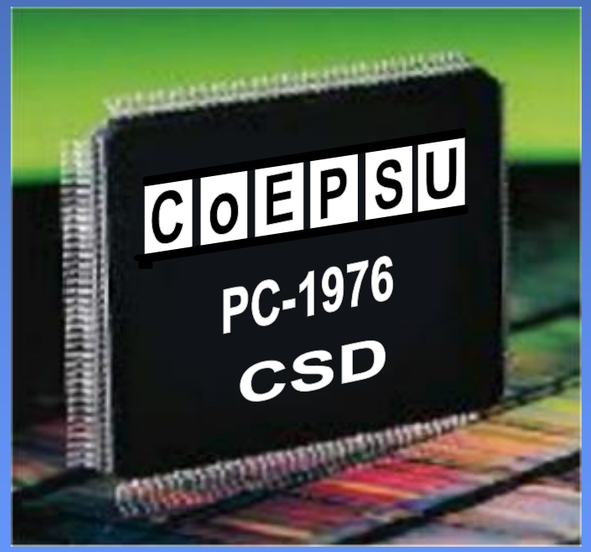
Department of Computer Engineering
Faculty of Engineering
Prince of Songkla University

240-319

Embedded System Developer Module



March 23, 2023

Associate Prof. Dr. Panyayot Chaikan
panyayot@coe.psu.ac.th

Chapter 6

การสื่อสารข้อมูลอนุกรมแบบอะซิงโครนัส

(Asynchronous Serial Communication)

เนื้อหา

การสื่อสารข้อมูลแบบอนุกรม

วงจร USART

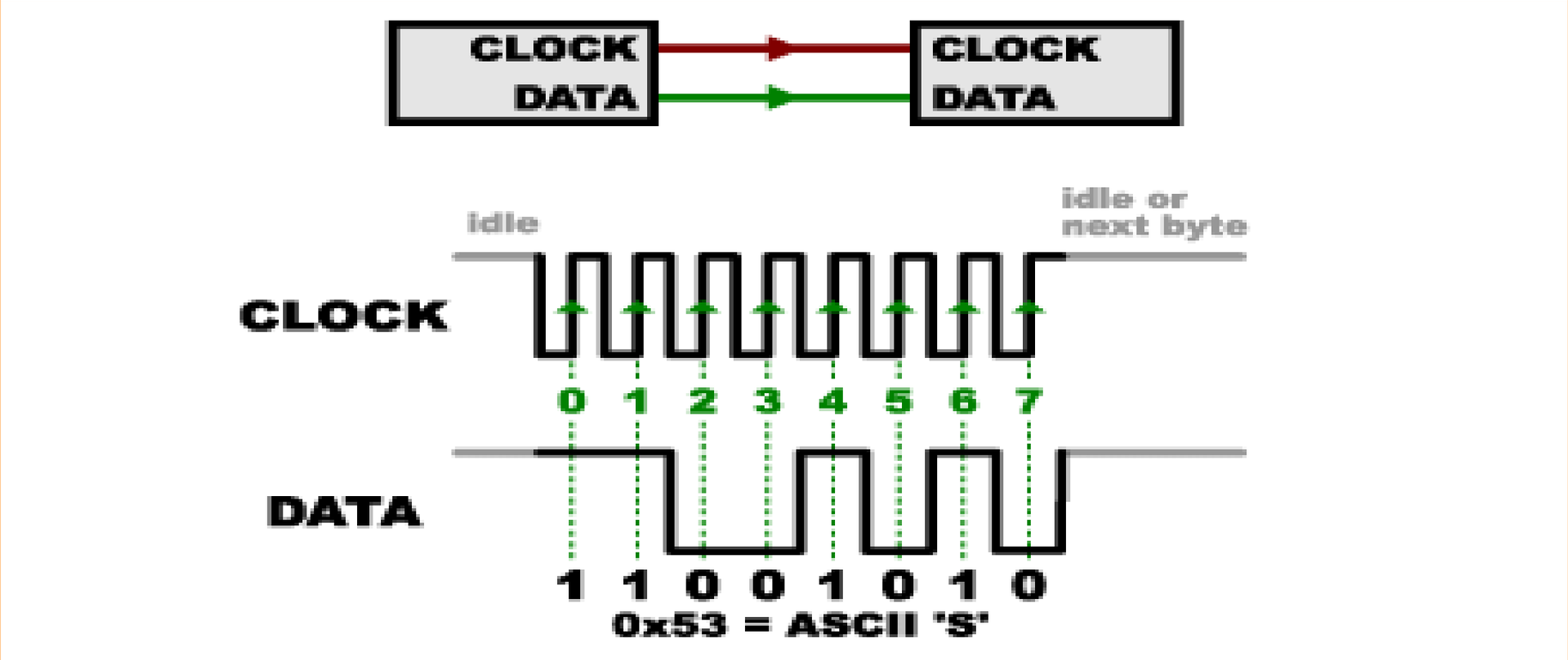
การสื่อสารข้อมูลอนุกรมแบบอะซิงโครนัส

เรจิสเตอร์ควบคุมการสื่อสารอนุกรม

การเขียนโปรแกรมสื่อสารข้อมูลอนุกรม

- การวนลูปตรวจสอบสถานะของการรับส่งข้อมูล
- การรับส่งข้อมูลโดยวิธีการอินเตอร์รัพต์

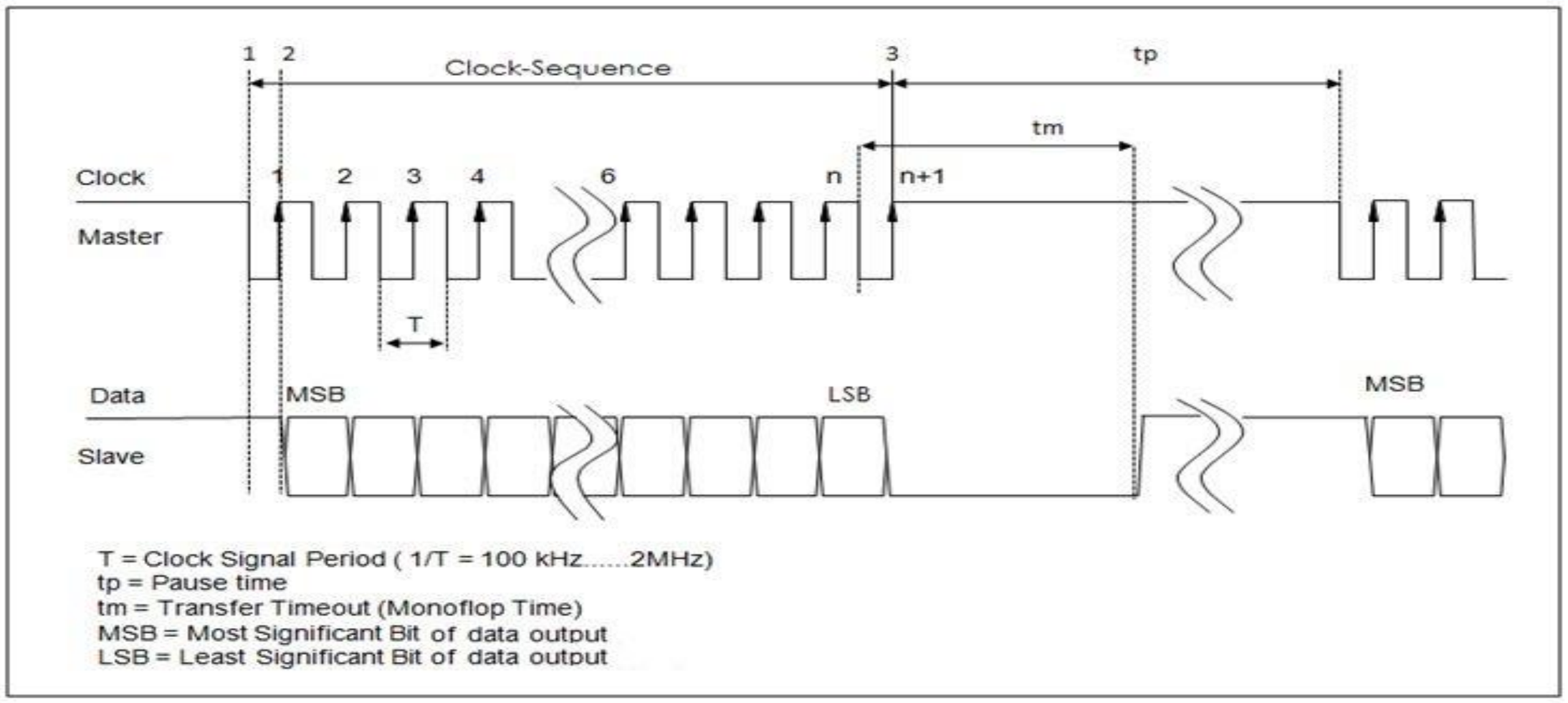
Synchronous Serial Communication



รูปจาก

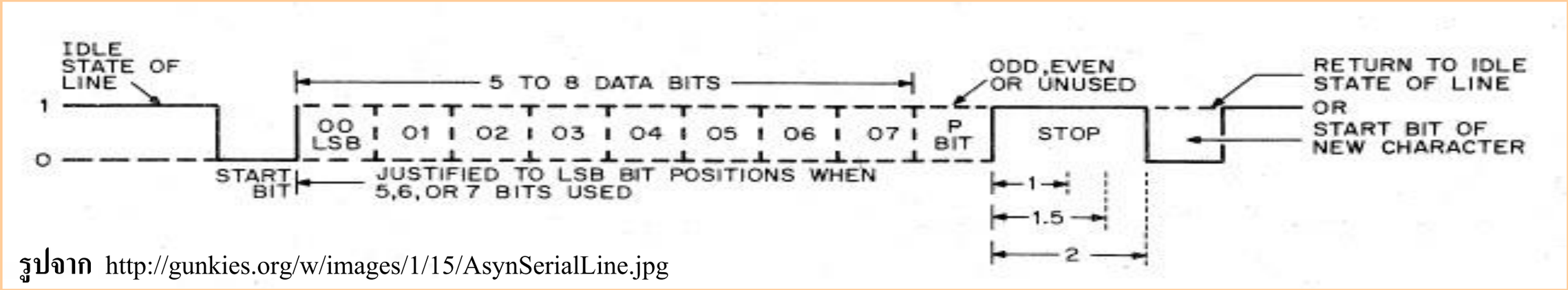
https://www.researchgate.net/profile/Ahmedul_Haque2/publication/315794059/figure/fig15/AS:479944775081992@1491439159812/An-example-of-synchronous-serial-communication-Mik.png

Synchronous Serial Communication

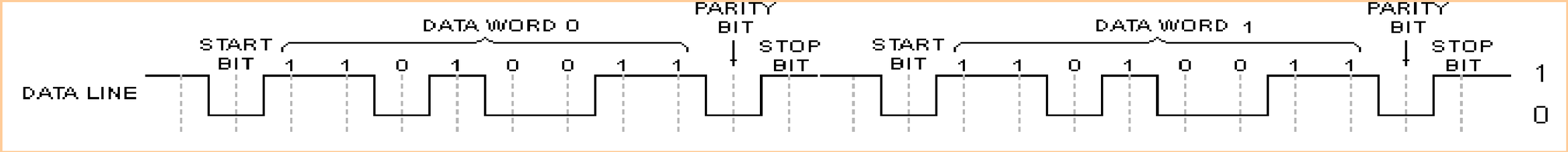


รูปจาก https://upload.wikimedia.org/wikipedia/commons/8/8d/Ssisingletransmission.jpg

