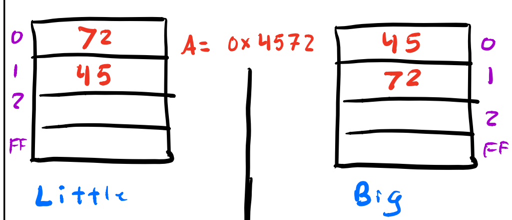
Data type	Size	Range	Alignment
bool	8 bits	0 to 1	1
char	8 bits	0 to 255	1
signed char	8 bits	-128 to 127	1
unsigned char	8 bits	0 to 255	1
signed short	16 bits	-32768 to 32767	2
unsigned short	16 bits	0 to 65535	2
signed int	16 bits	-32768 to 32767	2
unsigned int	16 bits	0 to 65535	2
signed long	32 bits	-2^{31} to 2^{31}-1	2
unsigned long	32 bits	0 to 2^{32}-1	2
signed long long	64 bits	-2^{63} to 2^{63}-1	2
unsigned long long	64 bits	0 to 2^{64}-1	2
float	32 bits		2
double	64 bits		2

Bits	Name	Range
n	*n*-bit integer (general case)	*Signed:* (-2^{n-1}) to ($2^{n-1}-1$) *Unsigned:* 0 to ($2^n - 1$)
8	byte, octet	*Signed:* −128 to +127 *Unsigned:* 0 to +255
16	halfword, word	*Signed:* −32,768 to +32,767 *Unsigned:* 0 to +65,535
32	word, doubleword, longword	*Signed:* −2,147,483,648 to +2,147,483,647 *Unsigned:* 0 to +4,294,967,295
64	doubleword, longword, long long, quad, quadword	*Signed:* −9,223,372,036,854,775,808 to +9,223,372,036,854,775,807 *Unsigned:* 0 to +18,446,744,073,709,551,615
128	octaword	*Signed:* −170,141,183,460,469,231,731,687,303,715,884,105,728 to +170,141,183,460,469,231,731,687,303,715,884,105,727 *Unsigned:* 0 to +340,282,366,920,938,463,463,374,607,431,768,211,455

specifier	Output	Example
d *or* i	Signed decimal integer	392
u	Unsigned decimal integer	7235
o	Unsigned octal	610
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (uppercase)	7FA
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation (mantissa/exponent), lowercase	3.9265e+2
E	Scientific notation (mantissa/exponent), uppercase	3.9265E+2
g	Use the shortest representation: %e or %f	392.65
G	Use the shortest representation: %E or %F	392.65
a	Hexadecimal floating point, lowercase	-0xc.90fep-2
A	Hexadecimal floating point, uppercase	-0XC.90FEP-2
c	Character	a
s	String of characters	sample
p	Pointer address	b8000000
n	Nothing printed. The corresponding argument must be a pointer to a signed int. The number of characters written so far is stored in the pointed location.	
%	A % followed by another % character will write a single % to the stream.	%

```c
int unsignedmin = 0;
printf("------------------------------------------------------------------------------------------------\n");
printf("| Data Type              |    Size (in bytes)    | Minimum              | Maximum              |\n");
printf("------------------------------------------------------------------------------------------------\n");
printf("| char                   | %d                    |%-21hhi| %-21hhi|\n", CHAR_BIT, SCHAR_MIN, SCHAR_MAX);
printf("| short int              | %d                    |%-21hd| %-21hd|\n",      sizeof(short int),              SHRT_MIN, SHRT_MAX);
printf("| int                    | %d                    |%-21d| %-21d|\n",        sizeof(int),                    INT_MIN, INT_MAX);
printf("| long int               | %d                    |%-21ld| %-21ld|\n",      sizeof(long int),               LONG_MIN, LONG_MAX);
printf("| long long int          | %d                    |%-21lld| %-21lld|\n",    sizeof(long long int),          LLONG_MIN, LLONG_MAX);
printf("| unsigned char          | %d                    |%-21hhu| %-21hhu|\n",    sizeof(unsigned char),          CHAR_MIN, UCHAR_MAX);
printf("| unsigned short int     | %d                    |%-21hu| %-21hu|\n",      sizeof(unsigned short int),     unsignedmin, USHRT_MAX);
printf("| unsigned int           | %d                    |%-21u| %-21u|\n",        sizeof(unsigned int),           unsignedmin, UINT_MAX);
printf("| unsigned long int      | %d                    |%-21u| %-21lu|\n",       sizeof(unsigned long int),      unsignedmin, ULONG_MAX);
printf("| unsigned long long int | %d                    |%-21u| %-21llu|\n",      sizeof(unsigned long long int), unsignedmin, ULLONG_MAX);
printf("| float                  | %d                    |%-21g| %-21g|\n",        sizeof(float),                  FLT_MIN, FLT_MAX);
printf("| double                 | %d                    |%-21g| %-21g|\n",        sizeof(double),                 DBL_MIN, DBL_MAX);
printf("------------------------------------------------------------------------------------------------\n");
return 0;
```

MSP430 - Little Endian



The LSB of the data is placed at the byte w/ the lowest address.

