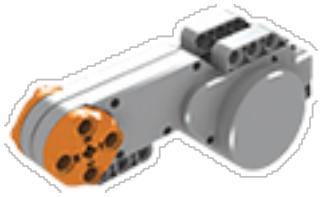
ECRobot C API

This page presents the API for managing sensors/actuators of the NXT. The documentation is largely inspired from http://lejos-osek.sourceforge.net/ecrobot_c_api.htm. This documentation focuses on the basic NXT sensors/actuators (motors, touch sensor, ultrasonic sensor, LCD display and NXT internal). For a more comprehensive documentation see http://lejos-osek.sourceforge.net/ecrobot_c_api.htm.

ECRobot API consists of low level device API and ECRobot wrapper API (prefix of the API is ecrobot_) that is designed for real-time control application programming. It is better to use the original API to reduce the run time overhead. When you use ECRobot API, you need to include ecrobot_interface.h in your source code.

1/ Servo Motor API



Servo Motor API	Description
int nxt_motor_get_count(U32 *n*)	gets Servo Motor revolution count in degree.Count may be negative or positive. Counts are not reset to zero when reaching 360 (resp. -360). Parameters: *n*: NXT_PORT_A, NXT_PORT_B, NXT_PORT_C Returns: Servo Motors revolution in degree
void nxt_motor_set_count(U32 *n*, int *count*)	sets Servo Motor revolution count in degree. Parameters: *n*: NXT_PORT_A, NXT_PORT_B, NXT_PORT_C *count*: Servo Motor revolution value Returns: none
void nxt_motor_set_speed(U32 *n*, int *speed_percent*, int *brake*)	sets Servo Motor PWM value and brake mode. Parameters: *n*: NXT_PORT_A, NXT_PORT_B, NXT_PORT_C *speed_percent*: -100 to 100 *brake*: 0 (float), 1 (brake)

	Returns: none
S32 ecrobot_get_motor_rev(U8 port_id)	gets Servo Motor revolution value in degree. Wrapper of nxt_motor_get_count. Parameters: *port_id*: NXT_PORT_A, NXT_PORT_B, NXT_PORT_C Returns: Servo Motors revolution in degree
void ecrobot_set_motor_speed(U8 port_id, S8 speed)	sets Servo Motor PWM value. Wrapper of nxt_motor_set_speed, but brake mode is fixed as brake. Parameters: *port_id*: NXT_PORT_A, NXT_PORT_B, NXT_PORT_C *speed*: -100 to 100 Returns: none
void ecrobot_set_motor_mode_speed(U8 port_id, S32 mode, S8 speed)	sets Servo Motor brake mode and PWM value. Wrapper of nxt_motor_set_speed Parameters: *port_id*: NXT_PORT_A, NXT_PORT_B, NXT_PORT_C *mode: 0(float), 1(brake)* *speed*: -100 to 100 Returns: none

2/ Touch Sensor API

Touch Sensor API	Description
U8 ecrobot_get_touch_sensor(U8 *port_id*)	gets Touch Sensor status.Parameters: *port_id*: NXT_PORT_S1, NXT_PORT_S2, NXT_PORT_S3, NXT_PORT_S4 Returns: 0 (not touched), 1 (touched)

3/ Ultrasonic Sensor API

Ultrasonic Sensor has its brain to communicate with the ARM7 via another I2C communication channel. ecrobot_get_sonar_sensor sends a protocol data to communicate with the Ultrasonic Sensor. However, actual data transaction between the ARM7 and the Ultrasonic Sensor is done by an ISR triggered by this function call, so there is one execution cycle delay to achieve consistent data acquisition.

Ultrasonic Sensor API	Description
void ecrobot_init_sonar_sensor(U8 *port_id*)	initializes a port for I2C communication for Ultrasonic Sensor. This function should be implemented in the device initialize hook routine. Parameters: *port_id*: NXT_PORT_S1, NXT_PORT_S2, NXT_PORT_S3, NXT_PORT_S4 Returns: none
S32 ecrobot_get_sonar_sensor(U8 *port_id*)	gets Ultrasonic Sensor measurement data in cm via I2C. Parameters: *port_id*: NXT_PORT_S1, NXT_PORT_S2, NXT_PORT_S3, NXT_PORT_S4 Returns: -1 to 255 (-1: not ready for measurement)

void ecrobot_term_sonar_sensor(U8 *port_id*)	terminates I2C communication used for Ultrasonic Sensor. This function should be implemented in the device terminate hook routine. Parameters: *port_id*: NXT_PORT_S1, NXT_PORT_S2, NXT_PORT_S3, NXT_PORT_S4 Returns: none

4/ LCD display API

LCD display API	Description
void display_update(void)	updates LCD display. Without a call to display_update, nothing appears on the LCD. Parameters: none Returns: none
void display_clear(U32 *updateToo*)	clears LCD display buffer. Parameters: *updateToo*: 0 (not update LCD display), 1(update LCD display after clearing the buffer) Returns: none
void display_goto_xy(int *x*, int *y*)	specifies text display position. Top left is (0,0). Parameters: *x*: horizontal position (0 to 15) *y*: vertical position (0 to 7) Returns: none

void display_string(const char *_str_)	stores string into LCD display buffer. Parameters: _str_: supported ASCII characters are 0x00 to 0x7F Returns: none
void display_hex(U32 _val_, U32 _places_)	stores integer value into LCD display buffer to display hex expression. Parameters: _val_: integer value to be displayed _places_: number of characters to be reserved Returns: none
void display_unsigned(U32 _val_, U32 _places_)	stores unsigned integer value into LCD display buffer. Parameters: _val_: unsigned integer value to be displayed _places_: number of characters to be reserved Returns: none
void display_int(int _val_, U32 _places_)	stores signed integer value into LCD display buffer. Parameters: _val_: signed integer value to be displayed _places_: number of characters to be reserved Returns: none
void ecrobot_status_monitor(const CHAR *_target_name_)	displays NXT internal status (application name, system tick, battery voltage, raw A/D data, motor revolutions, Bluetooth connection status, and Ultrasonic Sensor data) on the LCD. It is recommended to invoke this API in a low speed periodical Task (i.e. 500msec). Parameters: _target_name_: target name string Returns: none

5/ NXT internal API

NXT internal API	Description
U8 ecrobot_is_ENTER_button_pressed(void)	returns the status of ENTER (ENTR) button. Parameters: none Returns: Status of ENTER (ENTR) button 1: button is pressed 0: button is not pressed
U8 ecrobot_is_RUN_button_pressed(void)	returns the status of RUN button. Parameters: none Returns: Status of RUN button 1: button is pressed 0: button is not pressed
U16 ecrobot_get_battery_voltage(void)	gets battery voltage data. Parameters: none Returns: battery voltage in mV. (i.e. 9000 = 9.000V)
U32 systick_get_ms(void)	gets system tick in msec. system tick is started when the NXT is turned on (not started when an application begins) Parameters: none Returns: system tick in msec
U32 ecrobot_get_systick_ms(void)	gets system tick in msec. Wrapper of systick_get_ms.

	Parameters: none Returns: system tick in msec
void systick_wait_ms(U32 *ms*)	waits for specified msec. Parameters: *ms*: wait time in msec Returns: none