การสื่อสารแบบอนุกรม

Serial Communication

รศ.ณรงค์ บวบทอง

หัวข้อ

บทนำ

การสื่อสารข้อมูลแบบอนุกรม เป็นการรับส่งข้อมูลทีละบิต แทนที่จะทำการรับส่งข้อมูลพร้อมกันทุกบิตในเวลาเดียวกัน ข้อดีของการสื่อสารแบบนี้คือ ใช้จำนวนสายในการสื่อสารน้อย สามารถรับส่งได้ในระยะ ทางที่ไกล ๆ แต่ก็มีข้อเสียในด้านเวลา เพราะต้องใช้เวลาในการสื่อสารมาก เมื่อเทียบกับการสื่อสารแบบขนาน อีกทั้งโอกาสเกิดการผิดพลาดของข้อมูลก็สูงกว่าแบบขนาน

รูปแบบของการสื่อสาร

รูปแบบของการสื่อสาร แบ่งได้ 3 แบบ คือ

1. แบบซิมเพล็กซ์ (Simplex) เป็นการสื่อสารทางเดียว
2. แบบฮาล์ฟดูเพล็กซ์(Half-duplex) เป็นการสื่อสารได้ทั้งสองทาง แต่จะต้องผลัดกันรับ-ส่ง
3. แบบฟูลดูเพล็กซ์(Full-duplex) เป็นการสื่อสารได้ทั้งสองทางและทำได้ในเวลาเดียวกัน

รูปแบบการสื่อสารข้อมูลแบบอนุกรม

การติดต่อแบบอนุกรมเมื่อแบ่งตามลักษณะของการส่งข้อมูล แบ่งได้ 2 แบบ คือ

1. การสื่อสารแบบซิงโครนัส (Synchronous)
2. การสื่อสารแบบอะซิงโครนัส(Asynchronous)
3. การส่งข้อมูลแบบไอโซโครนัส (Isochronous Transmission)

การสื่อสารข้อมูลแบบซิงโครนัส (Synchronous)

เป็นการส่งข้อมูลเป็นบล็อก ครั้งละหลายๆไบต์ สัญญาณนาฬิกาที่ช่วยให้การทำงานของตัวส่งและตัวรับสอดคล้องกัน อาจจะถูกเข้ารหัสอยู่ในชุดของข้อมูลนั้นหรือแยกอิสระออกเป็นสายต่างหากก็ได้ ตัวอย่างการรับส่งข้อมูลแบบนี้ได้แก่โปรโตคอล High-Level Data Link Control

โปรโตคอล High-Level Data Link Control (HDLC)

โปรโตคอล High-Level Data Link Control (HDLC) ที่ใช้กับโมเด็มแวน (WAN) ซึ่งมีลักษณะแฟรม (Fram) ดังนี้

ชื่อฟิลด์	ขนาด
Flag Field(F)	8 บิต
Address Field(A)	8 บิต
Control Field(C)	8 or 16 บิต
Information Field(I)	เปลี่ยนแปลงได้ บางแฟลมก็ไม่ได้ใช้
Frame Check Sequence(FCS)	16 or 32 บิต
Closing Flag Field(F)	8 บิต

การสื่อสารข้อมูลแบบอะซิงโครนัส (Asynchronous)

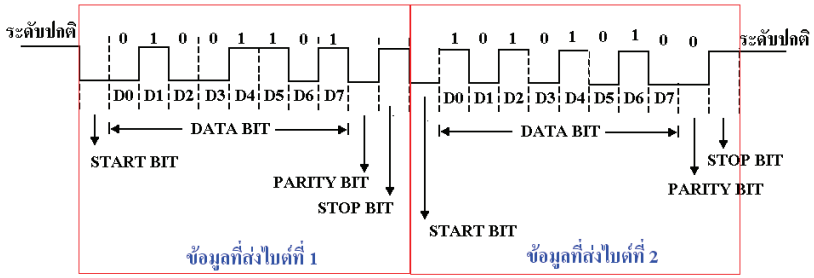
การสื่อสารแบบนี้ใช้มากในเครื่องไมโครคอมพิวเตอร์พีซี รูปแบบการสื่อสารจะเป็นการรับและส่งข้อมูลครั้งละ 1 ไบต์

ตัวอย่างโปรโตคอลอะซิงโครนัส Serial Commincation - Example Protocols

- Morse code
- RS-232 - Recommended Standard 232
- RS422, RS-423, RS-485
- I2C - Inter-Integrated Circuit
- SPI - Serial Peripheral Interface
- USB - Universal Serial Bus
- Firewire
- Ethernet
- Serial ATA - Serial Advanced TEchnology Attachment
- Serial Attach SCSI - Serial Attached Small Computer System
- Interface
- SONET - Synchronous Optical Network
- PCI Express - Peripheral Component Interconnect Express

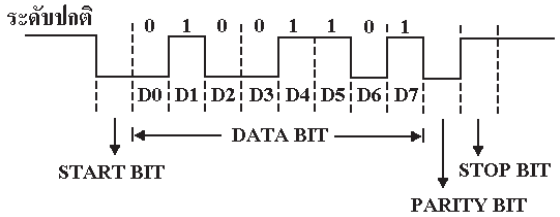
การสื่อสารแบบอะซิงโครนัส (Asynchronous)

การสื่อสารแบบนี้ใช้มากในเครื่องไมโครคอมพิวเตอร์พีซี รูปแบบการส่งข้อมูลจะเป็นการส่งครั้งละ 1 ไบต์ โดยมีรูปแบบดังนี้



การสื่อสารแบบอะซิงโครนัส (Asynchronous) (ต่อ)

