Serial Communication

Characters and Strings

Compute

- **Transistors and gates**
- **Processor**

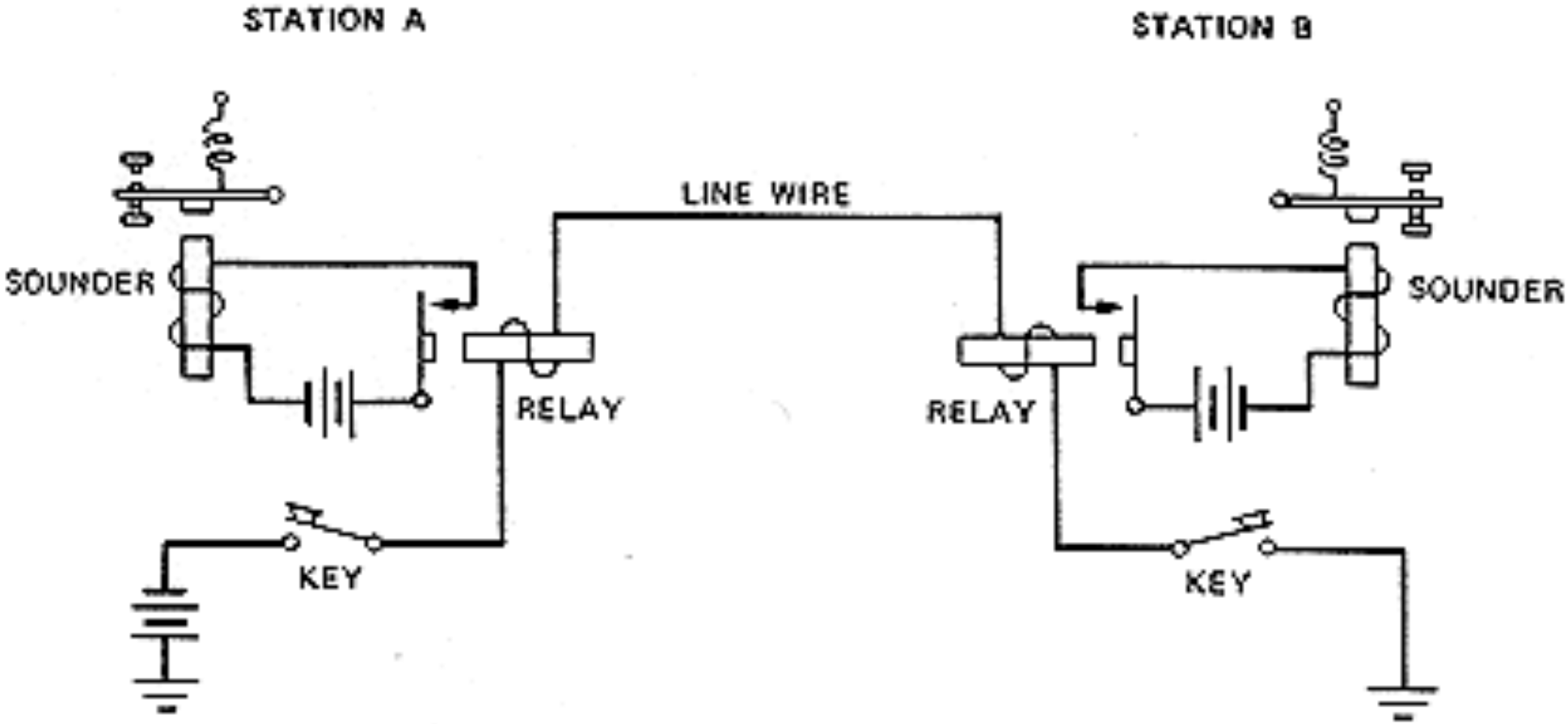
Control

- **Peripherals: sensors and actuators**

Communicate

- **Wires**
- **Memory**

SIMPLEX TELEGRAPH

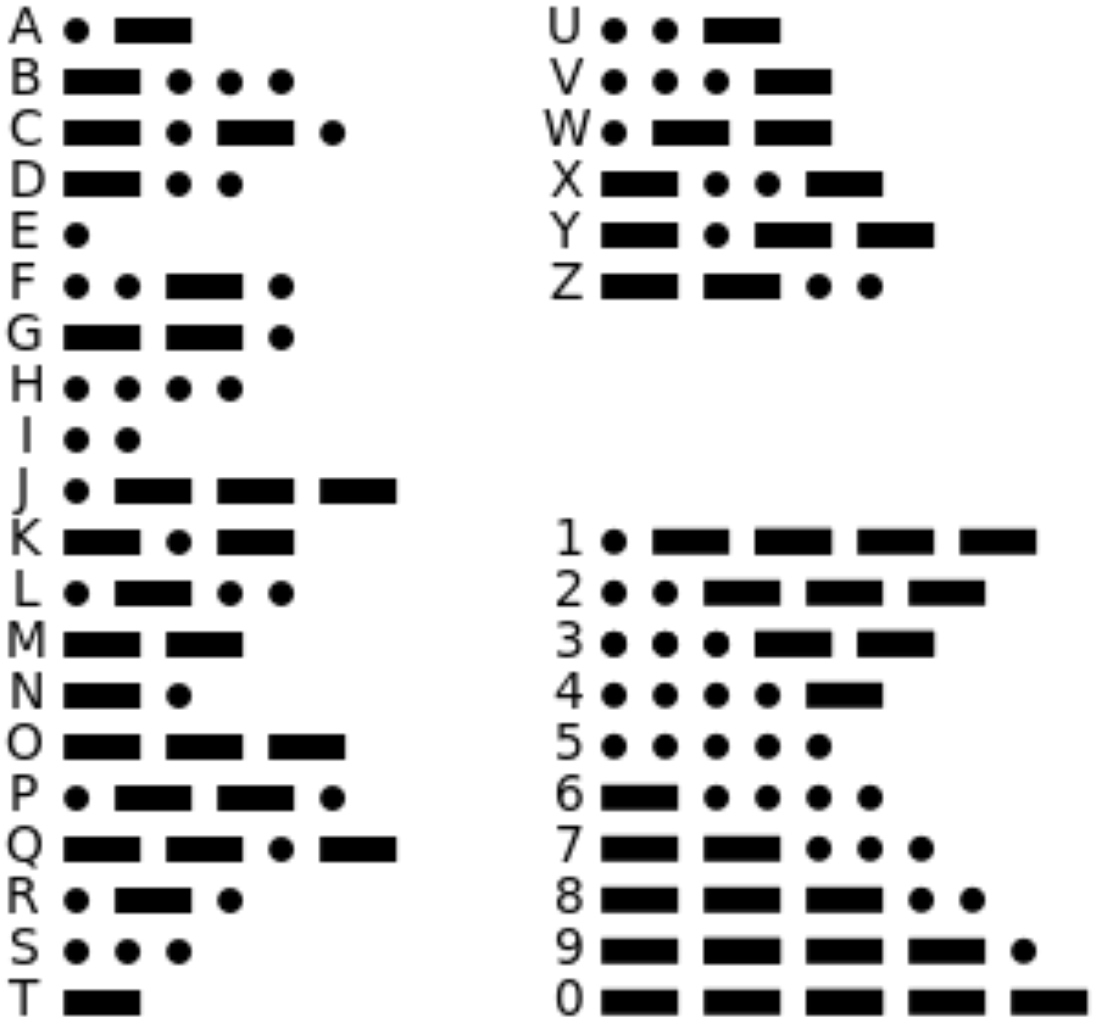


Elementary neutral telegraph circuit.

http://people.seas.harvard.edu/~jones/cscie129/nu_lectures/lecture5/elecmag_tel/morse_tel.html

