

การทดลองที่ 2 : การสื่อสารอนุกรม (Serial Communications)

วัตถุประสงค์

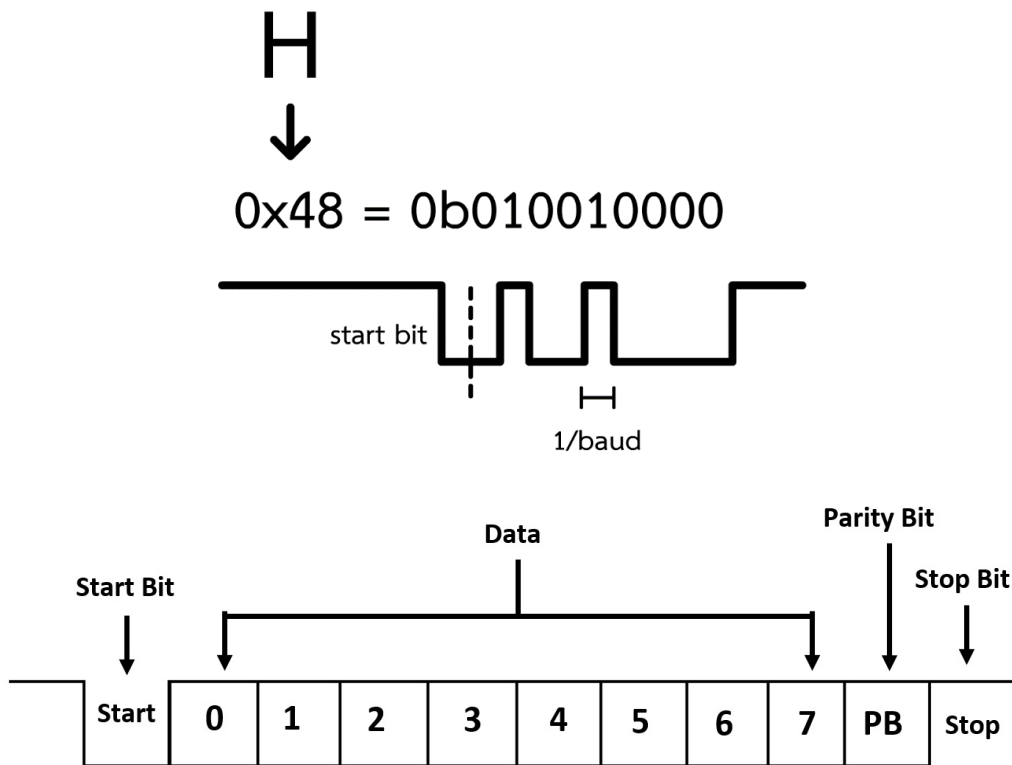
1. เพื่อให้นักศึกษารู้จักพอร์ตสื่อสารอนุกรม
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมใช้งานพอร์ตอนุกรมเบื้องต้นได้เพื่อใช้ในการติบค่าต่าง ๆ
3. เพื่อให้เรียนรู้และใช้งานคำสั่งพอร์ตอนุกรมต่าง ๆ ของ Arduino IDE

บทนำ

การสื่อสารภายนอกของบอร์ด Arduino จะใช้พอร์ตที่เรียกว่าพอร์ตอนุกรม (Serial Port) พอร์ต Serial เป็นพอร์ตการสื่อสารที่เป็นมาตรฐานที่ใช้กันมาตั้งแต่เมาส์ และ คีย์บอร์ดของคอมพิวเตอร์ สำหรับในไมโครคอนโทรลเลอร์นั้น เรายังใช้ Serial Port ในการเชื่อมต่ออุปกรณ์อื่นอีกด้วย ไม่ว่าจะเป็นข้อมูลของเซนเซอร์ต่าง ๆ เช่น เซนเซอร์วัดค่าแสง เซนเซอร์วัดค่าอุณหภูมิ พิกัด GPS , อ่านบัตร RFID และอุปกรณ์ออกแบบเฉพาะงานอย่างเช่น Serial LCD character สำหรับแสดงผล และ Serial Servo Control สำหรับงานควบคุมมอเตอร์ servo จำนวนมาก ๆ เป็นต้น

การสื่อสารผ่านพอร์ตอนุกรม

เมื่อใช้งานบอร์ดทดลอง Arduino หรือ ESP8266, ESP32 ใ้ใช้งานอย่างง่าย หากเราต้องการเขียนโปรแกรมลงบนบอร์ดเหล่านี้ เราก็จะเสียบสาย USB เข้ากับคอมพิวเตอร์ แล้วจะใช้งานเขียนโปรแกรมได้ก็ต้องเลือกพอร์ต COMX แบบนี้ การเชื่อมต่อที่เกิดขึ้นนี้เรียกว่า การสื่อสารผ่านพอร์ตอนุกรม (Serial Port Communication) ซึ่งเป็นการสื่อสารระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ หรือระหว่างไมโครคอนโทรลเลอร์กับอุปกรณ์อื่น ๆ บอร์ดไมโครคอนโทรลเลอร์ Arduino ทุกุ่นนั้นมีพอร์ตสำหรับสื่อสารอย่างน้อย 1 พอร์ต การสื่อสารนี้สามารถเรียก UART หรือ USART ในการใช้งานการสื่อสาร เราจะใช้หมายเลข 0 (RX) และหมายเลข 1 (TX) ของบอร์ด โดยถ้าเราใช้งานขาทั้งสองนี้ในการสื่อสารแล้วจะไม่สามารถใช้ขาทั้งสองมาเป็นขา input หรือ output โดย Serial protocol เป็น protocol พื้นฐานในงานด้าน embedded เนื่องจากเป็น protocol ที่เรียบง่ายและข้อกำหนดไม่ยุ่งยาก โดย serial เป็นการนำข้อมูลที่ต้องการจะส่งมาเปลี่ยนให้อยู่ในรูปแบบของ binary และส่งไปทีละ bit ต่อกันในสายสัญญาณ



