3.10 Asynchronous Serial Communications

EE302 – Real time and embedded systems

1

Overview

- Aims
  - To introduce basic asynchronous communications handling
- Learning outcomes – you should be able to…
  - Explain serial communications, asynchronous communications, framing, parity, baud rate, symbol rate, bit rate, effective data rate
  - Describe (with particular attention to timing) the line coding of asynchronous serial communications
  - Explain how multi-byte messages/packets can be communicated
  - Show how asynchronous serial communications can be implemented in software using low level "bit banging"
  - Interpret an asynchronous serial specification and calculate the effective data rate and timing error tolerance

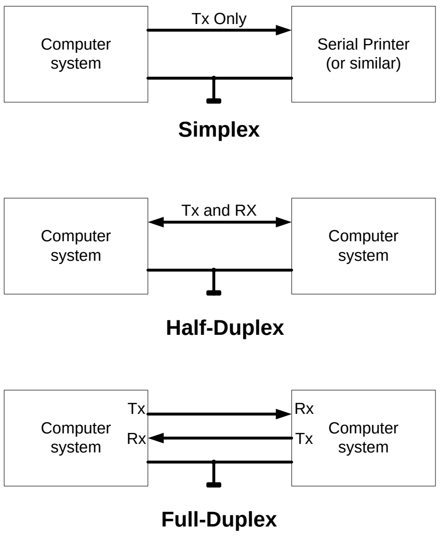
2

Introduction

- Why choose Serial over Parallel Communication?
  - Parallel cables can be bulky and expensive
  - Serial communication requires fewer wire
  - Serial data can be transmitted over longer distances
- Most modern high speed interfaces are serial
  - Examples: FireWire, USB, PCI Express

Async serial connection types

Async serial uses a point-to-point connection



Async serial properties

5

- Asynchronous serial communications is used to send/receive serial data **without explicitly synchronising** the clocks of transmitter and receiver
  - E.g. RS232 (and similar interfaces)
- The receiver must synchronise using only the transmitted signal and some prior agreed/configured timing
- Data is sent one byte at a time and each byte is packaged in an **asynchronous word** which includes start/stop bits (overhead) to aid synchronisation
- The bits in each asynchronous word (data + overhead) are mapped to **symbols** which are transmitted sequentially (serially)
  - The rate at which symbols are transmitted/received is called the **symbol rate** or **Baud rate** (e.g. 9600 baud)
- Async serial communication is point-to-point and usually full duplex (i.e. separate Transmit and Receive lines), but half duplex and simplex can also be used

10 November 2020

5

Error detection

6

- If desired, async serial can use a parity bit so that single bit errors can be detected by receiver
  - The transmitter calculates a parity bit for the data and adds the parity bit to the asynchronous word as another overhead bit
  - Receiver checks the parity of the received async word against the parity bit
- Reminder: how it works
  - Parity counts the number of 1’s in the data bits to be transmitted
  - Transmitter and receiver can agree to use even or odd parity
    - EVEN: parity bit is 1 if number of 1’s in data is odd (so number of 1’s with parity bit is now even)
    - ODD: parity bit is 1 if number of 1’s in data is even (so number of 1’s with parity bit is now odd)

10 November 2020

6

Async serial line coding 1

- Data is transmitted as asynchronous words
  - These are clearly delimited by stop and start bits
  - The start bit is used by receiver to synchronise and read subsequent bits correctly
- An asynchronous word is used to transmit a single byte (8 bits) of data
- Between words (i.e. after the stop bit of one word and before the start bit of another), the line will be at its idle level

Async serial line coding 2

- An asynchronous word consists of the following parts ordered as listed
  - A start bit
  - The 8 bits of data to be transmitted, **least significant bit (LSb) first**
  - Optionally, a parity bit
    - The use of parity in an async communication system can specified as None (don't use parity), Even, or Odd.
  - Finally one or two stop bits (depending on configuration)
    - The number of stop bits must be specified for any async serial communication system
    - Two stop bits gives the receiver more time to process received data before the next asynchronous word could possibly arrive

Async serial line coding 3

- For transmission the bits are mapped to symbols
- In async serial, just two voltage levels are used, so the alphabet size is 2 and hence the number of bits per symbol is 1.
  - The two symbols are voltage levels referred to as Mark (high) and Space (low)
- When the line is idle (between asynchronous words), it is at the Mark level
- The rate at which symbols are presented is called the symbol rate (or baud rate)
  - The symbol period is 1/symbolRate

