Mechatronics Engineering

Serial Communications

Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science
German University in Cairo

Serial Vs. Parallel

Serial	Parallel
one bit at a time	one byte/word at a time
sync or async	sync
no data skew	data skew
noise immunity	crosstalk
simplex, half, or full duplex	simplex or half duplex
complex design	simple design
two/four wires	*n*-wires

GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Types of Serial Comm

- **RS232.** Peer-to-peer (i.e. communications between two devices)
- **RS485.** Multi-point (i.e. communications between two or more devices)
- **USB** (Universal Serial Bus). Replaced RS232 on desktop computers.
- **CAN** (Controller Area Network). Multi-point. Popular in the automotive industry.
- **SPI** (Serial Peripheral Interface). (1979) Developed by Motorola. Synchronous master/slave communications.
- **I2C** (Inter-Integrated Circuit). (1982) Developed by Philips. Multimaster communications. Also called two-wire interface (TWI).

Serial Comm. Comparisons

	S/A	Type	Duplex	#Device	speed (kbps)	distance (ft)	Wires
RS-232	A	peer	full	2	20	30	2+
RS-422	A	multi-drop	half	10	10,000	4,000	1+
RS-485	A	multi-point	half	32	10,000	4,000	2
I2C	S	multi-master	half	?	3,400	<10	2
SPI	S	multi-master	full	?	>1,000	<10	3+
Microwire	S	master/slave	full	?	>625	<10	3+
1-Wire	A	master/slave	half	?	16	1,000	1

GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Serial Comm?

- Serial data is transmitted between devices one bit at a time using agreed upon electrical signals.
- In what order are the series of bits shifted across the data line?
- What constitutes a CLK event?
- How does the peripheral know when it is supposed to receive bytes and when it is supposed to transmit bytes?

The USART

- Universal Asynchronous Receiver/ Transmitter.
- The interface between the microprocessor and the serial port.
- responsible for breaking apart bytes of data and transmitting them one bit at a time
- receives serialized bits and converts them back into bytes

USART Specifics

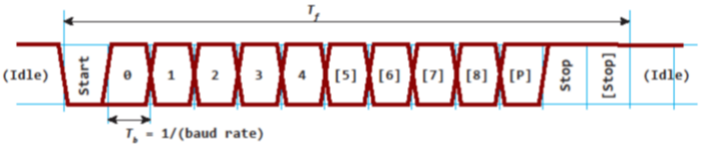
- Two wires Rx and Tx
- Full duplex
- Synchronous or Asynchronous?
- Requires:
  - start/stop bit
  - parity
  - pre-defined timing
  - frame size

USART Options

Start Bits	Stop Bits
1	1, 2
Parity	**Frame Sizes**
Even/Odd/None	5,6,7,8,9

GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

USART Frames



Start bit, always low

Parity bit, can be even, odd, or none

Stop bit, always high

Idle, not transfers, always high

USART Frames

First Frame

Second Frame

$$P_{odd} = b_7 \oplus b_6 \oplus b_5 \oplus b_4 \oplus b_3 \oplus b_2 \oplus b_1 \oplus b_0 \oplus 1$$

$$P_{even} = b_7 \oplus b_6 \oplus b_5 \oplus b_4 \oplus b_3 \oplus b_2 \oplus b_1 \oplus b_0 \oplus 0$$

USART0 Control and Status Registers

UCSR0A

0xC0	7	6	5	4	3	2	1	0
Name	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Read/write	R	R/W	R	R	R	R	R/W	R/W
Default	0	0	1	0	0	0	0	0

- RXC0 Receive Complete. Data received but not yet read.
- TXC0 Transmit Complete. Set when all data has transmitted.
- UDRE0 Data Register Empty. Set when UDR0 register is ready for new data.
- FE0 Frame Error. Set when UDR0 has data in which the first stop bit is zero (Frame Error)
- DOR0 Data Overrun. Set when the UDR0 was not read before the next frame arrived.
- UPE0 Parity Error. Set when the next frame in UDR0 has a parity error.
- U2X0 Double Transmission speed. For asynchronous operation to speed double
- MPCM0 Multi-processor communication mode. If no addressing information is provided, incoming data is ignored when this is set.

GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

USART0 Control and Status Registers

UCSR0B

0xC1	7	6	5	4	3	2	1	0
Name	RXCIE0	TXCIEO	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Default	0	0	0	0	0	0	0	0

- RXCIE0 RX Complete Interrupt Enable. Allows interrupts upon receive completes.
- TXCIE0 Transmit Complete Interrupt Enable.Allows interrupts upon transmit completes.
- UDRIE0 Data Register Empty Interrupt Enable. Allows interrupts upon empty data reg.
- RXEN0 Receive Enable. Allows data receive, and reconfigures the pin for receives.
- TXEN0 Transmit Enable. Allows data transmits,and reconfigures the pin for transmission.
- UCSZ02 Character size bit 2. The other bits are in register UCSR0C.
- RXB80 Receive Data Bit 8. This is the 8th data bit of a received character. The other bits are in UDR0.
- TXB80 Transmit Data Bit 8. This is the 8th data bit of a character to be transmitted. The other bits go in UDR0. This bit must be written first.

GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

USART0 Control and Status Registers

UCSR0C	0xC2	7	6	5	4	3	2	1	0
	Name	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0
	Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	Default	0	0	0	0	0	1	1	0

UMSEL	MODE
00	Asynch USART
01	Synch. USART
10	----
11	Master SPI (MSPIM)

UPM0	Parity MODE
00	Disabled
01	----
10	Enabled, Even
11	Enabled, Odd

UCSZ	Frame Size
000	5-bit
001	6-bit
010	7-bit
011	8-bit
100	---
101	---
110	---
111	9-bit

UCPOL0 Clock Polarity. Set to transmit on falling edge and sample on rising edge.
Clear to transmit on rising edge and sample on falling edge.

GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

USART0 Baud Rate Registers

UBRR0H

0xC5	7	6	5	4	3	2	1	0
Name	–	–	–	–	UBRR011	UBRR010	UBRR09	UBRR08
Read/write	R	R	R	R	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

UBRR0L

0xC4	7	6	5	4	3	2	1	0
Name	UBRR07	UBRR06	UBRR05	UBRR04	UBRR03	UBRR02	UBRR01	UBRR00
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

USART0 Baud Rates

Baud Rate (bps)	U2X0=0 UBRR0	U2X0=1 UBRR0
2400	0x01A0	0x0340
4800	0x00CF	0x01A0
9600	0x0067	0x00CF
14400	0x0044	0x008A
19200	0x0033	0x0067
28800	0x0022	0x0044
38400	0x0019	0x0033
57600	0x0010	0x0022
76800	0x000C	0x0019
115200	0x0008	0x0010
230400	0x0003	0x0008
250000	0x0003	0x0007
500000	0x0001	0x0003
100000	0x0000	0x0001

U2X0 = 0

$$\left[\frac{10^6}{\text{Baud Rate}}\right] - 1$$

U2X0 = 1

$$\left[\frac{2 \times 10^6}{\text{Baud Rate}}\right] - 1$$

$$\text{Baud rate} = \frac{\text{Fosc}}{(\text{UBRR}+1) \times 16}$$

GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

UDR0: UART**0** I/O Data Register

0xC6	7	6	5	4	3	2	1	0
Name	UD7	UD6	UD5	UD4	UD3	UD2	UD1	UD0
Read/ write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

USART Example Initialize

```
#define BAUD_PRESCALER 0x0067 //9600 baud
Void USART_init(void)
{
   // Set The Baud Rate Prescale Rate register
   UBRR0 = BAUD_PRESCALER;
   // Set frame format: 8data, 1 stop bit.
   UCSR0C = ((0<<USBS0) | (1 << UCSZ01)|(1<<UCSZ00));
   // Enable receiver and transmitter
   UCSR0B = ((1<<RXEN0)|(1<<TXEN0));
}
```


USART Example TX

- write the bytes to be transmitted to the UDR register.
- make sure that this register is empty before doing this
  - avoid overwriting data that has not yet been transmitted.
  - keep checking the UDRE bit in the UCSR0A register before writing

```
void USART_send( unsigned char data)
{
  //while the transmit buffer is not empty loop
  while(!(UCSR0A & (1<<UDRE0)));
  // when the buffer is empty write data to the transmitter
  UDR0 = data;
}
```

USART Example Program

```
void setup(void)
{
   USART_init();
   USART_send('a');   // Send a
   USART_send(0x61);  // Send a
   USART_send('\r');  // Send carriage return
   USART_send('\n');  // Send linefeed
}
```


USART Example Send String

```
// sends the characters from the string one at a
//time to the USART
void USART_putstring(char* StringPtr)
{
   while(*StringPtr != 0x00)
   {
       USART_send(*StringPtr);
       StringPtr++;
   }
}
```

Usage: `USART_putstring("Hello There.");`

```
       volatile uint8_t buffer[20];
       USART_putstring(buffer);
```

GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

USART Example RX

- Keep checking the RXC bit in the UCSR0A register to see if data has been received
- Read the data from the UDR0 register.

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


USART Interrupts

```
#define BAUD_PRESCALER 0x0067
void USART_interrupt_init(void)
{
cli();
//Set the baud rate prescale rate register
UBRR0 = BAUD_PRESCALER;
// Enable receiver And transmitter and Rx interrupt
UCSR0B = ((1<<RXEN0)|(1<<TXEN0)|(1<< RXCIE0));
// Set frame format: 8data, 1 stop bit.
UCSR0C = ((0<<USBS0)|(1 << UCSZ01)|(1<<UCSZ00));
sei();
}
```

USART Interrupts

```
volatile int i=0;
volatile uint8_t buffer[20];
volatile uint8_t StrRxFlag=0;
ISR(USART0_RX_vect)
   buffer[i]=UDR0;  //Read USART data register
   //check for carriage return terminator and increment
   //buffer index
   if (buffer[i++]=='\r')
   {
       // if terminator detected
       StrRxFlag=1;  //Set String received flag
       buffer[i - 1]=0x00; //Set string terminator To 0x00
       i=0; //Reset buffer index
   }
}
```

GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science