USART and Serial Communication

ECET 209 – Lecture 3

Introduction to Microcontrollers

Overview

- STDIO
- Alternate functions & the USART
- ASCII data

Advantages to C

- Stated on the first day that there are several advantages to using the C programming language
  - Standard Input and Output Functions
    - Allow us to use the built in functions to utilize the serial port to transmit serial data to the PC

STDIO

- Standard C functions that allow a method of sending or gathering information from your program.
- Ability to format numeric and text information.

The printf() function

- One of the standard I/O functions that allows us to send text information to the terminal window through the serial USART.

- For example:

  printf(“Hello World”);

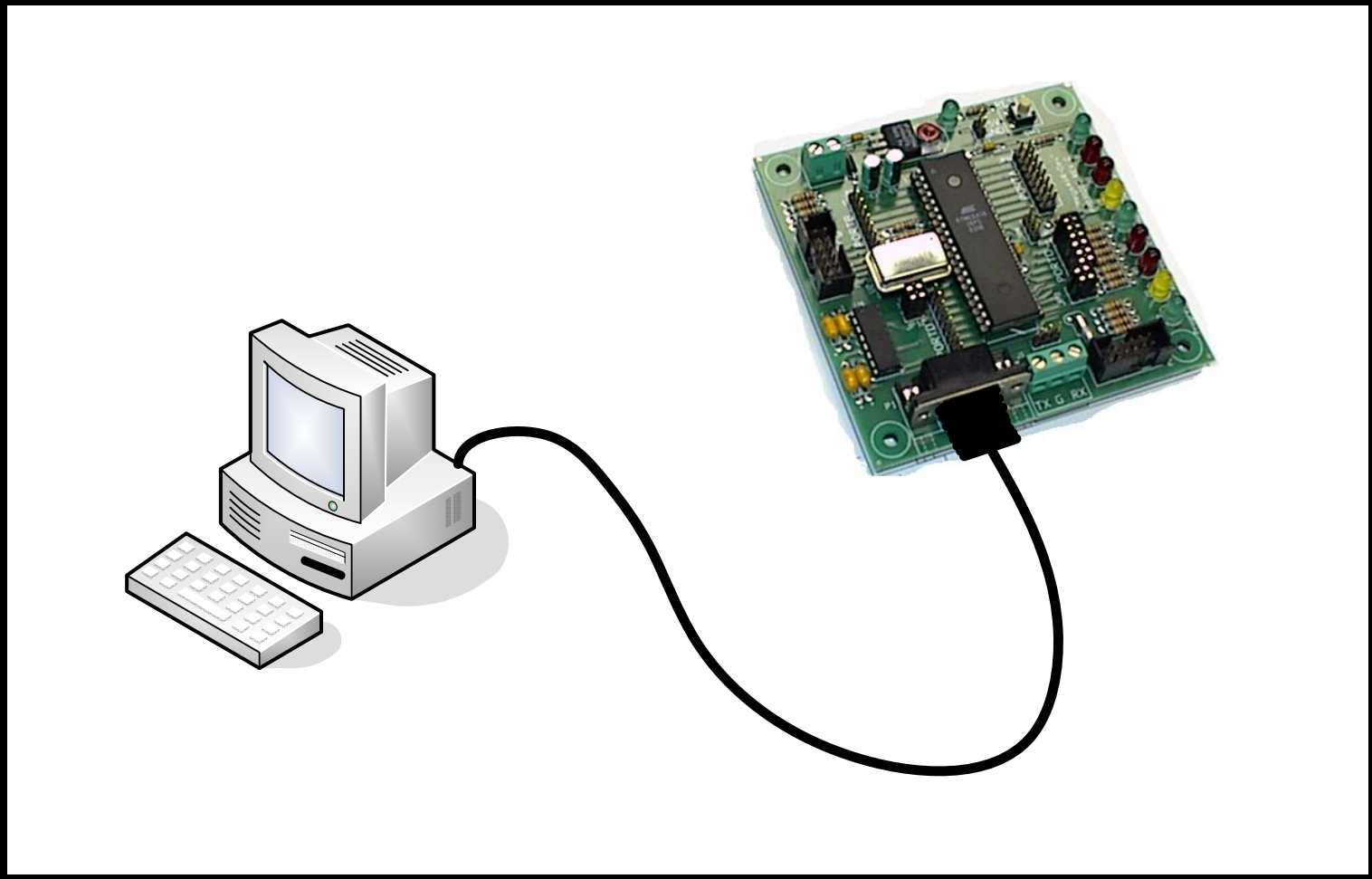
Hello World

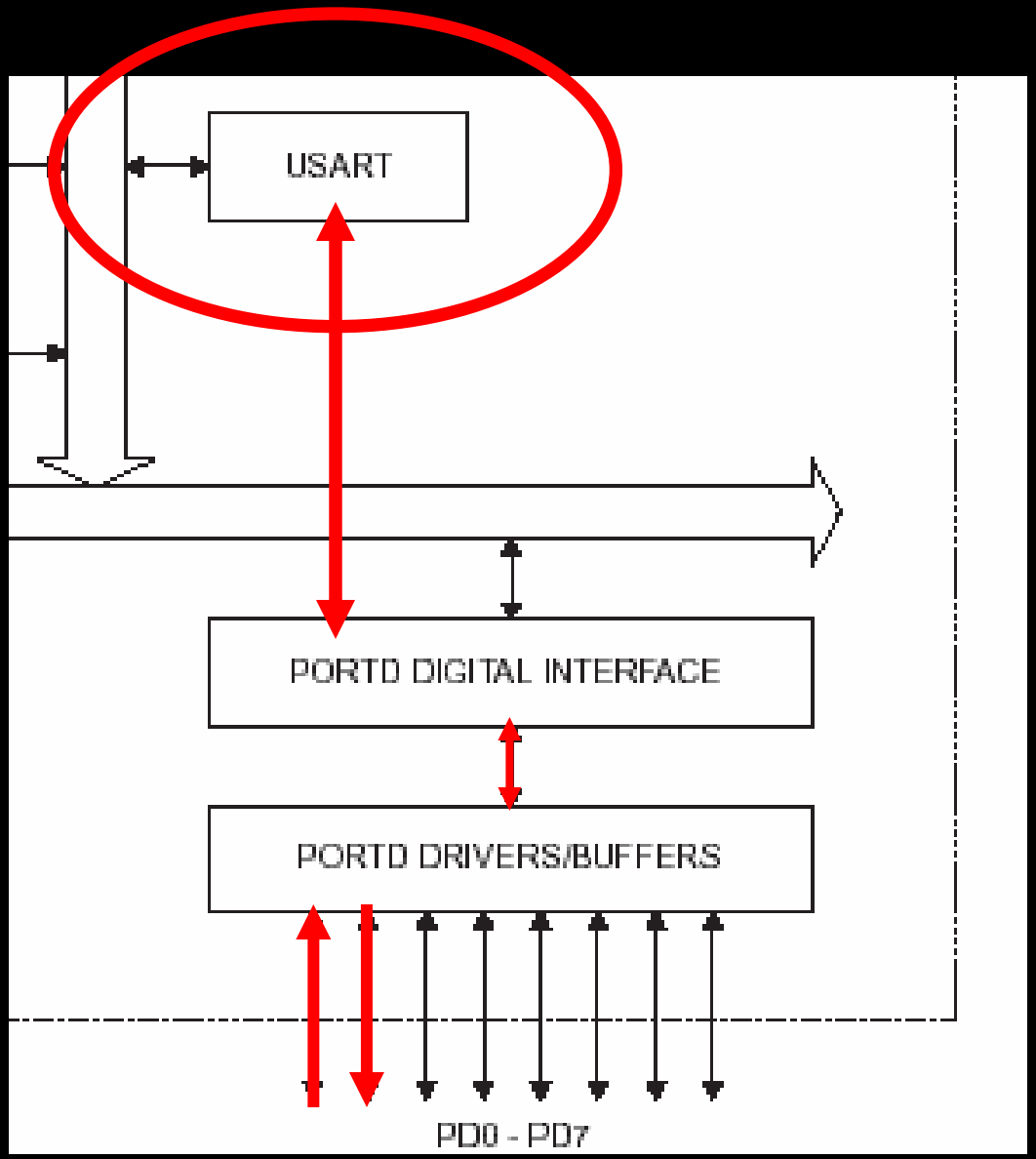
Alternate Functions

- Many Port Pins have alternate (specialized) functions.

