The Fastest Verification

ZeBu ™

MIPI DSI transactor API

1.1

Fri Mar 1 2013 17:22:51

Copyright Notice

Proprietary Information

Right to Copy Documentation

The license agreement with EVE permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:
This document is duplicated with the permission of EVE, for the exclusive use of _____ and its employees. This is copy number ____.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader s responsibility to determine the applicable regulations and to comply with them.

Disclaimer

EVE AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 ZEBU_IP Namespace Reference

Namespaces

- namespace MIPI_DSI

4.2 ZEBU_IP::MIPI_DSI Namespace Reference

Classes

- struct Video_Cnt_t
- class DSI

Typedefs

- typedef enum ZEBU_IP::MIPI_DSI::VideoRefreshUnit_t videoRefreshUnit
- typedef enum ZEBU_IP::MIPI_DSI::VideoRotate_t videoRotate
- typedef void(∗ UserMenuCB_t)(void ∗userData)

  User menu callback prototype.

Enumerations

- enum Disp_UpdateEv_t { DCSCommand = 1, Display_Sync_V = 2, Display_Sync_H = 4 }
- enum PixelCode_t { RGB_565 = 0x0E, RGB_666 = 0x1E, RGB_666_LP = 0x2E, RGB_888 = 0x3E, Unknown = -1 }

  Pixel Coding.
- enum VideoRefreshUnit_t { videoRefreshFrame, videoRefreshLine, videoRefreshUnknown }

  Refresh period unit (frame or line)
- enum VideoRotate_t { rotateNone = 0, rotate90 = 90, rotate180 = 180, rotate270 = 270 }
- enum DSIMode_t { VIDEO_MODE = 0, DCS_CMD_MODE = 1 }

4.2.1 Typedef Documentation

4.2.1.1 typedef void(* ZEBU_IP::MIPI_DSI::UserMenuCB_t)(void *userData)

User menu callback prototype.

4.2.1.2 typedef enum ZEBU_IP::MIPI_DSI::VideoRefreshUnit_t ZEBU_IP::MIPI_DSI::videoRefreshUnit

4.2.1.3 typedef enum ZEBU_IP::MIPI_DSI::VideoRotate_t ZEBU_IP::MIPI_DSI::videoRotate

4.2.2 Enumeration Type Documentation

4.2.2.1 enum ZEBU_IP::MIPI_DSI::Disp_UpdateEv_t

Enumerator:

DCSCommand
Display_Sync_V
Display_Sync_H

4.2.2.2 enum ZEBU_IP::MIPI_DSI::DSIMode_t

Enumerator:

VIDEO_MODE
DCS_CMD_MODE

4.2.2.3 enum ZEBU_IP::MIPI_DSI::PixelCode_t

Pixel Coding.

Enumerator:

RGB_565
RGB_666
RGB_666_LP
RGB_888
Unknown

4.2.2.4 enum ZEBU_IP::MIPI_DSI::VideoRefreshUnit_t

Refresh period unit (frame or line)

Enumerator:

videoRefreshFrame
videoRefreshLine
videoRefreshUnknown

4.2.2.5 enum ZEBU_IP::MIPI_DSI::VideoRotate_t

Enumerator:

rotateNone

rotate90

rotate180

rotate270

Chapter 5

Class Documentation

5.1 ZEBU_IP::MIPI_DSI::DSI Class Reference

`#include <DSI.hh>`

Public Member Functions

- DSI (void) VS_SO_EXPORT

  MIPI_DSI Constructor.
- ~DSI (void) VS_SO_EXPORT

  MIPI_DSI Destructor.
- void setName (const char *name) VS_SO_EXPORT

  Set name of MIPI_DSI transactor that will appear in all messages prefixes.
- const char * getName (void) VS_SO_EXPORT

  Return link name (for debug purpose)
- void setDebugLevel (uint lvl) VS_SO_EXPORT

  Set debug level.
- void setLog (FILE *stream, bool stdoutDup=false) VS_SO_EXPORT

  Set log stream.
- bool setLog (char *fname, bool stdoutDup=false) VS_SO_EXPORT

  Open log file.
- PixelCode_t getPixelCoding (void) VS_SO_EXPORT

  Get Pixel coding mode.
- uint getWidth (void) VS_SO_EXPORT

  Get video width (number of pixels per line)
- uint getHeight (void) VS_SO_EXPORT

  Get video height (number of lines per frame)
- uint getHBP (void) VS_SO_EXPORT

  Get Horizontal Back Porsch length (in pixels)
- uint getVBP (void) VS_SO_EXPORT

  Get Vertical Back Porsch length (in lines)
- uint getHFP (void) VS_SO_EXPORT

  Get Horizontal Front Porsch (in pixels)

