

Panel 1

Prior to Le07

# Variable types

ME430 Mechatronics

1

Panel 2

Two primary variable types:

Integer

Floating point

2

Panel 3

|      |         | Unsigned Range |     | Signed Range |     |
|------|---------|----------------|-----|--------------|-----|
|      |         | Min            | Max | Min          | Max |
| char | 8-bits  |                |     |              |     |
| int  | 16 bits |                |     |              |     |
| long | 32 bits |                |     |              |     |

3

Panel 4

**TABLE 2-1: INTEGER DATA TYPE SIZES AND LIMITS**

| Type                  | Size    | Minimum        | Maximum       |
|-----------------------|---------|----------------|---------------|
| char <sup>(1,2)</sup> | 8 bits  | -128           | 127           |
| signed char           | 8 bits  | -128           | 127           |
| unsigned char         | 8 bits  | 0              | 255           |
| int                   | 16 bits | -32,768        | 32,767        |
| unsigned int          | 16 bits | 0              | 65,535        |
| short                 | 16 bits | -32,768        | 32,767        |
| unsigned short        | 16 bits | 0              | 65,535        |
| short long            | 24 bits | -8,388,608     | 8,388,607     |
| unsigned short long   | 24 bits | 0              | 16,777,215    |
| long                  | 32 bits | -2,147,483,648 | 2,147,483,647 |
| unsigned long         | 32 bits | 0              | 4,294,967,295 |

4

Panel 5

Overflow issues with variables

char x;

x = 250;

x = 260;

f = 200+100;

5

Panel 6

The ASCII conversion table

| ASCII Character Format |           |      |     |       |      |     |     |      |     |     |      |
|------------------------|-----------|------|-----|-------|------|-----|-----|------|-----|-----|------|
| Dec                    | Hex       | Dec  | Hex | Dec   | Hex  | Dec | Hex | Hex  |     |     |      |
| 0                      | mll       | 0x00 | 32  | space | 0x20 | 64  | @   | 0x40 | 96  | ,   | 0x60 |
| 1                      | soh       | 0x01 | 33  | !     | 0x21 | 65  | A   | 0x41 | 97  | a   | 0x61 |
| 2                      | stx       | 0x02 | 34  | *     | 0x22 | 66  | B   | 0x42 | 98  | b   | 0x62 |
| 3                      | etx       | 0x03 | 35  | #     | 0x23 | 67  | C   | 0x43 | 99  | c   | 0x63 |
| 4                      | eot       | 0x04 | 36  | 0x    | 0x24 | 68  | D   | 0x44 | 100 | d   | 0x64 |
| 5                      | enq       | 0x05 | 37  | %     | 0x25 | 69  | E   | 0x45 | 101 | e   | 0x65 |
| 6                      | ack       | 0x06 | 38  | &     | 0x26 | 70  | F   | 0x46 | 102 | f   | 0x66 |
| 7                      | bell      | 0x07 | 39  | '     | 0x27 | 71  | G   | 0x47 | 103 | g   | 0x67 |
| 8                      | backspace | 0x08 | 40  | (     | 0x28 | 72  | H   | 0x48 | 104 | h   | 0x68 |
| 9                      | tab       | 0x09 | 41  | )     | 0x29 | 73  | I   | 0x49 | 105 | i   | 0x69 |
| 10                     | lf        | 0x0A | 42  | *     | 0x2A | 74  | J   | 0x4A | 106 | j   | 0x6A |
| 11                     | vt        | 0x0B | 43  | +     | 0x2B | 75  | K   | 0x4B | 107 | k   | 0x6B |
| 12                     | np        | 0x0C | 44  | ,     | 0x2C | 76  | L   | 0x4C | 108 | l   | 0x6C |
| 13                     | cr        | 0x0D | 45  | -     | 0x2D | 77  | M   | 0x4D | 109 | m   | 0x6D |
| 14                     | so        | 0x0E | 46  | .     | 0x2E | 78  | N   | 0x4E | 110 | n   | 0x6E |
| 15                     | si        | 0x0F | 47  | /     | 0x2F | 79  | O   | 0x4F | 111 | o   | 0x6F |
| 16                     | dle       | 0x10 | 48  | 0     | 0x30 | 80  | P   | 0x50 | 112 | p   | 0x70 |
| 17                     | del       | 0x11 | 49  | 1     | 0x31 | 81  | Q   | 0x51 | 113 | q   | 0x71 |
| 18                     | dc2       | 0x12 | 50  | 2     | 0x32 | 82  | R   | 0x52 | 114 | r   | 0x72 |
| 19                     | dc3       | 0x13 | 51  | 3     | 0x33 | 83  | S   | 0x53 | 115 | s   | 0x73 |
| 20                     | dc4       | 0x14 | 52  | 4     | 0x34 | 84  | T   | 0x54 | 116 | t   | 0x74 |
| 21                     | nak       | 0x15 | 53  | 5     | 0x35 | 85  | U   | 0x55 | 117 | u   | 0x75 |
| 22                     | syn       | 0x16 | 54  | 6     | 0x36 | 86  | V   | 0x56 | 118 | v   | 0x76 |
| 23                     | etb       | 0x17 | 55  | 7     | 0x37 | 87  | W   | 0x57 | 119 | w   | 0x77 |
| 24                     | can       | 0x18 | 56  | 8     | 0x38 | 88  | X   | 0x58 | 120 | x   | 0x78 |
| 25                     | em        | 0x19 | 57  | 9     | 0x39 | 89  | Y   | 0x59 | 121 | y   | 0x79 |
| 26                     | eof       | 0x1A | 58  | :     | 0x3A | 90  | Z   | 0x5A | 122 | z   | 0x7A |
| 27                     | esc       | 0x1B | 59  | ;     | 0x3B | 91  | [   | 0x5B | 123 | {   | 0x7B |
| 28                     | fr        | 0x1C | 60  | <     | 0x3C | 92  | \   | 0x5C | 124 |     | 0x7C |
| 29                     | gs        | 0x1D | 61  | =     | 0x3D | 93  | ]   | 0x5D | 125 | }   | 0x7D |
| 30                     | rs        | 0x1E | 62  | >     | 0x3E | 94  | ^   | 0x5E | 126 | ~   | 0x7E |
| 31                     | us        | 0x1F | 63  | ?     | 0x3F | 95  |     | 0x5F | 127 | del | 0x7F |

