Computer Club

10-27-2015

Arduino Update

- Project 1 - Blinking LED

  Using digital pin 13 we turned an LED on and off.

  pinMode - Sets up a pin as input or output

  pinMode(pin#, OUTPUT) || pinMode(pin#, INPUT)

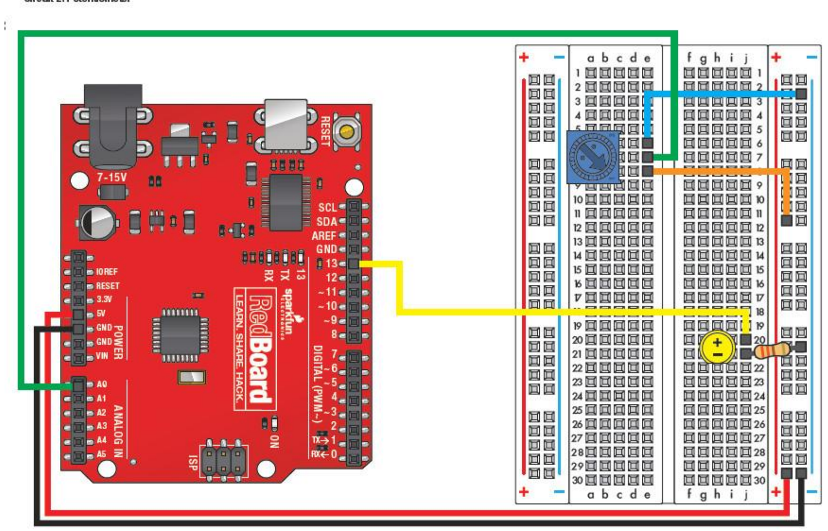
  digitalWrite(pinNumber,VALUE)