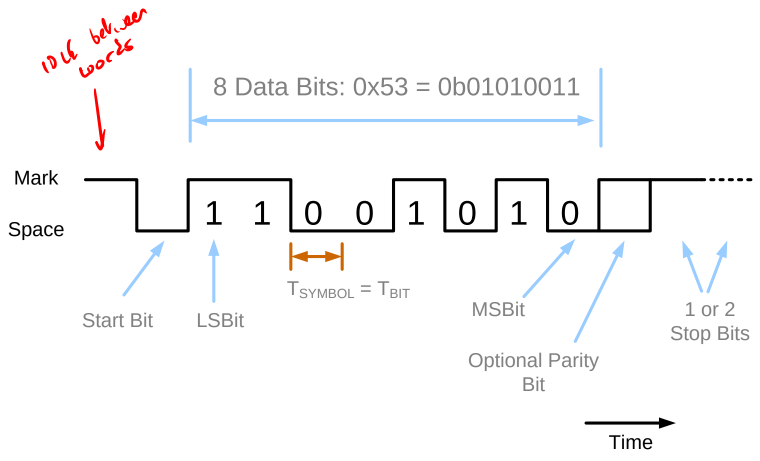
NOTE

- *A symbol is a single signalling level or pulse communicated over a transmission medium*
- *The number of bits encoded by a symbol depends on the number of unique symbols in the "alphabet" for that communication scheme*
  - *An alphabet size of 2^b implies each symbol encodes b bits*
  - *At each symbol time, one symbol from the alphabet is chosen (dictated by the bits you want to encode) to transmit*
  - *E.g. suppose you can transmit one of 8 voltage levels over a line. The size of the alphabet = 8 = 2^3. Therefore, each symbol encodes 3 bits. To transmit the bit pattern 101, set the line to voltage level 5.*

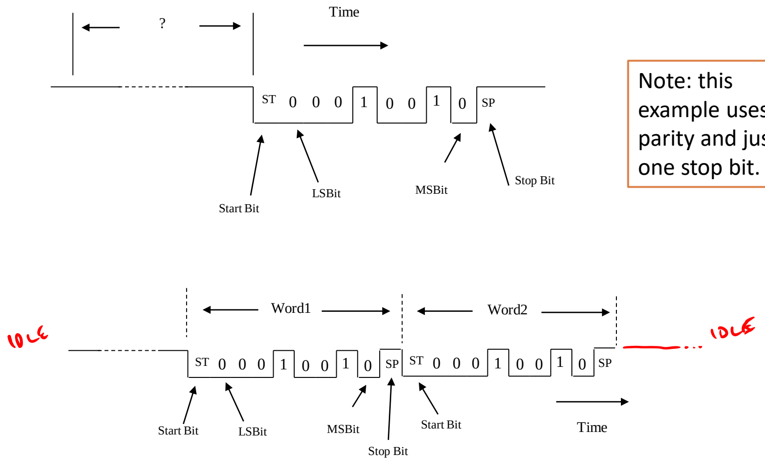
Asynchronous serial specification

- Asynchronous serial communications is specified using the following format:
  - baudRate, numDataBitsPerWord, parity, numStopBits
- Examples:
  - 9600, 8, N, 1
    - 9600 symbols/sec, 8 data bits per word, no parity, 1 stop bit
  - 28800, 8, E, 2
    - 28800 symbols/sec, 8 data bits per word, even parity, 2 stop bits
  - 4800, 8, O, 1
    - 4800 symbols/sec, 8 data bits per word, odd parity, 1 stop bit

Line coding example – single word



Transmitting multiple words



Note: this example uses no parity and just one stop bit.

Bit banging

- The most basic technique for implementing async serial communications from an MCU is called "bit banging"
- In this technique
  - the Tx and Rx communication lines are connected directly to digital I/O ports
  - The MCU drives the Tx line high and low at the correct times to transmit a well formed asynchronous word
  - Similarly the MCU must check (poll) the Rx line at regular intervals to detect the start bit and hence the subsequent bits in an asynchronous word
- How can this be done in practice…

"Bit banging" transmitter (simple version)

```
Main…

// Trivial app: send a dot once every second while running
loop()
  transmitByte('.')
  delay(ONE_SECOND)

setup()
  configure a pin of I/O port for output and call this pin Tx
  set Tx = MARK // initially at the idle or stop level

// blocking call: transmit complete Byte before returning
transmitByteCompletely(val)
  // start bit, 8 data bits (LSbit first), no parity, stop bit
  set Tx = SPACE // start bit
  delay(SYMBOL_PERIOD)
  if bit0 of val is 1, set Tx = MARK, else set Tx = SPACE
  delay(SYMBOL_PERIOD)
  …
  if bit7 of val is 1, set Tx = MARK, else set Tx = SPACE
  delay(SYMBOL_PERIOD)
  set Tx = MARK // stop bit
  delay(SYMBOL_PERIOD)
```

