บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 ได้จัดให้มีส่วนประกอบภายใน เพื่ออำนวยความสะดวกแก่ผู้ใช้ เช่น ไทม์เมอร์/เคาเตอร์ พอร์ตอนุกรม (Serial Port) และสำหรับไมโครคอนโทรลเลอร์บางตัว อาจมีส่วนอื่นเพิ่มเติมเข้ามาอีก เช่นเบอร์ AT80C515, AT80C535 มีวงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล (Analog to Digital Converter)

2.1.1 คุณสมบัติทั่วไปของไมโครคอนโทรลเลอร์ตระกูล MCS-51

เนื่องจากคุณสมบัติของไมโครคอนโทรลเลอร์แต่ละเบอร์นั้นจะมีความแตกต่างกันในรายละเอียด ในที่นี้จะขออ้างถึงเบอร์ AT89S52 ของบริษัท Atmel ซึ่งเป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต มีคุณสมบัติดังนี้

-มีหน่วยความจำแบบแฟลช (Flash Memory) ขนาด 8 กิโลไบท์

-โปรแกรมผ่านพอร์ตอนุกรมมาตรฐาน SPI (SPI Serial Interface)

-สามารถโปรแกรมและลบซ้ำได้นับ 1000 ครั้ง

-มีหน่วยความจำแบบ EEPROM ขนาด 2 กิโลไบต์

-สามารถโปรแกรมและลบซ้ำได้นับ 1000 ครั้ง

-ใช้แหล่งจ่ายไฟกระแสตรงขนาด 5 โวลต์ (ทำงานในช่วง 4-6 โวลต์)

-ทำงานด้วยสัญญาณนาฬิกาตั้งแต่ 0-24 MHz

-สามารถป้องกันหน่วยความจำได้ 3 ระดับ

-มีหน่วยความจำข้อมูล (RAM) ขนาด 256 ไบต์

-มี 32 บิตอิสระสามารถเข้าถึงระดับบิตได้

-มีไทเมอร์/ เคาเตอร์ขนาด 16-bit ทั้งหมด 2 ตัว

-รองรับอินเตอร์รัปต์ได้ 8 แหล่ง

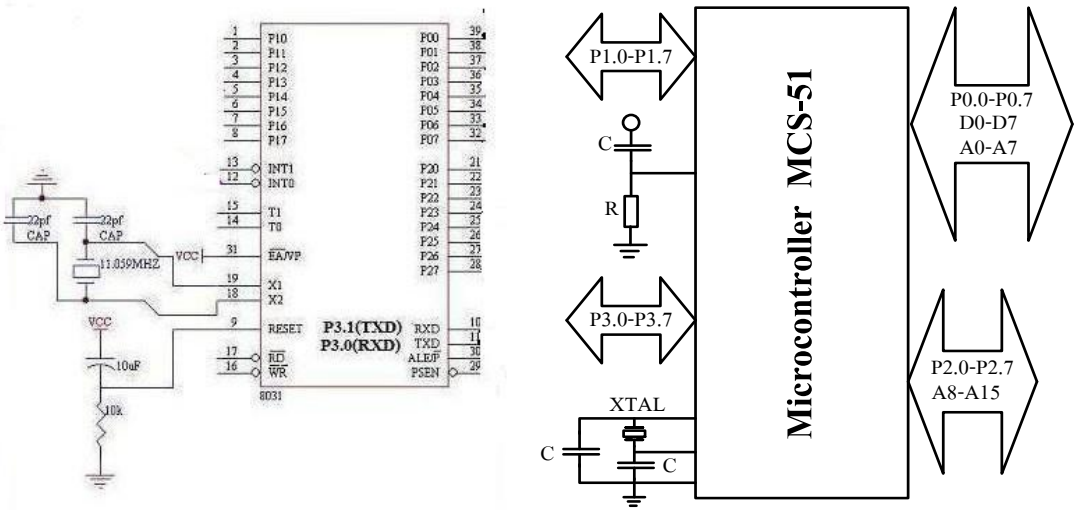
-สามารถสื่อสารข้อมูลแบบอนุกรมได้ด้วย UART Channel

-มีโหมดการทำงานแบบ Low Power Idle และ Power Down เพื่อประหยัดพลังงาน

-มี Watchdog Timer เพื่อให้การทำงานของระบบสามารถ Reset ได้อัตโนมัติ

2.1.2 สถาปัตยกรรมของไมโครคอนโทรลเลอร์ตระกูล MCS-51 (เบอร์ AT89S52)

การจัดเรียงขาของ MCS-51(เบอร์ AT89S52) เป็นไปตามรูปที่ 2.1



รูปที่ 2.1 การจัดขาของ MCS-51(เบอร์ AT89S52)

VCC ต่อไฟเลี้ยง (Supply Voltage 5 V)

GND ต่อกราวด์ (Ground)

Port 0 (P0.0-P0.7) เป็นพอร์ต 2 ทิศทางขนาด 8 บิตสามารถทำงานได้ 2 หน้าที่คือเป็นพอร์ตอินพุตเอาต์พุตทั่วไป และใช้เป็นพอร์ตสำหรับติดต่อกับหน่วยความจำภายนอกคือรับส่งข้อมูล และกำหนดแอดเดรสไบต์ต่ำ (A0-A7)

