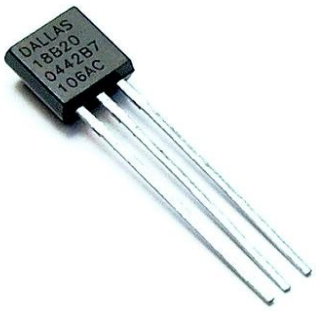
การวัดอุณหภูมิด้วยไอซี DS18B20 Digital Thermometer

สารบัญ



1. บทนำ

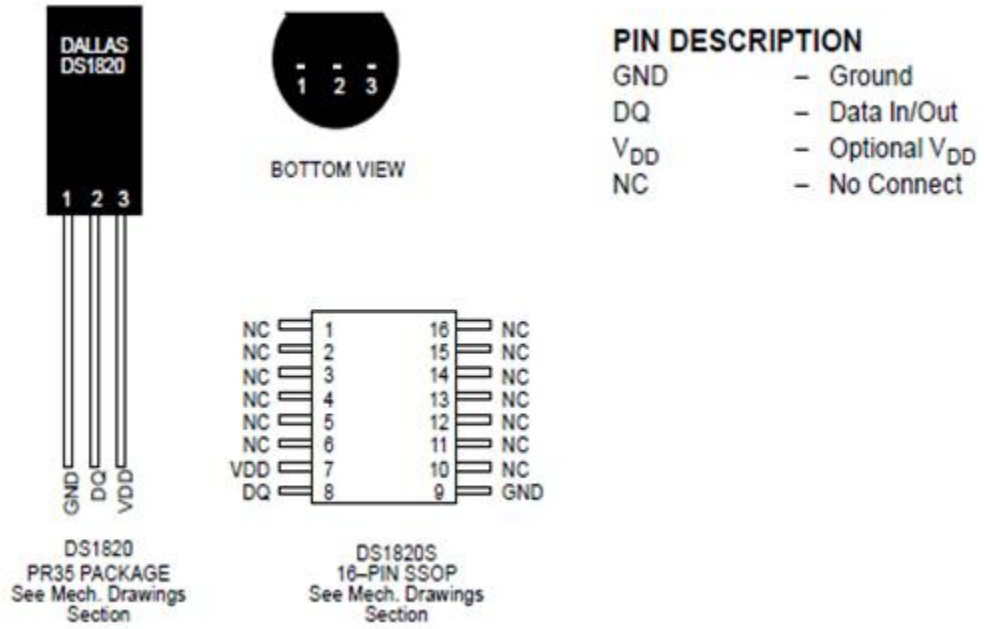
ไอซีสำหรับวัดอุณหภูมิมีอยู่หลายแบบ ถ้าแบ่งตามสัญญาณเอาต์พุตจะแบ่งได้เป็นสองประเภทคือ ไอซีที่ให้เอาต์พุตแบบแอนะล็อก และแบบดิจิตอล โดยไอซีแบบดิจิตอล จะมีวงจรแปลงสัญญาณ ADC (Analog to Digital Converter) รวมอยู่ภายใน บางตระกูล สามารถโปรแกรมการทำงานได้ ในส่วนเอาท์พุท

สำหรับการเชื่อมต่อกับไมโครคอนโทรลเลอร์ มักจะเป็นการสื่อสารข้อมูลแบบที่ใช้ขาสัญญาณจำนวนน้อย เช่น SPI I2C หรือ 1-Wire (OneWire) เพื่อประหยัดขา I/O ในการเชื่อมต่อ ในหัวข้อนี้กล่าวถึง การใช้งานไอซี DS18B20 Digital Thermometer ของบริษัท Dallas ซึ่งใช้ โปรโตคอลสื่อสารแบบ 1-Wire

2. DS1820

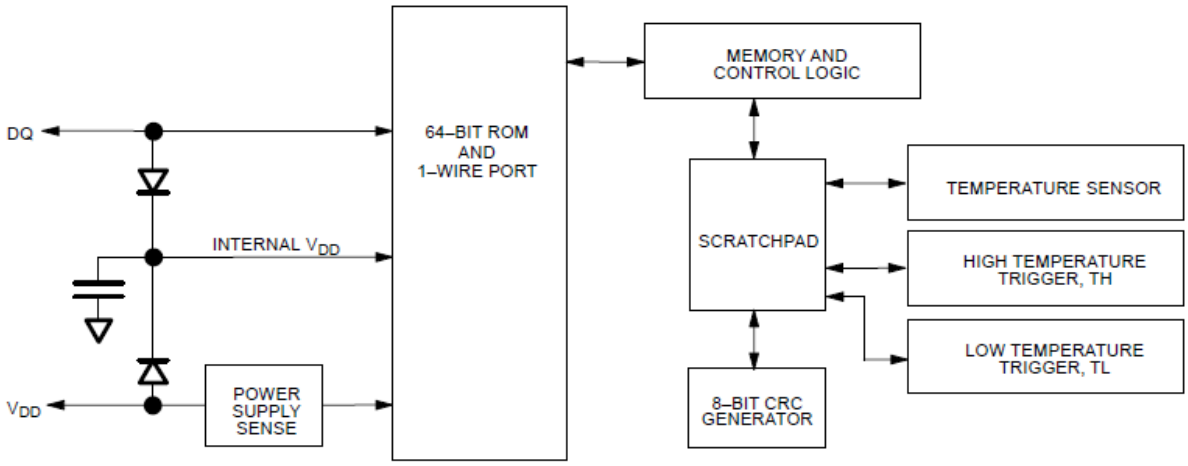
คุณลักษณะทั่วไป

- DS1820 ซึ่งเป็นไอซีดิจิตอลเทอร์โมมิเตอร์แบบโปรแกรมค่าความละเอียด และใช้การอินเตอร์เฟสแบบ 1-Wire และไม่ต้องต่ออุปกรณ์ภายนอก ไอซีตระกูลนี้มีหลายเบอร์ ขึ้นอยู่กับค่าความละเอียด เช่น
    - DS1820 ค่าที่อ่านได้ 9–bit ความละเอียด 0.5 °C
    - DS18B20 ค่าที่อ่านได้ 12–bit ความละเอียด 0.0625 °C
- ใช้แรงดันไฟเลี้ยง Vdd (หรือ Vcc) ได้ในช่วง 3.0V ถึง 5.5V
- ย่านการวัดตั้งแต่ –55°C ถึง +125°C หรือ –67°F ถึง +257°F
- ใช้เวลาการแปลง 200 ms สำหรับข้อมูล 9 บิต และ750 msec สำหรับข้อมูล 12 บิต
- มี 3 ขา (ตัวถังแบบ TO-92) คือ Gnd (Pin 1), DQ (Pin 2), Vdd (Pin 3)
- ใช้งานได้สองแบบ: normal mode (ใช้ทั้ง 3 ขา) และ parasite power mode (ใช้เพียง 2 ขา คือ DQ และ GND ในขณะที่ขา Vdd จะต่อกับขา Gnd)
- สามารถนำไอซีมาพ่วงต่อกันในบัสเดียว (เส้นสัญญาณ DQ) ได้หลายอุปกรณ์
- ในการใช้งาน จะต้องต่อ pull-up 4.7kΩ ที่ขา DQ กับแรงดันไฟเลี้ยง
- ไอซีแต่ละตัวมีหมายเลขเฉพาะตัว ขนาด 64 บิต (64-bit serial code)
- สำหรับตระกูล DS18B20 มีค่ารหัสตระกูลเป็น 28h (0x28) เป็นไบต์แรกของหมายเลขอุปกรณ์



รูปที่ 1 ลักษณะตัวถังของ DS1820

3. บล็อกไดอะแกรมของ DS1820



รูปที่ 2 บล็อกไดอะแกรมของ DS1820

ROM ขนาด 64 บิต

ไอซีตระกูล DS1820 แต่ละตัวจะมีรหัสประจำตัว เก็บไว้ใน ROM ขนาด 64 บิต โดย 8 บิตแรกจะเป็นรหัสของตระกูลย่อยในกลุ่มนี้เช่น

- รหัส 10H เป็นไอซีตระกูล DS18S20 หรือ DS1820
- รหัส 28H เป็นไอซีตระกูล DS18B20
- รหัส 22H เป็นไอซีตระกูล DS18S22

ส่วนที่สองมีขนาด 48 บิต เป็นรหัสประจำตัว (serial number) ของ ไอซีแต่ละตัว ซึ่งจะไม่ซ้ำกันเลย ทำให้สามารถต่อไอซีหลายๆตัว เข้ากับบัส 1-Wire ชุดเดียวกันได้ และเวลาอ่านค่าอุณหภูมิได้ก็ตรวจได้ว่าเป็นค่าของไอซีตัวใด โดยดูจากรหัสนี้

ส่วนสุดท้ายมีขนาด 8 บิต เรียกว่า CRC () ของเลข56บิตที่กล่าวมาแล้ว ตามรูปที่ 3

ข้อมูลขนาด 64 บิตใน ROM		
รหัส CRC ขนาด 8 บิต	Serial Number ขนาด 48 บิต	รหัสตระกูล (เช่น 28H)
MSB LSB	MSB LSB	MSB LSB