Asynchronous Serial Communication



รูปจาก http://gunkies.org/w/images/1/15/AsynSerialLine.jpg



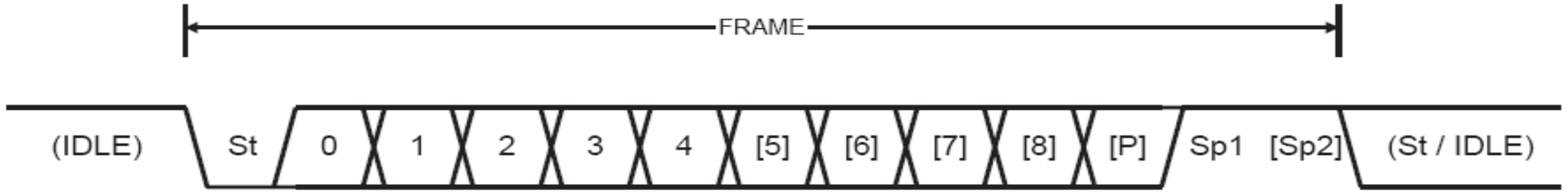
รูปจาก https://allpinouts.org/img/conn_rs232-f1.gif

ข้อตกลงระหว่างผู้รับและผู้ส่ง

- ส่งแบบ full or half-duplex
- จำนวนของบิตต่อหนึ่งตัวอักษร
- วิธีการเรียงบิต (endianness)
- ความเร็วในการรับส่ง (บิตต่อวินาที)
- มีการใช้บิตภาวะคู่หรือคี่ (Parity bit) หรือไม่
- หากมีการใช้บิตภาวะคู่หรือคี่ จะต้องตกลงกันว่าจะใช้บิตภาวะคู่ หรือ บิตภาวะคี่
- จำนวนบิตหยุด (Stop bit)

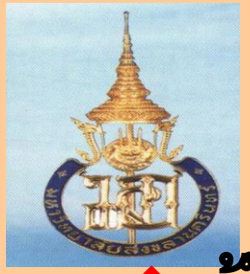
Asynchronous communication: Frame Format

- ผู้ใช้เลือกได้ถึง 30 รูปแบบการส่งใน AVR



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxDn or TxDn). An IDLE line must be high.

รูปจาก ATmega48A/48PA/88A/88PA/168A/168PA/328/328 datasheet, Atmel Corp, 2010.

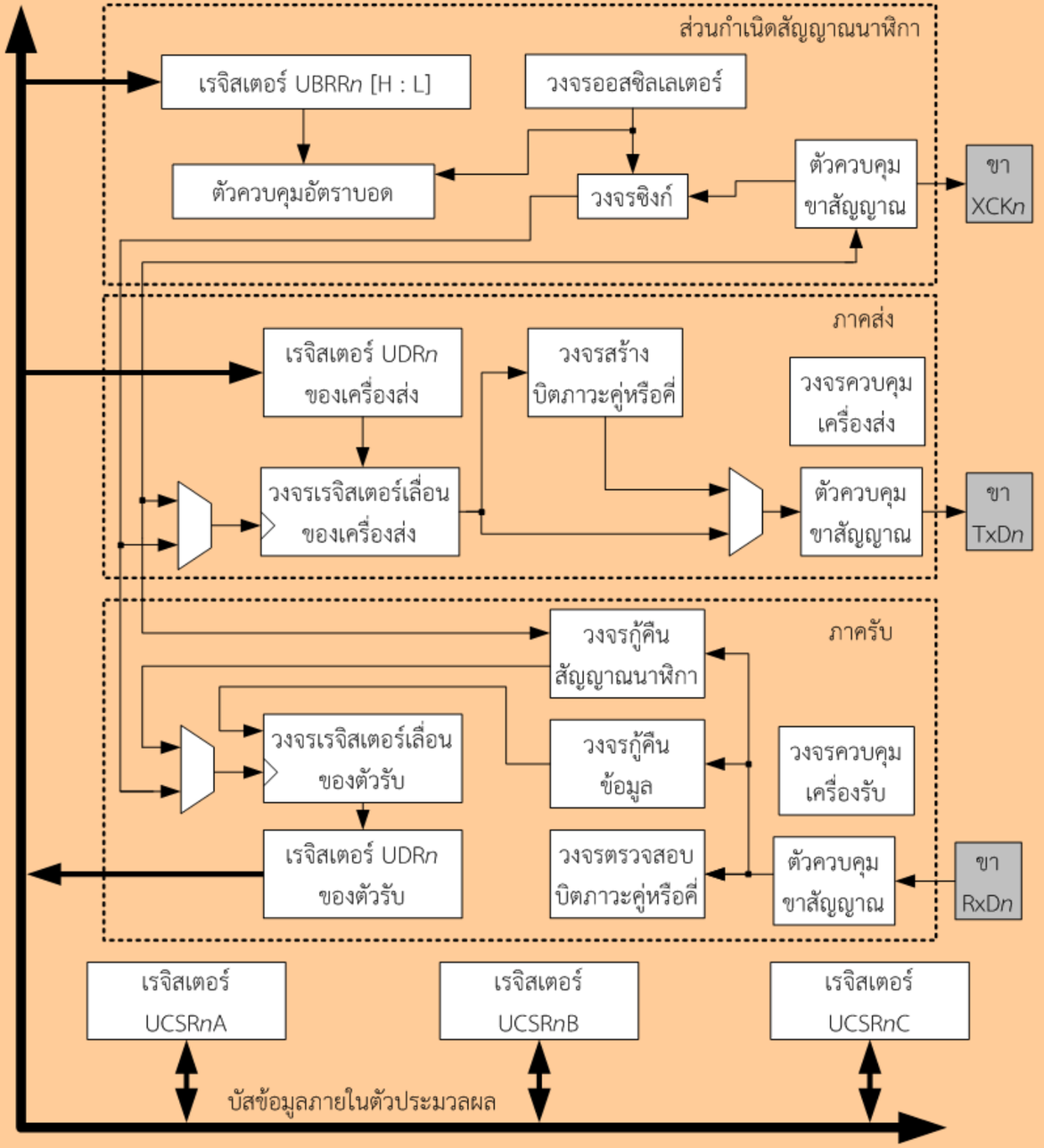


Asynchronous communication: Frame Format

- ผู้ใช้เลือกได้ถึง 30 รูปแบบการส่งใน AVR
- 1 start bit
- 5, 6, 7, 8, or 9 data bits
- no, even or odd parity bit
- 1 or 2 stop bits
- ใช้มอดูลยูสาร์ท (USART : Universal Synchronous /Asynchronous Receiver and Transmitter)

มอดูลยูสาร์ท

USART



ความสามารถของมอดูลยูสาร์ท

- รับส่งข้อมูลแบบสองทางเต็มอัตรา (Full Duplex)
- รับส่งข้อมูลแบบซิงโครนัสและอะซิงโครนัส
- วงจรกำเนิดอัตราบอดความเที่ยงตรงสูง
- สนับสนุนเฟรมข้อมูลขนาด 5, 6, 7, 8, or 9 บิต และบิตหยุดขนาด 1 และ 2 บิต
- เลือกบิตภาวะคู่หรือคี่ได้ หรือจะไม่ใช้บิตภาวะคู่หรือคี่ได้
- สามารถตรวจจับความผิดพลาดของเฟรมข้อมูลได้
- มีการกรองสัญญาณรบกวนของสัญญาณรับเข้า
- สนับสนุนการขัดจังหวะเมื่อส่งข้อมูลเสร็จ เมื่อข้อมูลในเรจิสเตอร์ส่งว่าง และเมื่อรับข้อมูลเสร็จ

เวกเตอร์การขัดจังหวะของ AVR

```
#define INT0_vect            _VECTOR(1)    /* External Interrupt Request 0 */
#define INT1_vect            _VECTOR(2)    /* External Interrupt Request 1 */
#define PCINT0_vect          _VECTOR(3)    /* Pin Change Interrupt Request 0 */
#define PCINT1_vect          _VECTOR(4)    /* Pin Change Interrupt Request 1 */
#define PCINT2_vect          _VECTOR(5)    /* Pin Change Interrupt Request 2 */
#define WDT_vect             _VECTOR(6)    /* Watchdog Time-out Interrupt */
#define TIMER2_COMPA_vect    _VECTOR(7)    /* Timer/Counter2 Compare Match A */
#define TIMER2_COMPB_vect    _VECTOR(8)    /* Timer/Counter2 Compare Match A */
#define TIMER2_OVF_vect      _VECTOR(9)    /* Timer/Counter2 Overflow */
#define TIMER1_CAPT_vect     _VECTOR(10)   /* Timer/Counter1 Capture Event */
#define TIMER1_COMPA_vect    _VECTOR(11)   /* Timer/Counter1 Compare Match A */
#define TIMER1_COMPB_vect    _VECTOR(12)   /* Timer/Counter1 Compare Match B */
#define TIMER1_OVF_vect      _VECTOR(13)   /* Timer/Counter1 Overflow */
#define TIMER0_COMPA_vect    _VECTOR(14)   /* TimerCounter0 Compare Match A */
#define TIMER0_COMPB_vect    _VECTOR(15)   /* TimerCounter0 Compare Match B */
#define TIMER0_OVF_vect      _VECTOR(16)   /* Timer/Couner0 Overflow */
#define SPI_STC_vect         _VECTOR(17)   /* SPI Serial Transfer Complete */
#define USART_RX_vect        _VECTOR(18)   /* USART Rx Complete */
#define USART_UDRE_vect      _VECTOR(19)   /* USART, Data Register Empty */
#define USART_TX_vect        _VECTOR(20)   /* USART Tx Complete */
#define ADC_vect             _VECTOR(21)   /* ADC Conversion Complete */
#define EE_READY_vect        _VECTOR(22)   /* EEPROM Ready */
#define ANALOG_COMP_vect     _VECTOR(23)   /* Analog Comparator */
#define TWI_vect             _VECTOR(24)   /* Two-wire Serial Interface */
#define SPM_READY_vect       _VECTOR(25)   /* Store Program Memory Read */
```

เรจิสเตอร์ควบคุมการทำงานของมอดูลยูสาร์ท

บิตที่ 7	บิตที่ 6	บิตที่ 5	บิตที่ 4	บิตที่ 3	บิตที่ 2	บิตที่ 1	บิตที่ 0	
RXC*n*	TXC*n*	UDRE*n*	FE*n*	DOR*n*	UPE*n*	U2X*n*	MPCM*n*	เรจิสเตอร์ UCSR*n*A
RXCIE*n*	TXCIE*n*	UDRIE*n*	RXEN*n*	TXEN*n*	UCSZ*n*2	RXB8*n*	TXB8*n*	เรจิสเตอร์ UCSR*n*B
UMSEL*n*1	UMSEL*n*0	UPM*n*1	UPM*n*0	USBS*n*	UCSZ*n*1	UCSZ*n*0	UCPOL*n*	เรจิสเตอร์ UCSR*n*C

UDRn – USART I/O Data Register n

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDRn (Read)
	TXB[7:0]								UDRn (Write)
Read/Write	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	

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	

UCSRnA – USART Control and Status Register n A

บิตที่ 7	บิตที่ 6	บิตที่ 5	บิตที่ 4	บิตที่ 3	บิตที่ 2	บิตที่ 1	บิตที่ 0
RXC*n*	TXC*n*	UDRE*n*	FE*n*	DOR*n*	UPE*n*	U2X*n*	MPCM*n*