International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

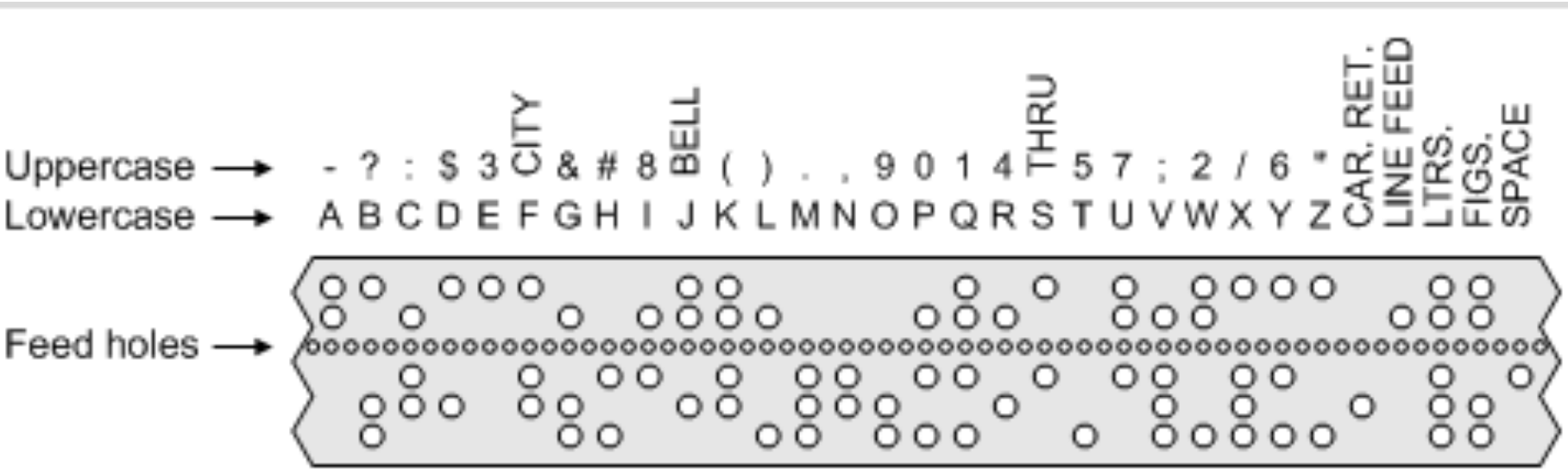


sos.c

Teletype

http://www.smecc.org/police_-__fire_-_civil_defense_communications.htm

Baudot Code



Paper tape showing the five-bit Baudot Code

Baud: Number of symbols per second

https://savzen.wordpress.com/tag/baudot/

```
% ascii
   2 3 4 5 6 7
 ------------
0:   0 @ P ' p
1: ! 1 A Q a q
2: " 2 B R b r
3: # 3 C S c s
4: $ 4 D T d t
5: % 5 E U e u
6: & 6 F V f v
7: ' 7 G W g w
8: ( 8 H X h x
9: ) 9 I Y i y
A: * : J Z j z
B: + ; K [ k {
C: , < L \ l |
D: - = M ] m }
E: . > N ^ n ~
F: / ? O _ o DEL
```

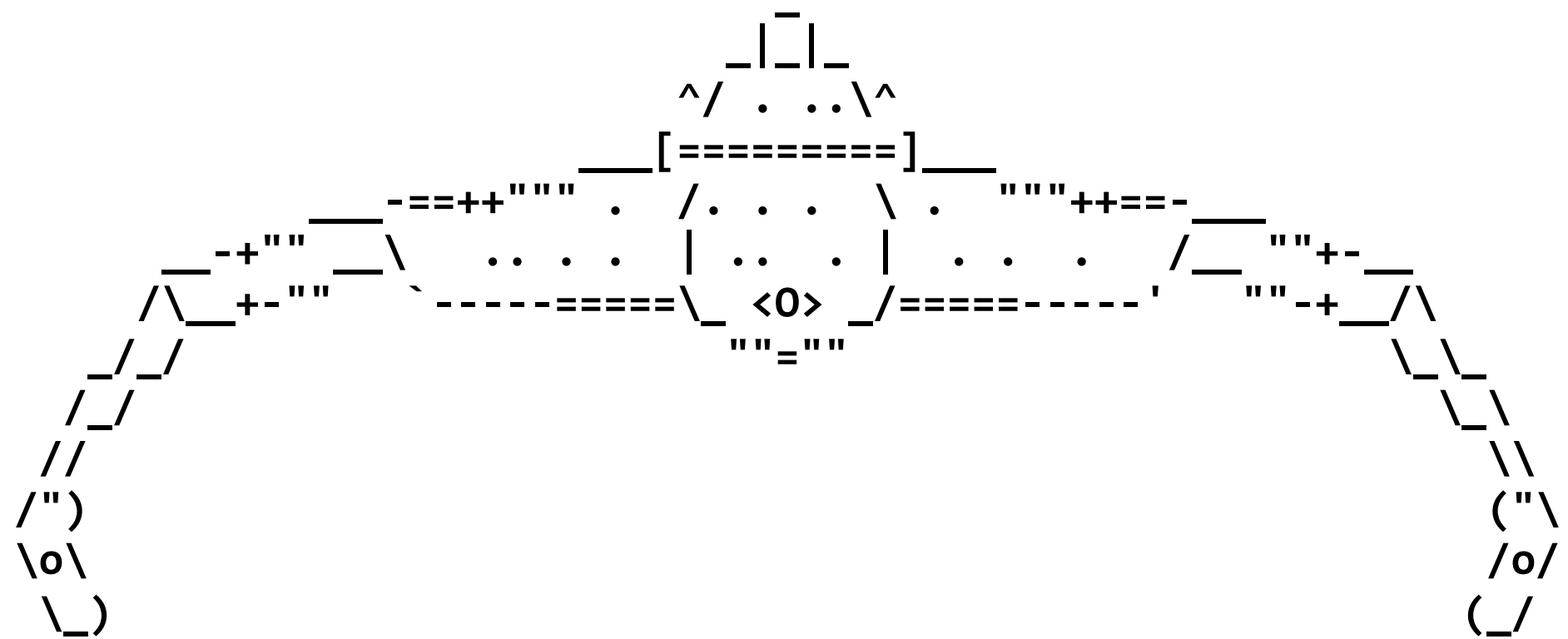
0x30 = ‘0’