Port 1 (P1.0-P1.7) เป็นพอร์ต 2 ทิศทางขนาด 8 บิต มีการต่อความต้านทาน (Pull-up Resistant) ไว้ภายใน ทำหน้าที่เป็นพอร์ตอินพุตเอาต์พุตทั่วไป นอกจากนี้ยังใช้งานเป็นขาอินพุตเอาต์พุตของไทเมอร์ 2

Port 2 (P2.0-P2.7) เป็นพอร์ต 2 ทิศทางขนาด 8 บิต มีการต่อความต้านทาน (Pull-up Resistant) ไว้ภายใน สามารถทำงานได้ 2 หน้าที่คือเป็นพอร์ตอินพุตเอาต์พุตทั่วไป และใช้เป็นพอร์ตสำหรับติดต่อกับหน่วยความจำภายนอกคือรับส่งข้อมูล และกำหนด แอดเดรสไบต์สูง (A8-A15)

Port 3 (P3.0-P3.7) เป็นพอร์ต 2 ทิศทางขนาด 8 บิตและ มีการต่อความต้านทาน (Pull-up Resistant) ไว้ภายใน ทำหน้าที่คือเป็นพอร์ตอินพุตเอาต์พุตทั่วไป ยังใช้งานเป็นสัญญาณควบคุมการติดต่อกับหน่วยความจำ การอินเตอร์รัปต์ และอื่นๆ

RST เป็นขาอินพุตที่ใช้รับสัญญาณสำหรับรีเซ็ตซีพียู ซีพียูจะถูกรีเซ็ตเมื่อขานี้เป็นลอจิก “1” นาน 2 แมชินไซเคิล หรือ 24 ไซเคิลของสัญญาณนาฬิกา

ALE/PROG ทำหน้าที่เป็นขาเอาต์พุตเมื่อซีพียูต้องการติดต่อกับหน่วยความจำภายนอกจะทำการส่งสัญญาณพัลส์ออกมาที่ขานี้เพื่อทำการแอดเดรสไบต์ต่ำของหน่วยความจำภายนอก และขานี้จะเป็นอินพุตเมื่ออยู่ในระหว่างโปรแกรมแฟลช

PSEN เป็นขาเอาต์พุตใช้ในการติดต่อกับหน่วยความจำโปรแกรมภายนอกคือเมื่อซีพียูทำการประมวลผลกับหน่วยความจำภายนอกขานี้จะแอคทีฟสองครั้ง

EA/VPP เป็นขาเอาต์พุตและต้องการลอจิก “0” เพื่อยอมให้ซีพียูสามารถเข้าถึงหน่วยความจำภายนอกได้ซึ่งอยู่ที่ตำแหน่ง 0000H ถึง FFFFH นอกจากนี้แล้วขานี้ยังใช้รับไฟ 12 โวลต์ เพื่อใช้ในระหว่างที่ทำการโปรแกรมแฟลช

XTAL1 เป็นขาอินพุตของวงจรออสซิลเลเตอร์แอมปลิไฟเออร์ และยังเป็นอินพุตของวงจรกำเนิดสัญญาณนาฬิกาภายใน

XTAL2 เป็นขาเอาต์พุตของวงจรออสซิลเลเตอร์แอมปลิไฟเออร์

2.1.3 สัญลักษณ์และคำสั่งของ MCS-51 เป็นไปตามตารางที่ 2.1

ตารางที่ 2.1 สัญลักษณ์และคำสั่งของ MCS-51

สัญลักษณ์	**ความหมาย**
Rn	หมายถึง รีจิสเตอร์ใช้งานทั่วไป R0-R7 ที่ถูกเลือกใช้ในขณะนั้น
@Ri	หมายถึง รีจิสเตอร์ใช้งานทั่วไป R0 หรือ R1 ที่เข้าถึงข้อมูลได้โดยอ้อม
direct	หมายถึง ข้อมูลขนาด 8 บิต ที่ใช้กำหนดค่าตำแหน่งหน่วยความจำสำหรับเก็บข้อมูลภายใน ประกอบด้วยหน่วยความจำเก็บข้อมูลทั่วไป ตำแหน่ง 0-127 และ หน่วยความจำ เก็บข้อมูลที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ ตำแหน่ง 128-255
bit	หมายถึง ค่าตำแหน่งของบิตข้อมูลในหน่วยความจำที่เข้าถึงในระดับบิต
#data	หมายถึง ข้อมูลขนาด 8 บิตที่กำหนดในคำสั่ง
#data16	หมายถึง ข้อมูลขนาด 16 บิตที่กำหนดในคำสั่ง
rel	หมายถึง ค่าตำแหน่งหน่วยความจำขนาด 8 บิต ที่คิดแบบมีเครื่องหมายในคำสั่ง SJMP และการกระโดดแบบมีเงื่อนไขทุกคำสั่ง
addr11	หมายถึง ค่าตำแหน่งหน่วยความจำขนาด 11 บิต ใช้เป็นค่าตำแหน่งหน่วยความจำปลายทาง (ภายในขอบเขต 2048 ตำแหน่ง) ในคำสั่ง AJMP และ ACALL

ตารางที่ 2.1 สัญลักษณ์และคำสั่งของ MCS-51 (ต่อ)

