

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 643924



D3.3

Firmware Documentation



Copyright © 2015 The EoT Consortium

The opinions of the authors expressed in this document do not necessarily reflect the official opinion of EOT partners or of the European Commission.

1. DOCUMENT INFORMATION

Deliverable Number	D3.3
Deliverable Name	Firmware documentation
Authors	N. Vallez (UCLM), J. L. Espinosa-Aranda (UCLM), J. M. Rico (UCLM), S. Krauss (DFKI), R. Reiser (DFKI), A. Pagani (DFKI), A. Dehghani (MOVIDIUS)
Responsible Author	Oscar Deniz (UCLM) e-mail:Oscar.Deniz@uclm.es phone: +34 926295300 Ext.6286
Keywords	EoT Firmware, API
WP	WP3
Nature	R
Dissemination Level	PU
Planned Date	01.03.2016
Final Version Date	29.02.2016
Reviewed by	O. Deniz (UCLM)
Verified by	C. Fedorczak (THALES)

2. DOCUMENT HISTORY

Person	Date	Comment	Version
N. Vallez	02.02.2016	Initial	0.1
J. L. Espinosa-Aranda	11.02.2016	WiFi Data Transfer section	0.2
N. Vallez	12.02.2016	Camera interface and Video Streaming sections	0.3
J. L. Espinosa-Aranda and N. Vallez	16.02.2016	Other vision libraries section and revision of previous sections	0.4
N. Vallez	17.02.2016	Revision of Other vision libraries section	0.5
J. L. Espinosa-Aranda	19.02.2016	Introduction and Computer vision: Sparse optical flow (LK point tracking) sections	0.6
N. Vallez	22.02.2016	Power Management section and revision	0.7
N. Vallez	23.02.2016	Revision of Introduction and Power Management sections	0.8
S. Krauss, R. Reiser	25.02.2016	DFKI sections and abstract added	0.9
A. Dehghani	25.02.2016	Motor control added	0.10
J. M. Rico	25.02.2016	Rotation Invariant Face Detection added	0.11
A. Dehghani	26.02.2016	CNN	0.12
O. Deniz	29.02.2016	Final version after Thales	0.13

3. ABSTRACT

This document describes the firmware software developed in EoT as of March 1st, 2016. This firmware is intended to be compiled for and executed on the Myriad 2 SoC along with a number of other hardware components such as Wifi, SD card, etc. The software modules have been generated according to the task structure of WP3 (Software). Some of the tasks described in this document were not in the original DoW but were added later as they were considered relevant for the project objectives (Other vision libraries, Motor control, CNN, audio input & codec). The main participating partners are UCLM, DFKI and Movidius. Deliverables D3.1 and D3.2 described the Control Mode software generated for desktop and Android platforms, along with the associated firmware side, and thus they will not be covered here.

4. TABLE OF CONTENTS

1.	Document Information.....	2
2.	Document History	3
3.	Abstract.....	4
4.	Table of Contents.....	5
5.	List of Figures.....	8
6.	Introduction	10
6.1.	Myriad 2.....	10
6.2.	RTEMS	10
6.3.	Myriad 2 programming paradigms	11
6.4.	EoT firmware	13
6.5.	EoT repository.....	14
7.	WiFi data transfer	16
7.1.	Introduction.....	16
7.1.	Known issues.....	17
7.2.	Unit tests	17
7.3.	Licensing	18
7.4.	Code.....	18
7.5.	Conclusions and Future work	19
8.	Camera interface	20
8.1.	Introduction.....	20
8.2.	Known issues.....	22
8.3.	Unit tests	22
8.4.	Licensing	23
8.5.	Code.....	24
8.6.	Conclusions and Future work	24
9.	Video streaming.....	25
9.1.	Introduction.....	25
9.2.	RTSP Server	26
9.3.	Supported Players	27
9.4.	Known issues.....	29
9.5.	Unit tests	29
9.6.	Code.....	29
9.7.	Conclusions and Future work	30
10.	Input buttons/DIP switches	31
10.1.	Introduction.....	31
10.2.	Unit tests	31
10.3.	Code.....	31
10.4.	Conclusions and Future work	32
11.	SD card management.....	33
11.1.	Introduction.....	33
11.2.	Known issues.....	33
11.3.	Unit tests	33
11.4.	Licensing	34
11.5.	Code.....	34
11.6.	Conclusions and Future work	34
12.	Bootloader	35
12.1.	Introduction.....	35

12.2.	Known issues	36
12.3.	Unit tests	36
12.4.	Code.....	36
12.5.	Conclusions and Future work	36
13.	Control mode API, embedded side.....	37
14.	Audio input & output	38
14.1.	Introduction.....	38
14.2.	Licensing	38
14.3.	Code.....	38
14.4.	Conclusions and Future work	38
15.	Computer vision: CNN	39
15.1.	Introduction.....	39
15.2.	Movidius Fathom CNN framework	44
15.3.	MvTensor Implementation Details.....	45
15.4.	GoogleNet Example	47
16.	Computer vision: Colour histogram matching	57
16.1.	Introduction.....	57
16.2.	Unit tests	57
16.3.	Code.....	57
16.4.	Conclusions and Future work	57
17.	Computer vision: Keypoint matching	58
17.1.	Introduction.....	58
17.2.	Known issues.....	59
17.3.	Unit tests	59
17.4.	Licensing	59
17.5.	Code.....	59
17.6.	Conclusions and Future work	59
18.	Computer vision: Rotation-invariant face detector	60
18.1.	Introduction.....	60
18.2.	Known issues.....	68
18.3.	Unit tests	68
18.4.	Licensing	70
18.5.	Code.....	70
18.6.	Conclusions and Future work	70
19.	Computer vision: Sparse optical flow (LK point tracking).....	71
19.1.	Introduction.....	71
19.2.	Known issues.....	74
19.3.	Unit tests	74
19.4.	Licensing	78
19.5.	Code.....	78
19.6.	Conclusions and Future work	79
20.	Power management	80
20.1.	Introduction.....	80
20.2.	Low power states	81
20.3.	Unit tests	82
20.4.	Code.....	82
20.5.	Conclusions and Future work	85
21.	Control mode API, Desktop side.....	86
22.	Control mode API, Android	87
23.	Other vision libraries	89
23.1.	Introduction.....	89

23.2.	Known issues	94
23.3.	Unit tests	95
23.4.	Licensing	109
23.5.	Code	114
23.6.	Conclusions and Future work	115
24.	Motor control	116
24.1.	Introduction	116
24.2.	Motion Control API	116
24.3.	Android App for Ground Robot control.	120
24.4.	Known issues	123
24.5.	Unit tests	123
24.6.	Licensing	123
24.7.	Code	124
24.8.	Conclusions and Future work	124
25.	Conclusions	125
26.	Glossary	126

Annexes:

Annex 1	WiFi Functions
Annex 2	Camera
Annex 3	RTSP
Annex 4	Buttons, Switches and LEDs
Annex 5	Crypto
Annex 6	SDCardIO
Annex 7	Elf Loader
Annex 8	FlashIO
Annex 9	Audio
Annex 10	Histogram Matching
Annex 11	Rotation-Invariant Face Detection
Annex 12	Cherokey 4WD

5. LIST OF FIGURES

Figure 1. Myriad 2 Block Diagram	10
Figure 2. Standard programming paradigm diagram	11
Figure 3. One Leon programming paradigm diagram.....	12
Figure 4. Bare metal programming paradigm diagram.....	12
Figure 5. EoT firmware structure	14
Figure 6. TI CC3100MOD WiFi module.....	16
Figure 7. MIPI camera used for development	20
Figure 8. Camera execution diagram	21
Figure 9. Camera frame retrieved from Pulga.....	23
Figure 10. RTSP video streaming.....	23
Figure 11. RTSP Player TCP configuration	28
Figure 12. VLC TCP configuration	29
Figure 13. RTSP streaming results with 2 clients	29
Figure 14. Buttons, DIP switches and LEDs	31
Figure 15. SD card slot	33
Figure 16. Flash memory layout	35
Figure 17. Architecture of a convolutional neural network.....	40
Figure 18. Kernel convolution	40
Figure 19. Architecture of LeNet.....	41
Figure 20. An illustration of the architecture of AlexNet CNN.....	42
Figure 21. ZF Net architecture	43
Figure 22. GoogleNet architecture	43
Figure 23. Movidius Tool converts into DNN using Conv/MatMul libraries	44
Figure 24. Movidius MDK including Fathom and Tensor CNN support.....	45
Figure 25. Core operation in MvMatMul Library.....	48
Figure 26. Face detection sequence diagram.....	60
Figure 27. Rotation matrix.....	62
Figure 28. Face detection dependencies	62
Figure 29. Memory positions and size	65
Figure 30. Wrong rotated image.....	67
Figure 31. Rotated images.....	69
Figure 32. Vtrack flow diagram	73
Figure 33. OpenCV optical flow example.....	78
Figure 34. Vtrack optical flow example.....	79
Figure 35. Myriad 2 power islands diagram	80
Figure 36. Include dependency graph for OsDrvCpr.h	83
Figure 37. OpenCV in the cloud paradigm.....	89
Figure 38. Add new app	90
Figure 39. Selecting web2py.....	90
Figure 40. Choosing the application name	91
Figure 41. MDK vs OpenCV benchmarks.....	99
Figure 42. QR first test image	104
Figure 43. QR second test image.....	104
Figure 44. QR third test image	105
Figure 45. QR fourth test image	106
Figure 46. QR fifth test image	106
Figure 47. QR sixth test image.....	107

Figure 48. QR seventh test image.....	108
Figure 49. Cherokey 4WD from DFRobot.	116
Figure 50. Connection between the EoT DevBoard and the Cherokey 4WD	119
Figure 51. Ground Robot Android App	120
Figure 52. Entering the IP address of the EoT board	121
Figure 53. Connecting and Connected to EoT board	121
Figure 54. Motion control using phone orientation	122

6. INTRODUCTION

Myriad 2

The EoT device is based on the Myriad 2 VPU. It supports two trillion 16-bit operations per second at a power consumption of 500mW. The architecture is based on 12 128-bit very long instruction word "SHAVE" processors and two 32-bit RISC processors (LeonOS and LeonRT). The chip includes 2-Mbytes of on-chip memory and 400-Gbytes per second of sustained internal memory bandwidth.

The off-chip memory requirement is for up 8-Gbits of 2 vy 32 LPDDR2 or LPDDR3 DRAM capable of up to 1,066MHz operation. It supports up to six full HD 60 frames per second camera inputs simultaneously via MIPI lanes.

Given its highly parallelized data processing architecture and on-chip memory fabric, Myriad-2 can achieve high-performance processing with low latency.

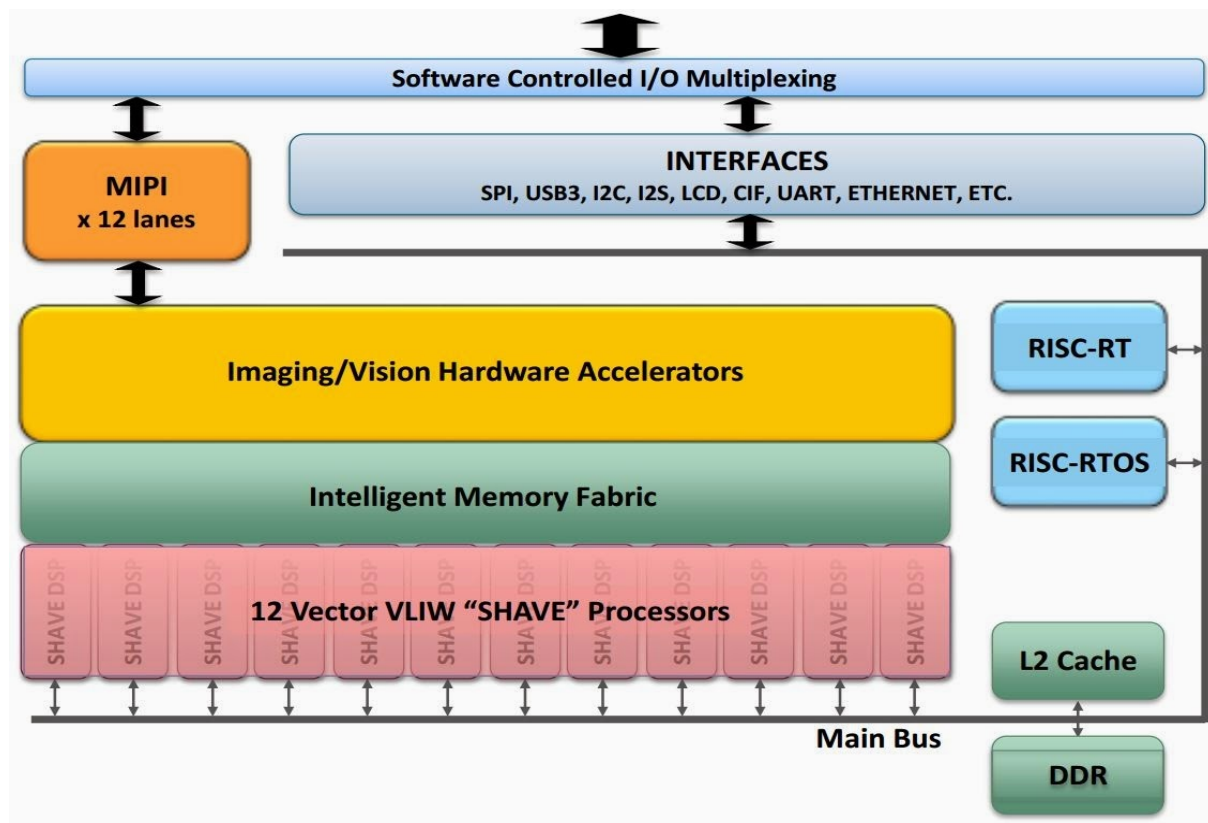


Figure 1. Myriad 2 Block Diagram

RTEMS

The operating system used by the EoT device is RTEMS. This OS is designed for real-time, embedded systems and supports various open API standards. In this project we will focus on POSIX. It also provides a port of the FreeBSD TCP/IP stack and support for several filesystems as the FAT filesystem.

RTEMS does not provide any form of memory management or processes. In POSIX terminology, it implements a single process, multithreaded environment.

This OS is distributed under a modified [GNU General Public License](#) (GPL), allowing linking RTEMS objects with other files without needing the full executable to be covered by the GPL.

■ Myriad 2 programming paradigms

6.3.1. Standard programming paradigm

This first approach concerning Myriad 2 is focused on using the implemented SIPP pipeline. SIPP is a programming paradigm that involves a graph of connected filters in which data is streamed from one filter to the next, on a scanline-by-scanline basis. Images are consumed in raster order not requiring DDR accesses (other than accessing any pipeline input or output images located in DDR, using DMA copies to/from local memory). In addition to the performance and power benefits of avoiding DDR accesses, the design can also reduce hardware costs, allowing stacked DDR to be omitted for certain types of applications.

As the standard programming paradigm for Myriad, this approach involves using RTEMS running on LeonOS and the SIPP scheduler on LeonRT. The advantage of this paradigm is that it provides parallelization in an easy to use environment. The SIPP scheduler itself is able to ensure parallel pipeline configurations for managing the HW filters and exterior interfaces with a low footprint so as to ensure LeonRT optimized utilization. The SIPP used number of SHAVEs is configurable, so any extra number of SHAVEs not used for line based pipelines will remain free to be used by the RTEMS operating system running on LeonOS for various other purposes including computer vision algorithms.

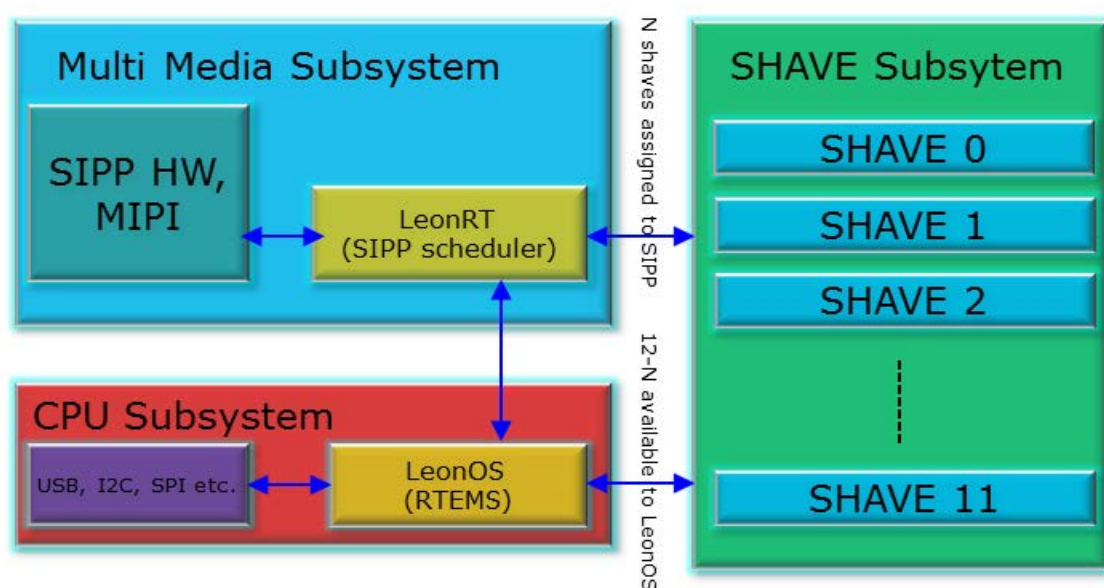


Figure 2. Standard programming paradigm diagram

6.3.2. The One Leon programming paradigm

Some applications might not require heavy line based processing. Such applications might choose to completely switch OFF the LeonRT processor and instead only use LeonOS with (or without) RTEMS. HW filters may still be used. Using this programming paradigm, Leon OS would control all of the applications running on the 12 SHAVE cores.

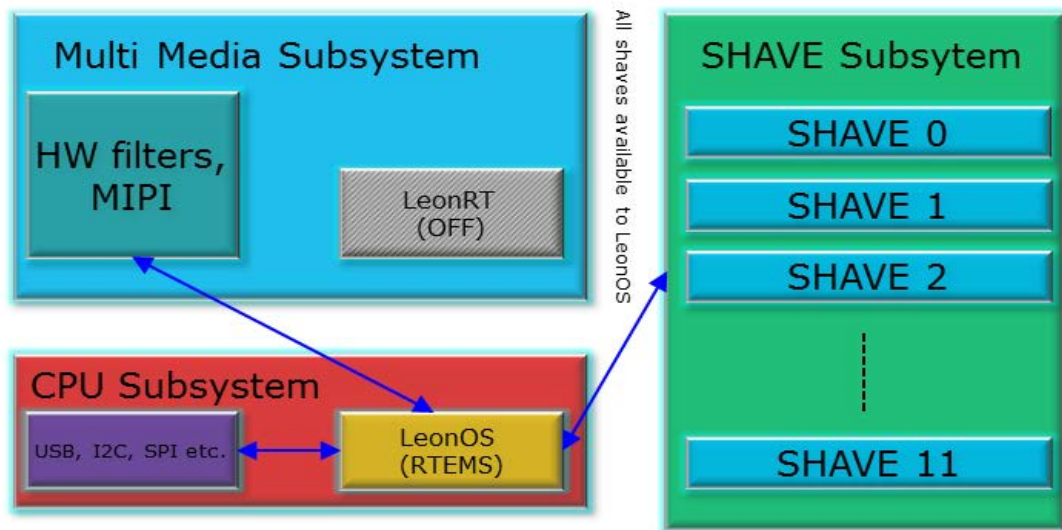


Figure 3. One Leon programming paradigm diagram

6.3.3. Bare metal programming paradigm

A bare metal programming paradigm will also be supported by the MDK build system. This will allow developer to use both LEON cores without any operating system, only minimal schedulers running to control the pipelines application. This paradigm requires more integration efforts but allows developers to write applications which will not be affected by any operating system overhead.

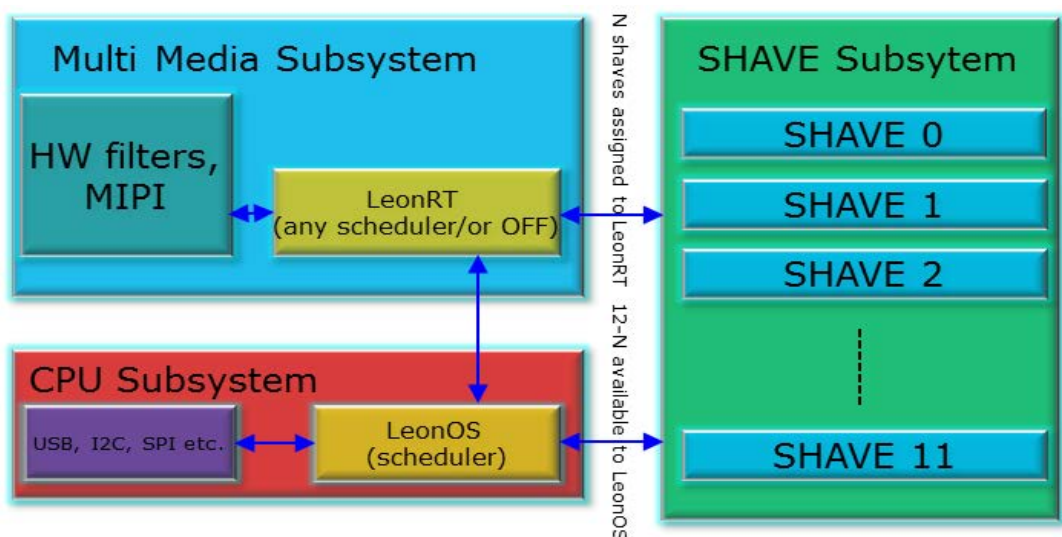


Figure 4. Bare metal programming paradigm diagram

6.3.4. Selected paradigm

With regard to EoT, "one Leon programming paradigm" has been the selected paradigm. This approach allows to deactivate the LeonRT when not necessary, reducing the energy consumption.

EoT firmware

The EoT firmware is divided in the following modules:

- **Wifi data transfer (Section 7).** This module provides a layer of functions for managing the WiFi chip.
- **Camera interface (Section 8).** This module provides software on top of the camera driver for retrieving frames.
- **Video streaming (Section 9).** This module provides an application for streaming the video captured through the camera interface.
- **Input buttons/DIP switches (Section 10).** This module provides functionality to check the status of the DIP switches and user pushbuttons in the board. It also provides functions to turn the two user LEDs on/off.
- **SD card management (Section 11).** This module provide the functionality for reading and writing data to the SD card.
- **Bootloader (Section 12).** This module allows the EoT board to be started on control mode or to run another uploaded application.
- **Control mode API, embedded side (Section 13).** This application is used for configuring and managing the EoT board.
- **Audio input & output (Section 14).** This module provides the functionality for reproducing a stored audio file.
- **Computer vision: CNN (Section 15).** This module includes an implementation of Convolutional Neural Networks for the EoT device.
- **Computer vision: Color histogram matching (Section 16).** This module includes an implementation of color histogram matching algorithms for the EoT device.
- **Computer vision: Keypoint matching (Section 17).** This module includes an implementation of keypoint matching algorithms for the EoT device
- **Computer vision: Rotation-invariant face detector (Section 18).** This module includes an implementation of rotation-invariant face detection algorithms for the EoT device.
- **Computer vision: Sparse optical flow (LK point tracking) (Section 19).** This module includes an implementation of sparse optical flow algorithms for the EoT device.
- **Power management (Section 20).** This module can be used in an application for placing the Myriad chip into a low-power state.
- **Control mode API, Desktop (Section 21).** This is a JAVA library and application for desktop PC that provides functions for interaction with the "Control Mode API embedded side" module.
- **Control mode API, Android (Section 22).** This is an Android library and application for mobile phones that provides functions for interaction with the "Control Mode API embedded side" module.

- **Other vision libraries (Section 23).** This module provides ports of several pre-existing vision libraries to the EoT platform.
- **Motor control (Section 24).** This module provides communication with motors.

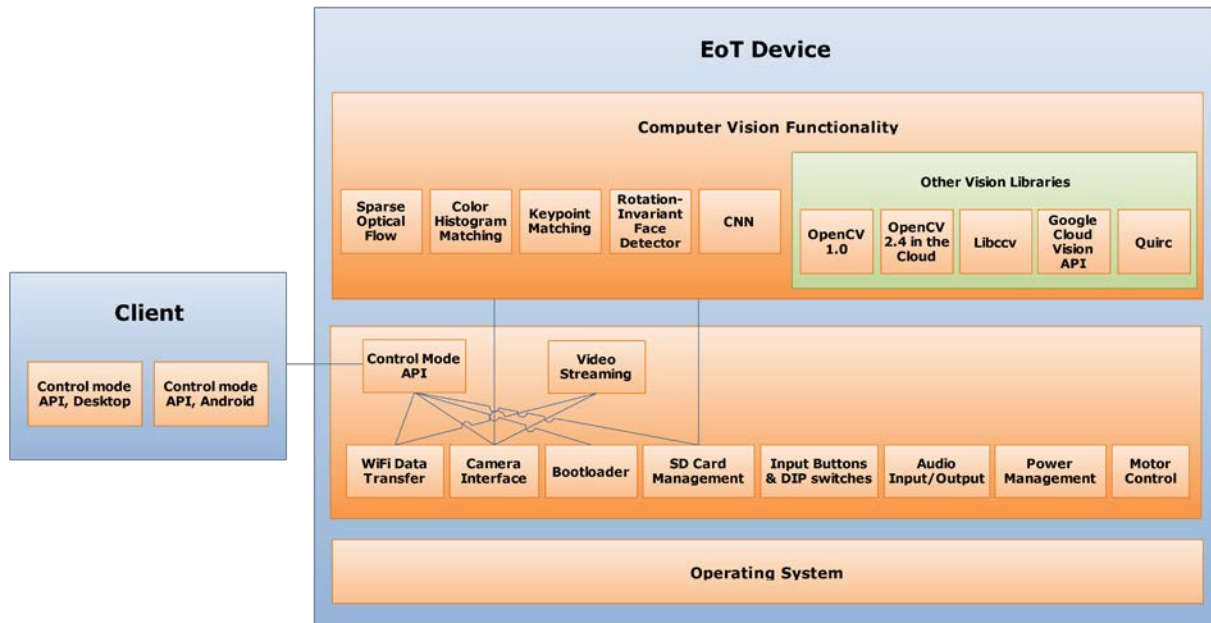


Figure 5. EoT firmware structure

EoT repository

The EoT repository is hosted in the following link:
<https://gitlab.com/espiaran/EoT>.

In this repository there is a first level referring to the 'Development platform' (Myriad, Desktop or Android). In turn, inside each platform there are directories for libraries (libs), unit tests (unittests) and examples and applications (apps). On the root directory there is also a documentation directory (doc) to keep any necessary documents. The structure graph is as follows:

```

root
|-----> doc
|-----> myriad
|         |-----> apps
|         |         |-----> sdcard_example
|         |         |-----> wifi_example
|         |         |-----> pulga_control_app
|         |         |-----> ...
|         |-----> libs
|         |         |-----> SDCardIO
|         |         |-----> Crypto
|         |         |-----> ...
|         |-----> tools
|         |         |-----> flasher
|         |-----> unittest

```

```

|----->sdcard
|----->crypto
|-----> ...
|
|-----> desktop
|-----> apps
|-----> libs
|-----> unittest
|
|-----> ...

```

Currently the repository contains 369642 lines of code. The cloc application (<http://cloc.sourceforge.net>) has been used to calculate this statistic. It is worth noting that some of the libraries have been ported, not implemented from scratch. The complete output of the program is:

```

cloc EoT --exclude-lang=CUDA,HTML,Groovy,Prolog,XML,HTML
1343 text files.
1195 unique files.
1400 files ignored.

```

<http://cloc.sourceforge.net> v 1.60 T=4.72 s (222.7 files/s, 124714.3 lines/s)

Language	files	blank	comment	code
C	512	32769	82365	217049
C++	174	26625	29754	106192
C/C++ Header	224	10495	27095	33886
Java	97	2107	7119	10872
make	41	358	693	1187
Python	3	113	67	441
IDL	1	2	0	15
SUM:	1052	72469	147093	369642

7. WIFI DATA TRANSFER

Introduction

In this task, software has been developed on top of the SPI driver, developed by Movidius, for the WiFi module CC3100MOD from Texas Instruments¹. This software provides a layer of functions for creating an ad-hoc WiFi, establishing a connection with another device using the desktop and mobile APIs defined in Tasks "Middleware API Desktop" and "Middleware API Android", sending/receiving data (of any kind) and closing connection. Moreover, the application developed in Task "Control Mode" directly depends on this.



Figure 6. TI CC3100MOD WiFi module

The CC3100MOD can create an ad-hoc network and has security and encryption (WPA2). The ad-hoc WiFi allows connection with the external configuration computer even without WiFi infrastructure. Furthermore, since the SSID is public, an additional security feature is used: a network password. When using WPA, the password is a string between 8 and 63 characters long. When in place, this password is necessary to use the EoT device. The password is stored in the CC3100MOD's flash memory and can be eventually changed (or removed) from EoT's Control mode.

The result of this task is a library called WifiFunctions, which is an abstraction layer between the Simplelink WiFi driver and the programmer. The functionality provided, apart from the Simplelink driver functions (check CC3100 programmer's guide at <http://www.ti.com/lit/ug/swru368a/swru368a.pdf> for more information), is as follows:

- Generation of access point.
- Connection to existing access points.
- Scanning of the WiFi spectrum to find the less saturated channel.

¹ see [http://www.ti.com/product/cc3100mod?keyMatch=CC3100MOD&tisearch= Search-EN](http://www.ti.com/product/cc3100mod?keyMatch=CC3100MOD&tisearch=Search-EN)

- Profile management for saving a previously generated access point, and reuse it when the device is restarted.
- Ping.
- Change own MAC address
- Set WiFi signal intensity and power policy

Known issues

- The Wifi chip only supports one client connection in AP mode. This was first observed by UCLM. This is not indicated neither in the Wifi manuals nor in the TI web. It is only mentioned in one post in the TI forum. We assume only one client will make the first connection to the EoT device. This, in fact, can be considered more secure. More clients can connect to the EoT device after connection to an external Wifi.
- Potential conflict detected with FlashIO in the first hardware setup proposed over the SPI bus. This conflict is solved with EoT Rev1 board.
- Nonblocking calls in the first version of WiFi driver do not work in server-related functions (accept and select, the timeout parameter does not work, so these calls block forever). Movidius subcontractor Emdalo sent additional code that uses the non-blocking call along with a wait, and another thread to signal the end of the wait, but this hack does not solve the problem completely. It will be solved in future versions.

Unit tests

The unit test for WifiFunctions can be found in the repository in the WorkPackage_3\myriad\unittests\wifiFunctions folder.

The following tests are performed:

- Save and restore a WiFi profile.
- Restore a WiFi profile when no profile has been saved previously.
- Generate an Access Point using the default configuration.
- Generate an Access Point using a saved profile.
- Change the WiFi device to Station mode (mode necessary to connect to other access point).
- Change the WiFi device to access point mode (mode necessary to generate an access point).

Expected output

The expected output of the tests must be similar to:

```
UART: =====
UART: Starting Wifi Functions test app
UART: =====
UART: .test Save Restore Profile OK
UART: .test Generate AP From Profile Index OK
UART: .Failed to get a profile
UART: No profile found on index 0
```

```
UART: test Save Restore Profile Error OK
UART: .test Generate AP From Default Profile OK
UART: .test set mode to Station OK
UART: .test set mode to AP OK
UART:
UART: OK (6 tests)
UART:
LOS: Application terminated successfully.
Batch execute: <echo $exit_opt>
exit
Batch execute: <$exit_opt>
```

Licensing

7.3.1. Simplelink library

simplelink.h - CC31xx/CC32xx Host Driver Implementation

Copyright (C) 2014 Texas Instruments Incorporated - <http://www.ti.com/>

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Texas Instruments Incorporated nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Code

The WifiFunctions library can be found in folder WorkPackage_3\myriad\libs\leon\WifiFunctions of EoT repository. An example using this library can be found in WorkPackage_3\myriad\apps\wifi_example.

The documentation for the relevant code is in Annex 1.

Conclusions and Future work

The implemented library WifiFunctions allows the user to manage all the parameters of the CC3100 module of the EoT board, including the functions of the Simplelink driver, and providing a set of functions which that reduces the programming complexity of developing software for the EoT platform.

This module will be used in every task of the project which needs communication between the EoT board and another device via WiFi. Currently it is used in other applications of EoT as the video streaming application and the control software tool.

Furthermore, the library allows to be expanded including new functionalities. This fact is important for the following steps of the project, in which demonstrators will develop new applications with new requirements that could require new methods related to the WiFi management or configuration.

8. CAMERA INTERFACE

Introduction

To provide an easy-to-use interface to connect to the camera and retrieve frames, a camera module has been developed on top of the more complex MDK camera driver (CamGeneric module).

In the absence of the NanEye camera, this module uses one of the black and white MIPI cameras in the development board (CAM_B1), reducing image size to (480x256) (See Figure 7). Since the NanEye will be supported by the CamGeneric module, changes will be added to this module but always retaining the previous functionality with the devel board camera, so that the camera to use is selected with, say, a parameter flag in the code at compile time.

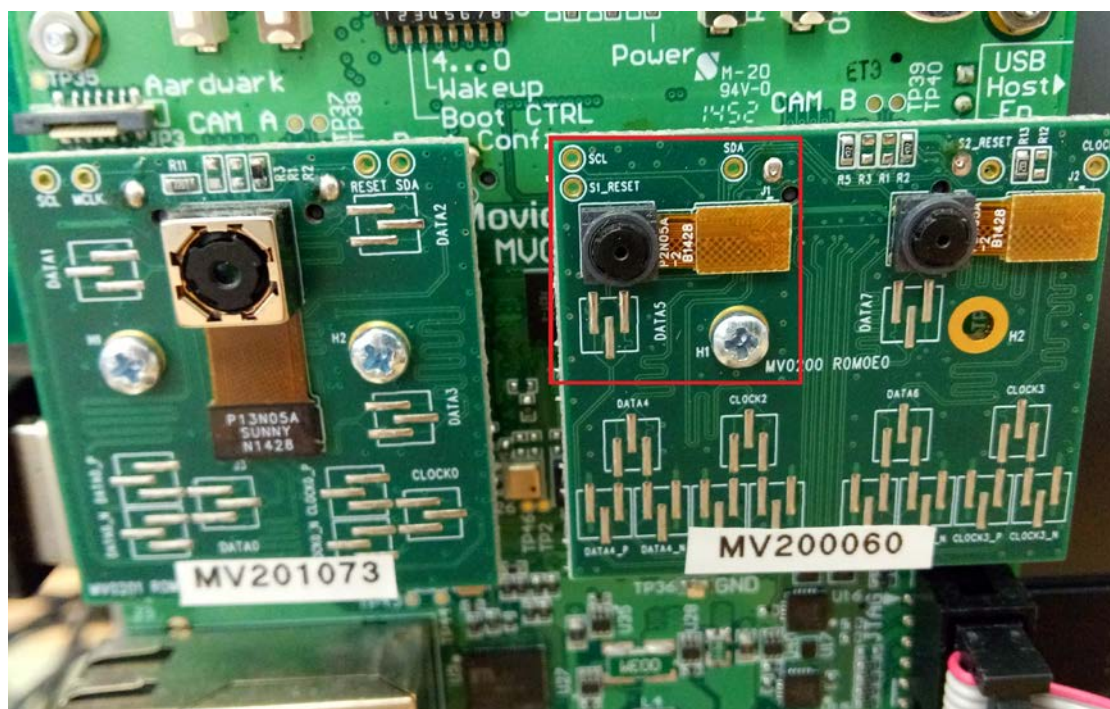


Figure 7. MIPI camera used for development

Using CamGeneric, the camera should be used according to the following scenario:

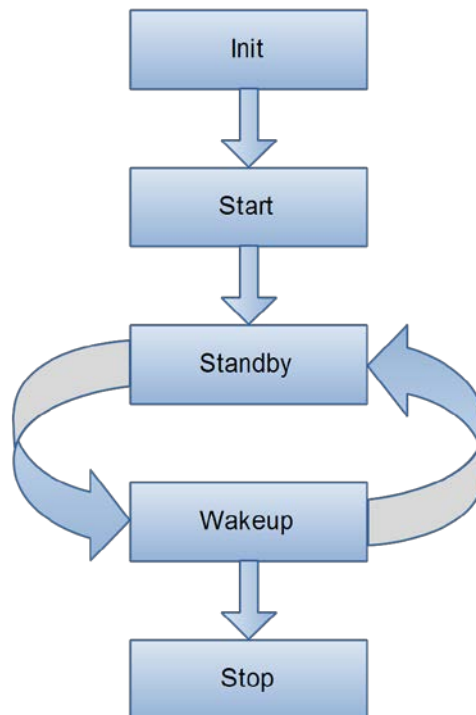


Figure 8. Camera execution diagram

In this case, the camera cannot be started and stopped more than one time. However its standby status can be used to simulate this behaviour and save power. There are two standby configurations:

- HOT STANDBY: the sensor is deactivated but still configured. This standby type is mostly used to save processor time by suspending the interrupts and is fast to recover from (activation last less than 0.5 milliseconds)
- COLD STANDBY: Wakeup out of this state implies full sensor reconfiguration. The wake up duration may last tens of milliseconds.

The EoT Camera module supports the HOT STANDBY option. In computer vision applications, the camera should be active only when needed.

The camera interface provides frames in a simple format:

```
typedef struct frameElements
{
    frameSpec spec;
    unsigned char* p1; // Pointer to first image plane
    unsigned char* p2; // Pointer to second image plane
    unsigned char* p3; // Pointer to third image
} frameBuffer;
```

Where spec defines the type of frame (height, width, ...). These structures are defined in file swcFrameTypes.h.

Since JPEG is one of the most used image compression standards, JPEG compression support has been added to this module. The Control Mode application (i.e. Pulga) and the RTSP streaming applications send the JPEG-compressed image obtained from the camera.

Known issues

The Camera module operates using the I2C buses for the communication with sensors. The I2C configuration function, *BoardInitialise*, sets up all GPIOs and performs the initialization of basic functions of the board and the external clock generator. In addition, this module needs to configure the interrupts and the interrupt service routine (ISR) to periodically update the new buffer where the frame is stored.

The above aspects should be taken into account when using this module together with module WifiFunctions. I2C buses should be initialised before starting the WiFi configuration because WifiFunctions uses the I2C buses to send the nHIB signal to the WiFi chip. The WifiFunctions I2C configuration does not affect the previous I2C configuration of the Camera module. On the contrary, using *BoardInitialise* after initialising the WiFi chip drops the communication between the main processor and the WiFi. With regard to the Camera ISR, notice that the WiFi driver disables all interrupts during WiFi chip initialization. Therefore, initialising the WiFi while the camera is capturing frames can block the interrupt that updates the image buffer. This problem can be solved by calling the *init_camera* method after initialising the WiFi chip.

The following steps show the order of instructions to use WifiFunctions and Camera modules together without deadlocks:

1. The following lines should be added as part of the board initialization. A good practice is to add them at the end of the *initClocksAndMemory* function in the *app_config.c* file of the application.

```
s32 boardStatus;  
boardStatus = BoardInitialise(EXT_PLL_CFG_148_24_24MHZ);  
if(boardStatus != B_SUCCESS ) {  
    printf("Error: board initialization failed with %d  
    status\n",boardStatus);  
    return -1;  
}
```
2. Call *generateAPFromProfileOnErrorDefault(0)* to create an AP with the default parameters.
3. Call *init_camera()* or other methods that call it in a new thread in order to capture images.
4. Rest of the code.

Finally, the camera module has been used in applications running in the LeonOS processor. This keeps the LeonRT processor off reducing the power consumption of the device. In order to use the LeonRT processor for camera streaming applications, synchronization mechanisms should be used to ensure the correct sequence of initialization operations as described above.

Unit tests

There are not specific tests for this module. However, two applications use it: *Pulga_Control_App* and *RTEMS_RTSP_Camera* and *Pulga_test* includes a test for

retrieving a camera snapshot. These applications can be found in folder WorkPackage_3\myriad\ of EoT repository.

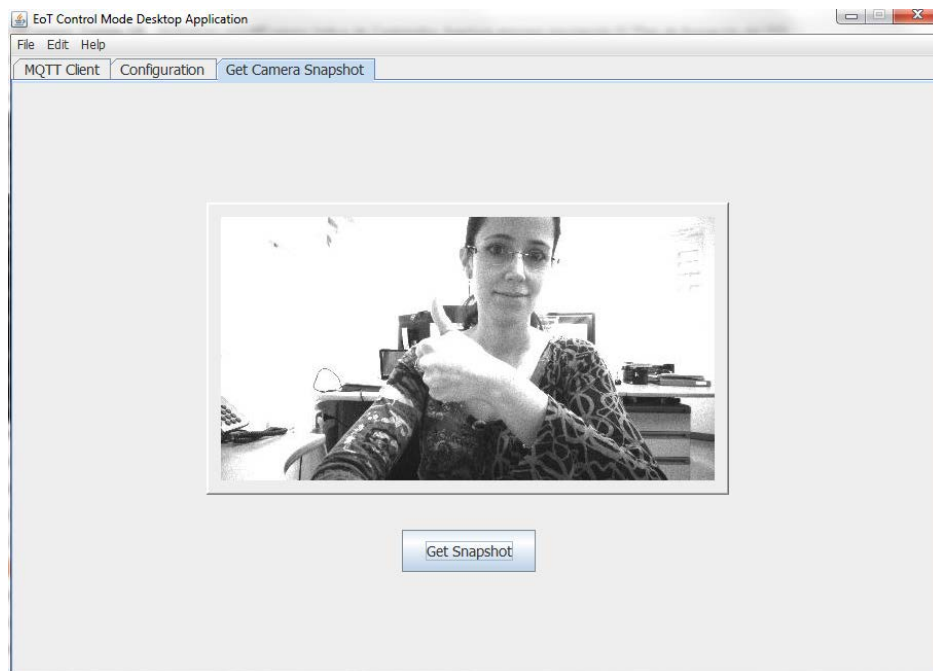


Figure 9. Camera frame retrieved from Pulga

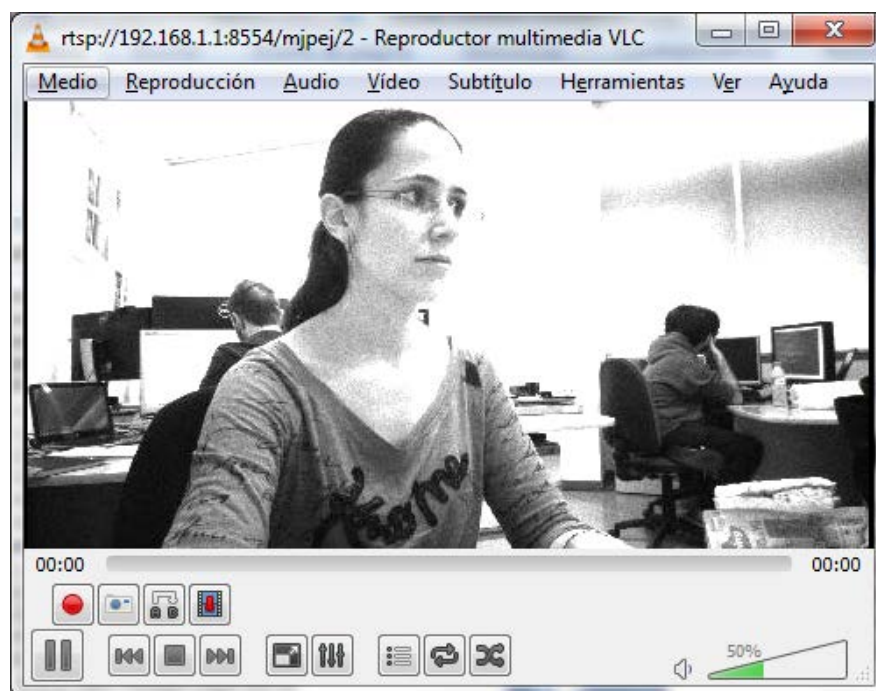


Figure 10. RTSP video streaming

Licensing

The code used to develop the JPEG conversion functionality is based on the one from:

Author: bkenwright@xbdev.net
URL: www.xbdev.net Date: 19-01-06

Code

The Camera code can be found in WorkPackage_3\myriad\libs\leon\Camera of EoT repository.

The documentation for the relevant code is in Annex 2.

Conclusions and Future work

A module for easily controlling the camera has been provided. This module is supported by RTEMS and runs in LeonOS. Its functionalities have been used in Pulga_Control_App and RTEMS_RTSP_Camera applications.

9. VIDEO STREAMING

Introduction

The Real-Time Streaming Protocol (RTSP – RFC 2326) has been selected for video streaming. RTSP is the most used protocol for video streaming from IP cameras. This protocol establishes and controls either a single or several time-synchronized streams of continuous media such as audio and video. It does not typically deliver the continuous stream itself. For that task it relies on other protocols such as RTP (RFC 3550).

As the RFC describes, the protocol supports the following operations:

- Retrieval of media from media server.
The client can request a presentation description via HTTP or some other method. If the presentation is being multicast, the presentation description contains the multicast addresses and ports to be used for the continuous media. If the presentation is to be sent only to the client via unicast, the client provides the destination for security reasons.
- Invitation of a media server to a conference.
A media server can be "invited" to join an existing conference, either to play back media into the presentation or to record all or a subset of the media in a presentation. This mode is useful for distributed teaching applications. Several parties in the conference may take turns "pushing the remote control buttons."
- Addition of media to an existing presentation.
Particularly for live presentations, it is useful if the server can tell the client about additional media becoming available.

The first scenario has been implemented in EoT. An RTSP server is running in the device and clients can connect to it and request video streaming.

RTSP is focused on the streaming control only. For that, different methods are defined in the RFC. SETUP, PLAY, RECORD, PAUSE, and TEARDOWN are the specific control methods while others are defined for setting different options of the protocol: DESCRIBE, ANNOUNCE, GET_PARAMETER, OPTIONS, REDIRECT and SET_PARAMETER.

