# การตรวจจับอุณหภูมิผ่านระบบทีดีเอ็ม

# TEMPERATURE DETECTION VIA TDM SYSTEM


นายมงคล วรรณโกษิตย์ 42015614

นายวีรศักดิ์ ตันชเสาวภาค 42015619



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

สาขาเทคโนโลยีโทรคมนาคม ภาควิชาเทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

# การตรวจจับอุณหภูมิผ่านระบบทีดีเอ็ม


| | | |
|---|---|---|
| โดย | นายมงคล | วรรณโกษิตย์ |
| | นายวีรศักดิ์ | ตันชเสาวภาค |
| **อาจารย์ที่ปรึกษา** | ผศ. อรลาภ | แสงอรุณ |
| **ปีการศึกษา** | 2543 | |


## บทคัดย่อ


ปริญญานิพนธ์ฉบับนี้ เสนอการออกแบบและสร้างระบบการส่งสัญญาณแบบ ทีดีเอ็ม โดยใช้การออกแบบไมโครคอนโทรลเลอร์ MCS-51 เป็นตัวควบคุมการมัลติเพล็กซ์ และใช้ไมโครคอนโทรลเลอร์ PIC 16F84 เป็นตัวจัดสัญญาณในการสื่อสารข้อมูลแบบอนุกรม โดยในปริญญานิพนธ์ฉบับนี้ออกแบบให้สามารถส่งข้อมูลได้ 2 ช่องสัญญาณใน 1 สายส่ง

**Temperture Detection Via TDM System**

**BY** MR. Mongkol Wunnakosit

MR. Weerasak Tunchasaowapak

**ADVISOR** Asst.prof. Ornlarp Saingaroon

**YEAR** 2000

---

## Abstract

This thesis presents construction and designed of the temperature detection system. The time division multiplexing and sending signal have controled by microcontroller (MCS-51).Other microcontroller (PIC16F84) sorts signal for communicate with serial ports. There have two channals with one transmittion line.

# กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ได้จัดทำขึ้นเป็นผลสำเร็จ ทางคณะผู้จัดทำต้องขอขอบพระคุณท่านบูรพาจารย์ทั้งหลายท่านผู้เขียนเอกสารและตำราอ้างอิงต่างๆ โดยเฉพาะอาจารย์ที่ปรึกษา ผศ.อรลาภ แสงอรุณ เป็นอย่างสูงที่ช่วยกรุณาถ่ายทอดความรู้ดูแลและเอาใจใส่กระผมเป็นอย่างดี จนทำให้โครงงานชิ้นนี้สำเร็จลุล่วงไปได้ด้วยดี และขอบพระคุณบิดา – มารดา ที่เฝ้าตรากตรำทำงานส่งเสียให้พวกเราเรียนจนสำเร็จ ขอบคุณสำหรับกำลังใจจากเพื่อนๆที่ช่วยเหลือและค่อยให้กำลังใจมาโดยตลอด ทั้งนี้คณะผู้จัดทำต้องขอขอบพระคุณภาควิชาเทคนิคอุตสาหกรรม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่ให้โอกาสคณะผู้จัดทำได้มีโอกาสเข้ามาศึกษา ณ สถาบันแห่งนี้

สุดท้ายนี้ทางคณะผู้จัดทำ ขอขอบพระคุณท่านอาจารย์ทุกท่านที่กรุณาประสิทธิ์ประสาทวิชาความรู้ รวมถึงแนวทางความคิดและแนวทางปฏิบัติให้แก่คณะผู้จัดทำ จนทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จผลตามเป้าหมายทุกประการ

นายมงคล วรรณโกษิตย์
นายวีรศักดิ์ ตันชเสาวภาค

# สารบัญ

# สารบัญ(ต่อ)

# สารบัญรูปภาพ

# สารบัญรูปภาพ(ต่อ)

# บทที่ 1

## บทนำ

สัญญาณไฟฟ้าที่ใช้ติดต่อสื่อสารกันในระบบมีอยู่ 2 ลักษณะคือ สัญญาณแอนะลอก(Analog) และสัญญาณดิจิตอล (Digital) ส่วนระบบการสื่อสารด้วยสัญญาณไฟฟ้า สามารถทำได้ โดยการสร้างลักษณะของรูปคลื่นข่าวสารที่ต้องการส่ง- รับ ในลักษณะต่างๆ การสื่อสารสัญญาณอาจจะเป็นแบบการใช้สาย เป็นช่องการสื่อสาร กระบวนการใช้เปลี่ยนลักษณะรูปคลื่นของสัญญาณไฟฟ้า ทางด้านผู้ส่ง เพื่อใช้สำหรับส่งออกไปในช่องทางการสื่อสาร เรียกว่ากระบวนการมอดูเลชั่น(Modulation) ส่วนทางด้านผู้รับข่าวสาร เป็นกระบวนการที่กลับกัน เรียกว่ากระบวนการดีมอดูเลชั่น (Demodulation) เพื่อสร้างสัญญาณข่าวสารให้มีลักษณะให้มีลักษณะของรูปคลื่น เหมือนกับลักษณะของรูปคลื่นข่าวสารก่อนที่จะผ่านกระบวนการมอดูเลชั่น ทางด้านส่ง การสื่อสารกันด้วยสัญญาณทางไฟฟ้า ตามกระบวนการที่กล่าวมา สามารถที่จะส่งสัญญาณข่าวสาร จากช่องสัญญาณการสื่อสารจำนวนมากออกไปพร้อมๆกัน ในช่องทางการสื่อสารเดียวได้ด้วยวิธีการมัลติเพล็กซ์

ในโครงการนี้ประยุกติใช้การส่งอุณหภูมิโดยใช้ IC DS1820 เป็นตัวตรวจจับ และใช้ไมโครคอนโทรลเลอร์เป็นตัวมัลติเพล็กซ์และรับส่งข้อมูล

## วัตถุประสงค์

1. เพื่อประยุกต์ใช้งานในการส่งสัญญาณระบบ TDM ใช้ในการตรวจวัดอุณหภูมิ
2. เพื่อศึกษาการเขียนโปรแกรมการใช้งานของไมโครคอนโทรลเลอร์ MCS 51
3. เพื่อศึกษาการเขียนโปรแกรมของ PIC 16F84
4. เพื่อศึกษาการรับการส่งสัญญาณแบบระบบ TDM
5. โครงงานนี้สามารถพัฒนาให้ไปใช้งานด้านอื่นได้อีกด้วย

## ขอบเขตของโครงงาน

ในการตรวจวัดอุณหภูมิ จะใช้ไอซี DS 1820 เป็นตัวตรวจจับแล้วส่งสัญญาณไบนารีไปยังไมโครคอนโทรลเลอร์ MCS 51ในการมัลติเพล็กซ์ จะใช้ไมโครคอนโทรลเลอร์ MCS 51 เพราะสะดวกและง่ายในการเขียนโปรแกรม ส่งเข้าไปยังส่วนของการแปลงสัญญาณข้อมูลแบบขนานเป็นอนุกรมจะใช้ PIC 16F84 เขียนโปรแกรมเพื่อส่งไปยังเครื่องรับ ในการใช้อุปกรณ์ไมโครคอนโทรลเลอร์ที่

สามารถเขียนโปรแกรมการทำงานได้จึง ทำให้ลดขนานของวงจรมาก มีประสิทธิภาพมาก และสามารถแก้ไขได้ง่าย ทำให้เกิดความสะดวดรวดเร็วในการประกอบวงจร

ขั้นตอนการดำเนินงาน

1. ศึกษาไมโครคอนโทรลเลอร์ MCS 51 และ PIC 16F84
2. ศึกษาการเขียนโปรแกรมภาษา Assembly
3. ศึกษาการตรวจวัดอุณหภูมิเพื่อส่งสัญญาณระบบ TDM
4. สร้างวงจรเครื่องรับและเครื่องส่ง
5. เมื่อสัญญาณส่งมาที่เครื่องรับแล้วนำไปแสดงผลที่ LCD

# บทที่ 2
# ทฤษฎีและหลักการ

**ทฤษฎีและหลักการแบ่งออกได้เป็น 2 ส่วนดังนี้**

2.1 ทฤษฎีและหลักการมัลติเพล็กซ์และการดีมัลติเพล็กซ์

2.2 ทฤษฎีและหลักการควบคุมและแสดงผล

## 2.1 การมัลติเพล็กซ์ และการดีมัลติเพล็กซ์

### 2.1.1 หลักการของระบบ Pulse Code Modulation (PCM)

ระบบ PCM ที่สมบูรณ์ สำหรับช่องการส่งสัญญาณ (Channel) จำนวน 1 ช่อง สัญญาณจะประกอบไปด้วยชุดเข้ารหัส (Encoder) สัญญาณแอนะลอกเป็นสัญญาณ PCM หรือชุดมอดูเลเตอร์ (Modulater) ซึ่งเป็นชุดปลายทางด้านส่ง และชุดถอดรหัส(Decoder) สัญญาณ PCM เป็นสัญญาณแอนะลอก หรือชุดดีมอดูเลเตอร์ (Demodulater) เพื่อคืนรูปสัญญาณกลับคืนซึ่งเป็นชุดปลายทางด้านรับ

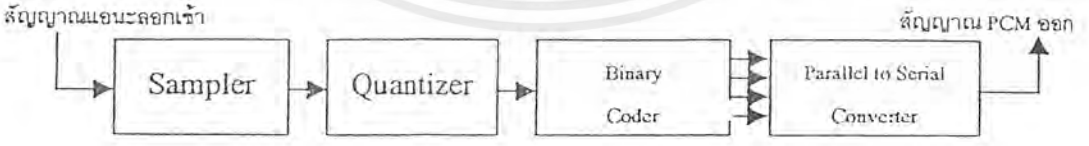
**2.1.2 มอดูเลเตอร์หรือตัวเข้ารหัส(Encoder)** เป็นวิธีการที่จะเปลี่ยน Analog Speech Signal ให้เป็นสัญญาณดิจิตอล ซึ่งแต่ละสัญญาณ จะถูกกำหนดให้เป็นขบวนของพัลส์ในรูปของรหัสไบนารี(Binary Code) การเปลี่ยนดังกล่าวจะต้องประกอบด้วยหลักการ 4 ประการคือ

การชักตัวอย่าง (Sampling)

การแบ่งย่าน Amplitude ออกเป็นระดับต่างๆ (Quantization)

การเข้ารหัส (Coding)

การแปลงข้อมูลขนานเป็นข้อมูลอนุกรม (Parallel to Serial Converter)




รูปที่ 1 บล็อกไดอะแกรมของมอดูเลเตอร์ PCM

รูปที่ 1 แสดงให้เห็นบล็อกไดอะแกรมของมอดูเลเตอร์ระบบ PCM วิธีการทำงานเป็นดังนี้ ขั้นแรก คือการชักตัวอย่างสัญญาณแอนะลอกด้วย Sampler จะทำให้ได้พัลส์ที่มีแอมปลิจูดเท่ากับแอมปลิจูดของสัญญาณแอนะลอก ณ เวลาที่ทำการสุ่มนั้นๆ เรียกว่า สัญญาณพัลส์ Pulse Amplitude Modulation (PAM) จากนั้นสัญญาณ PCM ก็จะได้รับการแปลงให้เป็นสัญญาณยอด

แบน (Flat-Top) และส่งไปยัง Quantizer ในขั้นนี้ Amplitude ของแต่ละตัวอย่าง ที่ชักมาได้จะถูกจัดให้อยู่ในระดับที่ตรงกันหรือใกล้เคียงกับระดับที่ได้กำหนดไว้ ต่อไปคือการนำแอมปลีจูดที่ได้การนำ Amplitude ที่ได้รับการจัดระดับแล้วไปทำการเข้ารหัสเป็นไบนารี (Binary Code) ซึ่งรหัสไบนารี ที่ให้นี้จะตรงกับระดับของ Amplitude ที่ได้ถูกแบ่งไว้ในตอนแรก รหัสไบนารีที่ได้จะได้รับการแปลงผันในรูปของรหัสไบนารีอนุกรม ซึ่งขบวนพัลส์เหล่านี้คือ สัญญาณ PCM ที่ถูกส่งออกไปในสายส่ง

**2.1.3 ดีมอดูเลเตอร์หรือตัวถอดรหัส(Decoder)** เป็นการเปลี่ยนสัญญาณดิจิตอลในรูปของขบวนพัลส์ PCM อนุกรมให้กลับอยู่ในรูปของสัญญาณ Analog Speech Signal กระบวนดังกล่าวประกอบด้วยหลักการ 3 ประการคือ

การแปลงสัญญาณอนุกรมเป็นสัญญาณขนาน(Serial to Parallel Convention)

การแปลงผันสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก(D/A)

การกรองผ่านความถี่ต่ำ(Low Pass Filter)


รูปที่ 2 แสดงบล็อกไดอะแกรมของระบบ PCM Demodulation

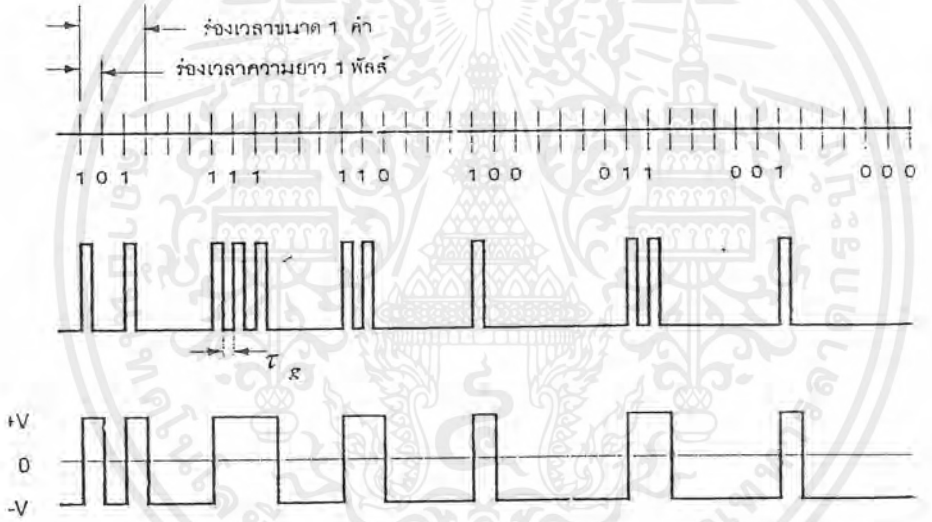
รูปที่ 2 แสดงบล็อกไดอะแกรมของระบบ PCM Demodulation ขั้นตอนการทำงานเป็นดังนี้ คือ วงจรแปลงผันสัญญาณอนุกรมเป็นสัญญาณขนานจะรับขบวนพัลส์ PCM อนุกรมเข้ามาและทำการแปลงสัญญาณให้เป็นสัญญาณขนาน สัญญาณ PCM ขนานที่ได้จะถูกส่งเข้าวงจรแปลงผันสัญญาณดิจิตอลเป็นสัญญาณแอนะลอกและคืนรูปสัญญาณแอนาลอกที่ได้ให้เป็นสัญญาณ Pulse Amplitude Modulation (PAM) ขั้นตอนสุดท้ายก็ จะนำสัญญาณPAM ที่ได้ ไปกรองเอาองค์ประกอบของความถี่สูงออกด้วยวงจรกรองผ่านความถี่ต่ำ (Low Pass Filter) จะได้สัญญาณคืนรูปเป็นสัญญาณ Analog Speech Signal ที่ปลายทางด้านส่ง ส่งมา

**2.1.4 การใช้สัญญาณไฟฟ้าแทนรหัสไบนารี**

รูปที่ 3 แสดงให้เห็นโดยการใช้พัลส์ทางไฟฟ้าแทนตัวเลขไบนารี โดยใช้ตัวเลขไบนารีสามหลักแทนสัญญาณตัวอย่างที่ผ่านการควอนไทซ์แล้วหนึ่งตัวอย่าง เรียกว่าหนึ่งเวิร์ด

(A Word) ในรูปที่ 3 ก แสดงร่องเวลาของพัลส์ (Pulse Time Slots) รูปที่ 3ข แสดงให้เห็นการแทนตัวเลขไบนารี "1" หนึ่งตัวด้วยพัลส์หนึ่งพัลส์ ในขณะที่ตัวเลขไบนารี "0" หนึ่งตัวด้วยการไม่มีพัลส์หนึ่งพัลส์ โดยมี $\tau_g$ เป็นส่วนของเวลาควบคุม (Guard Time) ในช่วงว่างของแต่ละคำนั้นสามารถนำมาใช้ประโยชน์ ในการส่งข่าวสารได้

ถ้าหากกำจัดเวลาคุมระหว่างพัลส์ออกไป จะได้รูปคลื่นตามที่แสดงในรูปที่ 3 ค ซึ่งมีระดับการเปลี่ยนของพัลส์อยู่ระหว่างแรงดันสองระดับ เมื่อพัลส์ที่อยู่ระดับแรงดันที่ต่ำกว่า ณ ร่องเวลาใดร่องเวลาหนึ่งพัลส์นั้นจะเป็นตัวแทนของตัวเลขไบนารี "0" หนึ่งตัว ในขณะที่ระดับแรงดันที่สูงกว่าจะเป็นตัวแทนของ "1" หนึ่งตัว การกำหนดระดับแรงดันอาจจะกำหนดให้อยู่ระหว่าง –V กับ + V ตามที่แสดงในรูปที่ 3 ค




รูปที่ 3 การใช้สัญญาณไฟฟ้าแทนตัวเลขไบนารี

สำหรับการสร้างพัลส์ PCM ตามรูปที่ 3 ข นั้นหากว่าพัลส์ที่นำมาสำหรับการชักตัวอย่างที่มีระดับของแรงดันระหว่าง 0 V ถึง + V ใดๆ จึงทำให้พัลส์ชักตัวอย่างที่มีระดับของแรงดันระหว่าง0 ถึง +V ใดๆ ไม่สามารถเป็นสัญญาณเปิดทางให้กับสัญญาณแอนะลอก ในครึ่งซีกทางขั้วลบได้ ดังนั้น ถ้าต้องการชักตัวอย่างสัญญาณด้วยพัลส์ชักตัวอย่างซึ่งมีลักษณะตามที่ได้กล่าวมาแล้วนั้น จะต้องทำการเลื่อนระดับของสัญญาณ ซึ่งมี 2 ขั้ว นั้นให้ไปอยู่ในช่วงของระดับแรงดันของพัลส์ชักตัวอย่าง ด้วยการเพิ่มแรงดันไฟตรงชดเชย (DC Offset) ให้กับสัญญาณ

แอนนะลอก ซึ่งนับว่าเป็นปัญหาประการหนึ่ง ส่วนการสร้างพัลส์ PCM ตามรูปที่ 3 ค นั้น ไม่จำเป็นต้องใช้รงดันไฟตรงชดเชยให้กับสัญญาณแอนะลอก ปัญหาดังกล่าวจึงไม่เกิดขึ้น เนื่องจากว่าพัลส์ชักตัวอย่าง จะเป็นพัลส์ที่ระดับของแรงดันอยู่ในช่วง –V ถึง +V

เนื่องจากว่าระบบการสื่อสารสัญญาณดิจิตอลโดยมากจะเป็นการส่งแบบอนุกรม ถ้าคำรหัสไบนารีอยู่ในรูปแบบขนาน จะต้องแปลงให้เป็นคำรหัสอนุกรม จะได้เอาพุตเป็นอนุกรมของพัลส์ ซึ่งกลุ่มของพัลส์ที่มีขนาดแน่นอนจำนวนหนึ่ง ก็คือตัวแทนของเลขรหัสหนึ่งจำนวน และกลุ่มของพัลส์ที่ได้รับการแปลงให้เป็นอนุกรมแล้วนี้ก็คือ สัญญาณ PCM นั้นเอง

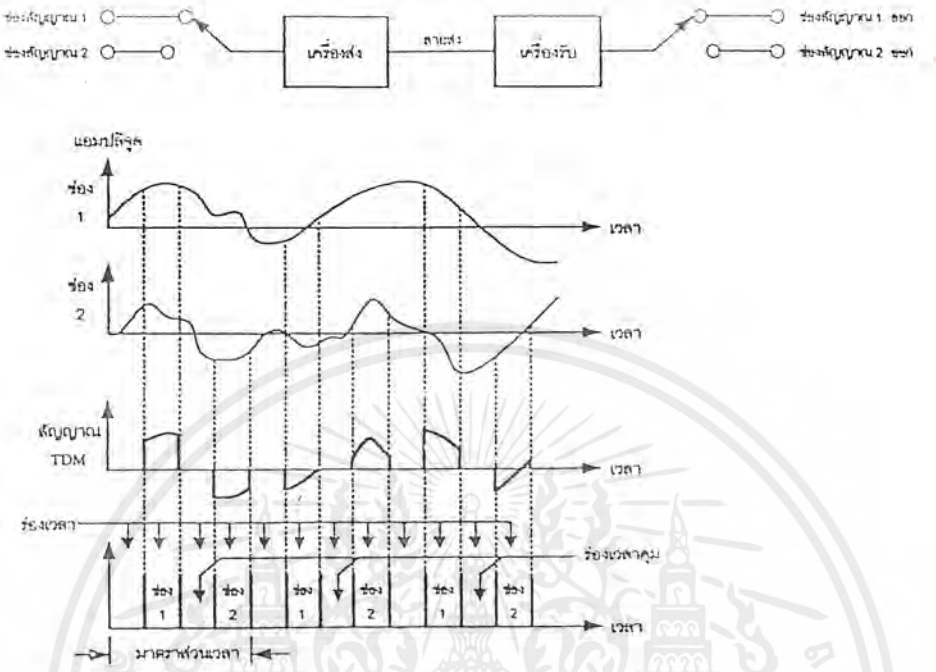
เมื่อนำบล็อกไดอะแกรมในรูปที่ 1 มาพิจารณาอีกครั้ง จะเห็นว่าอันที่จริงแล้ว ตัวเข้ารหัส PCM หรือมอดูเลเตอร์ PCM ใน 3 บล็อกแรกซึ่งประกอบด้วย กระบวนการชักตัวอย่าง การแบ่งย่าน Amplitude ออกเป็นระดับต่างๆ หรือการควอนไทซ์ และการเข้ารหัสไบนารี ก็คือกระบวนการที่สมบูรณ์ของการแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล (A/D Converter) นั้นเอง ดังนั้นกระบวนการถอดรหัส PCM หรือการดีมอดูเลเตอร์ PCM ตามบล็อกไดอะแกรมตามรูปที่ 2 ก็เป็นกระบวนการแปลงกลับคือ การแปลงสัญญาณดิจิตอลให้กลับสัญญาณแอนะลอก (D/A Converter)

ส่วนทฤษฎีที่นำสนอนั้นเป็นกระบวนการสำหรับการส่งช่องสัญญาณเพียงหนึ่งช่องสัญญาณเท่านั้น ถ้าหากต้องการส่งสัญญาณหลายๆสัญญาณออกไปบนช่องสื่อสารอันเดียวกัน จะต้องทำการศึกษาการมัลติเพล็กซ์และการซิงโครไนซ์

**2.1.5 การมัลติเพล็กซ์ (Multiplexing)**

การมัลติเพล็กซ์ก็คือ กระบวนการนำมาใช้เพื่อให้สามารถส่งสัญญาณหลายๆสัญญาณบนช่องการส่งสัญญาณเพียงหนึ่งช่อง และสามารถทำให้ปลายทางด้านรับ รับสัญญาณแต่ละสัญญาณที่ส่งมาได้ กระบวนการมัลติเพล็กซ์ทำได้ 2 วิธีคือ ส่งสัญญาณไปบนแถบความถี่ที่แตกต่างกันเรียกว่าการมัลติเพล็กซ์แบบแบ่งความถี่ (Frequency Division Multiplexing) การรับส่งสัญญาณที่เวลาต่างกันเรียกว่า การมัลติเพล็กซ์แบบแบ่งเวลา (Time Division Multiplexing) ในที่นี้จะกล่าวเฉพาะ การมัลติเพล็กซ์แบบแบ่งเวลาเท่านั้น

การมัลติเพล็กซ์แบบแบ่งเวลา (TDM) คือการส่งสัญญาณข่าวสาร จากแหล่งกำเนิดสัญญาณจำนวนมากออกไปบนช่องทางการสื่อสารอันเดียวกัน ที่เวลาที่แตกต่างกัน โดยการที่ส่งสัญญาณจากแหล่งกำเนิดหนึ่งๆ จะใช้เวลาชั่วขณะหนึ่งที่มีช่วงของเวลาที่แน่นอน




รูปที่ 4 แผนภาพผังงานของระบบ TDM และรูปคลื่น

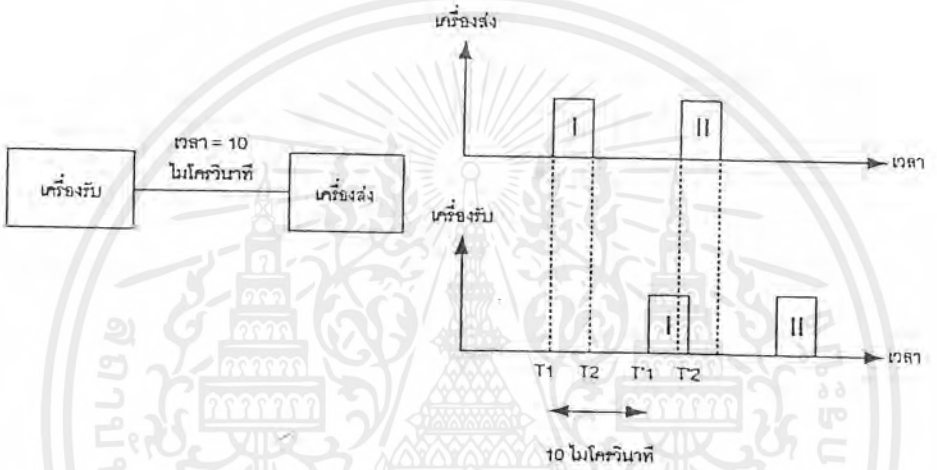
รูปที่ 4 แสดงแผนภาพผังงานของระบบ TDM ซึ่งการเชื่อมต่อของสวิตช์ที่เครื่องรับ กับสวิตช์ที่เครื่องส่งนั้น จะต่อเพื่อให้มีการส่งและรับสัญญาณจากแหล่งกำเนิด สัญญาณแหล่งอื่นๆ ด้วยคาบเวลาที่แน่นอนคาบหนึ่ง จากนั้นสวิตช์จะเปลี่ยนการเชื่อมต่อการส่งและรับสัญญาณจากแหล่งกำเนิด สัญญาณจากแหล่งที่สองต่อไป ด้วยคาบเวลาที่เท่ากับแต่เวลาที่แตกต่างกัน ดูในรูปที่ 4 จากรูปที่ 4 สังเกตุได้ว่ามาตราส่วนเวลา (Time Scale) ถูกแบ่งออกเป็นร่องเวลา (Time Slot) จำนวนหนึ่งซึ่งใช้สำหรับแยกช่องสัญญาณ ที่มาจากแหล่งกำเนิดสัญญาณที่ต่างกัน และระหว่างร่องเวลาแต่ละร่องจะมีร่องเวลาคุม (Guard Time) แทรกอยู่เพื่อป้องกันสัญญาณไขว้แทรก (Crosstalk) ระหว่างช่องสัญญาณ ตามรูปที่ 4 นี้แสดงลักษณะของสัญญาณที่ส่งเป็นสัญญาณ PAM (Pulse Amplitude Modulation)

ปัญหานึ่งที่เกิดขึ้นในระบบ TDM คือการซิงโครไนซ์ ของวงจรเวลาที่เครื่องส่งกับเวลาที่เครื่องรับซึ่งขณะความถี่ขณะทำงาน ต้องเท่ากันในขณะทั้งในเครื่องส่งและเครื่องรับ มิฉะนั้นทำให้การสวิทช์เปลี่ยนตำแหน่งที่เวลาแตกต่างกัน ผลที่อาจเกิดตามมาคือ SW1. อยู่ในตำแหน่ง

CH2 ในขณะที่ SW2. จะอยู่ที่ตำแหน่งที่ CH1. ดังนั้น จึงจำเป็นต้องทำให้แน่ใจ ได้ว่ามีการซิงโครไนซ์ของสวิทช์ทั้งสวิทช์ทั้งความถี่และตำแหน่ง

**2.1.6 การซิงโครไนซ์(Synchronization)**

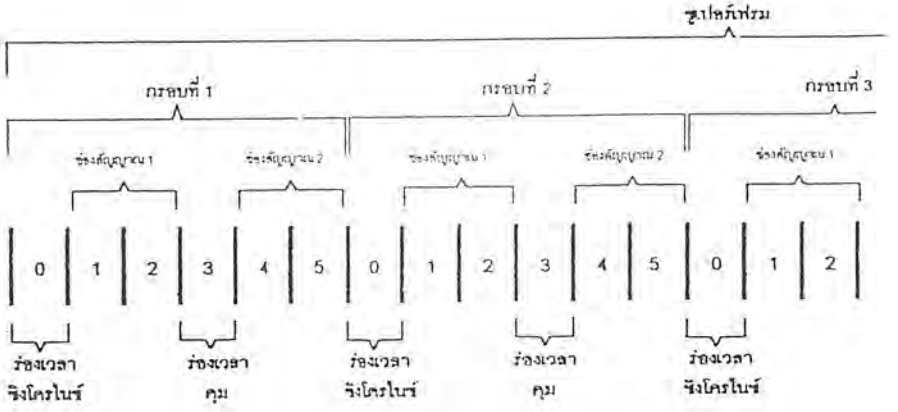
การซิงโครไนซ์คือ การทำให้เครื่องส่งและเครื่องรับปฏิบัติงานที่มาตราส่วนเวลา (Time Scale) ที่เท่ากันแต่ไม่จำเป็นต้องทำหน้าที่เดียวกันในเวลาเดียวกัน




รูปที่ 5 มาตราส่วนเวลา(Time Scale) ของเครื่องรับเครื่องส่ง

ถ้าระยะเวลาที่สัญญาณเดินทางจากเครื่องส่งไปยังเครื่องรับใช้เวลา 10 ไมโครวินาที มาตราส่วนเวลา (Time Scale) ของเครื่องรับจะตามหลังมาตราส่วนเวลาของเครื่องส่งอยู่ 10 ไมโครวินาที ดูในรูปที่ 5 ดังนั้นช่วงเวลา $T'_1$ , $T'_2$ ในมาตราส่วนเวลา (Time Scale) ของเครื่องรับจะต้องเท่ากับช่วงเวลา $T_1$, $T_2$ ในมาตราส่วนเวลาของเครื่องส่ง

การซิงโครไนซ์ทำโดยการทำได้โดยการส่งพัลส์พิเศษหรือลับของพัลส์ออกไปจากเครื่องส่ง ซึ่งจะต้องมีความแตกต่างจากพัลส์ของข่าวสาร โดยที่ในระบบดิจิตอลทั่วไป นั้นมีวิธีการส่งลำดับของพัลส์ซ้ำๆ กันออกไปเพื่อให้เกิดการซิงโครไนซ์ ตามวิธีดังกล่าวนี้จะต้องมีการจัดสรรร่องเวลา (TimeScale) ให้เป็นกลุ่มที่มีรูปแบบที่แน่นอน ด้วยวิธีการบางประการ โดยมีร่องเวลาที่ว่างเผื่อไว้ สำหรับส่งพัลส์ซิงโครไนซ์




รูปที่ 6 การจัดกรอบหนึ่งกรอบในระบบ TDM ซึ่งจัดสรรร่องเวลาจำนวน2 ร่อง ต่อ1ช่องสัญญาณ และร่องเวลาคุม 1 ร่อง

การจัดสรรร่องเวลาในระบบ TDM 2 ช่องสัญญาณดังแสดงในรูปที่ 6 หน่วยพื้นฐานหนึ่งหน่วยเรียกว่า กรอบ (Frame) ซึ่งประกอบด้วยร่องเวลาจำนวนหนึ่ง และแบ่งร่องเวลาหน่งร่องให้กับพัลส์ซิงโครไนซ์ โดยช่องสัญญาณช่องหนึ่งจะได้รับการจัดสรรร่องเวลาที่อยู่ติดกันจำนวนหนึ่งร่อง หรือมากว่าและมีร่องเวลาคุมแทรกอยู่ระห่างช่องสัญญาณแต่ละช่อง ถ้าการส่งพัลส์ซิงโครไนซ์เป็นอนุกรมของพัลส์ จะต้องทำการส่งพัลส์ซิงโครไนซ์จำนวนหนึ่งพัลส์ในหนึ่งกรอบ (Frame) แต่ทั้งนี้การส่งอนุกรมของพัลส์ซิงโครไนซ์ที่สมบูรณ์ จะต้องใช้กรอบจำนวนที่แน่นอนจำนวนหนึ่งรวมกันเป็นหนึ่งกลุ่มเรียกว่า ซูเปอร์เฟรม (Super Frame)

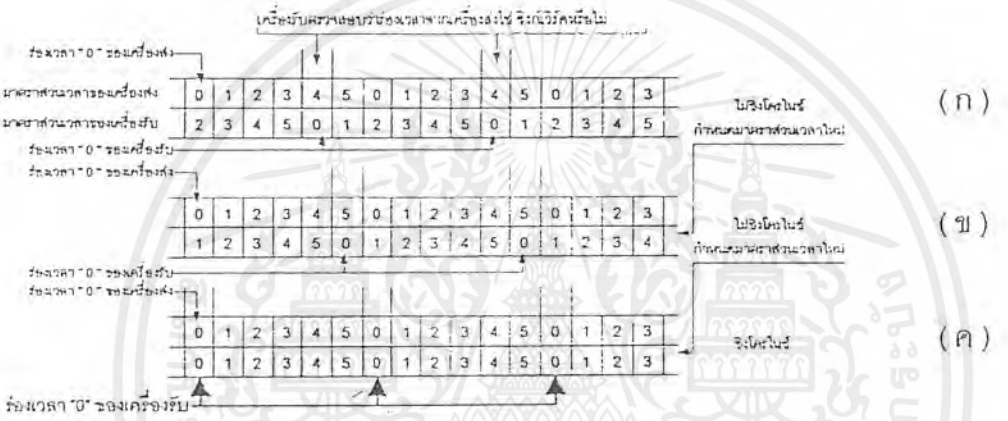
ดังนั้นในหนึ่งซูเปอร์เฟรมจะประกอบไปด้วย กลุ่มของกรอบ ที่มีจำนวนที่แน่นอนจำนวนหนึ่ง ในขณะที่กรอบ หนึ่งกรอบ จะประกอบด้วยกลุ่มของร่องเวลาที่จำนวนที่แน่นอนจำนวนหนึ่ง และกลุ่มของร่องเวลาแต่ละกลุ่ม จะทำหน้าที่เฉพาะตามที่ได้กำหนดไว้ เช่น ใช้ส่งพัลส์ของข่าวสาร ใช้ส่งซิงพัลส์ ส่งพัลส์ข้อมูลหรือเป็นร่องเวลาคุม

อนุกรมของพัลส์ซิงโครไนซ์ แบบดิจิตอล 1 ชุด เรียกว่าซิงโครไนซ์เวิร์ด (Synchronize Word) 1 คำ โดยการส่งพัลส์ซิงโครไนซ์จำนวน 1 บิต ออกไปในกรอบ แต่ละกรอบ

ในรูปที่ 7 แสดงการซิงโครไนเซชั่นชนิดหนึ่ง โดยเครื่องรับจะกำหนดมาตราส่วนเวลาให้กับตัวเองและจะทำการตรวจสอบว่า มาตราส่วนเวลา อันไหนที่ซิงโครไนซ์ กับมาตราส่วนเวลาของเครื่องส่งที่แสดงให้เห็นในรูปที่ 7 นั้นกรอบหนึ่งกรอบประกอบด้วย ร่องเวลาจำนวน 6 ร่อง แต่ละร่องเวลาแสดงด้วย 0, 1,2,3,4,5 โดยมีบิตของพัลส์ซิงโครไนซ์อยู่ในร่องเวลา 0 ของเครื่องส่ง

เครื่องรับจะตรวจสอบว่าบิตที่รับเข้ามาในร่องเวลา 0 ของเครื่องรับ ใช่ซิงโครไนซ์เวิร์ดหรือไม่ ถ้าใช่ก็แสดงว่าร่องเวลาของเครื่องรับซิงโครไนซ์กับร่องเวลาของ

อนุกรมของพัลส์ส่งเข้ามา ถ้าไม่ใช่ซิงโครไนซ์ (ดูรูปที่7 ข) เครื่องรับจะกำหนดร่องเวลาใหม่ ให้สัมพันธ์กับร่องเวลาของเครื่องส่ง ด้วยวิธีการหยุดร่องเวลาของเครื่องรับเองเป็นจำนวนหนึ่งร่องเวลา ผลลัพธ์แสดงในรูปที่ 7 ข จะเห็นว่าร่องเวลาของเครื่องรับ ยังคงนำหน้าร่องเวลาของเครื่องส่งอยู่เป็นจำนวน 1 ร่องเวลา วิธีหยุดร่องเวลาของเครื่องรับก็จะดำเนินต่อไปจนกระทั่ง เกิดการซิงโครไนซ์กับร่องเวลาของเครื่องส่ง ผลสุดท้ายก็จะทำให้เครื่องรับสามารถรับบิตซิงโครไนซ์เข้ามาที่ร่องเวลา 0 ของเครื่องรับได้ แสดงในรูปที่ 7 ค




รูปที่7 กระบวนการซิงโครไนเซชั่น

2.2 ทฤษฎีและหลักการควบคุมและแสดงผล

**2.2.1 ไมโครคอนโทรลเลอร์ เบอร์ 8051**

ไมโครคอนโทรลเลอร์ในตระกูล MCS-51 ที่มีขนาด 8 บิตประกอบด้วยไมโครคอนโทรลเลอร์เบอร์ต่างๆ ทุกๆ เบอร์จะมีสถาปัตยกรรมพื้นฐานที่เหมือนกัน เพียงแต่มีขนาดหน่วยความจำ-ภายใน และ ภายนอกที่แตกต่างกัน เพื่อความเหมาะสมในการนำเอาไปใช้งานตามความต้องการต่างๆ แต่เดิม 8051 ถูกสร้างด้วยวิธี HMOS I แต่ในปัจจุบันได้สร้างด้วยวิธี HMOS II จึงมีชื่อเป็น 8051AH ไมโครคอนโทรลเลอร์ในตระกูล 51 นั้น ถึงแม้ว่าจะมีหลายเบอร์แต่เราก็จะเรียกว่าเป็น "8051" ซึ่งหมายถึงไมโครคอนโทรลเลอร์ตระกูล51 นั้น ส่วนเบอร์ 8032 และ 8052 มีหน่วยความจำภายในเพิ่มขึ้นและมีวงจรนับ/จับเวลา ขนาด 16 บิต เพิ่มขึ้นมาดังตารางที่ 1

ตารางที่ 1 ไมโครคอนโทรลเลอร์ตระกูล 8051

| เบอร์ | หน่วยความจำภายใน | | จำนวนไทเมอร์/เคาน์เตอร์ | อินเตอร์รัพต์หมายเลข |
|---|---|---|---|---|
| | เก็บโปรแกรม | เก็บข้อมูล | | |
| 8052H | 8K x 8 ROM | 256 x 8 ROM | 3 x 16-Bit | 6 |
| 8051H | 4K x 8 ROM | 128 x 8 ROM | 2 x 16 Bit | 5 |
| 8051 | 4K x 8 ROM | 256 x 8 ROM | 2 x 16 Bit | 5 |
| 8032AH | ไม่มี | 128 x 8 ROM | 3 x 16-Bit | 6 |
| 8031AH | ไม่มี | 128 x 8 ROM | 2 x 16 Bit | 5 |
| 8031 | ไม่มี | 128 x 8 ROM | 2 x 16 Bit | 5 |
| 8751H | 4K x 8 EPROM | 128 x 8 ROM | 2 x 16 Bit | 5 |
| 80751H-12 | 4K x 8 EPROM | 128 x 8 ROM | 2 x 16 Bit | 5 |