คำสั่ง	ความหมาย
addr11	หมายถึง ค่าตำแหน่งหน่วยความจำขนาด 11 บิต ใช้เป็นค่าตำแหน่งหน่วยความจำปลายทาง (ภายในขอบเขต 2048 ตำแหน่ง) ในคำสั่ง AJMP และ ACALL
addr16	หมายถึง ค่าตำแหน่งหน่วยความจำขนาด 16 บิต ใช้เป็นค่าตำแหน่งหน่วยความจำปลายทาง (ภายในขอบเขต 65536 ตำแหน่ง) ในคำสั่ง LJMP และ LCALL
MOV A,Rn	ย้ายข้อมูลจาก Rn ไป A
MOV A,direct	ย้ายข้อมูลจากหน่วยความจำ direct ไป A
MOV A,@Ri	ย้ายข้อมูลจากหน่วยความจำที่เก็บอยู่ในตำแหน่ง Ri ไป A
MOV A,#data	ย้ายค่าคงที่ 8 บิตไปเก็บที่ A
MOV Rn,A	ย้ายข้อมูลจาก A ไป Rn
MOV Rn,direct	ย้ายข้อมูลจากหน่วยความจำ direct ไป Rn
MOV direct,A	ย้ายข้อมูลจาก A ไปยังหน่วยความจำ direct
MOV direct,Rn	ย้ายข้อมูลจาก Rn ไปยังหน่วยความจำ direct
MOV direct,direct	ย้ายข้อมูลระหว่างหน่วยความจำภายใน
MOV direct,@Ri	ย้ายข้อมูลจากหน่วยความจำที่เก็บอยู่ในตำแหน่ง Ri ไปยังหน่วยความจำ direct
MOV direct,#data	ย้ายค่าคงที่ 8 บิต ไปยังหน่วยความจำ direct
MOV @Ri,A	ย้ายข้อมูลใน A ไปยังหน่วยความจำ Ri
MOV @Ri,direct	ย้ายข้อมูลจากหน่วยความจำ direct ไปยังหน่วยความจำ Ri
MOV @Ri,#data	ย้ายค่าคงที่ 8 บิต ไปยังหน่วยความจำ Ri
MOV DPTR,#data16	ย้ายค่าคงที่ 16 บิต ไปยัง DPTR
MOVC A,@A+DPTR	ย้ายข้อมูลจากหน่วยความจำข้อมูลที่สัมพัทธ์กับ DPTR ไปยัง A
MOVC A,@A+PC	ย้ายข้อมูลจากหน่วยความจำข้อมูลที่สัมพัทธ์กับ PC ไปยัง A
MOVX A,@Ri	ย้ายข้อมูลจากหน่วยอินพุตที่เก็บอยู่ในตำแหน่ง Ri ไปยัง A
MOVX A,@DPTR	ย้ายข้อมูลจากหน่วยความจำที่เก็บอยู่ในตำแหน่ง DPTR ไปยัง A
MOVX @Ri,A	ย้ายข้อมูลที่เก็บอยู่ใน A ไปยังหน่วยเอาต์พุตตำแหน่ง Ri

ตารางที่ 2.1 สัญลักษณ์และคำสั่งของ MCS-51 (ต่อ)

สัญลักษณ์	ความหมาย
MOVX @DPTR,A	ย้ายข้อมูลที่เก็บอยู่ใน A ไปยังหน่วยความจำตำแหน่ง DPTR
MOV C,bit	ย้ายค่า bit ไปยัง Carry Flag
MOV bit,C	ย้ายค่าแฟล็ก Carry ไปยัง bit
JNC rel	กระโดด ถ้าค่าแฟล็ก Carry เป็น 0
JC rel	กระโดด ถ้าค่าแฟล็ก Carry เป็น 1
JB bit,rel	กระโดด ถ้าค่า bit เป็น 1
JNB bit,rel	กระโดด ถ้าค่า bit เป็น 0
JBC bit,rel	กระโดด ถ้าค่า bit เป็น 1 และเปลี่ยนค่า bit เป็น 0
JZ rel	กระโดดไปยังตำแหน่งที่สัมพัทธ์กับตำแหน่งปัจจุบัน ถ้าหากค่า A เป็นค่า 00H
JNZ rel	กระโดดไปยังตำแหน่งที่สัมพัทธ์กับตำแหน่งปัจจุบัน ถ้าหากค่า A ไม่เป็นค่า 00H
CJNE A,direct,rel	เปรียบเทียบค่า A กับหน่วยความจำ direct และกระโดดไปยังตำแหน่งที่สัมพัทธ์กับตำแหน่งปัจจุบัน ถ้าค่าไม่เท่ากัน
CJNE A,#data,rel	เปรียบเทียบค่า A กับค่าคงที่ และกระโดดไปยังตำแหน่งที่สัมพัทธ์กับตำแหน่งปัจจุบัน ถ้าค่าไม่เท่ากัน
CJNE Rn,#data,rel	เปรียบเทียบค่า Rn กับค่าคงที่ และกระโดดไปยังตำแหน่งที่สัมพัทธ์กับตำแหน่งปัจจุบัน ถ้าค่าไม่เท่ากัน
CJNE @Ri,#data,rel	เปรียบเทียบค่าใน Ri กับค่าคงที่ และกระโดดไปยังตำแหน่งที่สัมพัทธ์กับตำแหน่งปัจจุบัน ถ้าค่าไม่เท่ากัน
DJNZ Rn,rel	ลดค่าใน Rn และกระโดดไปยังตำแหน่งที่สัมพัทธ์กับตำแหน่งปัจจุบัน ถ้าค่าไม่เป็น 0
DJNZ direct,rel	ลดค่าในหน่วยความจำ direct และกระโดดไปยังตำแหน่งสัมพัทธ์กับตำแหน่งปัจจุบัน ถ้าค่าไม่เป็น 0
SJMP rel	กระโดดไปยังตำแหน่งสัมพัทธ์กับตำแหน่งปัจจุบัน
AJMP addr11	กระโดดไปยังตำแหน่งจากค่าแอดเดรส 11 บิต
LJMP addr16	กระโดดไปยังตำแหน่งจากค่าแอดเดรส 16 บิต