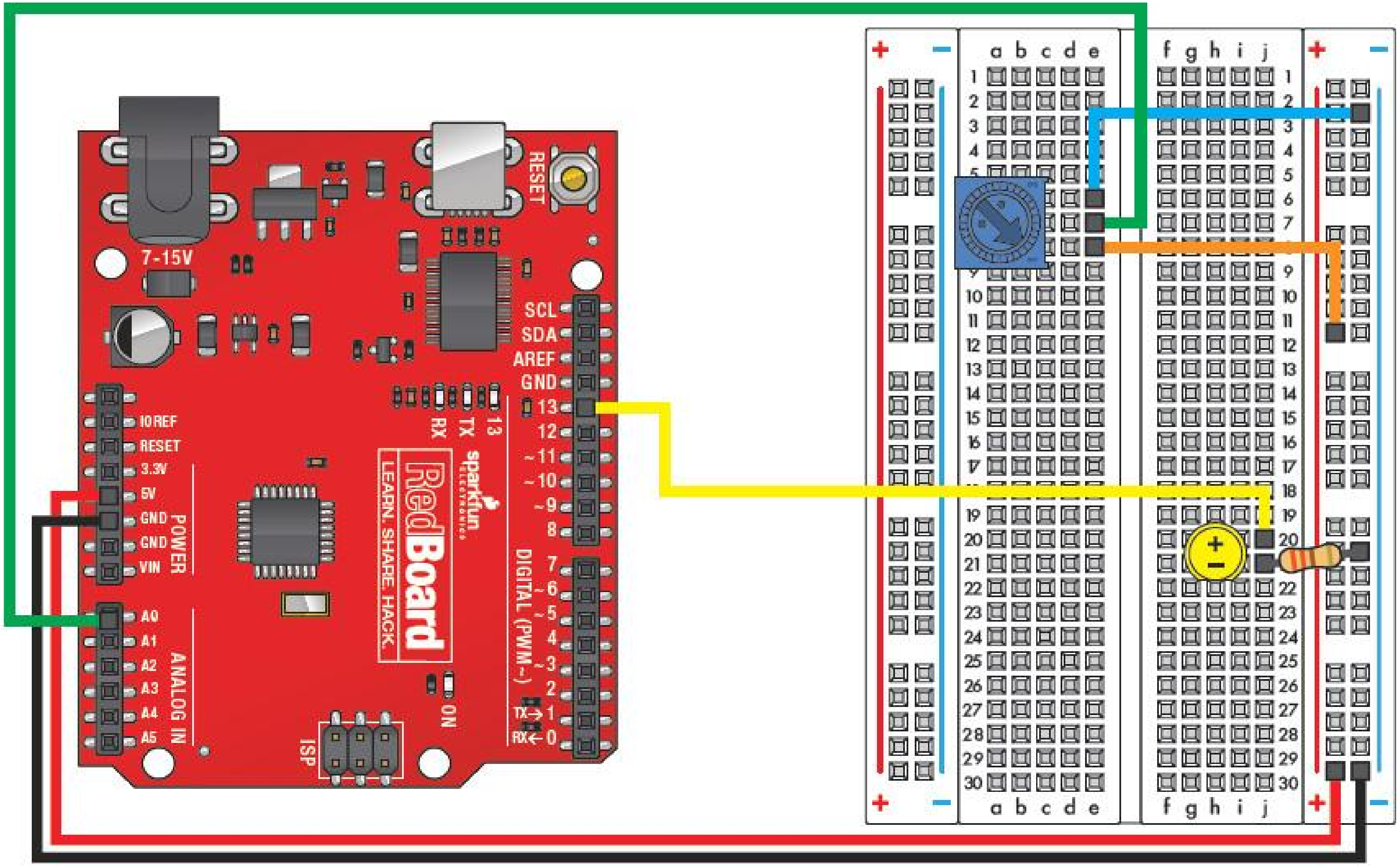
  VALUE = HIGH || LOW

  delay(# of Milliseconds)

Arduino Update

- Potentiometer
  - Used a Potentiometer to control the speed at which the LED on PIN 13 Blinks
    - Voltage Divider
  - PIN 13 - Digital Pin
    - HIGH = 5 VOLTS
    - LOW = 0 VOLTS
  - PIN A0 = Analog Input
- By adjusting the Potentiometer we raised and lowered the voltage to pin A0.



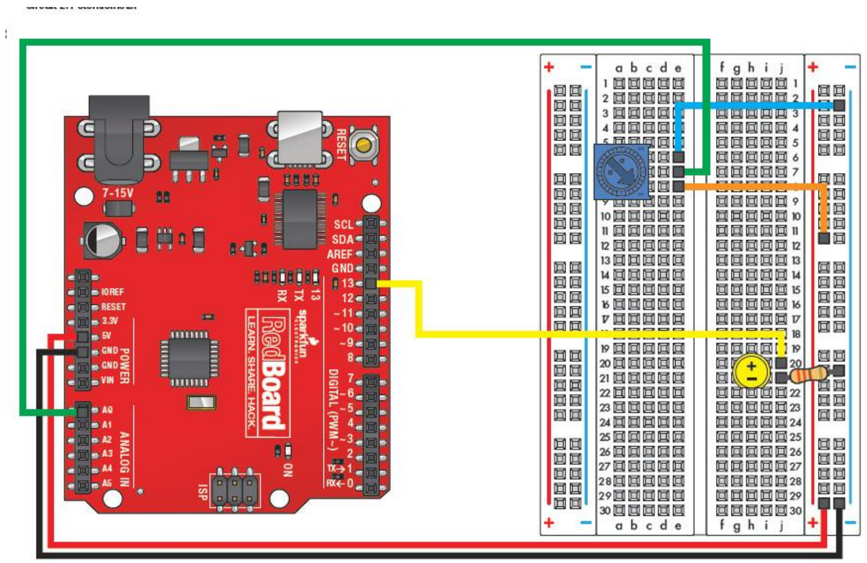

7-15V
RESET
SCL
SDA
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
RX
TX
13
IOREF
RESET
3.3V
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
ISP
ON
RedBoard
LEARN. SHARE. HACK.
sparkfun electronics
a b c d e
f g h i j
+
−

