Contents

1 Data Structure Index ........................................... 1

1.1 Data Structures ........................................... 1

2 File Index ..................................................... 3

2.1 File List ..................................................... 3

3 Data Structure Documentation .................................... 5

3.1 bth_inquiry_record Struct Reference ......................... 5

3.2 GLimageset Struct Reference .................................. 5

3.2.1 Detailed Description ....................................... 6

3.3 image_timing_data Struct Reference .......................... 6

3.3.1 Detailed Description ....................................... 6

3.4 motion_timing_data Struct Reference ......................... 7

3.4.1 Detailed Description ....................................... 7

3.5 my_error_mgr Struct Reference ................................ 7

3.6 Picture Struct Reference ..................................... 7

3.6.1 Detailed Description ....................................... 8

3.7 Pixel Struct Reference ....................................... 8

3.7.1 Detailed Description ....................................... 9

3.8 robot_motion_camera_record Struct Reference ............... 9

3.8.1 Detailed Description ....................................... 9

3.9 timing_thread_info Struct Reference ......................... 9

3.9.1 Detailed Description ....................................... 9
# File Documentation

## 4.1 MyroC-image-file.c File Reference

### 4.1.1 Detailed Description

### 4.1.2 Function Documentation

- **4.1.2.1 rLoadPicture()**
- **4.1.2.2 rSavePicture()**

## 4.2 MyroC.h File Reference

### 4.2.1 Detailed Description

### 4.2.2 Function Documentation

- **4.2.2.1 rBackward()**
- **4.2.2.2 rBeep()**
- **4.2.2.3 rBeep2()**
- **4.2.2.4 rConnect()**
- **4.2.2.5 rDisconnect()**
- **4.2.2.6 rDisplayImageProcessingErrors()**
- **4.2.2.7 rDisplayPicture()**
- **4.2.2.8 rFastForward()**
- **4.2.2.9 rFinishProcessing()**
- **4.2.2.10 rForward()**
- **4.2.2.11 rGetBattery()**
- **4.2.2.12 rGetBrightAll()**
- **4.2.2.13 rGetBrightTxt()**
- **4.2.2.14 rGetForwardness()**
- **4.2.2.15 rGetInfo()**
- **4.2.2.16 rGetIRAll()**
- **4.2.2.17 rGetIRTxt()**
- **4.2.2.18 rGetLightsAll()**
- **4.2.2.19 rGetLightTxt()**
- **4.2.2.20 rGetLine()**
- **4.2.2.21 rGetName()**
4.2.2.22 rGetObstacleAll() ......................................................... 35
4.2.2.23 rGetObstacleTxt() ....................................................... 36
4.2.2.24 rGetStall() ................................................................. 38
4.2.2.25 rHardStop() ................................................................. 39
4.2.2.26 rLoadPicture() ............................................................ 39
4.2.2.27 rMotors() ................................................................. 40
4.2.2.28 rSavePicture() ............................................................ 40
4.2.2.29 rSetBluetoothEcho() ................................................... 42
4.2.2.30 rSetConnection() .......................................................... 42
4.2.2.31 rSetForwardness() ....................................................... 43
4.2.2.32 rSetForwardnessTxt() ................................................. 43
4.2.2.33 rSetIRPower() ............................................................ 44
4.2.2.34 rSetLEDBack() ........................................................... 44
4.2.2.35 rSetLEDFront() ........................................................... 45
4.2.2.36 rSetName() ............................................................... 45
4.2.2.37 rSetVolume() ............................................................. 46
4.2.2.38 rStop() ................................................................. 46
4.2.2.39 rTakePicture() .......................................................... 46
4.2.2.40 rTurnLeft() .............................................................. 47
4.2.2.41 rTurnRight() ............................................................. 48
4.2.2.42 rTurnSpeed() ............................................................ 49
4.2.2.43 rWaitTimedImageDisplay() ......................................... 49

Index 51

Generated by Doxygen
Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

- bth_inquiry_record .................................................. 5
- GLImageset .................................................................. 5
- image_timing_data ..................................................... 6
- motion_timing_data ................................................... 7
- my_error_mgr ............................................................. 7
- Picture Struct for a picture object ................................. 7
- Pixel Struct for a pixel ................................................ 8
- robot_motion_camera_record ...................................... 9
- timing_thread_info .................................................... 9
## Chapter 2

### File Index

#### 2.1 File List

Here is a list of all documented files with brief descriptions:

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyroC-image-file.c</td>
<td>Implementation file for saving/loading jpeg images to/from a file</td>
<td>11</td>
</tr>
<tr>
<td>MyroC-utilities-with-getcurrentDirectory.h</td>
<td></td>
<td>??</td>
</tr>
<tr>
<td>MyroC-utilities.h</td>
<td></td>
<td>??</td>
</tr>
<tr>
<td>MyroC.h</td>
<td>Header for a C-based, my-robot package for the Scribbler 2</td>
<td>15</td>
</tr>
</tbody>
</table>
Chapter 3

Data Structure Documentation

3.1 bth_inquiry_record Struct Reference

Data Fields

- char `addrStr` [INQUIRY_STR_MAX]
- char `serviceName` [INQUIRY_STR_MAX]

The documentation for this struct was generated from the following file:

- MyroC-connect-windows.c

3.2 GLimageset Struct Reference

Collaboration diagram for GLimageset:
Data Fields

- `char * window_title`
- `pid_t display_pid`
- `int socket_id [2]`
- `pthread_mutex_t lock`
- `GLuint texture_objs [1]`
- `int height`
- `int width`
- `int process_status`
- `int image_num`
- `Pixel * pixs`
- `HANDLE toImgHandWr`
- `HANDLE fromImgHandRd`
- `PROCESS_INFORMATION procInfo`

### 3.2.1 Detailed Description

Struct for recording details of openGL process

The documentation for this struct was generated from the following files:

- `MyroC-display-linux-mac.c`
- `MyroC-display-windows-interface-manager.c`
- `MyroC-display.c`

### 3.3 image_timing_data Struct Reference

Data Fields

- `double delay_time`
- `int GLimageset_index`
- `int image_seq_number`
- `int thread_seq_number`

### 3.3.1 Detailed Description

Struct for timing display of image

The documentation for this struct was generated from the following files:

- `MyroC-display-linux-mac.c`
- `MyroC-display-windows-interface-manager.c`
- `MyroC-display-windows-opengl-img-handler.c`
- `MyroC-display.c`
3.4 motion_timing_data Struct Reference

#include <MyroC-utilities-with-getcurrentDirectory.h>

Data Fields
  - double duration_time
  - SOCKET_TYPE robot_socket_num
  - int motion_seq_num

3.4.1 Detailed Description

Struct for timing motion commands of given robot

The documentation for this struct was generated from the following files:
  - MyroC-utilities-with-getcurrentDirectory.h
  - MyroC-utilities.h

3.5 my_error_mgr Struct Reference

Data Fields
  - struct jpeg_error_mgr pub
  - jmp_buf setjmp_buffer

The documentation for this struct was generated from the following files:
  - MyroC-image-file-linux-mac.c
  - MyroC-image-file.c

3.6 Picture Struct Reference

Struct for a picture object.
#include <MyroC.h>

Collaboration diagram for Picture:
Data Fields

- **int height**
  *The actual height of the image, but no more than 266.*

- **int width**
  *The actual width of the image, but no more than 427.*

- **Pixel pix_array [266][427]**
  *The array of pixels comprising the image.*

3.6.1 Detailed Description

Struct for a picture object.

**Note**

Images from robot cameras have varying sizes, depending on the Fluke.
- *pix_array* is sufficiently large to accommodate any Fluke version.
- Images for the original Fluke are 192 (height) by 256 (width).
- Low-resolution images for the Fluke 2 are 266 by 427.
- High-resolution images (e.g., 800 by 1280) are not practical, due to memory constraints and thus are not available in MyroC.
- User-defined images may have any size, as long as height \( \leq 266 \) and width \( \leq 427 \).

Following standard mathematical convention for a 2D matrix, all references to a pixel are given within an array as \([\text{row}][\text{col}]\).

**Warning**

The Picture struct is defined to be sufficiently large to store several low-resolution camera images (340756 bytes each). Experimentally, an array of up to 94 (not 95) Pictures is allowed. However, the display of images requires that image data be copied, so display of many images may not work if a program hangs when working with Picture variables, the issue may involve lack of space on the runtime stack. To utilize a modest number of Pictures, use "ulimit -s" command, as needed, in a terminal window. For example, ulimit -s 32768. Sizes above 32768 may not be allowed in Linux or Mac OS X.

The documentation for this struct was generated from the following file:

- **MyroC.h**

3.7 Pixel Struct Reference

Struct for a pixel.

#include <MyroC.h>

Data Fields

- **unsigned char R**
  *The value of the red component.*

- **unsigned char G**
  *The value of the green component.*

- **unsigned char B**
  *The value of the blue component.*
3.8 robot_motion_camera_record Struct Reference

3.7.1 Detailed Description

Struct for a pixel.