- uint getVFP (void) VS_SO_EXPORT

  Get Vertical Front Porsch (in lines)
- uint getHSync (void) VS_SO_EXPORT

  Get Horizontal Sync length (in pixels)
- uint getVSync (void) VS_SO_EXPORT

  Get Vertical Sync length (in lines)
- float getFPS (void) VS_SO_EXPORT

  Get FPS (in Frames/s)
- void setMClkFreq (float freq) VS_SO_EXPORT

  Set Master Clock frequency.
- void setErrorInjector (uint level=0) VS_SO_EXPORT

  Set Error Injection.
- void setWidth (uint w) VS_SO_EXPORT

  Set video width.
- void setHeight (uint h) VS_SO_EXPORT

  Set video height.
- void rotateVisual (videoRotate value) VS_SO_EXPORT

  Performs rotation by specified value on visual window.
- void zoomInVisual (uint value, int offsetX, int offsetY, uint width, uint height) VS_SO_EXPORT

  Performs zoom in by specified value on visual window.
- void zoomOutVisual (uint value) VS_SO_EXPORT

  Performs zoom out by specified value on visual window.
- void registerEndOfFrame_CB (void(∗userCB)(void ∗context)=NULL, void ∗context=NULL) VS_SO_EXPORT

  Register a fonction that will be called by the transactor when a frame is received.
- void registerEndOfLine_CB (void(∗userCB)(void ∗context)=NULL, void ∗context=NULL) VS_SO_EXPORT

  Register a fonction that will be called by the transactor when a lince is received.
- void setDisplayTiming (uint Tvdl, uint Tvdh, uint Thdl, uint Thdh) VS_SO_EXPORT

  Set Timing values for TE line.
- void getDisplayTiming (uint &Tvdl, uint &Tvdh, uint &Thdl, uint &Thdh) VS_SO_EXPORT

  Get Timing values for TE line.
- void setEnableLaneDPHY (uint Nb_Lane) VS_SO_EXPORT

  Enable Transactor DPHY Lane number.
- float getCurrentLaneSpeed (void) VS_SO_EXPORT

  Retrieve ratio between Master clock and DPHY-Byte clock.
- bool getLaneModelInfo (unsigned int &Nb_Lane_Tx, unsigned int &Nb_Lane_Rx, char ∗LaneModel_Type, float &LaneModel_Version) VS_SO_EXPORT

  Get Lane Model Information.
- bool init (Board ∗zebu_board, const char ∗driverName) VS_SO_EXPORT

  Initialize and connect MIPI_DSI transactor.
- void config (DSIMode_t mode=VIDEO_MODE) VS_SO_EXPORT

  Sends the configuration parameters defined by the user to the transactor.
- void useZebuServiceLoop (bool activate=true) VS_SO_EXPORT

  Connect MIPI_DSI transactor call-back to ZeBu ports.
- void useZebuSvcLoop (bool activate=true) VS_SO_EXPORT

- void useZebuServiceLoop (int(∗zebuServiceLoopHandler)(void ∗context, int pending), void ∗context) VS_SO_EXPORT

  Enable call of ZeBu service loop by MIPI_DSI Xactor.
- void useZebuServiceLoop (int(∗zebuServiceLoopHandler)(void ∗context, int pending), void ∗context, const unsigned int portGroupNumber) VS_SO_EXPORT

  Enable call of ZeBu service loop by MIPI_DSI Xactor.
- void registerUserCB (void(∗userCB)(void ∗context)=NULL, void ∗context=NULL) VS_SO_EXPORT

  Register a fonction that will be called by the transactor when unable to send/receive message on ZeBu messages ports.
- void dsiServiceLoop (int(∗serviceCB)(void ∗context, int pending)=NULL, void ∗context=NULL) VS_SO_OBSOLETE

  Handle the current MIPI_DSI transactor instance arriving and pending messages.
- void serviceLoop (int(∗servicCB)(void ∗context, int pending)=NULL, void ∗context=NULL) VS_SO_EXPORT

  Handle the current MIPI_DSI transactor instance arriving and pending messages.
- void setZebuPortGroup (const uint portGroupNumber) VS_SO_EXPORT

  Set MIPI_DSI transactor ports group.
- void start (int nb_frames=-1) VS_SO_EXPORT

  Start transactor controlled clock.
- void halt (void) VS_SO_EXPORT

  Stop transactor controlled clock.
- bool isHalted (void) VS_SO_EXPORT

  Get transactor status.
- void launchDisplay (const char ∗name, uint refreshPeriod=1, VideoRefreshUnit_t refreshUnit=videoRefreshFrame, bool blocking=true, bool black_frame=true) VS_SO_EXPORT

  Launch GTK display window and start the GTK main loop in a separate thread.
- void launchDisplay (const char ∗name, uint refreshPeriod, VideoRefreshUnit_t refreshUnit, bool blocking, bool black_frame, uint width, uint height) VS_SO_EXPORT

  Launch GTK display window and start the GTK main loop in a separate thread.
- gtkWidgetp createWindow (const char ∗name, uint refreshPeriod=1, VideoRefreshUnit_t refreshUnit=videoRefreshFrame, bool blocking=true, bool black_frame=true) VS_SO_EXPORT