0x31 = ‘1’

0x40 = ‘@‘

0x41 = ‘A’

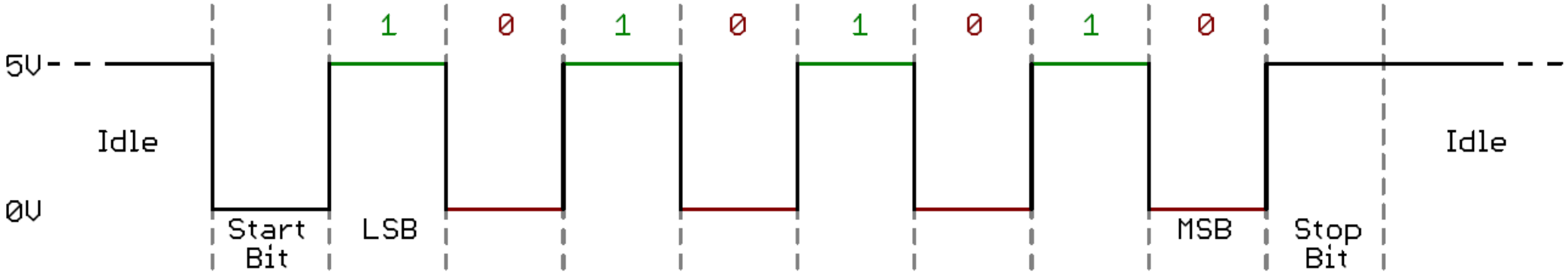
Klingon D-7M Cruiser (Gym Z. Quirk aka Taki Kogoma)



http://startrekasciiart.blogspot.com/

Asynchronous Serial Communication

(implicit clock)