The documentation for this struct was generated from the following file:

- MyroC.h

3.8 robot_motion_camera_record Struct Reference

#include <MyroC-utilities-with-getcurrentDirectory.h>

Data Fields

- SOCKET_TYPE robot_socket_num
- int camera_height
- int camera_width
- char * fluke_type
- int motion_seq_num
- pthread_t thread_motion_timer_id

3.8.1 Detailed Description

Struct for recording number of motion commands by a robot

The documentation for this struct was generated from the following files:

- MyroC-utilities-with-getcurrentDirectory.h
- MyroC-utilities.h

3.9 timing_thread_info Struct Reference

Data Fields

- int thread_seq_number
- pthread_t pthread

3.9.1 Detailed Description

Struct for acting timing-thread information

The documentation for this struct was generated from the following files:

- MyroC-display-linux-mac.c
- MyroC-display-windows-interface-manager.c
- MyroC-display-windows-opengl-img-handler.c
- MyroC-display.c
Chapter 4

File Documentation

4.1  MyroC-image-file.c File Reference

Implementation file for saving/loading jpeg images to/from a file.

```c
#include "MyroC.h"
#include "MyroC-utilities.h"
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <jpeglib.h>
#include <setjmp.h>
#include <pthread.h>
```

Include dependency graph for MyroC-image-file.c:

Data Structures

- struct my_error_mgr

Typedefs

- typedef struct my_error_mgr * my_error_ptr
Functions

- `void my_error_exit (j_common_ptr cinfo)`
- `void rSavePicture (Picture *pic, char *filename)`
  *Save a Picture to a .jpeg.*
- `Picture rLoadPicture (char *filename)`
  *Load a picture from a .jpeg file.*

4.1.1 Detailed Description

Implementation file for saving/loading jpeg images to/from a file.

Revision History

Version 1.0 based on a C++ package by April O'Neill, David Cowden, Dilan Ustek, Erik Opavsky, and Henry M. Walker


Revisions for Version 3.0 Henry M. Walker

C ported to Macintosh Linux/Mac differences required for connections — otherwise same code OpenGL used to display images, replacing ImageMagick same [new] code used for both Linux and Macintosh 1 process for robot control 1 process needed for each titled window (not each image, as in 2.2-2.4) Blocking options (negative duration parameter) utilize separate thread timer MyroC implementation files organized by user function as follows:

Revisions for Version 3.1 Henry M. Walker

Picture struct and image functions revised to allow 192 by 256 images from origial Fluke camera 266 x 427 low-resolution images from Fluke 2 (high-resolution (800 x 1280) too large for more than 2-4 variables on run-time stack) storage, retrieval, and display of any images up to 266 x 427

This program and all MyroC software is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License. Details may be found at [http://creativecommons.org/licenses/by-nc-sa/3.0/us/](http://creativecommons.org/licenses/by-nc-sa/3.0/us/)

Revisions for Version 3.2 Henry M. Walker

Practical range of rBeep duration identified as < 3.0014 seconds Image display and processing resolves several matters and adds functionality function rDisplayPicture completely rewritten over time OpenGL routines had encountered troubles on Linux, specifically glutHideWindow() hid images, but could not be restored, with difficulties depending on the graphics card on Mac, High Sierra generated compile warnings and restricted threads that could display images function rDisplayImageProcessingErrors added function rWaitTimedImageDisplay updated substantially

Revisions for Version 3.3 Henry M. Walker

MyroC ported to Windows 10, in addition to Linux and Mac OS X Additional refinements text parameter for rGetLightTxt expanded to "Middle" as well as "Center"

This program and all MyroC software is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License. Details may be found at [http://creativecommons.org/licenses/by-nc-sa/3.0/us/](http://creativecommons.org/licenses/by-nc-sa/3.0/us/)
implementation overview

indicates functions implemented in this file

0. LOW-LEVEL UTILITY (defined in MyroC-utilities.h implemented in MyroC-utilities.c) rSend rReceive rSet→ OpeningExchange rSetBluetoothEcho rSetEchoMode rCheckHardwareVersionSetCameraSize

1. GENERAL (mostly MyroC-general.c) 2. SENSOR (MyroC-sensors.c) rConnect (MyroC-connect.c) a← Scribbler Sensors rDisconnect (MyroC-connect.c) rGetLightsAll rSetConnection rGetLightTxt rFinish← Processing rGetRAAll rSetVolume rGetIRTxt rBeep rGetLine rBeep2 rSetName b.Fluke Sensors rGet← Name rGetObstacleAll rSetForwardness rGetObstacleTxt rGetForwardness rGetBrightAll rSetLEDFront r← GetBrightTxt rSetLEDBack rSetRPower rGetBattery rGetStall

Note:
MyroC-connect requires special
knowledge of a Mac or Linux
environment, yielding two versions:
MyroC-connect-mac.c
MyroC-connect-linux.c

1. MOVEMENT (MyroC-movement.c) 4. PICTURES (files as indicated) rTurnLeft rGetCameraSize (MyroC-camera.c) rTurnRight rTakePicture (MyroC-camera.c) rTurnSpeed rDisplayPicture (MyroC-display.c) rForward rWaitTimedImageDisplay(MyroC-display.c) rFastForward > rSavePicture (MyroC-image-file.c) rBackward > rLoadPicture (MyroC-image-file.c) rMotors rStop rHardStop

4.1.2 Function Documentation

4.1.2.1 rLoadPicture()

Picture rLoadPicture ( char * filename )

Load a picture from a .jpeg file.

Parameters

| filename | the name of the file |

Precondition

file must exist
file extension must be .jpeg or .jpg
file must be no larger than 266 by 427

Returns

locally-declared Picture

Generated by Doxygen
Postcondition

calling program should store returned Picture in a previously-declared struct

4.1.2.2 rSavePicture()

void rSavePicture (  
    Picture * pic,  
    char * filename  
)

Save a Picture to a .jpeg.

This section contains functions for taking and manipulating images. All images are constrained with height $\leq 266$ and width $\leq 427$

images from robot cameras have varying sizes, depending on the Fluke images for the original Fluke are 192 (height) by 256 (width) low-resolution images for the Fluke 2 are 266 by 427 high-resolution images for the fluke 2 are 800 by 1280 (not supported)

Bluetooth communication constrains the time required for the Fluke to take a picture. Typical times: original fluke: 2-4 seconds Fluke 2 (low res): 4-6 seconds Fluke 2 (high res): 25-30 seconds

BASED ON TIMINGS AND MEMORY CONSIDERATIONS, Myro C ALLOWS ONLY LOW RESOLUTION IMAGES

the Picture struct allows direct access to Pixel data. Pictures can be saved and loaded as .jpeg files

Note

Following standard mathematical convention for a 2D matrix, all references to a pixel are given within an array as [row][col]

user-defined images may have any size, as long as height $\leq 266$ and width $\leq 427$

Following standard mathematical convention for a 2D matrix, all references to a pixel are given within an array as [row][col]

Warning

The Picture struct is defined to be sufficiently large to store several low-resolution camera images (340756 bytes each). Experimentally, an array of up to 94 (not 95) Pictures is allowed. However, the display of images requires that image data be copied, so display of many images may not work. If a program hangs when working with Picture variables, the issue may involve lack of space on the runtime stack. To utilize a modest number of Pictures, use "ulimit -s" command, as needed, in a terminal window. For example, ulimit -s 32768. Sizes above 32768 may not be allowed in Linux or Mac OS X. Save a Picture to a .jpeg

Parameters

<table>
<thead>
<tr>
<th>pic</th>
<th>pointer to an RGB picture struct from Scribbler 2 camera</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>the name of the file</td>
</tr>
</tbody>
</table>
4.2 MyroC.h File Reference

Header for a C-based, my-robot package for the Scribbler 2.

This graph shows which files directly or indirectly include this file:

```
MyroC.h
MyroC-image-file.c
```

Data Structures

- `struct Pixel`
  
  Struct for a pixel.

- `struct Picture`
  
  Struct for a picture object.

Macros

- `#define ROBOT_CONNECTION_TYPE` int

Precondition

  filename ends with .jpeg or .jpg.

Postcondition

  If the file does not exist, a new file will be created.
  If the file exists, the file will be overwritten.

this code is a slightly modified version of write_JPEG_file(char *, int) from example.c from the libjpeg.v8d library directory.
Functions

- **ROBOT_CONNECTION_TYPE rConnect (const char *address)**
  
  connects program to Scribbler

- **void rDisconnect ()**
  
  stop Scribbler motion and close Bluetooth

- **void rSetConnection (ROBOT_CONNECTION_TYPE new_socket_num)**
  
  set current connection to the socket number

- **void rFinishProcessing ()**
  
  all timed motions/image displays completed, all robots stopped, and all robot Bluetooth connections closed

- **void rSetVolume (char highMute)**
  
  Set sound to high volume (H) or mute (M) highMute set volume of Scribbler.

- **void rBeep (double duration, int frequency)**
  
  Beeps with the given duration and frequency.

- **void rBeep2 (double duration, int freq1, int freq2)**
  
  Generates two notes for the prescribed duration.

- **void rSetName (const char *name)**
  
  Change name stored in the robot to the 16-byte name given.

- **const char * rGetName ()**
  
  Get the name of the robot.

