เครื่องตั้งเวลาหยอดเหรียญ

PROGRAMABLE TIMER FOR VENDING MACHINE

โดย

1.นายไตรรงค์ ดวงชมภู รหัสประจำตัว 39013271

2.นายธีระพล เที่ยงธรรม รหัสประจำตัว 39013274

3.นายสมบูรณ์ ชูทอง รหัสประจำตัว 39013294

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

สาขาวิชาเทคโนโลยีอิเล็กทรอนิกส์ ภาควิชาเทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

หัวข้อปริญญานิพนธ์	เครื่องตั้งเวลาหยอดเหรียญ		
	PROGRAMABLE TIMER FOR VENDING MACHINE		
จัดทำโดย	นายไตรรงค์	ดวงชมภู	รหัสประจำตัว 39013271
	นายธีระพล	เที่ยงธรรม	รหัสประจำตัว 39013274
	นายสมบูรณ์	ชูทอง	รหัสประจำตัว 39013294
สาขาวิชา	เทคโนโลยีอิเล็กทรอนิกส์		
ภาควิชา	เทคนิคอุตสาหกรรม		
อาจารย์ที่ปรึกษา	ผู้ช่วยศาตราจารย์ประดิษฐ์ วัชรพิบูลย์		
ปีการศึกษา	2541		

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังอนุมัติให้นับปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

_____ประธานกรรมการ
(_____)

_____กรรมการ
(_____)

_____กรรมการ
(_____)

_____กรรมการ
(_____)

_____กรรมการ
(_____)

_____กรรมการ
(_____)

ลิขสิทธิ์ของคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เครื่องตั้งเวลาหยอดเหรียญ

PROGRAMABLE TIMER FOR VENDING MACHINE

โดย	นายไตรรงค์ ดวงชมภู	รหัสประจำตัว 39013271
	นายธีระพล เที่ยงธรรม	รหัสประจำตัว 39013274
	นายสมบูรณ์ ชูทอง	รหัสประจำตัว 39013294
อาจารย์ที่ปรึกษา	ผู้ช่วยศาตราจารย์ประดิษฐ์ วัชรพิบูลย์	
ปีการศึกษา	2541	

บทคัดย่อ

โครงงานเครื่องหยอดเหรียญ ตั้งเวลาได้ โดยใช้ไมโครคอนโทรลเลอร์ ควบคุมหน้าที่การทำงานของกลไก จุดมุ่งหมายหลัก เพื่อเป็นการศึกษาถึงการทำงานของไมโครคอนโทรลเลอร์ และการทำงานของระบบกลไก และเพื่อเป็นการพัฒนาทักษะในการเขียนโปรแกรมใช้งานในระบบควบคุม ได้อย่างมีประสิทธิภาพสูงสุด

PROGRAMMABLE TIMER FOR VENDING MACHINE

BY	**Mr. Trairong**	**Doungchompoo**	**39013271**
	Mr. Teerapon	**Theingtham**	**39013274**
	Mr. Somboon	**Choothong**	**39013294**
ADVISOR	**Assistant prof. Pradit wacharapiboon**		
YEAR	**1998**		

Abstract

This thesis present the using of microcontroller to control the mechanical of the vending machine with programmable timer. The main purpose is to study the function of microcontroller AT89C51 and AT89C2051 for development the skill of writing the assembling language

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จลงด้วยความเรียบร้อยโดยได้รับการสนับสนุนและการให้คำปรึกษาจากหลายฝ่าย ฉนั้นผู้จัดทำจึงขอขอบพระคุณท่านอาจารย์ประดิษฐ์ วัชรพิบูลย์ เป็นอย่างสูงในการให้คำปรึกษาชี้แนะ และติชมแก้ไข พร้อมทั้งขอขอบคุณ คุณก้องเกียรติ เอนกพงษ์พันธ์ ที่ให้ความช่วยเลือทางด้านอุปกรณ์ทดลอง คุณรัชตภาคย์ ปัตตะวังศ์ ที่ช่วยให้คำแนะนำทางด้านการเขียนโปรแกรม ขอบคุณเพื่อนๆทุกท่าน ที่ให้ความสนุบสนุน คำชี้แนะ และช่วยเหลือในการทำโครงงาน ซึ่งทำให้โครงงานนี้สำเร็จลุล่วงไปได้ด้วยดี ตามวัตถุประสงค์

คณะผู้จัดทำ
นายไตรรงค์ ดวงชมภู
นายธีระพล เที่ยงธรรม
นายสมบูรณ์ ชูทอง

สารบัญ

บทที่ 1
บทนำ

วัตถุประสงค์

1. เพื่อศึกษาการเขียนโปรแกรมในระบบควบคุม
2. เพื่อศึกษาระบบกลไกควบคุมการหยอดเหรียญ
3. เพื่อศึกษาการเชื่อมต่อข้อมูลของระบบไมโครคอนโทรลเลอร์
4. เพื่อศึกษาระบบการทำงานแบบแมคคาโทรนิกส์
5. สามารถนำไปประยุกต์ใช้งานได้

แนวความคิดและที่มา

เนื่องจากเครื่องตั้งเวลาหยอดเหรียญโดยทั่วๆ ไปจะมีส่วนแสดงผลอยู่เพียงชุดเดียว ซึ่งจะทำให้การควบคุมดูแลได้จำกัด เมื่อมีการนำระบบ Multiprocessor System มาใช้จะทำให้ตัวแสดงผลสามารถมีถึง 2 ชุด และสามารถถ่ายโอนข้อมูลซึ่งกันและกันได้ จะทำให้การควบคุมมีความสามารถเพิ่มขึ้น และการทำงานของ Microprocessor ของส่วนแสดงผลทั้งสอง จะทำงานโดยตัววงจรสร้างฐานเวลาตัวเดียวกัน ทำให้ข้อมูลของทั้งสองเครื่องตรงกัน และทำงานร่วมกันได้อย่างสมบูรณ์

ประโยชน์ที่คาดว่าจะได้รับ

1. ได้รับความรู้ความเข้าใจในการเขียนโปรแกรมควบคุมระบบไมโครคอนโทรลเลอร์
2. ได้รับความรู้ความเข้าใจในระบบกลไกระบบความคุมการหยอดเหรียญ
3. เข้าใจระบบการเชื่อมต่อข้อมูลของไมโครคอนโทรลเลอร์
4. ได้รับความรู้ความเข้าใจในระบบแมคคาโทรนิกส์

ในการสร้างเครื่องหยอดเหรียญขึ้นมานี้ เพื่อศึกษาการเขียนการพัฒนาโปรแกรมและการใช้ไมโครคอนโทรลเลอร์ในการควบคุมไอซีต่างๆ แบบอนุกรม และนำมาควบคุมระบบแมคคานิค พร้อมทั้งศึกษาการทำงานเชื่อมต่อข้อมูลระหว่างไมโครคอนโทรลเลอร์เอง และเชื่อมต่อกับระบบแมคคานิค โดยวงจรจะมีไอซีไมโครคอนโทรลเลอร์ ตระกูล MCS-51 เบอร์ AT89C51 และ AT89C2051 ของบริษัท ATMAL มีหน่วยความจำขนาด 4 กิโลไบท์ สามารถโปรแกรมได้ 1000 ครั้ง มีหน่วยความจำแบบแรม 8 บิท ขนาด 128 ไบท์ (Internal RAM) และการทำงานทั้งหมดของ

ระบบใช้ แอลซีดี(LCD) เป็นตัวแสดงผล ใช้ไอซี DALLUS 1202 เป็น อาร์ทีซี (RTC) สร้างฐานเวลาจริง ใช้ไอซี 93C46 เป็น อีสแควพรอม (EEPROM) ในการบันทึกข้อมูล ใช้ไอซี MAX691 เป็นแบตเตอรี่ในการแบ็คอัพ(Back up) ข้อมูลแสดงจำนวนเวลาของเหรียญ ใช้ไอซี opto coupler 4N33 และ 4N25 ในการเชื่อมต่อใช้งานบังคับโซลินอยด์(Solinoids) และการทำงานทางด้านแมคคานิค ใช้ไอซี 75176 ดปืนตัวเชื่อมต่อข้อมูลระหว่าง ไอซีไมโครคอนโทรลเลอร์ AT89C51(Master) และAT89C2051 (Slave) ระบบแมคคานิคใช้ โซลีนอยด์เป็นตัวควบคุมการจัดเก็บเหรียญ คืนเหรียญ และคัดเหรียญออก โดยใช้วัสดุพลาสติกในเครื่องต้นแบบ

เครื่องหยอดเหรียญนี้มีการติดต่อกับผู้ใช้ ประกอบไปด้วยฟังค์ชั่น(Function) 2 อย่างคือ

- ตั้งวันเวลาของเครื่อง
- ตั้งเวลาต่อเหรียญ

บทที่ 2
หลักการและการทำงานทั่วไป

จากหลักการทำงานโดยทั่วไปจะมีการทำงานที่เราสามารถแบ่งเป็นภาคใหญ่ๆ ได้ 2 ภาค คือ ภาคคอนโทรล (control) และ ภาค ไดรเวอร์ (driver) โดยส่วนแสดงผลจะเป็นใช้ แอลซีดี (LCD)ในการแสดงผล โดยจะแสดงทั้งหมด 2 ชุดคือที่ตัวเครื่องหลัก (master) และตัวเครื่องรอง (slave) ซึ่งการแสดงผลจะมีการแสดงผลจำนวนเหรียญ เวลานับถอยหลัง วันเดือนปีปัจจุบัน และจะมีฟังค์ชั่น(function) การโปรแกรมการทำงานของเครื่องคือจำนวนเวลาต่อเหรียญ 1 เหรียญ และตั้งเวลาปัจจุบัน โดยจะมีคีย์สำหรับการโปรแกรมได้โดยตัวผู้ใช้เอง โดยการทำงานเราสามารถแบ่งเป็นส่วนๆ ดังนี้

ภาคคอนโทรล

ประกอบไปด้วย ไอซีหลายๆตัวมาทำงานร่วมกัน ประกอบไปด้วย

1.ซีพียู (CPU) เบอร์ AT89C51 และ AT89C2051 ของบริษัท ATMEL

โครงสร้างของ MCS-51

ไมโครคอนโทรลเลอร์แบบชิพเดี่ยวตระกูล MCS-51 นี้ผลิตโดยบริษัทอินเทลมีอยู่ด้วยกันหลายเบอร์ซึ่งมีรายละเอียดดังตารางที่ 2.1

Device	ROMless version	EPROM version	ROM Bytes	RAM Bytes	8-Bit I/O ports	16-Bit Timer/Counters	Programmable counter Array (PCA)	UART	Serial Expansion port(SEP)	Global Serial Channel (GSC)	DMA Channels	A/D Channels	Interrupt Sources/Vectors	Power Down and Idle Modes
8051	8031	-	4K	128	4	2		/					6/6	
8051AH	8031AH	8751H 8751BH	4K	128	4	2		/					6/6	
8052AH	8032AH	8752BH	8K	256	4	3		/					8/8	
89C51BH	89C31BH	87C51	4K	128	4	2		/					6/6	/
89C52	89C32	-	8K	256	4	3		/					8/6	/
83C51FA	89C51FA	87C51FA	8K	256	4	3	/	/					14/7	/
83C51FB	89C51FA	87C51FB	16K	256	4	3	/	/					14/7	/
83C152JA	89C152JA	-	8K	256	6	2		/		/	2		10/11	/
-	89C152JB	-	-	256	7	2		/		/	2		10/11	/
83C152JC-	89C152JC	-	8K	256	6	2		/		/	2		10/11	/
-	89C152JD	-	-	256	7	2		/		/	2		10/11	/
83C452	89C452	87C452P	8K	256	6	2		/					8/8	/

ตารางที่ 2.1

ตระกูล MCS-51 ได้ถูกออกแบบมาเพื่อใช้ในอุตสาหกรรมมาตรฐานขนาด 8 บิทไมโครคอนโทรลเลอร์ และให้มีความสามารถในการงานควบคุมประยุกต์ใช้งานในเรื่อง sequential real time control , close loop control และ data control และมีส่วนคล้ายกับ MCS-48 แต่จะทำงานได้เร็วกว่าเป็น 2 ถึง 5 เท่า รวมทั้งอุปกรณ์ที่เพิ่มขึ้นตามลักษณะหลักทั่ว ๆ ไป

คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51

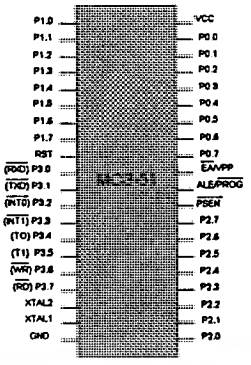
- ต้องการแหล่งจ่ายไฟ +5 V ชุดเดียว
- มีหน่วยความจำโปรแกรม (Program Memory) ขนาด 4 กิโลไบท์สำหรับเบอร์ 8051 และ 8031 , 8032 ไม่มีหน่วยความจำชุดนี้ ส่วน 8052 มีหน่วยความจำถึง 8 กิโลไบท์
- มีวงจรออสซิลเลเตอร์ และวงจรนาฬิกาบนชิป
- ตัวโพรเซสเซอร์สามารถใช้งานแบบบูลีนได้ สำหรับการใช้กับกระบวนการงานควบคุม
- มีหน่วยความจำสำหรับเก็บข้อมูล (Data Memory) ขนาด 128 ไบท์ สำหรับ 8052 มีถึง 256 ไบท์
- หน่วยความจำสำหรับโปรแกรมและดาต้า (Program Memory และ Data Memory แยกจากกันอย่างละ 64 กิโลไบท์)
- คำสั่งที่ใช้เวลาน้อยที่สุดประมาณ 1 us เมื่อทำงานที่ความถี่ 12 MHz
- มี Timer /Counter ขนาด 16 บิท 2 ชุด (สำหรับ 8052 มี 3 ชุด) ทำงานได้ 4 โหมด
- รับอินเตอร์รัพท์ได้ 6 แหล่ง 5 เวคเตอร์
- มีพอร์ทรับส่งข้อมูลอนุกรม (UART) 2 พอร์ท ทั้งรับและส่งในเวลาเดียวกันได้ (Full Duplex) เลือกรูปแบบการส่งข้อมูลได้ 4 โหมด
- มีคำสั่งในการทำ AND , OR หรือ COMPLEMENT ได้ทั้งแบบ 8 บิทและ 1 บิท

1. การจัดขาลักษณะภายนอกของ MCS-51

รูปที่ 2.23 แสดงการจัดขาตามลักษณะภายนอกของชิป MCS-51 ซึ่งจะมีการแบ่งกลุ่มการจัด

ขาตามสถาปัตยกรรมของ MCS-51 อยู่ 4 กลุ่ม คือ

1. กลุ่มขารับแหล่งจ่ายไฟฟ้า และระบบสัญญาณนาฬิกา
2. กลุ่มขาแอดเดรสและข้อมูล
3. กลุ่มขาควบคุม
4. กลุ่มขาพอร์ทแบบขนานและอนุกรม



รูปที่ 2.1 ลักษณะการจัดขาภายนอกของ MCS-51

ขาบางขาจะทำหน้าที่ได้สองหน้าที่ขึ้นอยู่กับการติดตั้งด้วยซอฟต์แวร์หรือฮาร์ดแวร์ เช่น ขาที่ 32-39 จะทำหน้าที่ได้เป็นกลุ่มขาแอดเดรสและข้อมูลหรือจะทำหน้าที่เป็นกลุ่มขาพอร์ทแบบขนานเป็นต้น รายละเอียดหน้าที่ขาแต่ละขาจะมีดังนี้

ขา Vss (ขา 20)

เป็นขาสำหรับต่อลงดิน

ขา Vcc (ขา40)

เป็นขาที่ต่อแรงดันไฟกระแสตรงขนาด 5V และใช้สำหรับการโปรแกรม

ขา PORT 0 (P0.0-P0.7/AD0-AD7) (ขา 32-39)

ทำหน้าที่เป็นพอร์ทไอโอ 8 บิทแบบ Open Drain Bidirectional สามารถ ที่จะรับโหลดทีทีแอลได้ 8 ตัว การเขียนค่า '1' ไปที่พอร์ทนี้ จะเป็นการปล่อยลอย ขาของพอร์ทนี้ ทำให้มันทำงานเป็นอินพุท มีสถานะอิมพีแดนซ์สูง ในการให้พอร์ทนี้บริการแบบไอโอและอีกหน้าที่หนึ่งของพอร์ท 0 จะทำงานเป็นมัลติเพลกซ์ ด้วยสัญญาณแอดเดรสไบท์ต่ำกับบัสข้อมูล สำหรับการใช้งานด้านหน่วยความจำภายนอก ในการใช้งานแบบนี้จะใช้ลักษณะภายในเป็นตัวพูลอัพนอกจากหน้าที่หลัก 2 หน้าที่ ดังกล่าวแล้ว พอร์ท 0 ยังใช้งานพิเศษเป็นตัวส่งข้อมูลออกทางพอร์ทนี้ เมื่อใช้บริการทางด้านการตรวจสอบโปรแกรม ROM ภายใน และการโปรแกรมตัว EPROM ภายใน ถ้าใช้งานในลักษณะนี้การพูลอัพจากภายนอกจะต้องต่อด้วยค่า 10 กิโลโอห์ม

ขา PORT 1 (P1.0-P1.7) (ขา 1-8)

เป็นพอร์ทไอโอ 8 บิทแบบ Open Drain Bidirectional พร้อมด้วยการพูลอัพภายใน ถ้าเป็นพอร์ทเอาท์พุท บัฟเฟอร์สามารถขับโหลดทีทีแอลตระกูลแอลเอสไอได้ 4 ตัว พอร์ท 1 เมื่อถูกเขียนค่า '1' ด้วยโปรแกรมมันจะมีสถานะสูงด้วยการพูลอัพภายใน การให้สถานะเช่นนี้ จะ

เป็นการ Initial ใช้งานพอร์ทนี้ให้เป็นอินพุท ขณะที่พอร์ท 1 เป็นอินพุท การให้สัญญาณลงต่ำจะเป็นการจ่ายกระแสออกเนื่องจากการพูลอัพภายใน ในเบอร์ 8052 ขา P1.0 และ P1.7 จะใช้งานเป็น T2 และ T2EX โดยขา T2 จะทำหน้าที่รับสัญญาณจากภายนอกให้ตัวจับเวลา 2 ทำงาน และขา T2EX จะเป็นอินพุทผ่านเข้าตัวจับเวลา 2 ถูกกระตุ้นให้ทำงานแบบปกติตามโปรแกรมที่ติดตั้งไว้ หรือ เค็ปเจอร์

ขา PORT 2 (P2.0-P2.7) (ขา 21-28)

เป็นพอร์ทไอโอ 8 บิทแบบ Open Drain Bidirectional ด้วยการพูลอัพภายใน พอร์ท 2 ที่ทำหน้าที่เป็นบัฟเฟอร์เอาท์พุทสามารถจ่ายโหลดทีทีแอลตระกูลแอลเอสได้ 4 ตัว อีกหน้าที่หนึ่งของพอร์ทจะถูกใช้งานเป็นตัวส่งแอดเดรสไบท์สูงด้วย เมื่อใช้งานร่วมกับหน่วยความจำภายนอกเพื่อให้แอดเดรสได้ถึง 16 บิท ด้วยการใช้งานแบบนี้มันจะมีพูลอัพภายในที่ช่วยให้การส่งค่า '1' ได้ระดับที่แน่นอนนอกจากการใช้งานสำหรับแอดเดรสอันดับสูงยังใช้เป็นขาควบคุมในการใช้งานตรวจสอบ และเขียนโปรแกรมเบอร์ 8751 และตรวจสอบโปรแกรมภายใน 8051

ขา PORT 3 (P3.0-P3.7) (ขา 10-17)

เป็นพอร์ทไอโอ 8 บิท แบบพูลอัพภายใน นอกจากทำเป็นพอร์ทไอโอที่สามารถรับโหลดทีทีแอลพวกตระกูลแอลเอสได้ 4 ตัว แล้วยังมีอีกหน้าที่หนึ่งของตระกูล MCS-51 ตามรายการข้างล่างนี้ด้วย

ขาพอร์ท	ขา	การทำงานตามฟังก์ชั่นพิเศษ
P3.0	10	RxD พอร์ทอนุกรมอินพุท
P3.1	11	TxD พอร์ทอนุกรมเอาท์พุท
P3.2	12	$\overline{\text{INT0}}$ อินเตอร์รัพท์ภายนอกตัวที่ 1
P3.3	13	$\overline{\text{INT1}}$ อินเตอร์รัพท์ภายนอกตัวที่ 2
P3.4	14	T0 สัญญาณกระตุ้นเข้า ที่ตัวจับเวลา /ตัวนับ 0
P3.5	15	T1 สัญญาณกระตุ้นเข้าที่ตัวจับเวลา/ตัวนับ 1
P3.6	16	$\overline{\text{WR}}$ สัญญาณควบคุมการเขียน
P3.7	17	$\overline{\text{RD}}$ สัญญาณควบคุมการอ่าน

การที่จะให้ทำงานตามฟังก์ชั่นข้างบนได้ จะต้องติดตั้งโปรแกรมด้วยการส่งค่า '1' ไปแลทช์ไว้ก่อนที่ให้ทำงานตามฟังก์ชั่นข้างบน

ขา RST (ขา 9)

ต้องคงสถานะค่าสูงเป็นเวลาประมาณอย่างน้อยสองวัฏจักรระหว่างที่ออสซิเลเตอร์ทำงานขณะที่ต้องการรีเซททั้งระบบงาน โดยจะต่อรีซิสเตอร์พูลดาวน์จากขา RST ไปลงดิน และเพื่อให้ตัวชิปรีเซทได้โดยอัตโนมัติ ขณะเปิดไฟจะใช้คาปาซิเตอร์ ต่อคร่อมระหว่างขา RST กับขา Vcc

ขา ALE/$\overline{\text{PROG}}$ *(ขา 30)*

เป็นขาแอดเดรสแลทช์อีนาเบิ้ลด้วยการส่งพัลส์ออกไปใช้สำหรับแลทช์ค่าแอดเดรสไบท์ต่ำจากพอร์ท 0 ในระหว่างการเข้าถึงข้อมูลจากหน่วยความจำภายใน ALE จะถูกส่งสัญญาณนาฬิกาออกมาอัตราความเร็วคงที่ ที่ 1/8 ของความถี่ออสซิเลเตอร์ตลอดเวลาแม้ว่าบางช่วงจังหวะจะไม่มีการเข้าถึงข้อมูลจากภายใน ดังนั้นจึงสามารถที่จะใช้สัญญาณจากขานี้เป็นตัวจับเวลาภายนอกหรือเป็นความถี่สัญญาณนาฬิกา แต่อย่างไรก็ตามความถี่สัญญาณนี้จะลดความถี่ช้าลงไปเท่าหนึ่งระหว่างการทำงานแบบการเข้าถึงของหน่วยความจำข้อมูลภายนอก ขานี้ยังจะใช้เป็นสัญญาณพัลซ์เข้า สำหรับการควบคุมการโปรแกรม EPROM ภายในชิป

ขา $\overline{\text{PSEN}}$ *(ขา 29)*

Program Storage Enable เป็นสโตรบอ่านข้อมูลจากโปรแกรมหน่วยความจำภายนอก เมื่อชิปทำงานด้วยโปรแกรมภายนอก ขา PSEN จะสร้างสโตรบต่ำสองครั้งภายในแต่ละวัฏจักรแมชชีนสัญญาณจะมีสถานะสูง หรือพัลส์ต่ำทั้งสองลูกจะหายไป เมื่อทำงานในช่วงการอ่านหรือเขียนข้อมูลจากหน่วยความจำข้อมูลภายนอกและ PSEN จะไม่มีพัลส์ส่งออกถ้าชิปทำงานด้วยโปรแกรมหน่วยความจำภายใน

ขา $\overline{\text{EA}}$ */Vpp (ขา 31)*

มีสถานะสูง ตัวซีพียูในชิปจะทำงานตามโปรแกรมที่อยู่ในหน่วยความจำภายใน การทำให้ EA สถานะต่ำจะเป็นการควบคุมให้ซีพียูทำงานตามโปรแกรมหน่วยความจำภายนอก ซึ่งขยายโปรแกรมได้ยาวถึง 64 กิโลไบท์ ในตัว 8031 AH และ 8032 AH ขา EA จะต้องต่อลงดินเช่นกันแม้ว่าจะไม่มี ROM อยู่ภายในก็ตาม ในตัว 8751 H จะใช้ขานี้จ่ายแรงดันขนาด 21V ขณะทำการเขียนโปรแกรมเข้า EPROM ของชิป 8751 H ตัวนี้

ขา XTAL1 (ขา19)

ใช้เป็นตัวอินพุทเข้าสู่ตัวออสซิเลเตอร์ขยายแบบ Invert

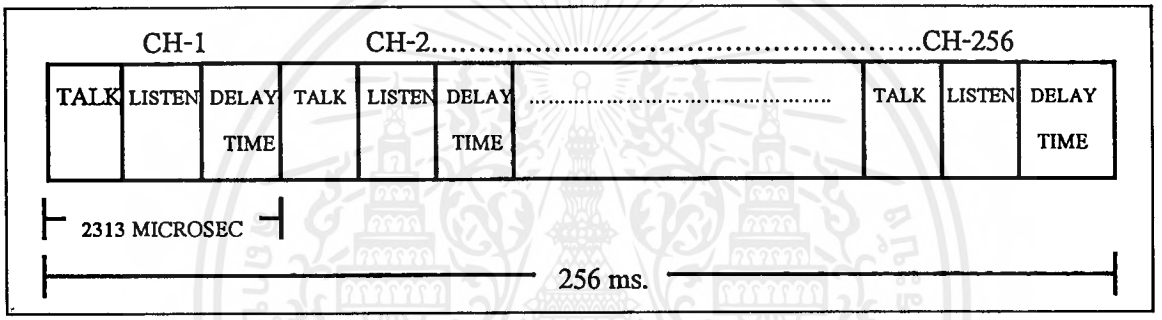
ขา XTAL2 (ขา 18)

ใช้เป็นตัวเอาท์พุทจากตัวออสซิเลเตอร์ขยายแบบ Invert

การสื่อสารระบบ Multiprocessor

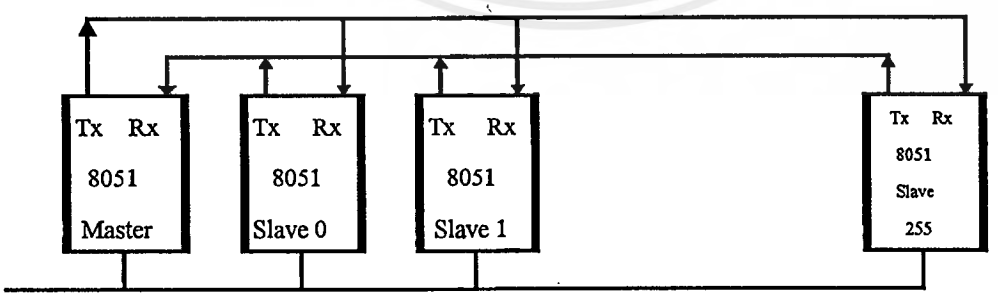
1. การสื่อสารข้อมูลแบบอนุกรมโดยใช้หลักการของมัลติโปรเซสเซอร์