"Bit banging" transmitter using a loop

```
Main…

// Trivial app: send a dot once every second while running
loop()
  transmitByte('.')
  delay(ONE_SECOND)

setup()
  configure a pin of I/O port for output and call this pin Tx
  set Tx = MARK // initially at the idle or stop level

// blocking call: transmit complete Byte before returning
transmitByteCompletely(val)
  // start bit, 8 data bits (LSbit first), no parity, stop bit
  set Tx = SPACE // start bit
  delay(SYMBOL_PERIOD)
  for i=0 to 7 // there are 8 bits in the byte, bit 0 is the LSB
    if bit i of val is 1, set Tx = MARK, else set Tx = SPACE
    delay(SYMBOL_PERIOD - LOOP_OVERHEAD)
  Tx = MARK // stop bit
  delay(SYMBOL_PERIOD)
```

Contd. – additional details

```
// to extract the bit value at bitIndex in the value x
bitValue = (x RSHIFT by bitIndex) BitwiseAND 0b00000001
---------------------------------------------------------------

// the equivalent C code fragment is
bitValue = (x >> bitIndex) & 0x1;
---------------------------------------------------------------

// Therefore the C code to examine each bit in a value, x,
// and take some action based on the bit values is…
for (bitIndex = 0; bitIndex < 8; bitIndex++) {
    // what is the value of bit numbered bitIndex in x
    bitValue = (x >> bitIndex) & 0x1;

    if (bitValue) // if bitValue is a 1, do something
      …
}
```

Self test question

17

Q. Modify the bit banging transmitter so that instead of blocking the superloop until the entire byte has been sent, it just sends a single bit each time it is called.

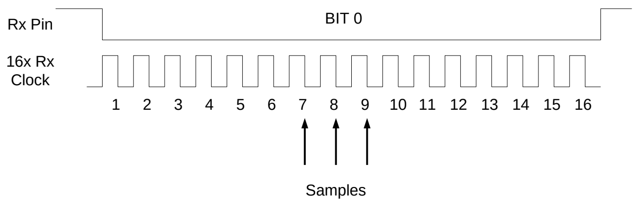
- *Hints: The superloop period should be changed to the symbol time and the transmit function will need one or more static variables to keep track of what stage it is at in the transmission (i.e. at the start bit, at some bit of the data, etc.)*

10 November 2020

17

Asychronous - Reception

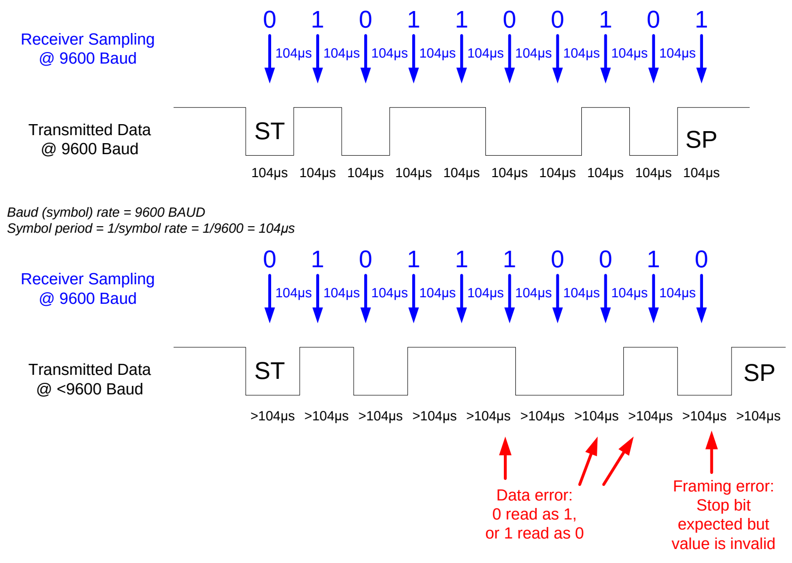
18



10 November 2020

18

Asynchronous – Tx/Rx timing tolerence


Receiver Sampling @ 9600 Baud
0 1 0 1 1 0 0 1 0 1
104µs 104µs 104µs 104µs 104µs 104µs 104µs 104µs 104µs
Transmitted Data @ 9600 Baud
ST
SP
104µs 104µs 104µs 104µs 104µs 104µs 104µs 104µs 104µs 104µs
Baud (symbol) rate = 9600 BAUD
Symbol period = 1/symbol rate = 1/9600 = 104µs
Receiver Sampling @ 9600 Baud
0 1 0 1 1 1 0 0 1 0
104µs 104µs 104µs 104µs 104µs 104µs 104µs 104µs 104µs
Transmitted Data @ <9600 Baud
ST
SP
>104µs >104µs >104µs >104µs >104µs >104µs >104µs >104µs >104µs >104µs
Data error: 0 read as 1, or 1 read as 0
Framing error: Stop bit expected but value is invalid

Tx/Rx timing tolerence contd.