  Launch GTK display window, the GTK main loop is handled by the testbench.
- gtkWidgetp createWindow (const char ∗name, uint refreshPeriod, VideoRefreshUnit_t refreshUnit, bool blocking, bool black_frame, uint width, uint height) VS_SO_EXPORT

  Launch GTK display window, the GTK main loop is handled by the testbench.
- void launchVisual (const char ∗name) VS_SO_EXPORT

  Launch GTK visual window and start the GTK main loop in a separate thread.
- gtkWidgetp createVisual (const char ∗name) VS_SO_EXPORT

  Launch GTK visual window, the GTK main loop is handled by the testbench.
- gtkWidgetp createDrawingArea (uint refreshPeriod=1, VideoRefreshUnit_t refreshUnit=videoRefreshFrame, bool blocking=true, bool black_frame=true) VS_SO_EXPORT

  Create a GTK display drawing area widget, the testbench can include the widget in its own window and handles the GTK main loop.
- bool registerUserMenuItem (UserMenuCB_t userFunc, void ∗userData=NULL, char ∗label=NULL, char ∗accel=NULL, char ∗stock_id=NULL, char ∗tooltip=NULL) VS_SO_EXPORT

  Register a user item in GTK action menu.
- void destroyDisplay (void) VS_SO_EXPORT

Destroy transactor GTK ressources.

- void destroyVisual (void) VS_SO_EXPORT

  Destroy transactor Visual window.
- bool openDumpFile (char *fileName, bool mode=false) VS_SO_EXPORT

  Open dump file and start dumping.
- bool closeDumpFile (void) VS_SO_EXPORT

  Stop dumping and close dump file.
- bool stopDump (void) VS_SO_EXPORT

  Stop dumping after next end of frame.
- bool restartDump (void) VS_SO_EXPORT

  Restart dumping in current file after next end of frame.
- bool openMonitorFile (char *fileName, uint level=0) VS_SO_EXPORT

  Open monitor file and start monitoring DSI packets.
- bool closeMonitorFile (void) VS_SO_EXPORT

  Stop monitoring and close monitor file.
- bool stopMonitor (void) VS_SO_EXPORT

  Stop monitoring after next end of frame.
- bool restartMonitor (void) VS_SO_EXPORT

  Restart monitoring in current file after next end of frame.
- bool saveFrame (const char *fileName, const char *fileFormat) VS_SO_EXPORT

  Save next RGB Frame into file with specified format.
- bool saveFrame (const char *fileName, const char *fileFormat, uint frame_start, uint frame_num) VS_SO_EXPORT

  Save RGB Frame into file with specified format.
- bool save (const char *clockName) VS_SO_EXPORT

  Save MIPI_DSI transactor state.
- bool configRestore (const char *clockName) VS_SO_EXPORT

  Configure MIPI_DSI transactor after restore.

Static Public Member Functions

- static const char * getVersion (void) VS_SO_EXPORT

  Return the current DSI transactor version.
- static bool isDriverPresent (ZEBU::Board *board) VS_SO_EXPORT

  Check MIPI_DSI Transactor presence.
- static bool firstDriver (ZEBU::Board *board) VS_SO_EXPORT

  Look for first intance of MIPI_DSI Transactor.
- static bool nextDriver (void) VS_SO_EXPORT

  Search for next MIPI_DSI Transactor instance.
- static const char * getInstanceName (void) VS_SO_EXPORT

  Get name of current MIPI_DSI Transactor instance.
- static bool IsDriverPresent (ZEBU::Board *board) VS_SO_OBSOLETE

  Check MIPI_DSI Transactor presence.
- static bool FirstDriver (ZEBU::Board *board) VS_SO_OBSOLETE

  Look for first intance of MIPI_DSI Transactor.
- static bool NextDriver (void) VS_SO_OBSOLETE

  Search for next MIPI_DSI Transactor instance.
- static const char * GetInstanceName (void) VS_SO_OBSOLETE

  Get name of current MIPI_DSI Transactor instance.

5.1.1 Constructor & Destructor Documentation

5.1.1.1 ZEBU_IP::MIPI_DSI::DSI::DSI (void)

MIPI_DSI Constructor.

5.1.1.2 ZEBU_IP::MIPI_DSI::DSI::~DSI (void)

MIPI_DSI Destructor.

5.1.2 Member Function Documentation

5.1.2.1 bool ZEBU_IP::MIPI_DSI::DSI::closeDumpFile (void)

Stop dumping and close dump file.

Return values

true	if succesful

5.1.2.2 bool ZEBU_IP::MIPI_DSI::DSI::closeMonitorFile (void)

Stop monitoring and close monitor file.

Return values

true	if succesful

5.1.2.3 void ZEBU_IP::MIPI_DSI::DSI::config (DSIMode_t *mode* = VIDEO_MODE)

Sends the configuration parameters defined by the user to the transactor.

Parameters

mode	Selects either VIDEO_MODE or DCS_CMD_MODE (default is VIDEO_MODE)

5.1.2.4 bool ZEBU_IP::MIPI_DSI::DSI::configRestore (const char * *clockName*)

Configure MIPI_DSI transactor after restore.