The Code...

```
int sensorPin = 0;    // The potentiometer is connected to
int ledPin = 13;      // The LED is connected to digital pin 13

void setup() // this function runs once when the sketch starts up
{
  pinMode(ledPin, OUTPUT);
}


void loop() // this function runs repeatedly after setup() finishes
{

  int sensorValue;
  sensorValue = analogRead(sensorPin);
  digitalWrite(ledPin, HIGH);     // Turn the LED on
  delay(sensorValue);             // Pause for sensorValue
  digitalWrite(ledPin, LOW);      // Turn the LED off
  delay(sensorValue);             // Pause for sensorValue
}
```

Globals….

```
int sensorPin = 0;     // The
potentiometer is connected to

int ledPin = 13;       // The LED is
connected to digital pin 13
```



Global variables are variables that are accessible to all of the functions in a program (accessible globally)

Be careful with global variables as they could be unintentionally modified.

Use const keyword to make them “Constant” - Unable to be modified.

setup

```
void setup(){

  pinMode(ledPin, OUTPUT);

}
```

- When programming the Arduino, setup is the function that can initialize values and set pin modes.
- It is called only once.

loop

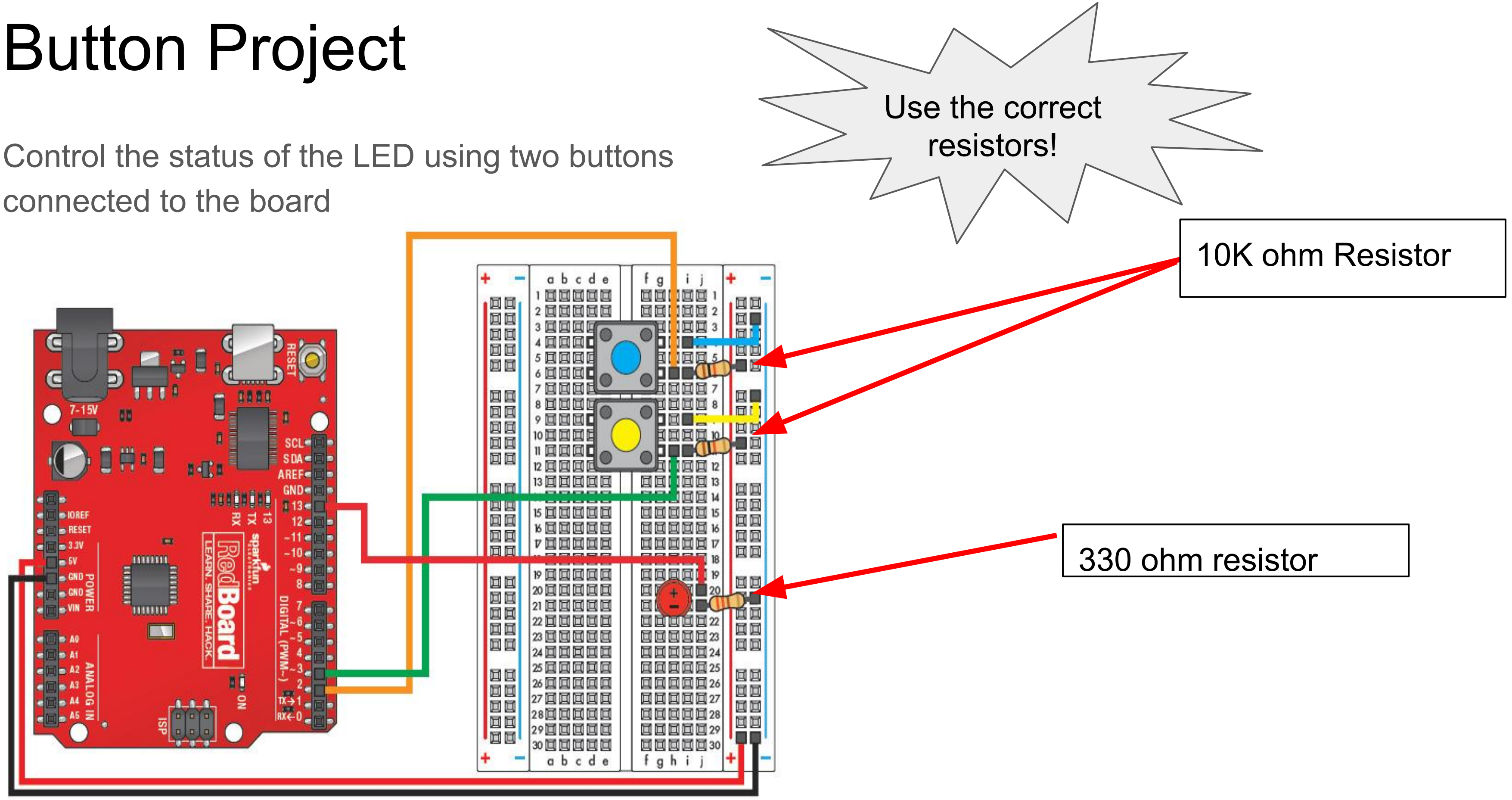
```
void loop(){

  int sensorValue;

  sensorValue =analogRead(sensorPin);

  digitalWrite(ledPin, HIGH);

  delay(sensorValue);

  digitalWrite(ledPin, LOW);

  delay(sensorValue);

}
```