ตารางที่ 2.1 สัญลักษณ์และคำสั่งของ MCS-51 (ต่อ)

คำสั่ง	ความหมาย
AJMP addr11	กระโดดไปยังตำแหน่งจากค่าแอดเดรส 11 บิต
LJMP addr16	กระโดดไปยังตำแหน่งจากค่าแอดเดรส 16 บิต
JMP @A+DPTR	กระโดดไปยังตำแหน่งที่สัมพัทธ์กับ A+DPTR
ACALL addr11	ไปทำโปรแกรมย่อยจากค่าแอดเดรส 11 บิต
LCALL addr16	ไปทำโปรแกรมย่อยจากค่าแอดเดรส 16 บิต
RET	สิ้นสุดการทำโปรแกรมย่อย แล้วกลับไปยังตำแหน่งถัดไปที่กระโดดมา
RETI	สิ้นสุดการทำโปรแกรมย่อย อินเตอร์รัปท์
INC A	เพิ่มค่าใน A
INC Rn	เพิ่มค่าใน Rn
INC direct	เพิ่มค่าในหน่วยความจำ Direct
INC @Ri	เพิ่มค่าในหน่วยความจำที่ตำแหน่ง Ri
INC DPTR	เพิ่มค่าใน DPTR
DEC A	ลดค่าใน A
DEC Rn	ลดค่าใน Rn
DEC direct	ลดค่าในหน่วยความจำ Direct
DEC @Ri	ลดค่าในหน่วยความจำที่เก็บอยู่ใน Ri
DEC DPTR	ลดค่าใน DPTR
MUL AB	คูณ A กับ B แล้วเก็บค่าใน BA
DIV AB	หาร A กับ B แล้วเก็บค่าใน A เก็บเศษใน B
DA A	ทำ Decimal Adjust ค่าใน A
ADD A,Rn	บวกค่า Rn กับ A
ADD A,direct	บวกค่าในหน่วยความจำ Direct กับ A เก็บผลลัพธ์ใน A
ADD A,@Ri	บวกค่าในหน่วยความจำตำแหน่ง Ri กับ A เก็บผลลัพธ์ใน A
ADD A,#data	บวกค่าคงที่ 8 บิต กับ A เก็บผลลัพธ์ใน A
ADDC A,Rn	บวกค่า Rn กับ A พร้อมแฟล็ก Carry เก็บผลลัพธ์ใน A
ADDC A,direct	บวกค่าในหน่วยความจำ Direct กับ A พร้อมแฟล็ก Carry เก็บผลลัพธ์ใน A

ตารางที่ 2.1 สัญลักษณ์และคำสั่งของ MCS-51 (ต่อ)

คำสั่ง	ความหมาย
ADDC A,@Ri	บวกค่าในหน่วยความจำตำแหน่ง Ri กับ A พร้อมแฟล็ก Carry เก็บผลลัพธ์ใน A
ADDC A,#data	บวกค่าคงที่ 8 บิต กับ A พร้อมแฟล็ก Carry เก็บผลลัพธ์ใน A
SUBB A,Rn	ลบค่า Rn กับ A พร้อมแฟล็ก Borrow เก็บผลลัพธ์ใน A
SUBB A,direct	ลบค่าในหน่วยความจำ Direct กับ A พร้อมแฟล็ก Borrow เก็บผลลัพธ์ใน A
SUBB A,@Ri	ลบค่าในหน่วยความจำที่เก็บอยู่ใน Ri กับ A พร้อมแฟล็ก Borrow เก็บผลลัพธ์ใน A
SUBB A,#data	ลบค่าคงที่ 8 บิต กับ A พร้อมแฟล็ก Borrow เก็บผลลัพธ์ใน A
CLR A	ทำค่าใน A ให้เป็นศูนย์
CPL A	กลับค่าบิตใน A เป็นตรงข้ามทุกบิต
RL A	หมุนบิตใน A ไปทางซ้าย 1 บิตและบิต 0 เป็นค่าจากบิต 7
RLC A	หมุนบิตใน A ไปทางซ้าย 1 บิตและค่าจากบิต 7 นำไปเก็บในแฟล็ก Carry และบิตที่อยู่ในแฟล็ก Carry ไปเป็นบิตที่ 0
RR A	หมุนบิตใน A ไปทางขวา 1 บิตและบิต 7 เป็นค่าจากบิต 0
RRC A	หมุนบิตใน A ไปทางขวา 1 บิตและค่าจากบิต 0 นำไปเก็บในแฟล็ก Carry และบิตที่อยู่ในแฟล็ก Carry ไปเป็นบิตที่ 7
SWAP A	สลับค่าสี่บิตซ้ายกับสี่บิตขวาภายใน A
PUSH direct	ย้ายข้อมูลหน่วยความจำ direct ไปเก็บยัง Stack
POP direct	ย้ายข้อมูลจาก Stack ไปยังหน่วยความจำ direct
CLR C	ทำค่าแฟล็ก Carry ให้เป็น 0
CLR bit	ทำค่า bit ให้เป็น 0
SETB C	ทำค่าแฟล็ก Carry ให้เป็น 1
SETB bit	ทำค่า bit ให้เป็น 1
CPL C	กลับค่าแฟล็ก Carry ให้เป็นตรงข้าม
CPL bit	กลับค่า bit ให้เป็นตรงข้าม
ORL C,bit	OR ค่า bit กับแฟล็ก Carry เก็บผลลัพธ์ใน C
ORL C,/bit	OR ค่า not bit กับแฟล็ก Carry เก็บผลลัพธ์ใน C
ANL C,bit	AND ค่า bit กับแฟล็ก Carry เก็บผลลัพธ์ใน C
ANL C,/bit	AND ค่า not bit กับแฟล็ก Carry เก็บผลลัพธ์ใน C

