hochschule aschaffenburg
university of applied sciences

# *Commands for EV3-API*

## Initialization:

| command | explanation |
|---|---|
| **#include <ev3.h>** | including Header |
| **InitEV3();** | Initialization of all EV3-Functions |
| **FreeEV3();** | Closing of all EV3-Functions |

## Display:

| command | explanation |
|---|---|
| bool **LcdClean();** | Erase Display |
| **LcdPrintf(<Color>,<Text>,…);** | Working like printf() |

**Help:**

| parameter | type | explanation | value |
|---|---|---|---|
| **<Color>** | char | Color of text | 1: black text<br>0: wihteteext with black background |
| **<Text>** | char* | Pointer to text | e.g. “Hello EV3“ |

## Break:

| command | explanation |
|---|---|
| void **Wait(<Zeit_ms>);** | Break code for a given time |

**Help:**

| parameter | type | explanation | value |
|---|---|---|---|
| **<Zeit_ms>** | unsigned long | time in ms<br>possible to use the given macros or type directly the value | MS_1…10 :1<br>10…100 :10<br>50…500 :50<br>100…900 :100<br>SEC_1…10 :1<br>5…20 :5<br>30<br>MIN_1 |

## Inputs (Sensoren):

**Commands:**

| command | explanation |
|---|---|
| int **setAllSensorMode(<Mode>, <Mode>,<Mode>,<Mode>);** | Allocate of the sensor types of all 4 ports in the correct order (IN_1, IN_2, IN_3, IN_4) |
| int **readSensor(<Input>);** | Readout of the actual sensor data |
| int **setIRBeaconCH(<Input>, <Channel>);** | Set Channel of the Beacon for Readout (default: Ch. 1 ) |

**Help:**

| parameter | type | explanation | value |
|---|---|---|---|
| **<Input>** | int | Input-Ports | IN_1, IN_2, IN_3, IN_4 |
| **<Mode>** | char | Name und Mode of the connected Sensors | See next table |
| **<Channel>** | int | Channel of the Beacon.<br>Needed for: IR_SEEK and IR_REMOTE | BEACON_CH_1,<br>BEACON_CH_2,<br>BEACON_CH_3,<br>BEACON_CH_4 |

**Sensor type:**

| Sensor | <Mode> | explanation | return value |
|---|---|---|---|
| **No Sensor** | NO_SEN | No sensor to the port connected | -1 |
| **Touch sensor** | TOUCH_PRESS | Return of state (2 states possible) | Not pressed: 0<br>pressed: 1 |
| **Light sensor** | COL_REFLECT | Return of the reflected light intensities in % | 0 to 100 |
| | COL_AMBIENT | Return of room light intensities in % | 0 to 100 |
| | COL_COLOR | Return of color | 0: transparent<br>1: black<br>2: blue<br>3: green<br>4: yellow<br>5: red<br>6: white<br>7: brown |
| **Sonar sensor** | US_DIST_MM | Return of distance in mm | 0 to 2550 |
| **Gyroscope** | GYRO_ANG | Return of angle in ° | -180 to 180 |
| | GYRO_RATE | Return of gear rate in °/s | -440 to 440 |
| **EV3 Infrared** | IR_PROX | Return of distance in % (up to 70cm) | 0 (Near) to 100 (Far) |
| | IR_SEEK | Position of the Beacon | -25 to 25 |
| | IR_REMOTE | Controlling EV3 with Beacon | BEACON_OFF<br>BEACON_UP_LEFT<br>BEACON_DOWN_LEFT<br>BEACON_UP_RIGHT<br>BEACON_DOWN_RIGHT<br>BEACON_UP<br>BEACON_DIAG_UP_LEFT |

| | | | |
|---|---|---|---|
| | | | BEACON_DIAG_UP_RIGHT<br>BEACON_DOWN<br>BEACON_ON<br>BEACON_LEFT<br>BEACON_RIGHT |
| **NXT Infrared** | NXT_IR_SEEKER | | 1 to 9 |
| **NXT Temperature** | NXT_TEMP_C | Temperature in °C | -55 to 128 |
| | NXT_TEMP_F | Temperature in °F | -67 to 262 |

## Outputs (Motoren):

**Controlling:**

| command | explanation |
|---|---|
| void **OnFwdReg(<Output>, <Speed>);** | Forwards/backwards with given speed |
| void **OnRevReg(<Output>, <Speed>);** | |
| void **OnFwdSync(<Output>, <Speed>);** | synchronized forwards/backwards with given speed (only working with two motors) |
| void **OnRevSync(<Output>, <Speed>);** | |
| void **Off(<Output>);** | Switch off motors |
| void **RotateMotor(<Output>, <Speed>, <Angle>);** | Rotate with given speed for a defined angle (Code stops till the angle is reached) |