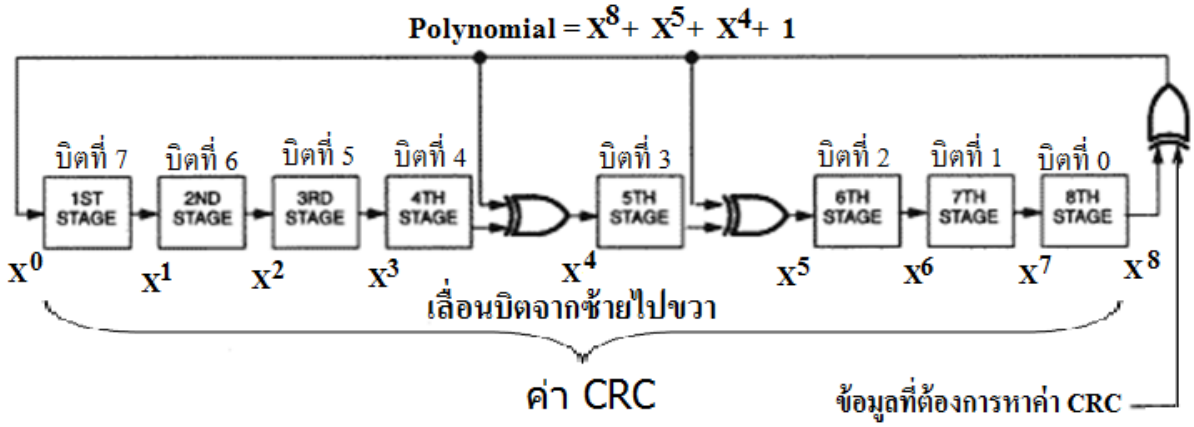
รูปที่ 3 ข้อมูลใน ROM

CRC หรือ cyclic redundancy check เป็นค่าที่ไอซีสร้างขึ้นมาเพื่อให้ตัวอ่านสามารถตรวจสอบได้ว่า ค่าที่อ่านได้จาก ROM นั้นมีข้อผิดพลาดหรือไม่ โดยเมื่อตัวอ่านอ่านข้อมูล 64 บิตนี้ได้ ก็นำไปสร้างค่า CRC ขึ้นมาแล้วนำมาเปรียบเทียบกับ ค่า CRC ที่อ่านออกมาได้ ถ้าไม่ตรงกันแสดงว่าค่าที่อ่านได้มีข้อผิดพลาด

การคำนวณหาค่า CRC ใช้ฟังก์ชั่นโพลิโนเมียล (Polynomial function) ดังนี้

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

โดยหาค่าได้ด้วยการ XORและเลื่อนข้อมูลดังรูปที่ 4 นี้



รูปที่ 4 วิธีการหาค่า CRC

ตัวอย่างการคำนวณค่า CRC และการตรวจสอบ

สมมุติค่าที่อ่านจาก ROM 93 00 00 04 C6 A1 81 28

CRC = 93

Serial No = 00 00 04 C6 A1 81

Famiry = 28

	0	1	2	3	4	5	6	7	A	7	6	5	4	3	2	1	0	
CRC เริ่มต้น	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	28
	0	0	0	0	0	0	0	0	0								0	
	0	0	0	0	0	0	0	0	0								0	
	0	0	0	0	0	0	0	0	1								1	
	1	0	0	0	1	1	0	0	0								0	
	0	1	0	0	0	1	1	0	1								1	
	1	0	1	0	1	1	1	1	1								0	
	1	1	0	1	1	0	1	1	1								0	
	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	81
	0	1	1	1	0	0	0	0	0								0	
	0	0	1	1	1	0	0	0	0								0	
	0	0	0	1	1	1	0	0	0								0	
	0	0	0	0	1	1	1	0	0								0	
	0	0	0	0	0	1	1	1	1								0	
	1	0	0	0	1	1	1	1	1								0	
	1	1	0	0	1	0	1	1	0								1	
	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0	1	A1
	0	0	1	1	0	0	1	0	0								0	
	0	0	0	1	1	0	0	1	1								0	
	1	0	0	0	0	0	0	0	0								0	
	0	1	0	0	0	0	0	0	0								0	
	0	0	1	0	0	0	0	0	1								1	
	1	0	0	1	1	1	0	0	0								0	
	0	1	0	0	1	1	1	0	1								1	
	1	0	1	0	1	0	1	1	1	1	1	0	0	0	1	1	0	C6

บิตที่ 7 CRC = 0 XOR กับ บิตที่ 0 ของ 28 คือ 0 ได้ผลลัพท์เป็น 0 (ให้เป็น A)

บิตที่ 1,2,3, 6, 7 ของ CRC รอบถัดไปได้จากบิตที่ 0,1,2,5,6 จากตัวเดิม

บิตที่ 0 ของ CRC รอบถัดไป = A

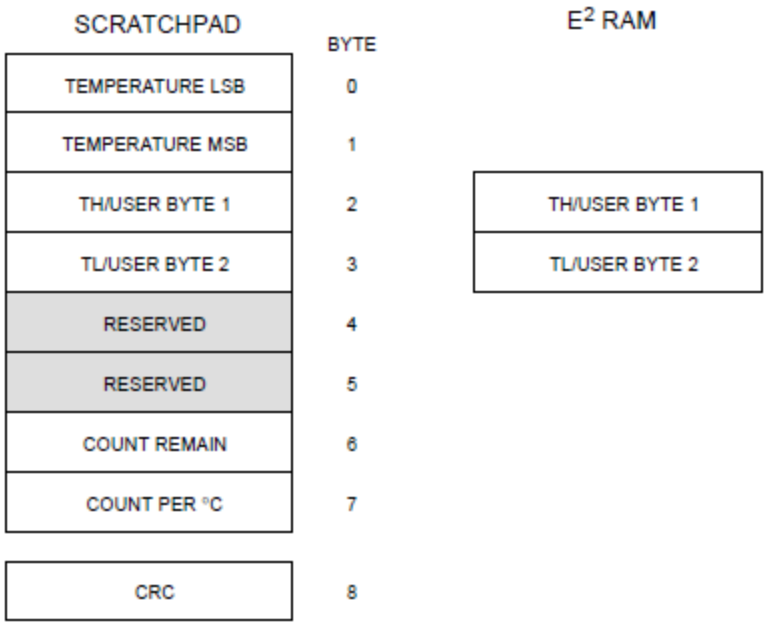
บิตที่ 4 ของ CRC รอบถัดไป = A xor บิตที่ 3

บิตที่ 5 ของ CRC รอบถัดไป = A xor บิตที่ 4

		1	0	1	0	1	0	1	1	1		1	1	0	0	0	1	1	0		C6	
		1	1	0	1	1	0	0	1	0									1			
		0	1	1	0	1	1	0	0	1									1			
		1	0	1	1	1	0	1	0	0									0			
		0	1	0	1	1	1	0	1	1									0			
		1	0	1	0	0	0	1	0	0									0			
		0	1	0	1	0	0	0	1	0									1			
		0	0	1	0	1	0	0	0	1									1			
		1	0	0	1	1	0	0	0	0		0	0	0	0	0	1	0	0		04	
		0	1	0	0	1	1	0	0	0									0			
		0	0	1	0	0	1	1	0	1									1			
		1	0	0	1	1	1	1	1	1									0			
		1	1	0	0	0	0	1	1	1									0			
		1	1	1	0	1	1	0	1	1									0			
		1	1	1	1	1	0	1	0	0									0			
		0	1	1	1	1	1	0	1	1									0			
		1	0	1	1	0	0	1	0	0		0	0	0	0	0	0	0	0		00	
		0	1	0	1	1	0	0	1	1									0			
		1	0	1	0	0	0	0	0	0									0			
		0	1	0	1	0	0	0	0	0									0			
		0	0	1	0	1	0	0	0	0									0			
		0	0	0	1	0	1	0	0	0									0			
		0	0	0	0	1	0	1	0	0									0			
		0	0	0	0	0	1	0	1	1									0			
		1	0	0	0	1	1	1	0	0		0	0	0	0	0	0	0	0		00	
		0	1	0	0	0	1	1	1	1									0			
		1	0	1	0	1	1	1	1	1									0			
		1	1	0	1	1	0	1	1	1									0			
		1	1	1	0	0	0	0	1	1									0			
		1	1	1	1	1	1	0	0	0									0			
		0	1	1	1	1	1	1	0	0									0			
		0	0	1	1	1	1	1	1	1									0			
		1	**0**	**0**	**1**	**0**	**0**	**1**	**1**	0		1	0	0	1	0	0	1	1		93	
		0	1	0	0	1	0	0	1	0									1			
		0	0	1	0	0	1	0	0	0									0			
		0	0	0	1	0	0	1	0	0									0			
		0	0	0	0	1	0	0	1	0									1			
		0	0	0	0	0	1	0	0	0									0			
		0	0	0	0	0	0	1	0	0									0			
		0	0	0	0	0	0	0	1	0									1			
		0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	1												

