# Universal Asynchronous Receive Transmit

Microprocessor Interfacing

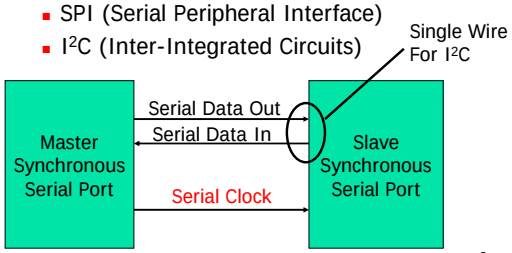
## Overview

- Serial Standard
- EIA Standard
- Hardware UART
- Configuring UART

2

## Synchronous Serial Standard

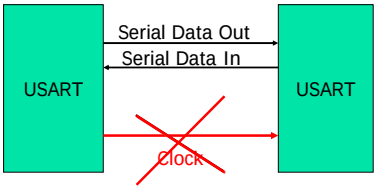
- Master Synchronous Serial Port Module
  - SPI (Serial Peripheral Interface)
  - I²C (Inter-Integrated Circuits)




3

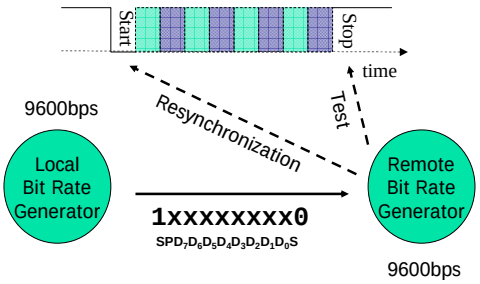
## Asynchronous Serial Standard

- Universal Synchronous/Asynchronous Receiver Transmitter (USART or UART)




4

## Asynchronous Resynchronization




5

## Asynchronous Serial Standard

10 Bits

1 Start bit => 0
7 to 8 Data bits LSB to MSB
1 Parity Bit optional
1 Stop bit

Even Parity Gen. = ($\Sigma$ $d_n$) mod 2
Odd Parity Gen. = [($\Sigma$ $d_n$)+1] mod 2


Even Parity Test : ($\Sigma$ dn+P) mod 2 == 0
Odd Parity Test : [($\Sigma$ dn)+1+P] mod 2 == 0

6

## ASCII (pronounced ASK-key)

- American Standard Code for Information Inter-change
  - 7-bits
  - First 32 codes for printer control
  - Last 96 printable characters
  - IBM added 8th bit

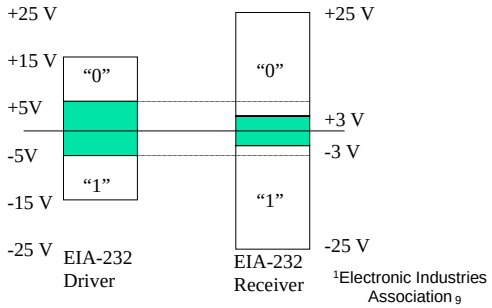
7

## PIC24F UART Protocols

- EIA-232 (RS-232)
- EIA-485 (RS-485)
- LIN bus (Local Interconnect Network)
- IrDA (Infrared Data Association)

8

## EIA-232 Voltages (EIA[1] Voltages)

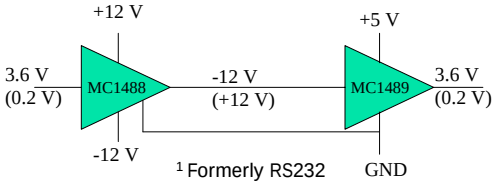



[1]Electronic Industries Association

9

## EIA-232E of 1991

- EIA-232 E common asynchronous serial line standard[1]. EIA-232 is the EIA equivalent of ITU-T V.24 and V.28.

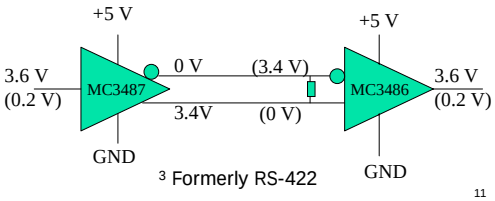



[1] Formerly RS232

10

## EIA-422

- EIA422A serial line standard[3] which specifies differential drivers and receivers

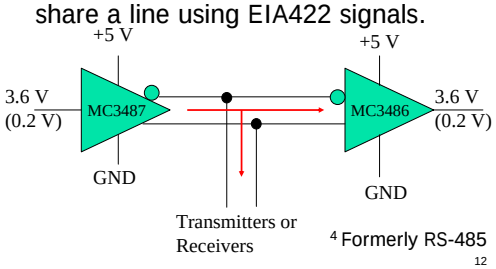



[3] Formerly RS-422

11

## EIA-485

- EIA485[4] allows multiple devices to share a line using EIA422 signals.