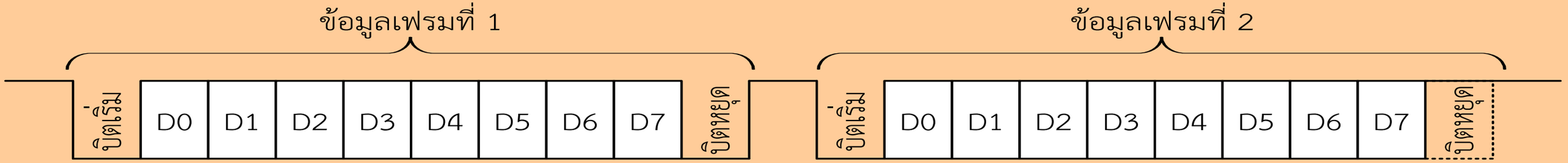
- RXC*n* ย่อมาจาก USART*n* Receive Complete
- TXC*n* ย่อมาจาก USART*n* Transmit Complete เป็นตัวบ่งชี้ซึ่งมีค่าเป็นตรรกะสูงก็ต่อเมื่อข้อมูลของเฟรมถูกส่งออกไปจากวงจรเรจิสเตอร์เลื่อนครบหมดแล้วทุกบิต รวมทั้งบิตภาวะคู่หรือคี่ (ถ้ามี) และบิตหยุดก็ถูกส่งออกไปหมดแล้วด้วยเช่นกัน
- บิต UDRE*n* ย่อมาจาก USART*n* Data Register Empty เป็นตัวบ่งชี้ว่าวงจรบัฟเฟอร์ภาคส่งพร้อมที่จะรับข้อมูลใหม่สำหรับส่งออกเป็นข้อมูลเฟรมถัดไป

UCSRnA – USART Control and Status Register n A

บิตที่ 7	บิตที่ 6	บิตที่ 5	บิตที่ 4	บิตที่ 3	บิตที่ 2	บิตที่ 1	บิตที่ 0
RXC*n*	TXC*n*	UDRE*n*	FE*n*	DOR*n*	UPE*n*	U2X*n*	MPCM*n*

- FEn ย่อมาจาก Frame Error n เป็นตัวบ่งชี้ว่าวงจรภาครับตรวจพบความผิดพลาดของเฟรมข้อมูล
- DORn ย่อมาจาก Data OverRun n เป็นตัวบ่งชี้ว่าเกิดสถานะที่ข้อมูลเฟรมที่ได้รับก่อนหน้านี้ยังมิได้ถูกนำไปใช้งาน แต่วงจรภาครับตรวจพบข้อมูลเฟรมถัดไปส่งเข้ามา
- UPEn ย่อมาจาก USART Parity Error n เป็นตัวบ่งชี้ว่าข้อมูลที่ได้รับเมื่อตรวจสอบกับบิตภาวะคู่หรือคี่แล้วเกิดความผิดพลาด

ตัวอย่างกรณีเฟรมข้อมูลเกิดความผิดพลาด



- เฟรมใดมีความผิดพลาดของข้อมูล ?

UCSRnA – USART Control and Status Register n A

บิตที่ 7	บิตที่ 6	บิตที่ 5	บิตที่ 4	บิตที่ 3	บิตที่ 2	บิตที่ 1	บิตที่ 0
RXC*n*	TXC*n*	UDRE*n*	FE*n*	DOR*n*	UPE*n*	U2X*n*	MPCM*n*

- U2X*n* มาจาก Double the USART Transmission Speed *n* ส่งผลให้ค่าอัตราบอดของการรับส่งข้อมูลสูงกว่าเดิมสองเท่า
- MPCM*n* ย่อมาจาก Multi-processor Communication Mode เป็นบิตควบคุมที่ใช้ในการเปิดทางการติดต่อสื่อสารกันระหว่างตัวประมวลผลหลายตัว

UCSRnB – USART Control and Status Register n B

บิตที่ 7	บิตที่ 6	บิตที่ 5	บิตที่ 4	บิตที่ 3	บิตที่ 2	บิตที่ 1	บิตที่ 0
RXCIE*n*	TXCIE*n*	UDRIE*n*	RXEN*n*	TXEN*n*	UCSZ*n*2	RXB8*n*	TXB8*n*

- RXCIE*n* ย่อมาจาก RX Complete Interrupt Enable *n* ใช้ในการเปิดทางการตอบรับการขัดจังหวะของตัวประมวลผลเมื่อรับข้อมูลครบหนึ่งเฟรม
- TXCIE*n* ย่อมาจาก TX Complete Interrupt Enable *n* ใช้ในการเปิดทางการตอบรับการขัดจังหวะของตัวประมวลผลเมื่อส่งข้อมูลเสร็จหนึ่งเฟรม
- UDRIE*n* ย่อมาจาก USART Data Register Empty Interrupt Enable *n* ใช้ในการเปิดทางการตอบรับการขัดจังหวะของตัวประมวลผลเมื่อข้อมูลในเรจิสเตอร์ UDR*n* ถูกส่งออกไปหมดแล้ว
- RXEN*n* ย่อมาจาก Receiver Enable *n* เป็นบิตที่ใช้ในการเปิดการทำงานของวงจรภาครับของ USART*n*

UCSRnB – USART Control and Status Register n B

บิตที่ 7	บิตที่ 6	บิตที่ 5	บิตที่ 4	บิตที่ 3	บิตที่ 2	บิตที่ 1	บิตที่ 0
RXCIE*n*	TXCIE*n*	UDRIE*n*	RXEN*n*	TXEN*n*	UCSZ*n*2	RXB8*n*	TXB8*n*

- TXEN*n* ย่อมาจาก Transmitter Enable *n* เป็นบิตที่ใช้ในการเปิดการทำงานของวงจรภาคส่ง
- UCSZ*n*2 ย่อมาจาก Character Size *n* 2 เป็นบิตที่ 2 ที่ใช้กำหนดขนาดของอักขระสำหรับการรับส่งข้อมูล
- RXB8*n* ย่อมาจาก Receive Data Bit 8 *n* เป็นที่สำหรับเก็บข้อมูลบิตที่มีนัยสำคัญสูงสุด ซึ่งก็คือบิตที่ 8 ของการรับข้อมูลอนุกรมที่รับเข้ามา
- TXB8*n* ย่อมาจาก Transmit Data Bit 8 *n* เป็นที่สำหรับเก็บข้อมูลบิตที่มีนัยสำคัญสูงสุด ซึ่งก็คือบิตที่ 8 ของการข้อมูลที่จะส่งออกไปทางขา TxD*n*

UCSRnC – USART Control and Status Register n C

บิตที่ 7	บิตที่ 6	บิตที่ 5	บิตที่ 4	บิตที่ 3	บิตที่ 2	บิตที่ 1	บิตที่ 0
UMSEL*n*1	UMSEL*n*0	UPM*n*1	UPM*n*0	USBS*n*	UCSZ*n*1	UCSZ*n*0	UCPOL*n*

- UMSEL*n* ย่อมาจาก USART Mode Select มีจำนวน 2 บิต คือ UMSEL*n*1 และ UMSEL*n*0 ใช้ในการเลือกแบบวิธีในการทำงานของมอดูลยูสาร์ท
- UPM*n* ย่อมาจาก USART Parity Mode *n* มีจำนวน 2 บิต คือ UPM*n*1 และ UPM*n*0 ใช้สำหรับการเปิดหรือปิดการใช้บิตภาวะคู่หรือคี่ในเฟรมข้อมูล
- USBS*n* ย่อมาจาก USART Stop Bit Select *n* ใช้สำหรับเลือกจำนวนของบิตหยุด
- UCSZn ย่อมาจาก USART Character Size n มีจำนวน 2 บิต คือ UCSZn1 และ UCSZn0 ใช้สำหรับกำหนดขนาดของเฟรมข้อมูล
- UCPOLn ย่อมาจาก USART Clock Polarity *n* บิตนี้ใช้งานเฉพาะกรณีที่วงจร USARTn ทำงานในแบบซิงโครนัส

การกำหนดขนาดของเฟรมข้อมูล

UCSZn2	UCSZn1	UCSZn0	ขนาดของอักขระที่ต้องการส่ง
0	0	0	5 บิต
0	0	1	6 บิต
0	1	0	7 บิต
0	1	1	8 บิต
1	1	1	9 บิต
หมายเหตุ ค่า 100_2-110_2 ถูกสงวนไว้ มิให้ใช้งาน			

การตั้งแบบวิธีการทำงานของมอดูลยูสาร์ท

UMSEL*n*1	UMSEL*n*0	แบบวิธีการทำงานของวงจร USART*n*
0	0	อะซิงโครนัส
0	1	ซิงโครนัส
1	0	ค่านี้ถูกสงวนไว้ มิให้ใช้งาน
1	1	มาสเตอร์เอสพีไอ (Master SPI)

การตั้งบิตภาวะคู่หรือคี่ของการรับส่งข้อมูล

UPM*n*1	UPM*n*0	แบบวิธีการกำหนดบิตภาวะคู่หรือคี่
0	0	ไม่ใช้งานบิตภาวะคู่หรือคี่
0	1	ค่านี้ถูกสงวนไว้ มิให้ใช้งาน
1	0	เปิดใช้งานบิตภาวะคี่
1	1	เปิดใช้งานบิตภาวะคู่

$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$

$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

การเลือกขนาดของบิตหยุด

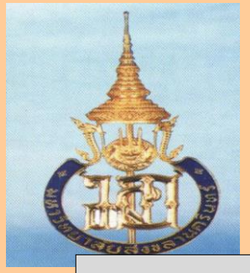
บิตที่ 7	บิตที่ 6	บิตที่ 5	บิตที่ 4	บิตที่ 3	บิตที่ 2	บิตที่ 1	บิตที่ 0
UMSEL*n*1	UMSEL*n*0	UPM*n*1	UPM*n*0	USBS*n*	UCSZ*n*1	UCSZ*n*0	UCPOL*n*

USBSn	Stop Bit(s)
0	1-bit
1	2-bit

การคำนวณค่า baud rate สำหรับเรจิสเตอร์ UBRR*n*

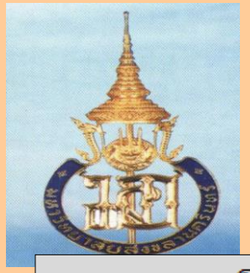
$$BAUD = \frac{2^{U2Xn} fosc}{16(UBRRn + 1)}$$

- BAUD คือ อัตราบอดในการรับส่งข้อมูล มีหน่วยเป็นบิตต่อวินาที
- fosc คือ ค่าความถี่ที่ไมโครคอนโทรลเลอร์กำลังทำงานอยู่
- UBRR*n* คือ ค่าที่ตั้งให้กับเรจิสเตอร์ UBRR*n*
- U2X*n* คือ ค่าตรรกะของบิตที่ 1 ในเรจิสเตอร์ UCSR*n*A ซึ่งหากเป็นตรรกะสูงจะมีค่าเท่ากับ 1 ในสมการนี้ แต่หากเป็นตรรกะต่ำจะมีค่าเป็น 0