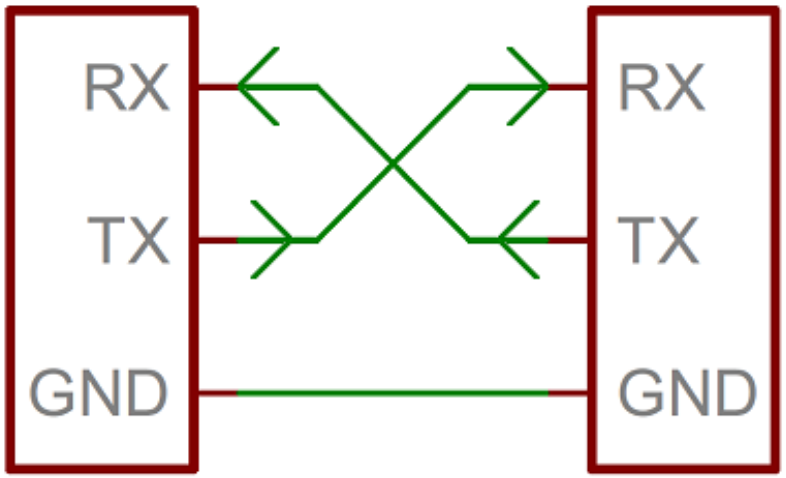


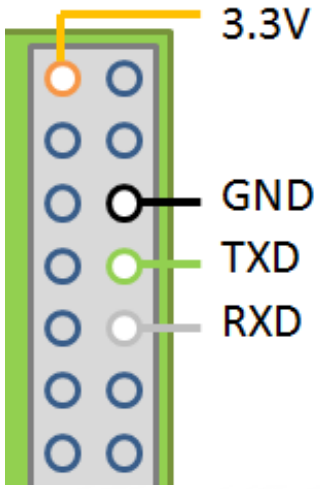
8-bits, no parity, 1 stop bit (8N1)