ในการติดต่อสื่อสารข้อมูลแบบอนุกรมโดยใช้หลักการไมโครโปรเซสเซอร์ จำเป็นที่จะต้องกำหนดค่าแอดเดรสของตัวเอง เพื่อจะได้ไม่แย่งกันทำงาน โดยตัว Master จะเป็นตัวส่งแอดเดรสไบต์ของตัว Slave ออกไป (Slave มีได้ 256 ตัว) แล้วตามด้วย ดาต้าไบต์ ซึ่งมีข้อมูลส่งไปยัง Slave เวลาช่วงแรกนี้เราถือว่าเป็นการ Talk หลังจากในช่วงเวลานี้ตัว Master ก็จะรอสัญญาณตอบกลับจากตัว Slave เวลาช่วงนี้เราถือว่าเป็นการ Listen ถ้าเราต้องการติดต่อกับตัว Slave ทั้ง 256 ตัวเราสามารถเขียนรูปแบบการติดต่อได้ดังรูป 2.2



รูปที่ 2.2 แสดงแบบการติดต่อระหว่าง Master กับ Slave ทั้ง 256 ตัว

การต่อสัญญาณจากตัว Master ไปยัง Slave ทั้ง 256 ตัว ดังแสดงในรูป 2.3



รูปที่ 2.3 การสื่อสารระบบ Multiprocessor Mode แบบ Full Duplex

2. รูปแบบสัญญาณที่ส่งออกจากตัว Master

MCS-51 กำหนดรูปแบบการรับส่งข้อมูลได้ 2 แบบคือ Single Processor Mode โดยเซ็ท SM2=0 และ Multiprocessor Mode โดยเซ็ท SM2=1 การส่งข้อมูลจะต้องเป็น (โหมด 2,3) และโหมด 3 เท่านั้นเพราะส่งข้อมูลได้ 11 บิต โดยบิตแรกคือ Start bit อีก 8 บิต ข้อมูลตามด้วยบิตที่ 9 ซึ่งใช้เป็นตัวบอกว่าไบต์ที่ส่งไปนี้ แอดเดรสไบต์ หรือดาต้าไบต์

ทางด้านส่ง บิต 9=1 หมายถึง แอดเดรสไบต์

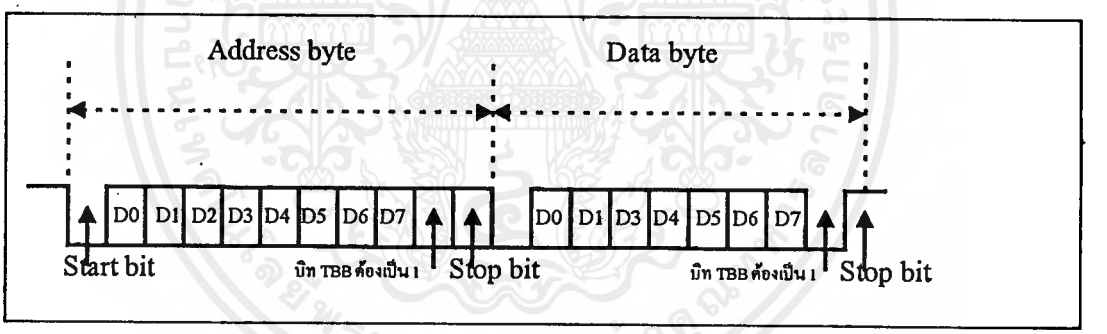
บิต 9=0 หมายถึง ดาต้าไบต์

ทางด้านรับ บิต 9=1 หมายถึง แอดเดรสไบต์

บิต 9=0 หมายถึง ดาต้าไบต์

*** การโปรแกรมให้บิตที่ 9 เป็น 1 หรือ 0 เราจะต้องโปรแกรมที่บิต TB8 ใน SCON

- เมื่อการทำงานแบบ Multiprocessor Mode RI จะ SET เมื่อค่าบิตที่ 9 ที่รับเข้ามามีค่าเป็น หนึ่ง มีค่าเป็น 1



รูปที่ 2.4 สัญญาณที่ขา TxD ของตัว Master

หมายถึง :- บิต TB8 ทางด้านส่งจะเข้าไปเก็บใน RB8 ทางด้านรับ บิตนี้อยู่ใน SCON การส่งข้อมูลโหมดนี้จะต้องส่ง แอดเดรสไบต์ ออกไปก่อน บิตที่ 9 (TB8) ต้องเป็น 1 หลังจากนั้นให้เคลียร์ TB8 เพื่อส่ง ดาต้าไบต์ ดูรูปที่ 2.4 ประกอบ

เบอร์	หน่วยความจำ ภายใน		ตัวจับเวลา/ ตัวนับจำนวน	อินเตอร์รัพท์
	โปรแกรม	ข้อมูล		
8052 AH	8K x 8 ROM	256 x 8 RAM	3 x 16 BIT	6
8051 AH	4K x 8 ROM	128 x 8 RAM	2 x 16 BIT	5
8051 .	4K x 8 ROM	128 x 8 RAM	2 x 16 BIT	5
8032 AH	NO ROM	256 x 8 RAM	3 x 16 BIT	6
8031 AH	NO ROM	128 x 8 RAM	2 x 16 BIT	5
8031 .	NO ROM	128 x 8 RAM	2 x 16 BIT	5
8751 H .	4Kx8 EPROM	128 x 8 RAM	2 x 16 BIT	5
8752 H .	8Kx8 EPROM	256 x 8 RAM	3 x 16 BIT	6

ตารางที่ 2.2 ตารางรายละเอียดของตระกูล MCS-51

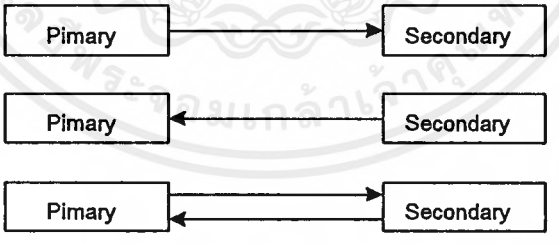
ตามตารางที่ 2.2 MCS-51 ทั้งสามกลุ่ม คือ กลุ่มที่มี ROM ไม่มี ROM และพวก EPROM จะมีขาใช้งานเหมือนกันหมด ยกเว้นขา 1 จะใช้งานเป็น T2 และขา 2 เป็น T2EX ในเบอร์ 8032/8052 ตลอดถึงจังหวะเวลา และคุณสมบัติทางไฟฟ้าทั้งสามจะแตกต่างกันเฉพาะการโปรแกรมบนชิป MCS-51 เท่านั้น ซึ่งแต่ละแบบจัดไปตามความต้องการของผู้ใช้เช่น 8751 จะมี 4 กิโลไบท์ของ Ultraviolet Erasable Programmable Read Only Memory (EPROM) เหมาะสำหรับการพัฒนาเครื่องต้นแบบ และการผลิตอุปกรณ์ที่มีจำนวนจำกัด เมื่อต้องการจะเขียนโปรแกรมเข้า EPROM จะมีตัวเขียนโปรแกรมพิเศษสำหรับเขียนโปรแกรมที่ผู้ออกแบบเขียนขึ้นมาได้ ถ้าโปรแกรมมีบักหรือส่วนผิดพลาดที่ต้องการจะแก้ไข ก็สามารถแก้ไขได้โดยการนำตัว 8751 นี้ไปล้างโปรแกรมเดิมออกด้วยแสงอุลตร้าไวโอเลท และอัดข้อมูลโปรแกรมที่ได้แก้ไขแล้วเข้าไปใหม่ ทำเช่นนี้จนกระทั่งได้โปรแกรมสมบูรณ์ และเมื่อต้องการผลิตจำนวนมากก็สามารถที่จะใช้ MCS-51 เบอร์ 8051 ที่มี 4 กิโลไบท์ ของ ROM ซึ่งจะถูกอัดข้อมูลโปรแกรมตามความต้องการของผู้ออกแบบโดยโรงงาน ผู้ผลิตชิปเบอร์นี้ การผลิตลักษณะนี้จะถูกกว่าการใช้เบอร์ 8751 แต่โปรแกรมภายในจะไม่สามารถลบ และโปรแกรมใหม่ได้หลังการผลิตไปแล้ว

<u>หลักการและโครงสร้างการสื่อสารข้อมูลแบบอนุกรม</u>

หลักการและโครงสร้างการสื่อสารข้อมูลแบบอนุกรมที่การรับและการส่งข้อมูลเป็นการใช้สายสัญญาณในการส่งข้อมูลจำนวนน้อย ซึ่งโดยปกติจะใช้เพียงๆ 1 คู่คือสายสัญญาณที่เป็นข้อมูลและสายกราวด์ (Ground) เปรียบเทียบกันโดยข้อมูลจะส่งออกหรือรับเข้าจะเป็นลักษณะที่เป็นบิท(Bit) ต่อบิท(Bit) ซึ่งเมื่อเราทำการเปรียบเทียบกับการสื่อสารข้อมูลแบบขนานที่จำนวนข้อมูลและอัตราการส่งข้อมูลที่เท่ากันแล้วจะพบว่า การสื่อสารข้อมูลแบบอนุกรมนี้จะต้องใช้เวลาในการรับ-ส่งข้อมูลมากกว่าแน่นอน แต่ข้อดีของการสื่อสารข้อมูลแบบอนุกรม คือ การใช้สายสัญญาณน้อยกว่าและสามารถที่จะส่งข้อมูลได้ไกลกว่าแม้ว่าอัตราการลดทอนและการเพี้ยนของสัญญาณที่มีผลจากความยาวของของสายสัญญาณจะมีค่าเท่ากับการสื่อสารข้อมูลแบบขนาน แต่การสื่อสารข้อมูลแบบอนุกรมมีวิธีที่ใช้ในการที่จะลดผลของการลดทอนของสัญญาณนี้โดยการรับ-ส่งสัญญาณแบบดิฟเฟอร์เรนเชียล (Differential) ดังนั้นการสื่อสารข้อมูลแบบอนุกรมจึงเหมาะสมกับการสื่อสารข้อมูลในระยะทางไกลๆ การสื่อสารข้อมูลที่ต้องใช้จำนวนสายหือช่องสัญญาณในการรับ-ส่งข้อมูลจำนวนน้อย เช่นการสื่อสารข้อมูลท้องถิ่น (Local Area Netwoek : LAN)

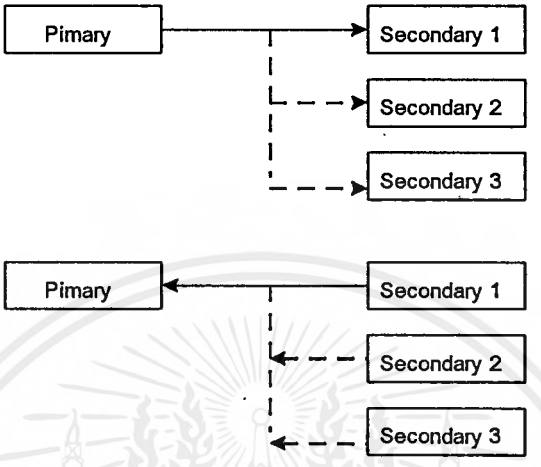
จากที่ได้กล่าวมาแล้วการสื่อสารข้อมูลแบบอนุกรมยังสามารถที่จะแบ่งตามหลักลักษณะของทิศทางในการสื่อสารข้อมูล ตามโครงสร้างและความต้องการของระบบ ได้ดังนี้คือ

1.การสื่อสารข้อมูลในทิศทางเดียวตลอดเวลา :ซิมเพล็ก (Simplex)



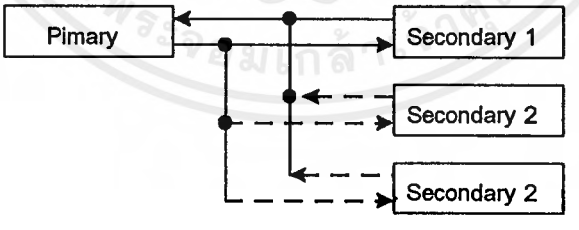
รูป2.5 แสดงโครงสร้างการสื่อสารข้อมูลแบบซิมเพล็ก (Simplex)

2.การสื่อสารข้อมูลใน 2 ทิศทางแต่สลับเวลากัน : ฮาฟดูเพล็ก (Half-Duplex)



รูป2.6 แสดงโครงสร้างการสื่อสารข้อมูลแบบฮาฟดูเพล็ก (Half-Duplex)

3.การสื่อสารข้อมูลใน 2 ทิศทางตลอดเวลา : ฟูลดูเพล็ก (Full-Duplex)



รูป2.7 แสดงโครงสร้างการสื่อสารข้อมูลแบบฟูลดูเพล็ก (Full-Duplex)

ซิมเพล็ก (Simplex) : เป็นการสื่อสารข้อมูลในทิศทางใดก็จะใช้ทิศทางนั้นตลอดไปไม่มีการเปลี่ยนแปลงทิศทาง เช่นการส่งกระจายเสียงของสถานีวิทยุไปยังเครื่องรับ หรือการส่งข้อมูลไปยังวิทยุติดตามตัว

ฮาฟดูเพล็ก (Half-Duplex) เป็นการสื่อสารข้อมูลใน 2 ทิศทาง แต่ขณะเวลาหนึ่งนั้น สัญญาณจะไปได้ในทิศทางเดียวเท่านั้น ดังนั้นอุปกรณ์แต่ละตัวที่จะเชื่อมต่อหรือการสื่อสารข้อมูลในลักษณะนี้ จะต้องได้ทั้งตัวรับและตัวส่ง ซึ่งมีชื่อว่า ทรานซิฟเวอร์ (Tranceiver) แบะจะต้องมีวงจรที่เลือก ณ.เวลานั้นจะทำงานเป็นตัวรับหรือตัวส่ง

ฟูลดูลเพล็ก (Full-Doplex) เป็นการสื่อสารข้อมูลที่คล้ายกับการสื่อสารแบบฮาฟดูเพล็ก (Half-Duplex) เพียงแต่ในขณะเวลาเดียวกันสามารถที่จะทำหน้าที่ได้ทั้งรับและส่งในเวลาเดียวกันได้

การสื่อสารข้อมูลแบบอนุกรมตามมาตราฐาน RS-232C

การสื่อสารข้อมูลแบบอนุกรมที่ใช้งานอยู่ในปัจจุบันนี้ได้มีการกำหนดมาตราฐานการรับ-ส่งข้อมูลไว้หลายแบบด้วยกัน แต่ที่ได้รับความนิยมนำไปใช้กันอย่างแพร่หลาย คือการสื่อสารข้อมูลแบบอนุกรมตามมาตราฐาน RS-232C เหตุที่เป็นที่นิยมเนื่องจากเป็นการสื่อสารข้อมูลที่ใช้ในเครื่องคอมพิวเตอร์ (Computer) IBM PC ซึ่งเป็นเครื่องคอมพิวเตอร์ที่ใช้อย่างแพร่หลายจากอดีตมาถึงปัจจุบันมาตราฐานRS-232C

โครงสร้างการสื่อสารข้อมูลเป็นแบบจุดต่อจุดเท่านั้น โดยมีลักษณะสมบัติทางไฟฟ้าและทางกายภาพ ดังแสดงการดังรูป

รูป2.8 แสดงโครงสร้างลักษณะคุณสมบัติทางไฟฟ้าแบบไม่สมดุล

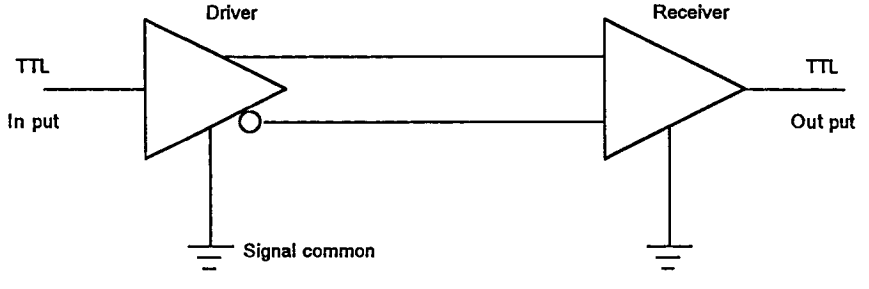
ข้อจำกัดของการสื่อสารข้อมูลแบบอนุกรมมาตราฐาน RS-232C

ข้อจำกัดนี้ประกอบกันด้วย 3 อย่างด้วยกันคือ

- ระยะทาง ในเคเบิล (Cable) RS-232C ค่าความจุไฟฟ้าระหว่างสายสัญญาณต่างๆ กับกราวด์ เป็นพารามิเตอร์หลักที่จำกัดระยะทางการใช้เคเบิลนั้น ตามข้อกำหนดมีว่าค่าความจุไฟฟ้าที่มองไปจากวงจรขับ (Driver) จะต้องไม่เกิน 2500pF การเพิ่มระยะทางของเคเบิล ก็จะเพิ่มค่าความจุไฟฟ้าที่มาเป็นภาระของวงจรขับ ธรรมดาที่ใช้เคเบิลที่ใช้อยู่จะมีความจุไฟฟ้า 2500pF ที่ความยาวประมาณ 15 เมตร
- อัตราการส่งข้อมูล จะต้องไม่เกิน 20,000บิท/วินาที ปัญหานี้เกี่ยวข้องกับความต้านทานกับค่าความจุของสายเคเบิล (Cable) เช่นกัน RS-232C นั้นกำหนดค่า ความต้านทานอินพุท (In put) ของวงจรภาครับค่อนข้างสูง 3000-7000 Ω
- สัญญาณรบกวน สามารถทำให้เกิดข้อมูลผิดพลาด ซึ่งเกิดจากแหล่งกำเนิดภายนอกเช่น มอเตอร์ไฟฟ้า สถานีวิทยุ และจากแหล่งกำเนิดภายใน เช่นความต้านทานไม่สมบูรณ์ การกระจายของกระแสจากสายสัญญาณต่างๆ

การสื่อสารข้อมูลแบบอนุกรมตามมาตราฐาน RS-422A

ในการออกแบบการสื่อสารข้อมูลที่กล่าวมา ได้มีการพยายามที่จะออกแบบให้การสื่อสารข้อมูลได้รวดเร็วยิ่งขึ้น และมีระยะทางในการสื่อสารข้องมูลได้มากขึ้นด้วย ซึ่งที่ผ่านมาการสื่อสารข้อมูลมาตราฐาน RS-232C ได้ออกแบบมาเพื่อใช้เชื่อมโยงกับโมเด็ม (Modem) เท่านั้นจึงไม่ได้คำนึงถึงความเร็วและระยะทางในการสื่อสาร แต่ในปัจจุบันได้มีการออกแบบมาเพื่อรองรับความต้องการของการใช้งาน ที่ต้องการให้รับ-ส่งข้อมูลได้ไกลยิ่งขึ้น คือมาตราฐาน RS-422A ซึ่งจะใช้สัญญาณแบบดิฟเฟอร์เรนเชียล (Differentail) ดังในรูป

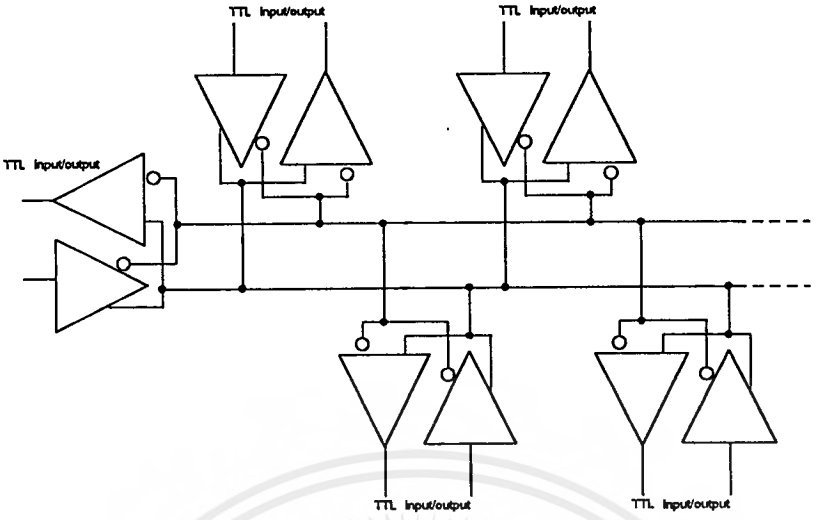


รูป 2.9 แสดงโครงสร้างลักษณะคุณสมบัติทางไฟฟ้าแบบสมดุล

หลักการก็คือสัญญาณที่จะรับ-ส่งจะเป็นการเปรียบเทียบระหว่างสัญญาณ 2 เส้นเปรียบเทียบกับมาตราฐาน RS-232C ที่สัญญาณทุกสัญญาณทุกสัญญาณจะเทียบกราวด์ (Ground) ซึ่งในการสื่อสารในระยะทางไกลๆ แล้วสัญญาณจะถูกลดทอนไปถึงจุดๆหนึ่งสัญญาณนั้นก็จะผิดพลาดไปจากความเป็นจริงก็จะทำการรับ-ส่งผิดพลาดขึ้น แต่สำหรับสัญญาณดิฟเฟอร์เรนเชียล (Differentail) แล้วการลดทอนของสัญญาณจะไปลดทอนทั้งสองสายด้วยค่าที่เท่ากันหรือใกล้เคียงกันและความแตกต่างของสัญญาณทั้งสองเส้น จากตัวส่งไปยังตัวรับก็ยังคงมีค่าเท่าเดิมหรือแตกต่างกันเล็กน้อย จึงทำให้ผลการลดลงของสัญญาณที่ระยะทางการสื่อสารไกลมีผลต่อสัญญาณดิฟเฟอร์เรนเชียล (Differentail) มีค่าน้อยกว่า การสื่อสารข้อมูลแบบนี้จึงสามารถส่งข้อมูลนี้จึงสามารถส่งข้อมูลได้ไกลกว่าและอัตราการสื่อสารข้อมูลสูงกว่า

การสื่อสารข้อมูลแบบอนุกรมตามมาตราฐาน RS-485

การสื่อสารข้อมูลตามมาตราฐานที่กล่าวมาข้างต้น นั้นเป็นมาตราฐานการสื่อสารข้อมูลในแบบที่ใช้กันระหว่างอุปกรณ์ หรือจุดต่อจุด (Point-to-Point) ส่วน RS-422A นั้นเป็นมาตราฐานที่พัฒนามาจาก RS-232C ให้ได้ระยะทางไกลขึ้น และอัตราการสื่อสารมากขึ้น แต่ยังเป็นการสื่อสารข้อมูลระหว่างอุปกรณ์หนึ่งไปยังอุปกรณ์อื่นๆ ได้อัตราสูงสุด 10 เท่าตัว ไม่สามารถส่งย้อนกลับจากอุปกรณ์ 10 ตัวได้ หรือกล่าวได้ว่าอุปกรณ์การสื่อสารข้อมูลตามมาตราฐาน RS-422A นั้นเป็นการสื่อสารข้อมูลแบบซมเพล็ก (Simplex) คือทิศทางการสื่อสารข้อมูลเป็นทางเดียวตลอดเวลา ดังนั้นถ้าต้องการออกแบบระบบบให้เป็นโครงข่ายข้อมูลก็ไม่สามารถที่จะออกแบบได้ จึงได้มีการพัฒนามาตราฐานการสื่อสารข้อมูลแบบใหม่เพื่อรองรับการต้องการนี้ คือมาตราฐาน RS-845 ซึ่งเป็นมาตราฐานที่อาศัยหลักการของสัญญาณแบบดิฟเฟอร์เรนเชียล (Differentail)เช่นเดียวกับมาตราฐาน RS-422A แต่การสื่อสารข้อมูลได้ทั้งสองทิศทางในสายสัญญาณเพียงคู่เดียว ซึ่งการสื่อสารข้อมูลแบบฮาฟดูเพล็ก (Half-Duplex) จากผลการทดลองการใช้สัญญาณในลักษณะดิฟเรนเชียล (Differrential) นี้จะทำให้ระยะทางและความเร็วในการสื่อสารข้อมูลเช่นเดียวกับมาตราฐาน RS-422A แต่มาตราฐาน RS-485 สามารถที่จะสื่อสารกันระหว่างอุปกรณ์ทั้งการรับและกาส่งได้หลายจุดที่เรียกว่ามัลติพ้อยท์ คอมมิวนิเคชั่น (Multipoint communication) โครงสร้างในการสื่อสารข้อมูลแบบ RS-485 ดังแสดงดังรูป



รูป2.10 แสดงโครงสร้างการสื่อสารของมูลแบบอนุกรมตามมาตราฐาน RS-485

ตารางเปรียบมาตราฐานการสื่อสารข้อมูลของ EIA

พารามิเตอร์ (Parameter)	RS-232C	RS-423-A	RS-422A	RS-485
โหมดการทำงาน	Single-ended	Single-ended	Differential	Differential
ความยาวของคู่สายและตัวส่งที่รับได้	1 ตัวส่ง 1 ตัวรับ	1 ตัวส่ง 10 ตัวรับ	1 ตัวส่ง 10 ตัวรับ	32 ตัวส่ง 32 ตัวรับ
ความยาวของคู่สายสัญญาณรับส่งข้อมูล	50 ฟุต	4000 ฟุต	4000 ฟุต	4000 ฟุต
อัตราการส่งข้อมูลสูงสุด(บิทต่อวินาที)	20 k	100 k	10 M	10 M
แรงดันไฟฟ้าโหมดร่วมสูงสุด	±2.5 V	± 6 V	6 V -2.5 V	12 -7 V
Driver load	±5 V ต่ำสุด ±15 V สูงสุด	±3.6V ต่ำสุด ±6.0 V สูงสุด	±2 V ต่ำสุด	±1.5.V ต่ำสุด
Driver load Ω	3 k ถึง 7 k	450 ต่ำสุด	100 ต่ำสุด	60 ต่ำสุด
Drive slew rate	3 K/uS สูงสุด		NA	NA
กระแสลิมิตเมื่อกระแสลัดวงจร	500mA ลัดวงจรกลับVCCหรือGND	150mA ลัดวงจรกลับกับGND	150mA ลัดวงจรกลับกับGND	150mA ลัดวงจรกลับกับGND 150mA ลัดวงจรกลับกับ8Vกับ12V
ความต้านทานเอาท์พุท(output)ของตัวส่ง Ω	NA - power on 300 - power off	NA - power on 60K - power off	NA - power on 60K - power off	120 K power on, off
ความต้านทานอินพุท(input)ของตัวรับ Ω	3 K ถึง 7K	4K	4K	12K
ความไวของตัวรับ	±3V	± 200 mA	± 200 mA	± 200 mA