www.ucbeginner.com

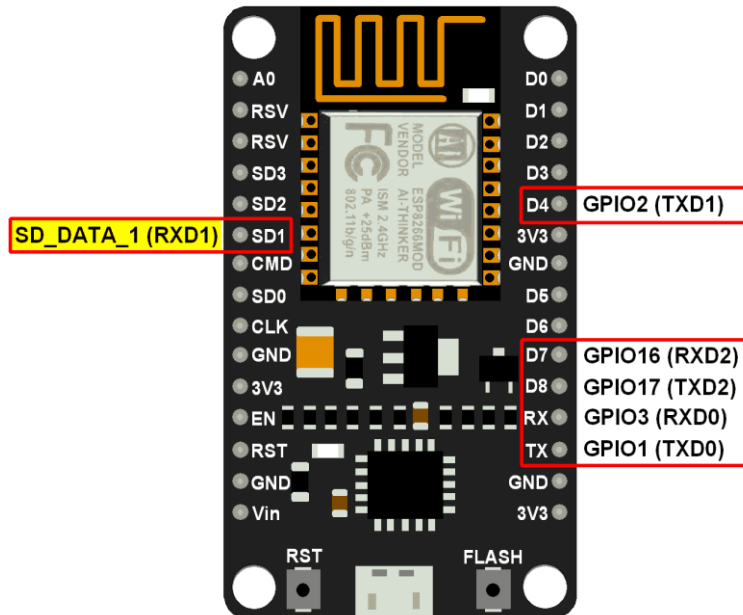
การสื่อสารพอร์ตอนุกรม

โดยเมื่อถูกส่งผ่านสายสัญญาณเรียบร้อยแล้ว ข้อมูลจะถูกเก็บไว้ใน “Buffer”

โดยการสื่อสารอนุกรมจะมีข้อกำหนดอยู่ 4 อย่าง คือ

1. Data bits คือ จำนวน bit ต่อการส่ง 1 ครั้ง
 2. Synchronization bits คือ ข้อกำหนดของการรับข้อมูลโดยจะกำหนดจุดเริ่มต้นของข้อมูลและจุดสิ้นสุดของข้อมูลโดยทั่วไปจะกำหนดให้เมื่ออยู่ในสถานะว่าง (ไม่มีข้อมูลอยู่บนสายสัญญาณ) สายสัญญาณจะมี logic เป็น 1 และเมื่อมีการเริ่มส่งข้อมูลจะมี start bit เป็น logic 0 เป็นเวลาเท่ากับข้อมูล 1 bit และตามด้วยข้อมูลและจะหยุดรับเมื่อครบจำนวน Data bits
 3. Parity bits คือ bit ที่ตรวจสอบความถูกต้องโดยปกติจะไม่ใช้
 4. Baud rate คือ ความเร็วของข้อมูลที่อยู่บนสายส่ง สามารถบอกถึงเวลาของข้อมูลแต่ละ bit
- ในการเริ่มต้นใช้งานการสื่อสารแบบ Serial นั้นมีคำสั่งต่างๆที่จำเป็นดังนี้

สำหรับพอร์ตที่ใช้ในการสื่อสารของ NodeMCU ESP8266 จะมีพอร์ต ดังนี้



RXD0 and TXD0 are the serial control and bootloading pins. They are primarily used for communicating with the ESP module.

คำสั่งสำหรับการใช้งานพอร์ตอนุกรมสำหรับ Arduino IDE

สำหรับคำสั่งของ Arduino ที่ใช้ในการสื่อสาร Serial Port มีตัวอย่างดังนี้

- **void serial.begin(rate)** กำหนดอัตราบอดของการรับส่งข้อมูล หน่วยเป็นบิตต่อวินาที (bits per second :bps baud rate)
- **int serial.available()** ใช้ตรวจสอบว่า buffer รับข้อมูลไว้หรือไม่ โดยจะคืนค่าจำนวนไบต์ที่อยู่ในบัฟเฟอร์
- **int serial.read()** ส่งค่าที่รับจากพอร์ตอนุกรมออกมา (Serial)
- **void Serial.flush()** เคลียร์บัฟเฟอร์ของพอร์ตอนุกรม(Serial) ให้อว่าง
- **void Serial.print()** พิมพ์ข้อมูล ออกทางพอร์ตอนุกรม (ไม่ขึ้นบรรทัดใหม่)
- **void Serial.println()** พิมพ์ข้อมูล ออกทางพอร์ตอนุกรม แต่ขึ้นบรรทัดใหม่ด้วย

นอกจากนี้ ฟังก์ชันอื่น ๆ ดูได้จาก : <https://www.arduino.cc/reference/en/language/functions/communication/serial/>

begin()

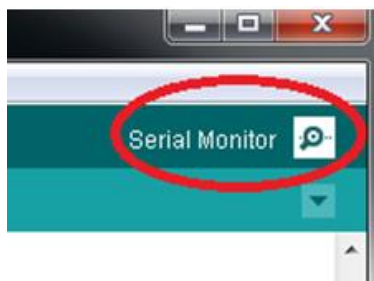
เป็นคำสั่งสำหรับเริ่มต้นการทำงานของการทำงานของการสื่อสารผ่านพอร์ตอนุกรม โดยการเซตอัตราในการส่งและรับข้อมูล มีหน่วยเป็นต่อวินาที (baud rate) โดยการสื่อสารระหว่างบอร์ดไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ต้องเซต baud rate ตามค่าต่อไปนี้ 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, หรือ 115200 ตัวเลขที่มีค่ามากหมายถึงอัตราการสื่อสารที่เร็วเพิ่มมากขึ้น

ยกตัวอย่างหากเราต้องการเซตค่า baud rate เป็น 9600 เพื่อเริ่มต้นการสื่อสารกับคอมพิวเตอร์ให้ใช้คำสั่งตามรูป

```
void setup() {  
  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps  
}  
  
void loop() {}
```