- **void rSetForwardness (char *direction)**
  
  specifies which end of the Scribbler is considered the front

- **void rSetForwardnessTxt (char *direction)**
  
  alternative to rSetForwardness for compatibility with earlier MyroC

- **char * rGetForwardness ()**
  
  Gets the forwardness of the Scribbler.

- **void rSetLEDFront (int led)**
  
  Set the front [fluke] LED on or off.

- **void rSetLEDBack (double led)**
  
  Set the the intensity of the back fluke LED.

- **double rGetBattery ()**
  
  Get the current voltage from the Scribbler batteries; Maximum charge from 6 batteries could be up to 6 volts; Manufacturer suggests batteries should be changed below 4.1 volts.

- **int rGetStall (int sampleSize)**

- **void rSetBluetoothEcho (char onOff)**
  
  Turn on and off the printing transmission and echong of Bluetooth transmissions All robot commands involve the transmission of a command over Bluetooth Scribbler commands are always 9 bytes Fluke commands have varying lengths The fluke echos most, but not all, of the commands For many commands, the fluke also echos 11 bytes of sensor data.

- **void rGetLightsAll (int lightSensors[3], int sampleSize)**
  
  Get the average values of each of the three light sensors in an array. Values of each light sensor can somewhat (typically under 5%-10%). To even out variability, the sensor can be queried sampleSize times and an average obtained.

- **int rGetLightTxt (const char *sensorName, int sampleSize)**
  
  Get the average values of a specified light sensor. Values of each light sensor can somewhat (typically under 5%-10%). To even out variability, the sensor can be queried sampleSize times and an average obtained.

- **void rGetIRAll (int irSensors[2], int sampleSize)**
  
  Get an array of true/false values regarding the presence of obstacle based on the average values of each of the three IR sensors. Since readings of each light sensor can vary substantially, each sensor can be queried sampleSize times and an average obtained.

- **int rGetIRTxt (const char *sensorName, int sampleSize)**

Generated by Doxygen
Use specified IR sensor to determine if obstacle is present. Since values of each light sensor can vary substantially, the sensor can be queried sampleSize times and an average obtained.

- void rGetLine (int lineSensors[2], int sampleSize)
  Use Scribbler 2 line sensors of Scribbler to check for a black line on a white surface under the robot. Since values of each light sensor can vary substantially (successive readings may differ by several hundred or more), the sensor can be queried sampleSize times and an average obtained.

- void rSetIRPower (int power)
  Set the amount of power for the dongle’s IR sensors.

- void rGetObstacleAll (int obstSensors[3], int sampleSize)
  Get the average values of the three obstacle sensors in an array. Since readings of each obstacle sensor can vary substantially (successive readings may differ by several hundred or more), each sensor can be queried sampleSize times and an average obtained.

- int rGetObstacleTxt (const char *sensorName, int sampleSize)
  Get the average values of a specified obstacle (IR) sensor. Since values of each obstacle sensor can vary substantially (successive readings may differ by several hundred or more), the sensor can be queried sampleSize times and an average obtained.

- void rGetBrightAll (int brightSensors[3], int sampleSize)
  Read the Fluke’s virtual light sensors. Since readings of each brightness sensor can vary substantially (successive readings may differ by 5000-10000), each sensor can be queried sampleSize times and an average obtained.

- int rGetBrightTxt (char *sensorName, int sampleSize)
  Reads one of the Fluke’s virtual light sensors. Each sensor reports a total intensity in the left, middle, or right of the Fluke’s camera. Since values of each obstacle sensor can vary substantially (successive readings may differ by 5000-10000), the sensor can be queried sampleSize times and an average obtained.

- void rGetInfo (char *infoBuffer)
  returns information about the robot’s dongle, firmware, and communication mode as a 60 character array in infoBuffer.

- void rTurnLeft (double speed, double time)
  turn Scribbler left for a specified time and speed

- void rTurnRight (double speed, double time)
  turn Scribbler right for a specified time and speed

- void rTurnSpeed (char *direction, double speed, double time)
  turn Scribbler in direction for a specified time and speed

- void rForward (double speed, double time)
  moves Scribbler forward for a specified time and speed

- void rFastForward (double time)
  moves Scribbler forward at the largest possible speed for a specified time

- void rBackward (double speed, double time)
  moves Scribbler backward for a specified time and speed

- void rMotors (double leftSpeed, double rightSpeed)
  move robot with given speeds for the left and right motors continues until given another motion command or disconnected (non-blocking)

- void rStop ()
  directs robot to stop movement

- void rHardStop ()
  cuts power to the motor of the robot

- Picture rTakePicture ()
  Use the camera to take a photo.

- void rSavePicture (Picture *pic, char *filename)
  Save a Picture to a .jpeg.

- Picture rLoadPicture (char *filename)
  Load a picture from a .jpeg file.

- void rDisplayPicture (Picture *pic, double duration, const char *windowTitle)
  Display a picture in a new window.

- void rDisplayImageProcessingErrors ()
  Turn on and off the printing of error message for image processing.

- void rWaitTimedImageDisplay ()
  Wait until all timed, non-blocking image window timers are complete.
4.2.1 Detailed Description

Header for a C-based, my-robot package for the Scribbler 2.

Revision History

Version 1.0 based on a C++ package by April O'Neill, David Cowden, Dilan Ustek, Erik Opavsky, and Henry M. Walker


Revisions for Version 3.0 Henry M. Walker

C ported to Macintosh Linux/Mac differences required for connections — otherwise same code OpenGL used to display images, replacing ImageMagick same [new] code used for both Linux and Macintosh 1 process for robot control 1 process needed for each titled window (not each image, as in 2.2-2.4) Blocking options (negative duration parameter) utilize separate thread timer MyroC implementation files organized by user function as follows:

Revisions for Version 3.1 Henry M. Walker

Picture struct and image functions revised to allow 192 by 256 images from origial Fluke camera 266 x 427 low-resolution images from Fluke 2 (high-resolution (800 x 1280) too large for more than 2-4 variables on run-time stack) storage, retrieval, and display of any images up to 266 x 427

Revisions for Version 3.2 Henry M. Walker

Practical range of rBeep duration identified as <= 3.0014 seconds Image display and processing resolves several matters and adds functionality function rDisplayPicture completely rewritten over time OpenGL rountines had encountered troubles on Linux, specifically glutHideWindow() hid images, but could not be restored, with difficulties depending on the graphics card on Mac, High Sierra generated compile warnings and restricted threads that could display images function rDisplayImageProcessingErrors added function rWaitTimedImageDisplay updated substantially

Revisions for Version 3.3 Henry M. Walker

MyroC ported to Windows 10, in addition to Linux and Mac OS X Additional refinements text parameter for rGetLightTxt expanded to "Middle" as well as "Center"

This program and all MyroC software is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License. Details may be found at http://creativecommons.org/licenses/by-nc-sa/3.0/us/

4.2.2 Function Documentation

4.2.2.1 rBackward()

```c
void rBackward(
    double speed,
    double time
)
```

moves Scribbler backward for a specified time and speed
4.2 MyroC.h File Reference

Parameters

| speed | the rate at which the robot should move backward linear range: -1.0 specifies move forward at full speed 0.0 specifies no forward/backward movement 1.0 specifies move backward at full speed |
| time | specifies the duration of the turn if negative: the robot starts backward (non-blocking) other processing proceeds, and the robot continues backward until given another motion command or disconnected (non-blocking) if zero: robot starts moving backward (non-blocking) other processing proceeds if positive: robot moves backward for the given duration, in seconds |

moves Scribbler backward for a specified time and speed

Parameters

| speed | the rate at which the robot should move backward linear range: -1.0 specifies move forward at full speed 0.0 specifies no forward/backward movement 1.0 specifies move backward at full speed |
| time | specifies the duration of the turn if negative: the robot continues to go backward until given another motion command or disconnected (non-blocking) if nonnegative: robot moves backward for the given duration, in seconds |

4.2.2.2 rBeep()

void rBeep ( double duration, int frequency )

Beeps with the given duration and frequency.

Parameters

| duration | length of note in seconds |
| frequency | frequency of pitch in cycles per second (hertz) |

Precondition

0.0 < duration > 0.0 <= 3.0 seconds (Scribbler 2 hangs if duration > 3.0014 seconds; although reason unknown, experimentation suggests part of Scribbler 2 firmware)

4.2.2.3 rBeep2()

void rBeep2 ( double duration, int freq1, int freq2 )

Generates two notes for the prescribed duration.