- Assumptions
  - we assume that the receiver can identify the exact middle of the first symbol period (i.e. the start bit) in an asynchronous word – this is a best case scenario and we should ideally allow more margin in case this cannot be achieved
  - If we accumulate timing error greater or equal to ±0.5 symbol period, over the duration of the asynchronous word, then the receiver will misidentify a symbol
- Therefore the maximum timing error we can tolerate is:

$$maxErrorFraction = \frac{0.5}{totalBitsPerAsyncWord}$$

$$maxErrorPercent = 100 \times maxErrorFraction$$

Self test questions

Q. If the async serial specification is 9600,8,N,1, then what is the max timing error fraction we can tolerate?

Q. If the async serial specification is 2400,8,N,1, then what is the max timing in microseconds that we can tolerate on each symbol?

Q. If the async serial specification is 4800,8,E,2, then what is the max timing error percent we can tolerate?

Effective data rate

- Baud rate = symbol rate = symbols per second
  - For async serial we have 1 bit per symbol, so raw bit rate (in bits per second) is equal to the symbol rate in this case
- Effective data rate

$$effectiveDataRate = bitRate \times \frac{dataBitsPerAsyncWord}{totalBitsPerAsyncWord}$$

  - Express in bps or kbps
- The number of data bits per word is essentially always 8
- The total bits per async word includes all overhead and data bits and depends on the spec (was parity used? One or two stop bits?)

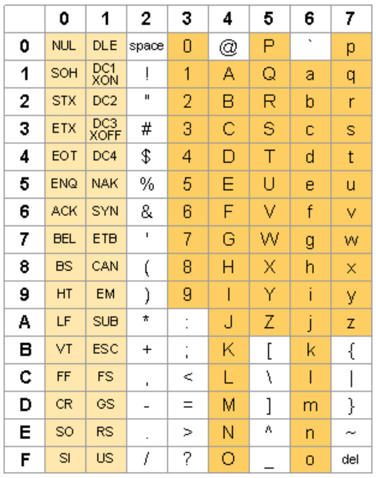
Self test questions

Q. If the async serial specification is 9600,8,N,1, then what is the effective data rate?

Q. If the async serial specification is 4800,8,E,2, then what is the effective data rate?

Data format

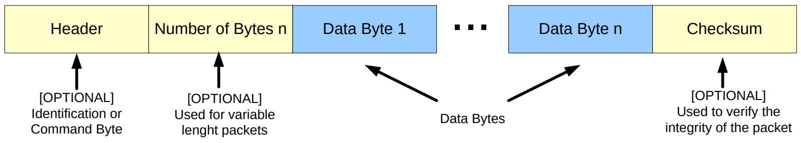
- Originally async serial used to communicate between main frames and input terminals and printers
  - Hyperterminal on Windows simulates old input terminals
- Standard codes were created to allow text interoperability and communication
  - E.g. American Standard Code for Information Interchange (ASCII)
- In ASCII Just 7 bits required to represent most common characters in American English
  - Codes 0x00..0x1F are control codes which aid printing (e.g. line feed, LF) and communications (e.g. ACK)
- E.g. “See\n” = 0x53, 0x65, 0x65, 0x0D
- NOTE: async serial can also transmit binary data – ASCII is only used for text interchange



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

Ref. http://ascii-table.com/

Asynchronous – Multi-Byte Packet


Header
Number of Bytes n
Data Byte 1
...
Data Byte n
Checksum
[OPTIONAL] Identification or Command Byte
[OPTIONAL] Used for variable lenght packets
Data Bytes
[OPTIONAL] Used to verify the integrity of the packet

Example:
GPS data is communicated over async serial in accordance with NMEA.
Serial configuration is 4800,8,N,1.
GPS data is transmitted as "sentences" of ASCII coded text, e.g.

```
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47
$GPGSV,2,1,08,01,40,083,46,02,17,308,41,12,07,344,39,14,22,228,45*75
```

Each sentence starts with '$' and ends with CR+LF and has at most 80 visible characters
The first text after the $ (e.g. GPGGA) is a header which identifies the sentence type
Sentences end with a checksum of the form *xx where xx are two hex digits

Example/Self test questions

Explain how to sample and receive data

Explain use of parity bit

Explain framing error

Draw async bit pattern for data byte 0x43, assuming configuration is: 9600,8,E,1

A GPS using the NMEA standard transmits serial data using the configuration: 4800,8,N,1

- *If the average message size is 60 data bytes, how many messages per second can be transmitted?*

Write the pseudocode for a bit banging transmitter which transmits a character at a time at 4800 baud. Be specific about any timing/delay values used.

27