[4] Formerly RS-485

12

## UART Drivers and Standards

- Use MC1488 and MC1489 drives with PIC for EIA232, requires –12 VDC, +12 VDC, and +5 VDC supply
- Use MC3487 and MC3486 drivers with PIC for EIA422 requires +5 VDC supply
- MAX232A provides two EIA232 outputs and two EIA232 inputs requires +5 VDC supply

13

## EIA Terminology – Telecom Terms

- DTE – Data terminal equipment (computer)
- DCE – Data communication equipment (modem)
- RxD – Receiver data
- TxD – Transmitter data
- DCD – Data carrier detect (valid modem connection)

14

## EIA Terminology

- DTR – Data terminal ready (Computer on and software is ready)
- DSR -- Data set ready (Modem on and software ready)
- RTS – Request to send (DTE wants to send character)
- CTS – Clear to send (DCE acknowledge to RTS from DTE)
- RI – Ring indicator from telephone

15

## DE-9 and DB-25 Connector

| Signal | DE-9 | DB-25 |
|---|---|---|
| DCD | 1 | 8 |
| RxD | 2 | 3 |
| TxD | 3 | 2 |
| DTR | 4 | 20 |
| GND | 5 | 7 |
| DSR | 6 | 6 |
| RTS | 7 | 4 |
| CTS | 8 | 5 |
| RI | 9 | 22 |

16

## Standard and Null Modem Cable

**Using DE-9**

| DTE Null | DTE | DCE Standard |
|---|---|---|
| Open | DCD (1) | 1 |
| 2 | RxD (2) | 2 |
| 3 | TxD (3) | 3 |
| 4 | DTR (4) | 4 |
| 5 | GND (5) | 5 |
| Open | DSR (6) | 6 |
| 7 | RTS (7) | 7 |
| Open | CTS (8) | 8 |
| Open | RI (9) | 9 |

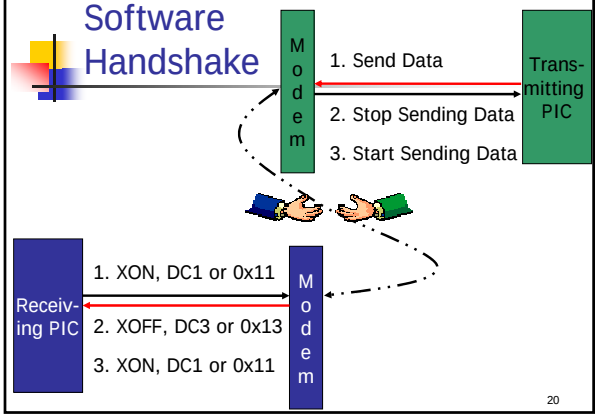
17

## Baud Rate

- Baud is state changes per second.
- When two states ( 0 and 1) then baud equals bits per second or bps
- Standard baud rates for PIC 300, 1200, 2400, 9600, 19200, 38400, 57600, 76800, 96000, 115200 and others
- Baud rate specified in UxBRG and BRGH

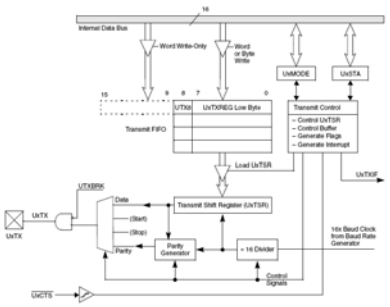
18

## Software Handshake



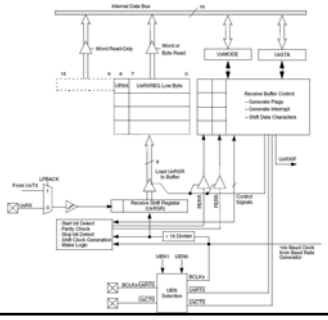

20

## Hardware UART Tx



21

## Hardware UART Rx



22

## Hardware UART (BRGH=1)

Hardware Timed

- Receive ($4 \bullet (X+1) \bullet T_{cyc}$ = 1 bit period)
  - Wait for 0
  - Wait $2 \bullet (X+1) \bullet T_{cyc}$, If 0 continue else restart
  - Wait $4 \bullet (X+1) \bullet T_{cyc}$, get $D_0$
  - Wait $4 \bullet (X+1) \bullet T_{cyc}$, get $D_1$
  - . . .
  - Wait $4 \bullet (X+1) \bullet T_{cyc}$, get $D_7$ or parity
  - Wait $4 \bullet (X+1) \bullet T_{cyc}$, If 1 okay else frame error.

23

## Hardware UART (BRGH=0)

- Receive ($16 \bullet (X+1) \bullet T_{cyc}$ = 1 bit period)
  - Wait for 0
  - Wait $8 \bullet (X+1) \bullet T_{cyc}$, If 0 continue else restart
  - Wait $16 \bullet (X+1) \bullet T_{cyc}$, get $D_0$
  - Wait $16 \bullet (X+1) \bullet T_{cyc}$, get $D_1$
  - . . .
  - Wait $16 \bullet (X+1) \bullet T_{cyc}$, get $D_7$ or parity
  - Wait $16 \bullet (X+1) \bullet T_{cyc}$, If 1 okay else frame error.