Generated by Doxygen
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>duration</td>
<td>length of note in seconds</td>
</tr>
<tr>
<td>freq1</td>
<td>frequency of first pitch in cycles per second (hertz)</td>
</tr>
<tr>
<td>freq2</td>
<td>frequency of second pitch in cycles per second (hertz)</td>
</tr>
</tbody>
</table>

Precondition

0.0 < duration <= 3.0 seconds (Scribbler 2 hangs if duration > 3.0014 seconds; although reason unknown, experimentation suggests part of Scribbler 2 firmware)

4.2.2.4 rConnect()

ROBOT_CONNECTION_TYPE rConnect (const char * address)

connects program to Scribbler

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>address</td>
<td>string, giving name of workstation port or a Scribbler Bluetooth designation</td>
</tr>
</tbody>
</table>

several string formats are possible

Linux and Mac:
a communications port, such as "/dev/rfcomm0"
a Scribbler 2 fluke serial number, such as "245787"
a full IPRE serial number, such as "IPRE245787"
a Fluke 2 serial number (hexadecimal), such as "021F"
a full Fluke 2 serial number, such as "Fluke2-021F"

Linux only:
a MAC address, such as "00:1E:19:01:0E:13"

Mac only:
any substring of a complete device file name, as long as the resulting device is unique
some possibilities include
a complete device file name, such as

- "/dev/tty.IPRE6-365877-DevB"
- "/dev/tty.Fluke2-0958-Fluke2"
  a fluke or fluke2 serial number
  full path of symbolic link to a device filename string or substring in /dev

uniqueness is ensured by requiring 4 hex digits
or 6 decimal digits
Returns

if a connection is made the socket number/id of communications port is returned otherwise, the program exits, as robot processing cannot continue

Note

on Linux and Mac computers, sockets are ints, so rConnect returns an int on Windows computers, sockets have type SOCKET, so rConnect returns a SOCKET together, these are defined as a ROBOT_CONNECT

ON_TYPE

Postcondition

subsequent communications will take place through this robot address, unless changed by rSetConnection

connects program to Scribbler

Parameters

| address | string, giving name of workstation port or a Scribbler Bluetooth designation |

several string formats are possible a communications port, such as "/dev/rfcomm0" a MAC address, such as "00:1E:19:01:0E:13" a Scribbler 2 fluke serial number, such as "245787" a full IPRE serial number, such as "IPRE245787" a Fluke 2 serial number (hexadecimal), such as "021F" a full Fluke 2 serial number, such as "Fluke2-021F"

Returns

if a connection is made the socket number of communications port is returned otherwise, the program exits, as robot processing cannot continue

Postcondition

subsequent communications will take place through this socket, unless changed by rSetConnection

Parameters

| address | string, giving name of workstation port or a Scribbler Bluetooth designation |

several string formats are possible a complete device file name, such as

- "/dev/rfcomm0"
- "/dev/tty.IPRE6-365877-DevB"
- "/dev/tty.Fluke2-0958-Fluke2" any substring of a complete device file name, as long as the resulting device is unique some possibilities include
  - a Scribbler 2 fluke serial number, such as "245787"
  - a full IPRE serial number, such as "IPRE6-245787"
  - a Fluke 2 serial number (hexadecimal), such as "021F"
• a full Fluke 2 serial number, such as "Fluke2-021F" uniqueness is ensured by requiring 4 hex digits or 6 decimal digits the full path of a symbolic link to a device file name string or substring in /dev

Note: Connections via a MAC address, such as "00:1E:19:01:0E:13", is NOT available, for MyroC on Mac OS X, since AF_BLUETOOTH socket types are not defined in /usr/include/sys/socket.h Type man socket for additional information

Returns

if a connection is made the socket number of communications port is returned otherwise, the program exits, as robot processing cannot continue

Postcondition

subsequent communications will take place through this socket, unless changed by rSetConnection

4.2.2.5 rDisconnect()

void rDisconnect()

stop Scribbler motion and close Bluetooth

Postcondition

motion for the current robot is stopped, blocking until any non-blocking motion time has expired i.e., if a motion timer is set, this procedure blocks when the timer completes, then the motion stops else, procedure stops motion immediately Bluetooth for the current robot is closed motion for the current robot is stopped, blocking until any non-blocking motion time has expired i.e., if a motion timer is set, this procedure blocks, then stops motion else, procedure stops motion immediately Bluetooth for the current robot is closed

4.2.2.6 rDisplayImageProcessingErrors()

void rDisplayImageProcessingErrors()

Turn on and off the printing of error message for image processing.

Note

Some operating systems (e.g., Mac OX S High Sierra) display an error message whenever a new process for image display starts
4.2.2.7 rDisplayPicture()

void rDisplayPicture (  
    Picture * pic,  
    double duration,  
    const char * windowTitle )

Display a picture in a new window.

Parameters

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>pic</strong></td>
<td>pointer to an RGB picture struct from Scribbler 2 camera</td>
</tr>
</tbody>
</table>
| **duration** | if duration > 0, operation is blocking  
                  if duration <= 0, operation is non-blocking  
                  for duration != 0, picture displayed for abs(duration) seconds or until picture closed manually or until the program terminates  
                  if duration == 0, picture displayed until closed manually |
| **windowTitle** | The title of the window that appears. white spaces will be replaced with underscores. |

Precondition

windowTitle is less than 100 characters.
Postcondition

    image is displayed for the duration specified,
    EXCEPT all display windows are closed when the main program terminates.

Warning

    If images are displayed with a non-blocking option, and if the user wants images to appear for a full duration,
    use rWaitTimedImageDisplay or rFinishProcessing to block processing until all image timers are finished. Otherwise, program termination may close windows prematurely.

Display a picture.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pic</td>
<td>pointer to an RGB picture struct from Scribbler 2 camera</td>
</tr>
</tbody>
</table>
| duration  | if duration > 0, operation is blocking
            if duration <= 0, operation is non-blocking
            for duration != 0, picture displayed for abs(duration) seconds or until picture closed manually or until the program terminates
            if duration == 0, picture displayed until closed manually |
| windowTitle | The title of the window that appears. white spaces will be replaced with underscores. |

Precondition

    windowTitle is less than 100 characters.

Postcondition

    image is displayed for the duration specified,
    EXCEPT all display windows are closed when the main program terminates.

Warning

    If images are displayed with a non-blocking option, and if the user wants images to appear for a full duration,
    use rWaitTimedImageDisplay or rFinishProcessing to block processing until all image timers are finished. Otherwise, program termination may close windows prematurely.

4.2.2.8 rFastForward()

    void rFastForward ( double time )

    moves Scribbler forward at the largest possible speed for a specified time
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>time</code></td>
<td>Specifies the duration of the turn. If negative, the robot starts forward (non-blocking) other processing proceeds, and the robot continues forward until given another motion command or disconnected (non-blocking). If zero, the robot starts moving forward (non-blocking); other processing proceeds. If positive, the robot moves forward for the given duration, in seconds.</td>
</tr>
</tbody>
</table>

**Warning**

- may take longer than usual to execute

**Moves Scribbler forward at the largest possible speed for a specified time**

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>time</code></td>
<td>Specifies the duration of movement. If negative, the robot continues to move forward until given another motion command or disconnected (non-blocking). If nonnegative, the robot moves forward for the given duration, in seconds.</td>
</tr>
</tbody>
</table>

**Warning**

- may take longer than usual to execute

### 4.2.2.9 rFinishProcessing()

```cpp
void rFinishProcessing ()
```

All timed motions/image displays completed, all robots stopped, and all robot Bluetooth connections closed.

Same result as rDisconnect for all robots plus rCompleteImageDisplay.

**Postcondition**

- Blocks until all timed robot motions are complete, and all timed image displays no longer visible.
- Upon completion, all timed images are invisible, all robot motion is halted, and all robot Bluetooth connections closed.

**Produces same result as calling rDisconnect for all robots and also calling rCompleteImageDisplay**

**Postcondition**

- Procedure blocks until all timed robot motions are complete, and all timed image displays no longer visible.
- Upon completion, all timed image displays are closed, all robot motion is halted, and all robot Bluetooth connections closed.

### 4.2.2.10 rForward()

```cpp
void rForward (
    double speed,
    double time)
```

Moves Scribbler forward for a specified time and speed.
**Parameters**

<table>
<thead>
<tr>
<th>speed</th>
<th>the rate at which the robot should move forward</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>linear range: -1.0 specifies move backward at full speed</td>
</tr>
<tr>
<td></td>
<td>0.0 specifies no forward/backward movement</td>
</tr>
<tr>
<td></td>
<td>1.0 specifies move forward at full speed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>time</th>
<th>specifies the duration of the turn</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>if negative: the robot starts forward (non-blocking) other processing proceeds, and the robot continues forward until given another motion command or disconnected (non-blocking)</td>
</tr>
<tr>
<td></td>
<td>if zero: robot starts moving forward (non-blocking); other processing proceeds</td>
</tr>
<tr>
<td></td>
<td>if positive: robot moves forward for the given duration, in seconds</td>
</tr>
</tbody>
</table>

moves Scribbler forward for a specified time and speed

**Parameters**

<table>
<thead>
<tr>
<th>speed</th>
<th>the rate at which the robot should move forward</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>linear range: -1.0 specifies move backward at full speed</td>
</tr>
<tr>
<td></td>
<td>0.0 specifies no forward/backward movement</td>
</tr>
<tr>
<td></td>
<td>1.0 specifies move forward at full speed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>time</th>
<th>specifies the duration of movement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>if negative: robot continues to move forward until given another motion command or disconnected (non-blocking) if nonnegative: robot moves forward for the given duration, in seconds</td>
</tr>
</tbody>
</table>

---

### 4.2.2.11 rGetBattery()

```c
double rGetBattery ( )
```

Get the current voltage from the Scribbler batteries; Maximum charge from 6 batteries could be up to 6 volts; Manufacturer suggests batteries should be changed below 4.1 volts.