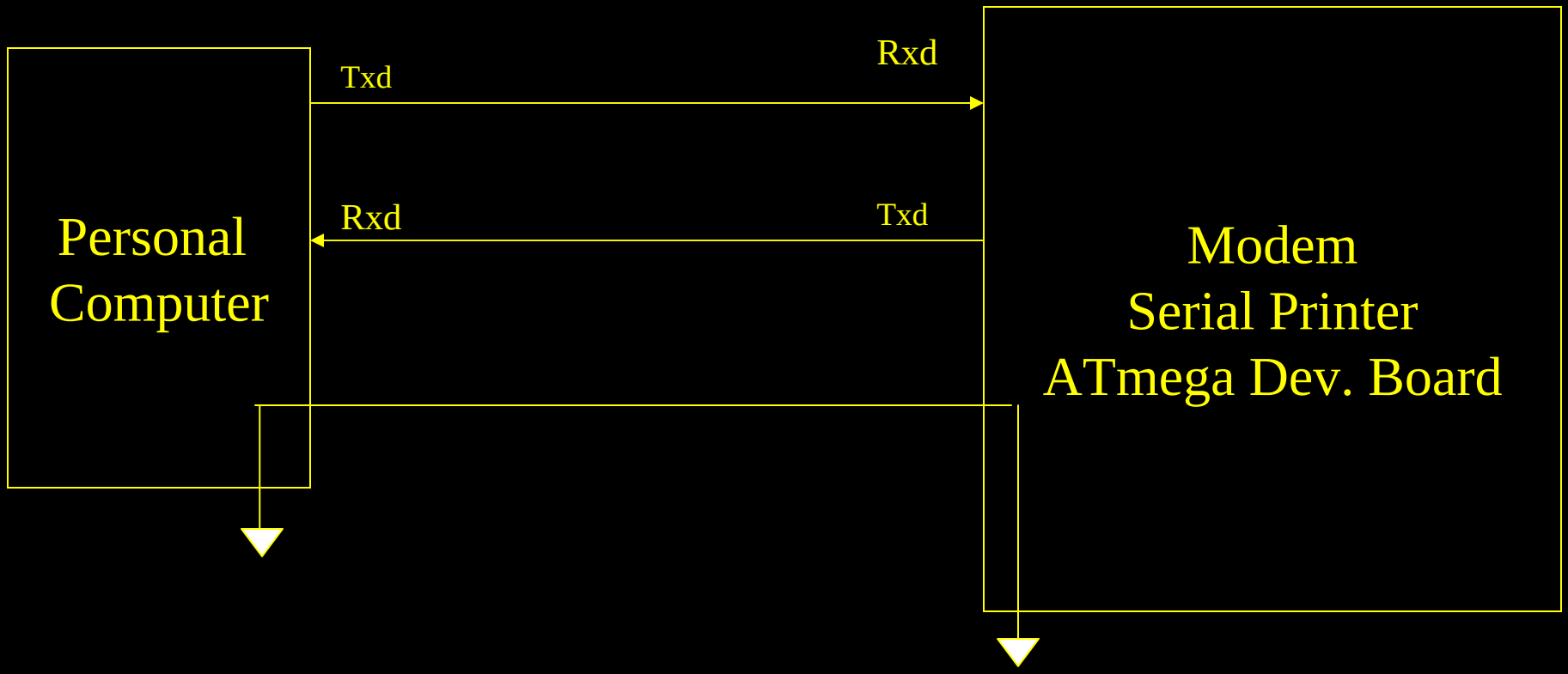
- One example is the serial port.
- Specifically used to communicate with the PC.