**2.2.1.1 คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51**

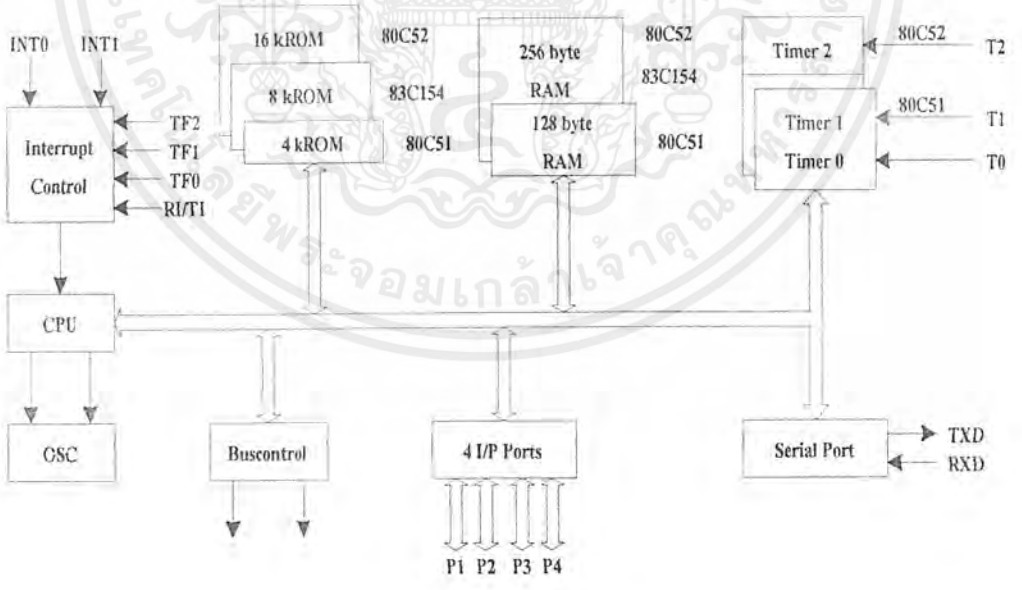
1. ต้องการแหล่งจ่ายไฟ + 5 V. ชุดเดียว
2. มีหน่วยความจำโปรแกรม (Program Memory) ขนาด 4 กิโลไบต์สำหรับเบอร์ 8051 และ

8031 สำหรับเบอร์ 8052 มีหน่วยความจำถึง 8 กิโลไบต์

3. มีหน่วยความจำสำหรับเก็บข้อมูล (Data Memory) ขนาด 128 ไบต์ สำหรับเบอร์ 8052 ขึ้นไปมีถึง 256 ไบต์
4. มีหน่วยความจำสำหรับเก็บโปรแกรมและข้อมูลแยกจากกันอย่างละ 64 กิโลไบต์
5. มีไทเมอร์เคาน์เตอร์ ขนาด 16 บิต 2 ชุด (สำหรับเบอร์ 8052 มี 3 ชุด) ทำงานได้ 4 โหมด
6. รับอินเตอร์รัพท์ได้ 6 แหล่ง 5 เวคเตอร์ สำหรับเบอร์ 8052 ขึ้นไปมี 8 แหล่ง 6 เวคเตอร์
7. มีพอร์ตรับส่งข้อมูลอนุกรม (UART) 2 พอร์ตแบบ Full Duplex เลือกรูปได้ 4 โหมด
8. มีคำสั่งในการทำ AND, OR หรือ Complement ได้ทั้งแบบ 8 บิตและ 1 บิต
9. มีวงจรออสซิลเลเตอร์ภายใน

**2.2.1.2 โครงสร้างของไมโครคอนโทรเลอร์ตระกูล MCS-51**

ภายใน 8051 จะประกอบด้วยเกจชนิดต่างๆ เช่น AND, OR, NOT ซึ่งแต่ละเกจเหล่านี้จะนำมาออกแบบให้มีหน้าที่การทำงานต่างๆ เช่น วงจรถอดรหัสคำสั่ง, วงจรสัญญาณนาฬิกา เป็นต้น โครงสร้างภายในของ 8051 จะประกอบด้วยส่วนย่อยๆ ดังรูปที่ 8




รูปที่ 8 โครงสร้างของ 8051

โครงสร้างของ 8051 จะประกอบด้วย 3 ส่วน หลักๆ ดังนี้

**1. ซีพียู (Central Processing Unit)** ส่วนนี้จะทำหน้าที่สร้างสัญญาณควบคุมการติดต่อกับส่วนอื่นๆ เรียกว่าวงจรควบคุม (Control Unit) สัญญาณที่สร้างจากวงจรควบคุม ได้แก่สัญญาณสำหรับการติดต่อกับหน่วยความจำ, อุปกรณ์รับข้อมูลเข้าหรือส่งข้อมูลออก ซึ่งส่วนควบคุมการขัดจังหวะ และส่วนควบคุมบัสก็เป็นส่วนหนึ่งของวงจรควบคุมด้วย การสร้างสัญญาณวงจรควบคุมจากซีพียูนี้ จะทำการสร้างสัญญาณ โดยการถอดรหัสจากคำสั่งที่มีการกำหนดไว้ และสัญญาณที่สร้างขึ้นมาจะอ้างอิงกับสัญญาณนาฬิกา ที่สร้างขึ้นจากวงจรออสซิลเลเตอร์เพื่อให้ทุกๆ ส่วนทำงานประสานกันอย่างถูกต้อง ในซีพียูยังประกอบด้วยส่วนประมวลผล (Arithmetic Logic Unit) ทำหน้าที่ประมวลผลข้อมูล เช่น การบวก, ลบ, คูณ หรือหารข้อมูลแล้วนำผลลัพธ์ไปเก็บไว้ในหน่วยความจำที่ต้องการ

**2. หน่วยความจำ (Memory)** มีไว้สำหรับจดจำข้อมูล ซึ่งในการนำข้อมูลเข้าและออกจากหน่วยความจำ เราจำเป็นต้องรู้ตำแหน่งของหน่วยความจำ (Address) ในการนำข้อมูลเข้าไปเก็บในหน่วยความจำเรียกว่า การเขียนข้อมูล และการนำข้อมูลออกจากหน่วยความจำ เรียกว่าการอ่านข้อ-มูล ในไมโครคอนโทรลเลอร์ 8051 ข้อมูล ในแต่ละตำแหน่งจะมีขนาด 8 บิต ดังนั้นแต่ละตำแหน่งของหน่วยความจำสามารถเก็บความจำข้อมูลมีค่าได้ระหว่าง $00000000_2$ ถึง $11111111_2$ หรือ 00H ถึง 0FFH ในการติดต่อหน่วยความจำจะต้องมีสัญญาณสามกลุ่มคือ

2.1 ตำแหน่งที่ต้องการติดต่อกับหน่วยความจำซึ่งในไมโครคอนโทรลเลอร์8051 จะมีหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลที่มีขนาดสูงสุดชนิดละ 65536 ตำแหน่ง (64 กิโลไบต์ ) ดังนั้นการอ้างตำแหน่งของหน่วยความจำจะต้องใช้เส้นแสดงตำแหน่งของหน่วยความจำจะต้องใช้เส้นแสดงตำแหน่งในเลขฐาน 2 ทั้งหมด 16 เส้น ( $2^{16}$ เท่ากับ 65536 )

2.2 ข้อมูลที่อ่านหรือเขียนกับหน่วยความจำในตำแหน่งที่เราต้องการ

2.3 สัญญาณควบคุมที่จะส่งไปยังหน่วยความจำ เพื่อจะบอกกับหน่วยความจำ ว่าต้องการอ่านหรือเขียนข้อมูล โดยวงจรถอดรหัสคำสั่ง จะทำการสร้างสัญญาณควบคุมจากคำสั่งที่อ่านเข้ามาจากหน่วยความจำโปรแกรม

**3. อุปกรณ์อินพุต/เอาพุต (Input / Output Device)** เป็นส่วนใช้ส่งข้อมูลเข้า หรือนำข้อมูลออกจากไมโครคอนโทรลเลอร์ 8051 ทำให้สามารถติดต่อกับอุปกรณ์ภายในอุปกรณ์

อินพุต/เอาท์พุต ได้แก่ อินพุต/เอาท์พุตพอร์ตแบบขนาน วงจรนับ/จับเวลา 0 วงจรนับ/จับเวลา 1 พอร์ตสื่อสารอนุกรม

3.1 พอร์ตแบบขนาน เป็นที่สำหรับใช้รับส่งข้อมูลซึ่งเป็นสัญญาณดิจิตอลเข้าหรือออกจากตัวไมโครคอนโทรลเลอร์เบอร์ 8051 มีทั้งหมด 4 พอร์ต โดยแต่ละพอร์ตจะรับส่งข้อมูลได้ 8 บิต มีพอร์ต P0, P1, P2 และ P3 บางพอร์ตใช้งานได้มากกว่า 1 อย่าง

3.2 วงจรนับเวลา/จับเวลา 0 และวงจรนับ/จับเวลา 1 เป็น วงจรที่สามารถทำการนับจำนวนไซเคิลของสัญญาณที่ต่อ จากภายนอกของไมโครคอนโทรลเลอร์เบอร์ 8051 หรือจำนวนของสัญญาณนาฬิกาภายในไมโครคอนโทรลเลอร์เบอร์ 8051 ก็ได้ สามารถตั้งค่าเริ่มต้นของการนับและอ่านค่าการนับได้โดยซีพียู

3.3 พอร์ตอนุกรม ซีพียูจะอ่านและเขียนข้อมูล พอร์ตอนุกรมเป็นแบบ 8 บิต แต่ละข้อมูลจะถูกส่งออกมาจากไมโครคอนโทรลเลอร์เบอร์ 8051 เรียงไปทีละบิตออกจากขา TXD และ ในการรับข้อมูลก็จะรับเข้ามาทีละบิตทางขา RXD และ จัดเรียงใหม่เป็น 8 บิต เพื่อให้ซีพียูอ่านไปใช้งานต่อไปในไมโครคอนโทรลเลอร์เบอร์ 8051 มีพอร์ตใช้งานได้หลายแบบทำให้สะดวกแก่การนำไปใช้งานต่างๆ ได้มากมาย การนำพอร์ตไปใช้งานจะต้องเขียนโปรแกรมขึ้นมาควบคุม

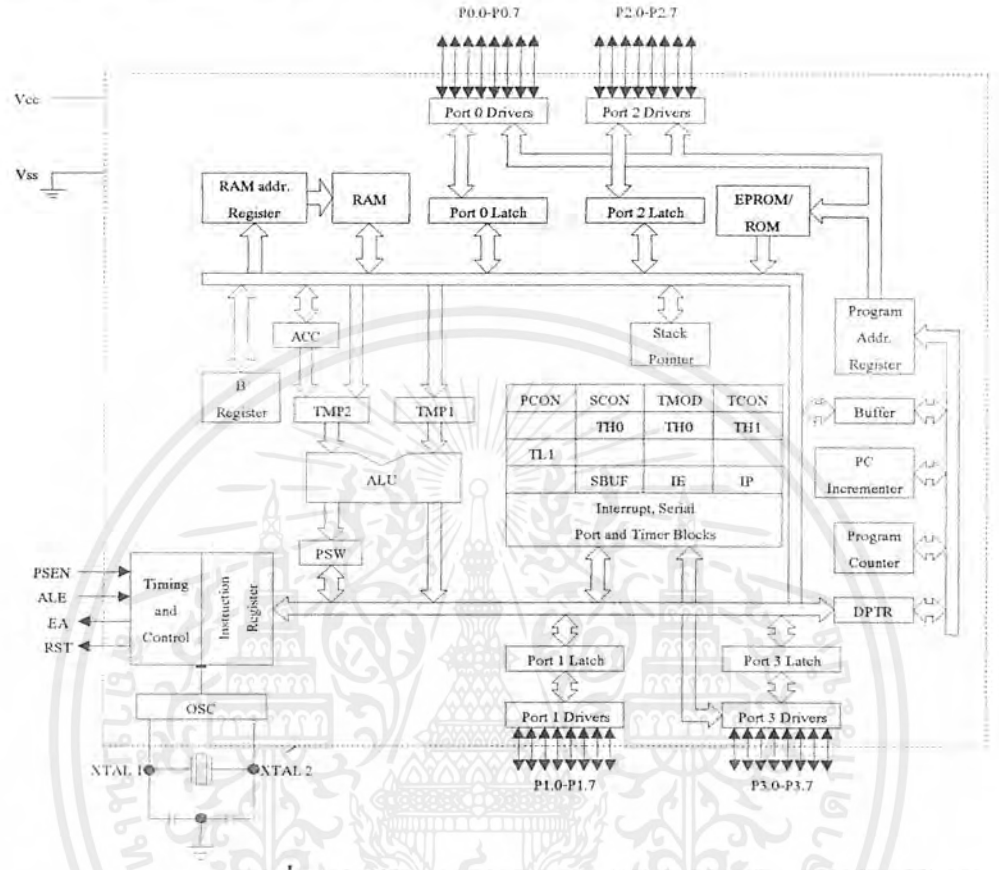
**2.2.1.3 สถาปัตยกรรมภายในของ 8051**

MCS-51 ใช้เทคโนโลยีในการผลิตแบบ NMOS และ CMOS เบอร์ 8032 และ 8052 จะมี ROM BASIC อยู่ภายในจึงสะดวกสำหรับโปรแกรมเมอร์ที่จะเขียนโปรแกรมด้วยภาษาเบสิก โครง-สร้างภายในสำหรับเบอร์ 8051 ดังแสดงดังรูปที่ 9 ซึ่งจะอธิบายถึงส่วนประกอบย่อยๆ ภายในตัวของไมโครคอนโทรลเลอร์เบอร์ 8051 และ สัญญาณจากภายในจะต่อออกสู่ภายนอกทางขาของไมโครคอนโทรลเลอร์ 8051

ไมโครคอนโทรลเลอร์เบอร์ 8051บรรจุอยู่ภายในวงจรรวมแบบDIP (Dual Inline Package) แบบ 40 ขาดังนี้

| | |
|---|---|
| Vcc (ขา 40) | ต่อกับไฟ +5V. |
| Vss (ขา 20) | เป็นขา GND |
| Port 0 (ขา 32-39) | มี 8 บิต คือ P0.0 – P0.7 มีโครงสร้างแบบ Open-Drain Bi-Directional โดยสามารถใช้งานได้ 2 หน้าที่คือ Address Bus และ Data Bus นอกจากนี้ยังใช้งานเป็นอินพุตเอาท์พุตได้ |
| Port 1 (ขา 1-8) | มี 8 บิต คือ P1.0 – P1.8 ใช้งานเป็นอินพุตและเอาท์พุตพอร์ตทั่วไป |

Port 2 (ขา 21-28) มี 8 บิต คือ P2.0 – P2.7 ใช้งานเป็นอินพุต/เอาท์พุตพอร์ตทั่วไปแล้ว ยังใช้เป็นตัวส่ง Address ไบต์สูง เพื่อติดต่อกับหน่วยความจำภายนอก




รูปที่ 9 สถาปัตยกรรมภายในของไมโครคอนโทรลเลอร์ เบอร์ 8051

Port 3 (ขา 10-17) มี 8 บิต คือ P3.0 – P3.7 ใช้งานเป็นอินพุต/เอาท์พอร์ตทั่วไป ใช้งานในหน้าที่พิเศษดังนี้

3.0/RXD (Serial Input Port) ใช้รับข้อมูลแบบอนุกรม3.1/TXD (Serial Output Port) ใช้ส่งข้อมูลแบบอนุกรม

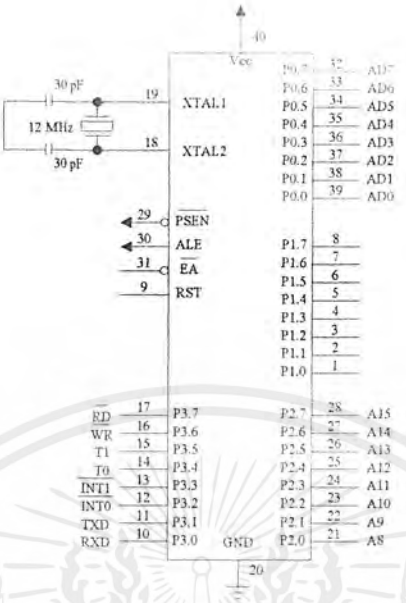
3.2/INT0 (External Interrupt 0) ใช้รับสัญญาณขัดจังหวะจากภายนอกเบอร์ 0

3.3/INT1 (External Interrupt 1) ใช้รับสัญญาณขัดจังหวะจากภายนอก เบอร์1

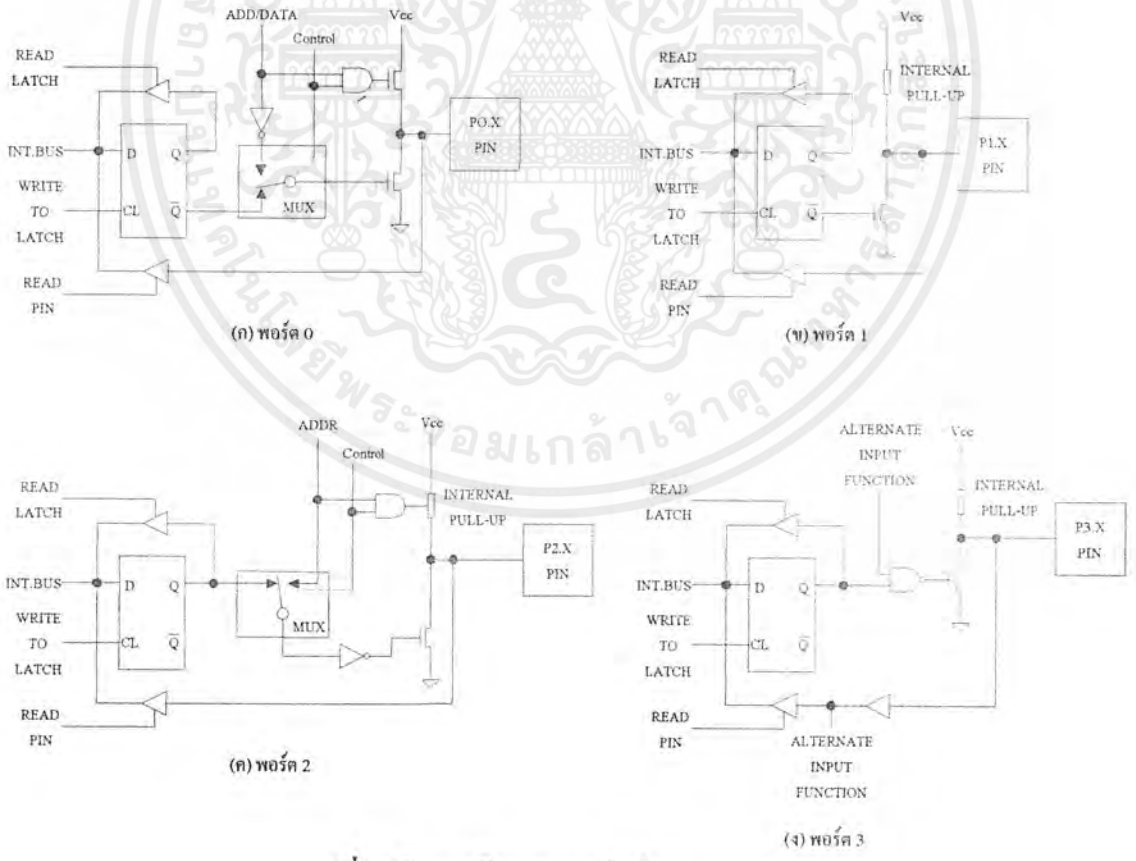
3.4/T0 (Counter 0 External Input) ใช้เป็นอินพุตให้วงจรนับ / จับเวลา ชุดที่ 0

3.5/T1 (Counter 1 External Input) ใช้เป็นอินพุตให้วงจรนับ / จับ

| | |
|---|---|
| | เวลา ชุดที่ 1 |
| | 3.6/WR (External Data Memory Write Strobe) ขาควบคุมการเขียนข้อมูลจากหน่วยความจำภายนอก |
| | 3.7/RD (External Data Memory Read Strobe) ขาควบคุมการอ่านข้อมูลจากหน่วยความจำภายนอก |
| RST (ขา 9) | Reset ใช้สำหรับรีเซ็ตวงจรภายในชิพ เพื่อเริ่มต้นการทำงานใหม่ในการรีเซ็ตต้องป้อนลอจิก "1" นานอย่างน้อย 2 แมชชีนไซเคิล |
| ALE (ขา 30) | Address Latch Enable เป็นขาส่งสัญญาณออกไปภายนอก เพื่อควบคุมการ Latch ค่า Address ไบต์ต่ำจากพอร์ต 0 |
| PSEN (ขา 29) | Program Strobe Enable เป็นสัญญาณเพื่ออ่านคำสั่งจากหน่วยความจำโปรแกรมภายนอก เมื่อขานี้ Active มีลอจิกเป็น "0" จะอ่านโปรแกรมจาก หน่วยความจำภายนอก และ ถ้าเป็นการอ่านโปรแกรมภายในขานี้จะไม่มีการ Active |
| EA (ขา 31) | External Access เป็น ขาที่ใช้สำหรับเลือกว่าให้ทำงานจากหน่วยความจำ<br>โปรแกรมภายในหรือหน่วยความจำภายนอกชิพ เมื่อขานี้ Active จะมีลอจิกเป็น "0" จะเป็นการทำงานตามคำสั่งในหน่วยความจำโปรแกรมภายนอก |
| XTAL1 (ขา 19) | ใช้ต่อคริสตัลภายนอก โดยอินพุตเข้าสู่วงจรออสซิลเลเตอร์ |
| XTAL2 (ขา 18) | ใช้ต่อคริสตัลภายนอก โดยเอาท์พุตออกจากวงจรออสซิลเลเตอร์ |




รูปที่ 10 ขาต่างๆ ของ 8051

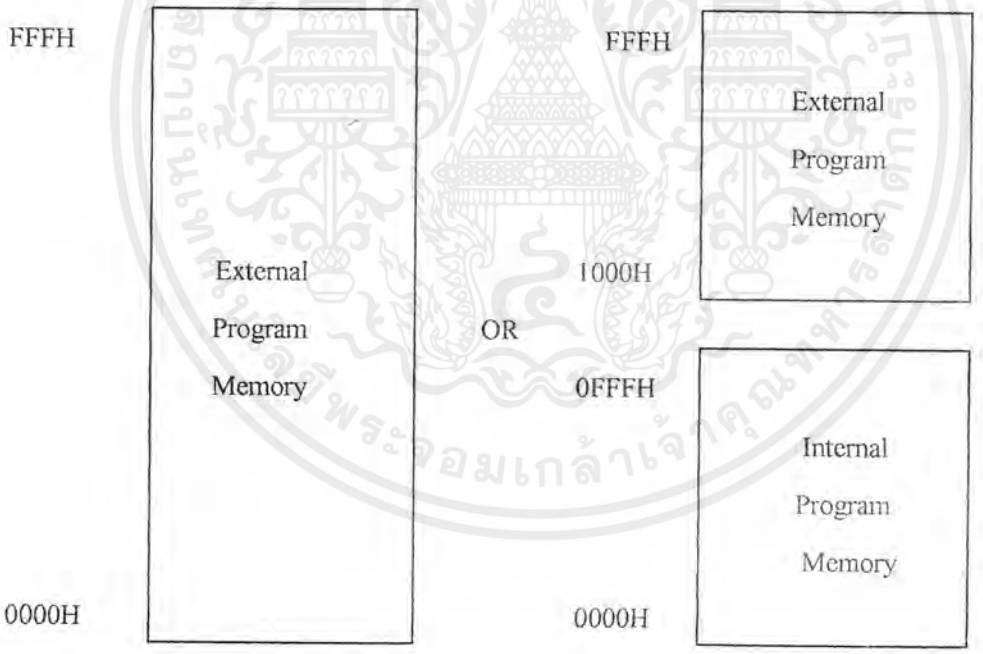



รูปที่ 11 โครงสร้างของพอร์ตต่างๆ ของ 8051

**2.2.1.4 การจัดการหน่วยความจำของ 8051**

หน่วยความจำของ 8051 แบ่งออกเป็น 2 แบบตามลักษณะการใช้งาน ดังนี้

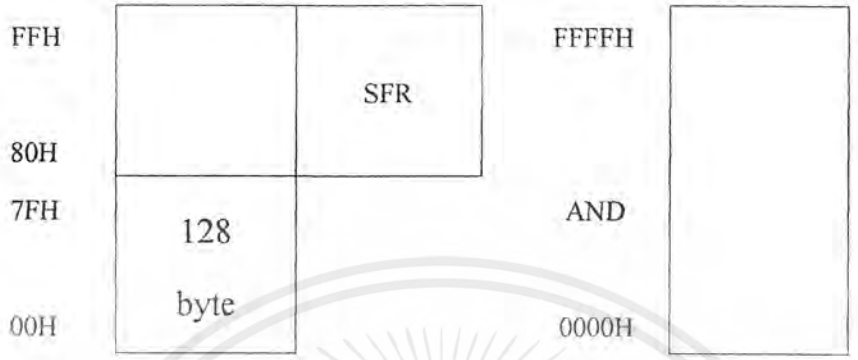
**1. หน่วยความจำโปรแกรม (Program Memory)** เป็นหน่วยความจำที่ใช้เก็บคำสั่งในรูปของภาษาเครื่องซึ่งต้องการให้ 8051 ทำงาน เมื่อ 8051 ทำงานก็จะอ่านข้อมูลที่เก็บในหน่วยความจำโปรแกรมไปทำการถอดรหัสแล้วสร้างสัญญาณควบคุมส่วนอื่นๆ ตามการทำงานของแต่ละคำสั่งนั้น หน่วยความจำนี้เป็นแบบรอม และ ผู้ใช้ต้องเขียนข้อมูลในแต่ละตำแหน่งของหน่วยความจำเป็นภาษาเครื่องของ 8051 ตามลำดับการทำงานที่ต้องการ ส่วนที่เป็นหน่วยความจำโปรแกรมก็ คือ รอมขนาด 4 กิโลไบต์ ดังรูปที่ 12




รูปที่ 12 การจัดพื้นที่หน่วยความจำสำหรับโปรแกรมของ 8051

**2. หน่วยความจำข้อมูล (Data Memory)** เป็นหน่วยความจำที่ใช้เก็บข้อมูล สามารถอ่าน

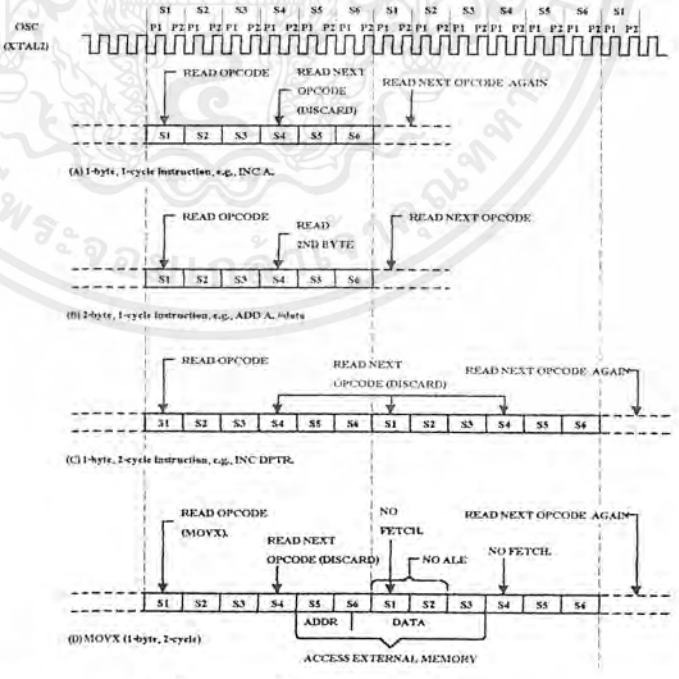
และเขียนข้อมูลได้ ซึ่งหน่วยความจำภายในมีขนาดเพียง 128 ไบต์ ส่วนหน่วยความจำภายนอกไอซีมีขนาด 64 กิโลไบต์ ดังรูปที่ 13




รูปที่ 13 การจัดพื้นที่หน่วยความจำข้อมูล

**2.2.1.5 ฐานเวลาในการทำงานของไมโครคอนโทรลเลอร์**

แมชชีนไซเคิล (Machine Cycle) คือรอบการทำงานของคำสั่ง เป็นค่าที่น้อยที่สุดในการทำคำสั่งใดคำสั่งหนึ่ง ถ้าเป็นคำสั่งที่ซับซ้อนมากก็ต้องใช้เวลานาน 2-3 แมชชีนไซเคิล 1 แมชชีนไซเคิล จะประกอบด้วยสัญญาณนาฬิกาจำนวน 12 ลูก โดยสัญญาณนาฬิกาแต่ละลูกเรียกว่า "เฟส" (Phase) สัญญาณนาฬิกา 2 เฟส รวมกันเป็น 1 สเตท (State) เพราะฉะนั้นใน 1 แมชชีนไซเคิลจึงมี 6 สเตท




รูปที่ 14 ผังเวลาของไมโครคอนโทรลเลอร์เบอร์ 8051

**2.2.1.6 การทำงานของ 8051**

เมื่อป้อนไฟเลี้ยงให้กับไมโครคอนโทรลเลอร์เบอร์ 8051 ซึ่งมีวงจรรีเซ็ตเมื่อปิดเครื่องจะเกิดการรีเซ็ตการทำงานภายในไมโครคอนโทรลเลอร์เบอร์ 8051 เริ่มจากภาคโปรแกรมเคาน์เตอร์ส่งค่าตำแหน่งหน่วยความจำสำหรับโปรแกรมลงไปบนเส้นทางหมายเลข 1 เส้นทางนี้ มีขนาด 16 บิต ค่าตำแหน่งหน่วยความจำนี้ถูกส่งไปเก็บไว้ที่ Program ADDR Register ค่าตำแหน่งหน่วยความจำจะปรากฏลงบนบัส 16 บิต หมายเลข 2 ถ้าเป็นค่าตำแหน่งหน่วยความจำแรกหลังจากการรีเซ็ต ค่าตำแหน่งหน่วยความจำจะเป็น 0000H หน่วยความจำสำหรับโปรแกรมการเลือกได้ว่าเป็นรอมภายในหรือภายนอก 8051 โดยการป้อนสภาวะลอจิกเข้าไปที่ 8051 ทางขา EA ซึ่งต่ออยู่กับส่วนของวงจรเวลาและควบคุม ถ้าป้อนสัญญาณลอจิก 0 เข้าที่ขา EA เป็นการเลือกใช้รอมภายใน 8051 โดยที่วงจรเวลา และ ควบคุมจะสร้างสัญญาณไปยังรอมภายใน ให้ส่งข้อมูลที่เป็นคำสั่งจากตำแหน่งที่ถูกชี้ด้วยค่าตำแหน่งที่ส่งมายังเส้นทางหมายเลข 2 ข้อมูลจากรอมถูกส่งไปยังเส้นทางหมายเลข 3 ที่เรียกว่าเส้นทางข้อมูลภายใน แล้วนำไปเก็บไว้ที่รีจิสเตอร์ IR (Instruction Register) เพื่อส่งไปให้กับวงจรเวลาและควบคุมทำการถอดรหัสแล้วควบคุมการทำงานส่วนอื่นๆ ต่อไป ในกรณีที่เลือกรอมภายนอกโดยป้อนลอจิก 1 เข้าที่ขา EA จะทำให้วงจรเวลาและควบคุมส่งสัญญาณไปยังพอร์ต 0 และพอร์ต 2 เพื่อส่งค่าตำแหน่งหน่วยความจำบนเส้นทางหมายเลข 2 ออกไปชี้หน่วยความจำภายนอก จากนั้นจะอ่านข้อมูลที่เป็นคำสั่งกลับเข้ามาทางพอร์ต 0 ไปยังเส้นทางข้อมูลภายในแล้วไปเก็บไว้ที่รีจิสเตอร์ IR เพื่อทำงานต่อไปเหมือนกับตอนอ่านคำสั่งจากรอมภายใน การทำงานในช่วงค่าตำแห่งในหน่วยความจำไปยังหน่วยความจำ แล้วอ่านข้อมูลที่เป็นคำสั่งกลับเข้ามาเก็บไว้ในรีจิสเตอร์ IR เรียกว่า "ช่วงของการเฟตช์" (Fetch) ช่วงต่อไปจะเป็นช่วงของการทำงานตามคำสั่งเรียกว่า "Execute Cycle"

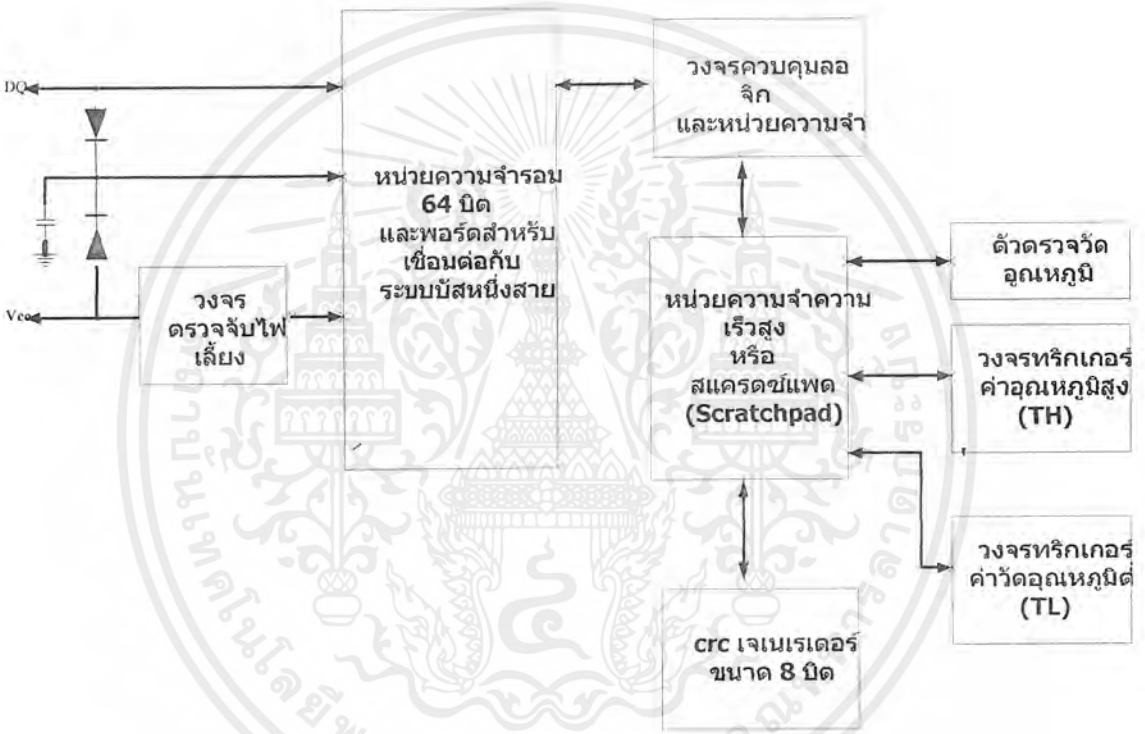
**2.2.2ไอซีตรวจจับอุณหภูมิ DS 1820**

เป็นไอซีตรวจจับอุณหภูมิที่ใช้ในการติดต่อแบบระบบบัสหนึ่งสาย มีขาต่อใช้งานเพียง 3 ขา คือ DQ ซึ่งเป็นขาเชื่อมต่อบัส ขาต่อไฟเลี้ยงภายนอก และขากราวด์ ดังในการแสดงการจัดขาของ ไอซี DS 1820 ในรูปที่ 15 และมีโครงสร้างภายในแสดงในรูปที่

หัวใจสำคัญของ DS 1820 อยู่ที่ตัวจับอุณหภูมิและหน่วยความจำความเร็วสูง ที่เรียกว่าสแครตช์แพด (scratchpad) ซึ่งมีขนาด 9 ไบต์ มีการจัดสรรหน่วยความจำส่วนนี้แสดงในรูปที่

เมื่อวัดอุณหภูมิได้ก็นำค่าที่วัดได้นี้มาเก็บไว้ที่สแครตช์แพด ที่ไบต์ 0 และ 1 ทั้งนี้เนื่องจากไอซี DS 1820 สามารถให้ข้อมูลของอุณหภูมิได้ละเอียดถึง 16 บิต เมื่อนำมาแปลงข้อมูลเลข

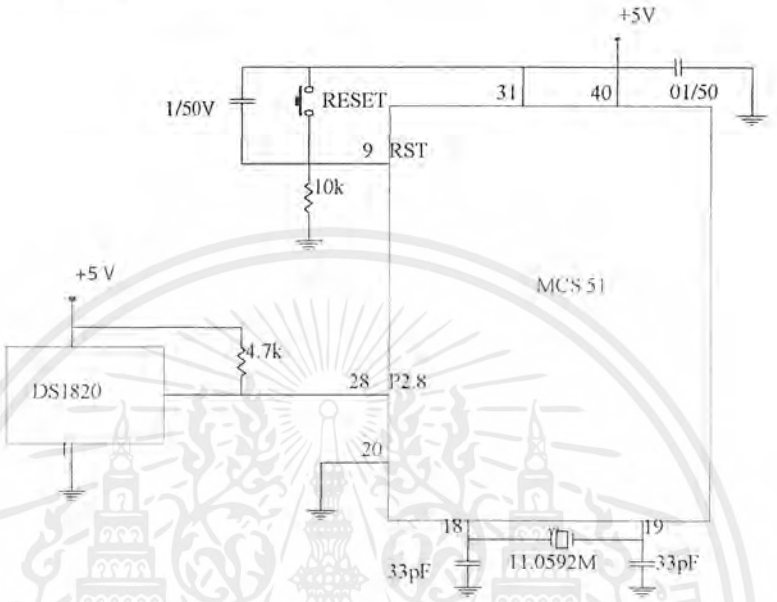
ฐานสิบ จึงสามารถแสดงความละเอียดของค่าอุณหภูมิได้ถึง 0.5 องศาเซลเซียสและ 0.9 องศาฟาเรนไฮต์โดยมีย่านวัดอุณหภูมิ –55 ถึง +125 องศาเซลเซียล หรือ-67 ถึง +257 องศาฟาเรนไฮต์ โดยค่าขององศาฟาเรนไฮต์ ต้องใช้การแปลงหน่วยเข้ามาช่วย ใช้เวลาในการแปลงค่าอุณหภูมิเป็นข้อมูลดิจิตอลประมาณ 200 มิลลิวินาที สามารถกำหนดขอบเขตของอุณหภูมิที่วัดได้ และแจ้งเตือนเมื่อค่าอุณหภูมิสูงขึ้นหรือต่ำลง ถึงค่าที่กำหนด โดยค่าอุณหภูมิที่กำหนดนี้จะเก็บไว้ที่สแครตซ์แพด ในไบต์ที่ 2 และที่ 3




รูปที่ 15 โครงสร้างการทำงานภายในของไอซีตรวจจับอุณหภูมิ

**2.2.2.1 การควบคุมการทำงานของ DS1820**

ในการติดต่อกับไอซี DS1820 จะมีคำสั่งที่ต้องส่งให้แก่ DS 1820 เพื่อกำหนดรูปแบบการทำงาน คำสั่งที่ใช้มากที่สุดมีด้วยกัน 3 คำสั่งดังนี้




รูปที่ 16 การเชื่อมต่อ DS1820 กับไมโครคอนโทรลเลอร์

1. คำสั่งไม่ติดต่อกับหน่วยความจำรอมหรือสคิปรอม9 (Skip ROM) เนื่องจาก ในการใช้งาน DS1820 โดยปกติแล้วจะมี DS1820 อยู่บนสายสัญญาณเพียงสายเดียว จึงไม่จำเป็นต้องใช้ข้อมูลกำหนดแอดเดรส ดังนั้นจึงไม่จำเป็นต้องติดต่อหน่วยความจำรอมเพื่ออ่านข้อมูล ข้อมูลของคำสั่งสคิปรอมที่ต้องส่งให้ DS 1820 คือ 0CCH
2. คำสั่งในการแปลงอุณหภูมิ (Convert T) มีค่าเท่ากับ 44H เมื่อส่งคำสั่งนี้ให้ DS1820 จะต้องทำการวนลูปอย่างน้อย 200 มิลลิวินาที เพื่อให้DS1820 ได้ใช้เวลานี้ ในการแปลงค่าอุณหภูมิเป็นข้อมูลดิจิตอล มาเก็บไว้ในสแตรตช์แพด
3. คำสั่งในการอ่านข้อมูลจากสแตรตช์แพด (Read Scratchpad) มีค่าเท่ากับ 0BEH เมื่อส่งคำสั่งนี้ DS1820 จะทยอยส่งข้อมูลค่าอุณหภูมิออกมาทั้งหมด 9 ไบต์

