แปลงเป็นเลขฐาน 16 = F_h
5. นำค่า 10000H – F_h = T
6. TH0 = ไบท์สูงของ T, TLO = ไบต์ต่ำของ T

ตัวอย่างที่ 1.4 จับเวลาทุกๆ 1500 μsec (X'tal = 18 MHz, โดยใช้ MCS-51)

วิธีคำนวณ

1. กำหนดเวลาความถี่ที่ต้องการให้ Timer Interrupt : f (ในกรณีที่เป็นเวลาให้เปลี่ยนเป็นความถี่ โดยใช้สูตร f = 1/T)

$$f = 1/1500\ \mu sec$$

2. เอาค่าความถี่ของ X-TAL มาหารด้วย 12 = f_{12}

$$f_{12} = 18\ MHz/12 = 1.5\ MHz$$

3. นำค่า $f_{12}/f = F$

$$F = 1.5\ MHz/\ (1/1500\mu sec) = 2250$$

4. นำค่า F แปลงเป็นเลขฐาน 16 = F_h

$$F_h = 8CAH$$

5. นำค่า $10000H - F_h = T$

$$T = 10000H-8CAH = F736H$$

ดังนั้น TH0 หรือ TH1 = F7H

TL0 หรือ TL1 = 36H

หรืออาจคิดแบบไม่ใช้สูตรได้ดังนี้

- เวลาที่ใช้ทั้งหมดใน 1 machine cycle = 12/18MHz
  = 0.666 μsec
- จำนวนครั้งในเวลา 1500 μsec = 1500 μsec / 0.666 μsec
  = 2250 ครั้ง

SE-ED

ดังนั้น ค่าที่ต้องการ = 65536-2250 = 63286 = F736H

TH0, TH1 = F7H

TL0, TL1 = 36H

ตัวอย่างที่ 1.5 จับเวลาทุกๆ 1500 μsec แล้วสร้าง pulse ออกมา ในขณะเดียวกันให้หลอด LED วิ่งจากซ้ายไปขวาที่ port 1 (ใช้ Timer 0, Mode1)

TMOD (Timer/Counter Mode Control Register)

Gate 1	C/T 1	M1	M0	Gate 0	C/T 0	M1	M0
0	0	0	0	0	0	0	1

TMOD = 0x01

IE (Interrupt Enable Register)

EA	-	ET2	ES	ET1	EX1	ET0	EX0
1	-	0	0	0	0	1	0

IE = 0x82

วิธีคำนวณค่า TH0, TL0

1. f = 1/1500 μsec
2. f_{12} = 11.0592 MHz/12 = 0.9216 MHz
3. F = 0.9216MHz / (1/1500 μsec) = 1832
4. F_h = 728H
5. T = 10000H-728H = F8D8H

ดังนั้น TH0 = F8H และ TL0 = D8H สามารถเขียนโปรแกรมได้ดังนี้

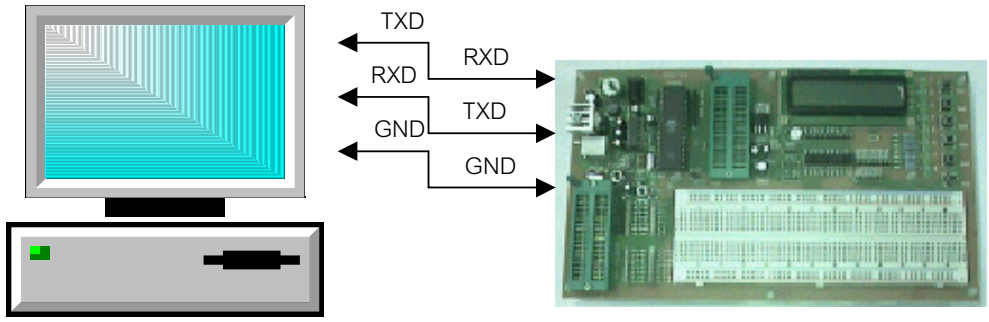
```
#include<reg51.h>
 unsigned char LED = {1, 2, 4, 8, 16, 32, 64, 128};
 void main (void)
```

```
    {
        TMOD = 0x01;
        TH0 = 0xF8;
        TL0 = 0xD8;
        IE = 0x82;
        TR0 = 1;
        Int x, y;
        While (1)
        {
        for (x = 0; x<8; x++)
        {
          P1 = LED [x];
          for (y-0; y<2000; y++);
        }
        }
    }
void overtime (void) interrupt 1
    {
        P2 = ~ P2;
        TH0 = 0xF8;
        TL0 = 0xD8;
        TF0 = 0;
    }
```