6

Panel 7

### 2.1.2 Floating-point Types

32-bit floating-point types are native to MPLAB C18 using either the `double` or `float` data types. MPLAB C18 utilizes the IEEE-754 floating-point standard to represent floating-point types. The ranges of the floating-point type are documented in Table 2-2.

**TABLE 2-2: FLOATING-POINT DATA TYPE SIZES AND LIMITS**

| Type   | Size    | Minimum Exponent | Maximum Exponent | Minimum Normalized                  | Maximum Normalized                                     |
|--------|---------|------------------|------------------|-------------------------------------|--|
| float  | 32 bits | -126             | 128              | $2^{-126} \approx 1.17549435e - 38$ | $2^{128} * (2 \cdot 2^{-15}) \approx 6.80564693e + 38$ |
| double | 32 bits | -126             | 128              | $2^{-126} \approx 1.17549435e - 38$ | $2^{128} * (2 \cdot 2^{-15}) \approx 6.80564693e + 38$ |

7

Panel 8

| Sign | 8-bit biased exponent <i>E</i> | 23-bit unsigned fraction <i>f</i> |
|------|--------------------------------|-----------------------------------|
| ±    | $e_7e_6e_5e_4e_3e_2e_1e_0$     | $d_0d_1d_2d_3 \dots d_{23}$       |

8

Panel 9

Floating points aren't exact

```
float x = 0.1;
x = 0 0111 1011 100 1100 1100 1100 1100 1101
```

Actually get's stored as:

```
x = 0.1000000005
```

```
f = .1;
f2 = 0.100000005;
```

```
if (f == f2)
    printf("The two are equal");
else
    printf("The two are different");
```

For this reason PCs use more than 32 bits

9

Panel 10

| Operator                   | Symbol   | Associativity |
|----------------------------|----------|---------------|
| Address                    | &        | L to R        |
| Dereferencing              | *        |               |
| Logical NOT                | !        | R to L        |
| Bit inverse                | ~        |               |
| Increment                  | ++       |               |
| Decrement                  | --       |               |
| Negation                   | -        |               |
| Cast (see note**below)     | (<type>) |               |
| Multiplication             | *        | L to R        |
| Division                   | /        |               |
| Modulus                    | %        |               |
| Addition                   | +        | L to R        |
| Subtraction                | -        |               |
| Shift Bits Left            | <<       | L to R        |
| Shift Bits Right           | >>       |               |
| Less Than                  | <        | L to R        |
| Less Than or Equal To      | <=       |               |
| Greater Than               | >        |               |
| Greater Than or Equal To   | >=       |               |
| Equal To                   | ==       |               |
| Not Equal To               | !=       |               |
| Bit AND                    | &        | L to R        |
| Exponentiation (for float) | ^        | L to R        |
| Exclusive OR (for int)     |          |               |
| Bit OR                     |          | L to R        |
| Logical AND                | &&       | L to R        |
| Logical OR                 |          | L to R        |
| Assignment                 | =        | R to L        |
| Additive Assignment        | +=       |               |
| Multiplicative Assignment  | *= etc.  |               |

10

Panel 11

Integer math issues

```
char y;
```

```
y = 3/4 * 100;
```



```
y = 100*8/4;
```



11

Panel 12

Arrays

In addition to integers and floating point numbers, you'll probably also need to be able to use arrays. Arrays are lists that have an index. They can be very handy to go through lots of data.

Let's see an example of using an array:

```
int i[] = { 1, 2, 3, 4, 5};

int y[5];
y[0] = 10;
y[1] = 20;
y[2] = 30;
y[3] = 40;
y[4] = 50;
```

Strings

```
char listchar[]={'t','o','y',' ','b','o','a','t'};
char listchar2[]="toy boat";
```

```
char numchar[2];
numchar[0]= 0x61;
numchar[1]= 0x62;
```

```
printf(" %c ",listchar[1]);
printf(" %c ",numchar[1]);
```

12