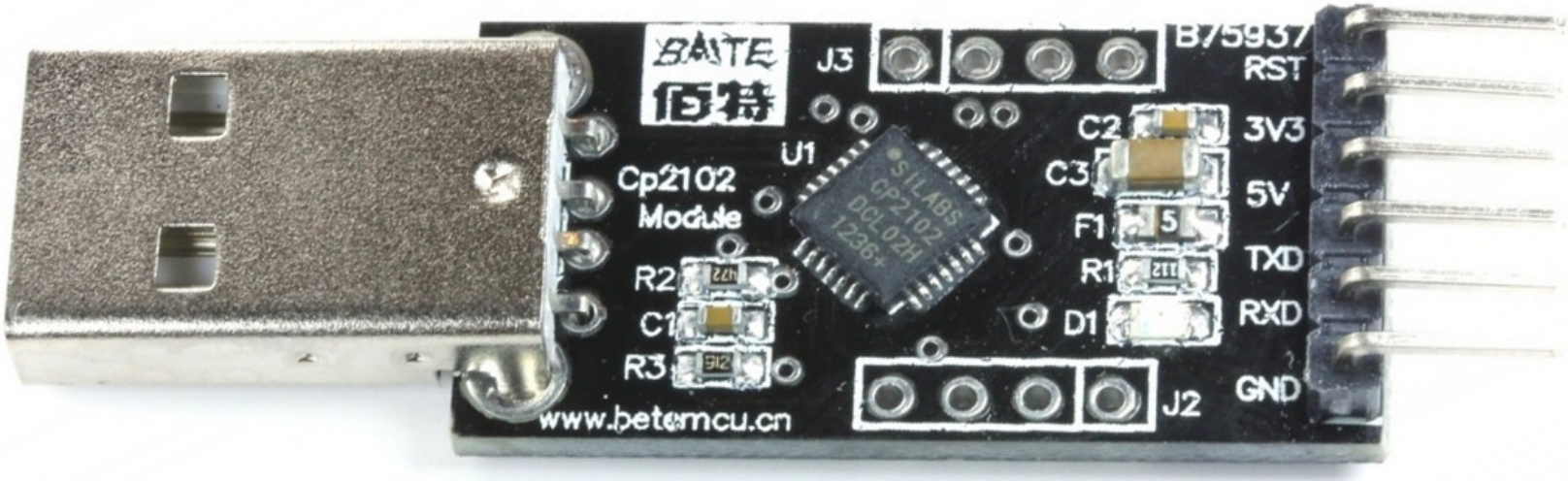
9600 baud = 9600 bits/sec

(1000000 usecs)/9600 ~ 104 usec/bit

serial.c



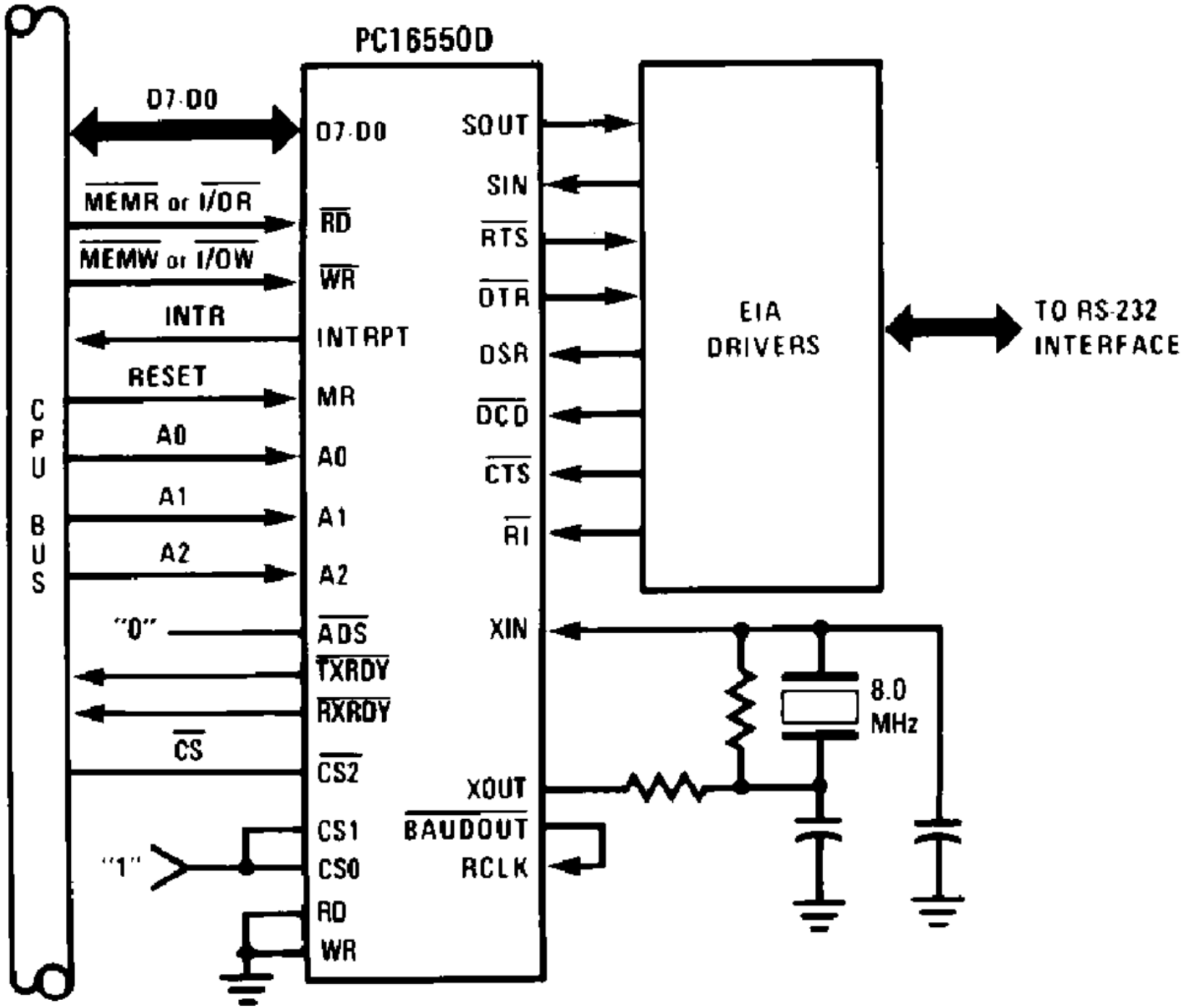




https://learn.sparkfun.com/tutorials/serial-communication

`uart.c`

Universal Asynchronous Receiver-Transmitter


PC16550D
D7-D0
D7-D0
SOUT
SIN
RTS
DTR
DSR
DCD
CTS
RI
XIN
XOUT
BAUDOUT
RCLK
MEMR or I/OR
RD
MEMW or I/OW
WR
INTR
INTRPT
RESET
MR
A0
A0
A1
A1
A2
A2
"0"
ADS
TXRDY
RXRDY
CS
CS2
CS1
"1"
CS0
RD
WR
CPU BUS
EIA DRIVERS
TO RS-232 INTERFACE
8.0 MHz

```
// BCM2835-ARM-Peripherals.pdf
// Sec 2: Mini-UART, SPI0, SPI1, pp 8-19
struct UART {
    int data; // I/O Data
    int ier;  // Interupt enable
    int iir;  // Interupt identify/fifo
    int lcr;  // line control register
    int mcr;  // modem control register
    int lsr;  // line status
    int msr;  // modem status
    int scratch;
    int cntl; // control register
    int stat; // status register
    int baud; // baud rate register
} ;
```

```
// Strings

char *s1 = “hello, world\n”;

char s2[] = {'h','e','l','l','o',' ',
             'w','o','r','l','d','\n','\0'};

// Is s1 the same as s2?
```

```
// Strings
char *s = “hello, world\n”;

// What is strlen(s)?

// How many bytes is “hello, world\n”?
```