2.2 ไตรแอก (TRIAC)

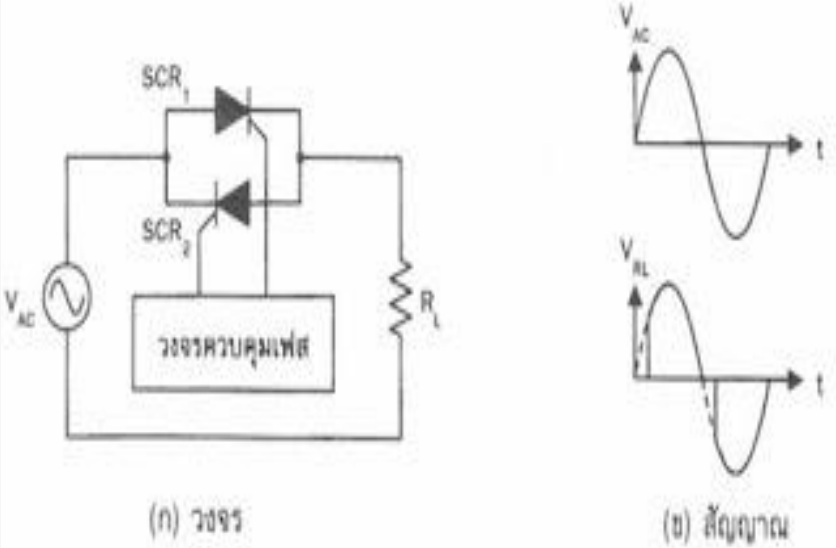
ไตรแอกถูกพัฒนาขึ้นมาใช้งานได้กับแรงดันไฟสับ เพื่อแก้ไขข้อบกพร่องของ SCR ไตรแอกนำกระแสได้สองทิศทาง ทั้งช่วงบวกและช่วงลบของแรงดันไฟฟ้าสลับ โดยทำหน้าทีเป็นสวิตช์ปิด-เปิดแรงดันไฟสลับ ไ

ตรแอกถูกสร้างขึ้นมาให้ใช้งานได้กับประแสสูง ๆ ดังนั้นเวลาทำงานมักเกิดความร้อนขึ้น จึงต้องยึดติดแผ่นระบายความร้อน สวิตช์ไตรแอกดีกว่าสวิตช์ธรรมดาหลายประการ คือ ทำงานได้เร็ว ควบคุมให้ทำงานได้ง่าย และไม่มีหน้าสัมผัสจึงไม่เกิดประกายไฟขณะทำงาน ดัดแปลงไปใช้งานได้กว้างขวางโครงสร้างของไตรแอกเสมือนการรวม SCR สองตัวไว้ด้วยกัน ต่อขนานหันหัวกลับทางกัน ขาเกตของ SCR ทั้งสองต่อร่วมกัน วงจรเทียบเท่าของไตรแอกเหมือนกับวงจรเทียบเท่าของ SCR สองตัวต่อรวมกัน คุณสมบัติในการทำงานของไตรแอกเหมือนกับการทำงานของ SCR แต่สามารถทำงานแรงดันไฟสลับทั้ง 2 ช่วง การนำกระแสของตัวไตรแอก ขึ้นอยู่กับการควบคุมแรงดันกระตุ้นที่ขา G โดยจัดขั้วแรงดันกระตุ้นขา G ให้เหมาะสมถูกต้องกับแรงดันไบอัสที่ขา A1 , A2ไตรแอกจะสามารถนำกระแสได้ดี

การนำไตรแอกไปใช้งาน จำเป็นต้องเลือกสภาวะการทำงานของไตรแอก โดยเลือกสภาวะที่ไตรแอกทำงานได้ดี คือ เลือกสภาวะที่กระแสแอโนด (IA) กับแระแสเกต (IG) เสริมกัน สังเกตได้จากการจ่ายแรงดันให้ขา A2 และขา G ต้องมีขั้วแรงดันเหมือนกัน การทำให้ไตรแอกที่นำกระแสและหยุดนำกระแส ทำได้ 2 วิธี คือ ตัดแหล่งจ่ายแรงดันที่จ่ายให้ขา A2 และขา A1 ออกชั่วขณะและลดกระแสที่ไหลผ่านตัวไตรแอกให้ต่ำกว่าค่ากระแสโฮลดิ้ง (IH)