The MSB of the data is placed at the byte w/ the lowest address.

Metric Prefix	Symbol	Multiplier (Traditional Notation)	Exponential	Description
Yotta	Y	1,000,000,000,000,000,000,000,000	10^{24}	Septillion
Zetta	Z	1,000,000,000,000,000,000,000	10^{21}	Sextillion
Exa	E	1,000,000,000,000,000,000	10^{18}	Quintillion
Peta	P	1,000,000,000,000,000	10^{15}	Quadrillion
Tera	T	1,000,000,000,000	10^{12}	Trillion
Giga	G	1,000,000,000	10^{9}	Billion
Mega	M	1,000,000	10^{6}	Million
kilo	k	1,000	10^{3}	Thousand
hecto	h	100	10^{2}	Hundred
deca	da	10	10^{1}	Ten
base	b	1	10^{0}	One
deci	d	1/10	10^{-1}	Tenth
centi	c	1/100	10^{-2}	Hundredth
milli	m	1/1,000	10^{-3}	Thousandth
micro	μ	1/1,000,000	10^{-6}	Millionth
nano	n	1/1,000,000,000	10^{-9}	Billionth
pico	p	1/1,000,000,000,000	10^{-12}	Trillionth
femto	f	1/1,000,000,000,000,000	10^{-15}	Quadrillionth
atto	a	1/1,000,000,000,000,000,000	10^{-18}	Quintillionth
zepto	z	1/1,000,000,000,000,000,000,000	10^{-21}	Sextillionth
yocto	y	1/1,000,000,000,000,000,000,000,000	10^{-24}	Septillionth

Putty, moba, Serial app.

- Putty can only display 8-bit ASCII chars, everything else will be gibberish.
- Serial App translates serial packets sent to it and represents this data graphically.
  - ↳ must first send header byte and then break up data sent into 1 byte chunks

Configuring Serial App packets:

```c
char index = 0;
// Use character pointers to send one byte at a time
char *myPointer = (char* )&myData;

UART_putCharacter(0x55);                    // Send header
for(index = 0; index < 4; index++)
{    // Send 4-bytes of myData
    UART_putCharacter(myPointer[index]);
}
```

```c
void ADC_setup(void)
{
    P6SEL = 0x07;                               // Enable A/D channel inputs for x, y, and z (0000 0111)
    ADC12CTL0 = ADC12ON + ADC12MSC + ADC12SHT0_8;  // Turn on ADC12, extend sampling time to avoid overflow of results
    ADC12CTL1 = ADC12SHP + ADC12CONSEQ_1;       // Use sampling timer, repeated sequence
    ADC12MCTL0 = ADC12INCH_0;                   // ref += AVcc, channel = A0
    ADC12MCTL1 = ADC12INCH_1;                   // ref += AVcc, channel = A1
    ADC12MCTL2 = ADC12INCH_2 + ADC12EOS;        // ref += AVcc, channel = A2, end sequence
    ADC12IE = 0x02;                             // Enable ADC12IFG.1
    ADC12CTL0 |= ADC12ENC;                      // Enable conversions
}
```

```c
#pragma vector = ADC12_VECTOR
__interrupt void ADC12ISR(void)
{
    ADCXval = ADC12MEM0;                    // Move ADC12MEM0 (x) to ADCXval
    ADCYval = ADC12MEM1;                    // Move ADC12MEM1 (y) to ADCYval
    ADCZval = ADC12MEM2;                    // Move ADC12MEM2 (z) to ADCZval
    __bic_SR_register_on_exit(LPM0_bits);   // Exit LPM0
}
```

- Voltage Resolution: smallest change of an input analog signal that causes a change in the digital output.
  - ↳ Resolution of 12 bits,
- Reference Voltage: dials in the minimum and maximum values read by the ADC.
  - ↳ -5v = 0, 10v = 4095.
- Getting the voltage right is important because it maximizes the amount of resolution you get from the ADC.
- ADC resolution: $2^n - 1$ where n = bits of resolution.

  ADC 12 has 12 bits, so 4095 different input voltage levels.