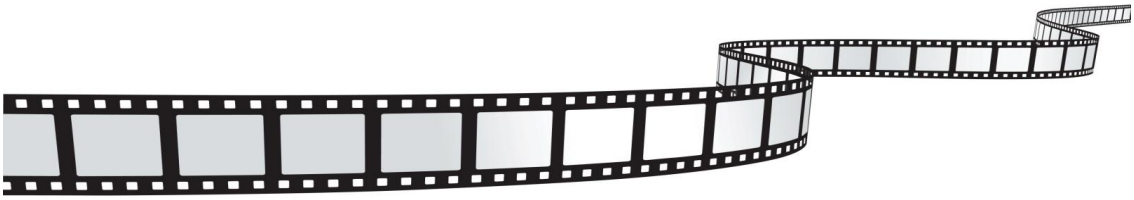
18.08.10



Movidius

Contents

1 Introduction **12**

2 Module Index **13**

2.1 Modules 13

3 Data Structure Index **14**

3.1 Data Structures 14

4 File Index **16**

4.1 File List 16

5 Module Documentation **18**

5.1 Bicubic Warp API 18

5.1.1 Detailed Description 18

5.1.2 Function Documentation 18

5.2 Board 182 21

5.2.1 Detailed Description 21

5.2.2 Function Documentation 21

5.2.3 Variable Documentation 21

5.3 Camera Generic 22

5.3.1 Detailed Description 22

5.3.2 Function Documentation 22

5.4 Event Loop API 27

5.4.1 Detailed Description 28

5.4.2 Typedef Documentation 28

5.4.3 Enumeration Type Documentation 28

5.4.4 Function Documentation 28

5.5 Event Queue 33

5.5.1 Detailed Description 33

5.5.2 Typedef Documentation 33

5.5.3 Function Documentation . . . 34
5.6 I2C Slave API . . . 35
5.6.1 Detailed Description . . . 35
5.6.2 Function Documentation . . . 35
5.7 Image Warp . . . 37
5.7.1 Detailed Description . . . 37
5.7.2 Function Documentation . . . 37
5.8 JPEG Encoder API . . . 38
5.8.1 Detailed Description . . . 38
5.8.2 Macro Definition Documentation . . . 39
5.8.3 Enumeration Type Documentation . . . 39
5.8.4 Function Documentation . . . 39
5.9 JPEG Encoder Parallel API . . . 41
5.9.1 Detailed Description . . . 41
5.9.2 Macro Definition Documentation . . . 41
5.9.3 Enumeration Type Documentation . . . 42
5.9.4 Function Documentation . . . 42
5.10 LCD Generic API . . . 43
5.10.1 Detailed Description . . . 43
5.10.2 Function Documentation . . . 44
5.11 Leon IPC API . . . 50
5.11.1 Detailed Description . . . 50
5.11.2 Function Documentation . . . 50
5.12 Leon L1 Cache . . . 54
5.12.1 Detailed Description . . . 54
5.12.2 Function Documentation . . . 54
5.13 World Message Protocol API . . . 58
5.13.1 Detailed Description . . . 60
5.13.2 Macro Definition Documentation . . . 60
5.13.3 Typedef Documentation . . . 61
5.13.4 Enumeration Type Documentation . . . 61
5.13.5 Function Documentation . . . 61
5.13.6 Variable Documentation . . . 63
5.14 API . . . 64
5.14.1 Detailed Description . . . 64
5.14.2 Macro Definition Documentation . . . 64

5.14.3 Function Documentation . . . 65
5.14.4 Variable Documentation . . . 65
5.15 SPI Slave API . . . 66
5.15.1 Detailed Description . . . 66
5.15.2 Macro Definition Documentation . . . 66
5.15.3 Function Documentation . . . 67
5.15.4 Variable Documentation . . . 67
5.16 Opipe . . . 68
5.16.1 Detailed Description . . . 68
5.17 Pattern Generator API . . . 69
5.17.1 Detailed Description . . . 69
5.17.2 Function Documentation . . . 69
5.18 Unit Test API . . . 71
5.18.1 Detailed Description . . . 71
5.18.2 Function Documentation . . . 71
5.19 VCS Unit Test API . . . 76
5.19.1 Detailed Description . . . 77
5.19.2 Function Documentation . . . 77
5.20 VCS Test Environment API . . . 83
5.20.1 Detailed Description . . . 83
5.20.2 Function Documentation . . . 84
5.21 Debug Tracer . . . 90
5.21.1 Detailed Description . . . 90
5.21.2 Macro Definition Documentation . . . 90

6 Data Structure Documentation **91**

6.1 _SwLink Struct Reference . . . 91
6.1.1 Field Documentation . . . 91
6.2 AeAwbCfg Struct Reference . . . 92
6.2.1 Field Documentation . . . 92
6.3 AeAwbPatchStats Struct Reference . . . 93
6.3.1 Field Documentation . . . 93
6.4 AfCfg Struct Reference . . . 93
6.4.1 Field Documentation . . . 93
6.5 AfPatchStats Struct Reference . . . 94
6.5.1 Field Documentation . . . 94
6.6 BlcCfg Struct Reference . . . 95

6.6.1 Field Documentation . . . 95
6.7 ChromaDnsCfg Struct Reference . . . 95
6.7.1 Field Documentation . . . 95
6.8 ChromaGenCfg Struct Reference . . . 96
6.8.1 Field Documentation . . . 96
6.9 client_tx_frame_header_t Struct Reference . . . 97
6.9.1 Field Documentation . . . 98
6.10 ColCombCfg Struct Reference . . . 99
6.10.1 Field Documentation . . . 99
6.11 ColConvCfg Struct Reference . . . 99
6.11.1 Field Documentation . . . 99
6.12 ConvCfg Struct Reference . . . 100
6.12.1 Field Documentation . . . 100
6.13 DBuffer Struct Reference . . . 100
6.13.1 Detailed Description . . . 100
6.13.2 Field Documentation . . . 101
6.14 DbyrCfg Struct Reference . . . 101
6.14.1 Field Documentation . . . 102
6.15 DogCfg Struct Reference . . . 102
6.15.1 Field Documentation . . . 102
6.16 eventQueue_t Struct Reference . . . 103
6.16.1 Detailed Description . . . 103
6.16.2 Field Documentation . . . 103
6.17 eventQueueItem_t Struct Reference . . . 103
6.17.1 Detailed Description . . . 104
6.17.2 Field Documentation . . . 104
6.18 FeatMaintenance Class Reference . . . 105
6.18.1 Constructor & Destructor Documentation . . . 106
6.18.2 Member Function Documentation . . . 106
6.18.3 Field Documentation . . . 106
6.19 fmResourceCfg Struct Reference . . . 106
6.19.1 Field Documentation . . . 106
6.20 HarrisCfg Struct Reference . . . 107
6.20.1 Field Documentation . . . 107
6.21 ipipeServerInfo Struct Reference . . . 107
6.21.1 Field Documentation . . . 107

6.22 LscCfg Struct Reference . . . 108
6.22.1 Field Documentation . . . 108
6.23 LtmCfg Struct Reference . . . 108
6.23.1 Field Documentation . . . 108
6.24 LumaDnsCfg Struct Reference . . . 108
6.24.1 Field Documentation . . . 109
6.25 LumaDnsRefCfg Struct Reference . . . 109
6.25.1 Field Documentation . . . 109
6.26 LutCfg Struct Reference . . . 110
6.26.1 Field Documentation . . . 110
6.27 MBuffer Struct Reference . . . 110
6.27.1 Field Documentation . . . 110
6.28 MedianCfg Struct Reference . . . 111
6.28.1 Field Documentation . . . 111
6.29 MessageRingBuffer Struct Reference . . . 112
6.30 mestHandler_t Struct Reference . . . 112
6.30.1 Field Documentation . . . 112
6.31 MipiRxCfg Struct Reference . . . 112
6.31.1 Field Documentation . . . 113
6.32 ofResourceCfg Struct Reference . . . 113
6.32.1 Field Documentation . . . 113
6.33 oMipiTxLoopbackParam Struct Reference . . . 113
6.33.1 Detailed Description . . . 114
6.33.2 Field Documentation . . . 114
6.34 OpipeGlobal Struct Reference . . . 115
6.34.1 Field Documentation . . . 115
6.35 OpipeS Struct Reference . . . 116
6.35.1 Detailed Description . . . 119
6.35.2 Field Documentation . . . 119
6.36 OpticalFlow Class Reference . . . 125
6.36.1 Constructor & Destructor Documentation . . . 126
6.36.2 Member Function Documentation . . . 126
6.36.3 Field Documentation . . . 126
6.37 OsVirtualChannel Struct Reference . . . 127
6.38 PBuffer Struct Reference . . . 127
6.38.1 Field Documentation . . . 127

6.39 PhysicalChannel Struct Reference . . . 127
6.40 PixelPipe Class Reference . . . 128
6.40.1 Constructor & Destructor Documentation . . . 129
6.40.2 Member Function Documentation . . . 129
6.40.3 Field Documentation . . . 130
6.41 PlgFifoElemS Struct Reference . . . 130
6.41.1 Field Documentation . . . 130
6.42 PlgFifoS Struct Reference . . . 131
6.42.1 Field Documentation . . . 131
6.43 PlgIspFullStruct Struct Reference . . . 131
6.43.1 Field Documentation . . . 132
6.44 PlgSource Struct Reference . . . 133
6.44.1 Field Documentation . . . 133
6.45 PlgSrcIsp Struct Reference . . . 134
6.45.1 Field Documentation . . . 135
6.46 PluginServerCtrl Struct Reference . . . 135
6.46.1 Field Documentation . . . 135
6.47 ppThresholds_t Struct Reference . . . 136
6.47.1 Field Documentation . . . 136
6.48 RawCfg Struct Reference . . . 136
6.48.1 Field Documentation . . . 137
6.49 RectRgn Struct Reference . . . 137
6.49.1 Field Documentation . . . 138
6.50 SBuffer Struct Reference . . . 138
6.50.1 Field Documentation . . . 138
6.51 send_out_tx_buffer_header_t Struct Reference . . . 139
6.51.1 Member Function Documentation . . . 139
6.51.2 Field Documentation . . . 139
6.52 SendOutElement_t Struct Reference . . . 139
6.52.1 Field Documentation . . . 140
6.53 SendOutInitCfg_t Struct Reference . . . 140
6.53.1 Field Documentation . . . 140
6.54 SharpenCfg Struct Reference . . . 140
6.54.1 Field Documentation . . . 141
6.55 SigmaDnsCfg Struct Reference . . . 141
6.55.1 Field Documentation . . . 141

6.56 SourceServerCtrlT Struct Reference . . . 142
6.56.1 Field Documentation . . . 142
6.57 spiSlaveCommunicationConfiguration_t Struct Reference . . . 142
6.58 t_ppFifoCfg Struct Reference . . . 143
6.58.1 Field Documentation . . . 143
6.59 t_pPipeResourceCfg Struct Reference . . . 143
6.59.1 Field Documentation . . . 143
6.60 t_pPipeShaveConfig Struct Reference . . . 144
6.60.1 Field Documentation . . . 144
6.61 TrigerCaptQue Struct Reference . . . 144
6.61.1 Field Documentation . . . 144
6.62 TriggerCaptElement Struct Reference . . . 145
6.62.1 Field Documentation . . . 145
6.63 UpfirdnCfg Struct Reference . . . 145
6.63.1 Field Documentation . . . 145
6.64 VirtualChannel Struct Reference . . . 146

7 File Documentation **147**

7.1 aeApi.h File Reference . . . 147
7.1.1 Detailed Description . . . 147
7.1.2 Function Documentation . . . 147
7.2 bicubicWarpApi.h File Reference . . . 148
7.2.1 Detailed Description . . . 148
7.3 Board182Api.h File Reference . . . 148
7.3.1 Detailed Description . . . 148
7.4 CamGenericApi.h File Reference . . . 149
7.4.1 Detailed Description . . . 149
7.5 dbgTracerApi.h File Reference . . . 149
7.5.1 Detailed Description . . . 150
7.6 disparityMapApi.h File Reference . . . 150
7.6.1 Function Documentation . . . 150
7.7 eventLoop.h File Reference . . . 150
7.7.1 Detailed Description . . . 151
7.8 eventQueue.h File Reference . . . 151
7.8.1 Detailed Description . . . 152
7.9 featureMaintenanceApi.h File Reference . . . 152
7.9.1 Detailed Description . . . 153

7.9.2 Typedef Documentation . . . 153
7.9.3 Function Documentation . . . 153
7.9.4 Variable Documentation . . . 153
7.10 fifoCommApi.h File Reference . . . 154
7.10.1 Detailed Description . . . 154
7.10.2 Function Documentation . . . 154
7.11 fifoCommInitApi.h File Reference . . . 157
7.11.1 Detailed Description . . . 157
7.11.2 Function Documentation . . . 158
7.12 FrameMgrApi.h File Reference . . . 160
7.12.1 Function Documentation . . . 160
7.13 I2CSlaveApi.h File Reference . . . 161
7.13.1 Detailed Description . . . 161
7.14 imageWarp.h File Reference . . . 161
7.14.1 Detailed Description . . . 161
7.14.2 Function Documentation . . . 162
7.15 imageWarp.h File Reference . . . 162
7.15.1 Detailed Description . . . 162
7.15.2 Function Documentation . . . 162
7.16 imageWarp.h File Reference . . . 163
7.16.1 Detailed Description . . . 163
7.17 IpipeServerApi.h File Reference . . . 163
7.17.1 Macro Definition Documentation . . . 164
7.17.2 Function Documentation . . . 164
7.17.3 Variable Documentation . . . 165
7.18 JpegEncoderApi.h File Reference . . . 166
7.18.1 Detailed Description . . . 166
7.19 JpegEncoderApi.h File Reference . . . 166
7.19.1 Detailed Description . . . 167
7.20 LcdApi.h File Reference . . . 167
7.20.1 Detailed Description . . . 168
7.21 LeonIPCApi.h File Reference . . . 168
7.21.1 Detailed Description . . . 169
7.22 LeonL1CacheApi.h File Reference . . . 169
7.22.1 Detailed Description . . . 170
7.23 MDKdox-Components-intro.txt File Reference . . . 170

7.24 memManagerApi.h File Reference . . . 170
7.24.1 Enumeration Type Documentation . . . 170
7.24.2 Function Documentation . . . 171
7.25 MemMgrApi.h File Reference . . . 172
7.25.1 Macro Definition Documentation . . . 172
7.25.2 Typedef Documentation . . . 172
7.25.3 Function Documentation . . . 172
7.26 MessageProtocol.h File Reference . . . 172
7.26.1 Detailed Description . . . 174
7.27 MessRingBuff.h File Reference . . . 174
7.27.1 Detailed Description . . . 174
7.28 MestApi.h File Reference . . . 174
7.28.1 Detailed Description . . . 175
7.28.2 Macro Definition Documentation . . . 176
7.28.3 Function Documentation . . . 177
7.28.4 Variable Documentation . . . 180
7.29 MipiSendApi.h File Reference . . . 180
7.29.1 Function Documentation . . . 180
7.30 MipiSendApi.h File Reference . . . 180
7.30.1 Function Documentation . . . 180
7.31 Opipe.h File Reference . . . 181
7.31.1 Detailed Description . . . 181
7.31.2 Function Documentation . . . 182
7.32 OpipeBlocks.h File Reference . . . 186
7.32.1 Detailed Description . . . 187
7.33 OpipeDefs.h File Reference . . . 187
7.33.1 Detailed Description . . . 188
7.33.2 Macro Definition Documentation . . . 189
7.33.3 Typedef Documentation . . . 190
7.33.4 Function Documentation . . . 190
7.34 opticalFlowApi.h File Reference . . . 190
7.34.1 Detailed Description . . . 191
7.34.2 Macro Definition Documentation . . . 191
7.34.3 Typedef Documentation . . . 191
7.34.4 Function Documentation . . . 192
7.34.5 Variable Documentation . . . 192

7.35 OsDrvSpiSlaveCP.h File Reference . . . 192
7.35.1 Detailed Description . . . 193
7.36 OsMessageProtocol.h File Reference . . . 193
7.36.1 Detailed Description . . . 194
7.37 PatternGeneratorApi.h File Reference . . . 194
7.37.1 Detailed Description . . . 195
7.38 PixelPipeApi.h File Reference . . . 195
7.38.1 Detailed Description . . . 196
7.38.2 Macro Definition Documentation . . . 196
7.38.3 Typedef Documentation . . . 196
7.38.4 Function Documentation . . . 196
7.38.5 Variable Documentation . . . 197
7.39 PlgFifoApi.h File Reference . . . 198
7.39.1 Macro Definition Documentation . . . 198
7.39.2 Typedef Documentation . . . 198
7.39.3 Function Documentation . . . 198
7.40 PlgIspFullApi.h File Reference . . . 198
7.40.1 Typedef Documentation . . . 199
7.40.2 Enumeration Type Documentation . . . 199
7.40.3 Function Documentation . . . 199
7.41 PlgSourceApi.h File Reference . . . 199
7.41.1 Enumeration Type Documentation . . . 200
7.41.2 Function Documentation . . . 200
7.42 PlgSrcIspApi.h File Reference . . . 200
7.42.1 Enumeration Type Documentation . . . 201
7.42.2 Function Documentation . . . 201
7.43 sendOutApi.h File Reference . . . 202
7.43.1 Typedef Documentation . . . 202
7.43.2 Function Documentation . . . 202
7.43.3 Variable Documentation . . . 203
7.44 sendOutApi.h File Reference . . . 203
7.44.1 Typedef Documentation . . . 203
7.44.2 Function Documentation . . . 203
7.44.3 Variable Documentation . . . 204
7.45 SuspendLrtLpApi.h File Reference . . . 204
7.45.1 Detailed Description . . . 204

7.45.2 Macro Definition Documentation 204
7.45.3 Function Documentation 205
7.46 testUtilsApi.h File Reference 205
7.46.1 Detailed Description 205
7.46.2 Function Documentation 205
7.47 UnitTestApi.h File Reference 208
7.47.1 Detailed Description 209
7.48 UnitTestApi.h File Reference 209
7.48.1 Detailed Description 210
7.49 VcsHooksApi.h File Reference 210
7.49.1 Detailed Description 211

Index **212**

Movidius

Chapter 1

Introduction

Components are software modules that are conditionally compiled. This document documents the API for each of these modules

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

API 64
Bicubic Warp API 18
Board 182 21
Camera Generic 22
Debug Tracer 90
Event Loop API 27
Event Queue 33
I2C Slave API 35
Image Warp 37
JPEG Encoder API 38
JPEG Encoder Parallel API 41
LCD Generic API 43
Leon IPC API 50
Leon L1 Cache 54
Opipe 68
Pattern Generator API 69
SPI Slave API 66
Unit Test API 71
VCS Test Environment API 83
VCS Unit Test API 76
World Message Protocol API 58

Movidius

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

_SwLink . . . 91
AeAwbCfg . . . 92
AeAwbPatchStats . . . 93
AfCfg . . . 93
AfPatchStats . . . 94
BlcCfg . . . 95
ChromaDnsCfg . . . 95
ChromaGenCfg . . . 96
client_tx_frame_header_t . . . 97
ColCombCfg . . . 99
ColConvCfg . . . 99
ConvCfg . . . 100
DBuffer . . . 100
DbyrCfg . . . 101
DogCfg . . . 102
eventQueue_t
    An event queue type . . . 103
eventQueueItem_t
    An event queue item type . . . 103
FeatMaintenance . . . 105
fmResourceCfg . . . 106
HarrisCfg . . . 107
ipipeServerInfo . . . 107
LscCfg . . . 108
LtmCfg . . . 108
LumaDnsCfg . . . 108
LumaDnsRefCfg . . . 109
LutCfg . . . 110
MBuffer . . . 110
MedianCfg . . . 111
MessageRingBuffer . . . 112
mestHandler_t . . . 112
MipiRxCfg . . . 112

ofResourceCfg . . . 113
oMipiTxLoopbackParam
Mipi-TX loopback debug params . . . 113
OpipeGlobal . . . 115
OpipeS
Main O-Pipe data struct . . . 116
OpticalFlow . . . 125
OsVirtualChannel . . . 127
PBuffer . . . 127
PhysicalChannel . . . 127
PixelPipe . . . 128
PlgFifoElemS . . . 130
PlgFifoS . . . 131
PlgIspFullStruct . . . 131
PlgSource . . . 133
PlgSrcIsp . . . 134
PluginServerCtrl . . . 135
ppThresholds_t . . . 136
RawCfg . . . 136
RectRgn . . . 137
SBuffer . . . 138
send_out_tx_buffer_header_t . . . 139
SendOutElement_t . . . 139
SendOutInitCfg_t . . . 140
SharpenCfg . . . 140
SigmaDnsCfg . . . 141
SourceServerCtrlT . . . 142
spiSlaveCommunicationConfiguration_t . . . 142
t_ppFifoCfg . . . 143
t_pPipeResourceCfg . . . 143
t_pPipeShaveConfig . . . 144
TrigerCaptQue . . . 144
TriggerCaptElement . . . 145
UpfirdnCfg . . . 145
VirtualChannel . . . 146

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

aeApi.h . . . 147
bicubicWarpApi.h . . . 148
Board182Api.h . . . 148
CamGenericApi.h . . . 149
dbgTracerApi.h . . . 149
disparityMapApi.h . . . 150
eventLoop.h . . . 150
eventQueue.h . . . 151
featureMaintenanceApi.h . . . 152
fifoCommApi.h
    Application configuration Leon header . . . 154
fifoCommInitApi.h
    Shave Fifo communication init functions . . . 157
FrameMgrApi.h . . . 160
I2CSlaveApi.h . . . 161
shave/imageWarp.h
    Image warp component API . . . 161
yn/compShvDynApps/imageWarpDynlib/imageWarp.h
    Image warp component API . . . 162
yn/leon/imageWarp.h . . . 163
IpipeServerApi.h . . . 163
JpegM2EncoderDyn/leon/JpegEncoderApi.h . . . 166
JpegM2EncoderParallel/leon/JpegEncoderApi.h . . . 166
LcdApi.h . . . 167
LeonIPCApi.h . . . 168
LeonL1CacheApi.h . . . 169
memManagerApi.h . . . 170
MemMgrApi.h . . . 172
MessageProtocol.h . . . 172
MessRingBuff.h . . . 174
MestApi.h
    MEST Component Functions API . . . 174
MipiSend/leon/MipiSendApi.h . . . 180

pipe2/MipiSend/leon/MipiSendApi.h . . . 180
Opipe.h
Opipe - API . . . 181
OpipeBlocks.h
Opipe - SIPP blocks config data structs . . . 186
OpipeDefs.h
Opipe - definitions . . . 187
opticalFlowApi.h . . . 190
OsDrvSpiSlaveCP.h . . . 192
OsMessageProtocol.h . . . 193
PatternGeneratorApi.h . . . 194
PixelPipeApi.h . . . 195
PlgFifoApi.h . . . 198
PlgIspFullApi.h . . . 198
PlgSourceApi.h . . . 199
PlgSrcIspApi.h . . . 200
common/leon/sendOutApi.h . . . 202
pipe2/common/leon/sendOutApi.h . . . 203
SuspendLrtLpApi.h
Suspend LRT component: API definitions . . . 204
testUtilsApi.h
Software test library . . . 205
leon/UnitTestApi.h . . . 208
cs/leon/UnitTestApi.h . . . 209
VcsHooksApi.h . . . 210

Chapter 5

Module Documentation

5.1 Bicubic Warp API

Bicubic Warp API.

Functions

- int bicubicWarpInit (bicubicWarpContext *ctx)
  Initialize bicubic block.
- int bicubicWarpProcessFrame (bicubicWarpContext *ctx)
  Run bicubic block.
- void bicubicWarpGenerateMeshRT (bicubicWarpContext *ctx)
  Generate rectified coordinates based on rotation and translation (RT) relative to the center.
- void bicubicWarpGenerateMeshHomographyRTP (bicubicWarpContext *ctx)
  Generate rectified coordinates based on the homography (OpenCV style)
- void bicubicWarpGenerateMeshFromLUTMaps (bicubicWarpContext *ctx)
  Generate rectified coordinates from LUT maps.

5.1.1 Detailed Description

Bicubic Warp API. This is the functions API for the Bicubic Warp component

5.1.2 Function Documentation

void bicubicWarpGenerateMeshFromLUTMaps (bicubicWarpContext * ctx)

Generate rectified coordinates from LUT maps.

Parameters

in	*ctx*	- pointer to bicubicWarpContext

Returns

int

void bicubicWarpGenerateMeshHomographyRTP (bicubicWarpContext * ctx)

Generate rectified coordinates based on the homography (OpenCV style)

Parameters

in	*ctx*	- pointer to bicubicWarpContext

Returns

int

void bicubicWarpGenerateMeshRT (bicubicWarpContext * ctx)

Generate rectified coordinates based on rotation and translation (RT) relative to the center.

Parameters

in	*ctx*	- pointer to bicubicWarpContext

Returns

int

int bicubicWarpInit (bicubicWarpContext * ctx)

Initialize bicubic block.

Parameters

in	*ctx*	- pointer to bicubicWarpContext

Returns

int

int bicubicWarpProcessFrame (bicubicWarpContext * ctx)

Run bicubic block.

Parameters

in	*ctx*	- pointer to bicubicWarpContext

Returns

int

5.2 Board 182

Board Setup Functions API.

Functions

- s32 BoardInitialise (u32 clockConfiguration)

  This function performs the initialization of basic functions of MV0182 board: I2C buses, external clock generator and sets up all GPIOS.

Variables

- tyAppDeviceHandles gAppDevHndls

5.2.1 Detailed Description

Board Setup Functions API. This is the API to board setup library implementation.

5.2.2 Function Documentation

s32 BoardInitialise (u32 clockConfiguration)

This function performs the initialization of basic functions of MV0182 board: I2C buses, external clock generator and sets up all GPIOS.

Parameters

`in`	*clock-Configuration*	- External PLL clock configuration

Returns

initialization status

5.2.3 Variable Documentation

tyAppDeviceHandles gAppDevHndls

5.3 Camera Generic

Generic Camera API.

Functions

- camErrorType CamInit (GenericCameraHandle *hndl, GenericCamSpec *camSpec, CamUserSpec *userSpec, callbacksListStruct *cbStruct, I2CM_Device *pI2cHandle)

  This will initialize the camera component (the sensor and all the myriad components connected in the frames data path) for a specific sensor.
- camErrorType CamStart (GenericCameraHandle *hndl)

  This will start the sensor and the interrupts system.
- camErrorType CamStandby (GenericCameraHandle *hndl, camStatus_type standbyType)

  This will put the sensor and related myriad logic in a standby mode.
- camErrorType CamWakeup (GenericCameraHandle *hndl)

  This will wake up the sensor and related myriad logic from the standby mode.
- camErrorType CamStop (GenericCameraHandle *hndl)

  Resets the camera component (the sensor and the related myriad logic)
- camErrorType CamSetupCallbacks (GenericCameraHandle *hndl, sensorCallbacksListType *cbList)

  Set or change the existing callbacks used for the sensor configuration.
- camErrorType CamSetupInterrupts (GenericCameraHandle *hndl, camIsrType managedInterrupt, u32 notifiedInterrupts, u32 clearedInterrupts, interruptsCallbacksListType *cbList, u32 interruptsLevel, u32 routeLineInterrupt, u32 routeFrameInterrupt)

  Set or change the existing/the default interrupt events for a camera module.
- unsigned int CamGetFrameCounter (GenericCameraHandle *hndl)

  Get the counter of frames/lines (depending on the managed ISR types) received inside CIF block of myriad component (always 0 for SIPP receivers)

5.3.1 Detailed Description

Generic Camera API. This is the API for generic camera component

5.3.2 Function Documentation

unsigned int CamGetFrameCounter (GenericCameraHandle * hndl)

Get the counter of frames/lines (depending on the managed ISR types) received inside CIF block of myriad component (always 0 for SIPP receivers)

Parameters

hndl	- Pointer to the camera handle

Returns

Frame number

camErrorType CamInit (GenericCameraHandle * hndl, GenericCamSpec * camSpec, CamUserSpec * userSpec, callbacksListStruct * cbStruct, I2CM_Device * pI2cHandle)

This will initialize the camera component (the sensor and all the myriad components connected in the frames data path) for a specific sensor.

Parameters

in	*hndl*	- Pointer to the empty camera handle which will be filled back with all camera configurations at return (pseudo return parameter)
in	*camSpec*	- Fixed camera configuration
in	*userSpec*	- Dynamic camera configuration
in	*cbStruct*	- structured list of pointers to all user callback functions which will be called by CamGeneric
in	*pI2cHandle*	- I2C handle, NULL if no default I2C configuration need to be performed by CamGeneric

Returns

camErrorType - CAM_SUCCESS if camera initialization was OK, other values if it failed

camErrorType CamSetupCallbacks (GenericCameraHandle * hndl, sensorCallbacksListType * cbList)

Set or change the existing callbacks used for the sensor configuration.

Parameters

in	*hndl*	- Pointer to the camera handle
in	*cbList*	- List of callbacks pointers

Returns

camErrorType - CAM_SUCCESS if the camera callbacks list was updated OK, other values if it failed

camErrorType CamSetupInterrupts (GenericCameraHandle * hndl, camIsrType managedInterrupt, u32 notifiedInterrupts, u32 clearedInterrupts, interruptsCallbacksListType * cbList, u32 interruptsLevel, u32 routeLineInterrupt, u32 routeFrameInterrupt)

Set or change the existing/the default interrupt events for a camera module.

Parameters

in	*hndl*	- Pointer to the camera handle
in	*managed-Interrupt*	- The event on which CamGeneric will perform local HW management (DMA restart, ISRs clearing) (values: see camIsrType) and will call the allocation cbf
in	*notified-Interrupts*	- The event(s) on which CamGeneric will call the notification cbf (values: combined values of camIsrType type)
in	*cleared-Interrupts*	- The event(s) which will be cleared locally in in CamGeneric ISR handler (values: combined values of camIsrType type)

in	*cbList*	- The list of callbacks to be used by the ISR events
in	*interruptsLevel*	- The level of priority to be assigned to the camera interrupt; values between 0 - 15 (not recommended)
in	*routeLine-Interrupt*	- The id of the rerouted line interrupt, in case CamGeneric runs on Leon OS (values: 0 = default, IRQ_DYNAMIC_0 .. IRQ_DYNAMIC_11, seeDrvIcbDefines.h) For CIF receivers, this must have same value as the routeFrameInterrupt
in	*routeFrame-Interrupt*	- The id of the rerouted frame interrupt, in case CamGeneric runs on Leon OS (values: 0 = default, IRQ_DYNAMIC_0 .. IRQ_DYNAMIC_11, seeDrvIcbDefines.h)

Returns

camErrorType CAM_SUCCESS if the camera interrupts registers were updated OK, other values if it failed

camErrorType CamStandby (GenericCameraHandle * hndl, camStatus_type standbyType)

This will put the sensor and related myriad logic in a standby mode.

Parameters

in	*hndl*	- Pointer to the camera handle
in	*standbyType*	- The standby type to be performed: COLD or HOT

Returns

camErrorType - CAM_SUCCESS if camera standby passing was OK, other values if it failed

camErrorType CamStart (GenericCameraHandle * hndl)

This will start the sensor and the interrupts system.

Parameters

in	*hndl*	- Pointer to the camera handle

Returns

camErrorType - CAM_SUCCESS if camera started OK, other values if it failed

camErrorType CamStop (GenericCameraHandle * hndl)

Resets the camera component (the sensor and the related myriad logic)

Parameters

in	*hndl*	- Pointer to the camera handle

Returns

camErrorType - CAM_SUCCESS if camera was stopped OK, other values if it failed

camErrorType CamWakeup (GenericCameraHandle * hndl)

This will wake up the sensor and related myriad logic from the standby mode.

Parameters

in	*hndl*	- Pointer to the camera handle

Returns

camErrorType - CAM_SUCCESS if camera wake up was OK, other values if it failed

5.4 Event Loop API

Event Loop functions API.

Typedefs

- typedef s32(∗ eventLoopCallback_t)(eventQueueItem_t ∗event)

  Type of event loop callback for each event type;.
- typedef enum eventState_t eventState_t

  Enum of different states of an event.

Enumerations

- enum eventState_t { BUSY, DONE, REQUIRED, DROPPED }

  Enum of different states of an event.

Functions

- void eventLoopInit ()

  Initialize the event loop queuese.
- s32 eventLoopSetCallback (eventType_t event, eventLoopCallback_t cb)

  Add an event loop callback for an event type.
- void eventLoopPushCmd (eventQueueItem_t ∗event)

  Add an event to the command queue.
- void eventLoopPush (eventQueueItem_t ∗event)

  Add an event to the event loop.
- void eventLoopRun ()

  Run the event loop.
- void eventLoopReset ()

  Clear the event loop.
- void eventLoopStartTimer ()

  Start event loop timer.
- void eventLoopSetEventState (eventState_t evState, eventQueueItem_t ∗event, eventLoopCallback_t evCallbackPtr)

  Set the event state.
- void eventLoopClearBusyFlag (eventQueueItem_t ∗event, eventLoopCallback_t evCallbackPtr)

  Clear the busy callback flag.
- void eventLoopChangeEventType (eventQueueItem_t ∗event, eventType_t newEventType)

  Change the event type to newEventType.
- void eventLoopDropEvent (eventQueueItem_t ∗event, eventLoopCallback_t evCallbackPtr)

  Drop an event from the loop.
- void eventLoopSetReleaseCallback (eventType_t event, eventLoopCallback_t onReleaseCb)

  Set event done callback.
- void eventLoopReleaseEvent (eventQueueItem_t ∗event, eventLoopCallback_t evCallbackPtr)

  Mark event done or free it, depending on the event state.

5.4.1 Detailed Description

Event Loop functions API. API to implement simple scheduler in order to avoid interrupts on chip

5.4.2 Typedef Documentation

typedef s32(∗ eventLoopCallback_t)(**eventQueueItem_t** ∗event)

Type of event loop callback for each event type;.

typedef enum **eventState_t eventState_t**

Enum of different states of an event.

5.4.3 Enumeration Type Documentation

enum **eventState_t**

Enum of different states of an event.

Enumerator

BUSY
DONE
REQUIRED
DROPPED

5.4.4 Function Documentation

void eventLoopChangeEventType (**eventQueueItem_t** ∗ event, eventType_t newEventType)

Change the event type to newEventType.

Parameters

in	*event*	- event queue item
in	*newEventType*	- event type to which the event item should be changed

Note

Changing the event type is a bad practice. This function is usually used to avoid event copying (if the data is big) Use this function only if you know the event loop well.

Returns

void

void eventLoopClearBusyFlag (**eventQueueItem_t** ∗ event, **eventLoopCallback_t** evCallbackPtr)

Clear the busy callback flag.

Parameters

in	*event*	- event queue item
in	*evCallbackPtr*	- pointer to event loop callback

Returns

void

void eventLoopDropEvent (**eventQueueItem_t** * event, **eventLoopCallback_t** evCallbackPtr)

Drop an event from the loop.

Parameters

in	*event*	- event queue item
in	*evCallbackPtr*	- pointer to event loop callback

Returns

void

void eventLoopInit ( )

Initialize the event loop queuese.

Returns

void

void eventLoopPush (**eventQueueItem_t** * event)

Add an event to the event loop.

Parameters

in	*event*	- event queue item

Returns

void

void eventLoopPushCmd (**eventQueueItem_t** * event)

Add an event to the command queue.

Parameters

in	*event*	- event queue item

Returns

void

void eventLoopReleaseEvent (**eventQueueItem_t** * event, **eventLoopCallback_t** evCallbackPtr)

Mark event done or free it, depending on the event state.

Parameters

in	*event*	- event queue item
in	*evCallbackPtr*	- pointer to event loop callback

Returns

void

void eventLoopReset ( )

Clear the event loop.

Returns

void

void eventLoopRun ( )

Run the event loop.

Returns

void

s32 eventLoopSetCallback (eventType_t event, **eventLoopCallback_t** cb)

Add an event loop callback for an event type.

Parameters

in	*event*	- type of event
in	*cb*	- Type of event loop callback for each event type

Returns

1

void eventLoopSetEventState (**eventState_t** evState, **eventQueueItem_t** * event, **eventLoopCallback_t** evCallbackPtr)

Set the event state.

Parameters

in	*evState*	- event state to be set
in	*event*	- event whose state is to be changed
in	*evCallbackPtr*	- pointer to event loop callback

Returns

void

void eventLoopSetReleaseCallback (eventType_t event, **eventLoopCallback_t** onReleaseCb)

Set event done callback.

Parameters

in	*event*	- event queue item
in	*onReleaseCb*	- pointer to event loop callback

Returns

void

void eventLoopStartTimer ( )

Start event loop timer.

Returns

void

5.5 Event Queue

Event Queue functions API.

Data Structures

- struct eventQueueItem_t

  An event queue item type.
- struct eventQueue_t

  An event queue type.

Typedefs

- typedef struct eventQueueItem_t eventQueueItem_t

  An event queue item type.
- typedef struct eventQueue_t eventQueue_t

  An event queue type.

Functions

- u64 eventLoopTimestamp ()

  Get system timestamp in clock-ticks.
- void eventQueueInit (eventQueue_t *self)

  Initialize an event queue.
- void eventQueuePush (eventQueue_t *self, eventQueueItem_t *item)

  Push a new item on the event queue param[in] self - event queue to which will be added a new item param[in] item - new event queue item.
- eventQueueItem_t * eventQueuePop (eventQueue_t *self)

  Get item of the head of list and remove it from list.
- void eventQueueReturnToParent (eventQueueItem_t *item)

  Return item to parent queue.

5.5.1 Detailed Description

Event Queue functions API. Event Queue API and structs to aid the event loop component

5.5.2 Typedef Documentation

typedef struct **eventQueue_t eventQueue_t**

An event queue type.

typedef struct **eventQueueItem_t eventQueueItem_t**

An event queue item type.

5.5.3 Function Documentation

u64 eventLoopTimestamp ( )

Get system timestamp in clock-ticks.

void eventQueueInit (**eventQueue_t** * self)

Initialize an event queue.

Parameters

in	*self*	- event queue to be initialised

Returns

void

eventQueueItem_t* eventQueuePop (**eventQueue_t** * self)

Get item of the head of list and remove it from list.

Parameters

in	*self*	- event queue whose head is to be removed

Returns

Head of the event queue

void eventQueuePush (**eventQueue_t** * self, **eventQueueItem_t** * item)

Push a new item on the event queue param[in] self - event queue to which will be added a new item param[in] item - new event queue item.

Returns

void

void eventQueueReturnToParent (**eventQueueItem_t** * item)

Return item to parent queue.

Parameters

in	*item*	- event queue item to be added back to the parent queue

Returns

void

5.6 I2C Slave API

I2C Slave Functions API.

Functions

- s32 I2CSlaveInit (i2cSlaveHandle_t *hndl, i2cSlaveAddrCfg_t *config)
- void I2CSlaveSetupCallbacks (i2cSlaveHandle_t *hndl, i2cReadAction *cbReadAction, i2cWriteAction *cbWriteAction)

5.6.1 Detailed Description

I2C Slave Functions API.

Component Usage

In order to use the component the following steps are ought to be done:

1. Declare a variable of i2cSlaveAddrCfg_t type
2. Initialize the members of the variable above declared
3. Declare a variable of i2cSlaveHandle_t type (handler)
4. Initialize a member of the handler declared, i2cConfig_t
5. Call I2CSlaveSetupCallbacks() function, having as parameters address of handler declared, result of i2cRead function, result of i2cRead function
6. Call I2CSlaveInit() function, having as parameters the address of the handler declared, I2C block, and the address of I2CSlaveAddrCfg variable

i2cReadAction() callback is called every time a RD_REQ interrupt is triggered, meaning that the master has issued a read request and it is waiting for data. The function can be used to prepare the data to be send to the master.

i2cWriteAction() callback is called when the slave has finished reading the information sent by the master and also a stop bit has been received, signaling that the master has finished the transfer. It provides the data that was received from master.

5.6.2 Function Documentation

s32 I2CSlaveInit (i2cSlaveHandle_t * hndl, i2cSlaveAddrCfg_t * config)

This will initialize the I2C Slave Component

Parameters

in	*hndl*	- pointer to the I2C slave handler
in	*config*	- pointer to I2CSlaveAddr configuration

Returns

s32 - 0 - on success

void I2CSlaveSetupCallbacks (i2cSlaveHandle_t * hndl, i2cReadAction * cbReadAction, i2cWriteAction * cbWriteAction)

Set callback functions for the I2C component

Parameters

in	*hndl*	- pointer to I2C slave handler
in	*cbReadAction*	- pointer to read function handler
in	*cbWriteAction*	- pointer to write function handler

Returns

void

5.7 Image Warp

Component used to warp a frameBuffer dynamically.

Functions

- u32 IMGWARP_start (swcShaveUnit_t svu, meshStruct *mesh, tileList *tileNodes, frameBuffer *inputFb, frameBuffer *outputFb, unsigned short padding)

  Sets up and launches one dynamic image warp on a specifically requested SHAVE.
- void IMGWARP_cleanup (void)

  Cleans up and prepare already used shave for running other dynamic apps.

5.7.1 Detailed Description

Component used to warp a frameBuffer dynamically.

5.7.2 Function Documentation

void IMGWARP_cleanup (void)

Cleans up and prepare already used shave for running other dynamic apps.

Returns

void

u32 IMGWARP_start (**swcShaveUnit_t** svu, meshStruct * mesh, tileList * tileNodes, frameBuffer * inputFb, frameBuffer * outputFb, unsigned short padding)

Sets up and launches one dynamic image warp on a specifically requested SHAVE.

Parameters

in	*svu*	- shave index to use
in	*mesh*	- pointer to the mesh to use
in	*tileNodes*	- pointer to the tileNodes to use
in	*inputFb*	- input framebuffer to process
in	*outputFb*	- output buffer that will be written with the result
in	*padding*	- outside mesh padding

Returns

operation status - 0 if success, error code otherwise (to be enhanced in the future)

5.8 JPEG Encoder API

JPEG Encoder Functions API.

Macros

- #define PARTITION_0 (0)
- #define PARTITION_1 (1)
- #define PARTITION_2 (2)
- #define PARTITION_3 (3)
- #define PARTITION_4 (4)
- #define PARTITION_5 (5)
- #define PARTITION_6 (6)
- #define PARTITION_7 (7)
- #define PARTITION_8 (8)
- #define PARTITION_9 (9)
- #define PARTITION_10 (10)
- #define PARTITION_11 (11)

Enumerations

- enum { JPEG_420_PLANAR, JPEG_422_PLANAR, JPEG_444_PLANAR }

Functions

- u32 JPEG_encode (frameBuffer imgInfo, u8 *outputLocal, u32 shvNo, u32 inBuffSizeShave, int jpegFormat)

  The JPEG encoding algorithm.