Temperature Sensor เป็นตัวตรวจวัดอุณหภูมิ

High Temperature Trigger, TH และ Low Temperature Trigger, TL เป็นค่าอุณหภูมิสูงสุด และค่าอุณหภูมิต่ำสุด ที่ผู้ใช้กำหนดไว้ ค่าทั้งสองนี้เก็บอยู่ใน E2 RAM เมื่อทำงาน DS1820 จะนำมาเก็บไว้ที่ ไบต์ที่ 2 และ 3 ของ Scratchpad ตามรูปที่ 5



รูปที่ 5 การเก็บข้อมูลของหน่วยความจำ

หน่วยความจำ หน่วยความจำภายใน DS1820 ประกอบด้วยกัน 2 ส่วน คือ **Scratchpad** และ **E^2 RAM**

- **E^2 RAM** เป็นหน่วยความจำที่ไม่ต้องการไฟเลี้ยง (Nonvolatile memory) ใช้เก็บค่า T_H และ T_L ของผู้ใช้
- **Scratchpad** เป็นหน่วยความจำใช้เก็บค่าต่างๆเวลาทำงาน โดย
  - ไบต์ที่ 0 และ ไบต์ที่ 1 เก็บค่าอุณหภูมิที่อ่านได้
  - ไบต์ที่ 2 และ ไบต์ที่ 3 ใช้เก็บค่า T_H และ T_L ที่อ่านเอามาจาก **E^2 RAM**
  - ไบต์ที่ 4 และ ไบต์ที่ 5 สำรอง ปกติจะอ่านได้เป็นลอจิก 1 ทั้งหมด
  - ไบต์ที่ 6 และ ไบต์ที่ 7 เป็นรีจิสเตอร์ตัวนับ ใช้สำหรับเพิ่มค่าความละเอียด
  - ไบต์ที่ 8 เป็นค่า CRC ของข้อมูล 8 ไบต์ข้างต้น

การอ่านค่าอุณหภูมิที่ได้จาก DS1820

ค่าที่อ่านได้เป็นเลข 2 ‘complement 9 บิต

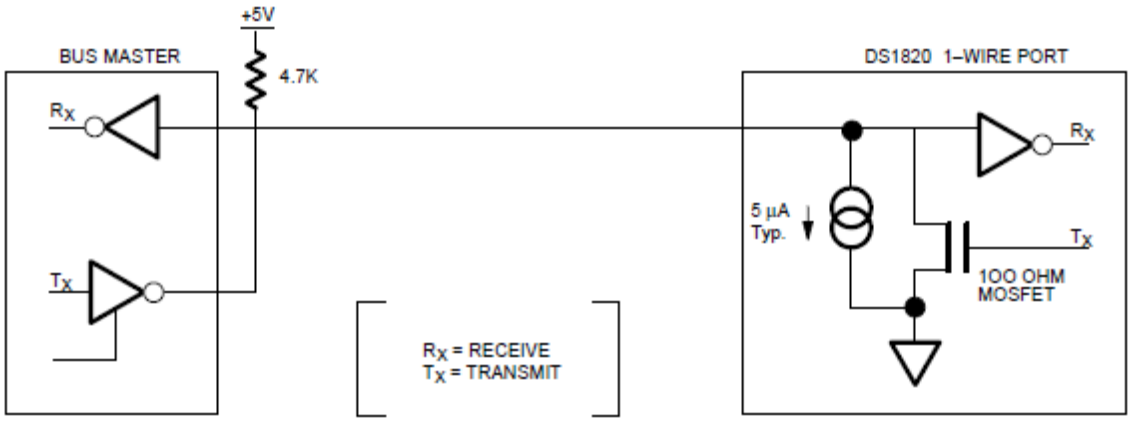
อุณหภูมิ	ค่าที่อ่านได้		วิธีแปลง	
TEMPERATURE	DIGITAL OUTPUT(Binary)	DIGITAL OUTPUT (Hex)	แปลงเป็นเลขฐานสิบ	หารด้วย 2
+125°C	00000000 11111010	00FA	250	+125
+25°C	00000000 00110010	0032h	50	+25
+1/2°C'	00000000 00000001	0001h	1	+1/2
+0°C	00000000 00000000	0000h	0	+0
–1/2°C	11111111 11111111	FFFFh	-1	-1/2
–25°C	11111111 11001110	FFCEh	-50	-25
–55°C	11111111 10010010	FF92h	-110	-55
FF92h = 1111 1111 1001 0010 บิตซ้ายมือสุดเป็น 1 แสดงว่าเป็นเลขลบ				
FF92h เป็นเลขลบ แปลงเป็น 2'scomplement = 006E = 110 แสดงว่าเป็น -110				

การอ่านค่าอุณหภูมิที่ได้จาก DS18B20

ค่าที่อ่านได้เป็นเลข 2 ‘complement 9 บิต ถึง 12 บิต

อุณหภูมิ	ค่าที่อ่านได้		วิธีแปลง	
TEMPERATURE	DIGITAL OUTPUT(Binary)	DIGITAL OUTPUT (Hex)	แปลงเป็นเลขฐานสิบ	หารด้วย16
+125°C	0000 0111 1101 0000	07D0h	2000	+125
+25.0625°C	0000 0001 1001 0010	0191h	401	+25.0625
+10.125°C	0000 0000 1010 0010	00A2h	162	+10.125
+1/2°C'	0000 0000 0000 1000	0008h	8	+1/2
+0°C	0000 0000 0000 0000	0000h	0	+0
–1/2°C	1111 1111 1111 1000	FFF8h	-8	-1/2
-10.125	1111 1111 0101 1110	FF5Eh	-162	-10.125
–25.0625°C	1111 1110 0110 1111	FE6Fh	-401	-25.0625
–55°C	1111 1100 1001 0010	FC90h	-880	-55
FC90 เป็นเลขลบ แปลงเป็น 2'scomplement = 0370 = -880 เมื่อหารด้วย 16 เท่ากับ -55				