ความหมายของบิต



Start Bit บอกจุดเริ่มต้นข้อมูล มีขนาด 1 บิต

Data Bit ค่าข้อมูลมีได้ 5 ถึง 8 บิต

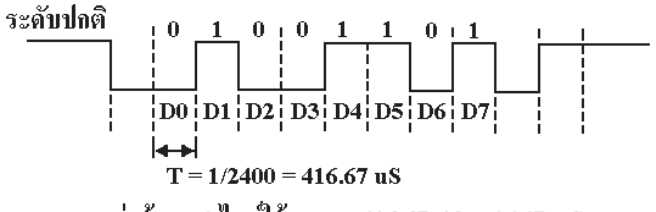
Parity Bit บิตสำหรับใช้ตรวจสอบความผิดพลาดของข้อมูล มีได้ 0 ถึง 1 บิต

Stop Bit บิตใช้บอกจุดสิ้นสุดข้อมูล มีได้ 1 1.5 และ 2 บิต

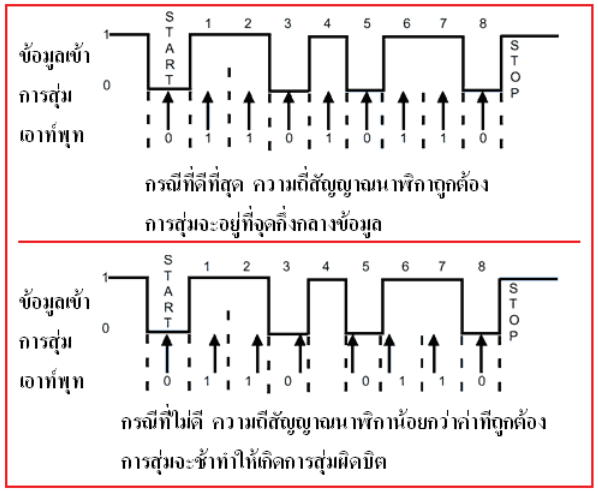
ความเร็วในการสื่อสาร

ความเร็วในการสื่อสาร หมายถึงจำนวนบิตที่ใช้รับส่งข้อมูลต่อวินาที โดยปกติจะมีค่าเท่ากับ 110 150 300 1200 2400 4800 9600 และ 19200 บิตต่อวินาที อัตราความเร็วนี้บางครั้งก็เรียกว่าอัตราบอด (Baud rate) ทั้งตัวส่งและตัวรับต้องกำหนดให้มีความเร็วในการสื่อสารเท่ากัน

ตัวอย่างข้อมูลส่งด้วยความเร็ว 2400 บิตต่อวินาที ดังนั้นแต่ละบิตใช้เวลาส่งเท่ากับ 1/2400 = 416.67 ไมโครวินาที

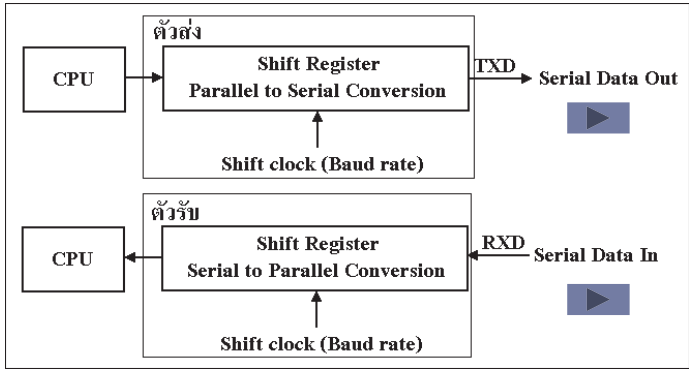


การสุ่มข้อมูลแบบอะซิงโครนัส

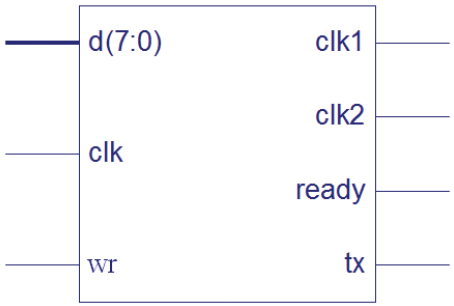


การสุ่มข้อมูลแบบอะซิงโครนัส (Asynchronous)

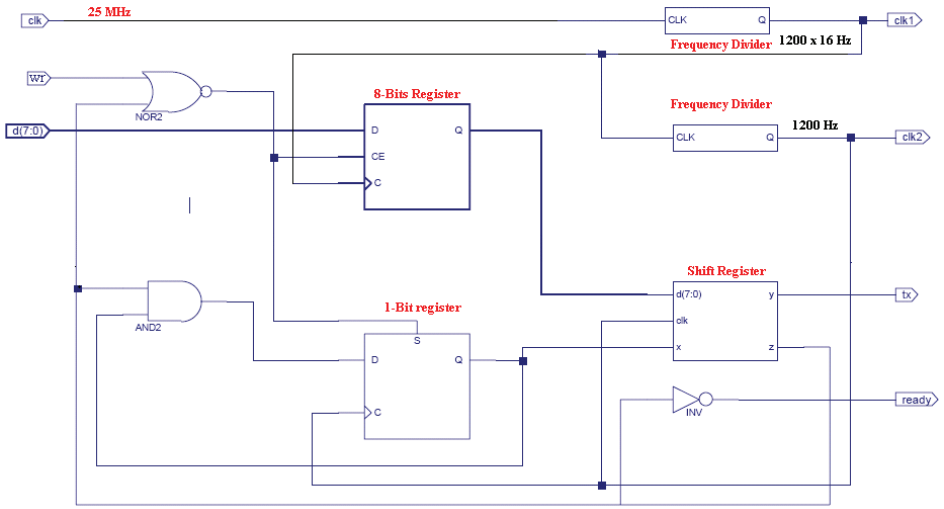
หลักการรับ-ส่งข้อมูลแบบอนุกรม



ตัวอย่างภาคส่งข้อมูลแบบอนุกรม



Block Diagram



การสื่อสารแบบข้อมูลแบบไอโซโครนัส (Isochronous Transmission)