ตัวอย่างการใช้คำสั่ง begin

เมื่อมีการเซตค่า baud rate ให้กับไมโครคอนโทรลเลอร์ ก็สามารถเริ่มต้นสื่อสารกับคอมพิวเตอร์ได้ทันที ในการพัฒนาโปรเจกต์ที่เกี่ยวข้องกับไมโครคอนโทรลเลอร์จำเป็นต้องมีโปรแกรมสำหรับ monitor การสื่อสารแบบอนุกรม เพื่อตรวจสอบดูว่า ไมโครคอนโทรลเลอร์นั้นสามารถสื่อสารกับกับคอมพิวเตอร์อย่างถูกต้องหรือไม่ บอร์ด Arduino หรือ NodeMCU เองก็มีโปรแกรม Serial Monitor มาให้ใช้งานอยู่แล้วโดยไม่ต้องไปหาโปรแกรมมาเพิ่มเติม โดยการใช้งาน โปรแกรม Serial monitor ของ Arduino หรือ NodeMCU เริ่มต้นจากเปิดโปรแกรม Arduino IDE ขึ้นมาดังที่อธิบายในบทที่ 1 โดยโปรแกรม Serial Monitor จะอยู่ที่มุมขวาบนของโปรแกรม ตามรูป



การเปิดโปรแกรม Serial Monitor


การใช้ Serial Monitor มีประโยชน์ในการตรวจสอบการทำงานของไมโครคอนโทรลเลอร์ ว่าทำงานถูกต้องตามที่เรต้องการหรือไม่ หัวข้อต่อไปเป็นการสั่งให้ไมโครคอนโทรลเลอร์ส่งข้อมูลมาที่คอมพิวเตอร์ด้วยคำสั่ง print()

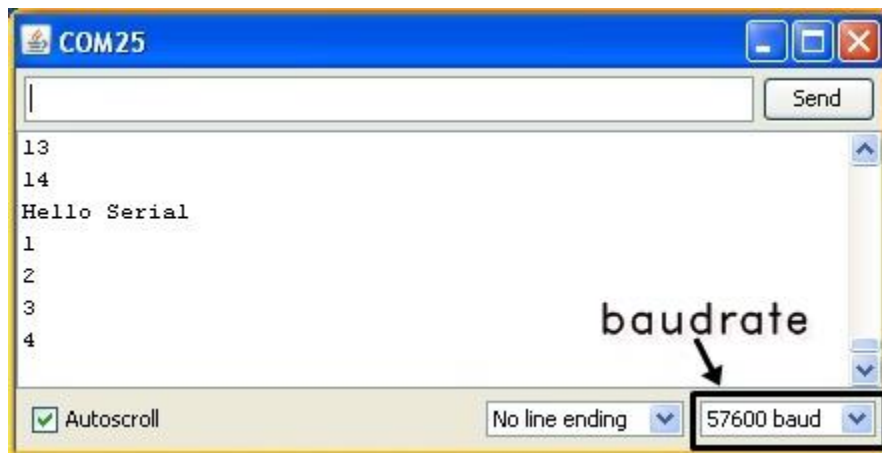
การเขียนโปรแกรม Serial Port กับ Arduino หรือ NodeMCU

ทดลองเขียนโปรแกรมส่งข้อมูลจาก Arduino หรือ NodeMCU ไปยังคอมพิวเตอร์โดยผ่านการสื่อสาร Serial Port โดยเปิดโปรแกรม Arduino หรือ NodeMCU แล้วทำการเขียนโค้ดลงไปดังนี้

```
void setup() {  
  Serial.begin(9600);  
  
}  
void loop() {  
  Serial.println("Hello ITE");  
  delay(100);  
}
```

ตัวอย่างโค้ดส่งข้อมูลผ่านพอร์ตอนุกรม

จากนั้นทำการคอมไพล์และเบิร์นลงบอร์ดให้เรียบร้อย จากนั้นกดปุ่ม  บนหน้าจอของโปรแกรม Arduino หรือ NodeMCU ทำการเลือก Baud Rate ตั้งแต่ 300, 1200, 4800, 9600, 19200, 38400, 57600, 115200 โดยปรับบอดเรตดังรูป จากนั้นสังเกตและบันทึกผลการทดลองลงในตาราง



ตัวอย่างการปรับบอดเรต

ตารางที่ 1 บันทึกผลการทดลองเมื่อปรับบอดเรต

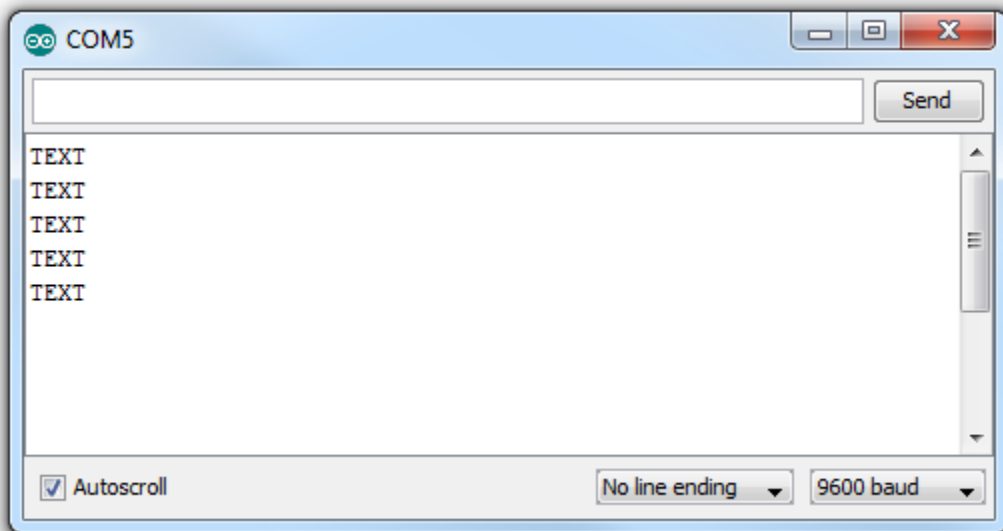
Baud Rate	สิ่งที่ปรากฏบนหน้าจอ
-----------	----------------------

1200	
4800	
9600	
19200	
38400	
57600	
115200	

คำถาม ? : เหตุใดบางค่าของ Baud Rate จึงปรากฏข้อความที่ไม่ใช่ Hello ITE

คำสั่งนี้เป็นการสั่งให้ Arduino ส่งข้อมูลผ่านทางพอร์ตอนุกรม ในรูปแบบ ASCII คำสั่ง print สามารถใช้ได้หลายรูปแบบทั้งแบบตัวเลขจำนวนเต็ม ตัวเลขทศนิยม(ค่าพื้นฐานคือทศนิยม 2 ตำแหน่ง) ตัวอักษร หรือแบบข้อความ