ตัวอย่างการตั้งค่าอัตราบอด

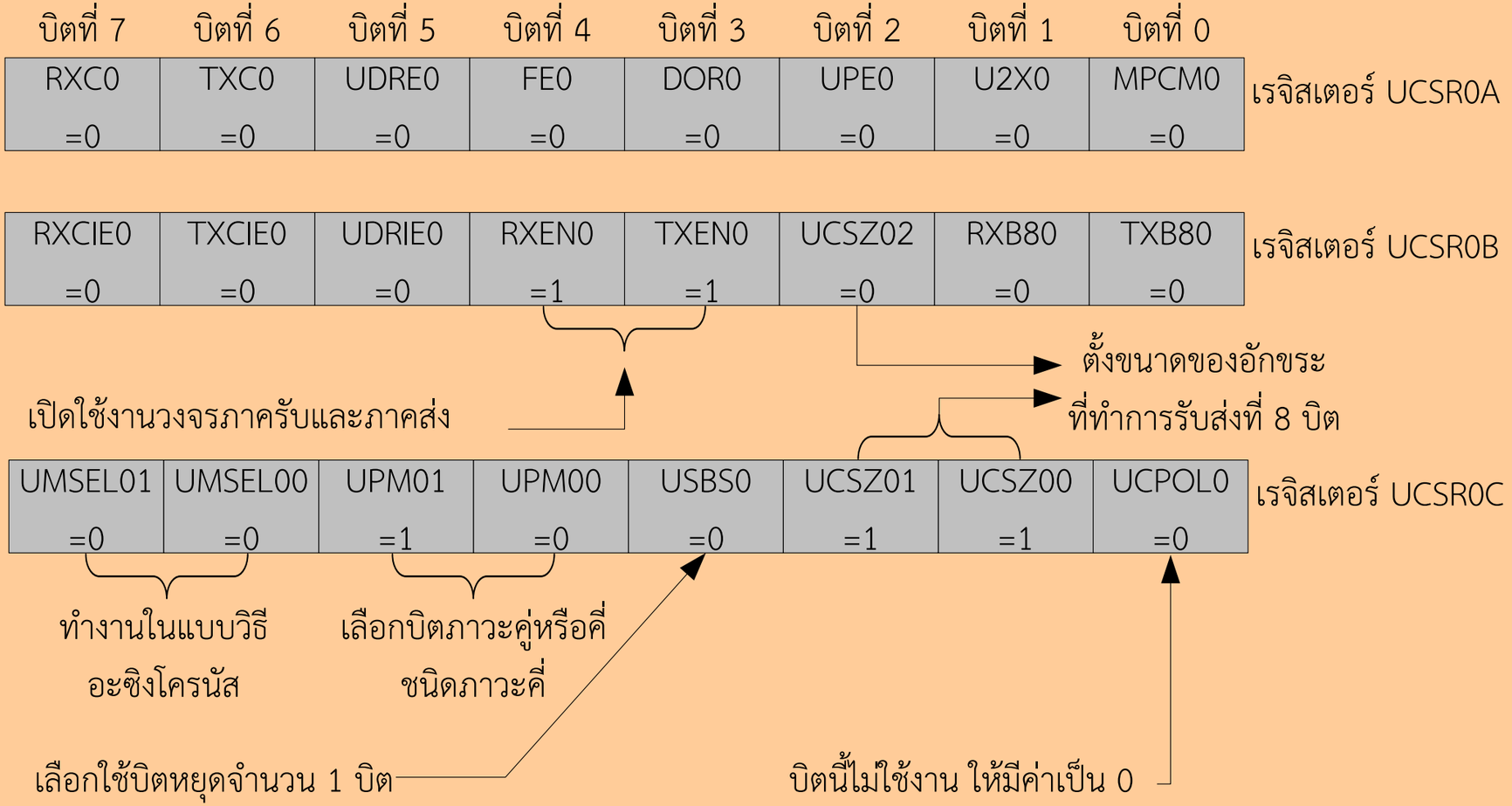
ความถี่ของตัวประมวลผล	ความเร็วในการรับ/ส่ง (bps)	ค่าที่ตั้งในเรจิสเตอร์ UBRR	สถานะของบิต U2X*n*	ความคลาดเคลื่อนของอัตราเร็วในการรับส่ง
16.0000 MHz	9600	103	0	0.16 %
16.0000 MHz	9600	207	1	0.16 %
16.0000 MHz	14400	68	0	0.64 %
16.0000 MHz	14400	138	1	-0.08 %
16.0000 MHz	19200	51	0	0.16 %
16.0000 MHz	19200	103	1	0.16 %
16.0000 MHz	28800	34	0	-0.79 %
16.0000 MHz	28800	68	1	3.52 %
16.0000 MHz	38400	25	0	0.16 %
16.0000 MHz	38400	51	1	0.16 %
16.0000 MHz	115200	8	0	-3.55 %
16.0000 MHz	115200	16	1	2.12 %
16.0000 MHz	230400	8	1	-3.55 %



ตัวอย่างการตั้งค่าอัตราบอด

ความถี่ของตัวประมวลผล	ความเร็วในการรับ/ส่ง (bps)	ค่าที่ตั้งในเรจิสเตอร์ UBRR	สถานะของบิต U2X*n*	ความคลาดเคลื่อนของอัตราเร็วในการรับส่ง
18.4320 MHz	9600	119	0	0.00 %
18.4320 MHz	9600	239	1	0.00 %
18.4320 MHz	14400	79	0	0.00 %
18.4320 MHz	14400	159	1	0.00 %
18.4320 MHz	19200	59	0	0.00 %
18.4320 MHz	19200	119	1	0.00 %
18.4320 MHz	28800	39	0	0.00 %
18.4320 MHz	28800	79	1	0.00 %
18.4320 MHz	38400	29	0	0.00 %
18.4320 MHz	38400	59	1	0.00 %
18.4320 MHz	115200	9	0	0.00 %
18.4320 MHz	115200	19	1	0.00 %
18.4320 MHz	230400	9	1	0.00 %

ตัวอย่างการตั้งค่ามอดูลยูสาร์ท



ตัวอย่างที่ 6.1 การตั้งค่าสำหรับ USART

Assembly Code Example(1)

```
USART_Init:
  ; Set baud rate
  out   UBRRnH, r17
  out   UBRRnL, r16
  ; Enable receiver and transmitter
  ldi   r16, (1<<RXENn)|(1<<TXENn)
  out   UCSRnB,r16
  ; Set frame format: 8data, 2stop bit
  ldi   r16, (1<<USBSn)|(3<<UCSZn0)
  out   UCSRnC,r16
  ret
```

C Code Example(1)

```
#define FOSC 1843200 // Clock Speed
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1
void main( void )
{
...
  USART_Init(MYUBRR)
...
}
void USART_Init( unsigned int ubrr)
{
  /*Set baud rate */
  UBRR0H = (unsigned char)(ubrr>>8);
  UBRR0L = (unsigned char)ubrr;
  Enable receiver and transmitter */
  UCSR0B = (1<<RXEN0)|(1<<TXEN0);
  /* Set frame format: 8data, 2stop bit */
  UCSR0C = (1<<USBS0)|(3<<UCSZ00);
}
```

- n ให้ใช้ค่า 0 (zero)

ตัวอย่างที่ 6.2 การตั้งค่า 9600bps บนซีพียู 10 MHz

```
//*************************************************************************
//USART_init: initializes the USART system
//*************************************************************************

void USART_init(void)
{
UCSRA = 0x00;                    //control register initialization
UCSRB = 0x08;                    //enable transmitter
UCSRC = 0x86;                    //async, no parity, 1 stop bit,
                                 //8 data bits
                                 //Baud Rate initialization

UBRRH = 0x00;
UBRRL = 0x40;
}
```

- เวลาใช้งานจริงต้องระบุด้วยว่าใช้ USART ตัวใด กรณี ATMEGA328P ค่า n ให้ใช้ค่า 0 (zero) เช่น UBRR0H, UBRR0L เป็นต้น

ตัวอย่างที่ 6.3 การส่งเฟรมข้อมูลขนาด 5-8 bit

Assembly Code Example

```
USART_Transmit:
  ; Wait for empty transmit buffer
  in r16, UCSRnA
  sbrs r16, UDREn
  rjmp USART_Transmit
  ; Put data (r16) into buffer, sends the data
  out  UDRn,r16
  ret
```

- กรณีใช้วิธี polling

C Code Example

```
void USART_Transmit( unsigned char data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRnA & (1<<UDREn)) )
        ;
  /* Put data into buffer, sends the data */
  UDRn = data;
}
```

ตัวอย่างที่ 6.4 การส่งเฟรมข้อมูลขนาด 9 bit

Assembly Code Example

```
USART_Transmit:
  ; Wait for empty transmit buffer
  in r16, UCSRnA
  sbrs r16, UDREn
  rjmp USART_Transmit
  ; Copy 9th bit from r17 to TXB8
  cbi  UCSRnB,TXB8
  sbrc r17,0
  sbi  UCSRnB,TXB8
  ; Put LSB data (r16) into buffer, sends the data
  out  UDRn,r16
  ret
```

C Code Example

```
void USART_Transmit( unsigned int data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRnA & (1<<UDREn)) )
        ;
  /* Copy 9th bit to TXB8 */
  UCSRnB &= ~(1<<TXB8);
  if ( data & 0x0100 )
    UCSRnB |= (1<<TXB8);
  /* Put data into buffer, sends the data */
  UDRn = data;
}
```

ตัวอย่างที่ 6.5 การรับข้อมูลจาก USART

Assembly Code Example

```
USART_Receive:
  ; Wait for data to be received
  in r16, UCSRnA
  sbrs r16, UDREn
  rjmp USART_Receive
  ; Get and return received data from buffer
  in   r16, UDRn
  ret
```

C Code Example

```
unsigned char USART_Receive( void )
{
  /* Wait for data to be received */
  while ( !(UCSRnA & (1<<RXCn)) )
        ;
  /* Get and return received data from buffer */
  return UDRn;
}
```

ตัวอย่างที่ 6.6

- จงเขียนโปรแกรมเพื่อส่งข้อมูล จาก AVR ออกทาง Serial port ซึ่งต่อกับเครื่องคอมพิวเตอร์ที่กำลังรันโปรแกรม Hyperterminal อยู่ ส่งข้อความ “Hello World”
- กำหนดค่า baudrate = 9600 bps, 1 stop bit, no parity