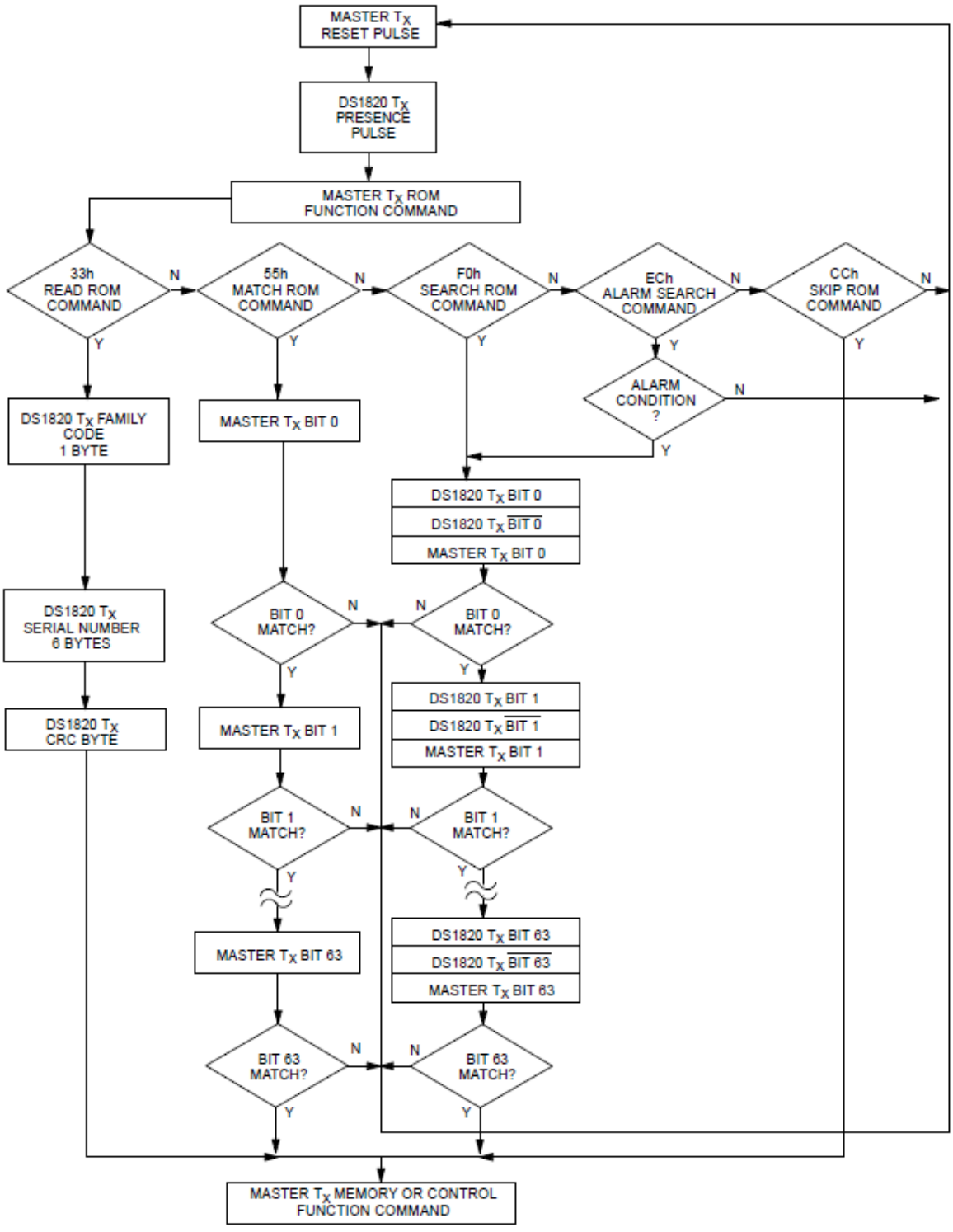
4. การทำงานของ DS1820



รูปที่ 6 การเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับ DS1820

การใช้งาน DS1820 ต้องต่ออยู่กับอุปกรณ์ควบคุม เช่นไมโครคอนโทรลอร์ ผ่านระบบบัส 1-Wire ซึ่งระบบนี้จะจัดให้อุปกรณ์ควบคุมเป็นตัวมาสเตอร์ (Master) ตามรูปที่ 6 การทำงานทุกอย่าตัวมาสเตอร์ ต้องเป็นผู้ส่งเช่นถ้าต้องการค่าอุณหภูมิ มาสเตอร์ตั้งส่งคำสั่งไปบอกให้ DS1820 ส่งค่าอุณหภูมิมาให้ เมื่อได้ค่าแล้ว จะทำอะไรต่อ มาสเตอร์ก็ต้องส่งคำสั่งไปใหม่ การทำงานแบ่งเป็น 2 แบบหลักๆ

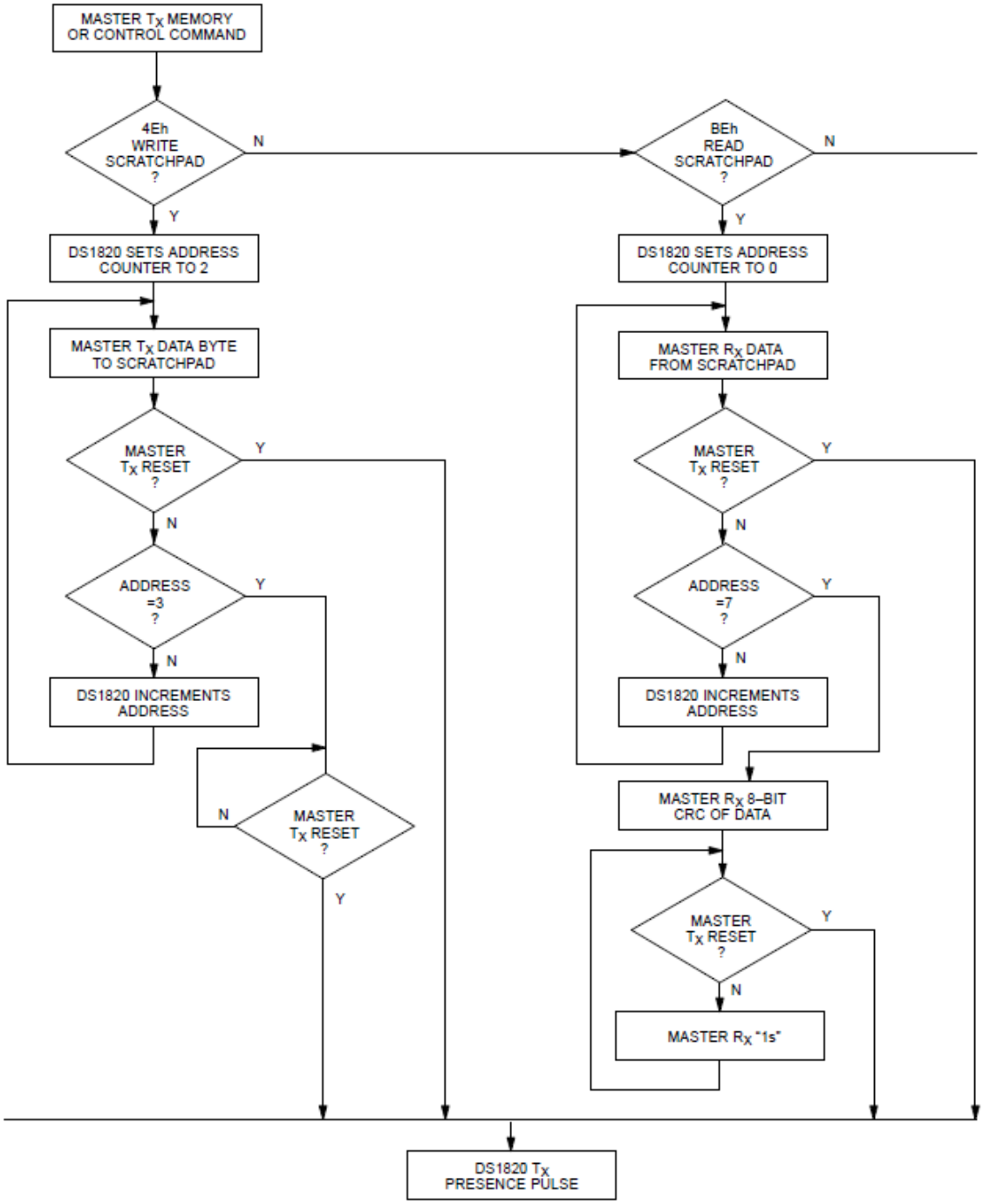
- การทำงานเกี่ยวกับ ROM มีโฟลว์ชาร์ต ตามรูปที่ 7
- การทำงานเกี่ยวกับหน่วยความจำ มีโฟลว์ชาร์ต ตามรูปที่ 8



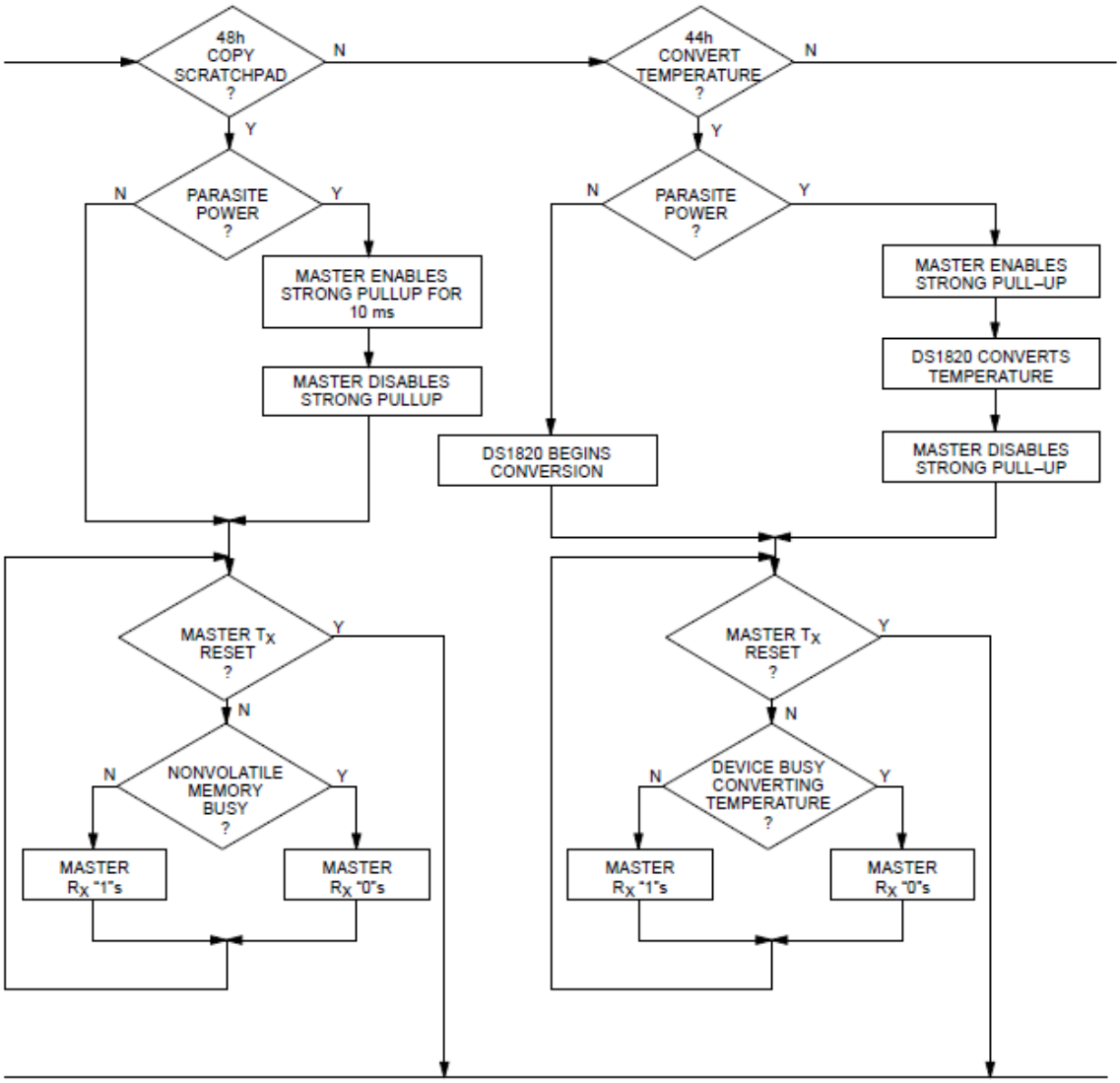
รูปที่ 7 โฟลว์ชาร์ตการทำงานเกี่ยวกับ ROM