<u>คุณสมบัติของการสื่อสารข้อมูลแบบ RS-485</u>

คุณสมบัติของการสื่อสารข้อมูลแบบ RS-485ที่แตกต่างจาก RS-422A

คุณลักษณะเฉพาะของตัวส่ง RS485

■ ตัวส่ง 1 ตัว สามารถที่จะขับโหลด (Load) ได้สูงสุด 32 ตัว โดยที่โหลด (Load) 1 ชุด ประกอบด้วยตัวส่ง 1 ตัวและตัวรับหนึ่งตัวและค่าของความต้านทานที่ตกคร่อมระหว่างคู่สายสัญญาณมีค่า 60Ω

■ เอ้าท์พุท (Out put) ของตัวส่งในสภาวะออฟ (Off) มีกระแสรั่วไหลไม่เกิน 100uA ในช่วงแรงดันไฟฟ้า โหมดร่วมระหว่าง -7 ถึง +12 V

■ เอาท์พุท (Out put) ของตัวส่งให้แรงดันไฟฟ้าเอาท์พุท (Out put) 1.5 V ถึง 5V ในช่วงแรงดันไฟฟ้าร่วมระหว่าง -7 ถึง +12 V

■ ตัวส่งมีวงจรป้องกันตัวเองที่ส่วนเอาท์พุท (Out put) ในกรณีที่ตัวส่งหลายๆ ส่งข้อมูลออกมาพร้อมๆกันคุณสมบัติของตัวรับ RS-485

■ ค่าความต้านทานอินพุท (Input) มีค่าสูง โดยมีค่าไม่น้อยกว่า 12 กิโลโอหม์ (Kilohm)

■ ตัวรับมีค่าแรงดันไฟฟ้าอินพุทโหมดร่วม ระหว่าง ค่า -7 V ถึง +12 V ตัวรับสามารถตอบสนองต่อสัญญาณที่แตกต่างกันจากสัญญาณที่โหมดร่วมได้±200mA(น้อยที่สุด)

ภาคไดรเวอร์

ภาคนี้ประกอบไปด้วยไอซีหลายๆตัวนำมาประกอบกันเพื่อไปควบคุมการทำงานของอุปกรณ์ต่างๆ ภายในตัวเครื่อง ซึ่งมี ดังนี้คือ

MAX691

เป็น Microprocessor Supervisory Circuits ใช้สำหรับจัดการเกี่ยวกับการ มอนิเตอร์เพาเวอร์ซัพพลาย (Power Supply Monitorning) และการควบคุมแบตเตอรี่ และ watcholog timer การสำรองข้อมูล โดยการรับแรงดันจากแบตเตอรี่ลิเธี่ยม 3 โวลท์ ที่ขา VBATT และแรงดัน VOUT จาก MAX691 ต่อที่ขา VTC ที่ขา VOUT ที่ระดับแรงดันปกติ 5 โวลท์ จะเสมือนถูกต่อกับ VCC ภายใน MAX691 ดังนั้น แบตเตอรี่จะถูกใช้งานในช่วงไฟดับเท่านั้น

Watchdog ticer

เป็นวงจรที่ทำหน้าที่ตรวจสอบการทำงานของระบบ Microprocessor ว่าทำงานเป็นสภาวะปกติหรือไม่ ถ้าระบบทำงานผิดปกติหรือเกิดอาการแฮงค์ วงจรส่วนนี้จะทำการ Reset CPU ให้เริ่มทำงานใหม่อีกครั้งหลักการทำงานคือ CPU จะต้องส่งสัญญาณไปกระตุ้นที่ขา WDI (Watchdog input) ของ MAX691 โดยที่ขา OSC IN และ OSC SEL ของ MAX691 ไม่ได้ใช้งานปล่อยลอยเอาไว้ CPU จะต้องเปลี่ยนสภาวะ (Toggle) ที่ขา WDI ทุก 1.6 วินาที โดยใช้คำสั่ง P2.7 เพื่อให้แน่ใจว่า Software ทำงานได้ถูกต้องถ้า Hardware หรือ Software เกิดทำงานผิดพลาด ซึ่งเป็นผลให้สภาวะที่ขา WDI ไม่เปลี่ยนแปลงตามที่กำหนด MAX691 จะส่งสัญญาณ Reset เพื่อ Reset ให้ CPU กลับไปทำงานใหม่อีกครั้ง และที่ขา Reset นี้จะส่ง Pluse Reset ออกมาทุกๆ 1.6 วินาที จนกว่าจะมีการเปลี่ยนสภาวะที่ขา WDI อีกครั้ง

IC DS1202 (Real time colck/calender)

IC ตัวนี้จะทำหน้าที่เป็นเวลาและปฏิทิน ซึ่งจะให้ข้อมูลเกี่ยวกับเวลา เช่น วินาที นาที ชั่วโมง และวัน เดือน ปี ไมโครคอนโทรเลอร์ ที่ต้องการนำข้อมูลนี้ไปใช้งาน ทำได้ด้วยวิธีการอนุกรม โดยมีสัญญาณนาฬิกา เพื่อกำหนดจังหวะการส่งข้อมูล ใน IC ประกอบด้วย static RAM ขนาด 24 Byte และใช้ Oscilltor ซึ่งมี Cytral ความถี่ 32.768 kHz

IC93C46 (memory)

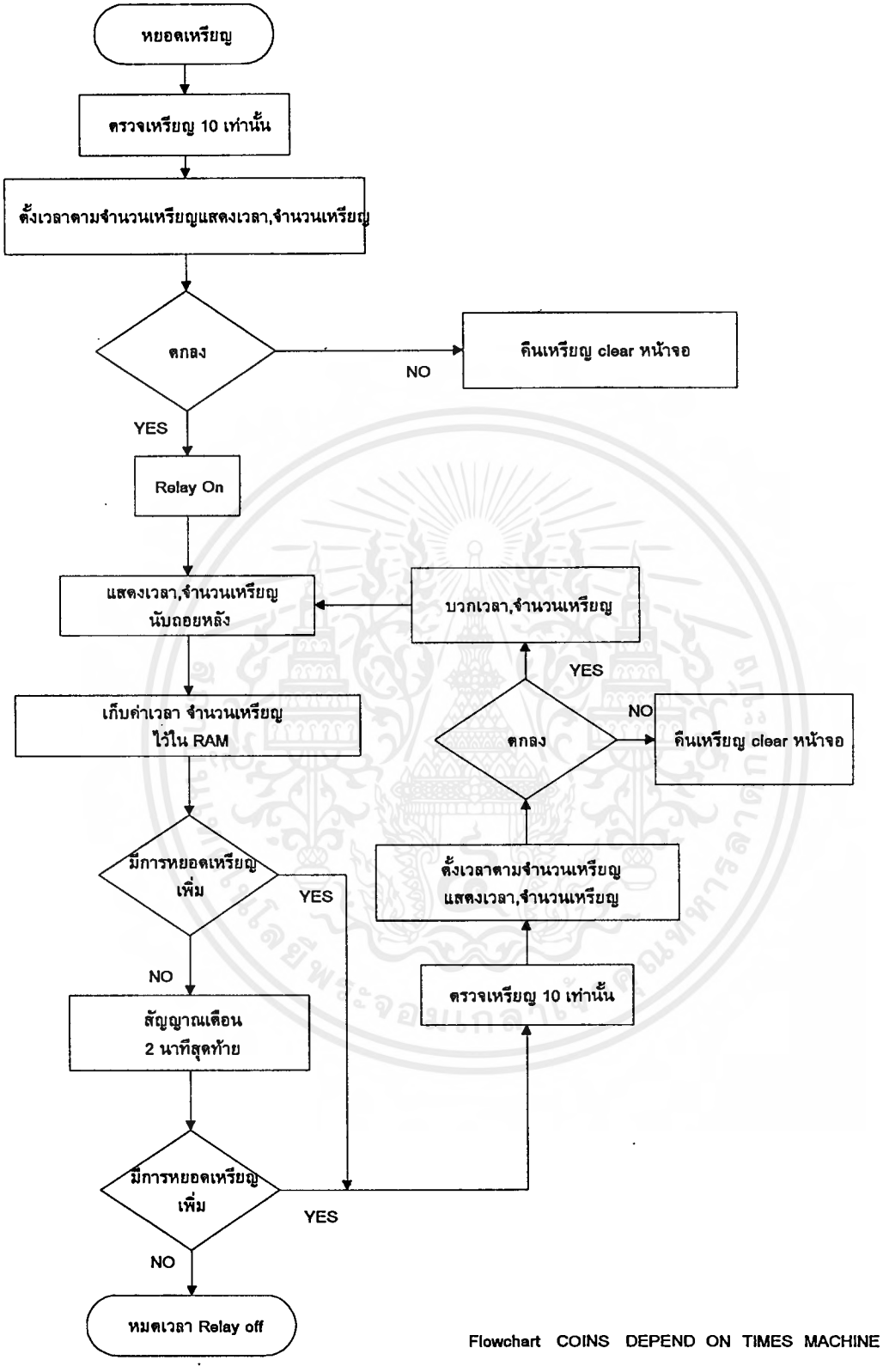
เป็น EEPROM โดยการบันทึกข้อมูลมีส่วนแสดงผลเป็น LCD มีการจัดเก็บข้อมูลเมื่อไม่มีไฟเลี้ยงหรือ Power Supply และเราสามารถนำโปรแกรมที่เราบันทึกไว้ในไอซีตัวนี้ออกมาใช้งานได้ เพื่อเป็นการเก็บข้อมูลที่จำเป็นของตัวเครื่องไว้

IC 4N33

เป็นออปโตคัปเปิ้ล (อุปกรณ์เชื่อมต่อผ่านแสง) ออปโตคัปเปิ้ลหรือที่เรียกอีกชื่อว่าออปโตไอโซเลเตอร์เป็นอุปกรณ์สารกึ่งตัวนำที่ประกอบด้วยตัวกำเนิดแสง ท่อนำแสงให้แสงผ่านระยะใกล้ๆ และตัวรับแสงประกอบรวมในตัวถังเดียวกัน โดยทั่วไปเป็นแบบ DIP ออปโตคัปเปิ้ลที่ใช้ในการเปลี่ยนสัญญาณแสงแล้วจึงแปลงกลับจากแสงคือเป็นไฟฟ้า

การทำงานของเครื่อง

1. ใส่เหรียญ
2. ตรวจขนาดของเหรียญ โดยจะ Sensor ขนาดซึ่งจะเอาเฉพาะเหรียญ 10 บาท เท่านั้น ถ้าเป็นเหรียญขนาดอื่นจะถูกคัดออก
3. จอแสดงผลจะแสดงจำนวนเหรียญที่หยอดมาและบอกว่าตามจำนวนเหรียญที่ ใส่เข้ามานั้นจะได้รับเวลาทำงานทั้งหมดเท่าไหร่
4. หน้าจอจะถามความแน่ในว่าต้องการที่จะให้เครื่องทำงานจริงรึเปล่าถ้าหาก กด cancle เครื่องจะทำการคืนเหรียญทางช่องคืนเหรียญ แต่ถ้ากด yes แสดงความแน่ใจ เครื่องก็จะเก็บเหรียญเข้าช่องเก็บเหรียญและจะทำการ Run เครื่องให้ทำงานหน้าจอก็จะแสดงจำนวนเหรียญและเวลาก็จะเริ่มนับถอยหลัง และเครื่องจะทำการเก็บข้อมูลทั้งหมดทั้งจำนวนเหรียญและเวลาที่นับถอยหลังอยู่ เข้าไปเก็บใน RAM เพื่อกรณีที่มีการหยอดเหรียญเข้ามาเพิ่ม เครื่องจะทำการบวกจำนวนเหรียญและเวลาที่จะได้รับตามจำนวนเหรียญมาบวกเข้ากับเวลาที่กำลัง Run อยู่ ขณะนั้น
5. เมื่อเหรียญที่ใส่เข้ามาตอนแรกจะหมดเวลาการใช้งานของเครื่องใน 2 นาทีสุดท้าย เครื่องจะทำการส่งสัญญาณเตือนว่าเวลาจะหมดแล้ว และเวลาจะนับถอยหลังไปเรื่อยๆ ถ้าไม่มีการหยอดเหรียญเพิ่ม จนเวลาหมดเครื่องก็จะหยุดทำงาน
6. กรณีที่มีการหยอดเหรียญเพิ่มเข้ามาไม่ว่าเวลาใด ในขณะที่เวลากำลังนับถอยหลังอยู่ หรือหลังจากสัญญาณเตือนแล้วต้องการหยอดเหรียญเพิ่ม เครื่องก็จะทำการตรวจขนาดก่อนแล้วก็จะถามความแน่ใจ ถ้ากด cancle เครื่องจะคืนเหรียญแต่ถ้ากด Yes เครื่องจะเก็บเหรียญเข้าช่องเก็บเหรียญและจะนำจำนวนเหรียญและเวลาที่จะได้รับจากจำนวนของเหรียญที่หยอดเข้ามาใหม่บวกเข้ากับข้อมูลที่มีอยู่ใน RAM เครื่องก็จะทำงานต่อจนกว่าจะหมดเวลาและไม่มีการหยอดเหรียญเพิ่ม

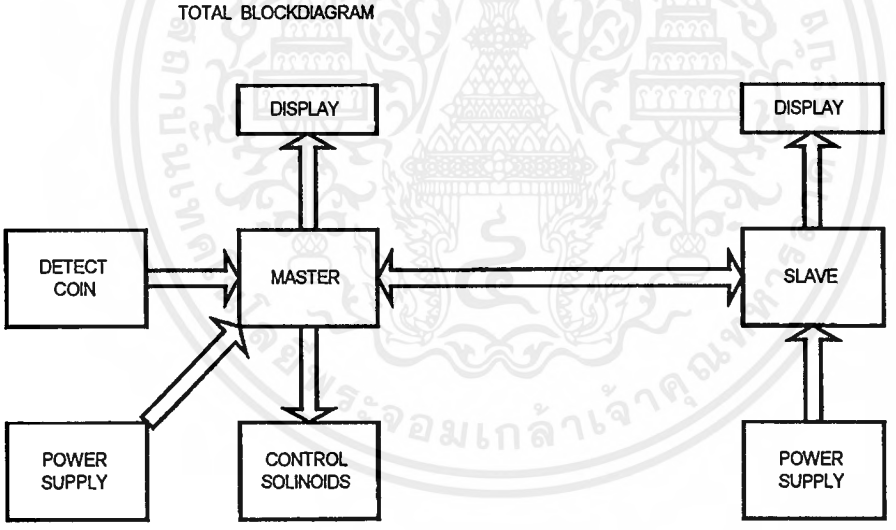


รูปที่ 2.11 Flowchart การทำงานของเครื่อง

บทที่ 3

การออกแบบ

โครงงานเครื่องตั้งเวลาหยอดเหรียญนี้ การออกแบบทั้งหมดเราเริ่มจากการคิดหลักการทั้งหมดของเครื่องว่าน่าจะมีส่วนประกอบอะไรบ้าง และเราสามารถนำอุปกรณ์และวงจรอะไรที่สามารถทำงานได้ตามที่เราต้องการ มาประกอบในเครื่องเราบ้าง โดยแนวความคิดหลักๆ จะมีอยู่ 2 ส่วนใหญ่ๆ คือ ภาคไมโครคอนโทรลเลอร์ และภาคกลไก โดยภาคไมโครคอนโทรลเลอร์ หรือภาคคอนโทรลเลอร์ เราจะนำมาควบคุมการทำงานทั้งหมดของเครื่องรวมทั้งควบคุมภาคกลไกด้วย และภาคกลไกก็จะทำหน้าที่ ควบคุมเหรียญที่ผู้ใช้หยอดเข้ามาในเครื่อง โดยมีการทำงานต่างๆ เช่น คัดเหรียญทิ้ง คืนเหรียญ แล้วก็เก็บเหรียญเข้าที่เก็บเหรียญ โดยในตอนแรกเราได้ ออกแบบมาเป็นบล็อกไดอะแกรม (Blockdiagram) คร่าวๆ ดังนี้



รูปที่ 3.1 บล็อกไดอะแกรมรวม

จากบล็อกไดอะแกรมของเครื่องจะมีภาคต่างๆ ที่ทำงานดังต่อไปนี้

- ภาคไมโครคอนโทรลเลอร์หลัก (Master)
- ภาคไมโครคอนโทรลเลอร์รอง (Slave)
- ภาคตรวจนับจำนวนเหรียญ (Detect coin)
- ภาคควบคุมอุปกรณ์ระบบกลไก (Control solinoids)

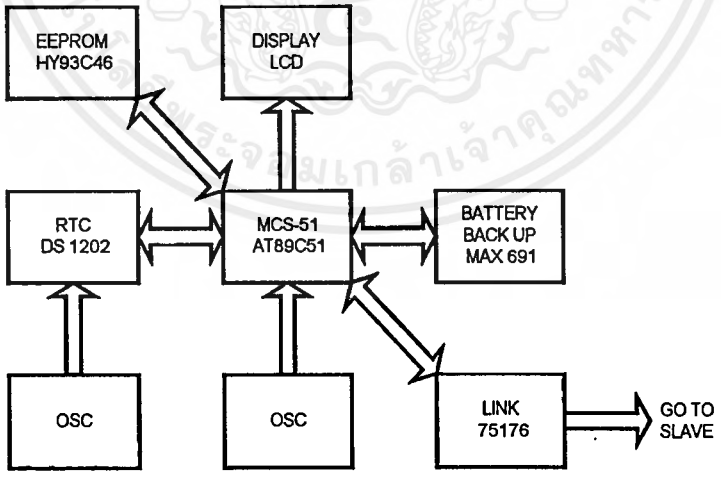
- ภาคจ่ายไฟ (Power supply)
- ภาคแสดงผล (Display)

โดยการออกแบบและการทำงานของภาคต่างๆ มีดังนี้

3.1 ภาคไมโครคอนโทรลเลอร์หลัก (Master)

ซึ่งในภาคนี้การออกแบบก็จะเริ่มจากการที่เราจะต้องเลือก ว่าจะใช้ไมโครคอนโทรลเลอร์เบอร์อะไร มาเป็นตัวควบคุมหลักในภาคนี้ ซึ่งเราได้ใช้ไอซีไมโครคอนโทรลเลอร์ตระกูล MCS51 มาเป็นตัวควบคุมโดยใช้เบอร์ AT89C51 ซึ่งมีคุณสมบัติคือเป็นไมโครคอนโทรเลอร์ในตระกูล MCS-51 มีหน่วยความจำชนิด FLASH MEMORY ขนาด 4 kByte สามารถโปรแกรมได้ 1,000 ครั้ง มีหน่วยความจำแบบ RAM 8 bit ขนาด 128 Byte (Internal RAM) และทำงานที่แรงดัน 2.7-6 โวลท์ ใช้ความถี่ของสัญญาณนาฬิกา (clock) ตั้งแต่ 0-24 MHz มี Port input/output ขนาด 32 bit (Port 1 และ Port 3) แต่ละ Port มีความต้านทานพูลอัพภายใน ยกเว้น P1.0 และ P1.1 ต้องต่อความต้านทานพูลอัพภายนอกเอง สามารถป้องกันการคัดลอกข้อมูล มีวงจร Timmer และ Counter ขนาด 16 bit 2 channal มีสัญญาณ Interrupt 5 แหล่ง สามารถแบ่งระดับความสำคัญได้ 2 ระดับ

ซึ่งในภาคนี้มีบล็อกไดอะแกรมดังนี้คือ



รูปที่ 3.2 บล็อกไดอะแกรมภาคไมโครคอนโทรลเลอร์หลัก

จากบล็อกไดอะแกรมดังแสดงในรูปที่ 2.3 จะเห็นว่าในภาคนี้จะประกอบไปด้วยไอซีหลายๆ ตัวซึ่งแต่ละตัวจะมีหน้าที่ต่างๆ ดังนี้คือ

1. MAX691

เป็น Microprocessor Supervisory Circuits ใช้สำหรับจัดการเกี่ยวกับการ มอนิเตอร์เพาเวอร์ซัพพลาย (Power Supply Monitorning) และการควบคุมแบตเตอรี่ และ watcholog timer การสำรองข้อมูล โดยการรับแรงดันจากแบตเตอรี่ลิเธี่ยม 3 โวลท์ ที่ขา VBATT และแรงดัน VOUT จาก MAX691 ต่อที่ขา VTC ที่ขา VOUT ที่ระดับแรงดันปกติ 5 โวลท์ จะเสมือนถูกต่อกับ VCC ภายใน MAX691 ดังนั้น แบตเตอรี่จะถูกใช้งานในช่วงไฟดับเท่านั้น

Watchdog ticer

เป็นวงจรที่ทำหน้าที่ตรวจสอบการทำงานของระบบ Microprocessor ว่าทำงานเป็นสภาวะปกติหรือไม่ ถ้าระบบทำงานผิดปกติหรือเกิดอาการแฮงค์ วงจรส่วนนี้จะทำการ Reset CPU ให้เริ่มทำงานใหม่อีกครั้งหลักการทำงานคือ CPU จะต้องส่งสัญญาณไปกระตุ้นที่ขา WDI (Watchdog input) ของ MAX691 โดยที่ขา OSC IN และ OSC SEL ของ MAX691 ไม่ได้ใช้งานปล่อยลอยเอาไว้ CPU จะต้องเปลี่ยนสภาวะ (Toggle) ที่ขา WDI ทุก 1.6 วินาที โดยใช้คำสั่ง P2.7 เพื่อให้แน่ใจว่า Software ทำงานได้ถูกต้องถ้า Hardware หรือ Software เกิดทำงานผิดพลาด ซึ่งเป็นผลให้สภาวะที่ขา WDI ไม่เปลี่ยนแปลงตามที่กำหนด MAX691 จะส่งสัญญาณ Reset เพื่อ Reset ให้ CPU กลับไปทำงานใหม่อีกครั้ง และที่ขา Reset นี้จะส่ง Pluse Reset ออกมาทุกๆ 1.6 วินาที จนกว่าจะมีการเปลี่ยนสภาวะที่ขา WDI อีกครั้ง

2. IC DS1202 (Real time colck/calender)

IC ตัวนี้จะทำหน้าที่เป็นเวลาและปฏิทิน ซึ่งจะให้ข้อมูลเกี่ยวกับเวลา เช่น วินาที นาที ชั่วโมง และวัน เดือน ปี ไมโครคอนโทรเลอร์ ที่ต้องการนำข้อมูลนี้ไปใช้งาน ทำได้ด้วยวิธีการอนุกรม โดยมีสัญญาณนาฬิกา เพื่อกำหนดจังหวะการส่งข้อมูล ใน IC ประกอบด้วย static RAM ขนาด 24 Byte และใช้ Oscilltor ซึ่งมี Cytral ความถี่ 32.768 kHz

การทำงานของไมโครคอนโทรเลอร์

เมื่อเปิด Switch CPU จะอ่านข้อมูลจาก Memory flag ของเวลา และวันที่จากนั้นทำการตรวจสอบว่าต้องแสดงอะไรออกที่ Display บ้าง โดยกำหนด memory โดยใช้ IC96C46 เป็น EPROM โดยการบันทึกข้อมูลมีส่วนแสดงผลเป็น LCD มีการจัดเก็บข้อมูลเมื่อไม่มีไฟเลี้ยงหรือ Power Supply

Micro controller จะรับข้อมูลจาก RTC DS1202 และรับการตั้งเวลาข้อมูลโดย keyboard Matrix 3X2 เพื่อ set ฟังค์ชั่นการทำงาน

3. HY93C46

เป็นไอซีอีสแคว์พรอม EEPROM ซึ่งจะทำหน้าที่เป็นหน่วยความจำให้กับไมโครคอนโทรลเลอร์ โดยจะทำหน้าที่เก็บค่าต่างๆ ที่เกิดขึ้นในการทำงานของเครื่องตั้งเวลาหยอดเหรียญนี้เช่น ค่าเวลาสูงสุดที่รับเข้ามาได้รวม ค่าจำนวนเหรียญสูงสุดที่ผู้ใช้ทำการหยอดเข้ามา

4. 75176

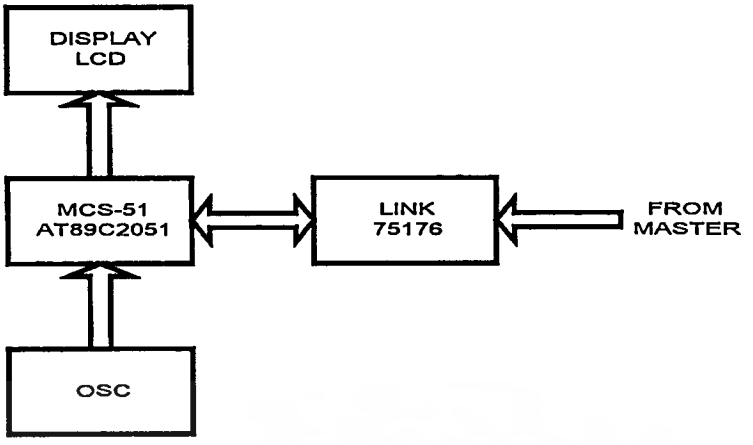
เป็นไอซีที่ทำการส่งผ่านข้อมูลระหว่างไมโครคอนโทรลเลอร์2 ตัวในที่นี้คือไมโครคอนโทรลเลอร์ตัวหลัก AT89C51 และไมโครคอนโทรลเลอร์ตัวรอง AT89C2051 ซึ่งไอซีตัวนี้เป็นไอซีที่เชื่อมต่อข้อมูลโดยเฉพาะของไมโครคอนโทรลเลอร์ตระกูลนี้ ในการทำงานของไอซีตัวนี้ในเครื่องหยอดเหรียญตั้งเวลานี้จะทำการนำข้อมูลจากไมโครคอนโทรลเลอร์หลัก ส่งไปยังไมโครคอนโทรลเลอร์รองเพื่อทำการแสดงผลที่เป็นข้อมูลของเครื่องออกทางหน้าจอแสดงผลทางเครื่องมอนิเตอร์ของเครื่องหยอดเหรียญซึ่งมีไว้สำหรับเจ้าของกิจการ ในกรณีที่เรานำไปใช้งาน นั้นก็หมายถึงว่าในเครื่องตั้งเวลาหยอดเหรียญเครื่องนี้จะมีการแสดงผลถึง 2 ชุดแสดงผล

5. Display

ในภาคแสดงผลของเครื่องตั้งเวลาหยอดเหรียญนี้เป็นภาคแสดงผลที่ออกมาจากตัวไมโครคอนโทรลเลอร์ ซึ่งเราใช้ ตัวแสดงผลแบบ LCD

3.2 ภาคไมโครคอนโทรลเลอร์รอง (Slave)

ในภาคนี้เราก็จำเป็นที่จะต้องใช้ไมโครอคนโทรลเลอร์มาควบคุมเหมือนกันซึ่งในที่นี้เราใช้ไมโครคอนโทรลเลอร์เบอร์ AT89C2051 ซึ่งคุณสมบัติก็เหมือนกันกับไมโครคอนโทรลเลอร์เบอร์ AT89C51 เพียงแต่ว่ามีขนาดเล็กกว่าและมีขาอยู่เพียง 20 ขา และมีพอร์ทการทำงานที่น้อยกว่าแต่เราก็ทำการแสดงผลโดย LCD เหมือนเดิมโดยต่อออกมาจากพอร์ทที่มีอยู่ในตัวไมโครคอนโทรลเลอร์ตัวนี้ โดยจะมีบล็อกไดอะแกรมของภาคนี้ดังนี้



รูปที่ 3.3 บล็อกไดอะแกรมภาคไมโครคอนโทรลเลอร์รอง

จะเห็นได้ว่าลักษณะของบล็อกไดอะแกรมจะคล้ายๆ กับภาคไมโครคอนโทรลเลอร์หลัก เพียงแต่จะไม่มี ไอซี MAX691 ซึ่งเป็นแบตเตอรี่ต่อเพื่อรักษาข้อมูลเวลาปิดเครื่อง ไอซี DS 1202 ซึ่งเป็นไอซีสร้างฐานเวลาแต่ว่าไมโครคอนโทรลเลอร์ตัวนี้จะดึงข้อมูลของฐานเวลามาจากไมโครคอนโทรลเลอร์หลัก ซึ่งจะทำให้ทั้งสองภาคใช้ฐานเวลาเดียวกัน และก็ไอซี HY94C46 ซึ่งเป็นอีสแควร์พรอม ซึ่เราก็ใช้ดึงข้อมูลที่เป็นส่วนหน่วยความจำที่เก็บไว้มาจากไมโครคอนโทรลเลอร์หลัก

3.3 ภาคตรวจนับจำนวนเหรียญ (Detect coin)

ในภาคนี้จะทำการนับเหรียญที่ผู้ใช้หยอดเข้ามา โดยใช้เซ็นเซอร์ (sensor) ซึ่งจะทำการนับจำนวนเหรียญที่เข้ามาแล้วแสดงผลออกทางจอแสดงผล LCD แล้วจะแสดงเวลาซึ่งเราจะตั้งไว้ว่าเหรียญที่หยอดเข้ามา 1 เหรียญจะได้เวลากี่นาที ไมโครคอนโทรลเลอร์จะทำการคูณแล้วแสดงผลออกมาทางหน้าจอด้วย ตัวตรวจจับเหรียญนี้จะทำการนับเหรียญที่เข้ามาที่ละเหรียญเมื่อทำการนับจนถึง 8 เหรียญแล้วจะทำการสั่งงานไปที่ภาคควบคุมระบบกลไก ให้ทำการเปิดช่องด้านบนสุดของเครื่องออก ซึ่งเมื่อมีการหยอดเหรียญเข้ามาอีก จะทำให้เหรียญตกลงไปในช่องคืนเหรียญ คือผู้ใช้ไม่สามารถทำการหยอดเหรียญได้อีกต่อ จะต้องมีการตอบตกลงว่าจะให้เครื่องทำงานตามจำนวนเหรียญและจำนวนเวลาที่ แสดงออกมาทางหน้าจอหรือไม่ ถ้าผู้ใช้ตอบตกลงเครื่องจะทำการเก็บเหรียญเข้าที่เก็บเหรียญ แต่ถ้าผู้ใช้ไม่ตอบตกลง เครื่องจะทำการคืนเหรียญโดยเหรียญจะหล่นลงไปในช่องคืนเหรียญ แล้วผู้ใช้สามารถทำการหยอดเข้ามาใหม่ได้

ในภาคนี้เราใช้ไอซีไมโครคอนโทรลเลอร์ AT89C2051 มาเป็นตัวตรวจจับและนับจำนวนเหรียญที่ผู้ใช้ทำการหยอดเข้ามา โดยตัวเซ็นเซอร์ที่ใช้จะใช้ตัวเซ็นเซอร์แบบโฟโต้ไดโอด โดยจะทำการตรวจจับเหรียญที่ผู้ใช้หยอดเข้ามา และเมื่อมีการหยอดเข้ามาครบ 8 เหรียญ ตัวไมโครคอนโทรลเลอร์ก็จะไปสั่งงานให้กลไกทำงานเพื่อที่จะทำให้ไม่สามารถหยอดเหรียญได้อีกต่อไป

3.4 ภาคควบคุมอุปกรณ์ระบบกลไก (Control solinoids)

ในภาคนี้จะทำหน้าที่ไปควบคุมระบบกลไกของเครื่องตั้งเวลาหยอดเหรียญนี้ โดยเราจะใช้โซลีนอยด์ (Solinoid) ซึ่งการทำงานของตัวโซลีนอยด์นี้ จะทำงานโดยเมื่อมีการจ่ายไฟให้กับตัวโซลีนอยด์ ตัวโซลีนอยด์จะทำการดึงตัวมันเอง ซึ่งเราใช้หลักการนี้มาใช้ทำตัวคัดเหรียญทิ้งโดยการทำช่องว่างไว้ เป็นช่องแล้วใช้ตัวโซลีนอยด์นี้เป็นตัวกันเอาไว้ เมื่อเราจ่ายไฟให้กับโซลีนอยด์ ตัวโซลีนอยจะมีการดึงตัวเองเกิดขึ้น ช่องที่ปิดอยู่ก็จะถูกเปิด ทำให้เราสามารถคัดเหรียญไปเก็บไว้ หรือให้ไหลไปในที่ต่างๆ ที่เราทำเป็นร่องไว้ได้

โดยในภาคนี้จะมี ไอซีที่เรานำมาใช้งานอยู่ 2 ตัวคือ ไอซีเบอร์ 74LS245 ซึ่งเราจะใช้ทำหน้าที่เป็นตัวอินเวอร์ส (Inverst) ซึ่งเราจะนำไปใช้ประกอบกับโซลีนอยด์ซึ่งเอาไว้ในกรณีที่เราต้องการ กลับคุณสมบัติการทำงานของตัวโซลีนอยด์ ซึ่งในไอซีตัวนี้จะมีบัฟเฟอร์ (Buffer) ซึ่งจะไม่ทำให้การแสในการควบคุมโซลีนอยด์ลดลง ไอซีอีกตัวคือ ไอซีเบอร์ 4N33 ซึ่งมันคือไอซี ที่มีคุณสมบัติ เป็นออปโตคัปเปิ้ล (อุปกรณ์เชื่อมต่อผ่านแสง) ออปโตคัปเปิ้ลหรือที่เรียกอีกชื่อว่าออปโตไอโซเลเตอร์เป็นอุปกรณ์สารกึ่งตัวนำที่ประกอบด้วยตัวกำเนิดแสง ท่อนำแสงให้แสงผ่านระยะใกล้ๆ และตัวรับแสงประกอบรวมในตัวถังเดียวกัน โดยทั่วไปเป็นแบบ DIP ออปโตคัปเปิ้ลที่ใช้ในการเปลี่ยนสัญญาณแสงแล้วจึงแปลงกลับจากแสงคือเป็นไฟฟ้า ซึ่งเรานำมันมาเป็นตัวเพิ่มกระแสให้เพิ่มขึ้นเพื่อใช้ขับโซลีนอยด์ให้สามารถทำงานได้

3.5 ภาคจ่ายไฟ (Power supply)

ในภาคนี้เป็นภาคที่สำคัญมากเพราะหากไม่มีภาคนี้แล้วทั้งเครื่องก็ไม่สามารถที่จะทำงานได้ เพราะว่าภาคนี้จะทำการจ่ายไฟให้กับวงจรทุกวงจร ทุกๆ ภาคที่มีอยู่ในเครื่องนี้ ซึ่งในที่นี้เราใช้ไฟ ดีซี (DC) ขนาด 5 โวลท์ ซึ่งเราจะจ่ายไฟให้กับทุกวงจรทุกภาคด้วยแรงไฟที่เท่ากันหมดคือ 5 โวลท์ เราใช้วงจรเร็กกูเลท ในการสร้างแรงดันกระแสตรงขนาด 5 โวลท์ จากแหล่งจ่ายไปกระแสสลับที่เข้ามาขนาด 220 โวลท์จากแหล่งจ่ายไฟฟ้ากระแสสลับทั่วไปตามบ้านเรา

3.6 ภาคแสดงผล (Display)

ในการแสดงผลของเครื่องตั้งเวลาหยอดเหรียญนี้เราใช้หน้าจอแสดงผลแบบ LCD

บทที่ 4
การพัฒนาการเขียนโปรแกรม

การพัฒนาการเขียนโปรแกรม เราทำโดยวิธีการจำลองพอร์ทต่างๆ ของ ไอซี AT89C51 และ ไอซี AT89C2051 โดยเราทำการเขียนโปรแกรมแล้วนำมาทดสอบกับวงจรที่เรานำมาต่อกันเป็นภาคต่างๆ คือภาคการแสดงผล ภาคการควบคุมโซลีนอยด์วาวล์ ภาคควบคุมการเชื่อมต่อข้อมูลของระบบระหว่างไอซีไมโครคอนโทรลเลอร์ทั้งสองตัวคือ AT89C51 และ AT89C2051 เนื่องจาก ไอซีทั้งสองตัวนี้เราสามารถทำการโปรแกรมลงไปในตัวไอซีได้ถึง 1000 ครั้ง ดังนั้นเราสามารถเขียนโปรแกรมแล้วนำมาทดสอบการทำงานเป็นภาคๆ แล้วทำการทดลองทีละวงจรทีละภาคได้แล้วจึงทำการทดสอบทุกวงจรโดยรวมอีกครั้งหนึ่งเพื่อทดสอบการทำงานร่วมกันทั้งหมดว่าสามารถทำงานร่วมกันได้ ในการเขียนโปรแกรมเราทำการเขียนโดยใช้คอมพิวเตอร์แล้วเมื่อได้โปรแกรมที่ทำการทดสอบโดยการทดสอบในโปรแกรมของเครื่องคอมพิวเตอร์แล้ว ก็ทำการโปรแกรมที่ทำงานแล้วลงไปในตัวไอซีแล้วจึงทำการนำไปทดลองกับตัววงจรใช้งานจริงอีกที

โปรแกรมการใช้งาน

ในตัวโปรแกรมการใช้งานจะประกอบไปด้วยการทำงานดังนี้ โดยเริ่มแรก CPU จะทำการเซ็ทค่าเริ่มต้นต่างๆ ให้กับระบบ จากนั้นจะทำการแสดงข้อมูลต่างๆ ออกมาทางหน้าจอแสดงผลคือจอ LCD โดยจะแสดงค่าเวลา วัน เดือน ปี และจำนวนเหรียญที่หยอดเข้ามา โดยเริ่มแรกค่าจำนวนเหรียญที่เข้ามาจะเป็น 0 ในการเปิดเครื่องครั้งแรก และแสดงค่าเวลานับถอยหลังซึ่งก็เหมือนกับค่าของจำนวนเหรียญ ที่หยอดเข้ามาคือเมื่อมีการเปิดเครื่องครั้งแรกไม่มีเหรียญเข้ามาเวลาที่ต้องการนับถอยหลังก็จะไม่มี คือจะทำการเซ็ทค่าเริ่มต้นไว้ให้เท่ากับ 0 ไว้ก่อน ในตัวโปรแกรมจะมีส่วนต่างๆ ของตัวโปรแกรม 3 ชุดคือ

1. ชุดโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์หลัก(master)

ในโปรแกรมชุดนี้จะมีโปรแกรมที่เราสามารถตั้งเซ็ทค่าเวลาคือตั้งเวลาวันเดือนปีได้ และสามารถโปรแกรมการตั้งค่าเวลาต่อจำนวนเหรียญได้คือ ในจำนวนเหรียญที่หยอดเข้ามา 1 เหรียญ เราสามารถตั้งค่าเวลาของมันได้เป็น 5,10,15 และ 20 นาที และจะมีการทำการนับจำนวนเหรียญที่เข้ามาโดยผ่านจากตัววงจรนับเหรียญ ที่ผ่านเข้ามายังตัวเซ็นเซอร์ และทำการคำนวณออกมาเป็นค่าเวลารวมแล้วทำการเริ่มนับเวลาถอยหลังเมื่อมีการรับคำสั่งเข้ามาว่าผู้ใช้ตอบตกลงการทำงานของเครื่อง

2. ชุดโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์รอง(slave)

ในโปรแกรมการทำงานชุดนี้จะแสดงผลเหมือนกับตัวไมโครคอนโทรลเลอร์หลักคือแสดงจำนวนเหรียญที่ผู้ใช้หยอดเข้ามา และแสดงจำนวนเวลาที่นับถอยหลัง แต่จะไม่มีฟังค์ชั่นที่สำหรับการตั้งวันเดือนปี และตั้งค่าเวลาต่อจำนวนเหรียญ แต่จะมีฟังค์ชั่นอย่างอื่นแทนคือผู้ใช้สามารถตรวจสอบจำนวนเหรียญทั้งหมดที่มีการหยอดเข้ามาหลังจากการรีเซ็ทครั้งสุดท้ายจำนวนกี่เหรียญ และค่าเวลารวมทั้งหมดว่ามีการใช้เวลาไปทั้งหมดเป็นจำนวน กี่ชั่วโมงกี่นาที

3. ชุดโปรแกรมควบคุมการตรวจจับเหรียญ

ในโปรแกรมการทำงานชุดนี้จะมีการนับจำนวนเหรียญที่เข้ามาโดยผ่านจากตัวเซนเซอร์แล้วจะมีการนับเหรียญจำนวน 8 เหรียญ เมื่อมีการนับครบ 8 เหรียญแล้ว โปรแกรมจะสั่งงานให้ไปควบคุมจงจรที่ควบคุมโซลีนอยด์วาล์ว ให้ทำการเปิดแล้วจะไม่สามารถหยอดเหรียญเข้าเครื่องต่อไปได้อีกเพราะเหรียญจะตกลงไปในช่องคืนเหรียญแทน

บทที่ 5

การทดลองและวิธีการใช้งาน

การตั้งโปรแกรมของ Function การทำงาน

การตั้งโปรแกรม

- ตั้งเวลาต่อเหรียญ
- ตั้งเวลา update

ใช้ปุ่มควบคุม 3 ปุ่ม

โดยที่

ปุ่ม 1 คือ Enter

ปุ่ม 2 คือ เลื่อนไปทางซ้าย และ ปรับตัวเลข

ปุ่ม 3 คือ เลื่อนไปทางขวา

- การทำงานในตอนแรกจะทำในโหมดปกติ หน้าจอจะแสดง เวลานับถอยหลัง จำนวนเหรียญ ฯลฯ เรียกว่า หน้าจอปกติ
- เมื่อมีการเข้าหน้าจอเมนู เพื่อจะทำการโปรแกรม จะมีเพียงหน้าจอที่ตัวเครื่อง Master เท่านั้นที่แสดงผล ส่วนหน้าจอที่เครื่อง Slave จะแสดงผลในโหมหน้าจอปกติ
- เมื่อมีการกดปุ่ม 1 หน้าจอจะแสดง MENU ออกมาแล้วเราจะเลือก function จากหน้าจอเมนูนี้
- เมื่อมีการทำงานในโหมด การตั้งโปรแกรม จะมีการเข้าสู่โหมดหน้าจอแบบโหมดต่างๆ คือ MENU, TIME/DATE, TIME PER COIN แต่หากปล่อยทิ้งไว้โดยไม่มีการกดปุ่มใดปุ่มหนึ่งเกินเวลา 15 วินาที หน้าจอก็จะเข้าสู่การทำงานในโหมดปกติเองอัตโนมัติ

การทำงานในโหมด MENU

ในขณะที่หน้าจอแสดงผลในโหมดหน้าจอปกติ เมื่อมีการกดปุ่ม 1 หน้าจอที่เครื่อง Master จะเปลี่ยนเป็นการทำงานในโหมดเมนู ซึ่งหน้าจอจะแสดงดังนี้

MENU
1 2 3

โดยที่เลข 1 หมายถึง ตั้งวันเวลา TIME/DATE

เลข 2 หมายถึง ตั้งเวลาของเหรียญ TIME PER COIN

เลข 3 หมายถึง กลับเข้าสู่โหมดการทำงานปกติ

และเมื่อเข้าสู่การทำงานในโหมด MENU นี้ หมายเลข 1 จะกระพริบค้างไว้

โดยปุ่ม 1 หมายถึง การEnterที่ต้องการเข้าสู่การทำงานในโหมดนั้น

ปุ่ม 2 หมายถึง เลื่อนไปทางซ้าย

ปุ่ม 3 หมายถึง เลื่อนไปทางขวา

การปรับ เมื่อเรากดปุ่ม 3 ตัวเลขที่กระพริบอยู่จะเลื่อนไปเป็นตัวเลขที่อยู่ทางขวาให้กระพริบแทน โดยการเลือกจะเป็นการเลือกแบบวนไปเรื่อยๆ คือ ถ้าเรากดปุ่ม 3 ไปเรื่อยๆมันก็จะวิ่งจาก 1 ไป 2,จาก 2 ไป 3 และก็จาก 3 ไป 1 ใหม่ ไปเรื่อยๆ และปุ่ม 2 ก็เหมือนกันแต่ว่าจะเปลี่ยนจากการเลื่อนไปทางขวามาเป็นเลื่อนไปทางซ้ายแทน และเมื่อเราได้หมายเลขที่เราต้องการแล้วนั้น เราก็ทำการกดปุ่ม 1 เครื่องก็จะทำงานในโหมดนั้นทันที

การทำงานในโหมด TIME/DATE

จากหน้าจอ MENU กดปุ่ม 2 และ 3 โดยการเลือกจะมีการกระพริบไปตามหมายเลขคือ 1,2,3 หากเลข 1 กำลังกระพริบ และมีการกดปุ่ม 1 หมายถึงการ Enter ให้เข้าสู่โหมด การตั้งวันและเวลา

โดยที่หน้าจอจะแสดงดังนี้

TIME PER DATE
XX:XX:XX: XX:XX:XX

ซึ่ง XX ที่ 1 หมายถึง ชั่วโมง 0-23

XX ที่ 2 หมายถึง นาที 00-59

XX ที่ 3 หมายถึง วินาที 00-59

XX ที่ 4 หมายถึง วันที่ 1-31 (วันที่ RUN ตามเดือน 29 วัน = เดือน 2

30 วัน = เดือน 4,6,9,11

31 วัน = เดือน 1,3,5,7,8,10,12)

XX ที่ 5 หมาถึง เดือน 1-12

XX ที่ 6 หมายถึง ปี 00-99 (ค.ศ.)

โดย วันเวลาที่แสดงอยู่คือ วันเวลาปัจจุบันที่ต้องการเปลี่ยนแปลงแก้ไข

ปุ่ม 1 จะทำหน้าที่ Enter การ set วันเวลา และกลับเข้าสู่การทำงานในโหมด MENU

ปุ่ม 2 จะทำหน้าที่ ปรับตัวเลข

ปุ่ม 3 จะทำหน้าที่ เลื่อน step ของ ชั่วโมง,นาที,วัน,เดือน,ปี ไปทางขวาเรื่อยๆ และวนกลับมาที่ชั่วโมงใหม่เมื่อเลื่อนไปจนถึงปีแล้ว

การทำงาน

ในโหมด TIME/DATE

เมื่อเริ่มการทำงานในโหมดนี้ หน้าจอจะแสดงผลออก ตามรูป โดยที่ตัวเลขของชั่วโมงจะกระพริบอยู่ คือพร้อมที่จะทำการตั้งชั่วโมงใหม่ การจะตั้งก็โดยการกดปุ่ม 2 กดไปเรื่อยๆจนกว่าจะได้จำนวนชั่วโมงที่ต้องการ โดยตัวเลขจะแสดงตั้งแต่ 0 ถึง 23 และจะ RUN ไปเรื่อยๆตามการกดปุ่ม 2 เมื่อถึงเลข 23 ก็จะวนกลับมา 0 ใหม่

และเมื่อได้ตัวเลขชั่วโมงที่ต้องการแล้ว ก็กดปุ่ม 3 จากนั้นตัวเลขที่แสดงจำนวน นาที จะกระพริบ เราก็สามารถปรับจำนวนนาทีได้ โดยการเลือกตัวเลข จากปุ่ม 2 โดยตัวเลขจะ RUN ตั้งแต่ 00 ถึง 59 เมื่อได้จำนวนนาทีที่เราต้องการแล้วก็ทำการกดปุ่ม 3 จากนั้นตัวเลขที่แสดงวันที่จะกระพริบ ซึ่งเราสามารถเลือกได้โดยการกดปุ่ม 2 ว่าต้องการจำนวนตัวเลขวันที่อะไร โดยตัวเลขวันที่จะ Run ตามจำนวนเดือนคือ

ถ้าเป็นเดือน 2 วันที่จะ RUN ตั้งแต่ 1 ถึง 29

ถ้าเป็นเดือน 4,6,9,11 วันที่จะ RUN ตั้งแต่ 1 ถึง 30

ถ้าเป็นเดือน 1,3,5,7,8,10,12 วันที่จะ RUN ตั้งแต่ 1 ถึง 31

และเมื่อได้วันที่ที่ต้องการแล้วก็กดปุ่ม 3 จากนั้นตัวเลขที่แสดงจำนวนเดือนจะกระพริบ เราก็เลือกตัวเลขจากการกดปุ่ม 2 โดยตัวเลขจะ RUN ตั้งแต่ 1 ถึง 12 วนไปเรื่อยๆ เมื่อเราได้จำนวนเดือนที่เราต้องการแล้วก็ทำการกดปุ่ม 3 จากนั้นตัวเลขที่แสดงจำนวนปีจะกระพริบ เราสามารถเลือกตัวเลขจากการกดปุ่ม 2 ซึ่งตัวเลขจะ RUN ตั้งแต่ 00 ถึง 99 และเมื่อได้จำนวนตัวเลขปีที่เราต้องการแล้ว ก็ทำการกดปุ่ม 3 ตัวเลขที่แสดงจำนวนชั่วโมงจะวนมากระพริบซ้ำอีกครั้ง วนอยู่อย่างนี้จนกว่าจะมีการกดปุ่ม 1 หน้าจอจึงจะกลับไปทำงานในโหมด MENU และเครื่องจะทำการโหลดข้อมูลที่เราตั้งไปใหม่เมื่อครู่เก็บไว้ในหน่วยความจำ

หมายเหตุ การทำงานในโหมดนี้ จะกลับเข้าสู่การทำงานในโหมด MENU ทันทีเมื่อมีการกดปุ่ม 1 ไม่ว่าในขณะนั้นจะมีการตั้งตัวเลขในหน่วยอะไรก็ตาม และจะทำการ save วันเวลาที่แสดงอยู่ปัจจุบันเข้าไปเก็บไว้ในหน่วยความจำ

การทำงานในโหมด TIME PER COIN

จากหน้าจอ MENU กดปุ่ม 2 และ 3 โดยการเลือกจะมีการกระพริบไปตามหมายเลขคือ 1,2,3 หากเลข 2 กำลังกระพริบ และมีการกดปุ่ม 1 หมายถึงการ Enter เข้าสู่โหมด การตั้งเวลาของเหรียญ
โดยที่หน้าจอจะแสดงดังนี้

TIME PER COIN
2 5 10 15 20

โดย ตัวเลขที่กระพริบอยู่คือ ค่าเวลาที่เราตั้งไว้ปัจจุบัน มีค่าเวลาเป็นหน่วย นาที ต่อ 1 เหรียญ
ปุ่ม 1 จะทำหน้าที่ Enter และกลับเข้าสู่การทำงานในโหมด MENU
ปุ่ม 2 จะทำหน้าที่ เลื่อนไปทางซ้าย
ปุ่ม 3 จะทำหน้าที่ เลื่อนไปทางขวา

การทำงาน
ในโหมด TIME PER COIN

เมื่อเริ่มการทำงานในโหมดนี้ หน้าจอจะแสดงผลออกมาดังรูป โดยที่ตัวเลขที่กระพริบอยู่คือ ค่าเวลาที่เราตั้งไว้ปัจจุบัน มีหน่วยเป็น นาที ต่อ 1 เหรียญ เมื่อมีการกดปุ่ม 2 จะมีการเลื่อนไปทางซ้าย สมมุติว่าตอนแรก เลข 10 กระพริบอยู่ หมายความว่าตอนนี้เราตั้งเวลาไว้ที่ 10 นาทีต่อ 1 เหรียญ เมื่อเรากดปุ่ม 2 เลข 5 ก็จะกระพริบแทน และเมื่อมีการกดปุ่ม 2 อีก ก็จะเลื่อนกลายเป็นเลข 2 กระพริบแทน และถ้ากดปุ่ม 2 อีกครั้งตัวเลขที่กระพริบก็จะวนมาที่ เลข 20 คือจะวนมาทางซ้ายเรื่อยๆ และปุ่ม 3 ก็เช่นกันเพียงแต่ว่าปุ่ม 3 จะเลื่อนไปทางขวา เมื่อได้ค่าเวลาตามที่เราต้องการแล้ว ก็ทำการกดปุ่ม 1 เครื่องจะทำการ Set ค่าเวลาต่อเหรียญให้เป็นเวลาใหม่ ตามค่าตัวเลขที่กำลังกระพริบอยู่ และจะทำการกลับเข้าสู่หน้าจอการทำงานในโหมด MENU ตามเดิม

ตัวอย่าง เมื่อเราเลือกให้ เลข 2 กระพริบ แล้วเรากดปุ่ม 1 หมายถึงเครื่องจะทำการ Set ตัวเองให้ตั้งเวลา 2 นาที(นับถอยหลัง) ต่อ 1 เหรียญ

บทที่ 6

บทสรุป และแนวทางการพัฒนา

จากการทำโครงงานเครื่องตั้งเวลาหยอดเหรียญ(programable timer for vending machine) มีอุปสรรคในการทำงานอยู่บ้างพอสมควร ซึ่งสามารถแก้ไขและปรับปรุงไปได้ด้วยดี ซึ่งปัญหาต่างๆที่มีสามารถกล่าวได้ ดังต่อไปนี้

1. ปัญหาอันเนื่องมาจากการเขียนโปรแกรม ซึ่งการเขียนโปรแกรมโดยให้โปรแกรมทำงานร่วมกับระบบ แมคคานิก ในตอนแรกไม่สามารถควบคุมได้ถึง 100 เปอร์เซนต์ เพราะว่าการให้โปรแกรมไปควบคุมการทำงานของวงจรที่จะไปควบคุม โซลีนอยด์ นั้นทำงานร่วมกันได้ไม่ดีเท่าที่ควร

2. ปัญหาอันเนื่องมาจาก อุปกรณ์ในการเขียนและบันทึกโปรแกรมลงบนตัวไอซีไมโครคอนโทรลเลอร์ ไม่พร้อมทำให้เสียเวลาในการเขียและบันทึกโปรแกรมมาก อีกทั้งการทำแผ่นพลาสติกต้นแบบนั้น ทำได้ยาก ต้องเสียเวลาในการทำมาก

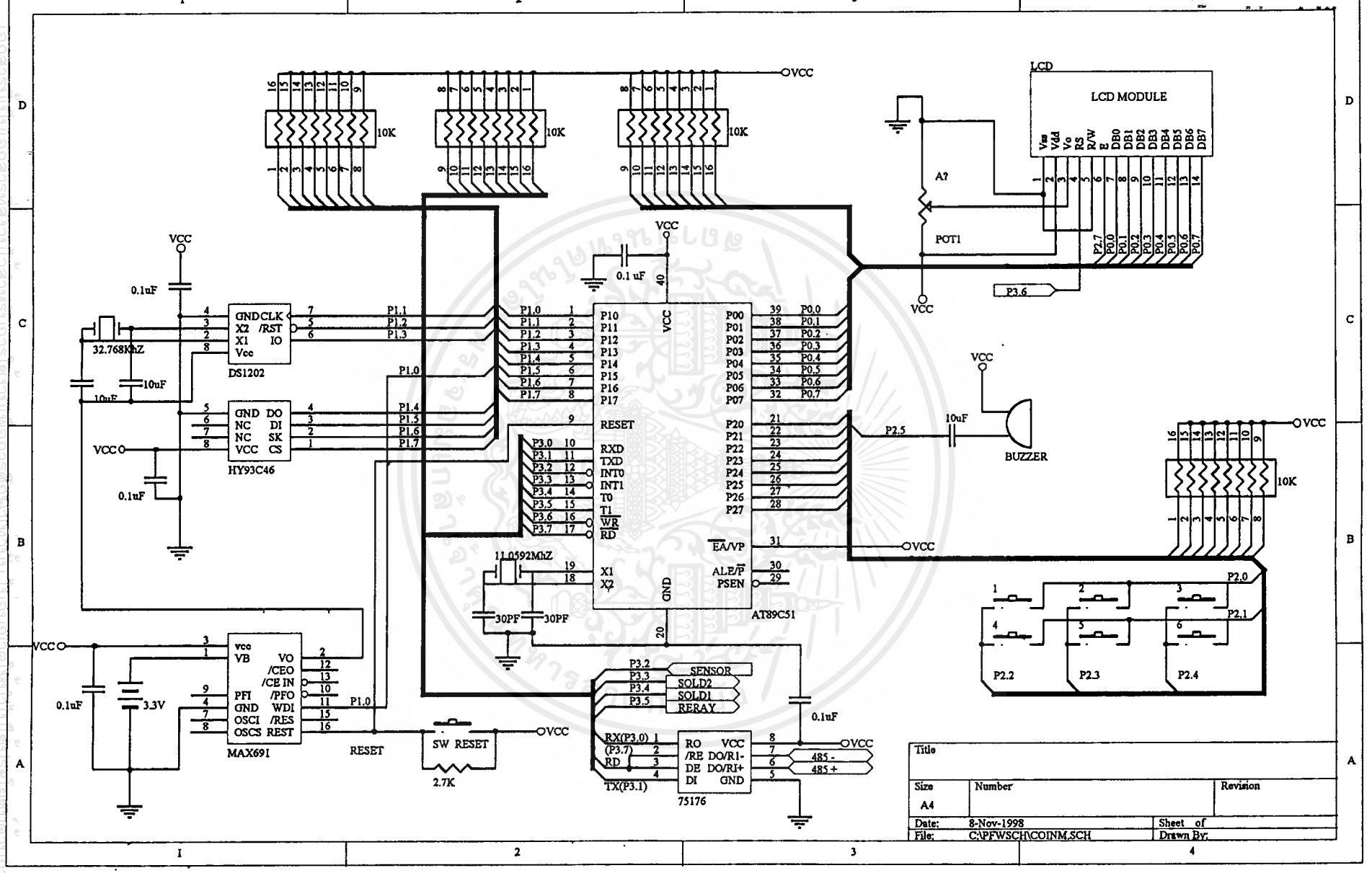
บทสรุป จากการทำโครงงานเครื่องตั้งเวลาหยอดเหรียญ(programable timer for vending machine) จากการทำงานและการทดลองได้เป็นไปตามวัตถุประสงค์คือเพื่อพัฒนาทักษะในการเขียนโปรแกรม และออกแบบศึกษาการใช้งานไอซีที่อ่าน เขียนข้อมูลแบบอนุกรมได้เป็นอย่างดี ส่วนอุปสรรคที่เกิดขึ้นนั้น ก็สามารถแก้ไขไปได้ด้วยดี ทำให้ได้รับความรู้และประสบการณ์ต่างๆ จากการทำโครงงานคือ ได้ทักษะในการเขียนโปรแกรมควบคุมอุปกรณ์ต่างๆ และศึกษาการใช้งานไอซีต่างๆ ซึ่งอ่านและเขียนข้อมูลแบบอนุกรมและการทำงานของอุปกรณ์ระบบแมคคานิก

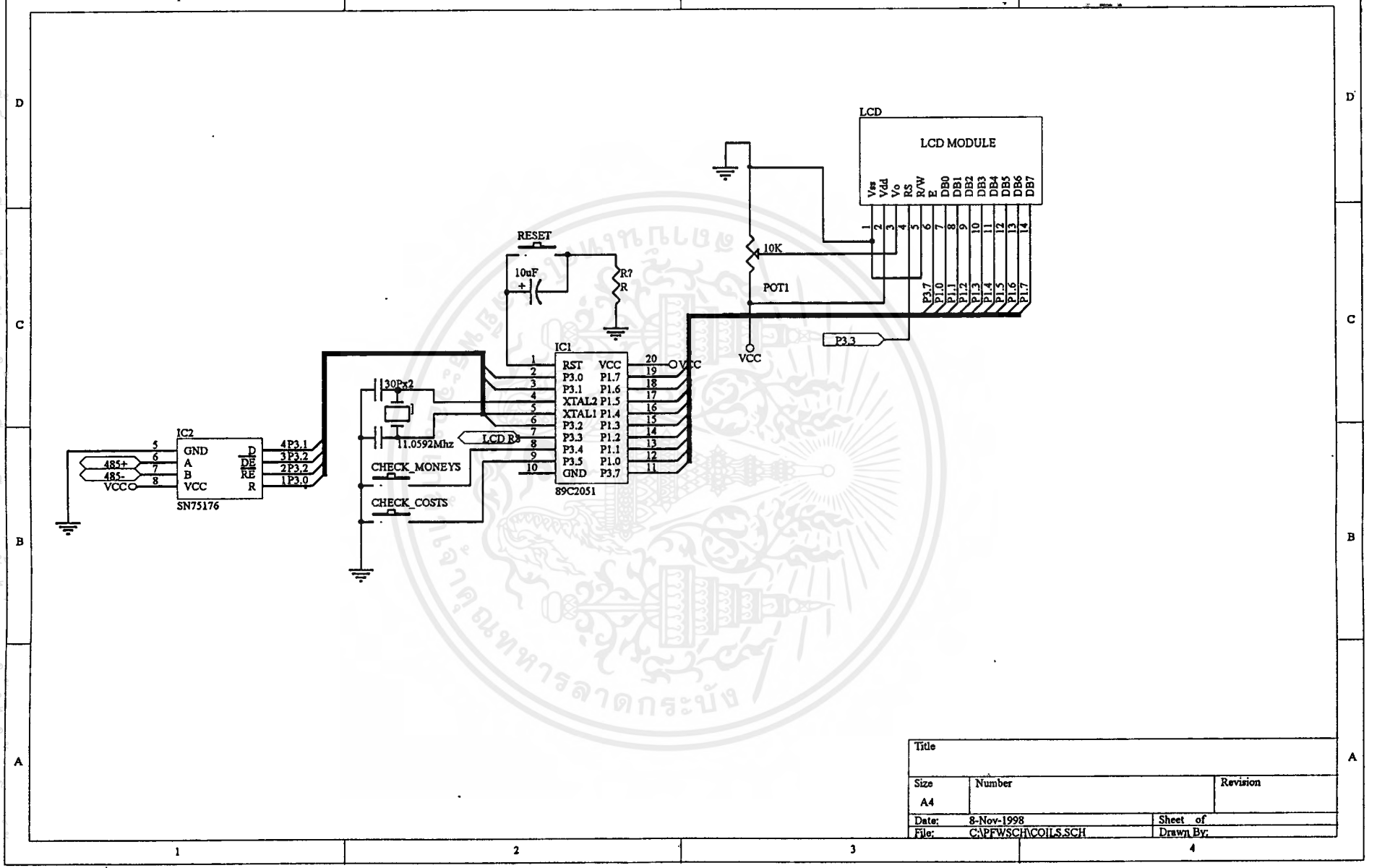
นอกจากนี้ยังได้เรียนรู้ถึงการทำงานร่วมกันเป็น กลุ่มซึ่งเป็นส่วนเสริมที่จะนำไปประยุกต์ใช้กับชีวิตการทำงานในอนาคตได้ดียิ่งขึ้น

หนังสืออ้างอิง

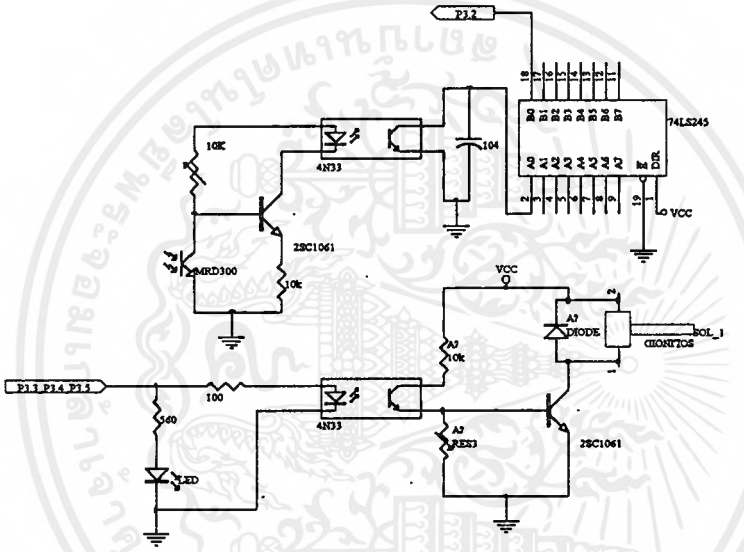
1. สุนทร วิทูสุรพจน์,การใช้งานไมโครคอนโทรลเลอร์ ตระกูล 8051,บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน),2537
2. ผศ.สมยศ จุณณะปิยะ,การใช้งานไมโครคอนโทรลเลอร์ ตระกูล MCS-51,คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง,2537
3. http://www.atmel.com/
4. http://www.intel.com/
5. http://www.dallas.com/
6. http://www.mxim.com/

ภาคผนวก ก.
รายละเอียดทางฮาร์ดแวร์


LCD
LCD MODULE
Vss Vdd Vo RS R/W E DB0 DB1 DB2 DB3 DB4 DB5 DB6 DB7
A?
POT1
P2.7 P0.0 P0.1 P0.2 P0.3 P0.4 P0.5 P0.6 P0.7
P3.6
VCC
10K
0.1uF
32.768KhZ
10uF
GND CLK
X2 /RST
X1 IO
Vcc
DS1202
GND DO
NC DI
NC SK
VCC CS
HY93C46
P1.0 P1.1 P1.2 P1.3 P1.4 P1.5 P1.6 P1.7
P10 P11 P12 P13 P14 P15 P16 P17
RESET
RXD TXD INT0 INT1 T0 T1 WR RD
P00 P01 P02 P03 P04 P05 P06 P07
P20 P21 P22 P23 P24 P25 P26 P27
EA/VP
ALE/P
PSEN
11.0592MhZ
30PF
AT89C51
BUZZER
P2.5
P2.0 P2.1 P2.2 P2.3 P2.4
vcc
VB VO
/CEO
/CE IN
PFI /PFO
GND WDI
OSCI /RES
OSCS REST
MAX691
3.3V
RESET
SW RESET
2.7K
P3.2 SENSOR
P3.3 SOLD2
P3.4 SOLD1
P3.5 RERAY
RX(P3.0)
(P3.7)
RD
TX(P3.1)
RO VCC
/RE DO/RI-
DE DO/RI+
DI GND
75176
485 -
485 +
Title
Size A4
Number
Revision
Date: 8-Nov-1998
Sheet of
File: C:\PFWSCH\COINM.SCH
Drawn By:

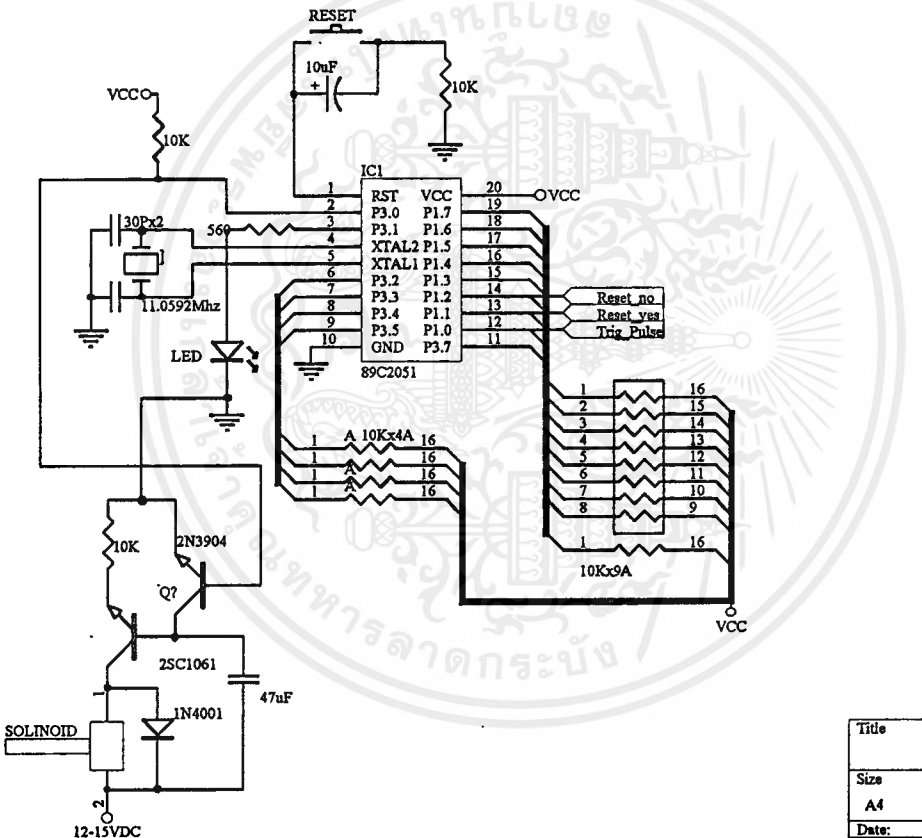

LCD
LCD MODULE
Vss Vdd Vo RS R/W E DB0 DB1 DB2 DB3 DB4 DB5 DB6 DB7
1 2 3 4 5 6 7 8 9 10 11 12 13 14
10K
POT1
VCC
P3.7 P1.0 P1.1 P1.2 P1.3 P1.4 P1.5 P1.6 P1.7
P3.3
RESET
10uF
R?
R
IC1
RST VCC
P3.0 P1.7
P3.1 P1.6
XTAL2 P1.5
XTAL1 P1.4
P3.2 P1.3
P3.3 P1.2
P3.4 P1.1
P3.5 P1.0
GND P3.7
89C2051
30Px2
11.0592Mhz
LCD R3
CHECK_MONEYS
CHECK_COSTS
IC2
GND D
A DE
B RE
VCC R
SN75176
485+
485-
VCC
4P3.1
3P3.2
2P3.2
1P3.0
Title
Size A4
Number
Revision
Date: 8-Nov-1998
File: C:\PFWSCH\COILS.SCH
Sheet of
Drawn By:

วงจร Slave



Title			
Size: B	Number		Revision
Date:	8-Nov-1998	Sheet of	
File:	C:\PEWSCH\DRIVER.SCH	Drawn By:	

วงจร Control solinoid



Title		
Size A4	Number	Revision
Date: 8-Nov-1998		Sheet of
File: C:\PFWSCH\DET_COIN.SCH		Drawn By:

รูปที่ Detect coin

ภาคผนวก ข.
โปรแกรมใช้งาน

ภาคผนวก ข(1).

โปรแกรมใช้งาน MASTER

```
;PROGRAM FOR MASTER  COIN
 ;---------- PORT EQU ----------
 RS      EQU P3.6
 ENABLE  EQU P2.7
 DATA_LCD EQU P0
 WDI     EQU P1.0
 BEEPP   EQU P2.5
 SLED    EQU P2.6
 SENSOR  EQU P3.2
 SOLD1   EQU P3.4
 SOLD2   EQU P3.3
; RELAY   EQU P2.6
 RELAY   EQU P3.5
 SEND    EQU P3.7
 EEPDO       EQU   P1.4
 EEPDI       EQU   P1.5
 EEPCLK      EQU   P1.6
 EEPCS       EQU   P1.7

;---------- MEMORY ----------
 EEPAD     EQU 02
 EEPLSB    EQU 03
 EEPMSB    EQU 04
 LINE      EQU 03
 HOUR      EQU 30H
 MINUTE    EQU 31H
 SEC       EQU 32H
 DATE      EQU 33H
 MOUNTH    EQU 34H
 YEAR      EQU 35H
```

```
DAY        EQU 36H
HEX_MSB    EQU 37H
HEX_LSB    EQU 38H
COIL1_MSB  EQU 39H
COIL1_LSB  EQU 3AH
COIL2_MSB  EQU 3BH
COIL2_LSB  EQU 3CH
TPCM       EQU 3DH
MIN_CAP    EQU 3EH
COUNTER_CAP EQU 3FH

LCD1       EQU 40H
LCD20      EQU 53H
TIME_MSB   EQU 54H
TIME_LSB   EQU 55H
TIME_MSB_CAP EQU 56H
TIME_LSB_CAP EQU 57H
COIN_SEND  EQU 58H
;-------------------------EEPROM EQU -------------------------
TPC        EQU 63
EETCOIN    EQU 62
EELCOIN    EQU 61
EETIME     EQU 60

;-------------------------DATA EQU -----------------------------
DELAY_SCAN  EQU 07FH
;------------------------------------------------------------
;--------- BIT  EQU -----------------------------------------
BEEPF      EQU 00
```

```
        ORG 0000H
        LJMP MAIN
        LJMP DETECT_COIL
        ORG 0BH
        MOV TL0,#0
        MOV TH0,#10
        CPL WDI
        RETI

MAIN:

        MOV COIL1_MSB,#0
        MOV COIL1_LSB,#0
        CLR EEPCLK
        CLR EEPCS
        CALL EEPEN

        CLR  SOLD1
        CLR  SOLD2
;       CLR  RELAY
        MOV A,#100
        CALL DELAY_MS
;       SETB SLED
        SETB SENSOR

        CALL INIT    ;INIT PAGE
        CLR IT0
```

```
    SETB EA
    MOV TMOD,#0
    SETB EX0
    SETB ET0
    MOV TL0,#0
    MOV TH0,#10
    SETB TR0
;-----------------------------END OF INIT STATE-----------------
MAIN_PROGRAM:
    MOV COIL1_MSB,#0
    MOV COIL1_LSB,#0
    MOV EEPAD,#TPC
    CALL EEPRD
    MOV TPCM,EEPLSB
WAITX:

    MOV DPTR,#L2FROM
    CALL INMEM

    MOV EEPAD,#EETCOIN
    CALL EEPRD
    MOV DPH,EEPMSB
    MOV DPL,EEPLSB
    CALL HTOD
    MOV COIL2_MSB,R2
    MOV COIL2_LSB,R3
    MOV A,R2
    CALL BCDTOHEX
    MOV 50H,HEX_MSB
    MOV 51H,HEX_LSB
```

```
MOV A,R3
CALL BCDTOHEX
MOV 52H,HEX_MSB
MOV 53H,HEX_LSB

MOV EEPAD,#EETIME
CALL EEPRD
MOV DPH,EEPMSB
MOV DPL,EEPLSB
CALL HTOD
MOV A,R2
CALL BCDTOHEX
MOV  45H,HEX_MSB
MOV  46H,HEX_LSB
MOV  A,R3
CALL BCDTOHEX
MOV  47H,HEX_MSB
MOV  48H,HEX_LSB

MOV LINE,#2
CALL SCAN


CALL SHOWTIME
CALL SEND_DATA

ORL P2,#00011111B
CLR P2.1
JNB P2.2,SET_FUNCTION
JNB P2.3,SET_FUNCTION
```

```
        JNB P2.4,SET_FUNCTION

        MOV A,TIME_LSB
        CJNE A,#0,DEC_COUN
        MOV A,TIME_MSB
        CJNE A,#0,DEC_COUN
        CLR RELAY
        MOV MIN_CAP,MINUTE
        JMP WAITX
DEC_COUN:
        SETB RELAY
        CALL COUNTER
        JMP WAITX



SET_FUNCTION:
        CALL INKEY
        MOV B,#3
        DEC A
        MUL AB
        MOV DPTR,#FUNCTION
        JMP @A+DPTR


;------- COUNTER TIME ----------------
COUNTER:
        CALL TIME_DISPLAY
        MOV A,MIN_CAP
```

```
        CJNE  A,MINUTE,DEC_MIN
        RET
DEC_MIN:
        MOV MIN_CAP,MINUTE
        MOV A,TIME_LSB
        CLR CY
        SUBB A,#1
        MOV TIME_LSB,A
        JC NOBEEP
        CJNE A,#2,STEEE
        CLR BEEPF
        JB BEEPF,STEEE
        CALL BEEP
        JMP STEEE
NOBEEP:
        MOV A,TIME_MSB
        SUBB A,#0
        MOV TIME_MSB,A
STEEE:
        MOV EEPAD,#EETIME
        MOV EEPMSB,TIME_MSB
        MOV EEPLSB,TIME_LSB
        CALL EEPWR
        RET

BEEP:
      MOV B,#4
SOUND2:
      MOV R7,#4
SOUND11:
```

```
    MOV R6,#0FFH
SOUND1:
    CPL BEEPP
    MOV A,#0BFH
    DJNZ ACC,$
    DJNZ R6,SOUND1
    DJNZ R7,SOUND11
    MOV A,#0FFH
    CALL DELAY_MS
    DJNZ B,SOUND2
    SETB BEEPF
    RET

FUNCTION:
    LJMP FUNCTION1
    LJMP FUNCTION2
    LJMP FUNCTION3
    JMP 0
F3L1:   DB "C/T= XX TCOIN = XXXX"
F3L2:   DB "1.CLEAR ALL YES.EXIT"
TEST1:   DB "0123456789ABCDEF0123"
F2L1:   DB "XX MINUTE PER 1 COIN"
F2L2:   DB "1.INC YES.SET MINUTE"
SDATEL2:  DB "1.DATE 2.MOUN 3.YEAR"
STIMEL2:  DB " 1.HOUR 2.MIN 3.SEC "
F1L1:   DB " SET TIME AND DATE  "
F1L2:   DB "1.TIME 2.DATE 3.END "
CLEAR_D:  DB "                "
TEST2:   DB "ABCDEFGHIJKLMNOPQRST"
L1FROM:   DB "HH:MM:SS.  DD/MM/YY."
```

```
L1FROMC:   DB "COIN=0000  TIME=0000"
L2FROM:    DB "TIME=XXXX TCOIN=XXXX"
YES_NO:    DB "YES.=OK   NO.=REJECT"
;--------DISPLAY TOTAL COIN  AND CLEAR ALL-----------------------------
FUNCTION3:
        MOV DPTR,#F3L2
        CALL INMEM
        MOV LINE,#2
        CALL SCAN
        MOV DPTR,#F3L1
        CALL INMEM
        MOV EEPAD,#TPC
        CALL EEPRD
        MOV DPH,#0
        MOV DPL,EEPLSB
        CALL HTOD
        MOV A,R3
        CALL BCDTOHEX
        MOV 45H,HEX_MSB
        MOV 46H,HEX_LSB
        MOV EEPAD,#EETCOIN
        CALL EEPRD
        MOV DPH,EEPMSB
        MOV DPL,EEPLSB
        CALL HTOD
        MOV A,R2
        CALL BCDTOHEX
        MOV 50H,HEX_MSB
        MOV 51H,HEX_LSB
        MOV A,R3
```

```
        CALL BCDTOHEX
        MOV 52H,HEX_MSB
        MOV 53H,HEX_LSB
        MOV LINE,#1
        CALL SCAN
ERRK:
        CALL INKEY
        CJNE A,#1,CHECK_STYES
        MOV EEPAD,#TPC
        MOV EEPMSB,#0
        MOV EEPLSB,#1
        CALL EEPWR
        MOV EEPAD,#EETCOIN
        MOV EEPMSB,#0
        MOV EEPLSB,#0
        CALL EEPWR
        MOV EEPAD,#EELCOIN
        MOV EEPMSB,#0
        MOV EEPLSB,#0
        CALL EEPWR
        MOV EEPAD,#EETIME
        MOV EEPMSB,#0
        MOV EEPLSB,#0
        CALL EEPWR
        MOV TIME_MSB,#0
        MOV TIME_LSB,#0
        JMP FUNCTION3

CHECK_STYES:  CJNE A,#4,ERRK
        JMP 0000
```

```
;--------- TIME PER COIN MODE ----------------------------
FUNCTION2:
    MOV DPTR,#F2L2
    CALL INMEM
    MOV LINE,#2
    CALL SCAN
    MOV EEPAD,#TPC
    CALL EEPRD
    MOV TPCM,EEPLSB
SHOW_COINP:
    MOV DPTR,#F2L1
    CALL INMEM
    MOV DPH,#0
    MOV DPL,TPCM
    CALL HTOD
    MOV A,R3
    ANL A,#0F0H
    SWAP A
    ADD A,#30H
    MOV 40H,A
    MOV A,R3
    ANL A,#0FH
    ADD A,#30H
    MOV 41H,A
    MOV LINE,#1
    CALL SCAN
NEW_KEYC:
    CALL INKEY
    CJNE A,#1,CHECK_YESC
    INC TPCM
```

```
        MOV A,TPCM
        ORL A,#0
        JZ COIN_ZERO
        CLR CY
        SUBB A,#21
        JNC COIN_ZERO
        JMP COIN_INC
COIN_ZERO:
       MOV TPCM,#1
COIN_INC:
       JMP SHOW_COINP
CHECK_YESC:
        CJNE A,#4,NEW_KEYC
        MOV EEPAD,#TPC
        MOV EEPLSB,TPCM
        CALL EEPWR
        JMP 0000
 IN_COIL:
        MOV A,COIL1_LSB
        CLR CY
        ADD A,#1
        MOV COIL1_LSB,A
        MOV A,COIL1_MSB
        ADDC A,#0
        MOV COIL1_MSB,A
         MOV DPTR,#L1FROMC
        CALL INMEM
        MOV DPH,COIL1_MSB
        MOV DPL,COIL1_LSB
        CALL HTOD
```

```
MOV A,R2
CALL BCDTOHEX
MOV 45H,HEX_MSB
MOV 46H,HEX_LSB
MOV A,R3
CALL BCDTOHEX
MOV 47H,HEX_MSB
MOV 48H,HEX_LSB
MOV EEPAD,#TPC
CALL EEPRD
MOV R2,#0
MOV R3,EEPLSB
MOV DPH,COIL1_MSB
MOV DPL,COIL1_LSB
CALL DPMUL
MOV TIME_MSB_CAP,DPH
MOV TIME_LSB_CAP,DPL
CALL HTOD
MOV A,R2
CALL BCDTOHEX
MOV 50H,HEX_MSB
MOV 51H,HEX_LSB
MOV A,R3
CALL BCDTOHEX
MOV 52H,HEX_MSB
MOV 53H,HEX_LSB
MOV LINE,#1
CALL SCAN
MOV DPTR,#YES_NO
CALL INMEM
```

```
        MOV LINE,#2
        CALL SCAN
CHECK_YES_NO:
        CLR P2.0
        SETB P2.1
        SETB P2.2
        SETB P2.3
        JNB P2.2,PRESS_YES
        JNB P2.3,PRESS_NO
        CALL SEND_DATA
        MOV A,TIME_LSB
        CJNE A,#0,DEC_COUNK
        MOV A,TIME_MSB
        CJNE A,#0,DEC_COUNK
        CLR RELAY
        MOV MIN_CAP,MINUTE
        JMP CHECK_YES_NO
DEC_COUNK:
        SETB RELAY
        CALL COUNTER
        JMP CHECK_YES_NO
PRESS_YES:
        JNB P2.2,$
        SETB SOLD2
        MOV A,#2FH
        CALL DELAY_MS
        MOV EEPAD,#EETCOIN
        CALL EEPRD
        MOV A,EEPLSB
        CLR CY
```

```
        ADD A,COIL1_LSB
        MOV EEPLSB,A
        MOV A,EEPMSB
        ADDC A,#0
        MOV EEPMSB,A
        MOV EEPAD,#EETCOIN
        CALL EEPWR
        MOV A,TIME_LSB_CAP
        CLR CY
        ADD A,TIME_LSB
        MOV TIME_LSB,A
        MOV A,TIME_MSB_CAP
        ADDC A,TIME_MSB
        MOV TIME_MSB,A
        MOV EEPAD,#EETIME
        MOV EEPMSB,TIME_MSB
        MOV EEPLSB,TIME_LSB
        CALL EEPWR
        MOV EEPAD,#EELCOIN
        MOV EEPMSB,COIL1_MSB
        MOV EEPLSB,COIL1_LSB
        CALL EEPWR
        MOV COIN_SEND,COIL1_LSB
        MOV A,#1
        CALL DELAY_SEC
        CLR SOLD2
        JMP MAIN_PROGRAM
PRESS_NO:
        JNB P2.3,$
        SETB SOLD1
```

```
        MOV A,#2
        CALL DELAY_SEC
        CLR SOLD1
        JMP MAIN_PROGRAM
; ********** DPMUL SUB **********
; IN  = DPTR   MULTIPLER
;     = R2,R3  MULTIPLICAND
; OUT = DPTR   RESULT
;     = CARRY FLAG SET WHEN RESULT OVERFLOW
; REG = A,R1,R2,R3,R4,R5,R6,DPTR
DPMUL:  MOV   R4,#0         ;CLEAR RESULT
        MOV   R5,#0
        MOV   R6,#0         ;CLEAR MUL FLAG
        MOV   R1,#16
DPMUL1: MOV   A,R5          ;*2
        ADD   A,R5
        MOV   R5,A
        MOV   A,R4
        ADDC  A,R4
        MOV   R4,A
        JNC   DPMUL2        ;NO CARRY
        MOV   R6,#1         ;OVERFLOW (SET FLAG)
DPMUL2: MOV   A,R3
        RLC   A
        MOV   R3,A
        MOV   A,R2
        RLC   A
        MOV   R2,A
        JNC   DPMUL3
        MOV   A,R5          ;R4R5=R4R5+DPTR
```

```
        ADD   A,DPL
        MOV   R5,A
        MOV   A,R4
        ADDC  A,DPH
        MOV   R4,A
        JNC   DPMUL3        ;NO CARRY
        MOV   R6,#1         ;OVERFLOW (SET FLAG)
DPMUL3: DJNZ  R1,DPMUL1
        MOV   DPH,R4
        MOV   DPL,R5
        MOV   A,R6
        MOV   C,ACC.0       ;LOAD MUL FLAG TO CARRY
        RET
; ********** EEPRD SUB **********
; 93C46 READ WORD SUBROUTINE
; IN  = R2 ADDRESS (0-63)
; OUT = R3,R4 DATA
; REG = A,R2,R3,R4
EEPRD:  MOV   A,R2
        ANL   A,#00111111B  ;MAKE SURE DATA=6 BIT
        ORL   A,#10000000B  ;OPCODE
        SETB  EEPCS         ;CS=1
        SETB  EEPDI         ;START BIT
        LCALL CLOCK
        MOV   R2,#8         ;OPCODE,ADDRESS
EEPRD2: RLC   A
        MOV   EEPDI,C
        LCALL CLOCK
        DJNZ  R2,EEPRD2
        MOV   R2,#8         ;READ DATA (8 BIT FIRST)
```

```
EEPRD4: LCALL  CLOCK
        MOV    C,EEPDO
        RLC    A
        DJNZ   R2,EEPRD4
        MOV    R3,A
        MOV    R2,#8          ;READ DATA (8 BIT LAST)
EEPRD5: LCALL  CLOCK
        MOV    C,EEPDO
        RLC    A
        DJNZ   R2,EEPRD5
        MOV    R4,A
        CLR    EEPCS          ;CS=0
        RET
CLOCK:  SETB   EEPCLK         ;SYNC CLOCK
        NOP
        NOP
        CLR    EEPCLK
        NOP
        NOP
        RET
; ********** EEPWR SUB **********
; 93C46 WRITE WORD SUBROUTINE
; IN  = R2 ADDRESS (0-63)
;     = R3,R4 DATA
; REG = A,R2
EEPWR:  MOV    A,R2
        ANL    A,#00111111B   ;MAKE SURE DATA=6 BIT
        ORL    A,#01000000B   ;OPCODE
        SETB   EEPCS          ;CS=1
        SETB   EEPDI          ;START BIT
```

```
    LCALL CLOCK
    MOV   R2,#8         ;OPCODE,ADDRESS
EEPWR2: RLC   A
    MOV   EEPDI,C
    LCALL CLOCK
    DJNZ  R2,EEPWR2
    MOV   R2,#8         ;WRITE DATA (8 BIT FIRST)
    MOV   A,R3
EEPWR4: RLC   A
    MOV   EEPDI,C
    LCALL CLOCK
    DJNZ  R2,EEPWR4
    MOV   R2,#8         ;WRITE DATA (8 BIT LAST)
    MOV   A,R4
EEPWR5: RLC   A
    MOV   EEPDI,C
    LCALL CLOCK
    DJNZ  R2,EEPWR5
    SETB EEPDO
    CLR   EEPCS         ;CS=0
    NOP
    NOP
    SETB  EEPCS         ;CS=1 (CHECK STATUS)
    NOP
    NOP
    JNB   EEPDO,$       ;WAIT FOR BUSY
    NOP
    NOP
    CLR   EEPCS         ;CS=0
    RET
```

```
; ********** EEPER SUB **********
; 93C46 ERASE WORD SUBROUTINE
; IN  = R2 ADDRESS (0-63)
; REG = A,R2
EEPER:  MOV   A,R2
        ANL   A,#00111111B    ;MAKE SURE DATA=6 BIT
        ORL   A,#11000000B    ;OPCODE
        SETB  EEPCS           ;CS=1
        SETB  EEPDI           ;START BIT
        LCALL CLOCK
        MOV   R2,#8           ;OPCODE,ADDRESS
EEPER2: RLC   A
        MOV   EEPDI,C
        LCALL CLOCK
        DJNZ  R2,EEPER2
        SETB EEPDO
        CLR   EEPCS           ;CS=0
        NOP
        NOP
        SETB  EEPCS           ;CS=1 (CHECK STATUS)
        NOP
        NOP
        JNB   EEPDO,$         ;WAIT FOR BUSY
        NOP
        NOP
        CLR   EEPCS           ;CS=0
        RET
; ********** EEPEN SUB **********
; 93C46 ENABLE SUBROUTINE
; REG = A,R2
```

```
EEPEN:  MOV   A,#00110000B    ;OPCODE
        SETB  EEPCS           ;CS=1
        SETB  EEPDI           ;START BIT
        LCALL CLOCK
        MOV   R2,#8           ;OPCODE,ADDRESS
EEPEN2: RLC   A
        MOV   EEPDI,C
        LCALL CLOCK
        DJNZ  R2,EEPEN2
        CLR   EEPCS
        RET
EEPDS:
        MOV   A,#00000000B    ;OPCODE
        SETB  EEPCS           ;CS=1
        SETB  EEPDI           ;START BIT
        LCALL CLOCK
        MOV   R2,#8           ;OPCODE,ADDRESS
EEPDS2: RLC   A
        MOV   EEPDI,C
        LCALL CLOCK
        DJNZ  R2,EEPDS2
        CLR   EEPCS
        RET
;---------- FUNCTION 1 SET DATE TIME ----------
FUNCTION1:
        MOV DPTR,#F1L1
        CALL INMEM
        MOV LINE,#1
        CALL SCAN
        MOV DPTR,#F1L2
```

```
        CALL INMEM
        MOV LINE,#2
        CALL SCAN
        CALL INKEY
        DEC A
        MOV B,#3
        MUL AB
        MOV DPTR,#SET_DATETIME
        JMP @A+DPTR
SET_DATETIME:
        JMP SET_TIME
        JMP SET_DATE
        JMP 0000
        JMP 0
        JMP 0
        JMP 0
SET_TIME:
        MOV DPTR,#STIMEL2
        CALL INMEM
        CALL SCAN
        CALL SHOWTIME
IN_TIME:
        CALL INKEY
CHECK_HOUR:
        CJNE A,#1,CHECK_MINUTE
        MOV A,HOUR
        INC A
        DA A
        MOV B,A
        CLR CY
```

```
        SUBB A,#24H
        JC SET_HOURP
        MOV B,#0
SET_HOURP:
        MOV HOUR,B
        CALL WTIME
        JMP SET_TIME
CHECK_MINUTE:
        CJNE A,#2,CHECK_SEC
        MOV A,MINUTE
        INC A
        DA A
        MOV B,A
        CLR CY
        SUBB A,#60H
        JC SET_MINUTEP
        MOV B,#0
SET_MINUTEP:
        MOV MINUTE,B
        CALL WTIME
        JMP SET_TIME
CHECK_SEC:
        CJNE A,#3,CHECK_SET
        MOV A,SEC
        INC A
        DA A
        MOV B,A
        CLR CY
        SUBB A,#60H
        JC SET_SECP
```

```
        MOV B,#0
SET_SECP:
        MOV SEC,B
        CALL WTIME
        JMP SET_TIME
CHECK_SET:
        CJNE A,#4,SET_TIME
        JMP FUNCTION1
SET_DATE:
        MOV DPTR,#SDATEL2
        CALL INMEM
        CALL SCAN
        CALL SHOWTIME
IN_DATE:
        CALL INKEY
CHECK_DATE:
        CJNE A,#1,CHECK_MOUNTH
        MOV A,DATE
        INC A
        DA A
        MOV B,A
        CLR CY
        SUBB A,#32H
        JC SET_DATEP
        MOV B,#0
SET_DATEP:
        MOV DATE,B
        CALL WTIME
        JMP SET_DATE
CHECK_MOUNTH:
```

```
        CJNE A,#2,CHECK_YEAR
        MOV A,MOUNTH
        INC A
        DA A
        MOV B,A
        CLR CY
        SUBB A,#13H
        JC SET_MOUNTHP
        MOV B,#0
SET_MOUNTHP:
        MOV MOUNTH,B
        CALL WTIME
        JMP SET_DATE
CHECK_YEAR:
        CJNE A,#3,CHECK_SET2
        MOV A,YEAR
        INC A
        DA A
        MOV YEAR,A
        CALL WTIME
        JMP SET_DATE
CHECK_SET2:
        CJNE A,#4,SET_DATE
        JMP FUNCTION1
INKEY:
    MOV A,#1
    ORL P2,#00011111B
    CLR P2.1
    JNB P2.2,SET_DATA
    INC A
```

```
        JNB P2.3,SET_DATA
        INC A
        JNB P2.4,SET_DATA
        ORL P2,#00011111B
        CLR P2.0
        INC A
        JNB P2.2,SET_DATA
        INC A
        JNB P2.3,SET_DATA
        JMP INKEY
SET_DATA:
        JNB P2.2,$
        JNB P2.3,$
        JNB P2.4,$
        RET
; ********** HTOD SUB **********
; HEX TO DECIMAL
; IN  = DPTR
; OUT = R1,R2,R3
; REG = A,R0,R1,R2,R3,R4,R5,DPTR
HTOD:   CLR   A             ;CLEAR OUTPUT
        MOV   R1,A
        MOV   R2,A
        MOV   R3,A
        MOV   R4,#16        ;SHIFT 16 BIT
HTOD1:  MOV   A,DPL
        RLC   A
        MOV   DPL,A
        MOV   A,DPH
        RLC   A
```

```
        MOV   DPH,A
        MOV   R5,#3          ;ADD DECIMAL
        MOV   R0,#3          ;INDEX TO R3
HTOD2:  MOV   A,@R0
        ADDC  A,ACC
        DA    A
        MOV   @R0,A
        DEC   R0
        DJNZ  R5,HTOD2
        DJNZ  R4,HTOD1
        RET
; ********** DTOH SUB **********
; DECIMAL TO HEX
; IN  = R1,R2,R3
; OUT = DPTR
; REG = A,R0,R1,R2,R3,R4,R5,DPTR
DTOH:   MOV   R4,#16
DTOH1:  MOV   R5,#3          ;SHIFT & SUB
        MOV   R0,#1          ;INDEX TO R1
        CLR   C
DTOH2:  MOV   A,@R0
        RRC   A
        PUSH  PSW            ;-[
        JNB   ACC.7,DTOH3
        CLR   C
        SUBB  A,#30H
DTOH3:  JNB   ACC.3,DTOH4
        CLR   C
        SUBB  A,#03H
DTOH4:  MOV   @R0,A
```

```
    INC   R0
    POP   PSW          ;-]
    DJNZ  R5,DTOH2
    MOV   A,DPH
    RRC   A
    MOV   DPH,A
    MOV   A,DPL
    RRC   A
    MOV   DPL,A
    DJNZ  R4,DTOH1
    RET
SHOWTIME:
    MOV DPTR,#L1FROM
    CALL INMEM
    CALL TIME_DISPLAY
    MOV A,HOUR
    CALL BCDTOHEX
    MOV 40H,HEX_MSB
    MOV 41H,HEX_LSB
    MOV A,MINUTE
    CALL BCDTOHEX
    MOV 43H,HEX_MSB
    MOV 44H,HEX_LSB
    MOV A,SEC
    CALL BCDTOHEX
    MOV 46H,HEX_MSB
    MOV 47H,HEX_LSB
    CALL DATE_DISPLAY
    MOV A,DATE
    CALL BCDTOHEX
```

```
    MOV  4BH,HEX_MSB
    MOV  4CH,HEX_LSB
    MOV  A,MOUNTH
    CALL BCDTOHEX
    MOV  4EH,HEX_MSB
    MOV  4FH,HEX_LSB
    MOV  A,YEAR
    CALL BCDTOHEX
    MOV  51H,HEX_MSB
    MOV  52H,HEX_LSB
    MOV LINE,#1
    CALL SCAN
    RET
SEND_DATA:
SENDTX:
;   SERIAL PORT ( MODE 1 ) 9600 BAUD Tx
    PUSH ACC
    SETB SEND
    MOV PCON,#00H       ; SMOD = 0
    MOV SCON,#40H       ; serial (Mode1)
    MOV TMOD,#20H       ; Timer  (Mode0)
    MOV TH1,#0FDH       ; buad 9600
    SETB TR1
    CLR TI
    MOV A,HOUR
    CALL BCDTOHEX
    MOV A,HEX_MSB
    ANL A,#0FH
    ORL A,#00H
    MOV SBUF,A
```

```
JNB TI,$
CLR TI
MOV A,#1
CALL DELAY_MS
MOV A,HEX_LSB
ANL A,#0FH
ORL A,#10H
MOV SBUF,A
JNB TI,$
CLR TI
MOV A,#1
CALL DELAY_MS
      ;SEND  HOUR
MOV A,MINUTE
CALL BCDTOHEX
MOV A,HEX_MSB
ANL A,#0FH
ORL A,#20H
MOV SBUF,A
JNB TI,$
CLR TI
MOV A,#1
CALL DELAY_MS
MOV A,HEX_LSB
ANL A,#0FH
ORL A,#30H
MOV SBUF,A
JNB TI,$
CLR TI
MOV A,#1
```

```
CALL DELAY_MS
MOV A,DATE
CALL BCDTOHEX
MOV A,HEX_MSB
ANL A,#0FH
ORL A,#40H
MOV SBUF,A
JNB TI,$
CLR TI
MOV A,#1
CALL DELAY_MS
MOV A,HEX_LSB
ANL A,#0FH
ORL A,#50H
MOV SBUF,A
JNB TI,$
CLR TI
MOV A,#1
CALL DELAY_MS
MOV A,MOUNTH
CALL BCDTOHEX
MOV A,HEX_MSB
ANL A,#0FH
ORL A,#60H
MOV SBUF,A
JNB TI,$
CLR TI
MOV A,#1
CALL DELAY_MS
MOV A,HEX_LSB
```

```
ANL A,#0FH
ORL A,#70H
MOV SBUF,A
JNB TI,$
CLR TI
MOV A,#1
CALL DELAY_MS
MOV A,YEAR
CALL BCDTOHEX
MOV A,HEX_MSB
ANL A,#0FH
ORL A,#080H
MOV SBUF,A
JNB TI,$
CLR TI
MOV A,#1
CALL DELAY_MS
MOV A,HEX_LSB
ANL A,#0FH
ORL A,#090H
MOV SBUF,A
JNB TI,$
CLR TI
MOV A,#1
CALL DELAY_MS
MOV DPH,#0
MOV DPL,COIN_SEND
CALL HTOD
MOV A,R3
CALL BCDTOHEX
```

```
MOV A,HEX_MSB
ANL A,#0FH
ORL A,#0A0H
MOV SBUF,A
JNB TI,$
CLR TI
MOV A,#1
CALL DELAY_MS
MOV A,HEX_LSB
ANL A,#0FH
ORL A,#0B0H
MOV SBUF,A
JNB TI,$
CLR TI
MOV A,#1
CALL DELAY_MS
MOV A,COIL2_MSB
CALL BCDTOHEX
MOV A,HEX_MSB
ANL A,#0FH
ORL A,#0C0H
MOV SBUF,A
JNB TI,$
CLR TI
MOV A,#1
CALL DELAY_MS
MOV A,HEX_LSB
ANL A,#0FH
ORL A,#0D0H
MOV SBUF,A
```

```
    JNB TI,$
    CLR TI
    MOV A,#1
    CALL DELAY_MS
    MOV A,COIL2_LSB
    CALL BCDTOHEX
    MOV A,HEX_MSB
    ANL A,#0FH
    ORL A,#0E0H
    MOV SBUF,A
    JNB TI,$
    CLR TI
    MOV A,#1
    CALL DELAY_MS
    MOV A,HEX_LSB
    ANL A,#0FH
    ORL A,#0F0H
    MOV SBUF,A
    JNB TI,$
    CLR TI
    MOV A,#1
    CALL DELAY_MS
    POP ACC
    RET
DETECT_COIL:
    JNB SENSOR,$
    MOV A,#7FH
    CALL DELAY_MS
    MOV SP,#09H
    MOV 08H,#LOW IN_COIL
```

```
    MOV 09H,#HIGH IN_COIL
    RETI
BCDTOHEX:
;IN PUT ACC
   PUSH ACC
   SWAP A
   ANL A,#0FH
   ADD A,#30H
   MOV HEX_MSB,A
   POP ACC
   ANL A,#0FH
   ADD A,#30H
   MOV HEX_LSB,A
   RET
TIME_DISPLAY:
      MOV R2,#81H          ;read time from rtc
      CALL RTC_READ_CH     ;READ SECOUND
      MOV SEC,R3
      MOV R2,#83H
      CALL RTC_READ_CH     ;read minute
      MOV MINUTE,R3
      MOV R2,#85H          ;read HOUR
      CALL RTC_READ_CH
      MOV HOUR,R3
      RET
DATE_DISPLAY:
      MOV R2,#87H
      CALL RTC_READ_CH     ;DATE
      MOV DATE,R3
      MOV R2,#89H
```

```
    CALL RTC_READ_CH      ;MOUNTH
    MOV MOUNTH,R3
    MOV R2,#8DH
    CALL RTC_READ_CH      ;YEAR
    MOV YEAR,R3
    RET
RTC_DATA EQU  P1.3
RTC_CLK  EQU  P1.1
RTC_RST  EQU  P1.2
RTC_WRITE_CH:
    CLR   RTC_CLK      ;low CLK line
    CALL  DELAY
    SETB  RTC_RST      ;high RST line
    CALL  DELAY
    MOV   A,R2         ;write COMMAND byte
    CALL  RTC_WRITE_8BIT
    MOV   A,R3         ;and DATA byte
    CALL  RTC_WRITE_8BIT
    CLR   RTC_RST      ;low RST line
    CALL  DELAY
    RET
; Transfer 8-bits data to DS1202
; IN: A = data
RTC_WRITE_8BIT:
    MOV   R4,#08H      ;8 bits to transfer
WR8BIT1:
    RRC   A
    MOV   RTC_DATA,C
    SETB  RTC_CLK      ;Rising edge clock
    CALL  DELAY
```

```
    CLR   RTC_CLK
    CALL  DELAY
    DJNZ  R4,WR8BIT1
    RET
; Perform READ Command and read parameter byte from DS1202 RTC
; IN:  R2 = COMMAND
; OUT: R3 = DATA
RTC_READ_CH:
    CLR   RTC_CLK      ;low clk line
    CALL  DELAY
    SETB  RTC_RST      ;high RST line
    CALL  DELAY
    MOV   A,R2         ;write COMMAND byte
    CALL  RTC_WRITE_8BIT
;
    MOV   R4,#8        ;then read DATA byte
    CLR   A
RD_CH1: CLR   RTC_CLK
    CALL  DELAY
    MOV   C,RTC_DATA
    RRC   A
    SETB  RTC_CLK
    CALL  DELAY
    DJNZ  R4,RD_CH1
    MOV   R3,A
;
    CLR   RTC_RST      ;low RST line
    CALL  DELAY
    RET
DELAY:
```

```
    NOP     ;delay
    RET
WTIME:
    MOV R2,#8EH
    MOV R3,#00H
    CALL RTC_WRITE_CH
    MOV R2,#80H
    MOV R3,SEC              ;SEC
;   MOV R3,#00
    CALL RTC_WRITE_CH
    MOV R2,#82H
    MOV R3,MINUTE           ;MIN
    CALL RTC_WRITE_CH
    MOV R2,#84H             ;HOUR
    MOV R3,HOUR
    CALL RTC_WRITE_CH
    MOV R2,#86H             ;DATE
    MOV R3,DATE
    CALL RTC_WRITE_CH
    MOV R2,#88H             ;MOUNTH
    MOV R3,MOUNTH
    CALL RTC_WRITE_CH
    MOV R2,#8CH             ;YEAR
    MOV R3,YEAR
    CALL RTC_WRITE_CH
    MOV R2,#8EH
    MOV R3,#80H
    CALL RTC_WRITE_CH
    RET
```

```
SCAN:
    PUSH DPH
    PUSH DPL
    PUSH 00
    MOV A,#80H
;SET LINE
    CLR RS
    CJNE R3,#1,LINE2
    MOV DATA_LCD,#80H
    JMP XLINE
LINE2:
    MOV DATA_LCD,#0C0H
XLINE:
    CALL EPLUSE
    MOV B,#20
    MOV R0,#40H
 L1:
    MOV  A,@R0
    CALL WRITE
    INC R0
    DJNZ B,L1
    POP 00
    POP DPL
    POP DPH
    RET
INMEM:
;  DPTR IS OFFSET DATA TO 40 ==> 53H (16)
;     PUSH DPH
;     PUSH DPL
    MOV R2,#20
```

```
        MOV R0,#40H
    IN1: MOV A,#0
        MOVC A,@A+DPTR
        MOV @R0,A
        INC DPTR
        INC R0
        DJNZ R2,IN1
    ;    POP DPL
    ;    POP DPH
        RET
    WRITE:
        PUSH ACC
        SETB RS
        POP ACC
        MOV DATA_LCD,A
        CALL EPLUSE
        CALL WAITBF
        RET
    EPLUSE: SETB ENABLE
        PUSH ACC
        MOV A,#01
        CALL DELAY_MS
        CLR  ENABLE
        POP ACC
        RET
    WAITBF:
       PUSH ACC
       MOV A,#4
       CALL DELAY_MS
       POP ACC
```

```
    RET
INIT:
    CLR RS
    MOV DATA_LCD,#01
    MOV DATA_LCD,#38H     ;DISPLAY 5*7 DOT 1 LINE
    CALL EPLUSE
    CALL WAITBF
    MOV DATA_LCD,#0FH
    CALL EPLUSE
    CALL WAITBF
    MOV DATA_LCD,#6
    CALL EPLUSE
    CALL WAITBF
    MOV DATA_LCD,#1
    CALL EPLUSE
    CALL WAITBF
    MOV DATA_LCD,#80H
    CALL EPLUSE
    CALL WAITBF
    RET
$INCLUDE "DELAY.ASM"
END
```

ภาคผนวก ข(2).
โปรแกรมใช้งาน SLAVE

```
;PROGRAM FOR SLAVE COIN
LINE    EQU 03
 RS     EQU P3.3
 ENABLE  EQU P3.7
 DATA_LCD EQU P1
 SEND    EQU P3.2
 DELAY_SCAN EQU 07FH
     ORG 0000H
     JMP STARTX
       ORG 0023H            ;INTER VECTOR RX TX
       PUSH ACC
       PUSH 00
       MOV R0,#30H
       MOV A,SBUF
       PUSH ACC
       ANL A,#0F0H
       SWAP A
       ORL 00,A
       POP ACC
       ANL A,#0FH
       ORL A,#00110000B
       MOV @R0,A
       CLR RI
       POP 00
       POP ACC
       RETI
STARTX:
     CLR SEND
     MOV PCON,#00H        ; SMOD 0
     MOV SCON,#50H
```

```
    MOV TH1,#0FDH
    MOV TMOD,#20H
    SETB TR1
    MOV IE,#90H
    CALL INIT    ;INIT PAGE
STARTY:
    MOV  DPTR,#L1FROM
    CALL INMEM
    MOV 40H,30H
    MOV 41H,31H
    MOV 43H,32H
    MOV 44H,33H
    MOV 48H,34H
    MOV 49H,35H
    MOV 4BH,36H
    MOV 4CH,37H
    MOV 4EH,38H
    MOV 4FH,39H
    MOV  LINE,#1
    CALL SCAN
    MOV DPTR,#L2FROM
    CALL INMEM
    MOV 4EH,3AH
    MOV 4FH,3BH
    MOV 43H,3CH
    MOV 44H,3DH
    MOV 45H,3EH
    MOV 46H,3FH
    MOV LINE,#2
    CALL SCAN
```

```
    JMP STARTY
CLEAR_D:  DB "              "
TEST1:    DB "0123456789ABCDEF"
TEST2:    DB "ABCDEFGHIJKLMNOT"
L1FROM:   DB "HH:MM. DD/MM/YY"
L2FROM:   DB "TC=0000 LCOIN=00"
SCAN:
    PUSH DPH
    PUSH DPL
    PUSH 00
    MOV A,#80H
;SET LINE
    CLR RS
    CJNE R3,#1,LINE2
    MOV DATA_LCD,#80H
    JMP XLINE
LINE2:
    MOV DATA_LCD,#0C0H
XLINE:
    CALL EPLUSE
    MOV B,#20
    MOV R0,#40H
 L1:
    MOV  A,@R0
    CALL WRITE
    INC R0
    DJNZ B,L1
    POP 00
    POP DPL
    POP DPH
```

```
        RET
INMEM:
;  DPTR IS OFFSET DATA TO 40 ==> 4FH (16)
;       PUSH DPH
;       PUSH DPL
        MOV R2,#16
        MOV R0,#40H
  IN1: MOV A,#0
        MOVC A,@A+DPTR
        MOV @R0,A
        INC DPTR
        INC R0
        DJNZ R2,IN1
;       POP DPL
;       POP DPH
        RET
WRITE:
       PUSH ACC
       SETB RS
       POP ACC
       MOV DATA_LCD,A
       CALL EPLUSE
       CALL WAITBF
       RET
EPLUSE: SETB ENABLE
        PUSH ACC
        MOV A,#01
        CALL DELAY_MS
        CLR  ENABLE
        POP ACC
```

```
    RET
WAITBF:
    PUSH ACC
    MOV A,#4
    CALL DELAY_MS
    POP ACC
    RET
INIT:
    CLR RS
    MOV DATA_LCD,#01
    MOV DATA_LCD,#38H    ;DISPLAY 5*7 DOT 1 LINE
    CALL EPLUSE
    CALL WAITBF
    MOV DATA_LCD,#0FH
    CALL EPLUSE
    CALL WAITBF
    MOV DATA_LCD,#6
    CALL EPLUSE
    CALL WAITBF
    MOV DATA_LCD,#1
    CALL EPLUSE
    CALL WAITBF
    MOV DATA_LCD,#80H
    CALL EPLUSE
    CALL WAITBF
    RET
 $INCLUDE "DELAY.ASM"
 END
```

ภาคผนวก ข(3).

โปรแกรมใช้งาน DETECT COIN

```
;PROGRAM FOR CONTROL SOLINOID
PULSE   EQU  P3.7
CONT    EQU  P3.0
STATS   EQU  P3.1
KEY1    EQU  P1.0
KEY2    EQU  P1.1
;-------------------------------------------------
COUNTER  EQU  30H
     ORG 0000
;---------- INIT STATE -------------------------
     SETB PULSE
     CLR CONT
     MOV COUNTER,#0
     CLR STATS
     MOV A,#100
     CALL DELAY_MS
     SETB STATS
     CALL DELAY_MS
     CLR STATS
     CALL DELAY_MS
     SETB STATS
     CALL DELAY_MS
     CLR STATS
;-------------------------------------------------
START:
     SETB PULSE
     JB PULSE,$
     JNB PULSE,$
     SETB STATS
     MOV A,#100
```

```
    CALL DELAY_MS
    CLR STATS
    INC COUNTER
    MOV A,COUNTER
    CJNE A,#8,START
    SETB CONT
CONT1:
    SETB KEY1
    SETB KEY2
    JNB KEY1,0
    JNB KEY2,0
    JMP CONT1
;PROGRAM DELAY TIME
;USED REGISTER A
;SUB DELAY_SEC AND DELAY_MS
;PROGRAM BY TEERAPON AND TEAM , KMITL
DELAY_MS:
            ;MOV ANC  CALL 1.085 X 3 uS
    PUSH ACC      ;2.170138889 uS
    PUSH B        ;2.170138889 uS
    MOV B,#231    ;2.170138889 uS
DD:
    DJNZ B,$      ;500 uS AT 11.0592 MHZ
    DJNZ B,$      ;500 uS AT 11.0592 MHZ
    DJNZ ACC,DD   ;2.170138889 uS
    POP B         ;2.170138889 uS
    POP ACC       ;2.170138889 uS
    RET           ;2.170138889 uS
DELAY_SEC:
    PUSH ACC
```

```
    PUSH B
    MOV B,A
DDD:
    MOV A,#246
    CALL DELAY_MS   ;250 mS
    CALL DELAY_MS   ;500 mS
    CALL DELAY_MS   ;750 mS
    CALL DELAY_MS   ;1000 mS
    DJNZ B,DDD
    POP B
    POP ACC
    RET
END
```

□

ภาคผนวก ค.

DATA SHEET

Features

- Compatible with MCS-51™ Products
- 2K Bytes of Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- 2.7V to 6V Operating Range
- Fully Static Operation: 0 Hz to 24 MHz
- Two-Level Program Memory Lock
- 128 x 8-Bit Internal RAM
- 15 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial UART Channel
- Direct LED Drive Outputs
- On-Chip Analog Comparator
- Low Power Idle and Power Down Modes

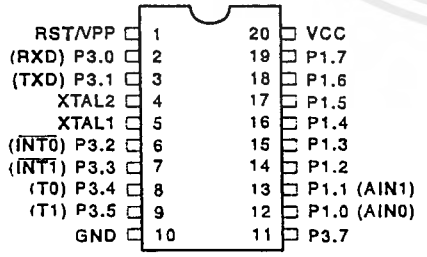
Description

The AT89C2051 is a low-voltage, high-performance CMOS 8-bit microcomputer with 2K Bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high density nonvolatile memory technology and is compatible with the industry standard MCS-51™ instruction set. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C2051 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

The AT89C2051 provides the following standard features: 2K Bytes of Flash, 128 bytes of RAM, 15 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, a precision analog comparator, on-chip oscillator and clock circuitry. In addition, the AT89C2051 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The Power Down Mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

Pin Configuration

PDIP/SOIC

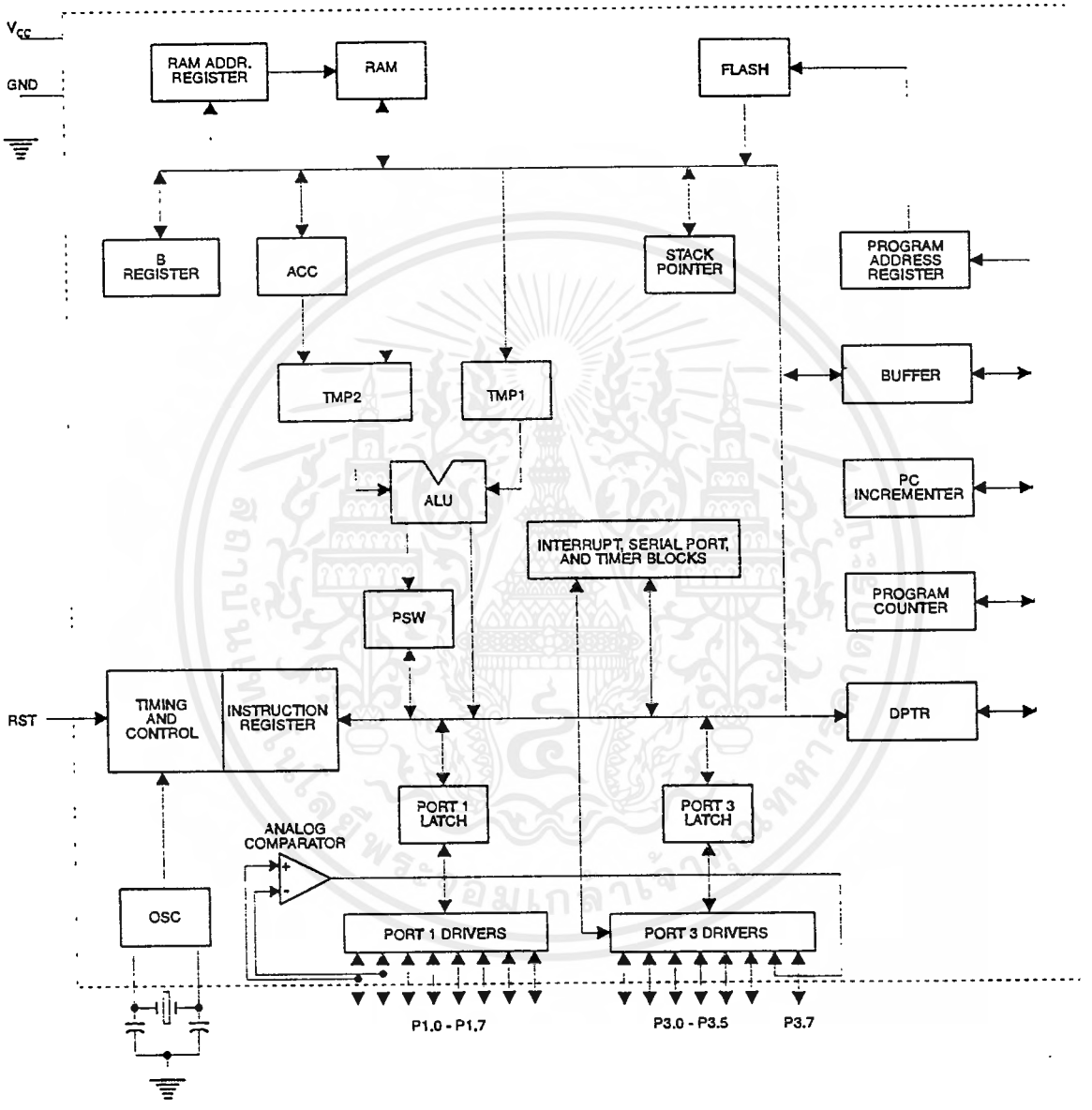


8-Bit Microcontroller with 2K Bytes Flash

AT89C2051

0368D-B-12/97

Block Diagram



Pin Description

V_{CC}
Supply voltage.

GND
Ground.

Port 1
Port 1 is an 8-bit bidirectional I/O port. Port pins P1.2 to P1.7 provide internal pullups. P1.0 and P1.1 require external pullups. P1.0 and P1.1 also serve as the positive input (AIN0) and the negative input (AIN1), respectively, of the on-chip precision analog comparator. The Port 1 output buffers can sink 20 mA and can drive LED displays directly. When 1s are written to Port 1 pins, they can be used as inputs. When pins P1.2 to P1.7 are used as inputs and are externally pulled low, they will source current (I_{IL}) because of the internal pullups.

Port 1 also receives code data during Flash programming and verification.

Port 3
Port 3 pins P3.0 to P3.5, P3.7 are seven bidirectional I/O pins with internal pullups. P3.6 is hard-wired as an input to the output of the on-chip comparator and is not accessible as a general purpose I/O pin. The Port 3 output buffers can sink 20 mA. When 1s are written to Port 3 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89C2051 as listed below:

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)

Port 3 also receives some control signals for Flash programming and verification.

RST
Reset input. All I/O pins are reset to 1s as soon as RST goes high. Holding the RST pin high for two machine cycles while the oscillator is running resets the device.

Each machine cycle takes 12 oscillator or clock cycles.

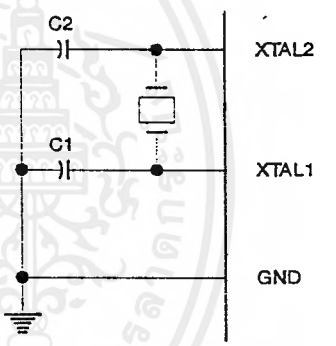
XTAL1
Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2
Output from the inverting oscillator amplifier.

Oscillator Characteristics

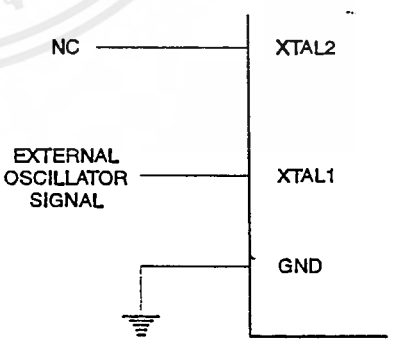
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Figure 1. Oscillator Connections



Note: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Figure 2. External Clock Drive Configuration



Pin Description

V_{CC}
Supply voltage.

GND
Ground.

Port 1
Port 1 is an 8-bit bidirectional I/O port. Port pins P1.2 to P1.7 provide internal pullups. P1.0 and P1.1 require external pullups. P1.0 and P1.1 also serve as the positive input (AIN0) and the negative input (AIN1), respectively, of the on-chip precision analog comparator. The Port 1 output buffers can sink 20 mA and can drive LED displays directly. When 1s are written to Port 1 pins, they can be used as inputs. When pins P1.2 to P1.7 are used as inputs and are externally pulled low, they will source current (I_{IL}) because of the internal pullups.

Port 1 also receives code data during Flash programming and verification.

Port 3
Port 3 pins P3.0 to P3.5, P3.7 are seven bidirectional I/O pins with internal pullups. P3.6 is hard-wired as an input to the output of the on-chip comparator and is not accessible as a general purpose I/O pin. The Port 3 output buffers can sink 20 mA. When 1s are written to Port 3 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89C2051 as listed below:

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)

Port 3 also receives some control signals for Flash programming and verification.

RST
Reset input. All I/O pins are reset to 1s as soon as RST goes high. Holding the RST pin high for two machine cycles while the oscillator is running resets the device.

Each machine cycle takes 12 oscillator or clock cycles.

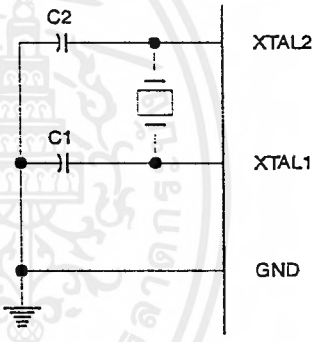
XTAL1
Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2
Output from the inverting oscillator amplifier.

Oscillator Characteristics

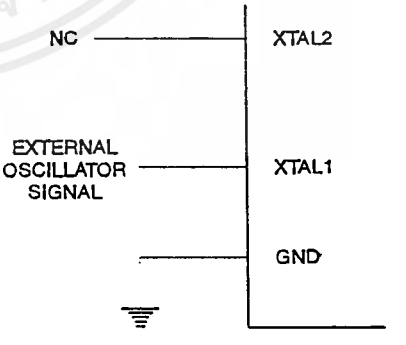
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Figure 1. Oscillator Connections



Note: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Figure 2. External Clock Drive Configuration



Special Function Registers

A map of the on-chip memory area called the Special Function Register (SFR) space is shown in the table below.

Note that not all of the addresses are occupied, and unoccupied addresses may not be implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have an indeterminate effect.

User software should not write 1s to these unlisted locations, since they may be used in future products to invoke new features. In that case, the reset or inactive values of the new bits will always be 0.

Table 1. AT89C2051 SFR Map and Reset Values

0F8H								0FFH
0F0H	B 00000000							0F7H
0E8H								0EFH
0E0H	ACC 00000000							0E7H
0D8H								0DFH
0D0H	PSW 00000000							0D7H
0C8H								0CFH
0C0H								0C7H
0B8H	IP XXX00000							0BFH
0B0H	P3 11111111							0B7H
0A8H	IE 0XX00000							0AFH
0A0H								0A7H
98H	SCON 00000000	SBUF XXXXXXXX						9FH
90H	P1 11111111							97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000		8FH
80H		SP 00000111	DPL 00000000	DPH 00000000			PCON 0XXX0000	87H

Restrictions on Certain Instructions

The AT89C2051 and is an economical and cost-effective member of Atmel's growing family of microcontrollers. It contains 2K bytes of flash program memory. It is fully compatible with the MCS-51 architecture, and can be programmed using the MCS-51 instruction set. However, there are a few considerations one must keep in mind when utilizing certain instructions to program this device.

All the instructions related to jumping or branching should be restricted such that the destination address falls within the physical program memory space of the device, which is 2K for the AT89C2051. This should be the responsibility of the software programmer. For example, LJMP 7E0H would be a valid instruction for the AT89C2051 (with 2K of memory), whereas LJMP 900H would not.

1. Branching instructions:
LCALL, LJMP, ACALL, AJMP, SJMP, JMP @A+DPTR

These unconditional branching instructions will execute correctly as long as the programmer keeps in mind that the destination branching address must fall within the physical boundaries of the program memory size (locations 00H to 7FFH for the 89C2051). Violating the physical space limits may cause unknown program behavior.

CJNE [...], DJNZ [...], JB, JNB, JC, JNC, JBC, JZ, JNZ With these conditional branching instructions the same rule above applies. Again, violating the memory boundaries may cause erratic execution.

For applications involving interrupts the normal interrupt service routine address locations of the 80C51 family architecture have been preserved.

2. MOVX-related instructions, Data Memory:
The AT89C2051 contains 128 bytes of internal data memory. Thus, in the AT89C2051 the stack depth is limited to 128 bytes, the amount of available RAM. External DATA memory access is not supported in this device, nor is external PROGRAM memory execution. Therefore, no MOVX [...] instructions should be included in the program.

A typical 80C51 assembler will still assemble instructions, even if they are written in violation of the restrictions mentioned above. It is the responsibility of the controller user to know the physical features and limitations of the device being used and adjust the instructions used correspondingly.

Program Memory Lock Bits

On the chip are two lock bits which can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the table below:

Lock Bit Protection Modes[1]

Program Lock Bits			
	LB1	LB2	Protection Type
1	U	U	No program lock features.
2	P	U	Further programming of the Flash is disabled.
3	P	P	Same as mode 2, also verify is disabled.

Note: 1. The Lock Bits can only be erased with the Chip Erase operation.

Idle Mode

In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

P1.0 and P1.1 should be set to '0' if no external pullups are used, or set to '1' if external pullups are used.

It should be noted that when idle is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

Power Down Mode

In the power down mode the oscillator is stopped, and the instruction that invokes power down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the power down mode is terminated. The only exit from power down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

P1.0 and P1.1 should be set to '0' if no external pullups are used, or set to '1' if external pullups are used.

Programming The Flash

The AT89C2051 is shipped with the 2K bytes of on-chip PEROM code memory array in the erased state (i.e., contents = FFH) and ready to be programmed. The code memory array is programmed one byte at a time. *Once the array is programmed, to re-program any non-blank byte, the entire memory array needs to be erased electrically.*

Internal Address Counter: The AT89C2051 contains an internal PEROM address counter which is always reset to 000H on the rising edge of RST and is advanced by applying a positive going pulse to pin XTAL1.

Programming Algorithm: To program the AT89C2051, the following sequence is recommended.

1. Power-up sequence:
   Apply power between V_{CC} and GND pins
   Set RST and XTAL1 to GND
2. Set pin RST to 'H'
   Set pin P3.2 to 'H'
3. Apply the appropriate combination of 'H' or 'L' logic levels to pins P3.3, P3.4, P3.5, P3.7 to select one of the programming operations shown in the PEROM Programming Modes table.

To Program and Verify the Array:

4. Apply data for Code byte at location 000H to P1.0 to P1.7.
5. Raise RST to 12V to enable programming.
6. Pulse P3.2 once to program a byte in the PEROM array or the lock bits. The byte-write cycle is self-timed and typically takes 1.2 ms.
7. To verify the programmed data, lower RST from 12V to logic 'H' level and set pins P3.3 to P3.7 to the appropiate levels. Output data can be read at the port P1 pins.
8. To program a byte at the next address location, pulse XTAL1 pin once to advance the internal address counter. Apply new data to the port P1 pins.
9. Repeat steps 5 through 8, changing data and advancing the address counter for the entire 2K bytes array or until the end of the object file is reached.
10. Power-off sequence:
    set XTAL1 to 'L'
    set RST to 'L'
    Turn V_{CC} power off

Data Polling: The AT89C2051 features Data Polling to indicate the end of a write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written data on P1.7. Once the write cycle has been completed, true data is valid on all outputs, and the next cycle may begin. Data Polling may begin any time after a write cycle has been initiated.

Ready/Busy: The Progress of byte programming can also be monitored by the RDY/BSY output signal. Pin P3.1 is pulled low after P3.2 goes High during programming to indicate BUSY. P3.1 is pulled High again when programming is done to indicate READY.

Program Verify: If lock bits LB1 and LB2 have not been programmed code data can be read back via the data lines for verification:

1. Reset the internal address counter to 000H by bringing RST from 'L' to 'H'.
2. Apply the appropriate control signals for Read Code data and read the output data at the port P1 pins.
3. Pulse pin XTAL1 once to advance the internal address counter.
4. Read the next code data byte at the port P1 pins.
5. Repeat steps 3 and 4 until the entire array is read.

The lock bits cannot be verified directly. Verification of the lock bits is achieved by observing that their features are enabled.

Chip Erase: The entire PEROM array (2K bytes) and the two Lock Bits are erased electrically by using the proper combination of control signals and by holding P3.2 low for 10 ms. The code array is written with all "1"s in the Chip Erase operation and must be executed before any non-blank memory byte can be re-programmed.

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 000H, 001H, and 002H, except that P3.5 and P3.7 must be pulled to a logic low. The values returned are as follows.

(000H) = 1EH indicates manufactured by Atmel
(001H) = 21H indicates 89C2051

Programming Interface

Every code byte in the Flash array can be written and the entire array can be erased by using the appropriate combination of control signals. The write operation cycle is self-timed and once initiated, will automatically time itself to completion.

All major programming vendors offer worldwide support for the Atmel microcontroller series. Please contact your local programming vendor for the appropriate software revision.

Flash Programming Modes

Mode		RST/VPP	P3.2/$\overline{PROG}$	P3.3	P3.4	P3.5	P3.7
Write Code Data[1][3]		12V		L	H	H	H
Read Code Data[1]		H	H	L	L	H	H
Write Lock	Bit - 1	12V		H	H	H	H
	Bit - 2	12V		H	H	L	L
Chip Erase		12V	(2)	H	L	L	L
Read Signature Byte		H	H	L	L	L	L

Notes:
1. The internal PEROM address counter is reset to 000H on the rising edge of RST and is advanced by a positive pulse at XTAL 1 pin.
2. Chip Erase requires a 10-ms $\overline{PROG}$ pulse.
3. P3.1 is pulled Low during programming to indicate RDY/$\overline{BSY}$.

Figure 3. Programming the Flash Memory

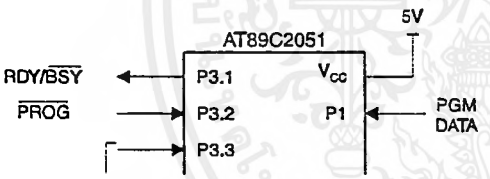
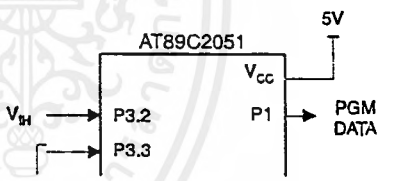


Figure 4. Verifying the Flash Memory



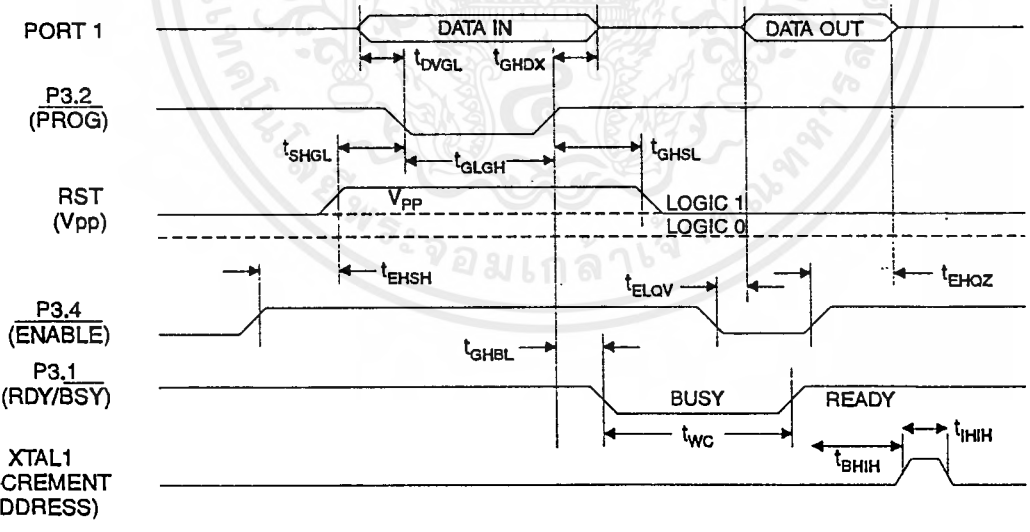
Flash Programming and Verification Characteristics

T_A = 0°C to 70°C, V_{CC} = 5.0 ± 10%

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Enable Voltage	11.5	12.5	V
I_{PP}	Programming Enable Current		250	µA
t_{DVGL}	Data Setup to $\overline{PROG}$ Low	1.0		µs
t_{GHDX}	Data Hold After $\overline{PROG}$	1.0		µs
t_{EHSH}	P3.4 ($\overline{ENABLE}$) High to V_{PP}	1.0		µs
t_{SHGL}	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
t_{GHSL}	V_{PP} Hold After $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ Width	1	110	µs
t_{ELQV}	$\overline{ENABLE}$ Low to Data Valid		1.0	µs
t_{EHQZ}	Data Float After $\overline{ENABLE}$	0	1.0	µs
t_{GHBL}	$\overline{PROG}$ High to $\overline{BUSY}$ Low		50	ns
t_{WC}	Byte Write Cycle Time		2.0	ms
t_{BHIH}	RDY/$\overline{BSY}$ to Increment Clock Delay	1.0		µs
t_{IHIL}	Increment Clock High	200		ns

Note: 1. Only used in 12-volt programming mode.

Flash Programming and Verification Waveforms



Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
DC Output Current	25.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

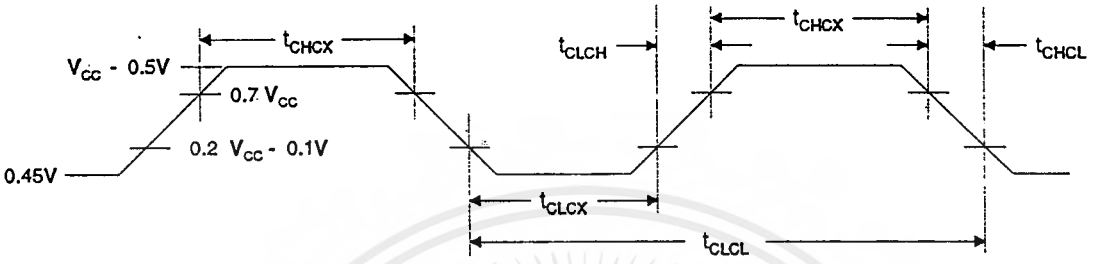
T_A = -40°C to 85°C, V_{CC} = 2.0V to 6.0V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low Voltage		-0.5	0.2 V_{CC} - 0.1	V
V_{IH}	Input High Voltage	(Except XTAL1, RST)	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V
V_{IH1}	Input High Voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC} + 0.5	V
V_{OL}	Output Low Voltage(1) (Ports 1, 3)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 2.7V		0.5	V
V_{OH}	Output High Voltage (Ports 1, 3)	I_{OH} = -80 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -30 µA	0.75 V_{CC}		V
		I_{OH} = -12 µA	0.9 V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1, 3)	V_{IN} = 0.45V		-50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1, 3)	V_{IN} = 2V, V_{CC} = 5V ± 10%		-750	µA
I_{LI}	Input Leakage Current (Port P1.0, P1.1)	0 < V_{IN} < V_{CC}		±10	µA
V_{OS}	Comparator Input Offset Voltage	V_{CC} = 5V		20	mV
V_{CM}	Comparator Input Common Mode Voltage		0	V_{CC}	V
RRST	Reset Pulldown Resistor		50	300	KΩ
C_{IO}	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
I_{CC}	Power Supply Current	Active Mode, 12 MHz, V_{CC} = 6V/3V		15/5.5	mA
		Idle Mode, 12 MHz, V_{CC} = 6V/3V P1.0 & P1.1 = 0V or V_{CC}		5/1	mA
	Power Down Mode(2)	V_{CC} = 6V P1.0 & P1.1 = 0V or V_{CC}		100	µA
		V_{CC} = 3V P1.0 & P1.1 = 0V or V_{CC}		20	µA

Notes: 1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
Maximum I_{OL} per port pin: 20 mA
Maximum total I_{OL} for all output pins: 80 mA
If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.

2. Minimum V_{CC} for Power Down is 2V.

External Clock Drive Waveforms



External Clock Drive

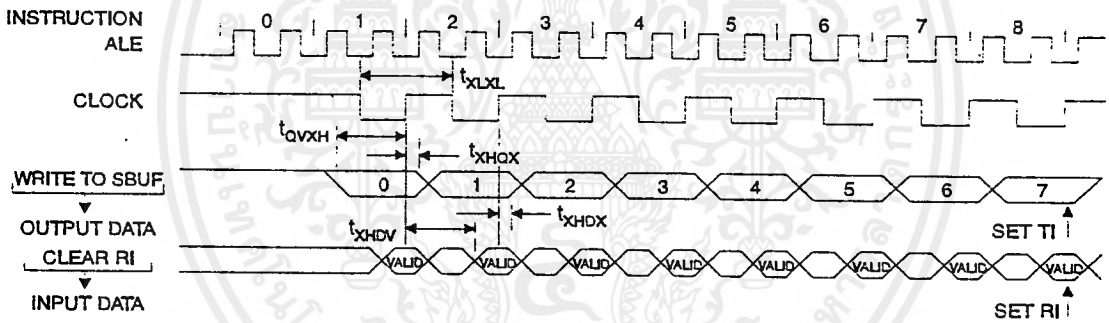
Symbol	Parameter	V_{CC} = 2.7V to 6.0V		V_{CC} = 4.0V to 6.0V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	12	0	24	MHz
t_{CLCL}	Clock Period	83.3		41.6		ns
t_{CHCX}	High Time	30		15		ns
t_{CLCX}	Low Time	30		15		ns
t_{CLCH}	Rise Time		20		20	ns
t_{CHCL}	Fall Time		20		20	ns

Serial Port Timing: Shift Register Mode Test Conditions

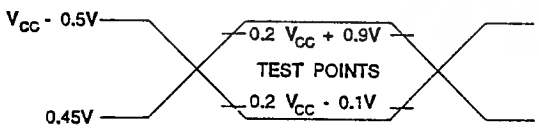
(V_{CC} = 5.0V ± 20%; Load Capacitance = 80 pF)

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
t_{XLXL}	Serial Port Clock Cycle Time	1.0		$12t_{CLCL}$		µs
t_{QVXH}	Output Data Setup to Clock Rising Edge	700		$10t_{CLCL}$-133		ns
t_{XHQX}	Output Data Hold After Clock Rising Edge	50		$2t_{CLCL}$-117		ns
t_{XHDX}	Input Data Hold After Clock Rising Edge	0		0		ns
t_{XHDV}	Clock Rising Edge to Input Data Valid		700		$10t_{CLCL}$-133	ns

Shift Register Mode Timing Waveforms

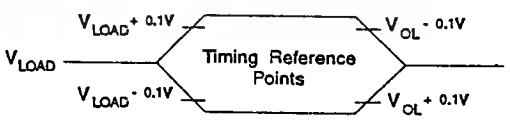


AC Testing Input/Output Waveforms[1]



Note: 1. AC Inputs during testing are driven at V_{CC} - 0.5V for a logic 1 and 0.45V for a logic 0. Timing measurements are made at V_{IH} min. for a logic 1 and V_{IL} max. for a logic 0.

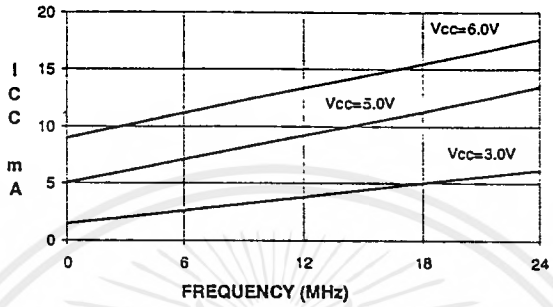
Float Waveforms[1]



Note: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when 100 mV change frothe loaded V_{OH}/V_{OL} level occurs.

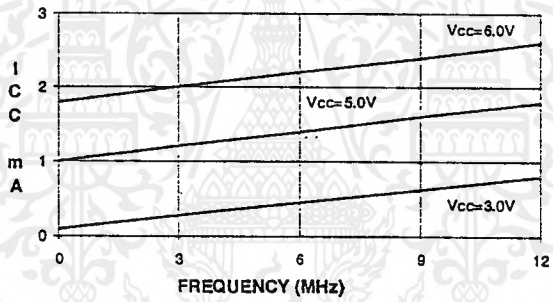
AT89C2051

TYPICAL ICC - ACTIVE (85°C)



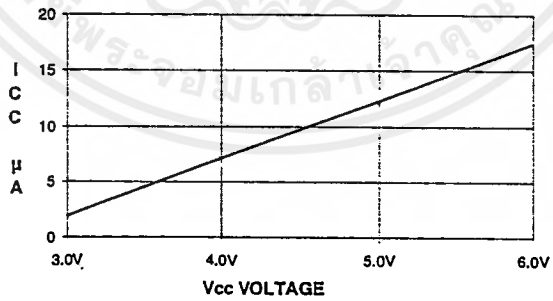
AT89C2051

TYPICAL ICC - IDLE (85°C)



AT89C2051

TYPICAL ICC vs. VOLTAGE- POWER DOWN (85°C)



Notes:
1. XTAL1 tied to GND for I_{CC} (power down)
2. P1.0 and P1.1 = V_{CC} or GND
3. Lock bits programmed

Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12	2.7V to 6.0V	AT89C2051-12PC AT89C2051-12SC	20P3 20S	Commercial (0°C to 70°C)
		AT89C2051-12PI AT89C2051-12SI	20P3 20S	Industrial (-40°C to 85°C)
		AT89C2051-12PA AT89C2051-12SA	20P3 20S	Automotive (-40°C to 105°C)
24	4.0V to 6.0V	AT89C2051-24PC AT89C2051-24SC	20P3 20S	Commercial (0°C to 70°C)
		AT89C2051-24PI AT89C2051-24SI	20P3 20S	Industrial (-40°C to 85°C)

Package Type	
20P3	20 Lead, 0.300" Wide, Plastic Dual In-line Package (PDIP)
20S	20 Lead, 0.300" Wide, Plastic Gull Wing Small Outline (SOIC)

Features

- Compatible with MCS-51™ Products
- 4K Bytes of In-System Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-Level Program Memory Lock
- 128 x 8-Bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial Channel
- Low Power Idle and Power Down Modes

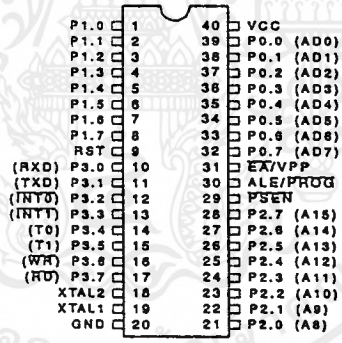
Description

The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4K bytes of Flash Programmable and Erasable Read Only Memory (PEROM). The device is manufactured using Atmel's high density nonvolatile memory technology and is compatible with the industry standard MCS-51™ instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

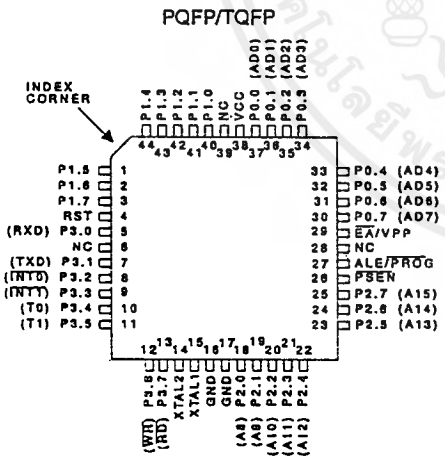
(continued)

Pin Configurations

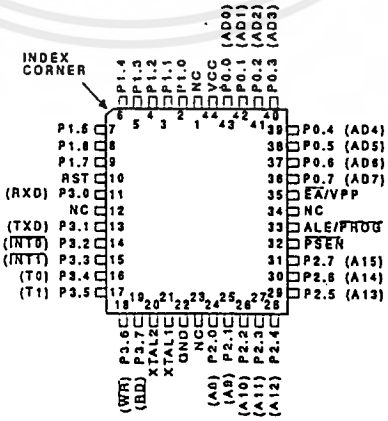
PDIP



PQFP/TQFP



PLCC

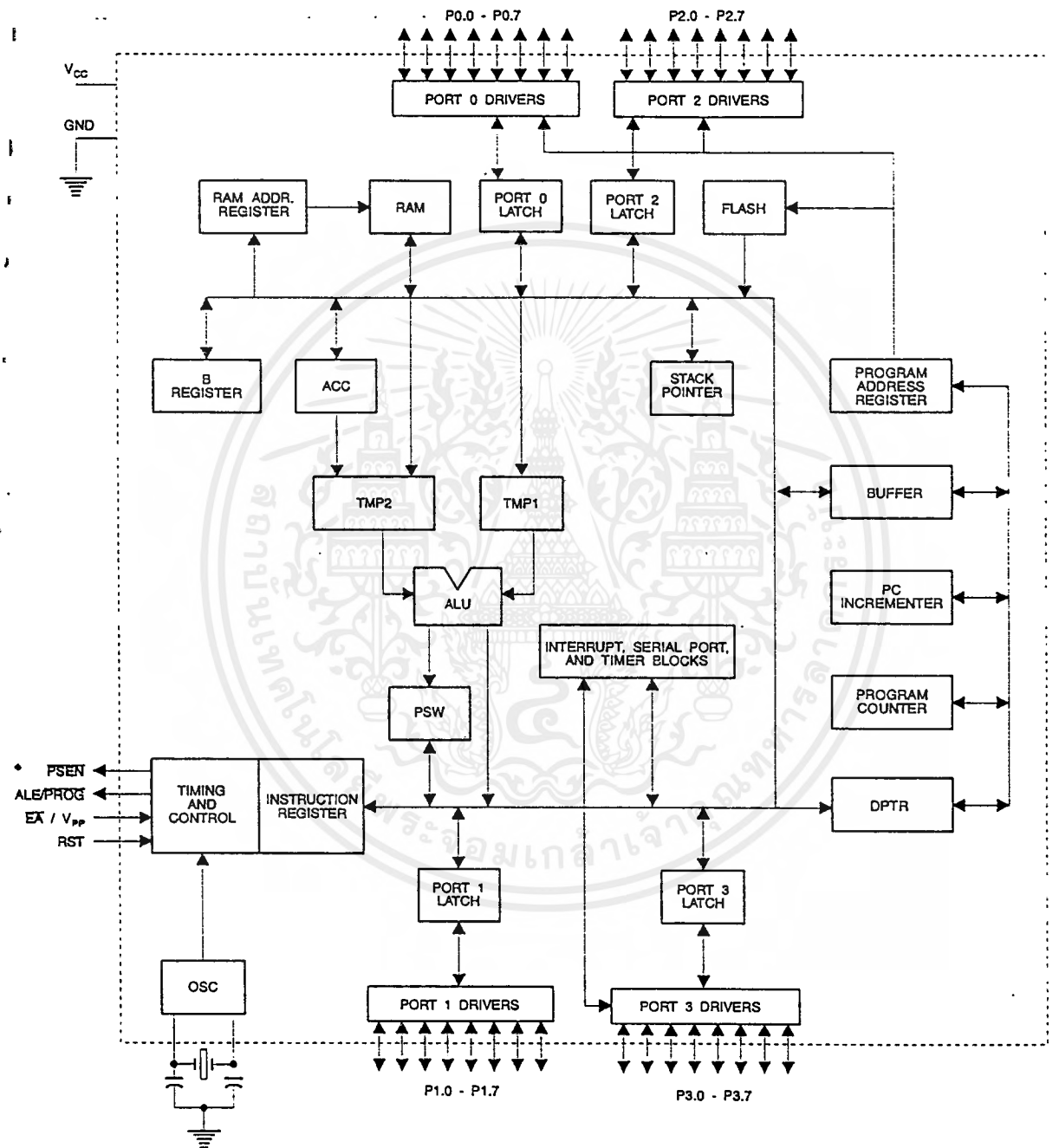


8-Bit Microcontroller with 4K Bytes Flash

AT89C51

0265F-A–12/97

lock Diagram



The AT89C51 provides the following standard features: 4K bytes of Flash, 128 bytes of RAM, 32 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator and clock circuitry. In addition, the AT89C51 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The Power Down Mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

Pin Description

V_{CC}
Supply voltage.

GND
Ground.

Port 0
Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs.

Port 0 may also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode P0 has internal pullups.

Port 0 also receives the code bytes during Flash programming, and outputs the code bytes during program verification. External pullups are required during program verification.

Port 1
Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

Port 1 also receives the low-order address bytes during Flash programming and verification.

Port 2
Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @ DPTR). In this application it uses strong internal pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

Port 3
Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89C51 as listed below:

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

Port 3 also receives some control signals for Flash programming and verification.

RST
Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$
Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during Flash programming.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

$\overline{PSEN}$
Program Store Enable is the read strobe to external program memory.

When the AT89C51 is executing code from external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory.

$\overline{EA}$/V_{PP}
External Access Enable. $\overline{EA}$ must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, $\overline{EA}$ will be internally latched on reset.

$\overline{EA}$ should be strapped to V_{CC} for internal program executions.

This pin also receives the 12-volt programming enable voltage (V_{PP}) during Flash programming, for parts that require 12-volt V_{PP}.

XTAL1
Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2
Output from the inverting oscillator amplifier.

Oscillator Characteristics

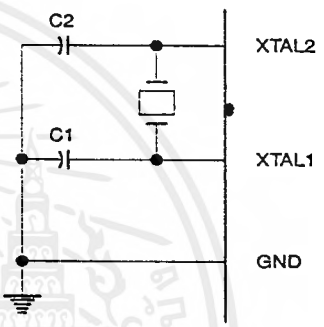
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Idle Mode

In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

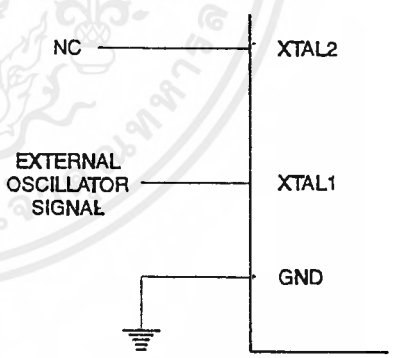
It should be noted that when idle is terminated by a hard ware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

Figure 1. Oscillator Connections



Note: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Figure 2. External Clock Drive Configuration



Status of External Pins During Idle and Power Down Modes

Mode	Program Memory	ALE	$\overline{PSEN}$	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power Down	Internal	0	0	Data	Data	Data	Data
Power Down	External	0	0	Float	Data	Data	Data

Power Down Mode

In the power down mode the oscillator is stopped, and the instruction that invokes power down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the power down mode is terminated. The only exit from power down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

Program Memory Lock Bits

On the chip are three lock bits which can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the table below:

When lock bit 1 is programmed, the logic level at the $\overline{EA}$ pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value, and holds that value until reset is activated. It is necessary that the latched value of $\overline{EA}$ be in agreement with the current logic level at that pin in order for the device to function properly.

Lock Bit Protection Modes

	Program Lock Bits			Protection Type
	LB1	LB2	LB3	
1	U	U	U	No program lock features.
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further programming of the Flash is disabled.
3	P	P	U	Same as mode 2, also verify is disabled.
4	P	P	P	Same as mode 3, also external execution is disabled.

Programming the Flash

The AT89C51 is normally shipped with the on-chip Flash memory array in the erased state (that is, contents = FFH) and ready to be programmed. The programming interface accepts either a high-voltage (12-volt) or a low-voltage (V_{CC}) program enable signal. The low voltage programming mode provides a convenient way to program the AT89C51 inside the user's system, while the high-voltage programming mode is compatible with conventional third party Flash or EPROM programmers.

The AT89C51 is shipped with either the high-voltage or low-voltage programming mode enabled. The respective top-side marking and device signature codes are listed in the following table.

	V_{PP} = 12V	V_{PP} = 5V
Top-Side Mark	AT89C51 xxxx yyww	AT89C51 xxxx-5 yyww
Signature	(030H)=1EH (031H)=51H (032H)=FFH	(030H)=1EH (031H)=51H (032H)=05H

The AT89C51 code memory array is programmed byte-by-byte in either programming mode. *To program any non-blank byte in the on-chip Flash Memory, the entire memory must be erased using the Chip Erase Mode.*

Programming Algorithm: Before programming the AT89C51, the address, data and control signals should be set up according to the Flash programming mode table and Figures 3 and 4. To program the AT89C51, take the following steps.

1. Input the desired memory location on the address lines.
2. Input the appropriate data byte on the data lines.
3. Activate the correct combination of control signals.
4. Raise $\overline{EA}/V_{PP}$ to 12V for the high-voltage programming mode.
5. Pulse ALE/$\overline{PROG}$ once to program a byte in the Flash array or the lock bits. The byte-write cycle is self-timed and typically takes no more than 1.5 ms. Repeat steps 1 through 5, changing the address and data for the entire array or until the end of the object file is reached.

$\overline{Data}$ Polling: The AT89C51 features $\overline{Data}$ Polling to indicate the end of a write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written datum on PO.7. Once the write cycle has been completed, true data are valid on all outputs, and the next cycle may begin. $\overline{Data}$ Polling may begin any time after a write cycle has been initiated.

Ready/$\overline{Busy}$: The progress of byte programming can also be monitored by the RDY/$\overline{BSY}$ output signal. P3.4 is pulled low after ALE goes high during programming to indicate BUSY. P3.4 is pulled high again when programming is done to indicate READY.

Program Verify: If lock bits LB1 and LB2 have not been programmed, the programmed code data can be read back via the address and data lines for verification. The lock bits cannot be verified directly. Verification of the lock bits is achieved by observing that their features are enabled.

Chip Erase: The entire Flash array is erased electrically by using the proper combination of control signals and by holding ALE/$\overline{PROG}$ low for 10 ms. The code array is written with all "1"s. The chip erase operation must be executed before the code memory can be re-programmed.

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 030H,

031H, and 032H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows.

(030H) = 1EH indicates manufactured by Atmel
(031H) = 51H indicates 89C51
(032H) = FFH indicates 12V programming
(032H) = 05H indicates 5V programming

Programming Interface

Every code byte in the Flash array can be written and the entire array can be erased by using the appropriate combination of control signals. The write operation cycle is self-timed and once initiated, will automatically time itself to completion.

All major programming vendors offer worldwide support for the Atmel microcontroller series. Please contact your local programming vendor for the appropriate software revision.

Flash Programming Modes

Mode		RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.6	P2.7	P3.6	P3.7
Write Code Data		H	L		H/12V	L	H	H	H
Read Code Data		H	L	H	H	L	L	H	H
Write Lock	Bit - 1	H	L		H/12V	H	H	H	H
	Bit - 2	H	L		H/12V	H	H	L	L
	Bit - 3	H	L		H/12V	H	L	H	L
Chip Erase		H	L	 (1)	H/12V	H	L	L	L
Read Signature Byte		H	L	H	H	L	L	L	L

Note: 1. Chip Erase requires a 10-ms PROG pulse.

Figure 3. Programming the Flash

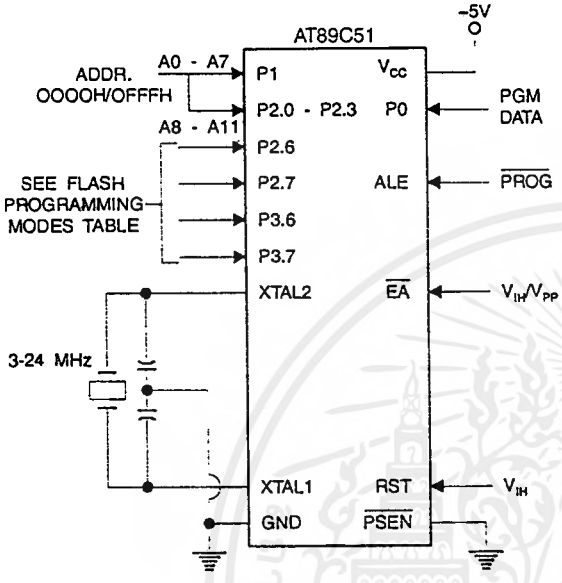
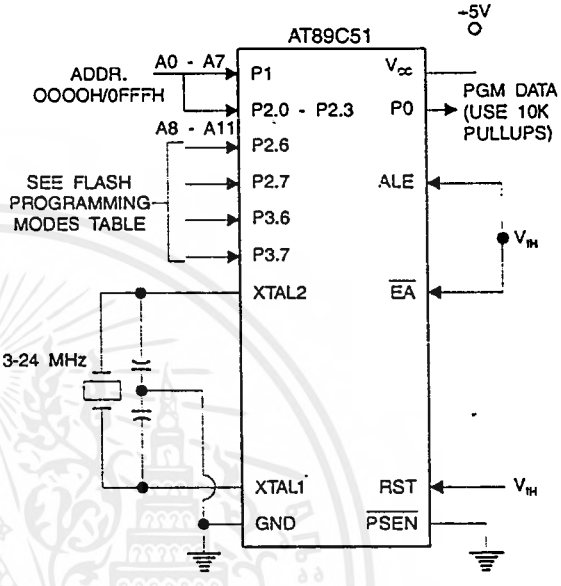


Figure 4. Verifying the Flash



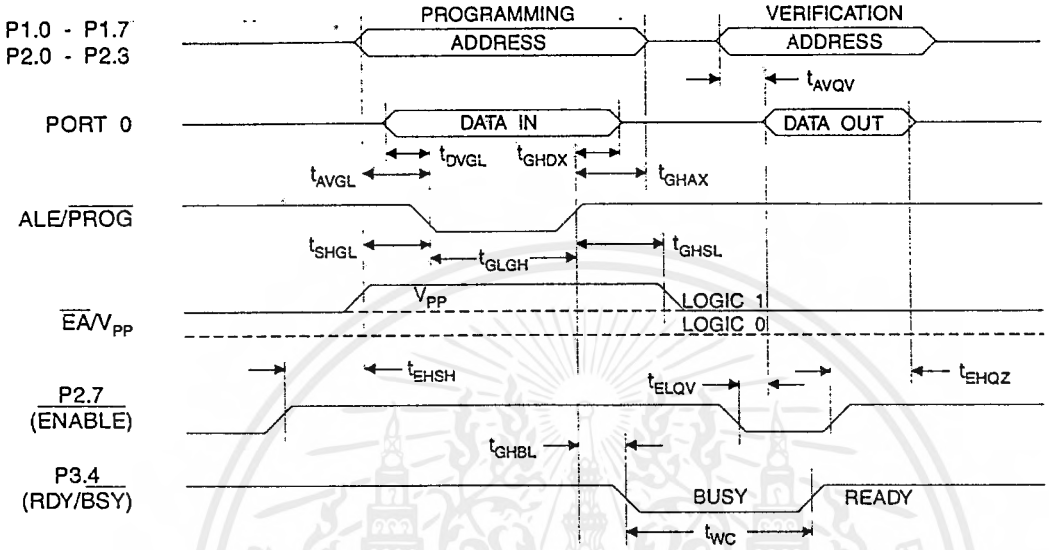
Flash Programming and Verification Characteristics

T_A = 0°C to 70°C, V_{CC} = 5.0 ± 10%

Symbol	Parameter	Min	Max	Units
V_{PP}(1)	Programming Enable Voltage	11.5	12.5	V
I_{PP}(1)	Programming Enable Current		1.0	mA
$1/t_{CLCL}$	Oscillator Frequency	3	24	MHz
t_{AVGL}	Address Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
t_{GHAX}	Address Hold After $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
t_{GHDX}	Data Hold After $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) High to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
t_{GHSL}(1)	V_{PP} Hold After $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ Width	1	110	µs
t_{AVQV}	Address to Data Valid		$48t_{CLCL}$	
t_{ELQV}	$\overline{ENABLE}$ Low to Data Valid		$48t_{CLCL}$	
t_{EHQZ}	Data Float After $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHBL}	$\overline{PROG}$ High to $\overline{BUSY}$ Low		1.0	µs
t_{WC}	Byte Write Cycle Time		2.0	ms

Note: 1. Only used in 12-volt programming mode.

Flash Programming and Verification Waveforms - High Voltage Mode (V_{PP} = 12V)



Flash Programming and Verification Waveforms - Low Voltage Mode (V_{PP} = 5V)