SE-ED

1.4.3 พอร์ตอนุกรม Serial Port

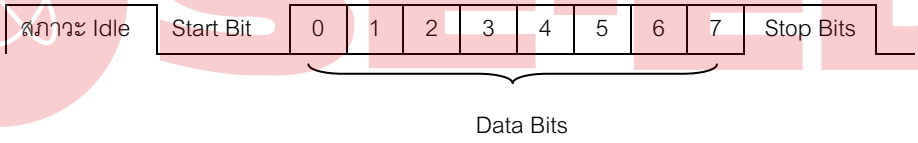
ในหน่วยนี้จะเรียนรู้เกี่ยวกับการทำงานพื้นฐานของพอร์ตอนุกรม (Serial Port) รวมทั้งการเชื่อมต่อ ระดับสัญญาณ ลักษณะของสัญญาณของพอร์ตอนุกรมตามมาตรฐาน RS-232 (Standard UART) จากนั้นจะมาศึกษาการทำงานพอร์ตอนุกรมของ MCS-51 และศึกษาเทคนิคการเขียนโปรแกรมควบคุมพอร์ตอนุกรม



ก) คอมพิวเตอร์ ข) แผงวงจร

รูปที่ 1.15 การเชื่อมต่อแบบ RS232

จากรูปที่ 1.15 แสดงการเชื่อมต่อสายสัญญาณระหว่างคอมพิวเตอร์กับบอร์ดไมโครคอนโทรลเลอร์โดย ลักษณะสัญญาณในการเชื่อมกันนั้นจะเป็นแบบ RS232 ซึ่งมีระดับสัญญาณลอจิก “1” ที่ระดับแรงดัน -3V ถึง -12V และลอจิก “0” ที่แรงดัน 3V ถึง 15 V



รูปที่ 1.16 สัญญาณ RS232

รูปข้างบนแสดงถึงลักษณะของสัญญาณที่ใช้ในการติดต่อ ผ่านมาตรฐาน RS-232 เพื่อให้อุปกรณ์ตัวหนึ่งกับอีกตัวหนึ่งสามารถสื่อสารกันได้ จึงต้องกำหนดมาตรฐานการสื่อสารขึ้นมา โดยลักษณะของสัญญาณนั้นจะประกอบด้วย 4 ส่วน คือ

Start Bit	เป็นบิทสำหรับการเริ่มต้นในการติดต่อสื่อสาร (logic 0 เสมอ)
Data Bits	เป็นบิทสำหรับเป็นข้อมูล มีทั้งหมด 8 บิท โดยเริ่มต้นด้วยบิทต่ำ
Stop Bits	เป็นบิทสำหรับการจบในการติดต่อสื่อสาร (logic 0 เสมอ)
Idle	เป็นสภาวะที่ขา RXD ของ MCS-51 รออยู่

ในแต่ละบิทจะมีการกำหนดความถี่ขึ้นมา เพื่อให้สามารถสื่อสารกันได้อย่างถูกต้อง โดยจะกำหนดให้ความกว้างของแต่ละบิทมีความถี่ค่าหนึ่ง ความถี่ค่านี้เรียกว่า Baud Rate หรืออัตราในการส่งข้อมูล เช่น Baud Rate 9600 หมายความว่า ใช้ความถี่ในการติดต่อสื่อสารที่ความถี่ 9600 Hz หรือสามารถส่งข้อมูลได้สูงสุด 9600 Bit/Sec นั่นเอง

■ การทำงานของวงจรพอร์ตอนุกรม

เริ่มต้นจากสร้างความถี่ในการสื่อสาร (Baud Rate) ก่อน โดย MCS-51 จะบังคับให้ใช้ Timer 1 เท่านั้น ซึ่งเราจะต้องกำหนดให้ Timer 1 ทำงานที่โหมด 2 ซึ่งคล้ายกับโหมด 1 ของ Timer 0 ในหน่วยที่ผ่านมา ซึ่งโหมด 1 เป็นแบบ 16 บิท แต่โหมด 2 จะทำงานแบบ 8 Bit Auto Roload หมายความว่า เมื่อสัญญาณนาฬิกาที่ได้จากวงจร Oscillator ถูกหารด้วย 12 แล้ว ป้อนเข้าที่รีจิสเตอร์ TL1 ค่าของ TL1 จะถูกเพิ่มค่าไปเรื่อยจนกลายเป็น 00H ค่าที่อยู่ในรีจิสเตอร์ TH1 ใหม่จะถูกใส่เข้าไปที่รีจีสเตอร์ TL1 ทันทีแล้ว TL1 ก็จะถูกเพิ่มไปเรื่อยๆ จนเป็น 00H ใหม่ ทำให้เอาต์พุทของ TL1 เป็นสัญญาณนาฬิกาค่าหนึ่ง