### 2.2.3 ทฤษฎี DOT MATRIX LCD MODULE

อุปกรณ์ในปัจจุบันนี้ในส่วนแสดงผลนั้นจะใช้ LCD เป็นส่วนใหญ่ ไม่ว่าจะเป็นเครื่องเล่น VIDEO, เครื่องถ่ายเอกสาร , เครื่องมือวัดคุมฯ , เครื่องคอมพิวเตอร์ เป็นต้น เราพอจะแบ่ง DOT MATRIX LCD MODULE นี้ออกได้เป็นพวกๆดังนี้

1. CHARCTER LCD MODULE
2. GRAPHIC LCD MODULE
3. SEGMENT DISPLAY TYPE LCD MODULE

โดยในแต่ละแบบนี้ก็จะมีส่วนประกอบใหญ่ๆแบ่งได้เป็น

1. DOT MATRIX LCD เป็นตัวแสดงผลให้เรามองเห็น ในลักษณะการปิดและเปิดตัวเองกับแสงก็คือส่วนของที่เป็นตัวกระจกบรรจุผลึก
2. DRIVER เป็นตัวรับสัญญาณจากตัวควบคุมมาขับผลึก LCD อีกทีหนึ่ง โดยมีเบอร์ที่นิยมใช้คือ HD44100H, MSM5259
3. CONTROLLER เป็นตัวรับข้อมูลจาก อุปกรณ์ภายนอกมาและ จัดการควบคุม LCD MODULE ให้ทำงานแสดงผลต่างๆเช่น โดยมีเบอร์ IC ที่นิยมใช้กันคือ HD4478 ซึ่งจะใช้แบบ CHARCTER LCD MODULE เป็นส่วนใหญ่และ IC HD61830 จะใช้ในแบบ GRAPHIC LCD MODULE

ในการศึกษาการทำงานและใช้งาน LCD MODULE นั้นไม่ใช่เรื่องยากเลยถ้าเราสามารถทำความเข้าใจในส่วนของ CONTROLLER ได้ก็เพียงพอแล้วและโดยมาก LCD MODULE ในแต่ละบริษัทจะใช้ CONTROLLER ที่มีหลักการทำงานเหมือนกันเป็นส่วนใหญ่ และใน LCD MODULE แต่ละขนาดจำนวนตัวอักษรหรือจำนวนบรรทัดก็มีหลักการทำงานแบบเดียวกันทั้งหมด IC ที่นิยมใช้เป็น CONTROLLER LCD คือเบอร์ HD4470 จะใช้แบบ CHARCTER LCD MODULE เป็นส่วนใหญ่และ IC HD61830 จะใช้ในแบบ GRAPHIC LCD MODULE

HD44780เป็นไอซีLSI ตัวหนึ่งใช้ควบคุม LCD โดยแสดงผลในรูปตัวอักษรหรือสัญลักษณ์ต่างๆ ตัวมันเองสามารถต่อใช้งานแบบ 4 BIT หรือ 8 BIT ก็ได้ โดยถ้าเราต่อแบบ 4 BIT จะต่อใช้งานที่ DB7-DB4 เท่านั้น โดยข้อมูลครั้งแรกที่ส่งนั้น HD44780 จะถือเป็นข้อมูล 4 BIT บน และข้อมูลที่ส่งต่อมาเป็นข้อมูล 4 BIT ล่าง

2.2.3.1 รายละเอียดของคำสั่ง HD44780

1. CLEAR DISPLAY

| RS | R/W | DB7 | | | | | | | DB0 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

รูปที่ 17 แสดงคำสั่ง CLEAR DISPLAY

คำสั่งนี้จะเป็นการเขียนช่องว่างหรือ SPACE (ASCII 20H) เข้าไปใน DDRAM ทั้งหมดและทำการ SET DD RAM ADDRESSER เป็นศูนย์ตัว CURSUR จะกลับไปอยู่ตำแหน่งบนสุดซ้ายมือของจอภาพ SET I/D=1 และ S ไม่มีการเปลี่ยน

2. RETURN HOME

| RS | R/W | DB7 | | | | | | | DB0 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | * |

รูปที่ 18 แสดงคำสั่ง RETURN HOME(* No effect)

คำสั่งนี้จะทำการ SET DD RAM ADDRESSER เป็นศูนย์ตัว CURSOR จะกลับไปอยู่ตำแหน่งบนสุดซ้ายมือของจอภาพไม่เปลี่ยน

3. ENTRY MODE SET

| RS | R/W | DB7 | | | | | | | DB0 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | I/D | S |

รูปที่ 19 แสดงคำสั่ง ENTRY HOME

BIT I/D : โดยจะเป็นตัวกำหนดให้ว่าเมื่อเขียนหรืออ่านข้อมูลแล้วจะทำให้ DD RAM ADDRESS เพิ่มขึ้นหนึ่งหรือลดลงหนึ่งโดย 1= เพิ่ม และ 0=ลดลงหนึ่ง

BIT S : เป็นการกำหนดแสดงผลโดยถ้า

S=1 จะเป็นการใส่ข้อมูลแล้วตัว CURSOR อยู่กับที่ข้อมูลจะถูกดันไปทางซ้าย

S=0 ข้อมูลจะอยู่กับที่ ตัว CURSOR จะถูกดันไปทางขวามือ

**4. DISPLAY ON/OFF CONTROL**

| RS | R/W | DB7 | | | | | | | DB0 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | D | C | B |

รูปที่ 20 แสดงคำสั่ง DISPLAY ON/OFF CONTROL

BIT D : ใช้กำหนดการเปิดปิดจอภาพ

D=1 จะ ON

D=0 จะ OFF

BIT C : จะแสดง CURSOR ให้ BIT C=1 และถ้าไม่ต้องการแสดง CURSOR BIT C=0 โดยตัว CURSOR จะอยู่ที่ LINE 8 ในแบบ 5X7 DOT และจะอยู่ที่ LINE 11 ในแบบ 5X10 DOT

BIT B : เป็น BITSET การกระพริบของ CURSOR โดย B=1 จะกระพริบ B=0 จะไม่กระพริบ โดยมีระยะเวลากระพริบประมาณ 379.2 ms

**5. CURSOR OR DISPLAY SHIFT**

| RS | R/W | DB7 | | | | | | | DB0 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | S/C | R/L | * | * |

รูปที่ 21 แสดงคำสั่ง CURSOR OR DISPLAY SHIFT(*No effect)

เป็นคำสั่งกำหนดให้ตำแหน่ง CURSOR หรือข้อมูลไปเกิดทางซ้ายหรือขวาโดยไม่ต้องใช้คำสั่งเขียนหรืออ่าน

ตารางที่ 2 แสดงค่าการเลื่อน CURSOR ไปแต่ละตำแหน่ง

| S/C | R/L | |
|---|---|---|
| 0 | 0 | ทำการย้าย CURSOR ไปจากตำแหน่งเดินไปทางซ้ายมือ 1 ตำแหน่ง |
| 0 | 1 | ทำการย้าย CURSOR ไปจากตำแหน่งเดินไปทางขวามือ 1 ตำแหน่ง |
| 1 | 0 | เป็นการดันตัวอักษรที่เกิดไปทางซ้าย |
| 1 | 1 | เป็นการดันตัวอักษรที่เกิดไปทางขวา |

**6. FUNCTION SET**

RS R/W DB7.....DB0

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | DL | N | F | * | * |

รูปที่ 22 แสดงคำสั่ง FUNCTION SET

BIT DL : เป็นการ SET การติดต่อจะให้เป็นแบบ 8 BIT หรือ 4 BIT ถ้าต้องการ 4BIT DL=1

N : เป็นการ SET บรรทัดการแสดงผล N=0 แสดง 1 บรรทัด N=1 แสดง 2 บรรทัด ในกรณีมากกว่า 2 บรรทัด ก็ให้ SET N=1

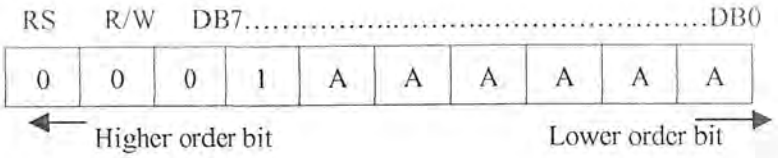
F : เป็นการ SET ขนาด DOT การแสดงผล 5X7 หรือ 5X10 โดย F=0 เป็นแบบ 5X7 และ F=1 เป็นแบบ 5X10

ตารางที่ 3 แสดงรายละเอียดย่อยของคำสั่ง FUNCTION SET

| N | F | No. of display line | Character Font | Duty factor | Remarks |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 5X7 dots | 1/8 | |
| 0 | 1 | 1 | 5X10 dots | 1/11 | |
| 0 | * | 2 | 5X7 dots | 1/16 | Cannot display 2 line with 5X10 dot character font |

- No effect

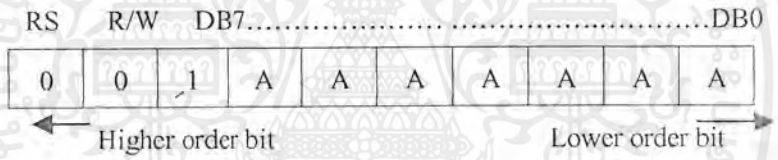
**7. SET CG RAM ADDRESS**




รูปที่ 23 แสดงคำสั่ง SET CG RAM ADDRESS

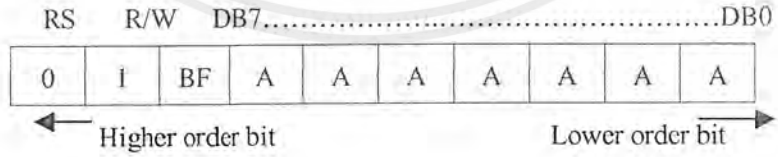
ใน HD44780 นั้นจะมีหน่วยความจำอยู่ 2 ชุดคือ DISPLAY DAT RAM ( DD RAM ) จำนวน 80X8 BIT และ CHARACTER GENERATOR ROM CG RAM จำนวน 512 BIT และ 7200 BIT คำสั่งนี้จะเป็นการ SET ADDRESS ใน CG RAM โดยต้องทำการ SET ADDRESS ก่อนเขียนหรืออ่านข้อมูลจาก CG RAM ด้วย

**8. SET DD RAM ADDRESS**




รูปที่ 24 แสดงคำสั่ง SET DD RAM ADDRESS

**9. READ BUSY FLAG AND ADDRESS**




รูปที่ 25 แสดงคำสั่ง READ BUSY FLAG AND ADDRESS

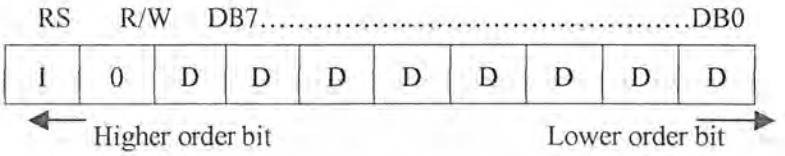
เป็นคำสั่งอ่านค่า BUSY FLAG ซึ่งจะเป็นตัวบอกว่าตัว HD44780 นี้อยู่ในขบวนการทำงานภายในอยู่หรืออยู่ในสภาพพร้อมรับข้อมูล โดย

BF=1 หรืออยู่ในขบวนการทำงานภายใน ไม่พร้อมที่จะรับข้อมูลหรือคำสั่ง

BF=0 พร้อมจะรับข้อมูลหรือคำสั่งแล้ว

และนอกจากนี้ยังเป็นคำสั่งอ่านข้อมูล ADDRESS ของ CG RAM หรือ DD RAM ด้วย

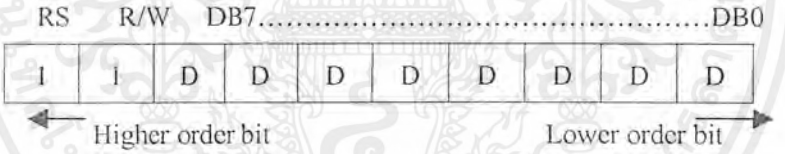
**10. WRITE DATA TO CG หรือ DD RAM**




รูปที่ 26 แสดงคำสั่ง WRITE DATA TO CG หรือ DD RAM

เป็นคำสั่งเขียนข้อมูลเข้าไปใน CG หรือ DD RAM โดยเมื่อเขียนข้อมูลและ ADDRESS จะเพิ่มหรือลดโดยอัตโนมัติตามคำสั่งที่ SET ใน ENTRY MODE ข้อกำหนดที่จะรู้ว่าเป็นการเขียนข้อมูลของ CG RAM หรือ DD RAM ทำได้โดยการ SET ADDRESS ของ CG RAM หรือ DD RAM ขึ้นมาก่อนจะเขียนข้อมูล

**11. READ DATA FROM CG RAM OR DD RAM**




รูปที่ 27 แสดงคำสั่ง WRITE DATA TO CG หรือ DD RAM

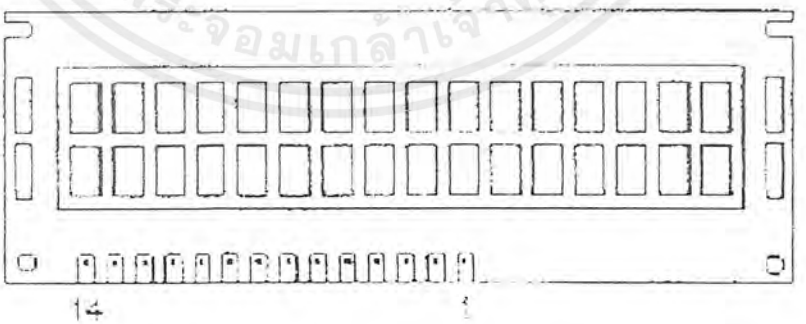
เป็นคำสั่งอ่านข้อมูลจาก CG RAM หรือ DD RAM โดยอ่านจาก DD RAM หรือ CG RAM นี้ควรจะใช้คำสั่ง SET ADDRESS ก่อนเพื่อให้รู้ว่าข้อมูลที่อ่านได้นั้นเป็น DD หรือ CG RAM จากตารางการทำงานจะเห็นได้ว่าการใช้งาน LCD MODULE นั้นง่าย เพียงแต่เราส่งคำสั่งเริ่มแรกและ SET ตามความต้องการในขนาดตัวอักษร, CURSOR หลังจากนั้นเราก็สามารถเขียนตัวอักษรเข้าไปใน DD RAM ได้

**ขาต่างๆในการต่อใช้งาน HD44780**

1. RS (REGISTER SELECTION) จะเป็นขาเลือก REGISTER ภายในซึ่งมีอยู่ 2 ตัว คือ INSTRUCTION REGISTER (IR) และ (DATA REGISTER (DR) โดยถ้าเป็น 1 จะเป็นการเลือก DATA และถ้าเป็น 0 จะเป็นการเลือก INSTRUCTION

2. R/W (READ/WRITE) เป็นตัวเลือกว่าจะเขียนหรือจะอ่านข้อมูลจากตัว IC โดย อ่านข้อมูล=1 และเขียนข้อมูล =0
3. EN(ENABLE SIGNAL) เป็นขากำหนดสภาพการรับเขียนอ่านข้อมูล
4. DB0-DB7 เป็นขาขอรับการส่งข้อมูลจากตัว IC
5. VDD ไฟเลี้ยงวงจร +5V
6. VSS เป็นขา GND
7. VO เป็นขารับ VOLTAGE ในการขับ LCD ให้สว่างหรือมืด

| RS | R/W | E | |
|---|---|---|---|
| 0 | 0 |  | IR write as internal operation (Display clear,etc.) |
| 0 | 1 |  | Read busy flag (DB7) and address counter (DB0-DB6) |
| 1 | 0 |  | DR Write as internal operation (DR to DD or CG RAM) |
| 1 | 1 |  | DR read as internal operation (DD or CG RAM to DR) |




รูปที่ 28 แสดงตำแหน่งขาของ LCD MODULE

# บทที่3
## หลักการและการออกแบบ

**แนวคิด**

เนื่องจากมีความจำเป็นในการนำสัญญาณที่วัดได้ที่อยู่ห่างไกลหลายๆสัญญาณ ในการตรวจจับอุณหภูมิ เพื่อนำมาแสดงผลบน LCD ที่ติดระยะไกล ดังนั้นเราจำเป็นต้องมีวิธีการที่เหมาะสม ในการส่งข้อมูลเพื่อเป็นการประหยัดต่อระบบโดยรวม ระบบ TDM เป็นวิธีหนึ่งที่สามารถส่งข้อมูลหลายๆช่องไปในสายเดียว ให้อยู่ในช่องเวลาเดียวกัน

**ขั้นตอนการออกแบบ**


รูปที่ 29 Block Diagram ของ ระบบ TDM

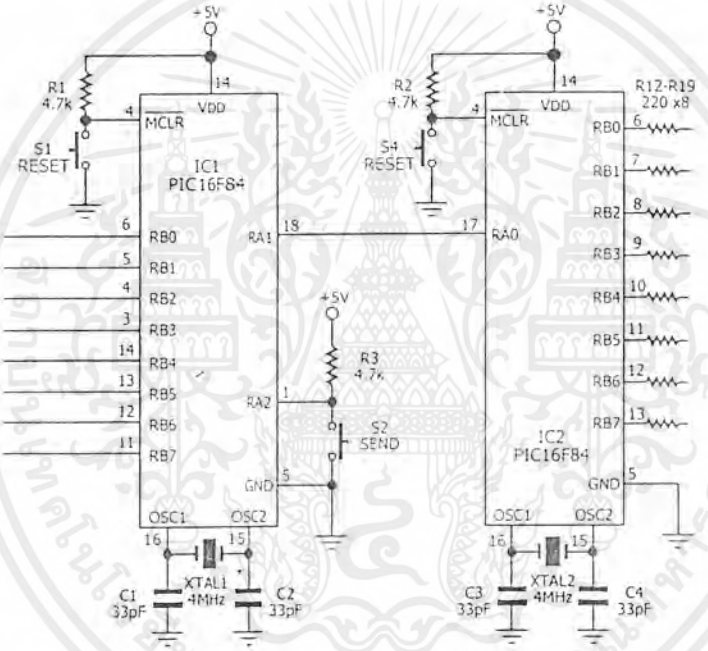
จากรูปที่ 29 Block Diagram ของระบบการส่งสัญญาณ TDM ประกอบไปด้วย 2 ส่วน คือ ส่วนแรกเป็นส่วนของภาคส่ง TDM ทำหน้าที่ในการพาสัญญาณที่ได้จากการตรวจจับอุณหภูมิ ที่ได้จาก Sensors ซึ่งจะอยู่ในรูปของไบนารี แล้วมาทำการมอดูเลเตอร์ แบบTDM โดยการควบคุมด้วยไมโครคอนโทรลเลอร์ MCS-51 สำหรับส่วนที่สองคือภาครับ สัญญาณแบบ TDM ทำหน้าทีในการรับ และแยกสัญญาณที่ได้จากการมัลติเพล็กซ์แบบ TDM ควบคุมด้วยไมโครคอนโทรลเลอร์ ออกเป็นแต่ละช่องสัญญาณ (Channel) จากนั้นทำการแยกข่าวสารจากการมอดูเลเตอร์แบบ TDM ในแต่ละช่องสัญญาณ

**3.1การสื่อสารข้อมูลระหว่าง PIC 16F84 แบบอนุกรม**

ระบบควบคุมในปัจจุบันนี้มีความซับซ้อนมากขึ้น จนในบางระบบไม่สามารถใช้ไมโครคอนโทรลเลอร์เพียง 1 ตัวควบคุมการทำงานได้หมด กอรปกับความนิยมที่ต้องการใช้ระบบควบคุมขนาดเล็กเข้ามาใช้งานเพิ่มขึ้น ทั้งนี้เพื่อต้องการเพิ่มประสิทธิภาพให้กับระบบ โดยการควบคุมระบบใหญ่จะบรรจุโมดูลของระบบควบคุมขนาดเล็กมากมายที่ทำหน้าที่แตกต่างกันออกไป โดยจะมีหน่วยควบคุมหลักคอยรับและส่งข้อมูลติดต่อกับโมดูลขนาดเล็กเหล่านั้น การทำงานใน

ลักษณะนี้บางที่เรียกว่าการทำงานแบบมัลติโปรเซสเซอร์ (Multi-processor) กล่าวคือเป็นระบบงานที่ใช้ไมโครโปรเซสเซอร์มากกว่า 1 ตัว ในการควบคุมระบบการทำงาน

ใน PIC 16F84 มีหน่วยความจำขนาด 1 กิโลเวิร์ด มีพอร์ตอินพุต เอาพุต 13 บิต จึงสามารถรองรับงานได้ในระดับหนึ่ง เมื่อต้องการนำ PIC 16F84 ไปควบคุมงานที่มีความซับซ้อนมากขึ้น ต้องการพอร์ตมากขึ้น ความจำเป็นที่ต้องการขยายระบบจึงตามมา และเนื่องจาก PIC 16F84 ไม่สามารถต่อใช้งานต่อหน่วย ความจำภายนอกได้ การขยายระบบโดยใช้ไมโครคอนโทรลเลอร์มาพ่วงต่อจึงเป็นแนวทางที่ดีที่สุด




รูปที่ 30 การเชื่อมต่อ PIC 16F84 ในลักษณะอนุกรม

### 3.1.1 การสื่อสารข้อมูลระหว่าง PIC16F84 ในลักษณะอนุกรม

ด้วยข้อจำกัดของจำนวนพอร์ต ทำให้เมื่อต้องการต่อพ่วง PIC16F84 จึ่งมีแนวทางในการต่อเชื่อมเพียงลักษณะเดียวคือ การต่อพ่วงในลักษณะอนุกรม(Serial-communication) เป็นวงจรการเชื่อมต่อ PIC16F84 2 ตัว เข้าด้วยกัน ในลักษณะอนุกรม ข้อมูลจาก IC1 ซึ่งได้มาจากการป้อนค่าโดยดิปสวิตซ์ จะถูกส่งมาทางขา RA1 ในลักษณะอนุกรม นั้นคือข้อมูลถูกเลื่อนมาจาก IC1 ครั้งละบิต โดยมีรูปแบบการส่งข้อมูลตามรูปที่ 31 ซึ่งประกอบด้วย บิตข้อมูล(data bit) และบิตหยุด (stop bit)


รูปที่ 31 รูปแบบการสื่อสารข้อมูลอนุกรมของ PIC 16F84

การสื่อสารข้อมูลระหว่าง PIC 16F84 2 ตัวในลักษณะอนุกรมในรูปที่ 31 กำหนดให้ IC1 เป็นตัวส่งข้อมูลออกมาทางขา RA1 ส่วน IC2 กำหนดเป็นตัวรับข้อมูล โดยข้อมูลเข้ามาทางที่ขา RA0 ทั้ง IC1 และ IC2 ได้รับการกำหนดให้ทำงานในโหมด XT ใช้สัญญาณจากนาฬิกาจากคริสตอล 4 MHz เท่ากัน และใช้โปรแกรมหน่วงเวลาที่มีคาบเท่ากัน

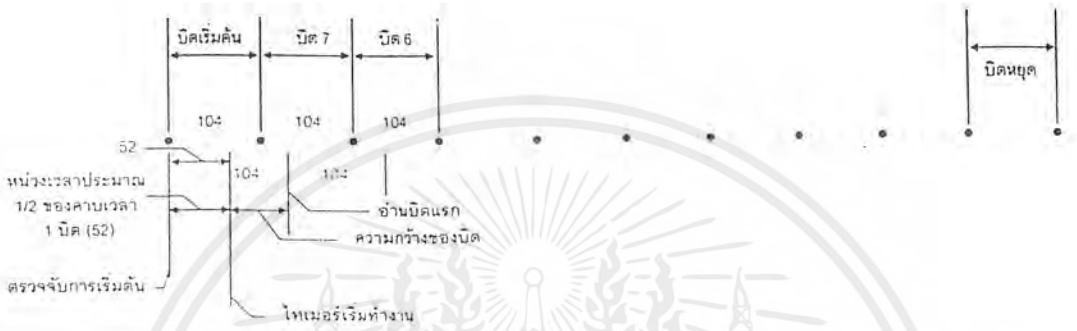
จุดหนึ่งที่ต้องคำนึงถึงก็คือ จังหวะในการรับ และ ส่งข้อมูล จะต้องสอดคล้องกัน ดังนั้นจึงต้องมี การกำหนดอัตราการรับ และส่งข้อมูลที่เรียกว่า อัตราบอด หรือ บอดเรต (Baud rate) มีหน่วยเป็นบิต ต่อ วินาที (bit per second: bps) จากการกำหนดรูปแบบข้อมูลในภาพที่ 31 ข้อมูลในแต่ละบิตมีคาบเวลา 104 ไซเคิล ของสัญญาณนาฬิกาภายในไมโครคอนโทรลเลอร์ เมื่อสัญญาณนาฬิกาภายในมีความถี่เท่ากับ ¼ ของความถี่จากแหล่งภายนอก ซึ่งในที่นี้ก็คือ คริสตอล ค่า 4 MHz ดังนั้นความถี่ของสัญญาณนาฬิกาใน PIC 16F84 จึงเท่ากับ 1 MHz คาบต่อ 1 ไซเคิล จึงเท่ากับ 1 วินาที ดังนั้นคาบเวลาของข้อมูล 1 บิต จึงเท่ากับ 104 ไมโครวินาที่ สามารถคำนวณอัตรา บอด ได้เท่ากับ 9.615 บิตต่อวินาที โดยประมาณ

### 3.1.2 กระบวนการรับส่งข้อมูล

พิจารณาในรูปที่ 30 และที่ 31 ประกอบกัน ในภาวะที่ไม่มีการรับ การส่งข้อมูล สถานะลอจิกที่ขา RA1 ของ IC1 ซึ่งต่อเข้ากับขา RA0 ของ IC2 จะเป็น "1" ค้างอยู่ตลอดเวลา ซึ่งในที่นี้เรียกว่า "มาร์ก" (mart) เมื่อขาที่ใช้ส่งข้อมูล(transmit data line: TD) ซึ่งก็คือขา RA1 ของ IC1 เป็น "0" ต่อจากสถานะมาร์กยาวนานถึง 104 ไซเคิลของสัญญาณนาฬิกาภายใน จะเรียกที่เกิดขึ้นนี้ว่า บิตเริ่มต้น อันเป็นการแจ้งให้ตัวรับเตรียมพร้อมในการรับข้อมูล

ทางด้านตัวรับ เมื่อเริ่มทำงาน ซีพียูจะวนการทำงานเพื่อรอบิตเริ่มต้นจากตัวส่ง เมื่อพบหรือตรวจจับบิตเริ่มต้นได้ โปรแกรมมอนิเตอร์ของตัวรับ จะควบคุมให้ซีพียูรอหรือหน่วงเวลาอีกครึ่งหนึ่งของคาบเวลาข้อมูล 1 บิต ซึ่งในที่นี้เท่ากับ 52 ไซเคิล ถ้าหากที่ขารับข้อมูล (receive

data: RD) ยังคงเป็นลอจิกที่ "0" ซีพียูจะเริ่มอ่านข้อมูลบิตแรกเข้าสู่หน่วยความจำต่อไป แต่ถ้าหากหลังจากการหน่วงเวลา 52 ไซเคิลแล้ว ที่ขารับข้อมูลเกิดลอจิก "1" ขึ้นมา แสดงว่า การส่งบิตเริ่มต้นจากตัวส่งเกิดความผิดพลาด ซีพียูจะยกเลิกการรับส่งข้อมูล แล้วกลับไปรอรับบิตเริ่มต้นใหม่อีกครั้ง ในรูปที่ 32 แสดงรายละเอียดของคาบเวลารับข้อมูลของตัวรับ




รูปที่ 32 คาบเวลาของการสื่อสารข้อมูลอนุกรม

หลังจากที่ตัวรับสามารถจะรับบิตเริ่มต้นได้ ก็จะเตรียมรับข้อมูลที่จะทยอยเข้ามา โดยบิต MSB ของข้อมูลจะถูกส่งเข้ามาก่อน จนครบ 8 บิต ตัวส่งก็จะส่งบิตหยุดไปที่ตัวรับ บิตหยุดเกิดจากการทำให้ขาส่งข้อมูลเป็น "1" หรืออยู่ในสถานะมาร์กยาวนานกว่า 104 ไซเคิลของสัญญาณนาฬิกาภายในไมโครคอนโทรลเลอร์ หรือยาวนานกว่า 104 ไมโครวินาที เมื่อทำการรับส่งข้อมูลในแต่ละไบต์หรือเวิร์ดแรกเรียบร้อยแล้ว ก็จะทำการรับส่งข้อมูลในไบต์หรือบิตต่อไป

**3.2 ทางภาคส่ง**


รูปที่ 33 ไดอะแกรมของภาคส่ง

เมื่อ ไอซี DS1820 ทำการตรวจจับอุณหภูมิแล้ว จะทำการติดต่อแบบระบบบัสหนึ่งสาย เพื่อทำการส่ง 1 ช่องสัญญาณ แต่ในโครงงานนี้เราจะใช้รับอินพุท(Input) 2 ช่องสัญญาณ ส่งต่อไปยังส่วนของการมัลติเพล็กซ์ จะมีสัญญาณข้อมูล 8บิต เป็นแบบขนาน ส่งไปยังภาคของการส่งสัญญาณ จะใช้ไอซี PIC 16F84 เป็นตัวแปลงสัญญาณข้อมูลแบบขนานเป็นสัญญาณข้อมูลแบบอนุกรม แล้วทำการส่งต่อไปยังภาครับโดยผ่านสายโทรศัพท์

### 3.2.1 การเชื่อม DS 1820 กับไมโครคอนโทรลเลอร์ MCS 51

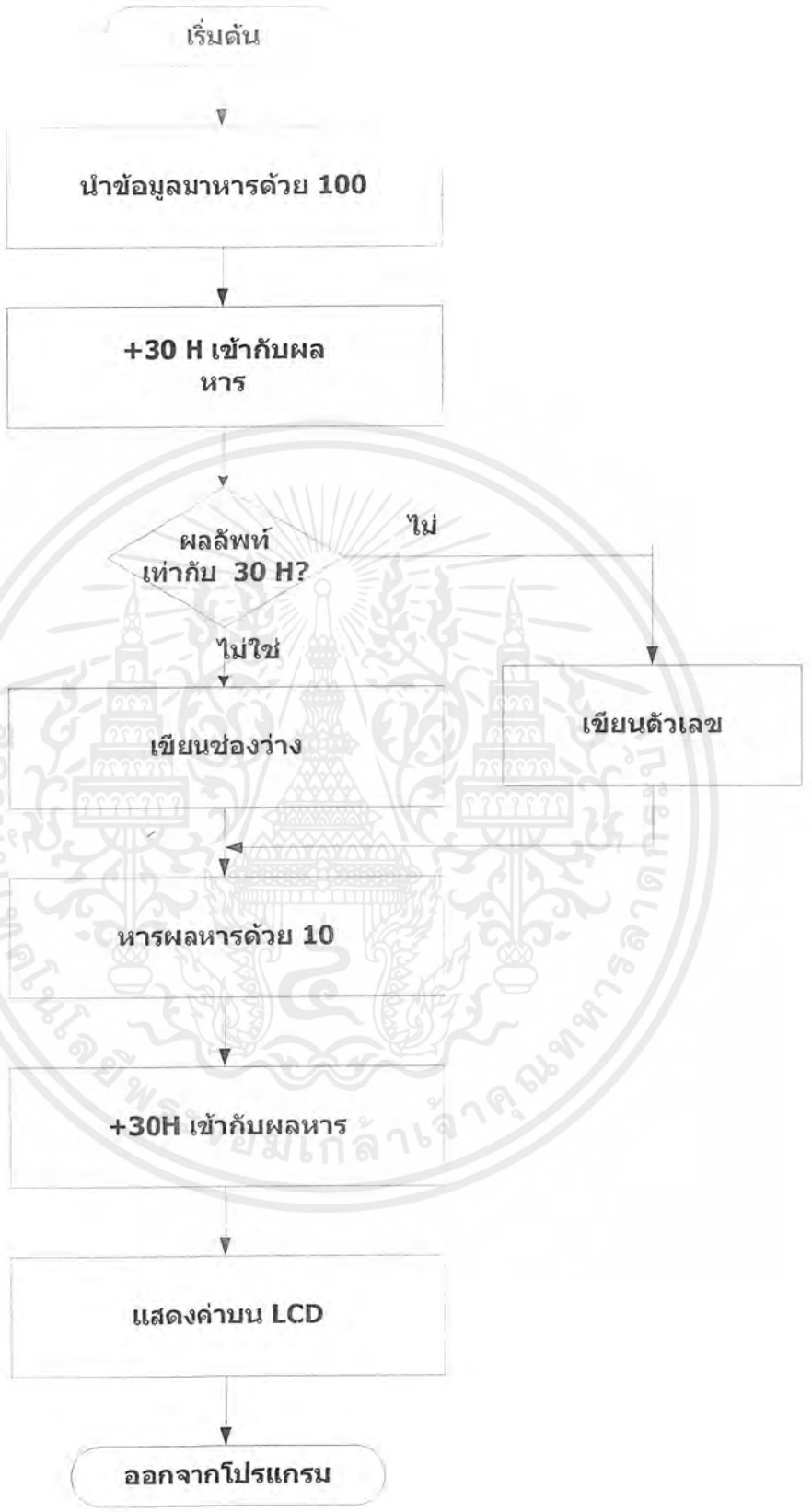
แสดงวงจรการเชื่อมต่อแสดงในรูปที่ 34 ใช้ขาพอร์ต เพียง 1 ขา เท่านั้น สำหรับการเชื่อมต่อกับ DS 1820 โดยต้องมีตัวต้านทาน 4.7 kΩ ต่อพูลอัปกับไฟเลี้ยง +5 V จากนั้นจึงทำการเขียนโปรแกรมเพื่อติดต่อกัน โดยใช้รูปแบบการติดต่อตามมาตรฐานระบบบัสหนึ่งสายของดัลดัส


รูปที่ 34 ไอซี DS1820

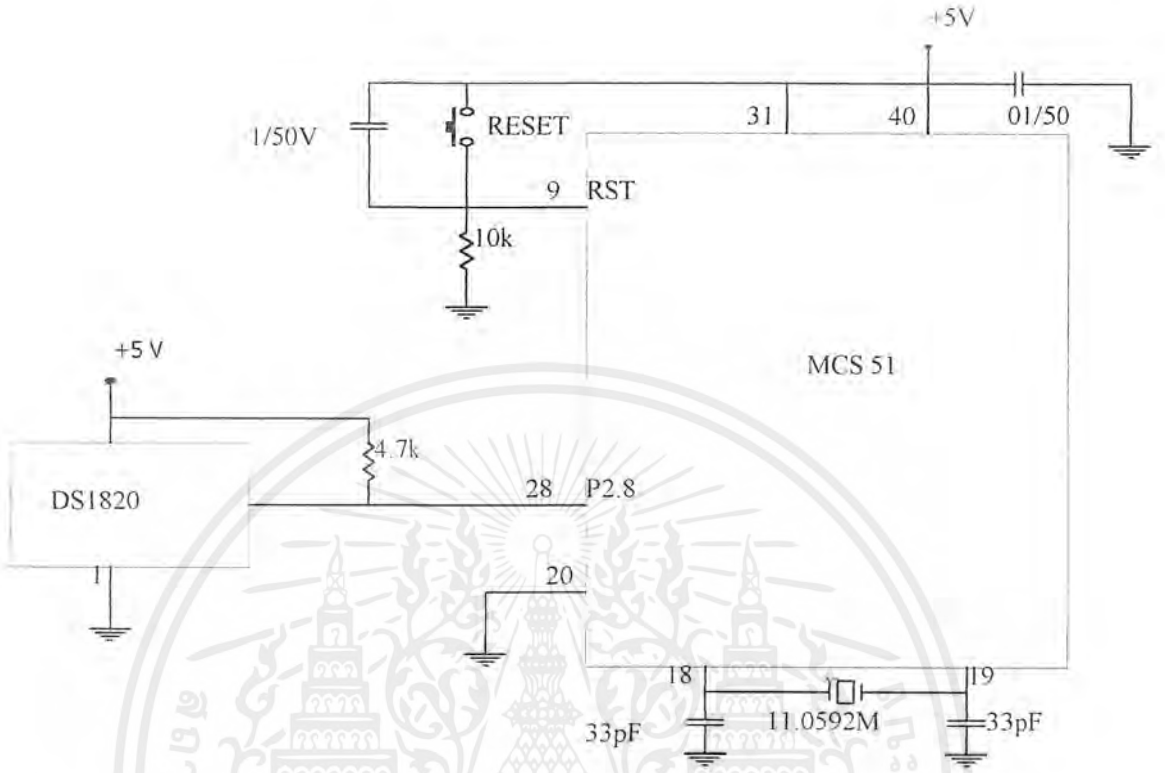
### 3.2.2 การเขียนโปรแกรมเพื่อติดต่อกับ DS1820

จากรายละเอียดของรูปแบบการสื่อสารในระบบบัสหนึ่งสายที่กล่าวมาข้างต้น สามารถนำมาใช้เพื่อเป็นการเขียนข้อมูลในการเขียนโปรแกรมติดต่อ โดยต้องเขียนโปรแกรมย่อย เพื่อสร้างไทม์สล็อตของฟังก์ชั่นต่างๆ ดังแสดงรายละเอียดของโฟลวชาร์ตและโปรแกรม

อนึ่งในการติดต่อกับ DS1820 นี้เพียง 2 ตัว จึงไม่จำเป็นต้องใช้คำสั่งเพื่อติดต่อกับหน่วยความจำรอมภายใน DS1820 นั้นคือ จะติดต่อแบบสคิปรอม (Skip ROM)




รูปที่ 35 โฟลวชาร์ตของ ไอซี DS1820 กับไมโครคอนโทรลเลอร์ MCS 51



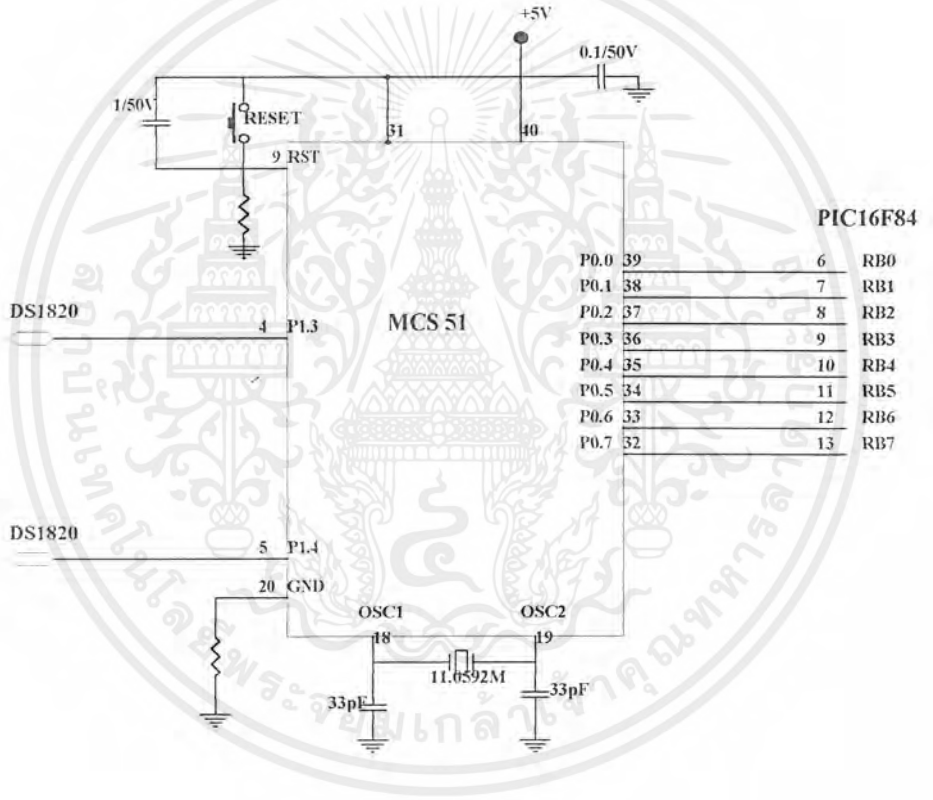

รูปที่ 36 เป็นวงจรการเชื่อมต่อไอซี DS1820 กับไมโครคอนโทรลเลอร์ MCS 51

รูปที่ 36 เป็นวงจรการเชื่อมต่อไอซี DS1820 กับไมโครคอนโทรลเลอร์ MCS 51 เมื่อไอซี DS 1820 ตรวจรับอุณหภูมิ จะส่งไปยังภาคมัลติเพล็กซ์ ไอซี DS1820 รับสัญญาณ 2 ช่องสัญญาณ จะมีขาใช้งาน 3 ขาคือ DQ เป็นการเชื่อมต่อระบบบัส ขาต่อไฟเลี้ยง และขาต่อขากราวด์