Typically, an RTPS connection works as follows. While the server is running and listening for incoming connections, the client starts the connection. Then RTSP messages ask the server to start the streaming. The server answers with the options of the stream and starts the RTP streaming. Now the client starts receiving the media and can control the streaming status with some of the control methods.

RTSP Server

Since the EoT RTSP only needs to support streaming and its status control, the developed service implements only the most important features needed to work in order to save memory, power and be more efficient.

The RTSP server controls connections and streaming using TCP sockets for the communication with clients. It allows multi-client streaming. In this case, the maximum number of clients is limited by the maximum number of different sockets that the WiFi chip CC3100 can support. Those are the different options implemented once the connection between the server and the client is established:

- OPTIONS
- DESCRIBE
- PLAY
- SETUP
- TEARDOWN

9.2.1. RTSP Server Options

- **OPTIONS** request returns the request types the server can accept.

```
C->S: OPTIONS rtsp://192.168.1.1:8554 RTSP/1.0
      CSeq: 2
      User-Agent: LibVLC/2.1.5 (LIVE555 Streaming Media v2014.05.27)

S->C: RTSP/1.0 200 OK
      CSeq: 2
      Public: DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE
```

- **DESCRIBE** request includes an RTSP URL and the type of reply data that the client can be handled.

```
C->S: DESCRIBE rtsp://192.168.1.1:8554 RTSP/1.0
      CSeq: 3
      User-Agent: LibVLC/2.1.5 (LIVE555 Streaming Media v2014.05.27)
      Accept: application/sdp

S->C: RTSP/1.0 200 OK
      CSeq: 3
      Content-Base: rtsp://192.168.1.1:8554
      Content-Type: application/sdp
      Content-Length: 93

      v=0
      o=- 1191391529 1 IN IP4 192.168.1.1
      s=
      t=0 0
      m=video 0 RTSP/AVP 26
      c=IN IP4 0.0.0.0
```

- **SETUP** request specifies how a single media stream must be transported. This must be done before a PLAY request.

```
C->S: SETUP rtsp://192.168.1.1:8554 RTSP/1.0
      CSeq: 4
      User-Agent: LibVLC/2.1.5 (LIVE555 Streaming Media v2014.05.27)
      Transport: RTP/AVP;unicast;interlaved=0-1
```

```
S->C: RTSP/1.0 200 OK
      CSeq: 4
      Transport: RTP/AVP;unicast;interlaved=0-1
      Session: 805308858
```

- **PLAY** request asks for media streaming.

```
C->S: SETUP rtsp://192.168.1.1:8554 RTSP/1.0
      CSeq: 5
      Range: npt=0.000-
      Session: 805308858
```

```
S->C: RTSP/1.0 200 OK
      CSeq: 5
      Session: 805308858
      RTP-Info: url=rtsp://192.168.1.1:8554
```

- **PAUSE** request temporarily halts media streams until a PLAY request.

```
C->S: PAUSE rtsp://192.168.1.1:8554 RTSP/1.0
      CSeq: 6
      Session: 805308858
```

```
S->C: RTSP/1.0 200 OK
      CSeq: 6
      Session: 805308858
```

- **TEARDOWN**. It terminates the session (stops media streams and frees session data on the server).

```
C->S: TEARDOWN rtsp://192.168.1.1:8554 RTSP/1.0
      CSeq: 7
      Session: 805308858
```

```
S->C: RTSP/1.0 200 OK
      CSeq: 7
```

After receiving the PLAY message, the server starts the stream. The stream is formed by JPEG images taken from the camera using the EoT Camera module (JPEG-compressed video, RFC 2435). If the client wants to restart the stream after a TEARDOWN request, a new connection has to be started.

Supported Players

The EoT RTSP server is compatible with standard clients that allow video streaming over TCP. Both RTSPPlayer for Android and VLC Player for PC (Windows or Linux) have been tested.

9.3.1. RTSPPlayer TCP configuration

In order to configure the Android app RTSPPlayer go to Settings (“Ajustes” in Figure 11) and scroll down to RTSP tunnel option. Then choose TCP as shown in the images.

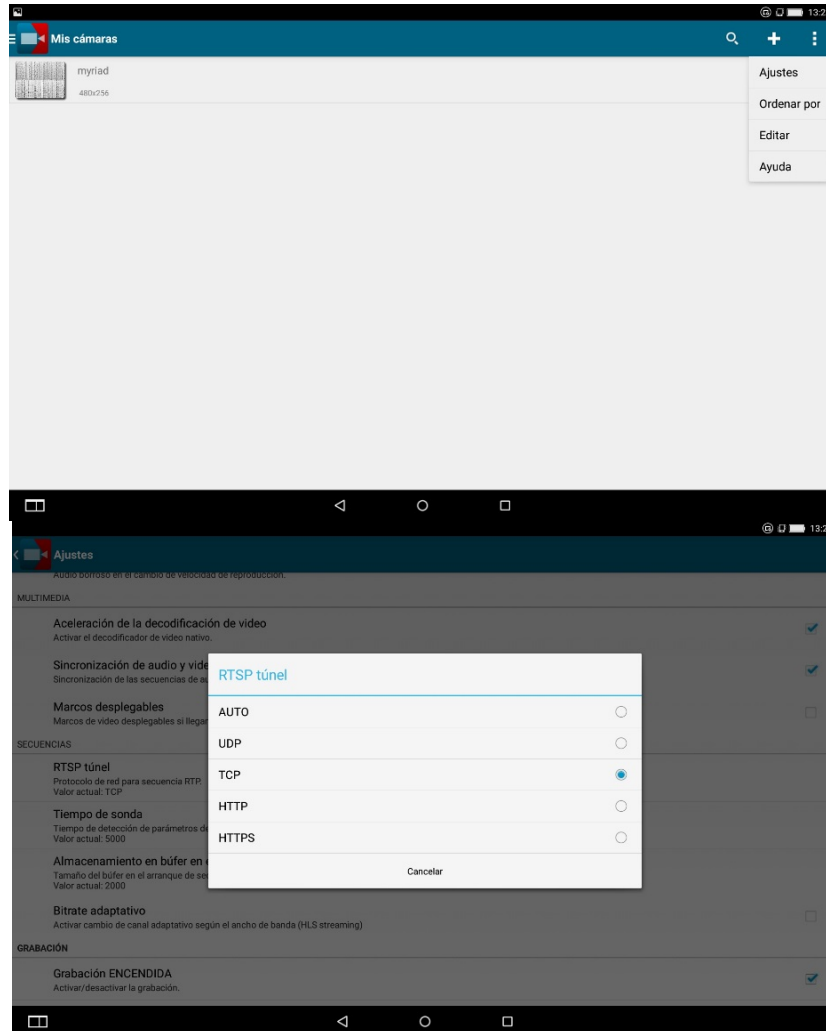


Figure 11. RTSP Player TCP configuration

9.3.2. VLC Player TCP configuration.

To configure VLC Player to work with our server, open preferences in Tools menu (Ctrl+P in Windows) and then in the Codecs tab check RTP over RTSP (TCP) as shown in Figure 12.

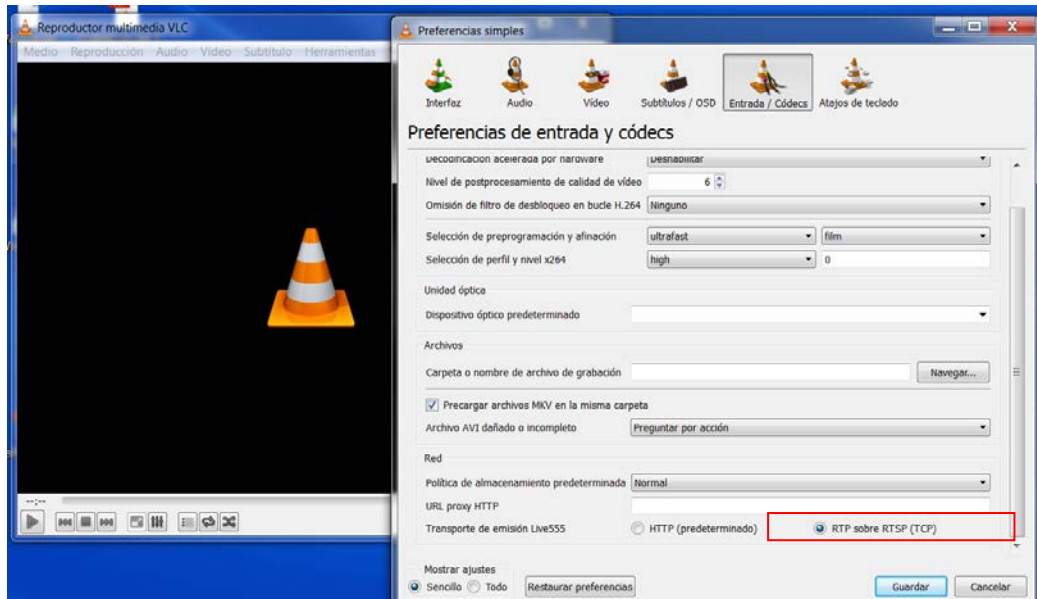


Figure 12. VLC TCP configuration

Known issues

While it is common to use TCP for control and UDP for streaming, the EoT RTSP server uses the same TCP socket for both tasks in order to allow for more clients connected (8 instead of 4) and leave free sockets for other applications.

Unit tests

There are no automatic tests for testing the functionalities of this module. Unit tests for this module would imply to develop an RTSP client. Instead, the RTSP server application, RTEMS_RTSP_Camera in the of EoT repository can be tested with VLC and RSP Player video players.

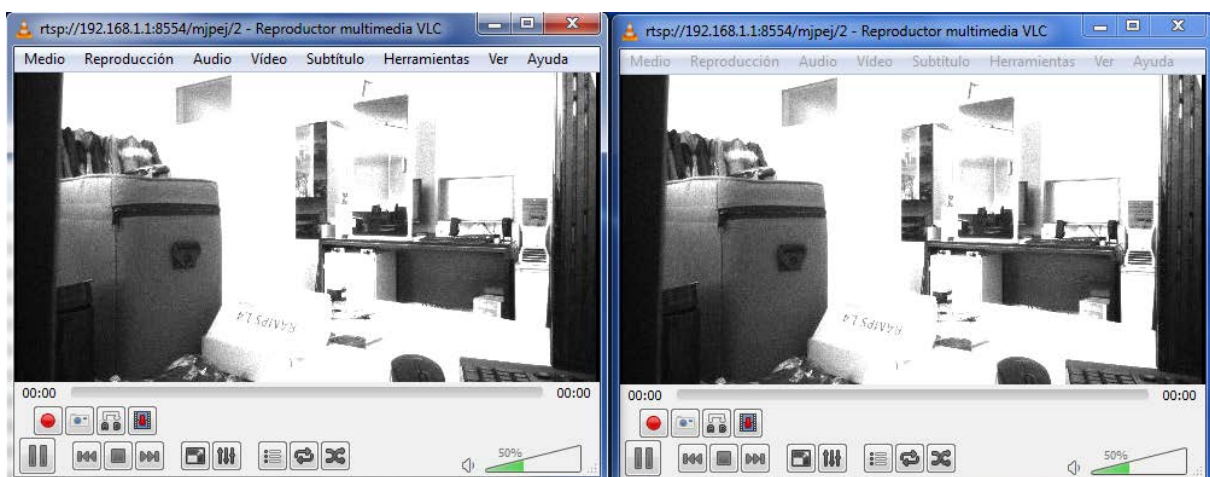


Figure 13. RTSP streaming results with 2 clients

Code

There is an example of how to use the RTSP server (RTEMS_RTSP_camera) in the WorkPackage_3\myriad\apps folder of the EoT repository. This example provides a RTSP server functionality running on RTEMS operating System and uses the Camera module to obtain each frame.

The documentation for the relevant code is in Annex 3.

Conclusions and Future work

The Real-Time Streaming Protocol (RTSP) has been implemented. The image streaming format supported is JPEG-compressed video. The server has all the basic capabilities of an RTSP server and supports streaming to multiple clients.

10. INPUT BUTTONS/DIP SWITCHES

Introduction

This module provides access to user input through push buttons and DIP switches as well as control of user LEDs. Using this module, an application can get minimal user input or signal application specific status. Furthermore, it enables using the DIP switch to decide whether to boot into the control mode or into the installed user app. The interface is fully documented through doxygen comments.

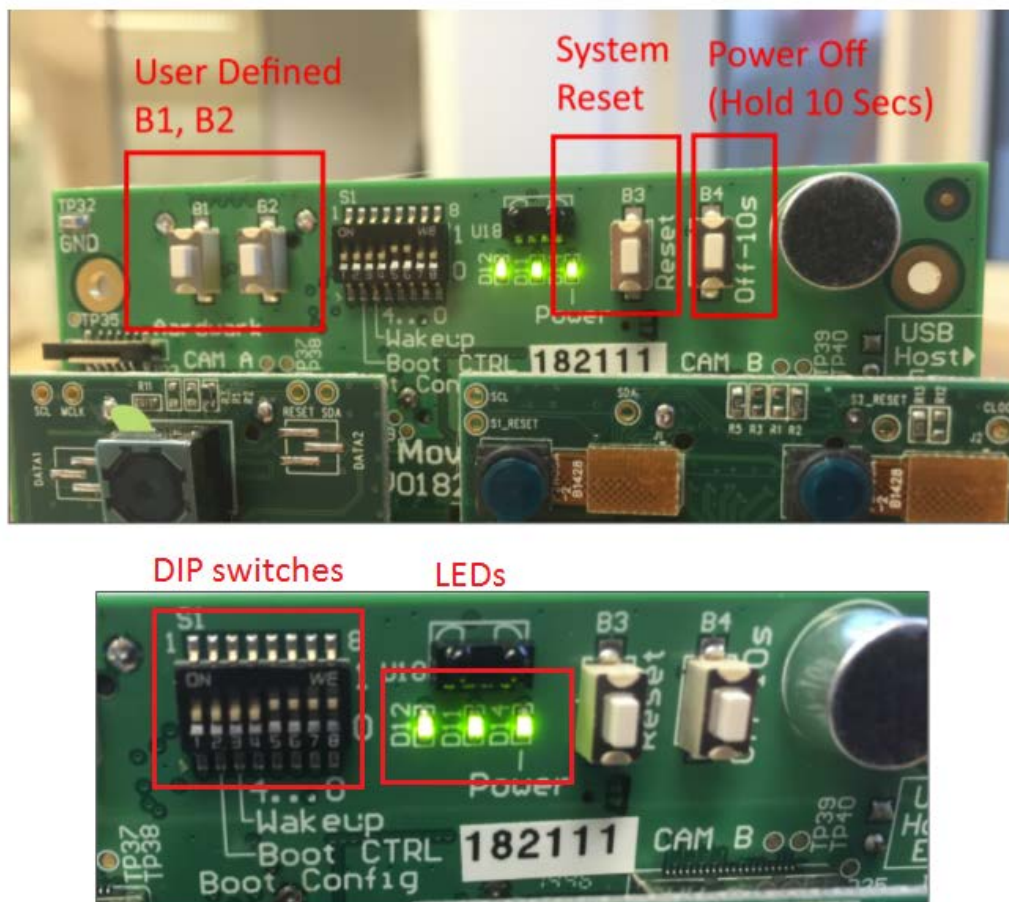


Figure 14. Buttons, DIP switches and LEDs

Unit tests

The functions of this module cannot be tested automatically, since they require user input or feedback.

Code

The code can be found in the GitLab repository of the EoT project at the following address: <https://gitlab.com/espiaran/EoT>.

The specific module is available in "WorkPackage_3/myriad/libs/leon/LEDs". It has no external dependencies.

The documentation for the relevant code is in Annex 4.

Conclusions and Future work

This module provides easy access to query the input buttons, the state of the DIP switch and to control the LEDs. The interface is documented through doxygen. A PDF version of the documentation is available as a separate file.

Although the module exposes all available functionality of the relevant hardware components, one possible future optimization has been identified. In particular, an alternative way of obtaining the state of the push button through the use of interrupts and an interrupt handler could be added. This may provide improved power efficiency in scenarios which make extended use of the push buttons.

11. SD CARD MANAGEMENT

Introduction

This module contains functions to provide access to files and directories on the SD card. It supports file level encryption (only file contents, not metadata) through the use of the EoT library "Crypto". Furthermore, it can also be used to mount and unmount the SD card. The module uses FAT32 as file system and builds on top of file system functionality provided by RTEMS. The module's interface is documented through doxygen comments.

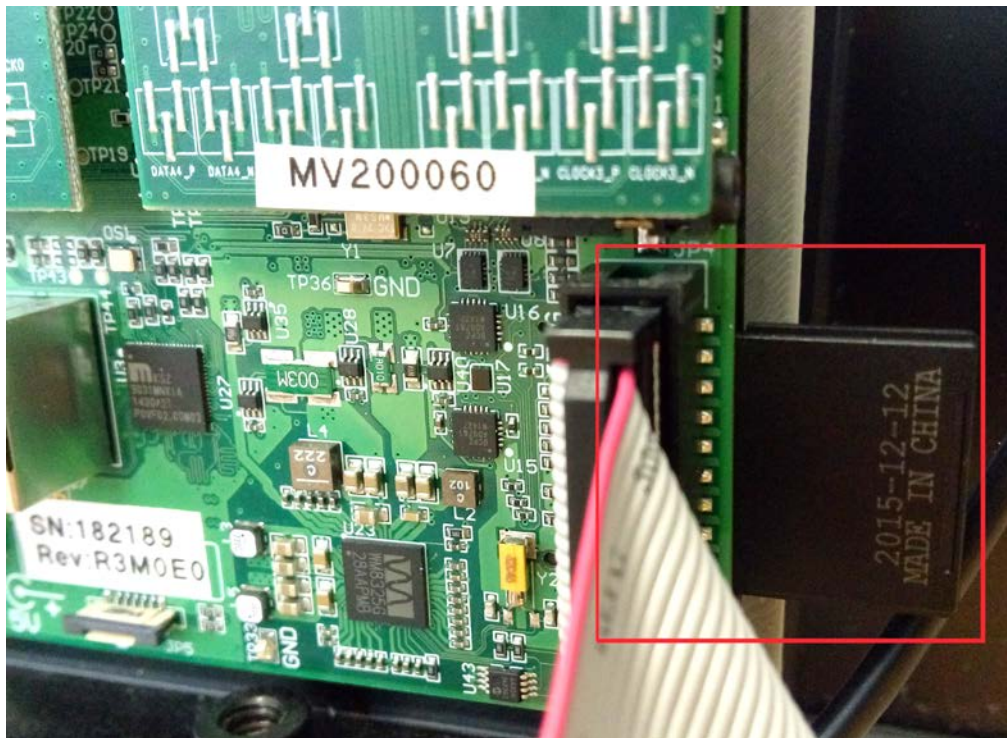


Figure 15. SD card slot

Known issues

There are two known issues which prevent some SD cards from working in the EoT device. The first issue is due to an incompatible formatting of the SD card. Such cases can be resolved by following the formatting instructions in the documentation of the Movidius MDK. The other issue appears to be related to the specific SD card in use. This may possibly be caused either by a defective SD card or a bug in RTEMS.

Unit tests

The unit tests for this module can be found in the following folder of the EoT repository: "WorkPackage_3/myriad/unittests/sdcard". A properly formatted, empty SD card is required for correct test execution. The instructions for formatting the SD card can be found in the MDK documentation from Movidius.

There are three groups of tests. The first group is concerned with testing the mounting and unmounting functionality. The second group tests the functions for accessing and manipulating directories, e.g. creating and deleting directories, listing directory entries, etc. The unit tests will print out on the command line whether they passed or failed. In case of failure, the names of the failed tests will be printed out. Furthermore, there are unit test for the crypto module, which can be found at "WorkPackage_3/myriad/unittests/crypto".

■ Licensing

The unit tests use the embUnit, a unit testing library for embedded systems, which uses the MIT/X Consortium License.

■ Code

The code can be found in the GitLab repository of the EoT project at the following address: <https://gitlab.com/espiaran/EoT>.

The specific module is available in "WorkPackage_3/myriad/libs/leon/SDCardIO". The unit testing library is available in "WorkPackage_3/myriad/libs/leon/embUnit". RTEMS itself is included in the Movidius MDK and also available at rtems.org.

The documentation for the relevant code is in Annexes 5 and 6.

■ Conclusions and Future work

This module implements file and directory management functionality for the SD card. Possible future improvements include a C++ interface (providing the same functionality currently exposed through the C interface) as well as a possible extension of the functionality to format the SD card appropriately in the EoT device. The latter would make it easier and more convenient to use SD cards with the EoT device.

12. BOOTLOADER

Introduction

The bootloader's task is to boot either into 'control mode' or to load an app, which the user has previously installed on the EoT device. The choice what to boot is made through the use of a DIP switch on the device. The bootloader itself, the control mode software as well as the user apps are all stored in the device's flash memory. The bootloader uses two other EoT libraries: FlashIO and ElfLoader. FlashIO provides a minimal filesystem for the flash memory to manage the placement and retrieval of the individual applications (bootloader, control mode and user apps) in the flash memory. Figure 16 shows the flash memory layout that is used. The bootloader is always stored at the beginning of the flash memory. This is necessary to ensure that the bootloader is started when the system boots. The control mode executable is stored at a fixed location after the bootloader. The remaining space can be used to store user apps or data files. Such files are managed through the use of an index table at the end of the flash memory. It stores information about the start and size of files stored in the flash memory. The start of files is always aligned to the size of the flash memory blocks.

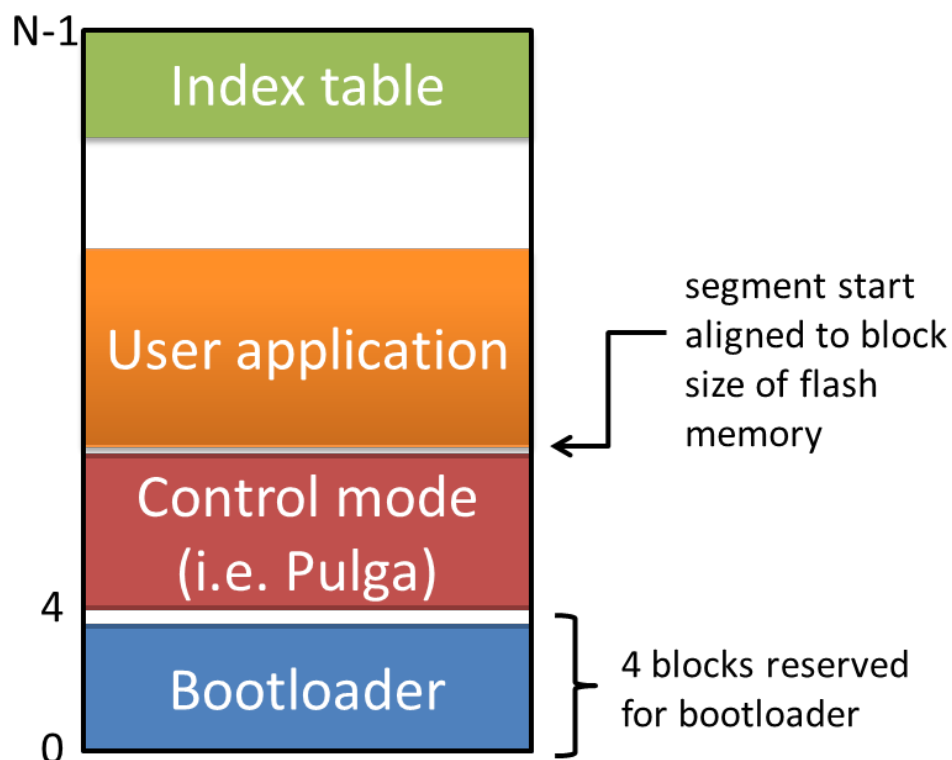


Figure 16. Flash memory layout

The ElfLoader library is used to parse ELF files and load them into memory (ELF files are the binary application files). This way the bootloader can load and start the execution of applications. The API of the ElfLoader and FlashIO module are documented through the use of doxygen comments in the source code.

■ Known issues

There is a known bug in the original development board, which prevents the use of the flash memory, while the WiFi module is connected. The issue will be resolved in board Rev 1.

■ Unit tests

Unit tests for accessing the flash memory through the use of the FlashIO module can be found in the EoT repository in the following folder: "WorkPackage_3/myriad/unittests/flash". There are currently no unit tests for the bootloader code and the ElfLoader module.

■ Code

The code can be found in the GitLab repository of the EoT project at the following address: <https://gitlab.com/espiaran/EoT>.

The specific module is available in "WorkPackage_3/myriad/apps/bootloader". The FlashIO module is available in "WorkPackage_3/myriad/libs/leon/FlashIO" and the ElfLoader in "WorkPackage_3/myriad/libs/leon/ElfLoader". The unit testing library is available in "WorkPackage_3/myriad/libs/leon/embUnit".

The documentation for the relevant code is in Annexes 7 and 8.

■ Conclusions and Future work

This module provides the bootloader, which is used for launching control mode and user apps. To that end, two supporting libraries have been implemented. One handles access to files stored in the flash memory and the other the parsing and loading of ELF executables. Currently, no further improvements are planned.

13. CONTROL MODE API, EMBEDDED SIDE

This module was described in deliverable D3.1.

14. AUDIO INPUT & OUTPUT

■ Introduction

The audio module provides access to the audio chip for playback and recording of audio signals. Furthermore, encoding and decoding of the Opus audio format is supported (a codec is needed for audio streaming). Audio data can be read from and written to files in the OGG container format. The audio chip is configured by this module through the I2C interface and audio data is transmitted through the I2S interface.

■ Licensing

Encoding and decoding of the Opus audio format is based on the reference implementation of the codec. The OGG container format is processed using the libogg reference implementation. Both implementations are covered under the three-clause BSD license. Opus is furthermore covered by several patents which are available under open-source compatible, royalty-free licenses. Details can be found at: <http://opus-codec.org/license/>.

■ Code

The code can be found in the GitLab repository of the EoT project at the following address: <https://gitlab.com/espiaran/EoT>.

The audio module is available in "WorkPackage_3/myriad/libs/leon/Audio". An example application showing its usage is available at "WorkPackage_3/myriad/apps/audio". At the time of writing, this example plays a .wav file stored on the SD card. The audio module also supports audio recording to the SD card. The libraries for the OGG container format and the Opus codec are available at "WorkPackage_3/myriad/libs/leon/Ogg" and "WorkPackage_3/myriad/libs/leon/Opus".

The documentation for the relevant code is in Annex 9.

■ Conclusions and Future work

This module implements reading and writing audio file, encoding and decoding audio data as well as sending and receiving audio data from/to the audio chip. Currently only a sampling rate of 48 kHz is supported. In future work, the module may be extended to also handle lower sampling frequencies. Also, work continues in order to provide a basic live input signal processing capability.

15. COMPUTER VISION: CNN

Introduction

Current methods for recognising subjects in images for video require a lot of hand tuning and other results are quite labour intensive and lacking in generality. Commonly used computer vision technique is histograms of oriented gradients (HoG) which uses histograms of oriented gradients as “visual words” and model the spatial distribution of these elements using a crude spatial pyramid². Such methods can recognize objects correctly without knowing exactly where they are.

By contrast, Convolutional Neural Networks (CNNs) are rapidly replacing existing machine-learning methods typified by HoG/SVM computer vision and machine learning. The driving force behind this move towards deep networks has been the huge gains in accuracy that have been made due to the introduction of CNNs on benchmarks like ImageNet compared to the previous incumbent methods. The broad deployment of CNNs has tracked the massive computational power afforded by the introduction of Graphics Processing Units (GPUs).

	Year	Place	Error (top-5) %	FLOPS	Weights	Watts	Secs
SuperVision	2012		16.40%				
SuperVision	2012	1	15.30%				
ISI	2012		29.98%				
ClariFai	2013		11.70%				
ZF	2013	3	13.51%				
NUS	2013	2	12.95%				
ClariFai	2013	1	11.20%				
Deeper Vision	2014	5	8.11%				
Andrew Howard	2014	4	0.10%				
MSRA	2014	3	7.35%				
VGG	2014	2	7.32%	30940.5	138344128	137.10	1109.96
GoogLeNet	2104	1	6.67%	3426.4	6990272	140.44	399.12
Google	2015		4.82%				
Microsoft	2015	1	4.94%				
DeepImage (Baidu)	2015		4.83%				

As an evolutionary step in neural networks, they are becoming a key method in applications, such as vision processing, handwriting recognition, robotics, and automotive self-driving systems.

² N. & T. B. Dalal, “Histograms of oriented gradients for human detection,” in *Computer Vision and Pattern Recognition (CVPR)*, 2005.

Conceptually CNNs are similar to ordinary Neural Networks in the sense that they are composed of a multitude of neurons with trainable weights and biases. Neurons individually receive some inputs and perform dot products (optionally followed by non-linearity). However, the whole network still introduces a single differentiable score function from the raw input image pixels to the class scores. Figure 17 shows the architecture of a CNN.

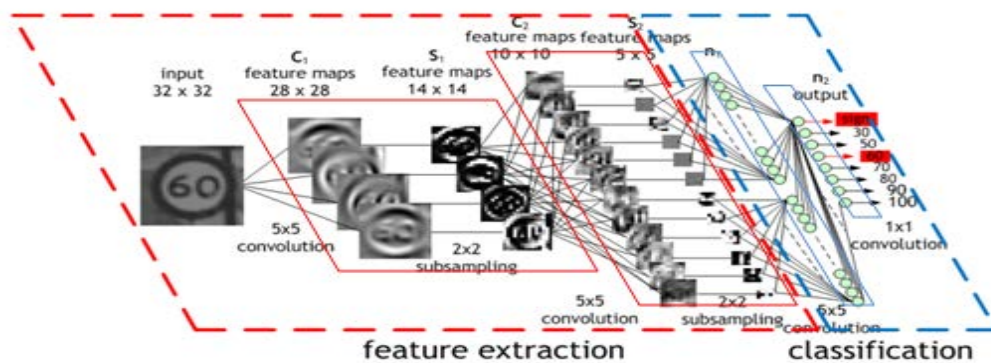


Figure 17. Architecture of a convolutional neural network³

A CNN consists of several layers which can be of four types:

- Convolutional:** This is the layer in the network that extracts feature maps by convolving a few filters across the width and height of the input layer. Every entry in the output then is interpreted as the output of a neuron that looks at only a small region in the input. Every filter consists of a set of learnable filters that are learnt through the back-propagation process. Figure 18 shows the kernel convolution process.

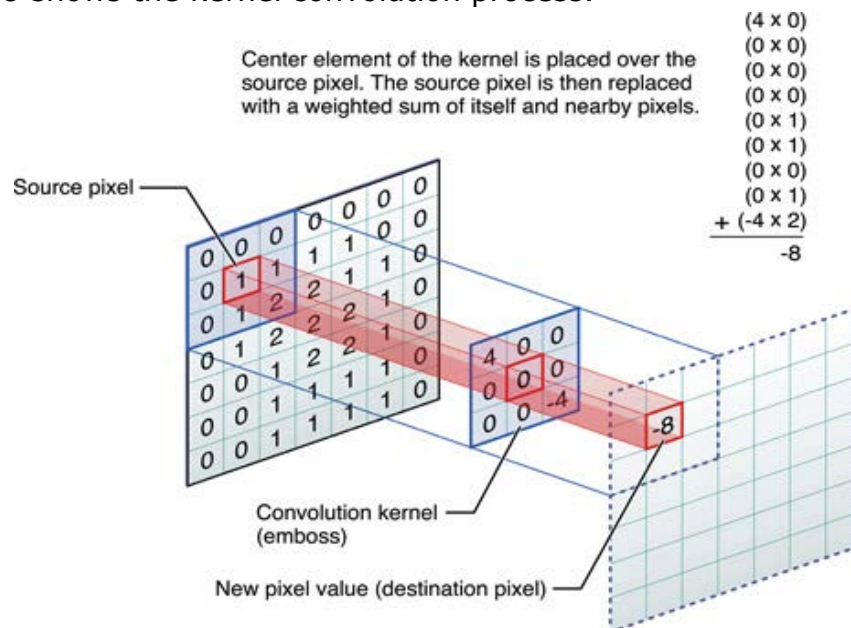


Figure 18. Kernel convolution⁴

³ <https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/>

- **Max-Pooling:** Each convolutional layer may be followed by a pooling layer, which takes small rectangular blocks of the convolutional layer's output and subsamples it to produce a single output from that block. The most popular pooling layer is max-pooling layers that takes the maximum value of entries in the block. However, other operations such as taking the average or a learned linear combination of elements in the block can be used.
- **ReLU:** This rectifier function is an activation function defined as $f(x) = \max(0, x)$, which can be used by neurons in the network. The rectifier activation function is used instead of a linear activation function to add non-linearity to the network. ReLU is used in the network mainly to its efficiency in computation compared to more conventional activation functions like the sigmoid $f(x) = \frac{1}{1+\exp(-x)}$ and hyperbolic tangent $f(x) = \tanh(x)$, without making a significant difference to generalisation accuracy.
- **Fully-Connected:** After several convolutional and max pooling layers, fully connected layers are applied. Similar to ordinary Neural Networks and as it comes from its name, each neuron in this layer is connected to all neurons in the previous layer.

The most common architectures in the field of Convolutional Networks:

- **LeNet.** Yann LeCun developed the first successful applications of CNNs in the 1990's. Of these, the best known is the [LeNet](#) architecture that was used to read zip codes, digits, etc. Figure 19 shows this network applied to digit classification.

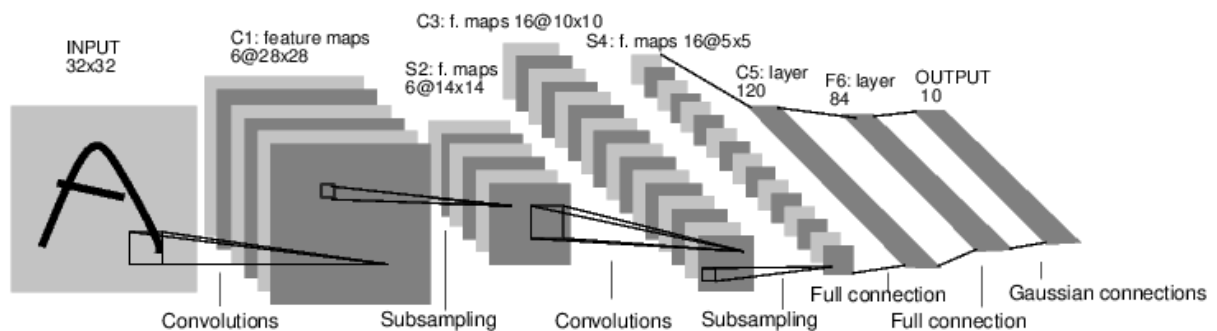


Figure 19. Architecture of LeNet⁵

- **AlexNet.** The first work that popularized CNNs in Computer Vision was [AlexNet⁶](#). AlexNet (Figure 20) was submitted to the [ImageNet ILSVRC](#)

4

<https://developer.apple.com/library/ios/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html>

⁵ Y. B. L. B. Y. & H. P. LeCun, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE 86*, no. 11, 1998.

[challenge](#) in 2012 and significantly outperformed the second runner-up (top 5 error of 16% compared to runner-up with 26% error). The Network had a similar architecture basic as LeNet, but was deeper, bigger, and featured Convolutional Layers stacked on top of each other (previously it was common to only have a single CONV layer immediately followed by a POOL layer).

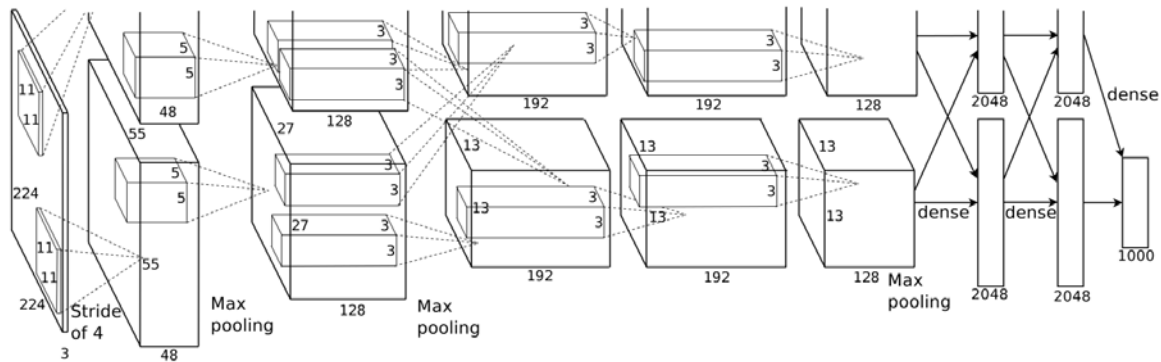


Figure 20. An illustration of the architecture of AlexNet CNN

- ZF Net.** The ILSVRC 2013 winner was a CNN from Matthew Zeiler and Rob Fergus. It became known as the [ZF Net](#)⁷. It was an improvement on AlexNet by tweaking the architecture hyper-parameters, in particular by expanding the size of the middle convolutional layers. Figure 21 shows the architecture of 8-layer convnet model. A 224 by 224 crop of an image (with 3 colour planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form ($6 \cdot 6 \cdot 256 = 9216$ dimensions). The final layer is a C -way softmax function, C being the number of classes. All filters and feature maps are square in shape.

⁶ A. I. S. a. G. E. H. Krizhevsky, "Imagenet classification with deep convolutional neural networks," in *In Advances in neural information processing systems*, 2012.

⁷ M. D. a. R. F. Zeiler, "Visualizing and understanding convolutional networks," in *Computer vision—ECCV*, 2014.

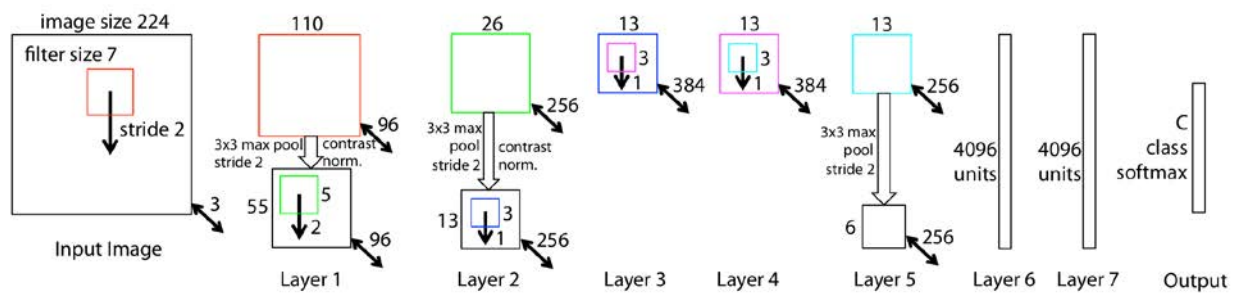


Figure 21. ZF Net architecture

- GoogLeNet.** The ILSVRC 2014 winner was a Convolutional Network from Szegedy et al. from Google⁸. Its main contribution was the development of an *Inception Module* that dramatically reduced the number of parameters in the network (4M, compared to AlexNet with 60M).

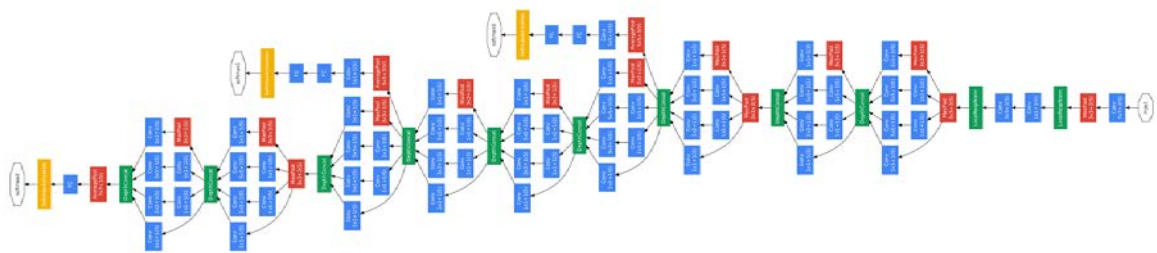


Figure 22. GoogleNet architecture

- VGGNet.** The runner-up in ILSVRC 2014 was the network from Karen Simonyan and Andrew Zisserman that became known as the **VGGNet**⁹. Its main contribution was in showing that ***the depth of the network is a critical component for good performance***. Their final best network contains 16 CONV/FC layers and, appealingly, features an extremely homogeneous architecture that only performs 3x3 convolutions and 2x2 pooling from the beginning to the end. It was later found that despite its slightly weaker classification performance, the VGG ConvNet features outperform those of GoogLeNet in multiple transfer learning tasks. Hence, ***the VGG network is currently the most preferred choice in the community when extracting CNN features from images***. In particular, their **pretrained model** is available for plug and play use in Caffe. **A downside of the VGGNet is**

⁸ C. W. L. Y. J. P. S. S. R. D. A. D. E. V. V. a. A. R. Szegedy, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

⁹ http://www.robots.ox.ac.uk/~vgg/research/very_deep/

that it is more expensive to evaluate and uses a lot more memory and parameters (140M).

■ Movidius Fathom CNN framework

The idea behind the Movidius Fathom (™) framework is to take an existing trained network from a CNN framework like Caffe, Torch7 or TensorFlow and translate and adapt it for use with a Movidius Myriad device. In this way, the current design flows used by deep-learning data-scientists to develop networks are unchanged and they can continue to use the flows, datasets and associated scripts they have used historically without modification to quickly and easily port the networks trained in the cloud on GPUs to the Myriad embedded platform as shown below. Fathom takes a pre-trained network does XML from Caffe and converted to run on Myriad2.

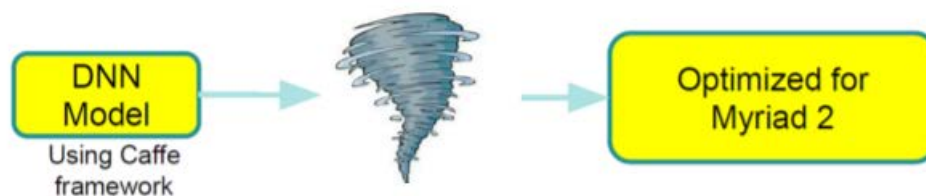


Figure 23. Movidius Tool converts into DNN using Conv/MatMul libraries

The Fathom framework and enabling Tensor high performance CNN libraries integrate with the rest of the MDK as shown below.

The Fathom framework builds on top of the mvTensor library which offers highly efficient 3D/4D convolution optimised for the following use-cases initially: 3x3s1xN, 3x3s2xN, 5x5s1xN, 5x5s2xN, 5x5s3xN. These small stride convolutions are supported in Version 1.0 and are required by GoogleNet. The library uses direct methods and matrix multiplication (mvMatMul) and supports both fp32, fp16 and 8-bit operations sustaining up to 80 GFLOPs at half precision.

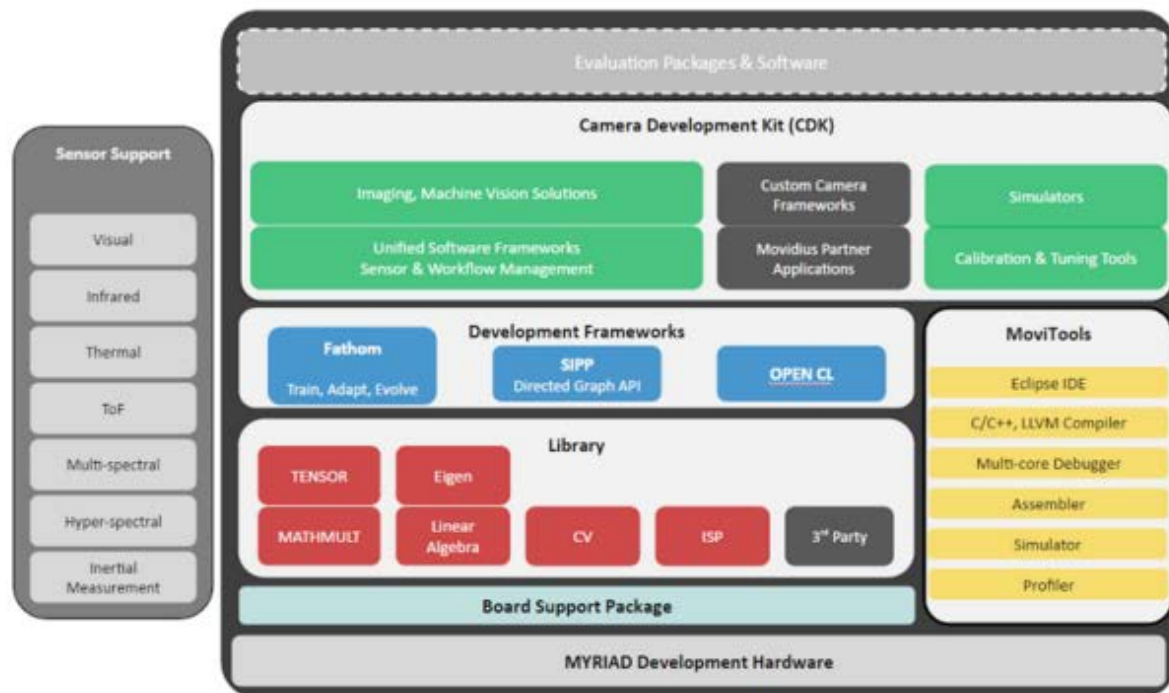


Figure 24. Movidius MDK including Fathom and Tensor CNN support

The initial version 1.0 release of the library is optimised for GoogLeNet execution with high performance and low power by optimally laying out the data and weights in the shared CMX memory with a goal of 15fps operation at less than the maximum TDP of 1.2W moving to 25-30fps over the longer term by leveraging 5x5 convolution hardware and reduced precision operations supported by Myriad2 hardware. As part of the roadmap the intention is to also support newer deep-learning frameworks like TensorFlow.