หลังจากที่ได้สัญญาณนาฬิกาจากวงจร Timer 1 แล้วสัญญาณจะถูกส่งไปยังวงจรหารความถี่ด้วย 32 กับ 16 โดยมีบิท SMOD ของรีจิสเตอร์ PCON เป็นตัวเลือกวงจรหาร หลังจากนั้นเราก็จะได้ความถี่ Baud Rate ป้อนให้กับวงจร Shift Registers ในการส่งข้อจาก Serial Port ผ่านทางขา TxD และ RxD ซึ่งจะมีความสัมพันธ์กับรีจิสเตอร์ SCON

■ วิธีการคำนวณหาค่า TH1 เพื่อสร้างความถี่ Baud Rate

1. กำหนดค่า Baud Rate ที่เราต้องการ : f
2. นำความถี่ของ X-Tal มาหาร 12 : F_{osc}
3. $F_{osc}/f = Ft$
4. Ft/32 = Co ; SMOD = 0 หรือ
   Ft/16 = Co ; SMOD = 1
5. 100H – Co = TH1

SE-ED

ตารางที่ 1.7 SCON (Serial Port Control Register) ตำแหน่งที่ 98h

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

บิท	ความหมาย
SM0	ใช้ในการสื่อสารแบบ Multi-Processor
SM1	บิทเลือก Mode ในการทำงาน
SM2	บิทเลือก Mode ในการทำงาน
REN	บิทเลือกว่าจะให้รับข้อมูลจาก Serial Port (1 : ให้รับได้)
TB8	ค่าของบิทที่ 9 ซึ่งจะส่งไปที่ขา TXD ใช้ในกรณีที่อยู่ใน Mode 2, 3
RB8	ค่าของบิทที่ 9 ซึ่งได้จากการรับข้อมูลจากขา RXD ใช้ในกรณีที่อยู่ใน Mode 2, 3
TI	จะเป็น “1” เมื่อทำการส่งข้อมูล 1 byte ออกไปที่ขา TXD (บิทนี้ต้องเคลียร์ที่ Software)
RI	จะเป็น “1” เมื่อได้รับข้อมูลจากขา RXD มาครบ 1 byte (บิทนี้ต้องเคลียร์ที่ Software)

SE-ED

■ ขั้นตอนการเขียนโปรแกรม Serial Port

1. กำหนดค่ารีจิสเตอร์ TMOD โดยให้ Timer 1 ทำงานที่โหมด 2 (8 Bit Auto Reload) เพราะฉะนั้น TMOD = 20H
2. กำหนดค่ารีจีสเตอร์ SCON โดยให้ทำงานที่โหมด 1 (8 Bit UART)
   เซ็ตบิท REN เพื่อให้สามารถรับข้อมูลได้
   เซ็ตบิท TI เพื่อให้สามารถส่งข้อมูลออกไปทาง Serial Port ได้

SMO	SM1	SM2	REN	TB8	RB8	TI	RI
0	1	0	1	0	0	1	0

SCON = 52H

3. กำหนดค่า TH1 ตามความถี่ Baud Rate ที่เรากำหนด (TH1 = FDH, 9600 Hz)
4. เปิดสัญญาณนาฬิกา ให้เข้าไปที่ TL1 โดยการเซ็ตบิท TR1
5. ในกรณีที่ต้องการให้ Interrupt จาก Serial Port ได้ ให้ทำการเซ็ตบิท EA =1 และ ES = 1 ของรีจิสเตอร์ IE