มาจากรากศัพท์ในภาษากรีก 2 คำ คือคำว่า iso หมายถึง เท่ากัน และคำว่า chronous ที่หมาย ถึง เวลา เมื่อนำมารวมกันจึงหมายความว่า เวลาที่เท่ากัน สำหรับคุณสมบัติที่สำคัญของการส่งข้อมูลแบบไอโซโครนัส คือ การส่งผ่านข้อมูลด้วยความเร็วสูงใน อัตราคงที่ และรับประกันเวลาในการส่ง

เนื่องจากการส่งข้อมูลแบบเรียลไทม์ เช่น ระบบออดิโอและวิดีโอ จำเป็นต้องส่งข้อมูล ด้วยความเร็วสูง ซึ่งการส่งข้อมูลแบบอะซิงโครนัส (มีการหน่วงเวลาเกิดขึ้นจากช่องว่างระหว่าง เฟรม) และซิงโครนัสก็ยังไม่สามารถรองรับได้ จึงเกิดการส่งข้อมูลแบบไอโซโครนัสขึ้นมา เพื่อใช้งานเรียลไทม์ ที่รับประกันข้อมูลที่จะส่งมาถึงด้วยอัตราเร็วคงที่

โดยจะนำการส่งข้อมูลแบบไอโซโครนัสมาใช้เพื่อส่งผ่านข้อมูลบนบัส 1394 หรือ เรียกว่า ไฟร์ไวร์ (FireWire) การส่งผ่านข้อมูลของไอโซโครนัสจะตั้งอยู่บนพื้นฐานของแพ็กเก็ต โดยขนาดของแพ็กเก็ตจะส่งผ่านอยู่บนแชนเนลที่ให้ไว้ และสามารถแปรผันจากเฟรมไปยังเฟรมได้ ส่วนขนาดของแพ็กเก็ตจะถูกจำกัดโดยแบนด์วิดธ์เท่าที่มีอยู่

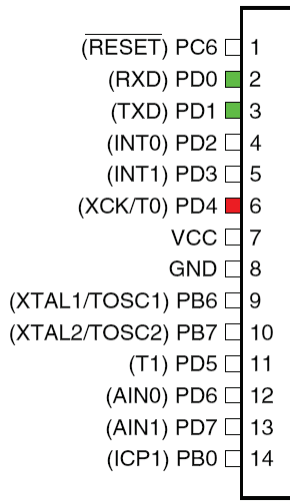
จาก

http://teacher.aru.ac.th/sawita/images/stories/file/PDF_Presentation/3503103_Networking/02092010_Network_Chapter5.pdf

การสื่อสารข้อมูลแบบอนุกรมของไมโครคอนโทรลเลอร์ AVR

ไมโครคอนโทรลเลอร์ AVR สามารถสื่อสารข้อมูลแบบอนุกรมได้โดยใช้โมดูล USART ((Universal Synchronous and Asynchronous serial Receiver and Transmitter) สำหรับ ATMega 16 ขาพอร์ทอนุกรมกำหนดไว้ที่

PD0	Serial input	RxD
PD1	Serial output	TxD

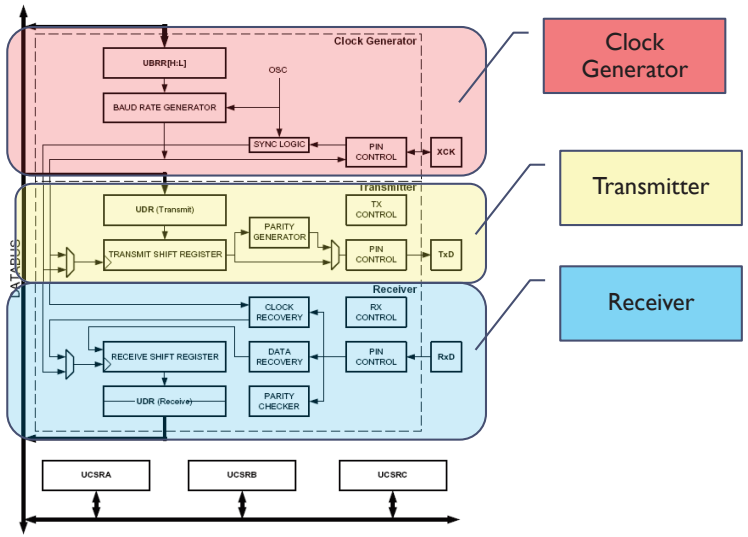


คุณลักษณะของ ATMega168-USART

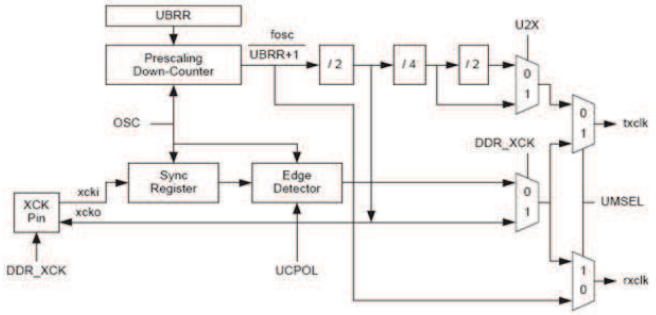
USART - Universal Synchronous Asynchronous Receiver Transmitter.