**Reading out:**

| command | explanation |
|---|---|
| int **MotorRotationCount(<Output>);** | Rotation angle of the motors in ° |
| void **ResetRotationCount(<Output>);** | Reset of rotation angle |
| char **MotorPower(<Output>);** | Actual motor speed |

**Help:**

| parameter | type | explanation | value |
|---|---|---|---|
| **<Output>** | int | Output-ports | OUT_A, OUT_B, OUT_C, OUT_D, OUT_AB, OUT_AC, OUT_AD, OUT_BC, OUT_BD, OUT_CD, OUT_ABC, OUT_BCD, OUT_ABCD, OUT_ALL |
| **<Speed>** | char | speed | 0 to 100 |
| **<Angle>** | int | angle in ° | |

# Buttons und LED:

### Functions for LED:

| command | explanation |
|---|---|
| void **SetLedPattern(<Pattern>);** | Changing color of LED behind buttons |
| void **SetLedWarning (<Value>);** | Activate/deactivate of warning.<br>LED color cannot be changed while warning is set. |

### Functions for buttons:

| command | explanation |
|---|---|
| word **ButtonWaitForAnyPress(<Time>);** | Waiting for button press for given time |
| bool **ButtonIsUp(<Button>);** | Check if button is pressed or not<br>(1: true, 0: false) |
| bool **ButtonIsDown(<Button>);** | |
| void **ButtonWaitForPress(<Button>);** | Waiting till a specific button is pressed |
| void **ButtonWaitForPressAndRelease(<Button>);** | Waiting till a specific button is pressed and released |

### Help:

| parameter | type | explanation | value |
|---|---|---|---|
| **<Pattern>** | byte | Color and modus of LED | LED_BLACK (0)<br>LED_GREEN (1)<br>LED_RED (2)<br>LED_ORANGE (3)<br>LED_GREEN_FLASH (4)<br>LED_RED_FLASH (5)<br>LED_ORANGE_FLASH (6)<br>LED_GREEN_PULSE (7)<br>LED_RED_PULSE (8)<br>LED_ORANGE_PULSE (9) |
| **<Value>** | bool | Activate or deactivate | 0: deactivate<br>1: activate |
| **<Time>** | uint | time in ms | 0: for endless waiting |
| **<Button>** | byte | Name of button | BTNEXIT, BTNRIGHT,<br>BTNLEFT, BTNCENTER,<br>BTNUP, BTNDOWN |

# Sounds:

### Commands:

| command | explanation |
|---|---|
| void **PlayTone(<Frequency>, <Duration>);** | Play one tone for a defined time |
| void **PlayTones(<Frequencies[]>);** | Play multiple tones |
| void **PlaySound(<Code>);** | Play a system sound |
| int **SoundState();** | Returns the current **<State>** |
| void **StopSound();** | Stop any playing sounds |
| bool **SoundTest();** | Returns TRUE when a Sound can be played |
| void **MuteSound ();** | Mute sounds (No sound can be played) |
| void **UnmuteSound ();** | Unmute sounds |
| void **ClearSound ();** | Synonym for StopSound() |

### Help:

| parameter | type | explanation | value |
|---|---|---|---|
| **<Frequency>** | unsigned short | Frequency of tone<br>Defined: Second to seventh octave<br>Example: All Tones of second octave<br>➔ | `TONE_C2, TONE_CS2, TONE_D2, TONE_DS2, TONE_E2, TONE_F2, TONE_FS2, TONE_G2, TONE_GS2, TONE_A2, TONE_AS2, TONE_B2` |
| **<Duration>** | unsigned short | Time in ms or with defines | NOTE_WHOLE, NOTE_HALF, NOTE_QUARTER, NOTE_EIGHT, NOTE_SIXTEEN |
| **<Frequencies[]>** | Tone | Array of the structure Tone<br>Tone: <Frequency> & <Duration> | Example:<br>Tone a[2] = {<br>{TONE_C4, NOTE_HALF},<br>{TONE_F6, NOTE_EIGHT}<br>}; |
| **<Code>** | byte | Names of system sounds | SOUND_CLICK, SOUND_DOUBLE_BEEP, SOUND_DOWN, SOUND_UP, SOUND_LOW_BEEP, SOUND_FAST_UP |
| **<State>** | int | States of the sound module | SOUND_STATE_IDLE, SOUND_STATE_SETUP_FILE, SOUND_STATE_FILE, SOUND_STATE_FILE_LOOPING, SOUND_STATE_TONE, SOUND_STATE_TONE_LOOPING SOUND_STATE_STOP |