Serial Data Connection




USART
PORTD DIGITAL INTERFACE
PORTD DRIVERS/BUFFERS
PD0 - PD7

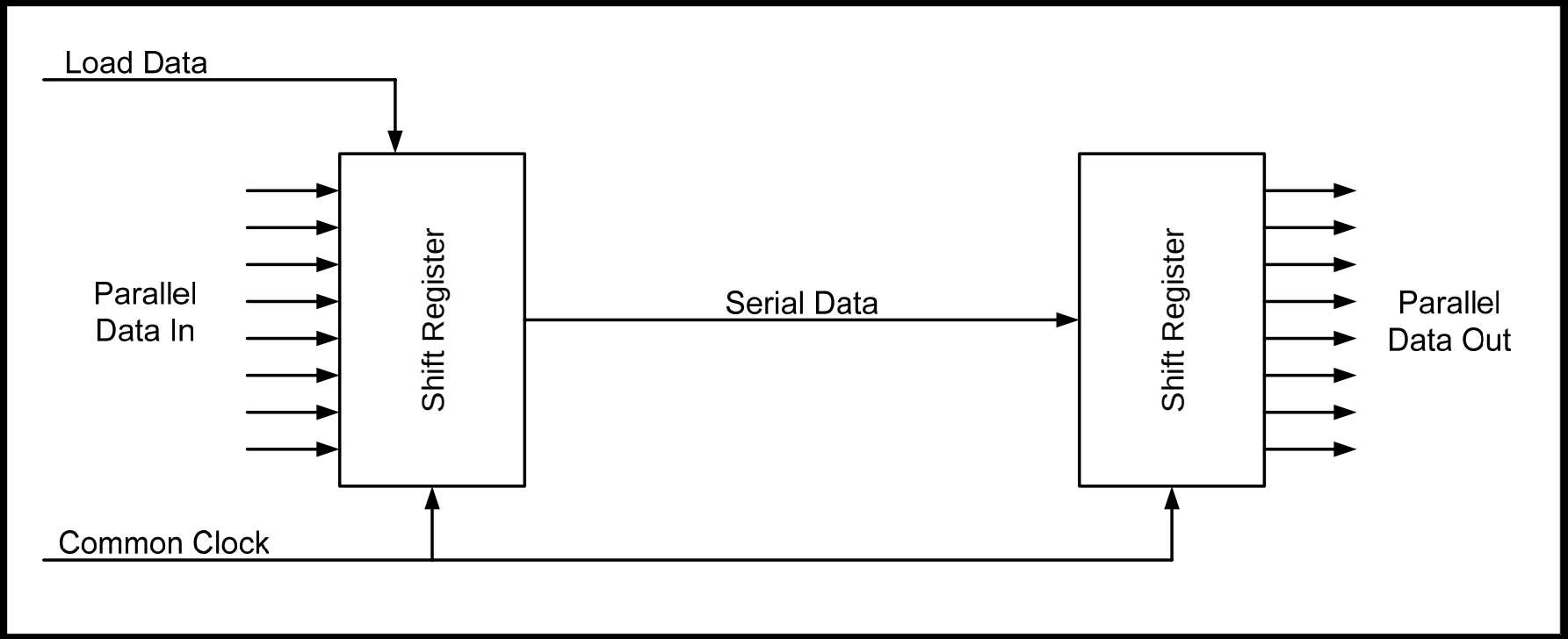
Typical data communication



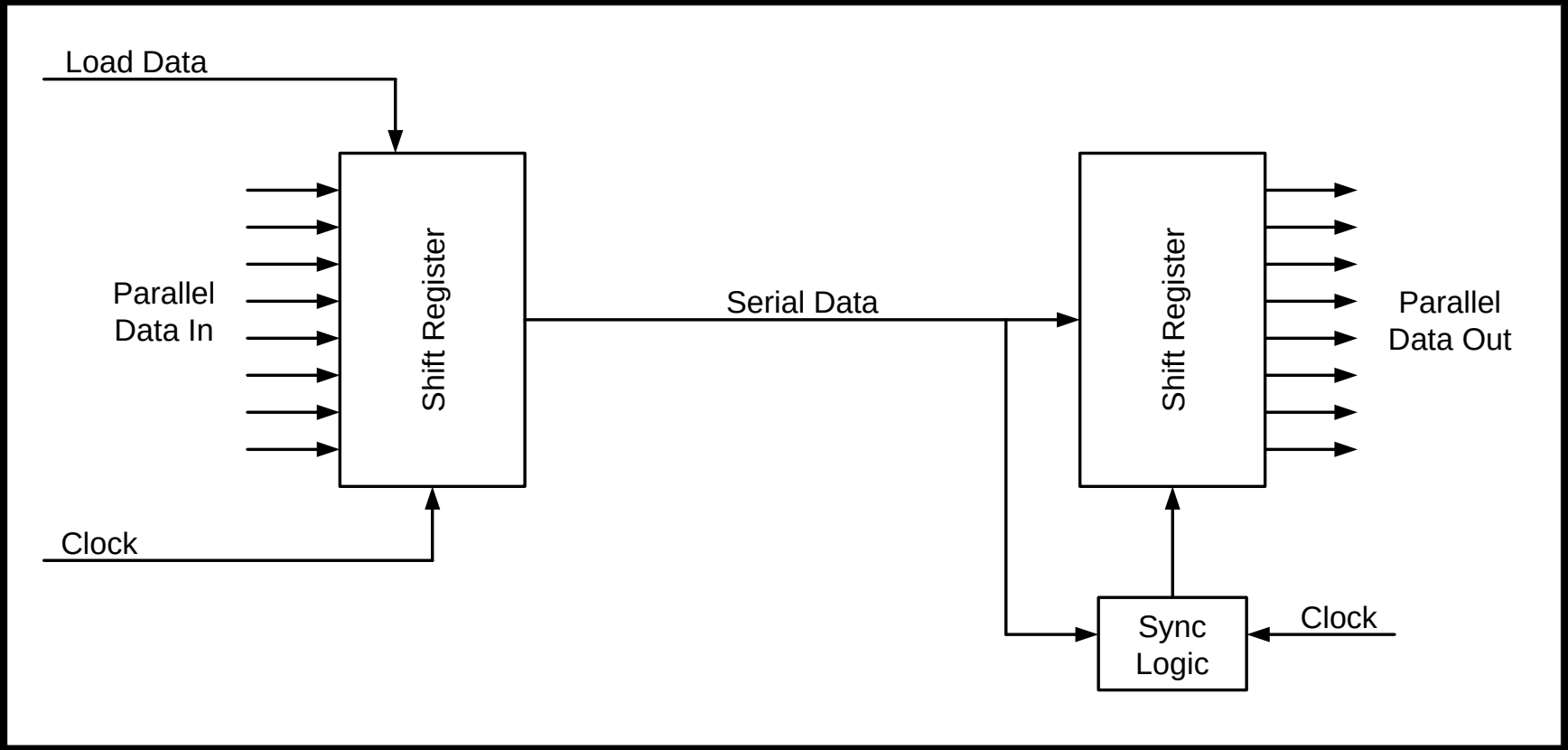
Serial Data Transmission

- Data is sent one bit at a time
- From a Parallel in / Serial out shift register
- To a Serial in / Parallel out shift register

Simplified Serial Transfer



Simplified Serial Transfer



PC serial ports e.g. COM1, COM2

- Standard RS-232 interface
- Asynchronous transmission
- Protocol must be the same on both transmitter and receiver
- Mostly used to send ASCII characters

