

- Battery and Power I/O pin. When a sensor is setup for the dirOutput direction, you can use this
 - nBatteryLevel
 - Integer - The voltage of the FRC main battery in millivolts. A value of 7500 indicates a voltage of 7.5 volts.
 - nBackupBatteryLevel
 - Integer - The voltage of the FRC backup battery in millivolts. A value of 7500 indicates a voltage of 7.5 volts.
- FRC Digital I/O
 - frcDigitalIODirection[port]
 - Array - This read/write variable is used to configure the direction (dirInput or dirOutput) of a digital I/O pin.
 - Ports: pio1 through pio18
 - frcDigitalIOValue[port]
 - Array - This array variable contains the current value of a digital variable to write the value of the pin (true or false). When direction is dirInput, you can read the current value of a pin (true or false).
 - Ports: pio1 through pio18
- FRC Relay Control
 - frcRelay[port]
 - Array Provides control for each of the 8 pairs FRC relays. Typically a pair of relay outputs will be connected to an IFI "Spike" motor control modules. Relays can be set to one of the four states-- 'off', 'forward', 'reverse' and 'brake',
 - frcRelay16[port]
 - Provides individual control for each of the 16 FRC relays. Relay can be set ON (true) or OFF (false).
- FRC Remote Controls
 - frcOIJoystickButtons[]
 - Port: oiButtonPort1Button1, oiButtonPort1Button2, oiButtonPort1Button3, oiButtonPort1Button4, all the way to oiButtonPort2,Button1... oiButtonPort4Button4
 - frcRF[]
 - Port: p1_x, p1_y, p1_aux, p1_wheel – p2, p3 and p4 are also valid.
- FRC Math Functions
 - sin(fRadians), cos(fRadians)
 - Returns the sine or cosine of the input parameter. The input value is a float value specified in radians.
 - asin(Sine)
 - Returns the arc-sine, i.e. the angle (in radians) whose sine value is “Sine”.
 - acos(Cos)
 - Returns the arc-cosine, i.e. the angle (in radians) whose cosine value is “Cos”.
 - atan(Tangent)
 - Returns the arc-tangent, i.e. the angle (in radians) whose tangent value is “Tangent”.Robotics Academy

- PI
 - Constant float variable containing the value of PI.
- sinDegrees(degrees), cosDegrees(degrees)
 - Returns the sine or cosine of the input parameter. The input value is float value specified in degrees.
- radiansToDegrees(radians), degreesToRadians(degrees)
 - Converts between degrees and radians. The converted values are normalized – 0 to 359 for degrees and 0 to 2 * PI for radians.
- abs(input), exp(input), sgn(input), sqrt(input)
 - Performs the appropriate math function on the input variable.
- srand(seed)
 - Sets the seed for the random number generator.
- random(range)
 - Returns an integer value in the range 0 to 'range' where 'range' should be a positive integer in the range 0..32767.
- Motors
 - bMotorReflected[port]
 - Boolean array. Indicates that the direction of a motor should be reflected 180 degrees. Useful when mechanical design results in a logical "reversed" condition of a motor.
 - Ports: port1 through port 16
 - Motor[port]
 - An array variable with one element for each of the possible motors. Used to sets the speed (-127 to +127) for a motor. To drive the motor on port1 at 50% of full power you use the statement motor[port1] = 64;
 - Ports: port1 through port 16
- Analog Sensors
 - SensorValue[port]
 - This array variable contains the current value of the sensor attached the port specified. Values range from 0 to 1023.
 - Ports: in1 through in16
- Timing/Timers
 - wait1Msec(nMSec); wait10Msec(nTenMSec);
 - Program execution will wait for the specified number of clock units. Units can be in either 1-millisecond or 10-millisecond counts.
 - The maximum interval that can be specified is either 32.767 seconds or 327.67 seconds depending on which function is used.
 - An alternative, and far less efficient, method to perform a wait is to continually execute a tight code loop looking to see if a timer has reached the desired interval. It is best to use the wait functions to insert a programmed delay in a program because tasks that are waiting do not consume any CPU cycles. This makes the most number of CPU cycles available for other tasks.
 - ClearTimer(theTimer);
 - Resets the value of the specified timer to zero.
 - time100[], time10[], time1[]

- These three four-element arrays hold the current value of the respective timers. Each of the timer values can be retrieved in units of 1, 10 and 100 milliseconds depending on which array is used. For example, `time1[T1]` retrieves the value of timer T1 in units of 1-msec and `time10[T1]` retrieves the value using a 10-msec tick. And `time100[T1]` retrieves the value using 100-msec tick.
 - Note that the arrays are “linked”. Setting `time1[T1] = 0;` will also reset the value of `time10[T1]` and `time100[T1]`.
 - Timers: T1, T2, T3, T4
 - `nSysTime`
 - This variable contains the value of the lower 16-bits of the internal 1-msec clock. This variable is reset when FRC is first powered on.
 - `nPgmTime`
 - This variable contains the value of the lower 16-bits of the internal 1-msec clock. This variable is reset when user program first starts running. This clock does not increment when the program is in a debugger "suspended" state which is useful during single step debugging as the clock does not increment.
 - `setClockTime(hours, minutes);`
 - This function can be used to set the time of day. The FRC maintains an internal system clock that counts minutes. Note that the system clock is reset to time 0:00 whenever the FRC is first powered on; the clock is not incremented when the FRC is powered off.
 - `nClockMinutes`
 - This read/write variable provides access to the FRC clock described above. The value ranges from 0 to 1439 before it wraps around back to 0. [Note: there are 1440 minutes in 24 hours].