ตามรูปที่ 7 นี้ เริ่มต้นการทำงานมาสเตอร์ต้องส่งสัญญานพัลส์เพื่อรีเซ็ต DS1820 หลังจากนั้น จะทำอะไรเกี่ยวกับ ROM ก็ส่งคำสั่งไปให้ DS1820 เช่น ถ้าต้องการอ่าน รหัสตระกูล serial number และ CRC ก็ส่งคำสั่ง 33H ไปที่ DS1820 หลังจากนั้น มาสเตอร์ก็รอรับข้อมูลทั้ง 8 ไบต์ ที่ DS1820 จะส่งออกมา

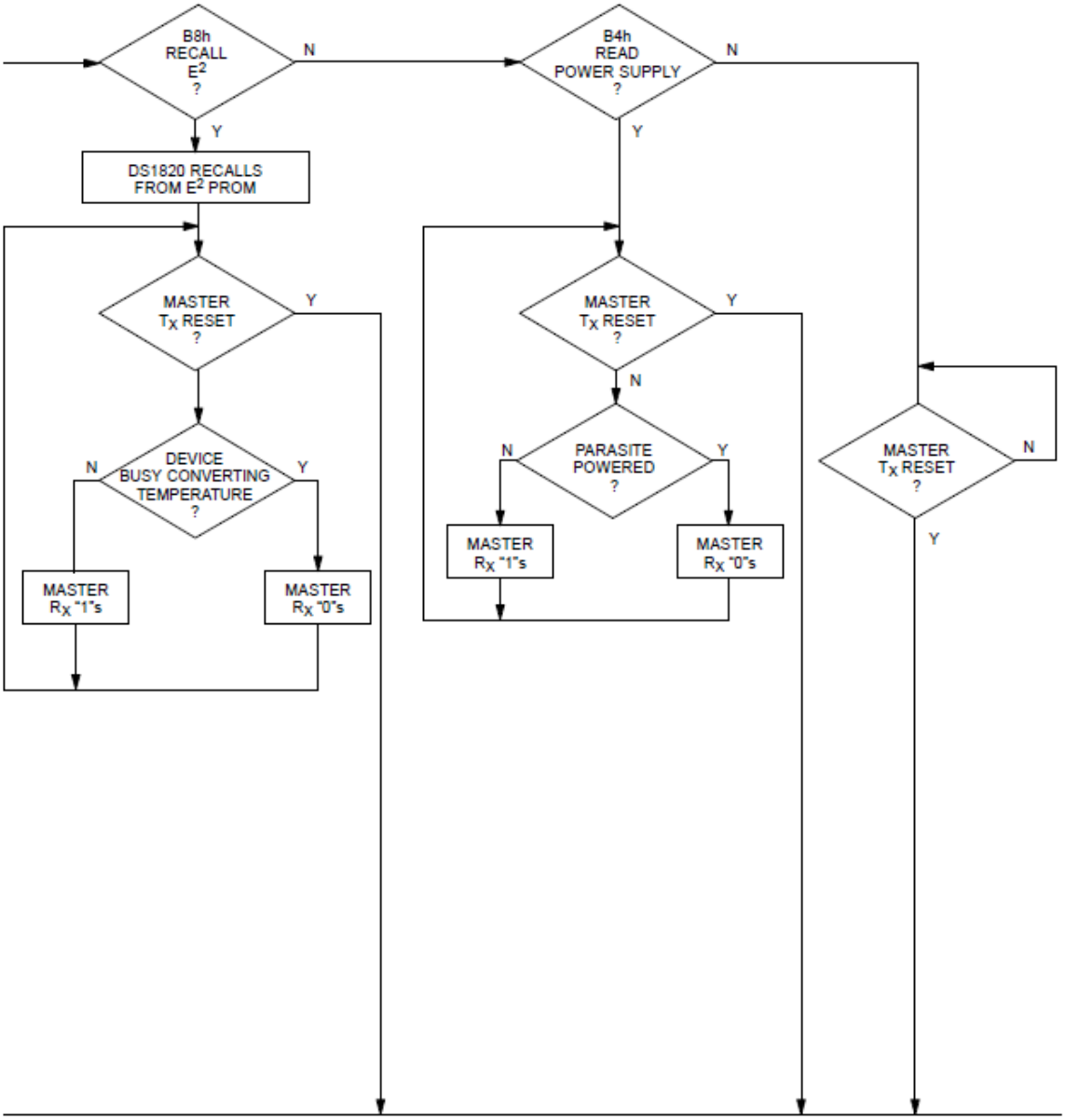
ส่วนการทำงานที่เกี่ยวกับหน่วยความจำก็เช่นเดียวกัน สามารถดูคำสั่งได้จาก โฟลว์ชาร์ตในรูปที่ 8



รูปที่ 8 โฟลว์ชาร์ตการทำงานเกี่ยวกับหน่วยความจำ



รูปที่ 8 โฟลว์ชาร์ตการทำงานเกี่ยวกับหน่วยความจำ (ต่อ)



รูปที่ 8 โฟลว์ชาร์ตการทำงานเกี่ยวกับหน่วยความจำ (ต่อ)

5. การสื่อสารแบบ 1-Wire

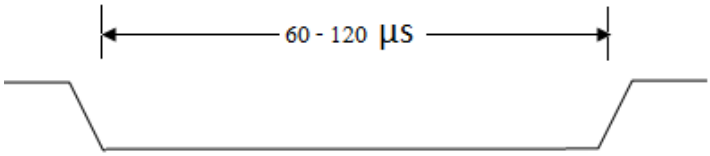
การสื่อสารแบบ 1-Wire จะกระทำผ่านสาย Data เพียงเส้นเดียวดังนั้นทั้งมาสเตอร์และ DS1820 ต้องทำหน้าที่เป็นทั้งผู้รับและผู้ส่ง เพียงแต่ต้องผลัดกันรับ-ผลัดกันส่ง ไม่สามารถทำพร้อมกันได้ ดังนั้นปัจจัยที่ต้องคำนึงถึงคือ

- เวลาตรงตามข้อกำหนด
- มาสเตอร์ต้องเป็นอุปกรณ์เริ่มต้นการสื่อสาร ด้วยสัญญาณรีเซ็ต

1) สัญญาณ มาสเตอร์ Write ลอจิก 1 มาสเตอร์ต้องดึงให้ขา Data เป็นลอจิก 0 ระหว่าง 1 ถึง 15 µs แล้วก็ปล่อยให้ขา Data กลับเป็น 1 จนกว่าจะครบเวลา ตามรูปที่ 9

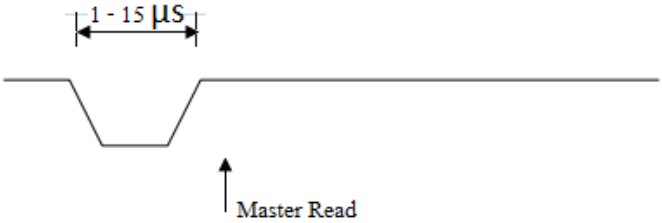
รูปที่ 9 สัญญาณ Write สำหรับ ลอจิก 1

2) สัญญาณมาสเตอร์ Write ลอจิก 0 มาสเตอร์ต้องดึงให้ขา Data เป็นลอจิก 0 อย่างน้อยสุด 60 µs แต่ไม่เกิน 120 µs แล้วก็ปล่อยให้ขา Data กลับเป็น 1 จนกว่าจะครบเวลา ตามรูปที่ 10