- ทำงานแบบ Full Duplex (การรับและส่งเป็นอิสระซึ่งกันและกัน)
- ทำงานได้ทั้งแบบ Asynchronous และ Synchronous
- Master or Slave Clocked Synchronous Operation
- High Resolution Baud Rate Generator
- รองรับการรับส่งข้อมูลแบบ 5, 6, 7, 8, or 9 Data Bits และ 1 หรือ 2 Stop Bits
- Odd or Even Parity Generation and Parity Check Supported by Hardware
- Data OverRun Detection
- Framing Error Detection
- Three Separate Interrupts on TX Complete, TX Data Register
- Empty and RX Complete

บล็อกไดอะแกรมอย่างง่ายๆของพอร์ทอนุกรม ATMega168 - USART



Clock Generation Logic, Block Diagram



Signal description:

- txclk Transmitter clock (internal signal).
- rxclk Receiver base clock (internal signal).
- xcki Input from XCK pin (internal signal). Used for synchronous slave operation.
- xcko Clock output to XCK pin (internal signal). Used for synchronous master operation.
- fosc System clock frequency.

การคำนวณหาอัตราบอต (Baud rate)

โหมดการทำงาน	อัตราบอต	ค่ารีจิสเตอร์ UBRR
อะซิงโครนัสปกติ (U2X = 0)	Baud = fosc/(16*(UBRR+1))	UBRR = (fosc/16*Baud)-1
อะซิงโครนัสทวีคูณ (U2X = 1)	Baud = fosc/(8*(UBRR+1))	UBRR = (fosc/8*Baud)-1
มาสเตอร์ซิงโครนัส	Baud = fosc/(2*(UBRR+1))	UBRR = (fosc/2*Baud)-1

UBRRnL and UBRRnH – USART baud rate registers

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Bit 15:12 – บิตสำรองเพื่อการใช้งาน บิตเหล่านี้ต้องให้เป็น 0

Bit 11:0 – UBRR11:0: USART บิตกำหนดอัตราบอด (baud rate) มีค่าได้ตั้งแต่ 0 - 4095

ตัวอย่างการกำหนดค่าอัตราบอต

$$\text{Error}[\%] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1\right) \bullet 100\%$$

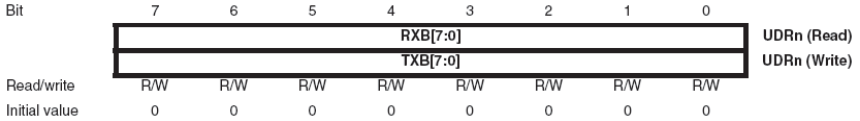
Baud rate (bps)	f_{osc} = 8.0000MHz			
	U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error
2400	207	0.2%	416	-0.1%
4800	103	0.2%	207	0.2%
9600	51	0.2%	103	0.2%
14.4k	34	-0.8%	68	0.6%
19.2k	25	0.2%	51	0.2%
28.8k	16	2.1%	34	-0.8%
38.4k	12	0.2%	25	0.2%
57.6k	8	-3.5%	16	2.1%
76.8k	6	-7.0%	12	0.2%
115.2k	3	8.5%	8	-3.5%
230.4k	1	8.5%	3	8.5%
250k	1	0.0%	3	0.0%
0.5M	0	0.0%	1	0.0%
1M	–	–	0	0.0%
Max.(1)	0.5Mbps		1Mbps	

รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานในโหมด USART

1. รีจิสเตอร์ UDR (USART I/O Data Register)
2. รีจิสเตอร์ UCSRA (USART Control and Status Register A)
3. รีจิสเตอร์ UCSRB (USART Control and Status Register B)
4. รีจิสเตอร์ UCSRC (USART Control and Status Register C)
5. รีจิสเตอร์ UBRRL และ UBRRH (USART Baud Rate Register)

รีจิสเตอร์ UDR (USART I/O Data Register)

- รีจิสเตอร์สำหรับอ่าน/เขียนข้อมูลขนาด 8 บิต โดยแบ่งเป็น RXB ใช้รับข้อมูลจากภายนอกและ TXB ใช้ส่งข้อมูลให้ภายนอก



รีจิสเตอร์ UCSRA (USART Control and Status Register A)

- รีจิสเตอร์ควบคุมการทำงานและแสดงสถานการณ์ทำงานของโมดูล USART ชุด A เกี่ยวข้องกับสถานะของการสื่อสารข้อมูล

บิตที่	7	6	5	4	3	2	1	0
ชื่อบิต	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM
Read/Write	R	R/W	R	R	R	R	R/W	R/W
ค่าเริ่มต้น	0	0	1	0	0	0	0	0

บิตที่ 7 - RXC: USART รับสมบูรณ์ (USART Receive Complete)
บิตนี้จะเป็น 1 เมื่อได้รับข้อมูลจากบัฟเฟอร์รับข้อมูล และจะเป็น 0 เมื่อบัฟเฟอร์ว่าง และถ้าใช้การอินเตอร์รัพท์ เมื่อบิตนี้เป็น 1 จะส่งสัญญาณไปอินเตอร์รัพท์ซีพียู

รีจิสเตอร์ UCSRA (USART Control and Status Register A)

บิตที่ 6 - TXC: USART ส่งสมบูรณ์ (USART Transmit Complete)
บิตนี้จะเป็น 1 เมื่อข้อมูลในบัฟเฟอร์ถูกส่งออกไปแล้วและยังไม่มีข้อมูลใหม่เข้ามา และจะเป็น 0 เมื่อบัฟเฟอร์ว่าง และถ้าใช้การอินเตอร์รัพท์ เมื่อบิตนี้เป็น 1 จะส่งสัญญาณไปอินเตอร์รัพท์ซีพียู