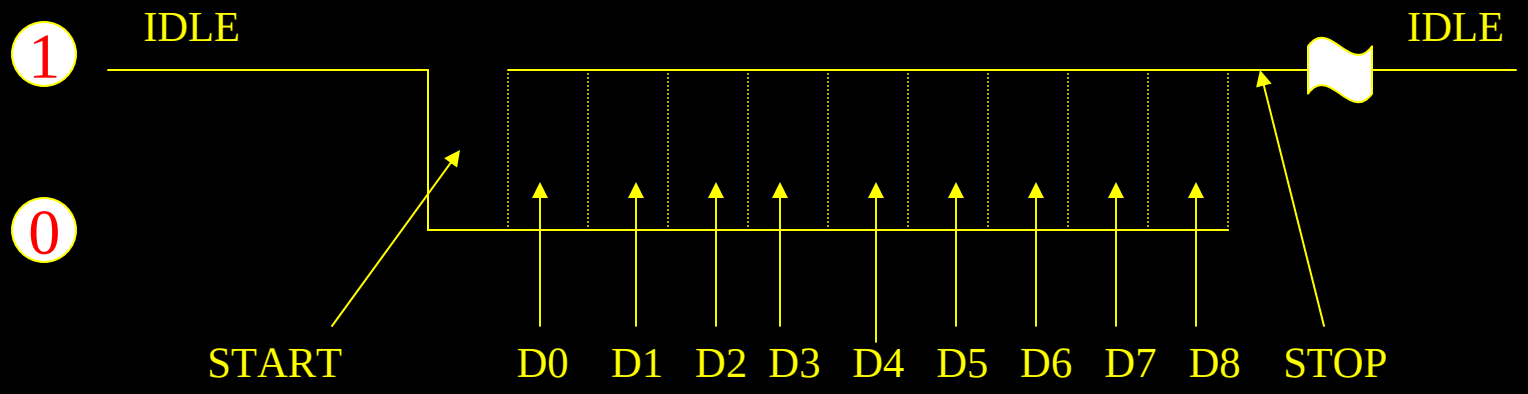
What is RS-232

- Standard developed years ago to standardize data communication between devices
- Specs logic levels:
  - ZERO +3V up to +24V
  - ONE -3V down to -24 V
- Includes synchronous and asynch modes

Asynchronous Data Format

- Idle Line defined as Logic “1”
- Requires START bit (Logic “0”)
- Specified number of data bits (5 - 8 typical)
  - Least significant bit is sent first
- Optional Even or Odd Parity bit
- Specified number of stop bits (1 - 2)
- Specified rate of transmission (baud rate)
- Baud rate = bits/sec
- Time for each bit = 1/Baud rate

Serial data protocol
TTL Levels

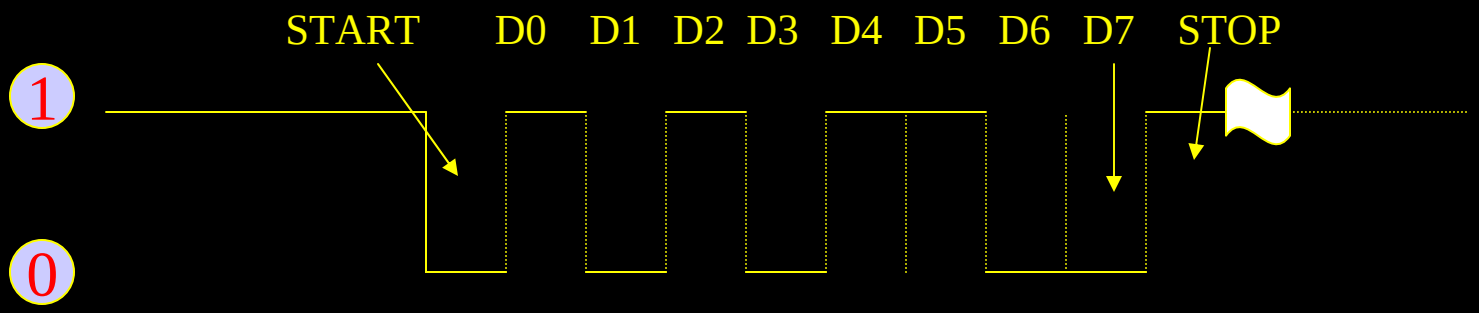


EXAMPLE: ASCII “5”