String Functions

strcat(s1,s2)	Concatenate s2 to s1
strncat(s1,s2,n)	Concatenate at most n characters of s2 to s1
strcpy(s1,s2)	Copy s2 to s1; Note the direction of the copy!
strncpy(s1,s2,n)	Copy first n characters of s2 to s1
strlen(s)	Return length of string s, not counting '\0'
strcmp(s1,s2)	Compare s1 with s2; Return integer less than zero, equal to zero, or greater than zero
strncmp(s1,s2,n)	Compare only the first n characters of s1 and s2
strchr(s,c)	Return a pointer to first occurrence of character c in string s; return NULL if not found
strrchr(s,c)	Return a pointer to last occurrence of character c in string s; return NULL if not found
strstr(s1,s2)	Return a pointer to the first occurrence of string s1 in string s2; return NULL if not found
strstr(s1,s2)	Return a pointer to the first occurence of string s1 in string s2; return zero if not found

```
int strlen(const char *str)
{
  const char *s;

  for (s = str; *s; ++s);

  return(s - str);
}

// What is strlen("\n")?
// What is strlen('\n')?
// What is strlen(NULL)?
```

```
/* ANSI sez:
 *   The `strcpy' function copies the string pointed to by `s2' (including
 *   the terminating null character) into the array pointed to by `s1'.
 *   If copying takes place between objects that overlap, the behavior
 *   is undefined.
 *   The `strcpy' function returns the value of `s1'.  [4.11.2.3]
 */
char *
strcpy(char *s1, const char *s2)
{
    char *s = s1;
    while ((*s++ = *s2++) != 0)
        ;
    return s1;
}
```

```
// Strings
char *s = “hello, world\n”;

char scopy[10];

strcpy(scopy, s);

// Problem?
```

```
/* ANSI sez:
 *   The `strncpy' function copies not more than `n' characters (characters
 *   that follow a null character are not copied) from the array pointed to
 *   by `s2' to the array pointed to by `s1'.  If copying takes place between
 *   objects that overlap, the behavior is undefined.
 *   If the array pointed to by `s2' is a string that is shorter than `n'
 *   characters, null characters are appended to the copy in the array
 *   pointed to by `s1', until `n' characters in all have been written.
 *   The `strncpy' function returns the value of `s1'.  [4.11.2.4]
 */
char *
strncpy(char *s1, const char *s2, int n)
{
    char *s = s1;
    while (n > 0 && *s2 != '\0') {
       *s++ = *s2++;
       --n;
    }
    while (n > 0) {
       *s++ = '\0';
       --n;
    }
    return s1;
}
```

```
// Strings
char *s = “hello, world\n”;

char scopy[10];

strncpy(scopy, s, 10);

// What will be in scopy?

strncpy(scopy, s, strlen(s));

// What will be in scopy?
```

```
// Strings
char *s = “hello, world\n”;

s[5] = ‘\0’;
puts(s);
```

Read:

The most expensive 1-byte mistake,

Did Ken, Dennis, and Brian choose wrong with NUL-terminated text strings?

Poul-Henning Kamp

http://queue.acm.org/detail.cfm?id=2010365

```
int *
intcpy(int *i1, const int i2)
{
    *i1 = i2;
    return i1;
}
```

```
printf(const char *format, …);
scanf(const char *format, …);

printf(“%d, %d”, 1, 2);
printf(“%d, %d, %d”, 1, 2, 3);
printf(“%d, %d, %d”, 1, 2);

int i1, i2;
sscanf(“1, 2”, “%d, %d”, &i1, &i2);

// Read documentation about
// how to handle functions
// with variable number of arguments
```