24

## Configuring UART

```
void InitUART(unsigned char baud,unsigned int config) {
// Fosc = 8 MHz
      switch(baud) {
            case BPS9600 : U1BRG=207;
                  break;
            case BPS19200 : U1BRG=103;
                  break;
            case BPS57600 : U1BRG=35;
                  break;
            default : SPBRG=832; // 2400
```

25

## Configuring UART

```
switch(config) {
        case POLL :  _U1RXIE=0;
                _U1TXIE=0;
                break;
        case RCONLY : _U1RXIE=1;
                _U1TXIE=0;
                break;
        case TXONLY :     _U1RXIE=0;
                _U1TXIE=1; TXIP=1;
                break;
        case TXRC :  _U1RXIE=1;
                _U1TXIE=1;
                break;
        default :    _U1RXIE=0;
                _U1TXIE=0;
        }
```

26

## Configuring UART

```
U1MODEbits.BRGH=1;        //data rate for sending
U1MODEbits.UARTEN=1;      //enable serial port pins
U1STAbits.UTXEN=1;        //enable transmitt
    }
```

27

## Clear Errors

```
void clearUARTError(void){
unsigned char dummy;
    if(OERR){
            TU1STAbits.OERR=0;
        }
    if(FERR){
            dummy=U1RXREG;
        }
}
```

28

## Send (Transmit) Character

```
void putch(char c) {
        while(!_U1TXIF) clearUARTError( );
        U1TXREG=c;
        T60us( );
        }

bit kbhit(void){
        return _U1RXIF;
}
```

29

## Receive Character

```
char getch(void) {
        while(!_U1RXIF) clearUARTError( );
        return U1RXREG;
        }

char getche(void) {
        unsigned char ch;
        ch=getch( );
        putch(ch);
        return ch;
        }
```

30

## Error Detection

- Parity Bits
  - Pro: Calculated by UART hardware
  - Cons: Does not detect pair reversals
- Hamming Code
  - Pro: Corrects errors
  - Con: Requires 12-bits per byte and bit oriented

31

## Error Detection

- Checksum
  - Pros: Simple algorithm
  - Cons: Does not detect – reordered byte, inserted or deleted zero-value bytes, and multiple errors which sum to zero
- Cyclic Redundancy Check
  - Pros: Detects – All single bit errors, all two-bit errors within order of polynomial, all odd number bit errors, and all burst error less than order of polynomial
  - Cons: Bit oriented. Does not detect – strings of zeros

32

## Hamming Code

- Error Correcting Code
- Bit Oriented
- (11,7) Hamming Code
- Uses even parity

| | $P_1$ | $P_2$ | $D_1$ | $P_3$ | $D_2$ | $D_3$ | $D_4$ | $P_4$ | $D_5$ | $D_6$ | $D_7$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Data Word | | | 0 | | 1 | 1 | 0 | | 1 | 0 | 1 |
| $P_1$ | **1** | | 0 | | 1 | | 0 | | 1 | | 1 |
| $P_2$ | | **0** | 0 | | | 1 | 0 | | | 0 | 1 |
| $P_3$ | | | | **0** | 1 | 1 | 0 | | | | |
| $P_4$ | | | | | | | | **0** | 1 | 0 | 1 |
| Tx Data | **1** | **0** | 0 | **0** | 1 | 1 | 0 | **0** | 1 | 0 | 1 |

33

## Checksum

- Message $M_i$ of length n
- Standard

$$Checksum = \left[\left(\sum_{i=0}^{n} M_i\right) \bmod 256\right]_{TwosComplement}$$

- Flecher-32

$$A = \left(1 + \sum_{i=0}^{n} M_i\right) \bmod 65535$$

$$B = \left(n + \sum_{i=0}^{n} (n-i) \bullet M_i\right) \bmod 65535$$

$$Checksum = B \bullet 65536 + A$$

- Adler-32

$$A = \left(1 + \sum_{i=0}^{n} M_i\right) \bmod 65521$$

$$B = \left(n + \sum_{i=0}^{n} (n-i) \bullet M_i\right) \bmod 65521$$

$$Checksum = B \bullet 65536 + A$$

34

## Cyclic Redundancy Check

- All arithmetic is Modula-2
  - By bit manipulation
  - By byte-wise table look-up
- *M(x)* is message
- *Q(x)* is quotient result
- *K(x)* key polynomial (example CRC-16 IBM)
- *R(x)* is remainder of result (CRC)

$$M(x) \bullet x^n = Q(x) \bullet K(x) + R(x)$$

$$\frac{M(x) \bullet x^n}{K(x)} = Q(x) + \frac{R(x)}{K(x)}$$

35

## Common CRCs

| Name | Polyomial | Use |
|---|---|---|
| CRC-1 | x+1 | Parity |
| CRC-5 USB | $x^5+x^2+1$ | USB |
| CRC-15 CAN | $x^{15}+x^{14}+x^{10}+x^8+x^7+x^4+x^3+1$ | CAN |
| CRC-16 IBM | $x^{16}+x^{15}+x^2+1$ | Serial |
| CRC-32 MPEG2 | See web | MPEG2 |
| CRC-32 IEEE | See web | Ethernet 802.3 |
| Others | See web | |

36

## Homework

- Chapter 8 (Handout)
  - 1 (Write helper function)
  - 2 (Read helper function)

38