เมื่อไอซี DS 1820 ตรวจจับอุณหภูมิ จะนำไปเก็บไว้ในสแตรตช์แพด(Scratchpad) ที่ไบต์ 0 และไบต์ 1 ไอซี DS1820 สามารถให้รายละเอียดของข้อมูลถึง 16 บิต

**3.2.3 ส่วนภาคมัลติเพล็กซ์**

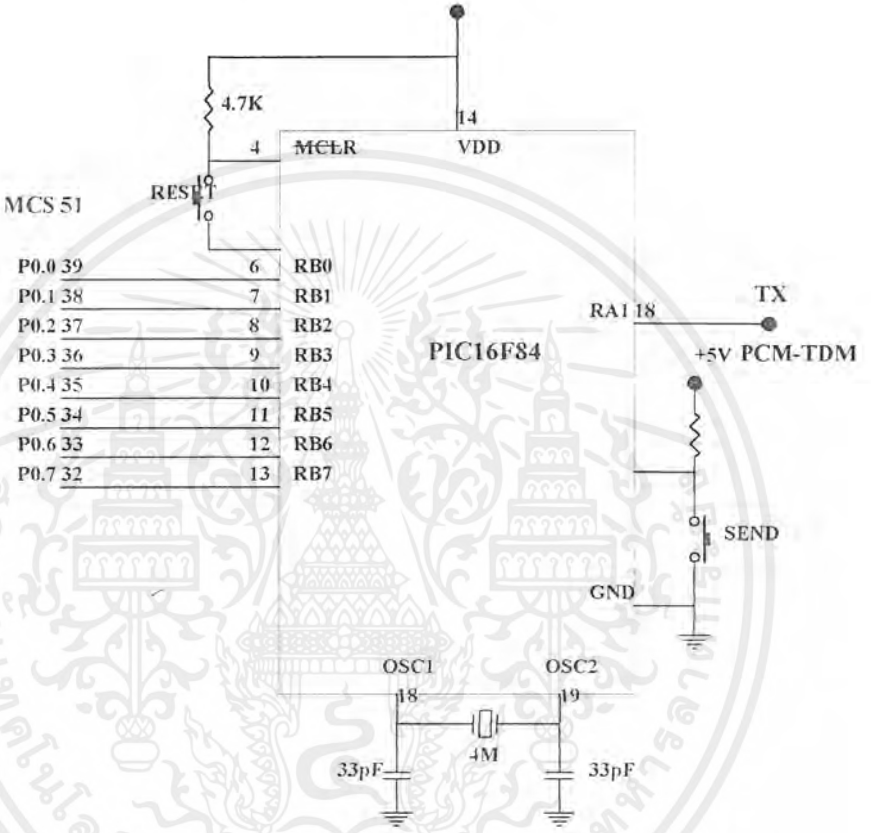
การมัลติเพล็กซ์ ในโครงงานนี้ จะใช้ การมัลติเพล็กซ์ แบบการแบ่งเวลา (TDM) คือการส่งข่าวสาร จากสัญญาณ อินพุต จาก ไอซี DS1820 สัญญาณ 2 ช่องสัญญาณ ออกไปทางการสื่อสารอันเดียว ที่เวลาที่ต่างกัน ในโครงงานนี้ จะใช้ไมโครคอนโทรลเลอร์ MCS 51 เป็นตัวมัลติเพล็กซ์ แล้วจะให้สัญญาณข้อมูล เป็นแบบขนาน 8 บิต




รูปที่ 37 ภาคมัลติเพล็กซ์

3.2.4 ส่วนของการแปลงสัญญาณข้อมูลแบบขนาน เป็น สัญญาณข้อมูลแบบอนุกรม

ในการส่งสัญญาณแบบ TDM เป็นสัญญาณข้อมูล ขนาด 8 บิต แบบขนาน จะทำการแปลง เป็นสัญญาณข้อมูลขนาด 8 บิต อนุกรม เราจะใช้ PIC 16F84

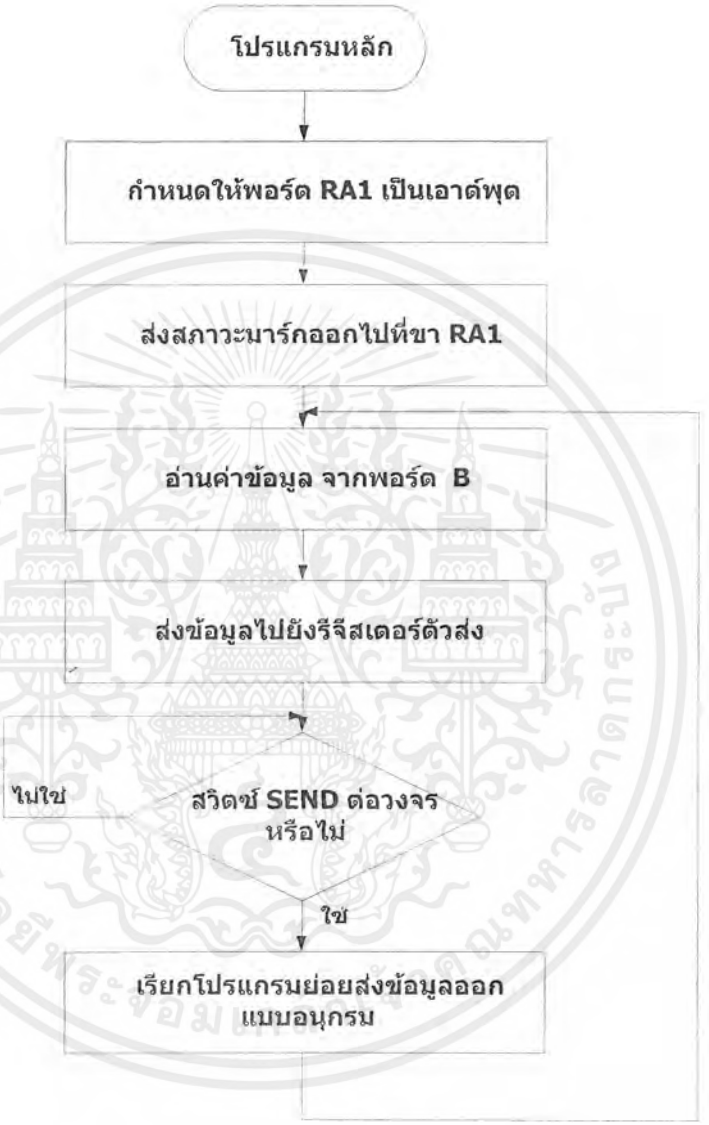



รูปที่ 38 วงจรการเชื่อมต่อของไมโครคอนโทรลเลอร์กับ PIC 16F84

การสื่อสารข้อมูลระหว่าง PIC 16F84 ในลักษณะอนุกรม กำหนดให้ IC1 เป็นตัวส่งข้อมูลออกมาทางขา RA1 ได้รับการกำหนดให้ทำงานในโหมด XT ใช้สัญญาณจากนาฬิกาจากคริสตอล 4 MHz เท่ากัน และใช้โปรแกรมหน่วงเวลาที่มีคาบเท่ากัน

จุดหนึ่งที่ต้องคำนึงถึงก็คือ จังหวะในการรับ และ ส่งข้อมูล จะต้องสอดคล้องกัน ดังนั้นจึงต้องมี การกำหนดอัตราการรับ และส่งข้อมูลที่เรียกว่า อัตราบอด หรือ บอดเรต (Baud rate) มีหน่วยเป็นบิต ต่อ วินาที (bit per second: bps) จากการกำหนดรูปแบบ ข้อมูลในแต่ละบิตมีคาบเวลา 104 ไซเคิล ของสัญญาณนาฬิกาภายในไมโครคอนโทรลเลอร์ เมื่อสัญญาณนาฬิกาภายในมีความถี่เท่ากับ ¼ ของความถี่จากแหล่งภายนอก ซึ่งในที่นี้ก็คือ คริสตอลค่า 4 MHz ดังนั้นความถี่ของสัญญาณนาฬิกาใน PIC 16F84 จึงเท่ากับ 1 MHz คาบต่อ 1 ไซเคิล จึงเท่ากับ 1

วินาที ดังนั้นคาบเวลาของข้อมูล 1 บิต จึงเท่ากับ 104 ไมโครวินาที สามารถคำนวณอัตรา บอด ได้เท่ากับ 9,615 บิตต่อวินาที โดยประมาณ




รูปที่ 39 โฟลวชาร์ตโปรแกรมแปลงสัญญาณขนานเป็นสัญญาณอนุกรม

3.3 ทางภาครับ

หลักการและแนวคิด

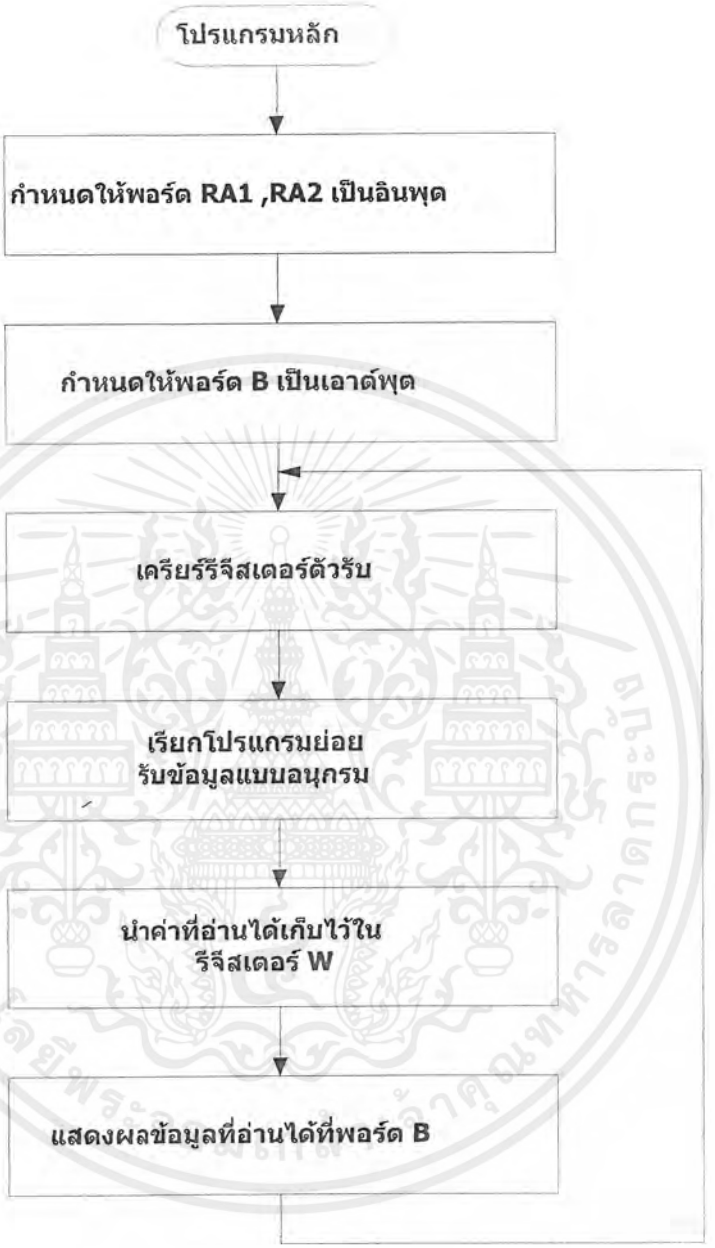
ในการรับสัญญาณมัลติเพล็กซ์ ที่ส่งมาตามสายโทรศัพท์ เมื่อสัญญาณส่งออกมาทางภาคส่ง จะเป็นสัญญาณข้อมูลอนุกรมขนาด 8 บิต ทำการแปลงสัญญาณข้อมูลขนานขนาด 8 บิต โดยการใช้ไอซี PIC 16F84 แล้วส่งเข้าส่วนดีมอดูเลเตอร์ โดยใช้ไมโครคอนโทรลเลอร์เป็นส่วนของดีมัลติเพล็กซ์ แล้วนำไปแสดงผลในส่วนของ LCD แยกสัญญาณเป็น 2 ช่องสัญญาณ


รูปที่ 40 แสดงรายละเอียดของภาครับ

3.3.1 ส่วนของการแปลงสัญญาณข้อมูลแบบอนุกรมเป็นสัญญาณข้อมูลแบบขนาน

เมื่อพิจารณาจากรูปที่ 40 ในสภาวะที่ยังไม่มีการรับแลการส่งสัญญาณข้อมูล สถานะลอจิกที่ขา RA1 ของ ไอซี PIC 16F84 จะอยู่ในสภาวะ "1" ค้างอยู่ตลอดเวลา เมื่อขาที่ใช้ส่งข้อมูลซึ่งก็คือขา RA1 เป็น "0" ต่อจากสถานะมาร์กยาวนานถึง 104 ไซเคิล ซึ่งสัญญาณภายใน จะเรียกสิ่งนี้ว่าบิตเริ่มต้น เพื่อแจ้งให้ตัวรับทำการเตรียมพร้อมในการรับข้อมูล

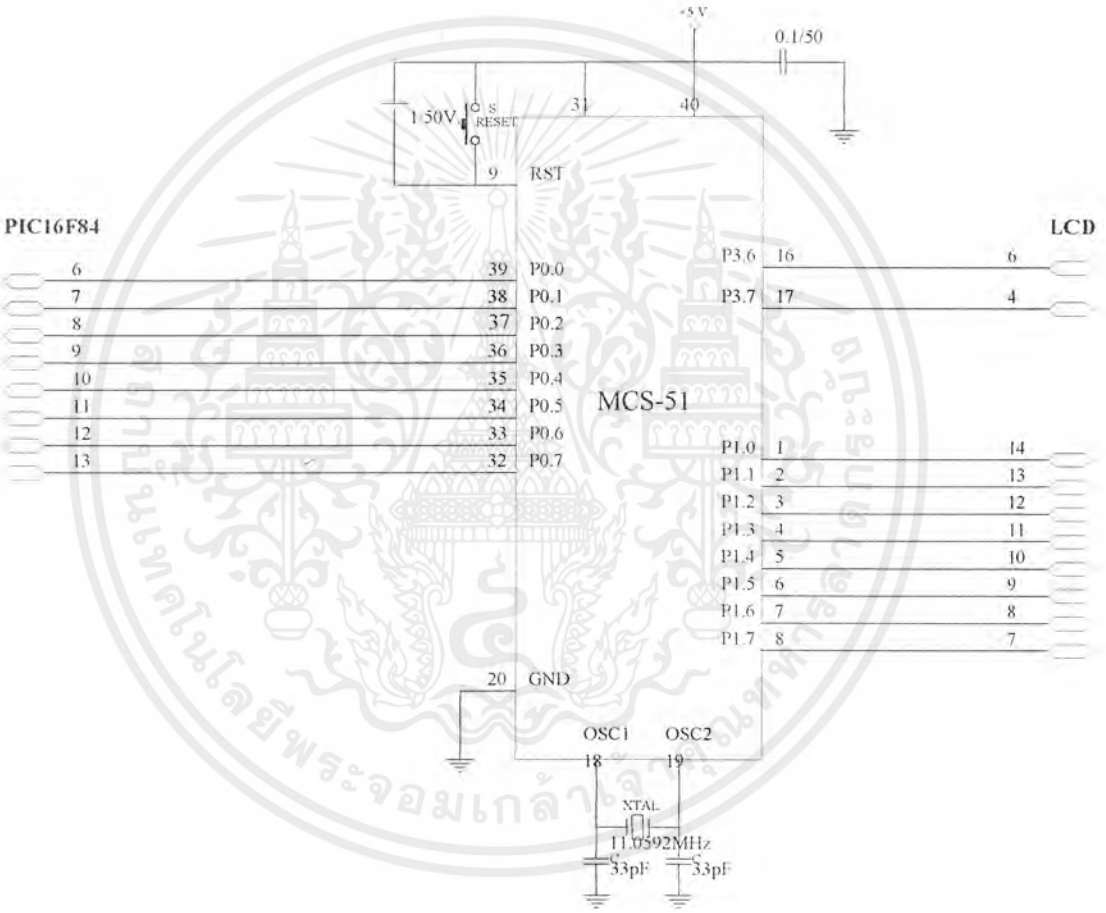
ในการรับสัญญาณข้อมูลของ PIC 16F84 จะรับสัญญาณข้อมูลจากภาคส่งเป็นแบบสัญญาณข้อมูลอนุกรมขนาด 8 บิต แปลงเป็นสัญญาณขนานขนาด 8 บิต จะเขียนได้เป็นไดอะแกรมได้ดังนี้




รูปที่ 41 โฟลวชาร์ตการแปลงสัญญาณข้อมูลแบบอนุกรมเป็นสัญญาณข้อมูลขนาน

### 3.3.2 ส่วนภาคดีมัลติเพล็กซ์

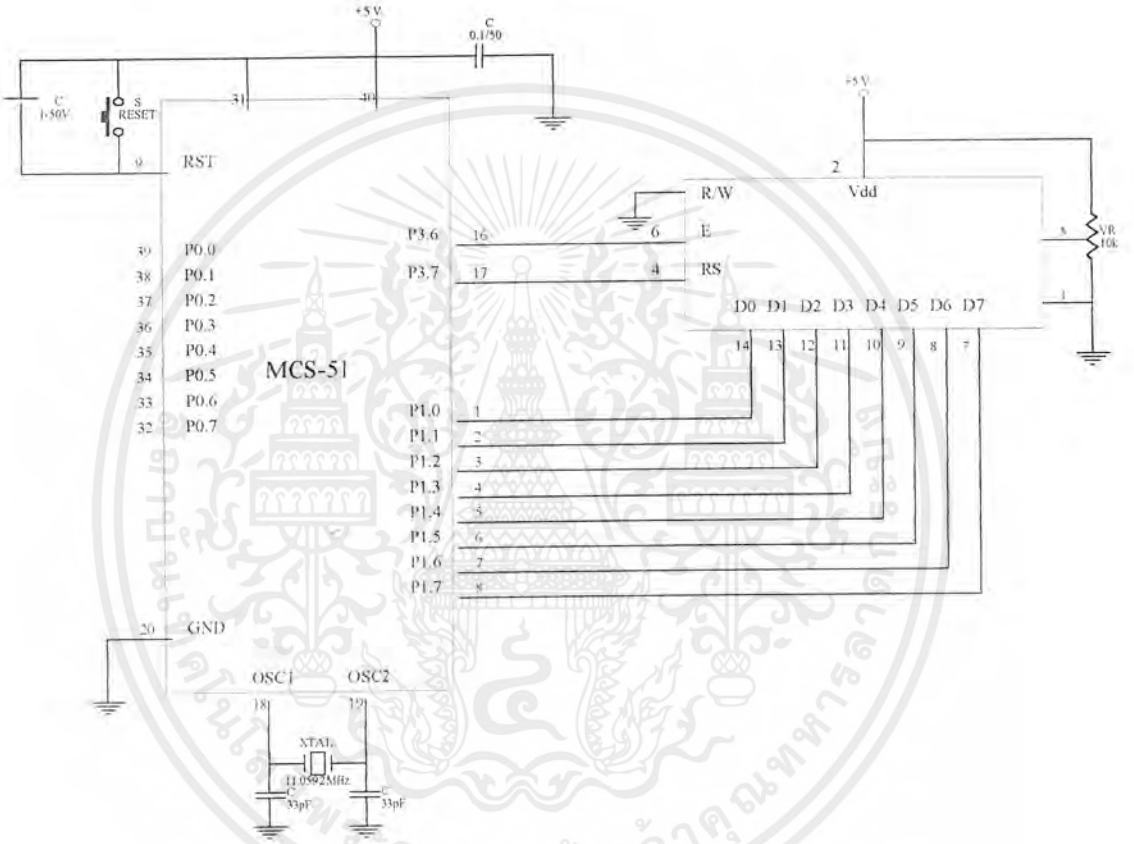
การดีมัลติเพล็กซ์คือ การคืนรูปสัญญาณให้เหมือนกับสัญญาณที่อินพุตของภาคส่ง ในโครงงานนี้ จะใช้ไมโครคอนโทรลเลอร์เป็นตัวมัลติเพล็กซ์สัญญาณ




รูปที่ 42 การดีมัลติเพล็กซ์

3.4 ภาคการแสดงผลผ่านทาง LCD

ในการแสดงผลทาง LCD นั้น จะประกอบไปด้วยส่วน 2 ส่วนใหญ่คือ ส่วนของฮาร์ดแวร์ (Hardware) และส่วนของซอร์ฟแวร์ (Software)




รูปที่ 43 บล็อกไดอะแกรมของส่วนนำสัญญาณเข้า LCD

จากบล็อกไดอะแกรม จะต้องมีการกำหนดโหมดการทำงานให้แก่โมดูล LCD ก่อนจากนั้น จากนั้นจึงค่อยส่งข้อมูล (data) ที่ต้องการแสดงผล เนื่องจากบัสข้อมูลของ LCD มี 8 เส้น คือ D0-D7 และเป็นทางผ่านของทั้งคำสั่งและข้อมูล ดังนั้นในการส่งคำสั่งและข้อมูล จึงต้องอาศัยการกำหนดสัญญาณลอจิกที่ขา RS หากที่ขาRS รับลอจิก “0” หมายความว่าข้อมูลที่ป้อนให้โมดูลที่ให้ในขณะนั้นเป็นคำสั่ง ถ้าหาก RS ได้ลอจิก “I” ข้อมูลที่ป้อนให้นั้นเป็นการนำไปใช้แสดงผล

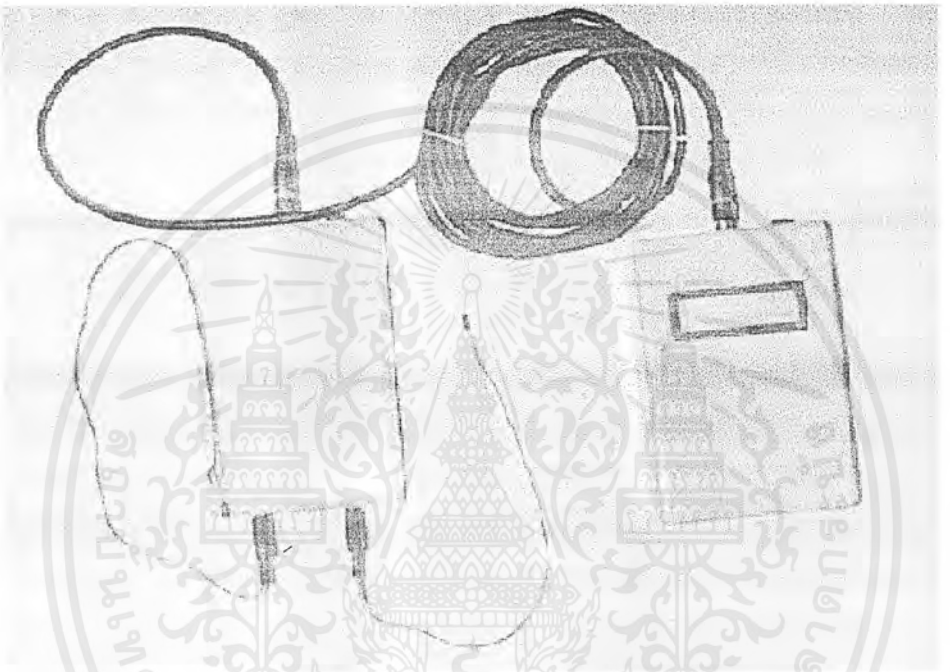
ในเมื่อเขียนหรืออ่านข้อมูลใน CGRAM และDDRAM เริ่มต้นต้องกำหนดแอสเดรสที่ต้องอ่านหรือต้องเขียนก่อน โดยใช้คำสั่งเลือกแอสเดรส จากนั้นกำหนดให้ขา RS เป็น "1" เพื่อแจ้งให้ตัวควบคุมภายในโมดูล LCD ทราบว่าข้อมูลที่ปรากฏออกไปเป็นข้อมูลปกติไม่ใช้คำสั่ง

ในการใช้งานโมดูล LCD โปรแกรมหน่วงเวลารอให้โมดูล LCD พร้อมทำงาน เมื่อเริ่มจ่ายไฟ ประมาณ 10 มิลลิวินาที จากนั้นกำหนดลอจิก ให้แก่ขา RS หน่วงเวลาอีกประมาณ 2 มิลลิวินาที เพื่อให้คอนโทรลเลอร์ใน LCD เพื่อให้รู้ว่าเป็นระหัสคำสั่งหรือเป็นข้อมูลที่ต้องการแสดงผล จากนั้นก็ส่งข้อมูลมาที่บัสข้อมูล D0-D7 ขั้นตอนต่อไปจะเป็นการส่งสัญญาณพัลส์ที่ขา E เพื่ออีนาเบิลโมดูล LCD ให้รับข้อมูลจากบัสข้อมูล พัลส์ที่ป้อนเข้าต้องเป็นพัลส์ขอบขาขึ้นด้วย จากนั้นทำการหน่วงเวลาอีก 2 มิลลิวินาที

# บทที่ 4

## การทดลองและผลการทดลอง

ในการทดลองนี้เราแบ่งการทดลองในการวัดที่อุณหภูมิต่างๆกัน โดยแบ่งเป็นที่ภาคส่งสามารถรับค่าอุณหภูมิได้ 2 ช่องสัญญาณ ส่งผ่านสายส่งเพียงชุดเดียว แล้วไปแสดงค่าอุณหภูมิยังภาครับที่จอแสดงผล LCD แสดงค่าอุณหภูมิที่ตรวจจับได้ทั้ง 2 ช่องสัญญาณ

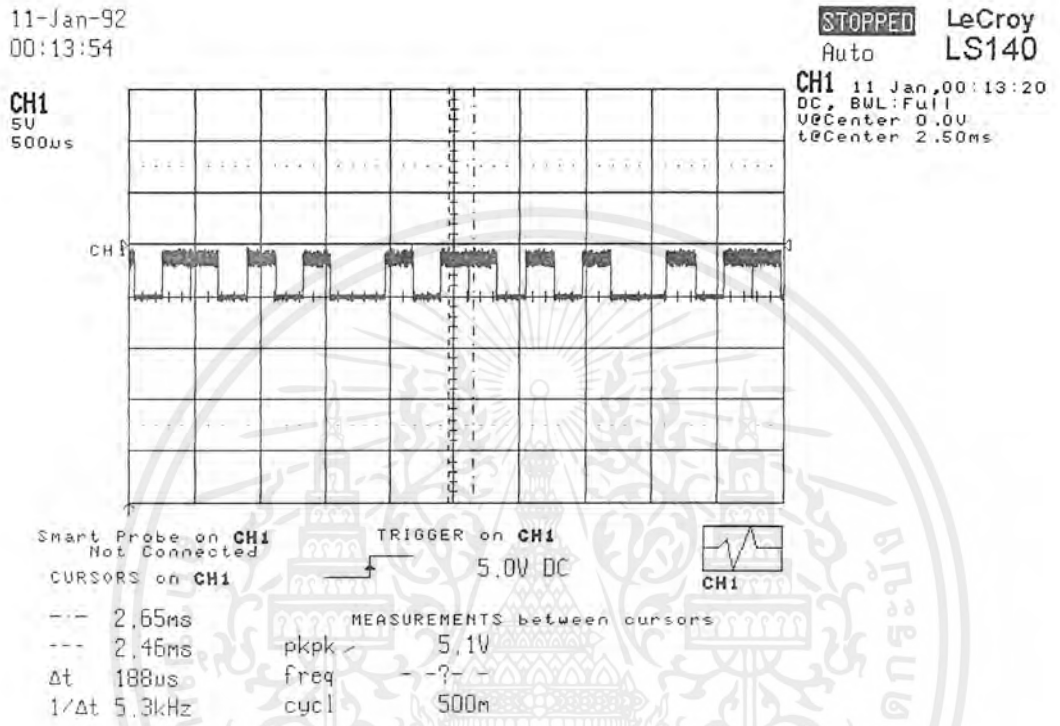


รูปที่ 45 รูปชิ้นงานจริงขณะทำการทดลอง

**ผลการทดลอง**

4.1 ผลการทดลองครั้งที่ I

การทดลองครั้งที่ 1 นี้ จะทำการตรวจจับค่าอุณหภูมิ 2 ค่า คือที่ CH 1 จะวางอยู่ที่อุณหภูมิห้องที่ 25 องศาเซลเซียส และที่ CH 2 ใช้นิ้วมือจับไว้ อุณหภูมิจะอยู่ที่ 30.5 องศาเซลเซียส แล้วทำการวัดสัญญาณที่ Output ของภาคส่ง จะได้รูปของ Pulse Data ที่อยู่ในลักษณะของข้อมูลที่เป็น Binary Bits




รูปที่ 46 รูปสัญญาณ Binary Bits ที่ Output ของภาคส่ง

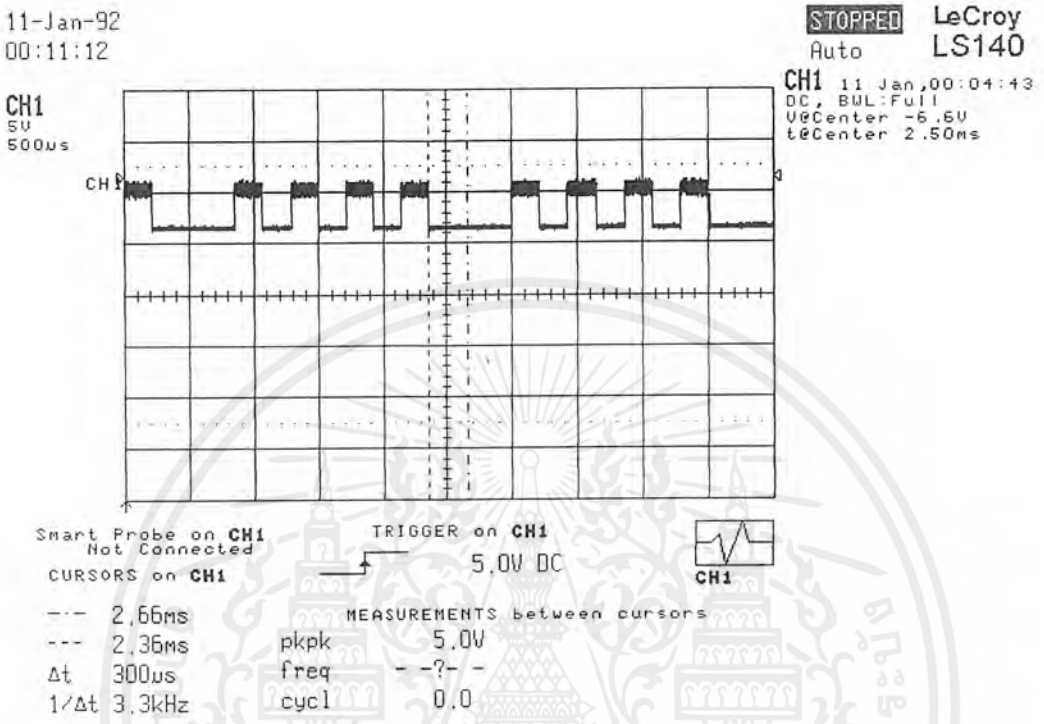
สัญญาณที่ได้จะมีลักษณะเป็น Pulse Data ที่อยู่ในรูปของ Binary Bits โดยใน Binary Bits นี้จะประกอบด้วย บิตข้อมูลที่มารวมกันกับบิตซิงโครไนซ์ เพื่อที่ภาครับจะนำบิตซิงโครไนซ์มาใช้ในกระบวนการดีมัลติเพล็กซ์ เพื่อนำบิตข้อมูลมาทำการแสดงผลของค่าอุณหภูมิทั้ง 2 CH. ที่จอแสดงผล LCD


รูปที่ 47 จอแสดงผล LCD ที่ภาครับ

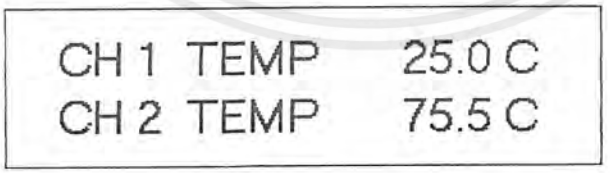
LCD จะทำการรับข้อมูลที่ผ่านกระบวนการดีมัลติเพล็กซ์ แล้วนำมาแสดงผล โดยแสดงผลได้พร้อมกันทั้ง 2 ช่องสัญญาณอย่างเป็นอิสระ

4.2 ผลการทดลองครั้งที่ 2

การทดลองครั้งที่ 2 นี้ จะทำการตรวจจับค่าอุณหภูมิ 2 ค่า คือที่ CH 1 จะวางอยู่ที่อุณหภูมิห้องที่ 25 องศาเซลเซียส และที่ CH 2 ใช้หัวแร้งนำไปวางอังไว้ระยะประมาณ 5 เซนติเมตร สังเกตุอุณหภูมิที่เปลี่ยนแปลงโดยดูจากจอแสดงผล LCD ที่ภาครับ จะเห็นว่าค่าอุณหภูมิที่ CH 2 จะเปลี่ยนแปลงได้โดยไม่มีผลต่อค่าอุณหภูมิห้อง 25 องศาเซลเซียส ที่ CH 1 อุณหภูมิที่ CH 2 จะขึ้นไปจนคงที่ที่อุณหภูมิ 75.5 องศาเซลเซียส ทำการวัดสัญญาณ Output ที่ภาคส่ง




รูปที่ 48รูปสัญญาณ Binary Bits ที่ Output ของภาคส่ง การทดลองที่2




รูปที่ 48 จอแสดงผล LCD ที่ภาครับ การทดลองที่ 2

# บทที่ 5

## บทสรุปและวิจารณ์

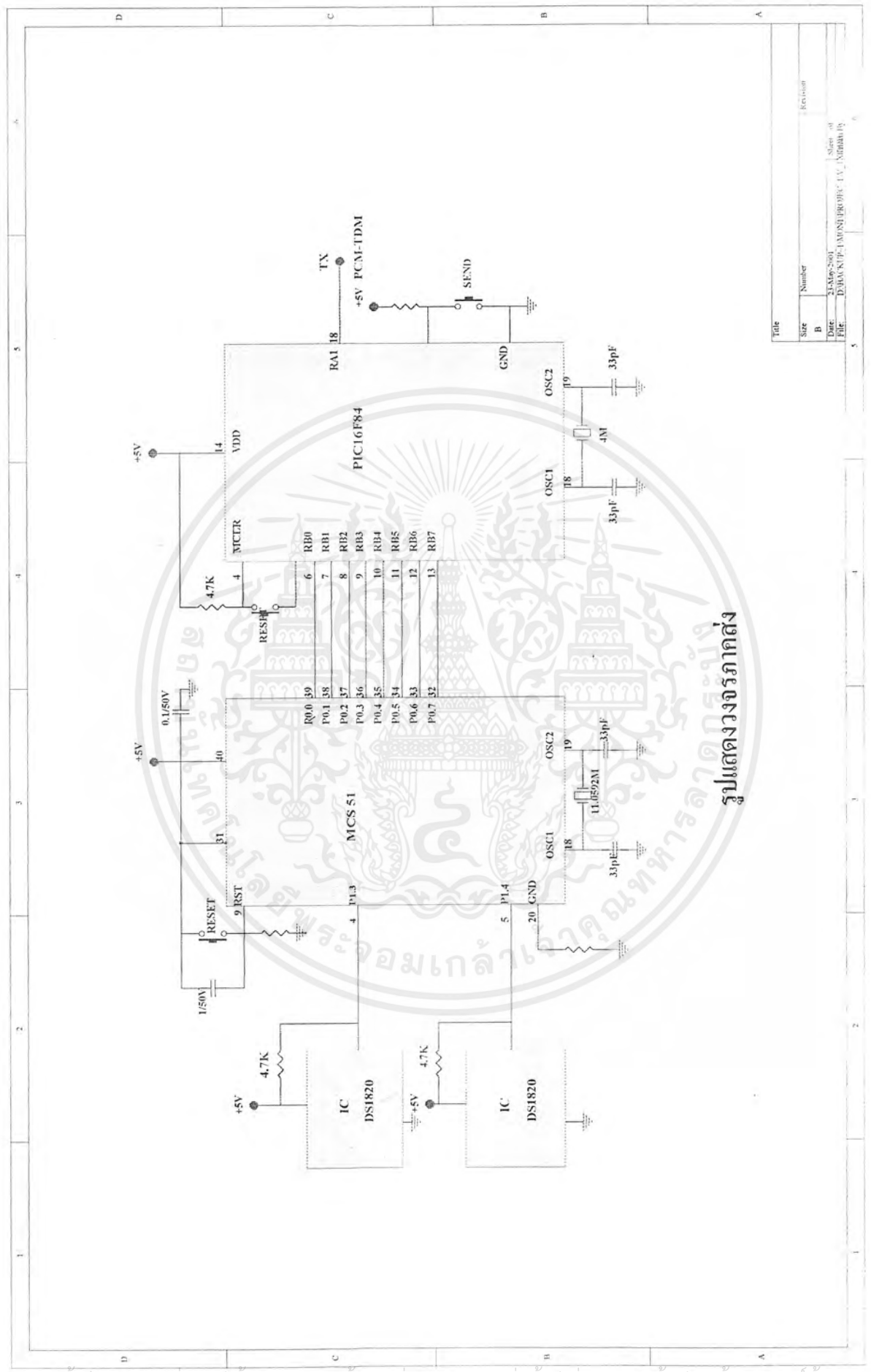
การตรวจจับอุณหภูมิในสถานที่แตกต่างกัน 2 แห่ง แล้วส่งสัญญาณไปแสดงผลอีกสถานที่อื่นที่ห่างไกลออกไป ในโครงงานนี้จะใช้การส่งในระบบ TDM ผ่านสายโคแอ็คเชียลไปแสดงผลที่ LCD ในการส่งไปแสดงผลที่ระยะไกลต้องทำ Line Coding แต่ในโครงงานนี้จะส่งได้ในระยะ 100 เมตร ดังนั้นในการส่งจึงจะไม่ได้ไกลเท่าที่ควร ในการวัดอุณหภูมิ ไม่ให้วัดที่อุณหภูมิที่สูงไม่เกิน 127 องศาเซลเซียล และติดลบไม่เกิน50 องศาเซลเซียล เพราะตามปกติแล้วในการวัดจะใช้ไม่ถึงตามที่จำกัดไว้ ในการจ่ายไฟให้กับโครงงานนี้ ต้องไม่เกิน 30 V เพราะทำให้วงจรภายในเสียหายได้ ดังนั้นต้องทำการปรับแหล่งจ่ายไฟให้อยู่ที่ไม่เกิน 12 V และไม่ต่ำกว่า 5 V จะทำให้การใช้งานมีประสิทธิภาพมาก