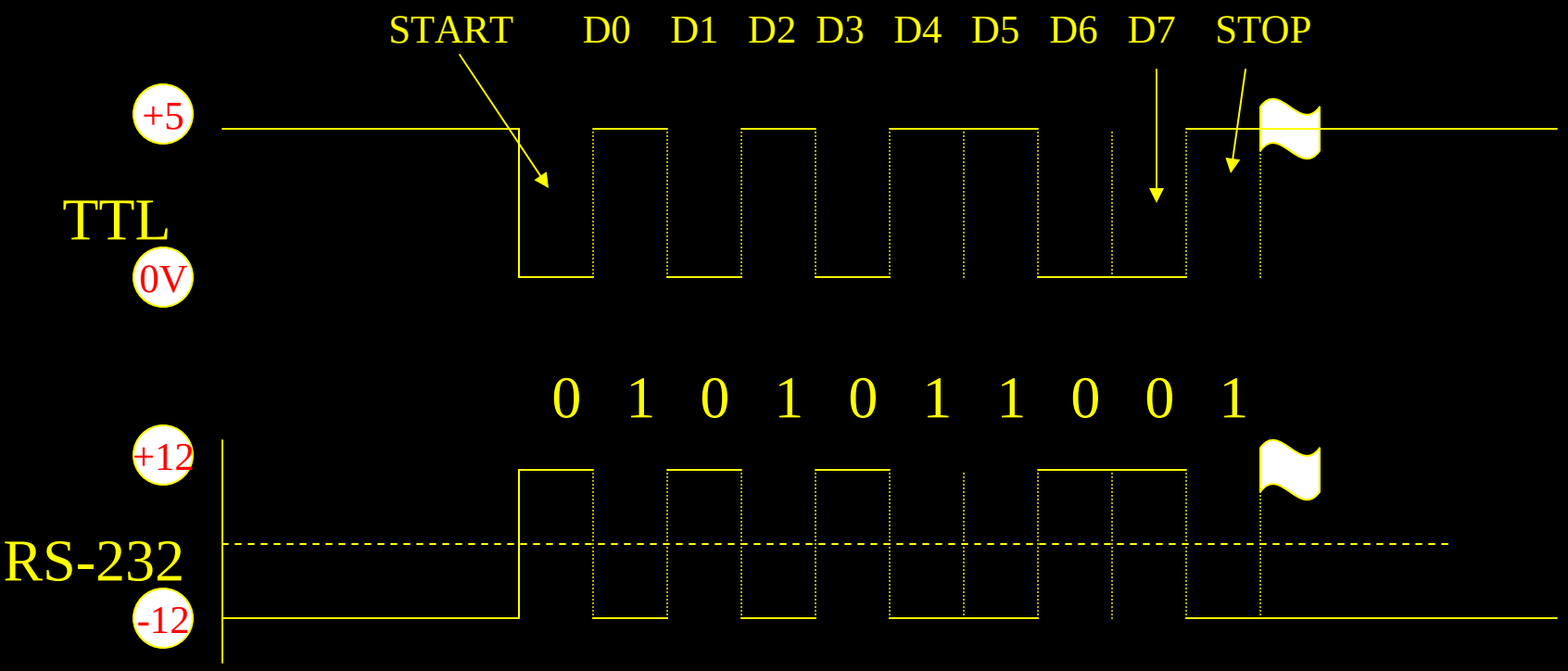
PROTOCOL: 8 DATA BITS, NO PARITY, 1 STOP BIT, 9600 BAUD

8-BIT ASCII “5” = 35H = 0011 0101

Note: TTL Levels



RS -232 LEVELS



Initializing the USART?

- Just as before (with the I/O Ports), we have to enable the USART function of the microcontroller

Configuration

- Configuring the USART:

  At a minimum, you MUST

  – set the baud rate
  – enable the transmitter and/or receiver
  – specify the number of data bits

Configuring the USART

- The baud rate is controlled through the UBRRL and UBRRH registers.
- Two options for determining the value to load into these registers
  - Select the value from a table
  - Calculate the value

Table 69. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	f_{osc} = 3.6864 MHz				f_{osc} = 4.0000 MHz				f_{osc} = 7.3728 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%
0.5M	–	–	0	-7.8%	–	–	0	0.0%	0	-7.8%	1	-7.8%
1M	–	–	–	–	–	–	–	–	–	–	0	-7.8%
Max [1]	230.4 kbps		460.8 kbps		250 kbps		0.5 Mbps		460.8 kbps		921.6 kbps	

1. UBRR = 0, Error = 0.0%

Manual Calculation

- The data sheets supply the following equation

$$BAUD = Fclk / (16 * (UBRR + 1))$$

(rearranged)

$$UBRR = (Fclk / (BAUD * 16)) - 1$$

UBRR = (Fclk / (BAUD * 16)) - 1

Manual Calculation

- Solve the equation

UBRR = (6,000,000 / (9600 * 16)) – 1

UBRR = 38.0625

Configuration

- UBRR = 38.0625 gets rounded to 38
- Assign the rounded value to UBRRL and UBRRH
  - set UBRRL equal to 38
  - set UBRRH equal to 0

C Code

- The C code to set the baud rate then becomes:

```
UBRRL = 38;          // set the baud rate to 9600
UBRRH = 0;           //     with 6MHz clock
```