**Returns**

percentage of battery voltage

---

### 4.2.2.12 rGetBrightAll()

```c
void rGetBrightAll ( int brightSensors[3],
                    int sampleSize )
```

Read the Fluke's virtual light sensors. Since readings of each brightness sensor can vary substantially (successive readings may differ by 5000-10000), each sensor can be queried sampleSize times and an average obtained.
Parameters

<table>
<thead>
<tr>
<th>brightSensors</th>
<th>array to store intensity values</th>
</tr>
</thead>
<tbody>
<tr>
<td>sampleSize</td>
<td>how many readings are taken for each sensor</td>
</tr>
</tbody>
</table>

Precondition

space already allocated for brightSensors array sampleSize > 0

Postcondition

brightSensors[0] gives average value for left sensor
brightSensors[1] gives average value for middle sensor
brightSensors[2] gives average value for right sensor
Brightness values near 0 represent bright light
Brightness values may extend to about 65535 for a dark region.

Read the Fluke's virtual light sensors. Since readings of each brightness sensor can vary substantially (successive readings may differ by 5000-10000), each sensor can be queried sampleSize times and an average obtained.

Parameters

<table>
<thead>
<tr>
<th>brightSensors</th>
<th>array to store intensity values</th>
</tr>
</thead>
<tbody>
<tr>
<td>sampleSize</td>
<td>how many readings are taken for each sensor</td>
</tr>
</tbody>
</table>

Precondition

space already allocated for brightSensors array sampleSize > 0

Postcondition

brightSensors[0] gives average value for left sensor brightSensors[1] gives average value for middle sensor brightSensors[2] gives average value for right sensor Brightness values near 0 represent bright light Brightness values may extend to about 65535 for a very dark region.

4.2.2.13 rGetBrightTxt()

int rGetBrightTxt ( char * sensorName, int sampleSize )

Reads one of the Fluke's virtual light sensors. Each sensor reports a total intensity in the left, middle, or right of the Fluke's camera Since values of each obstacle sensor can vary substantially (successive readings may differ by 5000-10000), the sensor can be queried sampleSize times and an average obtained.

Parameters

| sensorName | name of the bright sensor |

Generated by Doxygen
Precondition

sensorName is "left", "center", "middle", or "right" (not case sensitive)
designations "center" and "middle" are alternatives for the same bright sensor

Parameters

| sampleSize | how many readings are taken for the sensor |

Precondition

sampleSize > 0

Returns

reading from the specified bright sensor, averaged over sampleSize number of data samples
Brightness values near 0 represent bright light
Brightness values may extend to about 65535 for a very dark region.

Parameters

| sensorName | name of the bright sensor |

Precondition

sensorName is "left", "center", "middle", or "right" (not case sensitive) designations "center" and "middle" are alternatives for the same bright sensor

Parameters

| sampleSize | how many readings are taken for the sensor |

Precondition

sampleSize > 0

Returns

reading from the specified bright sensor, averaged over sampleSize number of data samples
Brightness values near 0 represent bright light
Brightness values may extend to about 65535 for a very dark region.

4.2.2.14 rGetForwardness()

cchar* rGetForwardness()

Gets the forwardness of the Scribbler.
Returns

- either "fluke-forward" or "scribbler-forward"

Gets the forwardness of the Scribbler

Returns

- either "fluke-forward" or "scribbler-forward"

### 4.2.2.15 rGetInfo()

```c
void rGetInfo ( char * infoBuffer )
```

returns information about the robot's dongle, firmware, and communication mode as a 60 character array in infoBuffer.

**Parameters**

- `infoBuffer` a pre-defined, 60-character array

**Postcondition**

- infoBuffer contains relevant robot information

### 4.2.2.16 rGetIRAll()

```c
void rGetIRAll ( int irSensors[2], int sampleSize )
```

Get an array of true/false values regarding the presence of obstacle based on the average values of each of the three IR sensors. Since readings of each light sensor can vary substantially, each sensor can be queried sampleSize times and an average obtained.
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>irSensors</code></td>
<td>array to store intensity values</td>
</tr>
<tr>
<td><code>sampleSize</code></td>
<td>how many readings are taken for each sensor</td>
</tr>
</tbody>
</table>

### Precondition

- space already allocated for `irSensors` array
- `sampleSize` > 0

### Postcondition

- `irSensors[0]` checks obstacle for left sensor
- `irSensors[1]` checks obstacle for right sensor
- for each `irSensors` array value
- return 0 indicates no obstacle detected
- return 1 indicates obstacle detected

Get an array of true/false values regarding the presence of obstacle based on the average values of each of the three IR sensors. Since readings of each light sensor can vary substantially, each sensor can be queried `sampleSize` times and an average obtained.

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>irSensors</code></td>
<td>array to store intensity values</td>
</tr>
<tr>
<td><code>sampleSize</code></td>
<td>how many readings are taken for each sensor</td>
</tr>
</tbody>
</table>

### Precondition

- space already allocated for `irSensors` array
- `sampleSize` > 0

### Postcondition

- `irSensors[0]` checks obstacle for left sensor
- `irSensors[1]` checks obstacle for right sensor
- for each `irSensors` array value
- return 0 indicates no obstacle detected
- return 1 indicates obstacle detected

#### 4.2.17 rGetIRTxt()

```c
int rGetIRTxt ( const char * sensorName, int sampleSize )
```

Use specified IR sensor to determine if obstacle is present. Since values of each light sensor can vary substantially, the sensor can be queried `sampleSize` times and an average obtained.

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sensorName</code></td>
<td>name of the light sensor</td>
</tr>
</tbody>
</table>
Precondition

sensorName is "left" or "right" (not case sensitive)

Parameters

| sampleSize | how many readings are taken for the sensor |

Precondition

sampleSize > 0

Returns

true/false (0/1) determination of obstacle, based on IR sensorName sensor, averaged over sampleSize number of data samples

Postcondition

return 0 indicates no obstacle detected
return 1 indicates obstacle detected

Parameters

| sensorName | name of the light sensor |

Precondition

sensorName is "left" or "right" (not case sensitive)

Parameters

| sampleSize | how many readings are taken for the sensor |

Precondition

sampleSize > 0

Returns

true/false (0/1) determination of obstacle, based on IR sensorName sensor, averaged over sampleSize number of data samples

Postcondition

return 0 indicates no obstacle detected return 1 indicates obstacle detected
4.2.2.18 rGetLightsAll()

```c
void rGetLightsAll ( int lightSensors[3], int sampleSize )
```

Get the average values of each of the three light sensors in an array. Values of each light sensor can somewhat (typically under 5%-10%). To even out variability, the sensor can be queried sampleSize times and an average obtained.

**Parameters**

<table>
<thead>
<tr>
<th>lightSensors</th>
<th>array to store intensity values</th>
</tr>
</thead>
<tbody>
<tr>
<td>sampleSize</td>
<td>how many readings are taken for each sensor</td>
</tr>
</tbody>
</table>

**Precondition**

space already allocated for lightSensors array sampleSize > 0

**Postcondition**

- lightSensors[0] gives average value for left sensor
- lightSensors[1] gives average value for middle sensor
- lightSensors[2] gives average value for right sensor
- Intensity values near 0 represent bright light
- Intensities may extend to about 65000 for a dark region.

**Parameters**

<table>
<thead>
<tr>
<th>lightSensors</th>
<th>array to store intensity values</th>
</tr>
</thead>
<tbody>
<tr>
<td>sampleSize</td>
<td>how many readings are taken for each sensor</td>
</tr>
</tbody>
</table>

**Precondition**

space already allocated for lightSensors array sampleSize > 0

**Postcondition**

- lightSensors[0] gives average value for left sensor
- lightSensors[1] gives average value for middle sensor
- lightSensors[2] gives average value for right sensor
- Intensity values near 0 represent bright light
- Intensities may extend to about 65000 for a very dark region.

4.2.2.19 rGetLightTxt()

```c
int rGetLightTxt ( const char * sensorName, int sampleSize )
```

Get the average values of a specified light sensor. Values of each light sensor can somewhat (typically under 5%-10%). To even out variability, the sensor can be queried sampleSize times and an average obtained.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensorName</td>
<td>name of the light sensor</td>
</tr>
</tbody>
</table>

Precondition

sensorName is "left", "center", "middle", or "right" (not case sensitive) designations "center" and "middle" are alternatives for the same light sensor

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sampleSize</td>
<td>how many readings are taken for the sensor</td>
</tr>
</tbody>
</table>

Precondition

sampleSize > 0

Returns

reading from the specified light sensor, averaged over sampleSize number of data samples if sensorName invalid, returns -1.0

Get the average values of a specified light sensor. Values of each light sensor can somewhat (typically under 5%-10%). To even out variability, the sensor can be queried sampleSize times and an average obtained.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensorName</td>
<td>name of the light sensor</td>
</tr>
</tbody>
</table>

Precondition

sensorName is "left", "center", "middle", or "right" (not case sensitive) designations "center" and "middle" are alternatives for the same light sensor

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sampleSize</td>
<td>how many readings are taken for the sensor</td>
</tr>
</tbody>
</table>

Precondition

sampleSize > 0

Returns

reading from the specified light sensor, averaged over sampleSize number of data samples if sensorName invalid, returns -1.0
4.2.2.20  rGetLine()

```c
void rGetLine(
    int lineSensors[2],
    int sampleSize
)
```

Use Scribbler 2 line sensors of Scribbler to check for a black line on a white surface under the robot. Since values of each light sensor can vary substantially, the sensor can be queried sampleSize times and an average obtained.