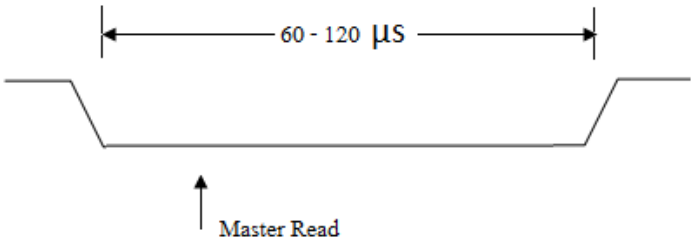
รูปที่ 10 สัญญาณ Write สำหรับ ลอจิก 0

3) สัญญาณมาสเตอร์ Read ลอจิก 1



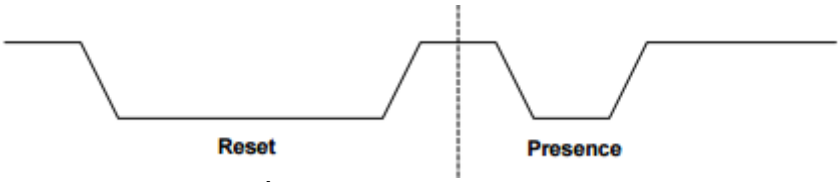
รูปที่ 11 มาสเตอร์อ่านได้ลอจิก 1

4) สัญญาณมาสเตอร์ Read ลอจิก 0



รูปที่ 12 มาสเตอร์อ่านได้ลอจิก 0

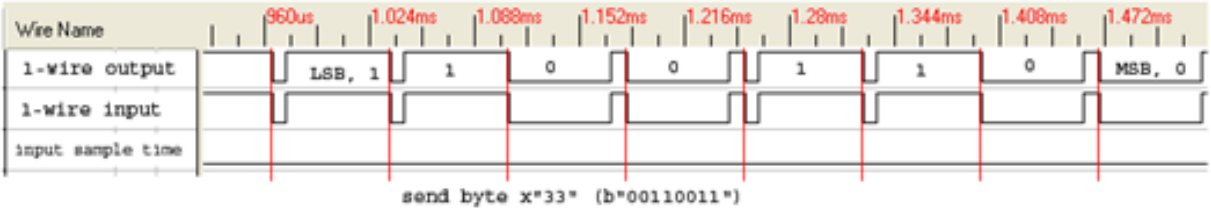
5) สัญญาณรีเซ็ต และ Presence เมื่อมาสเตอร์ ต้องการจะรีเซ็ต DS1820 มาสเตอร์จะทำให้สัญญาณ Data เป็น 0 ไม่น้อยกว่า 8 time slot หรือ 480 µs แล้วก็ปล่อยให้สัญญาณ Data กลับเป็น 1 ช่วงนี้เรียกว่ารีเซ็ตอุปกรณ์ ถ้ามีอุปกรณ์ต่ออยู่ อุปกรณ์ต้องตอบกลับโดยทำให้ขา Data เป็น 0 ภายใน 60 µs และเป็น 0 ไม่น้อยกว่า 60 µs ช่วงนี้เรียกว่า Presence ถ้าไม่มีสัญญาณ Presence ตอบกลับ Master ก็จะทราบได้ว่าไม่มีอุปกรณ์ต่ออยู่



รูปที่ 11 สัญญาณ Reset และ Presence

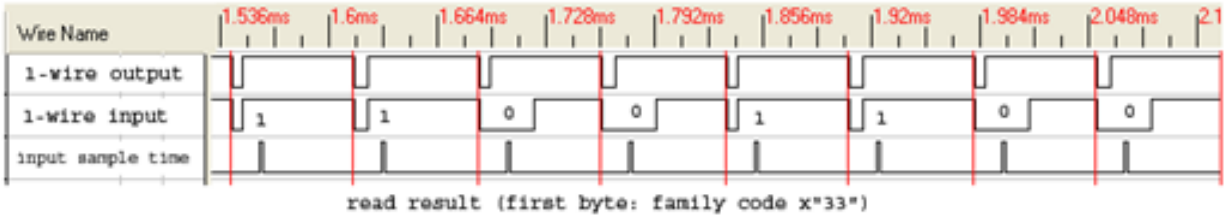
หมายเหตุ ขนาดเวลาของ Time slot ของอุปกรณ์แต่ละชนิดไม่เท่ากัน

ตัวอย่างมาสเตอร์ส่งข้อมูล b00110011



รูปที่ 12 ตัวอย่างการส่งสัญญาณ b00110011

ตัวอย่างมาสเตอร์อ่านข้อมูล 33H (b00110011)



รูปที่ 13 ตัวอย่างการอ่านสัญญาณ b00110011

6. ฟังก์ชั่นใน ไลบรารี่ Onewire สื่อสารแบบ 1-Wire สำหรับ Arduino

ไลบรารี่ Onewire สามารถดาวน์โหลดได้จาก http://playground.arduino.cc/Learning/OneWire

- OneWire myWire(pin)
- myWire.search(addrArray)
- myWire.reset_search()
- myWire.reset()
- myWire.select(addrArray)
- myWire.skip()
- myWire.write(num);
- myWire.write(num, 1);
- myWire.read()
- myWire.crc8(dataArray, length)

OneWire myWire(pin)

สร้างออปเจ็ค สำหรับ Onewire โดยการกำหนดตำแหน่งขา Data ของสัญญาณ Onewire เช่น

OneWire ds(11);

สร้างออปเจ็ค Onewire ชื่อว่า ds ต่ออยู่กับ D11 ของ Arduino

ขาหนึ่งขานี้สามารถต่อกับอุปกรณ์ได้หลายๆตัว แต่ถ้ามีอุปกรณ์มาก สามารถแบ่งอุปกรณ์เป็นกลุ่มๆ แต่ละกลุ่มก็ใช้ขาสัญญาณ 1 ขา เป็นการสร้างออปเจ็คขึ้นหลายตัว

myWire.search(addrArray)

ใช้สำหรับค้นหาอุปกรณ์ 1-Wire ตัวต่อไป ถ้าพบเมื่อออกจากฟังก์ชั่นจะให้ค่าเป็นจริง พร้อมกับใส่ค่าแอดเดรส ลงในตัวแปรอะเรย์ **addrArray** จำนวน 8 ไบต์ แต่ถ้าไม่มีอุปกรณ์ตัวต่อไปจะให้ค่าเป็นเท็จ

myWire.reset_search()

เริ่มต้นค้นหาอุปกรณ์ 1-Wire ใหม่ เมื่อพบจะนับเป็นอุปกรณ์ตัวแรก

myWire.reset()

เป็นฟังก์ชั่นสำหรับสั่งรีเซ็ตอุปกรณ์ 1-Wire โดยปกติใช้เมื่อจะเริ่มการสื่อสารกับอุปกรณ์ 1-Wire

myWire.select(addrArray)

ฟังก์ชั่นสำหรับเลือกสื่อสารกับอุปกรณ์ 1-Wire ตามค่าแอดเดรส ที่เก็บอยู่ในอะเรย์ **addrArray** ขนาด 8 ไบต์

myWire.skip()

ฟังก์ชั่นสำหรับการไม่เลือกอุปกรณ์ ฟังก์ชั่นนี้ทำงานได้ในกรณีที่ในระบบมีอุปกรณ์เพียงตัวเดียว

myWire.write(num)

ส่งคำสั่งหรือข้อมูลไปที่อุปกรณ์ 1-Wire 1 ไบต์

myWire.write(num, 1)

ส่งคำสั่งหรือข้อมูลไปที่อุปกรณ์ 1-Wire 1 ไบต์ แล้วคงไฟเลี้ยงไว้ที่บัส 1-Wire

myWire.read()

อ่านข้อมูลจากอุปกรณ์ 1-Wire 1 ไบต์

myWire.crc8(dataArray, length)

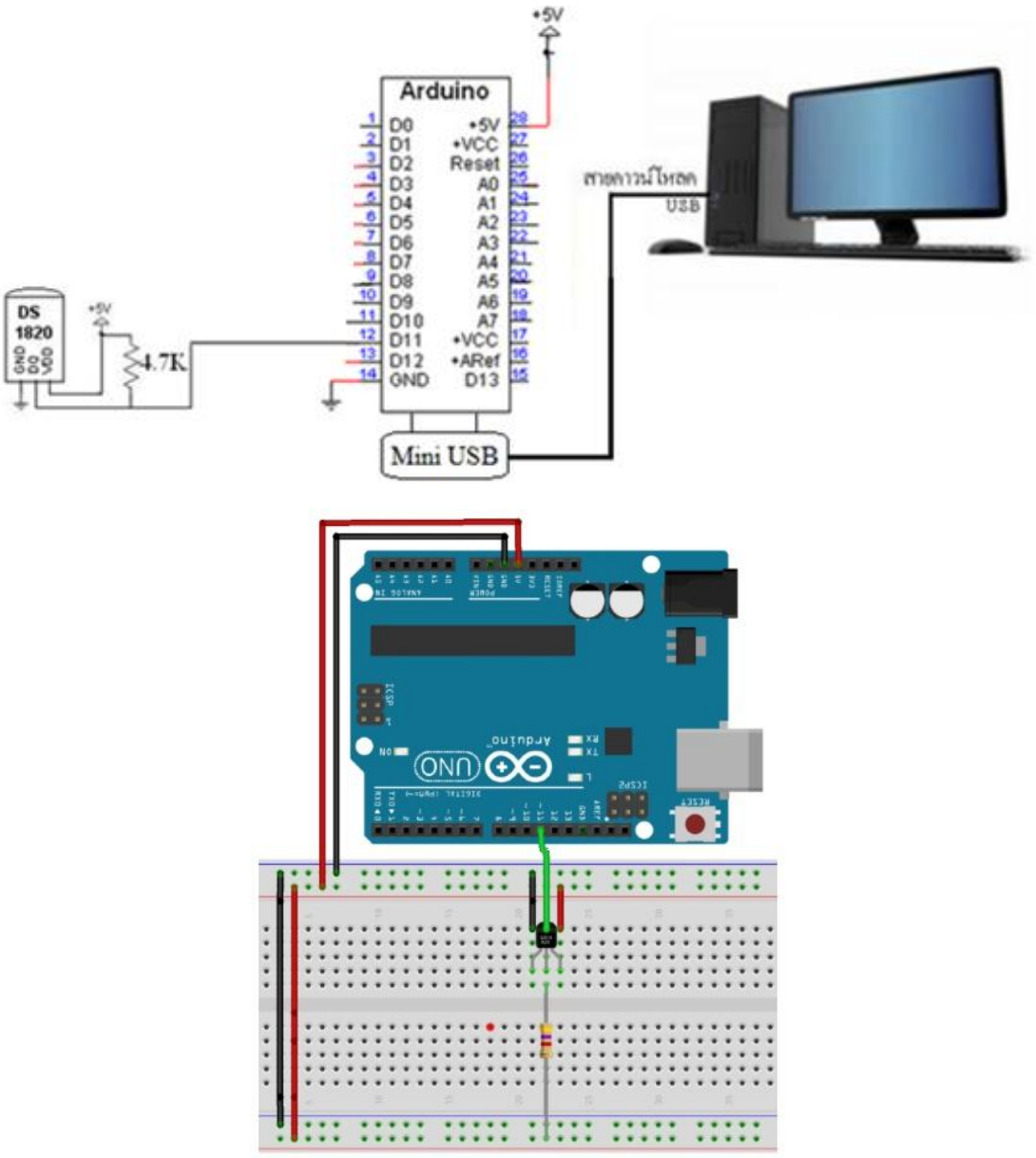
คำนวณหาค่า CRC ของข้อมูลในอะเรย์ **dataArray** ขนาด **length**

ที่มา https://www.pjrc.com/teensy/td_libs_OneWire.html

7. การทดลอง

7.1 การทดลองที่ 1

อ่านค่าใน ROM แล้วส่งออกแสดงที่ Serial Monitor



รูปที่ 15 การต่อ DS18B20 เข้ากับ Arduino

1) ประกอบวงจรตามรูปที่ 15
2) ในส่วนของมาสเตอร์เมื่อต้องการอ่านค่าแอดเดรส์จาก ROM ต้องทำงานดังนี้

มาสเตอร์	Data	Comment
Tx	Reset	พัลส์รีเซ็ต (480-490 us)
Rx	Presence	พัลส์ Presence
Tx	Search	ฟังก์ชั่น Serch

หมายเหตุ Tx คือ มาสเตอร์ส่งข้อมูลให้อุปกรณ์ 1-Wire

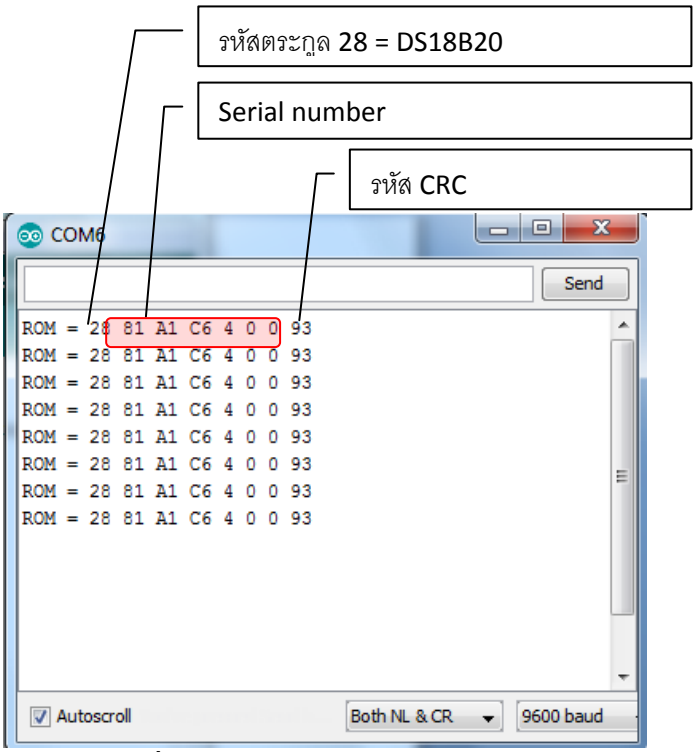
Rx คือ มาสเตอร์รับข้อมูลจากอุปกรณ์ 1-Wire

3) เขียนโปรแกรมที่ 1

โปรแกรม 1 อ่านค่าจาก ROM แล้วส่งออกทางพอร์ทอนุกรม

```
#include <OneWire.h>
// OneWire DS18S20, DS18B20, DS1822 Temperature Example

OneWire  ds(11);  // on pin 11

void setup(void) {
  Serial.begin(9600);
}

void loop(void) {
  byte i;
  byte addr[8];
  ds.reset();               //Reset
  delay(250);
  ds.search(addr);          //Search

  Serial.print("ROM =");            //Print address to serial port
  for( i = 0; i < 8; i++) {
    Serial.write(' ');
    Serial.print(addr[i], HEX);
  }
  Serial.println();
}
```

ผลการทำงาน



รูปที่ 16 ค่าที่ส่งออกทางพอร์ทอนุกรม

7.1 การทดลองที่ 2

อ่านค่าข้อมูล 9 ไบต์ แล้วแปลงเป็นค่าดิบ 12 บิต แล้วแปลงค่าเป็น ทั้งองศา C และองศา F แล้วส่งทั้งหมดออกที่ Serial Monitor

1) ประกอบวงจรตามรูปที่ 15

2) ในส่วนของมาสเตอร์เมื่อต้องการอ่านค่าอุณหภูมิจากหน่วยความจำต้องทำงานดังนี้

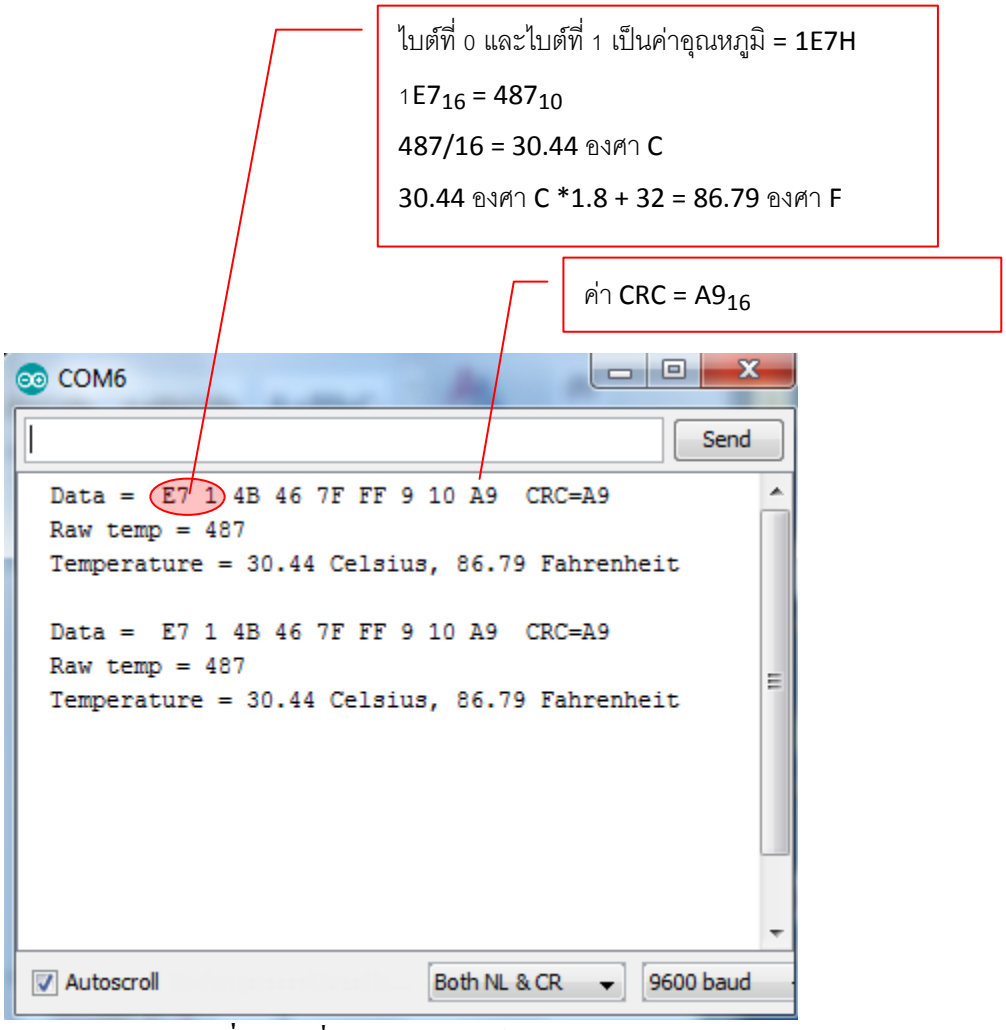
มาสเตอร์	Data	Comment
Tx	Reset_search	เริ่มต้นค้นหาอุปกรณ์
Tx	search(addr)	อ่านค่าแอดเดรส
Tx	Reset	รีเซ็ตอุปกรณ์
Tx	select(addr)	เลือกอุปกรณ์ตามค่าแอดเดรส
Tx	write(0x44,1)	สั่งให้อุปกรณ์เริ่มแปลงค่า
Tx	Reset	รีเซ็ตอุปกรณ์
Tx	select(addr)	เลือกอุปกรณ์ตามค่าแอดเดรส
Tx	write(0xBE)	ต้องการข้อมูลจากหน่วยความจำ
Rx	resd()	อ่านข้อมูล 9 ไบต์