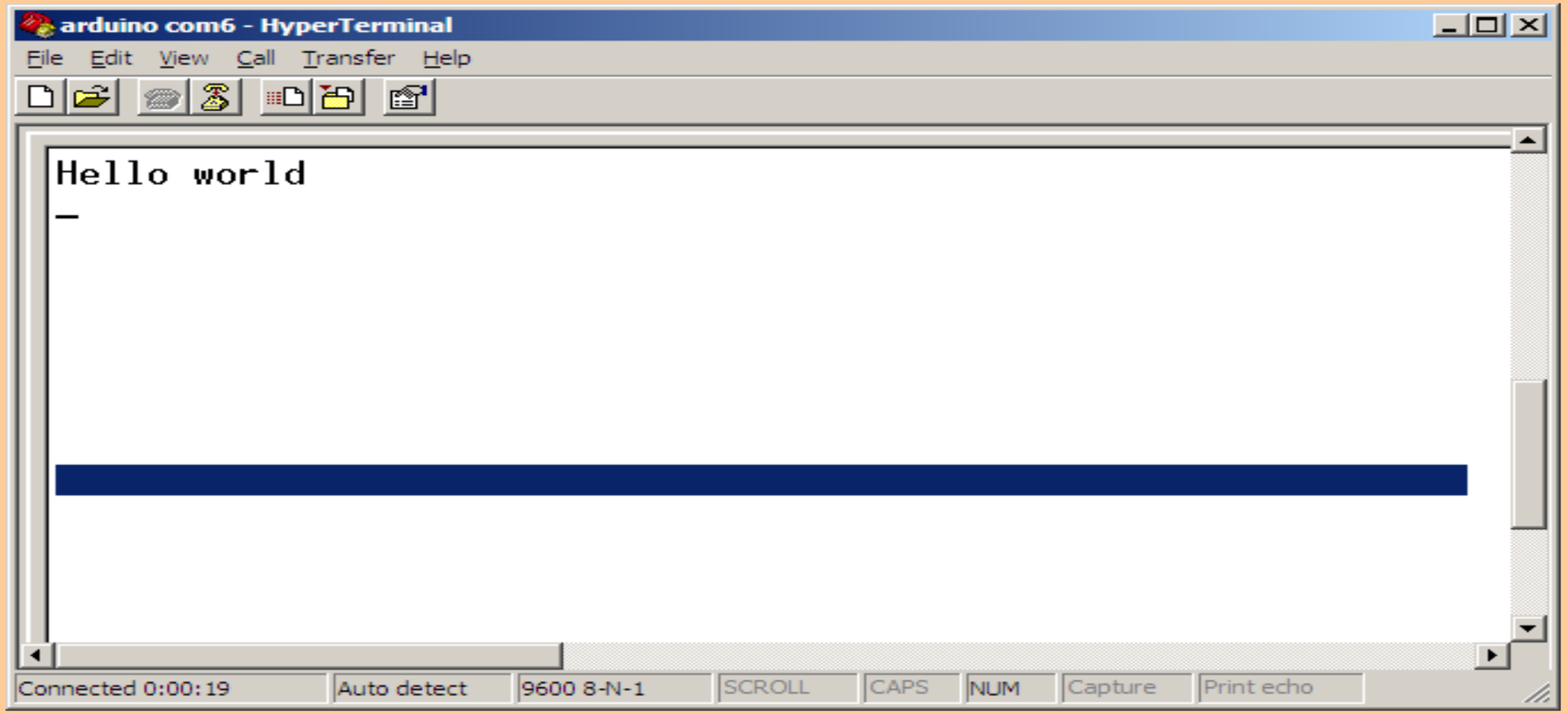
ตัวอย่างที่ 6.6

```
unsigned char TEXT[20] = {"Hello world"};
unsigned char i;


int main(void)
{
  UCSR0A = 0x00;
  UCSR0B = 0x08;     //enable transmitter but disable receiver
  UCSR0C = 0x06;     //select 1 stop bit and no parity asynchronous mode
  UBRR0H = 0;
  UBRR0L = 103;

  TEXT[11]= 13;  //carriage return
  TEXT[12]= 10;  //line feed

  for (i=0;i<13;i++)
  {
    USART_Transmit(TEXT[i]);
  }

  while(1)
  {
    do_nothing();
  }
}
```

```
//---------------------------------
void do_nothing(void)
{
}
```

```
#include <avr/io.h>
#include <avr/interrupt.h>


//---------------------------------------
void USART_Transmit( unsigned char data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSR0A & (1<<UDRE0)))
  ;
  UDR0 = data;
}
```

ตัวอย่างที่ 6.6


arduino com6 - HyperTerminal
File Edit View Call Transfer Help
Hello world
Connected 0:00:19
Auto detect
9600 8-N-1
SCROLL
CAPS
NUM
Capture
Print echo

ตัวอย่างที่ 6.7

- จงเขียนโปรแกรม AVR เพื่อทำการติดต่อกับเครื่องPC ซึ่งรันโปรแกรม Hyperterminal โดยให้ AVR นำข้อมูลที่รับได้จากพอร์ตอนุกรมมาทำการเพิ่มค่าขึ้น 2 ค่าแล้วส่งกลับไปยัง PC ดังตัวอย่าง

ข้อมูลที่ PC ส่งมา	ข้อมูลที่ AVR ส่งออกไป
‘A’	‘C’
‘1’	‘3’
‘d’	‘f’

- กำหนดค่า baudrate = 9600 bps, 1 stop bit, no parity

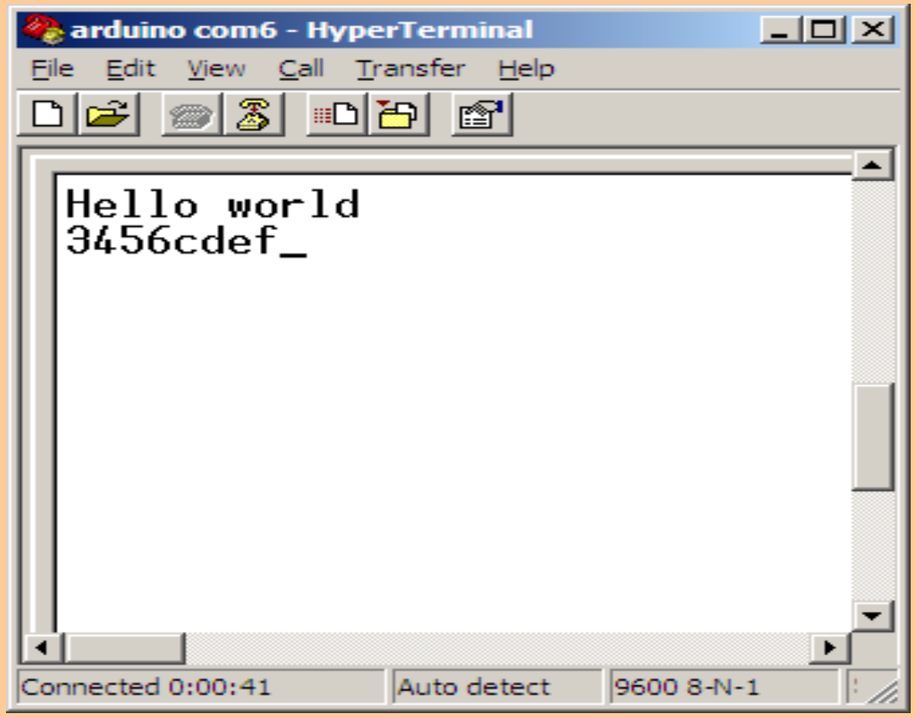
ตัวอย่างที่ 6.7

```
unsigned char TEXT[20] = {"Hello world"};
unsigned char i;


int main(void)
{
  UCSR0A = 0x00;
  UCSR0B = 0x18;     //enable transmitter and receiver
  UCSR0C = 0x06;     //select 1 stop bit and no parity asynchronous mode
  UBRR0H = 0;
  UBRR0L = 103;

  TEXT[11]= 13;  //carriage return
  TEXT[12]= 10;  //line feed

  for (i=0;i<13;i++)
  {
    USART_Transmit(TEXT[i]);
  }

  while(1)
  {
    i= USART_Receive();
    i+= 2;
    USART_Transmit(i);
  }
}
```

```
#include <avr/io.h>
#include <avr/interrupt.h>


//----------------------------------------
void USART_Transmit( unsigned char data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSR0A & (1<<UDRE0)))
  ;
  UDR0 = data;
}
```

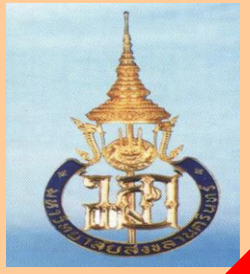
```
//----------------------------------------
unsigned char USART_Receive( void )
{
  /* Wait for data to be received */
  while ( !(UCSR0A & (1<<RXC0)) )
  ;
  /* Get and return received data from buffer */
  return UDR0;
}
```

ตัวอย่างที่ 6.7


arduino com6 - HyperTerminal
File Edit View Call Transfer Help
Hello world
3456cdef_
Connected 0:00:41
Auto detect
9600 8-N-1

เรจิสเตอร์ควบคุมใน ATmega32 และ ATmega328P

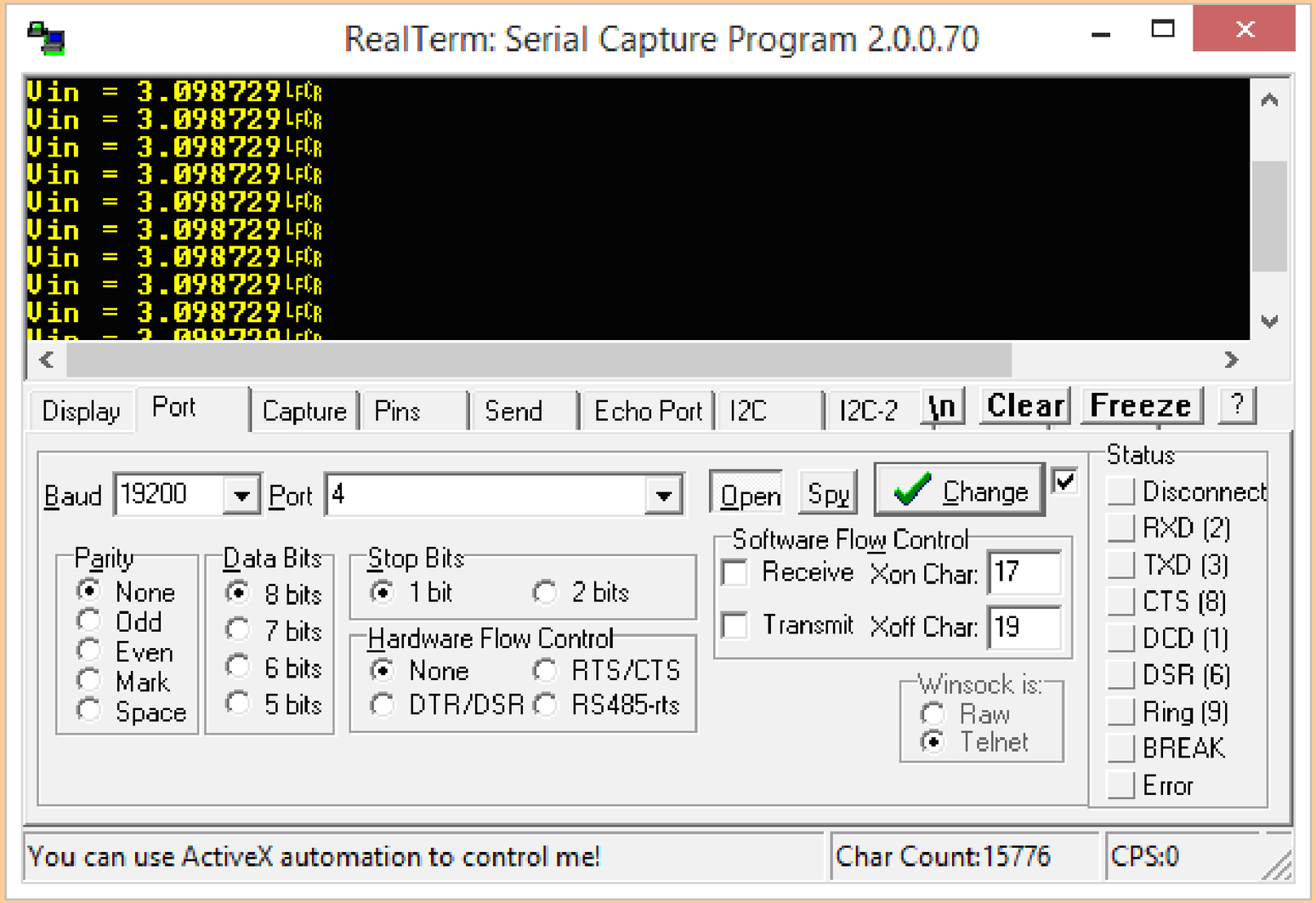
หน้าที่ใช้งานของเรจิสเตอร์	ชื่อเรจิสเตอร์ในตัวประมวลผล		หมายเหตุ
	ATmega328P	ATmega32	
สำหรับรับข้อมูลในภาครับ	UDR0	UDR	ทั้งภาครับและภาคส่งใช้ชื่อเดียวกันของ
สำหรับส่งข้อมูลในภาคส่ง	UDR0	UDR	
สำหรับเก็บบิตควบคุมและบิตสถานะตัวที่หนึ่ง	UCSR0A	UCSRA	-
สำหรับเก็บบิตควบคุมและบิตสถานะตัวที่สอง	UCSR0B	UCSRB	-
สำหรับเก็บบิตควบคุมและบิตสถานะตัวที่สาม	UCSR0C	UCSRC	-
สำหรับเก็บค่าควบคุมอัตราความเร็วในการรับส่งข้อมูล	UBRR0H	UBRRH	UBRR ไบต์สูง
	UBRR0L	UBRRL	UBRR ไบต์ต่ำ