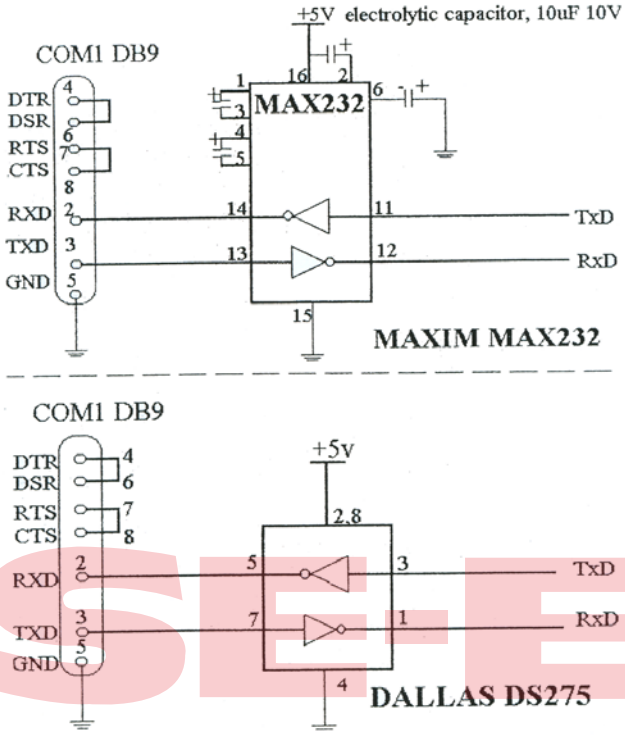
Note !	ถ้าต้องการใช้ Interrupt จาก Serial Port ได้ต้องเซ็ตบิท EA = 1, ES = 1 ในรีจิสเตอร์ IE

สำหรับ Turbo C ถ้าใช้คำสั่ง printf (“Message”); หมายถึง สั่งให้พิมพ์ข้อความ “Message” ลงบนจอภาพ ใน Keil Program ก็มีคำสั่งนี้เช่นกัน โดยใช้ร่วมกับการตั้ง Baud Rate เพื่อส่งข้อมูลออกทาง Serial Port เราสามารถเลือกความเร็วในการรับส่งข้อมูลได้จากการตั้ง Baud Rate เช่น ได้ค่ามาตรฐานในการรับส่งข้อมูล คือ 9600 bps และจะต้องตั้งค่าดังนี้

```
TMOD  = 0x20;
TH1   = 0xFD;
SCON  = 0x52;
TR    = 1;
```

ใน Turbo C เมื่อเราจะใช้คำสั่ง printf ( ) จะต้องประกาศที่หัวโปรแกรมด้วย # include <stdio.h> ใน Keil Program ก็เช่นกัน เพราะฟังก์ชันที่ใช้ติดต่อกับ Serial Port บรรจุอยู่ในไลบรารี stdio.h ในส่วนของวงจรจะใช้ไอซีสำเร็จรูปเบอร์ DS275 ที่แปลงระดับสัญญาณของ RS232 ให้เป็น TTL และจาก TTL ให้เป็นสัญญาณ RS 232 หรือจะใช้ MAX 232 ก็ได้

SE-ED



รูปที่ 1.17 IC MAX232, IC DS 275

ตัวอย่างที่ 1.6 การรับค่าทาง Serial Port

```
#include<reg51.h>
#include<stdio.h>
void main (void)
    {
        TMOD = 0x20;
        TH1 = 0xFD;
        SCON = 0x52;
        TR = 1;
        start: if(RI = =1)
        {
            RI = 0;
```

```
            P1 = SBUF;
        }
        goto start;
    }
```

ตัวอย่างที่ 1.7 Serial interrupt

IE (Interrupt Enable Register)

EA	-	ET2	ES	ET1	EX1	ET0	EX0
1	0	0	1	0	0	0	0

IE = 0x90

```
#include<reg51.h>
void main (void)
    {
        TMOD = 0x20;
        TH1 = 0xFD;
        SCON = 0x52;
        IE = 0x90;
        TR1 = 1;
        start:
        P0 = 0x55;
        delay (20000);
        P0 = 0xAA;
        delay (20000);
        goto start;

void serial_int (void) interrupt 4
      {
        if (TI = = 1)
        {
            TI = 0;
```

SE-ED

```
            SBUF = 'H';
        }
        if (RI = =1)
        {
            RI = 0;
            P1 = SBUF;
        }
    }
```

ตัวอย่างที่ 1.8 การส่งข้อมูลออกทาง Serial Port

```
#include<reg51.h>
#include<stdio.h>
void main (void)
    {
        init_serial ( );
        printf ("MCS-51 Serial Communication…\ n");
        printf ("Baud Rate 9600bps…\ n");
        while (1);              // stop
    }

void init_serial (void)
    {
        TMOD = 0x20;
        TH1 = 0xFD;         // Baud Rate = 9600
        SCON = 0x52;
        RT = 1;
    }
```