คำสั่ง println เป็นคำสั่งที่ทำหน้าที่เหมือนกับคำสั่ง print ทุกประการ เพียงแต่แตกต่างตรงที่คำสั่ง println จะมีการขึ้นบรรทัดใหม่ให้ทุกครั้งหลังจากจบคำสั่ง ดังแสดงในรูป



ผลของการใช้คำสั่ง println

ภายในฟังก์ชัน Serial.println นั้น จะมีรูปแบบคือ Serial.println(value ,format) โดยฟังก์ชันนี้

จะสามารถส่งค่า ออกมาในรูปแบบอื่นได้ ไม่ว่าจะเป็น แบบตัวอักษร(BYTE), เลขฐานสอง(BIN), เลขฐานแปด (OCT), เลขฐานสิบ(DEC), เลขฐานสิบหก(HEX) ให้นักศึกษาทำการทดลองเขียนโปรแกรม Serial.println ดังตารางข้างล่าง สังเกตและบันทึกผลการทดลองในตารางที่ 7.2

ตารางที่ 2 บันทึกผลการทดลองจากคำสั่ง println

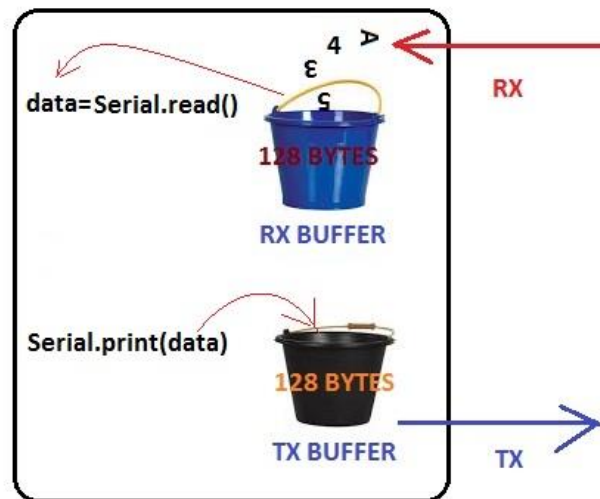
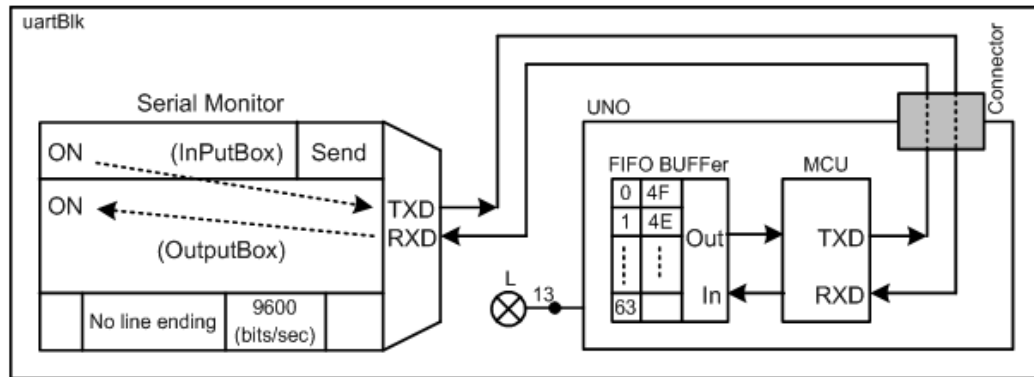
คำสั่ง	สิ่งที่ปรากฏบนหน้าจอ
Serial.println(78, OCT)	
Serial.println(78, DEC)	
Serial.println(78, HEX)	
Serial.println(1.23456, 0)	
Serial.println(1.23456, 2)	
Serial.println(1.23456, 4)	

available()

เป็นคำสั่งสำหรับอ่านค่าจำนวนของ byte ที่ถูกส่งมาให้กับบัฟเฟอร์ของ Serial เช่นหากผู้ใช้ส่งข้อมูลขนาด 3 byte มาที่บอร์ดไมโครคอนโทรลเลอร์ผ่านทาง Serial port หรือพิมพ์ผ่านทาง Serial Monitor ค่าที่ได้จากคำสั่ง available คือ 3 เป็นต้น โดยคำสั่งนี้สามารถใช้งานคู่กับ คำสั่ง read()

read()

เป็นคำสั่งสำหรับการอ่านค่าข้อมูลที่ถูกส่งเข้ามายัง Serial Port ของไมโครคอนโทรลเลอร์ ทีละ ไบท์ โดยจะมีการเก็บไว้ยัง Buffer ของ Serial Port ก่อนที่จะมีข้อมูลเข้ามา จะ return ค่าเป็น -1 ถ้าไม่มีข้อมูลเข้ามา



ตัวอย่างคำสั่ง เช่น

```

1 void setup()
2 {
3   pinMode(LED_BUILTIN, OUTPUT);
4   Serial.begin(9600);
5 }
6
7 void loop()
8 {
9   int a = Serial.read();
10  Serial.println(a);
11  delay(1000); // Wait for 1000 millisecond(s)
12 }

```

เปิดหน้าต่าง Serial Monitor สังเกตผลลัพธ์ที่ได้เป็นอย่างไร ?

จากนั้นทดลอง พิมพ์ abc ลงในช่องอินพุตของ Serial Monitor
ผลลัพธ์ที่ได้เป็นอย่างไร?

จากนั้น ทดลองเขียนโปรแกรม

```
1 void setup()
2 {
3   pinMode(LED_BUILTIN, OUTPUT);
4   Serial.begin(9600);
5 }
6
7 void loop()
8 {
9   int a = Serial.read();
10  int b = Serial.read();
11  Serial.println(a);
12  Serial.println(b);
13  delay(1000); // Wait for 1000 millisecond(s)
14 }
```

ผลลัพธ์ที่ได้เป็นอย่างไร ?

จะสังเกตได้ว่า ข้อมูลใน buffer จะถูกจับมาใส่ในตัวแปร ตามลำดับไปเรื่อย ๆ

แต่จะเห็นได้ว่าเมื่อครบทุกตัวที่ input ไปแล้ว จะ return -1 มาเรื่อย ๆ

ต่อไปทดลองใช้งาน คำสั่ง read ควบคู่ไปกับคำสั่ง available เป็นไปตามรูป

Example Serial Lab – read()

```
int incomingByte = 0;

void setup() {
  Serial.begin(115200);
  Serial.println("Hello, ESP32!");
}

void loop() {
  if(Serial.available() > 0)
  {
    incomingByte = Serial.read();
    Serial.println(incomingByte);
  }
  delay(500);
}
```

ตัวอย่างการใช้งาน คำสั่ง read และ available

ผลที่ได้เป็นอย่างไร ?

จะเห็นว่า available เป็นคำสั่งที่ใช้ตรวจสอบว่ามีข้อมูลอยู่ใน Buffer นั้นเองหรือไม่

หลังจากที่ได้เรียนรู้คำสั่งที่จำเป็นในการใช้งานการสื่อสารผ่านพอร์ทอนุกรมแล้ว หลังจากนี้จึงสามารถส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ได้ ทำให้การพัฒนาไมโครคอนโทรลเลอร์มีความสะดวกมากขึ้น

ทำการทดลองการรับข้อมูลจากคอมพิวเตอร์มายัง Arduino โดยใช้คำสั่ง Serial.ReadBytes() ซึ่งจะทำให้การอ่านตัวอักษรจากพอร์ทอนุกรมไปเก็บยังบัฟเฟอร์

Syntax

```
Serial.readBytes(buffer, length)
```

Parameters

Serial: serial port object. See the list of available serial ports for each board on the Serial main page

buffer: the buffer to store the bytes in. Allowed data types: array of char or byte.

length: the number of bytes to read. Allowed data types: int.

โดยทำการใช้โปรแกรม Arduino และทำการเขียนโค้ดลงไปดังนี้

Example Serial Lab – readBytes()

```
int incomingByte = 0;
const int BUFFER_SIZE = 50;
char buf[BUFFER_SIZE];

void setup() {
  Serial.begin(115200);
  Serial.println("Hello, ESP32!");
}

void loop() {
  if(Serial.available() > 0)
  {
```

```

incomingByte = Serial.readBytes(buf, BUFFER_SIZE);
Serial.println(incomingByte);

for(int i=0; i < incomingByte; i++)
  Serial.print(buf[i]);
}
delay(500);
}

```

ตัวอย่างการใช้คำสั่ง Serial.readBytes

เปิดหน้าต่าง Serial Monitor จากนั้นพิมพ์ข้อความว่า Hello_World
ผลลัพธ์ที่ได้ เป็นอย่างไร ? อธิบายผลที่ได้

คำสั่ง Serial.readBytesUntil

Example Serial Lab – readBytesUntil()

```

int incomingByte = 0;
const int BUFFER_SIZE = 50;
char buf[BUFFER_SIZE];

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.println("Hello, ESP32!");
}

void loop() {
  // put your main code here, to run repeatedly:
  if(Serial.available() > 0)
  {
    incomingByte = Serial.readBytesUntil('o',buf, BUFFER_SIZE);
    Serial.println(incomingByte);

    for(int i=0; i < incomingByte; i++)

```

```
Serial.print(buff[i]);  
}  
delay(500); // this speeds up the simulation  
}
```

ตัวอย่างการใช้คำสั่ง Serial.readBytesUntil()

เปิดหน้าต่าง Serial Monitor จากนั้นพิมพ์ข้อความว่า Hello_World

ผลลัพธ์ที่ได้ เป็นอย่างไร ? อธิบายผลที่ได้

คำสั่ง parseInt();

คำสั่งนี้เป็นการค้นหาจำนวนจริง (Int) ที่อยู่ในบัฟเฟอร์ที่อ่านค่าเข้ามา ว่ามีจำนวนใดบ้างที่เป็นจำนวนเต็ม หากไม่มีอินพุตเข้ามาจะทำการ return เป็น 0 โดยทดลองเขียนโค้ด ดังนี้

Example Serial Lab – read() parseInt

```
int val;  
void setup() {  
  Serial.begin(9600);  
}  
void loop() {  
  while (Serial.available() == 0) {  
    // Wait for user input  
  }  
  val = Serial.parseInt();  
  Serial.print("You entered: ");  
  Serial.println(val);  
}
```

เปิด Serial Monitor จากนั้นพิมพ์ abcd15235 แล้วกด Enter

ผลลัพธ์ที่ได้ เป็นอย่างไร

คำสั่ง serial.write()

คำสั่งนี้เป็นการส่งข้อมูลผ่านพอร์ตอนุกรมในรูปแบบของไบนารี โดยสามารถส่งข้อมูลได้ในรูปแบบของไบนารี โดยจะต่างกับ Serial.print() ที่จะส่งข้อมูลจากโดยแปลงจาก ASCII เป็น binary

Example Serial Lab – serial.write

```
void setup( )
{
  Serial.begin(9600);
}
void loop( )
{
  Serial.write(55); // the specified value is 55.
  // Serial.write( ) send the data as a byte with this value (55).
  int Bytestosend = Serial.write( "HelloWorldIoT" );
  Serial.println(bytesSent);
  // It sends the Arduino string.
  //The length of the string is a return parameter in this function.
}
```

เปิด Serial Monitor ผลลัพธ์ที่ได้ เป็นอย่างไร

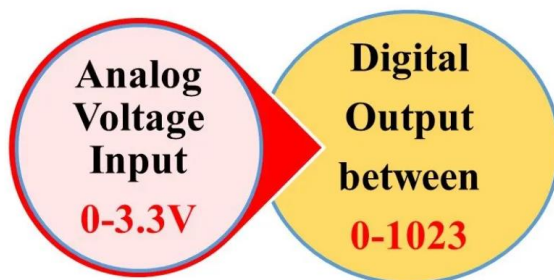
การทดลองที่ 3 : Analog to Digital Convertor

วัตถุประสงค์

1. เพื่อให้นักศึกษารู้จักการอ่านค่าเซ็นเซอร์โดยใช้หลักการของ ADC และวงจรแบ่งแรงดัน
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมอ่านค่าเซ็นเซอร์แสงด้วย LDR ได้