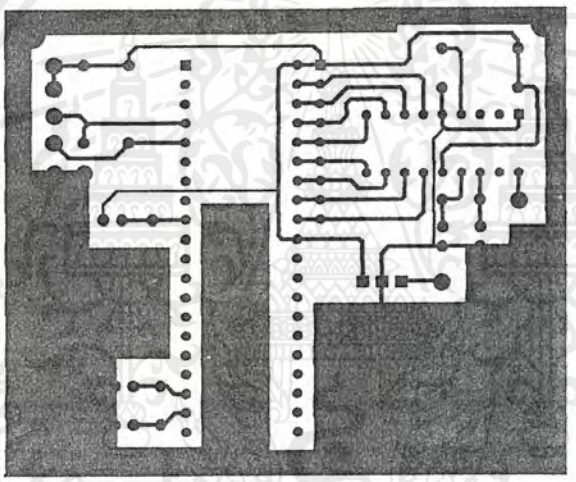
ในส่วนของโครงงานนี้ สามารถที่จะตรวจวัดอุณหภูมิ แล้วส่งไปตามสายโคแอ็คเชียล และแปลงสัญญาณที่ได้จากภาครับให้ไปแสดงผลที่ LCD เพื่อที่จะนำไปใช้งานในแหล่งที่เป็นอันตรายได้ และสามารถนำ ไปใช้ประโยชน์อย่างอื่นได้อีกมากมาย ขึ้นอยู่กับการจัดการกับข้อมูลและการเขียนโปรแกรม ซึ่งสามารถนำไปพัฒนาและปรับปรุงให้เหมาะสมกับการใช้งานต่อไป

# บรรณานุกรม

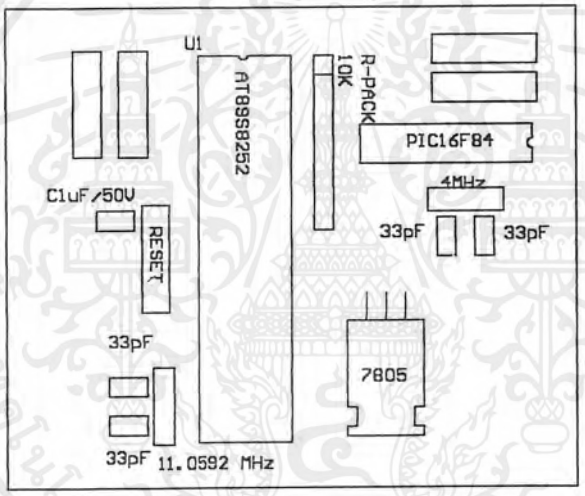

1. Pearson, J. E. Basic Communication Theory. Heartfordshire : Prentice-Hall International (UK) Ltd., 1992
2. กฤษดา ใจเย็น, ชัยวัฒน์ ลิ้มพรจิตรวิไล, เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ PIC16F84, กรุงเทพมหานคร, บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์
3. วรพจน์ กรแก้ววัฒนกุล, ชัยวัฒน์ ลิ้มพรจิตรวิไล, เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51, กรุงเทพมหานคร, บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์

# ภาคผนวก ก

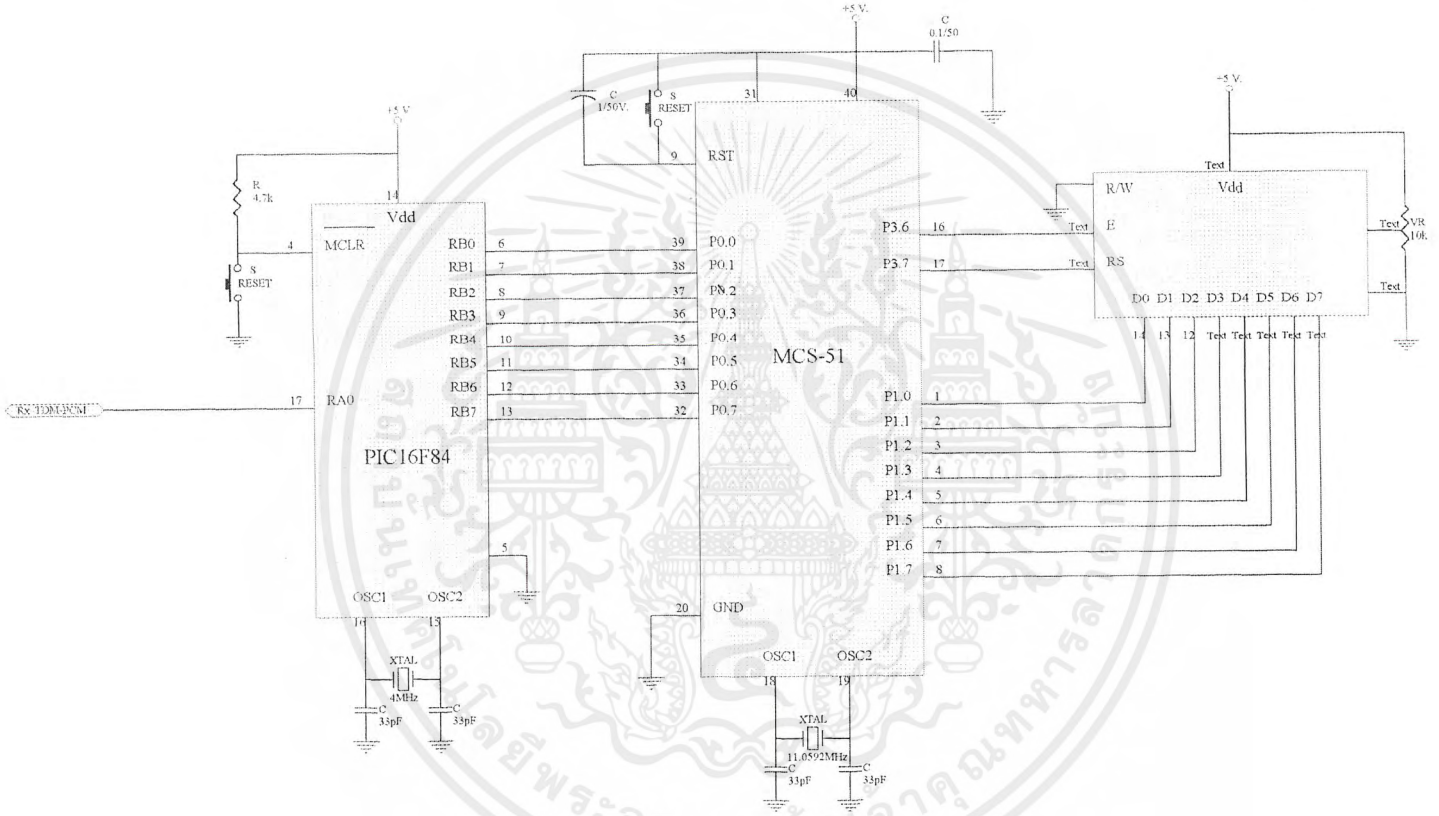




รูปแสดงลายแผ่นวงจรภาคส่ง

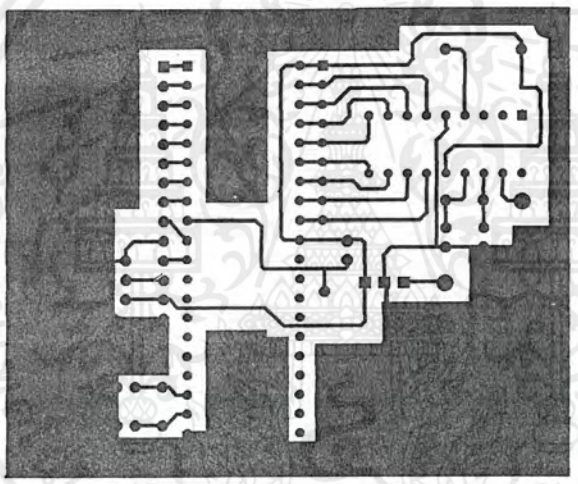



รูปแสดงการจัดวางอุปกรณ์ภาคส่ง

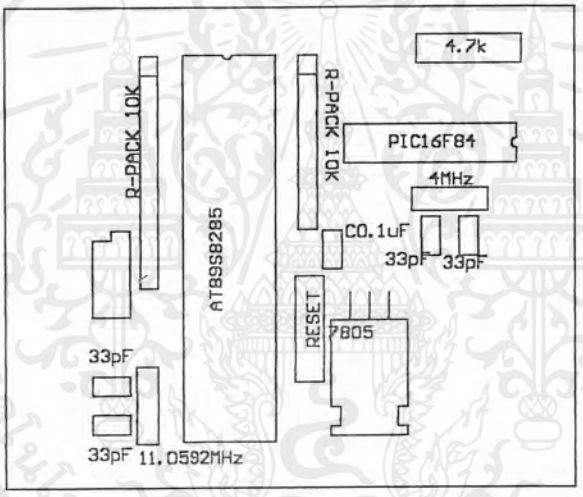



# รูปแสดงวงจรภาครับ

| | | |
|---|---|---|
| Title | | |
| Size: B | Number | Revision |
| Date: 23-May-2001 | | Sheet of |
| File: D:\BACKUP~1\MONITV_TX.DDB | | Drawn By: |



รูปแสดงลายแผ่นวงจรภาครับ




รูปแสดงการจัดวางอุปกรณ์ภาครับ

# ภาคผนวก ข

```
**********TX**********
;-------------------------------------------------------------------
; Define Port&Pin Name
;-------------------------------------------------------------------
ONEWIRE         BIT   P1.4    ; 1-Wire Interface (DS1820 Temp. Sensor)
ONEWIRE1        BIT   P1.3

;-------------------------------------------------------------------
; Define User Register
;-------------------------------------------------------------------
FLAG            EQU   02FH     ; User FLAG
BUSY            BIT   FLAG.0   ; Define BUSY as bit


ONEWIRE_DATA    EQU   032H     ; For keep ONEWIRE Data
ONEWIRE_DATA1   EQU   033H
TEMP            EQU   034H     ; For keep Temp. Data (Temp L only)
TEMP1           EQU   035H
TEMP2           EQU   036H     ; For keep Temp. Data (Temp L only)
TEMP3           EQU   037H
;-------------------------------------------------------------------
; Main Program.
;-------------------------------------------------------------------

                ORG   0000H    ; Reset Vector
                MOV   SP,#256-32
                MOV   P0,#0FFH
                MOV   P2,#0FFH ; Clear Databus
                SETB  ONEWIRE  ; Clear Onewire bus
                SETB  ONEWIRE1
```

```
MAIN:


LOOP:
                ACALL DS1820_RST        ; DS1820 Reset
                ACALL DS1820_PRES       ; DS1820 Presence
                MOV   ONEWIRE_DATA,#0CCH    ; Write Skip ROM
                ACALL DS1820_WR         ;
                MOV   ONEWIRE_DATA,#044H    ; Write Convert
Command
                ACALL DS1820_WR         ;

                SETB  BUSY              ; Set bit BUSY

PRES_CHK_LOOP:  ACALL DS1820_RST        ; DS1820 Reset
                ACALL DS1820_PRES       ; DS1820 Presence
                JB    BUSY,PRES_CHK_LOOP    ; Wait for Busy
                NOP                           ; Delay
                NOP                           ;
                NOP                           ;
                NOP                           ;

                ACALL DS1820_RST        ; DS1820 Reset
                ACALL DS1820_PRES       ; DS1820 Presence
                MOV   ONEWIRE_DATA,#0CCH    ; Write Skip ROM
                ACALL DS1820_WR
                MOV   ONEWIRE_DATA,#0BEH    ; Write Read
Scratchpad Command
                ACALL DS1820_WR
```

```
                ACALL DS1820_RD          ; Read DS1820
                MOV   TEMP,ONEWIRE_DATA  ; Get First Byte as
TEMP (L)

                ACALL DS1820_RST         ; DS1820 Reset
                ACALL DS1820_PRES        ; DS1820 Presence
                MOV   A,TEMP             ; Get Temp. Data
                CLR   C                  ; Clear Carry Flag
                RRC   A
                MOV   P2,A               ; Rotate ACC. to Right with
Carry

                MOV   TEMP1,A
                ANL   A,#11110000B
                ORL   A,#00000001B
                MOV   P0,A
                CALL  DELAY_10MS
                MOV   A,TEMP1
                SWAP  A
                ANL   A,#11110000B
                ORL   A,#00000010B
                MOV   P0,A

                ACALL DELAY_1s


CH2:            ACALL DS1820_RST1        ; DS1820 Reset
                ACALL DS1820_PRES1       ; DS1820 Presence
                MOV   ONEWIRE_DATA1,#0CCH  ; Write Skip ROM
```

```
                    ACALL DS1820_WR1          ;
                    MOV   ONEWIRE_DATA1,#044H     ; Write Convert
Command
                    ACALL DS1820_WR1          ;

                    SETB  BUSY                ; Set bit BUSY

PRES_CHK_LOOP1:     ACALL DS1820_RST1         ; DS1820 Reset
                    ACALL DS1820_PRES1        ; DS1820 Presence
                    JB    BUSY,PRES_CHK_LOOP1     ; Wait for Busy
                    NOP                             ; Delay
                    NOP                             ;
                    NOP                             ;
                    NOP                             ;

                    ACALL DS1820_RST1         ; DS1820 Reset
                    ACALL DS1820_PRES1        ; DS1820 Presence
                    MOV   ONEWIRE_DATA1,#0CCH     ; Write Skip ROM
                    ACALL DS1820_WR1
                    MOV   ONEWIRE_DATA1,#0BEH     ; Write Read
Scratchpad Command
                    ACALL DS1820_WR1

                    ACALL DS1820_RD1          ; Read DS1820
                    MOV   TEMP2,ONEWIRE_DATA1     ; Get First Byte as
TEMP (L)

                    ACALL DS1820_RST1         ; DS1820 Reset
                    ACALL DS1820_PRES1        ; DS1820 Presence
```

```
                MOV   A,TEMP2          ; Get Temp. Data
                CLR   C                ; Clear Carry Flag
                RRC   A
                MOV   P2,A             ; Rotate ACC. to Right with
Carry

                MOV   TEMP3,A
                ANL   A,#11110000B
                ORL   A,#00000011B
                MOV   P0,A
                CALL  DELAY_10MS
                MOV   A,TEMP3
                SWAP  A
                ANL   A,#11110000B
                ORL   A,#00000100B
                MOV   P0,A

                ACALL DELAY_1s         ; Delay
                AJMP  LOOP             ; Jump to loop

;-----------------------------------------------------------------
; DS1820 Data Read
;-----------------------------------------------------------------
DS1820_RD:      MOV   R4,#8            ; Set loop 8 times
                CLR   A                ; Clear ACC.
DS1820_RD_LOOP: CLR   ONEWIRE          ; Clear ONEWIRE
                NOP                    ; Delay
                NOP                    ;
                SETB  ONEWIRE          ; Set ONEWIRE
                NOP                    ; Delay
```

```
                NOP                             ;
                NOP                             ;
                NOP                             ;
                MOV   C,ONEWIRE          ; Get ONEWIRE to Carry
Flag
                ACALLONEWIRE_DELAY       ; Delay 75 us
                RRC   A                  ;
                DJNZ  R4,DS1820_RD_LOOP  ; Do until 8 times
                MOV   ONEWIRE_DATA,A     ; Move ACC. to
ONEWIRE_DATA
                RET                      ; Return

DS1820_RD1:     MOV   R4,#8              ; Set loop 8 times
                CLR   A                  ; Clear ACC.
DS1820_RD_LOOP1:CLR   ONEWIRE1           ; Clear ONEWIRE
                NOP                                 ; Delay
                NOP                                 ;
                SETB  ONEWIRE1           ; Set ONEWIRE
                NOP                                 ; Delay
                NOP                                 ;
                NOP                                 ;
                NOP                                 ;
                MOV   C,ONEWIRE1         ; Get ONEWIRE to Carry
Flag
                ACALLONEWIRE_DELAY               ; Delay 75 us
                RRC   A                  ;
                DJNZ  R4,DS1820_RD_LOOP1         ; Do until 8 times
                MOV   ONEWIRE_DATA1,A    ; Move ACC. to
ONEWIRE_DATA
                RET                              ; Return
```

```
;-------------------------------------------------------------------------
; DS1820 Data Write
;-------------------------------------------------------------------------
DS1820_WR:          MOV   R4,#8                 ; Set loop 8 times
                    MOV   A,ONEWIRE_DATA        ; Get
ONEWIRE_DATA
DS1820_WR_LOOP:     RRC   A                     ; Rotate ACC. with
Carry Flag
                    JNC   DS1820_WR_L           ; Carry Flag was set ?
                    CLR   ONEWIRE               ; Set => TX high
                    NOP                               ; Delay
                    NOP                               ;
                    NOP                               ;
                    NOP                               ;
                    SETB  ONEWIRE               ; Set ONEWIRE
                    ACALL ONEWIRE_DELAY         ; Delay 75 us
                    AJMP  DS1820_WR_NX          ; Jump to next write
DS1820_WR_L:        CLR   ONEWIRE               ; Clear => TX Low
                    ACALL ONEWIRE_DELAY         ; Delay 75 us
                    SETB  ONEWIRE               ; Set ONEWIRE
                    NOP                               ; Delay
                    NOP                               ;
                    NOP                               ;
                    NOP                               ;
DS1820_WR_NX:       DJNZ  R4,DS1820_WR_LOOP     ; Do until 8 times
                    RET                               ; Return
```

```
DS1820_WR1:             MOV   R4,#8              ; Set loop 8 times
                        MOV   A,ONEWIRE_DATA1    ; Get ONEWIRE_DATA
DS1820_WR_LOOP1:        RRC   A                  ; Rotate ACC. with Carry
Flag
                        JNC   DS1820_WR_L1       ; Carry Flag was set ?
                        CLR   ONEWIRE1           ; Set => TX high
                        NOP                              ; Delay
                        NOP                              ;
                        NOP                              ;
                        NOP                              ;
                        SETB  ONEWIRE1           ; Set ONEWIRE
                        ACALL ONEWIRE_DELAY              ; Delay 75 us
                        AJMP  DS1820_WR_NX1      ; Jump to next write
DS1820_WR_L1:           CLR   ONEWIRE1           ; Clear => TX Low
                        ACALL ONEWIRE_DELAY              ; Delay 75 us
                        SETB  ONEWIRE1           ; Set ONEWIRE
                        NOP                              ; Delay
                        NOP                              ;
                        NOP                              ;
                        NOP                              ;
DS1820_WR_NX1:          DJNZ  R4,DS1820_WR_LOOP1         ; Do until 8 times
                        RET                              ; Return


;-------------------------------------------------------------------------
; DS1820 Reset
;-------------------------------------------------------------------------
DS1820_RST:             CLR   ONEWIRE                    ; Clear ONEWIRE
                        ACALL DELAY_1ms          ; Delay
                        SETB  ONEWIRE            ; Set ONEWIRE
```

```
                    MOV   R4,#8              ; Delay
                    DJNZ  R4,$               ;
                    RET                      ; Return


DS1820_RST1:        CLR   ONEWIRE1           ; Clear ONEWIRE
                    ACALL DELAY_1ms          ; Delay
                    SETB  ONEWIRE1           ; Set ONEWIRE
                    MOV   R4,#8              ; Delay
                    DJNZ  R4,$               ;
                    RET                      ; Return

;-------------------------------------------------------------------------
; DS1820 Receive Presence Pulse
;-------------------------------------------------------------------------
DS1820_PRES:        MOV   R4,#8              ; Set Loop wait 1
DS1820_PRES_1:      MOV   R3,#0              ; Set Loop wait 2
DS1820_PRES_2:      JNB   ONEWIRE,DS1820_PRES_3          ; Check
ONEWIRE was clear?
                    DJNZ  R3,DS1820_PRES_2   ; Wait loop check 2
                    DJNZ  R4,DS1820_PRES_1   ; Wait loop check 1
                    RET                              ; Return
DS1820_PRES_3:      JNB   ONEWIRE,$                  ; Wait until
ONEWIRE set
                    MOV   R4,#8              ; Delay
                    DJNZ  R4,$               ;
                    CLR   BUSY               ; Clear BUSY Flag
                    RET                      ; Return
```

```
DS1820_PRES1:       MOV   R4,#8                    ; Set Loop wait 1
DS1820_PRES_11:     MOV   R3,#0                    ; Set Loop wait 2
DS1820_PRES_21:     JNB   ONEWIRE,DS1820_PRES_31   ; Check ONEWIRE
was clear?
                    DJNZ  R3,DS1820_PRES_21        ; Wait loop check 2
                    DJNZ  R4,DS1820_PRES_11        ; Wait loop check 1
                    RET                            ; Return
DS1820_PRES_31:     JNB   ONEWIRE1,$               ; Wait until ONEWIRE set
                    MOV   R4,#8                    ; Delay
                    DJNZ  R4,$                     ;
                    CLR   BUSY                     ; Clear BUSY Flag
                    RET                            ; Return

;---------------------------------------------------------------------
; Dummy Delay time ONEWIRE_DELAY, LCD_DELAY, 50u, 100u, 1m, 10m, 100m, 1s
;---------------------------------------------------------------------
ONEWIRE_DELAY:      MOV         R6,#012H       ; Each loop = 75 us
ONEWIRE_DELAY_1: NOP
                    NOP
                    DJNZ  R6,ONEWIRE_DELAY_1
                    RET

LCD_DELAY:          MOV         R7,#002        ; Do 2 times
LCD_DELAY_1:        MOV         R6,#0E6H       ; Each loop = 1 ms
LCD_DELAY_2:        NOP
                    NOP
                    DJNZ  R6,LCD_DELAY_2
                    DJNZ  R7,LCD_DELAY_1
                    RET
```

```
DELAY_50us:     MOV     R6,#00CH    ; Each loop = 50 us
DELAY_50us_1:   NOP
                NOP
                DJNZ    R6,DELAY_50us_1
                RET

DELAY_100us: MOV    R6,#017H        ; Each loop = 100 us
DELAY_100us_1:  NOP
                NOP
                DJNZ    R6,DELAY_100us_1
                RET

DELAY_1ms:      MOV     R6,#0E6H    ; Each loop = 1 ms
DELAY_1ms_1:    NOP
                NOP
                DJNZ    R6,DELAY_1ms_1
                RET

DELAY_10ms:     MOV     R7,#010     ; Do 10 times
DELAY_10ms_1:   MOV     R6,#0E6H    ; Each loop = 1 ms
DELAY_10ms_2:   NOP
                NOP
                DJNZ    R6,DELAY_10ms_2
                DJNZ    R7,DELAY_10ms_1
                RET

DELAY_100ms:    MOV     R7,#100     ; Do 100 times
DELAY_100ms_1:  MOV     R6,#0E6H    ; Each loop = 1 ms
DELAY_100ms_2:  NOP
                NOP
```

```
                DJNZ   R6,DELAY_100ms_2
                DJNZ   R7,DELAY_100ms_1
                RET

DELAY_1s:       MOV    R5,#50          ; Do 100 times
DELAY_1s_1:     ACALL DELAY_10ms
                DJNZ   R5,DELAY_1s_1
                RET
                END
```

```
;**********************************ภาคส่ง**********************************;
;************************** ****list          p=16f84*****************************
;-----------------------------------Config Define-----------------------------------;
CP_ON           equ     0x000f
CP_OFF          equ     0x3fff
PWT_ON          equ     0x3fff
PWT_OFF         equ     0x3ff7
WDT_ON          equ     0x3fff
WDT_OFF         equ     0x3ffb
LP_OSC          equ     0x3ffc
XT_OSC          equ     0x3ffd
HS_OSC          equ     0x3ffe
RC_OSC          equ     0x3fff
;-------------------------------------------------------------------------------------;
;               Code Protect OFF
;               Power Up Timer ON
;               Watch Dog Timer OFF
;               XT oscilator
                _config CP_OFF&PWT_OFF&XT_OSC

;-----------------------------Byte Define-------------------------------------;
STATUS          equ     0x03
PORTA           equ     0x05
PORTB           equ     0x06
SEND            equ     0x0c
COUNT           equ     0x0d
COUNT1          equ     0x0e
```

```
;----------------------------Bit Define----------------------------------------------;
PRO             equ     5
C               equ     0
W               equ     0
F               equ     1
                Org     0x000
Start           bsf     STATUS,RPO
                Movlw   b'00000100'
                Movwf   PORTA
                Movlw   b'11111111'
                Movwf   PORTB
                Bcf     STATUS,RPO
;----------------------MAIN-----------------------------------------;
                bsf     PORTA,1
Wait            movf    PORTB,W
                Movwf   SEND
                Btfsc   PORTA,2
                Goto    Wait
                Call    Ser_data
                Goto    Wait
;----------------------Send Serial Data Subroutine----------;
Ser_data
                Movlw   0x08
                Movwf   COUNT
                Bcf     PORTA,1
                Call    Delay96
Shft_b          rlf     SEND,F
                Bcf     PORTA,1
                Btfsc   STATUS,C
                Bsf     PORT,1
```

```
                    Call        Delay96
                    Decfsz      COUNT,F
                    Goto        Shft_b
                    Bsf         PORTA,1
                    Call        Delay96
                    Return
;----------------------Buadrate delay-----------------------------------------;
Delay96             movlw       0x21
                    Movwf       COUNT1
Delay1              decfsz      COUNT1,F
                    Goto        Delay1
                    return
                    end
```

```
*************************************************************************
*********************************RX**************************************
*************************************************************************
LCD_EN          EQU   P3.0
LCD_RS          EQU   P3.1
LCD_ADDR        EQU   030H          ; For keep LCD Address
LCD_DATA        EQU   031H          ; For keep LCD Data
TEMP1           EQU   032H
TEMP2           EQU   033H
BT              EQU   034H

                ORG   0000H
                SJMP  INITIAL
                ORG   0003H
                RETI
                ORG   0013H
                RETI


INITIAL:        MOV   SP,#256-32
                MOV   P0,#0FFH
                MOV   P1,#10000100B
                MOV   P3,#11111111B
                MOV   P2,#11111111B

MAIN:           ACALL  INIT_LCD
                MOV    LCD_ADDR,#000H
                ACALL  SET_ADDR_LCD
                MOV    DPTR,#TITLE_1
                ACALL  WRLINE_LCD
```

```
                MOV   LCD_ADDR,#040H
                ACALL SET_ADDR_LCD
                MOV   DPTR,#TITLE_2
                ACALL WRLINE_LCD
                CALL  DELAY_1S
                CALL  DELAY_1S
                MOV   LCD_ADDR,#000H
                ACALL SET_ADDR_LCD
                MOV   DPTR,#TITLE_3
                ACALL WRLINE_LCD
                MOV   LCD_ADDR,#040H
                ACALL SET_ADDR_LCD
                MOV   DPTR,#TITLE_4
                ACALL WRLINE_LCD

LOOP:           MOV   A,P0
                MOV   BT,A
                ANL   A,#00001111B
                CJNE  A,#00000001B,LOOP
                MOV   A,BT
                ANL   A,#11110000B
                MOV   TEMP1,A
LOOP1:          MOV   A,P0
                MOV   BT,A
                ANL   A,#00001111B
                CJNE  A,#00000010B,LOOP1
                MOV   A,BT
                SWAP  A
                ANL   A,#00001111B
                ORL   A,TEMP1
```

```
                MOV   P2,A
                MOV   TEMP1,A
                MOV   LCD_ADDR,#009H     ; Set Address 40H
                ACALLSET_ADDR_LCD        ;

                MOV   A,TEMP1                    ; Get Temp.
Data
                MOV   LCD_DATA,A         ; Write Temp. in 3
Decimal Digits
                ACALLHEX2LCD                     ;

                MOV   LCD_ADDR,#00DH     ; Set Address 44H
                ACALLSET_ADDR_LCD        ;

                MOV   A,TEMP1            ; Get Temp. Data
                JNB   ACC.0,WRITE_0C1    ; Check LSB was
set?
                MOV   LCD_DATA,#'5'      ; Set => Load '5'
                AJMP  WRITE_NEXT1         Jump to write LCD
WRITE_0C1:      MOV   LCD_DATA,#'0'      ; Not set => Load '0'
WRITE_NEXT1:    ACALLWRCHAR_LCD          ; Write to LCD


LOOP2:          MOV   A,P0
                MOV   BT,A
                ANL   A,#00001111B
                CJNE  A,#00000011B,LOOP2
                MOV   A,BT
                ANL   A,#11110000B
                MOV   TEMP2,A
```

```
LOOP3:          MOV   A,P0
                MOV   BT,A
                ANL   A,#00001111B
                CJNE  A,#00000100B,LOOP3
                MOV   A,BT
                SWAP A
                ANL   A,#00001111B
                ORL   A,TEMP2
                MOV   TEMP2,A

                MOV   LCD_ADDR,#049H     ; Set Address 40H
                ACALLSET_ADDR_LCD        ;

                MOV   A,TEMP2            ; Get Temp. Data

                MOV   LCD_DATA,A         ; Write Temp. in 3
Decimal Digits
                ACALLHEX2LCD             ;

                MOV   LCD_ADDR,#04DH     ; Set Address 44H
                ACALLSET_ADDR_LCD        ;

                MOV   A,TEMP2            ; Get Temp. Data
                JNB   ACC.0,WRITE_0C2    ; Check LSB was
set?
                MOV   LCD_DATA,#'5'      ; Set => Load '5'
                AJMP  WRITE_NEXT2        ; Jump to write LCD
WRITE_0C2:      MOV   LCD_DATA,#'0'      ; Not set => Load '0'
WRITE_NEXT2:    ACALLWRCHAR_LCD          ; Write to LCD
```

```
                    AJMP   LOOP                ; Jump to loop

;-------------------------------------------------------------------
; HEX Code to show LCD
; I/P:          LCD_DATA
;-------------------------------------------------------------------
HEX2LCD:        PUSH   ACC                 ; Push ACC.
                MOV    A,LCD_DATA          ; Get Data
                MOV    B,#100              ;
                DIV    AB                  ; Divide by 100
                ADD    A,#030H             ; Convert to ASCII
                CJNE   A,#030H,HEX2_LCD_NX ; Check x100 = 0 ?
                MOV    A,#' '              ; 0 => Write Space
HEX2_LCD_NX:    MOV    LCD_DATA,A          ;
                ACALL WRCHAR_LCD           ; Write x100
                MOV    A,B                 ;
                MOV    B,#10               ;
                DIV    AB                  ; Divide by 10
                ADD    A,#030H             ; Convert to ASCII
                MOV    LCD_DATA,A          ;
                ACALL WRCHAR_LCD           ; Write Lower HEX
Code
                MOV    A,B                 ; Get Remainder x1
                ADD    A,#030H             ; Convert to ASCII
                MOV    LCD_DATA,A          ;
                ACALL WRCHAR_LCD           ; Write Lower HEX
Code
                POP    ACC                 ; Pop ACC.
```

```
                RET                     ; Return


;*************************************************************************
**********************************
;*              LCD MODULE                                  *

INIT_LCD:   ACALL  DELAY_100ms
            CLR    LCD_RS
            MOV    P1,#00111000B
            ACALL  LCD_CLK
            ACALL  DELAY_10ms
            MOV    P1,#00111000B
            ACALL  LCD_CLK
            ACALL  LCD_OFF
            ACALL  LCD_CLR
            MOV    P1,#00000110B
            ACALL  LCD_CLK
            ACALL  LCD_HOME
LCD_CLR:    CLR    LCD_RS
            MOV    P1,#00000001B
            ACALL  LCD_CLK
            RET
LCD_HOME:   CLR    LCD_RS
            MOV    P1,#00000010B
            ACALL  LCD_CLK
            RET
LCD_OFF:    CLR    LCD_RS
```

```
                MOV   P1,#00001000B
                ACALL LCD_CLK
                RET
LCD_CLK:        SETB  LCD_EN
                ACALL LCD_DELAY
                CLR   LCD_EN
                ACALL LCD_DELAY
                RET
LCD_ON:         CLR   LCD_RS
                MOV   P1,#00001100B
                ACALL LCD_CLK
                RET
LCD_BLINK:      CLR   LCD_RS
                MOV   P1,#00001111B
                ACALL LCD_CLK
                RET
LCD_LSHF:       CLR   LCD_RS
                MOV   P1,#00011000B
                ACALL LCD_CLK
                RET

LCD_RSHF:       CLR   LCD_RS
                MOV   P1,#00011100B
                ACALL LCD_CLK
                RET
SET_ADDR_LCD:   CLR   LCD_RS
                MOV   A,LCD_ADDR
                SETB  ACC.7
                MOV   P1
                ACALL LCD_CLK
```

```
                RET
WRCHAR_LCD:     SETB   LCD_RS
                MOV    P1,LCD_DATA
                ACALL  LCD_CLK
                ACALL  LCD_ON
                RET
WRLINE_LCD:     MOV    R0,#0
WRLINE_LCD_1:   SETB   LCD_RS
                CLR    A
                MOVC   A,@A+DPTR
                MOV    P1,A
                ACALL  LCD_CLK
                INC    DPTR
                INC    R0
                CJNE   R0,#16,WRLINE_LCD_1
                ACALL  LCD_ON
                RET
```

```
;*************************************************************************
;*              DELAY MODULE                     *
;*************************************************************************

DELAY_1ms:   MOV    R6,#0E6H            ; Each loop = 1 ms
DELAY_1ms_1: NOP
             NOP
             DJNZ   R6,DELAY_1ms_1
```

```
                RET
LCD_DELAY:      MOV   7,#002
LCD_DELAY_1:    MOV   6,#0E6H
LCD_DELAY_2:    NOP
                NOP
                DJNZ  R6,LCD_DELAY_2
                DJNZ  R7,LCD_DELAY_1
                RET
DELAY_10ms:     MOV   7,#010
DELAY_10ms_1:   MOV   6,#0E6H
DELAY_10ms_2:   NOP
                NOP
                DJNZ  R6,DELAY_10ms_2
                DJNZ  R7,DELAY_10ms_1
                RET
DELAY_100ms:    MOV   7,#080
DELAY_100ms_1:  MOV   6,#0E6H
DELAY_100ms_2:  NOP
                NOP
DJNZ  R6,DELAY_10ms_2
DJNZ  R7,DELAY_10ms_1
                RET
DELAY_1s:       MOV   5,#50
DELAY_1s_1:     ACALL DELAY_10ms
                DJNZ  R5,DELAY_1s_1
                RET
DELAY_11s:      MOV   5,#050
DELAY_11s_1:    ACALL DELAY_10ms
                DJNZ  R5,DELAY_11s_1
                RET
```

```
:                    0123456789ABCDEF
                TITLE_1:        DB      'PCM-TDM  PROJECT'
                TITLE_2:        DB      ' Ning & Wee  2M '
                TITLE_3:        DB      'CH1_Temp:  . ',0DFH,'C '
                TITLE_4:        DB      'CH2_Temp:  . ',0DFH,'C '
                TITLE_5:        DB      '  Hour   Min '
                                END

;*************************************************************************
;*************************************************************************
```

```
;*********************************ภาครับ*************************************
;*********************************list        p=16f84*****************************
;-----------------------------------Config Define-------------------------------------;
CP_ON                   equ             0x000f
CP_OFF                  equ             0x3fff
PWT_ON                  equ             0x3fff
PWT_OFF                 equ             0x3ff7
WDT_ON                  equ             0x3fff
WDT_OFF                 equ             0x3ffb
LP_OSC                  equ             0x3ffc
XT_OSC                  equ             0x3ffd
HS_OSC                  equ             0x3ffe
RC_OSC                  equ             0x3fff
;-------------------------------------------------------------------------------------;
;               Code Protect OFF
;               Power Up Timer ON
;               Watch Dog Timer OFF
;               XT oscilator
                _config CP_OFF&PWT_OFF&XT_OSC

;-----------------------------Byte Define-------------------------------------------;
STATUS                  equ             0x03
PORTA                   equ             0x05
PORTB                   equ             0x06
SEND                    equ             0x0c
COUNT                   equ             0x0d
COUNT1                  equ             0x0e
```

```
;-------------------------------Bit Define-----------------------------------------------------:
PRO             equ     5
C               equ     0
W               equ     0
F               equ     1
                Org     0x000
                Goto    Start
Start           bsf     STATUS,RPO
                Movlw   b'00000101'
                Movwf   PORTA
                Movlw   b'11111111'
                Clrf    PORTB
                Bcf     STATUS,RPO
;----------------------MAIN-----------------------------------------;
                clrf    PORTB
                clrf    RCV
Again           call    Ser_in
                Movf    RCV
                Movf    RCV,W
                Movwf   PORTB
                Goto    Again
;----------------------Recieve Data Subroutine----------;
Ser_in
                Movlw   0x08
                Movwf   COUNT
S_bit           btfsc   PORTA,0
                Goto    S_bit
                Call    Delay96
                btfsc   PORTA,0
                Goto    S_bit
```

```
Jump            Call        Delay96
                Call        Delay96
                Movf        PORTA,W
                Movwf       TEMP
                Rrf         TEMP,F
                Rrf         RCV,F
                Decfsz      COUNT,F
                Goto        Jump
                Return
;----------------------Buadrate delay-----------------------------------;
Delay96         movlw       0x10
                Movwf       COUNT1
Delay1          decfsz      COUNT1,F
                Goto        Delay1
                return
                end
```

# ภาคผนวก ค

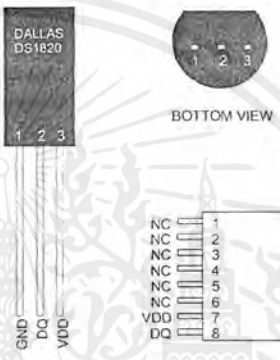
**DALLAS SEMICONDUCTOR**

# DS1820
## 1–Wire™ Digital Thermometer

### FEATURES

- Unique 1–Wire™ interface requires only one port pin for communication
- Multidrop capability simplifies distributed temperature sensing applications
- Requires no external components
- Can be powered from data line
- Zero standby power required
- Measures temperatures from –55°C to +125°C in 0.5°C increments. Fahrenheit equivalent is –67°F to +257°F in 0.9°F increments
- Temperature is read as a 9–bit digital value.
- Converts temperature to digital word in 200 ms (typ.)
- User–definable, nonvolatile temperature alarm settings
- Alarm search command identifies and addresses devices whose temperature is outside of programmed limits (temperature alarm condition)
- Applications include thermostatic controls, industrial systems, consumer products, thermometers, or any thermally sensitive system

### PIN ASSIGNMENT




DS1820
PR35 PACKAGE
See Mech. Drawings Section

DS1820S
16–PIN SSOP
See Mech. Drawings Section

### PIN DESCRIPTION

| | | |
|---|---|---|
| GND | – | Ground |
| DQ | – | Data In/Out |
| $V_{DD}$ | – | Optional $V_{DD}$ |
| NC | – | No Connect |

### DESCRIPTION

The DS1820 Digital Thermometer provides 9–bit temperature readings which indicate the temperature of the device.

Information is sent to/from the DS1820 over a 1–Wire interface, so that only one wire (and ground) needs to be connected from a central microprocessor to a DS1820. Power for reading, writing, and performing temperature conversions can be derived from the data line itself with no need for an external power source.

Because each DS1820 contains a unique silicon serial number, multiple DS1820s can exist on the same 1–Wire bus. This allows for placing temperature sensors in many different places. Applications where this feature is useful include HVAC environmental controls, sensing temperatures inside buildings, equipment or machinery, and in process monitoring and control.

## DETAILED PIN DESCRIPTION

| PIN 16–PIN SSOP | PIN PR35 | SYMBOL | DESCRIPTION |
|---|---|---|---|
| 9 | 1 | GND | **Ground.** |
| 8 | 2 | DQ | **Data Input/Output pin.** For 1–Wire operation: Open drain. (See "Parasite Power" section.) |
| 7 | 3 | $V_{DD}$ | **Optional $V_{DD}$ pin.** See "Parasite Power" section for details of connection. |

DS1820S (16–pin SSOP): All pins not specified in this table are not to be connected.