**Warning**

results of these sensors may be flakey!

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lineSensors</td>
<td>array to store line values detected</td>
</tr>
<tr>
<td>sampleSize</td>
<td>how many readings are taken for each sensor</td>
</tr>
</tbody>
</table>

**Precondition**

space already allocated for lineSensors array sampleSize > 0

**Postcondition**

- lineSensors[0] checks left sensor for line
- lineSensors[1] checks right sensor for line
- for each irSensors array value
- return 0 indicates line is identified
- return 1 indicates line is not identified

Use Scribbler 2 line sensors of Scribbler to check for a black line on a white surface under the robot. Since values of each light sensor can vary substantially, the sensor can be queried sampleSize times and an average obtained.

**Warning**

results of these sensors may be flakey!

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lineSensors</td>
<td>array to store line values detected</td>
</tr>
<tr>
<td>sampleSize</td>
<td>how many readings are taken for each sensor</td>
</tr>
</tbody>
</table>

**Precondition**

space already allocated for lineSensors array sampleSize > 0
4.2 MyroC.h File Reference

Postcondition

lineSensors[0] checks left sensor for line lineSensors[1] checks right sensor for line (left/right based on Scribbler-forward orientation)
for each line sensors array value return 0 indicates line is identified (light color under Scribbler) return 1 indicates line is not identified (dark color)

4.2.2.21 rGetName()

const char* rGetName ()

Get the name of the robot.

Returns

information about the name of the robot

Postcondition

the returned name is a newly-allocated 17-byte string

4.2.2.22 rGetObstacleAll()

void rGetObstacleAll ( 
    int obstSensors[3],
    int sampleSize )

Get the average values of the three obstacle sensors in an array. Since readings of each obstacle sensor can vary substantially (successive readings may differ by several hundred or more), each sensor can be queried sampleSize times and an average obtained.

Parameters

<table>
<thead>
<tr>
<th>obstSensors</th>
<th>array to store intensity values</th>
</tr>
</thead>
<tbody>
<tr>
<td>sampleSize</td>
<td>how many readings are taken for each sensor</td>
</tr>
</tbody>
</table>

Precondition

space already allocated for obstSensors array; sampleSize > 0

Postcondition

obstSensors[0] gives average value for left sensor
obstSensors[1] gives average value for middle sensor
obstSensors[2] gives average value for right sensor
Returned values are between 0 and 6400
Obstacle values near 0 represent no obstacle seen
Obstacle values may approach 6400 as obstacle gets close.
Warning

As battery degrades, sensor readings degrade, yielding systematically lower numbers.

Parameters

| obstSensors | array to store intensity values |
| sampleSize  | how many readings are taken for each sensor |

Precondition

space already allocated for obstSensors array sampleSize > 0

Postcondition

obstSensors[0] gives average value for left sensor obstSensors[1] gives average value for middle sensor obstSensors[2] gives average value for right sensor Returned values are between 0 and 6400 Obstacle values near 0 represent no obstacle seen Obstacle values may approach 6400 as obstacle gets close.

Warning

As battery degrades, sensor readings degrade, yielding systematically lower numbers.

4.2.2.23 rGetObstacleTxt()

int rGetObstacleTxt ( 
    const char * sensorName, 
    int sampleSize )

Get the average values of a specified obstacle (IR) sensor. Since values of each obstacle sensor can vary substantially (successive readings may differ by several hundred or more), the sensor can be queried sampleSize times and an average obtained.

Parameters

| sensorName | name of the obstacle sensor |

Precondition

sensorName is "left", "center", "middle", or "right" (not case sensitive) 
designations "center" and "middle" are alternatives for the same light sensor

Parameters

| sampleSize | how many readings are taken for the sensor |
Precondition

space already allocated for vals array; sampleSize > 0

Returns

reading from the specified obstacle sensor, averaged over sampleSize number of data samples
Returned values are between 0 and 6400
Obstacle values near 0 represent no obstacle seen
Obstacle values may approach 6400 as obstacle gets close.

Warning

As battery degrades, sensor values degrade, yielding systematically lower numbers.

Parameters

<table>
<thead>
<tr>
<th>sensorName</th>
<th>name of the obstacle sensor</th>
</tr>
</thead>
</table>

Precondition

sensorName is "left", "center", "middle", or "right" (not case sensitive) designations "center" and "middle" are alternatives for the same light sensor

Parameters

<table>
<thead>
<tr>
<th>sampleSize</th>
<th>how many readings are taken for the sensor</th>
</tr>
</thead>
</table>

Precondition

space already allocated for vals array sampleSize > 0

Returns

reading from the specified obstacle sensor, averaged over sampleSize number of data samples Returned values are between 0 and 6400 Obstacle values near 0 represent no obstacle seen Obstacle values may approach 6400 as obstacle gets close.

Warning

As battery degrades, sensor values degrade, yielding systematically lower numbers.
4.2.24 rGetStall()

```c
int rGetStall (int sampleSize)
```

Determine if robot has stalled

MyroC Reference Manual (http://wiki.robotorification.org/Myro_Reference_Manual) states "Every time you issue a move command, the stall sensor resets, and it needs to wait a short time to see whether the motors are stalled. This means that the sensor won't give accurate results if you test it too soon after the robot starts to move."

In practice, it may take 0.5-1.0 seconds for rGetStall to sense the robot is stalled

Parameters

<table>
<thead>
<tr>
<th>sampleSize</th>
<th>how many readings are taken for each sensor</th>
</tr>
</thead>
</table>

Precondition

sampleSize > 0

Returns

whether or not robot current has stalled

Postcondition

Returns 1 if the robot has stalled
Returns 0 otherwise.

Determine if robot has stalled

MyroC Reference Manual (http://wiki.robotorification.org/Myro_Reference_Manual) states Every time you issue a move command, the stall sensor resets, and it needs to wait a short time to see whether the motors are stalled. This means that the sensor won't give accurate results if you test it too soon after the robot starts to move.

In practice, it may take 0.5-1.0 seconds for rGetStall to sense the robot is stalled

Parameters

<table>
<thead>
<tr>
<th>sampleSize</th>
<th>how many readings are taken for each sensor</th>
</tr>
</thead>
</table>

Precondition

sampleSize > 0

Returns

whether or not robot current has stalled
Postcondition

Returns 1 if the robot has stalled Returns 0 otherwise.

4.2.2.25 rHardStop()

void rHardStop ( )
cuts power to the motor of the robot
cuts power to the motor of the robot

4.2.2.26 rLoadPicture()

Picture rLoadPicture ( 
    char * filename )
Load a picture from a .jpeg file.

Parameters

| filename | the name of the file |

Precondition

file must exist
file extension must be .jpeg or .jpg
file must be no larger than 266 by 427

Returns

Picture

Parameters

| filename | the name of the file |

Precondition

file must exist
file extension must be .jpeg or .jpg
file must be no larger than 266 by 427

Returns

locally-declared Picture
Postcondition

calling program should store returned Picture in a previously-declared struct

4.2.2.27  rMotors()

void rMotors {
    double leftSpeed,
    double rightSpeed )

move robot with given speeds for the left and right motors continues until given another motion command or discon-
nected (non-blocking)

Parameters

<table>
<thead>
<tr>
<th>leftSpeed</th>
<th>the rate at which the left wheel should turn</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>linear range: -1.0 specifies move backward at full speed</td>
</tr>
<tr>
<td></td>
<td>0.0 specifies no forward/backward movement</td>
</tr>
<tr>
<td></td>
<td>1.0 specifies move forwardward at full speed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>rightSpeed</th>
<th>the rate at which the right wheel should turn</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>linear range: -1.0 specifies move backward at full speed</td>
</tr>
<tr>
<td></td>
<td>0.0 specifies no forward/backward movement</td>
</tr>
<tr>
<td></td>
<td>1.0 specifies move forward at full speed</td>
</tr>
</tbody>
</table>

move robot with given speeds for the left and right motors continues until given another motion command or discon-
nected (non-blocking)

Parameters

<table>
<thead>
<tr>
<th>leftSpeed</th>
<th>the rate at which the left wheel should turn</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>linear range: -1.0 specifies move backward at full speed</td>
</tr>
<tr>
<td></td>
<td>0.0 specifies no forward/backward movement</td>
</tr>
<tr>
<td></td>
<td>1.0 specifies move forwardward at full speed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>rightSpeed</th>
<th>the rate at which the right wheel should turn</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>linear range: -1.0 specifies move backward at full speed</td>
</tr>
<tr>
<td></td>
<td>0.0 specifies no forward/backward movement</td>
</tr>
<tr>
<td></td>
<td>1.0 specifies move forward at full speed</td>
</tr>
</tbody>
</table>

4.2.2.28  rSavePicture()

void rSavePicture ( {
    Picture * pic,
    char * filename )

Save a Picture to a .jpeg.