การเชื่อมต่อเอวีอาร์กับพีซีผ่านระบบบัสยูเอสบี

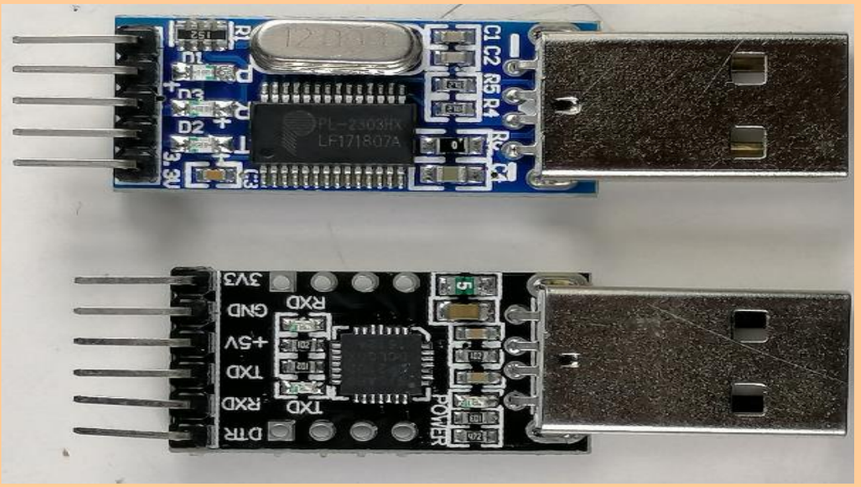
- ทำได้โดยใช้ไอซีแปลงยูเอสบีเป็น RS-232 ได้แก่
  - ไอซี FTD232R
  - ไอซี PL-2303HX
  - ไอซี CH340
- บนเครื่องพีซีจะมีการสร้างพอร์ตอนุกรมเสมือน (VCP: Virtual COM port)
- ผู้ใช้สามารถใช้โปรแกรมเลียนแบบเครื่องปลายทาง (Terminal Emulator) เพื่อติดต่อกับบอร์ดเอวีอาร์ได้
  - โปรแกรม HyperTerminal
  - โปรแกรม RealTerm เป็นต้น