MvTensor Implementation Details

Multi-channel spatial convolutions: This function satisfies the following requirements:

- Input: a volume of size $W1 \times H1 \times D1$.
- Hyper-parameters:
 - number of filters K ,
 - filters spatial extent F ,
 - stride S ,
 - amount of zero padding P .
- Output: a volume of size $W2 \times H2 \times D2$ where:
 - $W2 = (W1 - F + 2P) / S + 1$
 - $H2 = (H1 - F + 2P) / S + 1$ (i.e. width and height are computed equally by symmetry)
 - $D2 = K$
- With parameter sharing, $F \cdot F \cdot D1$ weights per filter, a total of $(F \cdot F \cdot D1) \cdot K$ weights and K biases.

- The d -th depth slice of output (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d -th filter over the input volume with a stride of S , and then offset by d -th bias.
- A common setting of the hyper-parameters is $F=3$, $S=1$, $P=1$.
- *Prefer a stack of small filter CONV to one large receptive field CONV layer.* Suppose that you stack three 3×3 CONV layers on top of each other (with nonlinearities in between, of course). In this arrangement, each neuron on the first CONV layer has a 3×3 view of the input volume. A neuron on the second CONV layer has a 3×3 view of the first CONV layer, and hence by extension a 5×5 view of the input volume. Similarly, a neuron on the third CONV layer has a 3×3 view of the 2nd CONV layer, and hence a 7×7 view of the input volume. Suppose that instead of these three layers of 3×3 CONV, we only wanted to use a single CONV layer with 7×7 receptive fields. These neurons would have a receptive field size of the input volume that is identical in spatial extent (7×7), but with several disadvantages.
- The neurons would be computing a linear function over the input, while the three stacks of CONV layers contain nonlinearities that make their features more expressive.
- If we suppose that all the volumes have C channels, then it can be seen that the single 7×7 CONV layer would contain $C \times (7 \times 7 \times C) = 49C$ parameters, while the three 3×3 CONV layers would only contain $3 \times (C \times (3 \times 3 \times C)) = 27C$ parameters. Intuitively, stacking CONV layers with tiny filters as opposed to having one CONV layer with big filters allows us to express more powerful features of the input, and with fewer parameters.
- As a practical disadvantage, we might need more memory to hold all the intermediate CONV layer results if we plan to do backpropagation.

Common rules of thumb for convolution function: The CONV should be using small filters (e.g. 3×3 or at most 5×5), using a stride of $S=1$, and crucially, padding the input volume with zeros in such way that the CONV layer does not alter the spatial dimensions of the input. That is, when $F=3$, then using $P=1$ will retain the original size of the input. When $F=5$, $P=2$. For a general F , it can be seen that $P=(F-1)/2$ preserves the input size. If you must use bigger filter sizes (such as 7×7 or so), it is only common to see this on the very first conv layer that is looking at the input image.

Max-pooling: It reduces the spatial size of the input, the amount of parameters, and computation in the network, by operating independently on every depth slice of the input using the Max or Average operations. The most common form of pooling is with size of 2×2 with a stride of 2, discarding 75% of the activations. Max and Average operations take a max or average over 4 numbers, respectively. The depth dimension remains unchanged. More generally, the pooling:

- Input: a volume of size $W_1 \times H_1 \times D_1$.
- Hyper-parameters:
 - their spatial extent F

- stride S
- Output: a volume of size $W2 \times H2 \times D2$, where:
 - $W2 = (W1 - F) / S + 1$
 - $H2 = (H1 - F) / S + 1$
 - $D2 = D1$
- **Zero parameters** (it computes a fixed function of the input).
- Not common to use zero-padding for Pooling layers.
- Two commonly seen variations of the max pooling: pooling with $F=3, S=2$ (also called overlapping pooling), and more commonly pooling with $F=2, S=2$.
- Pooling sizes with larger receptive fields are too destructive.
- Average pooling was often used historically but has recently fallen out of favour compared to the max pooling operation, which has been shown to work better in practice.

ReLU activation operations: It applies an elementwise activation nonlinear function, such as the $\max(0, x)$ thresholding at zero. This operation leaves the size of the input volume unchanged.

GoogleNet Example

In order to leverage their ability to learn complex functions, large amounts of data are required for training. Training a large convolutional network to produce state-of-the-art results can take weeks, even when using modern GPUs. Producing labels using a trained network can also be costly when dealing with web-scale datasets. These all make it more challenging to implement CNN for embedded systems.

The embedded system considered is Myriad2 with 12 x SHAVE VLIW vector processors and 2 x RISC processors. This section describes a specification of the GoogleNet Network on the Movidius Myriad MA2x50 family of processors. It involves two libraries: MvMatMul and MvTensor.

15.4.1. MvMatMul Library

This library is going to provide lower power, high throughput of matrix multiplication. Core operation is: $C = B' * A + C$ (see Figure 25). In CNN these matrices would be:

B	Weight matrix
A	Activation Matrix
C	Bias Matrix and also result

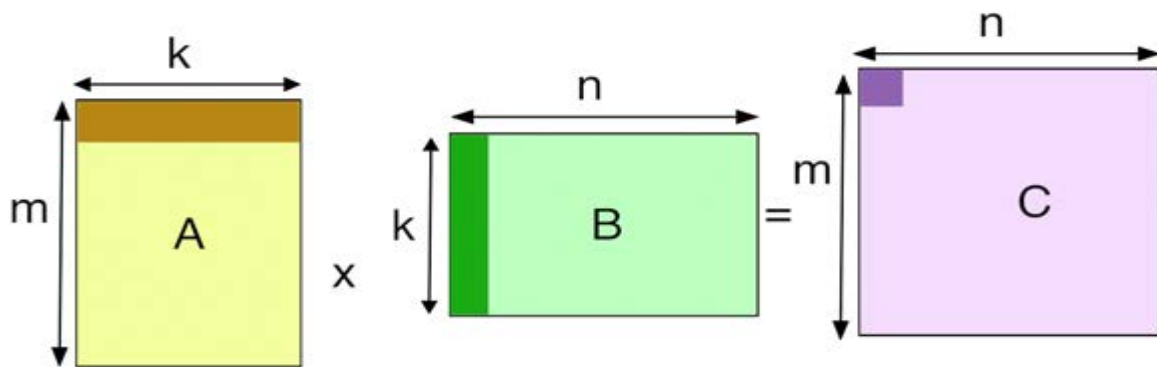


Figure 25. Core operation in MvMatMul Library

15.4.2. MvTensor Library

MvTensor is a library and framework for running tensor math operations efficiently on myriad family silicon. It comprises:

- Leon Runtime.
- SHAVE Code.
- Efficient SHAVE kernels for specific tensor/convolution operations.

Feature sets: it has the following feature sets:

- Run Kx3D Convolutions
- Linear scaling to across 1,2,3, 4, ..., 12 SHAVEs.
- Integrated DMA and data scheduling
 - taps, weights, activations in CMX or DDR.
- Support accumulation into output bias buffer.
- Support inline ReLu and pooling prior to output.

Supported 3D Convolutions: The supported 3D Convolutions are as:

- 3x3s1xN
- 3x3s2xN
- 5x5s1xN
- 5x5s2xN
- 5x5s3xN
- 7x7s1xN
- 7x7s2xN
- 7x7s3xN
- Where N is the number of channels and is 1,2,3 or a multiple of 8.

3D Convolution – Terminology:

- Feature map: an individual 'image' or 2D slice of CNN. Typically, a simple CNN stage will generate K output feature maps from C input feature maps. Feature maps are often called channels.
- Feature - a feature is computed by using a convolution kernel on an input feature map. For example, for a 3x3 convolution layer we say that the feature width is 3 and feature height is 3.

Parameters: The parameters involved in this library are:

- C input maps.
- K output maps
- Fh Feature Height (i.e. kernel width)
- Fw Features Width (i.e. kernel height)
- X feature map width
- Y feature map height

Examples:

- 2D Convolution over image

This is not required of library but good as example:

```
foreach X in mapWidth
  foreach Y in mapHeight
    output[X,Y] = conv2D(input[X,Y], taps)
```

- 3D Convolution over C input feature maps (CNN usecase)

```
foreach C in numInputMaps
  foreach X in mapWidth
    foreach Y in mapHeight
      output[X,Y] += conv2D(input[X,Y,C], taps[C])
```

- Kx3D Convolution to get K output maps from C input maps (CNN usecase)

Conceptually

consider input as 3d Volume C x X x Y
 consider output as 3d Volume K x X x Y
 consider convolution filter as having C x 3 x 3 taps

Example code:

```
foreach K in numOutputMaps
  foreach C in numInputMaps
    foreach X in mapWidth
      foreach Y in mapHeight
        output[X,Y,K] += conv2D(input[X,Y,C], taps[K,C])
```

Notes:

- note that we use different taps for each different layer
- Occasionally this gets called 4D convolution but it's really just a 3D convolution called K times.

15.4.3. MvTensor API Description

The purpose of the MvTensor API (Application Programming Interface) is to standardise communications between the Fathom framework and the underlying linear algebra and non-linear function libraries it relies upon which are implemented in MvTensor. The intent of this abstraction is to allow the Fathom framework to be developed independently of the mvTensor implementation details. This is particularly important as the first implementation of mvTensor will be very much tuned to the implementation of GoogLeNet and based upon existing fp32 and fp16 GEMM, GEMV and ReLU kernels that have already been

implemented for an initial port of LeNet to Myriad2. Abstracting via an API means that as other implementations of the underlying layers that take advantage of hardware acceleration, lower precision arithmetic, coefficient compression etc. are introduced they can be easily taken of advantage of without having to change the xml format used by Fathom as input meaning that existing trained networks can rapidly take advantage of advances in the underlying optimised hardware and software.

15.4.3.1. Scope:

- A single call to mvTensor does one operation of compute across multiple SHAVE processors and schedules data.
- C API.

15.4.3.2. Parameters

Table 1.1 summarizes the parameters required for MvTensor

Table 1.1 MvTensor Parameters.

type	Name	Comment
genData	input	input data structure
genData	output	output data structure
genData	weights	data structure for weights
genData	accumulation	optionally required buffer when precision is different
enum	preOperation	placeholder for now;
enum	op	operation type: kConv1x1 kConv3x3, kConv5x5, kConv7x7, kMaxPool3x3
int	opStrideX	operation stride in X direction
int	opStrideY	operation stride in Y direction
enum	postOp	kRelu, kReluMaxPool3x3, kNone
int	postOpStrideX	post operation stride in X direction
int	postOpStrideY	post operation stride in Y direction
tMyriadResources	firstProcessor	Myriad resources allocated <ul style="list-style-type: none"> • First SHAVE • Last SHAVE • DMA Link agent
tDebug		Debug struct. minimally store layer duration (ms) and error

		message string float ms; char debugmsg[120];
--	--	--

15.4.3.3. genData Container

The data structure is described in Table 1.2.

Table 1.2 Data Structure.

type	Name	Comment
void*	data	data pointer
int	dimX	elements in X dimension
int	dimY	elements in Y dimension
int	dimZ	elements in Z dimension
int	dimXStride	Stride is start of one line to start of next in elements
int	dimYStride	Stride is start of one line to start of next in elements
int	dimZStride	Stride is start of one line to start of next in elements
enum	datatype	fp16, u8f, int
enum	padStyle	zeros
enum	storageOrder	orderYXZ, orderZYZ, orderYZX,

15.4.3.4. Use-cases for storage

This section shows how the specified storage dimensions' map to data structures frequently used in CNN usecase. If a simple CNN stage may be described using the following parameters:

- C - input feature maps (channels)
- K - output feature maps
- X - columns in input feature map
- Y - rows in input feature map
- FW - Hpixels/columns spanned by the feature we are computing
- FH - Vpixels/rows spanned by the feature we are computing

Then they shall be mapped onto the storage structure as shown in Table 1.3.

Table 1.3: Data Storage

Storage Dimension	Activations	Weights	Output
X	X	{FH*FW}	X
Y	Y	C	Y

Z	C	K	K
typical storage order	orderYXZ	orderZYZ	orderYXZ

15.4.3.5. Addressing worked examples

Table 1.4 describes how data are addressed for different storage orders.

Table 1.4: Addressing worked examples

Storage Order	Address of element x,y,z
orderYXZ (Channel Minor)	$\text{data} + (y * \text{dimXStride} * \text{dimZStride} + x * \text{dimZStride} + z) * \text{sizeof}(\text{element})$
orderZYZ (column minor)	$\text{data} + (z * \text{dimYStride} * \text{dimXStride} + y * \text{dimXStride} + x) * \text{sizeof}(\text{element})$

15.4.4. Fathom Data Structure

15.4.4.1. t_MvTensorDebugInfo Struct Reference

Debug messages table.

```
#include <mvTensor.h>
```

Data Fields

- double [ms](#): Duration of the mvTensor call (in ms)
- char [debugMsg \[MV_TENSOR_DBG_MSG_SIZE\]](#): Debug messages.

Field Documentation: generated from the following file: `mvTensor.h`

- char [t_MvTensorDebugInfo::debugMsg\[MV_TENSOR_DBG_MSG_SIZE\]](#): Debug messages.
- double [t_MvTensorDebugInfo::ms](#) Duration of the mvTensor call (in ms).

15.4.4.2. t_mvTensorGenData Struct Reference

Basic data structure.

```
#include <mvTensor.h>
```

Data Fields

- void * [data](#): Data Pointer.
- int [dimX](#): Elements in x-dimension.
- int [dimY](#): Elements in y-dimension.
- int [dimZ](#): Elements in z-dimension.
- int [dimXStride](#): Stride (in bytes) in the x-direction.
- int [dimYStride](#): Stride (in bytes) in the y-direction.
- int [dimZStride](#): Stride (in bytes) in the z-direction.
- [t_MvTensorDataType datatype](#): Data type.
- [t_MvTensorPaddStyle paddStyle](#): Padding style.

- `t_MvTensorStorageOrder storageOrder`: Data storage order.

Field Documentation: generated from the following file "mvTensor.h"

- `void* t_mvTensorGenData::data`: Data Pointer.
- `t_MvTensorDataType t_mvTensorGenData::datatype`: Data type.
- `int t_mvTensorGenData::dimX`: Elements in x-dimension.
- `int t_mvTensorGenData::dimXStride`: Stride (in bytes) in the x-direction.
- `int t_mvTensorGenData::dimY`: Elements in y-dimension.
- `int t_mvTensorGenData::dimYStride`: Stride (in bytes) in the y-direction.
- `int t_mvTensorGenData::dimZ`: Elements in z-dimension.
- `int t_mvTensorGenData::dimZStride`: Stride (in bytes) in the z-direction.
- `t_MvTensorPaddStyle t_mvTensorGenData::paddStyle`: Padding style.
- `t_MvTensorStorageOrder t_mvTensorGenData::storageOrder`: Data storage order.

15.4.4.3. t_MvTensorMyriadResources Struct Reference

Myriad resources structure.

```
#include <mvTensor.h>
```

Data Fields

- `int firstShave`: Index of the first SHAVE.
- `int lastShave`: Index of the last SHAVE.
- `int dmaLinkAgent`: Link to the Direct Memory Access Agent.

Detailed Description

Myriad resources structure.

Field Documentation: generated from the following file "mvTensor.h":

- `int t_MvTensorMyriadResources::dmaLinkAgent` Link to the Direct Memory Access Agent.
- `int t_MvTensorMyriadResources::firstShave` Index of the first SHAVE.
- `int t_MvTensorMyriadResources::lastShave` Index of the last SHAVE.

15.4.4.4. t_MvTensorParam Struct Reference

MvTensor global parameters structure.

```
#include <mvTensor.h>
```

Data Fields

- `t_mvTensorGenData input`: Input data structure: pointer to input data, width, height, horizontal and vertical stride, data type and storage order.
- `t_mvTensorGenData output`: Output data structure: same type of information as for input.
- `t_mvTensorGenData weights`: Weights data structure: same type of information as for input.

- [t_mvTensorGenData accumulation](#): Intermediate data structure, needed only when operation needs to be done on another data type then the input/output are.
- [u32 preOperation](#): Pre-processing operation: TODO create enum type for this field.
- [t_MvTensorOp op](#): Filtering operation: Convolution, max-pooling.
- [int opStrideX](#): Operation stride in the X-direction.
- [int opStrideY](#): Operation stride in the Y-direction.
- [t_MvTensorPostOp postop](#): Post-operation: Relu.
- [int postOpStrideX](#): Stride in the X-direction for the post-processing operation.
- [int postOpStrideY](#): Stride in the Y-direction for the post-processing operation.
- [t_MvTensorMyriadResources myriadResources](#): Myriad resources structure: first shave, last shave and DMA link agent.
- [t_MvTensorDebugInfo debugInfo](#): Debug table: debug message and time of mvTensor function call (in ms)

Detailed Description

MvTensor global parameters structure.

Field Documentation: generated from the following file "mvTensor.h"

- [t_mvTensorGenData t_MvTensorParam::accumulation](#) Intermediate data structure: needed only when operation needs to be done on another data type then the input/output are.
- [t_MvTensorDebugInfo t_MvTensorParam::debugInfo](#) Debug table: debug message and time of mvTensor function call (in ms)
- [t_mvTensorGenData t_MvTensorParam::input](#) Input data structure: pointer to input data, width, height, horizontal and vertical stride, data type and storage order.
- [t_MvTensorMyriadResources t_MvTensorParam::myriadResources](#) Myriad resources structure: first shave, last shave and DMA link agent.
- [t_MvTensorOp t_MvTensorParam::op](#) Filtering operation: Convolution, max-pooling.
- [int t_MvTensorParam::opStrideX](#) Operation stride in the X-direction.
- [int t_MvTensorParam::opStrideY](#) Operation stride in the Y-direction.
- [t_mvTensorGenData t_MvTensorParam::output](#) Output data structure: same type of information as for input.
- [t_MvTensorPostOp t_MvTensorParam::postOp](#) Post-operation: Relu.
- [int t_MvTensorParam::postOpStrideX](#) Stride in the X-direction for the post-processing operation.
- [int t_MvTensorParam::postOpStrideY](#) Stride in the Y-direction for the post-processing operation.
- [u32 t_MvTensorParam::preOperation](#) Pre-processing operation: TODO create enum type for this field.
- [t_mvTensorGenData t_MvTensorParam::weights](#) Weights data structure: same type of information as for input.

15.4.5. mvTensor.h File Reference

MvTensor API – interface to MvTensor compute library.

```
#include <mv_types.h>
```

Data Structures

- `struct t_mvTensorGenData`: Basic data structure.
- `struct t_MvTensorMyriadResources`: Myriad resources structure.
- `struct t_MvTensorDebugInfo`: Debug messages table.
- `struct t_MvTensorParam`: MvTensor global parameters structure.

Macros

```
#define MV_TENSOR_DBG_MSG_SIZE 120: Message size.
```

Enumerations

- `enum t_MvTensorDataType {t_fp16, t_u8f, t_int}`: MvTensor data type structure.
- `enum t_MvTensorPaddStyle {zero}`: Padding style.
- `enum t_MvTensorStorageOrder {orderYXZ, orderZYX, orderYZX}`: MvTensor data storage order options.
- `enum t_MvTensorOp {kConv1x1, kConv3x3, kConv5x5, kConv7x7, kMaxPool3x3}`: Filtering types.
- `enum t_MvTensorPostOp {kRelu, kReluMaxPool3x3, kNone}`: Post-processing operations.

Functions

- `void mvTensor (t_MvTensorParam *mvTensorParam)`: mvTensor main function.
- `static u32 checkForErrors (t_MvTensorParam *mvTensorParam)`: Debug function.

15.4.5.1. MvTensor API – interface to MvTensor compute library.

Macro Definition Documentation

```
#define MV_TENSOR_DBG_MSG_SIZE 120: Message size.
```

Enumeration Type Documentation

`enum t_MvTensorDataType`: MvTensor data type structure.

Enumerator:

t_fp16 half precision floating point
t_u8f Unsigned byte.
t_int Integer.

`enum t_MvTensorOp`: Filtering types.

Enumerator:

kConv1x1 1x1 Convolution
kConv3x3 3x3 Convolution
kConv5x5 5x5 Convolution
kConv7x7 7x7 Convolution
kMaxPool3x3 3x3 Max-Pooling

`enum t_MvTensorPaddStyle`: Padding style.

Enumerator:

zero Zero-padding.

enum **t_MvTensorPostOp**: Post-processing operations.

Enumerator:

kRelu rectified linear unit (Relu) rectifier
kReluMaxPool3x3 Relu rectifier and 3x3 max-pooling.
kNone No post operation.

enum **t_MvTensorStorageOrder**: MvTensor data storage order options.

Enumerator:

orderYXZ Option1: YXZ channel minor.
orderZYX Option2: ZYX column minor.
orderYZX Option3: YZX.

Function Documentation

static u32 checkForErrors (**t_MvTensorParam** * mvTensorParam) [static]

Debug function: Transformation matrix (3x2) is obtained from a rotation degree and translation coefficients

Parameters

in	<i>mvTensor-Param</i>	- pointer to MvTensor data structure
----	-----------------------	--------------------------------------

Returns

u32 - 0 if there were no errors found / 1 if there is at least an error

void mvTensor (**t_MvTensorParam** * mvTensorParam)

mvTensor main function: Parameters

in	<i>mvTensor-Param</i>	pointer to a structure that holds information about: <ul style="list-style-type: none"> • Input • Output • Weights • Accumulation • Pre-operation type • Operation • Operation stride in the X-direction • Operation stride in the Y-direction • Post-operation type • Post-operation stride in the X-direction • Post-operation stride in the Y-direction • Myriad used resources • Debug informations
----	-----------------------	--

Returns void

16. COMPUTER VISION: COLOUR HISTOGRAM MATCHING

■ Introduction

Colour histogram matching is a simple method for comparing the visual similarity of images or image regions. Despite its simplicity, it can be quite effective, especially if an appropriate colour space is chosen. The process consists of two parts. First, the colour histograms of the images or image regions, which shall be compared, are computed. Then the matching score is computed based on a histogram distance metric. The histogram matching module implements the computation of the colour histograms as well as different distance metrics to compute the distance score. Currently, the histogram intersection, the Hellinger distance and the earth mover's distance are implemented.

■ Unit tests

Unit tests for histogram extraction and matching have been implemented and are available at "WorkPackage_3/myriad/unittests/HistogramMatching".

■ Code

The code can be found in the GitLab repository of the EoT project at the following address: <https://gitlab.com/espiaran/EoT>.

The histogram matching module is available in "WorkPackage_3/myriad/libs/leon/HistogramMatching".

The documentation for the relevant code is in Annex 10.

■ Conclusions and Future work

The histogram matching module implements the functionality to extract colour histograms from images and to compare them with different distance metrics.

17. COMPUTER VISION: KEYPOINT MATCHING

Introduction

The ability to detect and match keypoints in two different images is a fundamental functionality in many computer vision algorithms. Algorithms involving keypoint matching generally distinguish three different steps: feature detection, feature description and feature matching. In **feature detection**, the input image is searched for particular points with the following properties: the same 3D point should be detected under various viewpoints, scale changes and lighting conditions (repeatability) and the detected point should have a unique visual signature (description). For this reason, keypoint detectors usually detect points in an image with sufficient texture and corresponding to positions of high frequency – usually referred to as *corners*. A **feature descriptor** is an algorithm that computes the signature of a specific point in the image based on its visual appearance. The desirable properties of a descriptor are uniqueness of the description and invariance of the description under various changes (scale, brightness, rotation etc.). The descriptor itself is usually represented as a fixed-size vector of real numbers, but recent descriptors rather produce binary codes to increase the efficiency of the matching process. **Feature matching** amounts to the computation of a distance between feature descriptors. The distance must be chosen such that the difference between descriptors of the same 3D point seen under various conditions is generally small, whereas the difference between descriptors of different 3D points should generally be higher. In many cases the L1 or L2 distance can be chosen, but more complex distance definitions are possible.

In the last decades, many different algorithms for keypoint detection, description and matching have been proposed with various properties. Recent developments tend to favour fast detectors based on simple pixel intensity checks, as well as binary descriptors, as they are more suited to a large number of keypoints while keeping the matching process tractable. Among the recently published algorithms, we decided to implement the BRISK keypoint algorithm¹⁰, as it proved to be the most promising in terms of matching capabilities¹¹. BRISK uses a scale-space version of AGAST¹² as keypoint detector, a binary code based on difference of pixel pairs as descriptor, and a Hamming distance for matching.

¹⁰ Leutenegger, S., Chli, M., & Siegwart, R. Y. (2011, November). BRISK: Binary robust invariant scalable keypoints. In Computer Vision (ICCV), 2011 IEEE International Conference on (pp. 2548-2555).

¹¹ Figat, J., Kornuta, T., & Kasprzak, W. (2014). Performance Evaluation of Binary Descriptors of Local Features. In Computer Vision and Graphics (pp. 187-194). Springer International Publishing.

¹² E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In Proceedings of the European Conference on Computer Vision (ECCV), 2010.

17.1.1. Description of the API

The functionality “Keypoint matching” consists of three different modules:

- **vector<keypoints> detectFeaturesBRISK (input_image):** this function takes an image as input, applies the BRISK detection method and delivers the list of detected keypoints as simple 2D points with (x, y) coordinates
- **vector<descriptor> extractDescriptorBRISK (vector<keypoints>, input_image):** this function iterates over the list of keypoints provided as input and computes a binary descriptor for each provided 2D point
- **double computeHammingDistance(descriptor1, descriptor2):** this function computes the distance between two binary descriptors. It can be used in an exhaustive matching algorithm

Known issues

There are no known issues, currently.

Unit tests

Unit tests will be created for each of the described functions with fixed input parameters.

Licensing

An implementation of the described algorithm is available in OpenCV 3.0, which is available under the three-clause BSD license.

Code

Some parts of the functionality exist in Movidius' MDK: function extractDescriptorBRISK is partially covered by a similar function in the MDK.

Conclusions and Future work

This functionality has not been fully implemented yet and future work will focus on the implementation of the missing parts.

18. COMPUTER VISION: ROTATION-INVARIANT FACE DETECTOR

Introduction

The main objective in this task is to create a rotation-invariant face detector. The functionality will be initially achieved through the libccv library, which already includes a face detector. Libccv's face detector is based on an extension/modification of the well-known cascade of HAAR-like features with Adaboost, but features significantly faster computation and similar accuracy¹³.

To this effect, the application will load an image (initially from the SD Card), it will perform two rotations (+25 degrees and -25 degrees) and it will try to detect the faces in the image. If a face is detected in any image (input image or rotated images) the application will confirm that a face has been detected.

18.1.1. Code structure

The detector is first developed in the LeonOS processor, and parts of it will be increasingly offloaded to the SHAVEs in order to optimize speed. The first computation that is offloaded to the SHAVEs is the rotation of the input image. Two SHAVEs are used to perform the two required rotations. The next sequence diagram shows the execution trace of its more important aspects.

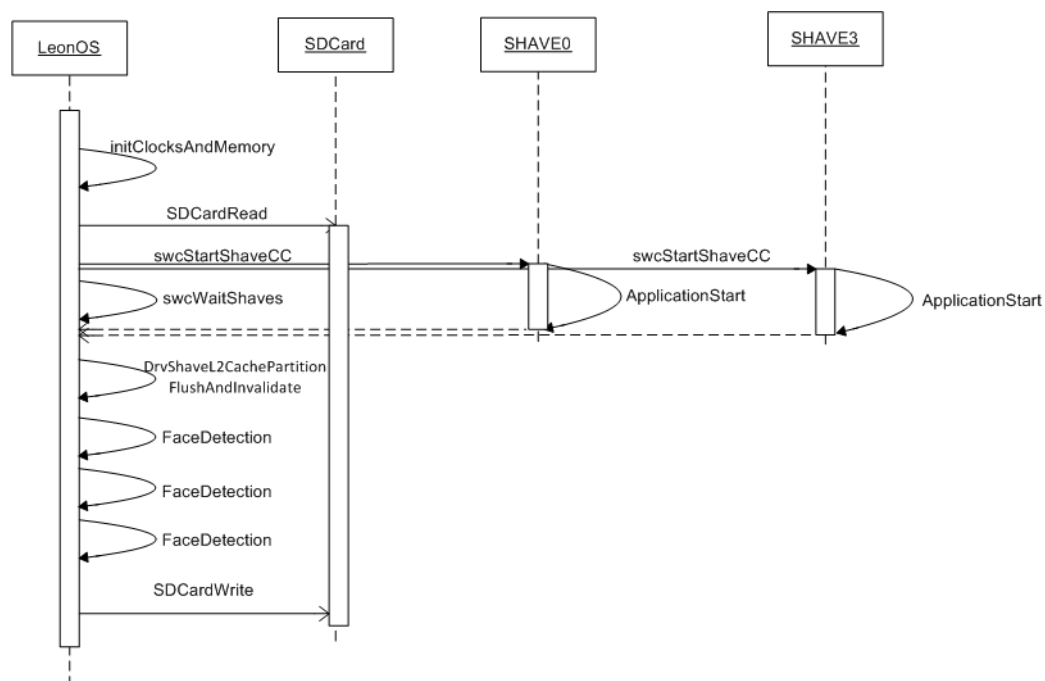


Figure 26. Face detection sequence diagram

¹³ Yotam Abramson, Bruno Steux, and Hicham Ghorayeb. 2007. Yet Even Faster (YEF) real-time object detection. Int. J. Intell. Syst. Technol. Appl. 2, 2/3 (February 2007), 102-112. DOI=<http://dx.doi.org/10.1504/IJISTA.2007.012476>

The following conclusions can be drawn from this sequence diagram:

- Read and write operations are performed by LeonOS processor.
- The `swcStartShaveCC` function is executed twice from LeonOS processor. This function starts the execution, in each SHAVE, `ApplicationStart` function.
- The execution of code in the two SHAVES is performed in parallel.
- Since the `swcStartShaveCC` function is synchronous, the `swcWaitShaves` function must be executed by LeonOS processor.
- When SHAVE processors finish, the `DrvShaveL2CachePartitionFlushAndInvalidate` function is executed.
- Finally, and sequentially, the three face detectors are executed by the LeonOS processor and the results are written in the SD Card.

In order to facilitate the scalability of the application, so that it is easier to add new SHAVES processors, the following structures and arrays have been created.

To contain the information of the image (rotation, path, faces detected and the image) we have the next structure:

```
struct data {
    char path_dst [250]; //Path where the rotate image will be saved.
    ccv_dense_matrix_t* img_dst; //Object ccv_dense_matrix. It contains all
    information relative
                                // to the final image.
    int degrees; //Degrees to rotate the original image
    int n_faces_detected; //Number of faces detected by bbf algorithm.
};
```

There are as many structures as images to rotate. These structures are passed in an array:

```
struct data array_attr[N_SHAVES] = {attrImgShave0, attrImgShave3};
```

Likewise, the entryptoints and SHAVES used are stored in arrays.

```
u32 entryPoints[N_SHAVES] = {
    (u32)&faceDetectionSHAVES0_ApplicationStart,
    (u32)&faceDetectionSHAVES3_ApplicationStart,
};

static swcShaveUnit_t SHAVE[N_SHAVES] = {0, 3};
```

This approach allows running, through loops, tasks associated to the SHAVES. In addition, it allows to add or to remove a SHAVE easily.

Two functions have been also developed to mark the faces detected. These functions receive x and y coordinates, and one measurement. With this data, the functions draw a square in the image. In addition, the color of the square can be selected. One of them draws a square on a gray scale image. The other draws a square on a color image.

```
void drawSquare (ccv_dense_matrix_t* img, int SquareX, int SquareY, int w,
u8 color);
void drawSquareColor (ccv_dense_matrix_t* img, int SquareX, int SquareY,
```

```
int w, u8 red, u8 green, u8 blue);
```

The `ccv_perspective_transform` function in `libccv` has been used to rotate. This method takes a transformation matrix and applies it to an image. In this case, `libccv` uses as coordinate origin the center of the image, so that the matrix used is:

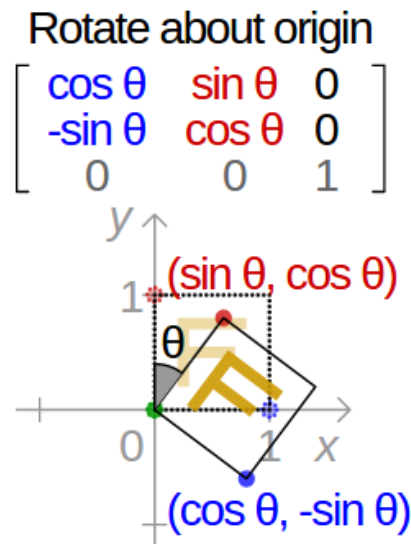


Figure 27. Rotation matrix

Since this function is executed by the SHAVEs processors, it has been necessary to include a part of `libccv` in the source path of the SHAVEs. Specifically, the dependencies of that function have to be included and compiled for the SHAVE processors. Figure 28 shows the dependencies.

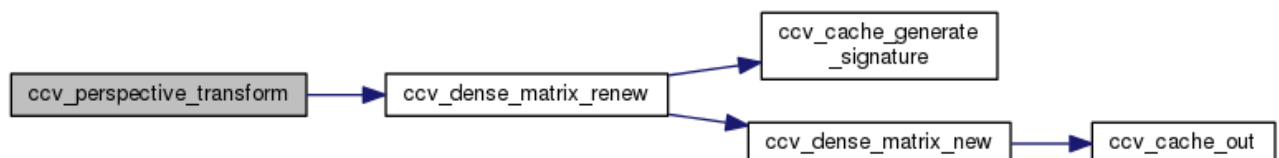


Figure 28. Face detection dependencies

The included files are the following:

- `ccv.h`
- `ccv_internal.h`
- `ccv_transform.c`: This file contains `ccv_perspective_transform` function.
- `ccv_memory.c`: This file contains `ccv_dense_matrix_renew` and `ccv_dense_matrix_new` functions.
- `ccv_cache.c`: This file contains `ccv_cache_generate_signature` y `ccv_cache_out` functions.
- `sha1.c`
- `sha1.h`

In file sha1.c we included a new function. Originally, the arpa/inet.h library is used by the libccv library. This library is used to change between little-endian and big-endian, according to the endianness where the program is executed. Both LeonOS and SHAVEs use little-endian architecture, so the CPU_swap_u32 function has been included in sha1.c. In this way, the dependence has been removed.

18.1.2. Optimizations

Three optimizations have been made in this application:

- The `ccv_perspective_transform` function is executed in the SHAVE processor. When executed in the LeonOS processor, a rotation operation takes between 0.5-0.6 seconds. Running the same operation in a SHAVE processor, its time was reduced to 0.27-0.3 seconds, that is almost half the time.
- Parallelizing rotations is the second optimization. This is achieved by the use of a number of SHAVEs running the same function. In this way, the two rotations take between 0.27-0.3 seconds.
- The last optimization is related to face detection. The parameters used by libccv in face detection have been modified. The times obtained can be seen in the following table. A window size of 90x90 and an interval of 3 have been selected. The window size is the minimum size of the object that can be detected while the interval is the number of images between the full size image and the half size one.

Attribute			
Window Size	Interval	Result	Time (Seconds)
24x24	5	1 detected	3,525186
	4	1 detected	2,929454
	3	1 detected	2,392324
	2	2 detected	1,865266
	1	1 detected	1,291151
30x30	5	1 detected	2,508927
	4	1 detected	2,148596
	3	1 detected	1,758166
	2	1 detected	1,401181
	1	1 detected	1,041092
70x70	5	1 detected	0,581443
	4	1 detected	0,528538
	3	1 detected	0,429481
	2	1 detected	0,351539
	1	1 detected	0,275125
90x90	5	1 detected	0,461311
	4	1 detected	0,380983
	3	1 detected	0,339476
	2	2 detected	0,275154
	1	1 detected	0,234491

18.1.3. How To

This section discusses the problems encountered in the implementation of this application and how they have been fixed. These problems are related to the use of SHAVE processors, so this section could be used as guide for SHAVE programming.

The example 004_SimpleCopyPlanes has been used as basis to develop this application. This is because example 004_SimpleCopyPlanes covers the minimum that rotation-invariant face detector application must have, this is, it must execute from LeonOS a function in a SHAVE processor.

Memory mapping

Initially, the example 004_SimpleCopyPlanes used the default memory mapping. Due to the fact that our application uses the SD Card, the default mapping produces the following error:

UART: Source 0 Internal 1 Error 2 0x2.

To fix it, a different memory mapping was included. This mapping can be found in the following path: mdk/common/scripts/ld/myriad2_shave_slices.ldscript

In this mapping, for each SHAVE, its memory is divided in two slices, text and data. When the `ccv_perspective_transform` function was included in the code executed by SHAVE0, the size of text slice was greater than the assigned slot. This caused an overlap between slices text and data of SHAVE0:

```
section S.shv0.cmx.data loaded at [0000000070008000,000000007004834f]
overlaps section S.shv0.cmx.text loaded at
[0000000070000000,0000000070010a03]
```

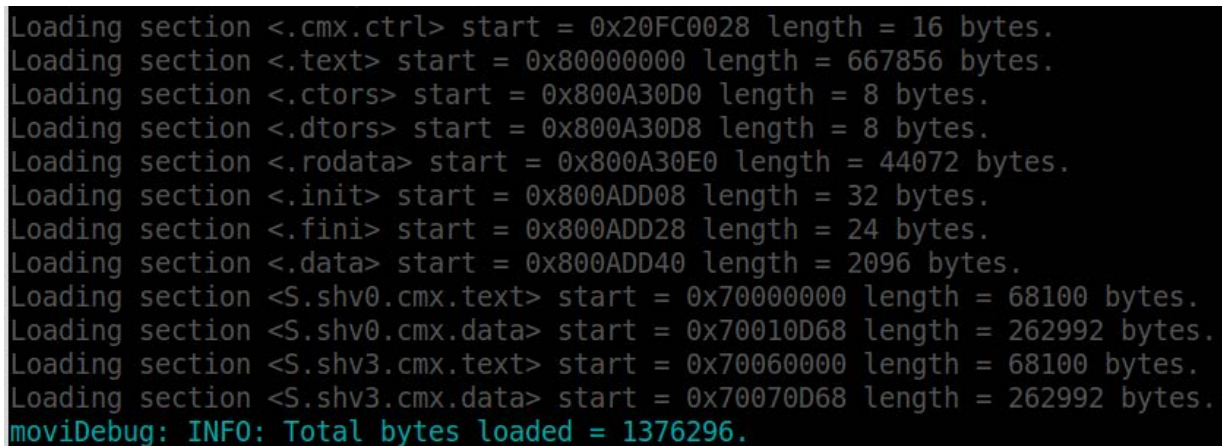
To fix this, it is necessary to change the beginning of the data slice for SHAVE0. Originally, the data slice of SHAVE0 had the next mapping.

```
. = 0x70008000;
S.shv0.cmx.data : {
    *(.shv0.S.data*)
    *(.shv0.S.rodata*)
    *(.shv0.S.__DATA__sect*)
    *(.shv0.S.__STACK__sect*)
    *(.shv0.S.__static_data*)
    *(.shv0.S.__HEAP__sect*)
    *(.shv0.S.__T__*)
    *(.lrt.shv0.S.data*)
    *(.lrt.shv0.S.rodata*)
    *(.lrt.shv0.S.__DATA__sect*)
    *(.lrt.shv0.S.__STACK__sect*)
    *(.lrt.shv0.S.__static_data*)
    *(.lrt.shv0.S.__HEAP__sect*)
    *(.lrt.shv0.S.__T__*)
}
```

It was changed to:


```
. = 0x70010d64;
S.shv0.cmx.data : {
    *(.shv0.S.data*)
    *(.shv0.S.rodata*)
    *(.shv0.S.__DATA__sect*)
    *(.shv0.S.__STACK__sect*)
    *(.shv0.S.__static_data*)
    *(.shv0.S.__HEAP__sect*)
    *(.shv0.S.__T__*)
    *(.lrt.shv0.S.data*)
    *(.lrt.shv0.S.rodata*)
    *(.lrt.shv0.S.__DATA__sect*)
    *(.lrt.shv0.S.__STACK__sect*)
    *(.lrt.shv0.S.__static_data*)
    *(.lrt.shv0.S.__HEAP__sect*)
    *(.lrt.shv0.S.__T__*)
}
```

Running the application, moviDebug informs of load sections, its memory positions and its size. In this case, the following image shows it.



```
Loading section <.cmx.ctrl> start = 0x20FC0028 length = 16 bytes.
Loading section <.text> start = 0x80000000 length = 667856 bytes.
Loading section <.ctors> start = 0x800A30D0 length = 8 bytes.
Loading section <.dtors> start = 0x800A30D8 length = 8 bytes.
Loading section <.rodata> start = 0x800A30E0 length = 44072 bytes.
Loading section <.init> start = 0x800ADD08 length = 32 bytes.
Loading section <.fini> start = 0x800ADD28 length = 24 bytes.
Loading section <.data> start = 0x800ADD40 length = 2096 bytes.
Loading section <S.shv0.cmx.text> start = 0x70000000 length = 68100 bytes.
Loading section <S.shv0.cmx.data> start = 0x70010D68 length = 262992 bytes.
Loading section <S.shv3.cmx.text> start = 0x70060000 length = 68100 bytes.
Loading section <S.shv3.cmx.data> start = 0x70070D68 length = 262992 bytes.
moviDebug: INFO: Total bytes loaded = 1376296.
```

Figure 29. Memory positions and size

As can be seen, data slice of SHAVE0 starts in the position 0x70010D68 and it allocates 262992 bytes. This means that SHAVE1 and SHAVE2 cannot be used because it will produce an overlap between data slice of SHAVE0 and slices of SHAVE1 and SHAVE2.

In this scenario, if either SHAVE1 or SHAVE2 is used, the following error will appear:

```
section  S.shv1.cmx.text    loaded  at  [0000000070020000,0000000070030a03]
overlaps          section  S.shv0.cmx.data    loaded          at
[0000000070010d68,00000000700510b7]
```

With the purpose of avoiding an overlap between SHAVE0 and SHAVE1 or SHAVE2, SHAVE3 has been used.

As with the SHAVE0, the slices assigned to SHAVE3 have been modified:

```

. = 0x70070d64;
S.shv3.cmx.data : {
    *(.shv3.S.data*)
    *(.shv3.S.rodata*)
    *(.shv3.S.__DATA__sect*)
    *(.shv3.S.__STACK__sect*)
    *(.shv3.S.__static_data*)
    *(.shv3.S.__HEAP__sect*)
    *(.shv3.S.__T__*)
    *(.lrt.shv3.S.data*)
    *(.lrt.shv3.S.rodata*)
    *(.lrt.shv3.S.__DATA__sect*)
    *(.lrt.shv3.S.__STACK__sect*)
    *(.lrt.shv3.S.__static_data*)
    *(.lrt.shv3.S.__HEAP__sect*)
    *(.lrt.shv3.S.__T__*)
}

```

APP_config.h

For each SHAVE used in the application, it is necessary to add a reference to it in the APP_UPA_CLOCKS variable. Otherwise, the execution will stop in the call to entryptoint of each SHAVE that is not added in APP_UPA_CLOCKS. This is done through the DEV_UPA_SHX structure. In this case, the variable is defined as follows:

```

#define APP_UPA_CLOCKS (DEV_UPA_SH0      | \
                        DEV_UPA_SH3      | \
                        DEV_UPA_SHAVE_L2 | \
                        DEV_UPA_CDMA     | \
                        DEV_UPA_CTRL     )

```

rtems_config.h

To use the SD Card, it is necessary to add in this file the following line:

```
#include <SDCardIORTEMSConfig.h>
```

The value of the variable CONFIGURE_MINIMUM_TASK_STACK_SIZE has been also changed to 16384.

Finally, BSP_SET_CLOCK function is as follows:

```
BSP_SET_CLOCK(12000, 200000, 1, 1, DEFAULT RTEMS_CSS_LOS_CLOCKS,
APP_MSS_CLOCKS, APP_UPA_CLOCKS, 0, 0);
```

For proper operation with the SD Card, the second parameter (Phase-locked loop) must be equal or greater than 100000. The default value in the example 004_SimpleCopyPlanes (260000) was tested, but some executions ended with exceptions. As a result of this, a known value was used. This value is 200000.

SHAVE cache

Immediately after the return of SHAVES (after `swcWaitShaves` function) the `DrvShaveL2CachePartitionFlushAndInvalidate` method must be called for each SHAVE used. Otherwise, data inconsistencies may occur. This function flushes the cache to the RAM memory. Without this, if two rotations with different degrees are performed, the resulting image would be as follows



Figure 30. Wrong rotated image

A part of image is rotated 70 degrees (the second execution). Nevertheless, there is a data inconsistency because another part of image is rotated 25 degrees (the first execution).

In addition, in file `app_config.c`, these two instructions must appear (these instructions are included in the example `004_SimpleCopyPlanes`)

```
#define L2CACHE_CFG      (SHAVE_L2CACHE_NORMAL_MODE)
DrvShaveL2CacheSetMode(L2CACHE_CFG);
```

These instructions configure the cache as a unique slice used by SHAVES.

Makefile

Changes made to the Makefile depend the name of application and the number of applications executed in each SHAVE.

In this case, the same entrypoint is executed in two different SHAVES, therefore the SHAVE applications section is as following:

```
SHAVE_APP_LIBS = $(faceDetectionSHAVESApp).mvlib
SHAVE0_APPS = $(faceDetectionSHAVESApp).shv0lib
SHAVE3_APPS = $(faceDetectionSHAVESApp).shv3lib
```

Variable `$(faceDetectionSHAVESApp)` indicates the application that is added in the `shvXlib` library.

Optionally, cleaning rules can be added to delete the files .mvlib and shvXlib generated.

Finally, the entrypoint of the SHAVE is called `ApplicationStart`. This is defined in `ENTRYPOINTS = -e ApplicationStart --gc-sections`. The main function in the source code of SHAVE must have the same name.

Known issues

At the time of writing, only the `ccv_perspective_transform` function is available for its use in SHAVE processors.

If the face is greater than the window size (90x90), the face will not be detected.