3.1 การแปลงแอนะล็อกเป็นดิจิตอล (Analog to Digital Convertor : ADC)

สัญญาณแอนะล็อกคือสัญญาณที่มีการเปลี่ยนแปลง แบบต่อเนื่องทั้งขนาดของค่าสัญญาณและเวลา ดังนั้นเมื่อพล็อตสัญญาณแอนะล็อกออกมาเป็นกราฟ จะมีลักษณะเป็นเส้นต่อเนื่องกัน ในขณะที่สัญญาณดิจิทัลคือสัญญาณที่มีการเปลี่ยนแปลงแบบขั้นทั้งขนาดของค่าสัญญาณและเวลา (Discrete in value and time) โดยมากแล้วสัญญาณดิจิทัลได้มาจากสัญญาณแอนะล็อกที่ผ่านกระบวนการ Sampling และการ Quantization ทางบอร์ด NodeMCU มี 1 ขา คือขา A0 รวมถึงขา REF (Analog Reference) โดยบางบอร์ดอย่างเช่น WeMos อาจมีสองขา คือ A0 และ ADC โดยที่ความละเอียดของ ADC อยู่ที่ 10 บิต (1024 ค่า, 0-1023) หมายความว่าหากอ่านค่าแรงดันไฟฟ้าเป็นสัญญาณแอนะล็อกได้ 5V (เทียบเท่าไฟเลี้ยงบอร์ด) แปลงมาเป็นค่าเป็นดิจิทัลแล้วบอร์ดจะเห็นเป็นค่า 1023 และในลักษณะเดียวกัน เมื่ออ่านค่าแรงดันไฟฟ้าแอนะล็อกได้ 0 V ค่าดิจิทัลที่บอร์ดเห็นจะเป็น 0 และใช้งานผ่านโปรแกรมด้วยฟังก์ชัน AnalogRead

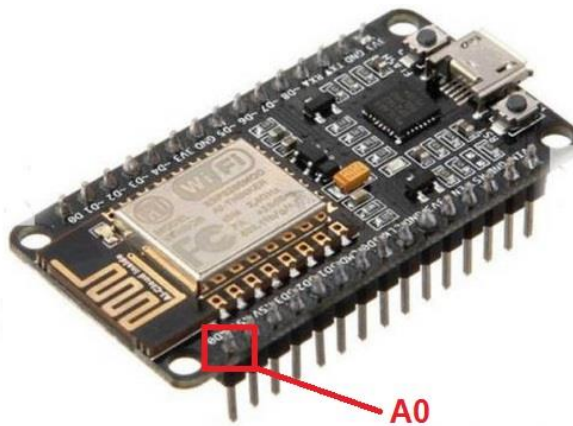


คำสั่งสำหรับการใช้งานการแปลงแอนะล็อกเป็นดิจิทัล : ADC

สำหรับคำสั่งของ Arduino ที่ใช้ใน ADC มีดังนี้

- **analogRead(pin)** ใช้สำหรับอ่านค่าแอนะล็อกจากพอร์ต Arduino โดยจะทำการอ่านค่า Analog มาให้เรา โดยค่าที่ Return กลับมาจะเป็นจำนวนเต็มตั้งแต่ 0 ไปจนถึง 1023 ดังนั้นการใช้คำสั่ง analogRead จำเป็นต้องมีตัวแปรมาเก็บค่าที่อ่านได้จากฟังก์ชันนี้ โดยต้องประกาศตัวแปรเป็นจำนวนเต็ม (Integer)

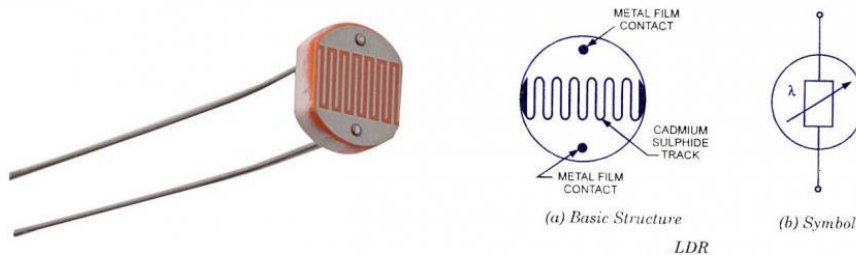
- **analogReference(type)** ใช้ในการอ้างอิงค่า Volt สูงสุดของ Analog Input เมื่อกำหนดค่า Type เป็น
 - **DEFAULT** : ใช้ 5V สำหรับบอร์ดที่จ่ายไฟ 5V และเป็น 3.3V สำหรับบอร์ดที่จ่ายไฟ 3.3V (สำหรับ NodeMCU ใช้ 3.3V)
 - **INTERNAL** : จะใช้ค่าอ้างอิงภายในบอร์ด 1.1V สำหรับ Atmega168/328 และ 2.56V สำหรับ ATmega8
 - **INTERNAL1V1** : จะใช้ค่าอ้างอิงภายในบอร์ด 1.1V สำหรับบอร์ด Arduino Mega เท่านั้น
 - **INTERNAL2V56** : จะใช้ค่าอ้างอิงภายในบอร์ด 2.56V สำหรับบอร์ด Arduino Mega เท่านั้น
 - **EXTERNAL** : จะใช้อ้างอิงภายนอกบอร์ดที่รับมาจากขา AREF (0-5V)
- **analogReadResolution(bit)** analogReadResolution() เป็นฟังก์ชันเพิ่มใน API อนุสาวรีย์ของบอร์ด NodeMCU (ESP8266) สามารถกำหนดขนาด (ให้เป็นบิต) ของค่าที่ส่งกลับมาโดย analogRead() ค่าเริ่มต้นจะอยู่ที่ 10 บิต (ค่าที่ส่งกลับอยู่ระหว่าง 0-1023)