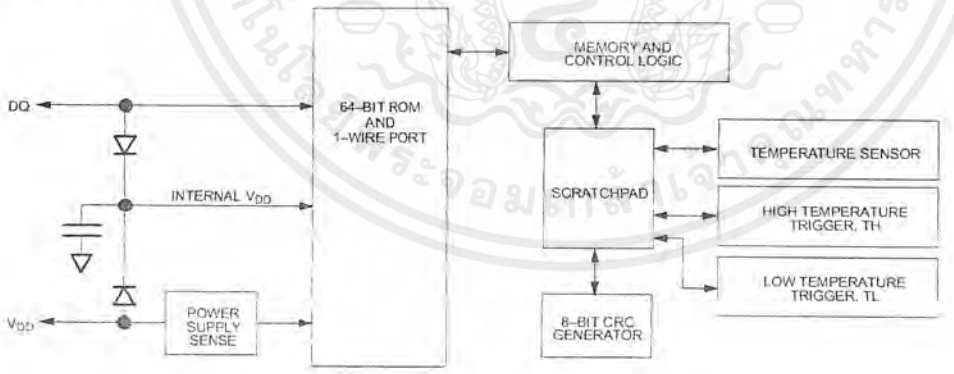
## OVERVIEW

The block diagram of Figure 1 shows the major components of the DS1820. The DS1820 has three main data components: 1) 64–bit lasered ROM, 2) temperature sensor, and 3) nonvolatile temperature alarm triggers TH and TL. The device derives its power from the 1–Wire communication line by storing energy on an internal capacitor during periods of time when the signal line is high and continues to operate off this power source during the low times of the 1–Wire line until it returns high to replenish the parasite (capacitor) supply. As an alternative, the DS1820 may also be powered from an external 5 volts supply.

Communication to the DS1820 is via a 1–Wire port. With the 1–Wire port, the memory and control functions will not be available before the ROM function protocol has been established. The master must first provide one of five ROM function commands: 1) Read ROM, 2) Match ROM, 3) Search ROM, 4) Skip ROM, or 5) Alarm Search. These commands operate on the 64–bit lasered ROM portion of each device and can single out a specific device if many are present on the 1–Wire line as well as indicate to the Bus Master how many and what types of devices are present. After a ROM function sequence has been successfully executed, the memory and control functions are accessible and the master may then provide any one of the six memory and control function commands.

One control function command instructs the DS1820 to perform a temperature measurement. The result of this measurement will be placed in the DS1820's scratchpad memory, and may be read by issuing a memory function command which reads the contents of the scratchpad memory. The temperature alarm triggers TH and TL consist of one byte EEPROM each. If the alarm search command is not applied to the DS1820, these registers may be used as general purpose user memory. Writing TH and TL is done using a memory function command. Read access to these registers is through the scratchpad. All data is read and written least significant bit first.

**DS1820 BLOCK DIAGRAM** Figure 1



## PARASITE POWER

The block diagram (Figure 1) shows the parasite powered circuitry. This circuitry "steals" power whenever the I/O or $V_{DD}$ pins are high. I/O will provide sufficient power as long as the specified timing and voltage requirements are met (see the section titled "1–Wire Bus System"). The advantages of parasite power are two–fold: 1) by parasiting off this pin, no local power source is needed for remote sensing of temperature, and 2) the ROM may be read in absence of normal power.

In order for the DS1820 to be able to perform accurate temperature conversions, sufficient power must be provided over the I/O line when a temperature conversion is taking place. Since the operating current of the DS1820 is up to 1 mA, the I/O line will not have sufficient drive due to the 5K pull–up resistor. This problem is particularly acute if several DS1820's are on the same I/O and attempting to convert simultaneously.

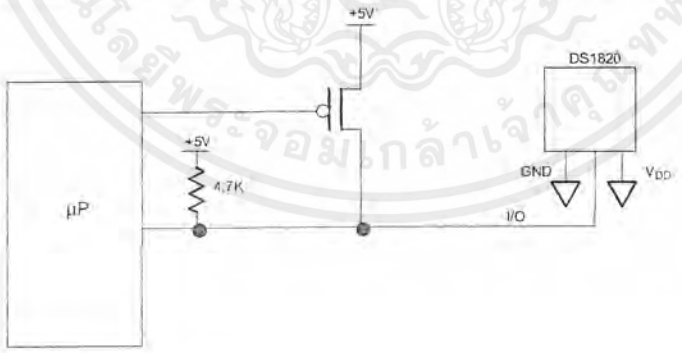
There are two ways to assure that the DS1820 has sufficient supply current during its active conversion cycle. The first is to provide a strong pull–up on the I/O line whenever temperature conversions or copies to the $E^2$ memory are taking place. This may be accomplished by using a MOSFET to pull the I/O line directly to the power supply as shown in Figure 2. The I/O line must be switched over to the strong pull–up within 10 µs maximum after issuing any protocol that involves copying to the $E^2$ memory or initiates temperature conversions. When using the parasite power mode, the $V_{DD}$ pin must be tied to ground.

Another method of supplying current to the DS1820 is through the use of an external power supply tied to the $V_{DD}$ pin, as shown in Figure 3. The advantage to this is that the strong pull–up is not required on the I/O line, and the bus master need not be tied up holding that line high during temperature conversions. This allows other data traffic on the 1–Wire bus during the conversion time. In addition, any number of DS1820's may be placed on the 1–Wire bus, and if they all use external power, they may all simultaneously perform temperature conversions by issuing the Skip ROM command and then issuing the Convert T command. Note that as long as the external power supply is active, the GND pin may not be floating.

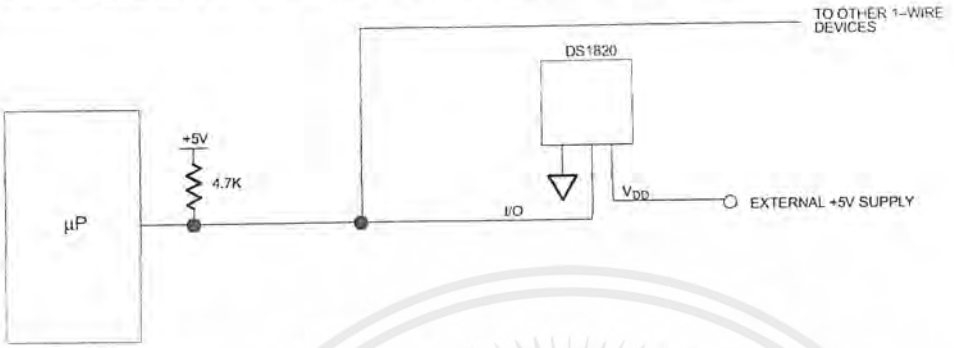
The use of parasite power is not recommended above 100°C, since it may not be able to sustain communications given the higher leakage currents the DS1820 exhibits at these temperatures. For applications in which such temperatures are likely, it is strongly recommended that $V_{DD}$ be applied to the DS1820.

For situations where the bus master does not know whether the DS1820's on the bus are parasite powered or supplied with external $V_{DD}$, a provision is made in the DS1820 to signal the power supply scheme used. The bus master can determine if any DS1820's are on the bus which require the strong pull–up by sending a Skip ROM protocol, then issuing the read power supply command. After this command is issued, the master then issues read time slots. The DS1820 will send back "0" on the 1–Wire bus if it is parasite powered; it will send back a "1" if it is powered from the $V_{DD}$ pin. If the master receives a "0", it knows that it must supply the strong pull–up on the I/O line during temperature conversions. See "Memory Command Functions" section for more detail on this command protocol.

**STRONG PULL–UP FOR SUPPLYING DS1820 DURING TEMPERATURE CONVERSION** Figure 2



## USING $V_{DD}$ TO SUPPLY TEMPERATURE CONVERSION CURRENT Figure 3




## OPERATION – MEASURING TEMPERATURE

The DS1820 measures temperature through the use of an on–board proprietary temperature measurement technique. A block diagram of the temperature measurement circuitry is shown in Figure 4.

The DS1820 measures temperature by counting the number of clock cycles that an oscillator with a low temperature coefficient goes through during a gate period determined by a high temperature coefficient oscillator. The counter is preset with a base count that corresponds to –55°C. If the counter reaches zero before the gate period is over, the temperature register, which is also preset to the –55°C value, is incremented, indicating that the temperature is higher than –55°C.

At the same time, the counter is then preset with a value determined by the slope accumulator circuitry. This circuitry is needed to compensate for the parabolic behavior of the oscillators over temperature. The counter is then clocked again until it reaches zero. If the gate period is still not finished, then this process repeats.

The slope accumulator is used to compensate for the non–linear behavior of the oscillators over temperature, yielding a high resolution temperature measurement. This is done by changing the number of counts necessary for the counter to go through for each incremental degree in temperature. To obtain the desired resolution, therefore, both the value of the counter and the number of counts per degree C (the value of the slope accumulator) at a given temperature must be known.

Internally, this calculation is done inside the DS1820 to provide 0.5°C resolution. The temperature reading is provided in a 16–bit, sign–extended two's complement reading. Table 1 describes the exact relationship of output data to measured temperature. The data is transmitted serially over the 1–Wire interface. The DS1820 can measure temperature over the range of –55°C to +125°C in 0.5°C increments. For Fahrenheit usage, a lookup table or conversion factor must be used.

Note that temperature is represented in the DS1820 in terms of a $^1/_2$°C LSB, yielding the following 9–bit format:

| MSB | | | | | | | | LSB |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |

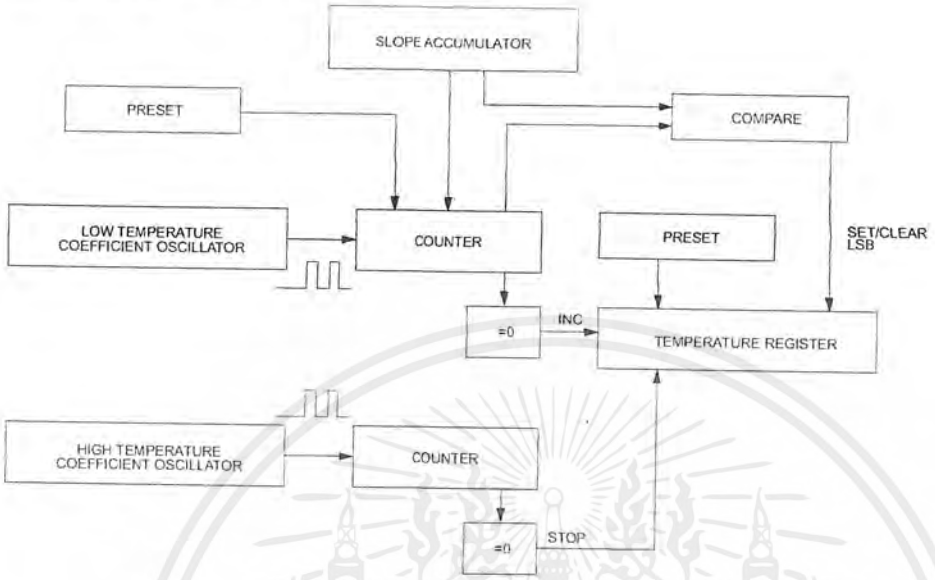
= –25°C

The most significant (sign) bit is duplicated into all of the bits in the upper MSB of the two–byte temperature register in memory. This "sign–extension" yields the 16–bit temperature readings as shown in Table 1.

Higher resolutions may be obtained by the following procedure. First, read the temperature, and truncate the 0.5°C bit (the LSB) from the read value. This value is TEMP_READ. The value left in the counter may then be read. This value is the count remaining (COUNT_REMAIN) after the gate period has ceased. The last value needed is the number of counts per degree C (COUNT_PER_C) at that temperature. The actual temperature may be then be calculated by the user using the following:

$$\text{TEMPERATURE} = \text{TEMP\_READ} - 0.25 + \frac{\text{(COUNT\_PER\_C} - \text{COUNT\_REMAIN)}}{\text{COUNT\_PER\_C}}$$

**TEMPERATURE MEASURING CIRCUITRY** Figure 4




**TEMPERATURE/DATA RELATIONSHIPS** Table 1

| TEMPERATURE | DIGITAL OUTPUT (Binary) | DIGITAL OUTPUT (Hex) |
|---|---|---|
| +125°C | 00000000 11111010 | 00FA |
| +25°C | 00000000 00110010 | 0032h |
| $+^1/_2$°C | 00000000 00000001 | 0001h |
| +0°C | 00000000 00000000 | 0000h |
| $-^1/_2$°C | 11111111 11111111 | FFFFh |
| −25°C | 11111111 11001110 | FFCEh |
| −55°C | 11111111 10010010 | FF92h |

## OPERATION – ALARM SIGNALING

After the DS1820 has performed a temperature conversion, the temperature value is compared to the trigger values stored in TH and TL. Since these registers are 8–bit only, the 0.5°C bit is ignored for comparison. The most significant bit of TH or TL directly corresponds to the sign bit of the 16–bit temperature register. If the result of a temperature measurement is higher than TH or lower than TL, an alarm flag inside the device is set. This flag is updated with every temperature measurement. As long as the alarm flag is set, the DS1820 will respond to the alarm search command. This allows many DS1820s to be connected in parallel doing simultaneous temperature measurements. If somewhere the temperature exceeds the limits, the alarming device(s) can be identified and read immediately without having to read non–alarming devices.

## 64–BIT LASERED ROM

Each DS1820 contains a unique ROM code that is 64–bits long. The first eight bits are a 1–Wire family code (DS1820 code is 10h). The next 48 bits are a unique serial number. The last eight bits are a CRC of the first 56 bits. (See Figure 5.) The 64–bit ROM and ROM Function Control section allow the DS1820 to operate as a 1–Wire device and follow the 1–Wire protocol detailed in the section "1–Wire Bus System". The functions required to control sections of the DS1820 are not accessible until the ROM function protocol has been satisfied. This protocol is described in the ROM function protocol flowchart (Figure 6). The 1–Wire bus master must first provide one of five ROM function commands: 1) Read ROM, 2) Match ROM, 3) Search ROM, 4) Skip ROM, or 5) Alarm Search. After a ROM functions sequence has been successfully executed, the functions specific to the DS1820 are accessible and the bus master may then provide and one of the six memory and control function commands.

## CRC GENERATION

The DS1820 has an 8–bit CRC stored in the most significant byte of the 64–bit ROM. The bus master can compute a CRC value from the first 56–bits of the 64–bit ROM and compare it to the value stored within the DS1820 to determine if the ROM data has been received error–free by the bus master. The equivalent polynomial function of this CRC is:

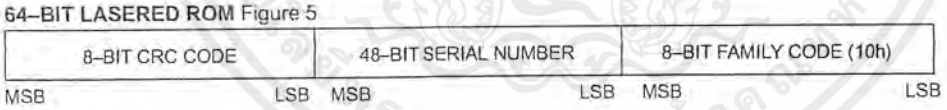
$$CRC = X^8 + X^5 + X^4 + 1$$

The DS1820 also generates an 8–bit CRC value using the same polynomial function shown above and provides this value to the bus master to validate the transfer of data bytes. In each case where a CRC is used for data transfer validation, the bus master must calculate a CRC value using the polynomial function given above and compare the calculated value to either the 8–bit CRC value stored in the 64–bit ROM portion of the DS1820 (for ROM reads) or the 8–bit CRC value computed within the DS1820 (which is read as a ninth byte when the scratchpad is read). The comparison of CRC values and decision to continue with an operation are determined entirely by the bus master. There is no circuitry inside the DS1820 that prevents a command sequence from proceeding if the CRC stored in or calculated by the DS1820 does not match the value generated by the bus master.

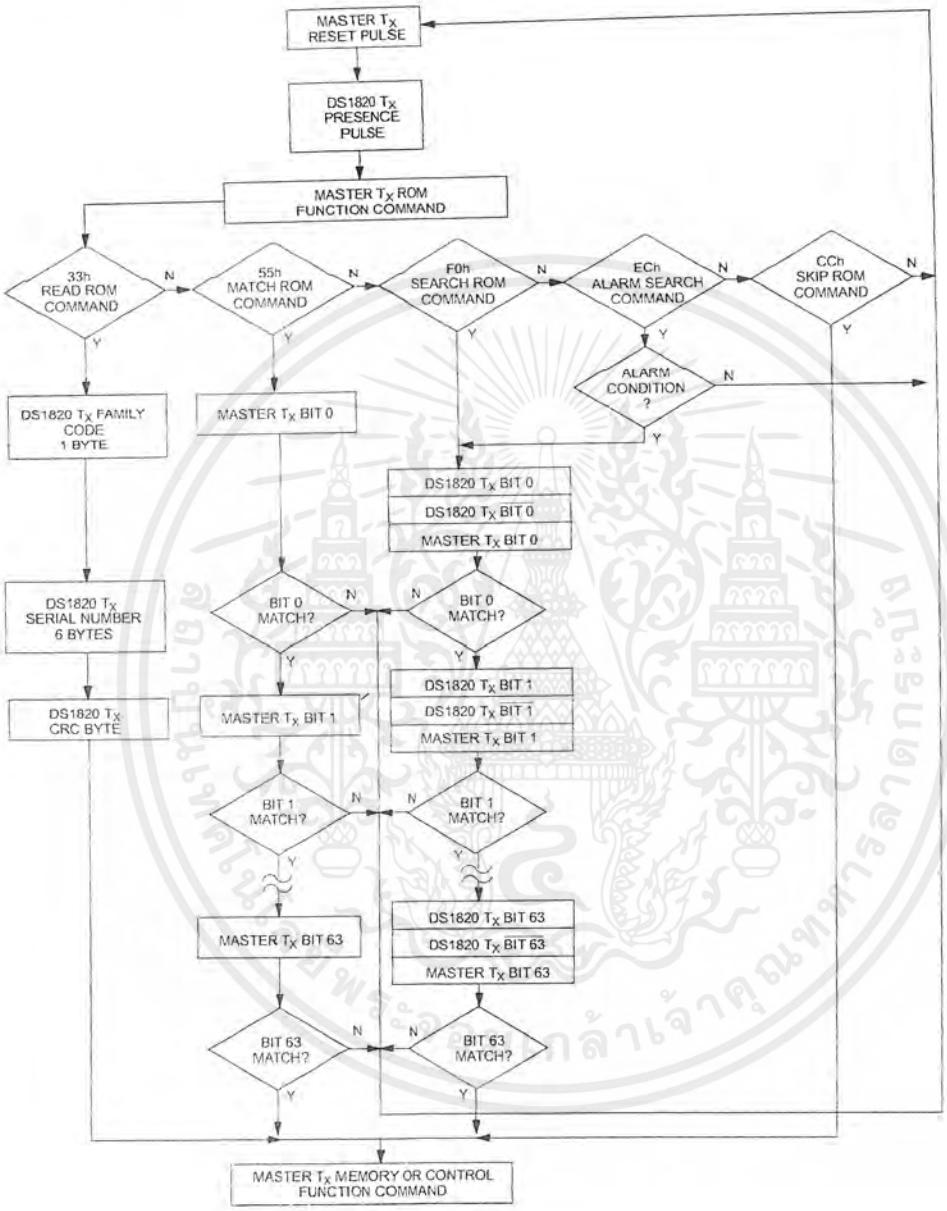
The 1–Wire CRC can be generated using a polynomial generator consisting of a shift register and XOR gates as shown in Figure 7. Additional information about the Dallas 1–Wire Cyclic Redundancy Check is available in Application Note 27 entitled "Understanding and Using Cyclic Redundancy Checks with Dallas Semiconductor *Touch Memory Products*".

The shift register bits are initialized to zero. Then starting with the least significant bit of the family code, one bit at a time is shifted in. After the 8th bit of the family code has been entered, then the serial number is entered. After the 48th bit of the serial number has been entered, the shift register contains the CRC value. Shifting in the eight bits of CRC should return the shift register to all zeros.

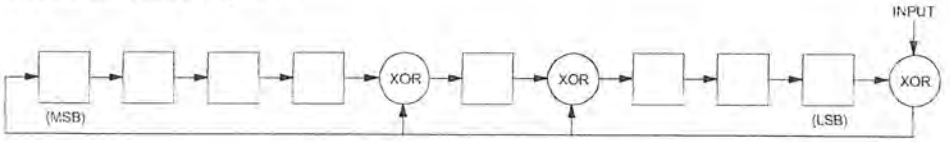
**64–BIT LASERED ROM** Figure 5



## ROM FUNCTIONS FLOW CHART Figure 6



## 1–WIRE CRC CODE Figure 7



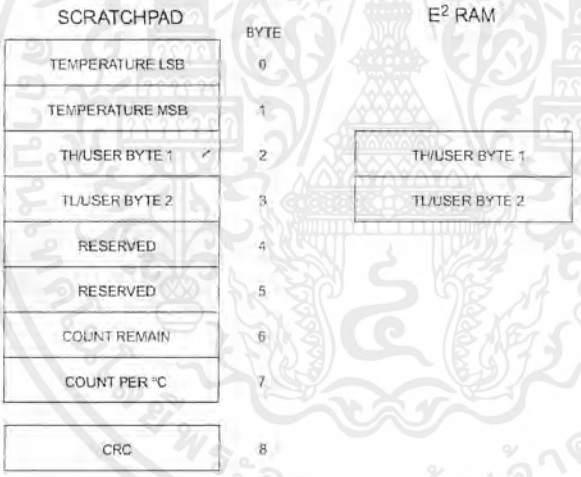

## MEMORY

The DS1820's memory is organized as shown in Figure 8. The memory consists of a scratchpad RAM and a nonvolatile, electrically erasable ($E^2$) RAM, which stores the high and low temperature triggers TH and TL. The scratchpad helps insure data integrity when communicating over the 1–Wire bus. Data is first written to the scratchpad where it can be read back. After the data has been verified, a copy scratchpad command will transfer the data to the nonvolatile ($E^2$) RAM. This process insures data integrity when modifying the memory.

The scratchpad is organized as eight bytes of memory. The first two bytes contain the measured temperature information. The third and fourth bytes are volatile copies of TH and TL and are refreshed with every power–on reset. The next two bytes are not used; upon reading back, however, they will appear as all logic 1's. The seventh and eighth bytes are count registers, which may be used in obtaining higher temperature resolution (see "Operation–measuring Temperature" section).

There is a ninth byte which may be read with a Read Scratchpad command. This byte contains a cyclic redundancy check (CRC) byte which is the CRC over all of the eight previous bytes. This CRC is implemented in the fashion described in the section titled "CRC Generation".

## DS1820 MEMORY MAP Figure 8



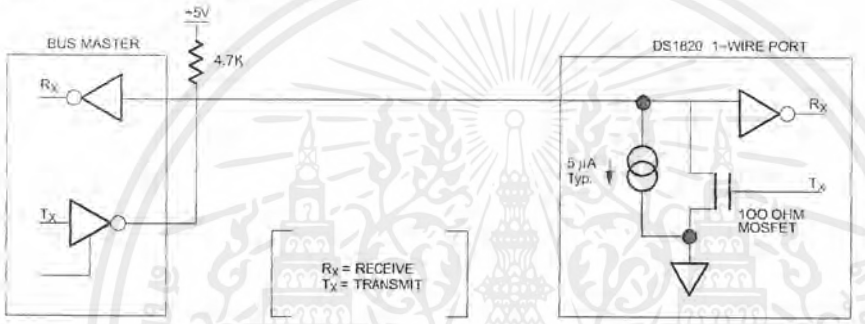
## 1–WIRE BUS SYSTEM

The 1–Wire bus is a system which has a single bus master and one or more slaves. The DS1820 behaves as a slave. The discussion of this bus system is broken down into three topics: hardware configuration, transaction sequence, and 1–Wire signaling (signal types and timing).

## HARDWARE CONFIGURATION

The 1–Wire bus has only a single line by definition; it is important that each device on the bus be able to drive it at the appropriate time. To facilitate this, each device attached to the 1–Wire bus must have open drain or 3–state outputs. The 1–Wire port of the DS1820 (I/O pin) is open drain with an internal circuit equivalent to that shown in Figure 9. A multidrop bus consists of a 1–Wire bus with multiple slaves attached. The 1–Wire bus requires a pullup resistor of approximately 5KΩ

**HARDWARE CONFIGURATION** Figure 9




The idle state for the 1–Wire bus is high. If for any reason a transaction needs to be suspended, the bus MUST be left in the idle state if the transaction is to resume. Infinite recovery time can occur between bits so long as the 1–Wire bus is in the inactive (high) state during the recovery period. If this does not occur and the bus is left low for more than 480 µs, all components on the bus will be reset.

## TRANSACTION SEQUENCE

The protocol for accessing the DS1820 via the 1–Wire port is as follows:

- Initialization
- ROM Function Command
- Memory Function Command
- Transaction/Data

## INITIALIZATION

All transactions on the 1–Wire bus begin with an initialization sequence. The initialization sequence consists of a reset pulse transmitted by the bus master followed by presence pulse(s) transmitted by the slave(s).

The presence pulse lets the bus master know that the DS1820 is on the bus and is ready to operate. For more details, see the "1–Wire Signaling" section.

## ROM FUNCTION COMMANDS

Once the bus master has detected a presence, it can issue one of the five ROM function commands. All ROM function commands are 8–bits long. A list of these commands follows (refer to flowchart in Figure 6):

### Read ROM [33h]
This command allows the bus master to read the DS1820's 8–bit family code, unique 48–bit serial number, and 8–bit CRC. This command can only be used if there is a single DS1820 on the bus. If more than one slave is present on the bus, a data collision will occur when all slaves try to transmit at the same time (open drain will produce a wired AND result).

### Match ROM [55h]
The match ROM command, followed by a 64–bit ROM sequence, allows the bus master to address a specific DS1820 on a multidrop bus. Only the DS1820 that exactly matches the 64–bit ROM sequence will respond to the following memory function command. All slaves that do not match the 64–bit ROM sequence will wait for a reset pulse. This command can be used with a single or multiple devices on the bus.

### Skip ROM [CCh]
This command can save time in a single drop bus system by allowing the bus master to access the memory functions without providing the 64–bit ROM code. If more than one slave is present on the bus and a read command is issued following the Skip ROM command, data collision will occur on the bus as multiple slaves transmit simultaneously (open drain pulldowns will produce a wired AND result).

### Search ROM [F0h]
When a system is initially brought up, the bus master might not know the number of devices on the 1–Wire bus or their 64–bit ROM codes. The search ROM command allows the bus master to use a process of elimination to identify the 64–bit ROM codes of all slave devices on the bus.

### Alarm Search [ECh]
The flowchart of this command is identical to the Search ROM command. However, the DS1820 will respond to this command only if an alarm condition has been encountered at the last temperature measurement. An alarm condition is defined as a temperature higher than TH or lower than TL. The alarm condition remains set as long as the DS1820 is powered up, or until another temperature measurement reveals a non–alarming value. For alarming, the trigger values stored in EEPROM are taken into account. If an alarm condition exists and the TH or TL settings are changed, another temperature conversion should be done to validate any alarm conditions.

### Example of a ROM Search
The ROM search process is the repetition of a simple 3–step routine: read a bit, read the complement of the bit, then write the desired value of that bit. The bus master performs this simple, 3–step routine on each bit of the ROM. After one complete pass, the bus master knows the contents of the ROM in one device. The remaining number of devices and their ROM codes may be identified by additional passes.

The following example of the ROM search process assumes four different devices are connected to the same 1–Wire bus. The ROM data of the four devices is as shown:

| | |
|---|---|
| ROM1 | 00110101... |
| ROM2 | 10101010... |
| ROM3 | 11110101... |
| ROM4 | 00010001... |

The search process is as follows:

1. The bus master begins the initialization sequence by issuing a reset pulse. The slave devices respond by issuing simultaneous presence pulses.
2. The bus master will then issue the Search ROM command on the 1–Wire bus.
3. The bus master reads a bit from the 1–Wire bus. Each device will respond by placing the value of the first bit of their respective ROM data onto the 1–Wire bus. ROM1 and ROM4 will place a 0 onto the 1–Wire bus, i.e., pull it low. ROM2 and ROM3 will place a 1 onto the 1–Wire bus by allowing the line to stay high. The result is the logical AND of all devices on the line, therefore the bus master sees a 0. The bus master reads another bit. Since the Search ROM data command is being executed, all of the devices on the 1–Wire bus respond to this second read by placing the complement of the first bit of their respective ROM data onto the 1–Wire bus. ROM1 and ROM4 will place a 1 onto the 1–Wire, allowing the line to stay high. ROM2 and ROM3 will place a 0 onto the 1–Wire, thus it will be pulled low. The bus master again observes a 0 for the complement of the first ROM data bit. The bus master has determined that there are some devices on the 1–Wire bus that have a 0 in the first position and others that have a 1.

The data obtained from the two reads of the 3–step routine have the following interpretations:

00 There are still devices attached which have conflicting bits in this position.

01 All devices still coupled have a 0–bit in this bit position.

10 All devices still coupled have a 1–bit in this bit position.

11 There are no devices attached to the 1–Wire bus.

4. The bus master writes a 0. This deselects ROM2 and ROM3 for the remainder of this search pass, leaving only ROM1 and ROM4 connected to the 1–Wire bus.
5. The bus master performs two more reads and receives a 0–bit followed by a 1–bit. This indicates that all devices still coupled to the bus have 0's as their second ROM data bit.
6. The bus master then writes a 0 to keep both ROM1 and ROM4 coupled.
7. The bus master executes two reads and receives two 0–bits. This indicates that both 1–bits and 0–bits exist as the third bit of the ROM data of the attached devices.
8. The bus master writes a 0–bit. This deselects ROM1 leaving ROM4 as the only device still connected.
9. The bus master reads the remainder of the ROM bits for ROM4 and continues to access the part if desired. This completes the first pass and uniquely identifies one part on the 1–Wire bus.
10. The bus master starts a new ROM search sequence by repeating steps 1 through 7.
11. The bus master writes a 1–bit. This decouples ROM4, leaving only ROM1 still coupled.
12. The bus master reads the remainder of the ROM bits for ROM1 and communicates to the underlying logic if desired. This completes the second ROM search pass, in which another of the ROMs was found.
13. The bus master starts a new ROM search by repeating steps 1 through 3.
14. The bus master writes a 1–bit. This deselects ROM1 and ROM4 for the remainder of this search pass, leaving only ROM2 and ROM3 coupled to the system.
15. The bus master executes two read time slots and receives two zeros.
16. The bus master writes a 0–bit. This decouples ROM3, and leaving only ROM2.
17. The bus master reads the remainder of the ROM bits for ROM2 and communicates to the underlying logic if desired. This completes the third ROM search pass, in which another of the ROMs was found.
18. The bus master starts a new ROM search by repeating steps 13 through 15.
19. The bus master writes a 1–bit. This decouples ROM2, leaving only ROM3.
20. The bus master reads the remainder of the ROM bits for ROM3 and communicates to the underlying logic if desired. This completes the fourth ROM search pass, in which another of the ROMs was found.

**Note the following:**
The bus master learns the unique ID number (ROM data pattern) of one 1–Wire device on each ROM Search operation. The time required to derive the part's unique ROM code is:

$$960\ \mu s + (8 + 3 \times 64)\ 61\ \mu s = 13.16\ ms$$

The bus master is therefore capable of identifying 75 different 1–Wire devices per second.

## I/O SIGNALING

The DS1820 requires strict protocols to insure data integrity. The protocol consists of several types of signaling on one line: reset pulse, presence pulse, write 0, write 1, read 0, and read 1. All of these signals, with the exception of the presence pulse, are initiated by the bus master.

The initialization sequence required to begin any communication with the DS1820 is shown in Figure 11. A reset pulse followed by a presence pulse indicates the DS1820 is ready to send or receive data given the correct ROM command and memory function command.

The bus master transmits (TX) a reset pulse (a low signal for a minimum of 480 µs). The bus master then releases the line and goes into a receive mode (RX). The 1–Wire bus is pulled to a high state via the 5K pull–up resistor . After detecting the rising edge on the

I/O pin, the DS1820 waits 15–60 µs and then transmits the presence pulse (a low signal for 60–240 µs).

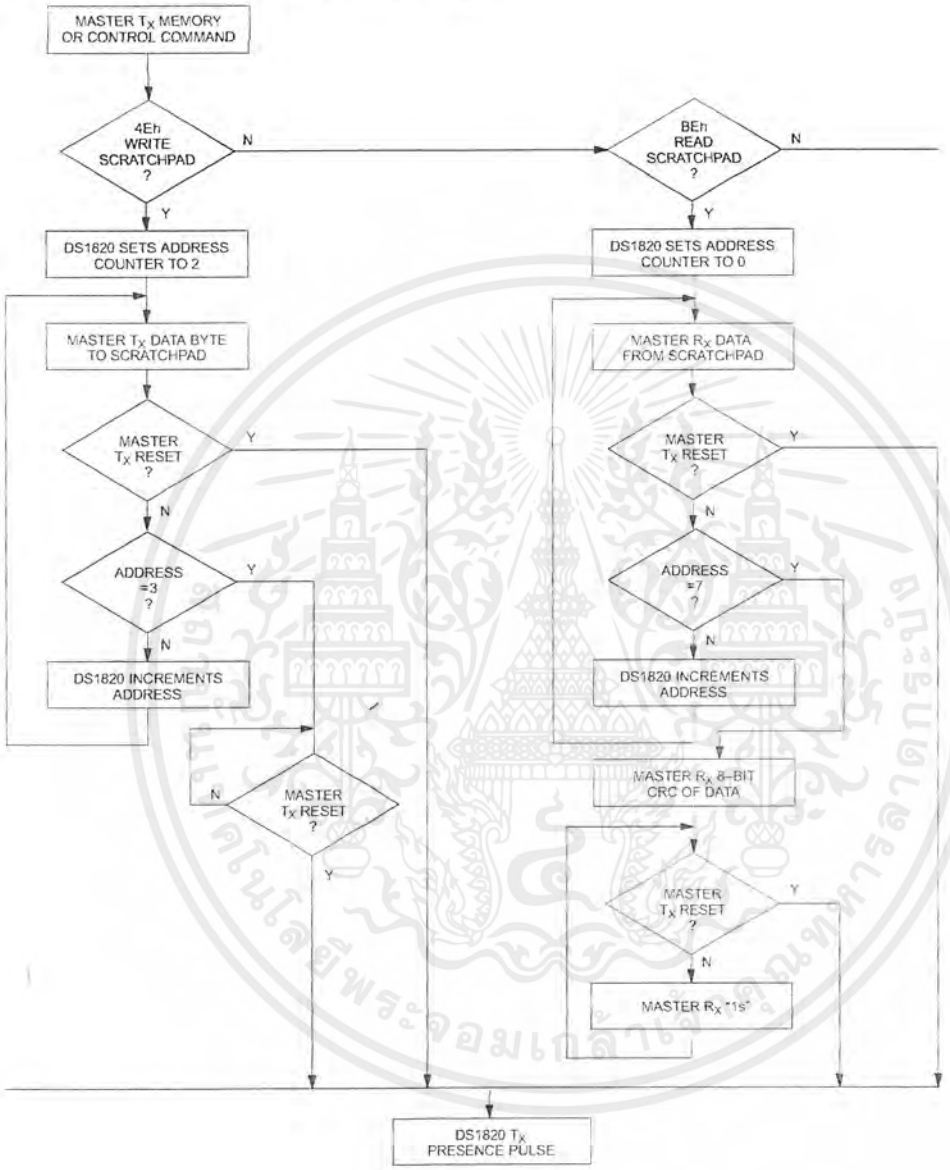
## MEMORY COMMAND FUNCTIONS

The following command protocols are summarized in Table 2, and by the flowchart of Figure 10.

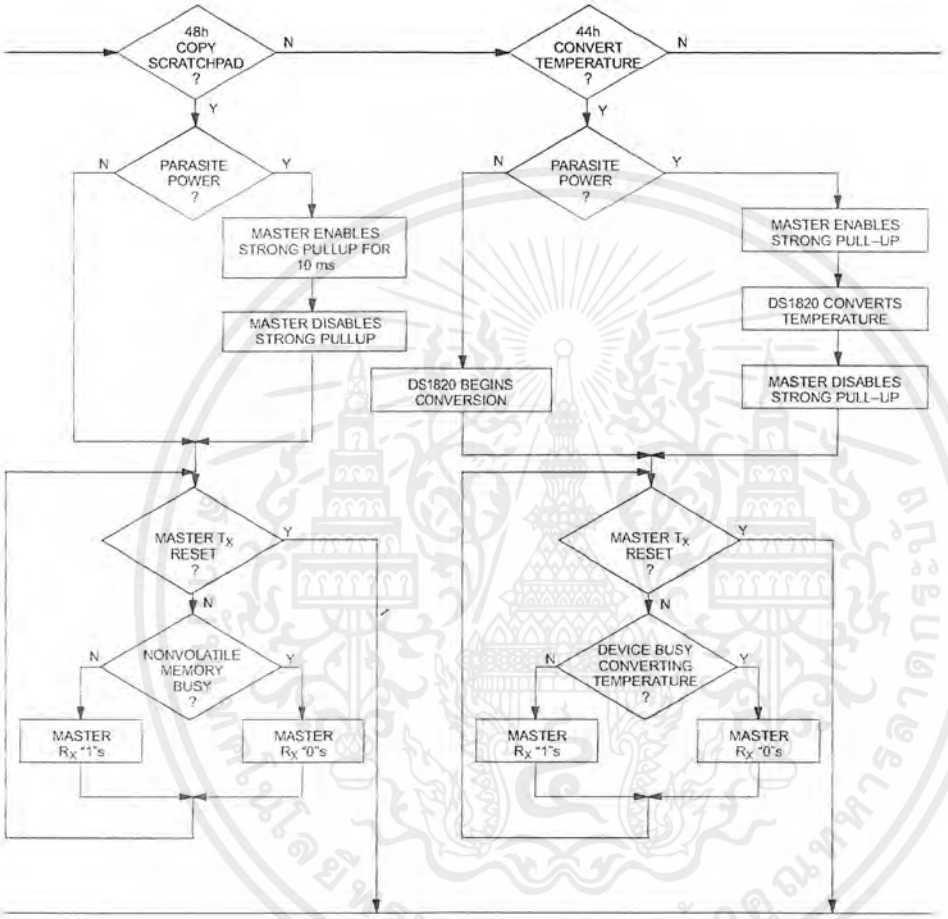
### Write Scratchpad [4Eh]

This command writes to the scratchpad of the DS1820, starting at address 2. The next two bytes written will be saved in scratchpad memory, at address locations 2 and 3. Writing may be terminated at any point by issuing a reset.

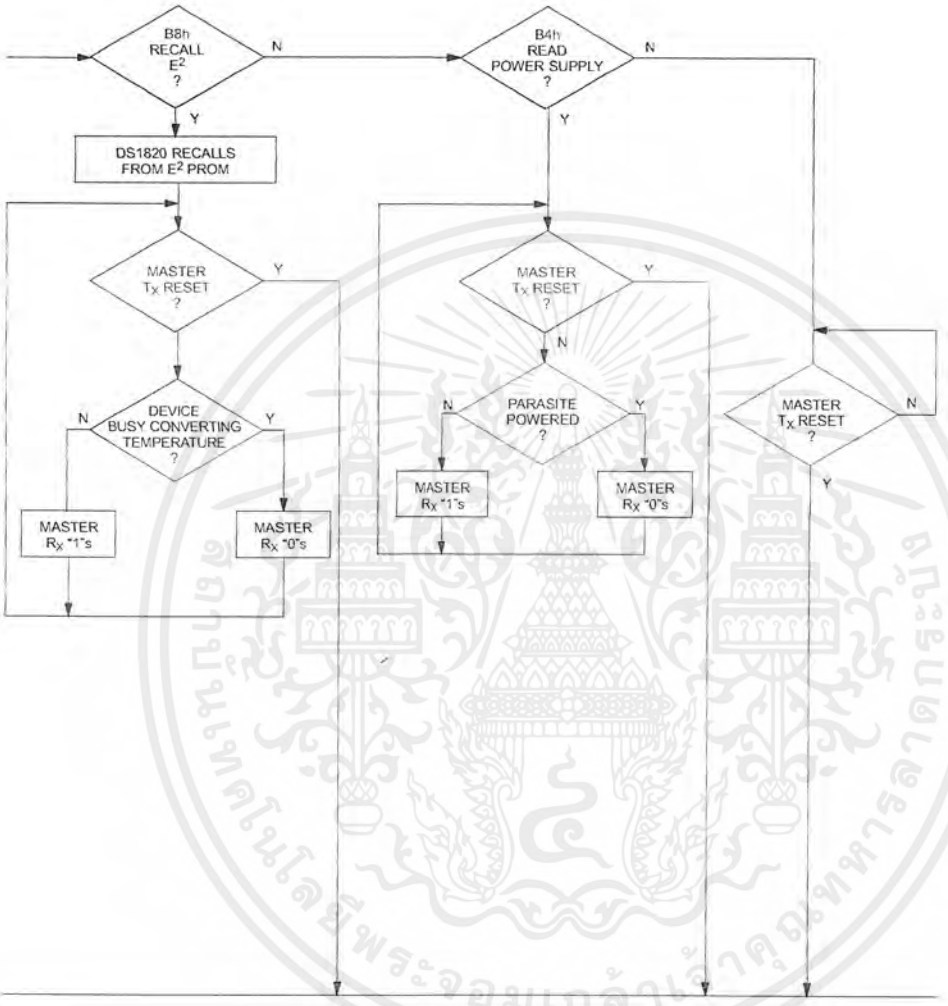
## MEMORY FUNCTIONS FLOW CHART Figure 10



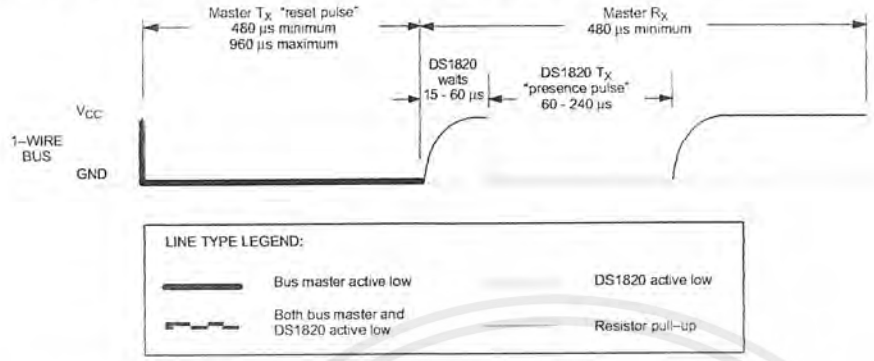
**MEMORY FUNCTIONS FLOW CHART** Figure 10 (cont'd)



**MEMORY FUNCTIONS FLOW CHART** Figure 10 (cont'd)



## INITIALIZATION PROCEDURE "RESET AND PRESENCE PULSES" Figure 11




## DS1820 COMMAND SET Table 2

| INSTRUCTION | DESCRIPTION | PROTOCOL | 1–WIRE BUS AFTER ISSUING PROTOCOL | NOTES |
|---|---|---|---|---|
| **TEMPERATURE CONVERSION COMMANDS** | | | | |
| Convert T | Initiates temperature conversion. | 44h | <read temperature busy status> | 1 |
| **MEMORY COMMANDS** | | | | |
| Read Scratchpad | Reads bytes from scratchpad and reads CRC byte. | BEh | <read data up to 9 bytes> | |
| Write Scratchpad | Writes bytes into scratchpad at addresses 2 and 3 (TH and TL temperature triggers). | 4Eh | <write data into 2 bytes at addr. 2 and addr. 3> | |
| Copy Scratchpad | Copies scratchpad into nonvolatile memory (addresses 2 and 3 only). | 48h | <read copy status> | 2 |
| Recall $E^2$ | Recalls values stored in nonvolatile memory into scratchpad (temperature triggers). | B8h | <read temperature busy status> | |
| Read Power Supply | Signals the mode of DS1820 power supply to the master. | B4h | <read supply status> | |

**NOTES:**

1. Temperature conversion takes up to 500 ms. After receiving the Convert T protocol, if the part does not receive power from the $V_{DD}$ pin, the I/O line for the DS1820 must be held high for at least 500 ms to provide power during the conversion process. As such, no other activity may take place on the 1–Wire bus for at least this period after a Convert T command has been issued.
2. After receiving the Copy Scratchpad protocol, if the part does not receive power from the $V_{DD}$ pin, the I/O line for the DS1820 must be held high for at least 10 ms to provide power during the copy process. As such, no other activity may take place on the 1–Wire bus for at least this period after a Copy Scratchpad command has been issued.

### Read Scratchpad [BEh]
This command reads the contents of the scratchpad. Reading will commence at byte 0, and will continue through the scratchpad until the 9th (byte–8, CRC) byte is read. If not all locations are to be read, the master may issue a reset to terminate reading at any time.

### Copy Scratchpad [48h]
This command copies the scratchpad into the $E^2$ memory of the DS1820, storing the temperature trigger bytes in nonvolatile memory. If the bus master issues read time slots following this command, the DS1820 will output "0" on the bus as long as it is busy copying the scratchpad to $E^2$; it will return a "1" when the copy process is complete. If parasite powered, the bus master has to enable a strong pull–up for at least 10 ms immediately after issuing this command.

### Convert T [44h]
This command begins a temperature conversion. No further data is required. The temperature conversion will be performed and then the DS1820 will remain idle. If the bus master issues read time slots following this command, the DS1820 will output "0" on the bus as long as it is busy making a temperature conversion; it will return a "1" when the temperature conversion is complete. If parasite powered, the bus master has to enable a strong pullup for 500 ms immediately after issuing this command.

### Recall E2 [B8h]
This command recalls the temperature trigger values stored in $E^2$ to the scratchpad. This recall operation happens automatically upon power–up to the DS1820 as well, so valid data is available in the scratchpad as soon as the device has power applied. With every read data time slot issued after this command has been sent, the device will output its temperature converter busy flag "0"=busy, "1"=ready.

### Read Power Supply [B4h]
With every read data time slot issued after this command has been sent to the DS1820, the device will signal its power mode: "0"=parasite power, "1"=external power supply provided.

## READ/WRITE TIME SLOTS
DS1820 data is read and written through the use of time slots to manipulate bits and a command word to specify the transaction.

### Write Time Slots
A write time slot is initiated when the host pulls the data line from a high logic level to a low logic level. There are two types of write time slots: Write One time slots and Write Zero time slots. All write time slots must be a minimum of 60 µs in duration with a minimum of a one µs recovery time between individual write cycles.

The DS1820 samples the I/O line in a window of 15 µs to 60 µs after the I/O line falls. If the line is high, a Write One occurs. If the line is low, a Write Zero occurs (see Figure 12).

For the host to generate a Write One time slot, the data line must be pulled to a logic low level and then released, allowing the data line to pull up to a high level within 15 µs after the start of the write time slot.

For the host to generate a Write Zero time slot, the data line must be pulled to a logic low level and remain low for 60 µs.

### Read Time Slots
The host generates read time slots when data is to be read from the DS1820. A read time slot is initiated when the host pulls the data line from a logic high level to logic low level. The data line must remain at a low logic level for a minimum of one µs; output data from the DS1820 is valid for 15 µs after the falling edge of the read time slot. The host therefore must stop driving the I/O pin low in order to read its state 15 µs from the start of the read slot (see Figure 12). By the end of the read time slot, the I/O pin will pull back high via the external pull–up resistor. All read time slots must be a minimum of 60 µs in duration with a minimum of a one µs recovery time between individual read slots.

Figure 13 shows that the sum of $T_{INIT}$, $T_{RC}$, and $T_{SAMPLE}$ must be less than 15 µs. Figure 14 shows that system timing margin is maximized by keeping $T_{INIT}$ and $T_{RC}$ as small as possible and by locating the master sample time towards the end of the 15 µs period.

## 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

# 8-Bit Microcontroller with 4K Bytes Flash

# AT89C51

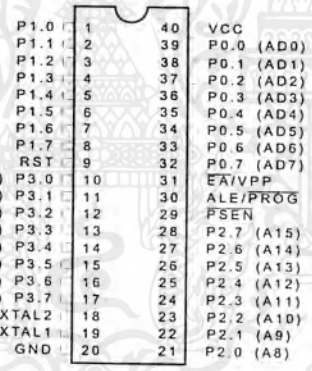
## Description

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.

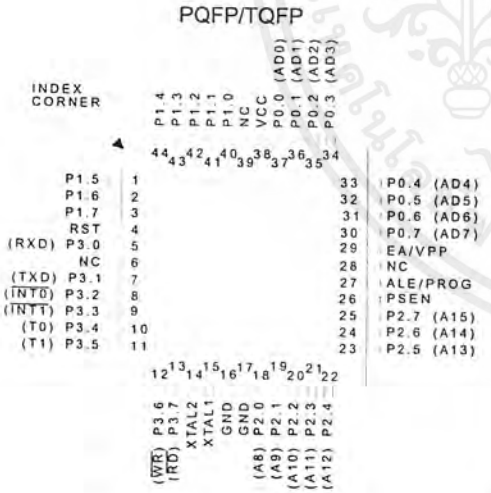
*(continued)*

## Pin Configurations

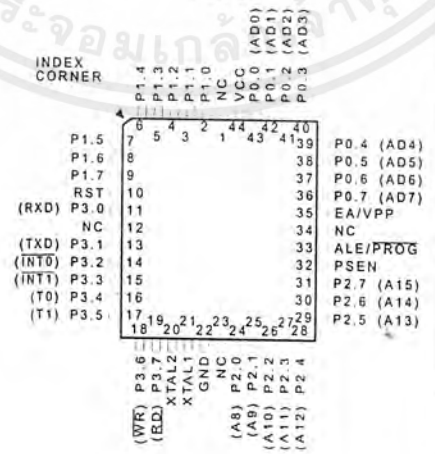
PDIP




PQFP/TQFP

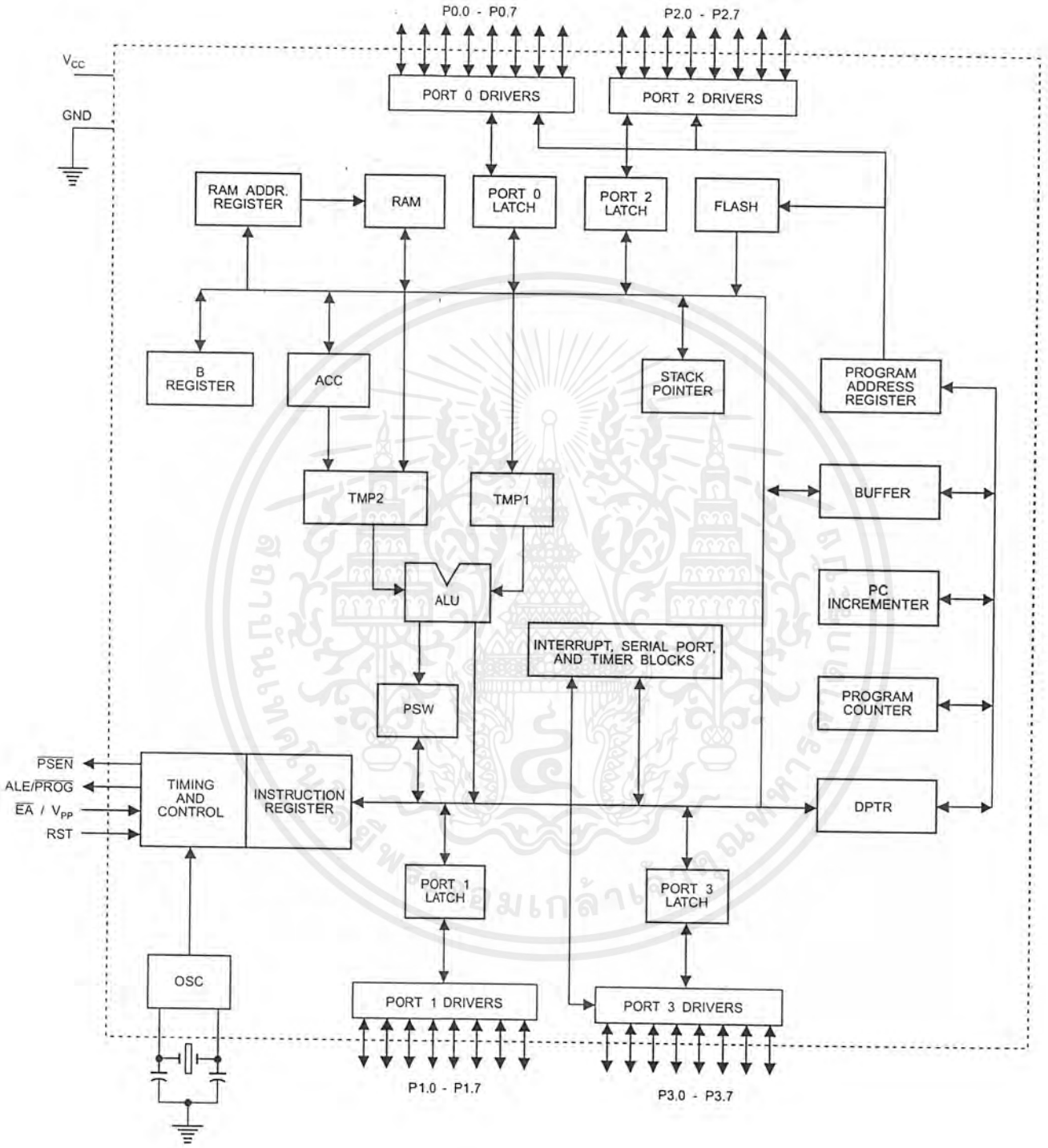



PLCC




0265F-A-12/97

## Block 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

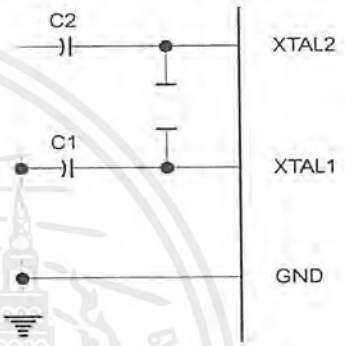
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected 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.

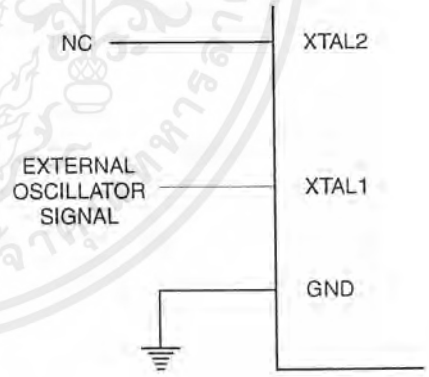
It should be noted that when idle is terminated by a hard ware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port 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<br>xxxx<br>yyww | AT89C51<br>xxxx-5<br>yyww |
| Signature | (030H)=1EH<br>(031H)=51H<br>(032H)=FFH | (030H)=1EH<br>(031H)=51H<br>(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

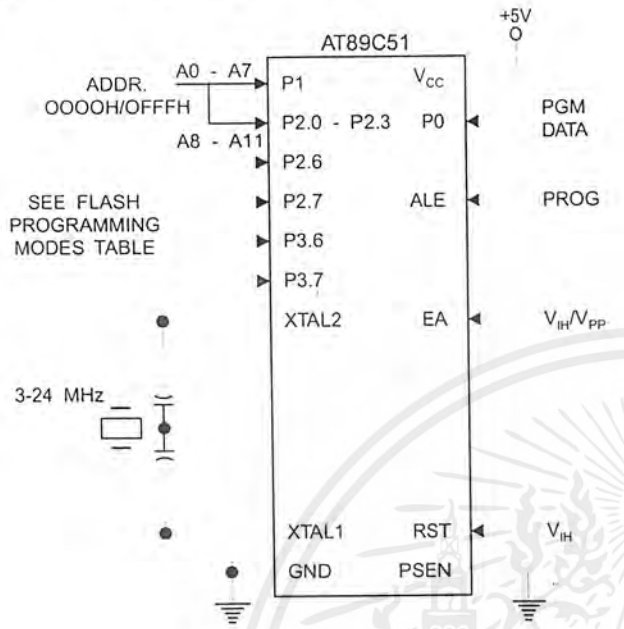
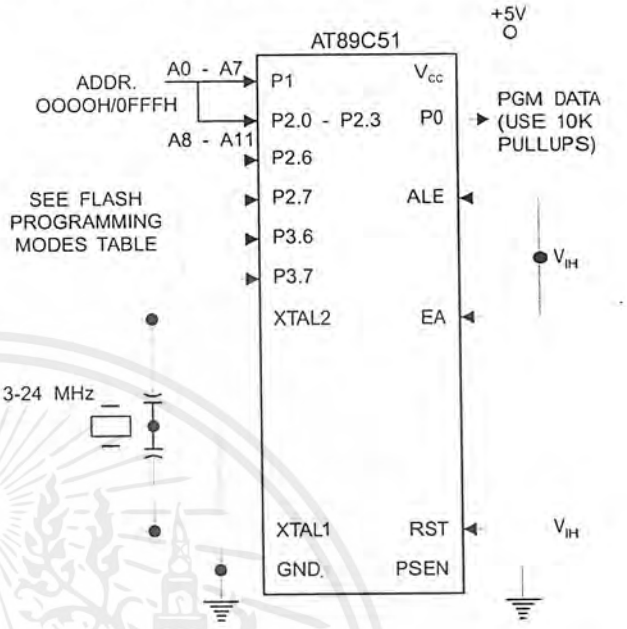



**Figure 4.** Verifying the Flash



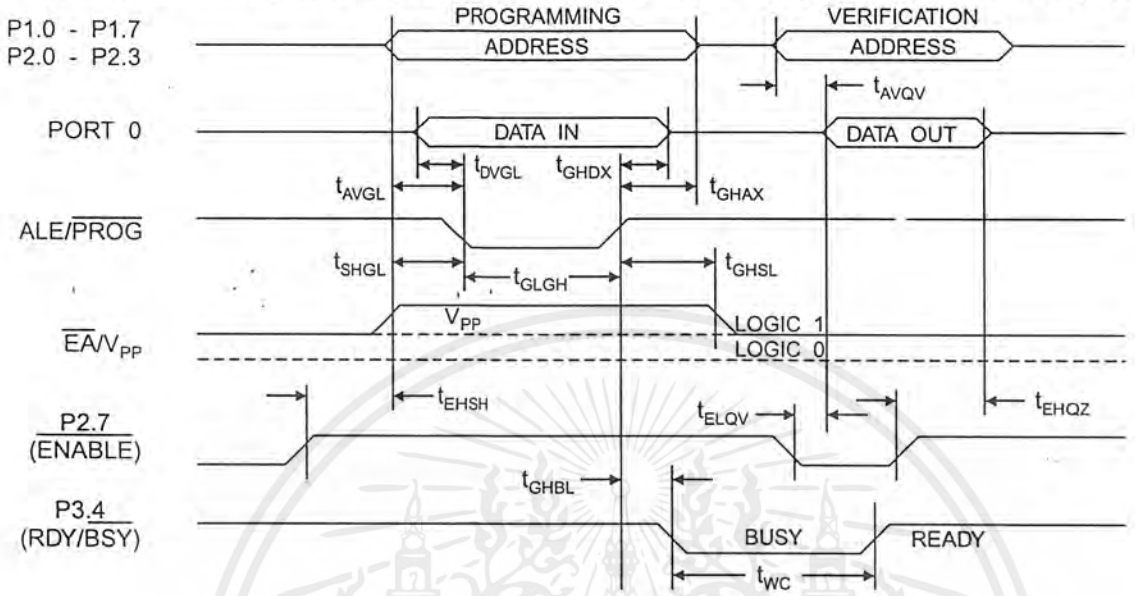

## Flash Programming and Verification Characteristics

$T_A$ = 0°C to 70°C, $V_{CC}$ = 5.0 ± 10%

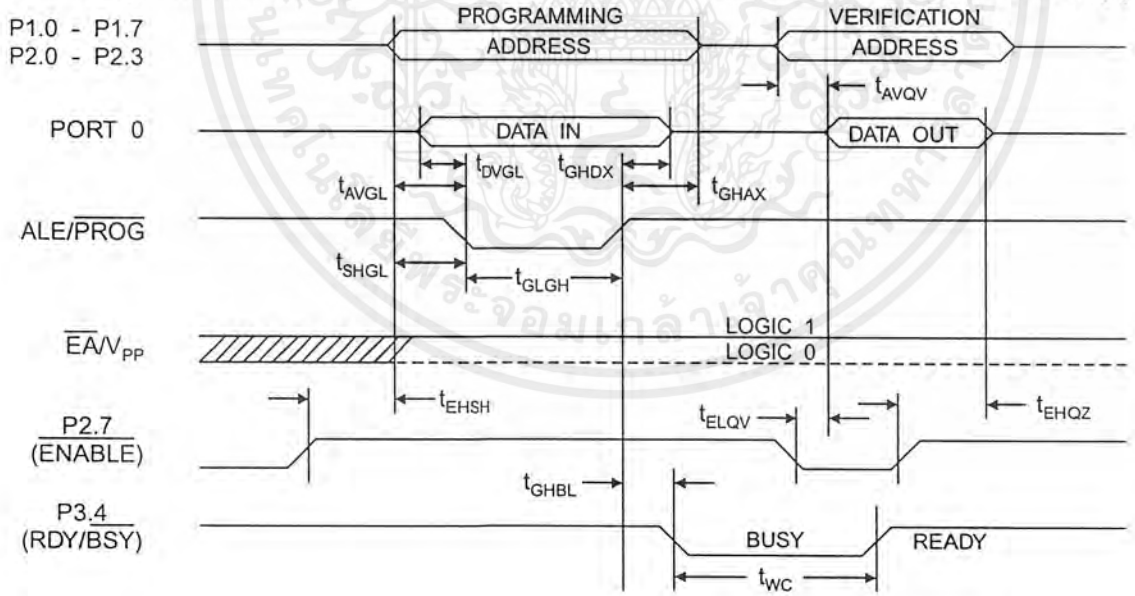
| Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|
| $V_{PP}$(1) | Programming Enable Voltage | 11.5 | 12.5 | V |
| $I_{PP}$(1) | Programming Enable Current | | 1.0 | mA |
| $1/t_{CLCL}$ | Oscillator Frequency | 3 | 24 | MHz |
| $t_{AVGL}$ | Address Setup to $\overline{PROG}$ Low | $48t_{CLCL}$ | | |
| $t_{GHAX}$ | 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)



## 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 | 15.0 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, $V_{CC}$ = 5.0V ± 20% (unless otherwise noted)

| Symbol | Parameter | Condition | Min | Max | Units |
|---|---|---|---|---|---|
| $V_{IL}$ | Input Low Voltage | (Except $\overline{EA}$) | -0.5 | 0.2 $V_{CC}$ - 0.1 | V |
| $V_{IL1}$ | Input Low Voltage ($\overline{EA}$) | | -0.5 | 0.2 $V_{CC}$ - 0.3 | 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 1,2,3) | $I_{OL}$ = 1.6 mA | | 0.45 | V |
| $V_{OL1}$ | Output Low Voltage(1) (Port 0, ALE, $\overline{PSEN}$) | $I_{OL}$ = 3.2 mA | | 0.45 | V |
| $V_{OH}$ | Output High Voltage (Ports 1,2,3, ALE, $\overline{PSEN}$) | $I_{OH}$ = -60 µA, $V_{CC}$ = 5V ± 10% | 2.4 | | V |
| | | $I_{OH}$ = -25 µA | 0.75 $V_{CC}$ | | V |
| | | $I_{OH}$ = -10 µA | 0.9 $V_{CC}$ | | V |
| $V_{OH1}$ | Output High Voltage (Port 0 in External Bus Mode) | $I_{OH}$ = -800 µA, $V_{CC}$ = 5V ± 10% | 2.4 | | V |
| | | $I_{OH}$ = -300 µA | 0.75 $V_{CC}$ | | V |
| | | $I_{OH}$ = -80 µA | 0.9 $V_{CC}$ | | V |
| $I_{IL}$ | Logical 0 Input Current (Ports 1,2,3) | $V_{IN}$ = 0.45V | | -50 | µA |
| $I_{TL}$ | Logical 1 to 0 Transition Current (Ports 1,2,3) | $V_{IN}$ = 2V, VCC = 5V ± 10% | | -650 | µA |
| $I_{LI}$ | Input Leakage Current (Port 0, $\overline{EA}$) | 0.45 < $V_{IN}$ < $V_{CC}$ | | ±10 | µA |
| 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 | | 20 | mA |
| | | Idle Mode, 12 MHz | | 5 | mA |
| | Power Down Mode(2) | $V_{CC}$ = 6V | | 100 | µA |
| | | $V_{CC}$ = 3V | | 40 | µA |

Notes: 1. Under steady state (non-transient) conditions, $I_{OL}$ must be externally limited as follows:
Maximum $I_{OL}$ per port pin: 10 mA
Maximum $I_{OL}$ per 8-bit port: Port 0: 26 mA
Ports 1, 2, 3: 15 mA
Maximum total $I_{OL}$ for all output pins: 71 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.

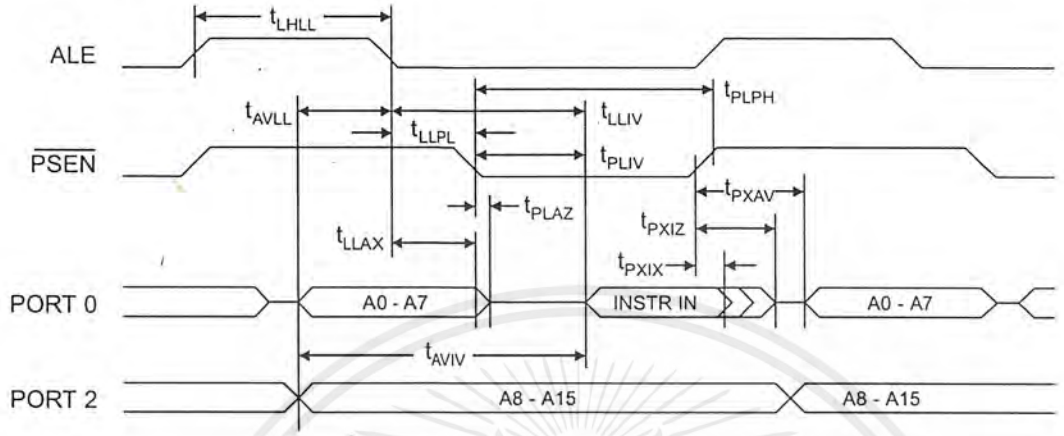
## AC Characteristics

(Under Operating Conditions; Load Capacitance for Port 0, ALE/$\overline{PROG}$, and $\overline{PSEN}$ = 100 pF; Load Capacitance for all other outputs = 80 pF)

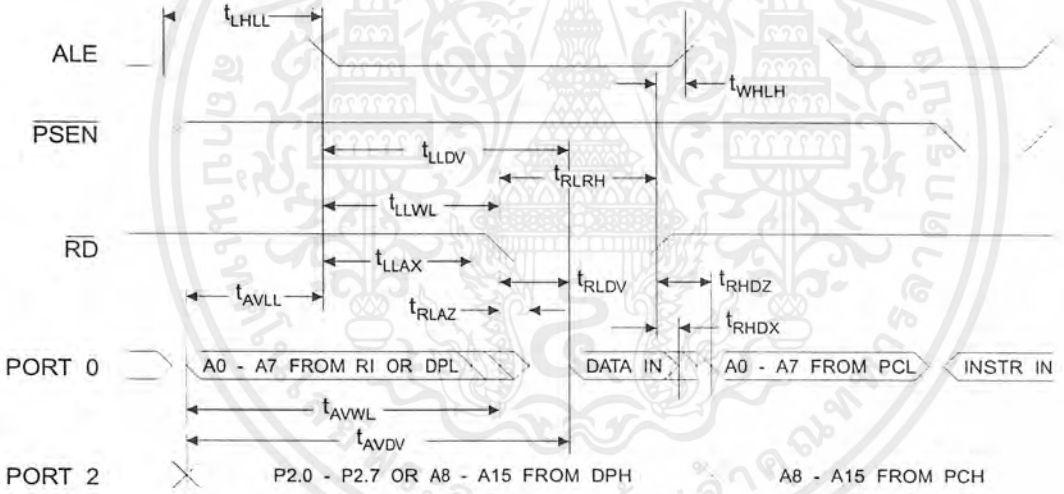
## External Program and Data Memory Characteristics

| Symbol | Parameter | 12 MHz Oscillator | | 16 to 24 MHz Oscillator | | Units |
|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | |
| $1/t_{CLCL}$ | Oscillator Frequency | | | 0 | 24 | MHz |
| $t_{LHLL}$ | ALE Pulse Width | 127 | | $2t_{CLCL}$-40 | | ns |
| $t_{AVLL}$ | Address Valid to ALE Low | 43 | | $t_{CLCL}$-13 | | ns |
| $t_{LLAX}$ | Address Hold After ALE Low | 48 | | $t_{CLCL}$-20 | | ns |
| $t_{LLIV}$ | ALE Low to Valid Instruction In | | 233 | | $4t_{CLCL}$-65 | ns |
| $t_{LLPL}$ | ALE Low to $\overline{PSEN}$ Low | 43 | | $t_{CLCL}$-13 | | ns |
| $t_{PLPH}$ | $\overline{PSEN}$ Pulse Width | 205 | | $3t_{CLCL}$-20 | | ns |
| $t_{PLIV}$ | $\overline{PSEN}$ Low to Valid Instruction In | | 145 | | $3t_{CLCL}$-45 | ns |
| $t_{PXIX}$ | Input Instruction Hold After $\overline{PSEN}$ | 0 | | 0 | | ns |
| $t_{PXIZ}$ | Input Instruction Float After $\overline{PSEN}$ | | 59 | | $t_{CLCL}$-10 | ns |
| $t_{PXAV}$ | $\overline{PSEN}$ to Address Valid | 75 | | $t_{CLCL}$-8 | | ns |
| $t_{AVIV}$ | Address to Valid Instruction In | | 312 | | $5t_{CLCL}$-55 | ns |
| $t_{PLAZ}$ | $\overline{PSEN}$ Low to Address Float | | 10 | | 10 | ns |
| $t_{RLRH}$ | $\overline{RD}$ Pulse Width | 400 | | $6t_{CLCL}$-100 | | ns |
| $t_{WLWH}$ | $\overline{WR}$ Pulse Width | 400 | | $6t_{CLCL}$-100 | | ns |
| $t_{RLDV}$ | $\overline{RD}$ Low to Valid Data In | | 252 | | $5t_{CLCL}$-90 | ns |
| $t_{RHDX}$ | Data Hold After $\overline{RD}$ | 0 | | 0 | | ns |
| $t_{RHDZ}$ | Data Float After $\overline{RD}$ | | 97 | | $2t_{CLCL}$-28 | ns |
| $t_{LLDV}$ | ALE Low to Valid Data In | | 517 | | $8t_{CLCL}$-150 | ns |
| $t_{AVDV}$ | Address to Valid Data In | | 585 | | $9t_{CLCL}$-165 | ns |
| $t_{LLWL}$ | ALE Low to $\overline{RD}$ or $\overline{WR}$ Low | 200 | 300 | $3t_{CLCL}$-50 | $3t_{CLCL}$+50 | ns |
| $t_{AVWL}$ | Address to $\overline{RD}$ or $\overline{WR}$ Low | 203 | | $4t_{CLCL}$-75 | | ns |
| $t_{QVWX}$ | Data Valid to $\overline{WR}$ Transition | 23 | | $t_{CLCL}$-20 | | ns |
| $t_{QVWH}$ | Data Valid to $\overline{WR}$ High | 433 | | $7t_{CLCL}$-120 | | ns |
| $t_{WHQX}$ | Data Hold After $\overline{WR}$ | 33 | | $t_{CLCL}$-20 | | ns |
| $t_{RLAZ}$ | $\overline{RD}$ Low to Address Float | | 0 | | 0 | ns |
| $t_{WHLH}$ | $\overline{RD}$ or $\overline{WR}$ High to ALE High | 43 | 123 | $t_{CLCL}$-20 | $t_{CLCL}$+25 | ns |

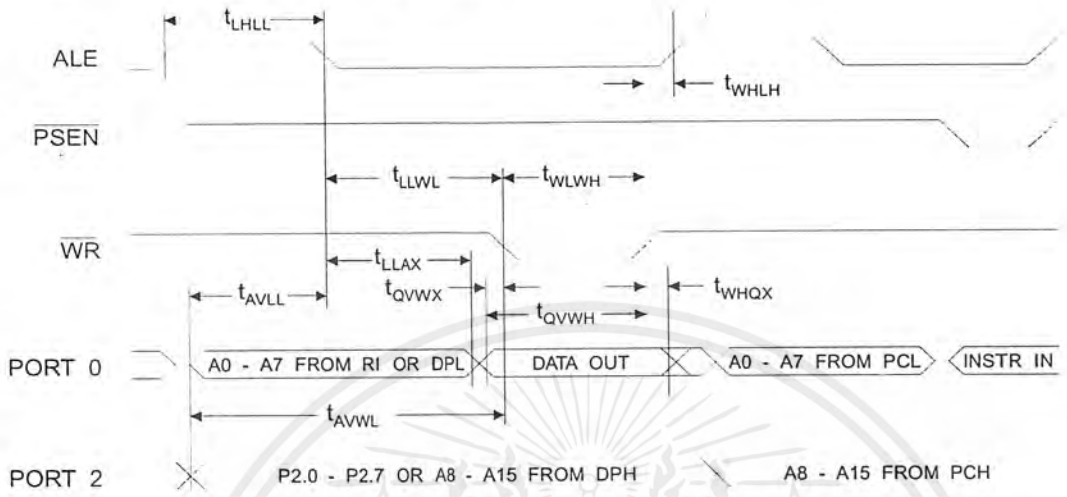
## External Program Memory Read Cycle



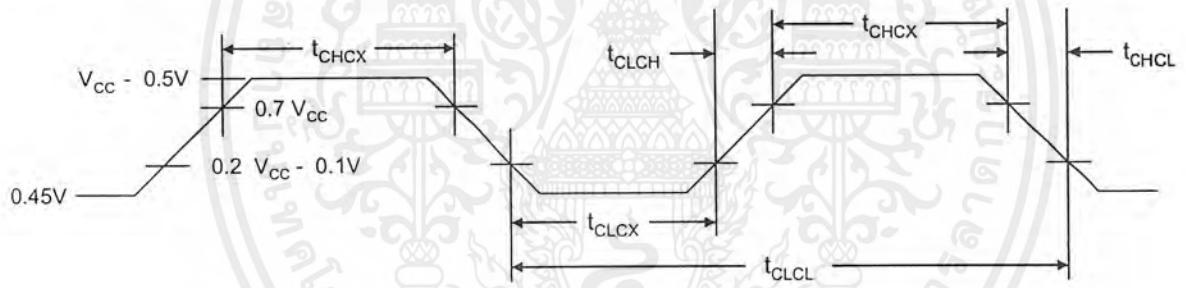

## External Data Memory Read Cycle



## External Data Memory Write Cycle




## External Clock Drive Waveforms




## External Clock Drive

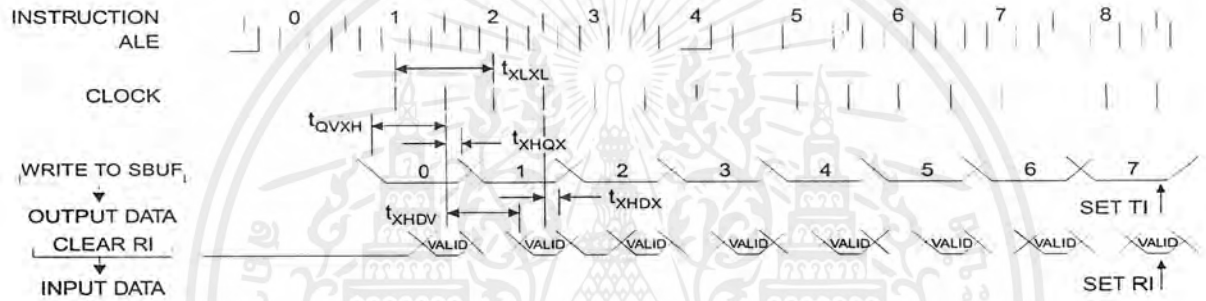
| Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|
| $1/t_{CLCL}$ | Oscillator Frequency | 0 | 24 | MHz |
| $t_{CLCL}$ | Clock Period | 41.6 | | ns |
| $t_{CHCX}$ | High Time | 15 | | ns |
| $t_{CLCX}$ | Low Time | 15 | | ns |
| $t_{CLCH}$ | Rise Time | | 20 | ns |
| $t_{CHCL}$ | Fall Time | | 20 | ns |

## Serial Port Timing: Shift Register Mode Test Conditions

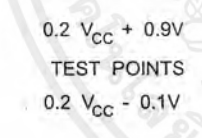
($V_{CC}$ = 5.0 V ± 20%; Load Capacitance = 80 pF)

| 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

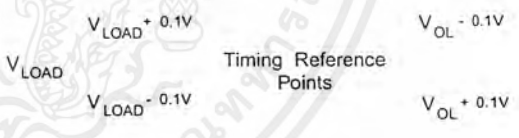



## AC Testing Input/Output Waveforms(1)


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 from the loaded $V_{OH}/V_{OL}$ level occurs.

## Ordering Information

| Speed (MHz) | Power Supply | Ordering Code | Package | Operation Range |
|---|---|---|---|---|
| 12 | 5V ± 20% | AT89C51-12AC | 44A | Commercial (0°C to 70°C) |
| | | AT89C51-12JC | 44J | |
| | | AT89C51-12PC | 40P6 | |
| | | AT89C51-12QC | 44Q | |
| | | AT89C51-12AI | 44A | Industrial (-40°C to 85°C) |
| | | AT89C51-12JI | 44J | |
| | | AT89C51-12PI | 40P6 | |
| | | AT89C51-12QI | 44Q | |
| | | AT89C51-12AA | 44A | Automotive (-40°C to 105°C) |
| | | AT89C51-12JA | 44J | |
| | | AT89C51-12PA | 40P6 | |
| | | AT89C51-12QA | 44Q | |
| 16 | 5V ± 20% | AT89C51-16AC | 44A | Commercial (0°C to 70°C) |
| | | AT89C51-16JC | 44J | |
| | | AT89C51-16PC | 40P6 | |
| | | AT89C51-16QC | 44Q | |
| | | AT89C51-16AI | 44A | Industrial (-40°C to 85°C) |
| | | AT89C51-16JI | 44J | |
| | | AT89C51-16PI | 40P6 | |
| | | AT89C51-16QI | 44Q | |
| | | AT89C51-16AA | 44A | Automotive (-40°C to 105°C) |
| | | AT89C51-16JA | 44J | |
| | | AT89C51-16PA | 40P6 | |
| | | AT89C51-16QA | 44Q | |
| 20 | 5V ± 20% | AT89C51-20AC | 44A | Commercial (0°C to 70°C) |
| | | AT89C51-20JC | 44J | |
| | | AT89C51-20PC | 40P6 | |
| | | AT89C51-20QC | 44Q | |
| | | AT89C51-20AI | 44A | Industrial (-40°C to 85°C) |
| | | AT89C51-20JI | 44J | |
| | | AT89C51-20PI | 40P6 | |
| | | AT89C51-20QI | 44Q | |

MICROCHIP

# PIC16F84A

## 18-pin *Enhanced* Flash/EEPROM 8-Bit Microcontroller

### Devices Included in this Data Sheet:

- PIC16F84A
- Extended voltage range device available (PIC16LF84A)

### High Performance RISC CPU Features:

- Only 35 single word instructions to learn
- All instructions single cycle except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- 1024 words of program memory
- 68 bytes of data RAM
- 64 bytes of data EEPROM
- 14-bit wide instruction words
- 8-bit wide data bytes
- 15 special function hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
  - External RB0/INT pin
  - TMR0 timer overflow
  - PORTB<7:4> interrupt on change
  - Data EEPROM write complete

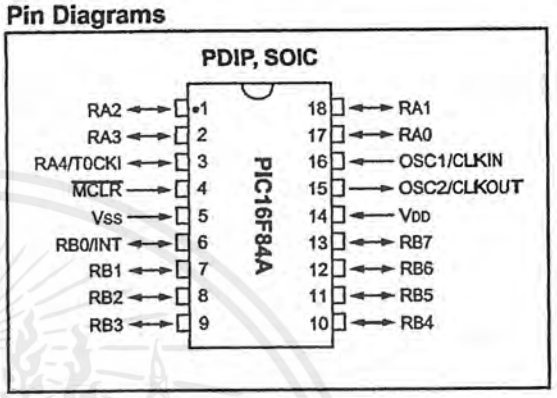
### Peripheral Features:

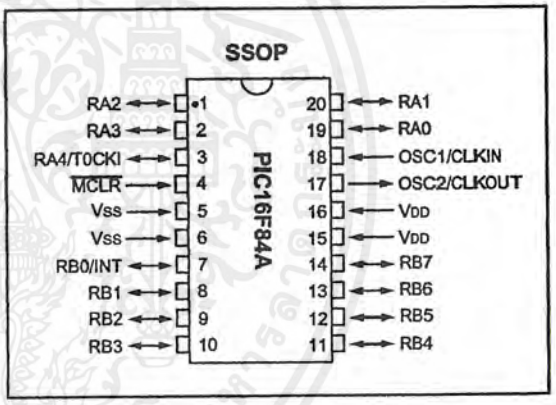
- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - 25 mA sink max. per pin
  - 25 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

### Special Microcontroller Features:

- 1000 erase/write cycles *Enhanced* Flash program memory
- 1,000,000 typical erase/write cycles EEPROM data memory
- EEPROM Data Retention > 40 years
- In-Circuit Serial Programming (ICSP™) - via two pins
- Power-on Reset (POR), Power-up Timer (PWRT), Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Code-protection
- Power saving SLEEP mode
- Selectable oscillator options

### Pin Diagrams




### CMOS *Enhanced* Flash/EERPOM Technology:

- Low-power, high-speed technology
- Fully static design
- Wide operating voltage range:
  - Commercial: 2.0V to 5.5V
  - Industrial: 2.0V to 5.5V
- Low power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 15 µA typical @ 2V, 32 kHz
  - < 0.5 µA typical standby current @ 2V

## Table of Contents



*To Our Valued Customers*

**Most Current Data Sheet**

To obtain the most up-to-date version of this data sheet, please check our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number. e.g., DS30000A is version A of document DS30000.

**Errata**

An errata sheet may exist for current devices, describing minor operational differences (from the data sheet) and recommended workarounds. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (602) 786-7277

When contacting a sales office or the literature center, please specify which device, revision of silicon and data sheet (include literature number) you are using.

**Corrections to this Data Sheet**

We constantly strive to improve the quality of all our products and documentation. We have spent a great deal of time to ensure that this document is correct. However, we realize that we may have missed a few things. If you find any information that is missing or appears in error, please:

- Fill out and mail in the reader response form in the back of this data sheet.
- E-mail us at webmaster@microchip.com.

We appreciate your assistance in making this a better document.

## 1.0 DEVICE OVERVIEW

This document contains device-specific information for the operation of the PIC16F84A device. Additional information may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023), which may be downloaded from the Microchip website. The Reference Manual should be considered a complementary document to this data sheet, and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

The PIC16F84A belongs to the mid-range family of the PICmicro™ microcontroller devices. A block diagram of the device is shown in Figure 1-1.

The program memory contains 1K words, which translates to 1024 instructions, since each 14-bit program memory word is the same width as each device instruction. The data memory (RAM) contains 68 bytes. Data EEPROM is 64 bytes.

There are also 13 I/O pins that are user-configured on a pin-to-pin basis. Some pins are multiplexed with other device functions. These functions include:

- External interrupt
- Change on PORTB interrupt
- Timer0 clock input

Table 1-1 details the pinout of the device with descriptions and details for each pin.

**FIGURE 1-1: PIC16F84A BLOCK DIAGRAM**

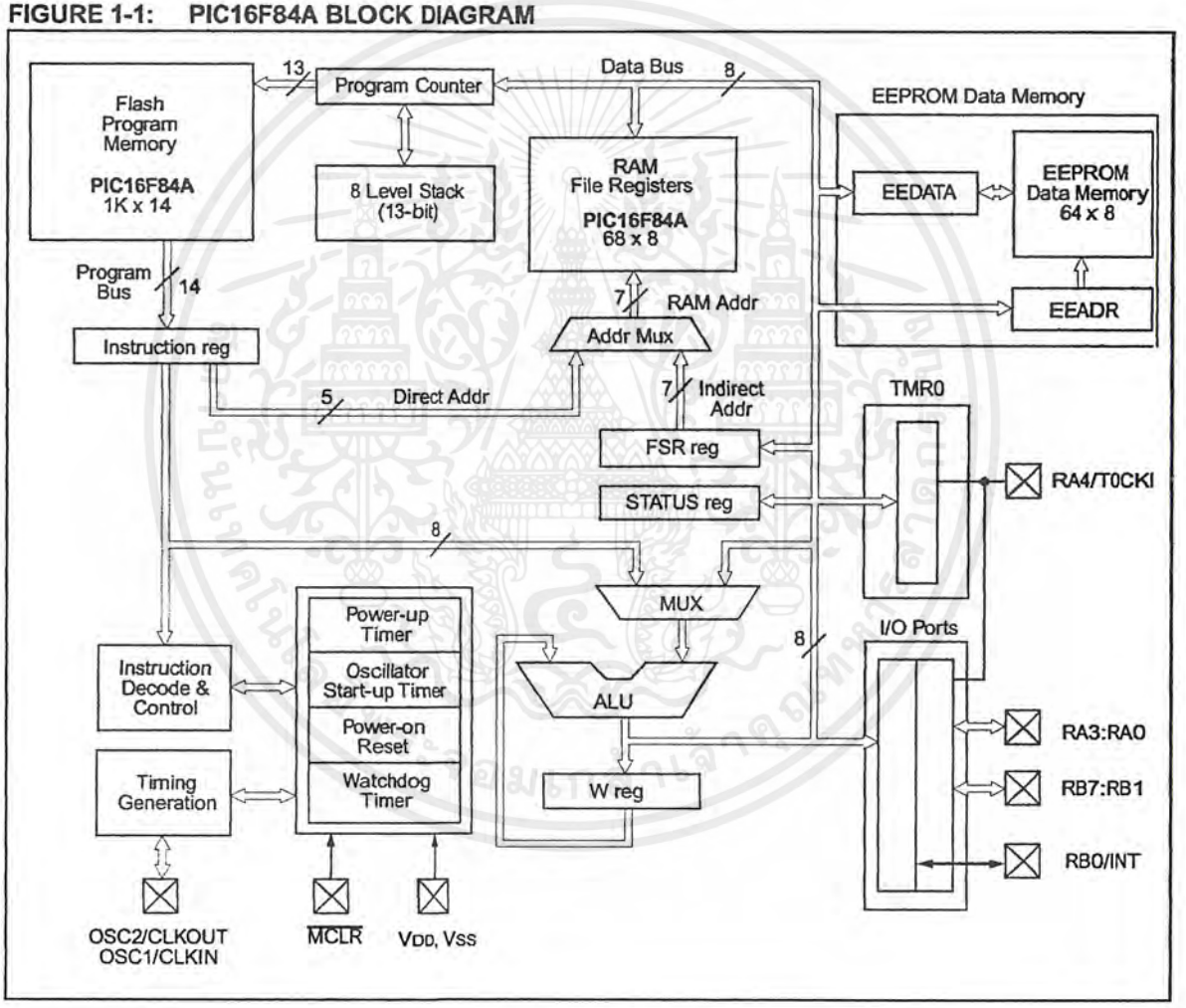


**TABLE 1-1 PIC16F84A PINOUT DESCRIPTION**

| Pin Name | DIP No. | SOIC No. | SSOP No. | I/O/P Type | Buffer Type | Description |
|---|---|---|---|---|---|---|
| OSC1/CLKIN | 16 | 16 | 18 | I | ST/CMOS (3) | Oscillator crystal input/external clock source input. |
| OSC2/CLKOUT | 15 | 15 | 19 | O | — | Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate. |
| $\overline{MCLR}$ | 4 | 4 | 4 | I/P | ST | Master clear (reset) input/programming voltage input. This pin is an active low reset to the device. |
| | | | | | | PORTA is a bi-directional I/O port. |
| RA0 | 17 | 17 | 19 | I/O | TTL | |
| RA1 | 18 | 18 | 20 | I/O | TTL | |
| RA2 | 1 | 1 | 1 | I/O | TTL | |
| RA3 | 2 | 2 | 2 | I/O | TTL | |
| RA4/T0CKI | 3 | 3 | 3 | I/O | ST | Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type. |
| | | | | | | PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs. |
| RB0/INT | 6 | 6 | 7 | I/O | TTL/ST (1) | RB0/INT can also be selected as an external interrupt pin. |
| RB1 | 7 | 7 | 8 | I/O | TTL | |
| RB2 | 8 | 8 | 9 | I/O | TTL | |
| RB3 | 9 | 9 | 10 | I/O | TTL | |
| RB4 | 10 | 10 | 11 | I/O | TTL | Interrupt on change pin. |
| RB5 | 11 | 11 | 12 | I/O | TTL | Interrupt on change pin. |
| RB6 | 12 | 12 | 13 | I/O | TTL/ST (2) | Interrupt on change pin. Serial programming clock. |
| RB7 | 13 | 13 | 14 | I/O | TTL/ST (2) | Interrupt on change pin. Serial programming data. |
| Vss | 5 | 5 | 5,6 | P | — | Ground reference for logic and I/O pins. |
| VDD | 14 | 14 | 15,16 | P | — | Positive supply for logic and I/O pins. |

Legend: I= input O = output I/O = Input/Output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

## 2.0 MEMORY ORGANIZATION

There are two memory blocks in the PIC16F84A. These are the program memory and the data memory. Each block has its own bus, so that access to each block can occur during the same oscillator cycle.

The data memory can further be broken down into the general purpose RAM and the Special Function Registers (SFRs). The operation of the SFRs that control the "core" are described here. The SFRs used to control the peripheral modules are described in the section discussing each individual peripheral module.

The data memory area also contains the data EEPROM memory. This memory is not directly mapped into the data memory, but is indirectly mapped. That is, an indirect address pointer specifies the address of the data EEPROM memory to read/write. The 64 bytes of data EEPROM memory have the address range 0h-3Fh. More details on the EEPROM memory can be found in Section 5.0.

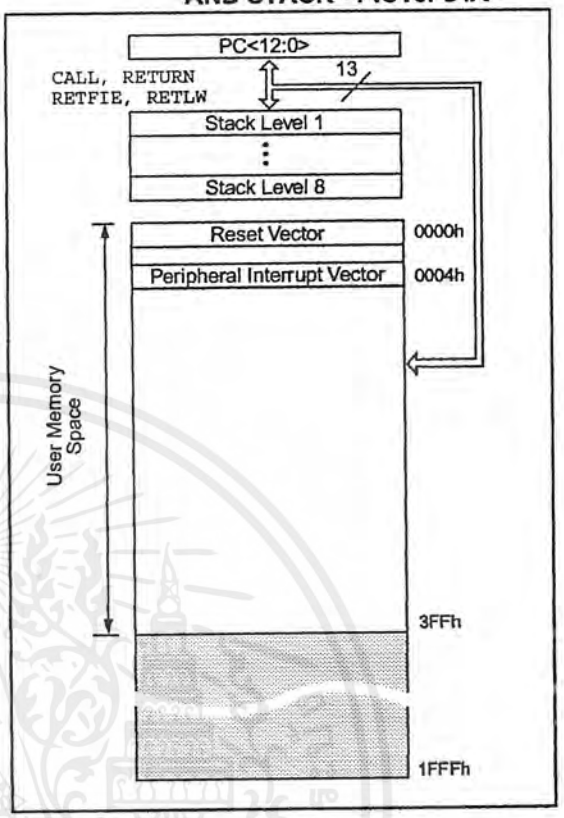
Additional information on device memory may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023).

### 2.1 Program Memory Organization

The PIC16FXX has a 13-bit program counter capable of addressing an 8K x 14 program memory space. For the PIC16F84A, the first 1K x 14 (0000h-03FFh) are physically implemented (Figure 2-1). Accessing a location above the physically implemented address will cause a wraparound. For example, for locations 20h, 420h, 820h, C20h, 1020h, 1420h, 1820h, and 1C20h will be the same instruction.

The reset vector is at 0000h and the interrupt vector is at 0004h.

**FIGURE 2-1: PROGRAM MEMORY MAP AND STACK - PIC16F84A**



## 2.2 Data Memory Organization

The data memory is partitioned into two areas. The first is the Special Function Registers (SFR) area, while the second is the General Purpose Registers (GPR) area. The SFRs control the operation of the device.

Portions of data memory are banked. This is for both the SFR area and the GPR area. The GPR area is banked to allow greater than 116 bytes of general purpose RAM. The banked areas of the SFR are for the registers that control the peripheral functions. Banking requires the use of control bits for bank selection. These control bits are located in the STATUS Register. Figure 2-1 shows the data memory map organization.

Instructions MOVWF and MOVF can move values from the W register to any location in the register file ("F"), and vice-versa.

The entire data memory can be accessed either directly using the absolute address of each register file or indirectly through the File Select Register (FSR) (Section 2.4). Indirect addressing uses the present value of the RP0 bit for access into the banked areas of data memory.

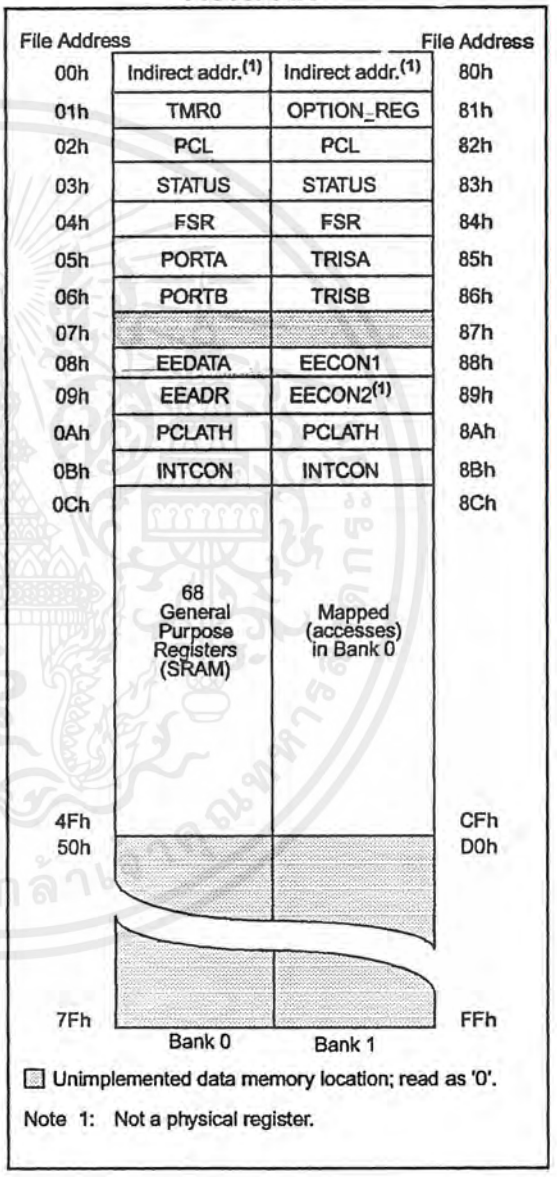
Data memory is partitioned into two banks which contain the general purpose registers and the special function registers. Bank 0 is selected by clearing the RP0 bit (STATUS<5>). Setting the RP0 bit selects Bank 1. Each Bank extends up to 7Fh (128 bytes). The first twelve locations of each Bank are reserved for the Special Function Registers. The remainder are General Purpose Registers implemented as static RAM.

### 2.2.1 GENERAL PURPOSE REGISTER FILE

Each General Purpose Register (GPR) is 8 bits wide and is accessed either directly or indirectly through the FSR (Section 2.4).

The GPR addresses in bank 1 are mapped to addresses in bank 0. As an example, addressing location 0Ch or 8Ch will access the same GPR.

**FIGURE 2-1: REGISTER FILE MAP - PIC16F84A**



### 2.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers (Figure 2-1 and Table 2-1) are used by the CPU and Peripheral functions to control the device operation. These registers are static RAM.

The special function registers can be classified into two sets, core and peripheral. Those associated with the core functions are described in this section. Those related to the operation of the peripheral features are described in the section for that specific feature.

**TABLE 2-1 REGISTER FILE SUMMARY**

| Addr | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on Power-on Reset | Value on all other resets (Note3) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bank 0** | | | | | | | | | | | |
| 00h | INDF | Uses contents of FSR to address data memory (not a physical register) | | | | | | | | ---- ---- | ---- ---- |
| 01h | TMR0 | 8-bit real-time clock/counter | | | | | | | | xxxx xxxx | uuuu uuuu |
| 02h | PCL | Low order 8 bits of the Program Counter (PC) | | | | | | | | 0000 0000 | 0000 0000 |
| 03h | STATUS (2) | IRP | RP1 | RP0 | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | 0001 1xxx | 000q quuu |
| 04h | FSR | Indirect data memory address pointer 0 | | | | | | | | xxxx xxxx | uuuu uuuu |
| 05h | PORTA (4) | — | — | — | RA4/T0CKI | RA3 | RA2 | RA1 | RA0 | ---x xxxx | ---u uuuu |
| 06h | PORTB (5) | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0/INT | xxxx xxxx | uuuu uuuu |
| 07h | | Unimplemented location, read as '0' | | | | | | | | ---- ---- | ---- ---- |
| 08h | EEDATA | EEPROM data register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 09h | EEADR | EEPROM address register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 0Ah | PCLATH | — | — | — | Write buffer for upper 5 bits of the PC (1) | | | | | ---0 0000 | ---0 0000 |
| 0Bh | INTCON | GIE | EEIE | T0IE | INTE | RBIE | T0IF | INTF | RBIF | 0000 000x | 0000 000u |
| **Bank 1** | | | | | | | | | | | |
| 80h | INDF | Uses contents of FSR to address data memory (not a physical register) | | | | | | | | ---- ---- | ---- ---- |
| 81h | OPTION_REG | $\overline{RBPU}$ | INTEDG | T0CS | T0SE | PSA | PS2 | PS1 | PS0 | 1111 1111 | 1111 1111 |
| 82h | PCL | Low order 8 bits of Program Counter (PC) | | | | | | | | 0000 0000 | 0000 0000 |
| 83h | STATUS (2) | IRP | RP1 | RP0 | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | 0001 1xxx | 000q quuu |
| 84h | FSR | Indirect data memory address pointer 0 | | | | | | | | xxxx xxxx | uuuu uuuu |
| 85h | TRISA | — | — | — | PORTA data direction register | | | | | ---1 1111 | ---1 1111 |
| 86h | TRISB | PORTB data direction register | | | | | | | | 1111 1111 | 1111 1111 |
| 87h | | Unimplemented location, read as '0' | | | | | | | | ---- ---- | ---- ---- |
| 88h | EECON1 | — | — | — | EEIF | WRERR | WREN | WR | RD | ---0 x000 | ---0 q000 |
| 89h | EECON2 | EEPROM control register 2 (not a physical register) | | | | | | | | ---- ---- | ---- ---- |
| 0Ah | PCLATH | — | — | — | Write buffer for upper 5 bits of the PC (1) | | | | | ---0 0000 | ---0 0000 |
| 0Bh | INTCON | GIE | EEIE | T0IE | INTE | RBIE | T0IF | INTF | RBIF | 0000 000x | 0000 000u |

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q = value depends on condition.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> is never transferred to PCLATH.

2: The $\overline{TO}$ and $\overline{PD}$ status bits in the STATUS register are not affected by a $\overline{MCLR}$ reset.

3: Other (non power-up) resets include: external reset through $\overline{MCLR}$ and the Watchdog Timer Reset.

4: On any device reset, these pins are configured as inputs.

5: This is the value that will be in the port output latch.

### 2.2.2.1 STATUS REGISTER

The STATUS register contains the arithmetic status of the ALU, the RESET status and the bank select bit for data memory.

As with any register, the STATUS register can be the destination for any instruction. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to device logic. Furthermore, the $\overline{TO}$ and $\overline{PD}$ bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper-three bits and set the Z bit. This leaves the STATUS register as `000u u1uu` (where `u` = unchanged).

Only the `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions should be used to alter the STATUS register (Table 7-2) because these instructions do not affect any status bit.

**Note 1:** The IRP and RP1 bits (STATUS<7:6>) are not used by the PIC16F84A and should be programmed as cleared. Use of these bits as general purpose R/W bits is NOT recommended, since this may affect upward compatibility with future products.

**Note 2:** The C and DC bits operate as a borrow and digit borrow out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

**Note 3:** When the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. The specified bit(s) will be updated according to device logic

**FIGURE 2-1: STATUS REGISTER (ADDRESS 03h, 83h)**

| R/W-0 | R/W-0 | R/W-0 | R-1 | R-1 | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| IRP | RP1 | RP0 | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **IRP**: Register Bank Select bit (used for indirect addressing)
The IRP bit is not used by the PIC16F84A. IRP should be maintained clear.

bit 6-5: **RP1:RP0**: Register Bank Select bits (used for direct addressing)
00 = Bank 0 (00h - 7Fh)
01 = Bank 1 (80h - FFh)
Each bank is 128 bytes. Only bit RP0 is used by the PIC16F84A. RP1 should be maintained clear.

bit 4: $\overline{TO}$: Time-out bit
1 = After power-up, `CLRWDT` instruction, or `SLEEP` instruction
0 = A WDT time-out occurred

bit 3: $\overline{PD}$: Power-down bit
1 = After power-up or by the `CLRWDT` instruction
0 = By execution of the `SLEEP` instruction

bit 2: **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1: **DC**: Digit carry/$\overline{borrow}$ bit (for `ADDWF` and `ADDLW` instructions) (For $\overline{borrow}$ the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0: **C**: Carry/$\overline{borrow}$ bit (for `ADDWF` and `ADDLW` instructions)
1 = A carry-out from the most significant bit of the result occurred
0 = No carry-out from the most significant bit of the result occurred
**Note:** For $\overline{borrow}$ the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low order bit of the source register.

2.2.2.2 OPTION_REG REGISTER

The OPTION_REG register is a readable and writable register which contains various control bits to configure the TMR0/WDT prescaler, the external INT interrupt, TMR0, and the weak pull-ups on PORTB.

**Note:** When the prescaler is assigned to the WDT (PSA = '1'), TMR0 has a 1:1 prescaler assignment.

**FIGURE 2-1: OPTION_REG REGISTER (ADDRESS 81h)**

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| $\overline{\text{RBPU}}$ | INTEDG | T0CS | T0SE | PSA | PS2 | PS1 | PS0 |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **$\overline{\text{RBPU}}$**: PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled (by individual port latch values)

bit 6: **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5: **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4: **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3: **PSA**: Prescaler Assignment bit
1 = Prescaler assigned to the WDT
0 = Prescaler assigned to TMR0

bit 2-0: **PS2:PS0**: Prescaler Rate Select bits

| Bit Value | TMR0 Rate | WDT Rate |
|---|---|---|
| 000 | 1 : 2 | 1 : 1 |
| 001 | 1 : 4 | 1 : 2 |
| 010 | 1 : 8 | 1 : 4 |
| 011 | 1 : 16 | 1 : 8 |
| 100 | 1 : 32 | 1 : 16 |
| 101 | 1 : 64 | 1 : 32 |
| 110 | 1 : 128 | 1 : 64 |
| 111 | 1 : 256 | 1 : 128 |

2.2.2.3 INTCON REGISTER

The INTCON register is a readable and writable register which contains the various enable bits for all interrupt sources.

**Note:** Interrupt flag bits get set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>).

**FIGURE 2-1: INTCON REGISTER (ADDRESS 0Bh, 8Bh)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-x |
|---|---|---|---|---|---|---|---|
| GIE | EEIE | T0IE | INTE | RBIE | T0IF | INTF | RBIF |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **GIE:** Global Interrupt Enable bit
1 = Enables all un-masked interrupts
0 = Disables all interrupts

**Note:** For the operation of the interrupt structure, please refer to Section •.

bit 6: **EEIE:** EE Write Complete Interrupt Enable bit
1 = Enables the EE write complete interrupt
0 = Disables the EE write complete interrupt

bit 5: **T0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4: **INTE:** RB0/INT Interrupt Enable bit
1 = Enables the RB0/INT interrupt
0 = Disables the RB0/INT interrupt

bit 3: **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2: **T0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 has overflowed (must be cleared in software)
0 = TMR0 did not overflow

bit 1: **INTF:** RB0/INT Interrupt Flag bit
1 = The RB0/INT interrupt occurred
0 = The RB0/INT interrupt did not occur

bit 0: **RBIF:** RB Port Change Interrupt Flag bit
1 = When at least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

## 2.3 PCL and PCLATH

The program counter (PC) specifies the address of the instruction to fetch for execution. The PC is 13 bits wide. The low byte is called the PCL register. This register is readable and writable. The high byte is called the PCH register. This register contains the PC<12:8> bits and is not directly readable or writable. All updates to the PCH register go through the PCLATH register.

### 2.3.1 STACK

The stack allows a combination of up to 8 program calls and interrupts to occur. The stack contains the return address from this branch in program execution.

Midrange devices have an 8 level deep x 13-bit wide hardware stack. The stack space is not part of either program or data space and the stack pointer is not readable or writable. The PC is PUSHed onto the stack when a CALL instruction is executed or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not modified when the stack is PUSHed or POPed.

After the stack has been PUSHed eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on).

## 2.4 Indirect Addressing; INDF and FSR Registers

The INDF register is not a physical register. Addressing INDF actually addresses the register whose address is contained in the FSR register (FSR is a *pointer*). This is indirect addressing.

**EXAMPLE 2-1: INDIRECT ADDRESSING**

- Register file 05 contains the value 10h
- Register file 06 contains the value 0Ah
- Load the value 05 into the FSR register
- A read of the INDF register will return the value of 10h
- Increment the value of the FSR register by one (FSR = 06)
- A read of the INDF register now will return the value of 0Ah.

Reading INDF itself indirectly (FSR = 0) will produce 00h. Writing to the INDF register indirectly results in a no-operation (although STATUS bits may be affected).

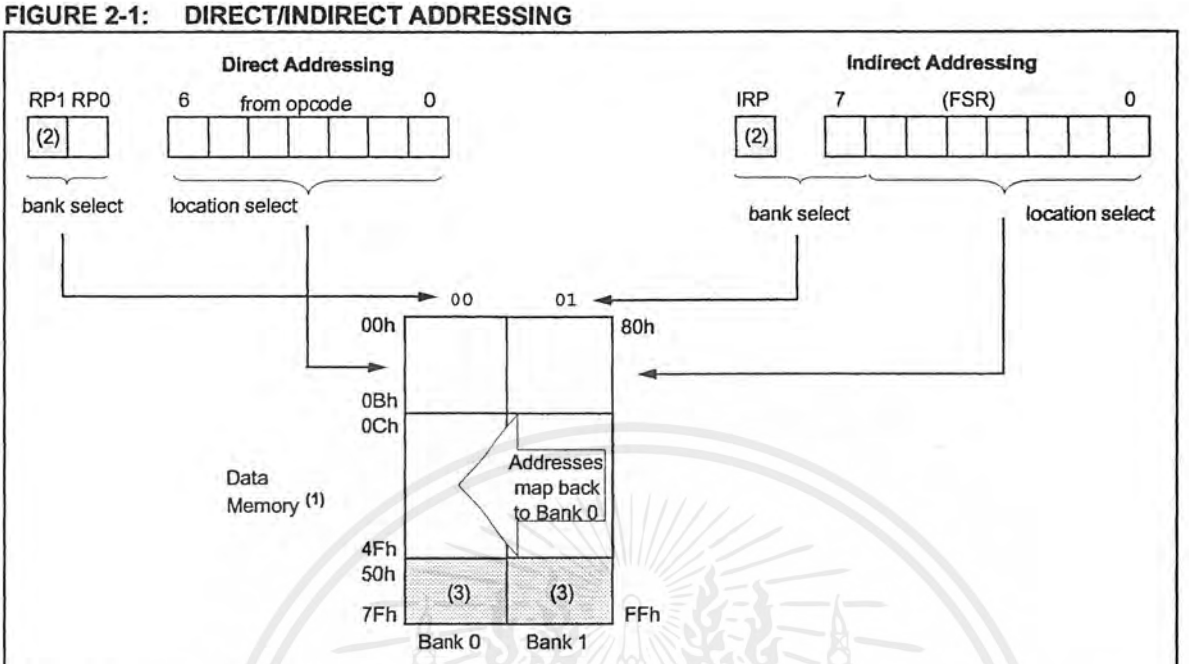
A simple program to clear RAM locations 20h-2Fh using indirect addressing is shown in Example 2-2.

**EXAMPLE 2-2: HOW TO CLEAR RAM USING INDIRECT ADDRESSING**

```
        movlw   0x20    ;initialize pointer
        movwf   FSR     ;  to RAM
NEXT    clrf    INDF    ;clear INDF register
        incf    FSR     ;inc pointer
        btfss   FSR,4   ;all done?
        goto    NEXT    ;NO, clear next
CONTINUE
        :               ;YES, continue
```

An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 2-1. However, IRP is not used in the PIC16F84A.

**FIGURE 2-1: DIRECT/INDIRECT ADDRESSING**




Note 1: For memory map detail see Figure 2-1.
2: Maintain as clear for upward compatiblity with future products.
3: Not implemented.

## 3.0 I/O PORTS

Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

Additional information on I/O ports may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023).

### 3.1 PORTA and TRISA Registers

PORTA is a 5-bit wide bi-directional port. The corresponding data direction register is TRISA. Setting a TRISA bit (=1) will make the corresponding PORTA pin an input, i.e., put the corresponding output driver in a hi-impedance mode. Clearing a TRISA bit (=0) will make the corresponding PORTA pin an output, i.e., put the contents of the output latch on the selected pin.

| **Note:** | On a Power-on Reset, these pins are configured as inputs and read as '0'. |
|---|---|

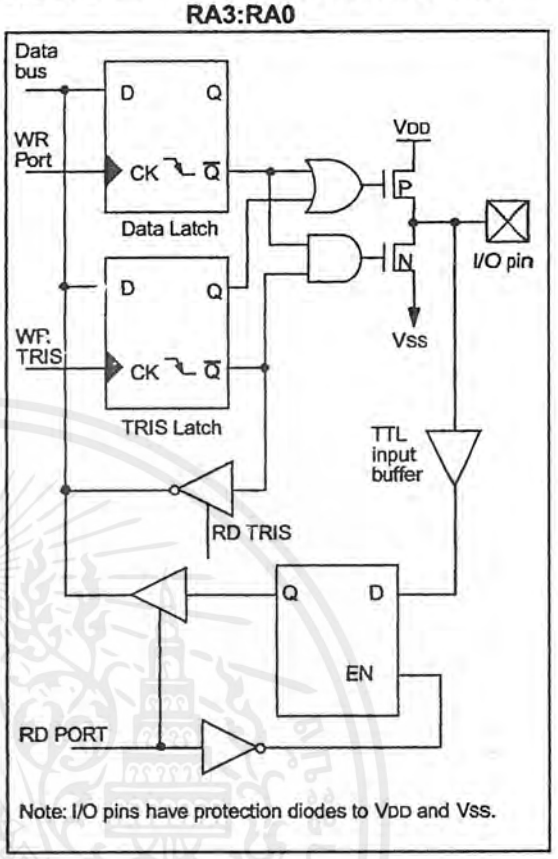
Reading the PORTA register reads the status of the pins whereas writing to it will write to the port latch. All write operations are read-modify-write operations. Therefore a write to a port implies that the port pins are read, this value is modified, and then written to the port data latch.

Pin RA4 is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA4/T0CKI pin is a Schmitt Trigger input and an open drain output. All other RA port pins have TTL input levels and full CMOS output drivers.

**EXAMPLE 3-1: INITIALIZING PORTA**

```
BCF     STATUS, RP0  ;
CLRF    PORTA        ; Initialize PORTA by
                     ; clearing output
                     ; data latches
BSF     STATUS, RP0  ; Select Bank 1
MOVLW   0x0F         ; Value used to
                     ; initialize data
                     ; direction
MOVWF   TRISA        ; Set RA<3:0> as inputs
                     ; RA4 as output
                     ; TRISA<7:5> are always
                     ; read as '0'.
```

**FIGURE 3-1: BLOCK DIAGRAM OF PINS RA3:RA0**




Note: I/O pins have protection diodes to VDD and VSS.

**FIGURE 3-2: BLOCK DIAGRAM OF PIN RA4**

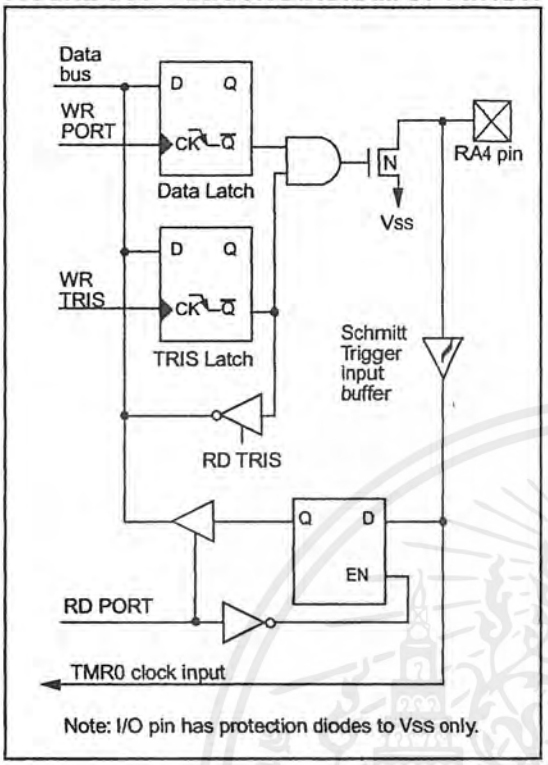



**TABLE 3-1 PORTA FUNCTIONS**

| Name | Bit0 | Buffer Type | Function |
|---|---|---|---|
| RA0 | bit0 | TTL | Input/output |
| RA1 | bit1 | TTL | Input/output |
| RA2 | bit2 | TTL | Input/output |
| RA3 | bit3 | TTL | Input/output |
| RA4/T0CKI | bit4 | ST | Input/output or external clock input for TMR0. Output is open drain type. |

Legend: TTL = TTL input, ST = Schmitt Trigger input

**TABLE 3-2 SUMMARY OF REGISTERS ASSOCIATED WITH PORTA**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on Power-on Reset | Value on all other resets |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 05h | PORTA | — | — | — | RA4/T0CKI | RA3 | RA2 | RA1 | RA0 | ---x xxxx | ---u uuuu |
| 85h | TRISA | — | — | — | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | ---1 1111 | ---1 1111 |

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are unimplemented, read as '0'

### 3.2 PORTB and TRISB Registers

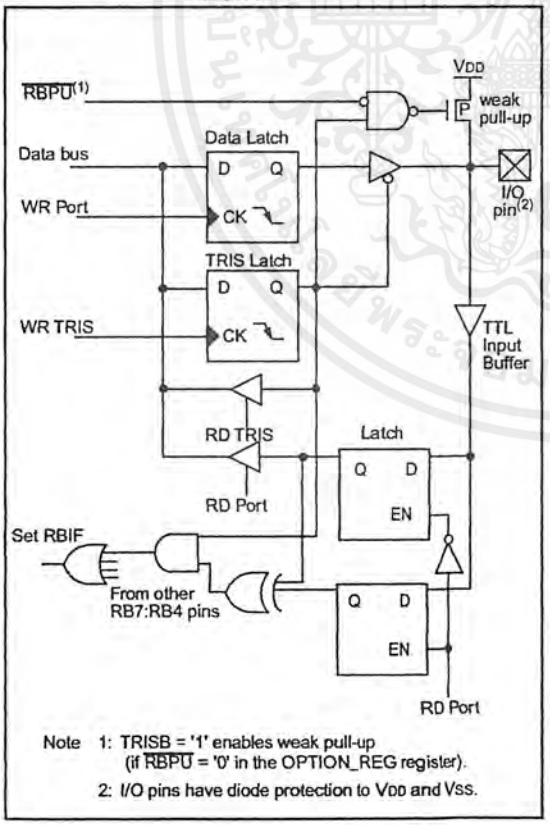
PORTB is an 8-bit wide bi-directional port. The corresponding data direction register is TRISB. Setting a TRISB bit (=1) will make the corresponding PORTB pin an input, i.e., put the corresponding output driver in a hi-impedance mode. Clearing a TRISB bit (=0) will make the corresponding PORTB pin an output, i.e., put the contents of the output latch on the selected pin.

**EXAMPLE 3-1: INITIALIZING PORTB**

```
BCF     STATUS, RP0     ;
CLRF    PORTB           ; Initialize PORTB by
                        ; clearing output
                        ; data latches
BSF     STATUS, RP0     ; Select Bank 1
MOVLW   0xCF            ; Value used to
                        ; initialize data
                        ; direction
MOVWF   TRISB           ; Set RB<3:0> as inputs
                        ; RB<5:4> as outputs
                        ; RB<7:6> as inputs
```

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit $\overline{\text{RBPU}}$ (OPTION<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

**FIGURE 3-3: BLOCK DIAGRAM OF PINS RB7:RB4**




Four of PORTB's pins, RB7:RB4, have an interrupt on change feature. Only pins configured as inputs can cause this interrupt to occur (i.e. any RB7:RB4 pin configured as an output is excluded from the interrupt on change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RB Port Change Interrupt with flag bit RBIF (INTCON<0>).

This interrupt can wake the device from SLEEP. The user, in the interrupt service routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB. This will end the mismatch condition.
b) Clear flag bit RBIF.

A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition, and allow flag bit RBIF to be cleared.

The interrupt on change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt on change feature. Polling of PORTB is not recommended while using the interrupt on change feature.

**FIGURE 3-4: BLOCK DIAGRAM OF PINS RB3:RB0**

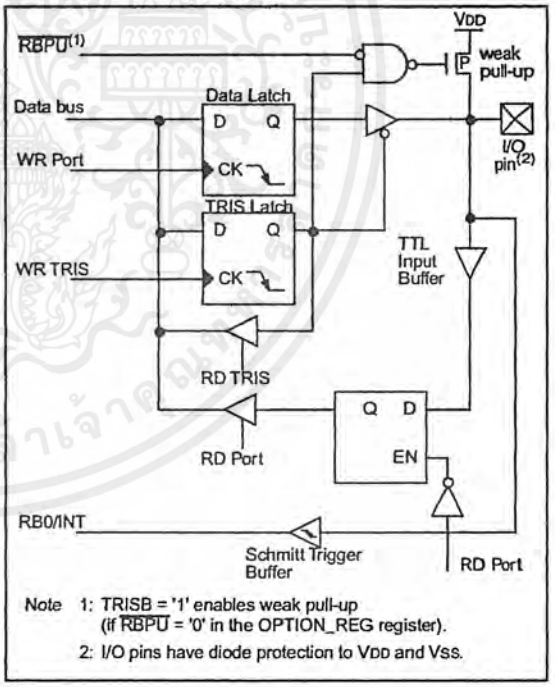


**TABLE 3-3 PORTB FUNCTIONS**

| Name | Bit | Buffer Type | I/O Consistency Function |
|---|---|---|---|
| RB0/INT | bit0 | TTL/ST(1) | Input/output pin or external interrupt input. Internal software programmable weak pull-up. |
| RB1 | bit1 | TTL | Input/output pin. Internal software programmable weak pull-up. |
| RB2 | bit2 | TTL | Input/output pin. Internal software programmable weak pull-up. |
| RB3 | bit3 | TTL | Input/output pin. Internal software programmable weak pull-up. |
| RB4 | bit4 | TTL | Input/output pin (with interrupt on change). Internal software programmable weak pull-up. |
| RB5 | bit5 | TTL | Input/output pin (with interrupt on change). Internal software programmable weak pull-up. |
| RB6 | bit6 | TTL/ST(2) | Input/output pin (with interrupt on change). Internal software programmable weak pull-up. Serial programming clock. |
| RB7 | bit7 | TTL/ST(2) | Input/output pin (with interrupt on change). Internal software programmable weak pull-up. Serial programming data. |

Legend: TTL = TTL input, ST = Schmitt Trigger.
Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.

**TABLE 3-4 SUMMARY OF REGISTERS ASSOCIATED WITH PORTB**

| Addr | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on Power-on Reset | Value on all other resets |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 06h | PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0/INT | xxxx xxxx | uuuu uuuu |
| 86h | TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 1111 1111 | 1111 1111 |
| 81h | OPTION_REG | $\overline{\text{RBPU}}$ | INTEDG | T0CS | T0SE | PSA | PS2 | PS1 | PS0 | 1111 1111 | 1111 1111 |

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.