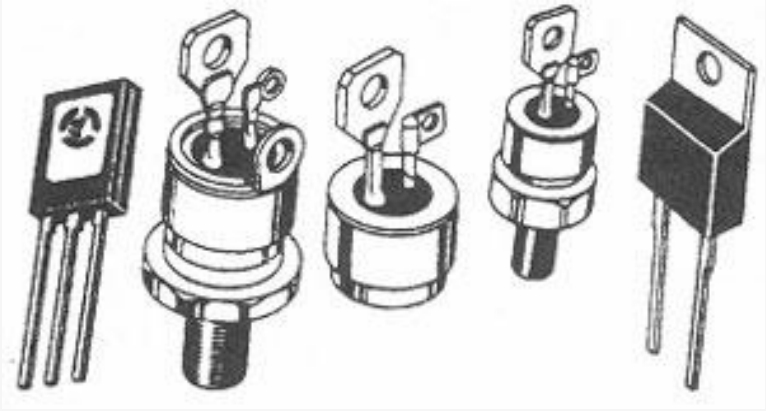
กราฟคุณสมบัติของไตรแอก เป็นกราฟบอกถึงสภาวะการทำงานของไตรแอกว่านำกระแสหรือไม่ ขณะนำกระแสไตรแอกสามารถทำงานได้หรือไม่ มีกระแสไหลผ่านตัวไตรแอกมากน้อยเพียงไร จุดทำงานที่กำหนดขึ้นนี้เหมาะสมกับการใช้งานหรือไม่ กราฟคุณสมบัติสามารถช่วยให้เลือกจุดทำงานได้เหมาะสมอุปกรณ์ใช้งานกับแรงดันไฟสลับ

SCR ถึงแม้สามารถนำไปใช้งานได้กับแรงดันไสลับ แต่ SCR จะนำกระแสได้เฉพาะช่วงครึ่งไซเกิลบวกของแรงดันไฟสลับเท่านั้น ส่วนครึ่งไซเกิลลบ SCR ไม่นำกระแส ทำให้แรงดันตกคร่อมสภาวะลดลงไปครึ่งหนึ่ง กำลังไฟฟ้าของสภาวะลดลงทำงานได้ไม่เต็มที่ หากต้องการให้สภาวะได้รับแรงดันไฟสลับช่วงไซเกิลลบด้วย ต้องจัดวงจรทำงานของ SCR ใหม่ โดยต่อวงจร SCR แบบขนาน 2 ตัว หันหัวกลับทางกัน จึงสามารถใช้ SCR ควบคุมแรงันไฟสลับได้ทั้งช่วงบวกและช่วงลบ ลักษณะของวงจรเบื้องต้นของ SCR ที่สามรถทำงานได้ทั้ง 2 ช่วงของแรงดันไฟกระแสสลับ แสดงการทำงานดังรูปที่ 2.2



รูปที่ 2.2 วงจรเบื้องต้นของ SCR ทำงานได้ 2 ช่วงของแรงดันไฟสลับ

ไตรแอก (Triac) เป็นอุปกรณ์จำพวกสารกึ่งตัวนำที่จัดอยู่ในสารกึ่งตัวนำประเภทไธริสเตอร์ เช่นเดียวกับ SCR ถูกพัฒนาขึ้นมาใช้งานได้กับแรงดันไฟสลับ เพื่อแก้ข้อบกพร่องของ SCR ไตรแอกสามารถนำกระแสได้ 2 ทิศทาง ทั้งช่วงบวกและช่วงลบของแรงดันไฟสลับ ดังนั้นวงจรใช้งานของไตรแอกจึงมักถูกนำไปใช้กับแรงดันไฟสลับ โดยตัวไตรแอกทำหน้าที่เป็นสวิตช์ปิด-เปิดแรงดันไฟสลับให้ผ่านไปยังสภาวะต่าง ๆ ไตรแอกถูกสร้างขึ้นมาให้ใช้งานได้กับกระแสสูง ๆ ทำให้ขณะไตรแอกทำงานจะเกิดความร้อนขึ้นสูง จึงต้องระมัดระวังในเรื่องการระบายความร้อนทำให้รูปร่างของไตรแอกที่สร้างขึ้นมา มีส่วนติดยึดกับแผ่นระบายความร้อนด้วยเสมอ ลักษณะและรูปร่างของไตรแอก แสดงดังรูปที่ 2.3



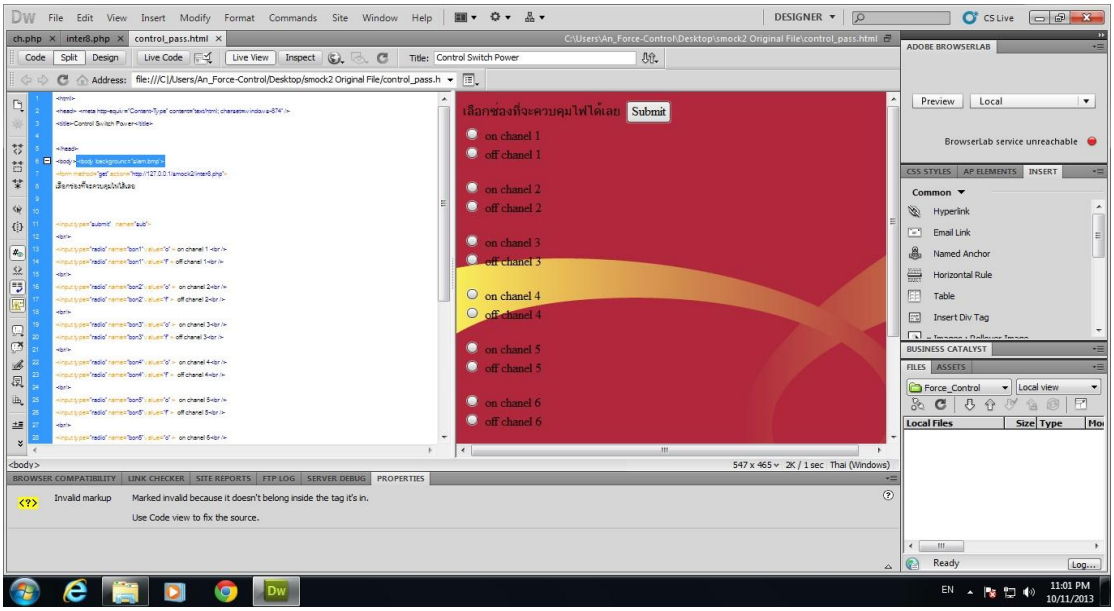
รูปที่ 2.3 ลักษณะของไตรแอกชนิดต่าง ๆ

ไตรแอกทำงานเป็นสวิตช์ทางอิเล็กทรอนิกส์ต่อแรงดันไฟสลับ มีข้อดีกว่าสวิตช์แบบกลไกหลายประการดังนี้

- การเปิด-ปิดของไตรแกเร็วกว่าสวิตช์กลไกหลายเท่า ทำให้การควบคุมให้สวิตช์ทำงานได้รวดเร็วขึ้น
- การควบคุมให้ไตรแอกทำงานในการปิด-เปิด วงจรไฟทำได้ง่าย โดยป้อนแรงดันไฟสลับค่าต่ำ ๆ เพียงเล็กน้อย ไปกระตุ้นขาเกต
- การปิด-เปิดวงจรไฟฟ้า ไม่มีการสัมผัสของหน้าสัมผัสเหมือนสวิตช์กลไกธรรมดาจึงไม่เกิดประกายไฟที่อาจทำให้เกิดเพลิงไหม้ได้ มีความปลอดภัยในการทำงานมากขึ้น
- สามารถดัดแปลงไปใช้งานกับวงจรต่าง ๆ ได้อย่างกว้างขวางมากมายไตรแอก (TRIAC)

2.3 Adobe Dreamweaver Software

Adobe Dreamweaver เป็นโปรแกรมสำหรับพัฒนาเว็บไซต์ ซึ่งมีคุณสมบัติครอบคลุมตั้งแต่การออกแบบและสร้างเว็บและสร้างเว็บเพจ การบริหารจัดการเว็บไซต์ ตลอดไปจนถึงการพัฒนาเว็บแอพพลิเคชั่นเบื้องต้น โปรแกรมนี้ได้รับความนิยมเป็นอย่างมากเพราะมีคุณสมบัติเด่นคือใช้งานง่าย มีเครื่องมือสำหรับวางข้อความ ภาพกราฟฟิก ตาราง แบบฟอร์ม มัลติมิเดีย รวมทั้งองค์ประกอบต่าง ๆ เพื่อโต้ตอบกับผู้ชมลงบนเว็บเพจได้ง่าย โดยผู้ใช้ไม่จำเป็นต้องรู้จักภาษา HTML,CSS, JavaScript และภาษาสคริปต์อื่น ๆ ดังนั้นจึงเหมาะสาหรับผู้เริ่มต้นและผู้ใช้ทั่วไป นอกจากนี้ยังมีคุณสมบัติขั้นสูงอีกมากมายสำหรับนักพัฒนาเว็บไซต์มือ โดยมีตัวอย่างการออกแบบ Website ตามรูปที่ 2.4



รูปที่ 2.4 การออกแบบ Website

2.4 Delphi 7

Delphi เป็น ซอฟต์แวร์ตัวแปลภาษาปาสกาล (Pascal) ตัวหนึ่ง ที่พัฒนาโดยบริษัท Borland ซึ่งรูปแบบของภาษานั้น ได้ถูกออกแบบระบบภาษาใหม่ เพื่อสนับสนุนการเขียนโปรแกรมแบบเชิงวัตถุ (Object oriented programming หรือเรียกแบบย่อว่า OOP) การเขียนโปรแกรมแบบเชิงวัตถุนั้นมีความแตกต่างกับการเขียนโปรแกรมแบบโครงสร้างอยู่ด้วยกันหลายจุด ซึ่งส่วนใหญ่แล้วความสามารถของโปรแกรมเชิงวัตถุ ยังคงต้องการความรู้ของการเขียนโปรแกรมแบบโครงสร้างเป็นพื้นฐาน ทั้งนี้เนื่องจากภาษาโปรแกรมเชิงวัตถุเป็นภาษาที่ออกแบบมาเพื่อเพิ่ม ประสิทธิภาพของการเขียนโปรแกรมในรูปแบบเดิม คือ ให้ง่ายต่อการพัฒนาต่อและสามารถนำไปใช้ได้