Parameters

<table>
<thead>
<tr>
<th>pic</th>
<th>pointer to an RGB picture struct from Scribbler 2 camera</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>the name of the file</td>
</tr>
</tbody>
</table>

Generated by Doxygen
Precondition

filename ends with .jpeg or .jpg.

Postcondition

If the file does not exist, a new file will be created.
If the file exists, the file will be overwritten.

This section contains functions for taking and manipulating images. All images are constrained with height ≤ 266
and width ≤ 427.

Images from robot cameras have varying sizes, depending on the Fluke images for the original Fluke are 192
(height) by 256 (width) low-resolution images for the Fluke 2 are 266 by 427 high-resolution images for the Fluke 2
are 800 by 1280 (not supported).

Bluetooth communication constrains the time required for the Fluke to take a picture. Typical times: original Fluke: 2-

BASED ON TIMINGS AND MEMORY CONSIDERATIONS, Myro C ALLOWS ONLY LOW RESOLUTION IMAGES.

The Picture struct allows direct access to Pixel data. Pictures can be saved and loaded as .jpeg files.

Note

Following standard mathematical convention for a 2D matrix, all references to a pixel are given within an array
as [row][col].

User-defined images may have any size, as long as height ≤ 266 and width ≤ 427.
Following standard mathematical convention for a 2D matrix, all references to a pixel are given within an array
as [row][col].

Warning

The Picture struct is defined to be sufficiently large to store several low-resolution camera images (340756
bytes each). Experimentally, an array of up to 94 (not 95) Pictures is allowed. However, the display of images
requires that image data be copied, so display of many images may not work. If a program hangs when working
with Picture variables, the issue may involve lack of space on the runtime stack. To utilize a modest number
of Pictures, use "ulimit -s" command, as needed, in a terminal window. For example, ulimit -s 32768. Sizes above
32768 may not be allowed in Linux or Mac OS X. Save a Picture to a .jpeg.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pic</td>
<td>Pointer to an RGB picture struct from Scribbler 2 camera</td>
</tr>
<tr>
<td>filename</td>
<td>The name of the file</td>
</tr>
</tbody>
</table>

Precondition

filename ends with .jpeg or .jpg.

Postcondition

If the file does not exist, a new file will be created.
If the file exists, the file will be overwritten.
this code is a slightly modified version of write_JPEG_file(char *, int) from example.c from the libjpeg.v8d library directory.

4.2.2.29 rSetBluetoothEcho()

void rSetBluetoothEcho ( 
    char onOff )

Turn on and off the printing transmission and echong of Bluetooth transmissions All robot commands involve the transmission of a command over Bluetooth Scribbler commands are always 9 bytes Fluke commands have varying lengths The fluke echos most, but not all, of the commands For many commands, the fluke also echos 11 bytes of sensor data.

Parameters

<table>
<thead>
<tr>
<th>onOff</th>
<th>char 'y' enables printing of Bluetooth communications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>char 'n' disables printing</td>
</tr>
<tr>
<td></td>
<td>other character values ignored</td>
</tr>
</tbody>
</table>

Turn on and off the printing transmission and echong of Bluetooth transmissions All robot commands involve the transmission of a command over Bluetooth Scribbler commands are always 9 bytes Fluke commands have varying lengths The fluke echos most, but not all, of the commands For many commands, the fluke also echos 11 bytes of sensor data.

Parameters

| onOff | char 'y' enables echoing char 'n' disables echoing other character values ignored |

4.2.2.30 rSetConnection()

void rSetConnection ( 
    ROBOT_CONNECTION_TYPE new_socket_num )

set current connection to the socket number

Parameters

| new_socket_num | the number of an open socket for communication |

Precondition

new_socket_num has been returned by rConnect the designated socket has not been closed
Note

a socket has type int on Linux or Mac; type SOCKET on Windows together these are defined as ROBOT Conexion_TYPE

set current connection to the socket number

Parameters

| new_socket_num | the number of an open socket for communication |

Precondition

current_socket_num has been returned by rConnect the designated socket has not been closed

4.2.2.31  rSetForwardness()

void rSetForwardness ( char * direction )
specifies which end of the Scribbler is considered the front

Parameters

| direction | identifies front direction |

Precondition

direction is either "fluke-forward" or "scribbler-forward" (not case sensitive)

specifies which end of the Scribbler is considered the front

Parameters

| direction | identifies front direction |

Precondition

direction is either "fluke-forward" or "scribbler-forward" (not case sensitive)

4.2.2.32  rSetForwardnessTxt()

void rSetForwardnessTxt ( char * direction )
alternative to rSetForwardness for compatibility with earlier MyroC

alternative to rSetForwardness for compatibility with earlier MyroC
4.2.2.33  rSetIRPower()

```c
void rSetIRPower (    
    int power    )
```

Set the amount of power for the dongle's IR sensors.

**Parameters**

| power        | the desired power level for the IR sensors |

**Precondition**

- power is between 0 and 255 (inclusive)
- Manufacturer notes: default value is 135
- if IR obstacle sensor is always high, try lowering IR power
- if IR obstacle sensor is always low, try raising IR power

**Parameters**

| power        | the desired power level for the IR sensors |

**Precondition**

- power is between 0 and 255 (inclusive)

4.2.2.34  rSetLEDBack()

```c
void rSetLEDBack (    
    double led    )
```

Set the intensity of the back fluke LED.

**Parameters**

| led         | intensity of the LED values between 0 and 1 provide a range of brightness from off to full intensity values bigger than 1 are treated as 1 (full brightness) values less than 0 are treated as 0 (LED off) |

Set the intensity of the back fluke LED.

**Parameters**

| led         | intensity of the LED values between 0 and 1 provide a range of brightness from off to full intensity values bigger than 1 are treated as 1 (full brightness) values less than 0 are treated as 0 (LED off). |
4.2.2.35  rSetLEDFront()

```c
void rSetLEDFront ( int led )
```

Set the front [fluke] LED on or off.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>led</code></td>
<td>value 1 turns on LED, value 0 turns off LED</td>
</tr>
</tbody>
</table>

**Precondition**

`led` must be 0 or 1

4.2.2.36  rSetName()

```c
void rSetName ( const char * name )
```

Change name stored in the robot to the 16-byte name given.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>name</code></td>
<td>specifies new name of robot</td>
</tr>
<tr>
<td></td>
<td>if &lt; 16 bytes given, name is filled with null characters</td>
</tr>
<tr>
<td></td>
<td>if &gt;= 16 bytes given, name is truncated to 15 bytes plus null</td>
</tr>
</tbody>
</table>

Change name stored in the robot to the 16-byte name given.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>name</code></td>
<td>specifies new name of robot if &lt; 16 bytes given, name is filled with null characters if &gt;= 16 bytes given, name is truncated to 15 bytes plus null</td>
</tr>
</tbody>
</table>

Generated by Doxygen
4.2.2.37 rSetVolume()

```c
void rSetVolume (char highMute)
```

Set sound to high volume (H) or mute (M) highMute set volume of Scribbler.

Precondition

highMute is 'H' to set for high volume or 'M' for mute

4.2.2.38 rStop()

```c
void rStop ()
```

directs robot to stop movement
directs robot to stop movement

4.2.2.39 rTakePicture()

```c
Picture rTakePicture ( )
```

Use the camera to take a photo.

This section contains functions for taking and manipulating images All images are constrained with height <= 266 and width <= 427

images from robot cameras have varying sizes, depending on the Fluke images for the original Fluke are 192 (height) by 256 (width) low-resolution images for the Fluke 2 are 266 by 427 high-resolution images for the fluke 2 are 800 by 1280

Bluetooth communication constrains the time required for the Fluke to take a picture Typical times: original fluke: 2-4 seconds Fluke 2 (low res): 4-6 seconds Fluke 2 (high res): 25-30 seconds

BASED ON TIMINGS AND MEMORY CONSIDERATIONS, Myro C ALLOWS ONLY LOW RESOLUTION IMAGES

the Picture struct allows direct access to Pixel data Pictures can be saved and loaded as .jpeg files

Note

Following standard mathematical convention for a 2D matrix, all references to a pixel are given within an array as [row][col]
user-defined images may have any size, as long as height <= 266 and width <= 427
Following standard mathematical convention for a 2D matrix, all references to a pixel are given within an array as [row][col]
Warning

The Picture struct is defined to be sufficiently large to store several low-resolution camera images (340756 bytes each). Experimentally, an array of up to 94 (not 95) Pictures is allowed. However, the display of images requires that image data be copied, so display of many images may not work if a program hangs when working with Picture variables, the issue may involve lack of space on the runtime stack. To utilize a modest number of Pictures, use “ulimit -s” command, as needed, in a terminal window. For example, ulimit -s 32768 Sizes above 32768 may not be allowed in Linux or Mac OS X.

Returns

Picture

This section contains functions for taking and manipulating images. All images are constrained with height <= 266 and width <= 427.

Images from robot cameras have varying sizes, depending on the Fluke images for the original Fluke are 192 (height) by 256 (width) low-resolution images for the Fluke 2 are 266 by 427 high-resolution images for the Fluke 2 are 800 by 1280.