ตัวต้านทานปรับค่าตามแสง

LDR (Light Dependent Resistor) คือตัวต้านทานปรับค่าตามแสง ตัวต้านทานชนิดนี้สามารถเปลี่ยนความนำไฟฟ้าได้เมื่อมีแสงมาตกกระทบ โฟโตริซิสเตอร์ (Photo Resistor) หรือ โฟโตคอนดักเตอร์ (Photo Conductor) เป็นตัวต้านทานที่ทำมาจากสารกึ่งตัวนำ (Semiconductor) ประเภทแคดเมียมซัลไฟด์ (Cds : Cadmium Sulfide) หรือแคดเมียมซีลีไนด์ (CdSe : Cadmium Selenide) ซึ่งทั้งสองตัวนี้ก็เป็นสารประเภทกึ่งตัวนำ เอามาฉาบลงบนแผ่นเซรามิกที่ใช้เป็นฐานรองแล้วต่อขาจากสารที่ฉาบ ไข่ออกมา โครงสร้างของ LDR

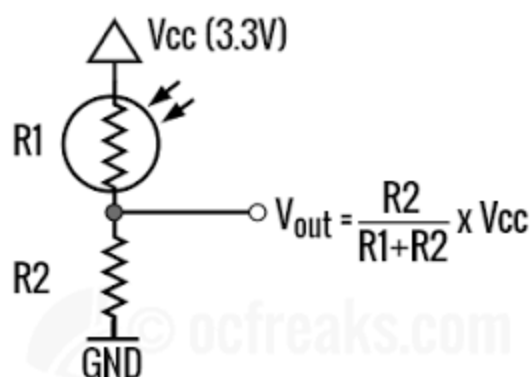
การทำงานของ LDR เมื่อเวลาที่มีแสงตกกระทบลงไปก็จะถ่ายทอดพลังงาน ให้กับสาร ที่ฉาบอยู่ ทำให้เกิด โหมดกับอิเล็กตรอนวิ่งกันพล่าน การที่มี โหมด กับอิเล็กตรอนอิสระนี้มากก็เท่ากับ ความต้านทานลดลงนั่นเอง ยิ่ง ความเข้มของแสงที่ตกกระทบมากเท่าไร ความต้านทานก็ยิ่งลดลงมากเท่านั้น ดังนั้นเมื่อ LDR ถูกแสงตกกระทบจะ ทำให้ ตัว LDR มีความต้านทานลดลง และเมื่อไม่มีแสงตกกระทบจะมีความต้านทานมากขึ้น



เราสามารถวัดโวลต์และอ่านค่าเป็นความสว่างของแสงจาก LDR ได้ สัญญาณที่ได้เป็นแบบ Analog ดังนั้นจึงทำได้ โดยผ่านทางขา Analog ของ Arduino

เนื่องจาก LDR ทนกำลังไฟฟ้าได้เพียงประมาณ 50 mW ดังนั้นถ้าเราใช้โอห์มมิเตอร์สเกล R วัดความต้านทานของ LDR อาจทำความเสียหายให้กับ LDR ได้ เราอาจวัดความต้านทานของ LDR โดยอาศัยวงจรแบ่งแรงดัน

วงจรแบ่งแรงดัน (Voltage Divider) และการคำนวณ



$$V_{out} = V_{in} \cdot \frac{R_2}{R_1 + R_2} \quad (1)$$

สมการการคำนวณวงจรแบ่งแรงดัน

โดย Vin คือ Vinut ที่เราใช้ ในที่นี้ NodeMCU คือ 3.3V

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

(2)

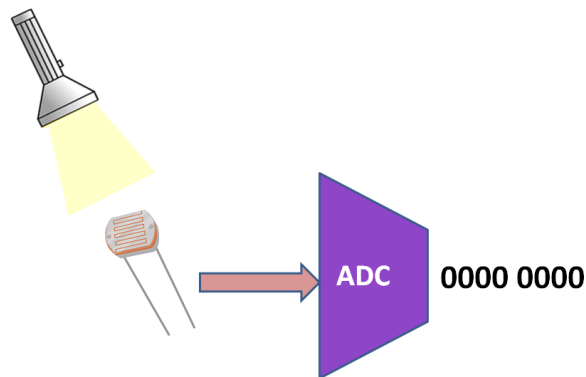
และสมการ การคำนวณ ADC

โดย Vref คือ 3.3V ของ NodeMCU

ADC คือ ค่าที่ขา analogRead อ่านค่าได้

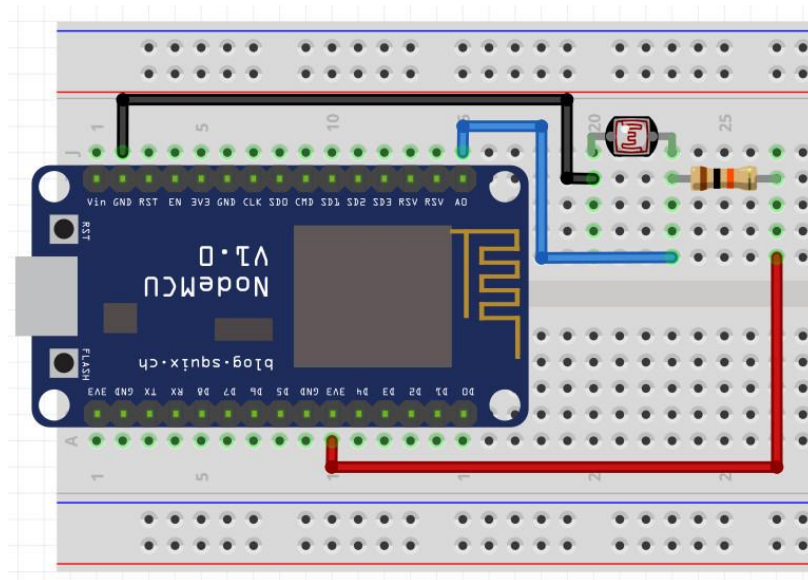
Vin คือ แรงดันของค่านั้น ๆ

จากสองสมการนี้ จะสังเกตได้ว่า V_{out} ของสมการที่ 1 คือ Vin ของสมการที่สอง นั่นเอง



