การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

รหัสวิชา 30127-2004 (2-3-3)
ดิจิทัลและไมโครคอนโทรลเลอร์

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

1. รูปแบบของการสื่อสารข้อมูล
2. รูปแบบข้อมูลของการสื่อสารแบบ UART
3. รูปแบบข้อมูลของการสื่อสารแบบ One-Wire
4. รีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ต UART ของไมโครคอนโทรลเลอร์
   - 4.1 รีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ต UART ของ AT89C51ED2
   - 4.2 รีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ต UART ของ PIC16F887
   - 4.3 รีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ต UART ของ ATMEGA32

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

5. การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ UARTของไมโครคอนโทรลเลอร์
   - 5.1 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ UART ของ AT89C51ED2
   - 5.2 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ UART ของ PIC16F887
   - 5.3 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ UART ของ ATMEGA32
6. การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ One-Wire ของไมโครคอนโทรลเลอร์
   - 6.1 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ One-Wire ของ AT89C51ED2
   - 6.2 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ One-Wire ของ PIC16F887
   - 6.3 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ One-Wire ของ ATMEGA32

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

1. รูปแบบของการสื่อสารข้อมูล

การสื่อสารข้อมูลสามารถแบ่งออกได้ 3 ลักษณะคือ

1.1 รูปแบบการสื่อสารข้อมูลที่แบ่งตามคุณสมบัติของสัญญาณ ซึ่งสามารถแบ่งออกได้เป็น 2 ประเภทคือ

1.1.1 การสื่อสารข้อมูลแบบอนาล็อก (Analog Signal)

1.1.2 การสื่อสารข้อมูลแบบดิจิตอล (Digital Signal)

1.2 รูปแบบการสื่อสารข้อมูลที่แบ่งตามทิศทางของการสื่อสารข้อมูล ซึ่งสามารถแบ่งออกได้เป็น 3 ประเภทคือ

1.2.1 การสื่อสารข้อมูลแบบซิมเพล็กซ์ (Simplex)

1.2.2 การสื่อสารข้อมูลแบบฮาล์ฟดูเพล็กซ์ (Half Duplex)

1.2.3 การสื่อสารข้อมูลแบบฟูลดูเพล็กซ์ (Full Duplex)

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

1.3 แบ่งตามลักษณะการสื่อสารข้อมูล ซึ่งสามารถแบ่งออกได้เป็น 2 ประเภทคือ

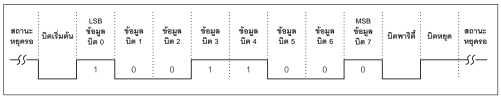
1.3.1 การสื่อสารข้อมูลแบบอนุกรม (Serial Data Transmission)

- การสื่อสารข้อมูลแบบอะซิงโครนัส (Asynchronous Data Transmission) ได้แก่ UART และ One-Wire เป็นต้น
- การสื่อสารข้อมูลแบบซิงโครนัส (Synchronous Data Transmission) ได้แก่ I2C และ SPI เป็นต้น

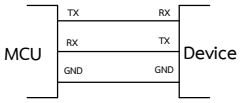
1.3.2 การสื่อสารข้อมูลแบบขนาน (Parallel Data Transmission)

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

2. รูปแบบข้อมูลของการสื่อสารแบบ UART

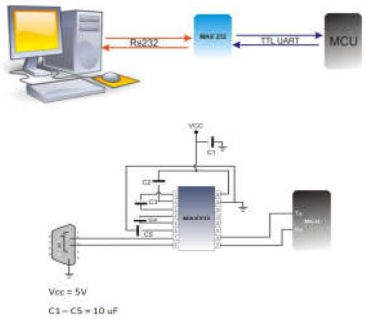


รูปแสดงรูปแบบของสัญญาณการสื่อสารข้อมูลในมาตรฐาน UART



รูปแสดงการเชื่อมต่อไมโครคอนโทรลเลอร์กับอุปกรณ์ภายนอกด้วยพอร์ต UART TTL

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire



รูปแสดงการเชื่อมต่อไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ด้วยพอร์ต UART TTL to RS232
(ที่มา : https://www.thaieasyelec.com)