Bluetooth communication constrains the time required for the Fluke to take a picture. Typical times: original fluke: 2-4 seconds Fluke 2 (low res): 4-6 seconds Fluke 2 (high res): 25-30 seconds

BASED ON TIMINGS AND MEMORY CONSIDERATIONS, Myro C ALLOWS ONLY LOW RESOLUTION IMAGES.

The Picture struct allows direct access to Pixel data. Pictures can be saved and loaded as .jpeg files.

Note

Following standard mathematical convention for a 2D matrix, all references to a pixel are given within an array as [row][col].

User-defined images may have any size, as long as height <= 266 and width <= 427.

Following standard mathematical convention for a 2D matrix, all references to a pixel are given within an array as [row][col].

Warning

The Picture struct is defined to be sufficiently large to store several low-resolution camera images (340756 bytes each). Experimentally, an array of up to 94 (not 95) Pictures is allowed. However, the display of images requires that image data be copied, so display of many images may not work if a program hangs when working with Picture variables, the issue may involve lack of space on the runtime stack. To utilize a modest number of Pictures, use “ulimit -s” command, as needed, in a terminal window. For example, ulimit -s 32768 Sizes above 32768 may not be allowed in Linux or Mac OS X. Use the camera to take a photo.

Returns

Picture

4.2.2.40 rTurnLeft()

void rTurnLeft (  
    double speed,  
    double time )

turn Scribbler left for a specified time and speed.
Parameters

| speed | the rate at which the robot should move left linear range: -1.0 specifies right turn at full speed 0.0 specifies no turn 1.0 specifies left turn at full speed |
| time | specifies the duration of the turn if negative: the robot starts to turn (non-blocking) other processing proceeds, and the robot continues to turn until given another motion command or disconnected (non-blocking) if zero: robot starts turning (non-blocking); other processing proceeds if positive: robot turns for the given duration, in seconds |

4.2.2.41 rTurnRight()

```cpp
void rTurnRight (  
        double speed,  
        double time )
```

Parameters

| speed | the rate at which the robot should move right linear range: -1.0 specifies left turn at full speed 0.0 specifies no turn 1.0 specifies right turn at full speed |
| time | specifies the duration of the turn if negative: the robot starts to turn (non-blocking) other processing proceeds, and the robot continues to turn until given another motion command or disconnected (non-blocking) if zero: robot starts turning (non-blocking); other processing proceeds if positive: robot turns for the given duration, in seconds if nonnegative: robot turns for the given duration, in seconds |

turn Scribbler left for a specified time and speed

Parameters

| speed | the rate at which the robot should move left linear range: -1.0 specifies right turn at full speed 0.0 specifies no turn 1.0 specifies left turn at full speed |
| time | specifies the duration of the turn if negative: the robot starts to turn (non-blocking) other processing proceeds, and the robot continues to turn until given another motion command or disconnected (non-blocking) if zero: robot starts turning (non-blocking); other processing proceeds if positive: robot turns for the given duration, in seconds |

turn Scribbler right for a specified time and speed

Parameters

| speed | the rate at which the robot should move right linear range: -1.0 specifies left turn at full speed 0.0 specifies no turn 1.0 specifies right turn at full speed |
| time | specifies the duration of the turn if negative: the robot starts to turn (non-blocking) other processing proceeds, and the robot continues to turn until given another motion command or disconnected (non-blocking) if zero: robot starts turning (non-blocking); other processing proceeds if positive: robot turns for the given duration, in seconds if nonnegative: robot turns for the given duration, in seconds |

turn Scribbler right for a specified time and speed

Parameters

| speed | the rate at which the robot should move right linear range: -1.0 specifies left turn at full speed 0.0 specifies no turn 1.0 specifies right turn at full speed |
| time | specifies the duration of the turn if negative: the robot starts to turn (non-blocking) other processing proceeds, and the robot continues to turn until given another motion command or disconnected (non-blocking) if zero: robot starts turning (non-blocking); other processing proceeds if positive: robot turns for the given duration, in seconds if nonnegative: robot turns for the given duration, in seconds |
4.2.2.42 `rTurnSpeed()`

```c
void rTurnSpeed (char * direction, double speed, double time )
```

turn Scribbler in direction for a specified time and speed

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>direction</em></td>
<td>direction of turn, based on looking from the center of the robot and facing forward</td>
</tr>
<tr>
<td><em>speed</em></td>
<td>the rate at which the robot should move forward linear range: -1.0 specifies turn at full speed 0.0 specifies no turn 1.0 specifies turn at full speed</td>
</tr>
<tr>
<td><em>time</em></td>
<td>specifies the duration of the turn if negative: the robot starts to turn (non-blocking) other processing proceeds, and the robot continues to turn until given another motion command or disconnected (non-blocking) if zero: robot starts turning (non-blocking); other processing proceeds if positive: robot turns for the given duration, in seconds</td>
</tr>
</tbody>
</table>

**Precondition**

- `direction` is "left" or "right", case insensitive

4.2.2.43 `rWaitTimedImageDisplay()`

```c
void rWaitTimedImageDisplay ( )
```

Wait until all timed, non-blocking image window timers are complete.

Generated by Doxygen
Postcondition
   wait until all timed [non-blocking] images have closed
   robot motion is unaffected by this function

Warning
   images opened with duration 0.0 do not close until images are updated or until the program terminates
   all other [timed, non-blocking] image windows are closed by this function

Postcondition
   wait until all timed [non-blocking] images have closed
   robot motion is unaffected by this function

Warning
   images opened with duration 0.0 do not close until images are updated or until the program terminates
   all other image windows are closed by this function
Index

bth_inquiry_record, 5
GLimageset, 5
image_timing_data, 6
motion_timing_data, 7
my_error_mgr, 7
MyroC-image-file.c, 11
   rLoadPicture, 13
   rSavePicture, 14
MyroC.h, 15
   rBackward, 18
   rBeep, 19
   rBeep2, 19
   rConnect, 20
   rDisconnect, 22
   rDisplayImageProcessingErrors, 22
   rDisplayPicture, 23
   rFastForward, 24
   rFinishProcessing, 25
   rForward, 25
   rGetBattery, 26
   rGetBrightAll, 26
   rGetBrightTxt, 27
   rGetForwardness, 28
   rGetRAll, 29
   rGetRTxt, 30
   rGetInfo, 29
   rGetLightTxt, 32
   rGetLightsAll, 31
   rGetLine, 33
   rGetName, 35
   rGetObstacleAll, 35
   rGetObstacleTxt, 36
   rGetStall, 37
   rHardStop, 39
   rLoadPicture, 39
   rMotors, 40
   rSavePicture, 40
   rSetBluetoothEcho, 42
   rSetConnection, 42
   rSetForwardness, 43
   rSetForwardnessTxt, 43
   rSetRPower, 43
   rSetLEDBack, 44
   rSetLEDFront, 45
   rSetName, 45
   rSetVolume, 45
   rStop, 46
   rTakePicture, 46
   rTurnLeft, 47
   rTurnRight, 48
   rTurnSpeed, 49
   rWaitTimedImageDisplay, 49
Picture, 7
Pixel, 8
   rBackward
      MyroC.h, 18
   rBeep
      MyroC.h, 19
   rBeep2
      MyroC.h, 19
   rConnect
      MyroC.h, 20
   rDisconnect
      MyroC.h, 22
   rDisplayImageProcessingErrors
      MyroC.h, 22
   rDisplayPicture
      MyroC.h, 23
   rFastForward
      MyroC.h, 24
   rFinishProcessing
      MyroC.h, 25
   rForward
      MyroC.h, 25
   rGetBattery
      MyroC.h, 26
   rGetBrightAll
      MyroC.h, 26
   rGetBrightTxt
      MyroC.h, 27
   rGetForwardness
      MyroC.h, 27
   rGetRAll
      MyroC.h, 28
   rGetRTxt
      MyroC.h, 29
   rGetInfo
      MyroC.h, 29
   rGetLightTxt
      MyroC.h, 30
   rGetLightsAll
      MyroC.h, 32
   rGetLightsAll
      MyroC.h, 31
   rGetLine
      MyroC.h, 33
   rGetName
INDEX

MyroC.h, 35
rGetObstacleAll
MyroC.h, 35
rGetObstacleTxt
MyroC.h, 36
rGetStall
MyroC.h, 37
rHardStop
MyroC.h, 39
rLoadPicture
MyroC-image-file.c, 13
MyroC.h, 39
rMotors
MyroC.h, 40
rSavePicture
MyroC-image-file.c, 14
MyroC.h, 40
rSetBluetoothEcho
MyroC.h, 42
rSetConnection
MyroC.h, 42
rSetForwardness
MyroC.h, 43
rSetForwardnessTxt
MyroC.h, 43
rSetIRPower
MyroC.h, 43
rSetLEDBack
MyroC.h, 44
rSetLEDFront
MyroC.h, 45
rSetName
MyroC.h, 45
rSetVolume
MyroC.h, 45
rStop
MyroC.h, 46
rTakePicture
MyroC.h, 46
rTurnLeft
MyroC.h, 47
rTurnRight
MyroC.h, 48
rTurnSpeed
MyroC.h, 49
rWaitTimedImageDisplay
MyroC.h, 49
robot_motion_camera_record, 9

timing_thread_info, 9