librcb4

Library to communicate with KONDO's RCB4 board (main processor of the KHR-3)

By Alfonso Arbona Gimeno
Katolab - Nagoya Institute of Technology
Supervisor: Professor Shohei KATO

Capabilities

- Read/write access to ROM, RAM and ICS.
- Direct servo control (individual or multiple at the same time).
- Pose selection (move all/some servos to a predefined position loaded in RAM).
- Multiple robots at the same time.
- Math, logic, call, jump and return functions implemented.
- Helper functions to do common tasks.

Initialization

- Create a connection to the servo:
  - `rcb4_connection* rcb4_init(const char* tty);`
  - `rcb4_connection` is a private structure with the connection information.
  - It automatically tries to find out the baudrate.
  - Can be called several times for different servos.
- Remember to close the connection!
  - `void rcb4_deinit(rcb4_connection* conn);`

Commands

- First create the command (uses malloc):
  - `rcb4_comm* rcb4_command_create(enum e_rcb4_command_types type);`
- Then fill the command data with the different functions (more in the following slides).
- Send the command to the robot:
  - `int rcb4_send_command(rcb4_connection* conn, rcb4_comm* comm, uint8_t* reply);`
- Delete the command (frees the memory):
  - `void rcb4_command_delete(rcb4_comm* comm);`

Commands

- If you want to reuse the same variable for a new command (instead of deleting and creating):
  - `int rcb4_command_recreate(rcb4_comm* comm, enum e_rcb4_command_types type);`
- Function types (`enum e_rcb4_command_types`):
  - **MOV** (copy from src to dst).
  - **Servocontrol** (ICS, single, constant, series, speed/stretch).
  - Logic (AND, OR, XOR, NOT).
  - Bitwise (Shift).
  - Arithmetic (Add, substract, multiply, divide, modulo).
  - Control (jump, call, ret, ping, version). *Implemented as a special function.*
- All functions return 0 on success, else if there was an error.

Servo control

- **ICS** allows to set the position and speed of some or all of the servos using data *already loaded in RAM*.
- **Single** allows to set the speed and position of a *single* servo.
- **Constant** sets individually the position of a group of servos and a *single speed for all of them*.
- **Series** is like *constant* but also allows the speed to be set individually (doesn't seem to work).
- **Speed/stretch** is not well documented so it's not implemented for now.

Pseudocode – Move two servos

- Initialize:
  - `conn = rcb4_init(“/dev/ttyUSB0”);`
- First servo:
  - `comm = rcb4_command_create(RCB4_COMM_SINGLE);`
  - `rcb4_command_set_servo(comm, servo_1, speed_1, position_1);`
  - `rcb4_send_command(conn, comm, NULL);`
- Second servo:
  - `rcb4_command_recreate(comm, RCB4_COMM_SINGLE);`
  - `rcb4_command_set_servo(comm, servo_2, speed_2, position_2);`
  - `rcb4_send_command(conn, comm, NULL);`
- Deinitialize:
  - `rcb4_command_delete(comm);`
  - `rcb4_deinit(conn);`

Parameters

- The **position** of a servo is defined as an 16 bit unsigned integer (0x0000 ~ 0xFFFF) but most servos can only move in a narrower range.
- The **speed** is an unsigned byte (**1** ~ 255), being 1 the slowest speed and 255 the fastest.
- Each servo has an ID from 1 to 36 but not all of them are physically present in the robot.

Pseudocode – Move some servos

- Initialize:
  - `conn = rcb4_init("/dev/ttyUSB0");`
- Create the command:
  - `comm = rcb4_command_create(RCB4_COMM_CONST);`
  - `rcb4_command_set_speed(comm, speed); // For both servos`
- Set the servos (speed argument ignored):
  - `rcb4_command_set_servo(comm, servo_1, 0x00, position_1);`
  - `rcb4_command_set_servo(comm, servo_2, 0x00, position_2);`
  - [...]
  - `rcb4_command_set_servo(comm, servo_n, 0x00, position_n);`
- Send the command:
  - `rcb4_send_command(conn, comm, NULL);`
- Deinitialize:
  - `rcb4_command_delete(comm);`
  - `rcb4_deinit(conn);`

Read sensor values

- Initialize:
  - `conn = rcb4_init(“/dev/ttyUSB0”);`
  - `uint16_t value;`
    - Needs at least 2 bytes (unsigned short, int, ...)
- Call the helper function:
  - `rcb4_ad_read(conn, SensorID, &value);`
    - It will return 0 on success and save it in the variable *value*.
    - *SensorID* is an integer from 0 to 10.
- Deinitialize:
  - `rcb4_command_delete(comm);`

Read/Write data

- Source
  - RAM
  - ICS
  - Literal
  - ROM
- Destination
  - RAM
  - ICS
  - COM
  - ROM

- `RCB4_COMM_MOV`
- `rcb4_command_set_src_*()`
  - Sets the source.
- `rcb4_command_set_dst_*()`
  - Sets the destination.
- User RAM address:
  - From 0x0090 to 0x048E
- Check *rcb4_cheatsheet.ods* if you want to read/write in other locations. Some are read-only!

Special commands

- `rcb4_jmp`
  - Jumps to address (assembly: JMP)
- `rcb4_call`
  - Calls a function in address (assembly: CALL)
- `rcb4_ret`
  - Returns from a function (assembly: RET)
- Jump and call accept a condition flag (execute only if the zero and/or carry flags are set/clear)
- `rcb4_command_debug_print`
  - Shows the hexadecimal dump of the command to be sent.
- `rcb4_util_usleep`
  - Sleeps for that amount of microseconds.