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

3. รูปแบบข้อมูลของการสื่อสารแบบ One-Wire

มาตรฐานการสื่อสารข้อมูลแบบ One-Wire เป็นรูปแบบการสื่อสารข้อมูลที่คิดค้นโดยบริษัท ดัลลัสเซมิคอนดักเตอร์ บางครั้งจึงเรียกระบบสื่อสารข้อมูลแบบนี้ว่า ระบบสื่อสารข้อมูลดัลลัสหนึ่งสาย (The Dallas 1-Wire Bus)

คุณสมบัติของไทม์สล็อต

อุปกรณ์มาสเตอร์เป็นอุปกรณ์เพียงตัวเดียวบนระบบบัสที่สามารถอินิเชียลสายสัญญาณได้ โดยอุปกรณ์มาสเตอร์จะเริ่มต้นไทม์สล็อตด้วยการทำให้สายสัญญาณเป็นลอจิกต่ำในช่วงเวลาหนึ่ง จากนั้นทำให้กลับมาเป็นลอจิกสูง ถ้าหากอุปกรณ์สเลฟต้องการส่งข้อมูลมายังอุปกรณ์มาสเตอร์ อุปกรณ์สเลฟจะเป็นตัวควบคุมสภาวะของสายสัญญาณต่อไป จนเสร็จสิ้นกระบวนการ แต่ถ้าอุปกรณ์มาสเตอร์ต้องการส่งข้อมูลก็จะสามารถดำเนินการต่อไปได้เลย

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

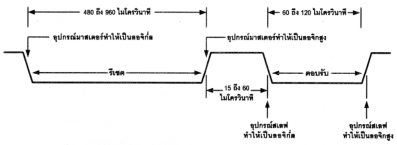
ฟังก์ชันของไทม์สล็อตที่กำหนดโดยอุปกรณ์มาสเตอร์ มีด้วยกัน 4 ฟังก์ชัน คือ

1. รีเซต (RESET) ใช้ในการเริ่มต้นติดต่อกับอุปกรณ์สเลฟ
2. อ่านข้อมูล (READ DATA) ใช้อ่านข้อมูลที่ส่งมาจากอุปกรณ์สเลฟ
3. เขียนข้อมูล ‘1’ (WRITE ONE) ใช้เขียนข้อมูล ‘1’ ไปยังอุปกรณ์สเลฟผ่านสายสัญญาณของระบบ
4. เขียนข้อมูล ‘0’ (WRITE ZERO) ใช้เขียนข้อมูล ‘0’ ไปยังอุปกรณ์สเลฟผ่านสายสัญญาณของระบบ

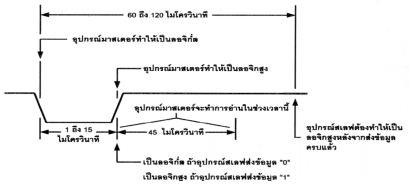
ฟังก์ชันของไทม์สล็อตในอุปกรณ์สเลฟ มีด้วยกัน 3 ฟังก์ชัน คือ

1. ตอบสนอง (PRESENCE) ใช้สำหรับตอบสนองการติดต่อจากอุปกรณ์มาสเตอร์ โดยอุปกรณ์สเลฟตัวที่ถูกเลือกจากอุปกรณ์มาสเตอร์ จะต้องส่งสัญญาณตอบสนองลงบนสายสัญญาณ เพื่อแจ้งให้อุปกรณ์มาสเตอร์ทราบว่า ขณะนี้สามารถติดต่อกันได้แล้ว
2. เขียนข้อมูล ‘1’ (WRITE ONE) ใช้สำหรับส่งข้อมูล ‘1’ ไปยังอุปกรณ์มาสเตอร์ผ่านสายสัญญาณของระบบ ซึ่งจะสัมพันธ์กับไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์
3. เขียนข้อมูล ‘0’ (WRITE ZERO) ใช้สำหรับส่งข้อมูล ‘0’ ไปยังอุปกรณ์มาสเตอร์ผ่านสายสัญญาณของระบบ ซึ่งจะสัมพันธ์กับไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