โปรแกรมเลียนแบบเครื่องปลายทาง

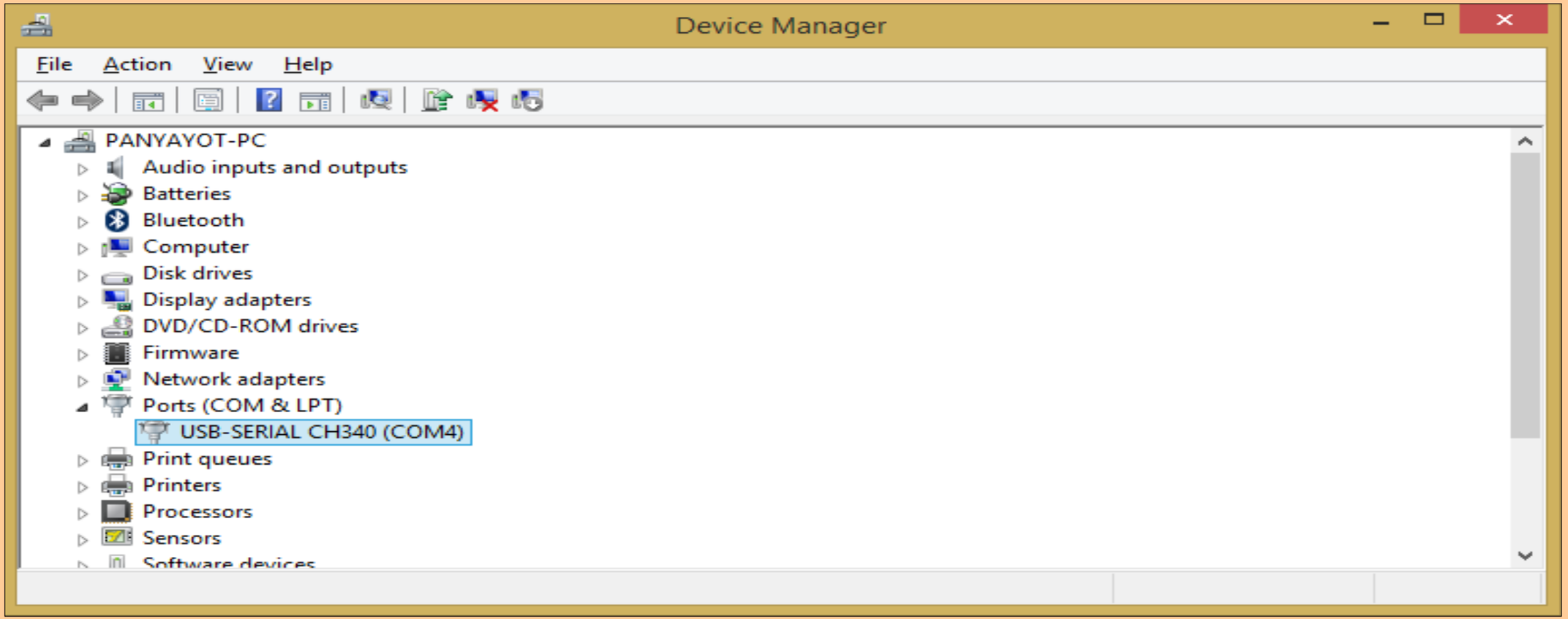


ตัวแปลงยูเอสบีเป็น RS-232

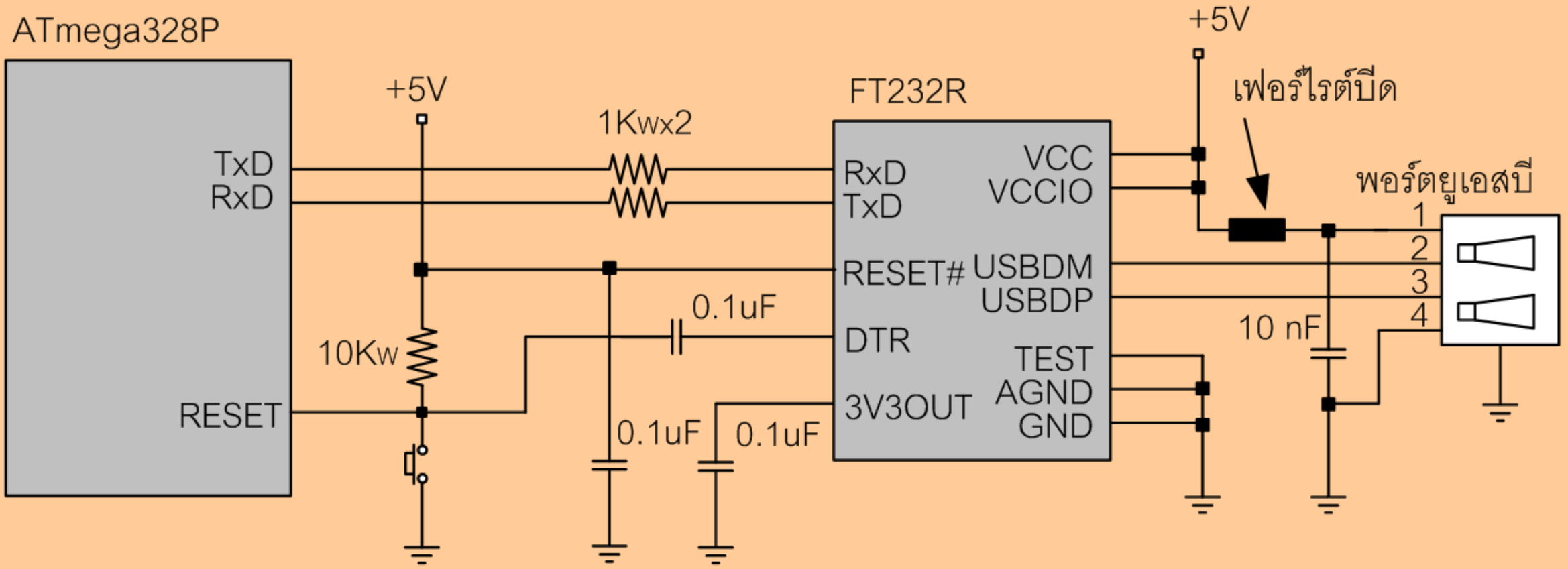
หมายเหตุ ถ่ายภาพโดยผู้จัดทำ



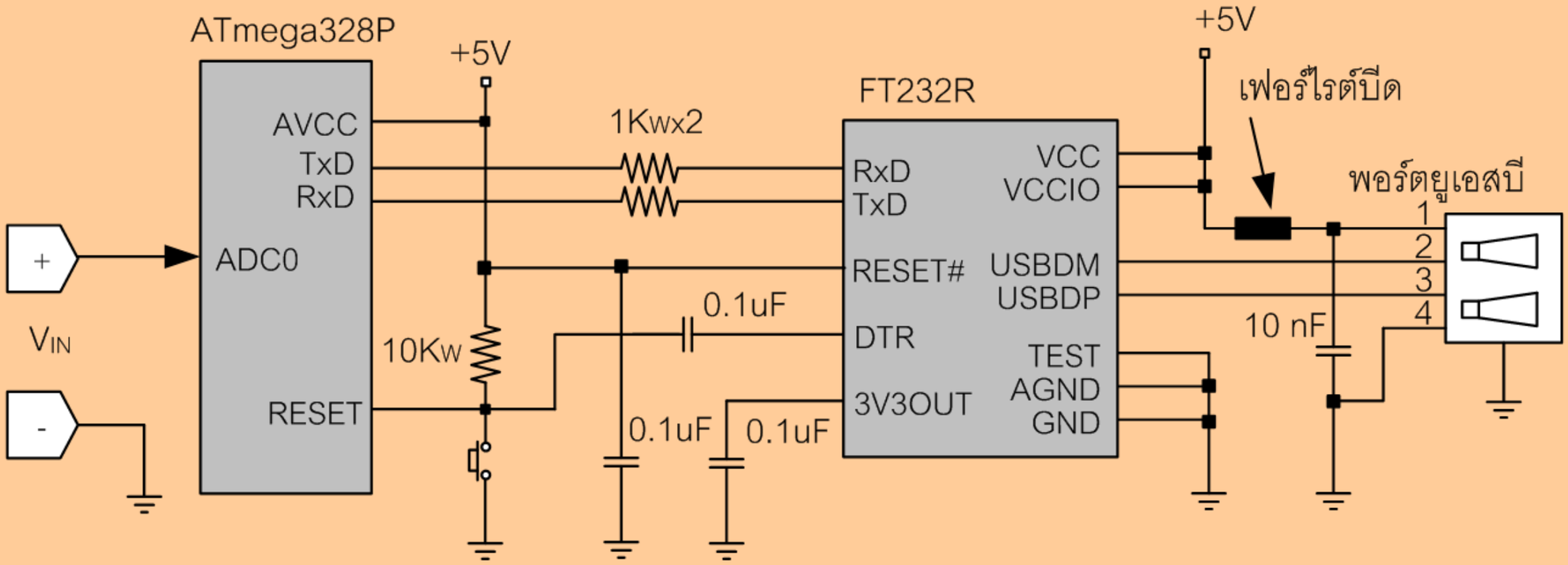
Virtual COM port



ตัวอย่างการใช้งานไอซี FT232R

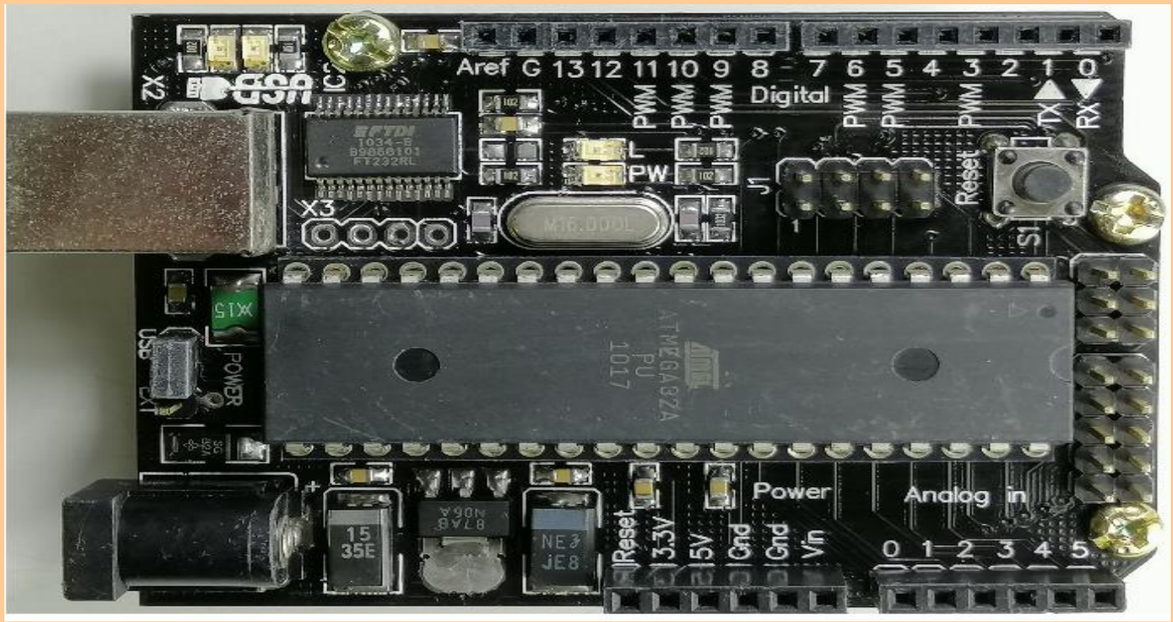


ตัวอย่างที่ 6.8



- อ่านค่าแรงดันแอนะล็อกจากขา ADC0 ไปแสดงผลบนเครื่องพีซีซึ่งติดตั้งโปรแกรมเลียนแบบเครื่องปลายทาง RealTerm ผ่านพอร์ตอนุกรม
- กำหนดให้ทำการรับส่งข้อมูลขนาด 8 บิต ที่ความเร็ว 19200 บิตต่อวินาทีโดยใช้บิตหยุดขนาด 1 บิต แต่ไม่มีการใช้บิตภาวะคู่หรือคี่

Arduino Boards



ตัวอย่างที่ 6.8

```
1  #include <avr/io.h>                    //เรียกใช้คลังโปรแกรม io.h
2  #define F_CPU 16000000UL               //กำหนดค่าความถี่ของตัวประมวลผลเท่ากับ 16 เมกะเฮิรทซ์
3  #include <avr/delay.h>                 //เรียกใช้คลังโปรแกรม delay.h
4  #define USART0_BAUDRATE 19200//ตั้งค่าอัตราบอดที่ 19200 บิตต่อวินาที
5  #define UBRR0_VALUE (((F_CPU/(USART0_BAUDRATE * 16UL))) - 1) //คำนวณค่าสำหรับตั้งใน UBRR0
6  #define ADC0 0                         //รับสัญญาณแอนะล็อกจากขา ADC0
7
8  unsigned int ADC_read(unsigned char a) //ฟังก์ชันสำหรับอ่านค่าแรงดันจากตัวแปลงแอนะล็อกเป็นดิจิทัล
9  {                                      //โดยรับค่าพารามิเตอร์ a ซึ่งระบุว่าจะอ่านจากช่องสัญญาณใด
10       ADMUX&=   0xF0;                  //ลบล้างค่าในสี่บิตล่างของ ADMUX ให้กลายเป็นตรรกะต่ำ
11       ADMUX|=    a;                    //ตั้งให้ ADMUX บิตสี่บิตล่างให้เท่ากับ a
12       ADCSRA |= (1<<ADSC);             //สั่งให้วงจรแปลงเริ่มทำการแปลงสัญญาณ
13       while ( !(ADCSRA & (1<<ADIF)) );  //วนซ้ำจนกว่าตัวบ่งชี้ ADIF มีค่าเป็นตรรกะสูง
14       ADCSRA |= 1<<ADIF;               //ลบล้างค่าในตัวบ่งชี้ ADIF ให้กลับมาเป็นตรรกะต่ำ
15       return ADC;                      //คืนค่าที่ได้จากการแปลงสู่โปรแกรมผู้เรียก
16 }
17
```

ตัวอย่างที่ 6.8

```
18  void init_USART0_module(void)   //ฟังก์ชันสำหรับตั้งรูปแบบการทำงานของมอดูลยูสาร์ท
19  {                                                   //เรจิสเตอร์ UBRR0 ใช้ตั้งความเร็วในการรับส่ง
20      UBRR0H = (uint8_t)(UBRR0_VALUE>>8);             //ตั้งค่าในเรจิสเตอร์ UBRR0H
21      UBRR0L = (uint8_t) UBRR0_VALUE;                 //ตั้งค่าในเรจิสเตอร์ UBRR0L
22      UCSR0C |= (1<<UCSZ01)|(1<<UCSZ00);              //ตั้งค่ารูปแบบข้อมูล 8 บิต, 1 stop bit, no parity
23      UCSR0B |= (1<<TXEN0);                           //เปิดทางภาคส่งของมอดูลยูสาร์ท
24  }
25
26  void USART0_Send_1_Byte(uint8_t data)//ฟังก์ชันสำหรับใช้ส่งตัวอักษร 1 ออกทางมอดูลยูสาร์ท
27  {
28      while(!(UCSR0A&(1<<UDRE0)))  ;          //วนซ้ำจนกว่าตัวบ่งชี้ UDRE0 จะเป็นตรรกะสูง
29      UDR0 = data;                            //ส่งค่าพารามิเตอร์ที่รับเข้าไปยังเรจิสเตอร์ภาคส่ง
30  }
31  void USART0_print_newline()     //สั่งให้ขึ้นบรรทัดใหม่สำหรับติดต่อกับโปรแกรมในไมโครซอฟต์วินโดวส์
32  {
33      USART0_Send_1_Byte('\n');   //ส่งตัวอักษรค่า line feed (ขึ้นบรรทัดใหม่) ออกทางพอร์ตอนุกรม
34      USART0_Send_1_Byte('\r');   //ส่งตัวอักษรค่า carriage return (ปัดแคร่) ออกทางพอร์ตอนุกรม
35  }
```

ตัวอย่างที่ 6.8

```
36  int main(void)
37  {
38      float volt;                     //ตัวแปร volt สำหรับเก็บค่าแรงดัน
39      uint16_t    Vadc0, i;           //ตัวแปร i สำหรับเป็นตัวนับการวนซ้ำ ตัวแปร Vadc0 ใช้รับค่าจากตัว ADC
40      char msg[32];                   //ตัวแปรสำหรับเก็บข้อความที่จะส่งออกไปแสดงผลยังเครื่อง PC
41      DDRC = 0x00;                    //สั่งให้พอร์ต C ทำหน้าที่รับเข้า
42      ADMUX = 0b01000101;             //เริ่มต้นให้เลือกค่าแรงดันอ้างอิงจากขา AVCC
43      ADCSRA = 0x87;                  //สั่งให้วงจรแปลงทำงานในแบบวิธีแปลงครั้งเดียว
44      init_USART0_module();           //สั่งให้มอดูลยูสาร์ทเริ่มทำงาน
45      while(1)                        //วนซ้ำการทำงานไม่รู้จบ
46      {
47          Vadc0 = ADC_read(ADC0);             //อ่านค่าสัญญาณแอนะล็อกใส่ตัวแปร Vadc0
48          volt = Vadc0/1023.00*5.00;          //คำนวณค่าแรงดันออกมาใส่ในตัวแปร volt
49          sprintf(msg,"Vin = %f", volt);      //เตรียมข้อความที่จะแสดงผลออกที่เครื่องพีซีใส่ใน msg
50          for (i=0;i<strlen(msg);i++)         //วนซ้ำส่งข้อมูลใน msg โดยตรวจสอบความยาวข้อมูลที่จะส่ง
51              USART0_Send_1_Byte(msg[i]);//ส่งข้อมูลใน msg ครั้งละ 1 ไบต์ออกสู่พีซีผ่านมอดูลยูสาร์ท
52          USART0_print_newline();             //สั่งให้โปรแกรมเลียนแบบเครื่องปลายทางขึ้นบรรทัดใหม่
53          _delay_ms(1000);                    //หน่วงเวลา 1 วินาที
54      }
55  }
```

ตัวอย่างหน้าจอโปรแกรมเลียนแบบเครื่องปลายทาง

RealTerm: Serial Capture Program 2.0.0.70

```
Vin  = 3.098729LFCR
Vin  = 3.098729LFCR
Vin  = 3.098729LFCR
Vin  = 3.098729LFCR
Vin  = 3.098729LFCR
Vin  = 3.098729LFCR
Vin  = 3.098729LFCR
Vin  = 3.098729LFCR
Vin  = 3.098729LFCR
```

Display | Port | Capture | Pins | Send | Echo Port | I2C | I2C-2 | \n | Clear | Freeze | ?

Baud 19200 Port 4 Open Spy Change

Parity: None, Odd, Even, Mark, Space

Data Bits: 8 bits, 7 bits, 6 bits, 5 bits

Stop Bits: 1 bit, 2 bits

Hardware Flow Control: None, RTS/CTS, DTR/DSR, RS485-rts

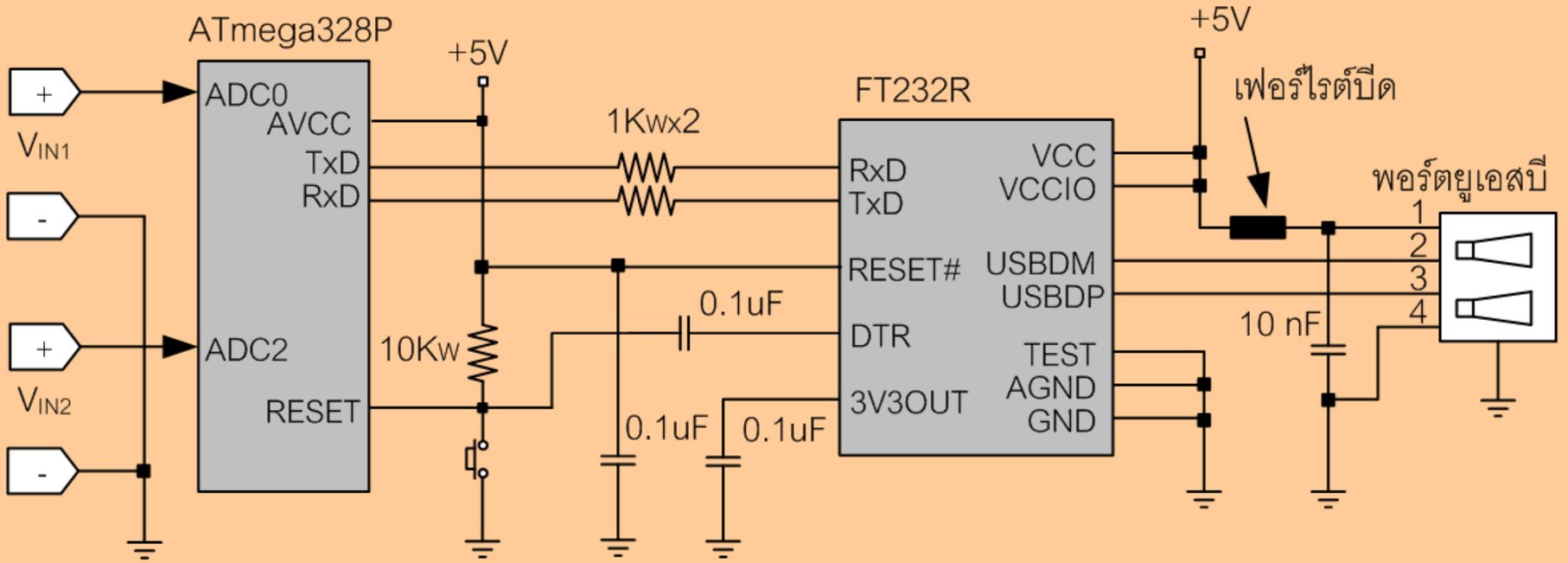
Software Flow Control: Receive, Transmit, Xon Char: 17, Xoff Char: 19

Winsock is: Raw, Telnet

Status: Disconnect, RXD (2), TXD (3), CTS (8), DCD (1), DSR (6), Ring (9), BREAK, Error

You can use ActiveX automation to control me! | Char Count:15776 | CPS:0

ตัวอย่างที่ 6.9 การอินเตอร์รัพต์กับ USART



- การรับส่งข้อมูลขนาด 8 บิต ที่ความเร็ว 19200 บิตต่อวินาทีโดยใช้บิตภาวะคี่ และบิตหยุดขนาด 2 บิต
- หากมีการกดปุ่ม “1” และปุ่ม “2” บนแป้นพิมพ์ของเครื่องพีซีจะมีการอ่านค่าแรงดัน VIN1 และ VIN2 ส่งไปแสดงผลตามลำดับ
- หากมีการกดปุ่ม “T” หรือ “t” บนแป้นพิมพ์ของเครื่องพีซีจะมีการสลับการอ่านจากช่องทางรับเข้าปัจจุบันไปอ่านค่าแรงดันแอนะล็อกอีกช่องทางหนึ่ง

ตัวอย่างที่ 6.9 การอินเตอร์รัพต์กับ USART

```
1  #include <avr/io.h>                    //เรียกใช้คลังโปรแกรม io.h
2  #define F_CPU 16000000UL               //กำหนดค่าความถี่ของตัวประมวลผลเท่ากับ 16 เมกะเฮิรทซ์
3  #include <avr/delay.h>                 //เรียกใช้คลังโปรแกรม delay.h
4  #include <avr/interrupt.h>             //เรียกใช้คลังโปรแกรม interrupt.h
5  #define USART0_BAUDRATE 19200//ตั้งค่าอัตราบอดที่ 19200 บิตต่อวินาที
6  #define UBRR0_VALUE (((F_CPU/(USART0_BAUDRATE * 16UL))) - 1) //คำนวณค่าสำหรับตั้งใน UBRR0
7  #define ADC0 0                         //สำหรับรับสัญญาณแอนะล็อกจากขา ADC0
8  #define ADC2 2                         //สำหรับรับสัญญาณแอนะล็อกจากขา ADC2
9  uint8_t adc_ip;                        //ตัวแปรส่วนกลางสำหรับเก็บค่าช่องทางแอนะล็อกที่ต้องรับเข้า
10 void init_USART0_module(void) //ฟังก์ชันสำหรับตั้งรูปแบบการทำงานของมอดูลยูสาร์ท
11 {                                      //เรจิสเตอร์ UBRR0 ใช้ตั้งความเร็วในการรับส่ง
12       UBRR0H = (uint8_t)(UBRR0_VALUE>>8);   //ตั้งค่าในเรจิสเตอร์ UBRR0H
13       UBRR0L = (uint8_t) UBRR0_VALUE;       //ตั้งค่าในเรจิสเตอร์ UBRR0L
14       UCSR0C |= (1<<UCSZ01)|(1<<UCSZ00);    //ตั้งค่ารูปแบบข้อมูล 8 บิต, 1 stop bit, no parity
15       UCSR0B |= (1<<TXEN0)|(1<<RXEN0);      //เปิดทางภาครับ/ส่งของมอดูลยูสาร์ท
16       UCSR0B |= (1<<RXCIE0);                //เปิดทางการขัดจังหวะจากภาครับของมอดูลยูสาร์ท
17 }
```

ตัวอย่างที่ 6.9 การอินเตอร์รัพต์กับ USART

```
18  ISR(USART_RX_vect)                  //ฟังก์ชันบริการการขัดจังหวะจากภาครับของมอดูลยูสาร์ท
19  {
20      switch (UDR0)                   //อ่านค่าจากเรจิสเตอร์ UDR0 และนำค่าที่ได้มาใช้ในการตัดสินใจ
21      {
22          case '1':   adc_ip = 1;   break;   //หากรับได้ตัวอักษร '1' ให้ตั้งตัวแปร adc_ip =1
23          case '2':   adc_ip = 2;   break;   //หากรับได้ตัวอักษร '2' ให้ตั้งตัวแปร adc_ip =2
24          case 't':                          //หากรับได้ตัวอักษร 't' หรือ
25          case 'T':   if (adc_ip==1)         //หากรับได้ตัวอักษร 'T' ให้ดูว่าค่าเก่าของ adc_ip เป็นอะไร
26                          adc_ip = 2;        //หากค่าเก่าของ adc_ip เป็น 1 ให้กลับค่าเป็น 2
27                      else
28                          adc_ip = 1;        //หากค่าเก่าของ adc_ip เป็น 2 ให้กลับค่าเป็น 1
29      }
30  }
31  int main(void)
32  {
33      adc_ip = 1;
34      float volt;                 //ตัวแปร volt สำหรับเก็บค่าแรงดัน
35      uint16_t    Vadc, i;        //ตัวแปร i สำหรับเป็นตัวนับการวนซ้ำ ตัวแปร Vadc ใช้รับค่าจากตัว ADC
36      char msg[32];               //ตัวแปรสำหรับเก็บข้อความที่จะส่งออกไปแสดงผลยังเครื่อง PC
```

ตัวอย่างที่ 6.9

```
37      DDRC = 0x00;              //สั่งให้พอร์ต C ทำหน้าที่รับเข้า
38      ADMUX = 0b01000101; //เริ่มต้นให้เลือกค่าแรงดันอ้างอิงจากขา AVCC
39      ADCSRA = 0x87;            //สั่งให้วงจรแปลงทำงานในแบบวิธีแปลงครั้งเดียว
40      init_USART0_module(); //สั่งให้มอดูลยูสาร์ทเริ่มทำงาน
41      sei();                    //เปิดทางการขัดจังหวะส่วนกลาง
42      while(1)                  //วนซ้ำการทำงานไม่รู้จบ
43      {
44          if (adc_ip==2)              //หากตัวแปรระบุช่องทางรับเข้าเท่ากับ 2 ให้อ่านจาก ADC2
45          {
46              Vadc = ADC_read(ADC2);      //อ่านค่าสัญญาณจากขา ADC2 ใส่ตัวแปร Vadc
47              volt = Vadc/1023.00*5.00;   //คำนวณค่าแรงดันออกมาใส่ในตัวแปร volt
48              sprintf(msg,"Vin2 = %f", volt); //เตรียมข้อความที่จะแสดงผลออกที่เครื่องพีซีใส่ใน msg
49          }
50          else                        //แต่หากตัวแปรระบุช่องทางรับเข้าไม่เท่ากับ 2 ให้อ่านจาก ADC0
51          {
52              Vadc = ADC_read(ADC0);      //อ่านค่าสัญญาณจากขา ADC0 ใส่ตัวแปร Vadc
53              volt = Vadc/1023.00*5.00;   //คำนวณค่าแรงดันออกมาใส่ในตัวแปร volt
54              sprintf(msg,"Vin1 = %f", volt); //เตรียมข้อความที่จะแสดงผลออกที่เครื่องพีซีใส่ใน msg
55          }
56          for (i=0;i<strlen(msg);i++)         //วนซ้ำส่งข้อมูลใน msg โดยตรวจสอบความยาวข้อมูลที่จะส่ง
57              USART0_Send_1_Byte(msg[i]);     //ส่งข้อมูลใน msg ครั้งละ 1 ไบต์ออกสู่พีซีผ่านมอดูลยูสาร์ท
58          USART0_print_newline();         //สั่งให้โปรแกรมเลียนแบบเครื่องปลายทางขึ้นบรรทัดใหม่
59          _delay_ms(1000);                //หน่วงเวลา 1 วินาที
60      }
```

ตัวอย่างที่ 6.9 การอินเตอร์รัพต์กับ USART

RealTerm: Serial Capture Program 2.0.0.70

```
Vin1  = 3.079179 LFCR
Vin1  = 3.079179 LFCR
Vin2  = 1.285435 LFCR
Vin2  = 1.285435 LFCR
Vin1  = 3.079179 LFCR
Vin1  = 3.084066 LFCR
Vin1  = 3.079179 LFCR
Vin1  = 3.084066 LFCR
Vin2  = 1.285435 LFCR
Vin2  = 1.285435 LFCR
                 CR
```

Display | Port | Capture | Pins | Send | Echo Port | I2C | I2C-2 | \n | Clear | Freeze | ?

Baud 19200 Port 4 Open Spy Change

Parity: None, Odd, Even, Mark, Space

Data Bits: 8 bits, 7 bits, 6 bits, 5 bits

Stop Bits: 1 bit, 2 bits

Hardware Flow Control: None, RTS/CTS, DTR/DSR, RS485-rts

Software Flow Control: Receive, Transmit, Xon Char: 17, Xoff Char: 19

Winsock is: Raw, Telnet

Status: Disconnect, RXD (2), TXD (3), CTS (8), DCD (1), DSR (6), Ring (9), BREAK, Error

You can use ActiveX automation to control me! | Char Count:5814 | CPS:0

ฟังก์ชันสำเร็จรูปใน Arduino

- Serial.read
- Serial.print
- Serial.println
- Serial.begin
- ใน Arduino Platform สามารถจัดการ Serial Communcation แบบ Interrupt ได้หรือไม่ ?

จบบทที่ 6