บิตที่ 5 – UDRE (USART data register empty)
บิตนี้จะเป็น 1 เมื่อ รีจิสเตอร์ข้อมูล UDR ว่าง พร้อมจะรับข้อมูลตัวใหม่

บิตที่ 4 – FE (Frame error)
บิตนี้จะเป็น 1 เมื่อ เฟรมข้อมูลผิดพลาด หรือบิตหยุดข้อมูลเป็น 0

รีจิสเตอร์ UCSRA (USART Control and Status Register A)

บิตที่ 3 – DOR (Data OverRun)
บิตนี้จะเป็น 1 เมื่อเกิดข้อผิดพลาดแบบ OverRun คือมีข้อมูลใหม่เข้ามาในขณะที่ข้อมูลเดิมยังไม่ถูกอ่านออกไป

บิตที่ 2 – PE (Parity Error)
บิตนี้จะเป็น 1 เมื่อเกิดข้อผิดพลาดแบบพาริตี้

บิตที่ 1 – U2X (Double the USART transmission speed)
ใช้กำหนดอัตราทวีคูณของการสื่อสาร

บิตที่ 0 – MPCM (Multi-processor communication mode)
บิตนี้เป็น 1 เมื่อต้องการใช้การสื่อสารแบบมัลติโปรเซสเซอร์

รีจิสเตอร์ UCSRB (USART Control and Status Register B)

- รีจิสเตอร์ควบคุมการทำงานและแสดงสถานการณ์ทำงานของโมดูล USART ชุด B เกี่ยวข้องกับการอินเตอร์รัพท์และขนาดของข้อมูลแบบ 9 บิต

บิตที่	7	6	5	4	3	2	1	0
ชื่อบิต	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
ค่าเริ่มต้น	0	0	0	0	0	0	0	0

บิตที่ 7 - RXCIE: (RX complete interrupt enable)
ให้บิตนี้เป็น 1 เมื่อต้องการให้เกิดการอินเตอร์รัพท์ เมื่อรับข้อมูลเข้ามา
บิตที่ 6 - TXCIE: (TX complete interrupt enable)
ให้บิตนี้เป็น 1 เมื่อต้องการให้เกิดการอินเตอร์รัพท์ เมื่อส่งข้อมูลแล้ว

รีจิสเตอร์ UCSRB (USART Control and Status Register B)

บิตที่ 5 - UDRIE: USART data register empty interrupt enable
ให้บิตนี้เป็น 1 เมื่อต้องการให้เกิดการอินเตอร์รัพท์ เมื่อรีจิสเตอร์ UDR ว่าง

บิตที่ 4 – RXEN: Receiver enable
ให้บิตนี้เป็น 1 เมื่อต้องการให้เกิดการรับข้อมูล

บิตที่ 3 – TXEN: Transmitter enable
ให้บิตนี้เป็น 1 เมื่อต้องการให้เกิดการส่งข้อมูล

บิตที่ 2 UCSZ2: Character size
ใช้กำหนดจำนวนบิตข้อมูล ใช้คู่กับ UCSZ1 และ UCSZ0 ใน UCSRC

บิตที่ 1 RXB8: Receive data bit 8
บิตที่ 8 ของการรับข้อมูล แบบ 9 บิต

บิตที่ 0 TXB8: Transmit data bit 8
บิตที่ 8 ของการส่งข้อมูล แบบ 9 บิต

รีจิสเตอร์ UCSRC (USART Control and Status Register C)

- รีจิสเตอร์ควบคุมการทำงานและแสดงสถานการณ์ทำงานของโมดูล USART ชุด C เกี่ยวข้องกับอัตราบอดในการรับส่ง

บิตที่	7	6	5	4	3	2	1	0
ชื่อบิต	UMSEL1	UMSEL0	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ค่าเริ่มต้น	0	0	0	0	0	1	1	0

บิตที่ 7:6 – UMSEL1:0 USART mode select

UMSEL1	UMSEL0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(Reserved)
1	1	Master SPI (MSPIM)(1)

รีจิสเตอร์ UCSRC (USART Control and Status Register C)

บิตที่ 5:4 – UPM1:0: Parity mode

บิตที่ 3 – USBS: Stop bit select

USBS	Stop bit(s)
0	1-bit
1	2-bit

บิตที่ 2:1 – UCSZ1:0: Character size
ใช้ร่วมกับกับ UCSZ2 ใน UCSRB

บิตที่ 0 – UCPOL: Clock polarity
ในแบบอะซิงโครนัสให้บิตนี้เป็น 0
ถ้าใช้ในแบบ ซิงโครนัส เป็นการกำหนด
ขอบสัญญาณ XCX

UPM1	UPM0	Parity mode
0	0	Disable
0	1	Reserved
1	0	Enable พาริตี้คู่
1	1	Enable พาริตี้คี่

UCSZ2	UCSZ1	UCSZ0	Character size
0	0	0	5 บิต
0	0	1	6 บิต
0	1	0	7 บิต
0	1	1	8 บิต
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9 บิต

UCPOL	Transmitted data changed (output of TxDn pin)	Received data sampled (input on RxDn pin)
0	Rising XCK edge	Falling XCK edge
1	Falling XCK edge	Rising XCK edge

ตัวอย่างการเขียนโปรแกรมใช้งาน USART

- ความถี่สัญญาณนาฬิกาของระบบเป็น 16 MHz
- ต้องการให้รับส่งข้อมูลแบบ Asynchronous ด้วยความเร็ว (Baud rate) 9600 บิตต่อวินาที
- ไม่มีบิตพาริตี้ ใช้ Stop bit 1 บิต