5.8.1 Detailed Description

JPEG Encoder Functions API. Jpeg Encoder functions API.

This is the API to a simple JPEG library implementing a jpeg encoding process.

Component Usage

In order to use the component the following steps are ought to be done:

1. User should declare variables for the 3 image planes:
   - extern unsigned char img_Y;
   - extern unsigned char img_U;
   - extern unsigned char img_V;
2. User should configure the image, by using a variable of jpegFrameSpec type, which is a structure whose fields are the 3 image planes, and information about image width and height
3. User should start applying the encode algorithm on the image, by calling JPEG_encode function
4. User can verify the processing result, by making a memory dump from "jpeg_buff", having the size equal to "outbytes"

5.8.2 Macro Definition Documentation

#define PARTITION_0 (0)

#define PARTITION_1 (1)

#define PARTITION_10 (10)

#define PARTITION_11 (11)

#define PARTITION_2 (2)

#define PARTITION_3 (3)

#define PARTITION_4 (4)

#define PARTITION_5 (5)

#define PARTITION_6 (6)

#define PARTITION_7 (7)

#define PARTITION_8 (8)

#define PARTITION_9 (9)

5.8.3 Enumeration Type Documentation

anonymous enum

Enumerator

JPEG_420_PLANAR

JPEG_422_PLANAR

JPEG_444_PLANAR

5.8.4 Function Documentation

u32 JPEG_encode (frameBuffer imgInfo, u8 * outputLocal, u32 shvNo, u32 inBuffSizeShave, int jpegFormat)

The JPEG encoding algorithm.

Parameters

in	*imgInfo*	- structure that contains info about input image
out	*output*	- pointer to the output buffer
in	*shaveNo*	- number of shaves to be used

in	*jpegFormat*	- JPEG_420_PLANAR or JPEG_422_PLANAR or JPEG_444_PLANAR

Returns

u32 length of output buffer

5.9 JPEG Encoder Parallel API

JPEG Encoder Parallel functions API.

Macros

- #define PARTITION_0 (0)
- #define PARTITION_1 (1)
- #define PARTITION_2 (2)
- #define PARTITION_3 (3)
- #define PARTITION_4 (4)
- #define PARTITION_5 (5)
- #define PARTITION_6 (6)
- #define PARTITION_7 (7)
- #define PARTITION_8 (8)
- #define PARTITION_9 (9)
- #define PARTITION_10 (10)
- #define PARTITION_11 (11)

Enumerations

- enum { JPEG_420_PLANAR, JPEG_422_PLANAR, JPEG_444_PLANAR }

Functions

- u32 JPEG_encode (frameBuffer imgInfo, u8 *outputLocal, u32 shvNo, u32 inBuffSizeShave, int jpegFormat)

  The JPEG encoding algorithm.

5.9.1 Detailed Description

JPEG Encoder Parallel functions API.

5.9.2 Macro Definition Documentation

#define PARTITION_0 (0)

#define PARTITION_1 (1)

#define PARTITION_10 (10)

#define PARTITION_11 (11)

#define PARTITION_2 (2)

#define PARTITION_3 (3)

#define PARTITION_4 (4)

#define PARTITION_5 (5)

#define PARTITION_6 (6)

#define PARTITION_7 (7)

#define PARTITION_8 (8)

#define PARTITION_9 (9)

5.9.3 Enumeration Type Documentation

anonymous enum

Enumerator

JPEG_420_PLANAR

JPEG_422_PLANAR

JPEG_444_PLANAR

5.9.4 Function Documentation

u32 JPEG_encode (frameBuffer imgInfo, u8 * outputLocal, u32 shvNo, u32 inBuffSizeShave, int jpegFormat)

The JPEG encoding algorithm.

Parameters

in	*imgInfo*	- structure that contains info about input image
out	*output*	- pointer to the output buffer
in	*shaveNo*	- number of shaves to be used
in	*inBuffSize-Shave*	- the size of the input buffer per shave
in	*jpegFormat*	- JPEG_420_PLANAR or JPEG_422_PLANAR or JPEG_444_PLANAR

Returns

u32 length of output buffer

5.10 LCD Generic API

LCD Generic Component Functions API.

Functions

- void LCDSetupCallbacks (LCDHandle *hndl, allocateLcdFn *assignFrame, frameReadyLcdFn *frameReady, freqLcdFn *highres, freqLcdFn *lowres)

  Set callbacks for assign new frame and frame ready.
- void LCDStop (LCDHandle *hndl)

  This will Stop the LCD interface.
- void LCDInit (LCDHandle *hndl, const LCDDisplayCfg *lcdsp, const frameSpec *fsp, unsigned int lcdNum)

  This will initialize the LCD interface.
- void LCDInitLayer (LCDHandle *hndl, int layer, frameSpec *fsp, LCDLayerOffset position)

  This will one of the LCD layers if fsp argument of LCDInit is NULL.
- void LCDEnYuv422i (void)

  This will enable YUV420p to Yuv422i format conversion on LCD interface.
- void LCDStart (LCDHandle *hndl)

  This will start the LCD interface.
- void LCDStartOneShot (LCDHandle *hndl)

  This will start the LCD interface in one-shot mode.
- int LCDQueueFrame (LCDHandle *handle, frameBuffer *VL1Buffer, frameBuffer *VL2Buffer, frameBuffer *GL1Buffer, frameBuffer *GL2Buffer)

  Queue a frame in One-Shot mode.
- int LCDCanQueueFrame (LCDHandle *handle)

  Tells you whether you can successfully queue a frame in one-shot mode.
- void LCDInitVL2Enable (LCDHandle *hndl)

  This will enable video layer 1.
- void LCDInitVL2Disable (LCDHandle *hndl)

  This will disable video layer 1.
- void LCDInitGLEnable (LCDHandle *hndl, int layer, const frameSpec *fr)

  This will enable graphical layer.
- void LCDInitGLDisable (LCDHandle *hndl, int layer)

  This will enable graphical layer.
- void LCDSetColorTable (LCDHandle *hndl, int layer, unsigned int *colorTable, int number)

  This will enable graphical layer.
- void LCDSetOutput (LCDHandle *hndl, lcdDatapath_t dataPath)

5.10.1 Detailed Description

LCD Generic Component Functions API. This is the API to the LCD controller subsystem component

5.10.2 Function Documentation

int LCDCanQueueFrame (LCDHandle * handle)

Tells you whether you can successfully queue a frame in one-shot mode.

Parameters

in	*handle*	- Pointer to the lcd handle

Returns

- zero, if the LCD already has two frames it hasn't given back.
- non-zero, if a new frame can be queued.

void LCDEnYuv422i (void)

This will enable YUV420p to Yuv422i format conversion on LCD interface.

Parameters

in	-	

Returns

void

void LCDInit (LCDHandle * hndl, const LCDDisplayCfg * lcdsp, const frameSpec * fsp, unsigned int lcdNum)

This will initialize the LCD interface.

Parameters

in	*hndl*	- Pointer to the lcd handle
in	*lcdsp*	- LCD configuration
in	*fsp*	- Frame specification
in	*lcdNum*	- LCD number 0 or 1

Returns

void

void LCDInitGLDisable (LCDHandle * hndl, int layer)

This will enable graphical layer.

Parameters

in	*hndl*	- Pointer to the lcd handle
in	*layer*	- 1 for GLl or 2 for GL2

Returns

void

void LCDInitGLEnable (LCDHandle * hndl, int layer, const frameSpec * fr)

This will enable graphical layer.

Parameters

in	*hndl*	- Pointer to the lcd handle
in	*layer*	- 1 for GLl or 2 for GL2
in	*fr*	- graphical layer frame spec

Returns

void

void LCDInitLayer (LCDHandle * hndl, int layer, frameSpec * fsp, LCDLayerOffset position)

This will one of the LCD layers if fsp argument of LCDInit is NULL.

Parameters

in	*hndl*	- Pointer to the lcd handle
in	*layer*	- LCD layer (VL1, VL2, GL1 or GL2)
in	*fsp*	- Frame spec
in	*position*	- layer position on the screen (from top-left corner)

Returns

void

void LCDInitVL2Disable (LCDHandle * hndl)

This will disable video layer 1.

Parameters

in	*hndl*	- Pointer to the lcd handle

Returns

void

void LCDInitVL2Enable (LCDHandle * hndl)

This will enable video layer 1.

Parameters

in	*hndl*	- Pointer to the lcd handle

Returns

void

int LCDQueueFrame (LCDHandle * handle, frameBuffer * VL1Buffer, frameBuffer * VL2Buffer, frameBuffer * GL1Buffer, frameBuffer * GL2Buffer)

Queue a frame in One-Shot mode.

Parameters

in	*handle*	- Pointer to the lcd handle
in	*VL1Buffer*	- Video Layer 1 frameBuffer pointer (may be NULL if the layer is not enabled)
in	*VL2Buffer*	- Video Layer 2
in	*GL1Buffer*	- Graphics Layer 1
in	*GL2Buffer*	- Graphics Layer 2

Returns

- 0 on success
- <0 on error (for example if two frames were already queued, and none were given back)

void LCDSetColorTable (LCDHandle * hndl, int layer, unsigned int * colorTable, int number)

This will enable graphical layer.

Parameters

in	*hndl*	- Pointer to the lcd handle
in	*layer*	- graphical layer number
in	*colorTable*	- color table
in	*number*	- number of colors from table

Returns

void

Note

needs to be called again after LcdStop call can be used during runtime.

void LCDSetOutput (LCDHandle * hndl, lcdDatapath_t dataPath)

Sets output datapath of LCD (Parallel port or Mipi)

Parameters

in	*hndl*	- Pointer to the lcd handle
in	*dataPath*	- type of datapath (LCD_TO_PARALLEL or LCD_TO_MIPI)

Returns

void

void LCDSetupCallbacks (LCDHandle * hndl, allocateLcdFn * assignFrame, frameReadyLcdFn * frameReady, freqLcdFn * highres, freqLcdFn * lowres)

Set callbacks for assign new frame and frame ready.

Parameters

in	*hndl*	- Pointer to the lcd handle
in	*assignFrame*	- Function pointer to get new frame
in	*frameReady*	- Function pointer for frame ready
in	*highres*	- Function pointer to set high frequency resolution
in	*lowres*	- Function pointer to set low frequency resolution

Returns

void

void LCDStart (LCDHandle * hndl)

This will start the LCD interface.

Parameters

in	*hndl*	- Pointer to the lcd handle

Returns

void

void LCDStartOneShot (LCDHandle * hndl)

This will start the LCD interface in one-shot mode.

Of the callbacks that can be set up with LCDSetupCallbacks, only the frameReady callback is used, and it is optional. In one-shot mode the LCD interface may own a minimum of zero, or a maximum of two frames: one that is currently being displayed, and one that is in the shadow registers. If you have successfully queued 3 frames, then you may assume that the first one you queued is done, and free to use for other purposes. The frame can become available much sooner then two consecutive frames, so it's best to use the frameReady callback to know when a frame is done.

Parameters

in	*hndl*	- Pointer to the lcd handle

Returns

void

void LCDStop (LCDHandle * hndl)

This will Stop the LCD interface.

Parameters

in	*hndl*	- Pointer to the lcd handle

Returns

void

5.11 Leon IPC API

Leon Inter Processor Communication Component Functions API.

Functions

- int LeonIPCTxInit (leonIPCChannel_t *msgChannel, uint32_t *messagePool, uint32_t messagePoolSize, uint32_t messageSize)

  This function initializes a protocol given a message pool and a size for it.
- int LeonIPCRxInit (leonIPCChannel_t *msgChannel, leonIPCCallback_t msgCallback, uint32_t irqNo, uint32_t irqPrio)

  This function initializes the Rx side of the communication.
- int LeonIPCRxReassignSinkThread (leonIPCChannel_t *channel)

  This function resets the receiver thread to the current calling thread.
- int LeonIPCRxReleaseSinkThread (leonIPCChannel_t *channel)

  This function resets the receiver thread. This function must be used before using the LeonIPCRxReassignSinkThread function.
- int LeonIPCSendMessage (leonIPCChannel_t *msgChannel, uint32_t *message)

  This function sends a message to the consumer.
- int LeonIPCWaitMessage (leonIPCChannel_t *msgChannel, uint32_t timeout)

  This function waits for a valid message to be present in the message queue.
- int LeonIPCNumberOfPendingMessages (leonIPCChannel_t *msgChannel, uint32_t *noOfmessages)

  This function waits for a valid message to be present in the message queue.
- int LeonIPCReadMessage (leonIPCChannel_t *msgChannel, uint32_t *message)

  This function waits for a valid message to be present in the message queue.

5.11.1 Detailed Description

Leon Inter Processor Communication Component Functions API. This is the API for Leon to Leon message passing

5.11.2 Function Documentation

int LeonIPCNumberOfPendingMessages (leonIPCChannel_t * msgChannel, uint32_t * noOfmessages)

This function waits for a valid message to be present in the message queue.

Parameters

in	*msgChannel*	- address of the communication channel
out	*noOfmessages*	- the number of messages available in the message queue

Returns

- IPC_SUCCESS - the operation finished successfully
- IPC_TX_NOTINITIALIZED - the transmitter was not initialized

int LeonIPCReadMessage (leonIPCChannel_t * msgChannel, uint32_t * message)

This function waits for a valid message to be present in the message queue.

Note

This function must be called in the same thread that called LeonIPCRxInit

Parameters

in	*msgChannel*	- address of the message channel
out	*message*	- pointer to the region of memory where the read message will be placed

Returns

- IPC_SUCCESS - the operation finished successfully
- IPC_TX_NOTINITIALIZED - the transmitter was not initialized

int LeonIPCRxInit (leonIPCChannel_t * msgChannel, leonIPCCallback_t msgCallback, uint32_t irqNo, uint32_t irqPrio)

This function initializes the Rx side of the communication.

Note

The function should only be called on the Rx side of the communication

Parameters

in	*msgChannel*	- address of a leonIPCChannel_t variable used for communication
in	*msgCallback*	- address of the callback to assign
in	*irqNo*	- the number of the interrupt used to notify the receiver that a message is available
in	*irqPrio*	- priority level of the notification interrupt

Returns

- IPC_SUCCESS - initialization finished successfully
- IPC_RX_ALREADY_INITIALIZED - Rx was already initialized

int LeonIPCRxReassignSinkThread (leonIPCChannel_t * channel)

This function resets the receiver thread to the current calling thread.

Note

The function should only be called on the Rx side of the communication after LeonIPCRxRelease-SinkThread is called

Parameters

in	*msgChannel*	- address of a leonIPCChannel_t variable used for communication

Returns

- IPC_SUCCESS - initialization finished successfully
- IPC_RX_ALREADY_INITIALIZED - Rx was already initialize
- RTEMS_NOT_OWNER_OF_RESOURCE - LeonIPCRxReleaseSinkThread should be called before using this function

int LeonIPCRxReleaseSinkThread (leonIPCChannel_t * channel)

This function resets the receiver thread. This function must be used before using the LeonIPCRx-ReassignSinkThread function.

Note

The function should only be called on the Rx side of the communication

Parameters

in	*msgChannel*	- address of a leonIPCChannel_t variable used for communication

Returns

- IPC_SUCCESS - initialization finished successfully
- IPC_RX_ALREADY_INITIALIZED - Rx was already initialized

int LeonIPCSendMessage (leonIPCChannel_t * msgChannel, uint32_t * message)

This function sends a message to the consumer.

Parameters

in	*msgChannel*	- address of the communication channel
in	*message*	- pointer to the message that needs to be sent

Returns

- IPC_SUCCESS - initialization finished successfully
- IPC_QUEUE_OVERFLOW - the message queue if full
- IPC_TX_ALREADY_INITIALIZED - Rx was already initialized

int LeonIPCTxInit (leonIPCChannel_t * msgChannel, uint32_t * messagePool, uint32_t messagePoolSize, uint32_t messageSize)

This function initializes a protocol given a message pool and a size for it.

Note

The function should only be called on the Tx side of the communication

Parameters

in	*msgChannel*	- address of a leonIPCChannel_t variable to initialize protocol
in	*messagePool*	- address of a uint32_t array for the message pool
in	*messagePool-Size*	- the maximum number of messages the queue can hold
in	*messageSize*	- the size of a single message in words

Returns

- IPC_SUCCESS - initialization finished successfully
- IPC_TX_ALREADY_INITIALIZED - Rx was already initialized

int LeonIPCWaitMessage (leonIPCChannel_t * msgChannel, uint32_t timeout)

This function waits for a valid message to be present in the message queue.

Parameters

in	*msgChannel*	- address of the communication channel
in	*timeout*	- the amount of time to wait for a message

Returns

- IPC_SUCCESS - initialization finished successfully
- IPC_TIMEOUT - the timeout expired before a valid message arrived

5.12 Leon L1 Cache

Utilities to work with Leon L1 Cache.

Functions

- void LeonL1CacheInitDiagAccess (void)
  Initialise diagnostic access to the cache.
- void LeonL1CacheDiagDisplay (tyCacheType cache)
  Display L1 cache contents.
- u32 LeonL1CacheDiagCountValidLines (tyCacheType cache)
  Count the valid L1 cache lines.
- void LeonL1CacheDisplayInfo (tyCacheType cache)
  Display L1 cache general information.

- u32 LeonL1CacheReadCacheTagMem (tyCacheType cache, u32 offset)
  Raw access to Tag and Data Memories.
- u32 LeonL1CacheReadCacheDataMem (tyCacheType cache, u32 offset)
  Raw access to Tag and Data Memories.

- u32 LeonL1DCacheReadTagForAddr (u32 address, u32 way)
  Read the L1 cache tag for a specific memory address.
- u32 LeonL1ICacheReadTagForAddr (u32 address, u32 way)
  Read the L1 cache tag for a specific memory address.

- u32 LeonL1DCacheReadDataWordForAddr (u32 address, u32 way)
  Read the L1 cache content for a specific memory address.
- u32 LeonL1ICacheReadDataWordForAddr (u32 address, u32 way)
  Read the L1 cache content for a specific memory address.

- u32 LeonL1CacheIsAddressDCached (u32 address)
  Check whether a specific memory address is cached either in data or instructions cache.
- u32 LeonL1CacheIsAddressICached (u32 address)
  Check whether a specific memory address is cached either in data or instructions cache.

5.12.1 Detailed Description

Utilities to work with Leon L1 Cache. Used for viewing and modifying Leon L1 Cache

5.12.2 Function Documentation

u32 LeonL1CacheDiagCountValidLines (tyCacheType cache)

Count the valid L1 cache lines.

Parameters

in	*cache*	- cache type(Data/Instructions)

Returns

number of valid lines

void LeonL1CacheDiagDisplay (tyCacheType cache)

Display L1 cache contents.

Parameters

in	*cache*	- cache type(Data/Instructions)

Returns

void

void LeonL1CacheDisplayInfo (tyCacheType cache)

Display L1 cache general information.

Parameters

in	*cache*	- cache type(Data/Instructions)

void LeonL1CacheInitDiagAccess (void)

Initialise diagnostic access to the cache.

Returns

void

u32 LeonL1CacheIsAddressDCached (u32 address)

Check whether a specific memory address is cached either in data or instructions cache.

Parameters

in	*address*	- address to be checked

Returns

- 1 - success;
- 0 - failure

u32 LeonL1CacheIsAddressICached (u32 address)

Check whether a specific memory address is cached either in data or instructions cache.

Parameters

in	*address*	- address to be checked

Returns

- 1 - success;
- 0 - failure

u32 LeonL1CacheReadCacheDataMem (tyCacheType cache, u32 offset)

Raw access to Tag and Data Memories.

Parameters

in	*cache*	- cache type(Data/Instructions)
in	*offset*	- the relative offset where the acces is requested

u32 LeonL1CacheReadCacheTagMem (tyCacheType cache, u32 offset)

Raw access to Tag and Data Memories.

Parameters

in	*cache*	- cache type(Data/Instructions)
in	*offset*	- the relative offset where the acces is requested

u32 LeonL1DCacheReadDataWordForAddr (u32 address, u32 way)

Read the L1 cache content for a specific memory address.

Parameters

in	*address*	- the requested memory address
in	*way*	- the cache way on which the request is made

Returns

The cached memory value

u32 LeonL1DCacheReadTagForAddr (u32 address, u32 way)

Read the L1 cache tag for a specific memory address.

Parameters

in	*address*	- the requested memory address
in	*way*	- the cache way on which the request is made

Returns

The cache tag value

u32 LeonL1ICacheReadDataWordForAddr (u32 address, u32 way)

Read the L1 cache content for a specific memory address.

Parameters

in	*address*	- the requested memory address
in	*way*	- the cache way on which the request is made

Returns

The cached memory value

u32 LeonL1ICacheReadTagForAddr (u32 address, u32 way)

Read the L1 cache tag for a specific memory address.

Parameters

in	*address*	- the requested memory address
in	*way*	- the cache way on which the request is made

Returns

The cache tag value

5.13 World Message Protocol API

Outside World Message Protocol API.

Data Structures

- struct PhysicalChannel
- struct VirtualChannel
- struct OsVirtualChannel

Macros

- #define MAX_COUNT_MESSAGING_PHYSICAL_CHANNELS 1
- #define MAX_COUNT_MESSAGING_VIRTUAL_CHANNELS 20
- #define DECLARE_MESSAGING_VIRTUAL_CHANNEL(vcHandle, channelName,channelId, priority,rx_fifo_size, tx_fifo_size, fifo_data_section)
- #define DECLARE_OS_MESSAGING_VIRTUAL_CHANNEL(vcHandle, channelName,channelId, priority,rx_fifo_size, tx_fifo_size, fifo_data_section)
- #define DEV_VIRTUAL_CHANNEL_DRIVER_TABLE_ENTRY
- #define DECLARE_COMMUNICATION_PROTOCOL_DRIVER_TABLE(drvTblName) rtems_driver_address_table drvTblName = DEV_VIRTUAL_CHANNEL_DRIVER_TABLE_ENTRY;

Typedefs

- typedef void ∗(PacketWriteRequestCallback_t)(u16 length, u8 channel, u8 flags)
- typedef void ∗(PacketReadRequestCallback_t)(u16 length, u8 channel, u8 flags)
- typedef s32(PacketWriteDoneCallback_t)(u16 length, u8 channel, u8 flags, void ∗buffer)
- typedef s32(PacketReadDoneCallback_t)(u16 length, u8 channel, u8 flags, void ∗buffer)
- typedef s32(PacketExchangeOverCallback_t)(s32 ∗channel, void ∗∗buffer, s32 ∗size)

Enumerations

- enum ChannelType { SPISLAVE }
- enum txPending_t { TX_IDLE, TX_PENDING }

Functions

- void MessagePassingInitialize (void)
- s32 MessagePassingInitializePhysicalChannel (PhysicalChannel ∗phyChannel, void ∗phyChannelContext, ChannelType ct)
- s32 BaseMessagePassingInitializePhysicalChannel (PhysicalChannel ∗phyChannel, void ∗phyChannelContext, ChannelType ct)
- s32 MessagePassingRegisterVirtualChannel (VirtualChannel ∗vc)
- PacketWriteRequestCallback_t ∗ MesasgePassingGetCbTxStart (PhysicalChannel pc)
- PacketWriteDoneCallback_t ∗ MesasgePassingGetCbTxDone (PhysicalChannel pc)
- PacketReadRequestCallback_t ∗ MesasgePassingGetCbRxStart (PhysicalChannel pc)

- PacketReadDoneCallback_t * MesasgePassingGetCbRxDone (PhysicalChannel pc)
- PacketExchangeOverCallback_t * MesasgePassingGetCbPeOver (PhysicalChannel pc)
- s32 MessagePassingWrite (u8 channel, void *buff, s32 size)
- s32 MessagePassingRead (u8 channel, void *buff, s32 size)
- s32 OsMessagePassingReadBlockEvent (u8 channel, void *buff, s32 size)
- void * MessagePassingGetDriverRxBuffer (u8 channel, s32 length)
- void MessagePassingFinalizeChannelRx (u8 channel)
- s32 MessagePassingFinalizePacketExchange (s32 *channel, void **buffer, s32 *size)
- VirtualChannel * MessagePassingGetVirtualChannel (u8 channelId)
- s32 BaseMessagePassingRead (VirtualChannel *vc, void *buff, s32 size)
- void BaseMessagePassingFinalizeChannelRx (VirtualChannel *vc)
- void OsMessagePassingInitialize (void)
- s32 OsMessagePassingInitializePhysicalChannel (PhysicalChannel *phyChannel, void *phyChannelContext, ChannelType ct)
- s32 OsMessagePassingRegisterVirtualChannel (PhysicalChannel *phyChannel, VirtualChannel *vc)
- PacketWriteRequestCallback_t * OsMesasgePassingGetCbTxStart (PhysicalChannel pcList)
- PacketReadRequestCallback_t * OsMesasgePassingGetCbRxStart (PhysicalChannel pcList)
- PacketWriteDoneCallback_t * OsMesasgePassingGetCbTxDone (PhysicalChannel pcList)
- PacketReadDoneCallback_t * OsMesasgePassingGetCbRxDone (PhysicalChannel pcList)
- PacketExchangeOverCallback_t * OsMesasgePassingGetCbPeOver (PhysicalChannel pcList)
- rtems_device_driver virtual_channel_initialize (rtems_device_major_number major, rtems_device_minor_number minor, void *e)
- rtems_device_driver virtual_channel_open (rtems_device_major_number major, rtems_device_minor_number minor, void *e)
- rtems_device_driver virtual_channel_close (rtems_device_major_number major, rtems_device_minor_number minor, void *e)
- rtems_device_driver virtual_channel_read (rtems_device_major_number major, rtems_device_minor_number minor, void *e)
- rtems_device_driver virtual_channel_write (rtems_device_major_number major, rtems_device_minor_number minor, void *e)
- rtems_device_driver virtual_channel_control (rtems_device_major_number major, rtems_device_minor_number minor, void *e)

Variables

- ChannelType PhysicalChannel::ct
- void * PhysicalChannel::context
- u8 VirtualChannel::id
- u8 VirtualChannel::name [32]
- u32 VirtualChannel::priority_level
- PhysicalChannel * VirtualChannel::phyChannel
- MessageRingBuffer VirtualChannel::rxFifo
- MessageRingBuffer VirtualChannel::txFifo
- VirtualChannel OsVirtualChannel::bmVC
- rtems_id OsVirtualChannel::rxWaitTaskId

5.13.1 Detailed Description

Outside World Message Protocol API.

5.13.2 Macro Definition Documentation

#define DECLARE_COMMUNICATION_PROTOCOL_DRIVER_TABLE(drvTblName) rtems_-driver_address_table drvTblName = **DEV_VIRTUAL_CHANNEL_DRIVER_TABLE_ENTRY**;

#define DECLARE_MESSAGING_VIRTUAL_CHANNEL(vcHandle, channelName, channelId, priority, rx_fifo_size, tx_fifo_size, fifo_data_section)

Value:

```
static u8 __attribute__((section(fifo_data_section))) message_rx_fifo_bm_#
      #channelId[rx_fifo_size]; \
    static u8 __attribute__((section(fifo_data_section))) message_tx_fifo_bm_##channelId[tx_fifo_size]; \
    static VirtualChannel vcHandle =                                        \
    {                                                                       \
     .name = channelName,                                                   \
          .id = channelId,                                                  \
          .priority_level = priority,                                       \
          .rxFifo = MESSAGE_RING_BUFFER(message_rx_fifo_bm_##channelId, rx_fifo_size), \
                                      .txFifo = MESSAGE_RING_BUFFER(message_tx_fifo_bm_##channelId,
       tx_fifo_size), \
                                      };
```

#define DECLARE_OS_MESSAGING_VIRTUAL_CHANNEL(vcHandle, channelName, channelId, priority, rx_fifo_size, tx_fifo_size, fifo_data_section)

Value:

```
static u8 __attribute__((section(fifo_data_section))) message_rx_fifo_os_#
      #channelId[rx_fifo_size]; \
    static u8 __attribute__((section(fifo_data_section))) message_tx_fifo_os_##channelId[tx_fifo_size]; \
    static OsVirtualChannel vcHandle =  {                                   \
        .bmVC = {                                                           \
            .name = channelName,                                            \
            .id = channelId,                                                \
            .priority_level = priority,                                     \
            .rxFifo = MESSAGE_RING_BUFFER(message_rx_fifo_os_##channelId, rx_fifo_size), \
            .txFifo = MESSAGE_RING_BUFFER(message_tx_fifo_os_##channelId, tx_fifo_size), \
        },                                                                  \
        .rxWaitTaskId = 0,                                                  \
    };
```

#define DEV_VIRTUAL_CHANNEL_DRIVER_TABLE_ENTRY

Value:

```
{       \
        virtual_channel_initialize,                     \
            virtual_channel_open,                       \
            virtual_channel_close,                      \
            virtual_channel_read,                       \
            virtual_channel_write,                      \
            virtual_channel_control                     \
            }
```

#define MAX_COUNT_MESSAGING_PHYSICAL_CHANNELS 1

#define MAX_COUNT_MESSAGING_VIRTUAL_CHANNELS 20

5.13.3 Typedef Documentation

typedef s32(PacketExchangeOverCallback_t)(s32 *channel, void **buffer, s32 *size)

typedef s32(PacketReadDoneCallback_t)(u16 length, u8 channel, u8 flags, void *buffer)

typedef void*(PacketReadRequestCallback_t)(u16 length, u8 channel, u8 flags)

typedef s32(PacketWriteDoneCallback_t)(u16 length, u8 channel, u8 flags, void *buffer)

typedef void*(PacketWriteRequestCallback_t)(u16 length, u8 channel, u8 flags)

5.13.4 Enumeration Type Documentation

enum **ChannelType**

Enumerator

SPISLAVE

enum **txPending_t**

Enumerator

TX_IDLE

TX_PENDING

5.13.5 Function Documentation

void BaseMessagePassingFinalizeChannelRx (**VirtualChannel** * vc)

s32 BaseMessagePassingInitializePhysicalChannel (**PhysicalChannel** * phyChannel, void * phyChannelContext, **ChannelType** ct)

s32 BaseMessagePassingRead (**VirtualChannel** * vc, void * buff, s32 size)

PacketExchangeOverCallback_t* MesasgePassingGetCbPeOver (**PhysicalChannel** pc)

PacketReadDoneCallback_t* MesasgePassingGetCbRxDone (**PhysicalChannel** pc)

PacketReadRequestCallback_t* MesasgePassingGetCbRxStart (**PhysicalChannel** pc)

PacketWriteDoneCallback_t* MesasgePassingGetCbTxDone (**PhysicalChannel** pc)

PacketWriteRequestCallback_t* MesasgePassingGetCbTxStart (**PhysicalChannel** pc)

void MessagePassingFinalizeChannelRx (u8 channel)

s32 MessagePassingFinalizePacketExchange (s32 * channel, void ** buffer, s32 * size)

void* MessagePassingGetDriverRxBuffer (u8 channel, s32 length)

VirtualChannel* MessagePassingGetVirtualChannel (u8 channelId)

void MessagePassingInitialize (void)

s32 MessagePassingInitializePhysicalChannel (**PhysicalChannel** * phyChannel, void * phyChannelContext, **ChannelType** ct)

s32 MessagePassingRead (u8 channel, void * buff, s32 size)

s32 MessagePassingRegisterVirtualChannel (**VirtualChannel** * vc)

s32 MessagePassingWrite (u8 channel, void * buff, s32 size)

PacketExchangeOverCallback_t* OsMesasgePassingGetCbPeOver (**PhysicalChannel** pcList)

PacketReadDoneCallback_t* OsMesasgePassingGetCbRxDone (**PhysicalChannel** pcList)

PacketReadRequestCallback_t* OsMesasgePassingGetCbRxStart (**PhysicalChannel** pcList)

PacketWriteDoneCallback_t* OsMesasgePassingGetCbTxDone (**PhysicalChannel** pcList)

PacketWriteRequestCallback_t* OsMesasgePassingGetCbTxStart (**PhysicalChannel** pcList)

void OsMessagePassingInitialize (void)

s32 OsMessagePassingInitializePhysicalChannel (**PhysicalChannel** * phyChannel, void * phyChannelContext, **ChannelType** ct)

s32 OsMessagePassingReadBlockEvent (u8 channel, void * buff, s32 size)

s32 OsMessagePassingRegisterVirtualChannel (**PhysicalChannel** * phyChannel, **VirtualChannel** * vc)

rtems_device_driver virtual_channel_close (rtems_device_major_number major, rtems_device_minor_number minor, void * e)

rtems_device_driver virtual_channel_control (rtems_device_major_number major, rtems_device_minor_number minor, void * e)

rtems_device_driver virtual_channel_initialize (rtems_device_major_number major, rtems_device_minor_number minor, void * e)

rtems_device_driver virtual_channel_open (rtems_device_major_number major, rtems_device_minor_number minor, void * e)

rtems_device_driver virtual_channel_read (rtems_device_major_number major, rtems_device_minor_number minor, void * e)

rtems_device_driver virtual_channel_write (rtems_device_major_number major, rtems_device_minor_number minor, void * e)

5.13.6 Variable Documentation

VirtualChannel OsVirtualChannel::bmVC

void* PhysicalChannel::context

ChannelType PhysicalChannel::ct

u8 VirtualChannel::id

u8 VirtualChannel::name[32]

PhysicalChannel* VirtualChannel::phyChannel

u32 VirtualChannel::priority_level

MessageRingBuffer VirtualChannel::rxFifo

rtems_id OsVirtualChannel::rxWaitTaskId

MessageRingBuffer VirtualChannel::txFifo

5.14 API

MessageProtocolRingBuffer API.

Data Structures

- struct MessageRingBuffer

Macros

- #define MESSAGE_RING_BUFFER(buffer, rb_size)

Functions

- void InitMessageRB (MessageRingBuffer *mrb, void *buffer, s32 size)
- void * getMessageRBWrPtr (MessageRingBuffer *mrb, s32 neededLength)
- void finishMessageRBWrite (MessageRingBuffer *mrb)
- void * getMessageRBRdPtr (MessageRingBuffer *mrb, s32 *availableLength)

Variables

- void * MessageRingBuffer::allMem
- s32 MessageRingBuffer::size
- s32 MessageRingBuffer::availableLength
- s32 MessageRingBuffer::activeWriteSize
- void * MessageRingBuffer::wrPtr
- void * MessageRingBuffer::rdPtr
- void * MessageRingBuffer::endPtr

5.14.1 Detailed Description

MessageProtocolRingBuffer API.

5.14.2 Macro Definition Documentation

#define MESSAGE_RING_BUFFER(buffer, rb_size)

Value:

```
{                                               \
        .allMem = buffer,                       \
            .wrPtr = buffer,                    \
            .rdPtr = buffer,                    \
            .endPtr = buffer,                   \
            .size = rb_size,                    \
            .availableLength = 0,               \
            }
```

5.14.3 Function Documentation

void finishMessageRBWrite (**MessageRingBuffer** * mrb)

void* getMessageRBRdPtr (**MessageRingBuffer** * mrb, s32 * availableLength)

void* getMessageRBWrPtr (**MessageRingBuffer** * mrb, s32 neededLength)

void InitMessageRB (**MessageRingBuffer** * mrb, void * buffer, s32 size)

5.14.4 Variable Documentation

s32 MessageRingBuffer::activeWriteSize

void* MessageRingBuffer::allMem

s32 MessageRingBuffer::availableLength

void* MessageRingBuffer::endPtr

void* MessageRingBuffer::rdPtr

s32 MessageRingBuffer::size

void* MessageRingBuffer::wrPtr

5.15 SPI Slave API

SPI Slave API.

Data Structures

- struct spiSlaveCommunicationConfiguration_t

Macros

- #define DRVSPI_CONFIGURATION(dev, cpol, cpha, bytesPerWord, dmaEnabled, hostIrqGpio, interruptLevel)

Functions

- rtems_status_code OsDrvSpiSlaveCPInitGlobally (spiHandler_t *handler, spiTxStartCallback_t *txStartCb, spiTxDoneCallback_t *txDoneCb, spiRxStartCallback_t *rxStartCb, spiRxDoneCallback_t *rxDoneCb, spiPeDoneCallback_t *peOverCb)
- rtems_status_code OsDrvSpiSlaveCPInit (spiHandler_t *handler, wordSizeBytes_t wordSizeBytes, dmaUsed_t useDma, spiSlaveBlock_t device, u32 cpol, u32 cpha, u32 interruptLevel, u32 hostIrqGpio, spiTxStartCallback_t *txStartCb, spiTxDoneCallback_t *txDoneCb, spiRxStartCallback_t *rxStartCb, spiRxDoneCallback_t *rxDoneCb, spiPeDoneCallback_t *peOverCb)
- rtems_status_code OsDrvSpiSlaveCPSendPacket (spiHandler_t *handler, u8 channel, u8 flags, s32 size, void *buff)

Variables

- spiSlaveBlock_t spiSlaveCommunicationConfiguration_t::device
- u32 spiSlaveCommunicationConfiguration_t::scpol
- u32 spiSlaveCommunicationConfiguration_t::scpha
- u32 spiSlaveCommunicationConfiguration_t::bpw
- dmaUsed_t spiSlaveCommunicationConfiguration_t::useDma
- u32 spiSlaveCommunicationConfiguration_t::hostGpioIrq
- u32 spiSlaveCommunicationConfiguration_t::irqLevel
- spiSlaveCommunicationConfiguration_t spiConfig

5.15.1 Detailed Description

SPI Slave API.

5.15.2 Macro Definition Documentation

#define DRVSPI_CONFIGURATION(dev, cpol, cpha, bytesPerWord, dmaEnabled, hostIrqGpio, interruptLevel)

Value:

```
spiSlaveCommunicationConfiguration_t
      spiConfig = {                  \
        .device = dev,                                          \
        .scpol = cpol,                                          \
        .scpha = cpha,                                          \
        .bpw = bytesPerWord,                                    \
        .useDma = dmaEnabled,                                   \
        .hostGpioIrq = hostIrqGpio,                             \
        .irqLevel = interruptLevel,                             \
    }
```

5.15.3 Function Documentation

rtems_status_code OsDrvSpiSlaveCPInit (spiHandler_t * handler, wordSizeBytes_t wordSizeBytes, dmaUsed_t useDma, spiSlaveBlock_t device, u32 cpol, u32 cpha, u32 interruptLevel, u32 hostIrqGpio, spiTxStartCallback_t * txStartCb, spiTxDoneCallback_t * txDoneCb, spiRxStartCallback_t * rxStartCb, spiRxDoneCallback_t * rxDoneCb, spiPeDoneCallback_t * peOverCb)

rtems_status_code OsDrvSpiSlaveCPInitGlobally (spiHandler_t * handler, spiTxStartCallback_t * txStartCb, spiTxDoneCallback_t * txDoneCb, spiRxStartCallback_t * rxStartCb, spiRxDoneCallback_t * rxDoneCb, spiPeDoneCallback_t * peOverCb)

rtems_status_code OsDrvSpiSlaveCPSendPacket (spiHandler_t * handler, u8 channel, u8 flags, s32 size, void * buff)

5.15.4 Variable Documentation

u32 spiSlaveCommunicationConfiguration_t::bpw

spiSlaveBlock_t spiSlaveCommunicationConfiguration_t::device

u32 spiSlaveCommunicationConfiguration_t::hostGpioIrq

u32 spiSlaveCommunicationConfiguration_t::irqLevel

u32 spiSlaveCommunicationConfiguration_t::scpha

u32 spiSlaveCommunicationConfiguration_t::scpol

spiSlaveCommunicationConfiguration_t spiConfig

dmaUsed_t spiSlaveCommunicationConfiguration_t::useDma

5.16 Opipe

The Opipe components is an optimized ISP pipeline composed of a chain of SIPP hardware accelerator filters.

Files

- file Opipe.h

  Opipe - API.
- file OpipeBlocks.h

  Opipe - SIPP blocks config data structs.
- file OpipeDefs.h

  Opipe - definitions.

5.16.1 Detailed Description

The Opipe components is an optimized ISP pipeline composed of a chain of SIPP hardware accelerator filters.

5.17 Pattern Generator API

Pattern Generator Component Functions API.

Functions

- unsigned int CreateHorizontalColorBars (frameBuffer *frame, unsigned int interlaced)

  Create an horizontal pattern at address specified in param list.
- unsigned int CreateVerticalColorBars (frameBuffer *frame, unsigned int interlaced)

  Create an vertical pattern at address specified in param list.
- unsigned int CreateColorStripesPattern (frameBuffer *frame)

  Create an specific horizontal pattern with 64 color stripes for first half and 8 color stripes for second half.
- unsigned int CreateLinearGreyPattern (frameBuffer *frame)

  Create an specific vertical pattern with 17 grey stripes, from black to white.
- unsigned int PatternCheck (frameBuffer *inputFrame, frameBuffer *outputFrame, unsigned int interlaced)

  Check if patterns are correct by comparing.

5.17.1 Detailed Description

Pattern Generator Component Functions API. This is the API to a Pattern Generator library implementing several types of color pattern generation, as well as pattern checking.

5.17.2 Function Documentation

unsigned int CreateColorStripesPattern (frameBuffer * frame)

Create an specific horizontal pattern with 64 color stripes for first half and 8 color stripes for second half.

Parameters

in	*frame*	- frame to make pattern

Returns

- 0 FAIL
- 1 SUCCESS

unsigned int CreateHorizontalColorBars (frameBuffer * frame, unsigned int interlaced)

Create an horizontal pattern at address specified in param list.

Parameters

in	*frame*	- frame to make pattern
in	*interlaced*	- flag for interlaced frames

Returns

- 0 FAIL
- 1 SUCCESS

unsigned int CreateLinearGreyPattern (frameBuffer * frame)

Create an specific vertical pattern with 17 grey stripes, from black to white.

Parameters

in	*frame*	- frame to make pattern

Returns

- 0 FAIL
- 1 SUCCESS

unsigned int CreateVerticalColorBars (frameBuffer * frame, unsigned int interlaced)

Create an vertical pattern at address specified in param list.

Parameters

in	*frame*	- frame to make pattern
in	*interlaced*	- lag for interlaced frames

Returns

- 0 FAIL
- 1 SUCCESS

unsigned int PatternCheck (frameBuffer * inputFrame, frameBuffer * outputFrame, unsigned int interlaced)

Check if patterns are correct by comparing.

Parameters

in	*inputFrame*	- input frame to test
in	*outputFrame*	- output frame to test
in	*interlaced*	- flag for interlaced frames

Returns

- 0 FAIL
- 1 MATCH

5.18 Unit Test API

Unit Test API.

Functions

- int unitTestInit (void)

  Initiate a new unit test.
- int unitTestFloatWithinRange (float actual, float expected, float percentageError)

  Checks if a floating point value is within a certain percentage of expected value.
- int unitTestExecutionWithinRange (float actual, float expected, float percentageError)

  Checks if a floating point value is within an acceptable margin expected value.
- int unitTestFloatAbsRangeCheck (float actual, float expected, float AbsError)

  Checks if a floating point value is within an acceptable margin expected value.
- int unitTestAssert (int value)

  Check if a logical condition is true or not.
- int unitTestLogPass (void)

  Log a passed test.
- int unitTestLogFail (void)

  Log a failed test.
- int unitTestLogResults (int passes, int fails)

  Set up the passed and failed tests results.
- void unitTestMemCompare (const void *pActualStart, const void *pExpectedStart, u32 lengthBytes)

  Compares two values.
- void unitTestMemCompareDeltaU8 (u8 *pActualStart, u8 *pExpectedStart, u32 lengthBytes, u8 delta)

  Compares two values, with threshold.
- void unitTestCompare (u32 actualValue, u32 expectedValue)
- int unitTestCheckSectionFail (char *sectionName)
- int unitTestFinalReport (void)

  Print the final results of the unit testing.

5.18.1 Detailed Description

Unit Test API. Used for test reporting

5.18.2 Function Documentation

int unitTestAssert (int value)

Check if a logical condition is true or not.

Parameters

in	*value*	- the value to be checked

Returns

0

int unitTestCheckSectionFail (char * sectionName)

@brief Check if there has been any failures in the last test section

The first section starts at the beginning of the test. When this function is called, it checks if there has been any failures in the previous tests and switches to a new a section. Any call of this function will only report failures in the tests run since the previous call of this function.

This functions prints a message if any failure has occurred, indicating the name of the section (passed as a parameter) and the number of failures. If the user does not want this message to be printed, set sectionName to NULL.

Parameters

in	*section_name*	- the name of the test section being checked /!\ This string has to be '\0' terminated. If set to NULL, no error message is printed.

Returns

the number of tests failed in the last section

void unitTestCompare (u32 actualValue, u32 expectedValue)

@brief Compares two values

Parameters

in	*actualValue*	- Actual value
in	*expectedValue*	- Expected value

Returns

void

int unitTestExecutionWithinRange (float actual, float expected, float percentageError)

Checks if a floating point value is within an acceptable margin expected value.

Note

unitTestExecutionWithinRange(actual,100, 10) passes if actual >90 and less than 110

Parameters

in	*actual*	- actual execution time
in	*expected*	- expected execution time
in	*percentage-Error*	- Accepted execution time error

Returns

0

int unitTestFinalReport (void)

Print the final results of the unit testing.

Returns

0

int unitTestFloatAbsRangeCheck (float actual, float expected, float AbsError)

Checks if a floating point value is within an acceptable margin expected value.

Parameters

in	*actual*	- existing float
in	*expected*	- expected float
in	*AbsError*	- accepted absolute error

Returns

0

int unitTestFloatWithinRange (float actual, float expected, float percentageError)

Checks if a floating point value is within a certain percentage of expected value.

Note

actual+1 is considered 100% deviation, actual is 0% deviation

Parameters

in	*actual*	- actual floating point value
in	*expected*	- expected floating point value

in	*percentageError*	- acceptable percent deviation as explained above

Returns

0

int unitTestInit (void)

Initiate a new unit test.

int unitTestLogFail (void)

Log a failed test.

Returns

0

int unitTestLogPass (void)

Log a passed test.

Returns

0

int unitTestLogResults (int passes, int fails)

Set up the passed and failed tests results.

Parameters

in	*passes*	- number of passed tests
in	*fails*	- number of failed tests

Returns

0

void unitTestMemCompare (const void * pActualStart, const void * pExpectedStart, u32 lengthBytes)

Compares two values.

Parameters

in	*pActualStart*	- Pointer to the actual value
in	*pExpectedStart*	- Pointer to the expected value
in	*lengthBytes*	- Value length in bytes

Returns

void

void unitTestMemCompareDeltaU8 (u8 * pActualStart, u8 * pExpectedStart, u32 lengthBytes, u8 delta)

Compares two values, with threshold.

Parameters

in	*pActualStart*	- Pointer to the actual value
in	*pExpectedStart*	- Pointer to the expected value
in	*lengthBytes*	- Value length in bytes
in	*delta*	- threshold

Returns

void

5.19 VCS Unit Test API

Unit Test API for VCS.

Functions

- void unitTestInit (void)

  Initialise the Unit Test Framework.
- void unitTestVerbosity (tyUnitVerbosity targetVerbosity)

  Set expected verbosity of the unitTest library.
- void unitTestAssert (int value)

  Assert that value passed is TRUE (non-zero)
- void unitTestCompare (u32 actualValue, u32 expectedValue)

  Compare 2 32 bit values, log error if they don't match.
- void unitTestCompare64 (u64 actualValue, u64 expectedValue)

  Compare 2 64 bit values, log error if they don't match.
- void unitTestReadDWordCheck (void *dWordAddress, u64 expectedValue)

  Read a dword from memory and compare against expected value. Log a failure if values don't match.
- void unitTestReadWordCheck (void *wordAddress, u32 expectedValue)

  Read a word from memory and compare against expected value. Log a failure if values don't match.
- void unitTestReadHalfCheck (void *address, u16 expectedValue)

  Read a 16 bit value from memory and compare against expected value. Log a failure if values don't match.
- void unitTestReadByteCheck (void *address, u8 expectedValue)

  Read a byte from memory and compare against expected value. Log a failure if values don't match.
- void unitTestReadBitCheck (void *wordAddress, u32 startBit, u32 numBits, u32 expectedValue)

  Read a word from memory and compare a number of bits from the value against an expected result. Log a failure if values don't match.
- void unitTestMemCompare (const void *pActualStart, const void *pExpectedStart, u32 lengthBytes)

  Compare two memory buffers for a given number of bytes.
- void unitTestCrcCheck (const void *pStart, u32 lengthBytes, u32 expectedCrc)

  Perform 32 bit CRC on the buffer of lengthBytes and compare against expectedCrc.
- void unitTestExecutionWithinRange (float actual, float expected, float percentageError)

  Checks if a floating point value is within an acceptable margin expected value.
- u32 unitTestFinalReport (void)

  Signal unit test Framework that testing is complete and generate report.
- void unitTestSetTestType (tyTestType testType)

  Set Test Type for either Positive (default) or Negative testing.
- void unitTestLogPass (void)

  Log a Pass in the Unit test framework.
- void unitTestLogFail (void)

  Log a Fail in the Unit test framework.

5.19.1 Detailed Description

Unit Test API for VCS. Used for test reporting

5.19.2 Function Documentation

void unitTestAssert (int value)

Assert that value passed is TRUE (non-zero)

Parameters

in	*value*	- Error logged if value == 0 or contains xx's

Returns

void

void unitTestCompare (u32 actualValue, u32 expectedValue)

Compare 2 32 bit values, log error if they don't match.

Parameters

in	*actualValue*	- actual value to be tested
in	*expectedValue*	- value for comparison

Returns

void

void unitTestCompare64 (u64 actualValue, u64 expectedValue)

Compare 2 64 bit values, log error if they don't match.

Parameters

in	*actualValue*	- actual value to be tested
in	*expectedValue*	- value for comparison

Returns

void

void unitTestCrcCheck (const void * pStart, u32 lengthBytes, u32 expectedCrc)

Perform 32 bit CRC on the buffer of lengthBytes and compare against expectedCrc.

The buffer may be byte aligned and there are no limitations on lengthBytes This is a high speed hardware accelerated CRC (takes negligible sim time even for multiMB buffers) The test will also fail if any xx's are found in the buffer.

Note

This function performs a raw read of the actual target memory! As such, it is the responsibility of the caller to ensure cache coherency (i.e. flush before use)

Parameters

in	*pStart*	- Start Pointer to buffer which should be checked
in	*expectedCrc*	- Start Pointer to buffer for CRC comparison
in	*lengthBytes*	- Number of bytes to compare

Returns

void

void unitTestExecutionWithinRange (float actual, float expected, float percentageError)

Checks if a floating point value is within an acceptable margin expected value.

Note

unitTestExecutionWithinRange(actual,100, 10) passes if actual >90 and less than 110

Parameters

in	*actual*	- actual execution time
in	*expected*	- expected execution time
in	*percentage-Error*	- Accepted execution time error

Returns

u32 unitTestFinalReport (void)

Signal unit test Framework that testing is complete and generate report.

Attention

This function MUST be called at the end of testing!

Note

This API supports returning the number of failures found but this is only valid when the API is used in "software mode" i.e. Where M2_SW_UNIT_TEST=yes makefile flag is specified In other cases the test system has no way to report back the number of failures and defaults to returning a fail

Returns

numFailures

void unitTestInit (void)

Initialise the Unit Test Framework.

Note

This function must be called before any other call in this API is valid

Returns

void

void unitTestLogFail (void)

Log a Fail in the Unit test framework.

This function call is provided for backwards compatiblity It allows for software tests which do their own checking. However it is strongly recommended to avoid this method as it does not allow for checking for xx's etc.

void unitTestLogPass (void)

Log a Pass in the Unit test framework.

This function call is provided for backwards compatiblity It allows for software tests which do their own checking. However it is strongly recommended to avoid this method as it does not allow for checking for xx's etc.

void unitTestMemCompare (const void * pActualStart, const void * pExpectedStart, u32 lengthBytes)

Compare two memory buffers for a given number of bytes.

Both pointers may be byte aligned and there are no limitations of lengthBytes This is a high speed hardware accelerated compare (takes negligible sim time even for multiMB buffers) The test will also fail if any xx's are found in either buffer. note This function can also be used to compare words, short etc as the system is all LE

Note

This function performs a raw read of the actual target memory! As such, it is the responsibility of the caller to ensure cache coherency (i.e. flush before use)

Parameters

in	*pActualStart*	- Start Pointer to buffer which should be checked
in	*pExpectedStart*	- Start Pointer to buffer for comparison

in	*lengthBytes*	- Number of bytes to compare

Returns

void

void unitTestReadBitCheck (void * wordAddress, u32 startBit, u32 numBits, u32 expectedValue)

Read a word from memory and compare a number of bits from the value against an expected result. Log a failure if values don't match.

Note

The Leon performs the actual READ

Parameters

in	*wordAddress*	- Address of word to read
in	*startBit*	- First bit to compare against (0-31)
in	*numBits*	- number of bits to compare
in	*expectedValue*	- numBits value to compare against [(numBits-1):0]

Returns

void

void unitTestReadByteCheck (void * address, u8 expectedValue)

Read a byte from memory and compare against expected value. Log a failure if values don't match.

Note

The Leon performs the actual READ

Parameters

in	*address*	- Address of word to read
in	*expectedValue*	- 8 bit value to compare against

Returns

void

void unitTestReadDWordCheck (void * dWordAddress, u64 expectedValue)

Read a dword from memory and compare against expected value. Log a failure if values don't match.

Note

The Leon performs the actual READ

Parameters

in	*dWordAddress*	- Address of dword to read
in	*expectedValue*	- 64 bit value to compare against

Returns

void

void unitTestReadHalfCheck (void * address, u16 expectedValue)

Read a 16 bit value from memory and compare against expected value. Log a failure if values don't match.

Note

The Leon performs the actual READ

Parameters

in	*address*	- Address of word to read
in	*expectedValue*	- 16 bit value to compare against

Returns

void

void unitTestReadWordCheck (void * wordAddress, u32 expectedValue)

Read a word from memory and compare against expected value. Log a failure if values don't match.

Note

The Leon performs the actual READ

Parameters

in	*wordAddress*	- Address of word to read
in	*expectedValue*	- 32 bit value to compare against

Returns

void

void unitTestSetTestType (tyTestType testType)

Set Test Type for either Positive (default) or Negative testing.

The normal default mode is that we expect all tests to PASS however this function allows us to invert that setting in the test logic and declare that we expect the following tests to FAIL While the testType == EXPECT_TESTS_TO_FAIL any tests that fail are effectively marked as passing and any tests that pass are effectively marked as failing

Parameters

in	*testType*	- (EXPECT_TESTS_TO_PASS,EXPECT_TESTS_TO_FAIL)

Returns

void

void unitTestVerbosity (tyUnitVerbosity targetVerbosity)

Set expected verbosity of the unitTest library.

Parameters

in	*targetVerbosity*	- either VERBOSITY_SILENT,VERBOSITY_QUIET,VERBOSITY-_DIFFS,VERBOSITY_ALL

Returns

void

5.20 VCS Test Environment API

Set of functions to allow test case interaction with the VCS Test Environment.

Functions

- void printInt (u32 value)

  Quickly display a single 32 bit unsigned value in the VCS output console.
- void printMsgInt (const char *msg, u32 value)

  Quickly display a msg followed by a 32 bit value.
- void testStateSet (u32 value)

  This function forces the AHB monitor register debug_test_state to a specific value.
- void testStateInc (void)

  This function increments the AHB monitor register debug_test_state to a specific value.
- void testStateAdd (u32 value)

  This function adds a value to the AHB monitor register debug_test_state to a specific value.
- void displayRawMemory (void *address, u32 length)

  This function does a dump to screen of the contents of a section of CMX memory.
- void dumpMemoryToFile (u32 address, u32 length)

  This function does a dump to screen the contents of a CMX memory range.
- void saveMemoryToFile (u32 address, u32 length, const char *fileName)

  This function does a dump to a file the contents of a memory range.
- void loadMemFromFile (char *pFileNameOpt, u32 optIndex, u32 fileOffset, u32 bytesToLoad, void *targetLoadAddress)

  This function loads some of all of a binary file into memory.
- void vcsHookFastMemCpy (void *dst, void *src, u32 length)
- void vcsHookFastMemSet (void *dst, u32 value, u32 length)
- void vcsHookVerilogEventTrigger (u32 eventCode)

  Trigger a verilog event (e.g. Power monitor)
- void vcsFastPuts (char *pString)

  Fast version of puts.
- void vcsHookFunctionCallParam6 (u32 function, u32 param1, u32 param2, u32 param3, u32 param4, u32 param5, u32 param6)

  This function implements a mechanism by which messages can be passed from software to the VCS hardware SOC simulator.

5.20.1 Detailed Description

Set of functions to allow test case interaction with the VCS Test Environment. This module allows the use of optimised routines which speed up simulation of test cases.

5.20.2 Function Documentation

void displayRawMemory (void * address, u32 length)

This function does a dump to screen of the contents of a section of CMX memory.

The results will be seen in the vcs run log and the memory will be displayed in its native 128 bit format from: addr -> [127..0]

Note

The address must be aligned to a 16 byte boundary

Parameters

in	*address*	- start address in RAM
in	*length*	- length of section to display

Returns

void

void dumpMemoryToFile (u32 address, u32 length)

This function does a dump to screen the contents of a CMX memory range.

The output is address followed by value. e.g.: Addr: 7020e900 Val: 00000000000000000000000000000000

The dump will be in little endian byte order as per system endianness.

Note

This function needs to be updated to add DDR support.

Parameters

in	*address*	- start address in CMX must have 16 byte alignment
in	*length*	- length of section to display

Returns

void

void loadMemFromFile (char * pFileNameOpt, u32 optIndex, u32 fileOffset, u32 bytesToLoad, void * targetLoadAddress)

This function loads some of all of a binary file into memory.

The input file path is relative to simulation folder.

For a less complex version of this see loadMemFromFileSimple.

Modes of operation of this API:

1. Fully qualified filename and path given

- This mode means optIndex = 0
- file loaded is exactly as per the string pFileNameOpt

2. Filename base + index
   - This mode means optIndex > 0
   - File loaded is built from sprintf(filename,"%s_%05d.bin",pFileNameOpt,optIndex)
   - Note: In this chase pFileNameOpt only represents the base of the filename
3. No Filename string passed and optIndex = 0
   - This mode means pFileNameOpt = 0
   - Filename is hardcoded to vector/vector_in.bin
4. No Filename string passed and optIndex > 0
   - This mode means pFileNameOpt = 0
   - Filename is hardcoded to sprintf(filename,"vector/vector_%05d.bin",optIndex)

Note

This function needs to be updated to add DDR support.

Parameters

in	*pFileNameOpt*	– See above for description
in	*optIndex*	– See above for description
in	*fileOffset*	– if this parameter is non zero the load occurs from fileOffset within the target file
in	*bytesToLoad*	– Number of bytes to load from file into memory. If this param is 0 then the whole file is loaded
in	*targetLoad-Address*	– Address in memory where the memory should be loaded

Returns

void

void printInt (u32 value)

Quickly display a single 32 bit unsigned value in the VCS output console.

Parameters

in	*value*	- 32 bit value to print

Returns

void

void printMsgInt (const char * msg, u32 value)

Quickly display a msg followed by a 32 bit value.

Parameters

in	*msg*	- string containing message to display (without)
in	*value*	- 32 bit value to print

Returns

void

void saveMemoryToFile (u32 address, u32 length, const char * fileName)

This function does a dump to a file the contents of a memory range.

The output file is written in the simulation folder.

File name is 3 digit index_address_len_LE.bin e.g.: 001_0xA0000000_0x00000008_LE.bin on first call 002_0xA0000000_0x00000008_LE.bin on second call etc.

The dump will be in little endian byte order as per system endianness.

Note

The file will be in little endian byte order as per system endianness

Parameters

in	*address*	- start address in CMX
in	*length*	- length of section to display
in	*fileName*	- file name

Returns

void

void testStateAdd (u32 value)

This function adds a value to the AHB monitor register debug_test_state to a specific value.

This function can be used to make it quicker to find a point of interest in waves. In cases where multiple tests are run, the user can call this function to add to the debug counter (which starts from 0). The debug_test_state variable can be monitored here: tc_fragrak.leon_ahb_monitor_i.debug_test_state in dve.

Parameters

in	*value*	- 32 bit value to use

Returns

void

void testStateInc (void)

This function increments the AHB monitor register debug_test_state to a specific value.

This function can be used to make it quicker to find a point of interest in waves. In cases where multiple tests are run, the user can call this function to increment the debug counter (which starts from 0). The debug_test_state variable can be monitored here: tc_fragrak.leon_ahb_monitor_i.debug_test_state in dve.

Input: 32 bit value to use

Returns

void

void testStateSet (u32 value)

This function forces the AHB monitor register debug_test_state to a specific value.

This function can be used to make it quicker to find a point of interest in waves. The user simply calls the fuction with a value and then monitors: tc_fragrak.leon_ahb_monitor_i.debug_test_state in dve.

Parameters

in	*value*	- 32 bit value to use

Returns

void

void vcsFastPuts (char * pString)

Fast version of puts.

Uses direct access to memory to avoid the need for leon looping through every byte

Note

The reason this isn't a direct replacement for puts is that the user needs to be very careful about cache-coherency. This issue means it is not a good fit for general purpose use.
Specialist use only

Parameters

in	*pString*	- Address of string to be printed in memory

Returns

void

void vcsHookFastMemCpy (void * dst, void * src, u32 length)

This function performs a fast memory copy in verilog

Operates on a byte basis. There are no alignment requirements

Note

Beware when using this function during profiling as this function behaves like an instant memory copy and takes no time. It is only intendend as a performnace boost during test development and should be replaced with DMA operations when tests are running to be representative of how the chip will really operate.
This function operates on the native memory. The user must ensure cache coherency.

Parameters

in	*dst*	- Address of destination buffer in RAM
in	*src*	- Address of source buffer in RAM
in	*length*	- size of copy in bytes

Returns

void

void vcsHookFastMemSet (void * dst, u32 value, u32 length)

This function performs a fast memory set in verilog

Operates on a byte basis. There are no alignment requirements

Note

Beware when using this function during profiling as this function behaves like an instant memory set and takes no time.
This function operates on the native memory. The user must ensure cache coherency.

Parameters

in	*dst*	- Address of destination buffer in RAM
in	*value*	- Value to set each byte to
in	*length*	- Number of bytes to set

Returns

void

void vcsHookFunctionCallParam6 (u32 function, u32 param1, u32 param2, u32 param3, u32 param4, u32 param5, u32 param6)

This function implements a mechanism by which messages can be passed from software to the VCS hardware SOC simulator.

Note

On real silicon calls to this function have no effect.

Returns

void

void vcsHookVerilogEventTrigger (u32 eventCode)

Trigger a verilog event (e.g. Power monitor)

Singals verilog testbench using an event code

Parameters

in	*eventCode*	- (e.g. VCS_HOOK_EVENT_POWER_MONITOR)

Returns

void

5.21 Debug Tracer

Debug Tracer module API.

Macros

- #define DEBUG_LOG_LEVEL_LOW LOG_LEVEL_INFO
- #define DEBUG_LOG_LEVEL_MEDIUM LOG_LEVEL_WARNING
- #define DEBUG_LOG_LEVEL_HIGH LOG_LEVEL_ERROR

5.21.1 Detailed Description

Debug Tracer module API. Header abstract API for debug trace logging

5.21.2 Macro Definition Documentation

#define DEBUG_LOG_LEVEL_HIGH LOG_LEVEL_ERROR

#define DEBUG_LOG_LEVEL_LOW LOG_LEVEL_INFO

#define DEBUG_LOG_LEVEL_MEDIUM LOG_LEVEL_WARNING

Chapter 6

Data Structure Documentation

6.1 _SwLink Struct Reference

`#include <OpipeDefs.h>`

Data Fields

- void * pipeRef
  ref to pipe it belongs to
- uint32_t prodId
  producer ID
- uint32_t allConsIdMask
  consumers ID mask
- uint32_t lastConsId
  last consumer ID
- PBuffer prodMon
  producer monitor
- PBuffer lastConsMon
  last consumer monitor(decrements producer OBFL)

6.1.1 Field Documentation

uint32_t _SwLink::allConsIdMask

consumers ID mask

uint32_t _SwLink::lastConsId

last consumer ID

PBuffer _SwLink::lastConsMon

last consumer monitor(decrements producer OBFL)

void* _SwLink::pipeRef

ref to pipe it belongs to

uint32_t _SwLink::prodId

producer ID

PBuffer _SwLink::prodMon

producer monitor

6.2 AeAwbCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- uint32_t firstPatchX
- uint32_t firstPatchY
- uint32_t patchWidth
- uint32_t patchHeight
- uint32_t patchGapX
- uint32_t patchGapY
- uint32_t nPatchesX
- uint32_t nPatchesY
- uint16_t darkThresh
- uint16_t brightThresh

6.2.1 Field Documentation

uint16_t AeAwbCfg::brightThresh

uint16_t AeAwbCfg::darkThresh

uint32_t AeAwbCfg::firstPatchX

uint32_t AeAwbCfg::firstPatchY

uint32_t AeAwbCfg::nPatchesX

uint32_t AeAwbCfg::nPatchesY

uint32_t AeAwbCfg::patchGapX

uint32_t AeAwbCfg::patchGapY

uint32_t AeAwbCfg::patchHeight

uint32_t AeAwbCfg::patchWidth

6.3 AeAwbPatchStats Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- uint32_t count [4]
- uint32_t accum [4]
- uint32_t altAccum [4]

6.3.1 Field Documentation

uint32_t AeAwbPatchStats::accum[4]

uint32_t AeAwbPatchStats::altAccum[4]

uint32_t AeAwbPatchStats::count[4]

6.4 AfCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- uint32_t firstPatchX
- uint32_t firstPatchY
- uint32_t patchWidth
- uint32_t patchHeight
- uint32_t nPatchesX
- uint32_t nPatchesY
- int32_t initialSubtractionValue
- int32_t f1Threshold
- int32_t f2Threshold
- int32_t f1Coeffs [11]
- int32_t f2Coeffs [11]

6.4.1 Field Documentation

int32_t AfCfg::f1Coeffs[11]

int32_t AfCfg::f1Threshold

int32_t AfCfg::f2Coeffs[11]

int32_t AfCfg::f2Threshold

uint32_t AfCfg::firstPatchX

uint32_t AfCfg::firstPatchY

int32_t AfCfg::initialSubtractionValue

uint32_t AfCfg::nPatchesX

uint32_t AfCfg::nPatchesY

uint32_t AfCfg::patchHeight

uint32_t AfCfg::patchWidth

6.5 AfPatchStats Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- int32_t UNDEFINED
- int32_t sum_all_green
- int32_t filter1_sum_max_green
- int32_t filter2_sum_max_green
- int32_t filter1_number_of_used_pixels_green
- int32_t filter1_sum_green
- int32_t filter2_number_of_used_pixels_green
- int32_t filter2_sum_green

6.5.1 Field Documentation

int32_t AfPatchStats::filter1_number_of_used_pixels_green

int32_t AfPatchStats::filter1_sum_green

int32_t AfPatchStats::filter1_sum_max_green

int32_t AfPatchStats::filter2_number_of_used_pixels_green

int32_t AfPatchStats::filter2_sum_green

int32_t AfPatchStats::filter2_sum_max_green

int32_t AfPatchStats::sum_all_green

int32_t AfPatchStats::UNDEFINED

6.6 BlcCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- uint32_t gr
- uint32_t r
- uint32_t b
- uint32_t gb

6.6.1 Field Documentation

uint32_t BlcCfg::b

uint32_t BlcCfg::gb

uint32_t BlcCfg::gr

uint32_t BlcCfg::r

6.7 ChromaDnsCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- uint32_t th_r
- uint32_t th_g
- uint32_t th_b
- uint32_t limit
- uint32_t hEnab
- uint32_t greyDesatSlope
- int32_t greyDesatOffset
- uint32_t greyCr
- uint32_t greyCg
- uint32_t greyCb
- uint32_t convCoeffCenter
- uint32_t convCoeffEdge
- uint32_t convCoeffCorner

6.7.1 Field Documentation

uint32_t ChromaDnsCfg::convCoeffCenter

uint32_t ChromaDnsCfg::convCoeffCorner

uint32_t ChromaDnsCfg::convCoeffEdge

uint32_t ChromaDnsCfg::greyCb

uint32_t ChromaDnsCfg::greyCg

uint32_t ChromaDnsCfg::greyCr

int32_t ChromaDnsCfg::greyDesatOffset

uint32_t ChromaDnsCfg::greyDesatSlope

uint32_t ChromaDnsCfg::hEnab

uint32_t ChromaDnsCfg::limit

uint32_t ChromaDnsCfg::th_b

uint32_t ChromaDnsCfg::th_g

uint32_t ChromaDnsCfg::th_r

6.8 ChromaGenCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- uint32_t epsilon
- uint32_t kr
- uint32_t kg
- uint32_t kb
- uint32_t lumaCoeffR
- uint32_t lumaCoeffG
- uint32_t lumaCoeffB
- uint32_t pfrStrength
- int32_t desatOffset
- uint32_t desatSlope

6.8.1 Field Documentation

int32_t ChromaGenCfg::desatOffset

uint32_t ChromaGenCfg::desatSlope

uint32_t ChromaGenCfg::epsilon

uint32_t ChromaGenCfg::kb

uint32_t ChromaGenCfg::kg

uint32_t ChromaGenCfg::kr

uint32_t ChromaGenCfg::lumaCoeffB

uint32_t ChromaGenCfg::lumaCoeffG

uint32_t ChromaGenCfg::lumaCoeffR

uint32_t ChromaGenCfg::pfrStrength

6.9 client_tx_frame_header_t Struct Reference

```
#include <sendOutApi.h>
```

Data Fields

- uint32_t frame_type
- uint32_t frame_format
- uint32_t frame_width
- uint32_t frame_height
- uint32_t frame_time_stamp_hi
- uint32_t frame_time_stamp_lo
- uint32_t frame_proc_time_stamp_hi
- uint32_t frame_proc_time_stamp_lo
- uint32_t frame_idx_req_hal
- uint32_t frame_idx_req_app
- uint32_t frame_idx_mipi_rx
- uint32_t frame_idx_process
- uint32_t header_height
- uint32_t slice_data_type
- uint32_t slice_y_offset
- uint32_t slice_y_size
- uint32_t slice_uv_offset
- uint32_t slice_uv_size
- uint32_t slice_total_number
- uint32_t slice_last_flag
- uint32_t debug_data_enable
- uint32_t camera_id
- uint32_t buff_width
- uint32_t buff_height
- uint32_t buff_stride
- uint32_t buff_pxl_size_nom
- uint32_t buff_pxl_size_denom
- uint32_t check_sum

6.9.1 Field Documentation

uint32_t client_tx_frame_header_t::buff_height

uint32_t client_tx_frame_header_t::buff_pxl_size_denom

uint32_t client_tx_frame_header_t::buff_pxl_size_nom

uint32_t client_tx_frame_header_t::buff_stride

uint32_t client_tx_frame_header_t::buff_width

uint32_t client_tx_frame_header_t::camera_id

uint32_t client_tx_frame_header_t::check_sum

uint32_t client_tx_frame_header_t::debug_data_enable

uint32_t client_tx_frame_header_t::frame_format

uint32_t client_tx_frame_header_t::frame_height

uint32_t client_tx_frame_header_t::frame_idx_mipi_rx

uint32_t client_tx_frame_header_t::frame_idx_process

uint32_t client_tx_frame_header_t::frame_idx_req_app

uint32_t client_tx_frame_header_t::frame_idx_req_hal

uint32_t client_tx_frame_header_t::frame_proc_time_stamp_hi

uint32_t client_tx_frame_header_t::frame_proc_time_stamp_lo

uint32_t client_tx_frame_header_t::frame_time_stamp_hi

uint32_t client_tx_frame_header_t::frame_time_stamp_lo

uint32_t client_tx_frame_header_t::frame_type

uint32_t client_tx_frame_header_t::frame_width

uint32_t client_tx_frame_header_t::header_height

uint32_t client_tx_frame_header_t::slice_data_type

uint32_t client_tx_frame_header_t::slice_last_flag

uint32_t client_tx_frame_header_t::slice_total_number

uint32_t client_tx_frame_header_t::slice_uv_offset

uint32_t client_tx_frame_header_t::slice_uv_size

uint32_t client_tx_frame_header_t::slice_y_offset

uint32_t client_tx_frame_header_t::slice_y_size

6.10 ColCombCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- float ccm [9]
- float ccmOff [3]
- uint32_t kr
- uint32_t kg
- uint32_t kb
- uint16_t * lut3D

6.10.1 Field Documentation

float ColCombCfg::ccm[9]

float ColCombCfg::ccmOff[3]

uint32_t ColCombCfg::kb

uint32_t ColCombCfg::kg

uint32_t ColCombCfg::kr

uint16_t* ColCombCfg::lut3D

6.11 ColConvCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- float mat [3 *3]
- float offset [3]

6.11.1 Field Documentation

float ColConvCfg::mat[3 *3]

float ColConvCfg::offset[3]

6.12 ConvCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- uint16_t mat5x5 [5 ∗5]

6.12.1 Field Documentation

uint16_t ConvCfg::mat5x5[5 ∗5]

6.13 DBuffer Struct Reference

`#include <OpipeDefs.h>`

Data Fields

- SBuffer cmx
  cmx view [mandatory]
- MBuffer ddr
  ddr view [optional, just for sinks/sources]
- void ∗ pipeRef
  ref to pipe it belongs to
- uint32_t unitID
  unit ID of the filter that gets fed
- uint32_t dir
  direction: IN/OUT: only relevant for SRK/SNK
- uint32_t sippBuffBase
  associated Sipp IOBuff Reg addr
- uint32_t nPlanes
  num planes
- uint32_t fmt
  bytes / pix format
- uint32_t irqRatePow
  IRQ rate of SIPP filter.
- uint32_t irqRate
  derived as 1<<irqRatePow

6.13.1 Detailed Description

[Full-DDR] + [circular-CMX] circular buffer pair DDR is only used for sink/source constructs

6.13.2 Field Documentation

SBuffer DBuffer::cmx

cmx view [mandatory]

MBuffer DBuffer::ddr

ddr view [optional, just for sinks/sources]

uint32_t DBuffer::dir

direction: IN/OUT: only relevant for SRK/SNK

uint32_t DBuffer::fmt

bytes / pix format

uint32_t DBuffer::irqRate

derived as 1<<irqRatePow

uint32_t DBuffer::irqRatePow

IRQ rate of SIPP filter.

uint32_t DBuffer::nPlanes

num planes

void* DBuffer::pipeRef

ref to pipe it belongs to

uint32_t DBuffer::sippBuffBase

associated Sipp IOBuff Reg addr

uint32_t DBuffer::unitID

unit ID of the filter that gets fed

6.14 DbyrCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- int32_t dewormGradientMul
- uint32_t dewormSlope
- int32_t dewormOffset
- int32_t lumaWeightR
- int32_t lumaWeightG
- int32_t lumaWeightB

6.14.1 Field Documentation

int32_t DbyrCfg::dewormGradientMul

int32_t DbyrCfg::dewormOffset

uint32_t DbyrCfg::dewormSlope

int32_t DbyrCfg::lumaWeightB

int32_t DbyrCfg::lumaWeightG

int32_t DbyrCfg::lumaWeightR

6.15 DogCfg Struct Reference

```
#include <OpipeBlocks.h>
```

Data Fields

- uint32_t thr
- uint32_t strength
- float sigma11
- float sigma15
- uint8_t coeffs11 [6]
- uint8_t coeffs15 [8]

6.15.1 Field Documentation

uint8_t DogCfg::coeffs11[6]

uint8_t DogCfg::coeffs15[8]

float DogCfg::sigma11

float DogCfg::sigma15

uint32_t DogCfg::strength

uint32_t DogCfg::thr

6.16 eventQueue_t Struct Reference

An event queue type.

`#include <eventQueue.h>`

Data Fields

- eventQueueItem_t * head
  Event queue item head.
- eventQueueItem_t * tail
  Event queue item tail.
- u32 nItems
  Number of items.

6.16.1 Detailed Description

An event queue type.

6.16.2 Field Documentation

eventQueueItem_t* eventQueue_t::head

Event queue item head.

u32 eventQueue_t::nItems

Number of items.

eventQueueItem_t* eventQueue_t::tail

Event queue item tail.

6.17 eventQueueItem_t Struct Reference

An event queue item type.

`#include <eventQueue.h>`

Data Fields

- volatile eventType_t type
  Type of this event.
- void * data

Private data for this event.

- u64 timestamp

  Timestamp of this event.
- struct eventQueueItem_t * next

  Next pointer for event queue.
- struct eventQueue_t * parent

  Which list we are in.

Event Flags

The following three flags should be modified through eventLoop interfaces only (check eventLoop.h)

- volatile u8 requiredFlags

  Flags to indicate required callbacks for the event processing.
- volatile u8 busyFlags

  Flags to indicate event is busy with a specific event handling.
- volatile u8 doneFlags

  Flags to indicate that a specific event handling is done.
- volatile u8 dropFlags

  Flags to indicate that a specific event was dropped by one of the callbacks.

6.17.1 Detailed Description

An event queue item type.

6.17.2 Field Documentation

volatile u8 eventQueueItem_t::busyFlags

Flags to indicate event is busy with a specific event handling.

void* eventQueueItem_t::data

Private data for this event.

volatile u8 eventQueueItem_t::doneFlags

Flags to indicate that a specific event handling is done.

volatile u8 eventQueueItem_t::dropFlags

Flags to indicate that a specific event was dropped by one of the callbacks.

struct **eventQueueItem_t*** eventQueueItem_t::next

Next pointer for event queue.

struct **eventQueue_t**∗ eventQueueItem_t::parent

Which list we are in.

volatile u8 eventQueueItem_t::requiredFlags

Flags to indicate required callbacks for the event processing.

u64 eventQueueItem_t::timestamp

Timestamp of this event.

volatile eventType_t eventQueueItem_t::type

Type of this event.

6.18 FeatMaintenance Class Reference

`#include <featureMaintenanceApi.h>`

Public Member Functions

- FeatMaintenance ()
- ∼FeatMaintenance ()
- void fmRun (const u32 frameNum, const frameBuffer ∗const fullFrame, const tvXYLoc features-Tmp[ ], const fp32 featuresErrorTmp[ ], FMFeatureThresholds_t ∗thresholds)
- void fmInit (FMSetupCfg ∗cfg, fmResourceCfg_t ∗resCfg)

Private Member Functions

- DynamicContext_t fmContext ALIGNED (64)
- DynamicContextInstances_elm fmContextInstanceData ALIGNED (64)
- DynamicContextInstancesPtr fmContextInstanceDataPtr ALIGNED (64)

Private Attributes

- uint32_t fmShaveNum
- uint8_t featPipeCacheData
- uint8_t featPipeCacheInstr
- FMSetupCfg g_fmCfg

6.18.1 Constructor & Destructor Documentation

FeatMaintenance::FeatMaintenance ( )

FeatMaintenance::~FeatMaintenance ( )

6.18.2 Member Function Documentation

DynamicContext_t fmContext FeatMaintenance::ALIGNED (64) [private]

DynamicContextInstances_elm fmContextInstanceData FeatMaintenance::ALIGNED (64) [private]

DynamicContextInstancesPtr fmContextInstanceDataPtr FeatMaintenance::ALIGNED (64) [private]

void FeatMaintenance::fmInit (FMSetupCfg * cfg, **fmResourceCfg_t** * resCfg)

void FeatMaintenance::fmRun (const u32 frameNum, const frameBuffer *const fullFrame, const tvXYLoc featuresTmp[], const fp32 featuresErrorTmp[], FMFeatureThresholds_t * thresholds)

6.18.3 Field Documentation

uint8_t FeatMaintenance::featPipeCacheData [private]

uint8_t FeatMaintenance::featPipeCacheInstr [private]

uint32_t FeatMaintenance::fmShaveNum [private]

FMSetupCfg FeatMaintenance::g_fmCfg [private]

6.19 fmResourceCfg Struct Reference

```
#include <featureMaintenanceApi.h>
```

Data Fields

- swcShaveUnit_t shaveNum
- uint8_t cachePartData
- uint8_t cachePartInstr

6.19.1 Field Documentation

uint8_t fmResourceCfg::cachePartData

uint8_t fmResourceCfg::cachePartInstr

swcShaveUnit_t fmResourceCfg::shaveNum

6.20 HarrisCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- uint32_t cfg
- u32f32 kValue

6.20.1 Field Documentation

uint32_t HarrisCfg::cfg

u32f32 HarrisCfg::kValue

6.21 ipipeServerInfo Struct Reference

`#include <IpipeServerApi.h>`

Data Fields

- void(∗ cbIcSetup)(icCtrl ∗ctrl)
- void(∗ cbIcReset)(void)
- void(∗ cbIcTearDown)(void)
- SourceServerCtrlT sourceServerCtrl [IPIPE_MAX_SOURCES_ALLOWED]
- PluginServerCtrl pluginServerCtrl [IPIPE_MAX_ISP_ALlOWED]
- TrigerCaptQue trigerCaptQue
- icStatusCode(∗ cbSourcesCommit)(icCtrl ∗ctrl)
- void(∗ cbDataWasSent)(FrameT ∗dataBuffer)
- uint32_t memFree
- void(∗ cbIcUserMsg)(void ∗eventStruct, uint32_t id)

6.21.1 Field Documentation

void(∗ ipipeServerInfo::cbDataWasSent)(FrameT ∗dataBuffer)

void(∗ ipipeServerInfo::cbIcReset)(void)

void(∗ ipipeServerInfo::cbIcSetup)(icCtrl ∗ctrl)

void(∗ ipipeServerInfo::cbIcTearDown)(void)

void(∗ ipipeServerInfo::cbIcUserMsg)(void ∗eventStruct, uint32_t id)

icStatusCode(∗ ipipeServerInfo::cbSourcesCommit)(icCtrl ∗ctrl)

uint32_t ipipeServerInfo::memFree

PluginServerCtrl ipipeServerInfo::pluginServerCtrl[IPIPE_MAX_ISP_ALlOWED]

SourceServerCtrlT ipipeServerInfo::sourceServerCtrl[IPIPE_MAX_SOURCES_ALLOWED]

TrigerCaptQue ipipeServerInfo::trigerCaptQue

6.22 LscCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- uint32_t lscWidth
- uint32_t lscHeight
- uint32_t lscStride
- uint16_t * pLscTable

6.22.1 Field Documentation

uint32_t LscCfg::lscHeight

uint32_t LscCfg::lscStride

uint32_t LscCfg::lscWidth

uint16_t* LscCfg::pLscTable

6.23 LtmCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- uint32_t thr
- uint16_t curves [16 *8]

6.23.1 Field Documentation

uint16_t LtmCfg::curves[16 *8]

uint32_t LtmCfg::thr

6.24 LumaDnsCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- float strength
- uint32_t alpha
- uint32_t bitpos
- uint8_t lut [32]
- uint32_t f2

6.24.1 Field Documentation

uint32_t LumaDnsCfg::alpha

uint32_t LumaDnsCfg::bitpos

uint32_t LumaDnsCfg::f2

uint8_t LumaDnsCfg::lut[32]

float LumaDnsCfg::strength

6.25 LumaDnsRefCfg Struct Reference

```
#include <OpipeBlocks.h>
```

Data Fields

- float gamma
- float angle_of_view
- uint64_t lutDist [256/8]
- uint8_t lutGamma0_32 [9]
- uint8_t lutGamma32_255 [9]
- uint32_t shift

6.25.1 Field Documentation

float LumaDnsRefCfg::angle_of_view

float LumaDnsRefCfg::gamma

uint64_t LumaDnsRefCfg::lutDist[256/8]

uint8_t LumaDnsRefCfg::lutGamma0_32[9]

uint8_t LumaDnsRefCfg::lutGamma32_255[9]

uint32_t LumaDnsRefCfg::shift

6.26 LutCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- uint32_t rangetop
- uint32_t size
- uint32_t rgnSize [2]
- uint16_t * table

6.26.1 Field Documentation

uint32_t LutCfg::rangetop

uint32_t LutCfg::rgnSize[2]

uint32_t LutCfg::size

uint16_t* LutCfg::table

6.27 MBuffer Struct Reference

`#include <OpipeDefs.h>`

Data Fields

- uint32_t base
  buffer base addr in CMX
- uint32_t height
  buffer height for circular progress
- uint32_t lineStride
  aux: line stride [bytes] = 8Byte aligned version of (lineW)
- uint32_t lineNo
- uint32_t lineW
  line width in Bytes
- uint32_t cpLines
  number of lines that actually got coppied in/out by DMA
- dmaTransactionList dmaDsc [N_DESCS]
  Dma descriptors.
- uint32_t curDsc
  cur descriptor [0..N_DESCS-1]

6.27.1 Field Documentation

uint32_t MBuffer::base

buffer base addr in CMX

uint32_t MBuffer::cpLines

number of lines that actually got coppied in/out by DMA

uint32_t MBuffer::curDsc

cur descriptor [0..N_DESCS-1]

dmaTransactionList MBuffer::dmaDsc[**N_DESCS**]

Dma descriptors.

uint32_t MBuffer::height

buffer height for circular progress

uint32_t MBuffer::lineNo

uint32_t MBuffer::lineStride

aux: line stride [bytes] = 8Byte aligned version of (lineW)

uint32_t MBuffer::lineW

line width in Bytes

6.28 MedianCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- uint32_t kernelSize
- uint32_t slope
- int32_t offset

6.28.1 Field Documentation

uint32_t MedianCfg::kernelSize

int32_t MedianCfg::offset

uint32_t MedianCfg::slope

6.29 MessageRingBuffer Struct Reference

`#include <MessRingBuff.h>`

Data Fields

- void * allMem
- s32 size
- s32 availableLength
- s32 activeWriteSize
- void * wrPtr
- void * rdPtr
- void * endPtr

6.30 mestHandler_t Struct Reference

`#include <MestApi.h>`

Data Fields

- void * privateData

6.30.1 Field Documentation

void* mestHandler_t::privateData

6.31 MipiRxCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- uint32_t cfg
- uint32_t xStartWidth [4]
- uint32_t yStartHeight [4]
- uint32_t sel01
- uint32_t sel23
- uint32_t mask [4]
- uint32_t black01
- uint32_t black23
- uint32_t vbp

6.31.1 Field Documentation

uint32_t MipiRxCfg::black01

uint32_t MipiRxCfg::black23

uint32_t MipiRxCfg::cfg

uint32_t MipiRxCfg::mask[4]

uint32_t MipiRxCfg::sel01

uint32_t MipiRxCfg::sel23

uint32_t MipiRxCfg::vbp

uint32_t MipiRxCfg::xStartWidth[4]

uint32_t MipiRxCfg::yStartHeight[4]

6.32 ofResourceCfg Struct Reference

`#include <opticalFlowApi.h>`

Data Fields

- u32 numShaves
- swcShaveUnit_t * shaveList
- u32 ** initEntry
- u32 ** runEntry
- uint8_t cachePartData
- uint8_t cachePartInstr

6.32.1 Field Documentation

uint8_t ofResourceCfg::cachePartData

uint8_t ofResourceCfg::cachePartInstr

u32** ofResourceCfg::initEntry

u32 ofResourceCfg::numShaves

u32** ofResourceCfg::runEntry

swcShaveUnit_t* ofResourceCfg::shaveList

6.33 oMipiTxLoopbackParam Struct Reference

Mipi-TX loopback debug params.

```
#include <OpipeDefs.h>
```

Data Fields

- uint32_t txID
  SIPP_MIPI_TX0_ID or SIPP_MIPI_TX1_ID.
- uint8_t * imgAddr
  full image base address
- uint32_t imgW
  full image width
- uint32_t imgH
  full image height
- uint32_t bpp
  bytes per pixel
- uint32_t hbp
  timing: horizontal back porch [pclk]
- uint32_t hfp
  timing: horizontal front porch [pclk]
- uint32_t hsync
  timing: horizontal sync [pclk]
- uint32_t vsync
  timing: vertical sync [lines]

6.33.1 Detailed Description

Mipi-TX loopback debug params.

6.33.2 Field Documentation

uint32_t oMipiTxLoopbackParam::bpp

bytes per pixel

uint32_t oMipiTxLoopbackParam::hbp

timing: horizontal back porch [pclk]

uint32_t oMipiTxLoopbackParam::hfp

timing: horizontal front porch [pclk]

uint32_t oMipiTxLoopbackParam::hsync

timing: horizontal sync [pclk]

uint8_t* oMipiTxLoopbackParam::imgAddr

full image base address

uint32_t oMipiTxLoopbackParam::imgH

full image height

uint32_t oMipiTxLoopbackParam::imgW

full image width

uint32_t oMipiTxLoopbackParam::txID

SIPP_MIPI_TX0_ID or SIPP_MIPI_TX1_ID.

uint32_t oMipiTxLoopbackParam::vsync

timing: vertical sync [lines]

6.34 OpipeGlobal Struct Reference

`#include <OpipeDefs.h>`

Data Fields

- uint32_t unitsInUse
- uint32_t cdmaReq
- uint32_t nSources
- uint32_t nSinks
- uint32_t nSwLinks
- uint32_t nPipes
- DBuffer * sources [32]
- DBuffer * sinks [32]
- SwLink * swCtrl [32]

6.34.1 Field Documentation

uint32_t OpipeGlobal::cdmaReq

uint32_t OpipeGlobal::nPipes

uint32_t OpipeGlobal::nSinks

uint32_t OpipeGlobal::nSources

uint32_t OpipeGlobal::nSwLinks

DBuffer∗ OpipeGlobal::sinks[32]

DBuffer∗ OpipeGlobal::sources[32]

SwLink∗ OpipeGlobal::swCtrl[32]

uint32_t OpipeGlobal::unitsInUse

6.35 OpipeS Struct Reference

Main O-Pipe data struct.

`#include <OpipeDefs.h>`

Data Fields

- uint32_t oldW
  prev W for res internal adjustment
- uint32_t oldH
  prev H for res internal adjustment
- uint32_t width
  Used to compute XXX_FRM_DIM above DBYR (including)
- uint32_t height
  Used to compute XXX_FRM_DIM above DBYR (including)
- uint32_t fullW
  Used to by lumaDns.
- uint32_t fullH
  Used to by lumaDns.
- uint32_t offX
  Used to by lumaDns.
- uint32_t offY
  Used to by lumaDns.
- uint32_t width2
  Used to compute XXX_FRM_DIM below DBYR.
- uint32_t height2
  Used to compute XXX_FRM_DIM below DBYR.
- uint32_t unitIDs
  Mask containing bit-enables for all filters used in this pipeline.
- uint32_t running
  flag = 1 if pipe is in progress
- uint32_t irqRate
  default to 8
- uint32_t format
  Bayer/Planar (shared by bayer blocks)
- uint32_t bayerPattern

Relevant if format == BAYER.

- uint32_t rawBits

  Bayer path bits (shared by bayer blocks)
- uint32_t enMask

  Filters enable mask (or-ed into SIPP_CONTROL_ADR)
- uint32_t id

  app general purpose indentification
- void * params [8]

  app general purpose params
- uint32_t cfg [SIPP_F_NUM]

  user CFG override for all filters
- uint8_t iPlanes [SIPP_F_NUM]

  Input planes (for each unit)
- uint8_t oPlanes [SIPP_F_NUM]

  Output planes (for each unit), mostly unused as matches iPlanes.
- uint32_t triggerSinkId

  SIPP ID of filter that can generate the line callbacks.
- uint32_t currHitLine

  notify user what is current hit line
- uint16_t targetLine [4]

  up to 4 lines per frame that can trigger the line-CB
- opipeCb cbLineHit

  line-reached callback (OPTIONAL)
- opipeCb cbEndOfFrame

  end of frame callback (OPTIONAL)
- opipeCb cbPreStart

  (OPTIONAL)
- BlcCfg * pBlcCfg

  Black Level Correction params.
- SigmaDnsCfg * pSigmaCfg

  Sigma Denoise params.
- LscCfg * pLscCfg

  Lens Shading params.
- RawCfg * pRawCfg

  Raw params.
- DbyrCfg * pDbyrCfg

  Debayer params.
- LtmCfg * pLtmCfg

  Local Tone Mapping params.
- DogCfg * pDogCfg

  Difference Of Gaussians params.
- LumaDnsCfg * pLumaDnsCfg

  Luma Denoise params.
- LumaDnsRefCfg * pLumaDnsRefCfg

  Luma Denoise Reference params.

- SharpenCfg * pSharpCfg

  Luma Sharpen params.
- ChromaGenCfg * pChrGenCfg

  Chroma Generation params.
- MedianCfg * pMedCfg

  Median params.
- ChromaDnsCfg * pChromaDnsCfg

  Chroma Denoise params.
- ColCombCfg * pColCombCfg

  Color Combination params.
- LutCfg * pLutCfg

  Lookup table params.
- ColConvCfg * pColConvCfg

  Color conversion params.
- ConvCfg * pConvCfg

  Convolution params.
- UpfirdnCfg * pUpfirdn0Cfg

  UPFIRDN_0 params.
- UpfirdnCfg * pUpfirdn12Cfg

  UPFIRDN_1 and UPFIRDN_2 shared params.
- MipiRxCfg * pMipiRxCfg [4]

  MipiRx params.
- HarrisCfg * pHarrisCfg

  Harris params.
- AeAwbCfg * aeCfg

  Ae/Awb config struct.
- AfCfg * afCfg

  Af config struct.
- AeAwbPatchStats * aeStats

  Ae/Awb output stats buff.
- AfPatchStats * afStats

  Af output stats buff.
- uint32_t * histLuma

  Luma histogram (256 entries)
- uint32_t * histRgb

  RGB histogram (3x128 entries)
- uint32_t cfgMask

  internally derived mask (to be OR=ed into SIPP_OPIPE_CFG_ADR)
- DBuffer cmxBuffs [16]

  internal
- uint32_t nCmxBuffs

  internal
- uint32_t nSrcs

  internal
- uint32_t nSnks

internal

- uint32_t nSwLinks
- DBuffer * srcs [MAX_SOURCES]

  internal
- DBuffer * snks [MAX_SINKS]

  internal
- SwLink swCtrl [4]
- uint32_t sippSrcIntEnMask

  goes OR-ed into SIPP_INT0_ENABLE_ADR
- uint32_t sippSnkIntEnMask

  goes OR-ed into SIPP_INT1_ENABLE_ADR
- uint32_t flags

  pipe flags
- uint32_t cdmaReq

  CMXDMA requester id (to control CMXDMA agent assignment)

6.35.1 Detailed Description

Main O-Pipe data struct.

6.35.2 Field Documentation

AeAwbCfg * OpipeS::aeCfg

Ae/Awb config struct.

AeAwbPatchStats * OpipeS::aeStats

Ae/Awb output stats buff.

AfCfg * OpipeS::afCfg

Af config struct.

AfPatchStats * OpipeS::afStats

Af output stats buff.

uint32_t OpipeS::bayerPattern

Relevant if format == BAYER.

opipeCb OpipeS::cbEndOfFrame

end of frame callback (OPTIONAL)

opipeCb OpipeS::cbLineHit

line-reached callback (OPTIONAL)

opipeCb OpipeS::cbPreStart

(OPTIONAL)

uint32_t OpipeS::cdmaReq

CMXDMA requester id (to control CMXDMA agent assignment)

uint32_t OpipeS::cfg[SIPP_F_NUM]

user CFG override for all filters

uint32_t OpipeS::cfgMask

internally derived mask (to be OR=ed into SIPP_OPIPE_CFG_ADR)

DBuffer OpipeS::cmxBuffs[16]

internal

uint32_t OpipeS::currHitLine

notify user what is current hit line

uint32_t OpipeS::enMask

Filters enable mask (or-ed into SIPP_CONTROL_ADR)

uint32_t OpipeS::flags

pipe flags

uint32_t OpipeS::format

Bayer/Planar (shared by bayer blocks)

uint32_t OpipeS::fullH

Used to by lumaDns.

uint32_t OpipeS::fullW

Used to by lumaDns.

uint32_t OpipeS::height

Used to compute XXX_FRM_DIM above DBYR (including)

uint32_t OpipeS::height2

Used to compute XXX_FRM_DIM below DBYR.

uint32_t* OpipeS::histLuma

Luma histogram (256 entries)

uint32_t* OpipeS::histRgb

RGB histogram (3x128 entries)

uint32_t OpipeS::id

app general purpose indentification

uint8_t OpipeS::iPlanes[SIPP_F_NUM]

Input planes (for each unit)

uint32_t OpipeS::irqRate

default to 8

uint32_t OpipeS::nCmxBuffs

internal

uint32_t OpipeS::nSnks

internal

uint32_t OpipeS::nSrcs

internal

uint32_t OpipeS::nSwLinks

uint32_t OpipeS::offX

Used to by lumaDns.

uint32_t OpipeS::offY

Used to by lumaDns.

uint32_t OpipeS::oldH

prev H for res internal adjustment

uint32_t OpipeS::oldW

prev W for res internal adjustment

uint8_t OpipeS::oPlanes[SIPP_F_NUM]

Output planes (for each unit), mostly unused as matches iPlanes.

void* OpipeS::params[8]

app general purpose params

BlcCfg* OpipeS::pBlcCfg

Black Level Correction params.

ChromaGenCfg* OpipeS::pChrGenCfg

Chroma Generation params.

ChromaDnsCfg* OpipeS::pChromaDnsCfg

Chroma Denoise params.

ColCombCfg* OpipeS::pColCombCfg

Color Combination params.

ColConvCfg* OpipeS::pColConvCfg

Color conversion params.

ConvCfg∗ OpipeS::pConvCfg

Convolution params.

DbyrCfg∗ OpipeS::pDbyrCfg

Debayer params.

DogCfg∗ OpipeS::pDogCfg

Difference Of Gaussians params.

HarrisCfg∗ OpipeS::pHarrisCfg

Harris params.

LscCfg∗ OpipeS::pLscCfg

Lens Shading params.

LtmCfg∗ OpipeS::pLtmCfg

Local Tone Mapping params.

LumaDnsCfg∗ OpipeS::pLumaDnsCfg

Luma Denoise params.

LumaDnsRefCfg∗ OpipeS::pLumaDnsRefCfg

Luma Denoise Reference params.

LutCfg∗ OpipeS::pLutCfg

Lookup table params.

MedianCfg∗ OpipeS::pMedCfg

Median params.

MipiRxCfg∗ OpipeS::pMipiRxCfg[4]

MipiRx params.

RawCfg* OpipeS::pRawCfg

Raw params.

SharpenCfg* OpipeS::pSharpCfg

Luma Sharpen params.

SigmaDnsCfg* OpipeS::pSigmaCfg

Sigma Denoise params.

UpfirdnCfg* OpipeS::pUpfirdn0Cfg

UPFIRDN_0 params.

UpfirdnCfg* OpipeS::pUpfirdn12Cfg

UPFIRDN_1 and UPFIRDN_2 shared params.

uint32_t OpipeS::rawBits

Bayer path bits (shared by bayer blocks)

uint32_t OpipeS::running

flag = 1 if pipe is in progress

uint32_t OpipeS::sippSnkIntEnMask

goes OR-ed into SIPP_INT1_ENABLE_ADR

uint32_t OpipeS::sippSrcIntEnMask

goes OR-ed into SIPP_INT0_ENABLE_ADR

DBuffer* OpipeS::snks[**MAX_SINKS**]

internal

DBuffer* OpipeS::srcs[**MAX_SOURCES**]

internal

SwLink OpipeS::swCtrl[4]

uint16_t OpipeS::targetLine[4]

up to 4 lines per frame that can trigger the line-CB

uint32_t OpipeS::triggerSinkId

SIPP ID of filter that can generate the line callbacks.

uint32_t OpipeS::unitIDs

Mask containing bit-enables for all filters used in this pipeline.

uint32_t OpipeS::width

Used to compute XXX_FRM_DIM above DBYR (including)

uint32_t OpipeS::width2

Used to compute XXX_FRM_DIM below DBYR.

6.36 OpticalFlow Class Reference

`#include <opticalFlowApi.h>`

Public Member Functions

- OpticalFlow ()
- ~OpticalFlow ()
- void ofInit (tvOpticalFlowCfg *algConfig, ofResourceCfg_t *resCfg)
- int ofRun (tvPyramidBuffer *pyrImgPrev, tvPyramidBuffer *pyrImgCur, tvXYLoc featuresPrev[ ], tvXYLoc featuresCur[ ], fp32 featuresError[ ], u32 featuresCount[ ], u32 numCells, u32 maxFeatPerCell)

Data Fields

- uint8_t * MEST_COORDS_HEAP
- uint8_t * MEST_RES_TMP_HEAP
- uint8_t * MEST_RES_HEAP

Private Member Functions

- DynamicContext_t ofContext ALIGNED (64)
- DynamicContextInstances_elm ofContextInstanceData ALIGNED (64)

- DynamicContextInstancesPtr ofContextInstanceDataPtr ALIGNED (64)

Private Attributes

- u32 numShaves
- swcShaveUnit_t * shaveList
- tvOpticalFlowCfg ofAlgConfig
- uint8_t cacheData
- uint8_t cacheInstr

6.36.1 Constructor & Destructor Documentation

OpticalFlow::OpticalFlow ( )

OpticalFlow::~OpticalFlow ( )

6.36.2 Member Function Documentation

DynamicContext_t ofContext OpticalFlow::ALIGNED (64) [private]

DynamicContextInstances_elm ofContextInstanceData OpticalFlow::ALIGNED (64) [private]

DynamicContextInstancesPtr ofContextInstanceDataPtr OpticalFlow::ALIGNED (64) [private]

void OpticalFlow::ofInit (tvOpticalFlowCfg * algConfig, **ofResourceCfg_t** * resCfg)

int OpticalFlow::ofRun (tvPyramidBuffer * pyrImgPrev, tvPyramidBuffer * pyrImgCur, tvXYLoc featuresPrev[ ], tvXYLoc featuresCur[ ], fp32 featuresError[ ], u32 featuresCount[ ], u32 numCells, u32 maxFeatPerCell)

6.36.3 Field Documentation

uint8_t OpticalFlow::cacheData [private]

uint8_t OpticalFlow::cacheInstr [private]

uint8_t* OpticalFlow::MEST_COORDS_HEAP

uint8_t* OpticalFlow::MEST_RES_HEAP

uint8_t* OpticalFlow::MEST_RES_TMP_HEAP

u32 OpticalFlow::numShaves [private]

tvOpticalFlowCfg OpticalFlow::ofAlgConfig [private]

swcShaveUnit_t* OpticalFlow::shaveList [private]

6.37 OsVirtualChannel Struct Reference

`#include <OsMessageProtocol.h>`

Data Fields

- VirtualChannel bmVC
- rtems_id rxWaitTaskId

6.38 PBuffer Struct Reference

`#include <OpipeDefs.h>`

Data Fields

- uint32_t height
  full buffer height
- uint32_t curLine
  line progress (e.g. DBYR written lines / DOGL read lines)
- uint32_t irqRatePow
  IRQ rate of SIPP filter.
- uint32_t irqRate
  derived as 1<<irqRatePow

6.38.1 Field Documentation

uint32_t PBuffer::curLine

line progress (e.g. DBYR written lines / DOGL read lines)

uint32_t PBuffer::height

full buffer height

uint32_t PBuffer::irqRate

derived as 1<<irqRatePow

uint32_t PBuffer::irqRatePow

IRQ rate of SIPP filter.

6.39 PhysicalChannel Struct Reference

`#include <MessageProtocol.h>`

Data Fields

- ChannelType ct
- void * context

6.40 PixelPipe Class Reference

`#include <PixelPipeApi.h>`

Public Member Functions

- PixelPipe ()
- void ppInit (t_pPipeResourceCfg *vpResource, pyramidAlgoType_t pyrAlg, cornerConfig_t corCfg)
- u32 ppRun (frameBuffer *in_img, tvFeatureCell **feature_cells, tvPyramidBuffer *frameBuffer, fp32 *thresholds, u32 num_pyr_levels, u32 num_pyrs, u32 cellGridDimension, u32 maxNumFeatures, u32 targetNumFeatures, ppThresholds_t *thresholdCfg)

Private Member Functions

- void initPixelPipe (t_pPipeResourceCfg *vpResource, pyramidAlgoType_t pyrAlg, cornerConfig_t corCfg)
- u32 pixelPipe (frameBuffer *in_img, tvFeatureCell **feature_cells, tvPyramidBuffer *frameBuffer, fp32 *thresholds, u32 num_pyr_levels, u32 num_pyrs, u32 cellGridDimension, u32 maxNumFeatures, u32 targetNumFeatures, ppThresholds_t *thresholdCfg)
- u8 shaveBuf[1088] __attribute__ ((aligned(64)))
- float harrisThresholds[MAX_NUM_CELLS] __attribute__ ((aligned(64)))
- DynamicContext_t ppContext ALIGNED (64)
- DynamicContextInstances_elm ppContextInstanceData ALIGNED (64)
- DynamicContextInstancesPtr ppContextInstanceDataPtr ALIGNED (64)
- DynamicContext_t corContext ALIGNED (64)
- DynamicContextInstances_elm corContextInstanceData ALIGNED (64)
- DynamicContextInstancesPtr corContextInstanceDataPtr ALIGNED (64)
- DynamicContext_t gaussContext ALIGNED (64)
- DynamicContextInstances_elm gaussContextInstanceData ALIGNED (64)
- DynamicContextInstancesPtr gaussContextInstanceDataPtr ALIGNED (64)

Private Attributes

- swcShaveUnit_t * gaussShavesList
- swcShaveUnit_t * cornerShavesList

- swcShaveUnit_t ppMasterShaveNum
- swcShaveUnit_t slaveShaves [NUM_SHAVES_SLAVE]
- u32 slaveNumShaves
- u32 gaussNumShaves
- u32 cornerNumShaves
- pixelPipeParams_t * pixelPipeParams
- u8 ppCacheData
- u8 ppCacheInstr
- fifoCommMasterHandler_t * masterHandler
- fifoCommSlaveHandler_t * slaveHandler
- fifoCommTask_t * shaveTaskTypes
- fifoCommTask_t * taskTypes

6.40.1 Constructor & Destructor Documentation

PixelPipe::PixelPipe ( )

6.40.2 Member Function Documentation

u8 shaveBuf [1088] PixelPipe::__attribute__ ((aligned(64))) [private]

float harrisThresholds [MAX_NUM_CELLS] PixelPipe::__attribute__ ((aligned(64))) [private]

DynamicContext_t ppContext PixelPipe::ALIGNED (64) [private]

DynamicContextInstances_elm ppContextInstanceData PixelPipe::ALIGNED (64) [private]

DynamicContextInstancesPtr ppContextInstanceDataPtr PixelPipe::ALIGNED (64) [private]

DynamicContext_t corContext PixelPipe::ALIGNED (64) [private]

DynamicContextInstances_elm corContextInstanceData PixelPipe::ALIGNED (64) [private]

DynamicContextInstancesPtr corContextInstanceDataPtr PixelPipe::ALIGNED (64) [private]

DynamicContext_t gaussContext PixelPipe::ALIGNED (64) [private]

DynamicContextInstances_elm gaussContextInstanceData PixelPipe::ALIGNED (64) [private]

DynamicContextInstancesPtr gaussContextInstanceDataPtr PixelPipe::ALIGNED (64) [private]

void PixelPipe::initPixelPipe (**t_pPipeResourceCfg** * vpResource, pyramidAlgoType_t pyrAlg, cornerConfig_t corCfg) [private]

u32 PixelPipe::pixelPipe (frameBuffer * in_img, tvFeatureCell ** feature_cells, tvPyramidBuffer * frameBuffer, fp32 * thresholds, u32 num_pyr_levels, u32 num_pyrs, u32 cellGridDimension, u32 maxNumFeatures, u32 targetNumFeatures, **ppThresholds_t** * thresholdCfg) [private]

void PixelPipe::ppInit (**t_pPipeResourceCfg** * vpResource, pyramidAlgoType_t pyrAlg, cornerConfig_t corCfg)

u32 PixelPipe::ppRun (frameBuffer * in_img, tvFeatureCell ** feature_cells, tvPyramidBuffer * frameBuffer, fp32 * thresholds, u32 num_pyr_levels, u32 num_pyrs, u32 cellGridDimension, u32 maxNumFeatures, u32 targetNumFeatures, **ppThresholds_t** * thresholdCfg)

6.40.3 Field Documentation

u32 PixelPipe::cornerNumShaves [private]

swcShaveUnit_t* PixelPipe::cornerShavesList [private]

u32 PixelPipe::gaussNumShaves [private]

swcShaveUnit_t* PixelPipe::gaussShavesList [private]

fifoCommMasterHandler_t* PixelPipe::masterHandler [private]

pixelPipeParams_t* PixelPipe::pixelPipeParams [private]

u8 PixelPipe::ppCacheData [private]

u8 PixelPipe::ppCacheInstr [private]

swcShaveUnit_t PixelPipe::ppMasterShaveNum [private]

fifoCommTask_t* PixelPipe::shaveTaskTypes [private]

fifoCommSlaveHandler_t* PixelPipe::slaveHandler [private]

u32 PixelPipe::slaveNumShaves [private]

swcShaveUnit_t PixelPipe::slaveShaves[**NUM_SHAVES_SLAVE**] [private]

fifoCommTask_t* PixelPipe::taskTypes [private]

6.41 PlgFifoElemS Struct Reference

`#include <PlgFifoApi.h>`

Data Fields

- uint32_t inputId
- void * value
- struct PlgFifoElemS * next

6.41.1 Field Documentation

uint32_t PlgFifoElemS::inputId

struct **PlgFifoElemS**∗ PlgFifoElemS::next

void∗ PlgFifoElemS::value

6.42 PlgFifoS Struct Reference

`#include <PlgFifoApi.h>`

Data Fields

- PlgType plg
- void(∗ triger)(void ∗pluginObj)
- FrameProducedCB cbList [PLGFIFO_MAX_NR_OF_INPUTS]
- FramePool ∗ outputPools
- uint32_t nOutputPools
- PlgFifoElem fifoList [PLGFIFO_SZ]
- PlgFifoElem ∗ fifoTop
- PlgFifoElem ∗ fifoBase
- volatile unsigned int fifoLevel

6.42.1 Field Documentation

FrameProducedCB PlgFifoS::cbList[**PLGFIFO_MAX_NR_OF_INPUTS**]

PlgFifoElem∗ PlgFifoS::fifoBase

volatile unsigned int PlgFifoS::fifoLevel

PlgFifoElem PlgFifoS::fifoList[**PLGFIFO_SZ**]

PlgFifoElem∗ PlgFifoS::fifoTop

uint32_t PlgFifoS::nOutputPools

FramePool∗ PlgFifoS::outputPools

PlgType PlgFifoS::plg

void(∗ PlgFifoS::triger)(void ∗pluginObj)

6.43 PlgIspFullStruct Struct Reference

`#include <PlgIspFullApi.h>`

Data Fields

- PlgType plg
- PlgIspBase base

- OpipeMF op
- uint8_t * cSigma
- uint8_t * cDbyrY
- uint8_t * cSharpY
- uint8_t * cLut
- uint8_t * cUpfirDn
- uint8_t * cDbyrIn
- icIspConfig * ispCfg
- icSize frmSz
- void(* procesStart)(void *plg, uint32_t seqNr, void *userData)
- void(* procesEnd)(void *plg, uint32_t seqNr, void *userData)
- void(* procesIspError)(void *plg, icSeverity severity, icError errorNo, void *userData)
- FramePool * outputPools
- volatile int32_t crtStatus
- FrameProducedCB cbList [1]
- PlgIspFullStatus status
- YuvScale scale

6.43.1 Field Documentation

PlgIspBase PlgIspFullStruct::base

FrameProducedCB PlgIspFullStruct::cbList[1]

uint8_t* PlgIspFullStruct::cDbyrIn

uint8_t* PlgIspFullStruct::cDbyrY

uint8_t* PlgIspFullStruct::cLut

volatile int32_t PlgIspFullStruct::crtStatus

uint8_t* PlgIspFullStruct::cSharpY

uint8_t* PlgIspFullStruct::cSigma

uint8_t* PlgIspFullStruct::cUpfirDn

icSize PlgIspFullStruct::frmSz

icIspConfig* PlgIspFullStruct::ispCfg

OpipeMF PlgIspFullStruct::op

FramePool* PlgIspFullStruct::outputPools

PlgType PlgIspFullStruct::plg

void(* PlgIspFullStruct::procesEnd)(void ***plg**, uint32_t seqNr, void *userData)

void(* PlgIspFullStruct::procesIspError)(void *plg, icSeverity severity, icError errorNo, void *userData)

void(* PlgIspFullStruct::procesStart)(void *plg, uint32_t seqNr, void *userData)

YuvScale PlgIspFullStruct::scale

PlgIspFullStatus PlgIspFullStruct::status

6.44 PlgSource Struct Reference

`#include <PlgSourceApi.h>`

Data Fields

- PlgType plg
- OpipeRx pRx
- MipiRxCfg rxCfg
- void(* eofEvent)(void *plg, FrameT *frame)
- void(* sofEvent)(void *plg, FrameT *frame)
- void(* hitEvent)(void *plg, FrameT *frame)
- FramePool * outputPools
- FrameT * frame
- uint32_t srcId
- uint32_t frameCnt
- icMipiConfig mipiRxData
- PlgSourceStatus status
- int downshift
- FrameT * inProcessFrame
- FrameT * producedFrame
- uint32_t receiverEofIdx
- uint32_t mipiCtrlEofIdx
- uint32_t conectedToController

6.44.1 Field Documentation

uint32_t PlgSource::conectedToController

int PlgSource::downshift

void(* PlgSource::eofEvent)(void *plg, FrameT *frame)

FrameT* PlgSource::frame

uint32_t PlgSource::frameCnt

void(* PlgSource::hitEvent)(void *plg, FrameT *frame)

FrameT* PlgSource::inProcessFrame

uint32_t PlgSource::mipiCtrlEofIdx

icMipiConfig PlgSource::mipiRxData

FramePool* PlgSource::outputPools

PlgType PlgSource::plg

FrameT* PlgSource::producedFrame

OpipeRx PlgSource::pRx

uint32_t PlgSource::receiverEofIdx

MipiRxCfg PlgSource::rxCfg

void(* PlgSource::sofEvent)(void *__plg__, FrameT *__frame__)

uint32_t PlgSource::srcId

PlgSourceStatus PlgSource::status

6.45 PlgSrcIsp Struct Reference

`#include <PlgSrcIspApi.h>`

Data Fields

- PlgType plg
- OpipeRxIsp pRxIsp
- OpipeRx pRxSkip
- FramePool * outputPools
- FrameT * frame
- PlgSrcIspStatus status
- void(* eofEvent)(void *plg, FrameT *frame)
- void(* sofEvent)(void *plg, FrameT *frame)
- void(* hitEvent)(void *plg, FrameT *frame)
- uint32_t srcId
- MipiRxCfg rxCfg
- icMipiConfig mipiRxData
- int downshift
- icIspConfig * curIspCfg
- icIspConfig * nxtIspCfg
- icSize frmSz
- YuvScale scale
- void(* procesIspError)(void *plg, PlgSrcIspErrors errorNo)

6.45.1 Field Documentation

icIspConfig* PlgSrcIsp::curIspCfg

int PlgSrcIsp::downshift

void(* PlgSrcIsp::eofEvent)(void *__plg__, FrameT *__frame__)

FrameT* PlgSrcIsp::frame

icSize PlgSrcIsp::frmSz

void(* PlgSrcIsp::hitEvent)(void *__plg__, FrameT *__frame__)

icMipiConfig PlgSrcIsp::mipiRxData

icIspConfig* PlgSrcIsp::nxtIspCfg

FramePool* PlgSrcIsp::outputPools

PlgType PlgSrcIsp::plg

void(* PlgSrcIsp::procesIspError)(void *__plg__, **PlgSrcIspErrors** errorNo)

OpipeRxIsp PlgSrcIsp::pRxIsp

OpipeRx PlgSrcIsp::pRxSkip

MipiRxCfg PlgSrcIsp::rxCfg

YuvScale PlgSrcIsp::scale

void(* PlgSrcIsp::sofEvent)(void *__plg__, FrameT *__frame__)

uint32_t PlgSrcIsp::srcId

PlgSrcIspStatus PlgSrcIsp::status

6.46 PluginServerCtrl Struct Reference

`#include <IpipeServerApi.h>`

Data Fields

- void(* cbConfigPlugin)(uint32_t ispInstance, void *cfg)

6.46.1 Field Documentation

void(* PluginServerCtrl::cbConfigPlugin)(uint32_t ispInstance, void *cfg)

6.47 ppThresholds_t Struct Reference

`#include <PixelPipeApi.h>`

Data Fields

- fp32 thresholdDecreaseVelocity
- fp32 thresholdIncreaseVelocity
- fp32 thresholdMin
- fp32 thresholdMax

6.47.1 Field Documentation

fp32 ppThresholds_t::thresholdDecreaseVelocity

fp32 ppThresholds_t::thresholdIncreaseVelocity

fp32 ppThresholds_t::thresholdMax

fp32 ppThresholds_t::thresholdMin

6.48 RawCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- uint32_t gainGr
- uint32_t gainR
- uint32_t gainB
- uint32_t gainGb
- uint32_t clampGr
- uint32_t clampR
- uint32_t clampB
- uint32_t clampGb
- uint32_t grgbImbalPlatDark
- uint32_t grgbImbalDecayDark
- uint32_t grgbImbalPlatBright
- uint32_t grgbImbalDecayBright
- uint32_t grgbImbalThr
- uint32_t dpcAlphaHotG
- uint32_t dpcAlphaHotRb
- uint32_t dpcAlphaColdG
- uint32_t dpcAlphaColdRb
- uint32_t dpcNoiseLevel
- uint32_t outputBits

6.48.1 Field Documentation

uint32_t RawCfg::clampB

uint32_t RawCfg::clampGb

uint32_t RawCfg::clampGr

uint32_t RawCfg::clampR

uint32_t RawCfg::dpcAlphaColdG

uint32_t RawCfg::dpcAlphaColdRb

uint32_t RawCfg::dpcAlphaHotG

uint32_t RawCfg::dpcAlphaHotRb

uint32_t RawCfg::dpcNoiseLevel

uint32_t RawCfg::gainB

uint32_t RawCfg::gainGb

uint32_t RawCfg::gainGr

uint32_t RawCfg::gainR

uint32_t RawCfg::grgbImbalDecayBright

uint32_t RawCfg::grgbImbalDecayDark

uint32_t RawCfg::grgbImbalPlatBright

uint32_t RawCfg::grgbImbalPlatDark

uint32_t RawCfg::grgbImbalThr

uint32_t RawCfg::outputBits

6.49 RectRgn Struct Reference

```
#include <OpipeDefs.h>
```

Data Fields

- uint16_t tlcX
- uint16_t tlcY
- uint16_t w
- uint16_t h
- uint16_t lStride

- uint32_t pStride
- uint16_t lineNo

6.49.1 Field Documentation

uint16_t RectRgn::h

uint16_t RectRgn::lineNo

uint16_t RectRgn::lStride

uint32_t RectRgn::pStride

uint16_t RectRgn::tlcX

uint16_t RectRgn::tlcY

uint16_t RectRgn::w

6.50 SBuffer Struct Reference

```
#include <OpipeDefs.h>
```

Data Fields

- uint32_t base
  buffer base addr in CMX
- uint32_t height
  buffer height for circular progress
- uint32_t lineStride
  aux: line stride [bytes] = 8Byte aligned version of (lineW)
- uint32_t lineNo
- uint32_t lineW
  line width in Bytes (ddr.lineW could be smaller on crop)

6.50.1 Field Documentation

uint32_t SBuffer::base

buffer base addr in CMX

uint32_t SBuffer::height

buffer height for circular progress

uint32_t SBuffer::lineNo

uint32_t SBuffer::lineStride

aux: line stride [bytes] = 8Byte aligned version of (lineW)

uint32_t SBuffer::lineW

line width in Bytes (ddr.lineW could be smaller on crop)

6.51 send_out_tx_buffer_header_t Struct Reference

`#include <sendOutApi.h>`

Public Member Functions

- uint8_t metadata[0] __attribute ((aligned(8)))
- uint8_t metadata[0] __attribute ((aligned(8)))

Data Fields

- uint32_t chunk
- char client_data [sizeof(client_tx_frame_header_t)]

6.51.1 Member Function Documentation

uint8_t metadata [0] send_out_tx_buffer_header_t::__attribute ((aligned(8)))

uint8_t metadata [0] send_out_tx_buffer_header_t::__attribute ((aligned(8)))

6.51.2 Field Documentation

uint32_t send_out_tx_buffer_header_t::chunk

char send_out_tx_buffer_header_t::client_data

6.52 SendOutElement_t Struct Reference

`#include <sendOutApi.h>`

Data Fields

- FrameT * buffer
- uint32_t outId
- uint32_t frmType
- SendOutCbSent sendOutCbSent
- InternalCbSent localCallback

6.52.1 Field Documentation

FrameT * SendOutElement_t::buffer

uint32_t SendOutElement_t::frmType

InternalCbSent SendOutElement_t::localCallback

uint32_t SendOutElement_t::outId

SendOutCbSent SendOutElement_t::sendOutCbSent

6.53 SendOutInitCfg_t Struct Reference

`#include <sendOutApi.h>`

Data Fields

- HdmiCfg_t * hdmiCfg
- MipiCfg_t * mipiCfg
- UsbCfg_t * usbCfg

6.53.1 Field Documentation

HdmiCfg_t * SendOutInitCfg_t::hdmiCfg

MipiCfg_t * SendOutInitCfg_t::mipiCfg

UsbCfg_t * SendOutInitCfg_t::usbCfg

6.54 SharpenCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- float sigma
- uint16_t sharpenCoeffs [4]
- uint16_t strengthDarken
- uint16_t strengthLighten
- uint16_t alpha
- uint16_t overshoot
- uint16_t undershoot
- uint16_t rangeStop0
- uint16_t rangeStop1
- uint16_t rangeStop2
- uint16_t rangeStop3
- uint16_t minThr

6.54.1 Field Documentation

uint16_t SharpenCfg::alpha

uint16_t SharpenCfg::minThr

uint16_t SharpenCfg::overshoot

uint16_t SharpenCfg::rangeStop0

uint16_t SharpenCfg::rangeStop1

uint16_t SharpenCfg::rangeStop2

uint16_t SharpenCfg::rangeStop3

uint16_t SharpenCfg::sharpenCoeffs[4]

float SharpenCfg::sigma

uint16_t SharpenCfg::strengthDarken

uint16_t SharpenCfg::strengthLighten

uint16_t SharpenCfg::undershoot

6.55 SigmaDnsCfg Struct Reference

```
#include <OpipeBlocks.h>
```

Data Fields

- uint32_t noiseFloor
- uint32_t thresh1P0
- uint32_t thresh2P0
- uint32_t thresh1P1
- uint32_t thresh2P1
- uint32_t thresh1P2
- uint32_t thresh2P2
- uint32_t thresh1P3
- uint32_t thresh2P3

6.55.1 Field Documentation

uint32_t SigmaDnsCfg::noiseFloor

uint32_t SigmaDnsCfg::thresh1P0

uint32_t SigmaDnsCfg::thresh1P1

uint32_t SigmaDnsCfg::thresh1P2

uint32_t SigmaDnsCfg::thresh1P3

uint32_t SigmaDnsCfg::thresh2P0

uint32_t SigmaDnsCfg::thresh2P1

uint32_t SigmaDnsCfg::thresh2P2

uint32_t SigmaDnsCfg::thresh2P3

6.56 SourceServerCtrlT Struct Reference

`#include <IpipeServerApi.h>`

Data Fields

- void(∗ cbStartSource)(uint32_t sourceInstance, icSourceConfig ∗sourceConfig)
- void(∗ cbStopSource)(uint32_t sourceInstance)
- void(∗ cbCfgDynamic)(uint32_t sourceInstance, icSourceConfigDynamic ∗dynCfg)
- FramePool ∗ pool
- IcQuerySource sourceDescription

6.56.1 Field Documentation

void(∗ SourceServerCtrlT::cbCfgDynamic)(uint32_t sourceInstance, icSourceConfigDynamic ∗dynCfg)

void(∗ SourceServerCtrlT::cbStartSource)(uint32_t sourceInstance, icSourceConfig ∗sourceConfig)

void(∗ SourceServerCtrlT::cbStopSource)(uint32_t sourceInstance)

FramePool∗ SourceServerCtrlT::pool

IcQuerySource SourceServerCtrlT::sourceDescription

6.57 spiSlaveCommunicationConfiguration_t Struct Reference

`#include <OsDrvSpiSlaveCP.h>`

Data Fields

- spiSlaveBlock_t device
- u32 scpol
- u32 scpha
- u32 bpw
- dmaUsed_t useDma

- u32 hostGpioIrq
- u32 irqLevel

6.58 t_ppFifoCfg Struct Reference

`#include <PixelPipeApi.h>`

Data Fields

- u8 fifo1
- u8 fifo2
- u8 fifo3
- u8 fifo4

6.58.1 Field Documentation

u8 t_ppFifoCfg::fifo1

u8 t_ppFifoCfg::fifo2

u8 t_ppFifoCfg::fifo3

u8 t_ppFifoCfg::fifo4

6.59 t_pPipeResourceCfg Struct Reference

`#include <PixelPipeApi.h>`

Data Fields

- t_pPipeShaveConfig * shaveConfig
- u8 * ppBuffs
- u32 ppBuffsSize
- u8 * ppCmxBuffs
- u32 ppCmxBuffsSize
- u8 ppCacheData
- u8 ppCacheInstr
- t_ppFifoCfg fifoCfg

6.59.1 Field Documentation

t_ppFifoCfg t_pPipeResourceCfg::fifoCfg

u8* t_pPipeResourceCfg::ppBuffs

u32 t_pPipeResourceCfg::ppBuffsSize

u8 t_pPipeResourceCfg::ppCacheData

u8 t_pPipeResourceCfg::ppCacheInstr

u8* t_pPipeResourceCfg::ppCmxBuffs

u32 t_pPipeResourceCfg::ppCmxBuffsSize

t_pPipeShaveConfig* t_pPipeResourceCfg::shaveConfig

6.60 t_pPipeShaveConfig Struct Reference

`#include <PixelPipeApi.h>`

Data Fields

- u32 gaussNoShaves
- swcShaveUnit_t * gaussShaveList
- u32 cornerNoShaves
- swcShaveUnit_t * cornerShaveList
- swcShaveUnit_t * ppShaveNum

6.60.1 Field Documentation

u32 t_pPipeShaveConfig::cornerNoShaves

swcShaveUnit_t* t_pPipeShaveConfig::cornerShaveList

u32 t_pPipeShaveConfig::gaussNoShaves

swcShaveUnit_t* t_pPipeShaveConfig::gaussShaveList

swcShaveUnit_t* t_pPipeShaveConfig::ppShaveNum

6.61 TrigerCaptQue Struct Reference

`#include <IpipeServerApi.h>`

Data Fields

- TriggerCaptElement queue [MAX_NR_OF_CAPTURE_PENDING]
- int32_t queueIn
- int32_t queueOut

6.61.1 Field Documentation

TriggerCaptElement TrigerCaptQue::queue[**MAX_NR_OF_CAPTURE_PENDING**]

int32_t TrigerCaptQue::queueIn

int32_t TrigerCaptQue::queueOut

6.62 TriggerCaptElement Struct Reference

`#include <IpipeServerApi.h>`

Data Fields

- FrameT * buffer
- void * config
- icSourceInstance source

6.62.1 Field Documentation

FrameT* TriggerCaptElement::buffer

void* TriggerCaptElement::config

icSourceInstance TriggerCaptElement::source

6.63 UpfirdnCfg Struct Reference

`#include <OpipeBlocks.h>`

Data Fields

- uint32_t kerSz
- uint32_t hN
- uint32_t hD
- uint32_t vN
- uint32_t vD
- uint8_t * hCoefs
- uint8_t * vCoefs
- uint16_t iW
- uint16_t iH
- uint16_t oW
- uint16_t oH

6.63.1 Field Documentation

uint8_t* UpfirdnCfg::hCoefs

uint32_t UpfirdnCfg::hD

uint32_t UpfirdnCfg::hN

uint16_t UpfirdnCfg::iH

uint16_t UpfirdnCfg::iW

uint32_t UpfirdnCfg::kerSz

uint16_t UpfirdnCfg::oH

uint16_t UpfirdnCfg::oW

uint8_t* UpfirdnCfg::vCoefs

uint32_t UpfirdnCfg::vD

uint32_t UpfirdnCfg::vN

6.64 VirtualChannel Struct Reference

`#include <MessageProtocol.h>`

Data Fields

- u8 id
- u8 name [32]
- u32 priority_level
- PhysicalChannel * phyChannel
- MessageRingBuffer rxFifo
- MessageRingBuffer txFifo

Chapter 7

File Documentation

7.1 aeApi.h File Reference

```
#include <FramePump.h>
#include "aeCommStructs.h"
```

Functions

- int simpleAEinit ()
- int simpleAEpostInit ()
- int simpleAEstart (const int fd, const int id, const char *name)
- int simpleAEprocessFrame (const int id, const FramePumpBuffer *frameIn)
- aeMessage_t simpleAEget ()

7.1.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved. For License Warranty see: common/license.txt

7.1.2 Function Documentation

aeMessage_t simpleAEget ( )

int simpleAEinit ( )

int simpleAEpostInit ( )

int simpleAEprocessFrame (const int id, const FramePumpBuffer * frameIn)

int simpleAEstart (const int fd, const int id, const char * name)

7.2 bicubicWarpApi.h File Reference

```
#include "bicubicWarpApiDefines.h"
```

Functions

- int bicubicWarpInit (bicubicWarpContext *ctx)

  Initialize bicubic block.
- int bicubicWarpProcessFrame (bicubicWarpContext *ctx)

  Run bicubic block.
- void bicubicWarpGenerateMeshRT (bicubicWarpContext *ctx)

  Generate rectified coordinates based on rotation and translation (RT) relative to the center.
- void bicubicWarpGenerateMeshHomographyRTP (bicubicWarpContext *ctx)

  Generate rectified coordinates based on the homography (OpenCV style)
- void bicubicWarpGenerateMeshFromLUTMaps (bicubicWarpContext *ctx)

  Generate rectified coordinates from LUT maps.

7.2.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved. For License Warranty see: common/license.txt

7.3 Board182Api.h File Reference

```
#include "Board182ApiDefines.h"
```

Functions

- s32 BoardInitialise (u32 clockConfiguration)

  This function performs the initialization of basic functions of MV0182 board: I2C buses, external clock generator and sets up all GPIOS.

Variables

- tyAppDeviceHandles gAppDevHndls

7.3.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see-: common/license.txt

7.4 CamGenericApi.h File Reference

```
#include "CamGenericApiDefines.h"
```

Functions

- camErrorType CamInit (GenericCameraHandle *hndl, GenericCamSpec *camSpec, CamUserSpec *userSpec, callbacksListStruct *cbStruct, I2CM_Device *pI2cHandle)

  This will initialize the camera component (the sensor and all the myriad components connected in the frames data path) for a specific sensor.
- camErrorType CamStart (GenericCameraHandle *hndl)

  This will start the sensor and the interrupts system.
- camErrorType CamStandby (GenericCameraHandle *hndl, camStatus_type standbyType)

  This will put the sensor and related myriad logic in a standby mode.
- camErrorType CamWakeup (GenericCameraHandle *hndl)

  This will wake up the sensor and related myriad logic from the standby mode.
- camErrorType CamStop (GenericCameraHandle *hndl)

  Resets the camera component (the sensor and the related myriad logic)
- camErrorType CamSetupCallbacks (GenericCameraHandle *hndl, sensorCallbacksListType *cbList)

  Set or change the existing callbacks used for the sensor configuration.
- camErrorType CamSetupInterrupts (GenericCameraHandle *hndl, camIsrType managedInterrupt, u32 notifiedInterrupts, u32 clearedInterrupts, interruptsCallbacksListType *cbList, u32 interruptsLevel, u32 routeLineInterrupt, u32 routeFrameInterrupt)

  Set or change the existing/the default interrupt events for a camera module.
- unsigned int CamGetFrameCounter (GenericCameraHandle *hndl)

  Get the counter of frames/lines (depending on the managed ISR types) received inside CIF block of myriad component (always 0 for SIPP receivers)

7.4.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2014, all rights reserved For License Warranty see: common/license.txt

7.5 dbgTracerApi.h File Reference

```
#include "logMsg.h"
```

Macros

- #define DEBUG_LOG_LEVEL_LOW LOG_LEVEL_INFO
- #define DEBUG_LOG_LEVEL_MEDIUM LOG_LEVEL_WARNING
- #define DEBUG_LOG_LEVEL_HIGH LOG_LEVEL_ERROR

7.5.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see-: common/license.txt

7.6 disparityMapApi.h File Reference

```
#include <disparityMapApiDefines.h>
```

Functions

- void sgbm (u8 *combinedCostCube, stereoUserConfig_t *userCfg, stereoTileConfig_t *tileCfg, unsigned short *dispMedianSubpixel, u8 *dispMedianInteger, u32 flag)

7.6.1 Function Documentation

void sgbm (u8 * combinedCostCube, stereoUserConfig_t * userCfg, stereoTileConfig_t * tileCfg, unsigned short * dispMedianSubpixel, u8 * dispMedianInteger, u32 flag)

7.7 eventLoop.h File Reference

```
#include "eventQueue.h"
```

Typedefs

- typedef s32(* eventLoopCallback_t)(eventQueueItem_t *event)

  Type of event loop callback for each event type;.
- typedef enum eventState_t eventState_t

  Enum of different states of an event.

Enumerations

- enum eventState_t { BUSY, DONE, REQUIRED, DROPPED }

  Enum of different states of an event.

Functions

- void eventLoopInit ()

  Initialize the event loop queuese.
- s32 eventLoopSetCallback (eventType_t event, eventLoopCallback_t cb)

  Add an event loop callback for an event type.
- void eventLoopPushCmd (eventQueueItem_t *event)

Add an event to the command queue.

- void eventLoopPush (eventQueueItem_t *event)

  Add an event to the event loop.

- void eventLoopRun ()

  Run the event loop.

- void eventLoopReset ()

  Clear the event loop.

- void eventLoopStartTimer ()

  Start event loop timer.

- void eventLoopSetEventState (eventState_t evState, eventQueueItem_t *event, eventLoopCallback_t evCallbackPtr)

  Set the event state.

- void eventLoopClearBusyFlag (eventQueueItem_t *event, eventLoopCallback_t evCallbackPtr)

  Clear the busy callback flag.

- void eventLoopChangeEventType (eventQueueItem_t *event, eventType_t newEventType)

  Change the event type to newEventType.

- void eventLoopDropEvent (eventQueueItem_t *event, eventLoopCallback_t evCallbackPtr)

  Drop an event from the loop.

- void eventLoopSetReleaseCallback (eventType_t event, eventLoopCallback_t onReleaseCb)

  Set event done callback.

- void eventLoopReleaseEvent (eventQueueItem_t *event, eventLoopCallback_t evCallbackPtr)

  Mark event done or free it, depending on the event state.

7.7.1 Detailed Description

Copyright

7.8 eventQueue.h File Reference

```
#include "EventType.h"
```

Data Structures

- struct eventQueueItem_t

  An event queue item type.

- struct eventQueue_t

  An event queue type.

Typedefs

- typedef struct eventQueueItem_t eventQueueItem_t

  An event queue item type.
- typedef struct eventQueue_t eventQueue_t

  An event queue type.

Functions

- u64 eventLoopTimestamp ()

  Get system timestamp in clock-ticks.
- void eventQueueInit (eventQueue_t *self)

  Initialize an event queue.
- void eventQueuePush (eventQueue_t *self, eventQueueItem_t *item)

  Push a new item on the event queue param[in] self - event queue to which will be added a new item param[in] item - new event queue item.
- eventQueueItem_t * eventQueuePop (eventQueue_t *self)

  Get item of the head of list and remove it from list.
- void eventQueueReturnToParent (eventQueueItem_t *item)

  Return item to parent queue.

7.8.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved. For License Warranty see: common/license.txt

7.9 featureMaintenanceApi.h File Reference

```
#include "mv_types.h"
#include "swcFrameTypes.h"
#include "vTrack.h"
#include "vPipePublicTypes.h"
#include "global_constants.h"
#include "theDynContext.h"
#include "featureMaintenanceTypes.h"
```

Data Structures

- struct fmResourceCfg
- class FeatMaintenance

Typedefs

- typedef u32 swcShaveUnit_t
- typedef struct fmResourceCfg fmResourceCfg_t

Functions

- class FeatMaintenance ALIGNED (64)
- FeatMaintenance ()
- ~FeatMaintenance ()
- void fmRun (const u32 frameNum, const frameBuffer *const fullFrame, const tvXYLoc featuresTmp[], const fp32 featuresErrorTmp[], FMFeatureThresholds_t *thresholds)
- void fmInit (FMSetupCfg *cfg, fmResourceCfg_t *resCfg)

Variables

- uint32_t fmShaveNum
- uint8_t featPipeCacheData
- uint8_t featPipeCacheInstr
- FMSetupCfg g_fmCfg

7.9.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved. For License Warranty see: common/license.txt

7.9.2 Typedef Documentation

typedef struct **fmResourceCfg fmResourceCfg_t**

typedef u32 **swcShaveUnit_t**

7.9.3 Function Documentation

class **FeatMaintenance** ALIGNED (64)

ALIGNED::FeatMaintenance ( )

void ALIGNED::fmInit (FMSetupCfg * cfg, **fmResourceCfg_t** * resCfg)

void ALIGNED::fmRun (const u32 frameNum, const frameBuffer *const fullFrame, const tvXYLoc featuresTmp[], const fp32 featuresErrorTmp[], FMFeatureThresholds_t * thresholds)

ALIGNED::~FeatMaintenance ( )

7.9.4 Variable Documentation

uint8_t featPipeCacheData

uint8_t featPipeCacheInstr

uint32_t fmShaveNum

FMSetupCfg g_fmCfg

7.10 fifoCommApi.h File Reference

Application configuration Leon header.

```
#include "fifoCommApiDefines.h"
```

Functions

- void fifoCommMasterAddTask (fifoCommTask_t *taskType, void *taskParameters)
  Add new task to be executed by the slaves.
- void * fifoCommMasterWaitTask (fifoCommTask_t *taskType)
  Wait for task to be executed.
- u32 fifoCommSlaveReadTask (fifoCommTask_t *taskType, u32 *result)
  Slave reads task from FIFO.
- void fifoCommSlaveNotifyTaskCompletion (fifoCommTask_t *taskType, void *taskParameters)
  Write a 32-bit value to a Shave FIFO.
- void fifoCommMasterRun (fifoCommMasterHandler_t *handler, void *params)
  Runs the master shave.
- void fifoCommSlaveRun (fifoCommSlaveHandler_t *handler)
  Runs the slave shave.

7.10.1 Detailed Description

Application configuration Leon header.

Copyright

All code copyright Movidius Ltd 2012, all rights reserved. For License Warranty see: common/license.txt

7.10.2 Function Documentation

void fifoCommMasterAddTask (**fifoCommTask_t** * taskType, void * taskParameters)

Add new task to be executed by the slaves.

Parameters

in	*taskType*	- task to be executed
in	*taskParameters*	- parameters of the task

Returns

void

void fifoCommMasterRun (**fifoCommMasterHandler_t** * handler, void * params)

Runs the master shave.

Parameters

in	*handler*	- master handler of the shave
in	*params*	- master parameters

Returns

void

void* fifoCommMasterWaitTask (**fifoCommTask_t** * taskType)

Wait for task to be executed.

Parameters

in	*taskType*	- task we are waiting for

Returns

void

void fifoCommSlaveNotifyTaskCompletion (**fifoCommTask_t** * taskType, void * taskParameters)

Write a 32-bit value to a Shave FIFO.

Parameters

in	*taskType*	- response FIFO which will be used
in	*taskParameters*	

Returns

void

u32 fifoCommSlaveReadTask (**fifoCommTask_t** * taskType, u32 * result)

Slave reads task from FIFO.

Parameters

in	*taskType*	- task to be read
in	*result*	- stores the value read from the FIFO

Returns

if the read was successful or not

void fifoCommSlaveRun (**fifoCommSlaveHandler_t** * handler)

Runs the slave shave.

Parameters

in	*handler*	- handler of the slave

Returns

void

7.11 fifoCommInitApi.h File Reference

shave Fifo communication init functions

```
#include "fifoCommApiDefines.h"
#include "OsDrvSvu.h"
```

Functions

- int fifoCommStartMaster (fifoCommMasterHandler_t *masterHandler, osDrvSvuHandler_t *handler, u32 *fifoCommMasterRun, void *masterParams, u32 shaveNumber)

  Open and start master.
- void fifoCommStartSlave (fifoCommSlaveHandler_t *slaveHandler, osDrvSvuHandler_t *handler, u32 *slaveRun, int shaveNumber)

  Open and start slave.
- void fifoCommWaitMaster (osDrvSvuHandler_t *handler)

  The function will wait for the master to finish its task. After the task is finished, the shave will be closed.
- void fifoCommWaitSlave (osDrvSvuHandler_t *handler)

  The function will wait for the slave to finish its task. After the task is finished, the shave will be closed.
- void fifoCommMasterInit (fifoCommMasterHandler_t *handler, fifoCommMasterCallback_t taskHandler)

  Master entry point. The function pointer which will be executed on the master shave.
- void fifoCommSlaveInit (fifoCommSlaveHandler_t *handler)

  Initialize slave handler.
- void fifoCommMasterRegisterTaskType (fifoCommMasterHandler_t *masterHandler, fifoCommTask_t *taskType, u32 taskFifoNr, u32 responseFifoNr)

  Create a new task type and register it to the master.
- void fifoCommSlaveRegisterTaskType (fifoCommSlaveHandler_t *slaveHandler, fifoCommTask_t *taskType, fifoCommTask_t *masterTask, fifoCommTaskCallback_t initHandler, fifoCommTaskCallback_t taskHandler)

  Set on slave handler the task which can be done by the slave.

7.11.1 Detailed Description

shave Fifo communication init functions

Copyright

All code copyright Movidius Ltd 2012, all rights reserved. For License Warranty see: common/license.txt

7.11.2 Function Documentation

void fifoCommMasterInit (**fifoCommMasterHandler_t** * handler, fifoCommMasterCallback_t taskHandler)

Master entry point. The function pointer which will be executed on the master shave.

Parameters

in	*handler*	- master handler to store the required private information.
in	*taskHandler*	- master entry point

Returns

void

void fifoCommMasterRegisterTaskType (**fifoCommMasterHandler_t** * masterHandler, **fifoCommTask_t** * taskType, u32 taskFifoNr, u32 responseFifoNr)

Create a new task type and register it to the master.

Parameters

in	*masterHandler*	- the already initialized master handler
in	*taskType*	- pointer to a taskType. This will be initialized by the function
in	*taskFifoNr*	- the FIFO from which the task will be executes
in	*responseFifoNr*	- the response FIFO number

Returns

void

void fifoCommSlaveInit (**fifoCommSlaveHandler_t** * handler)

Initialize slave handler.

Parameters

in	*handler*	- slave handler to store the required private information

Returns

void

void fifoCommSlaveRegisterTaskType (**fifoCommSlaveHandler_t** * slaveHandler, **fifoCommTask_t** * taskType, **fifoCommTask_t** * masterTask, fifoCommTaskCallback_t initHandler, fifoCommTaskCallback_t taskHandler)

Set on slave handler the task which can be done by the slave.

Parameters

in	*slaveHandler*	- handler of the used slave
in	*taskType*	- pointer to a taskType. This will be initialized by the function
in	*masterTask*	- pointer to the already initialized masterTaskType. The slave task will be linked to this type of task on master
in	*initHandler*	- slave init entry point. The function pointer which will be executed on the slave shave once, at the beginning
in	*taskHandler*	- slave entry point. The function pointer which will be executed on the slave shave each time a new task of this type was added

Returns

void

int fifoCommStartMaster (**fifoCommMasterHandler_t** * masterHandler, osDrvSvuHandler_t * handler, u32 * fifoCommMasterRun, void * masterParams, u32 shaveNumber)

Open and start master.

Parameters

in	*masterHandler*	- handler of the used master
in	*handler*	- handler of the used shave
in	*masterParams*	- parameter of the called function
in	*shaveUsed*	- shaved used for the master

Returns

void

void fifoCommStartSlave (**fifoCommSlaveHandler_t** * slaveHandler, osDrvSvuHandler_t * handler, u32 * slaveRun, int shaveNumber)

Open and start slave.

Parameters

in	*slaveHandler*	- handler of the used slave
in	*handler*	- handler of the used shave
in	*shaveNumber*	- slave entry point id

Returns

void

void fifoCommWaitMaster (osDrvSvuHandler_t * handler)

The function will wait for the master to finish its task. After the task is finished, the shave will be closed.

Parameters

in	*handler*	- shave handler

Returns

void

void fifoCommWaitSlave (osDrvSvuHandler_t * handler)

The function will wait for the slave to finish its task. After the task is finished, the shave will be closed.

Parameters

in	*handler*	- shave handler

Returns

void

7.12 FrameMgrApi.h File Reference

Functions

- int FrameMgrCreatePool (FramePool *pool, FrameT *frames, FrameProducedCB *callbacks, int32_t nCallbacks)
- FrameT * FrameMgrAcquireFrame (FramePool *pool)
- void FrameMgrProduceFrame (FrameT *frame)
- void FrameMgrReleaseFrame (FrameT *frame)
- FrameT * FrameMgrLockFrame (FramePool *pool, uint64_t frameSel, int32_t tsRel)
- void FrameMgrUnlockFrame (FrameT *frame)
- void FrameMgrIncreaseNrOfConsumer (FrameT *frame, uint32_t addNr)
- void FrameMgrAndAddTimeStamp (FrameT *oFrame, icTimestamp timeStamp)
- void FrameMgrAddTimeStampHist (FrameT *oFrame, FrameT *iFrame)
- void FrameMgrAddFrameToPool (FramePool *pool, FrameT *frame)

7.12.1 Function Documentation

FrameT* FrameMgrAcquireFrame (FramePool * pool)

void FrameMgrAddFrameToPool (FramePool * pool, FrameT * frame)

void FrameMgrAddTimeStampHist (FrameT * oFrame, FrameT * iFrame)

void FrameMgrAndAddTimeStamp (FrameT * oFrame, icTimestamp timeStamp)

int FrameMgrCreatePool (FramePool * pool, FrameT * frames, FrameProducedCB * callbacks, int32_t nCallbacks)

void FrameMgrIncreaseNrOfConsumer (FrameT * frame, uint32_t addNr)

FrameT* FrameMgrLockFrame (FramePool * pool, uint64_t frameSel, int32_t tsRel)

void FrameMgrProduceFrame (FrameT * frame)

void FrameMgrReleaseFrame (FrameT * frame)

void FrameMgrUnlockFrame (FrameT * frame)

7.13 I2CSlaveApi.h File Reference

```
#include "I2CSlaveApiDefines.h"
```

Functions

- s32 I2CSlaveInit (i2cSlaveHandle_t *hndl, i2cSlaveAddrCfg_t *config)
- void I2CSlaveSetupCallbacks (i2cSlaveHandle_t *hndl, i2cReadAction *cbReadAction, i2cWriteAction *cbWriteAction)

7.13.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2013, all rights reserved For License Warranty see: : common/license.txt

7.14 imageWarp.h File Reference

Image warp component API.

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

Functions

- void imageWarp (meshStruct *mesh, frameBuffer *inputFb, frameBuffer *outputFb, tileList *tileNodes, unsigned short paddingvalue, uint32_t nr_shaves_vsplit, uint32_t proc_shave_idx)

7.14.1 Detailed Description

Image warp component API.

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved. For License Warranty see: common/license.txt

7.14.2 Function Documentation

void imageWarp (meshStruct * mesh, frameBuffer * inputFb, frameBuffer * outputFb, tileList * tileNodes, unsigned short paddingvalue, uint32_t nr_shaves_vsplit, uint32_t proc_shave_idx)

Parameters

in	*mx*	- input mesh for x - coordinates related to input image coordinates
in	*my*	- input mesh for y- coordinates related to input image coordinates
in	*img*	- input image address
in	*shaves_used*	- number of shaves used to compute the entire algo
in	*out_img*	- Input tile of data, (u16 format) it has variable width/height size
in	*out_width*	- Input tile of data, (u16 format) it has variable width/height size
out	*tiles*	- structure of information for each input tile
out	*tile_no*	- number of tiles resulted from the computation

7.15 imageWarp.h File Reference

Image warp component API.

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

Functions

- void Entry (meshStruct *mesh, frameBuffer *inputFb, frameBuffer *outputFb, tileList *tileList, unsigned short paddingvalue)

7.15.1 Detailed Description

Image warp component API.

Copyright

7.15.2 Function Documentation

void Entry (meshStruct * mesh, frameBuffer * inputFb, frameBuffer * outputFb, tileList * tileList, unsigned short paddingvalue)

Parameters

in	*mx*	- input mesh for x - coordinates related to input image coordinates
in	*my*	- input mesh for y- coordinates related to input image coordinates
in	*img*	- input image address
in	*shaves_used*	- number of shaves used to compute the entire algo
in	*out_img*	- Input tile of data, (u16 format) it has variable width/height size
in	*out_width*	- Input tile of data, (u16 format) it has variable width/height size
out	*tiles*	- structure of information for each input tile
out	*tile_no*	- number of tiles resulted from the computation

7.16 imageWarp.h File Reference

```
#include <swcShaveLoader.h>
#include <swcFrameTypes.h>
#include "imageWarpDefines.h"
```

Functions

- u32 IMGWARP_start (swcShaveUnit_t svu, meshStruct *mesh, tileList *tileNodes, frameBuffer *inputFb, frameBuffer *outputFb, unsigned short padding)

  Sets up and launches one dynamic image warp on a specifically requested SHAVE.
- void IMGWARP_cleanup (void)

  Cleans up and prepare already used shave for running other dynamic apps.

7.16.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2013, all rights reserved For License Warranty see-: common/license.txt

7.17 IpipeServerApi.h File Reference

Data Structures

- struct TriggerCaptElement
- struct TrigerCaptQue
- struct SourceServerCtrlT
- struct PluginServerCtrl
- struct ipipeServerInfo

Macros

- #define MAX_NR_OF_CAPTURE_PENDING 8

Functions

- void setupIpipeServer ()
- void ipServerWasReset (void)
- void ipServerWasTornDown (void)
- void ipipeServerHandleInputEvent (icEvent *ev)
- void ipServerSendData (FrameT *frame, uint32_t outId)
- void ipServerSourceReady (icSourceInstance source)
- void ipServerSourceStopped (icSourceInstance source)
- void ipServerReadoutStart (icSourceInstance source, void *userData, uint32_t seqNo, icTimestamp ts)
- void ipServerLineHit (icSourceInstance source, void *userData, uint32_t seqNo, icTimestamp ts)
- void ipServerReadoutEnd (icSourceInstance source, void *userData, uint32_t seqNo, icTimestamp ts)
- void ipServerIspStart (int32_t ispIdx, uint32_t seqNo, void *userData)
- void ipServerIspEnd (int32_t ispIdx, uint32_t seqNo, void *userData)
- void ipServerIspConfigAccepted (int32_t ispIdx, uint32_t seqNo, void *userData)
- void ipServerIspStatsReady (int32_t ispIdx, uint32_t seqNo, void *userData)
- void ipServerFrameLocked (icSourceInstance source, FrameT *frame)
- int ipServerQueueGet (TriggerCaptElement **old)
- void ipServerIspReportError (int32_t srcIdx, icSeverity severity, icError errorNo, void *userData)
- FrameT * ipServerFrameMgrCreateList (uint32_t nrOfFrame)
- void ipServerFrameMgrAddBuffs (FrameT *frameList, uint32_t dataSizePl0, uint32_t dataSizePl1, uint32_t dataSizePl2)
- void ipServerRegSourceQuery (uint32_t id, const char *name, uint32_t attrs, uint32_t nrOfFramesInZsl, uint32_t outIdVid, uint32_t outIdStill, uint32_t outIdRaw, uint32_t outIdBin)
- void ipServerRegIspQuery (uint32_t id, const char *name, uint32_t attrs, uint32_t sourceId)
- void ipServerRegOutputQuery (uint32_t id, const char *name, uint32_t attrs, uint32_t dependentSources)
- void ipServerRegUserPlgQuery (uint32_t id, const char *name, uint32_t attrs)
- void ipServerQueryAddChild (void *parentDescP, void *childDescP)
- void ipServerSendUserMsgToLos (void *eventStruct, uint32_t id)

Variables

- volatile ipipeServerInfo gServerInfo
- icCtrl * pSrvIcCtrl

7.17.1 Macro Definition Documentation

#define MAX_NR_OF_CAPTURE_PENDING 8

7.17.2 Function Documentation

void ipipeServerHandleInputEvent (icEvent * ev)

void ipServerFrameLocked (icSourceInstance source, FrameT * frame)

void ipServerFrameMgrAddBuffs (FrameT * frameList, uint32_t dataSizePl0, uint32_t dataSizePl1, uint32_t dataSizePl2)

FrameT* ipServerFrameMgrCreateList (uint32_t nrOfFrame)

void ipServerIspConfigAccepted (int32_t ispIdx, uint32_t seqNo, void * userData)

void ipServerIspEnd (int32_t ispIdx, uint32_t seqNo, void * userData)

void ipServerIspReportError (int32_t srcIdx, icSeverity severity, icError errorNo, void * userData)

void ipServerIspStart (int32_t ispIdx, uint32_t seqNo, void * userData)

void ipServerIspStatsReady (int32_t ispIdx, uint32_t seqNo, void * userData)

void ipServerLineHit (icSourceInstance source, void * userData, uint32_t seqNo, icTimestamp ts)

void ipServerQueryAddChild (void * parentDescP, void * childDescP)

int ipServerQueueGet (**TriggerCaptElement** ** old)

void ipServerReadoutEnd (icSourceInstance source, void * userData, uint32_t seqNo, icTimestamp ts)

void ipServerReadoutStart (icSourceInstance source, void * userData, uint32_t seqNo, icTimestamp ts)

void ipServerRegIspQuery (uint32_t id, const char * name, uint32_t attrs, uint32_t sourceId)

void ipServerRegOutputQuery (uint32_t id, const char * name, uint32_t attrs, uint32_t dependentSources)

void ipServerRegSourceQuery (uint32_t id, const char * name, uint32_t attrs, uint32_t nrOfFramesInZsl, uint32_t outIdVid, uint32_t outIdStill, uint32_t outIdRaw, uint32_t outIdBin)

void ipServerRegUserPlgQuery (uint32_t id, const char * name, uint32_t attrs)

void ipServerSendData (FrameT * frame, uint32_t outId)

void ipServerSendUserMsgToLos (void * eventStruct, uint32_t id)

void ipServerSourceReady (icSourceInstance source)

void ipServerSourceStopped (icSourceInstance source)

void ipServerWasReset (void)

void ipServerWasTornDown (void)

void setupIpipeServer ( )

7.17.3 Variable Documentation

volatile **ipipeServerInfo** gServerInfo

icCtrl* pSrvIcCtrl

7.18 JpegEncoderApi.h File Reference

```
#include <swcFrameTypes.h>
```

Macros

- #define PARTITION_0 (0)
- #define PARTITION_1 (1)
- #define PARTITION_2 (2)
- #define PARTITION_3 (3)
- #define PARTITION_4 (4)
- #define PARTITION_5 (5)
- #define PARTITION_6 (6)
- #define PARTITION_7 (7)
- #define PARTITION_8 (8)
- #define PARTITION_9 (9)
- #define PARTITION_10 (10)
- #define PARTITION_11 (11)

Enumerations

- enum { JPEG_420_PLANAR, JPEG_422_PLANAR, JPEG_444_PLANAR }

Functions

- u32 JPEG_encode (frameBuffer imgInfo, u8 *outputLocal, u32 shvNo, u32 inBuffSizeShave, int jpegFormat)

  The JPEG encoding algorithm.

7.18.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see: common/license.txt

7.19 JpegEncoderApi.h File Reference

```
#include <swcFrameTypes.h>
```

Macros

- #define PARTITION_0 (0)
- #define PARTITION_1 (1)
- #define PARTITION_2 (2)
- #define PARTITION_3 (3)
- #define PARTITION_4 (4)
- #define PARTITION_5 (5)
- #define PARTITION_6 (6)
- #define PARTITION_7 (7)
- #define PARTITION_8 (8)
- #define PARTITION_9 (9)
- #define PARTITION_10 (10)
- #define PARTITION_11 (11)

Enumerations

- enum { JPEG_420_PLANAR, JPEG_422_PLANAR, JPEG_444_PLANAR }

Functions

- u32 JPEG_encode (frameBuffer imgInfo, u8 *outputLocal, u32 shvNo, u32 inBuffSizeShave, int jpegFormat)

  The JPEG encoding algorithm.

7.19.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see: common/license.txt

7.20 LcdApi.h File Reference

```
#include "LcdApiDefines.h"
```

Functions

- void LCDSetupCallbacks (LCDHandle *hndl, allocateLcdFn *assignFrame, frameReadyLcdFn *frameReady, freqLcdFn *highres, freqLcdFn *lowres)

  Set callbacks for assign new frame and frame ready.
- void LCDStop (LCDHandle *hndl)

  This will Stop the LCD interface.
- void LCDInit (LCDHandle *hndl, const LCDDisplayCfg *lcdsp, const frameSpec *fsp, unsigned int lcdNum)

  This will initialize the LCD interface.

- void LCDInitLayer (LCDHandle *hndl, int layer, frameSpec *fsp, LCDLayerOffset position)

  This will one of the LCD layers if fsp argument of LCDInit is NULL.
- void LCDEnYuv422i (void)

  This will enable YUV420p to Yuv422i format conversion on LCD interface.
- void LCDStart (LCDHandle *hndl)

  This will start the LCD interface.
- void LCDStartOneShot (LCDHandle *hndl)

  This will start the LCD interface in one-shot mode.
- int LCDQueueFrame (LCDHandle *handle, frameBuffer *VL1Buffer, frameBuffer *VL2Buffer, frameBuffer *GL1Buffer, frameBuffer *GL2Buffer)

  Queue a frame in One-Shot mode.
- int LCDCanQueueFrame (LCDHandle *handle)

  Tells you whether you can successfully queue a frame in one-shot mode.
- void LCDInitVL2Enable (LCDHandle *hndl)

  This will enable video layer 1.
- void LCDInitVL2Disable (LCDHandle *hndl)

  This will disable video layer 1.
- void LCDInitGLEnable (LCDHandle *hndl, int layer, const frameSpec *fr)

  This will enable graphical layer.
- void LCDInitGLDisable (LCDHandle *hndl, int layer)

  This will enable graphical layer.
- void LCDSetColorTable (LCDHandle *hndl, int layer, unsigned int *colorTable, int number)

  This will enable graphical layer.
- void LCDSetOutput (LCDHandle *hndl, lcdDatapath_t dataPath)

7.20.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see: common/license.txt

7.21 LeonIPCApi.h File Reference

```
#include "LeonIPCApiDefines.h"
```

Functions

- int LeonIPCTxInit (leonIPCChannel_t *msgChannel, uint32_t *messagePool, uint32_t messagePoolSize, uint32_t messageSize)

  This function initializes a protocol given a message pool and a size for it.
- int LeonIPCRxInit (leonIPCChannel_t *msgChannel, leonIPCCallback_t msgCallback, uint32_t irqNo, uint32_t irqPrio)

  This function initializes the Rx side of the communication.
- int LeonIPCRxReassignSinkThread (leonIPCChannel_t *channel)

This function resets the receiver thread to the current calling thread.

- int LeonIPCRxReleaseSinkThread (leonIPCChannel_t *channel)

  This function resets the receiver thread. This function must be used before using the LeonIPCRxReassignSinkThread function.
- int LeonIPCSendMessage (leonIPCChannel_t *msgChannel, uint32_t *message)

  This function sends a message to the consumer.
- int LeonIPCWaitMessage (leonIPCChannel_t *msgChannel, uint32_t timeout)

  This function waits for a valid message to be present in the message queue.
- int LeonIPCNumberOfPendingMessages (leonIPCChannel_t *msgChannel, uint32_t *noOfmessages)

  This function waits for a valid message to be present in the message queue.
- int LeonIPCReadMessage (leonIPCChannel_t *msgChannel, uint32_t *message)

  This function waits for a valid message to be present in the message queue.

7.21.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see: common/license.txt

7.22 LeonL1CacheApi.h File Reference

```
#include "LeonL1CacheDefines.h"
```

Functions

- void LeonL1CacheInitDiagAccess (void)

  Initialise diagnostic access to the cache.
- void LeonL1CacheDiagDisplay (tyCacheType cache)

  Display L1 cache contents.
- u32 LeonL1CacheDiagCountValidLines (tyCacheType cache)

  Count the valid L1 cache lines.
- void LeonL1CacheDisplayInfo (tyCacheType cache)

  Display L1 cache general information.

- u32 LeonL1CacheReadCacheTagMem (tyCacheType cache, u32 offset)

  Raw access to Tag and Data Memories.
- u32 LeonL1CacheReadCacheDataMem (tyCacheType cache, u32 offset)

  Raw access to Tag and Data Memories.

- u32 LeonL1DCacheReadTagForAddr (u32 address, u32 way)

  Read the L1 cache tag for a specific memory address.
- u32 LeonL1ICacheReadTagForAddr (u32 address, u32 way)

  Read the L1 cache tag for a specific memory address.

- u32 LeonL1DCacheReadDataWordForAddr (u32 address, u32 way)

  Read the L1 cache content for a specific memory address.
- u32 LeonL1ICacheReadDataWordForAddr (u32 address, u32 way)

  Read the L1 cache content for a specific memory address.

- u32 LeonL1CacheIsAddressDCached (u32 address)

  Check whether a specific memory address is cached either in data or instructions cache.
- u32 LeonL1CacheIsAddressICached (u32 address)

  Check whether a specific memory address is cached either in data or instructions cache.

7.22.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see-: common/license.txt

7.23 MDKdox-Components-intro.txt File Reference

7.24 memManagerApi.h File Reference

```
#include "memManagerAreas.h"
```

Enumerations

- enum MemMgrErrorCode {
  NO_ERROR = 0, MEMALLOC_INVALID_SIZE = 10, MEMALLOC_SIZE_EXCEEDED = 11,
  MEM_INVALID_ADDRESS = 12,
  MEM_EXPORT_ALLOCMEM = 13 }

Functions

- void * MemMgrAlloc (size_t size, MemMgrAreas area, unsigned int MemAlignment)
- MemMgrErrorCode MemMgrFree (void *address)
- MemMgrErrorCode MemMgrBufferExport (size_t bufSize, char *csvBuffer, size_t *csvLenght)
- void MemMgrExportDraw (int dispWidth)

7.24.1 Enumeration Type Documentation

enum **MemMgrErrorCode**

Enumerator

NO_ERROR

MEMALLOC_INVALID_SIZE

MEMALLOC_SIZE_EXCEEDED

MEM_INVALID_ADDRESS

MEM_EXPORT_ALLOCMEM

7.24.2 Function Documentation

void* MemMgrAlloc (size_t size, MemMgrAreas area, unsigned int MemAlignment)

Allocate memory equal to the specified size inside the given memory area

Parameters

in	*size*	- size of the allocation
in	*area*	- memory area where to allocate

Returns

address - address where memory was allocated

MemMgrErrorCode MemMgrBufferExport (size_t bufSize, char * csvBuffer, size_t * csvLenght)

Get memory's current state as a buffer structed into a .csv style format for easy export

Parameters

in	*bufSize*	- array of pointers to output lines
out	*csvBuffer*	- array of pointers to input lines
out	*csvLenght*	- array of values from convolution

Returns

error - error code

void MemMgrExportDraw (int dispWidth)

Graphically illustrate the current state of the memory mapped inside the memory manager

Parameters

in	*dispWidth*	- number of characters to display on one line

MemMgrErrorCode MemMgrFree (void * address)

Free up memory allocated at the inputed address

Parameters

in	*address*	- address where to free memory from

Returns

error - error code

7.25 MemMgrApi.h File Reference

Macros

- #define NO_VALID_MEM 0xFFFFFFFF
- #define MAX_MEMS_NR 9

Typedefs

- typedef uint32_t MemPoolId

Functions

- void MemMgrInit ()
- void MemMgrReset ()
- void MemMgrResetPool (MemPoolId pool)
- MemPoolId MemMgrAddPool (void *baseAdr, uint32_t nBytes)
- void * MemMgrAlloc (MemPoolId pool, uint32_t nBytes)
- uint32_t MemMgrGetFreeMem (MemPoolId pool)

7.25.1 Macro Definition Documentation

#define MAX_MEMS_NR 9

#define NO_VALID_MEM 0xFFFFFFFF

7.25.2 Typedef Documentation

typedef uint32_t **MemPoolId**

7.25.3 Function Documentation

MemPoolId MemMgrAddPool (void * baseAdr, uint32_t nBytes)

void* MemMgrAlloc (**MemPoolId** pool, uint32_t nBytes)

uint32_t MemMgrGetFreeMem (**MemPoolId** pool)

void MemMgrInit ( )

void MemMgrReset ( )

void MemMgrResetPool (**MemPoolId** pool)

7.26 MessageProtocol.h File Reference

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

Data Structures

- struct PhysicalChannel
- struct VirtualChannel

Macros

- #define MAX_COUNT_MESSAGING_PHYSICAL_CHANNELS 1
- #define MAX_COUNT_MESSAGING_VIRTUAL_CHANNELS 20
- #define DECLARE_MESSAGING_VIRTUAL_CHANNEL(vcHandle, channelName,channelId, priority,rx_fifo_size, tx_fifo_size, fifo_data_section)

Typedefs

- typedef void *(PacketWriteRequestCallback_t)(u16 length, u8 channel, u8 flags)
- typedef void *(PacketReadRequestCallback_t)(u16 length, u8 channel, u8 flags)
- typedef s32(PacketWriteDoneCallback_t)(u16 length, u8 channel, u8 flags, void *buffer)
- typedef s32(PacketReadDoneCallback_t)(u16 length, u8 channel, u8 flags, void *buffer)
- typedef s32(PacketExchangeOverCallback_t)(s32 *channel, void **buffer, s32 *size)

Enumerations

- enum ChannelType { SPISLAVE }
- enum txPending_t { TX_IDLE, TX_PENDING }

Functions

- void MessagePassingInitialize (void)
- s32 MessagePassingInitializePhysicalChannel (PhysicalChannel *phyChannel, void *phyChannelContext, ChannelType ct)
- s32 BaseMessagePassingInitializePhysicalChannel (PhysicalChannel *phyChannel, void *phyChannelContext, ChannelType ct)
- s32 MessagePassingRegisterVirtualChannel (VirtualChannel *vc)
- PacketWriteRequestCallback_t * MesasgePassingGetCbTxStart (PhysicalChannel pc)
- PacketWriteDoneCallback_t * MesasgePassingGetCbTxDone (PhysicalChannel pc)
- PacketReadRequestCallback_t * MesasgePassingGetCbRxStart (PhysicalChannel pc)
- PacketReadDoneCallback_t * MesasgePassingGetCbRxDone (PhysicalChannel pc)
- PacketExchangeOverCallback_t * MesasgePassingGetCbPeOver (PhysicalChannel pc)
- s32 MessagePassingWrite (u8 channel, void *buff, s32 size)
- s32 MessagePassingRead (u8 channel, void *buff, s32 size)
- s32 OsMessagePassingReadBlockEvent (u8 channel, void *buff, s32 size)
- void * MessagePassingGetDriverRxBuffer (u8 channel, s32 length)
- void MessagePassingFinalizeChannelRx (u8 channel)
- s32 MessagePassingFinalizePacketExchange (s32 *channel, void **buffer, s32 *size)
- VirtualChannel * MessagePassingGetVirtualChannel (u8 channelId)
- s32 BaseMessagePassingRead (VirtualChannel *vc, void *buff, s32 size)
- void BaseMessagePassingFinalizeChannelRx (VirtualChannel *vc)

7.26.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see: common/license.txt

7.27 MessRingBuff.h File Reference

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

Data Structures

- struct MessageRingBuffer

Macros

- #define MESSAGE_RING_BUFFER(buffer, rb_size)

Functions

- void InitMessageRB (MessageRingBuffer *mrb, void *buffer, s32 size)
- void * getMessageRBWrPtr (MessageRingBuffer *mrb, s32 neededLength)
- void finishMessageRBWrite (MessageRingBuffer *mrb)
- void * getMessageRBRdPtr (MessageRingBuffer *mrb, s32 *availableLength)

7.27.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see: common/license.txt

7.28 MestApi.h File Reference

MEST Component Functions API.

```
#include <stdint.h>
#include <mvMacros.h>
#include <MestPublicTypes.h>
```

Data Structures

- struct mestHandler_t

Macros

- #define ALIGNED(x) __attribute__((aligned(x)))
- #define MEST_GET_COORDS_HEAP_BYTES(ptsCnt) (ptsCnt * sizeof(uint32_t))
- #define MEST_GET_RES_HEAP_BYTES(ptsCnt) (ptsCnt * sizeof(uint32_t))
- #define MEST_GET_RES_BYTES(ptsCnt)

Functions

- mestError_t mestInitStaticConfigVGA (mestParamConfig_t *config)
- mestRuntimeConfig_t mestGetRuntimeConfig (const mestHandler_t *hnd)
- mestError_t mestSetRuntimeConfig (mestHandler_t *hnd, const mestRuntimeConfig_t *config)
- mestError_t mestMutexInit ()
- mestError_t mestMutexDeinit ()
- mestError_t mestInit (mestHandler_t *hnd, const mestMode_t mode, const mestResourceConfig_t *memory, const mestParamConfig_t *config)
- mestError_t mestDestroy ()
- mestError_t mestRun (mestHandler_t *hnd, frameBuffer prevFrame, frameBuffer currFrame, mestBulkOutput_t *result)
- uint32_t mestGetResultSize (mestHandler_t *hnd)
- int mestRunning (mestHandler_t *hnd)
- mestError_t mestAddROI (mestHandler_t *hnd, const mestROI_t *roi, const xyPos_t *locations, const uint32_t locationsCnt, const mestFeatureSelPolicy_t policy, mestROIId_t id)
- mestError_t mestRemoveROI (mestHandler_t *hnd, const mestROIId_t id)
- mestError_t mestRemoveAllROI (mestHandler_t *hnd)
- mestError_t mestUpdateROI (mestHandler_t *hnd, const mestROIId_t id, const mestROI_t *newRoi)
- mestError_t mestUpdateROIFeatures (mestHandler_t *hnd, const mestROIId_t id, xyPos_t *locations, uint32_t locationsCnt)
- mestROIId_t mestROIgetNextId (mestHandler_t *hnd)

Variables

- const uint16_t MEST_WRONG_INLIER_COST

7.28.1 Detailed Description

MEST Component Functions API.

Copyright

> All code copyright Movidius Ltd 2017, all rights reserved For License Warranty see: common/license.txt

This is the API to the MEST subsystem component

7.28.2 Macro Definition Documentation

#define ALIGNED(x) __attribute__((aligned(x)))

#define MEST_GET_COORDS_HEAP_BYTES(ptsCnt) (ptsCnt * sizeof(uint32_t))

Writes a buffer to a binary file.

Parameters

in	*buf*	- Address of the buffer to be written
in	*sz*	- Buffer length
in	*fn*	- Filename

#define MEST_GET_RES_BYTES(ptsCnt)

Value:

```
(\
        ALIGN_UP( (ptsCnt) * sizeof((*(mestMeta_t*)(0)).outputFlows[0]), 8 ) + \
        (ptsCnt) * sizeof((*(mestMeta_t*)(0)).costs[0]) + \
        sizeof(mestOutput_t) \
    )
```

#define MEST_GET_RES_HEAP_BYTES(ptsCnt) (ptsCnt ∗ sizeof(uint32_t))

7.28.3 Function Documentation

mestError_t mestAddROI (**mestHandler_t** ∗ hnd, const mestROI_t ∗ roi, const xyPos_t ∗ locations, const uint32_t locationsCnt, const mestFeatureSelPolicy_t policy, mestROIId_t id)

Adds a ROI to internal queue.

Parameters

in	*roi*	- ROI pointer to the element that needs to be added in queue
in	*locations*	- If needed, a custom target search points array can be supplied. Use NULL if it is configured for RASTER mode.
in	*locationsCnt*	- Size of the 'locations' array. Value is ignored if 'locations' is NULL.
in	*policy*	- Search policy for MEST. If is set to one of the RASTER modes, set 'locations' to NULL.
in	*id*	- ID of the current ROI

Returns

- MEST_ROI_SUCCESS MEST_ROI_ERROR MEST_ROI_INVALID_POINTS

mestError_t mestDestroy ( )

uint32_t mestGetResultSize (**mestHandler_t** ∗ hnd)

mestRuntimeConfig_t mestGetRuntimeConfig (const **mestHandler_t** ∗ hnd)

mestError_t mestInit (**mestHandler_t** ∗ hnd, const mestMode_t mode, const mestResourceConfig_t ∗ memory, const mestParamConfig_t ∗ config)

mestError_t mestInitStaticConfigVGA (mestParamConfig_t ∗ config)

mestError_t mestMutexDeinit ( )

mestError_t mestMutexInit ( )

mestError_t mestRemoveAllROI (**mestHandler_t** * hnd)

Removes all ROIs from queue.

Returns

- MEST_SUCCESS

mestError_t mestRemoveROI (**mestHandler_t** * hnd, const mestROIId_t id)

Removes a ROI from queue by its ID.

Parameters

in	*id*	- ID of the ROI to be removed

Returns

- MEST_SUCCESS MEST_ROI_NOT_FOUND

mestROIId_t mestROIgetNextId (**mestHandler_t** * hnd)

Get the next ROI ID that will be inserted in queue.

Returns

The ID of the next element in queue that will be inserted.

mestError_t mestRun (**mestHandler_t** * hnd, frameBuffer prevFrame, frameBuffer currFrame, mestBulkOutput_t * result)

int mestRunning (**mestHandler_t** * hnd)

mestError_t mestSetRuntimeConfig (**mestHandler_t** * hnd, const mestRuntimeConfig_t * config)

mestError_t mestUpdateROI (**mestHandler_t** * hnd, const mestROIId_t id, const mestROI_t * newRoi)

Replaces a ROI from queue specified by ID with a new ROI.

Parameters

in	*id*	- ID of the ROI to be replaced
in	*newRoi*	- new ROI that replaces the old one

Returns

- MEST_SUCCESS MEST_ROI_NOT_FOUND

mestError_t mestUpdateROIFeatures (**mestHandler_t** * hnd, const mestROIId_t id, xyPos_t * locations, uint32_t locationsCnt)

Replaces a ROI's list of points to search.

Parameters

in	*id*	- ID of the ROI to be updated
in	*locations*	- new array of coordinates
in	*locationsCnt*	- new array length

Returns

- MEST_SUCCESS MEST_ROI_NOT_FOUND MEST_ROI_QUEUE_EMPTY MEST_ROI_INVALID_POINTS

7.28.4 Variable Documentation

const uint16_t MEST_WRONG_INLIER_COST

7.29 MipiSendApi.h File Reference

Functions

- void mipiSendCreate (MipiCfg_t *cfg)
- void mipiSendInit (void)
- void mipiSendSentFrame (SendOutElement_t *task)
- void mipiSendFini (void)

7.29.1 Function Documentation

void mipiSendCreate (MipiCfg_t * cfg)

void mipiSendFini (void)

void mipiSendInit (void)

void mipiSendSentFrame (**SendOutElement_t** * task)

7.30 MipiSendApi.h File Reference

Functions

- void mipiSendCreate (MipiCfg_t *cfg)
- void mipiSendInit (void)
- void mipiSendSentFrame (SendOutElement_t *task)
- void mipiSendFini (void)

7.30.1 Function Documentation

void mipiSendCreate (MipiCfg_t * cfg)

void mipiSendFini (void)

void mipiSendInit (void)

void mipiSendSentFrame (**SendOutElement_t** * task)

7.31 Opipe.h File Reference

Opipe - API.

```
#include "OpipeBlocks.h"
#include "OpipeDefs.h"
```

Functions

- void OpipeTestInit ()
- void OpipeReset ()
- void OpipeInit (Opipe *p)
- uint32_t OpipeStart (Opipe *p)
- uint32_t OpipeWait (Opipe *p)
- void OpipeDetCfg (Opipe *p)
- DBuffer * OpipeCfgBuff (Opipe *p, uint32_t unitID, uint32_t flags, uint32_t cmxBuff, uint32_t cmxBuffH, uint32_t bpp)
- SwLink * OpipeSwLink (Opipe *p, uint32_t prodID, uint32_t allConsMask, uint32_t lastConsID)
- void defaultMipiTxLoopParams (oMipiTxLoopbackParam *mipiTxCfg, uint32_t txID, uint32_t iBuf, uint32_t bpp, uint32_t imgW, uint32_t imgH)
- void oCfgMipiTxLoopback (oMipiTxLoopbackParam *cfg)
- void oStartMipiTxLoopback (uint32_t txID)
- void OpipeWaitForRawStats ()
- void OpipeDelay (uint32_t numx100)
- void OpipeSetRes (Opipe *p, uint32_t newW, uint32_t newH)
- void OpipeForceU8fLuma (Opipe *p)

  Callback to override Luma format to u8f (default fp16)
- void DrvSetSliceDbyrLumaBuff (Opipe *p, u32 sWidth, u32 firstSlc)
- uint32_t DrvSetSippClkCtrlRegister (uint32_t mask_value)

7.31.1 Detailed Description

Opipe - API.

Author

andrei.lupas@movidius.com

Copyright

All code copyright Movidius Ltd 2015, all rights reserved. For License Warranty see: common/license.txt

7.31.2 Function Documentation

void defaultMipiTxLoopParams (**oMipiTxLoopbackParam** * mipiTxCfg, uint32_t txID, uint32_t iBuf, uint32_t bpp, uint32_t imgW, uint32_t imgH)

Set default timing values to MipiTx debug loopback structure.

Parameters

out	*mipiTxCfg*	reference to structure to be initialized
in	*txID*	sipp tx ID: SIPP_MIPI_TX0_ID or SIPP_MIPI_TX1_ID
in	*iBuf*	address of main full buffer to be sent over mipiTx
in	*bpp*	bytes per pixel
in	*imgW*	image width [pixels]
in	*imgH*	image height [pixels]

uint32_t DrvSetSippClkCtrlRegister (uint32_t mask_value) [inline]

Performs system initializations suitable for most tests. These include: clocks enabling, setting LeonL2 cache in WT mode, lowering Leon interrupt level, setup test module.

void DrvSetSliceDbyrLumaBuff (**Opipe** * p, u32 sWidth, u32 firstSlc)

Set SIPP image

Parameters

in	*p*	reference to pipeline of interest

void oCfgMipiTxLoopback (**oMipiTxLoopbackParam** * cfg)

Set the Mipi Tx->RX loopback mode as per argument struct

Parameters

in	*cfg*	reference to param structure

DBuffer∗ OpipeCfgBuff (**Opipe** ∗ p, uint32_t unitID, uint32_t flags, uint32_t cmxBuff, uint32_t cmxBuffH, uint32_t bpp)

Defines new CMX circular buffer

Parameters

in	*p*	reference to pipeline of interest
in	*unitID*	corresponding SIPP filter ID
in	*flags*	buffer flags: [DMA driven] and direction(mandatory)
in	*cmxBuff*	circular buffer base address
in	*lineW*	line width in pixels
in	*cmxBuffH*	circular buffer height [lines]
in	*bpp*	format (1,2,4 BPP or packed formats; see SIPP_FMT_∗)

Returns

reference to newly created buffer

void OpipeDelay (uint32_t numx100)

Wait NUMX100 groups of 100 nops (debug/test)

Parameters

in	*numx100*	number of 100NOP times to wait

void OpipeDetCfg (**Opipe** ∗ p)

Computes value to be OR-ed into SIPP_OPIPE_CFG_ADR

Parameters

in	*p*	reference to pipeline of interest

void OpipeForceU8fLuma (**Opipe** ∗ p)

Callback to override Luma format to u8f (default fp16)

void OpipeInit (**Opipe** ∗ p)

Performs individual pipeline initialization

Parameters

in	*p*	reference to pipeline to be initialized

void OpipeReset ( )

Performs Opipe general initialization

void OpipeSetRes (**Opipe** * p, uint32_t newW, uint32_t newH)

Set new resolution

Parameters

in	*p*	reference to pipeline of interest
in	*newW*	new main input width
in	*newH*	new main input height

uint32_t OpipeStart (**Opipe** * p)

Starts individual pipeline

Parameters

in	*p*	reference to pipeline to be started

SwLink* OpipeSwLink (**Opipe** * p, uint32_t prodID, uint32_t allConsMask, uint32_t lastConsID)

Performs system initializations suitable for most tests. These include: clocks enabling, setting LeonL2 cache in WT mode, lowering Leon interrupt level, setup test module.

void OpipeTestInit ( )

Performs system initializations suitable for most tests. These include: clocks enabling, setting LeonL2 cache in WT mode, lowering Leon interrupt level, setup test module.

uint32_t OpipeWait (**Opipe** * p)

Waits for completion of current pipeline (BLOCKING!)

Parameters

in	*p*	reference to pipeline to be waited for

void OpipeWaitForRawStats ( )

Wait till RAW stats are written to memory (test purpose mainly)

void oStartMipiTxLoopback (uint32_t txID)

Start MipiTx timing generator.

Parameters

in	*txID*	sipp tx ID: SIPP_MIPI_TX0_ID or SIPP_MIPI_TX1_ID

7.32 OpipeBlocks.h File Reference

Opipe - SIPP blocks config data structs.

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

Data Structures

- struct BlcCfg
- struct SigmaDnsCfg
- struct LscCfg
- struct RawCfg
- struct AeAwbCfg
- struct AfCfg
- struct AeAwbPatchStats
- struct AfPatchStats
- struct DbyrCfg
- struct LtmCfg
- struct DogCfg
- struct LumaDnsCfg
- struct LumaDnsRefCfg
- struct SharpenCfg
- struct ChromaGenCfg
- struct MedianCfg
- struct ChromaDnsCfg
- struct ColCombCfg
- struct LutCfg
- struct ColConvCfg
- struct UpfirdnCfg
- struct ConvCfg
- struct MipiRxCfg
- struct HarrisCfg

7.32.1 Detailed Description

Opipe - SIPP blocks config data structs.

Copyright

> All code copyright Movidius Ltd 2015, all rights reserved. For License Warranty see: common/license.txt

7.33 OpipeDefs.h File Reference

Opipe - definitions.

```
#include <DrvCmxDma.h>
#include "SippHwDefs.h"
```

Data Structures

- struct RectRgn
- struct PBuffer
- struct SBuffer
- struct MBuffer
- struct DBuffer
- struct _SwLink
- struct OpipeGlobal
- struct oMipiTxLoopbackParam

  Mipi-TX loopback debug params.
- struct OpipeS

  Main O-Pipe data struct.

- #define RDIR "../resources/"
- #define LUMA_BASED_BLEND 1
- #define IRQ_RATE_POW 3
- #define IRQ_RATE (1<<IRQ_RATE_POW)
- #define SIPP_IRQ_LEVEL 10
- #define ALIGNED(x)
- #define SECTION(x)
- #define D_IN 1
- #define D_OUT 2
- #define D_DMA 4
- #define DBYR_Y_H 32
- #define SHARP_Y_H 8
- #define LUT_H 16
- #define I_CBUFF_H (3*IRQ_RATE)
- #define O_CBUFF_H (2*IRQ_RATE)
- #define BPP(x) x
- #define NPL(x) x
- #define MAX_SOURCES 4
- #define MAX_SINKS 4
- #define N_DESCS 4
- #define CDMA_DEF_REQ 3
- #define SIPP_FMT_8BIT 0x1
- #define SIPP_FMT_16BIT 0x2
- #define SIPP_FMT_32BIT 0x4
- #define SIPP_FMT_PACK10 0x5

- #define SIPP_FMT_PACK12 0x3
- #define GEN_PREVIEW (1<<0)
- #define DETERMINED_CFG (1<<1)
- #define PIPE_RUNNING (1<<2)
- #define PREVIEW_ABLE (1<<3)
- #define LOST_IRQ (1<<4)
- #define CLEAN_EXIT (1<<5)
- #define DW_ALIGN(x) (((x) + 7) & (~7))
- typedef void(* deferCb)(struct _SwLink *swLink, uint32_t nLines)
- typedef struct _SwLink SwLink
- typedef struct OpipeS Opipe
- typedef void(* opipeCb)(Opipe *p)
- uint32_t OpipeFinished (Opipe *p)
- uint32_t OpipeDisable (Opipe *p)
- void oMemCompare (void *_refA, void *_refB, int len)

7.33.1 Detailed Description

Opipe - definitions.

Copyright

> All code copyright Movidius Ltd 2015, all rights reserved. For License Warranty see: common/license.txt

7.33.2 Macro Definition Documentation

#define ALIGNED(x)

#define BPP(x) x

#define CDMA_DEF_REQ 3

#define CLEAN_EXIT (1<<5)

#define D_DMA 4

#define D_IN 1

#define D_OUT 2

#define DBYR_Y_H 32

#define DETERMINED_CFG (1<<1)

#define DW_ALIGN(x) (((x) + 7) & (~7))

#define GEN_PREVIEW (1<<0)

#define I_CBUFF_H (3∗**IRQ_RATE**)

#define IRQ_RATE (1<<**IRQ_RATE_POW**)

#define IRQ_RATE_POW 3

#define LOST_IRQ (1<<4)

#define LUMA_BASED_BLEND 1

#define LUT_H 16

#define MAX_SINKS 4

#define MAX_SOURCES 4

#define N_DESCS 4

#define NPL(x) x

#define O_CBUFF_H (2∗**IRQ_RATE**)

#define PIPE_RUNNING (1<<2)

#define PREVIEW_ABLE (1<<3)

#define RDIR "../resources/"

#define SECTION(x)

#define SHARP_Y_H 8

#define SIPP_FMT_16BIT 0x2

#define SIPP_FMT_32BIT 0x4

#define SIPP_FMT_8BIT 0x1

#define SIPP_FMT_PACK10 0x5

#define SIPP_FMT_PACK12 0x3

#define SIPP_IRQ_LEVEL 10

7.33.3 Typedef Documentation

typedef void(∗ deferCb)(struct **_SwLink** ∗swLink, uint32_t nLines)

typedef struct **OpipeS Opipe**

typedef void(∗ opipeCb)(**Opipe** ∗p)

typedef struct **_SwLink SwLink**

7.33.4 Function Documentation

void oMemCompare (void * _refA, void * _refB, int len)

uint32_t OpipeDisable (**Opipe** * p)

uint32_t OpipeFinished (**Opipe** * p)

7.34 opticalFlowApi.h File Reference

```
#include "mv_types.h"
#include "theDynContext.h"
#include "vTrackOutput.h"
#include <mvMacros.h>
#include "global_constants.h"
#include "opticalFlowTypes.h"
```

Data Structures

- struct ofResourceCfg
- class OpticalFlow

Macros

- #define MAX_OF_TASKS 20
- #define MAX_SHAVES_OF 10

Typedefs

- typedef u32 swcShaveUnit_t
- typedef struct ofResourceCfg ofResourceCfg_t

Functions

- class OpticalFlow ALIGNED (64)
- OpticalFlow ()
- ~OpticalFlow ()
- void ofInit (tvOpticalFlowCfg *algConfig, ofResourceCfg_t *resCfg)
- int ofRun (tvPyramidBuffer *pyrImgPrev, tvPyramidBuffer *pyrImgCur, tvXYLoc features-Prev[ ], tvXYLoc featuresCur[ ], fp32 featuresError[ ], u32 featuresCount[ ], u32 numCells, u32 maxFeatPerCell)

Variables

- uint8_t * MEST_COORDS_HEAP
- uint8_t * MEST_RES_TMP_HEAP
- uint8_t * MEST_RES_HEAP
- u32 numShaves
- swcShaveUnit_t * shaveList
- tvOpticalFlowCfg ofAlgConfig
- uint8_t cacheData
- uint8_t cacheInstr

7.34.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved. For License Warranty see: common/license.txt

7.34.2 Macro Definition Documentation

#define MAX_OF_TASKS 20

#define MAX_SHAVES_OF 10

7.34.3 Typedef Documentation

typedef struct **ofResourceCfg ofResourceCfg_t**

typedef u32 **swcShaveUnit_t**

7.34.4 Function Documentation

class **OpticalFlow** ALIGNED (64)

void ALIGNED::ofInit (tvOpticalFlowCfg * algConfig, **ofResourceCfg_t** * resCfg)

int ALIGNED::ofRun (tvPyramidBuffer * pyrImgPrev, tvPyramidBuffer * pyrImgCur, tvXYLoc featuresPrev[ ], tvXYLoc featuresCur[ ], fp32 featuresError[ ], u32 featuresCount[ ], u32 numCells, u32 maxFeatPerCell)

ALIGNED::OpticalFlow ( )

ALIGNED::~OpticalFlow ( )

7.34.5 Variable Documentation

uint8_t cacheData

uint8_t cacheInstr

uint8_t* MEST_COORDS_HEAP

uint8_t* MEST_RES_HEAP

uint8_t* MEST_RES_TMP_HEAP

u32 numShaves

tvOpticalFlowCfg ofAlgConfig

swcShaveUnit_t* shaveList

7.35 OsDrvSpiSlaveCP.h File Reference

```
#include <DrvSpiSlaveCP.h>
#include <bsp.h>
#include <bsp/irq-generic.h>
#include <rtems/status-checks.h>
```

Data Structures

- struct spiSlaveCommunicationConfiguration_t

Macros

- #define DRVSPI_CONFIGURATION(dev, cpol, cpha, bytesPerWord, dmaEnabled, hostIrqGpio, interruptLevel)

Functions

- rtems_status_code OsDrvSpiSlaveCPInitGlobally (spiHandler_t *handler, spiTxStartCallback_t *txStartCb, spiTxDoneCallback_t *txDoneCb, spiRxStartCallback_t *rxStartCb, spiRxDoneCallback_t *rxDoneCb, spiPeDoneCallback_t *peOverCb)
- rtems_status_code OsDrvSpiSlaveCPInit (spiHandler_t *handler, wordSizeBytes_t wordSizeBytes, dmaUsed_t useDma, spiSlaveBlock_t device, u32 cpol, u32 cpha, u32 interruptLevel, u32 hostIrqGpio, spiTxStartCallback_t *txStartCb, spiTxDoneCallback_t *txDoneCb, spiRxStartCallback_t *rxStartCb, spiRxDoneCallback_t *rxDoneCb, spiPeDoneCallback_t *peOverCb)
- rtems_status_code OsDrvSpiSlaveCPSendPacket (spiHandler_t *handler, u8 channel, u8 flags, s32 size, void *buff)

Variables

- spiSlaveCommunicationConfiguration_t spiConfig

7.35.1 Detailed Description

7.36 OsMessageProtocol.h File Reference

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

Data Structures

- struct OsVirtualChannel

Macros

- #define DECLARE_OS_MESSAGING_VIRTUAL_CHANNEL(vcHandle, channelName,channel-Id, priority,rx_fifo_size, tx_fifo_size, fifo_data_section)
- #define DEV_VIRTUAL_CHANNEL_DRIVER_TABLE_ENTRY
- #define DECLARE_COMMUNICATION_PROTOCOL_DRIVER_TABLE(drvTblName) rtems-_driver_address_table drvTblName = DEV_VIRTUAL_CHANNEL_DRIVER_TABLE_ENTR-Y;

Functions

- void OsMessagePassingInitialize (void)
- s32 OsMessagePassingInitializePhysicalChannel (PhysicalChannel *phyChannel, void *phy-ChannelContext, ChannelType ct)
- s32 OsMessagePassingRegisterVirtualChannel (PhysicalChannel *phyChannel, VirtualChannel *vc)
- s32 OsMessagePassingReadBlockEvent (u8 channel, void *buff, s32 size)
- PacketWriteRequestCallback_t * OsMesasgePassingGetCbTxStart (PhysicalChannel pcList)
- PacketReadRequestCallback_t * OsMesasgePassingGetCbRxStart (PhysicalChannel pcList)
- PacketWriteDoneCallback_t * OsMesasgePassingGetCbTxDone (PhysicalChannel pcList)
- PacketReadDoneCallback_t * OsMesasgePassingGetCbRxDone (PhysicalChannel pcList)
- PacketExchangeOverCallback_t * OsMesasgePassingGetCbPeOver (PhysicalChannel pcList)
- rtems_device_driver virtual_channel_initialize (rtems_device_major_number major, rtems_-device_minor_number minor, void *e)
- rtems_device_driver virtual_channel_open (rtems_device_major_number major, rtems_device_-minor_number minor, void *e)
- rtems_device_driver virtual_channel_close (rtems_device_major_number major, rtems_device_-minor_number minor, void *e)
- rtems_device_driver virtual_channel_read (rtems_device_major_number major, rtems_device_-minor_number minor, void *e)
- rtems_device_driver virtual_channel_write (rtems_device_major_number major, rtems_device_-minor_number minor, void *e)
- rtems_device_driver virtual_channel_control (rtems_device_major_number major, rtems_device-_minor_number minor, void *e)

7.36.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see-: common/license.txt

7.37 PatternGeneratorApi.h File Reference

```
#include "PatternGeneratorApiDefines.h"
#include "swcFrameTypes.h"
```

Functions

- unsigned int CreateHorizontalColorBars (frameBuffer *frame, unsigned int interlaced)

  Create an horizontal pattern at address specified in param list.
- unsigned int CreateVerticalColorBars (frameBuffer *frame, unsigned int interlaced)

  Create an vertical pattern at address specified in param list.
- unsigned int CreateColorStripesPattern (frameBuffer *frame)

  Create an specific horizontal pattern with 64 color stripes for first half and 8 color stripes for second half.
- unsigned int CreateLinearGreyPattern (frameBuffer *frame)

  Create an specific vertical pattern with 17 grey stripes, from black to white.
- unsigned int PatternCheck (frameBuffer *inputFrame, frameBuffer *outputFrame, unsigned int interlaced)

  Check if patterns are correct by comparing.

7.37.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved. For License Warranty see: common/license.txt

7.38 PixelPipeApi.h File Reference

```
#include "mv_types.h"
#include "theDynContext.h"
#include <swcFrameTypes.h>
#include "vTrackPrivateTypes.h"
#include "pixelPipeHelper.h"
```

Data Structures

- struct t_pPipeShaveConfig

- struct ppThresholds_t
- struct t_ppFifoCfg
- struct t_pPipeResourceCfg
- class PixelPipe

Macros

- #define NUM_SHAVES_SLAVE 10

Typedefs

- typedef u32 swcShaveUnit_t
- typedef struct fifoCommMasterHandler_t fifoCommMasterHandler_t
- typedef struct fifoCommSlaveHandler_t fifoCommSlaveHandler_t
- typedef struct fifoCommTask_t fifoCommTask_t

Functions

- void UpdateCellThresholds (fp32 *thresholds, tvFeatureCell *featureCells, ppThresholds_t *thresholdCfg, u32 nCells, u32 targetNumFeatures)
- class PixelPipe ALIGNED (64)
- PixelPipe ()
- void ppInit (t_pPipeResourceCfg *vpResource, pyramidAlgoType_t pyrAlg, cornerConfig_t corCfg)
- u32 ppRun (frameBuffer *in_img, tvFeatureCell **feature_cells, tvPyramidBuffer *frameBuffer, fp32 *thresholds, u32 num_pyr_levels, u32 num_pyrs, u32 cellGridDimension, u32 maxNumFeatures, u32 targetNumFeatures, ppThresholds_t *thresholdCfg)
- void initPixelPipe (t_pPipeResourceCfg *vpResource, pyramidAlgoType_t pyrAlg, cornerConfig_t corCfg)
- u32 pixelPipe (frameBuffer *in_img, tvFeatureCell **feature_cells, tvPyramidBuffer *frameBuffer, fp32 *thresholds, u32 num_pyr_levels, u32 num_pyrs, u32 cellGridDimension, u32 maxNumFeatures, u32 targetNumFeatures, ppThresholds_t *thresholdCfg)
- u8 shaveBuf[1088] __attribute__ ((aligned(64)))

Variables

- struct t_pPipeShaveConfig ALIGNED
- swcShaveUnit_t * gaussShavesList
- swcShaveUnit_t * cornerShavesList
- swcShaveUnit_t ppMasterShaveNum
- swcShaveUnit_t slaveShaves [NUM_SHAVES_SLAVE]
- u32 slaveNumShaves
- u32 gaussNumShaves
- u32 cornerNumShaves
- pixelPipeParams_t * pixelPipeParams

- u8 ppCacheData
- u8 ppCacheInstr
- fifoCommMasterHandler_t * masterHandler
- fifoCommSlaveHandler_t * slaveHandler
- fifoCommTask_t * shaveTaskTypes
- fifoCommTask_t * taskTypes

7.38.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved. For License Warranty see: common/license.txt

7.38.2 Macro Definition Documentation

#define NUM_SHAVES_SLAVE 10

7.38.3 Typedef Documentation

typedef struct **fifoCommMasterHandler_t fifoCommMasterHandler_t**

typedef struct **fifoCommSlaveHandler_t fifoCommSlaveHandler_t**

typedef struct **fifoCommTask_t fifoCommTask_t**

typedef u32 **swcShaveUnit_t**

7.38.4 Function Documentation

u8 shaveBuf [1088] ALIGNED::__attribute__ ((aligned(64))) `[private]`

class **PixelPipe** ALIGNED (64)

void ALIGNED::initPixelPipe (**t_pPipeResourceCfg** * vpResource, pyramidAlgoType_t pyrAlg, cornerConfig_t corCfg) `[private]`

ALIGNED::PixelPipe ( )

u32 ALIGNED::pixelPipe (frameBuffer * in_img, tvFeatureCell ** feature_cells, tvPyramidBuffer * frameBuffer, fp32 * thresholds, u32 num_pyr_levels, u32 num_pyrs, u32 cellGridDimension, u32 maxNumFeatures, u32 targetNumFeatures, **ppThresholds_t** * thresholdCfg) `[private]`

void ALIGNED::ppInit (**t_pPipeResourceCfg** * vpResource, pyramidAlgoType_t pyrAlg, cornerConfig_t corCfg)

u32 ALIGNED::ppRun (frameBuffer * in_img, tvFeatureCell ** feature_cells, tvPyramidBuffer * frameBuffer, fp32 * thresholds, u32 num_pyr_levels, u32 num_pyrs, u32 cellGridDimension, u32 maxNumFeatures, u32 targetNumFeatures, **ppThresholds_t** * thresholdCfg)

void UpdateCellThresholds (fp32 * thresholds, tvFeatureCell * featureCells, **ppThresholds_t** * thresholdCfg, u32 nCells, u32 targetNumFeatures)

7.38.5 Variable Documentation

DynamicContextInstancesPtr gaussContextInstanceDataPtr ALIGNED

u32 cornerNumShaves

swcShaveUnit_t* cornerShavesList

u32 gaussNumShaves

swcShaveUnit_t* gaussShavesList

fifoCommMasterHandler_t* masterHandler

pixelPipeParams_t* pixelPipeParams

u8 ppCacheData

u8 ppCacheInstr

swcShaveUnit_t ppMasterShaveNum

fifoCommTask_t* shaveTaskTypes

fifoCommSlaveHandler_t* slaveHandler

u32 slaveNumShaves

swcShaveUnit_t slaveShaves[**NUM_SHAVES_SLAVE**]

fifoCommTask_t* taskTypes

7.39 PlgFifoApi.h File Reference

Data Structures

- struct PlgFifoElemS
- struct PlgFifoS

Macros

- #define PLGFIFO_SZ 32
- #define PLGFIFO_MAX_NR_OF_INPUTS 6

Typedefs

- typedef struct PlgFifoElemS PlgFifoElem
- typedef struct PlgFifoS PlgFifo

Functions

- void PlgFifoCreate (void *pluginObj)
- void PlgFifoConfig (void *pluginObj, uint32_t NoOfInputs)

7.39.1 Macro Definition Documentation

#define PLGFIFO_MAX_NR_OF_INPUTS 6

#define PLGFIFO_SZ 32

7.39.2 Typedef Documentation

typedef struct **PlgFifoS PlgFifo**

typedef struct **PlgFifoElemS PlgFifoElem**

7.39.3 Function Documentation

void PlgFifoConfig (void * pluginObj, uint32_t NoOfInputs)

void PlgFifoCreate (void * pluginObj)

7.40 PlgIspFullApi.h File Reference

Data Structures

- struct PlgIspFullStruct

Typedefs

- typedef struct PlgIspFullStruct PlgIspFull

Enumerations

- enum PlgIspFullStatus { PLG_ISPFULL_NOTMADE = 0, PLG_ISPFULL_CREATED = 1, PLG_ISPFULL_INUSE = 2 }

Functions

- void PlgIspFullCreate (void *pluginObject)
- void PlgIspFullConfig (void *pluginObject, icSize frameSz, uint32_t inFmt, uint32_t prevAble)

7.40.1 Typedef Documentation

typedef struct **PlgIspFullStruct PlgIspFull**

7.40.2 Enumeration Type Documentation

enum **PlgIspFullStatus**

Enumerator

PLG_ISPFULL_NOTMADE
PLG_ISPFULL_CREATED
PLG_ISPFULL_INUSE

7.40.3 Function Documentation

void PlgIspFullConfig (void * pluginObject, icSize frameSz, uint32_t inFmt, uint32_t prevAble)

void PlgIspFullCreate (void * pluginObject)

7.41 PlgSourceApi.h File Reference

Data Structures

- struct PlgSource

Enumerations

- enum PlgSourceStatus { PLG_SOURCE_NOTMADE = 0, PLG_SOURCE_CREATED = 1, PLG_SOURCE_INUSE = 2 }
- enum LineRefs { SOF_IDX_SRC = 0, EOF_IDX_SRC = 1, HIT_IDX_SRC = 2 }

Functions

- void PlgSourceCmxAlloc (uint32_t instNo, icSourceSetup *src, MemPoolId id)
- void PlgSourceStart (void *pluginObject, icSourceConfig *sourceConfig, uint32_t outFmt)
- void PlgSourceCreate (void *pluginObject, icSourceInstance instId)
- void PlgSourceSetLineHit (void *pluginObject, uint32_t lineNo)

7.41.1 Enumeration Type Documentation

enum **LineRefs**

Enumerator

SOF_IDX_SRC
EOF_IDX_SRC
HIT_IDX_SRC

enum **PlgSourceStatus**

Enumerator

PLG_SOURCE_NOTMADE
PLG_SOURCE_CREATED
PLG_SOURCE_INUSE

7.41.2 Function Documentation

void PlgSourceCmxAlloc (uint32_t instNo, icSourceSetup * src, **MemPoolId** id)

void PlgSourceCreate (void * pluginObject, icSourceInstance instId)

void PlgSourceSetLineHit (void * pluginObject, uint32_t lineNo)

void PlgSourceStart (void * pluginObject, icSourceConfig * sourceConfig, uint32_t outFmt)

7.42 PlgSrcIspApi.h File Reference

```
#include "PlgTypes.h"
#include "ipipe.h"
#include "Opipe.h"
#include "PlgSrcPipeDef.h"
#include "IspCommonUtils.h"
#include "DrvMipiDefines.h"
#include "MemMgrApi.h"
```

Data Structures

- struct PlgSrcIsp

Enumerations

- enum PlgSrcIspStatus { PLG_SRC_ISP_NOTMADE = 0, PLG_SRC_ISP_CREATED = 1, PLG_SRC_ISP_INUSE = 2 }
- enum LineRefsSrc { SOF_IDX = 0, EOF_IDX = 1, HIT_IDX = 2 }
- enum PlgSrcIspErrors { ERR_PLGSRCISP_NO_OUT_BUF, ERR_PLGSRCISP_NO_ISP_CFG }

Functions

- void PlgSrcIspCmxAlloc (PlgSrcIsp *obj, icSourceSetup *src, MemPoolId id)
- void PlgSrcIspStart (void *pluginObject, icSourceConfig *sourceConfig)
- void PlgSrcIspCreate (void *pluginObject, icSourceInstance instId)
- void PlgSrcIspSetLineHit (void *pluginObject, uint32_t lineNo)

7.42.1 Enumeration Type Documentation

enum **LineRefsSrc**

Enumerator

SOF_IDX

EOF_IDX

HIT_IDX

enum **PlgSrcIspErrors**

Enumerator

ERR_PLGSRCISP_NO_OUT_BUF

ERR_PLGSRCISP_NO_ISP_CFG

enum **PlgSrcIspStatus**

Enumerator

PLG_SRC_ISP_NOTMADE

PLG_SRC_ISP_CREATED

PLG_SRC_ISP_INUSE

7.42.2 Function Documentation

void PlgSrcIspCmxAlloc (**PlgSrcIsp** * obj, icSourceSetup * src, **MemPoolId** id)

void PlgSrcIspCreate (void * pluginObject, icSourceInstance instId)

void PlgSrcIspSetLineHit (void * pluginObject, uint32_t lineNo)

void PlgSrcIspStart (void * pluginObject, icSourceConfig * sourceConfig)

7.43 sendOutApi.h File Reference

Data Structures

- struct SendOutInitCfg_t
- struct client_tx_frame_header_t
- struct send_out_tx_buffer_header_t
- struct SendOutElement_t

Typedefs

- typedef void(* SendOutCbSent)(FrameT *frame, uint32_t outputId, uint32_t frmType)
- typedef void(* InternalCbSent)(void *task)

Functions

- void sendOutCreate (SendOutInitCfg_t *cfg)
- void sendOutInit (void)
- void sendOutControl (uint32_t camera_en_bit_mask, uint32_t frame_type_en_bit_mask, uint32_t frame_format_en_bit_mask)
- void sendOutSend (FrameT *frame, uint32_t outputId, uint32_t frmType, SendOutCbSent sendOutCbSent)
- void sendOutFini ()
- send_out_tx_buffer_header_t * sendOutGetBufferHeader (void)

Variables

- SendOutInitCfg_t sendOut_initCfg
- uint32_t dbgEnableOutput

7.43.1 Typedef Documentation

typedef void(* InternalCbSent)(void *task)

typedef void(* SendOutCbSent)(FrameT *frame, uint32_t outputId, uint32_t frmType)

7.43.2 Function Documentation

void sendOutControl (uint32_t camera_en_bit_mask, uint32_t frame_type_en_bit_mask, uint32_t frame_format_en_bit_mask)

void sendOutCreate (**SendOutInitCfg_t** * cfg)

void sendOutFini ( )

send_out_tx_buffer_header_t* sendOutGetBufferHeader (void)

void sendOutInit (void)

void sendOutSend (FrameT * frame, uint32_t outputId, uint32_t frmType, **SendOutCbSent** sendOutCbSent)

7.43.3 Variable Documentation

uint32_t dbgEnableOutput

SendOutInitCfg_t sendOut_initCfg

7.44 sendOutApi.h File Reference

Data Structures

- struct SendOutInitCfg_t
- struct client_tx_frame_header_t
- struct send_out_tx_buffer_header_t
- struct SendOutElement_t

Typedefs

- typedef void(* SendOutCbSent)(FrameT *frame, uint32_t outputId, uint32_t frmType)
- typedef void(* InternalCbSent)(void *task)

Functions

- void sendOutCreate (SendOutInitCfg_t *cfg)

- void sendOutInit (void)
- void sendOutControl (uint32_t camera_en_bit_mask, uint32_t frame_type_en_bit_mask, uint32_t frame_format_en_bit_mask)
- void sendOutSend (FrameT *frame, uint32_t outputId, uint32_t frmType, SendOutCbSent sendOutCbSent)
- void sendOutFini ()
- send_out_tx_buffer_header_t * sendOutGetBufferHeader (void)

Variables

- SendOutInitCfg_t sendOut_initCfg
- uint32_t dbgEnableOutput

7.44.1 Typedef Documentation

typedef void(* InternalCbSent)(void *task)

typedef void(* SendOutCbSent)(FrameT *frame, uint32_t outputId, uint32_t frmType)

7.44.2 Function Documentation

void sendOutControl (uint32_t camera_en_bit_mask, uint32_t frame_type_en_bit_mask, uint32_t frame_format_en_bit_mask)

void sendOutCreate (**SendOutInitCfg_t** * cfg)

void sendOutFini ( )

send_out_tx_buffer_header_t* sendOutGetBufferHeader (void)

void sendOutInit (void)

void sendOutSend (FrameT * frame, uint32_t outputId, uint32_t frmType, **SendOutCbSent** sendOutCbSent)

7.44.3 Variable Documentation

uint32_t dbgEnableOutput

SendOutInitCfg_t sendOut_initCfg

7.45 SuspendLrtLpApi.h File Reference

Suspend LRT component: API definitions.

Macros

- #define LP_INTERRUPT_PRIORITY 0xE
- #define LP_INTERRUPT_NUM IRQ_DYNAMIC_0

Functions

- void SuspendLrt (void)
- void switchStackAndEntersLp_asm (void)
- void restoreStackAndExitLp_asm (void)

7.45.1 Detailed Description

Suspend LRT component: API definitions.

Copyright

> All code copyright Movidius Ltd 2017, all rights reserved For License Warranty see-: common/license.txt

7.45.2 Macro Definition Documentation

#define LP_INTERRUPT_NUM IRQ_DYNAMIC_0

#define LP_INTERRUPT_PRIORITY 0xE

7.45.3 Function Documentation

void restoreStackAndExitLp_asm (void)

void SuspendLrt (void)

void switchStackAndEntersLp_asm (void)

7.46 testUtilsApi.h File Reference

Software test library.

```
#include <testUtilsApiDefines.h>
```

Functions

- void shaveProfileInit (performanceStruct *perfStruct, u32 stallsTypes)
  Initializes the performance counters and its performance structure.
- void shaveProfileReset (void)
  Resets values registered by the 4 counters.
- void shaveProfileStart (performanceStruct *perfStruct)
  Enables history registers and counters.
- u32 shaveGetFieldValue (performanceStruct *perfStruct, performanceCounterDef conterType)
  Gets value from given counterType.
- void shaveProfilePrint (performanceStruct *perfStruct)
  Prints all the default measured cycles.
- void shaveShowCounterValues (performanceStruct *perfStruct)
  Gets all the counter values and prints them.

7.46.1 Detailed Description

Software test library.

Copyright

> All code copyright Movidius Ltd 2016, all rights reserved. For License Warranty see: common/license.txt

7.46.2 Function Documentation

u32 shaveGetFieldValue (performanceStruct * perfStruct, performanceCounterDef conterType)

Gets value from given counterType.

Parameters

in	*perfStruct*	- performance structure where counter values are stored
in	*counterType*	- either instruction, stall, clock cycle or branch

Returns

> fieldValue - value of one of the 4 counterTypes retreved from perfStruct

void shaveProfileInit (performanceStruct * perfStruct, u32 stallsTypes)

Initializes the performance counters and its performance structure.

1: Includes

2: Source Specific #defines and types (typedef, enum, struct)

3: Static Local Data

4: Exported Functions (non-inline)

Parameters

in	*perfStruct*	- performance structure where counter values are stored
in	*stallsTypes*	- the types of stalls the user wants to count

Returns

> void

void shaveProfilePrint (performanceStruct * perfStruct)

Prints all the default measured cycles.

Parameters

in	*perfStruct*	- performance structure where counter values are stored

Returns

void

void shaveProfileReset (void)

Resets values registered by the 4 counters.

Returns

void

void shaveProfileStart (performanceStruct * perfStruct)

Enables history registers and counters.

Parameters

in	*perfStruct*	- performance structure where counter values are stored

Returns

void

void shaveShowCounterValues (performanceStruct * perfStruct)

Gets all the counter values and prints them.

Parameters

in	*perfStruct*	- performance structure where counter values are stored

Returns

void

7.47 UnitTestApi.h File Reference

```
#include "UnitTestDefines.h"
```

Functions

- int unitTestInit (void)

  Initiate a new unit test.

- int unitTestFloatWithinRange (float actual, float expected, float percentageError)

  Checks if a floating point value is within a certain percentage of expected value.
- int unitTestExecutionWithinRange (float actual, float expected, float percentageError)

  Checks if a floating point value is within an acceptable margin expected value.
- int unitTestFloatAbsRangeCheck (float actual, float expected, float AbsError)

  Checks if a floating point value is within an acceptable margin expected value.
- int unitTestAssert (int value)

  Check if a logical condition is true or not.
- int unitTestLogPass (void)

  Log a passed test.
- int unitTestLogFail (void)

  Log a failed test.
- int unitTestLogResults (int passes, int fails)

  Set up the passed and failed tests results.
- void unitTestMemCompare (const void *pActualStart, const void *pExpectedStart, u32 lengthBytes)

  Compares two values.
- void unitTestMemCompareDeltaU8 (u8 *pActualStart, u8 *pExpectedStart, u32 lengthBytes, u8 delta)

  Compares two values, with threshold.
- void unitTestCompare (u32 actualValue, u32 expectedValue)
- int unitTestCheckSectionFail (char *sectionName)
- int unitTestFinalReport (void)

  Print the final results of the unit testing.