Unit tests

The contents of the `testFiles` folder must be copied to the SDCard (`/mnt/sdcard/`) resulting in (`/mnt/sdcard/Rotation-invariant_faceDetector`)

- Files required:
 - `Rotation-invariant_faceDetector/lena.png`
 - `Rotation-invariant_faceDetector/face`

18.3.1. Output expected

There are two outputs, text and images

The text output is:

```
UART: ccv_perspective_transform Usec CPU time: 0.271619 seconds
UART:
UART: Original Image
UART: total : 0 detected
UART: [!] bbf Original Image Usec CPU time: 0.226114 seconds
UART:
UART: Rotated Image: 25°
UART: 230 214 157 157 -2.927335
UART: total : 1 detected
UART: [!] bbf Rotated Image: 25° Usec CPU time: 0.262913 seconds
UART:
UART: Rotated Image: -25°
UART: total : 0 detected
UART: [!] bbf Rotated Image: -25° Usec CPU time: 0.193256 seconds
UART:
UART: Face detected
```

The images generated by this app (`/mnt/sdcard/Rotation-invariant_faceDetector/Results`) are the following:



Figure 31. Rotated images

■ Licensing

Same as in Task "Other vision libraries".

■ Code

The rotation-invariant face detection application can be found in folder WorkPackage_3\myriad\apps\Rotation-invariant_faceDetector

Dependences

- libccv
- SDCardIO

The documentation for the relevant code is in Annex 11.

■ Conclusions and Future work

A rotation-invariant face detector has been developed with this application. This application can rotate, concurrently, an input image and executing, on each rotated image, the face detector. If there is any positive result, the face has been detected.

In addition, the `ccv_perspective_transform` function has been executed by the use of SHAVE processors.

Future work can consider:

- Face detection currently loads the input image from the SD Card. Camera will be integrated.
- Even though computational cost has been reduced, further optimizations can be made:
 - Face detection can be itself offloaded to SHAVE processors
 - The face detection algorithm can be partitioned and parallelized onto multiple SHAVEs
- The IMU sensor can be added to the module. This sensor can provide camera orientation in space, and thus may be used to perform a single rotation of the input image.
- Other face detectors may be considered (HOG+SVM, CNN-based, ...)

19. COMPUTER VISION: SPARSE OPTICAL FLOW (LK POINT TRACKING)

Introduction

Optical flow is the pattern of apparent motion of image objects between two consecutive frames caused by the movement of object or camera. It is 2D vector field where each vector is a displacement vector showing the movement of points from the first frame to the second.

The objective in this task is to obtain the functionality of Lucas-Kanade point tracking. This task's code depends on: Task "Camera interface".

In order to implement this module, at the time of writing two approaches have been followed, using the OpenCV 1.0 library port and using the vTrack module implemented internally by Movidius. The OpenCV solution only makes use of the LeonOS processor while vTrack uses also the SHAVEs, having a better performance.

19.1.1. OpenCV

OpenCV provides a sparse iterative version of the Lucas-Kanade optical flow in pyramids in a single function, **calcOpticalFlowPyrLK()**. To select the points to track, **goodFeaturesToTrack()** function is used. First, the algorithm takes the first frame, detects Harris corner points in it, and then iteratively tracks those points using Lucas-Kanade optical flow.

calcOpticalFlowPyrLK() receives the previous frame, previous points and the next frame, returning the next points along with status numbers which have a value of 1 if next point was found, and zero otherwise. In the next step, these points are passed as previous points.

19.1.2. vTrack

vTrack is a feature tracking algorithm implemented internally by Movidius (not in the MDK at the time of writing). The goal is to detect keypoints (features) on a frame and try to follow them on successive frames.

The output of the algorithm is a header with information about the frame (i.e. timestamp, frameId, number of features), a list of features and debug information (i.e. runtime of components, histogram based on age). Each feature in the list has an (x,y) location, an id, and an age.

To minimize the runtime and improve the tracking quality, the algorithm has the following capabilities:

- The number of SHAVEs is configurable. The user can minimize the latency of the data by using more SHAVEs

- Using gyro data, the algorithm is capable to estimate the new position of the features. With gyro assist the tracking quality and speed is improved.
- The algorithm is running on SHAVEs only, so the Leon just starts the process and can handle other tasks while vTrack is running.
- The algorithm can process multiple images in parallel

The algorithm has four major components:

1. vPipe: This is the interface of vTrack with the application. It is responsible for scheduling the other components of the algorithm based on the user setting
2. pixelPipe: Detects keypoints (features) on the input frame and generates the Gaussian pyramid for optical flow
3. opticalFlow: Finds the new position of the previous features on the current Gaussian pyramid
4. featureMaintenance: Maintains the list of tracked features. The component will assign id and age for the features, dropping the features which were not tracked correctly by the opticalFlow. If needed, it will also add the strongest features from pixelPipe.

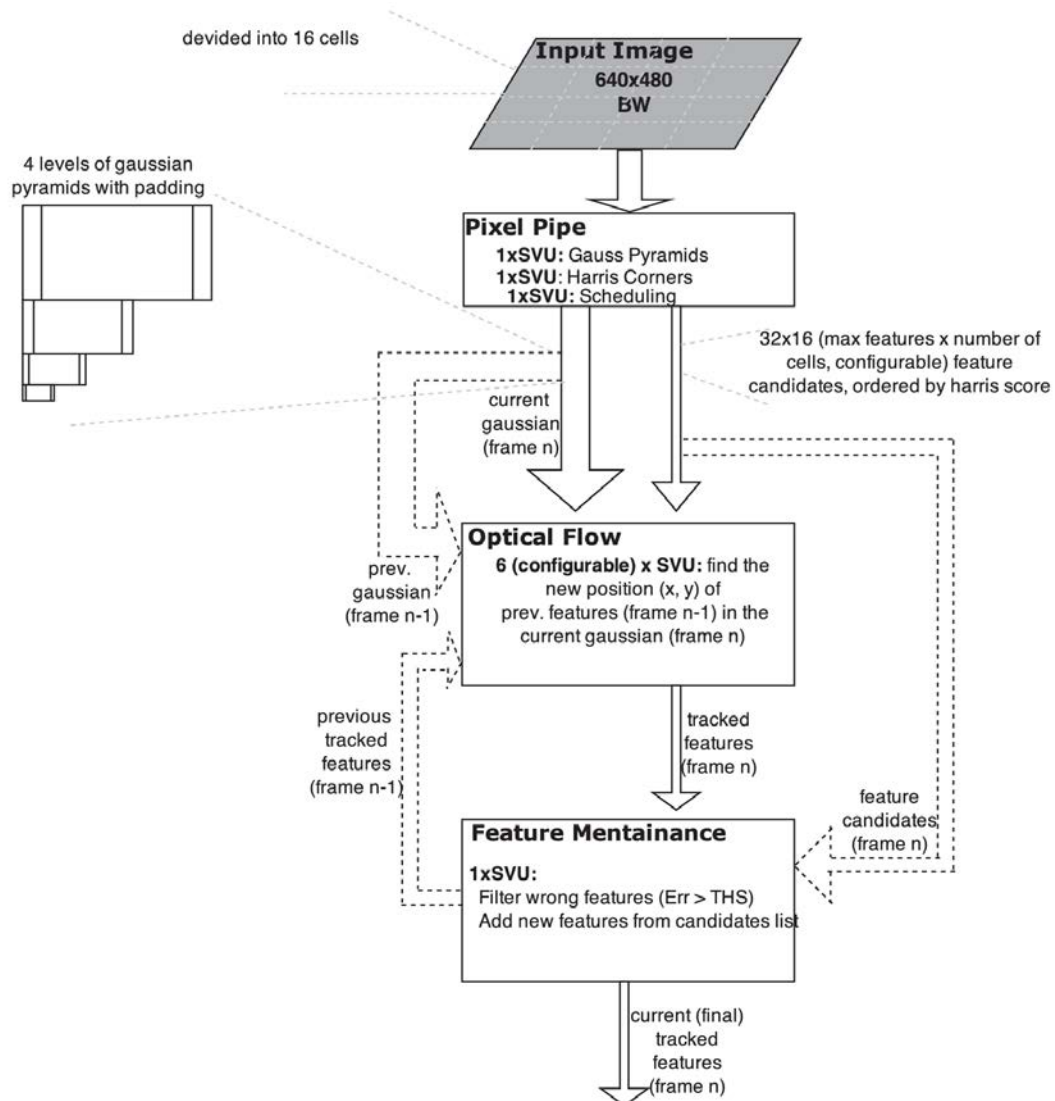


Figure 32. Vtrack flow diagram

It is possible to configure which of these components/steps is executed. Module vPipe contains a structure called vPipeTransitionTable in which all the possible configurations are stored. For example, the PP_FM_OF mode is defined as:

```
// VPIPE_MODE_PP_FM_OF
{
VPIPE_STATE_ERROR, // Next state for VPIPE_STATE_NOT_INIT
VPIPE_STATE_RUN_PIXEL_PIPE, // Next state for VPIPE_STATE_INIT
VPIPE_STATE_RUN_GYRO_PREDICT, // Next state for
VPIPE_STATE_RUN_PIXEL_PIPE
VPIPE_STATE_RUN_OF, // Next state for VPIPE_STATE_RUN_GYRO_PREDICT
VPIPE_STATE_RUN_FM, // Next state for VPIPE_STATE_RUN_OF
VPIPE_STATE_FILL_OUTPUT, // Next state for VPIPE_STATE_RUN_FM
VPIPE_STATE_DONE, // Next state for VPIPE_STATE_FILL_OUTPUT
VPIPE_STATE_ERROR, // Next state for VPIPE_STATE_FILL_DUMMY_OUTPUT
VPIPE_STATE_ERROR, // Next state for VPIPE_STATE_ERROR
VPIPE_STATE_INIT, // Next state for VPIPE_STATE_DONE
}
```

```
},
```

In which each parameter can contain `VPIPE_STATE_ERROR` value to deactivate it, or the `VPIPE_STATE_RUN_XXX` value to execute this step. For example if the user needs to deactivate the gyroscope in the previous example, the structure must be defined as follows:

```
// VPIPE_MODE_PP_FM_OF
{
VPIPE_STATE_ERROR, // Next state for VPIPE_STATE_NOT_INIT
VPIPE_STATE_RUN_PIXEL_PIPE, // Next state for VPIPE_STATE_INIT
VPIPE_STATE_ERROR, // Next state for VPIPE_STATE_RUN_PIXEL_PIPE
VPIPE_STATE_RUN_OF, // Next state for VPIPE_STATE_RUN_GYRO_PREDICT
VPIPE_STATE_RUN_FM, // Next state for VPIPE_STATE_RUN_OF
VPIPE_STATE_FILL_OUTPUT, // Next state for VPIPE_STATE_RUN_FM
VPIPE_STATE_DONE, // Next state for VPIPE_STATE_FILL_OUTPUT
VPIPE_STATE_ERROR, // Next state for VPIPE_STATE_FILL_DUMMY_OUTPUT
VPIPE_STATE_ERROR, // Next state for VPIPE_STATE_ERROR
VPIPE_STATE_INIT, // Next state for VPIPE_STATE_DONE
},
```

The selection of the mode to be run should be indicated in the function:

```
vPipeInit(volatile t_vPipeMode vp_mode, vpipeCallback_t* vpipeCallback, volatile
t_vPipeRes vp_resolution, frameSpec* input_frame_spec, float fov, float
camCenterX, float camCenterY).
```

An example running the `VPIPE_MODE_PP_FM_OF` mode:

```
vPipeInit(VPIPE_MODE_PP_FM_OF, vPipeDoneCb, Res_480, &camFrame[0].spec,
h_fov_degrees, cam_center_x, cam_center_y);
```

Known issues

19.2.1. OpenCV

The library is not currently fully optimized for Myriad 2, as it uses only the LeonOS processor.

19.2.2. vTrack

vTrack is currently in internal development by Movidius. At the time of writing, it has been only possible to test an unfinished binary of the example application.

The final library will be included in the next MDK version.

Unit tests

19.3.1. OpenCV

There are no unit tests.

19.3.2. vTrack

vTrack has 7 unit tests:

Test 1

This test will test the Gaussian image and corners generated by the pixelPipe component.

Expected output

=====

The results consist in printf seen using the debugger:

UART:

UART: Enable SHAVE L2 Cache

UART:

DEBUG: unitTestCrcCheck() : (addr:0x8001A400,0x0004CE00,0xAAA9135B)

=> PASS

DEBUG: unitTestCrcCheck() : (addr:0x80006900,0x00013B00,0x5E894CA0)

=> PASS

DEBUG: unitTestCrcCheck() : (addr:0x80001680,0x00005280,0x51422FAB)

=> PASS

DEBUG: unitTestCrcCheck() : (addr:0x80000000,0x00001680,0xAE3654B6)

=> PASS

DEBUG: unitTestCrcCheck() : (addr:0x70190AF0,0x00001900,0x7F949CDC)

=> PASS

UART: Feature count: 356

DEBUG:

DEBUG: moviUnitTest:PASSED

Afterwards, the debugger saves 5 pictures to the current folder.

The four levels of the gaussian pyramid:

pyr0_656x480_400.bw

pyr1_336x240_400.bw

pyr2_176x120_400.bw

pyr3_96x60_400.bw

Test 2

This test will test if the pixelPipe is finding corners correctly. It also tests if the featureMaintenance is able to correctly pass these features to the output buffers.

Expected output

=====

In the debugger window we will see the number of keypoints found by pixel pipe. The number of tracked features (FM output) is also shown. After each vTrack step, we should see the line:

DEBUG: unitTestAssert() : (value:0x00000001) => PASS

At the end of the test, the following line will appear:
DEBUG: moviUnitTest:PASSED

Afterwards, the debugger saves the input picture with all the detected corners:
features_640x480.bw

Test 3

This test will run vTrack in dummy mode. This means that vTrack will generate dummy features only. This can be used in a project to validate the datapath.

Expected output

=====

In the debugger window we will see the number of keypoints generated. After each vTrack step, we should see the line:

DEBUG: unitTestAssert() : (value:0x00000001) => PASS

At the end of the test, the following line will appear:
DEBUG: moviUnitTest:PASSED

Afterwards, the debugger saves the generated corners with black background:
features_640x480.bw

Test 4

The test will run vTrack several times using a test image and will report the following things:

- Number of points tracked
- Average runtime of vTrack
- Average runtime of each module
- Average tracking error of a feature

Expected output

=====

In the debugger window we will see the following information:

The coordinate of the points which have a tracking error (after 500 cycles) bigger than 0.1 (squared distance).

If the test passes we should see the line:

DEBUG: unitTestAssert() : (value:0x00000001) => PASS

After this the average runtime, the average error and the number of points are printed.

At the end of the test, the following line will appear:
DEBUG: moviUnitTest:PASSED

Afterwards, the debugger saves the input image with the position of each feature (after 500 cycles, so they will not be perfect):
features_640x480.bw

Test 5

The test will run vTrack several times obtaining pictures from the camera, reporting the following data:

- Number of points tracked
- Average runtime of vTrack
- Average runtime of each module
- Average tracking error of a feature

Expected output

=====

In the debugger window we will see the following information:

The coordinate of the points which have a tracking error (after 53 cycles) bigger than 0.1 (squared distance).

If the test passes, we should see the line:

```
DEBUG: unitTestAssert()      : (value:0x00000001)          => PASS
```

After this the average runtime, the average error and the number of points are printed.

At the end of the test, the following line will appear:

```
DEBUG: moviUnitTest:PASSED
```

Afterwards, the debugger saves the input image with the position of each feature (after 53 cycles, so they will not be perfect):

features_640x480.bw

Test 6

The test will generate input images for vTrack. On each image we will have only one corner at a given position. We will test if the corner coordinates are correct, and if it can be tracked by optical flow correctly.

We should test all points, but because this would take a very long time, we select several points from each portion of the image. The points will cover each cell, corners close to each border.

Expected output

=====

In the debugger window we will see the following information:

The coordinate of the points which have a tracking error (after 500 cycles) bigger than 0.1 (squared distance).

If the test passes, we should see the line several times:

```
"UART: Testing (x.000, X)" - where x is the currently tested line
```

If every point was tracked correctly, we should see the following line:

```
DEBUG: moviUnitTest:PASSED
```

Test 7

This test takes two real images and track features multiple times from one to another.

■ Licensing

19.4.1. OpenCV

Same as in Task "Other vision libraries".

19.4.2. vTrack

@copyright All code copyright Movidius Ltd 2012, all rights reserved.

■ Code

19.5.1. OpenCV

An example implementing the Lukas-Kanade algorithm can be found in WorkPackage_3\myriad\apps\OpticalFlow\OpticalFlowLK_OpenCV of EoT repository.

This example takes around 2 seconds to calculate the initial points, and 0.35 seconds for each new picture in the included picture example. Changing the parameters and depending on the configuration necessary for its application, these values can be reduced up to 1 second for the initial point calculation and 0.09 seconds per each new image.

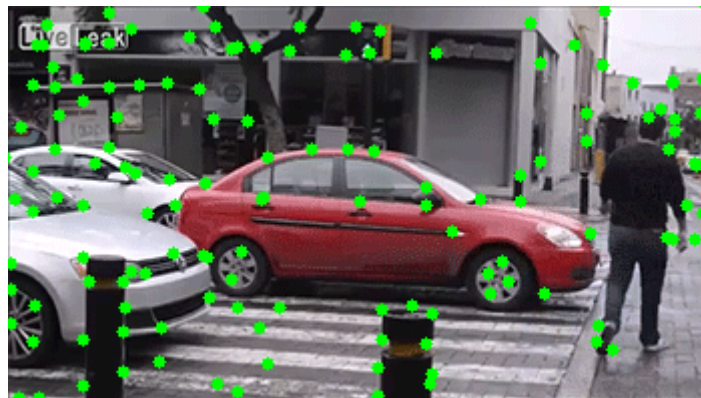


Figure 33. OpenCV optical flow example

Dependences

Same as in Task "Other vision libraries".

19.5.2. vTrack

The vTrack current version can be found in WorkPackage_3\myriad\apps\OpticalFlow\vTrack of EoT repository.



Figure 34. Vtrack optical flow example

Folder structure:

- modules: contains the vTrack modularized source code.
- myriad: contains the tests for vTrack.
- pc: contains the pc model of vTrack.

Required:

- You need to have an MDK in the same folder as vTrack to compile the application for the EoT device.
- For the pc model, you need OpenCv and MDK.

Conclusions and Future work

In this Section, two ways of using tracking algorithms using the EoT device have been explained. The first one uses the OpenCV library ported to EoT board, and the second uses the vTrack library which is being internally developed by Movidius engineers.

20. POWER MANAGEMENT

Introduction

Different components of the EoT device can be activated/deactivated in order to reduce power consumption. Some important elements, such as WiFi or camera, can be set in a low power consumption state. The power of the Myriad processor is controlled by the so-called *power islands*. In total, there are more than 20 power islands. Each power island controls the power supply of a certain component in the chip. The following are the main power island domains:

- **AON** - Always-on domain
- **CSS** - CPU Sub-system
- **SHAVE[11:0]** - SHAVE processor core island, one per processor
- **PMB** - Processor Memory block (CMX DMA, Mutex, Bicubic)
- **MSS** - Media Sub-system (Cameras, MIPI, SIPP filters)
- **DSS** - DDR Sub-system

The following diagram shows these power islands (best viewed in color).

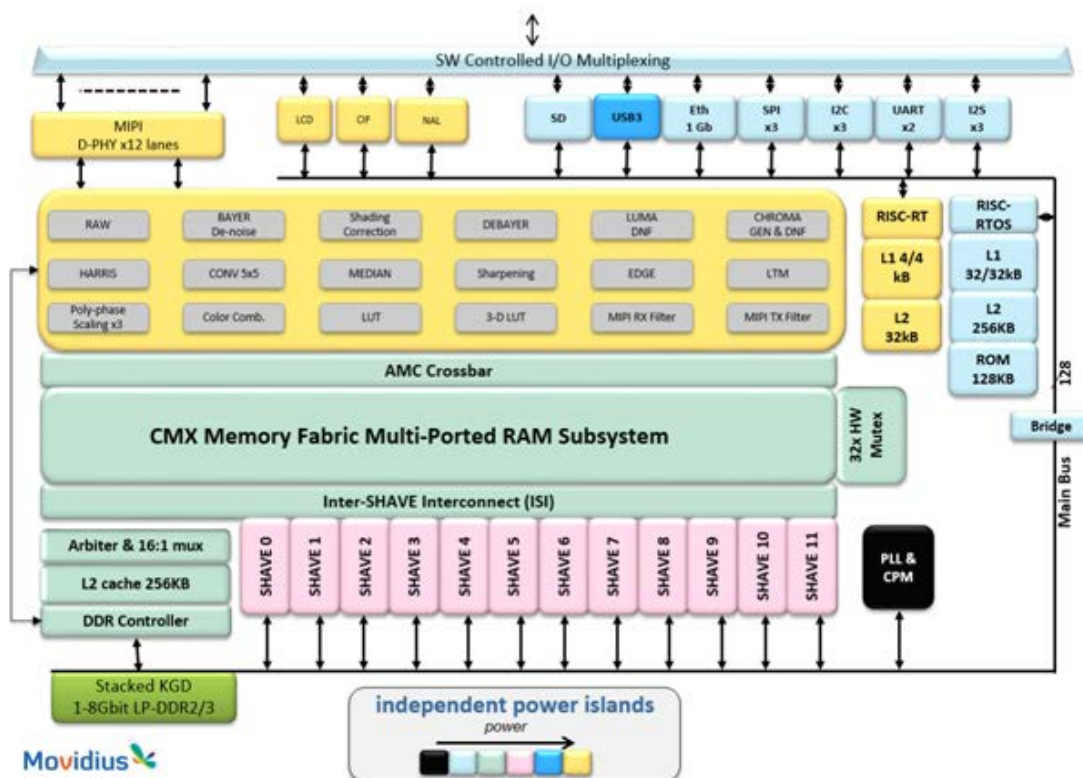


Figure 35. Myriad 2 power islands diagram

The primary power island (island 0 or CSS island) is special. It can be activated by a change of state on the wake-up signal or via a reset. The power islands can be controlled independently except:

- AON must be powered to start CSS domain

- CSS must be powered to start PMB, DSS or MSS domains
- PMB must be powered to start any of the SHAVE domains

For Example, to run code on SHAVE 8 processor from PMB you need: AON, CSS, PMB, and SHV[8] on.

The SHAVE drivers will automatically turn off SHAVE islands when not in use, but other islands need to be controlled at application level.

A low powered scenario may consider an application running from DRAM and having two different regions of interest. In that case, the core section of the application would live in the first part of the DRAM and other functions and data would live outside this region. On entering the low power state the first part of DRAM would be locked into the LeonOS L2 cache. This would allow for the DRAM to be put into self-refresh and all islands except the CSS would be disabled. Therefore, the clock may be reduced to lower frequencies and the device may wait until a specific wake-up event. To wake up the Myriad the needed islands are re-enabled.

There are functions to enable/disable power islands although their use is very complex at that level. Certain sequential steps and memory conditions should be taken into account to put the Myriad chip into a low power state. Movidius is currently working to make this functionality more usable for programmers¹⁴.

Low power states

20.2.1. Camera standby modes

For power saving or reconfigurability reasons, the camera can be put in standby. There are two types of standby modes:

- **HOT STANDBY:** the sensor is deactivated (but still configured), the MIPI controller and PHY and the MSS connections are still active and the CIF/SIPP are reconfigured but not restarted. This standby type is mostly used to save processor time by suspending the interrupts and is fast to recover from (activation in less than 0.5 milliseconds)
- **COLD STANDBY:** same as hot standby but the sensor is deactivated. Wakeup out of this state implies full sensor reconfiguration. The wake up duration may last tens of milliseconds, depending on the number of the sensor registers to configure. This standby type saves more power.

20.2.2. WiFi power policies

The WiFi subsystem supports predefined power management policies which allow a host application to guide the behaviour of the power-management algorithm¹⁵. The available policies are:

¹⁴ Myriad 2 Datasheet v1.06

¹⁵ CC3100\CC3200 SimpleLink™ WiFi® Network Processor Subsystem Programmer's Guide

- **Normal** (Default) – Features the best trade-off between traffic delivery time and power performance.
- **Always on** – The WiFi subsystem is kept fully active at all times, providing the best WLAN traffic performance. This policy is user-directed, whereby the user may provide the target latency figure.
- **Long Sleep Interval** – This low power mode comes with a desired max sleep time parameter. The parameter reflects the desired sleep interval between two consecutive wakeups for beacon reception. The WiFi module computes the desired time and wakes up to the next DTIM that does not exceed the specified time. The maximum allowed desired max sleep time parameter is two seconds.
- **Low latency power** – This device power management algorithm exploits opportunities to lower its power mode. Trade-off tends toward power conservation performance.

20.2.3. Myriad power states

The Myriad chip supports six different power states:

- **OFF** - All power off to all Myriad supplies, needs full system reset and boot time on power-up.
- **Deep Sleep** - All power islands off, only AON domain powered. Only IO and core supply are necessary. This state monitors the wakeup signal for wakeup condition. If the wakeup condition is met, it follows the wakeup sequence to start the Phase Locked Loop-based (PLL) clock generator and reload the boot image.
- **Sleep Mode A** - All power islands off, only AON domain and DRAM are powered. Only IO, core, and DRAM associated supplies are necessary. This state monitors the wakeup signal for wakeup condition. If the wakeup condition is met, it follows the wakeup sequence to start PLL and reload the boot image. In this mode boot image can be stored in DRAM (in self-refresh mode).
- **Sleep Mode B** - Similar to Sleep Mode Type A, applicable when size of boot image can fit in 256KB (L2 cache size).
- **LOW Power** - Only AON and CSS power domains are on. Wakeup can be from any GPIO pin or the wakeup pin. PLL can be on or off, depending on boot time versus low power trade-off.
- **Active** - Application dependent.

At the time of writing, these states are not yet exposed to the programmer in the MDK software, but future releases will.

Unit tests

There are not automatic tests for testing the functionalities of this module.

Code

WiFi and Camera power management is achieved through the corresponding EoT module functions. Device power islands are managed by the user using functions from the Myriad basic drivers.

20.4.1. WifiFunctions power management functions

- **_i32 setWlanPower** (_u8 power)
This function changes the operational power of the device.
- **_i32 setPowerPolicy** (_u8 policy)
This function sets the device power policy.
- **_i32 sleepWlanDevice** (int time)
This function is used to make the device enter sleep mode.

20.4.2. Camera power management functions

- **int standby_camera ()**
Puts the camera into 'hot standby' mode.
Returns:
-1 if there was an error. 0 otherwise.
- **int wakeup_camera ()**
Wakes up the camera from a standby mode.
Returns:
-1 if there was an error. 0 otherwise.

20.4.3. Myriad power management functions

OsDrvCpr.h File Reference

RTEMS CPR Header File.

```
#include <OsDrvCprDefines.h>
```

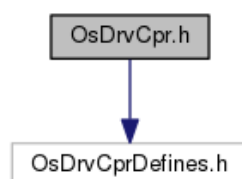


Figure 36. Include dependency graph for OsDrvCpr.h

- **int OsDrvCprPowerTurnOffIsland** (enum PowerIslandIndex *island_index*)
Turn off a single given power island, taking care of any delays that are needed. NOTE: this function does not reset anything in the power island before turning it off. It is recommended to reset peripherals, and to remove the clock to peripherals before turning the island they are in off.
Returns:
OS_MYR_DRV_SUCCESS on success, non-zero otherwise
- **int OsDrvCprPowerTurnOffIslandRaw** (u32 *islands_mask*, u32 *iso_ticks*, u32 *disable_ticks*)
Turn off one or more power islands, giving explicit isolation enable, and island disable delay values in clock ticks.

Parameters:

in	<i>islands_mask</i>	the index of each bit that is set to 1 tells which islands should be turned off
in	<i>iso_ticks</i>	the number of clock cycles to wait after enabling isolation, but before turning off power
in	<i>disable_ticks</i>	the number of clock cycles to wait after disabling power, before the function returns. If the number of ticks is 0 then it will wait until the power island status becomes inactive.

Returns:

OS_MYR_DRV_SUCCESS on success, non-zero otherwise

- **int OsDrvCprPowerTurnOnIsland (enum PowerIslandIndex *island_index*)**

Turn on a single given power island, taking care of any delays that are needed. NOTE: this function does not reset anything in the power island after turning it on, the user must do that manually.

Returns:

OS_MYR_DRV_SUCCESS on success, non-zero otherwise

- **int OsDrvCprPowerTurnOnIslandRaw (u32 *islands_mask*, u32 *trickle_ticks*, u32 *enable_ticks*)**

Turn on one or more power islands, giving explicit trickle, and enable delay values in clock ticks.

Parameters:

in	<i>islands_mask</i>	the index of each bit that is set to 1 tells which islands should be turned on
in	<i>trickle_ticks</i>	number of clock cycles to wait after enabling trickle power, but before turning on full power
in	<i>enable_ticks</i>	number of clock cycles to wait after enabling full power, but before turning of isolation. If the number of ticks is 0, then it will wait until power island status is active.

Returns:

OS_MYR_DRV_SUCCESS on success, non-zero otherwise

DrvCprDefinesMa2100.h File Reference

- **Enumerations**

enum PowerIslandIndex

Enumerator

POWER_ISLAND_CSS_DIGITAL
POWER_ISLAND_CSS_ANALOG
POWER_ISLAND_RETENTION
POWER_ISLAND_SHAVE_0
POWER_ISLAND_SHAVE_1
POWER_ISLAND_SHAVE_2
POWER_ISLAND_SHAVE_3
POWER_ISLAND_SHAVE_4
POWER_ISLAND_SHAVE_5
POWER_ISLAND_SHAVE_6
POWER_ISLAND_SHAVE_7
POWER_ISLAND_SHAVE_8
POWER_ISLAND_SHAVE_9
POWER_ISLAND_SHAVE_10
POWER_ISLAND_SHAVE_11

POWER_ISLAND_PMB
POWER_ISLAND_MSS_DIGITAL
POWER_ISLAND_MSS_ANALOG
POWER_ISLAND_DSS_DIGITAL
POWER_ISLAND_DSS_ANALOG

Conclusions and Future work

WiFi and Camera low power options have already been implemented in Camera and WifiFunctions modules. Regarding the Myriad low power modes, there are a set of very low level functions that support this task at a power-island level. Movidius' engineers are currently working to provide a higher level API for power management that abstracts hardware complexities to the programmer.

21. CONTROL MODE API, DESKTOP SIDE

This module was described in deliverable D3.1.

22. CONTROL MODE API, ANDROID

This module was described in deliverable D3.2.

noelya86@gmail.com

23. OTHER VISION LIBRARIES

Introduction

In this task, several pre-existing vision libraries have been ported to the EoT platform. This task, which was not in the original DoW, has been added with the idea of offering common and well-known libraries to the programmer-user of the EoT board.

23.1.1. OpenCV 1.0

OpenCV is a computer vision library released under a BSD license and hence it is free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications.

The goals of the OpenCV project were described as:

1. Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.
2. Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.
3. Advance vision-based commercial applications by making portable, performance-optimized code available for free-with a license that did not require to be open or free themselves.

Detailed information of all the functionalities of the library can be found in <http://www.cs.indiana.edu/cgi-pub/oleykin/website/OpenCVHelp/>.

23.1.2. OpenCV 2.4 in the cloud

Although OpenCV 1.0 has been ported to the EoT board, it is also important to have the possibility of running newer releases. To achieve this goal, an external server running an API to use OpenCV 2.4.3 is used. With this approach, it is actually possible to run any OpenCV version.

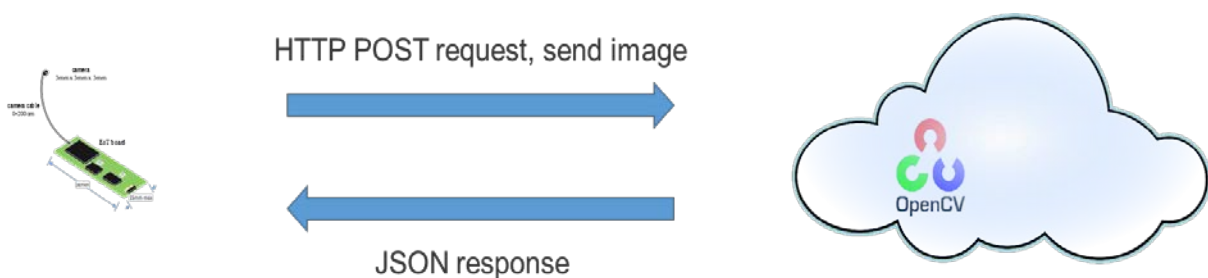


Figure 37. OpenCV in the cloud paradigm

Concretely, a free Pythonanywhere server has been used (www.pythonanywhere.com). This is a free service which provides a server with web2py and OpenCV 2.4.3 in which it is possible to program any computer vision capabilities.

23.1.2.1. Pythonanywhere configuration

In order to configure the Pythonanywhere server, the following steps should be followed:

1. Register for a free account in PythonAnywhere.com
2. Once you have registered and logged in, go to the **Web** tab, and **Add a new web app**

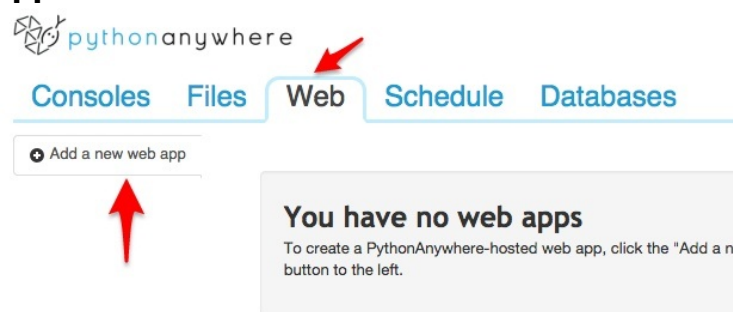


Figure 38. Add new app

3. Select web2py as your python framework.

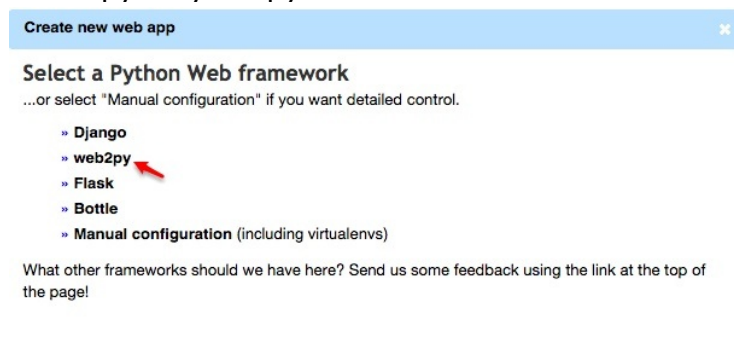


Figure 39. Selecting web2py

4. Select an admin password for web2py. Note that this is a web2py admin password, and it is different from your PythonAnywhere.com password.
5. Open a new tab, and go to web2py admin interface located at: <https://username.pythonanywhere.com/admin/default/index>

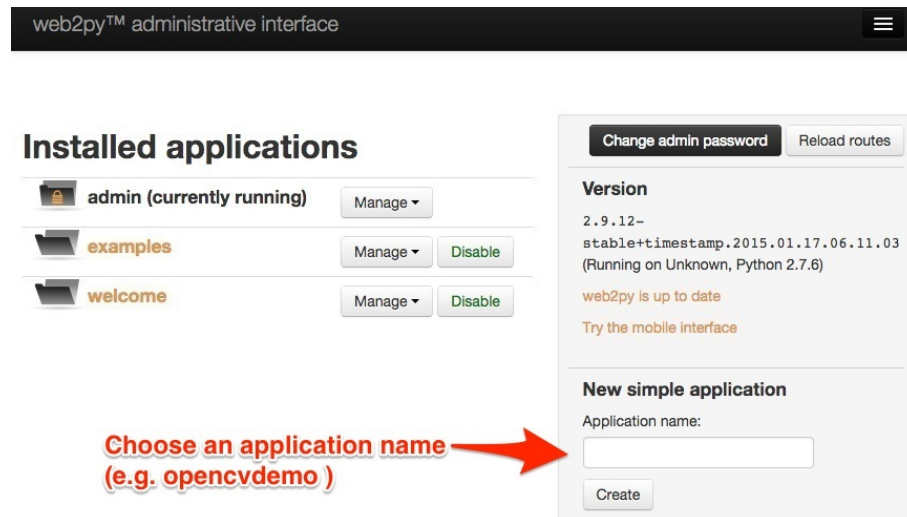


Figure 40. Choosing the application name

6. You should see your application folder under the **Files** tab on PythonAnywhere when you go down the directory structure `home/username/web2py/appname`
7. **To add OpenCV code to the web2py open** the following code: `home/username/web2py/applications/appname/controllers/default.py` and add new functions there. For testing, the following functions which use OpenCV to obtain the dimensions of the image have been developed. The program can receive a URL with the image and an image embedded in the HTTP request:

```
def image_dimensions():
    # Masquerade as Mozilla because some web servers may not
    # like python bots.
    hdr = {'User-Agent': 'Mozilla/5.0'}
    # Set up the request
    req = urllib2.Request(request.vars.url, headers=hdr)

    try:
        # Obtain the content of the url
        con = urllib2.urlopen( req )
        # Read the content and convert it into an numpy array
        im_array = np.asarray(bytearray(con.read()),
dtype=np.uint8)
        # Convert the numpy array into an image.
        im = cv2.imdecode(im_array, cv2.IMREAD_GRAYSCALE)
        # Get the width and heigh of the image.
        height, width = im.shape
        # Wrap up the width and height in an object and return
        # the encoded JSON.
        return json.dumps({"width" : width, "height" : height})

    except urllib2.HTTPError, e:
        return e.fp.read()

def process_image():
```

```

upfile = request.vars.imagen
fname = request.vars.imagen.filename

#return json.dumps({"Contenido del fichero" :
upfile.file.read()})
#return json.dumps({"Nombre del fichero" : fname})

im_array = np.asarray(bytearray(upfile.file.read()),
dtype=np.uint8)
# Convert the numpy array into an image.
im = cv2.imdecode(im_array, cv2.IMREAD_GRAYSCALE) #
cv2.imdecode(image, cv2.IMREAD_COLOR)

# PROCESS YOUR OPENCV IMAGE HERE. EXAMPLE: Get the width
and heigh of the image.
height, width = im.shape

# Wrap up the width and height in an object and return the
encoded JSON.
return json.dumps({"width" : width, "height" : height})

```

If the server is running, it is possible to check its functionality using the well-known command line utility *curl*. The following are the commands to test both examples, sending an URL of a picture and sending the image file:

URL:

```

curl -F url=http://example.com/image.jpg
http://username.pythonanywhere.com/appname/default/image_dimensions

```

Image:

```

curl -v -X POST
"http://eotest.pythonanywhere.com/opencvdemo/default/process_image"
-F imagen=@fichero_imagen.jpg

```

23.1.2.2. Creating your own server

It is also possible to create your own server using, for example, Python, Django and any OpenCV version, application or library (an example can be seen in <http://www.pyimagesearch.com/2015/05/11/creating-a-face-detection-api-with-python-and-opencv-in-just-5-minutes/>).

23.1.3. libccv

libccv is a minimalist open-source computer vision library for embedded devices developed by Liu Liu. It is meant to be easy to deploy and has a simple and well-organized code structure.

Its main characteristics are:

- It is portable and embeddable.
- It has a clean interface with cached image pre-processing.
- It includes several modern computer vision algorithms, for example:
 - Image classifier

- Frontal face detector, object detectors for pedestrians and cars
- Text detection algorithm
- Object tracking algorithm
- Feature point extraction algorithm

Detailed information of all the functionalities of the library can be found in <http://libccv.org/>.

23.1.4. Quirc

QR codes are a type of high-density matrix barcodes. Quirc is a library for extracting and decoding them from images. It is fast enough to be used with realtime video: extracting and decoding from VGA frame takes about 50 ms on a modern x86 core. Other features that make it a good choice for the purpose of EoT are:

- It has a robust and tolerant recognition algorithm. It can correctly recognise and decode QR codes which are rotated and/or oblique to the camera. It can also distinguish and decode multiple codes within the same image.
- It is easy to use, with a simple API described in a single commented header file.
- It is small and easily embeddable, with no dependencies other than standard C functions.
- It has a very small memory footprint: one byte per image pixel, plus a few kB per decoder object.
- It uses no global mutable state, and is safe to use in a multithreaded application.
- BSD-licensed, with almost no restrictions regarding use and/or modification.

This library will be particularly useful for the museum demonstrator.

23.1.5. Google Cloud Vision API

Recently Google has release the [Google Cloud Vision API](#). The limited preview of this tool encapsulates machine learning models that can learn and predict the content of an image as an easy-to-use REST API. Cloud Vision API quickly classifies images into thousands of categories.

The following set of Google Cloud Vision API features can be currently used:

- **Label/Entity Detection** detects the dominant entity within an image, from a broad set of object categories.
- **Optical Character Recognition** to retrieve text from an image providing automatic language identification.
- **Safe Search Detection** to detect inappropriate content within the studied picture.
- **Facial Detection** along with associated facial features such as eye, nose and mouth placement, and likelihood of over 8 attributes like joy and sorrow.

- **Landmark Detection** to identify popular natural and manmade structures, along with the associated latitude and longitude of the landmark.
- **Logo Detection** to identify product logos within an image.

The EoT project has recently received confirmation about participating in the limited preview, thus, it have been only possible to test the main functionality of the Cloud Vision API following the basic tutorials, which uses the “curl” application. As future work, EoT will provide tools and examples for obtaining results using the EoT device.

Known issues

23.2.1. OpenCV 1.0

The library is not currently optimized for Myriad 2, using only the LeonOS processor.

Limitations

There are some known limitations of the port which work in the original OpenCV:

- The Highgui-User Interface functionality is not supported in the embedded device.
- JPEG and other image formats are not supported. On the contrary, PNG is fully supported through the use of zlib/libpng.
- Video is not supported due to dependencies with libraries optimised for PCs.
- Some parts of the library have been modified to store data in general memory and not in the stack. However, in order to use cascade detectors and other code that requires a significant amount of stack memory to allocate pointers the RTEMS minimum task stack size must be increased using:

```
==#define  CONFIGURE_MINIMUM_TASK_STACK_SIZE      16384==
```

23.2.2. OpenCV 2.4 in the cloud

The functionality depends on the code implemented in the server.

23.2.3. libccv

The library is not currently optimized for Myriad 2, using only the LeonOS processor.

Limitations

There are some known limitations of the port (which work in the original libccv):

- JPEG is not supported
- Reading/writing sqlite3 files is not supported. This feature could be used in read/write methods of:

- ccv_scd
- ccv_convnet

23.2.4. Quirc

The library is not currently optimized for Myriad 2, using only the LeonOS processor.

Limitations

- If QR Code is rotated too much, as in Test 5, the QR Code is detected, but it cannot be decoded.
- If QR Code is too small, as in Test 8, the QR Code is detected, but it cannot be decoded.
- If test image is greater than 225 x 225 px, as in Test 2, Test 3, Test 4 and Test 7, the execution in Myriad produced an error.

```

•
  UART: Unexpected trap (0x09) at address 0x800367C4 Data access
  Exception
  UART: PSR: 0xF34010C1 PC: 0x800367C8 NPC: 0x800367CC TPC: 0x800367C4
  UART: G1: 0x800553C8 G2: 0x00000000 G3: 0x000B2E82 G4: 0x000003E8
  UART: G5: 0x8FFAAEEA G6: 0xFBABAEAA G7: 0xEBAEAA8A I0: 0x00000000
  UART: I1: 0x00000000 I2: 0x80045D00 I3: 0x800597A0 I4: 0x2BADABD0
  UART: I5: 0x00000000 I6: 0x800596F0 I7: 0x800367B4 Y: 0x00020496
  UART:
  UART: L0: 0x000001B9 L1: 0x00000001 L2: 0x80003D54 L3: 0x00000010
  UART: L4: 0x00000020 L5: 0x00000020 L6: 0x701FFDA0 L7: 0x701FFD9C
  UART: I0: 0x80045D00 I1: 0x00000509 I2: 0x800597C0 I3: 0x8005A734
  UART: I4: 0xA65FABD0 I5: 0x07E1EB2B I6: 0x80059758 I7: 0x80003A4C
  UART: SRA: 0x00000000 SA0:0x00000000 SA1: 0x00000000 SA2: 0x00000000
  UART: SA3: 0x00000000 SA4:0x00000000 SA5: 0x00000000

```

Similarly to OpenCV, in order to solve this problem the variable `CONFIGURE_MINIMUM_TASK_STACK_SIZE` (`rtems_config.h`) must be set to a larger value. The used value for all tests has been 22000.

Unit tests

23.3.1. OpenCV 1.0

The tests for OpenCV 1.0 can be found in the repository in the `WorkPackage_3\myriad\apps\OpenCV_examples` folder.

This application runs several examples using the OpenCV library. In most cases, the user needs to check the results saved in the SD card. Assertions check some common execution errors.

The contents of the `testFiles` folder (`OpenCVTests`) must be copied to the SD card (`/mnt/sdcard/`) resulting in `mnt/sdcard/OpenCVTests`.