1. จาก Baud rate 9600 บิตต่อวินาทีคำนวณหาค่า UBRR

   ใช้อะซิงโครนัสปกติ (U2X = 0)

   UBRR = (fosc/16*Baud)-1 = (16,000,000/(16*9600)) – 1 = 103

ดังนั้น

UBRR0H = 0x00

UBRR0L = 103 หรือ 0x67

ตัวอย่างการเขียนโปรแกรมใช้งาน USART (ต่อ)

- กำหนดค่า UCSR0A โดยพิจารณาจาก

  ใช้อะซิงโครนัสปกติ U2X = 0

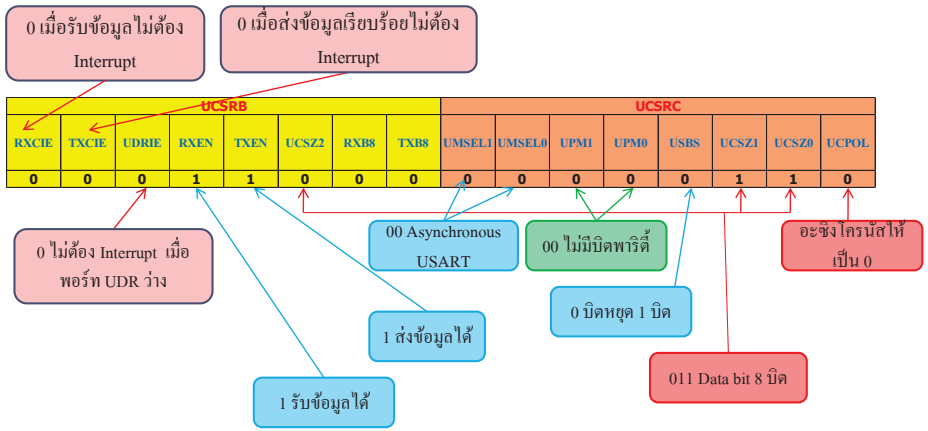
  UCSR0A = 0

บิตที่	7	6	5	4	3	2	1	0
ชื่อบิต	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM
กำหนดค่า	0	0	0	0	0	0	**0**	0

บิตสถานะได้จากการทำงาน ไม่ต้องกำหนด

ตัวอย่างการเขียนโปรแกรมใช้งาน USART (ต่อ)

- กำหนดค่า UCSRB และ UCSRC
- UCSR0B = 0b00011000 หรือ 0x18 UCSR0C = 0b00000110 หรือ 0x06

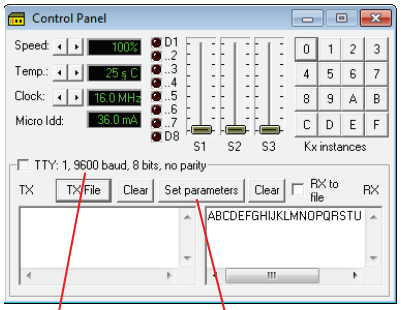


ตัวอย่างโปรแกรม Serial Output

```
#include <avr\io.h>              // Most basic include files
#include <util/delay.h>

void serial_init(void)
{
     // Set the baud rate and operation mode

    UBRR0H = 0;
    UBRR0L = 103;
    UCSR0A = 0;
    UCSR0B = 0b10011000;
    UCSR0C = 0b00000110;
     return;
}

void serial_write(char c)
{
      while ( !(UCSR0A & (1 << UDRE0)) );
      UDR0 = c;
}
```


ตัวอย่างโปรแกรม Serial Output (ต่อ)

```
int main (void)
{
    char i = 0x41;
    serial_init();
    while (1)
    {
        serial_write(i++);
        if (i >=91 )
        {
            i = 0x41;
            _delay_ms(300);
        }
    }
}
```



ตรวจสอบ Baud rate

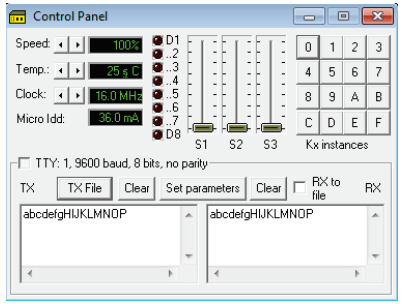
ตรวจสอบการทำงาน

ตัวอย่างโปรแกรม Serial Input – Output

```
#include <avr\io.h>              // Most basic include
files
#include <util/delay.h>

void serial_init(void)
{
     // Set the baud rate and operation mode

    UBRR0H = 0;
    UBRR0L = 103;
    UCSR0A = 0;
    UCSR0B = 0b10011000;
    UCSR0C = 0b00000110;
     return;
}

void serial_write(char c)
{
      while ( !(UCSR0A & (1 << UDRE0)) );
      UDR0 = c;
}
```


ตัวอย่างโปรแกรม Serial Input – Output (ต่อ)

```
int main (void)
{
    char i = 0x41;
    serial_init();
    while (1)
    {
        if ((UCSR0A & (1 << RXC0)))
        {
            i = UDR0;
            serial_write(i);
        }
    }
}
```



การจำลองการทำงานด้วย VMLAB

;X[inst_name] TTY(baud_rate [n_bits] [parity] [odd_parity] [n_stop_bits] [rx_display_as]) node_tx node_rx

X1 TTY(9600 8 0 0 1 1) PD0 PD1

Arduino : Serial

- if (Serial)