7.47.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see: common/license.txt

7.48 UnitTestApi.h File Reference

```
#include "UnitTestDefines.h"
```

Functions

- void unitTestInit (void)

  Initialise the Unit Test Framework.
- void unitTestVerbosity (tyUnitVerbosity targetVerbosity)

  Set expected verbosity of the unitTest library.
- void unitTestAssert (int value)

  Assert that value passed is TRUE (non-zero)
- void unitTestCompare (u32 actualValue, u32 expectedValue)

Compare 2 32 bit values, log error if they don't match.

- void unitTestCompare64 (u64 actualValue, u64 expectedValue)

  Compare 2 64 bit values, log error if they don't match.
- void unitTestReadDWordCheck (void *dWordAddress, u64 expectedValue)

  Read a dword from memory and compare against expected value. Log a failure if values don't match.
- void unitTestReadWordCheck (void *wordAddress, u32 expectedValue)

  Read a word from memory and compare against expected value. Log a failure if values don't match.
- void unitTestReadHalfCheck (void *address, u16 expectedValue)

  Read a 16 bit value from memory and compare against expected value. Log a failure if values don't match.
- void unitTestReadByteCheck (void *address, u8 expectedValue)

  Read a byte from memory and compare against expected value. Log a failure if values don't match.
- void unitTestReadBitCheck (void *wordAddress, u32 startBit, u32 numBits, u32 expectedValue)

  Read a word from memory and compare a number of bits from the value against an expected result. Log a failure if values don't match.
- void unitTestMemCompare (const void *pActualStart, const void *pExpectedStart, u32 lengthBytes)

  Compare two memory buffers for a given number of bytes.
- void unitTestCrcCheck (const void *pStart, u32 lengthBytes, u32 expectedCrc)

  Perform 32 bit CRC on the buffer of lengthBytes and compare against expectedCrc.
- void unitTestExecutionWithinRange (float actual, float expected, float percentageError)

  Checks if a floating point value is within an acceptable margin expected value.
- u32 unitTestFinalReport (void)

  Signal unit test Framework that testing is complete and generate report.
- void unitTestSetTestType (tyTestType testType)

  Set Test Type for either Positive (default) or Negative testing.
- void unitTestLogPass (void)

  Log a Pass in the Unit test framework.
- void unitTestLogFail (void)

  Log a Fail in the Unit test framework.

7.48.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see: common/license.txt

7.49 VcsHooksApi.h File Reference

```
#include "VcsHooksApiDefines.h"
```

Functions

- void printInt (u32 value)

  Quickly display a single 32 bit unsigned value in the VCS output console.
- void printMsgInt (const char *msg, u32 value)

  Quickly display a msg followed by a 32 bit value.
- void testStateSet (u32 value)

  This function forces the AHB monitor register debug_test_state to a specific value.
- void testStateInc (void)

  This function increments the AHB monitor register debug_test_state to a specific value.
- void testStateAdd (u32 value)

  This function adds a value to the AHB monitor register debug_test_state to a specific value.
- void displayRawMemory (void *address, u32 length)

  This function does a dump to screen of the contents of a section of CMX memory.
- void dumpMemoryToFile (u32 address, u32 length)

  This function does a dump to screen the contents of a CMX memory range.
- void saveMemoryToFile (u32 address, u32 length, const char *fileName)

  This function does a dump to a file the contents of a memory range.
- void loadMemFromFile (char *pFileNameOpt, u32 optIndex, u32 fileOffset, u32 bytesToLoad, void *targetLoadAddress)

  This function loads some of all of a binary file into memory.
- void vcsHookFastMemCpy (void *dst, void *src, u32 length)
- void vcsHookFastMemSet (void *dst, u32 value, u32 length)
- void vcsHookVerilogEventTrigger (u32 eventCode)

  Trigger a verilog event (e.g. Power monitor)
- void vcsFastPuts (char *pString)

  Fast version of puts.
- void vcsHookFunctionCallParam6 (u32 function, u32 param1, u32 param2, u32 param3, u32 param4, u32 param5, u32 param6)

  This function implements a mechanism by which messages can be passed from software to the VCS hardware SOC simulator.

7.49.1 Detailed Description

Copyright

Index

~FeatMaintenance
  FeatMaintenance, 106
  featureMaintenanceApi.h, 153
~OpticalFlow
  OpticalFlow, 126
  opticalFlowApi.h, 192
_SwLink, 91
  allConsIdMask, 91
  lastConsId, 91
  lastConsMon, 91
  pipeRef, 91
  prodId, 92
  prodMon, 92
__attribute
  send_out_tx_buffer_header_t, 139
__attribute__
  PixelPipe, 129
  PixelPipeApi.h, 196

ALIGNED
  FeatMaintenance, 106
  featureMaintenanceApi.h, 153
  MestApi.h, 176
  OpipeDefs.h, 189
  OpticalFlow, 126
  opticalFlowApi.h, 192
  PixelPipe, 129
  PixelPipeApi.h, 197
API, 64
  activeWriteSize, 65
  allMem, 65
  availableLength, 65
  endPtr, 65
  finishMessageRBWrite, 65
  getMessageRBRdPtr, 65
  getMessageRBWrPtr, 65
  InitMessageRB, 65
  rdPtr, 65
  size, 65
  wrPtr, 65
accum
  AeAwbPatchStats, 93
activeWriteSize
  API, 65
aeApi.h, 147
  simpleAEget, 147
  simpleAEinit, 147
  simpleAEpostInit, 147
  simpleAEprocessFrame, 147
  simpleAEstart, 147
AeAwbCfg, 92
  brightThresh, 92
  darkThresh, 92
  firstPatchX, 92
  firstPatchY, 92
  nPatchesX, 92
  nPatchesY, 92
  patchGapX, 92
  patchGapY, 92
  patchHeight, 92
  patchWidth, 93
AeAwbPatchStats, 93
  accum, 93
  altAccum, 93
  count, 93
aeCfg
  OpipeS, 119
aeStats
  OpipeS, 119
AfCfg, 93
  f1Coeffs, 93
  f1Threshold, 93
  f2Coeffs, 93
  f2Threshold, 93
  firstPatchX, 94
  firstPatchY, 94
  initialSubtractionValue, 94
  nPatchesX, 94
  nPatchesY, 94
  patchHeight, 94
  patchWidth, 94
afCfg
  OpipeS, 119
AfPatchStats, 94
  filter1_number_of_used_pixels_green, 94
  filter1_sum_green, 94
  filter1_sum_max_green, 94

filter2_number_of_used_pixels_green, 94
filter2_sum_green, 94
filter2_sum_max_green, 94
sum_all_green, 94
UNDEFINED, 94
afStats
OpipeS, 119
allConsIdMask
_SwLink, 91
allMem
API, 65
alpha
LumaDnsCfg, 109
SharpenCfg, 141
altAccum
AeAwbPatchStats, 93
angle_of_view
LumaDnsRefCfg, 109
availableLength
API, 65

b
BlcCfg, 95
BUSY
Event Loop API, 28
BPP
OpipeDefs.h, 189
base
MBuffer, 110
PlgIspFullStruct, 132
SBuffer, 138
BaseMessagePassingFinalizeChannelRx
World Message Protocol API, 61
BaseMessagePassingInitializePhysicalChannel
World Message Protocol API, 61
BaseMessagePassingRead
World Message Protocol API, 61
bayerPattern
OpipeS, 119
Bicubic Warp API, 18
bicubicWarpGenerateMeshFromLUTMaps, 18
bicubicWarpGenerateMeshHomographyRTP, 18
bicubicWarpGenerateMeshRT, 20
bicubicWarpInit, 20
bicubicWarpProcessFrame, 20
bicubicWarpApi.h, 148
bicubicWarpGenerateMeshFromLUTMaps
Bicubic Warp API, 18
bicubicWarpGenerateMeshHomographyRTP
Bicubic Warp API, 18
bicubicWarpGenerateMeshRT
Bicubic Warp API, 20
bicubicWarpInit
Bicubic Warp API, 20
bicubicWarpProcessFrame
Bicubic Warp API, 20
bitpos
LumaDnsCfg, 109
black01
MipiRxCfg, 113
black23
MipiRxCfg, 113
BlcCfg, 95
b, 95
gb, 95
gr, 95
r, 95
bmVC
World Message Protocol API, 63
Board 182, 21
BoardInitialise, 21
gAppDevHndls, 21
Board182Api.h, 148
BoardInitialise
Board 182, 21
bpp
oMipiTxLoopbackParam, 114
bpw
SPI Slave API, 67
brightThresh
AeAwbCfg, 92
buff_height
client_tx_frame_header_t, 98
buff_pxl_size_denom
client_tx_frame_header_t, 98
buff_pxl_size_nom
client_tx_frame_header_t, 98
buff_stride
client_tx_frame_header_t, 98
buff_width
client_tx_frame_header_t, 98
buffer
SendOutElement_t, 140
TriggerCaptElement, 145
busyFlags
eventQueueItem_t, 104

CDMA_DEF_REQ
OpipeDefs.h, 189
cDbyrIn
PlgIspFullStruct, 132
cDbyrY

PlgIspFullStruct, 132
CLEAN_EXIT
OpipeDefs.h, 189
cLut
PlgIspFullStruct, 132
cSharpY
PlgIspFullStruct, 132
cSigma
PlgIspFullStruct, 132
cUpfirDn
PlgIspFullStruct, 132
cacheData
OpticalFlow, 126
opticalFlowApi.h, 192
cacheInstr
OpticalFlow, 126
opticalFlowApi.h, 192
cachePartData
fmResourceCfg, 106
ofResourceCfg, 113
cachePartInstr
fmResourceCfg, 106
ofResourceCfg, 113
CamGenericApi.h, 149
CamGetFrameCounter
Camera Generic, 22
CamInit
Camera Generic, 22
CamSetupCallbacks
Camera Generic, 24
CamSetupInterrupts
Camera Generic, 24
CamStandby
Camera Generic, 25
CamStart
Camera Generic, 25
CamStop
Camera Generic, 25
CamWakeup
Camera Generic, 26
Camera Generic, 22
CamGetFrameCounter, 22
CamInit, 22
CamSetupCallbacks, 24
CamSetupInterrupts, 24
CamStandby, 25
CamStart, 25
CamStop, 25
CamWakeup, 26
camera_id
client_tx_frame_header_t, 98
cbCfgDynamic
SourceServerCtrlT, 142
cbConfigPlugin
PluginServerCtrl, 135
cbDataWasSent
ipipeServerInfo, 107
cbEndOfFrame
OpipeS, 119
cbIcReset
ipipeServerInfo, 107
cbIcSetup
ipipeServerInfo, 107
cbIcTearDown
ipipeServerInfo, 107
cbIcUserMsg
ipipeServerInfo, 107
cbLineHit
OpipeS, 119
cbList
PlgFifoS, 131
PlgIspFullStruct, 132
cbPreStart
OpipeS, 120
cbSourcesCommit
ipipeServerInfo, 107
cbStartSource
SourceServerCtrlT, 142
cbStopSource
SourceServerCtrlT, 142
ccm
ColCombCfg, 99
ccmOff
ColCombCfg, 99
cdmaReq
OpipeGlobal, 115
OpipeS, 120
cfg
HarrisCfg, 107
MipiRxCfg, 113
OpipeS, 120
cfgMask
OpipeS, 120
ChannelType
World Message Protocol API, 61
check_sum
client_tx_frame_header_t, 98
ChromaDnsCfg, 95
convCoeffCenter, 95
convCoeffCorner, 95
convCoeffEdge, 95
greyCb, 96

greyCg, 96
greyCr, 96
greyDesatOffset, 96
greyDesatSlope, 96
hEnab, 96
limit, 96
th_b, 96
th_g, 96
th_r, 96
ChromaGenCfg, 96
desatOffset, 96
desatSlope, 96
epsilon, 96
kb, 96
kg, 96
kr, 97
lumaCoeffB, 97
lumaCoeffG, 97
lumaCoeffR, 97
pfrStrength, 97
chunk
send_out_tx_buffer_header_t, 139
clampB
RawCfg, 137
clampGb
RawCfg, 137
clampGr
RawCfg, 137
clampR
RawCfg, 137
client_data
send_out_tx_buffer_header_t, 139
client_tx_frame_header_t, 97
buff_height, 98
buff_pxl_size_denom, 98
buff_pxl_size_nom, 98
buff_stride, 98
buff_width, 98
camera_id, 98
check_sum, 98
debug_data_enable, 98
frame_format, 98
frame_height, 98
frame_idx_mipi_rx, 98
frame_idx_process, 98
frame_idx_req_app, 98
frame_idx_req_hal, 98
frame_proc_time_stamp_hi, 98
frame_proc_time_stamp_lo, 98
frame_time_stamp_hi, 98
frame_time_stamp_lo, 98
frame_type, 98
frame_width, 98
header_height, 98
slice_data_type, 98
slice_last_flag, 98
slice_total_number, 98
slice_uv_offset, 98
slice_uv_size, 98
slice_y_offset, 99
slice_y_size, 99
cmx
DBuffer, 101
cmxBuffs
OpipeS, 120
coeffs11
DogCfg, 102
coeffs15
DogCfg, 102
ColCombCfg, 99
ccm, 99
ccmOff, 99
kb, 99
kg, 99
kr, 99
lut3D, 99
ColConvCfg, 99
mat, 99
offset, 99
common/leon/sendOutApi.h
dbgEnableOutput, 203
InternalCbSent, 202
sendOut_initCfg, 203
SendOutCbSent, 202
sendOutControl, 202
sendOutCreate, 202
sendOutFini, 202
sendOutGetBufferHeader, 202
sendOutInit, 202
sendOutSend, 203
conectedToController
PlgSource, 133
config
TriggerCaptElement, 145
context
World Message Protocol API, 63
ConvCfg, 100
mat5x5, 100
convCoeffCenter
ChromaDnsCfg, 95
convCoeffCorner
ChromaDnsCfg, 95

convCoeffEdge
ChromaDnsCfg, 95
cornerNoShaves
t_pPipeShaveConfig, 144
cornerNumShaves
PixelPipe, 130
PixelPipeApi.h, 197
cornerShaveList
t_pPipeShaveConfig, 144
cornerShavesList
PixelPipe, 130
PixelPipeApi.h, 197
count
AeAwbPatchStats, 93
cpLines
MBuffer, 110
CreateColorStripesPattern
Pattern Generator API, 69
CreateHorizontalColorBars
Pattern Generator API, 69
CreateLinearGreyPattern
Pattern Generator API, 70
CreateVerticalColorBars
Pattern Generator API, 70
crtStatus
PlgIspFullStruct, 132
ct
World Message Protocol API, 63
curDsc
MBuffer, 111
curIspCfg
PlgSrcIsp, 135
curLine
PBuffer, 127
currHitLine
OpipeS, 120
curves
LtmCfg, 108

DONE
Event Loop API, 28
DROPPED
Event Loop API, 28
D_DMA
OpipeDefs.h, 189
D_IN
OpipeDefs.h, 189
D_OUT
OpipeDefs.h, 189
DBYR_Y_H
OpipeDefs.h, 189
DBuffer, 100
cmx, 101
ddr, 101
dir, 101
fmt, 101
irqRate, 101
irqRatePow, 101
nPlanes, 101
pipeRef, 101
sippBuffBase, 101
unitID, 101
DEBUG_LOG_LEVEL_LOW
Debug Tracer, 90
DETERMINED_CFG
OpipeDefs.h, 189
DW_ALIGN
OpipeDefs.h, 189
darkThresh
AeAwbCfg, 92
data
eventQueueItem_t, 104
dbgEnableOutput
common/leon/sendOutApi.h, 203
pipe2/common/leon/sendOutApi.h, 204
dbgTracerApi.h, 149
DbyrCfg, 101
dewormGradientMul, 102
dewormOffset, 102
dewormSlope, 102
lumaWeightB, 102
lumaWeightG, 102
lumaWeightR, 102
ddr
DBuffer, 101
Debug Tracer, 90
DEBUG_LOG_LEVEL_LOW, 90
debug_data_enable
client_tx_frame_header_t, 98
defaultMipiTxLoopParams
Opipe.h, 182
deferCb
OpipeDefs.h, 190
desatOffset
ChromaGenCfg, 96
desatSlope
ChromaGenCfg, 96
device
SPI Slave API, 67
dewormGradientMul
DbyrCfg, 102
dewormOffset
DbyrCfg, 102

dewormSlope
DbyrCfg, 102
dir
DBuffer, 101
disparityMapApi.h, 150
sgbm, 150
displayRawMemory
VCS Test Environment API, 84
dmaDsc
MBuffer, 111
DogCfg, 102
coeffs11, 102
coeffs15, 102
sigma11, 102
sigma15, 102
strength, 102
thr, 102
doneFlags
eventQueueItem_t, 104
downshift
PlgSource, 133
PlgSrcIsp, 135
dpcAlphaColdG
RawCfg, 137
dpcAlphaColdRb
RawCfg, 137
dpcAlphaHotG
RawCfg, 137
dpcAlphaHotRb
RawCfg, 137
dpcNoiseLevel
RawCfg, 137
dropFlags
eventQueueItem_t, 104
DrvSetSippClkCtrlRegister
Opipe.h, 183
DrvSetSliceDbyrLumaBuff
Opipe.h, 183
dumpMemoryToFile
VCS Test Environment API, 84

EOF_IDX
PlgSrcIspApi.h, 201
EOF_IDX_SRC
PlgSourceApi.h, 200
ERR_PLGSRCISP_NO_ISP_CFG
PlgSrcIspApi.h, 201
ERR_PLGSRCISP_NO_OUT_BUF
PlgSrcIspApi.h, 201
enMask
OpipeS, 120
endPtr
API, 65
Entry
yn/compShvDynApps/imageWarpDynlib/imageWarp.h, 162
eofEvent
PlgSource, 133
PlgSrcIsp, 135
epsilon
ChromaGenCfg, 96
Event Loop API
BUSY, 28
DONE, 28
DROPPED, 28
REQUIRED, 28
Event Loop API, 27
eventLoopCallback_t, 28
eventLoopChangeEventType, 28
eventLoopClearBusyFlag, 28
eventLoopDropEvent, 29
eventLoopInit, 29
eventLoopPush, 29
eventLoopPushCmd, 29
eventLoopReleaseEvent, 30
eventLoopReset, 30
eventLoopRun, 30
eventLoopSetCallback, 30
eventLoopSetEventState, 30
eventLoopSetReleaseCallback, 32
eventLoopStartTimer, 32
eventState_t, 28
Event Queue, 33
eventLoopTimestamp, 34
eventQueue_t, 33
eventQueueInit, 34
eventQueueItem_t, 33
eventQueuePop, 34
eventQueuePush, 34
eventQueueReturnToParent, 34
eventLoop.h, 150
eventLoopCallback_t
Event Loop API, 28
eventLoopChangeEventType
Event Loop API, 28
eventLoopClearBusyFlag
Event Loop API, 28
eventLoopDropEvent
Event Loop API, 29
eventLoopInit
Event Loop API, 29
eventLoopPush
Event Loop API, 29

eventLoopPushCmd
Event Loop API, 29
eventLoopReleaseEvent
Event Loop API, 30
eventLoopReset
Event Loop API, 30
eventLoopRun
Event Loop API, 30
eventLoopSetCallback
Event Loop API, 30
eventLoopSetEventState
Event Loop API, 30
eventLoopSetReleaseCallback
Event Loop API, 32
eventLoopStartTimer
Event Loop API, 32
eventLoopTimestamp
Event Queue, 34
eventQueue.h, 151
eventQueue_t, 103
Event Queue, 33
head, 103
nItems, 103
tail, 103
eventQueueInit
Event Queue, 34
eventQueueItem_t, 103
busyFlags, 104
data, 104
doneFlags, 104
dropFlags, 104
Event Queue, 33
next, 104
parent, 104
requiredFlags, 105
timestamp, 105
type, 105
eventQueuePop
Event Queue, 34
eventQueuePush
Event Queue, 34
eventQueueReturnToParent
Event Queue, 34
eventState_t
Event Loop API, 28

f1Coeffs
AfCfg, 93
f1Threshold
AfCfg, 93
f2
LumaDnsCfg, 109
f2Coeffs
AfCfg, 93
f2Threshold
AfCfg, 93
FeatMaintenance, 105
∼FeatMaintenance, 106
ALIGNED, 106
FeatMaintenance, 106
featPipeCacheData, 106
featPipeCacheInstr, 106
FeatMaintenance, 106
featureMaintenanceApi.h, 153
fmInit, 106
fmRun, 106
fmShaveNum, 106
g_fmCfg, 106
featPipeCacheData
FeatMaintenance, 106
featureMaintenanceApi.h, 153
featPipeCacheInstr
FeatMaintenance, 106
featureMaintenanceApi.h, 153
featureMaintenanceApi.h, 152
∼FeatMaintenance, 153
ALIGNED, 153
FeatMaintenance, 153
featPipeCacheData, 153
featPipeCacheInstr, 153
fmInit, 153
fmResourceCfg_t, 153
fmRun, 153
fmShaveNum, 153
g_fmCfg, 153
swcShaveUnit_t, 153
fifo1
t_ppFifoCfg, 143
fifo2
t_ppFifoCfg, 143
fifo3
t_ppFifoCfg, 143
fifo4
t_ppFifoCfg, 143
fifoBase
PlgFifoS, 131
fifoCfg
t_pPipeResourceCfg, 143
fifoCommApi.h, 154
fifoCommMasterAddTask, 154
fifoCommMasterRun, 154
fifoCommMasterWaitTask, 156
fifoCommSlaveNotifyTaskCompletion, 156

fifoCommSlaveReadTask, 156
fifoCommSlaveRun, 156
fifoCommInitApi.h, 157
fifoCommMasterInit, 158
fifoCommMasterRegisterTaskType, 158
fifoCommSlaveInit, 158
fifoCommSlaveRegisterTaskType, 158
fifoCommStartMaster, 159
fifoCommStartSlave, 159
fifoCommWaitMaster, 159
fifoCommWaitSlave, 160
fifoCommMasterAddTask
fifoCommApi.h, 154
fifoCommMasterHandler_t
PixelPipeApi.h, 196
fifoCommMasterInit
fifoCommInitApi.h, 158
fifoCommMasterRegisterTaskType
fifoCommInitApi.h, 158
fifoCommMasterRun
fifoCommApi.h, 154
fifoCommMasterWaitTask
fifoCommApi.h, 156
fifoCommSlaveHandler_t
PixelPipeApi.h, 196
fifoCommSlaveInit
fifoCommInitApi.h, 158
fifoCommSlaveNotifyTaskCompletion
fifoCommApi.h, 156
fifoCommSlaveReadTask
fifoCommApi.h, 156
fifoCommSlaveRegisterTaskType
fifoCommInitApi.h, 158
fifoCommSlaveRun
fifoCommApi.h, 156
fifoCommStartMaster
fifoCommInitApi.h, 159
fifoCommStartSlave
fifoCommInitApi.h, 159
fifoCommTask_t
PixelPipeApi.h, 196
fifoCommWaitMaster
fifoCommInitApi.h, 159
fifoCommWaitSlave
fifoCommInitApi.h, 160
fifoLevel
PlgFifoS, 131
fifoList
PlgFifoS, 131
fifoTop
PlgFifoS, 131
filter1_number_of_used_pixels_green
AfPatchStats, 94
filter1_sum_green
AfPatchStats, 94
filter1_sum_max_green
AfPatchStats, 94
filter2_number_of_used_pixels_green
AfPatchStats, 94
filter2_sum_green
AfPatchStats, 94
filter2_sum_max_green
AfPatchStats, 94
finishMessageRBWrite
API, 65
firstPatchX
AeAwbCfg, 92
AfCfg, 94
firstPatchY
AeAwbCfg, 92
AfCfg, 94
flags
OpipeS, 120
fmInit
FeatMaintenance, 106
featureMaintenanceApi.h, 153
fmResourceCfg, 106
cachePartData, 106
cachePartInstr, 106
shaveNum, 106
fmResourceCfg_t
featureMaintenanceApi.h, 153
fmRun
FeatMaintenance, 106
featureMaintenanceApi.h, 153
fmShaveNum
FeatMaintenance, 106
featureMaintenanceApi.h, 153
fmt
DBuffer, 101
format
OpipeS, 120
frame
PlgSource, 133
PlgSrcIsp, 135
frame_format
client_tx_frame_header_t, 98
frame_height
client_tx_frame_header_t, 98
frame_idx_mipi_rx
client_tx_frame_header_t, 98
frame_idx_process

client_tx_frame_header_t, 98
frame_idx_req_app
client_tx_frame_header_t, 98
frame_idx_req_hal
client_tx_frame_header_t, 98
frame_proc_time_stamp_hi
client_tx_frame_header_t, 98
frame_proc_time_stamp_lo
client_tx_frame_header_t, 98
frame_time_stamp_hi
client_tx_frame_header_t, 98
frame_time_stamp_lo
client_tx_frame_header_t, 98
frame_type
client_tx_frame_header_t, 98
frame_width
client_tx_frame_header_t, 98
frameCnt
PlgSource, 133
FrameMgrAcquireFrame
FrameMgrApi.h, 160
FrameMgrAddFrameToPool
FrameMgrApi.h, 160
FrameMgrAddTimeStampHist
FrameMgrApi.h, 160
FrameMgrAndAddTimeStamp
FrameMgrApi.h, 160
FrameMgrApi.h, 160
FrameMgrAcquireFrame, 160
FrameMgrAddFrameToPool, 160
FrameMgrAddTimeStampHist, 160
FrameMgrAndAddTimeStamp, 160
FrameMgrCreatePool, 160
FrameMgrIncreaseNrOfConsumer, 160
FrameMgrLockFrame, 161
FrameMgrProduceFrame, 161
FrameMgrReleaseFrame, 161
FrameMgrUnlockFrame, 161
FrameMgrCreatePool
FrameMgrApi.h, 160
FrameMgrIncreaseNrOfConsumer
FrameMgrApi.h, 160
FrameMgrLockFrame
FrameMgrApi.h, 161
FrameMgrProduceFrame
FrameMgrApi.h, 161
FrameMgrReleaseFrame
FrameMgrApi.h, 161
FrameMgrUnlockFrame
FrameMgrApi.h, 161
frmSz
PlgIspFullStruct, 132
PlgSrcIsp, 135
frmType
SendOutElement_t, 140
fullH
OpipeS, 120
fullW
OpipeS, 120

g_fmCfg
FeatMaintenance, 106
featureMaintenanceApi.h, 153
gAppDevHndls
Board 182, 21
GEN_PREVIEW
OpipeDefs.h, 189
gServerInfo
IpipeServerApi.h, 165
gainB
RawCfg, 137
gainGb
RawCfg, 137
gainGr
RawCfg, 137
gainR
RawCfg, 137
gamma
LumaDnsRefCfg, 109
gaussNoShaves
t_pPipeShaveConfig, 144
gaussNumShaves
PixelPipe, 130
PixelPipeApi.h, 197
gaussShaveList
t_pPipeShaveConfig, 144
gaussShavesList
PixelPipe, 130
PixelPipeApi.h, 197
gb
BlcCfg, 95
getMessageRBRdPtr
API, 65
getMessageRBWrPtr
API, 65
gr
BlcCfg, 95
greyCb
ChromaDnsCfg, 96
greyCg
ChromaDnsCfg, 96
greyCr
ChromaDnsCfg, 96

greyDesatOffset
ChromaDnsCfg, 96
greyDesatSlope
ChromaDnsCfg, 96
grgbImbalDecayBright
RawCfg, 137
grgbImbalDecayDark
RawCfg, 137
grgbImbalPlatBright
RawCfg, 137
grgbImbalPlatDark
RawCfg, 137
grgbImbalThr
RawCfg, 137

h
RectRgn, 138
HIT_IDX
PlgSrcIspApi.h, 201
HIT_IDX_SRC
PlgSourceApi.h, 200
hCoefs
UpfirdnCfg, 145
hD
UpfirdnCfg, 145
hEnab
ChromaDnsCfg, 96
hN
UpfirdnCfg, 145
HarrisCfg, 107
cfg, 107
kValue, 107
hbp
oMipiTxLoopbackParam, 114
hdmiCfg
SendOutInitCfg_t, 140
head
eventQueue_t, 103
header_height
client_tx_frame_header_t, 98
height
MBuffer, 111
OpipeS, 121
PBuffer, 127
SBuffer, 138
height2
OpipeS, 121
hfp
oMipiTxLoopbackParam, 114
histLuma
OpipeS, 121
histRgb
OpipeS, 121
hitEvent
PlgSource, 133
PlgSrcIsp, 135
hostGpioIrq
SPI Slave API, 67
hsync
oMipiTxLoopbackParam, 114

I2C Slave API, 35
I2CSlaveInit, 35
I2CSlaveSetupCallbacks, 36
I2CSlaveApi.h, 161
I2CSlaveInit
I2C Slave API, 35
I2CSlaveSetupCallbacks
I2C Slave API, 36
I_CBUFF_H
OpipeDefs.h, 189
iH
UpfirdnCfg, 145
IMGWARP_cleanup
Image Warp, 37
IMGWARP_start
Image Warp, 37
iPlanes
OpipeS, 121
IRQ_RATE
OpipeDefs.h, 189
IRQ_RATE_POW
OpipeDefs.h, 189
iW
UpfirdnCfg, 146
id
OpipeS, 121
World Message Protocol API, 63
Image Warp, 37
IMGWARP_cleanup, 37
IMGWARP_start, 37
imageWarp
shave/imageWarp.h, 162
imageWarp.h, 161–163
imgAddr
oMipiTxLoopbackParam, 114
imgH
oMipiTxLoopbackParam, 115
imgW
oMipiTxLoopbackParam, 115
inProcessFrame
PlgSource, 133
initEntry
ofResourceCfg, 113

InitMessageRB
API, 65
initPixelPipe
PixelPipe, 129
PixelPipeApi.h, 197
initialSubtractionValue
AfCfg, 94
inputId
PlgFifoElemS, 130
InternalCbSent
common/leon/sendOutApi.h, 202
pipe2/common/leon/sendOutApi.h, 203
ipServerFrameLocked
IpipeServerApi.h, 164
ipServerFrameMgrAddBuffs
IpipeServerApi.h, 164
ipServerFrameMgrCreateList
IpipeServerApi.h, 165
ipServerIspConfigAccepted
IpipeServerApi.h, 165
ipServerIspEnd
IpipeServerApi.h, 165
ipServerIspReportError
IpipeServerApi.h, 165
ipServerIspStart
IpipeServerApi.h, 165
ipServerIspStatsReady
IpipeServerApi.h, 165
ipServerLineHit
IpipeServerApi.h, 165
ipServerQueryAddChild
IpipeServerApi.h, 165
ipServerQueueGet
IpipeServerApi.h, 165
ipServerReadoutEnd
IpipeServerApi.h, 165
ipServerReadoutStart
IpipeServerApi.h, 165
ipServerRegIspQuery
IpipeServerApi.h, 165
ipServerRegOutputQuery
IpipeServerApi.h, 165
ipServerRegSourceQuery
IpipeServerApi.h, 165
ipServerRegUserPlgQuery
IpipeServerApi.h, 165
ipServerSendData
IpipeServerApi.h, 165
ipServerSendUserMsgToLos
IpipeServerApi.h, 165
ipServerSourceReady
IpipeServerApi.h, 165
ipServerSourceStopped
IpipeServerApi.h, 165
ipServerWasReset
IpipeServerApi.h, 165
ipServerWasTornDown
IpipeServerApi.h, 165
IpipeServerApi.h, 163
gServerInfo, 165
ipServerFrameLocked, 164
ipServerFrameMgrAddBuffs, 164
ipServerFrameMgrCreateList, 165
ipServerIspConfigAccepted, 165
ipServerIspEnd, 165
ipServerIspReportError, 165
ipServerIspStart, 165
ipServerIspStatsReady, 165
ipServerLineHit, 165
ipServerQueryAddChild, 165
ipServerQueueGet, 165
ipServerReadoutEnd, 165
ipServerReadoutStart, 165
ipServerRegIspQuery, 165
ipServerRegOutputQuery, 165
ipServerRegSourceQuery, 165
ipServerRegUserPlgQuery, 165
ipServerSendData, 165
ipServerSendUserMsgToLos, 165
ipServerSourceReady, 165
ipServerSourceStopped, 165
ipServerWasReset, 165
ipServerWasTornDown, 165
ipipeServerHandleInputEvent, 164
pSrvIcCtrl, 166
setupIpipeServer, 165
ipipeServerHandleInputEvent
IpipeServerApi.h, 164
ipipeServerInfo, 107
cbDataWasSent, 107
cbIcReset, 107
cbIcSetup, 107
cbIcTearDown, 107
cbIcUserMsg, 107
cbSourcesCommit, 107
memFree, 107
pluginServerCtrl, 107
sourceServerCtrl, 108
trigerCaptQue, 108
irqLevel
SPI Slave API, 67
irqRate

DBuffer, 101
OpipeS, 121
PBuffer, 127
irqRatePow
DBuffer, 101
PBuffer, 127
ispCfg
PlgIspFullStruct, 132

JPEG Encoder API
JPEG_420_PLANAR, 39
JPEG_422_PLANAR, 39
JPEG_444_PLANAR, 39
JPEG Encoder Parallel API
JPEG_420_PLANAR, 42
JPEG_422_PLANAR, 42
JPEG_444_PLANAR, 42
JPEG_420_PLANAR
JPEG Encoder API, 39
JPEG Encoder Parallel API, 42
JPEG_422_PLANAR
JPEG Encoder API, 39
JPEG Encoder Parallel API, 42
JPEG_444_PLANAR
JPEG Encoder API, 39
JPEG Encoder Parallel API, 42
JPEG Encoder API, 38
JPEG_encode, 39
PARTITION_0, 39
PARTITION_1, 39
PARTITION_10, 39
PARTITION_11, 39
PARTITION_2, 39
PARTITION_3, 39
PARTITION_4, 39
PARTITION_5, 39
PARTITION_6, 39
PARTITION_7, 39
PARTITION_8, 39
PARTITION_9, 39
JPEG Encoder Parallel API, 41
JPEG_encode, 42
PARTITION_0, 41
PARTITION_1, 41
PARTITION_10, 41
PARTITION_11, 41
PARTITION_2, 41
PARTITION_3, 41
PARTITION_4, 41
PARTITION_5, 42
PARTITION_6, 42
PARTITION_7, 42
PARTITION_8, 42
PARTITION_9, 42
JPEG_encode
JPEG Encoder API, 39
JPEG Encoder Parallel API, 42
JpegEncoderApi.h, 166

kValue
HarrisCfg, 107
kb
ChromaGenCfg, 96
ColCombCfg, 99
kerSz
UpfirdnCfg, 146
kernelSize
MedianCfg, 111
kg
ChromaGenCfg, 96
ColCombCfg, 99
kr
ChromaGenCfg, 97
ColCombCfg, 99

LCD Generic API, 43
LCDCanQueueFrame, 44
LCDEnYuv422i, 45
LCDInit, 45
LCDInitGLDisable, 45
LCDInitGLEnable, 45
LCDInitLayer, 46
LCDInitVL2Disable, 46
LCDInitVL2Enable, 46
LCDQueueFrame, 46
LCDSetColorTable, 47
LCDSetOutput, 47
LCDSetupCallbacks, 47
LCDStart, 48
LCDStartOneShot, 48
LCDStop, 48
LCDCanQueueFrame
LCD Generic API, 44
LCDEnYuv422i
LCD Generic API, 45
LCDInit
LCD Generic API, 45
LCDInitGLDisable
LCD Generic API, 45
LCDInitGLEnable
LCD Generic API, 45
LCDInitLayer
LCD Generic API, 46
LCDInitVL2Disable

LCD Generic API, 46
LCDInitVL2Enable
LCD Generic API, 46
LCDQueueFrame
LCD Generic API, 46
LCDSetColorTable
LCD Generic API, 47
LCDSetOutput
LCD Generic API, 47
LCDSetupCallbacks
LCD Generic API, 47
LCDStart
LCD Generic API, 48
LCDStartOneShot
LCD Generic API, 48
LCDStop
LCD Generic API, 48
LOST_IRQ
OpipeDefs.h, 189
LP_INTERRUPT_NUM
SuspendLrtLpApi.h, 204
lStride
RectRgn, 138
LUMA_BASED_BLEND
OpipeDefs.h, 189
LUT_H
OpipeDefs.h, 189
lastConsId
_SwLink, 91
lastConsMon
_SwLink, 91
LcdApi.h, 167
Leon IPC API, 50
LeonIPCNumberOfPendingMessages, 50
LeonIPCReadMessage, 50
LeonIPCRxInit, 51
LeonIPCRxReassignSinkThread, 51
LeonIPCRxReleaseSinkThread, 52
LeonIPCSendMessage, 52
LeonIPCTxInit, 52
LeonIPCWaitMessage, 53
Leon L1 Cache, 54
LeonL1CacheDiagCountValidLines, 54
LeonL1CacheDiagDisplay, 55
LeonL1CacheDisplayInfo, 55
LeonL1CacheInitDiagAccess, 55
LeonL1CacheIsAddressDCached, 55
LeonL1CacheIsAddressICached, 55
LeonL1CacheReadCacheDataMem, 56
LeonL1CacheReadCacheTagMem, 56
LeonL1DCacheReadDataWordForAddr, 56
LeonL1DCacheReadTagForAddr, 56
LeonL1ICacheReadDataWordForAddr, 57
LeonL1ICacheReadTagForAddr, 57
LeonIPCApi.h, 168
LeonIPCNumberOfPendingMessages
Leon IPC API, 50
LeonIPCReadMessage
Leon IPC API, 50
LeonIPCRxInit
Leon IPC API, 51
LeonIPCRxReassignSinkThread
Leon IPC API, 51
LeonIPCRxReleaseSinkThread
Leon IPC API, 52
LeonIPCSendMessage
Leon IPC API, 52
LeonIPCTxInit
Leon IPC API, 52
LeonIPCWaitMessage
Leon IPC API, 53
LeonL1CacheApi.h, 169
LeonL1CacheDiagCountValidLines
Leon L1 Cache, 54
LeonL1CacheDiagDisplay
Leon L1 Cache, 55
LeonL1CacheDisplayInfo
Leon L1 Cache, 55
LeonL1CacheInitDiagAccess
Leon L1 Cache, 55
LeonL1CacheIsAddressDCached
Leon L1 Cache, 55
LeonL1CacheIsAddressICached
Leon L1 Cache, 55
LeonL1CacheReadCacheDataMem
Leon L1 Cache, 56
LeonL1CacheReadCacheTagMem
Leon L1 Cache, 56
LeonL1DCacheReadDataWordForAddr
Leon L1 Cache, 56
LeonL1DCacheReadTagForAddr
Leon L1 Cache, 56
LeonL1ICacheReadDataWordForAddr
Leon L1 Cache, 57
LeonL1ICacheReadTagForAddr
Leon L1 Cache, 57
limit
ChromaDnsCfg, 96
lineNo
MBuffer, 111
RectRgn, 138
SBuffer, 138

LineRefs
PlgSourceApi.h, 200
LineRefsSrc
PlgSrcIspApi.h, 201
lineStride
MBuffer, 111
SBuffer, 139
lineW
MBuffer, 111
SBuffer, 139
loadMemFromFile
VCS Test Environment API, 84
localCallback
SendOutElement_t, 140
LscCfg, 108
lscHeight, 108
lscStride, 108
lscWidth, 108
pLscTable, 108
lscHeight
LscCfg, 108
lscStride
LscCfg, 108
lscWidth
LscCfg, 108
LtmCfg, 108
curves, 108
thr, 108
lumaCoeffB
ChromaGenCfg, 97
lumaCoeffG
ChromaGenCfg, 97
lumaCoeffR
ChromaGenCfg, 97
LumaDnsCfg, 108
alpha, 109
bitpos, 109
f2, 109
lut, 109
strength, 109
LumaDnsRefCfg, 109
angle_of_view, 109
gamma, 109
lutDist, 109
lutGamma0_32, 109
lutGamma32_255, 109
shift, 109
lumaWeightB
DbyrCfg, 102
lumaWeightG
DbyrCfg, 102
lumaWeightR
DbyrCfg, 102
lut
LumaDnsCfg, 109
lut3D
ColCombCfg, 99
LutCfg, 110
rangetop, 110
rgnSize, 110
size, 110
table, 110
lutDist
LumaDnsRefCfg, 109
lutGamma0_32
LumaDnsRefCfg, 109
lutGamma32_255
LumaDnsRefCfg, 109

MEM_EXPORT_ALLOCMEM
memManagerApi.h, 171
MEM_INVALID_ADDRESS
memManagerApi.h, 170
MEMALLOC_INVALID_SIZE
memManagerApi.h, 170
MEMALLOC_SIZE_EXCEEDED
memManagerApi.h, 170
MAX_MEMS_NR
MemMgrApi.h, 172
MAX_OF_TASKS
opticalFlowApi.h, 191
MAX_SHAVES_OF
opticalFlowApi.h, 191
MAX_SINKS
OpipeDefs.h, 189
MAX_SOURCES
OpipeDefs.h, 189
MBuffer, 110
base, 110
cpLines, 110
curDsc, 111
dmaDsc, 111
height, 111
lineNo, 111
lineStride, 111
lineW, 111
MDKdox-Components-intro.txt, 170
MESSAGE_RING_BUFFER
API, 64
MEST_COORDS_HEAP
OpticalFlow, 126
opticalFlowApi.h, 192
MEST_GET_RES_BYTES