3) เขียนโปรแกรมที่ 2

โปรแกรม 2 อ่านค่าจากหน่วยความจำ แล้วแปลงเป็นองศา C และ F และส่งออกที่ Serial Monitor

```
#include <OneWire.h>
// OneWire DS18S20, DS18B20, DS1822 Temperature Example

OneWire  ds(11);  // on pin 10

void setup(void) {
  Serial.begin(9600);
}

void loop(void) {
  byte i;
  byte present = 0;
  byte data[12];
  byte addr[8];
  float celsius, fahrenheit;

  ds.reset_search();
  delay(250);
  ds.search(addr);
  ds.reset();
  ds.select(addr);
  ds.write(0x44,1);                 // start conversion, with parasite power on at the end
  delay(1000);                      // delay 1000ms

  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);                   // Read Scratchpad

  Serial.print("  Data = ");
  Serial.print(" ");
  for ( i = 0; i < 9; i++) {          // Read 9 bytes data
```

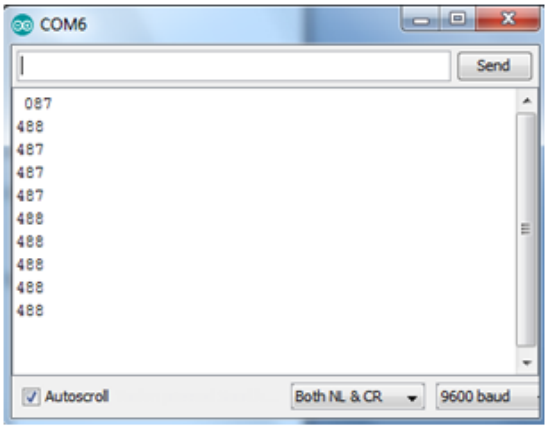
```
  data[i] = ds.read();
  Serial.print(data[i], HEX);
  Serial.print(" ");
 }
 Serial.print(" CRC=");
 Serial.print(OneWire::crc8(data, 8), HEX);
 Serial.println();

 // convert the data to actual temperature
 unsigned int raw = (data[1] << 8) | data[0];
 Serial.print("  Raw temp = ");
 Serial.println(raw);
 //------------------------------
 celsius = (float)raw / 16.0;
 fahrenheit = celsius * 1.8 + 32.0;
 Serial.print("  Temperature = ");
 Serial.print(celsius);
 Serial.print(" Celsius, ");
 Serial.print(fahrenheit);
 Serial.println(" Fahrenheit");
 Serial.println();
}
```



รูปที่ 17 ค่าที่ส่งออกทางพอร์ทอนุกรม

7.2 การทดลองที่ 3

จากโปรแกรมที่ 2 แก้ไขให้ส่งเฉพาะค่าอุณหูมิที่เป็นเลขจำนวนเต็ม 12 บิต (ค่า Raw) ออกที่พอร์ทอนุกรม แล้วใช้โปรแกรม Processing นำค่า Raw นี้ไปคำนวณ เพื่อแสดงออกทางจอภาพ

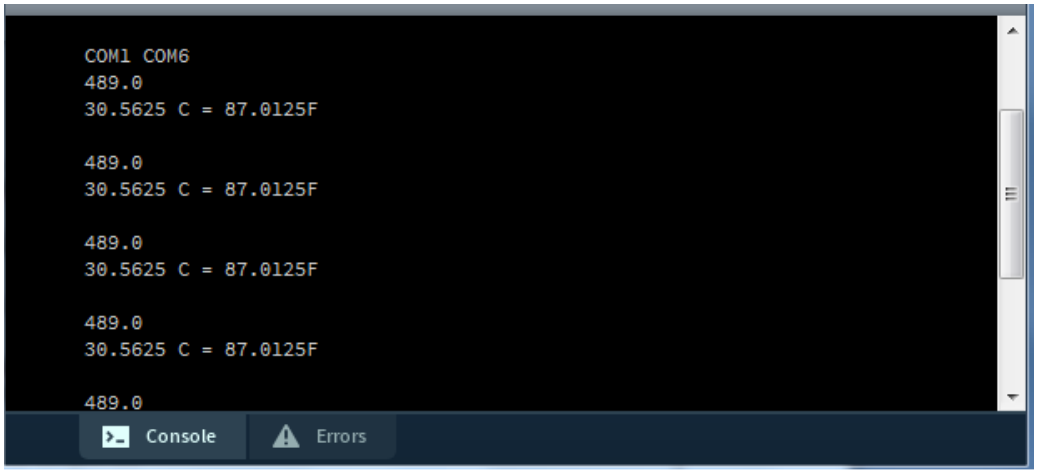


รูปที่ 16 ตัวอย่างค่าที่ส่งออกทางพอร์ทอนุกรม

1) ประกอบวงจรตามรูปที่ 15
2) แก้ไขโปรแกรมที่ 2 เมื่ออัปโหลดโปรแกรม แล้วให้เปิด Serial monitor เพื่อดูค่าที่ส่งออก ควรได้ค่าเป็นตัวเลขจำนวนเต็มเท่านั้น คล้ายๆรูปที่ 16
3) เขียนโปรแกรม Processing ตามโปรแกรมที่ 3 เมื่อรันโปรแกรมจะได้ผลตามรูปที่ 17

โปรแกรมที่ 3 โปรแกรม Processing รับข้อมูลจากพอร์อนุกรม แล้วคำนวณค่าแสดงออกทางคอนโซล

```
import processing.serial.*;
Serial  myPort;
int lf = 10 ;
void setup() {
  size(300, 300);
  println(Serial.list());
  myPort = new Serial(this, Serial.list()[1], 9600);
  smooth();
  background(255);   // Set the background to black
}
void draw()
{
  String myString = myPort.readStringUntil(lf);
  if (myString != null)
  {
      float raw = float (myString);
      println(raw);
      float degree = raw/16;
      print(degree);
      print(" C = ");
      print(degree*1.8+32);
      println( "F");
      println();
  }
}
```

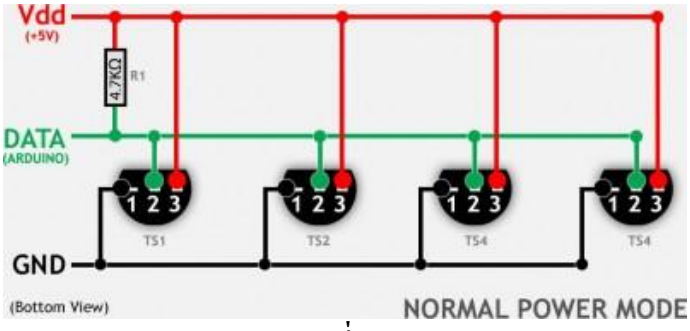


รูปที่ 17

8 วิธีการต่อ DS1820 หลายๆตัว

7.1 Normal Power Mode

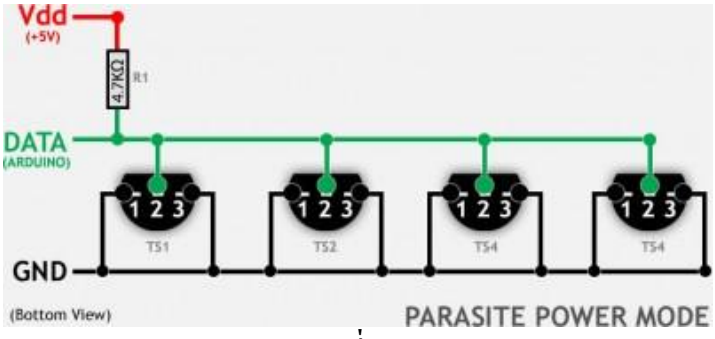
Below you will see how we can set multiple sensors in parallel using the regular powermode.



รูปที่ 18

7.2 Parasite Power Mode

Parasite mode basically drops the wire for Vdd and shorts pin 1 and 3 of the sensor, and the sensor will pull power from the data pin.



รูปที่ 19

ที่มา *http://www.tweaking4all.com/hardware/arduino/arduino-ds18b20-temperature-sensor/*

8. งานมอบหมาย

จากการทดลองที่ 3 ให้เปลี่ยนเป็นแสดงเป็นภาพที่สวยงามบนหน้าต่างของ Processing