Loop is called by the microprocessor right after setup is completed and is called until the microprocessor is turned off or reset.

```
do{
     loop();
}while(TRUE);
```

Button Project

Control the status of the LED using two buttons connected to the board



The code...

```
const int button1Pin = 2;  // pushbutton 1 pin
const int button2Pin = 3;  // pushbutton 2 pin
const int ledPin =  13;    // LED pin

void setup(){

     // Set up the pushbutton pins to be an input:

     pinMode(button1Pin, INPUT);
     pinMode(button2Pin, INPUT);


     // Set up the LED pin to be an output:

     pinMode(ledPin, OUTPUT);

}
```

The code….(continued!)

```
void loop()

{

  int button1State, button2State;

  button1State = digitalRead(button1Pin);

  button2State = digitalRead(button2Pin);
```

The code….(continued!)

```
  if (((button1State == LOW) || (button2State == LOW))
&& ! ((button1State == LOW) && (button2State == LOW))){

    digitalWrite(ledPin, HIGH);  // turn the LED on

  }else{

    digitalWrite(ledPin, LOW);  // turn the LED off

  }
```

if / else

==	Equivalence (equals to)
!=	Not equal to
&&	And
||	Or
!	Not

if / else (break it down…)

```
if (((button1State==LOW) || (button2State==LOW))  && !  ((button1State==LOW) && (button2State==LOW)))
```

Where are we going?

- Problem solving
    - Understanding conditions is key
    - Truth Tables:

&& (and)	T	F
T	T	F
F	F	F

|| (or)	T	F
T	T	T
F	T	F

- Guess what number the Arduino is thinking
    - Work in teams of 3 to 5 people
    - Red and Green LED (Red - Wrong, Green - Correct)
    - 4 Buttons (1 through 4)
    - Press a button the guess the number. LED will light if it is right or wrong.