MestApi.h, 177
MEST_RES_HEAP
OpticalFlow, 126
opticalFlowApi.h, 192
MEST_RES_TMP_HEAP
OpticalFlow, 126
opticalFlowApi.h, 192
mask
MipiRxCfg, 113
masterHandler
PixelPipe, 130
PixelPipeApi.h, 197
mat
ColConvCfg, 99
mat5x5
ConvCfg, 100
MedianCfg, 111
kernelSize, 111
offset, 111
slope, 111
memManagerApi.h
MEM_EXPORT_ALLOCMEM, 171
MEM_INVALID_ADDRESS, 170
MEMALLOC_INVALID_SIZE, 170
MEMALLOC_SIZE_EXCEEDED, 170
NO_ERROR, 170
memFree
ipipeServerInfo, 107
memManagerApi.h, 170
MemMgrAlloc, 171
MemMgrBufferExport, 171
MemMgrErrorCode, 170
MemMgrExportDraw, 171
MemMgrFree, 171
MemMgrAddPool
MemMgrApi.h, 172
MemMgrAlloc
memManagerApi.h, 171
MemMgrApi.h, 172
MemMgrApi.h, 172
MAX_MEMS_NR, 172
MemMgrAddPool, 172
MemMgrAlloc, 172
MemMgrGetFreeMem, 172
MemMgrInit, 172
MemMgrReset, 172
MemMgrResetPool, 172
MemPoolId, 172
NO_VALID_MEM, 172
MemMgrBufferExport
memManagerApi.h, 171
MemMgrErrorCode
memManagerApi.h, 170
MemMgrExportDraw
memManagerApi.h, 171
MemMgrFree
memManagerApi.h, 171
MemMgrGetFreeMem
MemMgrApi.h, 172
MemMgrInit
MemMgrApi.h, 172
MemMgrReset
MemMgrApi.h, 172
MemMgrResetPool
MemMgrApi.h, 172
MemPoolId
MemMgrApi.h, 172
MesasgePassingGetCbPeOver
World Message Protocol API, 61
MesasgePassingGetCbRxDone
World Message Protocol API, 61
MesasgePassingGetCbRxStart
World Message Protocol API, 61
MesasgePassingGetCbTxDone
World Message Protocol API, 61
MesasgePassingGetCbTxStart
World Message Protocol API, 61
MessRingBuff.h, 174
MessagePassingFinalizeChannelRx
World Message Protocol API, 61
MessagePassingFinalizePacketExchange
World Message Protocol API, 61
MessagePassingGetDriverRxBuffer
World Message Protocol API, 62
MessagePassingGetVirtualChannel
World Message Protocol API, 62
MessagePassingInitialize
World Message Protocol API, 62
MessagePassingInitializePhysicalChannel
World Message Protocol API, 62
MessagePassingRead
World Message Protocol API, 62
MessagePassingRegisterVirtualChannel
World Message Protocol API, 62
MessagePassingWrite
World Message Protocol API, 62
MessageProtocol.h, 172
MessageRingBuffer, 112
mestAddROI
MestApi.h, 177
MestApi.h, 174
ALIGNED, 176

mestAddROI, 177
mestDestroy, 177
mestGetResultSize, 177
mestGetRuntimeConfig, 177
mestInit, 177
mestInitStaticConfigVGA, 177
mestMutexDeinit, 177
mestMutexInit, 178
mestROIgetNextId, 178
mestRemoveAllROI, 178
mestRemoveROI, 178
mestRun, 178
mestRunning, 178
mestSetRuntimeConfig, 178
mestUpdateROI, 178
mestUpdateROIFeatures, 178
mestDestroy
MestApi.h, 177
mestGetResultSize
MestApi.h, 177
mestGetRuntimeConfig
MestApi.h, 177
mestHandler_t, 112
privateData, 112
mestInit
MestApi.h, 177
mestInitStaticConfigVGA
MestApi.h, 177
mestMutexDeinit
MestApi.h, 177
mestMutexInit
MestApi.h, 178
mestROIgetNextId
MestApi.h, 178
mestRemoveAllROI
MestApi.h, 178
mestRemoveROI
MestApi.h, 178
mestRun
MestApi.h, 178
mestRunning
MestApi.h, 178
mestSetRuntimeConfig
MestApi.h, 178
mestUpdateROI
MestApi.h, 178
mestUpdateROIFeatures
MestApi.h, 178
minThr
SharpenCfg, 141
mipiCfg
SendOutInitCfg_t, 140
mipiCtrlEofIdx
PlgSource, 134
MipiRxCfg, 112
black01, 113
black23, 113
cfg, 113
mask, 113
sel01, 113
sel23, 113
vbp, 113
xStartWidth, 113
yStartHeight, 113
mipiRxData
PlgSource, 134
PlgSrcIsp, 135
MipiSend/leon/MipiSendApi.h
mipiSendCreate, 180
mipiSendFini, 180
mipiSendInit, 180
mipiSendSentFrame, 180
MipiSendApi.h, 180
mipiSendCreate
MipiSend/leon/MipiSendApi.h, 180
pipe2/MipiSend/leon/MipiSendApi.h, 180
mipiSendFini
MipiSend/leon/MipiSendApi.h, 180
pipe2/MipiSend/leon/MipiSendApi.h, 180
mipiSendInit
MipiSend/leon/MipiSendApi.h, 180
pipe2/MipiSend/leon/MipiSendApi.h, 180
mipiSendSentFrame
MipiSend/leon/MipiSendApi.h, 180
pipe2/MipiSend/leon/MipiSendApi.h, 181

NO_ERROR
memManagerApi.h, 170
N_DESCS
OpipeDefs.h, 189
nCmxBuffs
OpipeS, 121
nItems
eventQueue_t, 103
NO_VALID_MEM
MemMgrApi.h, 172
nOutputPools
PlgFifoS, 131
NPL
OpipeDefs.h, 189
nPatchesX
AeAwbCfg, 92
AfCfg, 94

nPatchesY
AeAwbCfg, 92
AfCfg, 94
nPipes
OpipeGlobal, 115
nPlanes
DBuffer, 101
nSinks
OpipeGlobal, 115
nSnks
OpipeS, 121
nSources
OpipeGlobal, 115
nSrcs
OpipeS, 121
nSwLinks
OpipeGlobal, 115
OpipeS, 121
NUM_SHAVES_SLAVE
PixelPipeApi.h, 196
name
World Message Protocol API, 63
next
eventQueueItem_t, 104
PlgFifoElemS, 130
noiseFloor
SigmaDnsCfg, 141
numShaves
ofResourceCfg, 113
OpticalFlow, 126
opticalFlowApi.h, 192
nxtIspCfg
PlgSrcIsp, 135

O_CBUFF_H
OpipeDefs.h, 189
oCfgMipiTxLoopback
Opipe.h, 183
oH
UpfirdnCfg, 146
oMemCompare
OpipeDefs.h, 190
oMipiTxLoopbackParam, 113
bpp, 114
hbp, 114
hfp, 114
hsync, 114
imgAddr, 114
imgH, 115
imgW, 115
txID, 115
vsync, 115
oPlanes
OpipeS, 122
oStartMipiTxLoopback
Opipe.h, 186
oW
UpfirdnCfg, 146
ofAlgConfig
OpticalFlow, 126
opticalFlowApi.h, 192
ofInit
OpticalFlow, 126
opticalFlowApi.h, 192
ofResourceCfg, 113
cachePartData, 113
cachePartInstr, 113
initEntry, 113
numShaves, 113
runEntry, 113
shaveList, 113
ofResourceCfg_t
opticalFlowApi.h, 191
ofRun
OpticalFlow, 126
opticalFlowApi.h, 192
offX
OpipeS, 122
offY
OpipeS, 122
offset
ColConvCfg, 99
MedianCfg, 111
oldH
OpipeS, 122
oldW
OpipeS, 122
op
PlgIspFullStruct, 132
Opipe, 68
OpipeDefs.h, 190
Opipe.h, 181
defaultMipiTxLoopParams, 182
DrvSetSippClkCtrlRegister, 183
DrvSetSliceDbyrLumaBuff, 183
oCfgMipiTxLoopback, 183
oStartMipiTxLoopback, 186
OpipeCfgBuff, 183
OpipeDelay, 183
OpipeDetCfg, 185
OpipeForceU8fLuma, 185
OpipeInit, 185
OpipeReset, 185

OpipeSetRes, 185
OpipeStart, 185
OpipeSwLink, 185
OpipeTestInit, 186
OpipeWait, 186
OpipeWaitForRawStats, 186
OpipeBlocks.h, 186
opipeCb
OpipeDefs.h, 190
OpipeCfgBuff
Opipe.h, 183
OpipeDefs.h, 187
ALIGNED, 189
BPP, 189
CDMA_DEF_REQ, 189
CLEAN_EXIT, 189
D_DMA, 189
D_IN, 189
D_OUT, 189
DBYR_Y_H, 189
DETERMINED_CFG, 189
DW_ALIGN, 189
deferCb, 190
GEN_PREVIEW, 189
I_CBUFF_H, 189
IRQ_RATE, 189
IRQ_RATE_POW, 189
LOST_IRQ, 189
LUMA_BASED_BLEND, 189
LUT_H, 189
MAX_SINKS, 189
MAX_SOURCES, 189
N_DESCS, 189
NPL, 189
O_CBUFF_H, 189
oMemCompare, 190
Opipe, 190
opipeCb, 190
OpipeDisable, 190
OpipeFinished, 190
PIPE_RUNNING, 189
PREVIEW_ABLE, 189
RDIR, 189
SECTION, 189
SHARP_Y_H, 190
SIPP_FMT_16BIT, 190
SIPP_FMT_32BIT, 190
SIPP_FMT_8BIT, 190
SIPP_FMT_PACK10, 190
SIPP_FMT_PACK12, 190
SIPP_IRQ_LEVEL, 190
SwLink, 190
OpipeDelay
Opipe.h, 183
OpipeDetCfg
Opipe.h, 185
OpipeDisable
OpipeDefs.h, 190
OpipeFinished
OpipeDefs.h, 190
OpipeForceU8fLuma
Opipe.h, 185
OpipeGlobal, 115
cdmaReq, 115
nPipes, 115
nSinks, 115
nSources, 115
nSwLinks, 115
sinks, 116
sources, 116
swCtrl, 116
unitsInUse, 116
OpipeInit
Opipe.h, 185
OpipeReset
Opipe.h, 185
OpipeS, 116
aeCfg, 119
aeStats, 119
afCfg, 119
afStats, 119
bayerPattern, 119
cbEndOfFrame, 119
cbLineHit, 119
cbPreStart, 120
cdmaReq, 120
cfg, 120
cfgMask, 120
cmxBuffs, 120
currHitLine, 120
enMask, 120
flags, 120
format, 120
fullH, 120
fullW, 120
height, 121
height2, 121
histLuma, 121
histRgb, 121
iPlanes, 121
id, 121
irqRate, 121

nCmxBuffs, 121
nSnks, 121
nSrcs, 121
nSwLinks, 121
oPlanes, 122
offX, 122
offY, 122
oldH, 122
oldW, 122
pBlcCfg, 122
pChrGenCfg, 122
pChromaDnsCfg, 122
pColCombCfg, 122
pColConvCfg, 122
pConvCfg, 122
pDbyrCfg, 123
pDogCfg, 123
pHarrisCfg, 123
pLscCfg, 123
pLtmCfg, 123
pLumaDnsCfg, 123
pLumaDnsRefCfg, 123
pLutCfg, 123
pMedCfg, 123
pMipiRxCfg, 123
pRawCfg, 123
pSharpCfg, 124
pSigmaCfg, 124
pUpfirdn0Cfg, 124
pUpfirdn12Cfg, 124
params, 122
rawBits, 124
running, 124
sippSnkIntEnMask, 124
sippSrcIntEnMask, 124
snks, 124
srcs, 124
swCtrl, 124
targetLine, 125
triggerSinkId, 125
unitIDs, 125
width, 125
width2, 125
OpipeSetRes
Opipe.h, 185
OpipeStart
Opipe.h, 185
OpipeSwLink
Opipe.h, 185
OpipeTestInit
Opipe.h, 186
OpipeWait
Opipe.h, 186
OpipeWaitForRawStats
Opipe.h, 186
OpticalFlow, 125
~OpticalFlow, 126
ALIGNED, 126
cacheData, 126
cacheInstr, 126
MEST_COORDS_HEAP, 126
MEST_RES_HEAP, 126
MEST_RES_TMP_HEAP, 126
numShaves, 126
ofAlgConfig, 126
ofInit, 126
ofRun, 126
OpticalFlow, 126
OpticalFlow, 126
opticalFlowApi.h, 192
shaveList, 126
opticalFlowApi.h, 190
~OpticalFlow, 192
ALIGNED, 192
cacheData, 192
cacheInstr, 192
MAX_OF_TASKS, 191
MAX_SHAVES_OF, 191
MEST_COORDS_HEAP, 192
MEST_RES_HEAP, 192
numShaves, 192
ofAlgConfig, 192
ofInit, 192
ofResourceCfg_t, 191
ofRun, 192
OpticalFlow, 192
shaveList, 192
swcShaveUnit_t, 191
OsDrvSpiSlaveCP.h, 192
OsDrvSpiSlaveCPInit
SPI Slave API, 67
OsDrvSpiSlaveCPInitGlobally
SPI Slave API, 67
OsDrvSpiSlaveCPSendPacket
SPI Slave API, 67
OsMesasgePassingGetCbPeOver
World Message Protocol API, 62
OsMesasgePassingGetCbRxDone
World Message Protocol API, 62
OsMesasgePassingGetCbRxStart
World Message Protocol API, 62
OsMesasgePassingGetCbTxDone

World Message Protocol API, 62
OsMesasgePassingGetCbTxStart
World Message Protocol API, 62
OsMessagePassingInitialize
World Message Protocol API, 62
OsMessagePassingInitializePhysicalChannel
World Message Protocol API, 62
OsMessagePassingReadBlockEvent
World Message Protocol API, 62
OsMessagePassingRegisterVirtualChannel
World Message Protocol API, 62
OsMessageProtocol.h, 193
OsVirtualChannel, 127
outId
SendOutElement_t, 140
outputBits
RawCfg, 137
outputPools
PlgFifoS, 131
PlgIspFullStruct, 132
PlgSource, 134
PlgSrcIsp, 135
overshoot
SharpenCfg, 141

PLG_ISPFULL_CREATED
PlgIspFullApi.h, 199
PLG_ISPFULL_INUSE
PlgIspFullApi.h, 199
PLG_ISPFULL_NOTMADE
PlgIspFullApi.h, 199
PLG_SOURCE_CREATED
PlgSourceApi.h, 200
PLG_SOURCE_INUSE
PlgSourceApi.h, 200
PLG_SOURCE_NOTMADE
PlgSourceApi.h, 200
PLG_SRC_ISP_CREATED
PlgSrcIspApi.h, 201
PLG_SRC_ISP_INUSE
PlgSrcIspApi.h, 201
PLG_SRC_ISP_NOTMADE
PlgSrcIspApi.h, 201
PARTITION_0
JPEG Encoder API, 39
JPEG Encoder Parallel API, 41
PARTITION_1
JPEG Encoder API, 39
JPEG Encoder Parallel API, 41
PARTITION_10
JPEG Encoder API, 39
JPEG Encoder Parallel API, 41
PARTITION_11
JPEG Encoder API, 39
JPEG Encoder Parallel API, 41
PARTITION_2
JPEG Encoder API, 39
JPEG Encoder Parallel API, 41
PARTITION_3
JPEG Encoder API, 39
JPEG Encoder Parallel API, 41
PARTITION_4
JPEG Encoder API, 39
JPEG Encoder Parallel API, 41
PARTITION_5
JPEG Encoder API, 39
JPEG Encoder Parallel API, 42
PARTITION_6
JPEG Encoder API, 39
JPEG Encoder Parallel API, 42
PARTITION_7
JPEG Encoder API, 39
JPEG Encoder Parallel API, 42
PARTITION_8
JPEG Encoder API, 39
JPEG Encoder Parallel API, 42
PARTITION_9
JPEG Encoder API, 39
JPEG Encoder Parallel API, 42
pBlcCfg
OpipeS, 122
PBuffer, 127
curLine, 127
height, 127
irqRate, 127
irqRatePow, 127
pChrGenCfg
OpipeS, 122
pChromaDnsCfg
OpipeS, 122
pColCombCfg
OpipeS, 122
pColConvCfg
OpipeS, 122
pConvCfg
OpipeS, 122
pDbyrCfg
OpipeS, 123
pDogCfg
OpipeS, 123
pHarrisCfg
OpipeS, 123
PIPE_RUNNING

OpipeDefs.h, 189
PLGFIFO_SZ
PlgFifoApi.h, 198
pLscCfg
OpipeS, 123
pLscTable
LscCfg, 108
pLtmCfg
OpipeS, 123
pLumaDnsCfg
OpipeS, 123
pLumaDnsRefCfg
OpipeS, 123
pLutCfg
OpipeS, 123
pMedCfg
OpipeS, 123
pMipiRxCfg
OpipeS, 123
PREVIEW_ABLE
OpipeDefs.h, 189
pRawCfg
OpipeS, 123
pRx
PlgSource, 134
pRxIsp
PlgSrcIsp, 135
pRxSkip
PlgSrcIsp, 135
pSharpCfg
OpipeS, 124
pSigmaCfg
OpipeS, 124
pSrvIcCtrl
IpipeServerApi.h, 166
pStride
RectRgn, 138
pUpfirdn0Cfg
OpipeS, 124
pUpfirdn12Cfg
OpipeS, 124
PacketExchangeOverCallback_t
World Message Protocol API, 61
PacketReadDoneCallback_t
World Message Protocol API, 61
PacketReadRequestCallback_t
World Message Protocol API, 61
PacketWriteDoneCallback_t
World Message Protocol API, 61
PacketWriteRequestCallback_t
World Message Protocol API, 61
params
OpipeS, 122
parent
eventQueueItem_t, 104
patchGapX
AeAwbCfg, 92
patchGapY
AeAwbCfg, 92
patchHeight
AeAwbCfg, 92
AfCfg, 94
patchWidth
AeAwbCfg, 93
AfCfg, 94
Pattern Generator API, 69
CreateColorStripesPattern, 69
CreateHorizontalColorBars, 69
CreateLinearGreyPattern, 70
CreateVerticalColorBars, 70
PatternCheck, 70
PatternCheck
Pattern Generator API, 70
PatternGeneratorApi.h, 194
pfrStrength
ChromaGenCfg, 97
phyChannel
World Message Protocol API, 63
PhysicalChannel, 127
pipe2/MipiSend/leon/MipiSendApi.h
mipiSendCreate, 180
mipiSendFini, 180
mipiSendInit, 180
mipiSendSentFrame, 181
pipe2/common/leon/sendOutApi.h
dbgEnableOutput, 204
InternalCbSent, 203
sendOut_initCfg, 204
SendOutCbSent, 203
sendOutControl, 203
sendOutCreate, 204
sendOutFini, 204
sendOutGetBufferHeader, 204
sendOutInit, 204
sendOutSend, 204
pipeRef
_SwLink, 91
DBuffer, 101
PixelPipe, 128
__attribute__, 129
ALIGNED, 129
cornerNumShaves, 130

cornerShavesList, 130
gaussNumShaves, 130
gaussShavesList, 130
initPixelPipe, 129
masterHandler, 130
PixelPipe, 129
pixelPipe, 129
pixelPipeParams, 130
PixelPipe, 129
PixelPipeApi.h, 197
ppCacheData, 130
ppCacheInstr, 130
ppInit, 129
ppMasterShaveNum, 130
ppRun, 130
shaveTaskTypes, 130
slaveHandler, 130
slaveNumShaves, 130
slaveShaves, 130
taskTypes, 130
pixelPipe
PixelPipe, 129
PixelPipeApi.h, 197
PixelPipeApi.h, 195
__attribute__, 196
ALIGNED, 197
cornerNumShaves, 197
cornerShavesList, 197
fifoCommMasterHandler_t, 196
fifoCommSlaveHandler_t, 196
fifoCommTask_t, 196
gaussNumShaves, 197
gaussShavesList, 197
initPixelPipe, 197
masterHandler, 197
NUM_SHAVES_SLAVE, 196
PixelPipe, 197
pixelPipe, 197
pixelPipeParams, 197
ppCacheData, 197
ppCacheInstr, 197
ppInit, 197
ppMasterShaveNum, 197
ppRun, 197
shaveTaskTypes, 197
slaveHandler, 197
slaveNumShaves, 197
slaveShaves, 197
swcShaveUnit_t, 196
taskTypes, 197
UpdateCellThresholds, 197
pixelPipeParams
PixelPipe, 130
PixelPipeApi.h, 197
plg
PlgFifoS, 131
PlgIspFullStruct, 132
PlgSource, 134
PlgSrcIsp, 135
PlgIspFullApi.h
PLG_ISPFULL_CREATED, 199
PLG_ISPFULL_INUSE, 199
PLG_ISPFULL_NOTMADE, 199
PlgSourceApi.h
EOF_IDX_SRC, 200
HIT_IDX_SRC, 200
PLG_SOURCE_CREATED, 200
PLG_SOURCE_INUSE, 200
PLG_SOURCE_NOTMADE, 200
SOF_IDX_SRC, 200
PlgSrcIspApi.h
EOF_IDX, 201
ERR_PLGSRCISP_NO_ISP_CFG, 201
ERR_PLGSRCISP_NO_OUT_BUF, 201
HIT_IDX, 201
PLG_SRC_ISP_CREATED, 201
PLG_SRC_ISP_INUSE, 201
PLG_SRC_ISP_NOTMADE, 201
SOF_IDX, 201
PlgFifo
PlgFifoApi.h, 198
PlgFifoApi.h, 198
PLGFIFO_SZ, 198
PlgFifo, 198
PlgFifoConfig, 198
PlgFifoCreate, 198
PlgFifoElem, 198
PlgFifoConfig
PlgFifoApi.h, 198
PlgFifoCreate
PlgFifoApi.h, 198
PlgFifoElem
PlgFifoApi.h, 198
PlgFifoElemS, 130
inputId, 130
next, 130
value, 131
PlgFifoS, 131
cbList, 131
fifoBase, 131
fifoLevel, 131
fifoList, 131

fifoTop, 131
nOutputPools, 131
outputPools, 131
plg, 131
triger, 131
PlgIspFull
PlgIspFullApi.h, 199
PlgIspFullApi.h, 198
PlgIspFull, 199
PlgIspFullConfig, 199
PlgIspFullCreate, 199
PlgIspFullStatus, 199
PlgIspFullConfig
PlgIspFullApi.h, 199
PlgIspFullCreate
PlgIspFullApi.h, 199
PlgIspFullStatus
PlgIspFullApi.h, 199
PlgIspFullStruct, 131
base, 132
cDbyrIn, 132
cDbyrY, 132
cLut, 132
cSharpY, 132
cSigma, 132
cUpfirDn, 132
cbList, 132
crtStatus, 132
frmSz, 132
ispCfg, 132
op, 132
outputPools, 132
plg, 132
procesEnd, 132
procesIspError, 132
procesStart, 133
scale, 133
status, 133
PlgSource, 133
conectedToController, 133
downshift, 133
eofEvent, 133
frame, 133
frameCnt, 133
hitEvent, 133
inProcessFrame, 133
mipiCtrlEofIdx, 134
mipiRxData, 134
outputPools, 134
pRx, 134
plg, 134
producedFrame, 134
receiverEofIdx, 134
rxCfg, 134
sofEvent, 134
srcId, 134
status, 134
PlgSourceApi.h, 199
LineRefs, 200
PlgSourceCmxAlloc, 200
PlgSourceCreate, 200
PlgSourceSetLineHit, 200
PlgSourceStart, 200
PlgSourceStatus, 200
PlgSourceCmxAlloc
PlgSourceApi.h, 200
PlgSourceCreate
PlgSourceApi.h, 200
PlgSourceSetLineHit
PlgSourceApi.h, 200
PlgSourceStart
PlgSourceApi.h, 200
PlgSourceStatus
PlgSourceApi.h, 200
PlgSrcIsp, 134
curIspCfg, 135
downshift, 135
eofEvent, 135
frame, 135
frmSz, 135
hitEvent, 135
mipiRxData, 135
nxtIspCfg, 135
outputPools, 135
pRxIsp, 135
pRxSkip, 135
plg, 135
procesIspError, 135
rxCfg, 135
scale, 135
sofEvent, 135
srcId, 135
status, 135
PlgSrcIspApi.h, 200
LineRefsSrc, 201
PlgSrcIspCmxAlloc, 201
PlgSrcIspCreate, 201
PlgSrcIspErrors, 201
PlgSrcIspSetLineHit, 201
PlgSrcIspStart, 201
PlgSrcIspStatus, 201
PlgSrcIspCmxAlloc

PlgSrcIspApi.h, 201
PlgSrcIspCreate
PlgSrcIspApi.h, 201
PlgSrcIspErrors
PlgSrcIspApi.h, 201
PlgSrcIspSetLineHit
PlgSrcIspApi.h, 201
PlgSrcIspStart
PlgSrcIspApi.h, 201
PlgSrcIspStatus
PlgSrcIspApi.h, 201
PluginServerCtrl, 135
cbConfigPlugin, 135
pluginServerCtrl
ipipeServerInfo, 107
pool
SourceServerCtrlT, 142
ppBuffs
t_pPipeResourceCfg, 143
ppBuffsSize
t_pPipeResourceCfg, 143
ppCacheData
PixelPipe, 130
PixelPipeApi.h, 197
t_pPipeResourceCfg, 143
ppCacheInstr
PixelPipe, 130
PixelPipeApi.h, 197
t_pPipeResourceCfg, 144
ppCmxBuffs
t_pPipeResourceCfg, 144
ppCmxBuffsSize
t_pPipeResourceCfg, 144
ppInit
PixelPipe, 129
PixelPipeApi.h, 197
ppMasterShaveNum
PixelPipe, 130
PixelPipeApi.h, 197
ppRun
PixelPipe, 130
PixelPipeApi.h, 197
ppShaveNum
t_pPipeShaveConfig, 144
ppThresholds_t, 136
thresholdDecreaseVelocity, 136
thresholdIncreaseVelocity, 136
thresholdMax, 136
thresholdMin, 136
printInt
VCS Test Environment API, 85
printMsgInt
VCS Test Environment API, 85
priority_level
World Message Protocol API, 63
privateData
mestHandler_t, 112
procesEnd
PlgIspFullStruct, 132
procesIspError
PlgIspFullStruct, 132
PlgSrcIsp, 135
procesStart
PlgIspFullStruct, 133
prodId
_SwLink, 92
prodMon
_SwLink, 92
producedFrame
PlgSource, 134

queue
TrigerCaptQue, 144
queueIn
TrigerCaptQue, 144
queueOut
TrigerCaptQue, 145

r
BlcCfg, 95
REQUIRED
Event Loop API, 28
RDIR
OpipeDefs.h, 189
rangeStop0
SharpenCfg, 141
rangeStop1
SharpenCfg, 141
rangeStop2
SharpenCfg, 141
rangeStop3
SharpenCfg, 141
rangetop
LutCfg, 110
rawBits
OpipeS, 124
RawCfg, 136
clampB, 137
clampGb, 137
clampGr, 137
clampR, 137
dpcAlphaColdG, 137
dpcAlphaColdRb, 137

dpcAlphaHotG, 137
dpcAlphaHotRb, 137
dpcNoiseLevel, 137
gainB, 137
gainGb, 137
gainGr, 137
gainR, 137
grgbImbalDecayBright, 137
grgbImbalDecayDark, 137
grgbImbalPlatBright, 137
grgbImbalPlatDark, 137
grgbImbalThr, 137
outputBits, 137
rdPtr
API, 65
receiverEofIdx
PlgSource, 134
RectRgn, 137
h, 138
lStride, 138
lineNo, 138
pStride, 138
tlcX, 138
tlcY, 138
w, 138
requiredFlags
eventQueueItem_t, 105
restoreStackAndExitLp_asm
SuspendLrtLpApi.h, 205
rgnSize
LutCfg, 110
runEntry
ofResourceCfg, 113
running
OpipeS, 124
rxCfg
PlgSource, 134
PlgSrcIsp, 135
rxFifo
World Message Protocol API, 63
rxWaitTaskId
World Message Protocol API, 63

SOF_IDX
PlgSrcIspApi.h, 201
SOF_IDX_SRC
PlgSourceApi.h, 200
SPISLAVE
World Message Protocol API, 61
SBuffer, 138
base, 138
height, 138
lineNo, 138
lineStride, 139
lineW, 139
SECTION
OpipeDefs.h, 189
SHARP_Y_H
OpipeDefs.h, 190
SIPP_FMT_16BIT
OpipeDefs.h, 190
SIPP_FMT_32BIT
OpipeDefs.h, 190
SIPP_FMT_8BIT
OpipeDefs.h, 190
SIPP_FMT_PACK10
OpipeDefs.h, 190
SIPP_FMT_PACK12
OpipeDefs.h, 190
SIPP_IRQ_LEVEL
OpipeDefs.h, 190
SPI Slave API, 66
bpw, 67
device, 67
hostGpioIrq, 67
irqLevel, 67
OsDrvSpiSlaveCPInit, 67
OsDrvSpiSlaveCPInitGlobally, 67
OsDrvSpiSlaveCPSendPacket, 67
scpha, 67
scpol, 67
spiConfig, 67
useDma, 67
saveMemoryToFile
VCS Test Environment API, 86
scale
PlgIspFullStruct, 133
PlgSrcIsp, 135
scpha
SPI Slave API, 67
scpol
SPI Slave API, 67
sel01
MipiRxCfg, 113
sel23
MipiRxCfg, 113
send_out_tx_buffer_header_t, 139
__attribute, 139
chunk, 139
client_data, 139
sendOut_initCfg
common/leon/sendOutApi.h, 203
pipe2/common/leon/sendOutApi.h, 204

sendOutApi.h, 202, 203
SendOutCbSent
    common/leon/sendOutApi.h, 202
    pipe2/common/leon/sendOutApi.h, 203
sendOutCbSent
    SendOutElement_t, 140
sendOutControl
    common/leon/sendOutApi.h, 202
    pipe2/common/leon/sendOutApi.h, 203
sendOutCreate
    common/leon/sendOutApi.h, 202
    pipe2/common/leon/sendOutApi.h, 204
SendOutElement_t, 139
    buffer, 140
    frmType, 140
    localCallback, 140
    outId, 140
    sendOutCbSent, 140
sendOutFini
    common/leon/sendOutApi.h, 202
    pipe2/common/leon/sendOutApi.h, 204
sendOutGetBufferHeader
    common/leon/sendOutApi.h, 202
    pipe2/common/leon/sendOutApi.h, 204
sendOutInit
    common/leon/sendOutApi.h, 202
    pipe2/common/leon/sendOutApi.h, 204
SendOutInitCfg_t, 140
    hdmiCfg, 140
    mipiCfg, 140
    usbCfg, 140
sendOutSend
    common/leon/sendOutApi.h, 203
    pipe2/common/leon/sendOutApi.h, 204
setupIpipeServer
    IpipeServerApi.h, 165
sgbm
    disparityMapApi.h, 150
SharpenCfg, 140
    alpha, 141
    minThr, 141
    overshoot, 141
    rangeStop0, 141
    rangeStop1, 141
    rangeStop2, 141
    rangeStop3, 141
    sharpenCoeffs, 141
    sigma, 141
    strengthDarken, 141
    strengthLighten, 141
    undershoot, 141
sharpenCoeffs
    SharpenCfg, 141
shave/imageWarp.h
    imageWarp, 162
shaveConfig
    t_pPipeResourceCfg, 144
shaveGetFieldValue
    testUtilsApi.h, 205
shaveList
    ofResourceCfg, 113
    OpticalFlow, 126
    opticalFlowApi.h, 192
shaveNum
    fmResourceCfg, 106
shaveProfileInit
    testUtilsApi.h, 206
shaveProfilePrint
    testUtilsApi.h, 206
shaveProfileReset
    testUtilsApi.h, 206
shaveProfileStart
    testUtilsApi.h, 206
shaveShowCounterValues
    testUtilsApi.h, 208
shaveTaskTypes
    PixelPipe, 130
    PixelPipeApi.h, 197
shift
    LumaDnsRefCfg, 109
sigma
    SharpenCfg, 141
sigma11
    DogCfg, 102
sigma15
    DogCfg, 102
SigmaDnsCfg, 141
    noiseFloor, 141
    thresh1P0, 141
    thresh1P1, 141
    thresh1P2, 141
    thresh1P3, 142
    thresh2P0, 142
    thresh2P1, 142
    thresh2P2, 142
    thresh2P3, 142
simpleAEget
    aeApi.h, 147
simpleAEinit
    aeApi.h, 147
simpleAEpostInit
    aeApi.h, 147

simpleAEprocessFrame
aeApi.h, 147
simpleAEstart
aeApi.h, 147
sinks
OpipeGlobal, 116
sippBuffBase
DBuffer, 101
sippSnkIntEnMask
OpipeS, 124
sippSrcIntEnMask
OpipeS, 124
size
API, 65
LutCfg, 110
slaveHandler
PixelPipe, 130
PixelPipeApi.h, 197
slaveNumShaves
PixelPipe, 130
PixelPipeApi.h, 197
slaveShaves
PixelPipe, 130
PixelPipeApi.h, 197
slice_data_type
client_tx_frame_header_t, 98
slice_last_flag
client_tx_frame_header_t, 98
slice_total_number
client_tx_frame_header_t, 98
slice_uv_offset
client_tx_frame_header_t, 98
slice_uv_size
client_tx_frame_header_t, 98
slice_y_offset
client_tx_frame_header_t, 99
slice_y_size
client_tx_frame_header_t, 99
slope
MedianCfg, 111
snks
OpipeS, 124
sofEvent
PlgSource, 134
PlgSrcIsp, 135
source
TriggerCaptElement, 145
sourceDescription
SourceServerCtrlT, 142
sourceServerCtrl
ipipeServerInfo, 108
SourceServerCtrlT, 142
cbCfgDynamic, 142
cbStartSource, 142
cbStopSource, 142
pool, 142
sourceDescription, 142
sources
OpipeGlobal, 116
spiConfig
SPI Slave API, 67
spiSlaveCommunicationConfiguration_t, 142
srcId
PlgSource, 134
PlgSrcIsp, 135
srcs
OpipeS, 124
status
PlgIspFullStruct, 133
PlgSource, 134
PlgSrcIsp, 135
strength
DogCfg, 102
LumaDnsCfg, 109
strengthDarken
SharpenCfg, 141
strengthLighten
SharpenCfg, 141
sum_all_green
AfPatchStats, 94
SuspendLrt
SuspendLrtLpApi.h, 205
SuspendLrtLpApi.h, 204
LP_INTERRUPT_NUM, 204
restoreStackAndExitLp_asm, 205
SuspendLrt, 205
switchStackAndEntersLp_asm, 205
swCtrl
OpipeGlobal, 116
OpipeS, 124
SwLink
OpipeDefs.h, 190
swcShaveUnit_t
featureMaintenanceApi.h, 153
opticalFlowApi.h, 191
PixelPipeApi.h, 196
switchStackAndEntersLp_asm
SuspendLrtLpApi.h, 205

TX_IDLE
World Message Protocol API, 61
TX_PENDING
World Message Protocol API, 61

t_pPipeResourceCfg, 143
fifoCfg, 143
ppBuffs, 143
ppBuffsSize, 143
ppCacheData, 143
ppCacheInstr, 144
ppCmxBuffs, 144
ppCmxBuffsSize, 144
shaveConfig, 144
t_pPipeShaveConfig, 144
cornerNoShaves, 144
cornerShaveList, 144
gaussNoShaves, 144
gaussShaveList, 144
ppShaveNum, 144
t_ppFifoCfg, 143
fifo1, 143
fifo2, 143
fifo3, 143
fifo4, 143
table
LutCfg, 110
tail
eventQueue_t, 103
targetLine
OpipeS, 125
taskTypes
PixelPipe, 130
PixelPipeApi.h, 197
testStateAdd
VCS Test Environment API, 86
testStateInc
VCS Test Environment API, 86
testStateSet
VCS Test Environment API, 87
testUtilsApi.h, 205
shaveGetFieldValue, 205
shaveProfileInit, 206
shaveProfilePrint, 206
shaveProfileReset, 206
shaveProfileStart, 206
shaveShowCounterValues, 208
th_b
ChromaDnsCfg, 96
th_g
ChromaDnsCfg, 96
th_r
ChromaDnsCfg, 96
thr
DogCfg, 102
LtmCfg, 108
thresh1P0
SigmaDnsCfg, 141
thresh1P1
SigmaDnsCfg, 141
thresh1P2
SigmaDnsCfg, 141
thresh1P3
SigmaDnsCfg, 142
thresh2P0
SigmaDnsCfg, 142
thresh2P1
SigmaDnsCfg, 142
thresh2P2
SigmaDnsCfg, 142
thresh2P3
SigmaDnsCfg, 142
thresholdDecreaseVelocity
ppThresholds_t, 136
thresholdIncreaseVelocity
ppThresholds_t, 136
thresholdMax
ppThresholds_t, 136
thresholdMin
ppThresholds_t, 136
timestamp
eventQueueItem_t, 105
tlcX
RectRgn, 138
tlcY
RectRgn, 138
triger
PlgFifoS, 131
TrigerCaptQue, 144
queue, 144
queueIn, 144
queueOut, 145
trigerCaptQue
ipipeServerInfo, 108
TriggerCaptElement, 145
buffer, 145
config, 145
source, 145
triggerSinkId
OpipeS, 125
txFifo
World Message Protocol API, 63
txID
oMipiTxLoopbackParam, 115
txPending_t
World Message Protocol API, 61
type

eventQueueItem_t, 105

UNDEFINED
AfPatchStats, 94
undershoot
SharpenCfg, 141
Unit Test API, 71
unitTestAssert, 71
unitTestCheckSectionFail, 72
unitTestCompare, 72
unitTestExecutionWithinRange, 72
unitTestFinalReport, 73
unitTestFloatAbsRangeCheck, 73
unitTestFloatWithinRange, 73
unitTestInit, 74
unitTestLogFail, 74
unitTestLogPass, 74
unitTestLogResults, 74
unitTestMemCompare, 74
unitTestMemCompareDeltaU8, 75
unitID
DBuffer, 101
unitIDs
OpipeS, 125
UnitTestApi.h, 208, 209
unitTestAssert
Unit Test API, 71
VCS Unit Test API, 77
unitTestCheckSectionFail
Unit Test API, 72
unitTestCompare
Unit Test API, 72
VCS Unit Test API, 77
unitTestCompare64
VCS Unit Test API, 77
unitTestCrcCheck
VCS Unit Test API, 77
unitTestExecutionWithinRange
Unit Test API, 72
VCS Unit Test API, 78
unitTestFinalReport
Unit Test API, 73
VCS Unit Test API, 78
unitTestFloatAbsRangeCheck
Unit Test API, 73
unitTestFloatWithinRange
Unit Test API, 73
unitTestInit
Unit Test API, 74
VCS Unit Test API, 78
unitTestLogFail
Unit Test API, 74
VCS Unit Test API, 79
unitTestLogPass
Unit Test API, 74
VCS Unit Test API, 79
unitTestLogResults
Unit Test API, 74
unitTestMemCompare
Unit Test API, 74
VCS Unit Test API, 79
unitTestMemCompareDeltaU8
Unit Test API, 75
unitTestReadBitCheck
VCS Unit Test API, 80
unitTestReadByteCheck
VCS Unit Test API, 80
unitTestReadDWordCheck
VCS Unit Test API, 80
unitTestReadHalfCheck
VCS Unit Test API, 81
unitTestReadWordCheck
VCS Unit Test API, 81
unitTestSetTestType
VCS Unit Test API, 81
unitTestVerbosity
VCS Unit Test API, 82
unitsInUse
OpipeGlobal, 116
UpdateCellThresholds
PixelPipeApi.h, 197
UpfirdnCfg, 145
hCoefs, 145
hD, 145
hN, 145
iH, 145
iW, 146
kerSz, 146
oH, 146
oW, 146
vCoefs, 146
vD, 146
vN, 146
usbCfg
SendOutInitCfg_t, 140
useDma
SPI Slave API, 67

VCS Test Environment API, 83
displayRawMemory, 84
dumpMemoryToFile, 84
loadMemFromFile, 84
printInt, 85
printMsgInt, 85

saveMemoryToFile, 86
testStateAdd, 86
testStateInc, 86
testStateSet, 87
vcsFastPuts, 87
vcsHookFastMemCpy, 87
vcsHookFastMemSet, 88
vcsHookFunctionCallParam6, 88
vcsHookVerilogEventTrigger, 89
VCS Unit Test API, 76
unitTestAssert, 77
unitTestCompare, 77
unitTestCompare64, 77
unitTestCrcCheck, 77
unitTestExecutionWithinRange, 78
unitTestFinalReport, 78
unitTestInit, 78
unitTestLogFail, 79
unitTestLogPass, 79
unitTestMemCompare, 79
unitTestReadBitCheck, 80
unitTestReadByteCheck, 80
unitTestReadDWordCheck, 80
unitTestReadHalfCheck, 81
unitTestReadWordCheck, 81
unitTestSetTestType, 81
unitTestVerbosity, 82
vCoefs
UpfirdnCfg, 146
vD
UpfirdnCfg, 146
vN
UpfirdnCfg, 146
value
PlgFifoElemS, 131
vbp
MipiRxCfg, 113
vcsFastPuts
VCS Test Environment API, 87
vcsHookFastMemCpy
VCS Test Environment API, 87
vcsHookFastMemSet
VCS Test Environment API, 88
vcsHookFunctionCallParam6
VCS Test Environment API, 88
vcsHookVerilogEventTrigger
VCS Test Environment API, 89
VcsHooksApi.h, 210
virtual_channel_close
World Message Protocol API, 62
virtual_channel_control
World Message Protocol API, 62
virtual_channel_initialize
World Message Protocol API, 62
virtual_channel_open
World Message Protocol API, 62
virtual_channel_read
World Message Protocol API, 62
virtual_channel_write
World Message Protocol API, 62
VirtualChannel, 146
vsync
oMipiTxLoopbackParam, 115

w
RectRgn, 138
width
OpipeS, 125
width2
OpipeS, 125
World Message Protocol API
SPISLAVE, 61
TX_IDLE, 61
TX_PENDING, 61
World Message Protocol API, 58
BaseMessagePassingFinalizeChannelRx, 61
BaseMessagePassingInitializePhysicalChannel, 61
BaseMessagePassingRead, 61
bmVC, 63
ChannelType, 61
context, 63
ct, 63
id, 63
MesagePassingGetCbPeOver, 61
MesagePassingGetCbRxDone, 61
MesagePassingGetCbRxStart, 61
MesagePassingGetCbTxDone, 61
MesagePassingGetCbTxStart, 61
MessagePassingFinalizeChannelRx, 61
MessagePassingFinalizePacketExchange, 61
MessagePassingGetDriverRxBuffer, 62
MessagePassingGetVirtualChannel, 62
MessagePassingInitialize, 62
MessagePassingInitializePhysicalChannel, 62
MessagePassingRead, 62
MessagePassingRegisterVirtualChannel, 62
MessagePassingWrite, 62
name, 63
OsMesagePassingGetCbPeOver, 62
OsMesagePassingGetCbRxDone, 62
OsMesagePassingGetCbRxStart, 62
OsMesagePassingGetCbTxDone, 62

OsMesasgePassingGetCbTxStart, 62
OsMessagePassingInitialize, 62
OsMessagePassingInitializePhysicalChannel, 62
OsMessagePassingReadBlockEvent, 62
OsMessagePassingRegisterVirtualChannel, 62
PacketExchangeOverCallback_t, 61
PacketReadDoneCallback_t, 61
PacketReadRequestCallback_t, 61
PacketWriteDoneCallback_t, 61
PacketWriteRequestCallback_t, 61
phyChannel, 63
priority_level, 63
rxFifo, 63
rxWaitTaskId, 63
txFifo, 63
txPending_t, 61
virtual_channel_close, 62
virtual_channel_control, 62
virtual_channel_initialize, 62
virtual_channel_open, 62
virtual_channel_read, 62
virtual_channel_write, 62
wrPtr
API, 65

xStartWidth
MipiRxCfg, 113

yStartHeight
MipiRxCfg, 113
yn/compShvDynApps/imageWarpDynlib/imageWarp.h
Entry, 162