List of examples

1. Canny edge detector
 - Files required:
 - File: /mnt/sdcard/OpenCVTests/data/nature.png
 - Output:
 - File: /mnt/sdcard/OpenCVTests/results/resultEdge.png
2. Haar-based Cascade Face Detector
 - Files required:
 - /mnt/sdcard/OpenCVTests/data/lena.png
 - /mnt/sdcard/OpenCVTests/data/haarface.xml
 - Output:
 - /mnt/sdcard/OpenCVTests/results/resultFaceDetection.png
3. K-means algorithm
 - Output:
 - /mnt/sdcard/OpenCVTests/results/resultTestKmeans.png
4. Contours
 - Files required:
 - /mnt/sdcard/OpenCVTests/results/resultTestContours_1.png
 - Output:
 - /mnt/sdcard/OpenCVTests/results/resultTestContours_1.png
 - /mnt/sdcard/OpenCVTests/results/resultTestContours_2.png
5. Histogram/LUT
 - Files required:
 - /mnt/sdcard/OpenCVTests/data/baboon.png
 - Output:
 - /mnt/sdcard/OpenCVTests/results/resultDemHist.png
6. Delaunay Triangulation
 - Output:
 - /mnt/sdcard/OpenCVTests/results/delaunay/delaunay[n].png ([n]={0,1,2})
 - /mnt/sdcard/OpenCVTests/results/delaunay/delaunay[n]_s.png ([n]={0,1,2})
 - /mnt/sdcard/OpenCVTests/results/delaunay/voronoi.png
7. testPyramidSegmentation
 - Files required:
 - /mnt/sdcard/OpenCVTests/data/fruits.png
 - Output:
 - /mnt/sdcard/OpenCVTests/results/resultPyramidSegmentation.png
8. testSquareDetector
 - Files required:
 - /mnt/sdcard/OpenCVTests/data/pic[n].png ([n]={1,2,3,4,5,6})
 - Output:
 - /mnt/sdcard/OpenCVTests/results/squares/ResultSquarespic[n].png ([n]={1,2,3,4,5,6})
9. Watershed
 - Files required:
 - /mnt/sdcard/OpenCVTests/data/fruits.png
 - Output:
 - /mnt/sdcard/OpenCVTests/results/resultWatershed.png
10. Morphological Operations
 - Files required:

- /mnt/sdcard/OpenCVTests/data/baboon.png
- Output:
 - /mnt/sdcard/OpenCVTests/results/morphology/baboon_E_OC.png
 - /mnt/sdcard/OpenCVTests/results/morphology/baboon_E_ED.png
 - /mnt/sdcard/OpenCVTests/results/morphology/baboon_R_OC.png
 - /mnt/sdcard/OpenCVTests/results/morphology/baboon_R_ED.png
 - /mnt/sdcard/OpenCVTests/results/morphology/baboon_C_OC.png
 - /mnt/sdcard/OpenCVTests/results/morphology/baboon_C_ED.png
- 11. Fourier
 - Files required:
 - /mnt/sdcard/OpenCVTests/data/suit.png
 - Output:
 - /mnt/sdcard/OpenCVTests/results/dftres2.png
- 12. Inpainting
 - Files required:
 - /mnt/sdcard/OpenCVTests/data/lena.png
 - Output:
 - /mnt/sdcard/OpenCVTests/results/inpainted.png
- 13. Min Area Rectangle
 - Output:
 - /mnt/sdcard/OpenCVTests/results/rectCircle.png
- 14. Lucas Kanade
 - Files required:
 - /mnt/sdcard/OpenCVTests/data/suit.png
 - Output:
 - /mnt/sdcard/OpenCVTests/results/LKDemo.png
- 15. DOG
 - Files required:
 - /mnt/sdcard/OpenCVTests/data/DunLoghaire_320x240.png
 - Output:
 - /mnt/sdcard/OpenCVTests/results/DOG.png
- 16. Dilation
 - Files required:
 - /mnt/sdcard/OpenCVTests/data/lena_512x512_luma.png
 - Output:
 - /mnt/sdcard/OpenCVTests/results/dilate2.png
- 17. Harris Corners
 - Files required:
 - /mnt/sdcard/OpenCVTests/data/lena_512x512_luma.png
 - Output:
 - /mnt/sdcard/OpenCVTests/results/cornerharris.png
- 18. Median Filter
 - Files required:
 - /mnt/sdcard/OpenCVTests/data/ref_chroma_median_out_512x512_P444_8bpp.png
 - Output:
 - /mnt/sdcard/OpenCVTests/results/medianfilter.png

Expected output

```
UART: Thread 1 created
UART: Starting run
UART: SD card mounted? 1
UART: + Test 1 passed. OK
UART: + Test 2 passed. OK
UART: + Test 3 passed. OK
UART: + Test 4 passed. OK
UART: + Test 5 passed. OK
UART: + Test 6 passed. OK
UART: + Test 7 passed. OK
UART: + Test 8 passed. OK
UART: + Test 9 passed. OK
UART: + Test 10 passed. OK
UART: + Test 11 passed. OK
UART: + Test 12 passed. OK
UART: + Test 13 passed. OK
UART: + Test 14 passed. OK
UART: TEST 15 Usec CPU time: 181010
UART: TEST 16 Usec CPU time: 574738
UART: TEST 17 Usec CPU time: 1079811
UART: TEST 18 Usec CPU time: 598534
UART: Unmounting SD card...
UART: Tests ended
```

Benchmarks

The following examples of the MvCV library of the MDK have been implemented using OpenCV:

- 1) SimpleCrossCompilableCVPipe
- 2) SippTutC0
- 3) testHWHarrisCorners
- 4) testHwMedian

The computational time in seconds used for the OpenCV version VS the MvCv version can be shown in the following graph:

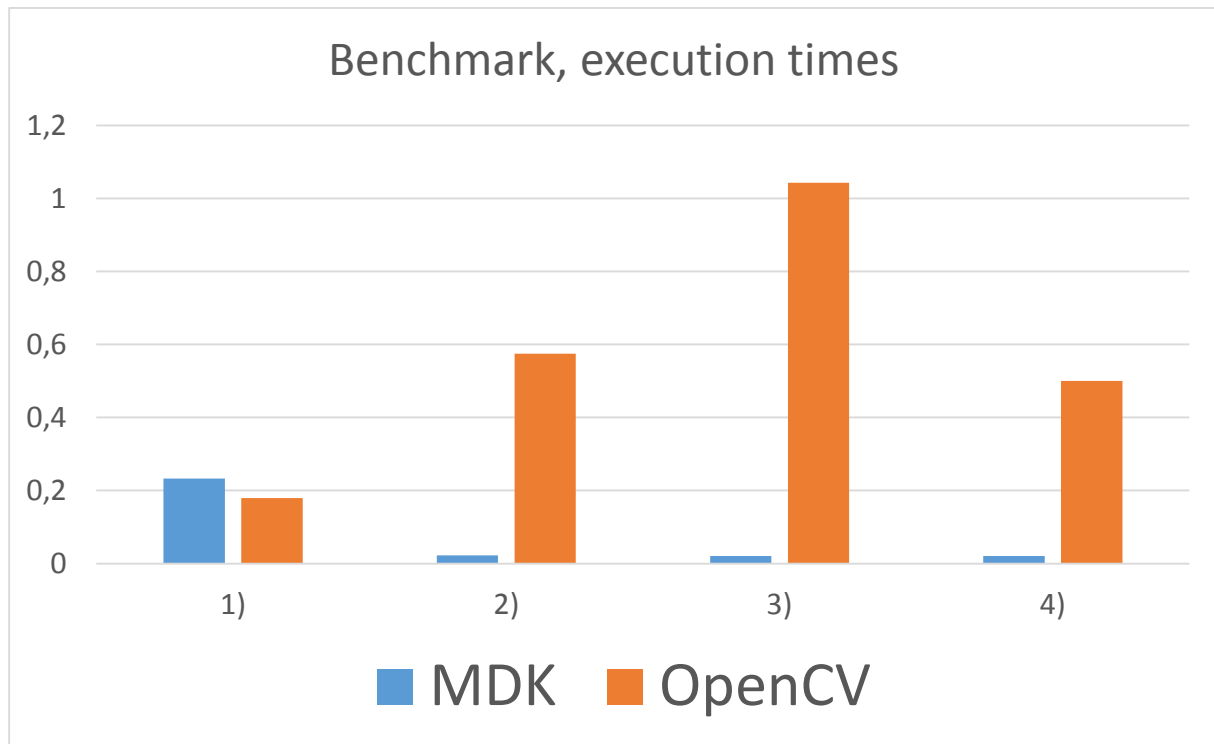


Figure 41. MDK vs OpenCV benchmarks

23.3.2. OpenCV 2.4 in the cloud

The test for OpenCV 2.4 in the cloud can be found in the repository in the WorkPackage_3\myriad\apps\RTEMS_API_REST_example folder.

This application connects to a server previously created in the cloud in <https://www.pythonanywhere.com>, which comes with OpenCV 2.4 pre-installed.

The sample application sends a POST request to the REST API of the server, including a picture which is read from the SD card. The server example returns its width and height in pixels.

The data of the access point must be changed in line 105:

```
connectToAP("AndroidAP", "tqwg4654", SL_SEC_TYPE_WPA_WPA2, 20);
```

The picture 0.9kb.jpg must be copied to the SD card (/mnt/sdcard/).

Expected output

UART: Mounted= 1 , opening file of 1147 bytes

UART:

UART: Closing file of 1147 bytes

UART:

UART: Connecting...

UART: [GENERAL EVENT]

UART: Connecting...

```
UART: Connecting...
UART: Own IP
UART: IP Address 192.168.43.222
UART: IP Address 50.19.109.98
UART: Received: HTTP/1.1 200 OK
UART: Server: ngx_openresty/1.4.3.6
UART: Date: Mon, 15 Feb 2016 17:39:53 GMT
UART: Content-Type: text/html; charset=utf-8
UART: Transfer-Encoding: chunked
UART: Connection: keep-alive
UART: Vary: Accept-Encoding
UART: X-Powered-By: web2py
UART: Set-Cookie: session_id_opencvdemo=37.29.212.73-93514171-09ee-4e78-bf29-6
UART: 7e9e6b1818f; Path=/
UART: Expires: Mon, 15 Feb 2016 17:39:53 GMT
UART: Pragma: no-cache
UART: Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check
UART: eck=0
UART: X-Clacks-Overhead: GNU Terry Pratchett
UART:
UART: 1b
UART: {"width": 26, "height": 24}
UART: 0
```

23.3.3. libccv

The test for libccv can be found in the repository in the WorkPackage_3\myriad\apps\libccv_examples folder.

This application runs several examples using the libccv library. Some of them compare the obtained results vs expected result, giving an assertion if the values are different. In others, the user needs to check the results saved in the SD card.

The contents of the testFiles folder (data and samples) must be copied to the SD card (/mnt/sdcard/).

List of examples

1. Canny edge detector.
 - Files required:
 - samples/blackbox.png
 - data/blackbox.canny.bin
 - Output:
 - File: samples/test1.png (1 if border has been detected, 0 if not)
2. Pedestrian cascade detector using Integral Channel Features.
 - Files required:
 - samples/pedestrian.icf
 - samples/pedestrian2.png

- Output:
 - Text: Number of pedestrians detected. For each detection, classification confidence and x, y, width and height of the rectangle.
- 3. Matrix addition operation.
- 4. Sobel operations.
 - Files required:
 - samples/chessbox.bmp
 - data/chessbox.sobel.x.bin
 - data/chessbox.sobel.y.bin
 - data/chessbox.sobel.u.bin
 - data/chessbox.sobel.v.bin
 - data/chessbox.sobel.x.3.bin
 - data/chessbox.sobel.y.3.bin
 - data/chessbox.sobel.x.5.bin
 - Output:
 - samples/test4.png (partial derivative on y within 5x5 window)
- 5. Otsu threshold calculation.
- 6. Image contrast modification.
 - Files required:
 - samples/nature.bmp
 - data/nature.contrast.0.5.bin
 - data/nature.contrast.1.5.bin
 - Output:
 - samples/test6_1.png (reducing contrast to 0.5)
 - samples/test6_2.png (increasing contrast 1.5 times)
- 7. Perspective transform on a picture.
 - Files required:
 - samples/chessbox.bmp
 - data/chessbox.perspective.transform.bin
 - Output:
 - samples/test7.png (picture rotated along y-axis for 30°)
- 8. Histogram of gradients calculation.
 - Files required:
 - samples/nature.bmp
 - Output:
 - samples/test8.png (values obtained)
- 9. Scale Invariant Feature Transform.
 - Files required:
 - samples/book.png
 - Output:
 - Text: List of keypoints matching.
- 10. CBLAS matrix multiplication.

11. Convolutional network of 11x11 on 225x225 with uniform weights.
12. Convolutional network of 5x5 on 27x27 with non-uniform weights.
13. Convolutional network of 5x5x4 on 27x27x8 partitioned by 2.
14. Stroke Width Transform (text detection).
 - Files required:
 - samples/text-detect.png
 - Output:
 - Text: Number of texts detected. For each detection, x, y, width and height of the rectangle.
15. Face Cascade Detector
 - Files required:
 - samples/face (folder with cascade.txt and stage-0.txt to stage-15.txt files)
 - samples/suit.png
 - Output:
 - Text: Number of faces detected. For each detection, classification confidence and x, y, width and height of the rectangle.

Expected output

```

UART: Thread 1 created
UART: Starting run
UART: Example 1, canny and write png
UART: Value of image: rows - 500, cols - 500, type - 1074794497
UART: Example 2, pedestrian cascade detector (several days)
UART: Loading files
UART: Detecting pedestrians
UART: Results:
UART: 20 36 50 152 0.074491
UART: total : 1 detected
UART: Example 3, matrix addition
UART: Example 4, sobel
UART: Example 5, otsu threshold
UART: Example 6, image contrast
UART: Example 7, perspective transform
UART: Example 8, HOG
UART: Example 9, SIFT
UART: Pictures read
UART: Keypoints calculated obj 1179
UART: Keypoints calculated img 1179
UART: 73.833488 2.090770 => 73.833488 2.090770
UART: 30.298519 2.555899 => 30.298519 2.555899
UART: 10.025833 2.887055 => 10.025833 2.887055
.....
.....
.....

```

```
UART: 105.916695 222.729172 => 105.916695 222.729172
UART: 105.916695 222.729172 => 105.916695 222.729172
UART: 117.846596 238.603943 => 117.846596 238.603943
UART: 75.254646 144.538391 => 75.254646 144.538391
UART: 338x289 on 338x289
UART: 1179 keypoints out of 1179 are matched
UART: elapsed time : 0
UART: Example 10, CBLAS mat multiplication
UART: Example 11, Convolutional network
UART: Example 12, convolutional network of 5x5 on 27x27 with non-uniform
weigh
UART: ts
UART: Example 13, convolutional network of 5x5x4 on 27x27x8 partitioned by 2
UART: Example 14, Text detect
UART: 64 179 104 13
UART: 182 178 67 13
UART: 89 197 136 18
UART: 183 197 34 13
UART: total : 4 detected
UART: Example 15, Detect faces
UART: 144 59 53 53 2.364821
UART: total : 1 detected
UART: Tests ended
```

23.3.4. Quirc

The test for Quirc can be found in the repository in the WorkPackage_3\myriad\apps\Quirc_example folder.

This application runs several examples using the Quirc library. For each test, the output will be the number of QR codes detected and their content. If an error occurs, the output will display its type.

The content of the testFiles folder must be copied to the SD card (/mnt/sdcard/) resulting in /mnt/sdcard/QuircTest/.

To obtain information about the used QR Codes, some QR Codes have been uploaded to <https://zxing.org/w/>.

List of examples and expected output

Test 1

This example is a basic test. The image contains a unique QR code, without margins on the sides. The main purpose is to check the proper behavior of the library.

Files required:

- File: /mnt/sdcard/QuircTest/TestQR.png

The image size is 150 x 150 px.

 Decode Succeeded		
Raw text	EoT	
Raw bytes	40 34 56 f5 40 ec 11 ec 11 ec 11 ec 11 ec 11 ec 11	
Barcode format	QR_CODE	
Parsed Result Type	TEXT	
Parsed Result	EoT	

Figure 42. QR first test image

In this example, the output is:

```
UART: Test 1
UART: num_codes: 1
UART: Data: EoT
UART: -----
```

Test 2

Second test image contains a QR Code and text. The main purpose is to check if Quirc library distinguish between QR Codes and text, decoding the code and ignoring the text.

In contrast to Test 1, this image has margins. In this way, we can know if Quirc looks for the QR Code.

Files required:

- File: /mnt/sdcard/QuircTest/TestTex.png

The image size is 490 x 490 px.

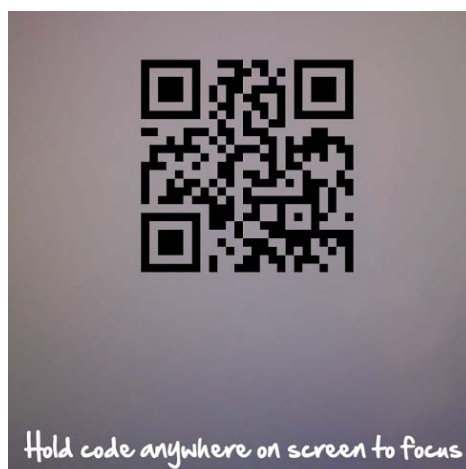


Figure 43. QR second test image

The output of the Test 2 is:

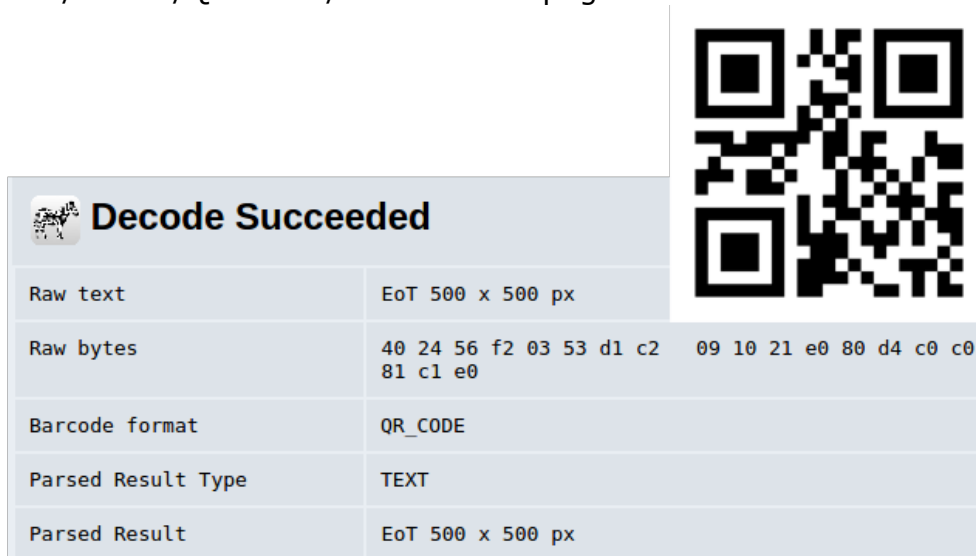
```
UART: Test 2
UART: num_codes: 1
UART: Data: http://www.qrdroid.com
UART: -----
```

Test 3

Just as Test 1, in Test 3 we will use an image without margins and its content will be a unique QR Code. As opposed to Test 1, the test image size is 500 x 500 px. Test 3 has as its objective to verify if Myriad can use Quirc library to decode images with this size.

Files required:

- File: /mnt/sdcard/QuircTest/Test500x500.png



Decode Succeeded	
Raw text	EoT 500 x 500 px
Raw bytes	40 24 56 f2 03 53 d1 c2 09 10 21 e0 80 d4 c0 c0 81 c1 e0
Barcode format	QR_CODE
Parsed Result Type	TEXT
Parsed Result	EoT 500 x 500 px

Figure 44. QR third test image

The output is:

```
UART: Test 3
UART: num_codes: 1
UART: Data: EoT 500 x 500 px
UART: -----
```


Test 4

The previous tests used images without rotation. Therefore, the main purpose is to check if Quirc library can decode rotated QR codes. In this case, the image size is 300 x 300 px.

Files required:

- File: /mnt/sdcard/QuircTest/TestRotated.png




Decode Succeeded

Raw text	Rotated Code EoT
Raw bytes	41 05 26 f7 46 17 46 56 42 04 36 f6 46 52 04 56 f5 40 ec
Barcode format	QR_CODE
Parsed Result Type	TEXT
Parsed Result	Rotated Code EoT

Figure 45. QR fourth test image

The output of Test 4 is:

```

UART: Test 4
UART: num_codes: 1
UART: Data: Rotated Code EoT
UART: -----

```

Test 5

Test 5 is an expansion of Test 4. Once again, the QR Code is rotated, but in Test 5 rotation is greater than Test 4. Test 5 has as its objective to verify if rotation affects the recognition and decoding.

Files required:

- File: /mnt/sdcard/QuircTest/TestRotatedError.png

The image size is 225 x 225 px.

**Figure 46. QR fifth test image**

The output is:

```
UART: Test 5
UART: num_codes: 1
UART: Data: ECC failure
UART: -----
```

Even though the example recognizes a QR Code, Quirc library cannot decode it.

Test 6

In this test, the QR Code will be red. The main objective is to check if Quirc library can recognize different colors. The image size is 50 x 50 px.

Files required:

- File: /mnt/sdcard/QuircTest/TestRedCode.png

Decode Succeeded	
Raw text	Red Code EoT
Raw bytes	40 c5 26 56 42 04 36 f6 46 52 04 56 f5 40 ec 11 ec 11 ec
Barcode format	QR_CODE
Parsed Result Type	TEXT
Parsed Result	Red Code EoT

Figure 47. QR sixth test image

The output is:

```
UART: Test 6
UART: num_codes: 1
UART: Data: Red Code EoT
UART: -----
```

Test 7

This test aims at verifying that the Quirc port can recognize and decode several codes in the same image. The test image contains two QR Codes, one is rotated.

Files required:

- File: /mnt/sdcard/QuircTest/TestTwoQR.png

The image size is 400 x 300 px.



Figure 48. QR seventh test image

The output of Test 7 is:

```
UART: Test 7
UART: num_codes: 2
UART: Data: EoT Data: http://eyesofthings.eu/
UART: -----
```

Test 8

The main purpose of this test is to check if the Quirc library port can work with very small sizes. The test image size is 25 x 25 px.

Files required:

- File: /mnt/sdcard/QuircTest/TestTooSmall.png

The output is:

```
UART: Test 8
UART: num_codes: 1
UART: Data: Invalid version
UART: -----
```

Like that test 5, Quirc library detects a QR Code, but the library cannot decode it. In this case, the error is different.

Benchmark

The following table shows the times in the execution of example program. For each test, there are 3 times: Time in loadCode (reads the image and creates the necessary structures), time in readQR (detects and reads the information) and total time.

From these times, we can draw the following conclusions.

- As expected, method loadCode uses most of the time.
- The higher image size, the higher time.
- The higher number of QR codes, the higher time to decode QR code.

Licensing

23.4.1. OpenCV 1.0

OpenCV is released under a BSD license and hence it is free for both academic and commercial use.

By downloading, copying, installing or using the software you agree to this license. If you do not agree to this license, do not download, install, copy or use the software.

License Agreement For Open Source Computer Vision Library

Copyright (C) 2000-2008, Intel Corporation, all rights reserved. Copyright (C) 2008-2011, Willow Garage Inc., all rights reserved. Third party copyrights are property of their respective owners.

- Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

The name of the copyright holders may not be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided by the copyright holders and contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the Intel Corporation or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but

not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

****3rd Party Modules****

The zlib/libpng License (Zlib)

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

This notice may not be removed or altered from any source distribution.

23.4.2. OpenCV 2.4 in the cloud

No license issues.

23.4.3. libccv

Libccv source code is distributed under BSD 3-clause License.

Files in directories ./samples are licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/>.

Copyright (c) 2010, Liu Liu
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of the authors nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

3rd Party Modules

kissfft:

Copyright (c) 2003-2010 Mark Borgerding

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- * Neither the author nor the names of any contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

dSFMT:

Copyright (c) 2007, 2008, 2009 Mutsuo Saito, Makoto Matsumoto and Hiroshima University.

Copyright (c) 2011, 2002 Mutsuo Saito, Makoto Matsumoto, Hiroshima University and The University of Tokyo.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- * Neither the name of the Hiroshima University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

SFMT:

Copyright (c) 2006,2007 Mutsuo Saito, Makoto Matsumoto and Hiroshima University.

Copyright (c) 2012 Mutsuo Saito, Makoto Matsumoto, Hiroshima University and The University of Tokyo.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- * Neither the names of Hiroshima University, The University of Tokyo nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

libebb:

see doc/index.html and examples/hello_world.c for explanation

webpage: <http://tinyclouds.org/libebb/>

git repository: <http://github.com/ry/libebb/tree/master>

(The MIT) LICENSE

Copyright © 2008 Ryah Dahl (ry@tinyclouds.org)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

multipart-parser-c:

Based on node-formidable by Felix Geisendörfer

Igor Afonov - afonov@gmail.com - 2012

MIT License - <http://www.opensource.org/licenses/mit-license.php>

sha1: TODO

sqlite3:

2001 September 15

The author disclaims copyright to this source code. In place of a legal notice, here is a blessing:

May you do good and not evil.
May you find forgiveness for yourself and forgive others.
May you share freely, never taking more than you give.

23.4.4. Quirc

<https://github.com/dlbeer/quirc>

Copyright (C) 2010-2012 Daniel Beer <dlbeer@gmail.com>

Permission to use, copy, modify, and/or distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Code

23.5.1. OpenCV 1.0

The WifiFunctions library can be found in the folder WorkPackage_3\myriad\libs\leon\OpenCV of EoT repository.

Dependences

- zlib
- libpng 1.4 (included inside the EoT OpenCV library)
- SDCardIO

23.5.2. OpenCV 2.4 in the cloud

An example with the server can be found in WorkPackage_3\myriad\apps\RTEMS_API_REST_example of EoT repository.

Dependences

- SDCardIO

- WifiFunctions

23.5.3. libccv

The libccv library can be found in the folder WorkPackage_3\myriad\libs\leon\libccv of EoT repository.

Dependences

- zlib
- cblas
- libpng

23.5.4. Quirc

The Quirc library can be found in the folder WorkPackage_3\myriad\libs\leon\Quirc of EoT repository.

Dependences

No external dependences.

Conclusions and Future work

Common vision libraries such as: OpenCV, libccv, or Quirc have been ported to the EoT device. Moreover, the EoT platform allows the connection with external computer vision libraries in the cloud, in this case, with OpenCV 2.4 through Pythonanywhere.

Since Google Cloud Vision API offers a wide range of possibilities for computer vision, native support will be added to the EoT platform. The work done in the integration of OpenCV 2.4 with Pythonanywhere, where POST requests to a web service are sent, will be used to achieve the access to the Google Cloud vision API.

24. MOTOR CONTROL

Introduction

Although the prices of the robot and drones has dropped considerably in the last few years, they lack of autonomous behaviour requiring human intervention to operate properly. The Motor Control module enables EoT board to provide autonomy and connectivity to robots and drones.

The Motor Control module provides EoT board with a means of communicating with a wide variety of devices such as robots, drones, actuators and sensors. By this mean, it is possible to control small robot cars and flying drones; open and close doors as required; monitor devices such as fridges, vending machines; or control systems like air conditioning and lighting which might not have access to internet or wireless connectivity.

There is no standard interface or protocol for communication between devices, at the moment. Therefore, the Motor Control module encapsulates the hardware details of the communication into an easy-to-use API, which could be extended to support multiple devices.

As an application, Motion Control Module of EoT board can be used to control a robot car. As the proof of concept for this application, the robot car Cherokey 4WD from DFRobot¹⁶ (Figure 49) was selected as it is low-cost and has a robust aluminium frame. Additionally, an Android App was developed to allow the remote control of a ground vehicle via WiFi.

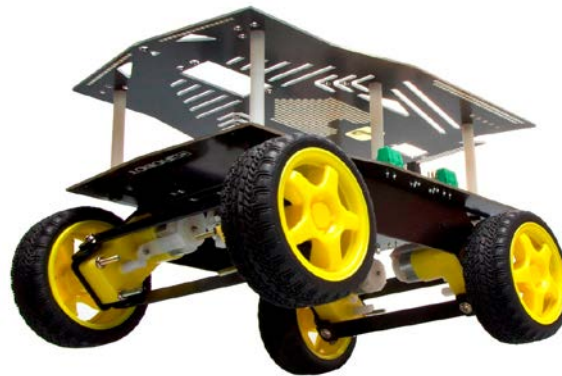


Figure 49. Cherokey 4WD from DFRobot.

The remote control of a robot car has two main components:

- Motion Control API
- Android app for Ground Robot control.

Motion Control API

The Motion Control API is a C/C++ library which provides access to the motion of a ground car or a drone in an easy-to-use interface. This API encapsulates the hardware specifics allowing to create easy-to-maintain code and port across

¹⁶ <http://www.dfrobot.com>

projects. The API was initially developed on Arduino before being ported to Raspberry Pi and finally to Myriad2.

The current implementation of the Motion Control API only provides function for controlling ground vehicles which are:

- Move forward,
- Move backward,
- Tank-turn left
- Tank-turn right
- Stop

Currently the Cherokey¹⁷ 4WD from DFRobot is the only fully supported robot by the Motion Control API. The incorporation of support for other robots or drones into the Motion Control API remains subject of future work.

The following codes shows sample code for controlling the Cherokey 4WD from the EoT board using the Motor Control Unit and the available functions for the Cherokey4WD on the Motion Control API as defined in the file Cherokey4WD.h.

```
// -- Other includes --
#include "Cherokey4WD/Cherokey4WD.h"

#define GPIO_M1_EN_PIN      45
#define GPIO_M2_EN_PIN      46
#define GPIO_M1_PWM_PIN     49
#define GPIO_M2_PWM_PIN     50

#define CHEROKEY_DIRECTION_FORWARD_STATE    1

int main(void)
{
    // -- Board initialization code --

    // Set up the cherokey
    Cherokey4WDSetup(GPIO_M1_EN_PIN, GPIO_M1_PWM_PIN, GPIO_M2_EN_PIN, GPIO_M2_PWM_PIN,
        CHEROKEY_DIRECTION_FORWARD_STATE);

    // Set the direction to forward at maximum speed
    Cherokey4WDForward(255, 255);

    // -- Wait for 1 second, so it keeps moving for the whole second --

    // Set the direction to backwards at half speed
    Cherokey4WDBackward(128, 128);

    // -- Wait for 1 second, so it keeps moving for the whole second --

    // Tank turn left at half speed
    Cherokey4WDTurnLeft(128, 128);

    // -- Wait for 1 second, so it keeps moving for the whole second --

    // Tank turn right at maximum speed
    Cherokey4WDTurnRight(255, 255);

    // -- Wait for 1 second, so it keeps moving for the whole second --

    // -- Board finalization code --

    return 0;
}
```

¹⁷ <http://goo.gl/7Gxi6n>

```

File Cherokey4WD.h
/**
 * Set up the pins which are going to be used to control the Cherokey4WD.
 * Set the pin state which corresponds to the forward direction (HIGH or LOW)
 * @param pin_m1_enable
 * @param pin_m1_pwm
 * @param pin_m2_enable
 * @param pin_m2_pwm
 * @param pin_state_forward
 */
void Cherokey4WDSetup(uint8 pin_m1_enable, uint8 pin_m1_pwm,
                     uint8 pin_m2_enable, uint8 pin_m2_pwm, uint8 pin_state_forward);

/**
 * Set the direction and speed for the wheels of the car
 * @param direction_left   Direction of the wheels on the left side
 * @param speed_left       Speed of the wheels on the left side (0 to 255)
 * @param direction_right  Direction of the wheels on the right side
 * @param speed_right      Speed of the wheels on the right side (0 to 255)
 */
void Cherokey4WDSetDirectionSpeed(uint8 direction_left, uint8 speed_left,
                                   uint8 direction_right, uint8 speed_right);

/**
 * Stop the car
 */
void Cherokey4WDStop();

/**
 * Moves the Cherokey 4WD forward at the specified speed (0 to 255) on each of the wheels
 * (left and right wheels)
 * @param speed_left
 * @param speed_right
 */
void Cherokey4WDForward(uint8 speed_left, uint8 speed_right);

/**
 * Moves the Cherokey 4WD backward at the specified speed (0 to 255) on each of the
 * wheels (left and right wheels)
 * @param speed_left
 * @param speed_right
 */
void Cherokey4WDBackward(uint8 speed_left, uint8 speed_right);

/**
 * Turns the Cherokey 4WD to the left at the specified speed (0 to 255) on each of the
 * wheels (left and right wheels)
 * @param speed_left
 * @param speed_right
 */
void Cherokey4WDTurnLeft(uint8 speed_left, uint8 speed_right);

/**
 * Turns the Cherokey 4WD to the right at the specified speed (0 to 255) on each of the
 * wheels (left and right wheels)
 * @param speed_left
 * @param speed_right
 */
void Cherokey4WDTurnRight(uint8 speed_left, uint8 speed_right);

```

Sample code for controlling Cherokey 4WD from EoT board.

Motion functions for Cherokey4WD as defined in the API file Cherokey4WD.h.

24.2.1. Cherokey 4WD

The Cherokee 4WD from DFRobot is controlled by four signals, two ENABLE for controlling the direction of the wheels and two PWM to control the speed. For more information on the Cherokee 4WD please refer to:

http://www.dfrobot.com/wiki/index.php?title=Cherokee_4WD_Mobile_Platform_%28SKU:ROB0102%29

24.2.2. EoT DevBoard and Cherokee 4WD Hardware interface

The Cherokee 4WD is controlled by four signals in the range of 0-5V, which is compatible with the pins in EoT Motor Control Header. The hardware connection between the Cherokee 4WD and EoT board is summarized in Table 1.

Figure 50 shows a schematic diagram of the connections.

Table 1. Connection between the EoT DevBoard and the Cherokee 4WD

EoT Motor Control Header Pin	Cherokee 4WD Pin
3: PWM0	D5:M1_PWM
4: PWM1	D7:M2_PWM
5: DIR0	D4:M1_EN
6: DIR1	D6:M2_EN
7: GND	GND

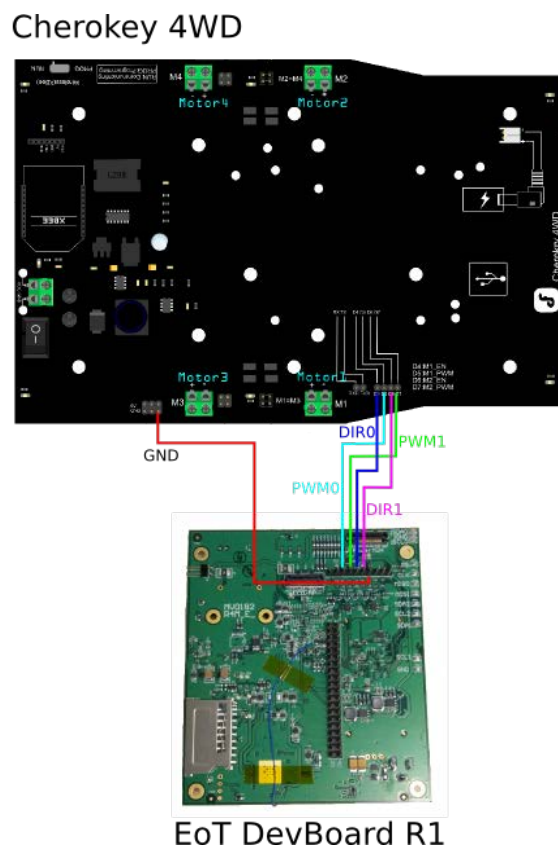


Figure 50. Connection between the EoT DevBoard and the Cherokee 4WD

■ Android App for Ground Robot control.

The Ground Robot Android app (called "GRApp", hereafter) provides an easy to use interface for controlling a ground robot (See Figure 51). As proof of concept, the communication between the GRApp and EoT board is performed using a custom designed protocol, namely the "Mobile Robot Control protocol". This protocol uses UDP sockets to send and receive data packages. This protocol is presented in section 24.3.1.

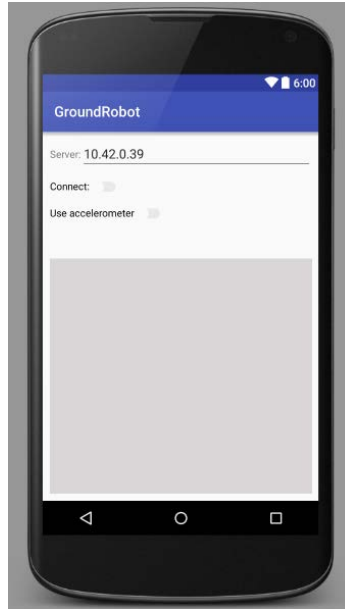


Figure 51. Ground Robot Android App

The user interface of GRApp has two modes for controlling a Ground Robot, the first mode uses the touch pad to create commands for moving forward, backward, tank turn left and tank turn right. The second mode the app uses the phone gyroscope to detect its orientation and issues the motion command according to this. GRApp needs the IP address of the EoT board in order to send the motion commands. The IP address should be introduced into the Server field (Figure 52).



Figure 52. Entering the IP address of the EoT board

After the IP of EoT board has been introduced into the Server field of GRApp, the "Connect" field should be set to *on* state. Then a connectivity test is performed to ensure the communication between GRApp and EoT board is successful. While the testing is performed the message "Connecting to Cherokee 4WD" is shown below the connect button. If the connection is successful, then the message "Connected to Cherokee 4WD" is shown (Figure 53).



Figure 53. Connecting and Connected to EoT board

Once GRApp succeeds in communicating with EoT board, it is possible to start sending motion commands to Cherokee 4WD. The large grey rectangle behaves as a touch pad where the top, bottom, right, left sides issue commands to move forward, move backward, turn right, turn left, respectively.

Alternately, the gyroscope and accelerometer of the phone could be used to issue the motion commands. By setting the switch "Use accelerometer" to on state, it would set the app into landscape mode and start issuing motion commands to EoT board. In this mode, the tilt angle of the phone controls the speed and direction of the motion. An indicator is drawn to shown the direction of motion sent to EoT board. Figure 54 shows an example of the screen produced by a light tilt to the front.

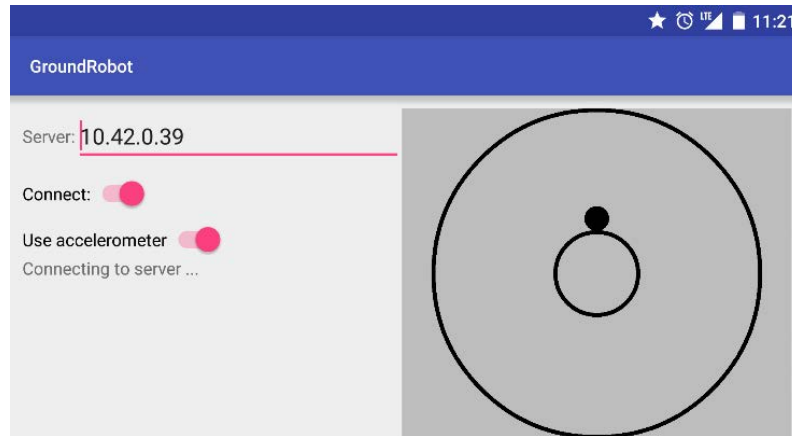


Figure 54. Motion control using phone orientation

24.3.1. Mobile Robot Control Protocol

The Ground Robot Android app provides an easy to use interface for remote control of a ground robot over WiFi. The Mobile Robot Control protocol was designed as a proof-of-concept for enabling a simple communication protocol between GRApp and EoT board. For the sake of robustness, this protocol could be replaced by other widely available protocols such as MQTT. At the time of writing this section, only one command, "*GroundRobotMotion*", is supported by this protocol. This command sets the speed and direction of the wheels. Table 2 summarizes this protocol.

Table 2. Ground Vehicle Motion Command.

Byte	Name	Value	Description
0	Start of packet	0xFF	Fixed
1	Protocol version	0x01	Version of the protocol used. Fixed.
2	Packet length	0x04	Number of bytes in the payload of the packet. It is counted from the byte 3 to the one byte before the CRC.
3	Command ID	0x02	ID of the command sent.
4	Direction	0xFF	Direction of the movement, the possible values are: 's' – stop 'f' – forward

			'b' – backward 'l' – left 'r' – right The ASCII code of the character is set as data, ie. To issue the stop command the ASCII value of 's' which is 0x73 is set as value of the field.
5	Speed left	0 – 255	Speed in the range of 0 to 255
6	Speed right	0 – 255	Speed in the range of 0 to 255
7	Checksum	0xXX	Checksum for validate integrity of the data. It is calculated by applying xor from byte 3 (Command ID) to one byte before the checksum.
8	End of packet	0x55	Fixed

The commands should be issued every time that the motion state of the vehicle is going to be changed. In other words, if a forward motion command is sent to vehicle, it would keep moving forward until a stop or other command is sent.

Known issues

Currently the Motion Control API does not have support for other robots than the Cherokey 4WD. Currently, work to support iCreate Kobuki Robot is in progress.

Unit tests

In order to test the hardware of the Motion Control module on EoT board, unit tests based on an Arduino Uno board were developed to simulate the presence of a robot. These tests change the status of the GPIO pins and send I2C messages to validate the required hardware components. The connection between Arduino and EoT boards should be performed as detailed in Table 3.

Table 3. Connections between Arduino Uno and EoT board for the Motion Control Hardware Tests

Arduino pin	Motor Control header pin
A4	I2C_SDA
A5	I2C_SCL
8	DIR0
9	DIR1
10	BRAKE0
11	BRAKE1
12	PWM0
13	PWM1

Licensing

This library contains proprietary intellectual property of Movidius Ltd. The library and its source code are protected by various copyrights and portions may also be

protected by patents or other legal protections. This software is licensed for use with the Myriad family of processors.

THIS CODE AND INFORMATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL THE COPYRIGHT OWNER BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
<http://www.movidius.com/>

■ Code

In order to compile and run the source code required for controlling the Cherokee 4WD using the EoT board, it is required a working installation of the Movidius MDK version $\geq 15.06.4$.

GRApp was tested to compile and work successfully using the Android API level 23. It was developed using Android Studio v 1.5.0.

The source code for the EoT board and the GRApp can be found at: <https://github.com/CTOmovidius/MvBot/tree/master/Robot/Cherokey4WD>.

The documentation for the relevant code is in Annex 12.

■ Conclusions and Future work

The Motor Control Unit allows interaction between EoT board and a wide set of devices such as robots, drones, doors and different sensors. As part of the Motion Control Unit, the Motion Control API was developed to provide an easy-to-use software interface to different robots. Currently only the Cherokee4WD is fully supported by this API. As the proof-of-concept, a sample application was created using EoT board to enable the remote control of Cherokee 4WD from an Android smartphone. Future work includes expanding the support for different robots such as iCreate Kobuki robot or drones like H8 mini and upgrading GRApp to use a standard communication protocol such as MQTT.

25. CONCLUSIONS

This document describes the firmware software developed in EoT as of March 1st, 2016. This firmware is intended to be compiled for and executed on the Myriad 2 SoC along with a number of other hardware components such as Wifi, SD card, etc. The software modules have been generated according to the task structure of WP3 (Software). Some of the tasks shown in this document were not in the original DoW but were added later since they were considered relevant for the project (Other vision libraries, Motor control, CNN, audio input & codec). The main participating partners are UCLM, DFKI and Movidius. Deliverables D3.1 and D3.2 described the Control Mode software generated for desktop and Android platforms, along with the associated firmware side, and thus they will not be covered here.

Work in EoT firmware will continue mainly to finalize the remaining CV modules, and in terms of testing and further optimization of existing modules.