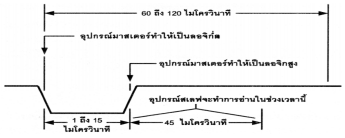


รูปแสดงคาบเวลาของไทม์สล็อตการรีเซตและตอบสนอง

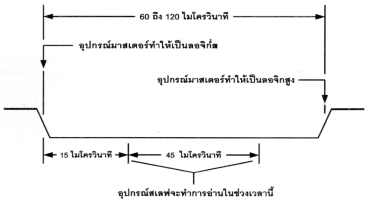


รูปแสดงไทม์สล็อตการอ่านข้อมูลของไมโครคอนโทรลเลอร์และการเขียนข้อมูลของอุปกรณ์สเลฟ

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire



รูปแสดงไทม์สล็อตการเขียนข้อมูลลอจิก ‘1’ ของไมโครคอนโทรลเลอร์



รูปแสดงไทม์สล็อตการเขียนข้อมูลลอจิก ‘0’ ของไมโครคอนโทรลเลอร์

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

4. รีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ต UART ของไมโครคอนโทรลเลอร์

4.1 รีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ต UART ของ AT89C51ED2 (P3.0,P3.1 และSBUF)

SCON: SERIAL PORT CONTROL REGISTER. BIT ADDRESSABLE.

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0 SCON. 7 Serial Port mode specifier. (NOTE 1).
SM1 SCON. 6 Serial Port mode specifier. (NOTE 1).
SM2 SCON. 5 Enables the multiprocessor communication feature in modes 2 & 3. In mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0. (See Table 9).
REN SCON. 4 Set/Cleared by software to Enable/Disable reception.
TB8 SCON. 3 The 9th bit that will be transmitted in modes 2 & 3. Set/Cleared by software.
RB8 SCON. 2 In modes 2 & 3, is the 9th data bit that was received. In mode 1, if SM2 = 0, RB8 is the stop bit that was received. In mode 0, RB8 is not used.
TI SCON. 1 Transmit interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or at the beginning of the stop bit in the other modes. Must be cleared by software.
RI SCON. 0 Receive interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or halfway through the stop bit time in the other modes (except see SM2). Must be cleared by software.

NOTE 1:

SM0	SM1	Mode	Description	Baud Rate
0	0	0	SHIFT REGISTER	Fosc./12
0	1	1	8-Bit UART	Variable
1	0	2	9-Bit UART	Fosc./64 OR Fosc./32
1	1	3	9-Bit UART	Variable

SERIAL PORT SET-UP:

Table 9

MODE	SCON	SM2 VARIATION
0 1 2 3	10H 50H 90H D0H	Single Processor Environment (SM2 = 0)
0 1 2 3	NA 70H B0H F0H	Multiprocessor Environment (SM2 = 1)

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

การกำหนดค่าให้ TH1 เพื่อกำเนิด Baud rate กรณีที่บิต PCON.7 เป็นลอจิก 0 คำนวณได้ดังนี้

TH1 = 256 - ((Crystal/384)/Baud)

ถ้า PCON.7 เป็นลอจิก 1 ค่า Baud rate จะเป็น 2 เท่า คำนวณได้ดังนี้

TH1 = 256 - ((Crystal/192)/Baud)

ตัวอย่าง สมมติว่า Crystal ความถี่ 11.0592 MHz ต้องการกำหนด Baud rate ที่ 9,600 bps จงคำนวณหาค่าที่จะโหลดลงใน TH1

วิธีคิด

TH1 = 256 - ((Crystal)/384)/Baud)

TH1 = 256 - ((11059200)/384)/9600)

TH1 = 256 - ((28800)/9600)

TH1 = 256 - 3 = 253

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

4.2 รีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ต UART ของ PIC16F887 (RC6 ,RC7 ,RCREG และTXREG)

REGISTER 12-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER(1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared x = Bit is unknown

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit
Asynchronous mode:
Don't care
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave
Don't care

bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enable interrupt and load the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 8-bit (RX9 = 0):
Don't care

bit 2 **FERR:** Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR:** Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** Ninth bit of Received Data
This can be address/data bit or a parity bit and must be calculated by user firmware.

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

REGISTER 12-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN(1)	SYNC	SENDB	BRGH	TRMT	TX9D
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared x = Bit is unknown

bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
Don't care
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9:** 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN:** Transmit Enable bit(1)
1 = Transmit enabled
0 = Transmit disabled

bit 4 **SYNC:** EUSART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **SENDB:** Send Break Character bit
Asynchronous mode:
1 = Send Sync Break on next transmission (cleared by hardware upon completion)
0 = Sync Break transmission completed
Synchronous mode:
Don't care

bit 2 **BRGH:** High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode

bit 1 **TRMT:** Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** Ninth bit of Transmit Data
Can be address/data bit or a parity bit.

Note 1: SREN/CREN overrides TXEN in Sync mode.

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

$$Desired\ Baud\ Rate = \frac{FOSC}{64([SPBRGH:SPBRG]+1)}$$

Solving for SPBRGH:SPBRG:

$$X = \frac{\frac{FOSC}{Desired\ Baud\ Rate}}{64} - 1$$

$$= \frac{\frac{16000000}{9600}}{64} - 1$$

$$= [25.042] = 25$$

$$Calculated\ Baud\ Rate = \frac{16000000}{64(25+1)}$$

$$= 9615$$

$$Error = \frac{Calc.\ Baud\ Rate - Desired\ Baud\ Rate}{Desired\ Baud\ Rate}$$

$$= \frac{(9615-9600)}{9600} = 0.16\%$$

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

REGISTER 12-3: BAUDCTL: BAUD RATE CONTROL REGISTER

R-0	R-1	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared x = Bit is unknown

bit 7 **ABDOVF:** Auto-Baud Detect Overflow bit
Asynchronous mode:
1 = Auto-baud timer overflowed
0 = Auto-baud timer did not overflow
Synchronous mode:
Don't care

bit 6 **RCIDL:** Receive Idle Flag bit
Asynchronous mode:
1 = Receiver is Idle
0 = Start bit has been received and the receiver is receiving
Synchronous mode:
Don't care

bit 5 **Unimplemented:** Read as '0'

bit 4 **SCKP:** Synchronous Clock Polarity Select bit
Asynchronous mode:
1 = Transmit inverted data to the RB7/TX/CK pin
0 = Transmit non-inverted data to the RB7/TX/CK pin
Synchronous mode:
1 = Data is clocked on rising edge of the clock
0 = Data is clocked on falling edge of the clock

bit 3 **BRG16:** 16-bit Baud Rate Generator bit
1 = 16-bit Baud Rate Generator is used
0 = 8-bit Baud Rate Generator is used

bit 2 **Unimplemented:** Read as '0'

bit 1 **WUE:** Wake-up Enable bit
Asynchronous mode:
1 = Receiver is waiting for a falling edge. No character will be received byte RCIF will be set. WUE will automatically clear after RCIF is set.
0 = Receiver is operating normally
Synchronous mode:
Don't care

bit 0 **ABDEN:** Auto-Baud Detect Enable bit
Asynchronous mode:
1 = Auto-Baud Detect mode is enabled (clears when auto-baud is complete)
0 = Auto-Baud Detect mode is disabled
Synchronous mode:
Don't care

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

4.3 รีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ต UART ของ ATMEGA32 (PD0,PD1 และUDR)

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXC: USART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the receiver is disabled, the receive buffer will be flushed and consequently the RXC bit will become zero. The RXC Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE bit).

- **Bit 6 – TXC: USART Transmit Complete**

This flag bit is set when the entire frame in the transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR). The TXC Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC Flag can generate a Transmit Complete interrupt (see description of the TXCIE bit).

- **Bit 5 – UDRE: USART Data Register Empty**

The UDRE Flag indicates if the transmit buffer (UDR) is ready to receive new data. If UDRE is one, the buffer is empty, and therefore ready to be written. The UDRE Flag can generate a Data Register empty Interrupt (see description of the UDRIE bit).

UDRE is set after a reset to indicate that the transmitter is ready.

- **Bit 4 – FE: Frame Error**

This bit is set if the next character in the receive buffer had a Frame Error when received. I.e., when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR) is read. The FE bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSRA.

- **Bit 3 – DOR: Data OverRun**

This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the receive Shift Register, and a new start bit is detected. This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

- **Bit 2 – PE: Parity Error**

This bit is set if the next character in the receive buffer had a Parity Error when received and the parity checking was enabled at that point (UPM1 = 1). This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

- **Bit 1 – U2X: Double the USART Transmission Speed**

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

- **Bit 0 – MPCM: Multi-processor Communication Mode**

This bit enables the Multi-processor Communication mode. When the MPCM bit is written to one, all the incoming frames received by the USART receiver that do not contain address information will be ignored. The transmitter is unaffected by the MPCM setting. For more detailed information see "Multi-processor Communication Mode" on page 157.

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – RXCIE: RX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the RXC Flag. A USART Receive Complete Interrupt will be generated only if the RXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC bit in UCSRA is set.

- **Bit 6 – TXCIE: TX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the TXC Flag. A USART Transmit Complete Interrupt will be generated only if the TXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC bit in UCSRA is set.

- **Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable**

Writing this bit to one enables interrupt on the UDRE Flag. A Data Register Empty Interrupt will be generated only if the UDRIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE bit in UCSRA is set.

- **Bit 4 – RXEN: Receiver Enable**

Writing this bit to one enables the USART Receiver. The Receiver will override normal port operation for the RxD pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE, DOR, and PE Flags.

- **Bit 3 – TXEN: Transmitter Enable**

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD pin when enabled. The disabling of the Transmitter (writing TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the transmit Shift Register and transmit Buffer Register do not contain data to be transmitted. When disabled, the transmitter will no longer override the TxD port.

- **Bit 2 – UCSZ2: Character Size**

The UCSZ2 bits combined with the UCSZ1:0 bit in UCSRC sets the number of data bits (Character Size) in a frame the receiver and transmitter use.

- **Bit 1 – RXB8: Receive Data Bit 8**

RXB8 is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDR.

- **Bit 0 – TXB8: Transmit Data Bit 8**

TXB8 is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. Must be written before writing the low bits to UDR.

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

Bit	7	6	5	4	3	2	1	0	
	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	0	0	0	0	1	1	0	

The UCSRC Register shares the same I/O location as the UBRRH Register. See the "Accessing UBRRH/ UCSRC Registers" on page 158 section which describes how to access this register.

- **Bit 7 – URSEL: Register Select**

This bit selects between accessing the UCSRC or the UBRRH Register. It is read as one when reading UCSRC. The URSEL must be one when writing the UCSRC.

- **Bit 6 – UMSEL: USART Mode Select**

This bit selects between Asynchronous and Synchronous mode of operation.

Table 63. UMSEL Bit Settings

UMSEL	Mode
0	Asynchronous Operation
1	Synchronous Operation

- **Bit 5:4 – UPM1:0: Parity Mode**

These bits enable and set type of parity generation and check. If enabled, the transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and compare it to the UPM0 setting. If a mismatch is detected, the PE Flag in UCSRA will be set.