การทดลอง วัดความสว่างด้วย LDR

1. ทำการเชื่อมต่อ Arduino Nano กับ LDR ดังรูป



2. เขียนโปรแกรมแล้วทำการ Upload ลงบอร์ด NodeMCU

Example 3.1 LDR – Light Sensor

```
void setup() {  
  Serial.begin(9600);  
}  
  
void loop() {  
  int sensorValue = analogRead(A0);  
  Serial.println(sensorValue);  
  delay(500);  
}
```

เปิดหน้าจอ Serial Monitor สังเกตผลการทดลองที่ได้

ค่า R ที่นักศึกษานำมาใช้มีค่า _____ ohm

- เมื่อ นำ LDR ปลอยไว้ตามปกติ ค่า ADC ที่อ่านได้จาก หน้าจอ Serial Monitor มีค่า _____ และจากการคำนวณด้วยสมการ (2) แล้วค่าแรงดัน หรือ Vin คือ _____ แสดงว่าในขณะนั้น LDR มีค่าความต้านทาน _____ ohm (คำนวณจากสมการที่ (1))
- ทดลองนำนิ้วปิดที่ LDR ค่า ADC ที่อ่านได้จาก หน้าจอ Serial Monitor มีค่า _____ และจากการคำนวณด้วยสมการ (2) แล้วค่าแรงดัน หรือ Vin คือ _____ แสดงว่าในขณะนั้น LDR มีค่าความต้านทาน _____ ohm (คำนวณจากสมการที่ (1))
- ทดลองนำไฟฉายจากโทรศัพท์ มาทดลองฉายที่ LDR ค่า ADC ที่อ่านได้จาก หน้าจอ Serial Monitor มีค่า _____ และจากการคำนวณด้วยสมการ (2) แล้วค่าแรงดัน หรือ Vin คือ _____ แสดงว่าในขณะนั้น LDR มีค่าความต้านทาน _____ ohm (คำนวณจากสมการที่ (1))

โจทย์การทดลอง

1. ให้นักศึกษาต่อวงจร Arduino กับหลอดไฟ LED โดยทำการเขียนโปรแกรมสั่งการผ่าน Serial Port เช่น เมื่อพิมพ์ตัวอักษร 'a' ลงไปบนหน้าจอ Serial Monitor จะทำให้ LED กระพริบ และเมื่อพิมพ์ตัวอักษร 'b' ลงบนหน้าจอ Serial Monitor จะทำให้ LED ดับ โดยเขียนโค้ดของการทดลอง

```
If (a == 'a') { digital... }
```
2. เขียนโปรแกรมรับค่าเซ็นเซอร์ความสว่างของแสง(ด้วย LDR) และเชื่อมต่อกับ LED โดยหากมีค่าความสว่างมาก ให้ LED สว่าง ถ้าไม่มีความสว่าง ให้ LED ดับ

การส่งงาน (ส่งภายในห้องเรียน)

- 1) ส่งเอกสารนี้ใน Microsoft teams ตอบคำถามในแต่ละส่วนให้เรียบร้อย
- 2) ทำโจทย์การทดลองทั้งสองข้อ ให้อาจารย์/TA ตรวจ

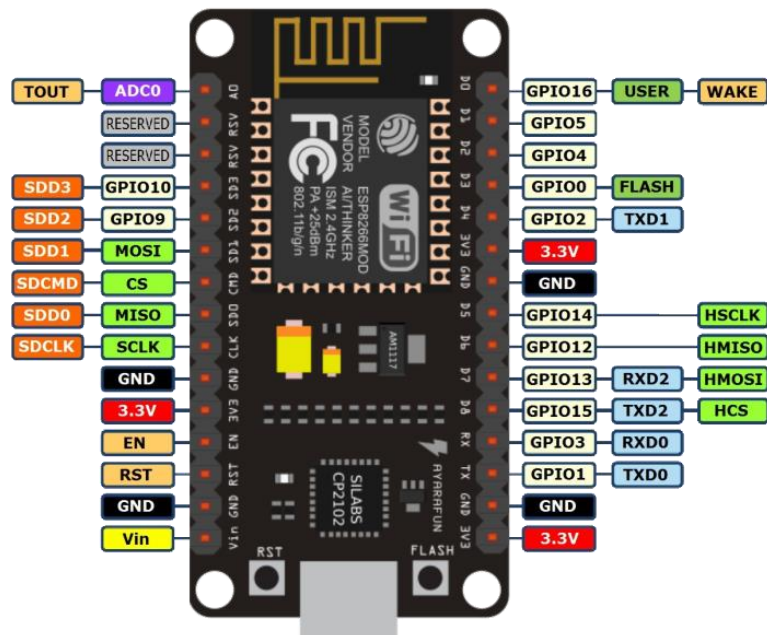
Assignment 1

- 1) จับคู่กับเพื่อน ศึกษาการส่งข้อมูลระหว่าง NodeMCU สองบอร์ด ด้วย Hardware Serial และ Software Serial โดยให้ส่งข้อมูลเซ็นเซอร์ความสว่างของแสงส่งไปยัง NodeMCU ของเพื่อนอีกบอร์ดหนึ่งแล้ว ตรวจสอบค่าที่ส่งมาให้ LED บนบอร์ดของเพื่อน สว่าง หรือดับ
- 2) ถ่ายวิดีโอ อัปโหลดลง youtube นำเสนองาน
** ส่งภายใน 2 สัปดาห์

Reference

การใช้งานพอร์ตต่าง ๆ ของ NodeMCU

สำหรับบอร์ด NodeMCU มีรายละเอียดของแต่ละขา ดังนี้



รูปที่ 1 ขาต่าง ๆ ของ NodeMCU ESP8266

ตาราง NodeMCU Port Details and GPIO Map

Pin	Function	ESP-8266 Pin
TX	TXD	TXD
RX	RXD	RXD
A0	Analog input, max 3.3V input	A0,ADC
D0	IO	GPI016
D1	IO, SCL	GPI05
D2	IO, SDA	GPI04
D3	IO, 10k Pull-up	GPI00
D4	IO, 10k Pull-up, BUILTIN_LED	GPI02

D5	IO, SCK	GPIO14
D6	IO, MISO	GPIO12
D7	IO, MOSI	GPIO13
D8	IO, 10k Pull-down, SS	GPIO15
G	Ground	GND
3V3	3.3V	3.3V
RST	Reset	RST

ASCII Code

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]