- If you prefer hexadecimal numbers

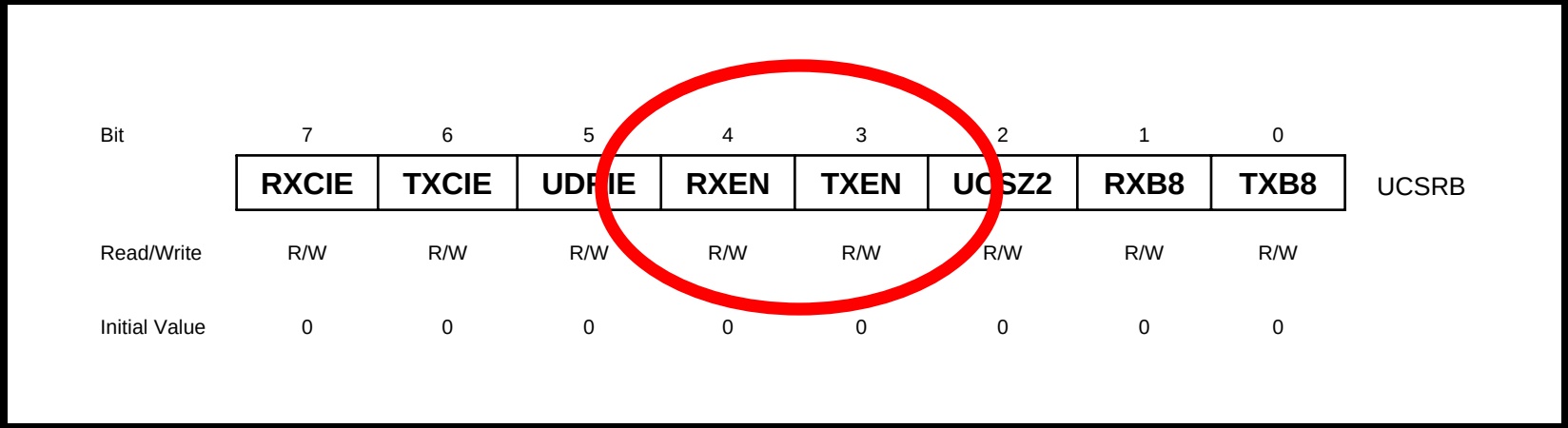
```
UBRRL = 0x26;
```

Manual Configuration

- Set the baud rate
- Enable the transmitter and receiver
- Specify the number of data bits

Enable TX and RX

- Bits to enable both the transmitter and receiver are located in the UCSRB register.



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/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

C Code

- The C code to set the TXEN and RXEN

```
UCSRB = 0x18;          // enable TX and RX
```

Manual Configuration

- Set the baud rate
- Enable the transmitter and receiver
- Specify the number of data bits

Number of Data Bits

- The UCSRC data register controls the number of data bits in the transmission.

```
UCSRC = 0x86;          // select 8 data bits
```

The C Initialization

```
UBRRL = 38;          // SET THE BAUD RATE
UBRRH = 0;           // 9600 BAUD, W/ 6MHz

UCSRB = 0x18;        // ENABLE TX & RX

UCSRC = 0x86;        // SET 8 DATA BITS
```

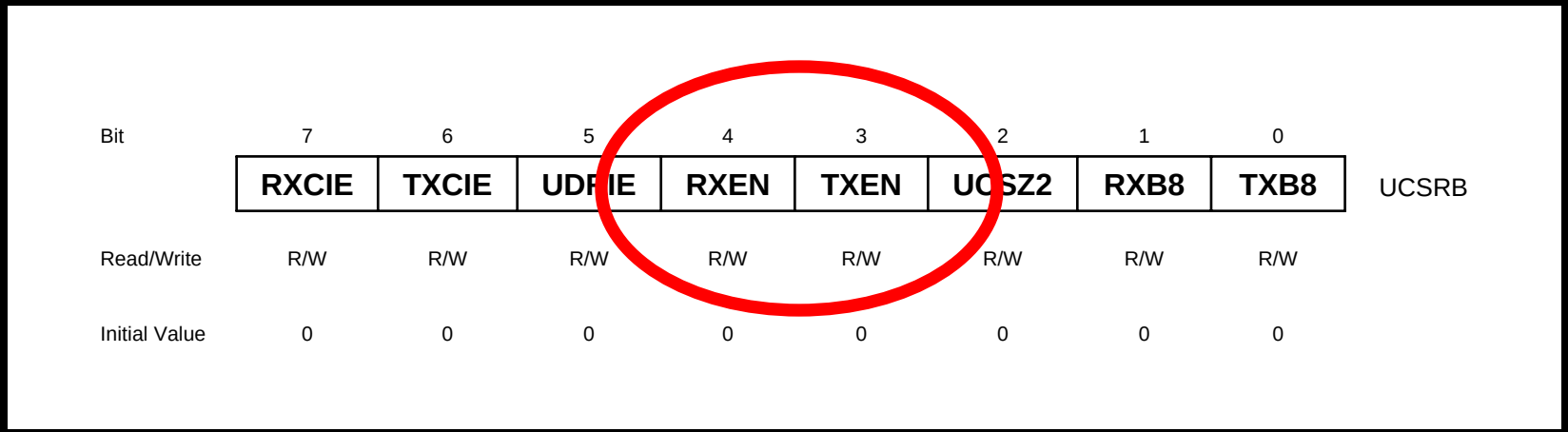
Questions?

Digital I/O vs. Serial Port

- Cannot be both (they share the same pins)
- When the USART is enabled, the internal circuitry “disconnects” the pin from the digital interface circuitry.