Table 64. UPM Bits Settings

UPM1	UPM0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

- **Bit 3 – USBS: Stop Bit Select**

This bit selects the number of Stop Bits to be inserted by the Transmitter. The Receiver ignores this setting.

Table 65. USBS Bit Settings

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

- **Bit 2:1 – UCSZ1:0: Character Size**

The UCSZ1:0 bits combined with the UCSZ2 bit in UCSRB sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

Table 66. UCSZ Bits Settings

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

- **Bit 0 – UCPOL: Clock Polarity**

This bit is used for Synchronous mode only. Write this bit to zero when Asynchronous mode is used. The UCPOL bit sets the relationship between data output change and data input sample, and the synchronous clock (XCK).

Table 67. UCPOL Bit Settings

UCPOL	Transmitted Data Changed (Output of TxD Pin)	Received Data Sampled (Input on RxD Pin)
0	Rising XCK Edge	Falling XCK Edge
1	Falling XCK Edge	Rising XCK Edge

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal Mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed Mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master Mode	$BAUD = \frac{f_{OSC}}{2(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

รูปแสดงการคำนวณหาค่าของรีจิสเตอร์ UBRR เพื่อกำหนดอัตราบอดเรต

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

Bit	15	14	13	12	11	10	9	8	
	URSEL	–	–	–	UBRR[11:8]				UBRRH
	UBRR[7:0]								UBRRL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	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	

The UBRRH Register shares the same I/O location as the UCSRC Register. See the "Accessing UBRRH/ UCSRC Registers" on page 158 section which describes how to access this register.

- **Bit 15 – URSEL: Register Select**

This bit selects between accessing the UBRRH or the UCSRC Register. It is read as zero when reading UBRRH. The URSEL must be zero when writing the UBRRH.

- **Bit 14:12 – Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, these bit must be written to zero when UBRRH is written.

- **Bit 11:0 – UBRR11:0: USART Baud Rate Register**

This is a 12-bit register which contains the USART baud rate. The UBRRH contains the four most significant bits, and the UBRRL contains the 8 least significant bits of the USART baud rate. Ongoing transmissions by the transmitter and receiver will be corrupted if the baud rate is changed. Writing UBRRL will trigger an immediate update of the baud rate prescaler.

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

5. การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ UART ของไมโครคอนโทรลเลอร์

5.1 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ UART ของ AT89C51ED2

5.1.1 การเขียนฟังก์ชันกำหนดค่าเริ่มต้นของโมดูล UART ของ AT89C51ED2

```
void uart_init(int baud){    //Use XTAL 11.0592MHz
    SCON = 0x50;   //RX Enable Data 8 bit Baud rate variable
    TMOD = (TMOD & 0x0f) | 0x20; // Timer1 mode 8 Auto reload
    TH1 = 256 – ((XTAL/384)/baud);
    TL1 = TH1;
    TR1 = 1;
}
```


การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

5.1.2 ฟังก์ชันการรับค่าจากพอร์ต UART ของ AT89C51ED2

```
unsigned char uart_getc(){
    while(RI == 0);
    RI = 0;
    return SBUF;
}
```

5.1.3 ฟังก์ชันการส่งค่าจากพอร์ต UART ของ AT89C51ED2

```
void uart_putc(unsigned char dat){ //Send 1 Character
    SBUF = dat;
    while(TI == 0);
    TI = 0;
}
void uart_puts(unsigned char *str){ //Send String
    while(*str != '\0') uart_putc(*str++);
}
```


การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

5.2 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ UART ของ PIC16F887

5.2.1 การเขียนฟังก์ชันกำหนดค่าเริ่มต้นของโมดูล UART ของ PIC16F887

```
void uart_init(unsigned int baud) { //Xtal = 20MHz
    unsigned int SpeedUart;
    SpeedUart = ((_XTAL_FREQ/baud)/16) – 1;
    SPBRG = SpeedUart;
    SPBRGH = SpeedUart >> 8;
    SPEN = 1; //Serial Port Enable & Continuous Enable
    CREN = 1;
    TXEN = 1; //Tx Enable & High Speed mode
    BRGH = 1;
}
```


การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

5.2.2 ฟังก์ชันการรับค่าจากพอร์ต UART ของ PIC16F887

```
unsigned char uart_getc(){
    unsigned char dat;
    while (!RCIF);
    dat = RCREG;
    return dat;
}
```

5.2.3 ฟังก์ชันการส่งค่าจากพอร์ต UART ของ PIC16F887

```
void uart_putc(unsigned char c){ //Send 1 Character
    while(!TRMT);
    TXREG = c;
}
void uart_puts (char *s) { //Send String
    while (*s) {
        uart_putc(*s);
        s++;
    }
}
```


การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

5.3 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ UART ของ ATMEGA32

5.3.1 การเขียนฟังก์ชันกำหนดค่าเริ่มต้นของโมดูล UART ของ ATMEGA32

```
void uart_init(unsigned int baud){ //Xtal = 16MHz
    uint32_t SpeedUart;
    SpeedUart = (uint32_t) (F_CPU/((uint32_t)16*baud)) – 1;
    UBRRL = SpeedUart;
    UBRRH = SpeedUart >> 8;
    UCSRB = (1<<RXEN)|(1<<TXEN);
    UCSRC = (1<<URSEL)|(3<<UCSZ0);
}
```


การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

5.3.2 ฟังก์ชันการรับค่าจากพอร์ต UART ของ ATMEGA32

```
unsigned char uart_getc(){
    unsigned char dat;
    while ( !(UCSRA & (1<<RXC)));
    dat = UDR;
    return dat;
}
```

5.3.3 ฟังก์ชันการส่งค่าจากพอร์ต UART ของ ATMEGA32

```
void uart_putc(unsigned char c){ //Send 1 Character
    while(!(UCSRA & (1 << UDRE)));
    UDR = c;
}
void uart_puts (char *s){ //Send String
    while (*s) {
        uart_putc(*s);
        s++;
    }
}
```

การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

6. การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ One-Wire ของไมโครคอนโทรลเลอร์

6.1 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ One-Wire ของ AT89C51ED2 (DS18S20)

```
sbit onewire = P3^2;                    // Bit data 1-wire bus
void delayOneWire(unsigned int usecond){ }
unsigned char ow_reset(){ }
bit read_bit (void){ }
unsigned char ReadByte(){ }
void WriteByte(unsigned char com){ }
```


การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

6.2 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ One-Wire ของ PIC16F887

```
#define onewire RD0        // Bit data 1-wire bus
#define TRISD1wire TRISD0
void delay_us(int us){ }
char ow_reset(){ }
char read_bit (void){ }
void write_bit(unsigned char DQ){ }
unsigned char ReadByte(void){ }
void WriteByte(char val){ }
```


การรับส่งข้อมูลอนุกรมในรูปแบบ UART และ One-Wire

6.3 การใช้เขียนฟังก์ชันภาษาซีเพื่อควบคุมการรับส่งข้อมูลอนุกรมในรูปแบบ One-Wire ของ ATMEGA32

```
#define BIT_DQ  6                  // 1-Wire Interface (DS18B20 Temp. Sensor)
#define SET_IN_DQ          (DDRD &= ~(1<<BIT_DQ))
#define SET_OUT_DQ         (DDRD |= (1<<BIT_DQ))
#define OUT_HIGH_DQ        (PORTD |= (1<<BIT_DQ))
#define OUT_LOW_DQ         (PORTD &= ~(1<<BIT_DQ))
#define IN_DQ              (PIND & (1<<BIT_DQ))
void delay_us(unsigned int time_us){ }
unsigned char ow_reset(void){ }
unsigned char read_bit (void){ }
void write_bit(char bitval){ }
unsigned char ReadByte(void){ }
void WriteByte(char val){ }
```