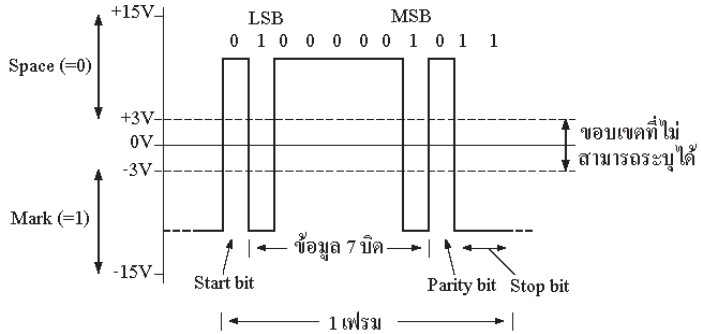
- available()
- begin()
- end()
- find()
- findUntil()
- flush()
- parseFloat()
- parseInt()
- peek()
- print()
- println()
- read()
- readBytes()
- readBytesUntil()
- setTimeout()
- write() begin()

อ้างอิง http://arduino.cc/en/Reference/Serial

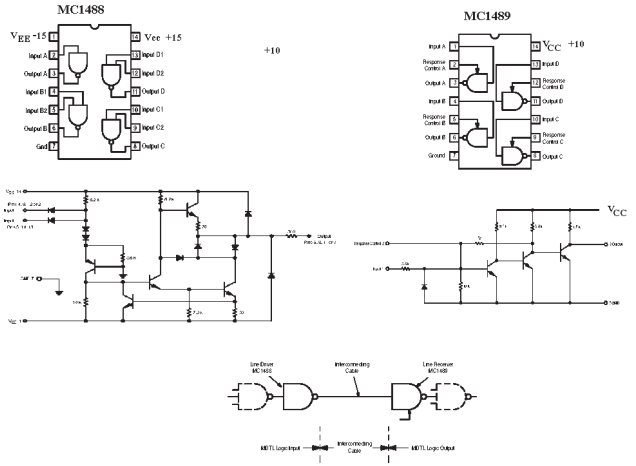
ตัวอย่างการส่งข้อมูล โดยใช้ Arduino

```
int analogValue = 0;    // variable to hold the analog value
void setup() {
    Serial.begin(9600);        // open the serial port at 9600 bps:
}

void loop() {
   analogValue = analogRead(0);     // read the analog input on pin 0:

  // print it out in many formats:
  Serial.println(analogValue);       // print as an ASCII-encoded decimal
  Serial.println(analogValue, DEC);  // print as an ASCII-encoded decimal
  Serial.println(analogValue, HEX);  // print as an ASCII-encoded hexadecimal
  Serial.println(analogValue, OCT);  // print as an ASCII-encoded octal
  Serial.println(analogValue, BIN);  // print as an ASCII-encoded binary

  // delay 10 milliseconds before the next reading:
  delay(10);
}
```

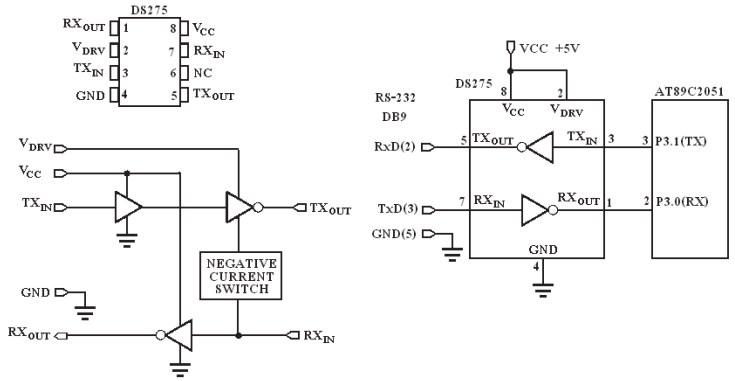

ระดับสัญญาณตามมาตราฐาน RS-232



Line EIA-232D Driver and Line Receivers



DS275 Line-Powered RS-232 Transceiver Chip



RXOUT	RS–232 Receiver Output	VDRV	Transmit driver +V
TXIN	RS–232 Driver Input	GND	System Ground (0V)
TXOUT	RS–232 Driver Output	NC	No Connection
RXIN	RS–232 Receive Input	VCC	System Logic Supply (+5V)

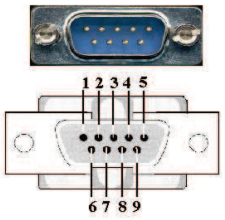
RS-232 Specs

SPECIFICATIONS	RS232
Mode of Operation	SINGLE-ENDED
Total Number of Drivers and Receivers on One Line	1 DRIVER1 RECVR
Maximum Cable Length	50 FT.
Maximum Data Rate	20kb/s
Maximum Driver Output Voltage	+/-25V
Driver Output Signal Level (Loaded Min.)Loaded	+/-5V to +/-15V
Driver Output Signal Level (Unloaded Max) Unloaded	+/-25V
Driver Load Impedance (Ohms)	3k to 7k
Max. Driver Current in High Z State (Power On)	N/A
Max. Driver Current in High Z State (Power Off)	+/-6mA @ +/-2v
Slew Rate (Max.)	30V/uS
Receiver Input Voltage Range	+/-15V
Receiver Input Sensitivity	+/-3V