26. GLOSSARY

AON	Always ON
AP	Access Point
API	Application Programming Interface
BRISK	Binary Robust Invariant Scalable Keypoints
BSD	Berkeley Software Distribution
CISC	Complex Instruction Set Computing
CNN	Convolutional Neural Network
CSS	CPU Subs-System
DDR	Double Data Rate
DIP	Dual in-line Package
DMA	Direct Memory Access
DSS	DDR Sub-System
DoW	Description of Work
GPIO	General Purpose Input/Output
GPL	General Public License
HD	Hard Disk
HOG	Histogram of Oriented Gradients
HTTP	Hypertext Transfer Protocol
HW	Hardware
I2C	Inter-Integrated Circuit
I2S	Integrated Interchip Sound
IO	Input/Output
ISR	Interrupt Service Routine
JPEG	Joint Photographic Experts Group
LED	Light-Emitting Diode
LK	Lucas-Kanade
MAC	Media Access Control
MDK	Movidijs Development Kit
MIPI	Mobile Industry Processor Interface
MQTT	Message Query Telemetry Transport
MSS	Media Sub-System
MvCv	Movidijs Computer Vision
OS	Operating System
PC	Personal Computer
PDF	Portable Document Format
PMB	Processor Memory Block
PNG	Portable Network Graphics
POSIX	Portable Operating System Interface
QR	Quick Response
REST	Representational State Transfer
RISC	Reduced Instruction Set Computing
RTEMS	Real-Time Executive for Multiprocessor Systems
RTP	Real-Time Transport Protocol
RTSP	Real-Time Streaming Protocol
SD	Secure Digital
SIFT	Scale-Invariant Feature Transform

SoC	System on Chip
SPI	Serial Peripheral Interface
SVM	Support Vector Machines
SW	Software
TCP/IP	Transmission Control Protocol / Internet Protocol
TI	Texas Instruments
VPU	Vision Processing Unit
WAV	Waveform Audio File
WEP	Wired Equivalent Privacy
WPA	WiFi Protected Access

Annexes

Annex 1

WifiFunctions

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	WifiConnectionState Struct Reference	5
3.1.1	Field Documentation	5
3.1.1.1	channel	5
3.1.1.2	mode	5
3.1.1.3	password	5
3.1.1.4	security	5
3.1.1.5	ssid_name	5
4	File Documentation	7
4.1	WifiFunctions.h File Reference	7
4.1.1	Detailed Description	9
4.1.2	Macro Definition Documentation	10
4.1.2.1	CLR_STATUS_BIT	10
4.1.2.2	DEFAULT_CHANNEL	10
4.1.2.3	DEFAULT_PASSWORD	10
4.1.2.4	DEFAULT_SECURITY	10
4.1.2.5	DEFAULT_SSID	10
4.1.2.6	GET_STATUS_BIT	10
4.1.2.7	IS_CONNECTED	10
4.1.2.8	IS_CONNECTION_FAILED	10
4.1.2.9	IS_IP_ACQUIRED	10
4.1.2.10	IS_IP_LEASED	10
4.1.2.11	IS_P2P_NEG_REQ_RECEIVED	10
4.1.2.12	IS_PING_DONE	10
4.1.2.13	IS_SMARTCONFIG_DONE	10

4.1.2.14	IS_SMARTCONFIG_STOPPED	10
4.1.2.15	IS_STA_CONNECTED	10
4.1.2.16	SET_STATUS_BIT	10
4.1.2.17	SL_STOP_TIMEOUT	10
4.1.3	Enumeration Type Documentation	10
4.1.3.1	ConnectionMode	10
4.1.3.2	e_AppStatusCodes	10
4.1.3.3	e_StatusBits	11
4.1.4	Function Documentation	11
4.1.4.1	configureSimpleLinkToDefaultState	11
4.1.4.2	connectToAP	11
4.1.4.3	disconnectFromAP	12
4.1.4.4	generateAP	12
4.1.4.5	generateAPFromDefaultProfile	12
4.1.4.6	generateAPFromProfile	13
4.1.4.7	generateAPFromProfileOnErrorDefault	14
4.1.4.8	generateAPSaveProfile	14
4.1.4.9	getHostIP	14
4.1.4.10	getLessSaturatedChannel	14
4.1.4.11	getOwnIP	15
4.1.4.12	getOwnMAC	15
4.1.4.13	getProfile	15
4.1.4.14	getStationIP	16
4.1.4.15	getWifiState	16
4.1.4.16	ping	16
4.1.4.17	pingToConnectedDevice	16
4.1.4.18	prettyIPv4	16
4.1.4.19	printPrettyIPv4_char	16
4.1.4.20	printPrettyIPv4_u32	16
4.1.4.21	printPrettyMAC	17
4.1.4.22	printWifiParams	17
4.1.4.23	removeProfiles	17
4.1.4.24	restoreProfile	17
4.1.4.25	saveCurrentProfile	17
4.1.4.26	saveProfile	17
4.1.4.27	scanWifi	18
4.1.4.28	scanWifiRestoreState	18
4.1.4.29	setOwnMAC	18
4.1.4.30	setPowerPolicy	18
4.1.4.31	setWifiState	19

4.1.4.32	setWlanPower	19
4.1.4.33	sleepWlanDevice	19
4.1.4.34	waitClients	19
4.1.4.35	wlanSetMode	19

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

WifiConnectionState	5
---	---

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

[WifiFunctions.h](#)

Library for common WiFi functions 7

Chapter 3

Data Structure Documentation

3.1 WifiConnectionState Struct Reference

```
#include <WifiFunctions.h>
```

Data Fields

- [_i8 ssid_name](#) [32]
- [_i8 password](#) [32]
- [_u8 security](#)
- [_u32 channel](#)
- [ConnectionMode mode](#)

3.1.1 Field Documentation

3.1.1.1 [_u32 channel](#)

3.1.1.2 [ConnectionMode mode](#)

3.1.1.3 [_i8 password](#)[32]

3.1.1.4 [_u8 security](#)

3.1.1.5 [_i8 ssid_name](#)[32]

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

- [WifiFunctions.h](#)

Chapter 4

File Documentation

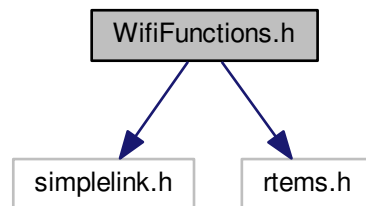
4.1 WifiFunctions.h File Reference

Library for common WiFi functions.

```
#include <simplelink.h>
```

```
#include <rtems.h>
```

Include dependency graph for WifiFunctions.h:



Data Structures

- struct [WifiConnectionState](#)

Macros

- `#define` [DEFAULT_SSID](#) "Myriad2Wifi"
- `#define` [DEFAULT_PASSWORD](#) "visilabap"
- `#define` [DEFAULT_SECURITY](#) `SL_SEC_TYPE_WPA_WPA2`
- `#define` [DEFAULT_CHANNEL](#) 8
- `#define` [SL_STOP_TIMEOUT](#) 0xFFFF
- `#define` [SET_STATUS_BIT](#)(status_variable, bit) `status_variable |= ((unsigned long)1<<(bit))`
- `#define` [CLR_STATUS_BIT](#)(status_variable, bit) `status_variable &= ~((unsigned long)1<<(bit))`
- `#define` [GET_STATUS_BIT](#)(status_variable, bit) `(0 != (status_variable & ((unsigned long)1<<(bit))))`
- `#define` [IS_PING_DONE](#)(status_variable) [GET_STATUS_BIT](#)(status_variable, [STATUS_BIT_PING_DONE](#))
- `#define` [IS_CONNECTED](#)(status_variable) [GET_STATUS_BIT](#)(status_variable, [STATUS_BIT_CONNECTION](#))

- `#define IS_STA_CONNECTED(status_variable) GET_STATUS_BIT(status_variable, STATUS_BIT_STA_CONNECTED)`
- `#define IS_IP_ACQUIRED(status_variable) GET_STATUS_BIT(status_variable, STATUS_BIT_IP_ACQUIRED)`
- `#define IS_IP_LEASED(status_variable) GET_STATUS_BIT(status_variable, STATUS_BIT_IP_LEASED)`
- `#define IS_CONNECTION_FAILED(status_variable) GET_STATUS_BIT(status_variable, STATUS_BIT_CONNECTION_FAILED)`
- `#define IS_P2P_NEG_REQ_RECEIVED(status_variable) GET_STATUS_BIT(status_variable, STATUS_BIT_P2P_NEG_REQ_RECEIVED)`
- `#define IS_SMARTCONFIG_DONE(status_variable) GET_STATUS_BIT(status_variable, STATUS_BIT_SMARTCONFIG_DONE)`
- `#define IS_SMARTCONFIG_STOPPED(status_variable) GET_STATUS_BIT(status_variable, STATUS_BIT_SMARTCONFIG_STOPPED)`

Enumerations

- `enum e_StatusBits {
STATUS_BIT_CONNECTION = 0, STATUS_BIT_STA_CONNECTED, STATUS_BIT_IP_ACQUIRED, STATUS_BIT_IP_LEASED,
STATUS_BIT_CONNECTION_FAILED, STATUS_BIT_P2P_NEG_REQ_RECEIVED, STATUS_BIT_SMARTCONFIG_DONE, STATUS_BIT_SMARTCONFIG_STOPPED }`
- `enum e_AppStatusCodes {
DEVICE_NOT_IN_STATION_MODE = -0x7D0, LAN_CONNECTION_FAILED = -0x7D0, SNTP_SEND_ERROR = DEVICE_NOT_IN_STATION_MODE - 1, SNTP_RECV_ERROR = SNTP_SEND_ERROR - 1,
SNTP_SERVER_RESPONSE_ERROR = SNTP_RECV_ERROR - 1, STATUS_BIT_PING_DONE = 31, STATUS_CODE_MAX = -0xBB8 }`
- `enum ConnectionMode { NOT_CONNECTED = 0, MODE_AP = 1, MODE_STATION = 2 }`

Functions

- `int generateAP (char *ssid_name, char *password, _u8 security, int channel)`
This function is used for generating an Access Point.
- `int generateAPSaveProfile (char *ssid_name, char *password, _u8 security, int channel)`
This function is used for generating an Access Point.
- `int generateAPFromProfile (int index)`
This function is used for generating an Access Point with the profile saved at a given index.
- `int generateAPFromDefaultProfile ()`
This function is used for generating an Access Point with default configuration defines on wifi_config.h.
- `int generateAPFromProfileOnErrorDefault (int index)`
This function is used for generating an Access Point with the profile saved at a given index. On error generate AP on default profile.
- `_i32 connectToAP (char *ssid_name, char *password, _u8 security, int timeout)`
This function is used for connecting to an Access Point.
- `_i32 wlanSetMode (int new_mode)`
This function is used for changing the operation mode of the device.
- `_i32 setWlanPower (_u8 power)`
This function is used for changing the operation power policy of the device.
- `_i32 setPowerPolicy (_u8 policy)`
This function is used for setting the power policy of the device.
- `_i32 sleepWlanDevice (int time)`
This function is used to put the device in low power consumption mode.
- `_i32 configureSimpleLinkToDefaultState ()`
This function configures the SimpleLink device in its default state. It:

- `_i32 pingToConnectedDevice` (int interval, int size, int request_timeout, int ping_attemp)

Pings to the last connected device.
- `_i32 ping` (int interval, int size, int request_timeout, int ping_attemp, _u32 ip)

Pings to an IP.
- void `waitClients` ()

Thread that waits until the first client connects.
- void `prettyIPv4` (_u32 val, _u8 *returnIP)

Gets IPv4 value into an array.
- void `printPrettyIPv4_u32` (_u32 ip)
- void `printPrettyIPv4_char` (_u8 *ip)
- void `printPrettyMAC` (_u8 *macAddressVal)
- void `printWifiParams` (WifiConnectionState state)
- int `disconnectFromAP` ()

Disconnects from a WLAN Access point.
- void `getOwnMAC` (_u8 *macAddressVal)
- void `setOwnMAC` (_u8 *macAddress)

Changes the device's MAC Address.
- _u32 `getOwnIP` ()
- _u32 `getHostIP` ()
- _u32 `getStationIP` ()
- int `scanWifi` (int scan_table_size, int channel, int timeout, SI_WlanNetworkEntry_t *netEntries)

Gets all available WiFi networks.
- int `scanWifiRestoreState` (int scan_table_size, int channel, int timeout, SI_WlanNetworkEntry_t *netEntries)

Gets all available WiFi networks and restore previous state.
- int `getLessSaturatedChannel` ()

Gets the number of less saturated channel.
- WifiConnectionState `getWifiState` ()

Gets the wifi connection state.
- void `setWifiState` (WifiConnectionState state)

Sets the wifi connection state.
- _i16 `saveCurrentProfile` ()

Saves the current profile.
- _i16 `saveProfile` (char *ssid_name, char *password, _u8 security, int channel)

Saves the given profile.
- _i16 `getProfile` (_i16 index, WifiConnectionState *profile)

Gets the stored profile at a given index.
- _i16 `restoreProfile` (int index)

Restores the stored profile at a given index.
- _i16 `removeProfiles` ()

Removes all the stored profiles.

4.1.1 Detailed Description

Library for common WiFi functions.

4.1.2 Macro Definition Documentation

4.1.2.1 `#define CLR_STATUS_BIT(status_variable, bit) status_variable &= ~((unsigned long)1<<(bit))`

4.1.2.2 `#define DEFAULT_CHANNEL 8`

4.1.2.3 `#define DEFAULT_PASSWORD "visilabap"`

4.1.2.4 `#define DEFAULT_SECURITY SL_SEC_TYPE_WPA_WPA2`

4.1.2.5 `#define DEFAULT_SSID "Myriad2Wifi"`

4.1.2.6 `#define GET_STATUS_BIT(status_variable, bit) (0 != (status_variable & ((unsigned long)1<<(bit))))`

4.1.2.7 `#define IS_CONNECTED(status_variable) GET_STATUS_BIT(status_variable, STATUS_BIT_CONNECTION)`

4.1.2.8 `#define IS_CONNECTION_FAILED(status_variable) GET_STATUS_BIT(status_variable, STATUS_BIT_CONNECTION_FAILED)`

4.1.2.9 `#define IS_IP_ACQUIRED(status_variable) GET_STATUS_BIT(status_variable, STATUS_BIT_IP_ACQUIRED)`

4.1.2.10 `#define IS_IP_LEASED(status_variable) GET_STATUS_BIT(status_variable, STATUS_BIT_IP_LEASED)`

4.1.2.11 `#define IS_P2P_NEG_REQ_RECEIVED(status_variable) GET_STATUS_BIT(status_variable, STATUS_BIT_P2P_NEG_REQ_RECEIVED)`

4.1.2.12 `#define IS_PING_DONE(status_variable) GET_STATUS_BIT(status_variable, STATUS_BIT_PING_DONE)`

4.1.2.13 `#define IS_SMARTCONFIG_DONE(status_variable) GET_STATUS_BIT(status_variable, STATUS_BIT_SMARTCONFIG_DONE)`

4.1.2.14 `#define IS_SMARTCONFIG_STOPPED(status_variable) GET_STATUS_BIT(status_variable, STATUS_BIT_SMARTCONFIG_STOPPED)`

4.1.2.15 `#define IS_STA_CONNECTED(status_variable) GET_STATUS_BIT(status_variable, STATUS_BIT_STA_CONNECTED)`

4.1.2.16 `#define SET_STATUS_BIT(status_variable, bit) status_variable |= ((unsigned long)1<<(bit))`

4.1.2.17 `#define SL_STOP_TIMEOUT 0xFFFF`

4.1.3 Enumeration Type Documentation

4.1.3.1 `enum ConnectionMode`

Enumerator

NOT_CONNECTED

MODE_AP

MODE_STATION

4.1.3.2 `enum e_AppStatusCodes`

Enumerator

DEVICE_NOT_IN_STATION_MODE

LAN_CONNECTION_FAILED
SNTP_SEND_ERROR
SNTP_RECV_ERROR
SNTP_SERVER_RESPONSE_ERROR
STATUS_BIT_PING_DONE
STATUS_CODE_MAX

4.1.3.3 enum e_StatusBits

Enumerator

STATUS_BIT_CONNECTION
STATUS_BIT_STA_CONNECTED
STATUS_BIT_IP_ACQUIRED
STATUS_BIT_IP_LEASED
STATUS_BIT_CONNECTION_FAILED
STATUS_BIT_P2P_NEG_REQ_RECEIVED
STATUS_BIT_SMARTCONFIG_DONE
STATUS_BIT_SMARTCONFIG_STOPPED

4.1.4 Function Documentation

4.1.4.1 _i32 configureSimpleLinkToDefaultState ()

This function configures the SimpleLink device in its default state. It:

- Sets the mode to STATION
- Configures connection policy to Auto and AutoSmartConfig
- Deletes all the stored profiles
- Enables DHCP
- Disables Scan policy
- Sets Tx power to maximum
- Sets power policy to normal
- Unregisters mDNS services
- Remove all filters

Parameters

<i>in</i>	<i>none</i>	
-----------	-------------	--

Returns

On success, zero is returned. On error, negative is returned

4.1.4.2 _i32 connectToAP (char * ssid_name, char * password, _u8 security, int timeout)

This function is used for connecting to an Access Point.

Parameters

in	<i>ssid_name</i>	is the name of the Access point
in	<i>password</i>	is the password of the Access Point
in	<i>security</i>	is the security of the WiFi network. It can be: <ul style="list-style-type: none"> • 0 or SL_SEC_TYPE_OPEN • 1 or SL_SEC_TYPE_WEP • 2 or SL_SEC_TYPE_WPA_WPA2
in	<i>timeout</i>	time in seconds trying connect

Returns

On success, zero is returned. On error, negative is returned

4.1.4.3 int disconnectFromAP ()

Disconnects from a WLAN Access point.

This function disconnects from the connected AP

Warning

If the WLAN disconnection fails, we will be stuck in this function forever.

4.1.4.4 int generateAP (char * ssid_name, char * password, _u8 security, int channel)

This function is used for generating an Access Point.

Parameters

in	<i>ssid_name</i>	is the name of the Access point
in	<i>password</i>	is the password of the Access Point
in	<i>security</i>	is the security of the WiFi network. It can be: <ul style="list-style-type: none"> • 0 or SL_SEC_TYPE_OPEN • 1 or SL_SEC_TYPE_WEP • 2 or SL_SEC_TYPE_WPA_WPA2
in	<i>channel</i>	is the channel where the network is generated

Returns

0 - if mode was set correctly

4.1.4.5 int generateAPFromDefaultProfile ()

This function is used for generating an Access Point with default configuration defines on wifi_config.h.

Returns

error if less than 0

4.1.4.6 int generateAPFromProfile (int *index*)

This function is used for generating an Access Point with the profile saved at a given index.

Parameters

<i>in</i>	<i>index</i>	where the profile is saved
-----------	--------------	----------------------------

Returns

error if less than 0

4.1.4.7 int generateAPFromProfileOnErrorDefault (int *index*)

This function is used for generating an Access Point with the profile saved at a given index. On error generate AP on default profile.

Parameters

<i>in</i>	<i>index</i>	where the profile is saved
-----------	--------------	----------------------------

Returns

error if less than 0

4.1.4.8 int generateAPSaveProfile (char * *ssid_name*, char * *password*, _u8 *security*, int *channel*)

This function is used for generating an Access Point.

Parameters

<i>in</i>	<i>ssid_name</i>	is the name of the Access point
<i>in</i>	<i>password</i>	is the password of the Access Point
<i>in</i>	<i>security</i>	is the security of the WiFi network. It can be: <ul style="list-style-type: none"> • 0 or SL_SEC_TYPE_OPEN • 1 or SL_SEC_TYPE_WEP • 2 or SL_SEC_TYPE_WPA_WPA2
<i>in</i>	<i>channel</i>	is the channel where the network is generated

Returns

index of the saved profile, less than 0 on error

4.1.4.9 _u32 getHostIP ()**4.1.4.10 int getLessSaturatedChannel ()**

Gets the number of less saturated channel.

Returns

the number of less saturated channel

4.1.4.11 `_u32 getOwnIP ()`

4.1.4.12 `void getOwnMAC (_u8 * macAddressVal)`

4.1.4.13 `_i16 getProfile (_i16 index, WifiConnectionState * profile)`

Gets the stored profile at a given index.

Parameters

<i>in</i>	<i>index</i>	where the profile is saved
-----------	--------------	----------------------------

Returns

less than 0 on error

4.1.4.14 _u32 getStationIP ()**4.1.4.15 WifiConnectionState getWifiState ()**

Gets the wifi connection state.

Returns

the wifi connection state

4.1.4.16 _i32 ping (int interval, int size, int request_timeout, int ping_attemp, _u32 ip)

Pings to an IP.

Parameters

<i>in</i>	<i>interval</i>	interval between ping commands
<i>in</i>	<i>size</i>	size of the ping package
<i>in</i>	<i>request_timeout</i>	timeout of the response
<i>in</i>	<i>ping_attemp</i>	times to retrying
<i>in</i>	<i>ip</i>	Address to ping

4.1.4.17 _i32 pingToConnectedDevice (int interval, int size, int request_timeout, int ping_attemp)

Pings to the last connected device.

Parameters

<i>in</i>	<i>interval</i>	interval between ping commands
<i>in</i>	<i>size</i>	size of the ping package
<i>in</i>	<i>request_timeout</i>	timeout of the response
<i>in</i>	<i>ping_attemp</i>	times to retrying

4.1.4.18 void prettyIPv4 (_u32 val, _u8 * returnIP)

Gets IPv4 value into an array.

Parameters

<i>in</i>	<i>val</i>	IP value code into a _u32 data type
<i>out</i>	<i>returnIP</i>	IP value code into an array of four numbers

4.1.4.19 void printPrettyIPv4_char (_u8 * ip)**4.1.4.20 void printPrettyIPv4_u32 (_u32 ip)**

4.1.4.21 void printPrettyMAC (_u8 * *macAddressVal*)

4.1.4.22 void printWifiParams (WifiConnectionState *state*)

4.1.4.23 _i16 removeProfiles ()

Removes all the stored profiles.

Returns

less than 0 on error

4.1.4.24 _i16 restoreProfile (int *index*)

Restores the stored profile at a given index.

Parameters

<i>in</i>	<i>index</i>	where the profile is saved
-----------	--------------	----------------------------

Returns

less than 0 on error

4.1.4.25 _i16 saveCurrentProfile ()

Saves the current profile.

Returns

index where the profile has been stored, less than 0 on error

4.1.4.26 _i16 saveProfile (char * *ssid_name*, char * *password*, _u8 *security*, int *channel*)

Saves the given profile.

Parameters

<i>in</i>	<i>ssid_name</i>	is the name of the Access point
<i>in</i>	<i>password</i>	is the password of the Access Point
<i>in</i>	<i>security</i>	is the security of the WiFi network. It can be: <ul style="list-style-type: none"> • 0 or SL_SEC_TYPE_OPEN • 1 or SL_SEC_TYPE_WEP • 2 or SL_SEC_TYPE_WPA_WPA2

in	<i>channel</i>	is the channel where the network is generated
----	----------------	---

Returns

index where the profile has been stored, less than 0 on error

Note

this function only support MODE_AP profiles

Warning

this function only support ONE profile

4.1.4.27 int scanWifi (int *scan_table_size*, int *channel*, int *timeout*, SI_WlanNetworkEntry_t * *netEntries*)

Gets all available WiFi networks.

Parameters

in	<i>scan_table_size</i>	the maximum wifi networks to return
in	<i>channel</i>	channel where scan. Have values between 1-11. Other value means all channels
in	<i>timeout</i>	timeout in seconds for wifi network scan
out	<i>netEntries</i>	array of found WiFi networks

Note

this function ends turning off the device, if you want to keep using it set on again

4.1.4.28 int scanWifiRestoreState (int *scan_table_size*, int *channel*, int *timeout*, SI_WlanNetworkEntry_t * *netEntries*)

Gets all available WiFi networks and restore previous state.

Parameters

in	<i>scan_table_size</i>	the maximum wifi networks to return
in	<i>channel</i>	channel where scan. Have values between 1-11. Other value means all channels
in	<i>timeout</i>	timeout in seconds for wifi network scan
out	<i>netEntries</i>	array of found WiFi networks

4.1.4.29 void setOwnMAC (_u8 * *macAddress*)

Changes the device's MAC Address.

Parameters

in	<i>macAddressVal</i>	new MAC address
----	----------------------	-----------------

Warning

After that the device can have a malfunction

4.1.4.30 _i32 setPowerPolicy (_u8 *policy*)

This function is used for setting the power policy of the device.

Parameters

<i>in</i>	<i>policy</i>	is the power policy to set. It can be: <ul style="list-style-type: none"> • SL_ALWAYS_ON_POLICY • SL_NORMAL_POLICY • SL_LOW_POWER_POLICY • SL_LONG_SLEEP_INTERVAL_POLICY
-----------	---------------	--

Returns

On success, zero is returned. On error, negative is returned

4.1.4.31 void setWifiState (WifiConnectionState state)

Sets the wifi connection state.

4.1.4.32 _i32 setWlanPower (_u8 power)

This function is used for changing the operation power policy of the device.

Parameters

<i>in</i>	<i>power</i>	is a number between 0-15, as dB offset from max power. 0 will set maximum power
-----------	--------------	---

Returns

On success, zero is returned. On error, negative is returned

4.1.4.33 _i32 sleepWlanDevice (int time)

This function is used to put the device in low power consumption mode.

Parameters

<i>in</i>	<i>time</i>	is a value between 100-2000 ms
-----------	-------------	--------------------------------

Returns

On success, zero is returned. On error, negative is returned

4.1.4.34 void waitClients ()

Thread that waits until the first client connects.

4.1.4.35 _i32 wlanSetMode (int new_mode)

This function is used for changing the operation mode of the device.

Parameters

<code>in</code>	<code>new_mode</code>	is the name of the Access point. It can be: <ul style="list-style-type: none">• <code>ROLE_STA</code>• <code>ROLE_AP</code>• <code>ROLE_P2P</code>
-----------------	-----------------------	--

Returns

`new_mode` value if it was successfully completed. Less than 0 on error

Annex 2

Camera

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	bitstring Struct Reference	5
3.1.1	Field Documentation	5
3.1.1.1	length	5
3.1.1.2	value	5
3.2	colorYCbCr Struct Reference	5
3.2.1	Field Documentation	5
3.2.1.1	Cb	5
3.2.1.2	Cr	5
3.2.1.3	Y	6
4	File Documentation	7
4.1	camera.h File Reference	7
4.1.1	Detailed Description	8
4.1.2	Macro Definition Documentation	8
4.1.2.1	CAM_BPP	8
4.1.2.2	CAM_FRAME_SIZE_BYTES	8
4.1.2.3	CAM_WINDOW_START_COLUMN	8
4.1.2.4	CAM_WINDOW_START_ROW	8
4.1.2.5	DDR_AREA	8
4.1.2.6	FIRST_INCOMING_BUF_ID	8
4.1.2.7	FIRST_OUTGOING_BUF_ID	8
4.1.2.8	MAX_USED_BUF	8
4.1.2.9	WINDOW_HEIGHT	8
4.1.2.10	WINDOW_WIDTH	8
4.1.3	Function Documentation	8

4.1.3.1	AllocateNextCamFrameBuf	8
4.1.3.2	getJpegFrame	9
4.1.3.3	init_camera	9
4.1.3.4	loop_camera	9
4.1.3.5	prepare_camera	9
4.1.3.6	prepareDriverData	9
4.1.3.7	reconfigure_camera	9
4.1.3.8	standby_camera	9
4.1.3.9	start_camera	10
4.1.3.10	stop_camera	10
4.1.3.11	take_snapshot	10
4.1.3.12	wakeup_camera	10
4.1.4	Variable Documentation	10
4.1.4.1	image	10
4.1.4.2	image_size_in_bytes	10
4.1.4.3	last_frame_buffer	10
4.2	jpeg_codec.h File Reference	11
4.2.1	Macro Definition Documentation	12
4.2.1.1	BYTE	12
4.2.1.2	Cb	12
4.2.1.3	Cr	12
4.2.1.4	DWORD	12
4.2.1.5	MAXBUFFERJPEG	12
4.2.1.6	SBYTE	12
4.2.1.7	SDWORD	12
4.2.1.8	SWORD	12
4.2.1.9	WORD	12
4.2.1.10	writebyte	12
4.2.1.11	writeword	12
4.2.1.12	Y	12
4.2.2	Function Documentation	12
4.2.2.1	convert2Jpeg	12

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

<code>bitstring</code>	5
<code>colorYCbCr</code>	5

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

camera.h		
	Management library for Camera	7
jpeg_codec.h	11

Chapter 3

Data Structure Documentation

3.1 bitstring Struct Reference

```
#include <jpeg_codec.h>
```

Data Fields

- [BYTE length](#)
- [WORD value](#)

3.1.1 Field Documentation

3.1.1.1 BYTE length

3.1.1.2 WORD value

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

- [jpeg_codec.h](#)

3.2 colorYCbCr Struct Reference

```
#include <jpeg_codec.h>
```

Data Fields

- [BYTE Y](#)
- [BYTE Cb](#)
- [BYTE Cr](#)

3.2.1 Field Documentation

3.2.1.1 BYTE Cb

3.2.1.2 BYTE Cr

3.2.1.3 BYTE Y

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

- [jpeg_codec.h](#)

Chapter 4

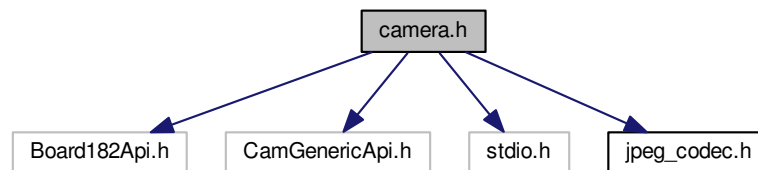
File Documentation

4.1 camera.h File Reference

Management library for Camera.

```
#include <Board182Api.h>
#include <CamGenericApi.h>
#include <stdio.h>
#include "jpeg_codec.h"
```

Include dependency graph for camera.h:



Macros

- #define MAX_USED_BUF 3
- #define FIRST_INCOMING_BUF_ID 1
- #define FIRST_OUTGOING_BUF_ID 0
- #define CAM_WINDOW_START_COLUMN 0
- #define CAM_WINDOW_START_ROW 8
- #define WINDOW_WIDTH 1920
- #define WINDOW_HEIGHT 1080
- #define CAM_BPP 1
- #define CAM_FRAME_SIZE_BYTES (WINDOW_WIDTH * WINDOW_HEIGHT * CAM_BPP)
- #define DDR_AREA __attribute__((section(".ddr.bss")))

Functions

- void prepareDriverData (void)
- frameBuffer * AllocateNextCamFrameBuf (void)

- int [loop_camera](#) ()
- int [stop_camera](#) ()
- int [init_camera](#) ()
- int [start_camera](#) ()
- int [standby_camera](#) ()
- int [wakeup_camera](#) ()
- void [prepare_camera](#) ()
- int [reconfigure_camera](#) ()
- void [take_snapshot](#) ()
- void [getJpegFrame](#) ()

Variables

- volatile unsigned char [image](#) [368640]
- int [image_size_in_bytes](#)
- colorYCbCr [last_frame_buffer](#) [122880]

4.1.1 Detailed Description

Management library for Camera.

4.1.2 Macro Definition Documentation

4.1.2.1 `#define CAM_BPP 1`

4.1.2.2 `#define CAM_FRAME_SIZE_BYTES (WINDOW_WIDTH * WINDOW_HEIGHT * CAM_BPP)`

4.1.2.3 `#define CAM_WINDOW_START_COLUMN 0`

4.1.2.4 `#define CAM_WINDOW_START_ROW 8`

4.1.2.5 `#define DDR_AREA __attribute__((section(".ddr.bss")))`

4.1.2.6 `#define FIRST_INCOMING_BUF_ID 1`

4.1.2.7 `#define FIRST_OUTGOING_BUF_ID 0`

4.1.2.8 `#define MAX_USED_BUF 3`

Application Includes

4.1.2.9 `#define WINDOW_HEIGHT 1080`

4.1.2.10 `#define WINDOW_WIDTH 1920`

4.1.3 Function Documentation

4.1.3.1 `frameBuffer* AllocateNextCamFrameBuf (void)`

This is the callback functions to be called on interrupts (and implicitly the buffers management behavior and notification behavior): `getFrame/getBlock/notification cbf`.

Returns

Pointer to the `frameBuffer` where the next frame should be stored.

4.1.3.2 void getJpegFrame ()

The frame is stored in 'last_frame_buffer' and compressed and stored in JPEG in 'image'. 'image_size_in_bytes' contains the size of the image buffer 'image'.

Returns

-1 if there was an error. 0 otherwise.

4.1.3.3 int init_camera ()

Performs all the camera configuration and initialises the camera.

Returns

-1 if there was an error. 0 otherwise.

4.1.3.4 int loop_camera ()

Performs all the camera configuration, initialises and starts the camera, and waits in a loop to continuously capture images.

Returns

-1 if there was an error. 0 otherwise.

4.1.3.5 void prepare_camera ()

Performs all the camera configuration, initialises and starts the camera, and puts the camera into 'hot standby' mode.

Returns

-1 if there was an error. 0 otherwise.

4.1.3.6 void prepareDriverData (void)

Initialises the data necessary to use the camera (CAM_B1).

4.1.3.7 int reconfigure_camera ()

Reconfigures camera interrupts. TODO: Complete the function.

Returns

-1 if there was an error. 0 otherwise.

4.1.3.8 int standby_camera ()

Puts the camera into 'hot standby' mode.

Returns

-1 if there was an error. 0 otherwise.

4.1.3.9 int start_camera ()

Starts the camera.

Returns

-1 if there was an error. 0 otherwise.

4.1.3.10 int stop_camera ()

Stops the camera.

Returns

-1 if there was an error. 0 otherwise.

4.1.3.11 void take_snapshot ()

Wakes up and puts in standby the camera in order to capture a frame. The frame is stored in 'last_frame_buffer' and compressed and stored in JPEG in 'image'. 'image_size_in_bytes' contains the size of the image buffer 'image'.

Returns

-1 if there was an error. 0 otherwise.

4.1.3.12 int wakeup_camera ()

Wakes up the camera from standby mode.

Returns

-1 if there was an error. 0 otherwise.

4.1.4 Variable Documentation**4.1.4.1 volatile unsigned char image[368640]**

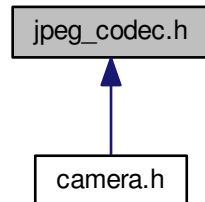
Where the jpegImage is stored. 368640 is the maximum needed by an image of 480x256x3

4.1.4.2 int image_size_in_bytes**4.1.4.3 colorYCbCr last_frame_buffer[122880]**

Where the yuv is stored 122880 is the maximum needed by an image of 480x256

4.2 jpeg_codec.h File Reference

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



Data Structures

- struct **APP0infotype**
- struct **SOF0infotype**
- struct **DQTinfotype**
- struct **DHTinfotype**
- struct **SOSinfotype**
- struct **colorYCbCr**
- struct **bitstring**

Macros

- #define **MAXBUFFERJPEG** 368640
- #define **BYTE** unsigned char
- #define **SBYTE** signed char
- #define **SWORD** signed short int
- #define **WORD** unsigned short int
- #define **DWORD** unsigned long int
- #define **SDWORD** signed long int
- #define **Y**(R, G, B) ((**BYTE**)(YRtab[(R)]+YGtab[(G)]+YBtab[(B)])>>16) - 128)
- #define **Cb**(R, G, B) ((**BYTE**)(CbRtab[(R)]+CbGtab[(G)]+CbBtab[(B)])>>16))
- #define **Cr**(R, G, B) ((**BYTE**)(CrRtab[(R)]+CrGtab[(G)]+CrBtab[(B)])>>16))
- #define **writebyte**(b) {image[image_size_in_bytes]=b;image_size_in_bytes++; }
- #define **writeword**(w) {image[image_size_in_bytes]=(w)/256;image_size_in_bytes++;image[image_size_in_↵_bytes]=(w)%256;image_size_in_bytes++;}

Functions

- void **convert2Jpeg** (**colorYCbCr** *imageBuffer, **WORD** Ximage_original, **WORD** Yimage_original)

4.2.1 Macro Definition Documentation

4.2.1.1 `#define BYTE unsigned char`

4.2.1.2 `#define Cb(R, G, B) ((BYTE)((CbRtab[(R)]+CbGtab[(G)]+CbBtab[(B)])>>16))`

4.2.1.3 `#define Cr(R, G, B) ((BYTE)((CrRtab[(R)]+CrGtab[(G)]+CrBtab[(B)])>>16))`

4.2.1.4 `#define DWORD unsigned long int`

4.2.1.5 `#define MAXBUFFERJPEG 368640`

4.2.1.6 `#define SBYTE signed char`

4.2.1.7 `#define SDWORD signed long int`

4.2.1.8 `#define SWORD signed short int`

4.2.1.9 `#define WORD unsigned short int`

4.2.1.10 `#define writebyte(b) {image[image_size_in_bytes]=b;image_size_in_bytes++; }`

4.2.1.11 `#define writeword(w) {image[image_size_in_bytes]=(w)/256;image_size_in_bytes++;image[image_↵
size_in_bytes]=(w)%256;image_size_in_bytes++;}`

4.2.1.12 `#define Y(R, G, B) ((BYTE)((YRtab[(R)]+YGtab[(G)]+YBtab[(B)])>>16) - 128)`

4.2.2 Function Documentation

4.2.2.1 `void convert2Jpeg (colorYCbCr * imageBuffer, WORD Ximage_original, WORD Yimage_original)`

Converts an image in YCbCr (YUV) to JPEG. The result is stored in 'image'. Variable 'image_size_in_bytes' contains the size of the image buffer 'image'.

Parameters

<i>*imageBuffer</i>	Pointer to the buffer where the image is stored.
<i>Ximage_original</i>	Width of the image.
<i>Yimage_original</i>	Height of the image.

Annex 3

RTSP

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	linkedList Struct Reference	5
3.1.1	Field Documentation	5
3.1.1.1	current	5
3.1.1.2	head	5
3.1.1.3	leaf	5
3.1.1.4	size	6
3.2	node Struct Reference	6
3.2.1	Field Documentation	6
3.2.1.1	next	6
3.2.1.2	prev	6
3.2.1.3	value	6
3.3	RtspClient Struct Reference	6
3.3.1	Detailed Description	7
3.3.2	Field Documentation	7
3.3.2.1	session	7
3.3.2.2	socketID	7
3.4	RtspServer Struct Reference	7
3.4.1	Detailed Description	7
3.4.2	Field Documentation	8
3.4.2.1	LocalAddr	8
3.4.2.2	nonBlocking	8
3.4.2.3	ServerSockID	8
3.5	RtspSession Struct Reference	8
3.5.1	Detailed Description	8
3.5.2	Field Documentation	8

3.5.2.1	clientSockID	8
3.5.2.2	messageN	8
3.5.2.3	sessionID	8
3.5.2.4	state	8
4	File Documentation	9
4.1	linkedList.h File Reference	9
4.1.1	Macro Definition Documentation	10
4.1.1.1	LL_VERSION	10
4.1.2	Typedef Documentation	10
4.1.2.1	item	10
4.1.3	Function Documentation	10
4.1.3.1	ll_clear	10
4.1.3.2	ll_create	10
4.1.3.3	ll_delete	10
4.1.3.4	ll_dump	10
4.1.3.5	ll_exists	10
4.1.3.6	ll_get_first	11
4.1.3.7	ll_get_index	11
4.1.3.8	ll_get_last	11
4.1.3.9	ll_get_next	11
4.1.3.10	ll_get_prev	11
4.1.3.11	ll_item_position	11
4.1.3.12	ll_pop_first	11
4.1.3.13	ll_pop_last	11
4.1.3.14	ll_print	11
4.1.3.15	ll_print_filter	11
4.1.3.16	ll_push_first	11
4.1.3.17	ll_push_last	11
4.1.3.18	ll_remove_item	11
4.1.3.19	ll_sort	11
4.2	rtsp.h File Reference	11
4.2.1	Macro Definition Documentation	13
4.2.1.1	BUF_SIZE	13
4.2.1.2	KJpegHeaderSize	13
4.2.1.3	KRtpHeaderSize	13
4.2.1.4	PACKAGE_LENGTHT	13
4.2.1.5	PORT_NUM	13
4.2.1.6	RTP_MARKER_END	13
4.2.1.7	RTP_MARKER_NO_END	13

4.2.2	Typedef Documentation	13
4.2.2.1	RtspClient	13
4.2.2.2	RtspServer	13
4.2.2.3	RtspSession	13
4.2.2.4	RtspState	13
4.2.3	Enumeration Type Documentation	13
4.2.3.1	RtspState	13
4.2.4	Function Documentation	14
4.2.4.1	closeSession	14
4.2.4.2	compare_RtspClients	15
4.2.4.3	generateSessionID	15
4.2.4.4	RTSP_step	15
4.2.4.5	runRTSPServer	15
4.2.4.6	sendImage	15
4.2.4.7	sendToAllClients	15
4.2.4.8	set_header	16
4.2.4.9	startRTSPServer	17

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

linkedList	5
node	6
RtspClient	6
RtspServer	7
RtspSession	8

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

linkedList.h	9
rtsp.h	11

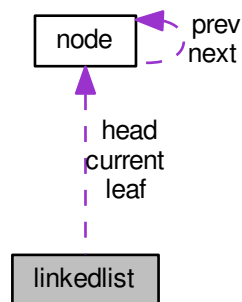
Chapter 3

Data Structure Documentation

3.1 linkedlist Struct Reference

```
#include <linkedlist.h>
```

Collaboration diagram for linkedlist:



Data Fields

- `item * head`
- `item * leaf`
- `item * current`
- unsigned int `size`

3.1.1 Field Documentation

3.1.1.1 `item* current`

3.1.1.2 `item* head`

3.1.1.3 `item* leaf`

3.1.1.4 unsigned int size

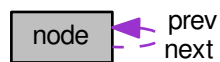
The documentation for this struct was generated from the following file:

- [linkedList.h](#)

3.2 node Struct Reference

```
#include <linkedList.h>
```

Collaboration diagram for node:



Data Fields

- void * [value](#)
- struct [node](#) * [prev](#)
- struct [node](#) * [next](#)

3.2.1 Field Documentation

3.2.1.1 struct node* next

3.2.1.2 struct node* prev

3.2.1.3 void* value

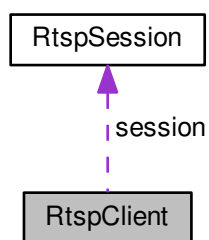
The documentation for this struct was generated from the following file:

- [linkedList.h](#)

3.3 RtspClient Struct Reference

```
#include <rtsp.h>
```

Collaboration diagram for RtspClient:



Data Fields

- [RtspSession](#) * [session](#)
- int [socketID](#)

3.3.1 Detailed Description

Information about the client.

3.3.2 Field Documentation

3.3.2.1 [RtspSession](#)* [session](#)

3.3.2.2 int [socketID](#)

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

- [rtsp.h](#)

3.4 RtspServer Struct Reference

```
#include <rtsp.h>
```

Data Fields

- [SISockAddrIn_t](#) [LocalAddr](#)
- [_i32](#) [ServerSockID](#)
- int [nonBlocking](#)

3.4.1 Detailed Description

Information about the server.

3.4.2 Field Documentation

3.4.2.1 SISockAddrIn_t LocalAddr

3.4.2.2 int nonBlocking

3.4.2.3 _i32 ServerSockID

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

- [rtsp.h](#)

3.5 RtspSession Struct Reference

```
#include <rtsp.h>
```

Data Fields

- [RtspState](#) state
- int [sessionID](#)
- int [clientSockID](#)
- int [messageN](#)

3.5.1 Detailed Description

Information about the session.