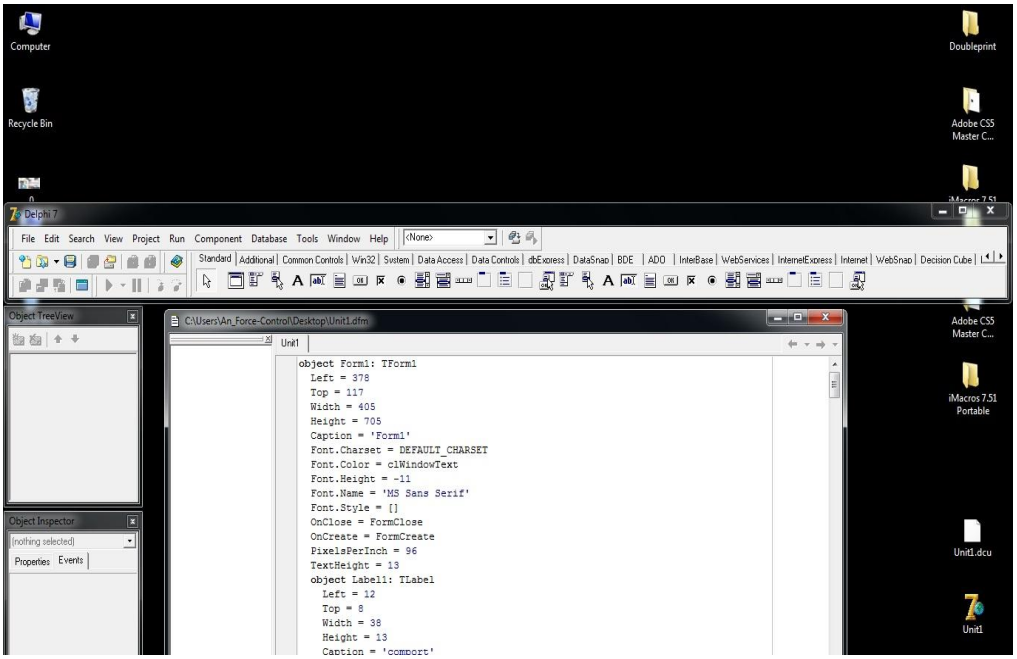
2.4.1 ความสามารถของ Delphi 7

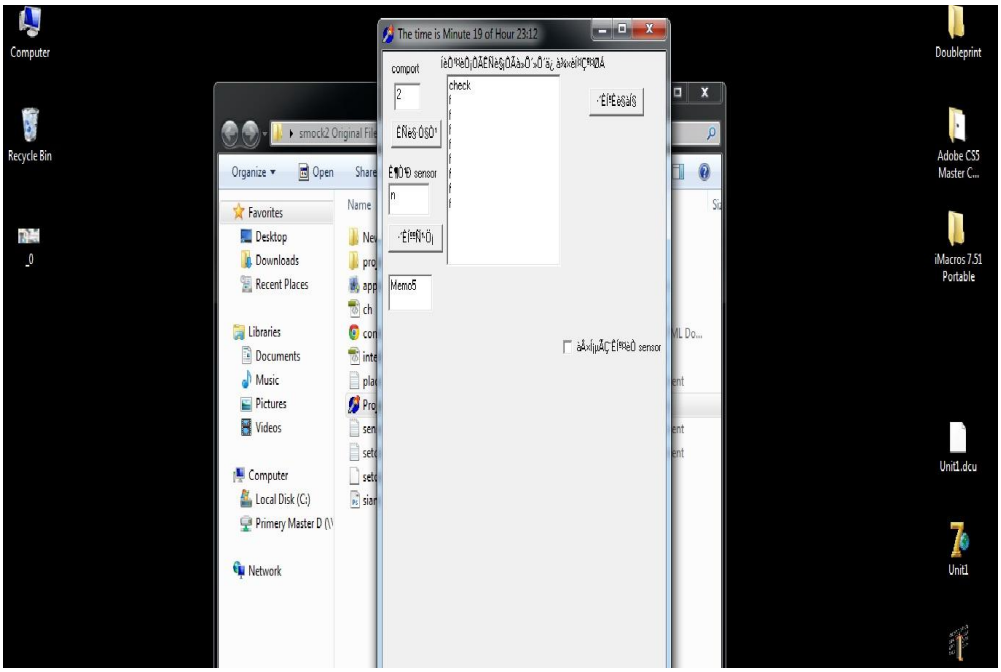
Delphi 7 นั้นมีความสามารถมากมาย ได้รับการตอนรับเป็นอย่างดีจากนักพัฒนาแอพพลิเคชันทั่วโลก รวมทั้งเมืองไทยด้วย ซึ่งจะเห็นได้จากการนำ Delphi 7 ประกอบการเรียนการสอน การฝึกอบรม ตลอดจนการนำไปสร้างเป็นซอฟต์แวร์เชิงพาณิชย์จำนวนมากและได้ยกมาอย่างคร่าว ๆ ดังนี้

1. สามารถสร้างแอพพลิเคชันสำหรับ Window
2.สามารถสร้างระบบงานด้านฐานข้อมูล
3. สามารถสร้างแอพพลิเคชันรองรับ .NET Web Service
4. สามารถใช้งานบน Linux ได้

โดยมีตัวอย่างการออกแบบโปรแกรมด้วย Delphi 7 ตามรูปที่ 2.5

รูปที่ 2.5 การออกแบบโปรแกรมด้วย Delphi 7





รูปที่ 2.5 (ต่อ) การออกแบบโปรแกรมด้วย Delphi 7