RS-232 vs RS-423

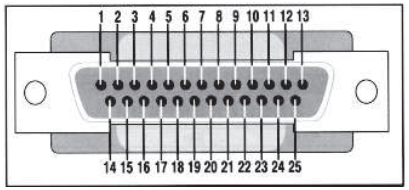
SPECIFICATIONS		RS232	RS423
Mode of Operation		SINGLE-ENDED	SINGLE-ENDED
Total Number of Drivers and Receivers on One Line		1 DRIVER 1 RECVR	1 DRIVER10 RECVR
Maximum Cable Length		50 FT.	4000 FT.
Maximum Data Rate		20kb/s	100kb/s
Maximum Driver Output Voltage		+/-25V	+/-6V
Driver Output Signal Level (Loaded Min.)	Loaded	+/-5V to +/-15V	+/-3.6V
Driver Output Signal Level (Unloaded Max)	Unloaded	+/-25V	+/-6V
Driver Load Impedance (Ohms)		3k to 7k	>=450
Max. Driver Current in High Z State Power On		N/A	N/A
Max. Driver Current in High Z State Power Off		+/-6mA @ +/-2v	+/-100uA
Slew Rate (Max.)		30V/uS	Adjustable
Receiver Input Voltage Range		+/-15V	+/-12V
Receiver Input Sensitivity		+/-3V	+/-200mV

RS232 Pin Assignments (DB9 PC signal set)



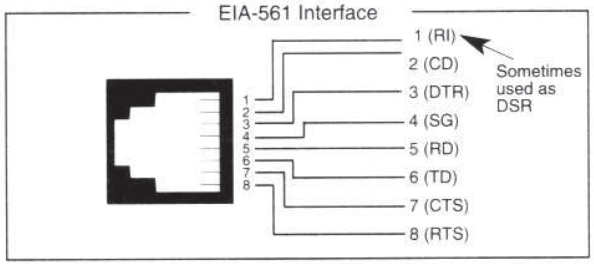
Pin 1 Received Line Signal Detector (Data Carrier Detect)	In
Pin 2 Received Data	In
Pin 3 Transmit Data	Out
Pin 4 Data Terminal Ready	Out
Pin 5 Signal Ground	
Pin 6 Data Set Ready	In
Pin 7 Request To Send	Out
Pin 8 Clear To Send	In
Pin 9 Ring Indicator	

RS232 Pin Assignments (DB25 PC signal set)

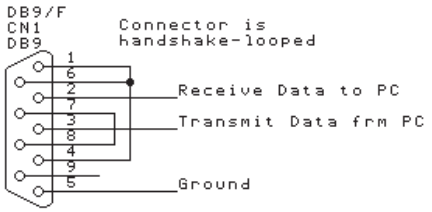


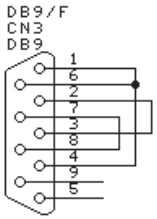
Pin 1 Protective Ground
Pin 2 Transmit Data
Pin 3 Received Data
Pin 4 Request To Send
Pin 5 Clear To Send
Pin 6 Data Set Ready
Pin 7 Signal Ground
Pin 8 Received Line Signal Detector (Data Carrier Detect)
Pin 20 Data Terminal Ready
Pin 22 Ring Indicator

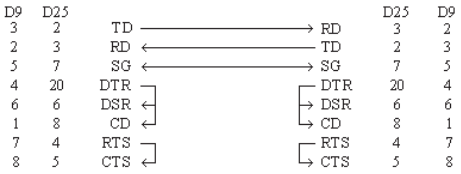
RS232 Pin Assignments on Modular Connector



การต่อสายแบบป้อนกลับ



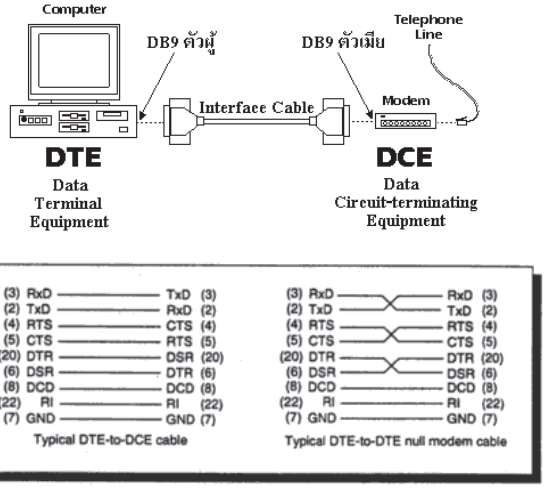




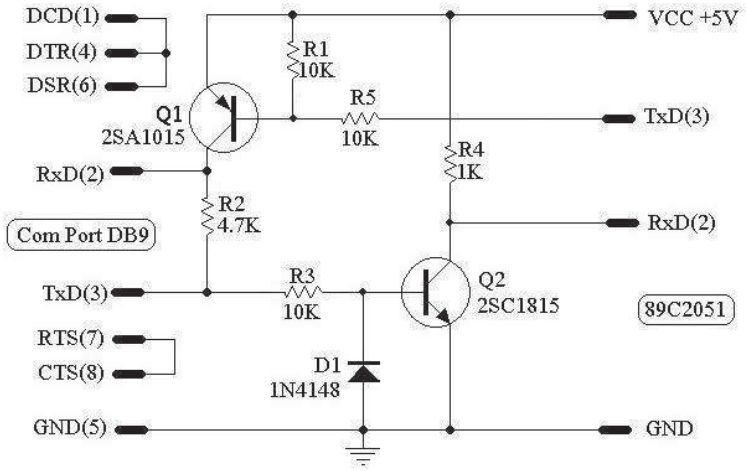
Handshake looping a PC serial connector

RS232 DB9 PC Loopback test plug

ตัวอย่างการต่อสายแบบครบ



วงจร RS232 Buffer ระหว่าง พีซีกับ ไมโครคอนโทรเลอร์โดยใช้ทรานซิสเตอร์



อ้างอิง

1. https://sites.google.com/site/eplearn/
2. http://arduino.cc/en/Reference/Serial