3.5.2 Field Documentation

3.5.2.1 int clientSockID

3.5.2.2 int messageN

3.5.2.3 int sessionID

3.5.2.4 RtspState state

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

- [rtsp.h](#)
-

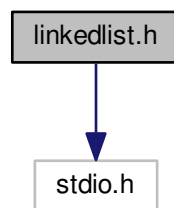
Chapter 4

File Documentation

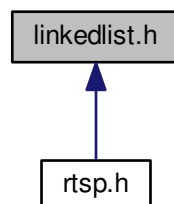
4.1 linkedlist.h File Reference

```
#include <stdio.h>
```

Include dependency graph for linkedlist.h:



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



Data Structures

- struct [node](#)

- struct [linkedList](#)

Macros

- #define [LL_VERSION](#) "1.0.0-b5"

Typedefs

- typedef struct [node](#) [item](#)

Functions

- [linkedList](#) * [ll_create](#) ()
- int [ll_push_first](#) ([linkedList](#) *, void *)
- int [ll_push_last](#) ([linkedList](#) *, void *)
- void * [ll_pop_first](#) ([linkedList](#) *)
- void * [ll_pop_last](#) ([linkedList](#) *)
- void * [ll_get_index](#) ([linkedList](#) *list, unsigned int index)
- void * [ll_get_first](#) ([linkedList](#) *)
- void * [ll_get_last](#) ([linkedList](#) *)
- void * [ll_get_next](#) ([linkedList](#) *)
- void * [ll_get_prev](#) ([linkedList](#) *)
- int [ll_exists](#) ([linkedList](#) *, void *, int(*) (void *, void *))
- int [ll_item_position](#) ([linkedList](#) *, void *, int(*) (void *, void *))
- int [ll_remove_item](#) ([linkedList](#) *, void *, int(*) (void *, void *))
- int [ll_sort](#) ([linkedList](#) *, int(*) (void *, void *))
- int [ll_clear](#) ([linkedList](#) *, void(*) (void *))
- int [ll_delete](#) ([linkedList](#) *, void(*) (void *))
- void [ll_print](#) ([linkedList](#) *, FILE *, void(*) (FILE *, void *))
- void [ll_print_filter](#) ([linkedList](#) *, FILE *, void(*) (FILE *, void *), int(*) (void *))
- void [ll_dump](#) ([linkedList](#) *, FILE *, void(*) (FILE *, void *))

4.1.1 Macro Definition Documentation

4.1.1.1 #define [LL_VERSION](#) "1.0.0-b5"

4.1.2 Typedef Documentation

4.1.2.1 typedef struct [node](#) [item](#)

4.1.3 Function Documentation

4.1.3.1 int [ll_clear](#) ([linkedList](#) *, void(*) (void *))

4.1.3.2 [linkedList](#)* [ll_create](#) ()

4.1.3.3 int [ll_delete](#) ([linkedList](#) *, void(*) (void *))

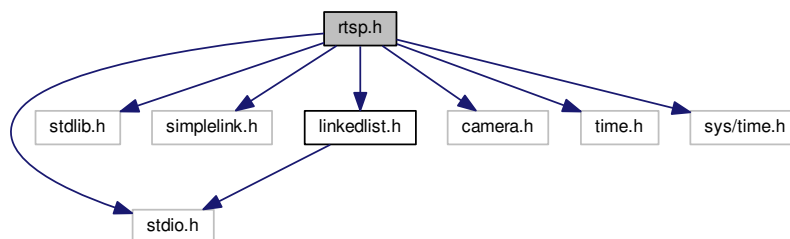
4.1.3.4 void [ll_dump](#) ([linkedList](#) *, FILE *, void(*) (FILE *, void *))

4.1.3.5 int [ll_exists](#) ([linkedList](#) *, void *, int(*) (void *, void *))

- 4.1.3.6 void* ll_get_first (linkedlist *)
- 4.1.3.7 void* ll_get_index (linkedlist * list, unsigned int index)
- 4.1.3.8 void* ll_get_last (linkedlist *)
- 4.1.3.9 void* ll_get_next (linkedlist *)
- 4.1.3.10 void* ll_get_prev (linkedlist *)
- 4.1.3.11 int ll_item_position (linkedlist * , void * , int (*)(void *, void *))
- 4.1.3.12 void* ll_pop_first (linkedlist *)
- 4.1.3.13 void* ll_pop_last (linkedlist *)
- 4.1.3.14 void ll_print (linkedlist * , FILE * , void (*)(FILE *, void *))
- 4.1.3.15 void ll_print_filter (linkedlist * , FILE * , void (*)(FILE *, void *) , int (*)(void *))
- 4.1.3.16 int ll_push_first (linkedlist * , void *)
- 4.1.3.17 int ll_push_last (linkedlist * , void *)
- 4.1.3.18 int ll_remove_item (linkedlist * , void * , int (*)(void *, void *))
- 4.1.3.19 int ll_sort (linkedlist * , int (*)(void *, void *))

4.2 rtsp.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "simplelink.h"
#include "linkedlist.h"
#include "camera.h"
#include "time.h"
#include "sys/time.h"
Include dependency graph for rtsp.h:
```



Data Structures

- struct [RtspSession](#)

- struct [RtspServer](#)
- struct [RtspClient](#)

Macros

- #define [PORT_NUM](#) 8554 /* Port where the server accepts connections*/
RTSP Server implementation.
- #define [BUF_SIZE](#) 1400 /* Size of the buffer where data is stored*/
- #define [KRtpHeaderSize](#) 12 /* Size of the RTP header*/
- #define [KJpegHeaderSize](#) 8 /* Size of the special JPEG payload header*/
- #define [PACKAGE_LENGTH](#) 1024 /* Maximun packet size*/
- #define [RTP_MARKER_END](#) 0x80 /* Last packet of the frame*/
- #define [RTP_MARKER_NO_END](#) 0 /* Not the last packet of the frame*/

Typedefs

- typedef enum [RtspState](#) [RtspState](#)
- typedef struct [RtspSession](#) [RtspSession](#)
- typedef struct [RtspServer](#) [RtspServer](#)
- typedef struct [RtspClient](#) [RtspClient](#)

Enumerations

- enum [RtspState](#) { [RTSP_STATE_STOP](#), [RTSP_STATE_PAUSE](#), [RTSP_STATE_PLAY](#), [RTSP_STATE_I↵NIT](#) }

Functions

- int [generateSessionID](#) ()
Generates a session ID for the RTSP server.
 - void [startRTSPServer](#) ([RtspServer](#) *server, _u8 *ownIP)
Initialises the data necessary for the RTSP server.
 - void [runRTSPServer](#) ([RtspServer](#) *server)
Starts accepting client connections and streaming.
 - void [sendToAllClients](#) ()
Sends the stream to several clients.
 - void [RTSP_step](#) ([RtspSession](#) *session)
Performs each step of the streaming configuring the message and sending the current image from the camera.
 - void [sendImage](#) (char *image, int width, int height, int length, int *sequenceNumber, int socketID, int timestamp) void [stopRTSPServer](#)([RtspServer](#) *server)
Sends the current image. If the final packet is larger than the maximun packet size, the image is fragmented and sent in several packages.
 - void [closeSession](#) ([RtspSession](#) *session)
Closes the RTSP session.
 - void [set_header](#) (char *RtpBuf, int RtpPacketSize, int m_SequenceNumber, int m_Timestamp, int offset, int rtpMarker, int width, int height)
Sets the packet header for the image to sent.
 - int [compare_RtspClients](#) (void *item1, void *item2)
Compares two clients.
-

4.2.1 Macro Definition Documentation

4.2.1.1 `#define BUF_SIZE 1400 /* Size of the buffer where data is stored*/`

4.2.1.2 `#define KJpegHeaderSize 8 /* Size of the special JPEG payload header*/`

4.2.1.3 `#define KRtpHeaderSize 12 /* Size of the RTP header*/`

4.2.1.4 `#define PACKAGE_LENGTH 1024 /* Maximum packet size*/`

4.2.1.5 `#define PORT_NUM 8554 /* Port where the server accepts connections*/`

RTSP Server implementation.

Implementation of the RTSP Server. This server supports multi-client streaming and JPEG-compressed video.

4.2.1.6 `#define RTP_MARKER_END 0x80 /* Last packet of the frame*/`

4.2.1.7 `#define RTP_MARKER_NO_END 0 /* Not the last packet of the frame*/`

4.2.2 Typedef Documentation

4.2.2.1 `typedef struct RtspClient RtspClient`

Information about the client.

4.2.2.2 `typedef struct RtspServer RtspServer`

Information about the server.

4.2.2.3 `typedef struct RtspSession RtspSession`

Information about the session.

4.2.2.4 `typedef enum RtspState RtspState`

Possible states of the server.

4.2.3 Enumeration Type Documentation

4.2.3.1 `enum RtspState`

Possible states of the server.

Enumerator

RTSP_STATE_STOP

RTSP_STATE_PAUSE

RTSP_STATE_PLAY

RTSP_STATE_INIT

4.2.4 Function Documentation

4.2.4.1 void closeSession (RtspSession * *session*)

Closes the RTSP session.

Parameters

in	<i>session</i>	The RtspSession struct that contains the session information.
----	----------------	---

4.2.4.2 int compare_RtspClients (void * *item1*, void * *item2*)

Compares two clients.

Parameters

in	<i>item1</i>	An RtspClient .
in	<i>item2</i>	An RtspClient .

Returns

0 - if the two clients are different

4.2.4.3 int generateSessionID ()

Generates a session ID for the RTSP server.

4.2.4.4 void RTSP_step ([RtspSession](#) * *session*)

Performs each step of the streaming configuring the message and sending the current image from the camera.

Parameters

in	<i>session</i>	The RtspSession struct that contains the session information.
----	----------------	---

4.2.4.5 void runRTSPServer ([RtspServer](#) * *server*)

Starts accepting client connections and streaming.

Parameters

in	<i>server</i>	The RtspServer struct that contains the server params.
----	---------------	--

4.2.4.6 void sendImage (char * *image*, int *width*, int *height*, int *length*, int * *sequenceNumber*, int *socketID*, int *timestamp*)

Sends the current image. If the final packet is larger than the maximum packet size, the image is fragmented and sent in several packages.

Closes the server socket and deletes all the clients.

Parameters

in	<i>server</i>	The RtspServer struct that contains the server params.
----	---------------	--

4.2.4.7 void sendToAllClients ()

Sends the stream to several clients.

4.2.4.8 void set_header (char * *RtpBuf*, int *RtpPacketSize*, int *m_SequenceNumber*, int *m_Timestamp*, int *offset*, int *rtpMarker*, int *width*, int *height*)

Sets the packet header for the image to sent.

Parameters

in	<i>RtpBuf</i>	Where the header is stored.
in	<i>RtpPacketSize</i>	Size of the final package (including header and body)
in	<i>m_SequenceNumber</i>	Sequence number of the current package.
in	<i>m_Timestamp</i>	Timestamp of the current frame.
in	<i>offset</i>	Is the offset of the current packet in the JPEG frame data.
in	<i>rtpMarker</i>	If the packet is the last one of the JPEG frame (0x80) or not (0).
in	<i>width</i>	Image width.
in	<i>height</i>	Image height.

4.2.4.9 void startRTSPServer (RtspServer * server, _u8 * ownIP)

Initialises the data necessary for the RTSP server.

Parameters

in	<i>server</i>	The RtspServer struct that contains the server params.
in	<i>ownIP</i>	The server IP address.

Annex 4

Buttons, Switches and LEDs

Contents

1	File Index	1
1.1	File List	1
2	File Documentation	3
2.1	LEDs.h File Reference	3
2.1.1	Detailed Description	4
2.1.2	Function Documentation	4
2.1.2.1	enableButtonsSwitchAndLeds()	4
2.1.2.2	readButton(u8 *buttonState)	4
2.1.2.3	readDip(u8 *dipState)	4
2.1.2.4	setLedModeConstant(u8 led, u8 state)	4
2.1.2.5	setLedModeContinuousPulsed(u8 led, enum Duration duration, enum DutyCycle dutyCycle)	4
2.1.2.6	setLedModePowerStatus(u8 led)	5
2.1.2.7	setLedModePulsedSequence(u8 led, enum SequenceLength sequenceLength, enum Duration duration, enum DutyCycle dutyCycle)	5
	Index	7

Chapter 1

File Index

1.1 File List

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

LEDs.h	Provides access to push buttons, DIP switches and LEDs	3
------------------------	--	---

Chapter 2

File Documentation

2.1 LEDs.h File Reference

Provides access to push buttons, DIP switches and LEDs.

```
#include <mv_types.h>
#include <DrvI2cDefines.h>
```

Enumerations

- enum **SequenceLength** { **SEQ_1_PULSE** = 0 << 4, **SEQ_2_PULSES** = 1 << 4, **SEQ_4_PULSES** = 2 << 4, **SEQ_7_PULSES** = 3 << 4 }
- enum **Duration** { **DUR_1000_MS** = 0 << 2, **DUR_250_MS** = 1 << 2, **DUR_125_MS** = 2 << 2, **DUR_62_5_MS** = 3 << 2 }
- enum **DutyCycle** { **DUTY_1_TO_1** = 0, **DUTY_1_TO_2** = 1, **DUTY_1_TO_3** = 2, **DUTY_1_TO_7** = 3 }

Functions

- I2CM_StatusType [enableButtonsSwitchAndLeds](#) ()
Enables manual control for LEDs, push buttons and the DIP switch. Needs to be called before any other function in this file can be used.
- I2CM_StatusType [readDip](#) (u8 *dipState)
Reads the current state of the first of the DIP switches.
- I2CM_StatusType [readButton](#) (u8 *buttonState)
Reads the current state of the push buttons.
- I2CM_StatusType [setLedModeConstant](#) (u8 led, u8 state)
Switches a LED on or off.
- I2CM_StatusType [setLedModeContinuousPulsed](#) (u8 led, enum Duration duration, enum DutyCycle dutyCycle)
Switches a LED into a mode in which it continuously emits light pulses.
- I2CM_StatusType [setLedModePulsedSequence](#) (u8 led, enum SequenceLength sequenceLength, enum Duration duration, enum DutyCycle dutyCycle)
Switches a LED into a mode in which it emits a sequence of light pulses of a defined length.
- I2CM_StatusType [setLedModePowerStatus](#) (u8 led)
Switches a LED into power state status mode.

2.1.1 Detailed Description

Provides access to push buttons, DIP switches and LEDs.

2.1.2 Function Documentation

2.1.2.1 I2CM_StatusType enableButtonsSwitchAndLeds ()

Enables manual control for LEDs, push buttons and the DIP switch. Needs to be called before any other function in this file can be used.

Returns

An error code in case of failure, I2CM_STAT_OK otherwise.

2.1.2.2 I2CM_StatusType readButton (u8 * *buttonState*)

Reads the current state of the push buttons.

Parameters

<i>buttonState</i>	The state of push buttons is written to this pointer. In case of an error the value is not changed. The value written can be one of the following: 0 - No button pushed. 1 - Button #1 pushed. 2 - Button #2 pushed. 3 - Both buttons pushed.
--------------------	---

Returns

An error code in case of failure, I2CM_STAT_OK otherwise.

2.1.2.3 I2CM_StatusType readDip (u8 * *dipState*)

Reads the current state of the first of the DIP switches.

Parameters

<i>dipState</i>	The state of DIP switch #1 (0 or 1) is written to this pointer. In case of an error the value is not changed.
-----------------	---

Returns

An error code in case of failure, I2CM_STAT_OK otherwise.

2.1.2.4 I2CM_StatusType setLedModeConstant (u8 *led*, u8 *state*)

Switches a LED on or off.

Parameters

<i>led</i>	The ID of the LED to switch. Either 0 or 1.
<i>state</i>	The desired state of the LED (0: off, 1: on).

Returns

An error code in case of failure, I2CM_STAT_OK otherwise.

2.1.2.5 I2CM_StatusType setLedModeContinuousPulsed (u8 *led*, enum Duration *duration*, enum DutyCycle *dutyCycle*)

Switches a LED into a mode in which it continuously emits light pulses.

Parameters

<i>led</i>	The ID of the LED to switch. Either 0 or 1.
<i>duration</i>	The 'on' time of the LED.
<i>dutyCycle</i>	The on/off ratio of the LED.

Returns

An error code in case of failure, I2CM_STAT_OK otherwise.

2.1.2.6 I2CM_StatusType setLedModePowerStatus (u8 led)

Switches a LED into power state status mode.

This mode can indicate one of the following 4 states as follows:

- power sequence failure: 4 pulses, 1000ms 'on' time, on/off ratio: 1:1
- PVDD low: continuous pulsed, 250ms 'on' time, on/off ratio: 1:3
- 'on' state: constant on
- 'sleep' state: continuous pulsed, 250ms 'on' time, on/off ratio: 1:7

Parameters

<i>led</i>	The ID of the LED to switch. Either 0 or 1.
------------	---

Returns

An error code in case of failure, I2CM_STAT_OK otherwise.

2.1.2.7 I2CM_StatusType setLedModePulsedSequence (u8 led, enum SequenceLength *sequenceLength*, enum Duration *duration*, enum DutyCycle *dutyCycle*)

Switches a LED into a mode in which it emits a sequence of light pulses of a defined length.

Parameters

<i>led</i>	The ID of the LED to switch. Either 0 or 1.
<i>sequenceLength</i>	Determines the length of the pulse sequence.
<i>duration</i>	The value determining the 'on' time of the LED.
<i>dutyCycle</i>	Determines the on/off ratio of the LED.

Returns

An error code in case of failure, I2CM_STAT_OK otherwise.

Index

enableButtonsSwitchAndLeds

LEDs.h, [4](#)

LEDs.h, [3](#)

enableButtonsSwitchAndLeds, [4](#)

readButton, [4](#)

readDip, [4](#)

setLedModeConstant, [4](#)

setLedModeContinuousPulsed, [4](#)

setLedModePowerStatus, [5](#)

setLedModePulsedSequence, [5](#)

readButton

LEDs.h, [4](#)

readDip

LEDs.h, [4](#)

setLedModeConstant

LEDs.h, [4](#)

setLedModeContinuousPulsed

LEDs.h, [4](#)

setLedModePowerStatus

LEDs.h, [5](#)

setLedModePulsedSequence

LEDs.h, [5](#)

Annex 5

Crypto

Contents

1	File Index	1
1.1	File List	1
2	File Documentation	3
2.1	Crypto.h File Reference	3
2.1.1	Detailed Description	3
2.1.2	Typedef Documentation	3
2.1.2.1	nonceGeneratorFunction	3
2.1.3	Function Documentation	4
2.1.3.1	CryptoDecrypt	4
2.1.3.2	CryptoEncrypt	5
2.1.3.3	CryptoSetKey	5
2.1.3.4	CryptoSetNonceGenerator	5
	Index	6

Chapter 1

File Index

1.1 File List

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

Crypto.h	This file provides functions for an AES 128bit Stream Cipher	3
--------------------------	--	---

Chapter 2

File Documentation

2.1 Crypto.h File Reference

This file provides functions for an AES 128bit Stream Cipher.

```
#include <mv_types.h>
```

Typedefs

- typedef u8 *(* nonceGeneratorFunction)(u32 state)

Defines an function pointer type.

Functions

- void [CryptoSetKey](#) (u8 key128[16])
Set a specific 128 bit key. (optional)
- void [CryptoSetNonceGenerator](#) ([nonceGeneratorFunction](#) generator)
Set a specific nonce generator. (optional)
- u32 [CryptoEncrypt](#) (const u8 *unencryptedBuffer, const u32 unencryptedBufferSize, const u32 offset, u8 *encryptedBuffer)
AES 128 bit Stream Cipher Encryption (CTR)
- u32 [CryptoDecrypt](#) (const u8 *encryptedBuffer, const u32 encryptedBufferSize, const u32 offset, u8 *decryptedBuffer)
AES 128 bit Stream Cipher Decryption (CTR)

2.1.1 Detailed Description

This file provides functions for an AES 128bit Stream Cipher.

2.1.2 Typedef Documentation

2.1.2.1 typedef u8 *(* nonceGeneratorFunction)(u32 state)

Defines an function pointer type.

2.1.3 Function Documentation

2.1.3.1 u32 CryptoDecrypt (const u8 * *encryptedBuffer*, const u32 *encryptedBufferSize*, const u32 *offset*, u8 * *decryptedBuffer*)

AES 128 bit Stream Cipher Decryption (CTR)

Parameters

in	<i>encryptedBuffer</i>	A pointer to an encrypted buffer
in	<i>encryptedBuffer-Size</i>	The number of bytes
in	<i>offset</i>	Number of bytes to offset from origin (default: 0)
out	<i>decryptedBuffer</i>	A pointer to a decrypted buffer

Returns

If decryption is completed, the function returns the number of bytes. Otherwise, a zero value is returned.

2.1.3.2 `u32 CryptoEncrypt (const u8 * unencryptedBuffer, const u32 unencryptedBufferSize, const u32 offset, u8 * encryptedBuffer)`

AES 128 bit Stream Cipher Encryption (CTR)

Parameters

in	<i>unencrypted-Buffer</i>	A pointer to an unencrypted buffer
in	<i>unencrypted-BufferSize</i>	The number of bytes
in	<i>offset</i>	Number of bytes to offset from origin (default: 0)
out	<i>encryptedBuffer</i>	A pointer to an encrypted buffer

Returns

If encryption is completed, the function returns the number of bytes. Otherwise, a zero value is returned.

2.1.3.3 `void CryptoSetKey (u8 key128[16])`

Set a specific 128 bit key. (optional)

2.1.3.4 `void CryptoSetNonceGenerator (nonceGeneratorFunction generator)`

Set a specific nonce generator. (optional)

Index

- Crypto.h, [3](#)
 - CryptoDecrypt, [4](#)
 - CryptoEncrypt, [5](#)
 - CryptoSetKey, [5](#)
 - CryptoSetNonceGenerator, [5](#)
 - nonceGeneratorFunction, [3](#)
- CryptoDecrypt
 - Crypto.h, [4](#)
- CryptoEncrypt
 - Crypto.h, [5](#)
- CryptoSetKey
 - Crypto.h, [5](#)
- CryptoSetNonceGenerator
 - Crypto.h, [5](#)
- nonceGeneratorFunction
 - Crypto.h, [3](#)

Annex 6

SDCardIO

Contents

1	File Index	1
1.1	File List	1
2	File Documentation	3
2.1	SDCardIO.h File Reference	3
2.1.1	Function Documentation	5
2.1.1.1	SDCardDirCd	5
2.1.1.2	SDCardDirCdUp	5
2.1.1.3	SDCardDirClose	5
2.1.1.4	SDCardDirCountDirectories	5
2.1.1.5	SDCardDirCountDirectoriesWithDir	5
2.1.1.6	SDCardDirCountFiles	6
2.1.1.7	SDCardDirCountFilesWithDir	6
2.1.1.8	SDCardDirExists	6
2.1.1.9	SDCardDirGetDirectoryName	6
2.1.1.10	SDCardDirGetFilename	6
2.1.1.11	SDCardDirGetName	6
2.1.1.12	SDCardDirGetPath	6
2.1.1.13	SDCardDirMakeDirectory	6
2.1.1.14	SDCardDirMakeDirectoryWithDir	6
2.1.1.15	SDCardDirOpen	6
2.1.1.16	SDCardDirOpenWithPath	6
2.1.1.17	SDCardDirRemoveDirectory	6
2.1.1.18	SDCardDirRemoveDirectoryRecursive	7
2.1.1.19	SDCardDirRemoveDirectoryRecursiveWithDir	7
2.1.1.20	SDCardDirRemoveDirectoryWithDir	7
2.1.1.21	SDCardEntryStatusDestroy	7
2.1.1.22	SDCardExists	7
2.1.1.23	SDCardFileClose	7
2.1.1.24	SDCardFileCopy	7
2.1.1.25	SDCardFileFlush	7

2.1.1.26	SDCardFileGetPostion	7
2.1.1.27	SDCardFileGetSize	7
2.1.1.28	SDCardFileOpen	8
2.1.1.29	SDCardFileOpenWithDir	9
2.1.1.30	SDCardFilePeek	9
2.1.1.31	SDCardFileRead	9
2.1.1.32	SDCardFileRemove	10
2.1.1.33	SDCardFileRemoveWithDir	10
2.1.1.34	SDCardFileRename	10
2.1.1.35	SDCardFileSetPosition	10
2.1.1.36	SDCardFileWrite	10
2.1.1.37	SDCardIsMounted	10
2.1.1.38	SDCardLs	10
2.1.1.39	SDCardLsWithDir	10
2.1.1.40	SDCardMount	10
2.1.1.41	SDCardRemoveAllFromDirectory	11
2.1.1.42	SDCardRemoveAllFromDirectoryWithDir	11
2.1.1.43	SDCardUnmount	11

Chapter 1

File Index

1.1 File List

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

SDCardIO.h	3
--------------------------------------	---

Chapter 2

File Documentation

2.1 SDCardIO.h File Reference

```
#include <stdbool.h>
#include <dirent.h>
#include <stdio.h>
#include <time.h>
#include <sys/syslimits.h>
#include <mv_types.h>
```

Functions

- bool [SDCardMount](#) (void)
Try to mount a SD Card. Returns true on success; otherwise returns false.
- bool [SDCardUnmount](#) (void)
Try to unmount a SD Card. Returns true on success; otherwise returns false.
- bool [SDCardIsMounted](#) (void)
Returns true if the SD Card is mounted; otherwise returns false.
- bool [SDCardExists](#) (const char *name)
Returns true if the directory/file called name exists; otherwise returns false.
- SDCardDir * [SDCardDirOpen](#) (void)
Opens the root directory of the SD Card.
- SDCardDir * [SDCardDirOpenWithPath](#) (const char *path)
Returns a valid pointer if successful; otherwise returns NULL.
- bool [SDCardDirClose](#) (SDCardDir **dirHandler)
Close the SDCardDir handler.
- bool [SDCardDirCd](#) (SDCardDir *dirHandler, const char *dirName)
Changes directory to dirName. Return true if the directory exists; otherwise returns false.
- bool [SDCardDirCdUp](#) (SDCardDir *dirHandler)
Changes directory by moving one directory up. Returns true if the directory exists; otherwise returns false.
- u32 [SDCardDirCountFiles](#) (const char *path)
Returns the total number of files in a directory specified by path.
- u32 [SDCardDirCountFilesWithDir](#) (const SDCardDir *dirHandler)
Returns the total number of files in the current directory.
- const char * [SDCardDirGetFilename](#) (const SDCardDir *dirHandler, const u32 index)
Return a filename if the index is valid; otherwise returns null.
- u32 [SDCardDirCountDirectories](#) (const char *path)

- Returns the total number of folders in a directory specified by path.*

 - u32 [SDCardDirCountDirectoriesWithDir](#) (const SDCardDir *dirHandler)

Returns the total number of folders in the directory.
 - const char * [SDCardDirGetDirectoryName](#) (const SDCardDir *dirHandler, const u32 index)
 - bool [SDCardDirMakeDirectory](#) (const char *path)

Create a sub-directory. Returns true on success; otherwise returns false.
 - bool [SDCardDirMakeDirectoryWithDir](#) (const SDCardDir *dirHandler, const char *dirName)

Create a sub-directory called dirName. Returns true on success; otherwise returns false.
 - bool [SDCardDirRemoveDirectory](#) (const char *path)

Removes the directory specified by path. The directory must be empty to succeed. Return true if successful; otherwise returns false.
 - bool [SDCardDirRemoveDirectoryWithDir](#) (const SDCardDir *dirHandler, const char *dirName)

Removes the directory specified by dirName. The directory must be empty to succeed. Return true if successful; otherwise returns false.
 - bool [SDCardDirRemoveDirectoryRecursive](#) (const char *path)

Removes the directory specified by path. Return true if successful; otherwise returns false.
 - bool [SDCardDirRemoveDirectoryRecursiveWithDir](#) (SDCardDir *dirHandler, const char *dirName)

Removes the directory specified by dirName. Return true if successful; otherwise returns false.
 - const char * [SDCardDirGetName](#) (const SDCardDir *dirHandler)

Returns the name of the directory.
 - const char * [SDCardDirGetPath](#) (const SDCardDir *dirHandler)

Returns the path. The returned path is absolute.
 - bool [SDCardDirExists](#) (const SDCardDir *dirHandler, const char *name)

Returns true if the directory/file called name exists; otherwise returns false.
 - bool [SDCardRemoveAllFromDirectory](#) (const char *path)

Delete all Elements inside the directory.
 - bool [SDCardRemoveAllFromDirectoryWithDir](#) (SDCardDir *dirHandler)

Delete all Elements inside the directory.
 - SDCardEntryStatus * [SDCardLs](#) (const char *path, int *size)

Returns an array of SDCardEntryStatus objects for all the files and directories in the directory. Returns a NULL if the directory is unreadable, does not exist or is empty.
 - SDCardEntryStatus * [SDCardLsWithDir](#) (const SDCardDir *dirHandler, int *size)

Returns an array of SDCardEntryStatus objects for all the files and directories in the directory. Returns a NULL if the directory is unreadable, does not exist or is empty.
 - void [SDCardEntryStatusDestroy](#) (SDCardEntryStatus **entries)

Frees an array of SDCardEntryStatus objects.
 - SDCardFile * [SDCardFileOpen](#) (const char *filename, const char *mode, bool enableEncryption)

Returns a valid pointer if successful; otherwise returns NULL.
 - SDCardFile * [SDCardFileOpenWithDir](#) (const SDCardDir *dirHandler, const char *filename, const char *mode, bool enableEncryption)

Returns a valid pointer if successful; otherwise returns NULL.
 - bool [SDCardFileClose](#) (SDCardFile **fileHandler)

Close the SDCardFile handler.
 - bool [SDCardFileRemove](#) (const char *filename)

Removes the file specified by the 'filename'(absolute path) given. Returns true if successful; otherwise return false.
 - bool [SDCardFileRemoveWithDir](#) (const SDCardDir *dirHandler, const char *filename)

Removes the file specified by the 'filename' given. Returns true if successful; otherwise return false.
 - bool [SDCardFileRename](#) (const char *currentFilename, const char *newFilename)

Rename the file 'currentFilename' to 'newFilename'. Return true if successful; otherwise returns false. If a file with the name 'newFilename' already exists, [SDCardFileRename\(\)](#) returns false.
 - bool [SDCardFileCopy](#) (const char *sourceFilename, const char *destFilename)
-

Copy the file 'sourceFilename' to 'destFilename'. Return true if successful; otherwise returns false. If a file with the name 'destFilename' already exists, [SDCardFileCopy\(\)](#) returns false.

- u32 [SDCardFileGetSize](#) (SDCardFile *fileHandler)
Returns the size of the file.
- u32 [SDCardFileWrite](#) (SDCardFile *fileHandler, const u8 *writeBuffer, const u32 writeBufferSize)
Writes at most 'writeBufferSize' bytes of data from 'writeBuffer' to the file. Returns the number of bytes that were actually written, or 0 if an error occurred.
- u32 [SDCardFileRead](#) (SDCardFile *fileHandler, u8 *readBuffer, const u32 readBufferSize)
Reads at most 'readBufferSize' bytes from the file into 'readBuffer', and returns the number of bytes read. If an error occurs, such as when attempting to read from a file opened in WriteOnly mode, this function returns 0.
- u32 [SDCardFilePeek](#) (SDCardFile *fileHandler, u8 *readBuffer, const u32 readBufferSize)
Reads at most 'readBufferSize' bytes from the file into 'readBuffer', without side effects (i.e., if you call [SDCardFileRead\(\)](#) after [SDCardFilePeek\(\)](#), you will get the same data). Returns the number of bytes read. If an error occurs, such as when attempting to peek a file opened in WriteOnly mode, this function returns 0.
- bool [SDCardFileSetPosition](#) (SDCardFile *fileHandler, const u32 pos)
Sets the current position from the fileHandler to 'pos'. Returns true on success, or false if an error occurred.
- u32 [SDCardFileGetPosition](#) (SDCardFile *fileHandler)
Returns the current position from the fileHandler.
- bool [SDCardFileFlush](#) (SDCardFile *fileHandler)
Flushes any buffered data to the file. Returns true if successful; otherwise returns false.

2.1.1 Function Documentation

2.1.1.1 bool SDCardDirCd (SDCardDir * dirHandler, const char * dirName)

Changes directory to dirName. Return true if the directory exists; otherwise returns false.

2.1.1.2 bool SDCardDirCdUp (SDCardDir * dirHandler)

Changes directory by moving one directory up. Returns true if the directory exists; otherwise returns false.

2.1.1.3 bool SDCardDirClose (SDCardDir ** dirHandler)

Close the SDCardDir handler.

Parameters

out	<i>dirHandler</i>	pointer to a SDCardDir object
-----	-------------------	-------------------------------

Returns

If the SDCardDir-handler is successfully closed, the function returns true. Otherwise, false is returned.

2.1.1.4 u32 SDCardDirCountDirectories (const char * path)

Returns the total number of folders in a directory specified by path.

2.1.1.5 u32 SDCardDirCountDirectoriesWithDir (const SDCardDir * dirHandler)

Returns the total number of folders in the directory.

2.1.1.6 u32 SDCardDirCountFiles (const char * *path*)

Returns the total number of files in a directory specified by path.

2.1.1.7 u32 SDCardDirCountFilesWithDir (const SDCardDir * *dirHandler*)

Returns the total number of files in the current directory.

2.1.1.8 bool SDCardDirExists (const SDCardDir * *dirHandler*, const char * *name*)

Returns true if the directory/file called name exists; otherwise returns false.

2.1.1.9 const char* SDCardDirGetDirectoryName (const SDCardDir * *dirHandler*, const u32 *index*)

Return a directory name if the index is valid; otherwise returns null.

2.1.1.10 const char* SDCardDirGetFilename (const SDCardDir * *dirHandler*, const u32 *index*)

Return a filename if the index is valid; otherwise returns null.

2.1.1.11 const char* SDCardDirGetName (const SDCardDir * *dirHandler*)

Returns the name of the directory.

2.1.1.12 const char* SDCardDirGetPath (const SDCardDir * *dirHandler*)

Returns the path. The returned path is absolute.

2.1.1.13 bool SDCardDirMakeDirectory (const char * *path*)

Create a sub-directory. Returns true on success; otherwise returns false.

2.1.1.14 bool SDCardDirMakeDirectoryWithDir (const SDCardDir * *dirHandler*, const char * *dirName*)

Create a sub-directory called dirName. Returns true on success; otherwise returns false.

2.1.1.15 SDCardDir* SDCardDirOpen (void)

Opens the root directory of the SD Card.

2.1.1.16 SDCardDir* SDCardDirOpenWithPath (const char * *path*)

Returns a valid pointer if successful; otherwise returns NULL.

2.1.1.17 bool SDCardDirRemoveDirectory (const char * *path*)

Removes the directory specified by path. The directory must be empty to succeed. Return true if successful; otherwise returns false.

2.1.1.18 bool SDCardDirRemoveDirectoryRecursive (const char * *path*)

Removes the directory specified by path. Return true if successful; otherwise returns false.

2.1.1.19 bool SDCardDirRemoveDirectoryRecursiveWithDir (SDCardDir * *dirHandler*, const char * *dirName*)

Removes the directory specified by dirName. Return true if successful; otherwise returns false.

2.1.1.20 bool SDCardDirRemoveDirectoryWithDir (const SDCardDir * *dirHandler*, const char * *dirName*)

Removes the directory specified by dirName. The directory must be empty to succeed. Return true if successful; otherwise returns false.

2.1.1.21 void SDCardEntryStatusDestroy (SDCardEntryStatus ** *entries*)

Frees an array of SDCardEntryStatus objects.

2.1.1.22 bool SDCardExists (const char * *name*)

Returns true if the directory/file called name exists; otherwise returns false.

2.1.1.23 bool SDCardFileClose (SDCardFile ** *fileHandler*)

Close the SDCardFile handler.

Parameters

out	<i>fileHandler</i>	pointer to a SDCardFile object
-----	--------------------	--------------------------------

Returns

If the SDCardFile-handler is successfully closed, the function returns true. Otherwise, false is returned.

2.1.1.24 bool SDCardFileCopy (const char * *sourceFilename*, const char * *destFilename*)

Copy the file 'sourceFilename' to 'destFilename'. Return true if successful; otherwise returns false. If a file with the name 'destFilename' already exists, [SDCardFileCopy\(\)](#) returns false.

2.1.1.25 bool SDCardFileFlush (SDCardFile * *fileHandler*)

Flushes any buffered data to the file. Returns true if successful; otherwise returns false.

2.1.1.26 u32 SDCardFileGetPostion (SDCardFile * *fileHandler*)

Returns the current position from the fileHandler.

2.1.1.27 u32 SDCardFileGetSize (SDCardFile * *fileHandler*)

Returns the size of the file.

2.1.1.28 SDCardFile* SDCardFileOpen (const char * *filename*, const char * *mode*, bool *enableEncryption*)

Returns a valid pointer if successful; otherwise returns NULL.

Parameters

in	<i>filename</i>	represent the file with the given name (absolute path)
in	<i>mode</i>	supports read ("r") and write ("w") access. For more details

See Also

[fopen](#)

Parameters

in	<i>enable-Encryption</i>	encrypt and decrypt automatically if true
----	--------------------------	---

Returns

If the file is successfully opened, the function returns a pointer to a SDCardFile object. Otherwise, a null pointer is returned.

2.1.1.29 SDCardFile* SDCardFileOpenWithDir (const SDCardDir * *dirHandler*, const char * *filename*, const char * *mode*, bool *enableEncryption*)

Returns a valid pointer if successful; otherwise returns NULL.

Parameters

in	<i>dirHandler</i>	holds the current path
in	<i>filename</i>	represent the file with the given name
in	<i>mode</i>	supports read ("r") and write ("w") access. For more details

See Also

[fopen](#)

Parameters

in	<i>enable-Encryption</i>	encrypt and decrypt automatically if true
----	--------------------------	---

Returns

If the file is successfully opened, the function returns a pointer to a SDCardFile object. Otherwise, a null pointer is returned.

2.1.1.30 u32 SDCardFilePeek (SDCardFile * *fileHandler*, u8 * *readBuffer*, const u32 *readBufferSize*)

Reads at most 'readBufferSize' bytes from the file into 'readBuffer', without side effects (i.e., if you call [SDCardFileRead\(\)](#) after [SDCardFilePeek\(\)](#), you will get the same data). Returns the number of bytes read. If an error occurs, such as when attempting to peek a file opened in WriteOnly mode, this function returns 0.

2.1.1.31 u32 SDCardFileRead (SDCardFile * *fileHandler*, u8 * *readBuffer*, const u32 *readBufferSize*)

Reads at most 'readBufferSize' bytes from the file into 'readBuffer', and returns the number of bytes read. If an error occurs, such as when attempting to read from a file opened in WriteOnly mode, this function returns 0.

Parameters

in	<i>fileHandler</i>	pointer to a SDCardFile object
out	<i>readBuffer</i>	a data array
in	<i>readBufferSize</i>	number of bytes

2.1.1.32 bool SDCardFileRemove (const char * filename)

Removes the file specified by the 'filename'(absolute path) given. Returns true if successful; otherwise return false.

2.1.1.33 bool SDCardFileRemoveWithDir (const SDCardDir * dirHandler, const char * filename)

Removes the file specified by the 'filename' given. Returns true if successful; otherwise return false.

2.1.1.34 bool SDCardFileRename (const char * currentFilename, const char * newFilename)

Rename the file 'currentFilename' to 'newFilename'. Return true if successful; otherwise returns false. If a file with the name 'newFilename' already exists, [SDCardFileRename\(\)](#) returns false.

2.1.1.35 bool SDCardFileSetPosition (SDCardFile * fileHandler, const u32 pos)

Sets the current position from the fileHandler to 'pos'. Returns true on success, or false if an error occurred.

2.1.1.36 u32 SDCardFileWrite (SDCardFile * fileHandler, const u8 * writeBuffer, const u32 writeBufferSize)

Writes at most 'writeBufferSize' bytes of data from 'writeBuffer' to the file. Returns the number of bytes that were actually written, or 0 if an error occurred.

Parameters

in	<i>fileHandler</i>	pointer to a SDCardFile object
in	<i>writeBuffer</i>	a data array
in	<i>writeBufferSize</i>	number of bytes

2.1.1.37 bool SDCardIsMounted (void)

Returns true if the SD Card is mounted; otherwise returns false.

2.1.1.38 SDCardEntryStatus* SDCardLs (const char * path, int * size)

Returns an array of SDCardEntryStatus objects for all the files and directories in the directory. Returns a NULL if the directory is unreadable, does not exist or is empty.

2.1.1.39 SDCardEntryStatus* SDCardLsWithDir (const SDCardDir * dirHandler, int * size)

Returns an array of SDCardEntryStatus objects for all the files and directories in the directory. Returns a NULL if the directory is unreadable, does not exist or is empty.

2.1.1.40 bool SDCardMount (void)

Try to mount a SD Card. Returns true on success; otherwise returns false.

2.1.1.41 bool SDCardRemoveAllFromDirectory (const char * *path*)

Delete all Elements inside the directory.

2.1.1.42 bool SDCardRemoveAllFromDirectoryWithDir (SDCardDir * *dirHandler*)

Delete all Elements inside the directory.

2.1.1.43 bool SDCardUnmount (void)

Try to unmount a SD Card. Returns true on success; otherwise returns false.

Index

SDCardDirCd
 SDCardIO.h, [5](#)

SDCardDirCdUp
 SDCardIO.h, [5](#)

SDCardDirClose
 SDCardIO.h, [5](#)

SDCardDirCountDirectories
 SDCardIO.h, [5](#)

SDCardDirCountDirectoriesWithDir
 SDCardIO.h, [5](#)

SDCardDirCountFiles
 SDCardIO.h, [5](#)

SDCardDirCountFilesWithDir
 SDCardIO.h, [6](#)

SDCardDirExists
 SDCardIO.h, [6](#)

SDCardDirGetDirectoryName
 SDCardIO.h, [6](#)

SDCardDirGetFilename
 SDCardIO.h, [6](#)

SDCardDirGetName
 SDCardIO.h, [6](#)

SDCardDirGetPath
 SDCardIO.h, [6](#)

SDCardDirMakeDirectory
 SDCardIO.h, [6](#)

SDCardDirMakeDirectoryWithDir
 SDCardIO.h, [6](#)

SDCardDirOpen
 SDCardIO.h, [6](#)

SDCardDirOpenWithPath
 SDCardIO.h, [6](#)

SDCardDirRemoveDirectory
 SDCardIO.h, [6](#)

SDCardDirRemoveDirectoryRecursive
 SDCardIO.h, [6](#)

SDCardDirRemoveDirectoryRecursiveWithDir
 SDCardIO.h, [7](#)

SDCardDirRemoveDirectoryWithDir
 SDCardIO.h, [7](#)

SDCardEntryStatusDestroy
 SDCardIO.h, [7](#)

SDCardExists
 SDCardIO.h, [7](#)

SDCardFileClose
 SDCardIO.h, [7](#)

SDCardFileCopy
 SDCardIO.h, [7](#)

SDCardFileFlush
 SDCardIO.h, [7](#)

SDCardFileGetPostion
 SDCardIO.h, [7](#)

SDCardFileGetSize
 SDCardIO.h, [7](#)

SDCardFileOpen
 SDCardIO.h, [7](#)

SDCardFileOpenWithDir
 SDCardIO.h, [9](#)

SDCardFilePeek
 SDCardIO.h, [9](#)

SDCardFileRead
 SDCardIO.h, [9](#)

SDCardFileRemove
 SDCardIO.h, [10](#)

SDCardFileRemoveWithDir
 SDCardIO.h, [10](#)

SDCardFileRename
 SDCardIO.h, [10](#)

SDCardFileSetPosition
 SDCardIO.h, [10](#)

SDCardFileWrite
 SDCardIO.h, [10](#)

SDCardIO.h, [3](#)
 SDCardDirCd, [5](#)
 SDCardDirCdUp, [5](#)
 SDCardDirClose, [5](#)
 SDCardDirCountDirectories, [5](#)
 SDCardDirCountDirectoriesWithDir, [5](#)
 SDCardDirCountFiles, [5](#)
 SDCardDirCountFilesWithDir, [6](#)
 SDCardDirExists, [6](#)
 SDCardDirGetDirectoryName, [6](#)
 SDCardDirGetFilename, [6](#)
 SDCardDirGetName, [6](#)
 SDCardDirGetPath, [6](#)
 SDCardDirMakeDirectory, [6](#)
 SDCardDirMakeDirectoryWithDir, [6](#)
 SDCardDirOpen, [6](#)
 SDCardDirOpenWithPath, [6](#)
 SDCardDirRemoveDirectory, [6](#)
 SDCardDirRemoveDirectoryRecursive, [6](#)
 SDCardDirRemoveDirectoryRecursiveWithDir, [7](#)
 SDCardDirRemoveDirectoryWithDir, [7](#)
 SDCardEntryStatusDestroy, [7](#)
 SDCardExists, [7](#)
 SDCardFileClose, [7](#)
 SDCardFileCopy, [7](#)
 SDCardFileFlush, [7](#)

- SDCardFileGetPostion, [7](#)
- SDCardFileGetSize, [7](#)
- SDCardFileOpen, [7](#)
- SDCardFileOpenWithDir, [9](#)
- SDCardFilePeek, [9](#)
- SDCardFileRead, [9](#)
- SDCardFileRemove, [10](#)
- SDCardFileRemoveWithDir, [10](#)
- SDCardFileRename, [10](#)
- SDCardFileSetPosition, [10](#)
- SDCardFileWrite, [10](#)
- SDCardIsMounted, [10](#)
- SDCardLs, [10](#)
- SDCardLsWithDir, [10](#)
- SDCardMount, [10](#)
- SDCardRemoveAllFromDirectory, [10](#)
- SDCardRemoveAllFromDirectoryWithDir, [11](#)
- SDCardUnmount, [11](#)
- SDCardIsMounted
 - SDCardIO.h, [10](#)
- SDCardLs
 - SDCardIO.h, [10](#)
- SDCardLsWithDir
 - SDCardIO.h, [10](#)
- SDCardMount
 - SDCardIO.h, [10](#)
- SDCardRemoveAllFromDirectory
 - SDCardIO.h, [10](#)
- SDCardRemoveAllFromDirectoryWithDir
 - SDCardIO.h, [11](#)
- SDCardUnmount
 - SDCardIO.h, [11](#)

Annex 7

ELF Loader

Contents

1	File Index	1
1.1	File List	1
2	File Documentation	3
2.1	ElfLoader.h File Reference	3
2.1.1	Detailed Description	3
2.1.2	Function Documentation	3
2.1.2.1	LoadElf(const char *elfBinaryName)	3
	Index	5

Chapter 1

File Index

1.1 File List

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

ElfLoader.h	Provides functionality to load an ELF file into memory	3
-----------------------------	--	-------------------

Chapter 2

File Documentation

2.1 ElfLoader.h File Reference

Provides functionality to load an ELF file into memory.

```
#include <mv_types.h>
```

Functions

- u32 [LoadElf](#) (const char *elfBinaryName)
Load an elf binary from the flash into the memory.

2.1.1 Detailed Description

Provides functionality to load an ELF file into memory.

2.1.2 Function Documentation

2.1.2.1 u32 LoadElf (const char * elfBinaryName)

Load an elf binary from the flash into the memory.

Parameters

in	<i>elfBinaryName</i>	holds the name of the binary
----	----------------------	------------------------------

Returns

If loading is completed, the function returns the entry point address. Otherwise, a zero value is returned.

Index

ElfLoader.h, [3](#)
LoadElf, [3](#)

LoadElf
ElfLoader.h, [3](#)

Annex 8

FlashIO

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	FlashFile Struct Reference	5
3.1.1	Detailed Description	5
4	File Documentation	7
4.1	FlashIO.h File Reference	7
4.1.1	Detailed Description	8
4.1.2	Enumeration Type Documentation	8
4.1.2.1	FlashFileError	8
4.1.2.2	FlashFileMode	8
4.1.3	Function Documentation	8
4.1.3.1	FlashFileAvailableMemory	8
4.1.3.2	FlashFileClose	8
4.1.3.3	FlashFileExists	9
4.1.3.4	FlashFileGetAvailableSpace	9
4.1.3.5	FlashFileGetDeviceID	9
4.1.3.6	FlashFileGetMaxSize	9
4.1.3.7	FlashFileGetPosition	9
4.1.3.8	FlashFileGetSize	9
4.1.3.9	FlashFileOpen	9
4.1.3.10	FlashFilePeek	9
4.1.3.11	FlashFileRead	9
4.1.3.12	FlashFileRemove	10
4.1.3.13	FlashFileRename	10
4.1.3.14	FlashFileSetPosition	10
4.1.3.15	FlashFileWrite	10

Index	11
-----------------------	----

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

FlashFile	5
-------------------------------------	---

Chapter 2

File Index

2.1 File List

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

FlashIO.h	This file provides functions for read/write files from/to the EEPROM	7
---------------------------	--	---

Chapter 3

Data Structure Documentation

3.1 FlashFile Struct Reference

```
#include <FlashIO.h>
```

3.1.1 Detailed Description

Object containing information to control a stream. [FlashFile](#) objects are created by a call of `FlashFileOpen`, which returns a pointer to one of these objects.

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

- [FlashIO.h](#)

Chapter 4

File Documentation

4.1 FlashIO.h File Reference

This file provides functions for read/write files from/to the EEPROM.