Disable TX and RX

- The BootLoader leaves the TX and RX enabled!
- Must always disable these bits located in the UCSRB register to use PORTD as general I/O



The C Code

- PORTD configured as an input with the internal pull-up resistors on.

```
DDRD = 0x00;        // configure PORTD for Input
PORTD = 0xFF;       // turn on the pull-up resistors
UCSRB = 0x00;       // disable the USART
```

printf()

- The printf() function allows us to output text with embedded numerical data also.

- for instance:

  printf(“The value is %d”, value);

  – assuming value was 25, then

The value is 25

Using STDIO

- The STDIO functions allow us to perform calculations and send the results out of the serial port (you will do this in Lab 2)

Using printf()

For example

```
degree_C = 5 * (degree_F – 32) / 9;

printf(“The temp in C is %d”, degree_C);
```

STDIO for Lab #2

printf("The temp in C is %d\n\r", degree_C);

- %d outputs the next argument as a signed decimal integer
- %u outputs the next argument as an unsigned decimal integer
- %x outputs the next argument as an unsigned hexadecimal integer using lower case letters
- %X outputs the next argument as an unsigned hexadecimal integer using upper case letters
- %c outputs the next argument as an ASCII character

What is ASCII Data?

- American Standard Code for Information Interchange (ASCII)
  - Method for encoding humanly readable characters
  - Uses 7 bits to represent
    - Upper and Lower case letters
    - Numbers and Symbols
    - Punctuation and Control Characters

MSB / LSB										
	Bit 6	0	0	0	0	1	1	1	1
	Bit 5	0	0	1	1	0	0	1	1
	Bit 4	0	1	0	1	0	1	0	1
Bit 3	Bit 2	Bit 1	Bit 0							
0	0	0	0	NUL	DLE	SP	0	@	P		p
0	0	0	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	STX	DC2		2	B	R	b	r
0	0	1	1	ETX	DC3	#	3	C	S	c	s
0	1	0	0	EOT	DC4	$	4	D	T	d	t
0	1	0	1	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	ACK	SYN	&	6	F	V	f	v
0	1	1	1	BEL	ETB		7	G	W	g	w
1	0	0	0	BS	CAN	(	8	H	X	h	x
1	0	0	1	HT	EM	)	9	I	Y	I	y
1	0	1	0	LF	SUB	*	:	J	Z	j	z
1	0	1	1	VT	ESC	+	;	K	[	k	{
1	1	0	0	FF	FS	,	<	L	\	l	|
1	1	0	1	CR	GS		=	M	]	m	}
1	1	1	0	SO	RS	.	>	N	^	n	~
1	1	1	1	SI	US		?	O	-	o	DEL

0x20

MSB									
Bit 6	0	0	0	0	1	1	1	1
Bit 5	0	0	1	1	0	0	1	1
Bit 4	0	1	0	1	0	1	0	1

LSB Bit 3	Bit 2	Bit 1	Bit 0								
0	0	0	0	NUL	DLE	SP	0	@	P		p
0	0	0	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	STX	DC2	"	2	B	R	b	r
0	0	1	1	ETX	DC3	#	3	C	S	c	s
0	1	0	0	EOT	DC4	$	4	D	T	d	t
0	1	0	1	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	ACK	SYN	&	6	F	V	f	v
0	1	1	1	BEL	ETB	'	7	G	W	g	w
1	0	0	0	BS	CAN	(	8	H	X	h	x
1	0	0	1	HT	EM	)	9	I	Y	I	y
1	0	1	0	LF	SUB	*	:	J	Z	j	z
1	0	1	1	VT	ESC	+	;	K	[	k	{
1	1	0	0	FF	FS	,	<	L	\	l	|
1	1	0	1	CR	GS	-	=	M	]	m	}
1	1	1	0	SO	RS	.	>	N	^	n	~
1	1	1	1	SI	US	/	?	O	-	o	DEL

0x30

MSB									
Bit 6	0	0	0	0	1	1	1	1
Bit 5	0	0	1	1	0	0	1	1
Bit 4	0	1	0	1	0	1	0	1

LSB									
Bit 3	Bit 2	Bit 1	Bit 0						
0	0	0	0	NUL	DLE	SP	0	@	P		p
0	0	0	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	STX	DC2		2	B	R	b	r
0	0	1	1	ETX	DC3	#	3	C	S	c	s
0	1	0	0	EOT	DC4	$	4	D	T	d	t
0	1	0	1	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	ACK	SYN	&	6	F	V	f	v
0	1	1	1	BEL	ETB		7	G	W	g	w
1	0	0	0	BS	CAN	(	8	H	X	h	x
1	0	0	1	HT	EM	)	9	I	Y	I	y
1	0	1	0	LF	SUB	*	:	J	Z	j	z
1	0	1	1	VT	ESC	+	;	K	[	k	{
1	1	0	0	FF	FS	,	<	L	\	l	|
1	1	0	1	CR	GS		=	M	]	m	}
1	1	1	0	SO	RS	.	>	N	^	n	~
1	1	1	1	SI	US		?	O	-	o	DEL

Lab 2

- Under Procedure 2
  - Display ASCII data on the Terminal
  - Set the toggles for a value between 0x30 and 0x70