Sends configuration parameters defined by the user to the transactor after a restore

Parameters

clockName	Name of the controlled clock

Return values

true	if succesful

5.1.2.5 gtkWidgetp ZEBU_IP::MIPI_DSI::DSI::createDrawingArea (uint *refreshPeriod* = 1, VideoRefreshUnit_t *refreshUnit* = videoRefreshFrame, bool *blocking* = true, bool *black_frame* = true)

Create a GTK display drawing area widget, the testbench can include the widget in its own window and handles the GTK main loop.

Parameters

name	Window name
refreshPeriod	Refresh period (optionnal, default is 1)
refreshUnit	Refresh period unit (optionnal, default is VideoRefreshFrame)
blocking	Display operating mode, if true ensure that all frames are displayed
black_frame	Clear display after each end of frame

5.1.2.6 gtkWidgetp ZEBU_IP::MIPI_DSI::DSI::createVisual (const char * *name*)

Launch GTK visual window, the GTK main loop is handled by the testbench.

Parameters

name	Window name

5.1.2.7 gtkWidgetp ZEBU_IP::MIPI_DSI::DSI::createWindow (const char * *name*, uint *refreshPeriod* = 1, VideoRefreshUnit_t *refreshUnit* = videoRefreshFrame, bool *blocking* = true, bool *black_frame* = true)

Launch GTK display window, the GTK main loop is handled by the testbench.

Parameters

name	Window name
refreshPeriod	Refresh period (optionnal, default is 1)
refreshUnit	Refresh period unit (optionnal, default is VideoRefreshFrame)
blocking	Display operating mode, if true ensure that all frames are displayed
black_frame	Clear display after each end of frame

5.1.2.8 gtkWidgetp ZEBU_IP::MIPI_DSI::DSI::createWindow (const char * *name*, uint *refreshPeriod*, VideoRefreshUnit_t *refreshUnit*, bool *blocking*, bool *black_frame*, uint *width*, uint *height*)

Launch GTK display window, the GTK main loop is handled by the testbench.

Parameters

name	Window name
refreshPeriod	Refresh period (optionnal, default is 1)
refreshUnit	Refresh period unit (optionnal, default is VideoRefreshFrame)
blocking	Display operating mode, if true ensure that all frames are displayed
black_frame	Clear display after each end of frame
width	Window width
height	Winfow height

5.1.2.9 void ZEBU_IP::MIPI_DSI::DSI::destroyDisplay (void)

Destroy transactor GTK ressources.

5.1.2.10 void ZEBU_IP::MIPI_DSI::DSI::destroyVisual (void)

Destroy transactor Visual window.

5.1.2.11 void ZEBU_IP::MIPI_DSI::DSI::dsiServiceLoop (int(*)(void *context, int pending) *serviceCB* = NULL, void * *context* = NULL)

Handle the current MIPI_DSI transactor instance arriving and pending messages.

Obsolete method - use DSI::serviceLoop() instead

This method check all the MIPI_DSI ports in order to send pending messages or receive incoming messages when possible. if the service callback pointer is NULL, the method exits right after having handled the messages. if a service callback is registered, it is called after each check of the ports. The pending argument will be set to 0 if no messages could be sent or received, otherwise it will be sent to a different value.

Parameters

serviceCB	pointer to the service callback function
context	pointer that will be given as argument to the service callback

5.1.2.12 static bool ZEBU_IP::MIPI_DSI::DSI::firstDriver (ZEBU::Board * *board*) [static]

Look for first intance of MIPI_DSI Transactor.

Parameters

board	ZeBu Board

Returns

bool

Return values

true	if first MIPI_DSI Transactor instance was found
false	if no MIPI_DSI Transactor was found

5.1.2.13 static bool ZEBU_IP::MIPI_DSI::DSI::FirstDriver (ZEBU::Board * *board*) [static]

Look for first intance of MIPI_DSI Transactor.

Parameters

board	ZeBu Board

Returns

bool

Return values

true	if first MIPI_DSI Transactor instance was found
false	if no MIPI_DSI Transactor was found

5.1.2.14 float ZEBU_IP::MIPI_DSI::DSI::getCurrentLaneSpeed (void)

Retrieve ratio between Master clock and DPHY-Byte clock.

5.1.2.15 void ZEBU_IP::MIPI_DSI::DSI::getDisplayTiming (uint & *Tvdl*, uint & *Tvdh*, uint & *Thdl*, uint & *Thdh*)

Get Timing values for TE line.

Parameters

Tvdl	- vsync low duration (in master clock cycles)
Tvdh	- vsync high duration (in master clock cycles)
Thdl	- hsync low duration (in master clock cycles)
Thdh	- hsync high duration (in master clock cycles)

5.1.2.16 float ZEBU_IP::MIPI_DSI::DSI::getFPS (void)

Get FPS (in Frames/s)

5.1.2.17 uint ZEBU_IP::MIPI_DSI::DSI::getHBP (void)

Get Horizontal Back Porsch length (in pixels)

5.1.2.18 uint ZEBU_IP::MIPI_DSI::DSI::getHeight (void)

Get video height (number of lines per frame)

5.1.2.19 uint ZEBU_IP::MIPI_DSI::DSI::getHFP (void)

Get Horizontal Front Porsch (in pixels)

5.1.2.20 uint ZEBU_IP::MIPI_DSI::DSI::getHSync (void)

Get Horizontal Sync length (in pixels)

5.1.2.21 static const char* ZEBU_IP::MIPI_DSI::DSI::getInstanceName (void) [static]

Get name of current MIPI_DSI Transactor instance.

Returns

const char*

Return values

Link	name string

5.1.2.22 static const char* ZEBU_IP::MIPI_DSI::DSI::GetInstanceName (void) [static]

Get name of current MIPI_DSI Transactor instance.

Returns

const char*

Return values

Link	name string

5.1.2.23 bool ZEBU_IP::MIPI_DSI::DSI::getLaneModelInfo (unsigned int & *Nb_Lane_Tx*, unsigned int & *Nb_Lane_Rx*, char * *LaneModel_Type*, float & *LaneModel_Version*)

Get Lane Model Information.

Parameters

Nb_Lane_Tx	- Number of Tx Lanes
Nb_Lane_Rx	- Number of Rx Lanes
LaneModel_Type	- Lane Model type ("PPI", "AFE", "AFE_DIV8")
LaneModel_Version	- Lane Model Version

Return values

true	upon success
false	if Lane Model Info not yet available or unknown

5.1.2.24 const char* ZEBU_IP::MIPI_DSI::DSI::getName (void)

Return link name (for debug purpose)

Returns

char*

5.1.2.25 PixelCode_t ZEBU_IP::MIPI_DSI::DSI::getPixelCoding (void)

Get Pixel coding mode.

Return values

PixelCode_t	RGB_565, RGB_666, RGB_666_LP, RGB_888 or Unknown

5.1.2.26 uint ZEBU_IP::MIPI_DSI::DSI::getVBP (void)

Get Vertical Back Porsch length (in lines)

5.1.2.27 static const char* ZEBU_IP::MIPI_DSI::DSI::getVersion (void) [static]

Return the current DSI transactor version.

Returns

char*

Return values

string	containing the current DSI Xactor version

5.1.2.28 uint ZEBU_IP::MIPI_DSI::DSI::getVFP (void)

Get Vertical Front Porsch (in lines)

5.1.2.29 uint ZEBU_IP::MIPI_DSI::DSI::getVSync (void)

Get Vertical Sync length (in lines)

5.1.2.30 uint ZEBU_IP::MIPI_DSI::DSI::getWidth (void)

Get video width (number of pixels per line)

5.1.2.31 void ZEBU_IP::MIPI_DSI::DSI::halt (void)

Stop transactor controlled clock.

5.1.2.32 bool ZEBU_IP::MIPI_DSI::DSI::init (Board * *zebu_board*, const char * *driverName*)

Initialize and connect MIPI_DSI transactor.

Parameters

zebu_board	ZeBu board
driverName	MIPI_DSI transactor instance name

Return values

true	if initialization successful otherwise false

5.1.2.33 static bool ZEBU_IP::MIPI_DSI::DSI::isDriverPresent (ZEBU::Board * *board*) [static]

Check MIPI_DSI Transactor presence.

Parameters

board	ZeBu Board

Returns

bool

Return values

true	if at least on MIPI_DSI Transactor was found
false	if no MIPI_DSI Transactor was found

5.1.2.34 static bool ZEBU_IP::MIPI_DSI::DSI::IsDriverPresent (ZEBU::Board * *board*) [static]

Check MIPI_DSI Transactor presence.

Parameters

board	ZeBu Board

Returns

bool

Return values

true	if at least on MIPI_DSI Transactor was found
false	if no MIPI_DSI Transactor was found

5.1.2.35 bool ZEBU_IP::MIPI_DSI::DSI::isHalted (void)

Get transactor status.

Return values

true	if transactor controlled clock stopped

5.1.2.36 void ZEBU_IP::MIPI_DSI::DSI::launchDisplay (const char * *name*, uint *refreshPeriod* = `1`, VideoRefreshUnit_t *refreshUnit* = videoRefreshFrame, bool *blocking* = `true`, bool *black_frame* = `true`)

Launch GTK display window and start the GTK main loop in a separate thread.

Parameters

name	Window name
refreshPeriod	Refresh period (optionnal, default is 1)
refreshUnit	Refresh period unit (optionnal, default is VideoRefreshFrame)
blocking	Display operating mode, if true ensure that all frames are displayed
black_frame	Clear display after each end of frame

5.1.2.37 void ZEBU_IP::MIPI_DSI::DSI::launchDisplay (const char * *name*, uint *refreshPeriod*, VideoRefreshUnit_t *refreshUnit*, bool *blocking*, bool *black_frame*, uint *width*, uint *height*)

Launch GTK display window and start the GTK main loop in a separate thread.

Parameters

name	Window name
refreshPeriod	Refresh period (optionnal, default is 1)
refreshUnit	Refresh period unit (optionnal, default is VideoRefreshFrame)
blocking	Display operating mode, if true ensure that all frames are displayed
black_frame	Clear display after each end of frame
width	Window width
height	Winfow height

5.1.2.38 void ZEBU_IP::MIPI_DSI::DSI::launchVisual (const char * *name*)

Launch GTK visual window and start the GTK main loop in a separate thread.

Parameters

name	Window name

5.1.2.39 static bool ZEBU_IP::MIPI_DSI::DSI::nextDriver (void) [static]

Search for next MIPI_DSI Transactor instance.

Prior to calling this method, the search must be initialized by calling the FirstDriver() method.

Returns

bool

Return values

true	if next MIPI_DSI Transactor instance was found
false	if no more MIPI_DSI Transactor was found

5.1.2.40 static bool ZEBU_IP::MIPI_DSI::DSI::NextDriver (void) [static]

Search for next MIPI_DSI Transactor instance.

Prior to calling this method, the search must be initialized by calling the FirstDriver() method.

Returns

bool

Return values

true	if next MIPI_DSI Transactor instance was found
false	if no more MIPI_DSI Transactor was found

5.1.2.41 bool ZEBU_IP::MIPI_DSI::DSI::openDumpFile (char * *fileName*, bool *mode* = false)

Open dump file and start dumping.

Parameters

fileName	dump file name
mode	(optionnal default is false). Specifies if pixels and lines count are referenced using active video only (false) or also include blanking area (true)

Return values

true	if succesful

5.1.2.42 bool ZEBU_IP::MIPI_DSI::DSI::openMonitorFile (char * *fileName*, uint *level* = 0)

Open monitor file and start monitoring DSI packets.

Parameters

fileName	monitor file name
level	Info level (optionnal default is 0) 0, no payload, only CRC result is sent 1, payload for video Pkts only, CRC is sent with its value 2, all payloads and CRC

Return values

true	if succesful

5.1.2.43 void ZEBU_IP::MIPI_DSI::DSI::registerEndOfFrame_CB (void(*)(void *context) *userCB* = NULL, void * *context* = NULL)

Register a fonction that will be called by the transactor when a frame is received.

Parameters

userCB	pointer to the function or NULL to disable previously recorded callback
context	pointer that will be given as argument to handler

5.1.2.44 void ZEBU_IP::MIPI_DSI::DSI::registerEndOfLine_CB (void(*)(void *context) *userCB* = NULL, void * *context* = NULL)

Register a fonction that will be called by the transactor when a lince is received.

Parameters

userCB	pointer to the function or NULL to disable previously recorded callback
context	pointer that will be given as argument to handler

5.1.2.45 void ZEBU_IP::MIPI_DSI::DSI::registerUserCB (void(*)(void *context) *userCB* = NULL, void * *context* = NULL)

Register a fonction that will be called by the transactor when unable to send/receive message on ZeBu messages ports.

Parameters

userCB	pointer to the function or NULL to disable previously recorded callback
context	pointer that will be given as argument to handler

5.1.2.46 bool ZEBU_IP::MIPI_DSI::DSI::registerUserMenuItem (UserMenuCB_t *userFunc*, void * *userData* = NULL, char * *label* = NULL, char * *accel* = NULL, char * *stock_id* = NULL, char * *tooltip* = NULL)

Register a user item in GTK action menu.

Parameters

userFun	Pointer to function to be called upon menu iotem activation
userData	User function argument
label	Item name in menu
accel	Key shortcut ("X", "<shift>X", "<contorl><alt>X")
stock_id	GTK stcok id
tooltip	Tool tip text

Return values

true	if succesful

5.1.2.47 bool ZEBU_IP::MIPI_DSI::DSI::restartDump (void)

Restart dumping in current file after next end of frame.

Return values

true	if succesful

5.1.2.48 bool ZEBU_IP::MIPI_DSI::DSI::restartMonitor (void)

Restart monitoring in current file after next end of frame.

Return values

true	if succesful

5.1.2.49 void ZEBU_IP::MIPI_DSI::DSI::rotateVisual (videoRotate *value*)

Performs rotation by specified value on visual window.

Parameters

value	angle to rotate by

5.1.2.50 bool ZEBU_IP::MIPI_DSI::DSI::save (const char * *clockName*)

Save MIPI_DSI transactor state.

Save the state of the transactor before the call to ZEBU_Board::save()

Parameters

clockName	Name of the controlled clock

Return values

true	if succesful

5.1.2.51 bool ZEBU_IP::MIPI_DSI::DSI::saveFrame (const char * *fileName*, const char * *fileFormat*)

Save next RGB Frame into file with specified format.

Parameters

fileName	filename to store the frame
fileFormat	file format: can be "jpg", "bmp" or "png"

Return values

true	if succesful

5.1.2.52 bool ZEBU_IP::MIPI_DSI::DSI::saveFrame (const char * *fileName*, const char * *fileFormat*, uint *frame_start*, uint *frame_num*)

Save RGB Frame into file with specified format.

Parameters

fileName	filename to store the frame
file_format	file format: can be "jpg", "bmp or "png"
frame_start	start frame to store to file
frame_num	frames number to store to file

Return values

true	if succesful

5.1.2.53 void ZEBU_IP::MIPI_DSI::DSI::serviceLoop (int(*)(void *context, int pending) *servicCB* = NULL, void * *context* = NULL)

Handle the current MIPI_DSI transactor instance arriving and pending messages.

This method check all the MIPI_DSI ports in order to send pending messages or receive incoming messages when possible. if the service callback pointer is NULL, the method exits right after having handled the messages. if a service callback is registered, it is called after each check of the ports. The pending argument will be set to 0 if no messages could be sent or received, otherwise it will be sent to a different value.

Parameters

serviceCB	pointer to the service callback function
context	pointer that will be given as argument to the service callback

5.1.2.54 void ZEBU_IP::MIPI_DSI::DSI::setDebugLevel (uint *lvl*)

Set debug level.

Parameters

lvl	debug level from 0 to 3, 0:no debug messages and 3: all debug messages

5.1.2.55 void ZEBU_IP::MIPI_DSI::DSI::setDisplayTiming (uint *Tvdl*, uint *Tvdh*, uint *Thdl*, uint *Thdh*)

Set Timing values for TE line.

Parameters

Tvdl	- vsync low duration (in master clock cycles)
Tvdh	- vsync high duration (in master clock cycles)
Thdl	- hsync low duration (in master clock cycles)
Thdh	- hsync high duration (in master clock cycles)

5.1.2.56 void ZEBU_IP::MIPI_DSI::DSI::setEnableLaneDPHY (uint *Nb_Lane*)

Enable Transactor DPHY Lane number.

Parameters

Nb_Lane	Number of lanes to enable

5.1.2.57 void ZEBU_IP::MIPI_DSI::DSI::setErrorInjector (uint *level* = `0`)

Set Error Injection.

Parameters

level	= 0, no error 1, error in Pkt Header 2, error in Payload 3, error in Pkt Header and Payload

5.1.2.58 void ZEBU_IP::MIPI_DSI::DSI::setHeight (uint *h*)

Set video height.

Parameters

h	number of lines per frame

5.1.2.59 void ZEBU_IP::MIPI_DSI::DSI::setLog (FILE * *stream*, bool *stdoutDup* = `false`)

Set log stream.

Parameters

stream	log stream
stdoutDup	Send log on both stdout and specified file

Return values

true	upon success otherwise false

5.1.2.60 bool ZEBU_IP::MIPI_DSI::DSI::setLog (char * *fname*, bool *stdoutDup* = `false`)

Open log file.

Parameters

fname	log filename
stdoutDup	Send log on both stdout and specified file

Return values

true	upon success otherwise false

5.1.2.61 void ZEBU_IP::MIPI_DSI::DSI::setMClkFreq (float *freq*)

Set Master Clock frequency.

Parameters

freq	Master Clock frequency in MHz

5.1.2.62 void ZEBU_IP::MIPI_DSI::DSI::setName (const char * *name*)

Set name of MIPI_DSI transactor that will appear in all messages prefixes.

Parameters

name	Transactor name

5.1.2.63 void ZEBU_IP::MIPI_DSI::DSI::setWidth (uint *w*)

Set video width.

Parameters

w	number of pixels per line

5.1.2.64 void ZEBU_IP::MIPI_DSI::DSI::setZebuPortGroup (const uint *portGroupNumber*)

Set MIPI_DSI transactor ports group.

Parameters

portGroup-Number	ZeBu port group number

5.1.2.65 void ZEBU_IP::MIPI_DSI::DSI::start (int *nb_frames* = -1)

Start transactor controlled clock.

Parameters

nb_frames	Number of frames or -1 to run forever

5.1.2.66 bool ZEBU_IP::MIPI_DSI::DSI::stopDump (void)

Stop dumping after next end of frame.

Return values

true	if succesful

5.1.2.67 bool ZEBU_IP::MIPI_DSI::DSI::stopMonitor (void)

Stop monitoring after next end of frame.

Return values

true	if succesful

5.1.2.68 void ZEBU_IP::MIPI_DSI::DSI::useZebuServiceLoop (bool *activate* = `true`)

Connect MIPI_DSI transactor call-back to ZeBu ports.

Parameters

activate	true if active

5.1.2.69 void ZEBU_IP::MIPI_DSI::DSI::useZebuServiceLoop (int(∗)(void ∗context, int pending) *zebuServiceLoopHandler*, void ∗ *context*)

Enable call of ZeBu service loop by MIPI_DSI Xactor.

When activated, the transactor will call ZeBu Board::serviceLoop() with the specified arguments instead of the DSI::serviceLoop() when the current operation cannot send/receive data to/from hardware See ZeBu API documentation to know more about the ZeBu service loop.