```
#include <stdbool.h>
#include <mv_types.h>
```

Data Structures

- struct [FlashFile](#)

Enumerations

- enum [FlashFileMode](#)
- enum [FlashFileError](#)

Functions

- [FlashFile](#) * [FlashFileOpen](#) (const char *filename, const [FlashFileMode](#) mode)
Returns a valid pointer if successful; otherwise returns NULL.
- bool [FlashFileClose](#) ([FlashFile](#) **fileHandler)
Close the [FlashFile](#) handler.
- u32 [FlashFileWrite](#) ([FlashFile](#) *fileHandler, const u8 *writeBuffer, const u32 writeBufferSize)
Writes at most 'writeBufferSize' bytes of data from 'writeBuffer' to the file. Returns the number of bytes that were actually written, or 0 if an error occurred.
- u32 [FlashFileRead](#) ([FlashFile](#) *fileHandler, u8 *readBuffer, const u32 readBufferSize)
Reads at most 'readBufferSize' bytes from the file into 'readBuffer', and returns the number of bytes read. If an error occurs, such as when attempting to read from a file opened in WriteOnly mode, this function returns 0.
- u32 [FlashFilePeek](#) (const [FlashFile](#) *fileHandler, u8 *readBuffer, const u32 readBufferSize)
Reads at most 'readBufferSize' bytes from the file into 'readBuffer', without side effects (i.e., if you call [FlashFileRead\(\)](#) after [FlashFilePeek\(\)](#), you will get the same data). Returns the number of bytes read. If an error occurs, such as when attempting to peek a file opened in WriteOnly mode, this function returns 0.
- bool [FlashFileSetPosition](#) ([FlashFile](#) *fileHandler, const u32 pos)
Sets the current position from the fileHandler to 'pos'. Returns true on success, or false if an error occurred. Note: This function will only change the position if the fileHandler is in ReadOnly mode.
- u32 [FlashFileGetPosition](#) (const [FlashFile](#) *fileHandler)
Returns the current position from the fileHandler.

- u32 [FlashFileGetAvailableSpace](#) ([FlashFile](#) *fileHandler)
Returns the available space for the fileHandler in bytes.
- bool [FlashFileRemove](#) (const char *filename)
Removes the file specified by the 'filename' given. Returns true if successful; otherwise return false.
- bool [FlashFileRename](#) (const char *oldFilename, const char *newFilename)
Rename the file 'oldFilename' to 'newFilename'. Return true if successful; otherwise returns false. If a file with the name 'newFilename' already exists, [FlashFileRename\(\)](#) returns false.
- u32 [FlashFileGetSize](#) (const char *filename)
Returns the size of the file.
- u32 [FlashFileGetMaxSize](#) (const char *filename)
Returns the maximum size of the file If the file is not existing or the maximum size is not set yet, this function returns 0.
- bool [FlashFileExists](#) (const char *filename)
Returns true if the file specified by 'filename' exists; otherwise returns false.
- u32 [FlashFileAvailableMemory](#) (void)
Returns the available space of the flash in bytes.
- const u8 * [FlashFileGetDeviceID](#) (void)
Returns the device id.

4.1.1 Detailed Description

This file provides functions for read/write files from/to the EEPROM.

4.1.2 Enumeration Type Documentation

4.1.2.1 enum FlashFileError

This enum describes the errors that may be returned by the errorFlashFile() function.

4.1.2.2 enum FlashFileMode

This enum is used with [FlashFileOpen\(\)](#) to describe the mode in which a [FlashFile](#) is opened.

4.1.3 Function Documentation

4.1.3.1 u32 FlashFileAvailableMemory (void)

Returns the available space of the flash in bytes.

4.1.3.2 bool FlashFileClose ([FlashFile](#) ** fileHandler)

Close the [FlashFile](#) handler.

Parameters

out	<i>fileHandler</i>	pointer to a FlashFile object
-----	--------------------	---

Returns

If the FlashFile-handler is successfully closed, the function returns true. Otherwise, false is returned.

4.1.3.3 bool FlashFileExists (const char * *filename*)

Returns true if the file specified by 'filename' exists; otherwise returns false.

4.1.3.4 u32 FlashFileGetAvailableSpace (FlashFile * *fileHandler*)

Returns the available space for the fileHandler in bytes.

4.1.3.5 const u8* FlashFileGetDeviceID (void)

Returns the device id.

4.1.3.6 u32 FlashFileGetMaxSize (const char * *filename*)

Returns the maximum size of the file. If the file is not existing or the maximum size is not set yet, this function returns 0.

4.1.3.7 u32 FlashFileGetPosition (const FlashFile * *fileHandler*)

Returns the current position from the fileHandler.

4.1.3.8 u32 FlashFileGetSize (const char * *filename*)

Returns the size of the file.

4.1.3.9 FlashFile* FlashFileOpen (const char * *filename*, const FlashFileMode *mode*)

Returns a valid pointer if successful; otherwise returns NULL.

Parameters

in	<i>filename</i>	represent the file with the given name
in	<i>mode</i>	the mode must be ReadOnly, WriteOnly or Append

Returns

If the file is successfully opened, the function returns a pointer to a [FlashFile](#) object. Otherwise, a null pointer is returned.

4.1.3.10 u32 FlashFilePeek (const FlashFile * *fileHandler*, u8 * *readBuffer*, const u32 *readBufferSize*)

Reads at most 'readBufferSize' bytes from the file into 'readBuffer', without side effects (i.e., if you call [FlashFileRead\(\)](#) after [FlashFilePeek\(\)](#), you will get the same data). Returns the number of bytes read. If an error occurs, such as when attempting to peek a file opened in WriteOnly mode, this function returns 0.

4.1.3.11 u32 FlashFileRead (FlashFile * *fileHandler*, u8 * *readBuffer*, const u32 *readBufferSize*)

Reads at most 'readBufferSize' bytes from the file into 'readBuffer', and returns the number of bytes read. If an error occurs, such as when attempting to read from a file opened in WriteOnly mode, this function returns 0.

Parameters

in	<i>fileHandler</i>	pointer to a FlashFile object
out	<i>readBuffer</i>	a data array
in	<i>readBufferSize</i>	number of bytes

4.1.3.12 bool FlashFileRemove (const char * *filename*)

Removes the file specified by the 'filename' given. Returns true if successful; otherwise return false.

4.1.3.13 bool FlashFileRename (const char * *oldFilename*, const char * *newFilename*)

Rename the file 'oldFilename' to 'newFilename'. Return true if successful; otherwise returns false. If a file with the name 'newFilename' already exists, [FlashFileRename\(\)](#) returns false.

4.1.3.14 bool FlashFileSetPosition ([FlashFile](#) * *fileHandler*, const u32 *pos*)

Sets the current position from the fileHandler to 'pos'. Returns true on success, or false if an error occurred. Note: This function will only change the position if the fileHandler is in ReadOnly mode.

4.1.3.15 u32 FlashFileWrite ([FlashFile](#) * *fileHandler*, const u8 * *writeBuffer*, const u32 *writeBufferSize*)

Writes at most 'writeBufferSize' bytes of data from 'writeBuffer' to the file. Returns the number of bytes that were actually written, or 0 if an error occurred.

Parameters

in	<i>fileHandler</i>	pointer to a FlashFile object
in	<i>writeBuffer</i>	a data array
in	<i>writeBufferSize</i>	number of bytes

Index

FlashFile, [5](#)
FlashFileAvailableMemory
 FlashIO.h, [8](#)
FlashFileClose
 FlashIO.h, [8](#)
FlashFileError
 FlashIO.h, [8](#)
FlashFileExists
 FlashIO.h, [8](#)
FlashFileGetAvailableSpace
 FlashIO.h, [9](#)
FlashFileGetDeviceID
 FlashIO.h, [9](#)
FlashFileGetMaxSize
 FlashIO.h, [9](#)
FlashFileGetPosition
 FlashIO.h, [9](#)
FlashFileGetSize
 FlashIO.h, [9](#)
FlashFileMode
 FlashIO.h, [8](#)
FlashFileOpen
 FlashIO.h, [9](#)
FlashFilePeek
 FlashIO.h, [9](#)
FlashFileRead
 FlashIO.h, [9](#)
FlashFileRemove
 FlashIO.h, [10](#)
FlashFileRename
 FlashIO.h, [10](#)
FlashFileSetPosition
 FlashIO.h, [10](#)
FlashFileWrite
 FlashIO.h, [10](#)
FlashIO.h, [7](#)
 FlashFileAvailableMemory, [8](#)
 FlashFileClose, [8](#)
 FlashFileError, [8](#)
 FlashFileExists, [8](#)
 FlashFileGetAvailableSpace, [9](#)
 FlashFileGetDeviceID, [9](#)
 FlashFileGetMaxSize, [9](#)
 FlashFileGetPosition, [9](#)
 FlashFileGetSize, [9](#)
 FlashFileMode, [8](#)
 FlashFileOpen, [9](#)
 FlashFilePeek, [9](#)
 FlashFileRead, [9](#)
 FlashFileRemove, [10](#)
 FlashFileRename, [10](#)
 FlashFileSetPosition, [10](#)
 FlashFileWrite, [10](#)

Annex 9

Audio

Contents

1	File Index	1
1.1	File List	1
2	File Documentation	3
2.1	/Developer/projects/EoT/WorkPackage_3/myriad/apps/audio/leon/Audio.h File Reference	3
2.1.1	Function Documentation	4
2.1.1.1	AudioGetDuration	4
2.1.1.2	AudioGetPosition	4
2.1.1.3	AudioGetVolume	4
2.1.1.4	AudiolsInPlaybackMode	4
2.1.1.5	AudiolsInRecordMode	4
2.1.1.6	AudiolsMuted	4
2.1.1.7	AudiolsPaused	4
2.1.1.8	AudiolsSeekable	4
2.1.1.9	AudiolsStopped	4
2.1.1.10	AudioPause	5
2.1.1.11	AudioPlay	5
2.1.1.12	AudioRecord	5
2.1.1.13	AudioResume	5
2.1.1.14	AudioSetMute	5
2.1.1.15	AudioSetPosition	5
2.1.1.16	AudioSetVolume	5
2.1.1.17	AudioStop	5
	Index	6

Chapter 1

File Index

1.1 File List

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

/Developer/projects/EoT/WorkPackage_3/myriad/apps/audio/leon/ Audio.h	3
---	---

Chapter 2

File Documentation

2.1 /Developer/projects/EoT/WorkPackage_3/myriad/apps/audio/leon/Audio.h File Reference

```
#include <stdbool.h>
#include <mv_types.h>
```

Functions

- u32 [AudioGetPosition](#) (void)
Returns the current position in milliseconds.
- void [AudioSetPosition](#) (u32 position)
Change the playback position, if the audio source is seekable.
- u32 [AudioGetDuration](#) (void)
Returns the total playback time in milliseconds.
- bool [AudioIsMuted](#) (void)
Returns true if the playback is muted; otherwise false.
- void [AudioSetMute](#) (bool enable)
Enable or disable the mute mode.
- bool [AudioIsSeekable](#) (void)
Returns true if the playback is seekable; false otherwise.
- u8 [AudioGetVolume](#) (void)
Returns the playback volume.
- void [AudioSetVolume](#) (int8_t volume)
Set the playback volume.
- bool [AudioPlay](#) (const char *filepath)
Start or resume playing the current source.
- void [AudioPause](#) (void)
Pause playing the current source.
- void [AudioStop](#) (void)
Stop playing, and reset the play position to the beginning.
- void [AudioResume](#) (void)
Resume playing/recording audio.
- bool [AudioRecord](#) (const char *filepath)
Start or resume recording.
- bool [AudioIsInPlaybackMode](#) (void)

Returns true if playback is active; otherwise false.

- bool [AudiolsPaused](#) (void)

Returns true if playback/record is paused; otherwise false.

- bool [AudiolsStopped](#) (void)

Returns true if playback/record is stopped; otherwise false.

- bool [AudiolsInRecordMode](#) (void)

Returns true if record is active; otherwise false.

2.1.1 Function Documentation

2.1.1.1 u32 AudioGetDuration (void)

Returns the total playback time in milliseconds.

2.1.1.2 u32 AudioGetPosition (void)

Returns the current position in milliseconds.

To change this position, use the [AudioSetPosition\(u32\)](#) method.

2.1.1.3 u8 AudioGetVolume (void)

Returns the playback volume.

2.1.1.4 bool AudiolsInPlaybackMode (void)

Returns true if playback is active; otherwise false.

2.1.1.5 bool AudiolsInRecordMode (void)

Returns true if record is active; otherwise false.

2.1.1.6 bool AudiolsMuted (void)

Returns true if the playback is muted; otherwise false.

2.1.1.7 bool AudiolsPaused (void)

Returns true if playback/record is paused; otherwise false.

2.1.1.8 bool AudiolsSeekable (void)

Returns true if the playback is seekable; false otherwise.

2.1.1.9 bool AudiolsStopped (void)

Returns true if playback/record is stopped; otherwise false.

2.1.1.10 void AudioPause (void)

Pause playing the current source.

2.1.1.11 bool AudioPlay (const char * *filepath*)

Start or resume playing the current source.

Parameters

<i>filepath</i>	path to the audio source.
-----------------	---------------------------

2.1.1.12 bool AudioRecord (const char * *filepath*)

Start or resume recording.

Parameters

<i>filepath</i>	save the audio data to file destination.
-----------------	--

2.1.1.13 void AudioResume (void)

Resume playing/recording audio.

2.1.1.14 void AudioSetMute (bool *enable*)

Enable or disable the mute mode.

2.1.1.15 void AudioSetPosition (u32 *position*)

Change the playback position, if the audio source is seekable.

Parameters

<i>position</i>	in milliseconds.
-----------------	------------------

2.1.1.16 void AudioSetVolume (int8_t *volume*)

Set the playback volume.

Parameters

<i>volume</i>	the range is from 0(silent) to 100(maximum), values outside this range will be clamped.
---------------	---

2.1.1.17 void AudioStop (void)

Stop playing, and reset the play position to the beginning.

Index

/Developer/projects/EoT/WorkPackage_3/myriad/apps/audio/leoAudio.h, [5](#)
Audio.h, [3](#)
Audio.h
AudioGetDuration, [4](#)
AudioGetPosition, [4](#)
AudioGetVolume, [4](#)
AudiolsInPlaybackMode, [4](#)
AudiolsInRecordMode, [4](#)
AudiolsMuted, [4](#)
AudiolsPaused, [4](#)
AudiolsSeekable, [4](#)
AudiolsStopped, [4](#)
AudioPause, [4](#)
AudioPlay, [5](#)
AudioRecord, [5](#)
AudioResume, [5](#)
AudioSetMute, [5](#)
AudioSetPosition, [5](#)
AudioSetVolume, [5](#)
AudioStop, [5](#)
AudioGetDuration
Audio.h, [4](#)
AudioGetPosition
Audio.h, [4](#)
AudioGetVolume
Audio.h, [4](#)
AudiolsInPlaybackMode
Audio.h, [4](#)
AudiolsInRecordMode
Audio.h, [4](#)
AudiolsMuted
Audio.h, [4](#)
AudiolsPaused
Audio.h, [4](#)
AudiolsSeekable
Audio.h, [4](#)
AudiolsStopped
Audio.h, [4](#)
AudioPause
Audio.h, [4](#)
AudioPlay
Audio.h, [5](#)
AudioRecord
Audio.h, [5](#)
AudioResume
Audio.h, [5](#)
AudioSetMute
Audio.h, [5](#)
AudioSetPosition

Annex 10

Histogram Matching

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	Rect Struct Reference	5
3.1.1	Detailed Description	5
4	File Documentation	7
4.1	ColorHistogram.h File Reference	7
4.1.1	Detailed Description	7
4.1.2	Function Documentation	7
4.1.2.1	computeHistogram(frameBuffer *image, int *histogramBuffer, u8 bins[])	7
4.1.2.2	computeHistogramInRoi(frameBuffer *image, int *histogramBuffer, u8 bins[], Rect roi)	8
4.2	HistogramMatching.h File Reference	8
4.2.1	Detailed Description	8
4.2.2	Function Documentation	8
4.2.2.1	earthMoversDistance(u16 bins, int histogramOne[], int histogramTwo[])	8
4.2.2.2	hellingerDistance(u16 bins, int histogramOne[], int histogramTwo[])	8
4.2.2.3	histogramIntersectionDistance(u16 bins, int histogramOne[], int histogramTwo[])	9
	Index	11

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Rect	Defines a rectangle in an image. The origin is in the upper left corner	5
----------------------	---	-------------------

Chapter 2

File Index

2.1 File List

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

ColorHistogram.h	
Provides functions to compute color histograms	7
HistogramMatching.h	
Provides functions to compute histogram matching scores	8

Chapter 3

Data Structure Documentation

3.1 Rect Struct Reference

Defines a rectangle in an image. The origin is in the upper left corner.

```
#include <ColorHistogram.h>
```

Data Fields

- unsigned int **x**
- unsigned int **y**
- unsigned int **width**
- unsigned int **height**

3.1.1 Detailed Description

Defines a rectangle in an image. The origin is in the upper left corner.

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

- [ColorHistogram.h](#)

Chapter 4

File Documentation

4.1 ColorHistogram.h File Reference

Provides functions to compute color histograms.

```
#include <mv_types.h>
#include <swcFrameTypes.h>
```

Data Structures

- struct [Rect](#)

Defines a rectangle in an image. The origin is in the upper left corner.

Functions

- void [computeHistogram](#) (frameBuffer *image, int *histogramBuffer, u8 bins[])
Computes the color histogram of the specified image.
- void [computeHistogramInRoi](#) (frameBuffer *image, int *histogramBuffer, u8 bins[], [Rect](#) roi)
Computes the color histogram in the specified region of interest in the image.

4.1.1 Detailed Description

Provides functions to compute color histograms.

4.1.2 Function Documentation

4.1.2.1 void computeHistogram (frameBuffer * image, int * histogramBuffer, u8 bins[])

Computes the color histogram of the specified image.

Parameters

<i>image</i>	A pointer to the image whose histogram shall be computed.
<i>histogramBuffer</i>	The buffer in which to store the histogram.

<i>bins</i>	The number of bins to use for each channel of the provided image.
-------------	---

4.1.2.2 void computeHistogramInRoi (frameBuffer * image, int * histogramBuffer, u8 bins[], Rect roi)

Computes the color histogram in the specified region of interest in the image.

Parameters

<i>image</i>	A pointer to the image whose histogram shall be computed.
<i>histogramBuffer</i>	The buffer in which to store the histogram.
<i>bins</i>	The number of bins to use for each channel of the provided image.
<i>roi</i>	The region (rectangle) of interest in which to compute the histogram.

4.2 HistogramMatching.h File Reference

Provides functions to compute histogram matching scores.

```
#include <mv_types.h>
```

Functions

- float [hellingerDistance](#) (u16 bins, int histogramOne[], int histogramTwo[])
Computes the Hellinger distance of a pair of histograms.
- float [histogramIntersectionDistance](#) (u16 bins, int histogramOne[], int histogramTwo[])
Computes a distance for a pair of histograms based on histogram intersection.
- float [earthMoversDistance](#) (u16 bins, int histogramOne[], int histogramTwo[])
Computes the earth mover's distance of a pair of histograms.

4.2.1 Detailed Description

Provides functions to compute histogram matching scores.

4.2.2 Function Documentation

4.2.2.1 float earthMoversDistance (u16 bins, int histogramOne[], int histogramTwo[])

Computes the earth mover's distance of a pair of histograms.

Parameters

<i>bins</i>	The number of bins to compare.
<i>histogramOne</i>	The first histogram to compare.
<i>histogramTwo</i>	The second histogram to compare.

Returns

The distance between the histograms. A floating point value in the range [0, 1].

4.2.2.2 float hellingerDistance (u16 bins, int histogramOne[], int histogramTwo[])

Computes the Hellinger distance of a pair of histograms.

Parameters

<i>bins</i>	The number of bins to compare.
<i>histogramOne</i>	The first histogram to compare.
<i>histogramTwo</i>	The second histogram to compare.

Returns

The distance between the histograms. A floating point value in the range [0, 1].

4.2.2.3 float histogramIntersectionDistance (u16 *bins*, int *histogramOne*[], int *histogramTwo*[])

Computes a distance for a pair of histograms based on histogram intersection.

Parameters

<i>bins</i>	The number of bins to compare.
<i>histogramOne</i>	The first histogram to compare.
<i>histogramTwo</i>	The second histogram to compare.

Returns

The distance between the histograms. A floating point value in the range [0, 1].

Index

- ColorHistogram.h, [7](#)
 - computeHistogram, [7](#)
 - computeHistogramInRoi, [8](#)
- computeHistogram
 - ColorHistogram.h, [7](#)
- computeHistogramInRoi
 - ColorHistogram.h, [8](#)
- earthMoversDistance
 - HistogramMatching.h, [8](#)
- hellingerDistance
 - HistogramMatching.h, [8](#)
- histogramIntersectionDistance
 - HistogramMatching.h, [9](#)
- HistogramMatching.h, [8](#)
 - earthMoversDistance, [8](#)
 - hellingerDistance, [8](#)
 - histogramIntersectionDistance, [9](#)
- Rect, [5](#)

Annex 11

Rotation-Invariant Face Detector

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	data Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Field Documentation	5
3.1.2.1	degrees	5
3.1.2.2	img_dst	5
3.1.2.3	n_faces_detected	5
3.1.2.4	path_dst	5
4	File Documentation	7
4.1	app_config.c File Reference	7
4.1.1	Detailed Description	7
4.1.2	Macro Definition Documentation	8
4.1.2.1	CMX_CONFIG_SLICE_15_8	8
4.1.2.2	CMX_CONFIG_SLICE_7_0	8
4.1.2.3	L2CACHE_CFG	8
4.1.3	Function Documentation	8
4.1.3.1	__attribute__	8
4.1.3.2	initClocksAndMemory	8
4.2	app_config.h File Reference	8
4.2.1	Detailed Description	9
4.2.2	Macro Definition Documentation	9
4.2.2.1	APP_MSS_CLOCKS	9
4.2.2.2	APP_UPA_CLOCKS	9
4.2.3	Function Documentation	9
4.2.3.1	initClocksAndMemory	9

4.3	main.c File Reference	9
4.3.1	Macro Definition Documentation	10
4.3.1.1	N_SHAVES	10
4.3.1.2	PATH_SOURCE	10
4.3.2	Function Documentation	10
4.3.2.1	drawSquare	10
4.3.2.2	drawSquareColor	11
4.3.2.3	Fatal_extension	11
4.3.2.4	POSIX_Init	11
4.3.3	Variable Documentation	11
4.3.3.1	ccv_bbf_custom	11
4.3.3.2	entryPoints	12
4.3.3.3	faceDetectionSHAVEs0_ApplicationStart	12
4.3.3.4	faceDetectionSHAVEs3_ApplicationStart	12
4.4	rtems_config.h File Reference	12
4.4.1	Detailed Description	13
4.4.2	Macro Definition Documentation	13
4.4.2.1	_RTEMS_CONFIG_H_	13
4.4.3	Function Documentation	13
4.4.3.1	BSP_SET_CLOCK	13
4.4.3.2	BSP_SET_L2C_CONFIG	13
4.5	shave_Start.c File Reference	13
4.5.1	Function Documentation	14
4.5.1.1	ApplicationStart	14

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

data	5
-----------------------	---

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

app_config.c	
Application configuration Leon file	7
app_config.h	
Application configuration Leon header	8
main.c	9
rtems_config.h	
RTEMS configuration Leon header	12
shave_Start.c	13

Chapter 3

Data Structure Documentation

3.1 data Struct Reference

Data Fields

- char **path_dst** [250]
- ccv_dense_matrix_t * **img_dst**
- int **degrees**
- int **n_faces_detected**

3.1.1 Detailed Description

This struct facilitates the scalability. It stores information relative to the rotate image and the result of bbf algorithm. For each SHAVE used there will be a struct.

3.1.2 Field Documentation

3.1.2.1 int degrees

Degrees to rotate the original image

3.1.2.2 ccv_dense_matrix_t* img_dst

Object ccv_dense_matrix. It contains all information relative to the final image.

3.1.2.3 int n_faces_detected

Number of faces detected by bbf algorithm.

3.1.2.4 char path_dst[250]

Path where the rotate image will be saved.

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

- **main.c**

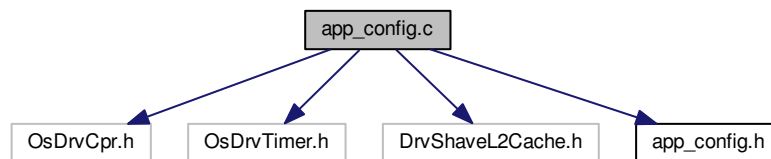
Chapter 4

File Documentation

4.1 app_config.c File Reference

Application configuration Leon file.

```
#include <OsDrvCpr.h>
#include "OsDrvTimer.h"
#include <DrvShaveL2Cache.h>
#include "app_config.h"
Include dependency graph for app_config.c:
```



Macros

- `#define CMX_CONFIG_SLICE_7_0 (0x11111111)`
- `#define CMX_CONFIG_SLICE_15_8 (0x11111111)`
- `#define L2CACHE_CFG (SHAVE_L2CACHE_NORMAL_MODE)`

Functions

- `CmxRamLayoutCfgType __attribute__((section(".cmx.ctrl")))`
- `int initClocksAndMemory (void)`

4.1.1 Detailed Description

Application configuration Leon file.

Copyright

All code copyright Movidius Ltd 2012, all rights reserved. For License Warranty see: [common/license.txt](#)

4.1.2 Macro Definition Documentation

4.1.2.1 `#define CMX_CONFIG_SLICE_15_8 (0x11111111)`

4.1.2.2 `#define CMX_CONFIG_SLICE_7_0 (0x11111111)`

4.1.2.3 `#define L2CACHE_CFG (SHAVE_L2CACHE_NORMAL_MODE)`

4.1.3 Function Documentation

4.1.3.1 `CmxRamLayoutCfgType __attribute__((section(".cmx.ctrl")))`

4.1.3.2 `int initClocksAndMemory (void)`

Setup all the clock configurations needed by this application and also the ddr

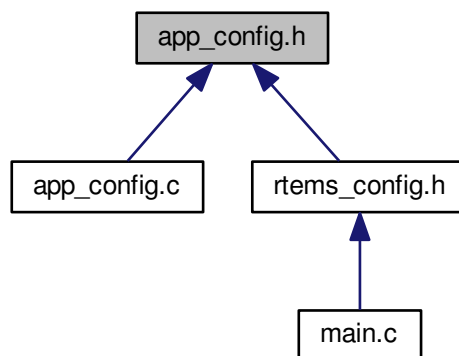
Returns

0 on success, non-zero otherwise

4.2 app_config.h File Reference

Application configuration Leon header.

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



Macros

- `#define APP_MSS_CLOCKS`
- `#define APP_UPA_CLOCKS`

Functions

- `int initClocksAndMemory (void)`

4.2.1 Detailed Description

Application configuration Leon header.

Copyright

All code copyright Movidius Ltd 2012, all rights reserved. For License Warranty see: common/license.txt

4.2.2 Macro Definition Documentation

4.2.2.1 #define APP_MSS_CLOCKS

Value:

```
(DEV_MSS_APB_SLV | \
  DEV_MSS_APB2_CTRL | \
  DEV_MSS_AXI_BRIDGE | \
  DEV_MSS_MXI_CTRL | \
  DEV_MSS_MXI_DEFSLV )
```

4.2.2.2 #define APP_UPA_CLOCKS

Value:

```
(DEV_UPA_SH0 | \
  DEV_UPA_SHAVE_L2 | \
  DEV_UPA_CDMA | \
  DEV_UPA_CTRL )
```

For each SHAVE used, is necessary add it in the UPA_CLOCKS variable. Otherwise, the execution will stop in call of SHAVE. DEV_UPA_SH0 | \ add the SHAVE0 DEV_UPA_SH3 | \ add the SHAVE3

4.2.3 Function Documentation

4.2.3.1 int initClocksAndMemory (void)

Setup all the clock configurations needed by this application and also the ddr

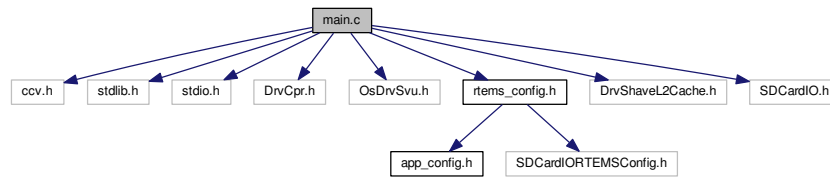
Returns

0 on success, non-zero otherwise

4.3 main.c File Reference

```
#include <ccv.h>
#include <stdlib.h>
#include <stdio.h>
#include <DrvCpr.h>
#include "OsDrvSvu.h"
#include "rtems_config.h"
#include <DrvShaveL2Cache.h>
#include <SDCardIO.h>
```

Include dependency graph for main.c:



Data Structures

- struct **data**

Macros

- `#define N_SHAVES 2`
- `#define PATH_SOURCE "/mnt/sdcard/Rotation-invariant_faceDetector/lena.png"`

Functions

- void **drawSquare** (ccv_dense_matrix_t *img, int SquareX, int SquareY, int w, u8 color)
This function is used for draw a square in a ccv_dense_matrix_t object. Image must be in grayscale.*
- void **drawSquareColor** (ccv_dense_matrix_t *img, int SquareX, int SquareY, int w, u8 red, u8 green, u8 blue)
This function is used for draw a square in a ccv_dense_matrix_t object. Image must be in RGB.*
- void **POSIX_Init** (void *args)
- void **Fatal_extension** (Internal_errors_Source the_source, bool is_internal, uint32_t the_error)

Variables

- ccv_bbf_param_t **ccv_bbf_custom**
- u32 **faceDetectionSHAVES0_ApplicationStart**
- u32 **faceDetectionSHAVES3_ApplicationStart**
- u32 **entryPoints** [N_SHAVES]

4.3.1 Macro Definition Documentation

4.3.1.1 `#define N_SHAVES 2`

Number of SHAVES used in this app

4.3.1.2 `#define PATH_SOURCE "/mnt/sdcard/Rotation-invariant_faceDetector/lena.png"`

Image path with this app works

4.3.2 Function Documentation

4.3.2.1 void drawSquare (ccv_dense_matrix_t * img, int SquareX, int SquareY, int w, u8 color)

This function is used for draw a square in a ccv_dense_matrix_t* object. Image must be in grayscale.

Parameters

in	<i>img</i>	image where square will be drawn
in	<i>SquareX</i>	x coordinate of the Start of square
in	<i>SquareY</i>	y coordinate of the Start of square
in	<i>w</i>	longitude
in	<i>color</i>	square color. The value must be between 0 and 255

4.3.2.2 void drawSquareColor (ccv_dense_matrix_t * *img*, int *SquareX*, int *SquareY*, int *w*, u8 *red*, u8 *green*, u8 *blue*)

This function is used for draw a square in a ccv_dense_matrix_t* object. Image must be in RGB.

Parameters

in	<i>img</i>	image where square will be drawn
in	<i>SquareX</i>	x coordinate of the Start of square
in	<i>SquareY</i>	y coordinate of the Start of square
in	<i>w</i>	longitude
in	<i>red</i>	red component The value must be between 0 and 255
in	<i>green</i>	green component The value must be between 0 and 255
in	<i>blue</i>	blue component The value must be between 0 and 255

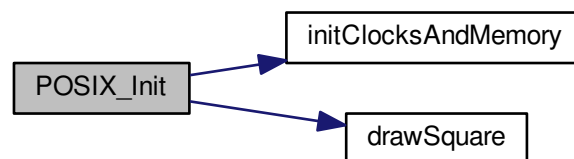
4.3.2.3 void Fatal_extension (Internal_errors_Source *the_source*, bool *is_internal*, uint32_t *the_error*)

4.3.2.4 void POSIX_Init (void * *args*)

<Data struct for the SHAVE0

<Data struct for the SHAVE3

Here is the call graph for this function:



4.3.3 Variable Documentation

4.3.3.1 ccv_bbf_param_t ccv_bbf_custom

Initial value:

```

= {
    .interval = 3,
    .min_neighbors = 2,
    .accurate = 1,

```



```

    .flags = 0,
    .size = {
        90,
        90,
    },
}

```

Custom settings for bbf detection. With this, the time of face detection is reduced from 3.5 seconds to 0.7 seconds. The original values can be found in ccv_bbf file from libccv library.

4.3.3.2 u32 entryPoints[N_SHAVES]

Initial value:

```

= {
    (u32) &faceDetectionSHAVEs0_ApplicationStart,
    (u32) &faceDetectionSHAVEs3_ApplicationStart,
}

```

Array of entypoints. It can facilitate the scalability

4.3.3.3 u32 faceDetectionSHAVEs0_ApplicationStart

4.3.3.4 u32 faceDetectionSHAVEs3_ApplicationStart

SHAVE entry point function serves as starting execution point on SHAVE. One for every SHAVE and execution point. Its name follow the next structure APP_NAME: faceDetectionSHAVEs n_shave: 0 and 3. entry point: Application-Start This fields must be declared in the makefile

4.4 rtems_config.h File Reference

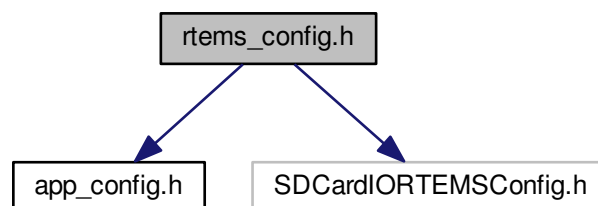
RTEMS configuration Leon header.

```

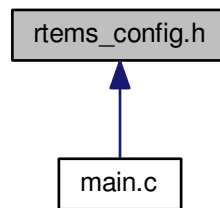
#include "app_config.h"
#include <SDCardIORTEMSConfig.h>

```

Include dependency graph for rtems_config.h:



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



Macros

- `#define _RTEMS_CONFIG_H_`

Functions

- **BSP_SET_CLOCK** (12000, 200000, 1, 1, DEFAULT_RTEMS_CSS_LOS_CLOCKS, APP_MSS_CLOCKS, APP_UPA_CLOCKS, 0, 0)
- **BSP_SET_L2C_CONFIG** (0, L2C_REPL_LRU, 0, L2C_MODE_COPY_BACK, 0, NULL)

4.4.1 Detailed Description

RTEMS configuration Leon header.

Copyright

All code copyright Movidius Ltd 2012, all rights reserved. For License Warranty see: common/license.txt

4.4.2 Macro Definition Documentation

4.4.2.1 `#define _RTEMS_CONFIG_H_`

4.4.3 Function Documentation

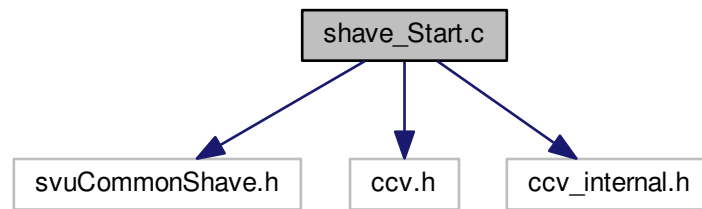
4.4.3.1 **BSP_SET_CLOCK** (12000 , 200000 , 1 , 1 , DEFAULT_RTEMS_CSS_LOS_CLOCKS , APP_MSS_CLOCKS , APP_UPA_CLOCKS , 0 , 0)

4.4.3.2 **BSP_SET_L2C_CONFIG** (0 , L2C_REPL_LRU , 0 , L2C_MODE_COPY_BACK , 0 , NULL)

4.5 shave_Start.c File Reference

```
#include <svuCommonShave.h>
#include "ccv.h"
#include "ccv_internal.h"
```

Include dependency graph for shave_Start.c:



Functions

- void **ApplicationStart** (ccv_dense_matrix_t *img_src, ccv_dense_matrix_t *img_dst, int degrees)
SHAVE entrypoint. ccv_perspective_transform will be running here.

4.5.1 Function Documentation

4.5.1.1 void ApplicationStart (ccv_dense_matrix_t * *img_src*, ccv_dense_matrix_t * *img_dst*, int *degrees*)

SHAVE entrypoint. ccv_perspective_transform will be running here.

Parameters

in	<i>img_src</i>	source image
in	<i>img_dst</i>	destination image
in	<i>degrees</i>	degrees to rotate

Index

- `_RTEMS_CONFIG_H_`
 - `rtems_config.h`, 13
 - `__attribute__`
 - `app_config.c`, 8
- `APP_MSS_CLOCKS`
 - `app_config.h`, 9
- `APP_UPA_CLOCKS`
 - `app_config.h`, 9
- `app_config.c`, 7
 - `__attribute__`, 8
 - `CMX_CONFIG_SLICE_15_8`, 8
 - `CMX_CONFIG_SLICE_7_0`, 8
 - `initClocksAndMemory`, 8
 - `L2CACHE_CFG`, 8
- `app_config.h`, 8
 - `APP_MSS_CLOCKS`, 9
 - `APP_UPA_CLOCKS`, 9
 - `initClocksAndMemory`, 9
- `ApplicationStart`
 - `shave_Start.c`, 14
- `BSP_SET_CLOCK`
 - `rtems_config.h`, 13
- `BSP_SET_L2C_CONFIG`
 - `rtems_config.h`, 13
- `CMX_CONFIG_SLICE_15_8`
 - `app_config.c`, 8
- `CMX_CONFIG_SLICE_7_0`
 - `app_config.c`, 8
- `ccv_bbf_custom`
 - `main.c`, 11
- `data`, 5
 - `degrees`, 5
 - `img_dst`, 5
 - `n_faces_detected`, 5
 - `path_dst`, 5
- `degrees`
 - `data`, 5
- `drawSquare`
 - `main.c`, 10
- `drawSquareColor`
 - `main.c`, 11
- `entryPoints`
 - `main.c`, 12
- `faceDetectionSHAVES0_ApplicationStart`
 - `main.c`, 12
- `faceDetectionSHAVES3_ApplicationStart`
 - `main.c`, 12
- `Fatal_extension`
 - `main.c`, 11
- `img_dst`
 - `data`, 5
- `initClocksAndMemory`
 - `app_config.c`, 8
 - `app_config.h`, 9
- `L2CACHE_CFG`
 - `app_config.c`, 8
- `main.c`, 9
 - `ccv_bbf_custom`, 11
 - `drawSquare`, 10
 - `drawSquareColor`, 11
 - `entryPoints`, 12
 - `faceDetectionSHAVES0_ApplicationStart`, 12
 - `faceDetectionSHAVES3_ApplicationStart`, 12
 - `Fatal_extension`, 11
 - `N_SHAVES`, 10
 - `PATH_SOURCE`, 10
 - `POSIX_Init`, 11
- `N_SHAVES`
 - `main.c`, 10
- `n_faces_detected`
 - `data`, 5
- `PATH_SOURCE`
 - `main.c`, 10
- `POSIX_Init`
 - `main.c`, 11
- `path_dst`
 - `data`, 5
- `rtems_config.h`, 12
 - `_RTEMS_CONFIG_H_`, 13
 - `BSP_SET_CLOCK`, 13
 - `BSP_SET_L2C_CONFIG`, 13
- `shave_Start.c`, 13
 - `ApplicationStart`, 14

Annex 12

Cherokee4WD

Contents

1	README	1
2	Module Index	3
2.1	Modules	3
3	File Index	5
3.1	File List	5
4	Module Documentation	7
4.1	Group title	7
4.1.1	Detailed Description	7
4.1.2	Function Documentation	7
4.1.2.1	Cherokey4WDBackward	7
4.1.2.2	Cherokey4WDForward	7
4.1.2.3	Cherokey4WDSetDirectionSpeed	8
4.1.2.4	Cherokey4WDSetup	8
4.1.2.5	Cherokey4WDStop	8
4.1.2.6	Cherokey4WDTurnLeft	8
4.1.2.7	Cherokey4WDTurnRight	8
4.2	Cherokey4WD from Myriad	10
5	File Documentation	11
5.1	Cherokey4WD.h File Reference	11
5.1.1	Detailed Description	11
5.2	Cherokey4WDArduino.c File Reference	11
5.2.1	Detailed Description	11
5.3	Cherokey4WDArduino.h File Reference	12
5.3.1	Detailed Description	12
5.4	Cherokey4WDArduinoTypes.h File Reference	12
5.4.1	Detailed Description	12
5.5	Cherokey4WDBoards.h File Reference	12
5.5.1	Detailed Description	12
5.6	Cherokey4WDMyriad.h File Reference	12

5.6.1 Detailed Description	12
5.7 Cherokey4WDRaspberryPi.h File Reference	12
5.7.1 Detailed Description	12
Index	13

Chapter 1

README

Library for controlling the Cherokey4WD robot.

Structure

- [Cherokey4WD.h](#), Cherokey4WD.c Main library files, provide functionality for moving forward/backward/turn left/turn right and stop.
- [Cherokey4WDBoards.h](#) Parser to select the proper includes according to the current board, the parser selects the proper board based on the available #define instructions.
- Cherokey4WD*BOARDNAME*.h,.c Board specific files

For install instructions refer to INSTALL_*BOARD*.txt files.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Group title	7
Cherokey4WD from Myriad	10

Chapter 3

File Index

3.1 File List

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

Cherokey4WD.h	11
Cherokey4WDArduino.c	11
Cherokey4WDArduino.h	12
Cherokey4WDArduinoTypes.h	12
Cherokey4WDBoards.h	12
Cherokey4WDMyriad.h	12
Cherokey4WDRaspberryPi.h	12

Chapter 4

Module Documentation

4.1 Group title

Group briefing.

Functions

- void [Cherokey4WDSetup](#) (uint8 pin_m1_enable, uint8 pin_m1_pwm, uint8 pin_m2_enable, uint8 pin_m2_pwm, uint8 pin_state_forward)
- void [Cherokey4WDSetDirectionSpeed](#) (uint8 direction_left, uint8 speed_left, uint8 direction_right, uint8 speed_right)
- void [Cherokey4WDStop](#) ()
- void [Cherokey4WDForward](#) (uint8 speed_left, uint8 speed_right)
- void [Cherokey4WDBackward](#) (uint8 speed_left, uint8 speed_right)
- void [Cherokey4WDTurnLeft](#) (uint8 speed_left, uint8 speed_right)
- void [Cherokey4WDTurnRight](#) (uint8 speed_left, uint8 speed_right)

4.1.1 Detailed Description

Group briefing. Detailed description

4.1.2 Function Documentation

4.1.2.1 void [Cherokey4WDBackward](#) (uint8 *speed_left*, uint8 *speed_right*)

Moves the Cherokey 4WD backward at the specified speed (0 to 255) on each of the wheels (left and right wheels)

Parameters

<i>speed_left</i>	
<i>speed_right</i>	

4.1.2.2 void [Cherokey4WDForward](#) (uint8 *speed_left*, uint8 *speed_right*)

Moves the Cherokey 4WD forward at the specified speed (0 to 255) on each of the wheels (left and right wheels)

Parameters

<i>speed_left</i>	
<i>speed_right</i>	

4.1.2.3 void Cherokey4WDSetDirectionSpeed (uint8 *direction_left*, uint8 *speed_left*, uint8 *direction_right*, uint8 *speed_right*)

Set the direction and speed for the wheels of the car

Parameters

<i>direction_left</i>	Direction of the wheels on the left side
<i>speed_left</i>	Speed of the wheels on the left side (0 to 255)
<i>direction_right</i>	Direction of the wheels on the right side
<i>speed_right</i>	Speed of the wheels on the right side (0 to 255)

4.1.2.4 void Cherokey4WDSetup (uint8 *pin_m1_enable*, uint8 *pin_m1_pwm*, uint8 *pin_m2_enable*, uint8 *pin_m2_pwm*, uint8 *pin_state_forward*)

Set up the pins which are going to be used to control the Cherokey4WD. Set the pin state which corresponds to the forward direction (HIGH or LOW)

Parameters

<i>pin_m1_enable</i>	
<i>pin_m1_pwm</i>	
<i>pin_m2_enable</i>	
<i>pin_m2_pwm</i>	
<i>pin_state_ - forward</i>	

4.1.2.5 void Cherokey4WDStop ()

Stop the car

4.1.2.6 void Cherokey4WDTurnLeft (uint8 *speed_left*, uint8 *speed_right*)

Turns the Cherokey 4WD to the left at the specified speed (0 to 255) on each of the wheels (left and right wheels)

Parameters

<i>speed_left</i>	
<i>speed_right</i>	

4.1.2.7 void Cherokey4WDTurnRight (uint8 *speed_left*, uint8 *speed_right*)

Turns the Cherokey 4WD to the right at the specified speed (0 to 255) on each of the wheels (left and right wheels)

Parameters

<i>speed_left</i>	
-------------------	--

<i>speed_right</i>	
--------------------	--

4.2 Cherokey4WD from Myriad

Controls the Cherokey4WD from the Myriad processor.

Controls the Cherokey4WD from the Myriad processor. Defines the required functions to control the Myriad pins

Chapter 5

File Documentation

5.1 Cherokey4WD.h File Reference

```
#include "Cherokey4WDBoards.h"
```

Functions

- void [Cherokey4WDSetup](#) (uint8 pin_m1_enable, uint8 pin_m1_pwm, uint8 pin_m2_enable, uint8 pin_m2_pwm, uint8 pin_state_forward)
- void [Cherokey4WDSetDirectionSpeed](#) (uint8 direction_left, uint8 speed_left, uint8 direction_right, uint8 speed_right)
- void [Cherokey4WDStop](#) ()
- void [Cherokey4WDForward](#) (uint8 speed_left, uint8 speed_right)
- void [Cherokey4WDBackward](#) (uint8 speed_left, uint8 speed_right)
- void [Cherokey4WDTurnLeft](#) (uint8 speed_left, uint8 speed_right)
- void [Cherokey4WDTurnRight](#) (uint8 speed_left, uint8 speed_right)

5.1.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see: common/license.txt

Created on: 20 Oct 2015 Author: dexamont

5.2 Cherokey4WDArduino.c File Reference

```
#include "Cherokey4WDArduino.h"
```

5.2.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see: common/license.txt

Created on: 27 Oct 2015 Author: dexamont

5.3 Cherokey4WDArduino.h File Reference

5.3.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see: common/license.txt

Created on: 27 Oct 2015 Author: dexamont

5.4 Cherokey4WDArduinoTypes.h File Reference

5.4.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see: common/license.txt

Created on: 27 Oct 2015 Author: dexamont

5.5 Cherokey4WDBoards.h File Reference

5.5.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see: common/license.txt

Created on: 20 Oct 2015 Author: dexamont

5.6 Cherokey4WDMyriad.h File Reference

5.6.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see: common/license.txt

Created on: 20 Oct 2015 Author: dexamont

5.7 Cherokey4WDRaspberryPi.h File Reference

5.7.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see: common/license.txt

Created on: 21 Oct 2015 Author: dexamont

Index

Cherokey4WD from Myriad, [10](#)
Cherokey4WD.h, [11](#)
Cherokey4WDArduino.c, [11](#)
Cherokey4WDArduino.h, [12](#)
Cherokey4WDArduinoTypes.h, [12](#)
Cherokey4WDBackward
 Group title, [7](#)
Cherokey4WDBoards.h, [12](#)
Cherokey4WDForward
 Group title, [7](#)
Cherokey4WDMyriad.h, [12](#)
Cherokey4WDRaspberryPi.h, [12](#)
Cherokey4WDSetDirectionSpeed
 Group title, [8](#)
Cherokey4WDSetup
 Group title, [8](#)
Cherokey4WDStop
 Group title, [8](#)
Cherokey4WDTurnLeft
 Group title, [8](#)
Cherokey4WDTurnRight
 Group title, [8](#)

Group title, [7](#)
 Cherokey4WDBackward, [7](#)
 Cherokey4WDForward, [7](#)
 Cherokey4WDSetDirectionSpeed, [8](#)
 Cherokey4WDSetup, [8](#)
 Cherokey4WDStop, [8](#)
 Cherokey4WDTurnLeft, [8](#)
 Cherokey4WDTurnRight, [8](#)