**Digital Sensors**

Digital ports range from dgtl1 to dgtl12.

**Bumper or limit switch:**
(to be placed at the top of the program)

```c
#pragma config(Sensor, dgtl1, touchSensor, sensorTouch)
```

(in the body of the program)

```c
int sensor_value;
sensor_value = SensorValue(touchSensor);
```

**Bumper and Limit Switch Characteristics:**
Digital output [0 1]
0 = switch is pressed

**Rotation counter (1/2 quad encoder):**
(to be placed at the top of the program)

```c
#pragma config(Sensor, dgtl2, ROT, sensorRotation)
```

(in the body of the program)

```c
int sensor_value;
sensor_value = SensorValue(ROT);
```

**Quad Encoder:**
(to be placed at the top of the program)

```c
#pragma config(Sensor, dgtl3, Quadencoder, sensorQuadEncoder)
```

NOTE: you also have to plug the second wire into the next dgtl port (in this case dgtl4).

(in the body of the program)

```c
int Dist;
SensorValue[rightEncoder] = 0; // Set the encoder so that it starts counting at 0
Dist = SensorValue(rightEncoder); //note that when setting the value you use ‘[ ]‘ versus ‘( )‘
//when calling the vale
```

**Sonar sensor:**
(to be placed at the top of the program)

```c
#pragma config(Sensor, dgtl9, sonarSensor, sensorSONAR_mm)
```

NOTE: you also have to plug the second wire into the next dgtl port (in this case dgtl10). You also need to plug the input wire into the first port and the output wire into the second port.

(in the body of the program)

```c
int sensor_value;
sensor_value = SensorValue(sonarSensor);
```
Analog Sensors Input and Output

Analog ports range from in1 to in8.

**Light sensor:**
(to be placed at the top of the program)

```c
#pragma config(Sensor, in1,    lightSensor, sensorReflection)
```
where in1 is the port declaration and LightSensor is the name of the sensor.

(in the body of the program)

```c
int sensor_value;
sensor_value = SensorValue(lightSensor);
```

*Light Sensor Characteristics:*
Analog output [0 1023]
0 = light; 1023 = dark

**Potentiometer:**
(to be placed at the top of the program)

```c
#pragma config(Sensor, in1, POT, sensorPotentiometer)
```

(in the body of the program)

```c
int sensor_value;
sensor_value = SensorValue(POT);
```

*Potentiometer Characteristics:*
Analog output [0 4095]
Has a movement range of approximately 260 degrees
Thus:

\[
\frac{4095}{260 \text{ degrees}} = 15.75 \text{ /degree}
\]
Motor Control

Motor ports range from port1 to port10.

(This configures two motors. One motor is plugged into the 2\textsuperscript{nd} motor port (the second variable) and is named rightMotor (third variable). The other motor is plugged into the 3\textsuperscript{rd} port and is named leftMotor. Note that you can reverse the motors in the configuration statement, like the rightMotor is in the example below.)

(to be placed at top of program)

\begin{verbatim}
#pragma config(Motor, port2,           rightMotor,    tmotorNormal, openLoop, reversed)
#pragma config(Motor, port3,           leftMotor,     tmotorNormal, openLoop)
\end{verbatim}

\begin{verbatim}
motor[leftMotor] = -127;   //activates the leftMotor full speed backwards.  
//Values:  –127 to 127.  negative numbers are //counterclockwise.  0 
//= stopped.
\end{verbatim}

For Servo control:

(to be placed at top of program)

\begin{verbatim}
#pragma config(Motor, port1, Servo, tmotorNormal, openLoop)
\end{verbatim}

\begin{verbatim}
motor[Servo] = -127;  //positions the servo motor to one of the extremes; range : -127  to 127
\end{verbatim}
Timers

- There are four timers that start as soon as the microprocessor is powered on, [T1, T2, T3, T4]. The times begin when the controller is turned on, but you can reset any of the timers at any time.

ClearTimer(T1);  // Resets the value of Timer "T1" back to zero seconds.

int valTime1, valTime10, valTime100;  // Create three integers to read the value of the timer.

valTime1 = time1[T1];  // Gets the value of Timer T1 in 1ms increments and stores it in a variable.
valTime10 = time10[T1];  // Gets the value of Timer T1 in 10ms increments and stores it in a variable.
valTime100 = time100[T1];  // Gets the value of Timer T1 in 100ms increments and stores it in a variable.