Parameters

zebuService-LoopHandler	pointer to ZeBu service loop handler
context	pointer that will be given as argument to handler

5.1.2.70 void ZEBU_IP::MIPI_DSI::DSI::useZebuServiceLoop (int(∗)(void ∗context, int pending) *zebuServiceLoopHandler*, void ∗ *context*, const unsigned int *portGroupNumber*)

Enable call of ZeBu service loop by MIPI_DSI Xactor.

When activated, the transactor will call ZeBu Board::serviceLoop() with the specified arguments instead

of the DSI::serviceLoop() when the current operation cannot send/receive data to/from hardware See ZeBu API documentation to know more about the ZeBu service loop.

Parameters

zebuService-LoopHandler	pointer to ZeBu service loop handler
context	pointer to data that will be given as argument to handler
portGroup-Number	ZeBu port group number

5.1.2.71 void ZEBU_IP::MIPI_DSI::DSI::useZebuSvcLoop (bool *activate* = `true`)

5.1.2.72 void ZEBU_IP::MIPI_DSI::DSI::zoomInVisual (uint *value*, int *offsetX*, int *offsetY*, uint *width*, uint *height*)

Performs zoom in by specified value on visual window.

Parameters

value	zoom to apply in %
offsetX	the offset in the X direction
offsetY	the offset in the Y direction
width	the width of the region to zoom in
height	the height of the region to zoom in

5.1.2.73 void ZEBU_IP::MIPI_DSI::DSI::zoomOutVisual (uint *value*)

Performs zoom out by specified value on visual window.

Parameters

value	zoom to apply in %

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

- DSI.hh

5.2 ZEBU_IP::MIPI_DSI::Video_Cnt_t Struct Reference

`#include <DSI.hh>`

Public Attributes

- unsigned char * pBuf
- uint size

5.2.1 Member Data Documentation

5.2.1.1 unsigned char* ZEBU_IP::MIPI_DSI::Video_Cnt_t::pBuf

5.2.1.2 uint ZEBU_IP::MIPI_DSI::Video_Cnt_t::size

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

- DSI.hh

Chapter 6

File Documentation

6.1 DSI.hh File Reference

```
#include <signal.h>
#include <stdexcept>
#include <string>
#include <list>
```

Classes

- struct ZEBU_IP::MIPI_DSI::Video_Cnt_t
- class ZEBU_IP::MIPI_DSI::DSI

Namespaces

- namespace ZEBU_IP
- namespace ZEBU_IP::MIPI_DSI

Macros

- #define VS_SO_OBSOLETE
- #define _MIPI_DSI_HH_

Typedefs

- typedef struct _GtkWidget * gtkWidgetp
- typedef enum ZEBU_IP::MIPI_DSI::VideoRefreshUnit_t ZEBU_IP::MIPI_DSI::videoRefreshUnit
- typedef enum ZEBU_IP::MIPI_DSI::VideoRotate_t ZEBU_IP::MIPI_DSI::videoRotate
- typedef void(* ZEBU_IP::MIPI_DSI::UserMenuCB_t)(void *userData)

  User menu callback prototype.

Enumerations

- enum ZEBU_IP::MIPI_DSI::Disp_UpdateEv_t { ZEBU_IP::MIPI_DSI::DCSCommand = 1, ZEBU_IP::MIPI_DSI::Display_Sync_V = 2, ZEBU_IP::MIPI_DSI::Display_Sync_H = 4 }
- enum ZEBU_IP::MIPI_DSI::PixelCode_t { ZEBU_IP::MIPI_DSI::RGB_565 = 0x0E, ZEBU_IP::MIPI_DSI::RGB_666 = 0x1E, ZEBU_IP::MIPI_DSI::RGB_666_LP = 0x2E, ZEBU_IP::MIPI_DSI::RGB_888 = 0x3E, ZEBU_IP::MIPI_DSI::Unknown = -1 }

  Pixel Coding.
- enum ZEBU_IP::MIPI_DSI::VideoRefreshUnit_t { ZEBU_IP::MIPI_DSI::videoRefreshFrame, ZEBU_IP::MIPI_DSI::videoRefreshLine, ZEBU_IP::MIPI_DSI::videoRefreshUnknown }

  Refresh period unit (frame or line)
- enum ZEBU_IP::MIPI_DSI::VideoRotate_t { ZEBU_IP::MIPI_DSI::rotateNone = 0, ZEBU_IP::MIPI_DSI::rotate90 = 90, ZEBU_IP::MIPI_DSI::rotate180 = 180, ZEBU_IP::MIPI_DSI::rotate270 = 270 }
- enum ZEBU_IP::MIPI_DSI::DSIMode_t { ZEBU_IP::MIPI_DSI::VIDEO_MODE = 0, ZEBU_IP::MIPI_DSI::DCS_CMD_MODE = 1 }

6.1.1 Macro Definition Documentation

6.1.1.1 #define _MIPI_DSI_HH_

6.1.1.2 #define VS_SO_OBSOLETE

6.1.2 Typedef Documentation

6.1.2.1 typedef struct _GtkWidget* gtkWidgetp

Index