SE-ED

คัมภีร์การใช้งาน ไมโครคอนโทรลเลอร์ MICROCONTROLLER MCS-51

หนังสือเล่มนี้ มีจุดมุ่งหมายที่จะพัฒนาระบบควบคุมอัตโนมัติ (Automation Control System) ตลอดจนหุ่นยนต์อุตสาหกรรม (Industrial Robot) ให้มีการแข่งขันและพัฒนาให้ทันกับโลกยุคปัจจุบัน ที่มีการแข่งขันทางด้านการตลาดและระบบเศรษฐกิจที่สูงขึ้น โดยจะเริ่มต้นอธิบายเกี่ยวกับไมโครคอนโทรลเลอร์ (MCS-51) ภาษาซี การออกแบบควบคุมการทำงานของ Input/Output port การควบคุม Stepping Motor การควบคุม DC Motor การควบคุม DC Servo Motor และจบด้วยการควบคุมหุ่นยนต์อุตสาหกรรม

หนังสือเล่มนี้เหมาะสำหรับนักศึกษาได้ใช้เรียนและค้นคว้าเพิ่มเติม รวมทั้งช่างไฟฟ้า ช่างอิเล็กทรอนิกส์ วิศวกรโรงงาน ตลอดจนผู้สนใจในงานควบคุมด้วยไมโครคอนโทรลเลอร์

พศ.ดร. เดชฤทธิ์ มณีธรรม

ปัจจุบัน

- หัวหน้าสาขาวิศวกรรมเมคคาทรอนิกส์ คณะครุศาสตร์อุตสาหกรรม มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี

การศึกษา

- ปริญญาเอก D. Eng. in Mechatronics Asian Institute of Technology
- ปริญญาโท สาขาเครื่องกล คณะครุศาสตร์อุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ
- ปริญญาตรี สาขาเทคโนโลยีขนถ่ายวัสดุ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ

ประสบการณ์และการทำงาน

- ศึกษาดูงานระบบ Industrial Robotics ที่สิงคโปร์
- ศึกษาดูงานระบบ Automation ที่อิตาลี
- ศึกษาดูงานระบบ Mechatronics ที่เกาหลีใต้
- ศึกษาดูงานระบบ Robotics ที่สาธารณรัฐประชาชนจีน
- ศึกษาดูงานระบบ Process Control ที่อินเดีย
- ศึกษาดูงานระบบ Control System ที่สวิตเซอร์แลนด์
- ศึกษาดูงานระบบ Process Control ที่อินเดีย
- ศึกษาดูงานระบบ Industrial Robotics ที่สหราชอาณาจักร
- ศึกษาดูงานระบบ Industrial Robotics ที่ออสเตรเลีย
- ศึกษาดูงานระบบ TwinCAT and PC Control ที่เยอรมนี
- เป็นวิศวกรโรงงานควบคุมระบบอัตโนมัติมากกว่า 10 ปี
- เป็นวิทยากรอบรมครูอาจารย์สังกัดกรมอาชีวศึกษา
- อาจารย์พิเศษโรงเรียนนายร้อยพระจุลจอมเกล้า
- เป็นวิทยากรสัมมนาให้กับบริษัทเอกชน และเป็นที่ปรึกษาระบบควบคุมอัตโนมัติให้กับบริษัทหลายแห่ง
- เป็นวิทยากรอบรมอาจารย์มหาวิทยาลัยหลายแห่ง ในหัวข้อฝึกอบรม
  - Pneumatic and Electro Pneumatic
  - Hydraulic and Electro Hydraulic
  - Programmable Logic Controller (PLC, SCADA and OPC)
  - Automation Control System
  - Microcontroller (MCS-51, PIC)

ผลงานที่เขียน

- คัมภีร์การใช้งาน ระบบนิวแมติกส์ (Pneumatics System)
- คัมภีร์การใช้งาน ไมโครคอนโทรลเลอร์ MCS-51
- คัมภีร์การใช้งาน ไมโครคอนโทรลเลอร์ PIC
- คัมภีร์การใช้งาน หุ่นยนต์ (ROBOT)
- คัมภีร์การใช้งาน ระบบไฮดรอลิกส์ (Hydraulics System)
- คัมภีร์การใช้งาน พีแอลซีเบคฮอฟฟ์ (PLC BECKHOFF)

ISBN 978-616-08-2595-0

9 786160 825950

250 บาท

วิทยาการและเทคโนโลยี/การควบคุมอัตโนมัติ

www.se-ed.com

sbc.fans

SE-ED