Bonus 1:

What Does this say???

```
lqef ef s fehzwa fgpflelglevt oezqak. dwsj ef tvm_jv_fejt_gz_sti_iv_lqa_kasw_oqswwatjaf
```

Bonus 2

```
x = raw_input("enter the password ");
y = "";
for c in x:
      y += chr(ord(c) ^ ord(" "));
if y == "nottherightcase":
      print "congratz the flag is "+y;
else:
      print "nope";
```

HINT 1:
^ is a bitwise exclusive or

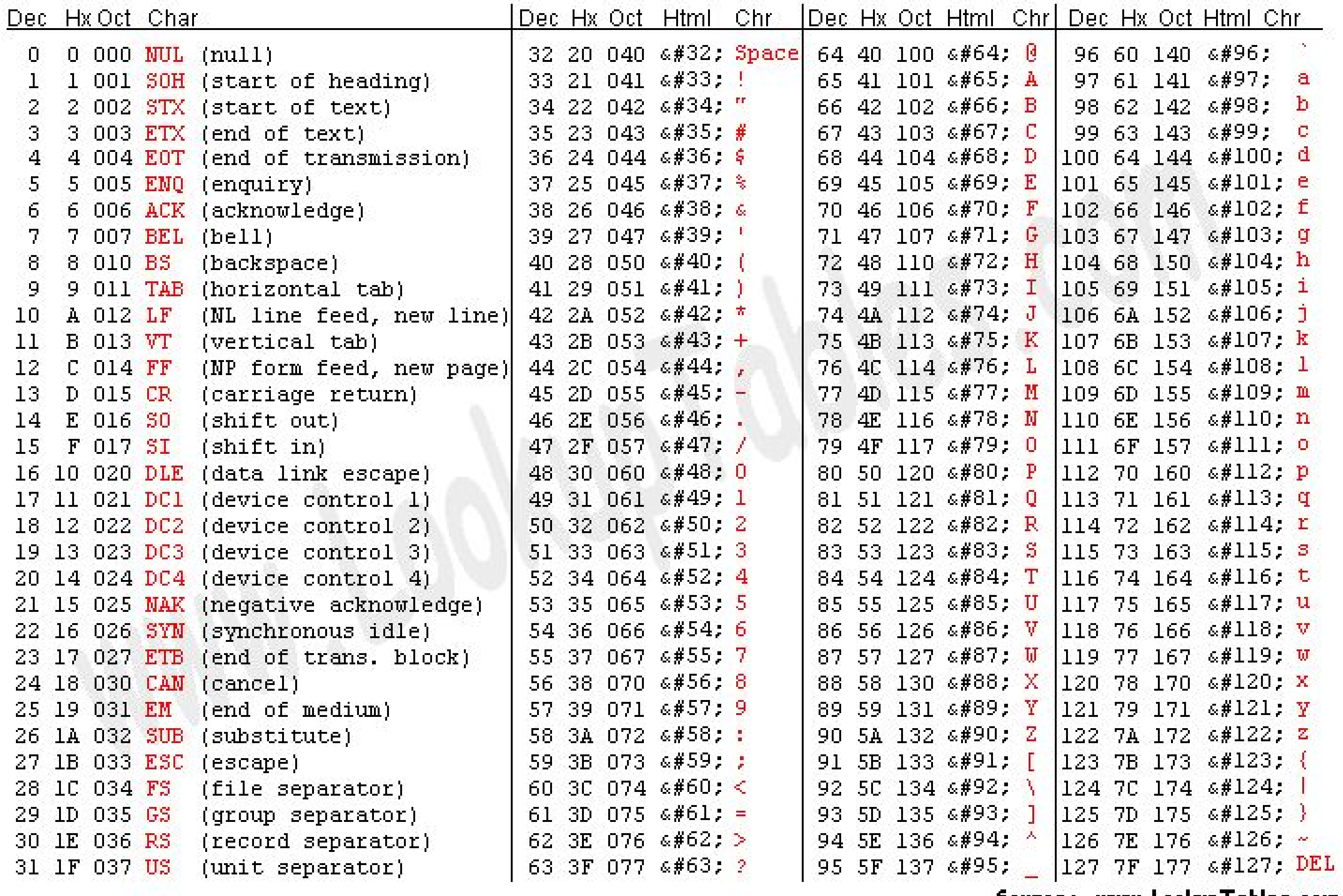
0101 ^ 1111 = 1010

HINT 2:

ASCII Character Table

HINT 3:
BINARY

ASCII TABLE



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL