ADLINK Vision Software

Manual Rev.: 2.2.1
Revision Date: Dec. 24, 2025

ADLINK TECHNOLOGY INC.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Revision History

Revision	Release Date	Description of Change(s)
1.0	Oct. 12th, 2021	Initial release
2.0	Feb. 17th, 2025	Add Camera Link product
2.1	Nov. 7th, 2025	Add new feature for Camera Link product
2.2.1	Dec. 24th, 2025	Add Callback function for Camera Link product

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

1. INTRODUCTION 6

4.11 SETUP WINDOWS ADLINK VISION SOFTWARE SDK 7
4.12 SETUP LINUX ADLINK VISION SOFTWARE SDK 8

2. FUNCTION LIST 9

3. SUPPORTED PRODUCT LIST 12

4. FUNCTION DESCRIPTION 12

4.13 THE DEFINITIONS AND ENUMERATIONS 12
4.14 THE CONFIGURATION FIELD DEFINITION 15
4.1.1. Advanced Configuration 15
4.1.1.1. PCIe-CPL64V Full 15
4.1.2. CameraLink File 27
4.1.2.1. PCIe-CPL64V Full 27
4.15 COMMON 31
4.1.3. AVS_GetProductNums 31
4.1.4. AVS_GetSDKInfo 31
4.1.5. AVS_GetProductHandle 32
4.1.6. AVS_GetProductInfo 32
4.1.7. AVS_GetProductType 33
4.1.8. AVS_GetProductSN 33
4.1.9. AVS_GetFeatureList 34
4.1.10. AVS_GetCardID 34
4.16 FEATURE – POWERCONTROLLER 36
4.1.11. AVS_PowCtrlGetPortNums 36
4.1.12. AVS_ PowCtrlSetPortEnable 36
4.1.13. AVS_ PowCtrlGetPortEnable 37
4.17 FEATURE – EXTERNAL POWER SUPPLY 38
4.1.14. AVS_ExPowSupGetNums 38
4.1.15. AVS_ExPowSupGetState 38
4.18 FEATURE – POWER OVER ETHERNET 40
4.1.16. AVS_PoEGetPortNums 40
4.1.17. AVS_PoESetPortEnable 40
4.1.18. AVS_PoEGetPortEnable 41
4.1.19. AVS_PoESetPowConsumCalcModel 41
4.1.20. AVS_PoEGetPowConsumCalcModel 42
4.1.21. AVS_PoESetProtectTempRange 42
4.1.22. AVS_PoEGetProtectTempRange 43
4.1.23. AVS_PoEGetProperty 44
4.1.24. AVS_PoEGetPortProperty 44
4.19 FEATURE – TRIGGER OVER ETHERNET 46
4.1.25. AVS_ToEGetPortNums 46
4.1.26. AVS_ToESendActionCommand 46
4.1.27. AVS_ToESetActionCommand 47
4.1.28. AVS_ToEGetActionCommand 47
4.1.29. AVS_ToESetTriggerSource 48
4.1.30. AVS_ToEGetTriggerSource 49
4.1.31. AVS_ToESetTriggerEnable 49
4.1.32. AVS_ToEGetTriggerEnable 50
4.1.33. AVS_ToESetTriggerActivation 50
4.1.34. AVS_ToEGetTriggerActivation 51
4.1.35. AVS_ToESetOneTriggerAllMode 51
4.1.36. AVS_ToEGetOneTriggerAllMode 52
4.1.37. AVS_ToESetExTriggerDebounce 52
4.1.38. AVS_ToEGetExTriggerDebounce 53

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.39. AVS_ToEGetExTriggerDebounceUnit 53
4.1.40. AVS_ToEGetTriggerCount 54
4.1.41. AVS_ToEResetTriggerCount 54
4.110 FEATURE – FAN 55
4.1.42. AVS_FanGetNums 55
4.1.43. AVS_FanGetSpeed 55
4.1.44. AVS_FanGetPWM 56
4.1.45. AVS_FanSetPWM 56
4.1.46. AVS_FanGetSmartFanEnable 57
4.1.47. AVS_FanSetSmartFanEnable 57
4.111 FEATURE – DIO 59
4.1.48. AVS_DIOGetCapability 59
4.1.49. AVS_DIOGetDINums 59
4.1.50. AVS_DIOGetDONums 60
4.1.51. AVS_DIOGetDICapability 60
4.1.52. AVS_DIOGetDOCapability 61
4.1.53. AVS_DIOSetDOState 62
4.1.54. AVS_DIOSetDOStates 62
4.1.55. AVS_DIOGetDOState 63
4.1.56. AVS_DIOGetDOStates 63
4.1.57. AVS_DIOGetDIState 64
4.1.58. AVS_DIOGetDIStates 65
4.1.59. AVS_DIOSetDIDebounceTime 65
4.1.60. AVS_DIOGetDIDebounceTime 66
4.1.61. AVS_DIOGetDIDebounceTimeUnit 67
4.1.62. AVS_DIOSetDOInvertState 67
4.1.63. AVS_DIOGetDOInvertState 68
4.1.64. AVS_DIOSetDITriggerEnable 68
4.1.65. AVS_DIOGetDITriggerEnable 69
4.1.66. AVS_DIOSetDOTriggerEnable 69
4.1.67. AVS_DIOGetDOTriggerEnable 70
4.1.68. AVS_DIOSetDITriggerCallback 71
4.1.69. AVS_DIOSetDIEdgeTrigger 71
4.1.70. AVS_DIOGetDIEdgeTrigger 72
4.1.71. AVS_DIOSetDITriggerPulseWidth 72
4.1.72. AVS_DIOGetDITriggerPulseWidth 73
4.1.73. AVS_DIOGetDITriggerPulseWidthUnit 74
4.1.74. AVS_DIOSetDOTriggerDelay 74
4.1.75. AVS_DIOGetDOTriggerDelay 75
4.1.76. AVS_DIOGetDOTriggerDelayUnit 75
4.1.77. AVS_DIOSetDOTriggerPulseWidth 76
4.1.78. AVS_DIOGetDOTriggerPulseWidth 76
4.1.79. AVS_DIOGetDOTriggerPulseWidthUnit 77
4.1.80. AVS_DIOTriggerDO 77
4.1.81. AVS_DIOTriggerDOs 78
4.1.82. AVS_DIOSetTrigger1vs1MappingTable 79
4.1.83. AVS_DIOGetTrigger1vs1MappingTable 79
4.1.84. AVS_DIOSetTrigger1vsMultipleMappingTable 80
4.1.85. AVS_DIOGetTrigger1vsMultipleMappingTable 81
4.112 FEATURE – FRAMEGRABBER 82
4.1.86. AVS_FrameGrabberGetDSNums 82
4.1.87. AVS_FrameGrabberStartAquisition 82
4.1.88. AVS_FrameGrabberStopAquisition 83
4.1.89. AVS_FrameGrabberGetFrame 83
4.1.90. AVS_FrameGrabberQueueBuffer 84
4.1.91. AVS_FrameGrabberGetDropFrame 84

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.92. AVS_FrameGrabberSetCallback *85*
4.1.93. AVS_FrameGrabberSetBufferCount *86*
4.113 FEATURE – ADVANCEDCONFIG 88
4.2.1. AVS_AdvancedConfigWritePars *88*
4.2.2. AVS_AdvancedConfigReadPars *88*
4.2.3. AVS_AdvancedConfigGetSupportedAdr *89*
4.114 FEATURE – CAMERALINK 90
4.2.4. AVS_CameraLinkImportCamFile *90*
4.2.5. AVS_CameraLinkExportCamFile *90*
4.2.6. AVS_CameraLinkGetSupportedSectionKey *91*
4.2.7. AVS_CameraLinkReadParameter *91*
4.2.8. AVS_CameraLinkWriteParameter *92*

5. SERIAL COMMUNICATION FUNCTIONS **94**

4.115 DEFINED DATA TYPE 94
4.116 CLFLUSHPORT 94
4.117 CLGETERRORTEXT 94
4.118 CLGETNUMBYTESAVAIL 95
4.119 CLGETNUMPORTS 95
4.120 CLGETPORTINFO 96
4.121 CLGETSUPPORTEDBAUDRATES 97
4.122 CLSERIALCLOSE 97
4.123 CLSERIALINIT 98
4.124 CLSERIALREAD 98
4.125 CLSERIALWRITE 99
4.126 CLSETBAUDRATE 99
4.127 STATUS CODES 100

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

1. Introduction

The AVS SDK provides the Power Controller device and Cameralink device functions. functions.

The AVS SDK use the C function interface and it could run on the Windows and Linux x86 platform. You can call these functions directly. However, the most functions need to call AVS_GetProductHandle() to get a handle by specifying a device index, and you could use this handle to operate corresponding functions.

- Requirements:
  - Platform: x86, x86-64
  - OS:
    Newer than Windows 7 or
    Newer than Ubuntu 12.04 LTS (Linux kernel newer than 3.13)
  - Devices: Vsion cards

- Example Code:

```
#ifdef __linux__
#include "/usr/local/include/ADLINK/AVS/AVS_SDK.h"
#include "/usr/local/include/ADLINK/AVS/Common/I_common.h"
#elif _WIN32
#include "Include/C++/AVS_SDK.h"
#include "Include/C++/Common/I_common.h"
#endif

using namespace std;
/*Show SDK Information*/
char buffer[256];
if (AVS_GetSDKInfo(buffer, sizeof(buffer)) == AVS_FUN_SUCCESS)
cout << buffer << endl;
...
/*Show Product Information*/
unsigned int nums = AVS_GetProductNums();
if (nums <= 0)
{
        cout << "Find no device!" << endl;
        return 0;
}

void* handle;
unsigned int devIndex = 0;
handle = AVS_GetProductHandle(devIndex);
if (!handle)
{
        cout << "AVS_GetProductHandle Error\n" << endl;
        return 0;
}


if (AVS_GetProductInfo(handle, buffer, sizeof(buffer)) == AVS_FUN_SUCCESS)
        cout << buffer << endl;
...
```

- There are main three parts in the AVS SDK.
  - General SDK functions. Described in the file "<include path>\AVS_SDK.h".

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

- These functions use for getting handle and retrieving the device information..
- Features functions. Described in the folder " <include path>\Feature ".
  - These functions use for operating the features of devices.
- The defined data types are described in the file "<include path>\Common\I_common.h"

- About sample codes:
  - There are two types of samples in the windows.
    1. The "GUI" sample is the entry point for quickly start to develop the sdk with GUI.
    2. The "Console" sample is the pure application and the same as linux sample.
  - The each console sample name is the SDK function name
    1. Check if the SDK function is working well by executing the console sample.
    2. Check the usage of the SDK function by searching the function name in the main.cpp source code.
  - The AVS Windows SDK provides the feature GUI tools under the Sample/Console folder.
    1. The feature GUI tool is an applications of executing the console sample.
    2. The feature GUI tools can help to check if the AVS SDK is working well and easily operate these features.
  - The SerialControl sample allows you to establish serial communication between a host and an external device.

4.11 Setup Windows ADLINK Vision Software SDK

1. Execute windows installation package.
2. The path of files:
   - **C++ Include path:**
     "%PROGRAMFILES%/ADLINK/AVS/Include/C++"
   - **C# Include path:**
     "%PROGRAMFILES%/ADLINK/AVS/Include/C#"
   - **VB.NET Include path:**
     "%PROGRAMFILES%/ADLINK/AVS/Include/VB.NET"
   - **Visual C++ library interface file:**
     "%PROGRAMFILES%/ADLINK/AVS/DLL/AVS_SDK.lib"
   - Libraries:
     "%WINDIR%/system32/AVS_SDK.dll"
   - Dependency Libraries:
     "%WINDIR%/system32/AVS_Product.dll"
     "%WINDIR%/system32/AVS_CoreSDK.dll"
     "%WINDIR%/system32/AVS_Utilities.dll"
3. Sample path, you could check how to use AVS SDK in each language sample:
   - **VC++ Sample:**
     "%PROGRAMFILES%/ADLINK/AVS/Sample/Console/VC++"
     "%PROGRAMFILES%/ADLINK/AVS/Sample/GUI/VC++"
   - **C# Sample:**
     "%PROGRAMFILES%/ADLINK/AVS/Sample/Console/C#"
     “%PROGRAMFILES%/ADLINK/AVS/Sample/GUI/C#"
   - **VB.NET Sample:**
     "%PROGRAMFILES%/ADLINK/AVS/Sample/Console/VB.NET"
     "%PROGRAMFILES%/ADLINK/AVS/Sample/GUI/VB.NET"
   - SampleGUITool. Provide GUI to execute sample. You can modify “AppPath.ini” to specify a default sample application path.
     "%PROGRAMFILES%/ADLINK/AVS/Sample/Console/"
   - Cameralink serial control sample
     "%PROGRAMFILES%/ADLINK/AVS/Sample/SerialControl/"
4. Serial communication registry:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

- **Registry Key:**
  " HKEY_LOCAL_MACHINE\software\cameralink"
- **Value Name:**
  " CLSERIALPATH"
- **Value Data:**
  " C:\cameralink\serial"
- If you have other cards providing their library already, please redefine the value of this registry key and copy clseradl.dll to the directory you defined. Then you can access all cards by the standard serial communication functions.

Library File	Description
clallserial.dll clseradl.dll	Dynamic link library. Required for Serail communication applications.

4.12 Setup Linux ADLINK Vision Software SDK

- Before start to install

1. Make sure your system support debian package
2. If your system has no support debian and dkms package, please untar these files and deploy the SDK and driver by yourself.

- Installation commands:
  Need to install the dkms first. If you had already installed dkms, you can skip this step. The command in Ubuntu OS:
  - If you can't install the dkms, you can update the package list.

  $sudo apt-get update
  $sudo apt-get install dkms

1. $tar xvf AVS-SDK_xxx.tar
   $cd AVS-SDK_xxx
   $./install.sh
   $sudo reboot

2. The path of files:
   - lib:
     /usr/local/lib/libAVS_SDK.so
   - include path:
     /usr/local/include/ADLINK/AVS
   - sample:
     /usr/local/src/ADLINK/AVS/Sample
   - Dependency Libraries:
     /usr/local/lib/libAVS_Product.so
     /usr/local/lib/libAVS_CoreSDK.so

3. Build Sample:
   $cd /usr/local/src/ADLINK/AVS/Sample
   $make

4. Execute Sample
   #cd /usr/local/src/ ADLINK/AVS/Sample/bin
   #sudo ./AVS_GetSDKInfo
   *This will show the driver and SDK version

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2. Function List

Category	Functino name	File
Common	AVS_GetProductNums()	AVS_SDK.h
	AVS_GetSDKInfo()	
	AVS_GetProductHandle()	
	AVS_GetProductInfo()	
	AVS_GetProductType()	
	AVS_GetFeatureList()	
	AVS_GetCardID()	
Power Controller	AVS_PowCtrlGetPortNums()	Feature/PowerController.h
	AVS_PowCtrlSetPortEnable()	
	AVS_PowCtrlGetPortEnable()	
External Power Supply	AVS_ExPowSupGetNums()	Feature/ExPowerSupply.h
	AVS_ExPowSupGetState()	
PoE (Power over Ethernet)	AVS_PoEGetPortNums()	Feature/PoE.h
	AVS_PoESetPortEnable()	
	AVS_PoEGetPortEnable()	
	AVS_PoESetPowConsumCalcModel()	
	AVS_PoEGetPowConsumCalcModel()	
	AVS_PoESetProtectTempRange()	
	AVS_PoEGetProtectTempRange()	
	AVS_PoEGetProperty()	
	AVS_PoEGetPortProperty()	
ToE (Trigger over Ethernet)	AVS_ToEGetPortNums()	Feature/ToE.h
	AVS_ToESendActionCommand()	
	AVS_ToESetActionCommand()	
	AVS_ToEGetActionCommand()	
	AVS_ToESetTriggerSource()	
	AVS_ToEGetTriggerSource()	
	AVS_ToESetTriggerEnable()	
	AVS_ToEGetTriggerEnable()	
	AVS_ToESetTriggerActivation()	
	AVS_ToESetTriggerActivation()	
	AVS_ToESetOneTriggerAllMode()	
	AVS_ToEGetOneTriggerAllMode()	
	AVS_ToESetExTriggerDebounce()	
	AVS_ToEGetExTriggerDebounceUnit()	
	AVS_ToEGetTriggerCount()	
	AVS_ToEResetTriggerCount()	

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Fan	AVS_FanGetNums()	Feature/Fan.h
	AVS_FanGetSpeed()	
	AVS_FanGetPWM()	
	AVS_FanSetPWM()	
	AVS_FanGetSmartFanEnable()	
	AVS_FanSetSmartFanEnable()	
DIO	AVS_DIOGetCapability()	Feature/DIO.h
	AVS_DIOGetDINums()	
	AVS_DIOGetDONums()	
	AVS_DIOGetDICapability()	
	AVS_DIOGetDOCapability()	
	AVS_DIOSetDOState()	
	AVS_DIOSetDOStates()	
	AVS_DIOGetDOState()	
	AVS_DIOGetDOStates()	
	AVS_DIOGetDIState()	
	AVS_DIOGetDIStates()	
	AVS_DIOSetDIDebounceTime()	
	AVS_DIOGetDIDebounceTime()	
	AVS_DIOGetDIDebounceTimeUnit()	
	AVS_DIOSetDOInvertState()	
	AVS_DIOGetDOInvertState()	
	AVS_DIOSetDITriggerEnable()	
	AVS_DIOGetDITriggerEnable()	
	AVS_DIOSetDOTriggerEnable()	
	AVS_DIOGetDOTriggerEnable()	
	AVS_DIOSetDITriggerCallback()	
	AVS_DIOSetDIEdgeTrigger()	
	AVS_DIOGetDIEdgeTrigger()	
	AVS_DIOSetDITriggerPulseWidth()	
	AVS_DIOGetDITriggerPulseWidth()	
	AVS_DIOGetDITriggerPulseWidthUnit()	
	AVS_DIOSetDOTriggerDelay()	
	AVS_DIOGetDOTriggerDelay()	
	AVS_DIOGetDOTriggerDelayUnit()	
	AVS_DIOSetDOTriggerPulseWidth()	
	AVS_DIOGetDOTriggerPulseWidth()	
	AVS_DIOGetDOTriggerPulseWidthUnit()	
	AVS_DIOTriggerDO()	
	AVS_DIOTriggerDOs()	

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

	AVS_DIOSetTrigger1vs1MappingTable()	
	AVS_DIOGetTrigger1vs1MappingTable()	
	AVS_DIOSetTrigger1vsMultipleMappingTable()	
	AVS_DIOGetTrigger1vsMultipleMappingTable()	
FrameGrabber	AVS_FrameGrabberGetDSNums()	Feature/FrameGrabber.h
	AVS_FrameGrabberStartAquisition()	
	AVS_FrameGrabberStopAquisition()	
	AVS_FrameGrabberQueueBuffer()	
	AVS_FrameGrabberGetFrame()	
	AVS_FrameGrabberGetDropFrame()	
	AVS_FrameGrabberSetCallback()	
	AVS_FrameGrabberSetBufferCount()	
AdvancedConfig	AVS_AdvancedConfigWritePars()	Feature/AdvancedConfig.h
	AVS_AdvancedConfigReadPars()	
	AVS_AdvancedConfigGetSupportedAdr()	
CameraLink	AVS_CameraLinkImportCamFile()	Feature/CameraLink.h
	AVS_CameraLinkExportCamFile()	
	AVS_CameraLinkGetSupportedSectionKey()	
	AVS_CameraLinkReadParameter()	
	AVS_CameraLinkWriteParameter()	

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3. Supported Product List

Product\Function	Common	Power Controller	External Power Supply	PoE	ToE	Fan	DIO	Frame Grabber	Advanced Config	CameraLink
PCIe-GIE Series	V		V	V		V				
PCIe-GIE Pro Series	V		V	V	V	V				
PCIe-10G Series	V		V	V	V	V				
PCI-U3 Series	V	V	V							
EOS JNX	V			V			V			
CPL64V	V							V	V	V

4. Function Description

4.13 The Definitions and enumerations

You could find the following definitions or enumerations in I_common.h:

AVS_FUN_RETURNVAL		
Enumeration	Comment	Error code
AVS_FUN_SUCCESS	Execute successfully	0
AVS_FUN_BUSY	System is busy	-100
AVS_FUN_FAILED	Execution failed	-1
AVS_FUN_INVALIDPARA	The parameters are invalid	-11
AVS_FUN_NOTINIT	The object is not initialized	-13
AVS_FUN_NOTDEFINED	Unsupported function	-5
AVS_FUN_BUFFERSIZEISTOOSMALL	The specified buffer or buffer size is too smaller for the operation	-101
AVS_FUN_TIMEOUT	The operation time is over the specified time	-102
AVS_FUN_ADV_ADDRESSERROR	The address of the advanced configuration is invalid	-200
AVS_FUN_ADV_VALUEERROR	The value of the advanced configuration is invalid	-201
AVS_FUN_ADV_ADDRESSCANNOTBEREAD	The address of the advanced configuration cannot be read	-202
AVS_FUN_ADV_ADDRESSCANNOTBEWRITEEN	The address of the advanced configuration cannot be written	-203
AVS_FUN_CL_FILENOTFOUND	The cameralink file is not found	-300
AVS_FUN_CL_SECTIONORKEYERROR	The sction or key of the cameralink parameter field is invalid	-301

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

AVS_FUN_CL_VALUEERROR	The value of the cameralink section and key is invalid	-302
AVS_FUN_FG_FRAMEERROR	• Format mismatch: tap configuration is incorrect. • Missing line data: including Line Valid missing. Note: The height setting is determined by the user.	-400

AVS_FEATURE_TYPE	
Enumeration	Comment
AVS_FEATURE_POWERCONTROLLER	Power controller
AVS_FEATURE_POE	Power over ethernet(PoE)
AVS_FEATURE_TOE	Trigger over ethernet(ToE)
AVS_FEATURE_DIO	Digital Input/Output
AVS_FEATURE_EXTERNALPOWERSUPPLY	External power supply
AVS_FEATURE_FAN	Fan control
AVS_FEATURE_FRAMGRABBER	Frame grabber
AVS_FEATURE_CAMERALINK	Cameralink
AVS_FEATURE_ADVCONFIG	ADLINK advanced configuration
AVS_FEATURE_NUMS	The number of the feature type definitions
AVS_FEATURE_NODEVICE	Unkonwn feature

AVS_PRODUCT_TYPE	
Enumeration	Comment
AVS_PRODUCT_UNKNOWN	Unkonwn product
AVS_PRODUCT_PCIEGIE64PLUS	PCIe-GIE64+
AVS_PRODUCT_PCIEGIE72	PCIe-GIE72
AVS_PRODUCT_PCIEGIE74	PCIe-GIE74
AVS_PRODUCT_PCIEGIE72PRO	PCIe-GIE72 PRO
AVS_PRODUCT_PCIEGIE74PRO	PCIe-GIE74 PRO
AVS_PRODUCT_PCIEGIENVQP1000	PCIe-GIENVQ-P1000
AVS_PRODUCT_PCIEGIENVQP2000	PCIe-GIENVQ-P2000
AVS_PRODUCT_PCIEGIEIMXMA2485	PCIe-GIEIMX
AVS_PRODUCT_PCIEU304	PCIe-U304
AVS_PRODUCT_PCIEU308	PCIe-U308
AVS_PRODUCT_PCIEU312	PCIe-U312
AVS_PRODUCT_PCIE10GPOE	PCIe-10GPoE

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

AVS_PRODUCT_PCIECPL64VFULL	PCIe-CPL64V Full
AVS_PRODUCT_NUMS	The number of the product type defination

AVS_EDGETRIGGER_TYPE	
Enumeration	Comment
AVS_EDGETRIGGER_FALLINGEDGE	Signal falling edge trigger
AVS_EDGETRIGGER_RISINGEDGE	Signal rising edge trigger
AVS_EDGETRIGGER_CHANGINGEDGE	Signal changing edge trigger
AVS_EDGETRIGGER_NUMS	The number of the edge trigger type

AVS_POEPROPERTY_TYPE	
Enumeration	Comment
AVS_POEPROPERTY_POEHWSETTING	PSE Hardware setting
AVS_POEPROPERTY_POEINITIALSTATE	PSE initial power
AVS_POEPROPERTY_OVERTEMPERATURE	Check if the device temperature is over high temperature (degree C)
AVS_POEPROPERTY_CURRENTTEMPERATURE	Get the current temperature
AVS_POEPROPERTY_CONSUMEPOWER	The consumer power(Watt)
AVS_POEPROPERTY_REMAINPOWER,	The left power(Watt)
AVS_POEPROPERTY_POWERBUDGET	The total power budget(Watt)
AVS_POEPROPERTY_NUMS	The number of the poe property

AVS_POEPORTPROPERTY_TYPE	
Enumeration	Comment
AVS_POEPORTPROPERTY_CURRENT	The current of a port (A)
AVS_POEPORTPROPERTY_VOLT	The volt of a port
AVS_POEPORTPROPERTY_POECLASS	The power class of a port
AVS_POEPORTPROPERTY_POWERGOOD	Check if the power is good of a port
AVS_POEPORTPROPERTY_LASTPOWERSTATE	Retrieve the last power state of a port
AVS_POEPORTPROPERTY_NUMS	The number of the poe port properties

AVS_DIOCAPABILITY_TYPE	
Enumeration	Comment
AVS_DIOCAPABILITY_DI	Support DI (FW/HW)
AVS_DIOCAPABILITY_DO	Support DO (FW/HW)
AVS_DIOCAPABILITY_1VS1TRIGGER	Support 1vs1 trigger mapping table (FW/HW)
VS_DIOCAPABILITY_1VSMULTIPLETRIGGER	Support 1vsMultiple trigger mapping table (FW/HW)
AVS_DIOCAPABILITY_NUMS	The number of the DIO capabilities

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

AVS_DIODICAPABILITY_TYPE	
Enumeration	Comment
AVS_DIODICAPABILITY_GET	Support get state
AVS_DIODICAPABILITY_DEBOUNCETIME	Support set/get DI debounce time (FW/HW)
AVS_DIODICAPABILITY_TRIGGEROFFSWITCH	Support DI turn off trigger (FW/HW)
AVS_DIODICAPABILITY_TRIGGERDETECTION	Support detect DI hw trigger (FW/HW)
AVS_DIODICAPABILITY_EDGETIRGGERTYPE	Support set/get DI edge trigger type (FW/HW)
AVS_DIODICAPABILITY_TRIGGERPULSEWIDTH	Support set/get DI trigger pulse width (FW/HW)
AVS_DIODICAPABILITY_SPECIALFUNCTION	This DI has special function, please refer the user manual or its product function interface
AVS_DIODICAPABILITY_NUMS	The number of the DIO's DI capabilities

AVS_DIODOCAPABILITY_TYPE	
Enumeration	Comment
AVS_DIODOCAPABILITY_SET	Support set state
AVS_DIODOCAPABILITY_GET	Support get state
AVS_DIODOCAPABILITY_INVERTSTATE	Support set/get invert the DO output state (FW/HW)
AVS_DIODOCAPABILITY_SWTRIGGER	Support SW trigger DO (FW/HW)
AVS_DIODOCAPABILITY_TRIGGEROFFSWITCH	Support DO turn off trigger (FW/HW)
AVS_DIODOCAPABILITY_TRIGGERDELAY	Support set/get DO delay trigger time (FW/HW)
AVS_DIODOCAPABILITY_TRIGGERPULSEWIDTH	Support set/get DO trigger pulse width (FW/HW)
AVS_DIODOCAPABILITY_SPECIALFUNCTION	This DO has special function, please refer the user manual or its product function interface
AVS_DIODOCAPABILITY_NUMS	The number of the DIO's DO capabilities

4.14 The Configuration Field Definition

4.1.1. Advanced Configuration

The AdvancedConfig feature allows you to config a device via the register address. Below are the definitions for each supported product.

4.1.1.1. PCIe-CPL64V Full

RW	Address	Description
		Parameter (Camfile Section.Key = Value)

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

[R]	0x1000	Information of the Card ID - The value depends on the the user settings
		DeviceInfo.CardID = 0 ~ 15
[R]	0x1001	Information of the Hardware Version - The value depends on the hardware revision
		DeviceInfo.HardwareVersion = 0 ~ 255 - The version is encoded in binary format. The MSB defines the major version (0 = A, 1 = B), and the lower three bits indicate the minor version (1–7)
[R]	0x1002	FPGA version release time - hhmm : hh = hour(24H)[15:8], mm = minutes[7:0]
		DeviceInfo.FPGADateCode1 = 0 ~ 65535 - The version code is displayed in decimal format and is used together with 0x1003 EX: Display : 1800, Data format：0x1800
[R]	0x1003	FPGA version release date 4 - YYYYMMDD : YYYY = year[31:16], MM = month[15:8], DD = day[7:0]
		DeviceInfo.FPGADateCode2 = 0 ~ 4294967295 - The version code is displayed in decimal format and is used together with 0x1002 EX: Display : 20241022, Data format：0x20241022
[RW]	0x1100	Scan mode settings of the card
		CaptureMode.ScanMode = 0x1:AreaScan / 0x0:LineScan
[RW]	0x1202	Grabber image width setting
		ImageFormat.SensorWidth = 32 ~ 65504 *Note8
[RW]	0x1203	Grabber image height setting
		ImageFormat.SensorHeight = 0 ~ 65535
[RW]	0x1204	Tap configuration for image format - When a camera has multiple taps (2-tap, 4-tap, 8-tap, 10-tap), each tap outputs a subset of pixels from each image line
		ImageFormat.Tap = 0x0:1 / 0x5:10 / 0x1:2 / 0x3:4 / 0x4:8
[RW]	0x1205	Pixel Size setting for image format - Setting outputs 8, 10, or 12 bits of data per pixel channel.
		ImageFormat.PixelSize = 0x1:10 / 0x2:12 / 0x0:8
[RW]	0x1206	Tap placement setting for image format - Tap Placement (Geometry) defines how pixel data is distributed across multiple parallel output taps in a Camera Link camera.
		ImageFormat.TapPlacement = 0x0:1x / 0x30:1x10 / 0x1:1x2 / 0x10:1x4 / 0x20:1x8
[RW]	0x1207	Pixel Format setting for image format - The Pixel Format feature allows you to choose the format of the image data transmitted by the frame grabber (only supported in unpacked format).
		ImageFormat.PixelFormat = 0x0:Mono / 0x1:RGB
[RW]	0x1208	Enable setting for FVAL - 0x0 = False – The frame-valid (FVAL) signal is ignored. - 0x1 = True – The frame-valid signal is used; the grabber relies on FVAL to identify the active frame period
		ImageFormat.FVALEnable = 0x0:False / 0x1:True
[RW]	0x1209	Enable setting for LVAL - 0x0 = False – The line-valid (LVAL) signal is ignored. - 0x1 = True – The line-valid signal is used; the grabber relies on LVAL to determine valid line intervals.
		ImageFormat.LVALEnable = 0x0:False / 0x1:True
[RW]	0x120A	Enable setting for DVAL - 0x0 = False – The data-valid (DVAL) signal is ignored. - 0x1 = True – The data-valid signal is used; the grabber uses it to identify valid pixel periods.
		ImageFormat.DVALEnable = 0x0:False / 0x1:True

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

[RW]	0x120B	Polarity of FVAL
		ImageFormat.FVALPolarity = 0x1:Falling / 0x0:Rising
[RW]	0x120C	Polarity of LVAL
		ImageFormat.LVALPolarity = 0x1:Falling / 0x0:Rising
[RW]	0x120D	Polarity of DVAL
		ImageFormat.DVALPolarity = 0x1:Falling / 0x0:Rising
[RW]	0x1300	High Pulse Width of Port0 CC1
		CameraControl.Port0CC1HighPulseWidth = 0 ~ 16777215 (unit: µs)
[RW]	0x1301	Low Pulse Width of Port0 CC1
		CameraControl.Port0CC1LowPulseWidth = 0 ~ 16777215 (unit: µs)
[RW]	0x1302	High Pulse Width of Port0 CC2
		CameraControl.Port0CC2HighPulseWidth = 0 ~ 16777215 (unit: µs)
[RW]	0x1303	Low Pulse Width of Port0 CC2
		CameraControl.Port0CC2LowPulseWidth = 0 ~ 16777215 (unit: µs)
[RW]	0x1304	Inverted Polarity of Port0 CC1
		CameraControl.Port0CC1InvertedPolarity = 0x1:Active / 0x0:Inactive
[RW]	0x1305	Inverted Polarity of Port0 CC2
		CameraControl.Port0CC2InvertedPolarity = 0x1:Active / 0x0:Inactive
[RW]	0x1306	Output Mode of Port0 CC1
		CameraControl.Port0CC1OutputMode = 0x1:Manual / 0x0:Trigger
[RW]	0x1307	Output Mode of Port0 CC2
		CameraControl.Port0CC2OutputMode = 0x1:Manual / 0x0:Trigger
[R]	0x1308	Number of trigger event received by Port0CC1Delay
		TriggerDelay.Port0CC1TriggerEventCount = 0 ~ 4294967295 *Note6
[R]	0x1309	Number of trigger event dropped by Port0CC1Delay
		TriggerDelay.Port0CC1TriggerDropCount = 0 ~ 4294967295 *Note6
[R]	0x130A	Number of trigger event received by Port0CC2Delay
		TriggerDelay.Port0CC2TriggerEventCount = 0 ~ 4294967295 *Note6
[R]	0x130B	Number of trigger event dropped by Port0CC2Delay
		TriggerDelay.Port0CC2TriggerDropCount = 0 ~ 4294967295 *Note6
[R]	0x130C	Number of trigger event received by Port0CameraControl1
		CameraControl.Port0CC1EventCount = 0 ~ 4294967295 *Note6
[R]	0x130D	Number of trigger event dropped by Port0CameraControl1
		CameraControl.Port0CC1DropCount = 0 ~ 4294967295 *Note6
[R]	0x130E	Number of trigger event received by Port0CameraControl2
		CameraControl.Port0CC2EventCount = 0 ~ 4294967295 *Note6
[R]	0x130F	Number of trigger event dropped by Port0CameraControl2
		CameraControl.PortCC2DropCount = 0 ~ 4294967295 *Note6
[RW]	0x1310	Port0 CC1 Status
		CameraControl.Port0CC1Status = 0x1:Active / 0x0:Inactive

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

[RW]	0x1311	Port0 CC2 Status
		CameraControl.Port0CC2Status = 0x1:Active / 0x0:Inactive
[RW]	0x1312	Port0 CC3 Status
		CameraControl.Port0CC3Status = 0x1:Active / 0x0:Inactive
[RW]	0x1313	Port0 CC4 Status
		CameraControl.Port0CC4Status = 0x1:Active / 0x0:Inactive
[RW]	0x1500	Encoder mode setting
		EncoderInput.Mode = 0x2:ABphaseX1 / 0x3:ABphaseX2 / 0x4:ABphaseX4 / 0x1:CW/CCW / 0x0:OUT/DIR
[RW]	0x1501	Compare Count sets the target count for comparison
		EncoderInput.CompareCount = 0 ~ 4294967295
[RW]	0x1502	Number of Encoder counter
		EncoderInput.EncoderValue = 0 ~ 4294967295
[RW]	0x1600	Reset the debug counter
		TriggerControl.ResetDebugCount = 0x0:False / 0x1:True *Note6
[R]	0x1601	FPGA time stamp (reserve)
		TriggerControl.FPGATimeStamp = 0 ~ 4294967295
[W]	0x1602	Send the software trigger command
		TriggerControl.SWTrigger = 0x1:True
[RW]	0x1700	Trigger Source of MultiCard output
		MultiCardTrigSrc.TriggerSource = 0x1:DI0 / 0x2:DI1 / 0x3:DI2 / 0x4:DI3 / 0x0:Disable / 0x5:EncoderComparer / 0x6:MultiCardSync / 0x7:SoftwareTrigger
[RW]	0x1701	Trigger Source of Digital Outout0
		TriggerOutputTrigSrc.DO0TriggerSource = 0x1:DI0 / 0x2:DI1 / 0x3:DI2 / 0x4:DI3 / 0x0:Disable / 0x5:EncoderComparer / 0x6:MultiCardSync / 0x7:SoftwareTrigger
[RW]	0x1702	Trigger Source of Digital Outout1
		TriggerOutputTrigSrc.DO1TriggerSource = 0x1:DI0 / 0x2:DI1 / 0x3:DI2 / 0x4:DI3 / 0x0:Disable / 0x5:EncoderComparer / 0x6:MultiCardSync / 0x7:SoftwareTrigger
[RW]	0x1703	Trigger Source of Digital Outout2
		TriggerOutputTrigSrc.DO2TriggerSource = 0x1:DI0 / 0x2:DI1 / 0x3:DI2 / 0x4:DI3 / 0x0:Disable / 0x5:EncoderComparer / 0x6:MultiCardSync / 0x7:SoftwareTrigger
[RW]	0x1704	Trigger Source of Digital Outout3
		TriggerOutputTrigSrc.DO3TriggerSource = 0x1:DI0 / 0x2:DI1 / 0x3:DI2 / 0x4:DI3 / 0x0:Disable / 0x5:EncoderComparer / 0x6:MultiCardSync / 0x7:SoftwareTrigger
[RW]	0x1705	Trigger Source of Port0 CC1
		TriggerOutputTrigSrc.Port0CC1TriggerSource = 0x1:DI0 / 0x2:DI1 / 0x3:DI2 / 0x4:DI3 / 0x0:Disable / 0x5:EncoderComparer / 0x6:MultiCardSync / 0x7:SoftwareTrigger
[RW]	0x1706	Trigger Source of Port0 CC2
		TriggerOutputTrigSrc.Port0CC2TriggerSource = 0x1:DI0 / 0x2:DI1 / 0x3:DI2 / 0x4:DI3 / 0x0:Disable / 0x5:EncoderComparer / 0x6:MultiCardSync / 0x7:SoftwareTrigger
[RW]	0x1800	Trigger Polarity of Digital Input0
		TriggerInput.DI0TrigPolarity = 0x1:Falling / 0x0:Rising
[RW]	0x1801	Trigger Polarity of Digital Input1
		TriggerInput.DI1TrigPolarity = 0x1:Falling / 0x0:Rising
[RW]	0x1802	Trigger Polarity of Digital Input2
		TriggerInput.DI2TrigPolarity = 0x1:Falling / 0x0:Rising

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

[RW]	0x1803	Trigger Polarity of Digital Input3
		TriggerInput.DI3TrigPolarity = 0x1:Falling / 0x0:Rising
[RW]	0x1804	Trigger Polarity of Encoder PairA
		TriggerInput.EnATrigPolarity = 0x1:Falling / 0x0:Rising
[RW]	0x1805	Trigger Polarity of Encoder PairB
		TriggerInput.EnBTrigPolarity = 0x1:Falling / 0x0:Rising
[RW]	0x1806	Digital Debounce Time of input - The debounce filter is turned on or off. DebounceTime = time constant value
		TriggerInput.DebounceTime = 0x0:0 / 0x1:200 / 0x2:500 / 0x3:800 (uint:ns)
[R]	0x180B	Digital Input0 status
		TriggerInput.DI0Status = 0x1:Active / 0x0:Inactive
[R]	0x180C	Digital Input1 status
		TriggerInput.DI1Status = 0x1:Active / 0x0:Inactive
[R]	0x180D	Digital Input2 status
		TriggerInput.DI2Status = 0x1:Active / 0x0:Inactive
[R]	0x180E	Digital Input3 status
		TriggerInput.DI3Status = 0x1:Active / 0x0:Inactive
[R]	0x180F	Number of DI0 trigger event received
		TriggerInput.DI0EventCount = 0 ~ 4294967295 *Note6
[R]	0x1810	Number of DI1 trigger event received
		TriggerInput.DI1EventCount = 0 ~ 4294967295 *Note6
[R]	0x1811	Number of DI2 trigger event received
		TriggerInput.DI2EventCount = 0 ~ 4294967295 *Note6
[R]	0x1812	Number of DI3 trigger event received
		TriggerInput.DI3EventCount = 0 ~ 4294967295 *Note6
[R]	0x1813	Number of Encoder Compare trigger event received
		TriggerInput.EncoderCompareEventCount = 0 ~ 4294967295 *Note6
[R]	0x1814	For the Master, it refers to the number of Multi-card Sync trigger events sent. For the Slave, it refers to the number of Multi-card Sync trigger events received.
		TriggerInput.MulticardSyncEventCount = 0 ~ 4294967295 *Note6
[R]	0x1815	Number of SW trigger event received
		TriggerInput.SWTriggerEventCount = 0 ~ 4294967295 *Note6
[RW]	0x1900	Delay Mode Setting of Digital Output0
		TriggerDelay.DO0DelayMode = 0x0:EncoderMode / 0x1:TimerMode
[RW]	0x1901	Delay Count Setting of Digital Output0
		TriggerDelay.DO0DelayCount = 0 ~ 4294967295 (TimerMode unit: μs , EncoderMode uint count) *Note5
[RW]	0x1902	Delay Mode Setting of Digital Output1
		TriggerDelay.DO1DelayMode = 0x0:EncoderMode / 0x1:TimerMode
[RW]	0x1903	Delay Count Setting of Digital Output1
		TriggerDelay.DO1DelayCount = 0 ~ 4294967295 (TimerMode unit: μs , EncoderMode uint count) *Note5
[RW]	0x1904	Delay Mode Setting of Digital Output2

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

		TriggerDelay.DO2DelayMode = 0x0:EncoderMode / 0x1:TimerMode
[RW]	0x1905	Delay Count Setting of Digital Output2
		TriggerDelay.DO2DelayCount = 0 ~ 4294967295 (TimerMode unit: µs , EncoderMode uint count) *Note5
[RW]	0x1906	Delay Mode Setting of Digital Output3
		TriggerDelay.DO3DelayMode = 0x0:EncoderMode / 0x1:TimerMode
[RW]	0x1907	Delay Count Setting of Digital Output3
		TriggerDelay.DO3DelayCount = 0 ~ 4294967295 (TimerMode unit: µs , EncoderMode uint count) *Note5
[RW]	0x1908	Delay Mode Setting of Port0 CC1
		TriggerDelay.Port0CC1DelayMode = 0x0:EncoderMode / 0x1:TimerMode
[RW]	0x1909	Delay Count Setting of Port0 CC1
		TriggerDelay.Port0CC1DelayCount = 0 ~ 4294967295 (TimerMode unit: µs , EncoderMode uint count) *Note5
[RW]	0x190A	Delay Mode Setting of Port0 CC2
		TriggerDelay.Port0CC2DelayMode = 0x0:EncoderMode / 0x1:TimerMode
[RW]	0x190B	Delay Count Setting of Port0 CC2
		TriggerDelay.Port0CC2DelayCount = 0 ~ 4294967295 (TimerMode unit: µs , EncoderMode uint count) *Note5
[RW]	0x190C	Inverted Polarity of Digital Output0
		TriggerOutput.DO0InvertedPolarity = 0x1:Active / 0x0:Inactive
[RW]	0x190D	Inverted Polarity of Digital Output1
		TriggerOutput.DO1InvertedPolarity = 0x1:Active / 0x0:Inactive
[RW]	0x190E	Inverted Polarity of Digital Output2
		TriggerOutput.DO2InvertedPolarity = 0x1:Active / 0x0:Inactive
[RW]	0x190F	Inverted Polarity of Digital Output3
		TriggerOutput.DO3InvertedPolarity = 0x1:Active / 0x0:Inactive
[RW]	0x1910	High Pulse Width of Digital Output0
		TriggerOutput.DO0HighPulseWidth = 0 ~ 16777215 (uint: µs)
[RW]	0x1911	Low Pulse Width of Digital Output0
		TriggerOutput.DO0LowPulseWidth = 0 ~ 16777215 (uint: µs)
[RW]	0x1912	High Pulse Width of Digital Output1
		TriggerOutput.DO1HighPulseWidth = 0 ~ 16777215 (uint: µs)
[RW]	0x1913	Low Pulse Width of Digital Output1
		TriggerOutput.DO1LowPulseWidth = 0 ~ 16777215 (uint: µs)
[RW]	0x1914	High Pulse Width of Digital Output2
		TriggerOutput.DO2HighPulseWidth = 0 ~ 16777215 (uint: µs)
[RW]	0x1915	Low Pulse Width of Digital Output2
		TriggerOutput.DO2LowPulseWidth = 0 ~ 16777215 (uint: µs)
[RW]	0x1916	High Pulse Width of Digital Output3
		TriggerOutput.DO3HighPulseWidth = 0 ~ 16777215 (uint: µs)
[RW]	0x1917	Low Pulse Width of Digital Output3
		TriggerOutput.DO3LowPulseWidth = 0 ~ 16777215 (uint: µs)

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

[RW]	0x1918	Output Mode setting of Digital Output0
		TriggerOutput.DO0OutputMode = 0x1:Manual / 0x0:Trigger
[RW]	0x1919	Output Mode setting of Digital Output1
		TriggerOutput.DO1OutputMode = 0x1:Manual / 0x0:Trigger
[RW]	0x191A	Output Mode setting of Digital Output2
		TriggerOutput.DO2OutputMode = 0x1:Manual / 0x0:Trigger
[RW]	0x191B	Output Mode setting of Digital Output3
		TriggerOutput.DO3OutputMode = 0x1:Manual / 0x0:Trigger
[R]	0x191C	Number of trigger event received by DO0Delay
		TriggerDelay.DO0TriggerEventCount = 0 ~ 4294967295 *Note6
[R]	0x191D	Number of trigger event dropped by DO0Delay
		TriggerDelay.DO0TriggerDropCount = 0 ~ 4294967295 *Note6
[R]	0x191E	Number of trigger event received by DO1Delay
		TriggerDelay.DO1TriggerEventCount = 0 ~ 4294967295 *Note6
[R]	0x191F	Number of trigger event dropped by DO1Delay
		TriggerDelay.DO1TriggerDropCount = 0 ~ 4294967295 *Note6
[R]	0x1920	Number of trigger event received by DO2Delay
		TriggerDelay.DO2TriggerEventCount = 0 ~ 4294967295 *Note6
[R]	0x1921	Number of trigger event dropped by DO2Delay
		TriggerDelay.DO2TriggerDropCount = 0 ~ 4294967295 *Note6
[R]	0x1922	Number of trigger event received by DO3Delay
		TriggerDelay.DO3TriggerEventCount = 0 ~ 4294967295 *Note6
[R]	0x1923	Number of trigger event dropped by DO3Delay
		TriggerDelay.DO3TriggerDropCount = 0 ~ 4294967295 *Note6
[RW]	0x1924	Digital Output0 Status
		TriggerOutput.DO0Status = 0x1:Active / 0x0:Inactive
[RW]	0x1925	Digital Output1 Status
		TriggerOutput.DO1Status = 0x1:Active / 0x0:Inactive
[RW]	0x1926	Digital Output2 Status
		TriggerOutput.DO2Status = 0x1:Active / 0x0:Inactive
[RW]	0x1927	Digital Output3 Status
		TriggerOutput.DO3Status = 0x1:Active / 0x0:Inactive
[R]	0x1928	Number of trigger event received by DO0TriggerOut
		TriggerOutput.DO0TriggerOutEventCount 0 ~ 4294967295 *Note6
[R]	0x1929	Number of trigger event dropped by DO0TriggerOut
		TriggerOutput.DO0TriggerOutDropCount = 0 ~ 4294967295 *Note6
[R]	0x192A	Number of trigger event received by DO1TriggerOut
		TriggerOutput.DO1TriggerOutEventCount = 0 ~ 4294967295 *Note6
[R]	0x192B	Number of trigger event dropped by DO1TriggerOut
		TriggerOutput.DO1TriggerOutDropCount = 0 ~ 4294967295 *Note6

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

[R]	0x192C	Number of trigger event received by DO2TriggerOut
		TriggerOutput.DO2TriggerOutEventCount = 0 ~ 4294967295 *Note6
[R]	0x192D	Number of trigger event dropped by DO2TriggerOut
		TriggerOutput.DO2TriggerOutDropCount = 0 ~ 4294967295 *Note6
[R]	0x192E	Number of trigger event received by DO3TriggerOut
		TriggerOutput.DO3TriggerOutEventCount = 0 ~ 4294967295 *Note6
[R]	0x192F	Number of trigger event dropped by DO3TriggerOut
		TriggerOutput.DO3TriggerOutDropCount = 0 ~ 4294967295 *Note6
[RW]	0x1A00	Multi-card synchronization setting status
		MultiCardSync.Mode = 0x1:Master / 0x0:Slave
[RW]	0x1B00	Retrigger Enable of Digital Output0
		Retrigger.DO0RetriggerEnable = 0x0:False / 0x1:True
[RW]	0x1B01	Retrigger Mode of Digital Output0
		Retrigger.DO0Mode = 0x0:EncoderMode / 0x1:TimerMode
[RW]	0x1B02	Retrigger Enable of Digital Output1
		Retrigger.DO1RetriggerEnable = 0x0:False / 0x1:True
[RW]	0x1B03	Retrigger Mode of Digital Output1
		Retrigger.DO1Mode = 0x0:EncoderMode / 0x1:TimerMode
[RW]	0x1B04	Retrigger Enable of Digital Output2
		Retrigger.DO2RetriggerEnable = 0x0:False / 0x1:True
[RW]	0x1B05	Retrigger Mode of Digital Output2
		Retrigger.DO2Mode = 0x0:EncoderMode / 0x1:TimerMode
[RW]	0x1B06	Retrigger Enable of Digital Output3
		Retrigger.DO3RetriggerEnable = 0x0:False / 0x1:True
[RW]	0x1B07	Retrigger Mode of Digital Output3
		Retrigger.DO3Mode = 0x0:EncoderMode / 0x1:TimerMode
[RW]	0x1B08	Retrigger Enable of Port0 CC1
		Retrigger.Port0CC1RetriggerEnable = 0x0:False / 0x1:True
[RW]	0x1B09	Retrigger Mode of Port0 CC1
		Retrigger.Port0CC1Mode = 0x0:EncoderMode / 0x1:TimerMode
[RW]	0x1B0A	Retrigger Enable of Port0 CC2
		Retrigger.Port0CC2RetriggerEnable = 0x0:False / 0x1:True
[RW]	0x1B0B	Retrigger Mode of Port0 CC2
		Retrigger.Port0CC2Mode = 0x0:EncoderMode / 0x1:TimerMode
[RW]	0x1B0C	Number of retrigger for Digital Output0
		Retrigger.DO0TriggerNums = 0 ~ 4294967295
[RW]	0x1B0D	Retrigger Interval for Digital Output0
		Retrigger.DO0Interval = 0 ~ 4294967295 (TimerMode unit: μs , EncoderMode uint count) *Note5
[RW]	0x1B0E	Number of retrigger for Digital Output1

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

		Retrigger.DO1TriggerNums = 0 ~ 4294967295
[RW]	0x1B0F	Retrigger Interval for Digital Output1
		Retrigger.DO1Interval = 0 ~ 4294967295 (TimerMode unit: μs , EncoderMode uint count) *Note5
[RW]	0x1B10	Number of retrigger for Digital Output2
		Retrigger.DO2TriggerNums = 0 ~ 4294967295
[RW]	0x1B11	Retrigger Interval for Digital Output
		Retrigger.DO2Interval = 0 ~ 4294967295 (TimerMode unit: μs , EncoderMode uint count) *Note5
[RW]	0x1B12	Number of retrigger for Digital Output3
		Retrigger.DO3TriggerNums = 0 ~ 4294967295
[RW]	0x1B13	Retrigger Interval for Digital Output3
		Retrigger.DO3Interval = 0 ~ 4294967295 (TimerMode unit: μs , EncoderMode uint count) *Note5
[RW]	0x1B14	Number of retrigger for Port0 CC1
		Retrigger.Port0CC1TriggerNums = 0 ~ 4294967295
[RW]	0x1B15	Retrigger Interval for Port0 CC1
		Retrigger.Port0CC1Interval = 0 ~ 4294967295 (TimerMode unit: μs , EncoderMode uint count) *Note5
[RW]	0x1B16	Number of retrigger for Port0 CC2
		Retrigger.Port0CC2TriggerNums = 0 ~ 4294967295
[RW]	0x1B17	Retrigger Interval for Port0 CC2
		Retrigger.Port0CC2Interval = 0 ~ 4294967295 (TimerMode unit: μs , EncoderMode uint count) *Note5
[R]	0x1B18	Number of trigger event received by DO0Retrigger
		Retrigger.DO0RetriggerEventCount = 0 ~ 4294967295 *Note6
[R]	0x1B19	Number of trigger event dropped by DO0Retrigger
		Retrigger.DO0RetriggerDropCount = 0 ~ 4294967295 *Note6
[R]	0x1B1A	Number of trigger event received by DO1Retrigger
		Retrigger.DO1RetriggerEventCount = 0 ~ 4294967295 *Note6
[R]	0x1B1B	Number of trigger event dropped by DO1Retrigger
		Retrigger.DO1RetriggerDropCount = 0 ~ 4294967295 *Note6
[R]	0x1B1C	Number of trigger event received by DO2Retrigger
		Retrigger.DO2RetriggerEventCount = 0 ~ 4294967295 *Note6
[R]	0x1B1D	Number of trigger event dropped by DO2Retrigger
		Retrigger.DO2RetriggerDropCount = 0 ~ 4294967295 *Note6
[R]	0x1B1E	Number of trigger event received by DO3Retrigger

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

		Retrigger.DO3RetriggerEventCount = 0 ~ 4294967295 *Note6
[R]	0x1B1F	Number of trigger event dropped by DO3Retrigger
		Retrigger.DO3RetriggerDropCount = 0 ~ 4294967295 *Note6
[R]	0x1B20	Number of trigger event received by Port0CC1Retrigger
		Retrigger.Port0CC1RetriggerEventCount = 0 ~ 4294967295 *Note6
[R]	0x1B21	Number of trigger event dropped by Port0CC1Retrigger
		Retrigger.Port0CC1RetriggerDropCount = 0 ~ 4294967295 *Note6
[R]	0x1B22	Number of trigger event received by Port0CC2Retrigger
		Retrigger.Port0CC2RetriggerEventCount = 0 ~ 4294967295 *Note6
[R]	0x1B23	Number of trigger event dropped by Port0CC2Retrigger
		Retrigger.Port0CC2RetriggerDropCount = 0 ~ 4294967295 *Note6
[R]	0x1C00	Number of DMA interrupt issued by FPGA
		Debug.DMAInterruptCount = 0 ~ 4294967295 *Note7
[R]	0x1C01	Data Stream Status Confirmation - The board is receiving data from the front-end device - The board is transmitting data to the host
		Debug.DataStreamStatus = 0 ~ 3 *Note1
[R]	0x1C02	Camera Link configuration status - Full / Medium / Base / Error / Not detected
		Debug.LEDStatus = 0 ~ 63 *Note2
[R]	0x1C03	PoCL Status for Port 0 (Base) & Port1 (Medium / Full)
		Debug.PoCLStatus = 0 ~ 3 (Reserved) *Note3
[R]	0x1C04	External device clock frequency detection
		Debug.CameralinkClock = 0 ~ 4294967295 (uint:KHz)
[R]	0x1C07	Number of DO0 Delay Buffer data count
		Debug.DO0DelayBufferCount = 0 ~ 255 *Note7
[R]	0x1C08	DO0 Delay Buffer overflow flag
		Debug.DO0DelayBufferOverflow = 0 ~ 1 *Note7
[R]	0x1C09	Number of DO1 Delay Buffer data count
		Debug.DO1DelayBufferCount = 0 ~ 255 *Note7
[R]	0x1C0A	DO1 Delay Buffer overflow flag
		Debug.DO1DelayBufferOverflow = 0 ~ 1 *Note7
[R]	0x1C0B	Number of DO2 Delay Buffer data count
		Debug.DO2DelayBufferCount = 0 ~ 255 *Note7

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

[R]	0x1C0C	DO2 Delay Buffer overflow flag
		Debug.DO2DelayBufferOverflow = 0 ~ 1 *Note7
[R]	0x1C0D	Number of DO3 Delay Buffer data count
		Debug.DO3DelayBufferCount = 0 ~ 255 *Note7
[R]	0x1C0E	DO3 Delay Buffer overflow flag
		Debug.DO3DelayBufferOverflow = 0 ~ 1 *Note7
[R]	0x1C0F	Number of CC1 Delay Buffer data count
		Debug.Port0CC1DelayBufferCount = 0 ~ 255 *Note7
[R]	0x1C10	CC1 Delay Buffer overflow flag
		Debug.Port0CC1DelayBufferOverflow = 0 ~ 1 *Note7
[R]	0x1C11	Number of CC2 Delay Buffer data count
		Debug.Port0CC2DelayBufferCount = 0 ~ 255 *Note7
[R]	0x1C12	CC2 Delay Buffer overflow flag
		Debug.Port0CC2DelayBufferOverflow = 0 ~ 1 *Note7
[R]	0x1C13	Number of lines received by FPGA
		Debug.Port0LineCount = 0 ~ 4294967295 *Note7
[R]	0x1C15	Number of frames received by FPGA
		Debug.Port0FrameCount = 0 ~ 4294967295 *Note7
[RW]	0x1C19	The PoCLSafePowerStatus includes the Clock Start Time, Clock Drop Time and Voltage Below Flag.
		Debug.PoCLSafePowerStatus = 0 ~ 7 (Reserved) *Note4
[RW]	0x1D00	Allows a PoCL device to extend its clock startup time beyond the default limit.
		PoCL.ClockStartupTimeTolerance = 0 ~ 3,000,000 unit: µs (Reserved) *Note5
[RW]	0x1D01	Allows a PoCL device to extend its clock drop out time beyond the default limit.
		PoCL.ClockDropOutTimeTolerance = 0 ~ 100,000 unit: µs (Reserved) *Note5
[RW]	0x1D02	Allows a PoCL device to extend its voltage drop duration beyond the default limit.
		PoCL.VoltageDropDurationTolerance = 0 ~ 20,000 unit: µs (Reserved) *Note5

Note1: 0x1C01 to read Debug.DataStreamStatus

Bit	Description
0	1 = cpl64v board device is receiving data from the front end device
1	1 = cpl64v board is transmitting data to the host

Note2: 0x1C02 to read Debug.LEDStatus (Camera Link configuration status & LED status)

Bit	Description
0	1 = Full camera mode is detected. LED is Green.
1	1 = Medium camera mode is detected. LED is Green blinking (2Hz)
2	1 = Base camera mode is detected. LED is Orange

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3	1 = Abnormal. LED is Orange blinking (2Hz)
4	1 = Camera link not detected. LED is OFF
5	0/1 = idle/capture period

Note3: 0x1C03 to read Debug.PoCLStatus

Bit	Description
0	Port 0 POCL active: 1: powering (for base)
1	Port 1 PoCL active: 1: powering (for medium and full)

Note4: 0x1C19 to read SafePower Status

Bit	Description
0	1 = Clock Start Time Flag Error
1	1 = Clock Drop Time Flag Error
2	1 = Voltage Below Flag Error

If 0x1C19 to the returned value is 1, you need to manually adjust the corresponding parameters 0x1D00~ 0x1D02

Note5: TimerMode and EncoderMode limitation, If TimerMode is selected, the effective bits are 24 bits, and the value is.

Mode	Range
TimerMode	0~16777215μs
EncoderMode	0 ~ 4294967295 count

Note6:
These Address are used for status tracking and are responsible for recording the trigger counts for each trigger path. The drop count should be zero, indicating that no trigger signals have been lost.

Address
0x1308, 0x1309, 0x130A, 0x130B, 0x130C, 0x130D, 0x130E, 0x130F, 0x180F, 0x1810, 0x1811, 0x1812, 0x1813, 0x1814, 0x1815, 0x191C, 0x191D, 0x191E, 0x191F, 0x1920, 0x1921, 0x1922, 0x1923, 0x1928, 0x1929, 0x192A, 0x192B, 0x192C, 0x192D, 0x192E, 0x192F, 0x1B18, 0x1B19, 0x1B1A, 0x1B1B, 0x1B1C, 0x1B1D, 0x1B1E, 0x1B1F, 0x1B20, 0x1B21, 0x1B22, 0x1B23

Information: The values read from the above address will be cleared by ResetDebugCount.(0x1600)

Note7:
The values read from the address will be cleared by AVS_FrameGrabberStartAquisition

Note8: (Image Buffer Declaration Requirements)
When allocating an image buffer, note the following alignment constraints.
The **total frame size must always be a multiple of 32 bytes**, and depending on pixel format and tap configuration, the **image width must satisfy specific alignment rules**.

General Requirement

Frame size requirement:
The total size of one frame must be a multiple of 32 bytes.

Width Alignment by Pixel Format and Tap Count

- **MONO 8-bit**

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

10-tap:
Width must be a multiple of 160 pixels. (160bytes)

1-tap / 2-tap / 4-tap / 8-tap:
Width must be a multiple of 32 pixels. (32bytes)

- **MONO 10-bit(unpacked)**

  8-tap:
  Width must be a multiple of 320 pixels. (640bytes)

  1-tap / 2-tap / 4-tap:
  Width must be a multiple of 16 pixels. (32bytes)

- **MONO 12-bit(unpacked)**

  1-tap / 2-tap / 4-tap:
  Width must be a multiple of 16 pixels. (32bytes)

- **RGB 8-bit(unpacked)**

  1-tap / 2-tap:
  Width must be a multiple of 8 pixels. (32bytes)

- RGB 12-bit(unpacked)

  1-tap:
  Width must be a multiple of 4 pixels. (32bytes)

4.1.2. CameraLink File

The CameraLink feature allows you to import a camfile to configure the device. Below are the definitions for each relevant product. One seting description will be like the following in a camfile:

[Section]
Key=value

4.1.2.1. PCIe-CPL64V Full

[CaptureMode]
ScanMode = AreaScan / LineScan

[ImageFormat]
SensorWidth = 32 ~ 65504
SensorHeight = 0 ~ 65535
Tap = 1 / 10 / 2 / 4 / 8
PixelSize = 10 / 12 / 8
TapPlacement = 1x / 1x10 / 1x2 / 1x4 / 1x8

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

PixelFormat = Mono / RGB
FVALEnable = False / True
LVALEnable = False / True
DVALEnable = False / True
FVALPolarity = Falling / Rising
LVALPolarity = Falling / Rising
DVALPolarity = Falling / Rising

[CameraControl]
Port0CC1HighPulseWidth = 0 ~ 16777215
Port0CC1LowPulseWidth = 0 ~ 16777215
Port0CC2HighPulseWidth = 0 ~ 16777215
Port0CC2LowPulseWidth = 0 ~ 16777215
Port0CC1InvertedPolarity = Active / Inactive
Port0CC2InvertedPolarity = Active / Inactive
Port0CC1OutputMode = Manual / Trigger
Port0CC2OutputMode = Manual / Trigger

[EncoderInput]
Mode = ABphaseX1 / ABphaseX2 / ABphaseX4 / CW/CCW / OUT/DIR
CompareCount = 0 ~ 4294967295

[MultiCardTrigSrc]
TriggerSource = DI0 / DI1 / DI2 / DI3 / Disable / EncoderComparer / MultiCardSync / SoftwareTrigger

[TriggerOutputTrigSrc]
DO0TriggerSource = DI0 / DI1 / DI2 / DI3 / Disable / EncoderComparer / MultiCardSync / SoftwareTrigger
DO1TriggerSource = DI0 / DI1 / DI2 / DI3 / Disable / EncoderComparer / MultiCardSync / SoftwareTrigger
DO2TriggerSource = DI0 / DI1 / DI2 / DI3 / Disable / EncoderComparer / MultiCardSync / SoftwareTrigger
DO3TriggerSource = DI0 / DI1 / DI2 / DI3 / Disable / EncoderComparer / MultiCardSync / SoftwareTrigger
Port0CC1TriggerSource = DI0 / DI1 / DI2 / DI3 / Disable / EncoderComparer / MultiCardSync / SoftwareTrigger
Port0CC2TriggerSource = DI0 / DI1 / DI2 / DI3 / Disable / EncoderComparer / MultiCardSync / SoftwareTrigger

[TriggerInput]
DI0TrigPolarity = Falling / Rising
DI1TrigPolarity = Falling / Rising
DI2TrigPolarity = Falling / Rising
DI3TrigPolarity = Falling / Rising
EnATrigPolarity = Falling / Rising
EnBTrigPolarity = Falling / Rising
DebounceTime = 0 / 200 / 500 / 800

[TriggerDelay]
DO0DelayMode = EncoderMode / TimerMode
DO0DelayCount = 0 ~ 4294967295
DO1DelayMode = EncoderMode / TimerMode
DO1DelayCount = 0 ~ 4294967295
DO2DelayMode = EncoderMode / TimerMode
DO2DelayCount = 0 ~ 4294967295

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

DO3DelayMode = EncoderMode / TimerMode
DO3DelayCount = 0 ~ 4294967295
Port0CC1DelayMode = EncoderMode / TimerMode
Port0CC1DelayCount = 0 ~ 4294967295
Port0CC2DelayMode = EncoderMode / TimerMode
Port0CC2DelayCount = 0 ~ 4294967295

[TriggerOutput]
DO0InvertedPolarity = Active / Inactive
DO1InvertedPolarity = Active / Inactive
DO2InvertedPolarity = Active / Inactive
DO3InvertedPolarity = Active / Inactive
DO0HighPulseWidth = 0 ~ 16777215
DO0LowPulseWidth = 0 ~ 16777215
DO1HighPulseWidth = 0 ~ 16777215
DO1LowPulseWidth = 0 ~ 16777215
DO2HighPulseWidth = 0 ~ 16777215
DO2LowPulseWidth = 0 ~ 16777215
DO3HighPulseWidth = 0 ~ 16777215
DO3LowPulseWidth = 0 ~ 16777215
DO0OutputMode = Manual / Trigger
DO1OutputMode = Manual / Trigger
DO2OutputMode = Manual / Trigger
DO3OutputMode = Manual / Trigger

[MultiCardSync]
Mode = Master / Slave

[Retrigger]
DO0RetriggerEnable = False / True
DO0Mode = EncoderMode / TimerMode
DO1RetriggerEnable = False / True
DO1Mode = EncoderMode / TimerMode
DO2RetriggerEnable = False / True
DO2Mode = EncoderMode / TimerMode
DO3RetriggerEnable = False / True
DO3Mode = EncoderMode / TimerMode
Port0CC1RetriggerEnable = False / True
Port0CC1Mode = EncoderMode / TimerMode
Port0CC2RetriggerEnable = False / True
Port0CC2Mode = EncoderMode / TimerMode
DO0TriggerNums = 0 ~ 4294967295
DO0Interval = 0 ~ 4294967295
DO1TriggerNums = 0 ~ 4294967295
DO1Interval = 0 ~ 4294967295
DO2TriggerNums = 0 ~ 4294967295
DO2Interval = 0 ~ 4294967295
DO3TriggerNums = 0 ~ 4294967295
DO3Interval = 0 ~ 4294967295
Port0CC1TriggerNums = 0 ~ 4294967295
Port0CC1Interval = 0 ~ 4294967295
Port0CC2TriggerNums = 0 ~ 4294967295
Port0CC2Interval = 0 ~ 4294967295

[PoCL] (Reserved)
ClockDropOutTimeTolerance = = 0 ~ 100000
ClockStartupTimeTolerance = 0 ~ 3000000

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

VoltageDropDurationTolerance = 0 ~ 20000

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.3. AVS_GetProductNums

Retrieve the number of the scanned product.

Syntax

C/C++

unsigned int AVS_GetProductNums()

Parameter(s)

Return Value

Return the number of the scanned product.

4.1.4. AVS_GetSDKInfo

Retrieve the SDK information.

Syntax

C/C++

int AVS_GetSDKInfo(char* data, unsigned int bufsize)

Parameter(s)

data

Need assign an allocated buffer to read the information of the SDK. The information format is:

Driver version:[Version]
SDK version:[Version]

Each device description is split by break line character "\r\n"

bufsize

The number of "data" bytes. The suggested size is 256.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.5. AVS_GetProductHandle

Retrieve the hadle of a product.

Syntax

C/C++

void* AVS_GetProductHandle(unsigned int index)

Parameter(s)

index

Assign a zero-based index value. The value can't bigger than maximum number of the scanned product.

Return Value

Return non-zero handle if success, or null if failed. This handle can be used to access product.

4.1.6. AVS_GetProductInfo

Retrieve the product information by a handle.

Syntax

C/C++

int AVS_GetProductInfo(void* handle, char* data, unsigned int bufsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

data

Need assign an allocated buffer to read the information of the device. These device information format is:

[Information type]: [Value]
...

Each device description is split by break line character "\r\n"

bufsize

The number of “data” bytes. The suggested size is 256.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.7. AVS_GetProductType

Retrieve the product type by a handle.

Syntax

C/C++

int AVS_GetProductType(void* handle)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

Return Value

Return AVS_PRODUCT_TYPE , please check the defined value in the I_common.h.

4.1.8. AVS_GetProductSN

Retrieve the product serial number (SN) by a handle.

Syntax

C/C++

int AVS_GetProductSN(void* handle, char* data, unsigned int bufsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

data

Need assign an allocated buffer to read the SN of the device.

bufsize

The number of “data” bytes. The suggested size is 32.

Return Value

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.9. AVS_GetFeatureList

Retrieve the feature list by a handle.

Syntax

C/C++

int AVS_GetFeatureList(void* handle, AVS_FEATURE_TYPE* list, unsigned int listsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

list

Need assign an allocated AVS_FEATURE_TYPE list to read the features of the device.

listsize

The number of"list" elements. The suggested size is AVS_FEATURE_NUMS.

Return Value

Return the number of the features.

4.1.10. AVS_GetCardID

Retrieve the card id of a product.

Syntax

C/C++

int AVS_GetCardID(void* handle, int* id)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

id

Return the card id of a product.

Return Value

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.16 Feature – PowerController

This feature is use to control the power device.

4.1.11. AVS_PowCtrlGetPortNums

Retrieve the number of the power ports.

Syntax

C/C++

int AVS_PowCtrlGetPortNums(void* handle, unsigned int* nums)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

nums

Return the number of the power ports.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.12. AVS_ PowCtrlSetPortEnable

Enable / disable the power ports.

Syntax

C/C++

int AVS_PowCtrlSetPortEnable(void* handle, int* list, unsigned int listsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

list

Assign a list to enable or disable each port.
Assign 0 to disable power or other to enable power.

listsize

The number of"list" elements. The suggested list size could be got from AVS_PowCtrlGetPortNums()

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.13. AVS_ PowCtrlGetPortEnable

Get the enable states of the power ports.

Syntax

C/C++

int AVS_PowCtrlGetPortEnable(void* handle, int* list, unsigned int listsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

list

Assign a list to get the enabled states of each port.
0 is disabled power or other is enabled power.

listsize

The number of"list" elements. The suggested list size could be got from AVS_PowCtrlGetPortNums()

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.17 Feature – External Power Supply

These functions provide the special functions of U3 series product.

4.1.14. AVS_ExPowSupGetNums

Retrieve the number of the external power supplies.

Syntax

C/C++

int AVS_ExPowSupGetNums(void* handle, unsigned int* nums)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

nums

Return the number of the external power supplies.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.15. AVS_ExPowSupGetState

Retrieve the states of the external power supplies.

Syntax

C/C++

int AVS_ExPowSupGetState(void* handle, int* statelist, unsigned int listsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

statelist

Assign a list to retrieve the external power supply states of each port.
0 is without power supply, the other is with power supply

listsize

The number of statelist elements. The suggested list size could be got from AVS_ExPowSupGetNums()

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.18 Feature – Power over Ethernet

These functions provide the PoE functions.

4.1.16. AVS_PoEGetPortNums

Retrieve the number of the power over ethernet.

Syntax

C/C++

int AVS_PoEGetPortNums(void* handle, unsigned int* nums)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

nums

Return the number of the power over ethernet.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.17. AVS_PoESetPortEnable

Enable / disable the power ports.

Syntax

C/C++

int AVS_PoESetPortEnable(void* handle, int* list, unsigned int listsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

list

Assign a list to enable or disable each port.
* 0 to disable power or other to enable power.

listsize

The number of¨list¨ elements. The suggested list size could be got from AVS_PowCtrlGetPortNums()

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.18. AVS_PoEGetPortEnable

Get the enable states of the power ports.

Syntax

C/C++

int AVS_PoEGetPortEnable(void* handle, int* list, unsigned int listsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

list

Assign a list to retrieve the enable state of each port.
* 0 to disable power or other to enable power.

listsize

The number of"list" elements. The suggested list size could be got from AVS_PowCtrlGetPortNums()

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.19. AVS_PoESetPowConsumCalcModel

Set the calculation properties of the power consumption, if set usePowClassBudget, the power consumption will be calculated by power class budget. Otherwise it will be calculated by the real consumption.

Syntax

C/C++

int AVS_PoESetPowConsumCalcModel(void* handle, int usePowClassBudget)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

usePowClassBudget

use the pow class budget to calculate the power consumption, if set 0, the consumption will be calculated by real consumption

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.20. AVS_PoEGetPowConsumCalcModel

Get the calculation properties of the power consumption, if usePowClassBudget = 1, the power consumption will be calculated by power class budget. Otherwise it will be calculated by the real consumption.

Syntax

C/C++

int AVS_PoEGetPowConsumCalcModel(void* handle, int* usePowClassBudget)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

usePowClassBudget

Use the power class budget to calculate the power consumption, if the value is 0, the consumption will be calculated by real consumption.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.21. AVS_PoESetProtectTempRange

Set the protected temperature range. (unit : centigrade degree)

Syntax

C/C++

int AVS_PoESetProtectTempRange(void* handle, int high_temperature, int recovery_temperature)

Parameter(s)

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

handle

Assign a product handle. Get from AVS_GetProductHandle()

high_temperature

Assign a high protected temperature.
The power will be closed when the temperature is over the value. (Maximum is 150)

recovery_temperature

Assign a recovery temperature.
After the temperature is over the high_temperature once,
when the temperature is cool down to this value, the power will be recovered.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer
AVS_FUN_RETURNVAL in the I_common.h.

4.1.22. AVS_PoEGetProtectTempRange

Get the protected temperature range. (unit : centigrade degree)

Syntax

C/C++

int AVS_PoESetProtectTempRange(void* handle, int high_temperature, int recovery_temperature)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

high_temperature

Retrieve the high protected temperature.
The power will be closed when the temperature is over the value. (Maximum is 150)

recovery_temperature

Retrieve the recovery temperature.
After the temperature is over the high_temperature once,
when the temperature is cool down to this value, the power will be recovered.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer
AVS_FUN_RETURNVAL in the I_common.h.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.23. AVS_PoEGetProperty

Get the PoE property

Syntax

C/C++

int AVS_PoEGetProperty(void* handle, AVS_POEPROPERTY_TYPE property, float* value)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

property

refer AVS_POEPROPERTY_TYPE:
* AVS_POEPROPERTY_POEHWSETTING : PSE Hardware setting.
* AVS_POEPROPERTY_POEINITIALSTATE : PSE initial power.
* AVS_POEPROPERTY_OVERTEMPERATURE : Check if the device temperature is over high temperature.
* AVS_POEPROPERTY_CURRENTTEMPERATURE : Retrieve the current temperature. (unit : centigrade degree)
* AVS_POEPROPERTY_CONSUMEPOWER : Retrieve the comsumer power. (unit : watt)
* AVS_POEPROPERTY_REMAINPOWER : Retrieve the remaining power. (unit : watt)
* AVS_POEPROPERTY_POWERBUDGET : Retrieve the total power budget. (unit : watt)

value

Retrieve the value of the property

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.
If the property is not support will be return AVS_FUN_INVAILDPAR

4.1.24. AVS_PoEGetPortProperty

Get the PoE port property

Syntax

C/C++

int AVS_PoEGetPortProperty(void* handle, unsigned int port, AVS_POEPORTPROPERTY_TYPE property, float* value)

Parameter(s)

handle

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Assign a product handle. Get from AVS_GetProductHandle()

port

Assign a port index, zero-based value.

property

refer AVS_POEPORTPROPERTY_TYPE :
* AVS_POEPORTPROPERTY_VOLT : Retrieve the voltage.
* AVS_POEPORTPROPERTY_CURRENT: Retrieve the electric current. (unit : A)
* AVS_POEPORTPROPERTY_POWERGOOD: Retrieve the output power. (unit : watt)
* AVS_POEPORTPROPERTY_POECLASS : Retrieve the power class. 0 is class 0, 1 is class 1 and so on. If the value is smaller than 0, it is not a valid value.
* AVS_POEPORTPROPERTY_LASTPOWERSTATE : Retrieve the last state of a poe port, 1 is power on, 0 is power off.

value

Retrieve the value of the property

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.
If the property is not support will be return AVS_FUN_INVAILDPAR

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.19 Feature – Trigger over Ethernet

These functions provide the ToE functions. The following is the set up steps:

1. Select a trigger source by using **AVS_ToESetTriggerSource**
2. Set up the action command by using **AVS_ToESetActionCommand**
3. Enable trigger by using **AVS_ToESetTriggerEnable**
4. Send the trigger command by using **AVS_ToESendActionCommand**
5. Check the trigger counts for confirming the command have been sent by using **AVS_ToEGetTriggerCount**

4.1.25. AVS_ToEGetPortNums

Retrieve the number of the trigger over ethernet.

Syntax

C/C++

int AVS_ToEGetPortNums(void* handle, unsigned int* nums)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

nums

Return the number of the trigger over ethernet.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.26. AVS_ToESendActionCommand

Send a toe command to ToE chip

Syntax

C/C++

int AVS_ToESendActionCommand(void* handle, unsigned int port)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

port

Specify a port to send a action command.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.27. AVS_ToESetActionCommand

Set a toe command, it will be send out when trigger the corresponding port. Please make sure you have already assigned an ip and the specified port is connected.

Syntax

C/C++

int AVS_ToESetActionCommand(void* handle, unsigned int port, unsigned long devicekey, unsigned long groupkey, unsigned long groupmask)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

port

Specify a port to send an action command.

devicekey

Specify a deivcekey.

groupkey

Specify a groupkey

groupmask

Specify a groupmask.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.28. AVS_ToEGetActionCommand

Get the toe command to ToE chip.

Syntax

C/C++

int AVS_ToEGetActionCommand(void* handle, unsigned int port, unsigned long* devicekey, unsigned long* groupkey, unsigned long* groupmask)

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

port

Specify a port to send an action command.

devicekey

Retrieve a deivcekey.

groupkey

Retrieve a groupkey

groupmask

Retrieve a groupmask.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.29. AVS_ToESetTriggerSource

Set the trigger source which could be software or hardware.

Syntax

C/C++

int AVS_ToESetTriggerSource(void* handle, int source)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

source

Set the triiger source. (0 is software, 1 is external hardware)

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.30. AVS_ToEGetTriggerSource

Get the trigger source which could be software or hardware.

Syntax

C/C++

int AVS_ToEGetTriggerSource(void* handle, int* source)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

source

Get the triiger source. (0 is software, 1 is external hardware)

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.31. AVS_ToESetTriggerEnable

Set the enable states of the toe trigger.

Syntax

C/C++

int AVS_ToESetTriggerEnable(void* handle, int enable)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

enable

Set the enable status of the ToE trigger.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.32. AVS_ToEGetTriggerEnable

Get the enable states of the toe trigger.

Syntax

C/C++

int AVS_ToEGetTriggerEnable(void* handle, int* enable)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

enable

Get the enable status of the ToE trigger.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.33. AVS_ToESetTriggerActivation

Set the edge trigger mode for all ToE port.

Syntax

C/C++

int AVS_ToESetTriggerActivation(void* handle, AVS_EDGETRIGGER_TYPE edgetrigger)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

edgetrigger

Set the edge trigger mode AVS_EDGETRIGGER_TYPE.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.34. AVS_ToEGetTriggerActivation

Get the edge trigger mode for all ToE port.

Syntax

C/C++

int AVS_ToEGetTriggerActivation(void* handle, AVS_EDGETRIGGER_TYPE* edgetrigger)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

edgetrigger

Get the edge trigger mode AVS_EDGETRIGGER_TYPE.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.35. AVS_ToESetOneTriggerAllMode

Set enable status of the one trigger input can trigger all trigger output. If disable this mode, one input will trigger related trigger output (such as di0 trigger do0, di1 trigger do1 ...)

Syntax

C/C++

int AVS_ToESetOneTriggerAllMode(void* handle, int enable)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

enable

Set the enable state of the one trigger all mode.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.36. AVS_ToEGetOneTriggerAllMode

Set enable status of the one trigger input can trigger all trigger output. If disable this mode, one input will trigger related trigger output (such as di0 trigger do0, di1 trigger do1 ...)

Syntax

C/C++

int AVS_ToEGetOneTriggerAllMode(void* handle, int* enable)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

enable

Get the enable state of the one trigger all mode.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.37. AVS_ToESetExTriggerDebounce

Set the hardware trigger debounce (Unit is got from AVS_ToEGetExTriggerDebounceUnit())

Syntax

C/C++

int AVS_ToESetExTriggerDebounce(void* handle, unsigned long debounce)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

debounce

Set the debounce time, input range: 0x0 ~ 0xFFFFFF.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.38. AVS_ToEGetExTriggerDebounce

Get the hardware trigger debounce (Unit is got from AVS_ToEGetExTriggerDebounceUnit())

Syntax

C/C++

int AVS_ToEGetExTriggerDebounce(void* handle, unsigned long* debounce)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

debounce

Get the debounce time, range: 0x0 ~ 0xFFFFFF.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.39. AVS_ToEGetExTriggerDebounceUnit

Get the hardware trigger debounce unit, the time unit is nano second

Syntax

C/C++

int AVS_ToEGetExTriggerDebounceUnit(void* handle, unsigned long* nanosec)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

nanosec

The number of the nano seconds.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.40. AVS_ToEGetTriggerCount

Get the number of trigger counts.

Syntax

C/C++

int AVS_ToEGetTriggerCount(void* handle, unsigned int port, unsigned int* triggerincount, unsigned int* triggeroutcount)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

port

Specify a port to get counts.

triggerincount

The number of the trigger input.

triggeroutcount

The number of the sending ToE commands.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.41. AVS_ToEResetTriggerCount

Reset the all trigger counts.

Syntax

C/C++

int AVS_ToEResetTriggerCount(void* handle)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.110 Feature – Fan

This feature is use to control the fans.

4.1.42. AVS_FanGetNums

Retrieve the number of the fans.

Syntax

C/C++

int AVS_FanGetNums (void* handle, unsigned int* nums)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

nums

Return the number of the fans.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.43. AVS_FanGetSpeed

Retrieve the speed of the fans. (Revolution Per Minute).

Syntax

C/C++

int AVS_FanGetSpeed(void* handle, unsigned int index, unsigned int* rpm)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

index

Assign an index of the fan

rpm

Retrieve the RPM of a specified fan.

Return Value

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.44. AVS_FanGetPWM

Retrieve the PWM duty cycle of a specified fan..

Syntax

C/C++

int AVS_FanGetPWM(void* handle, unsigned int index, int* dutycycle)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

index

Assign an index of the fan

dutycycle

Retrieve the Duty cycle of a specified fan. (0~100%)

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.45. AVS_FanSetPWM

Set the PWM duty cycle of specified fan.

Syntax

C/C++

int AVS_FanSetPWM(void* handle, unsigned int index, int dutycycle)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

index

Assign an index of the fan

dutycycle

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Assign a Duty cycle for a specified fan. (0~100%)

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.46. AVS_FanGetSmartFanEnable

Retrieve the enable state of the smart fan function

Syntax

C/C++

int AVS_FanGetSmartFanEnable(void* handle, unsigned int index, int* enable)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

index

Assign an index of the fan

enable

The enable state of the smart fan function

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.47. AVS_FanSetSmartFanEnable

Set the enable state of the smart fan function.

Syntax

C/C++

int AVS_FanSetSmartFanEnable(void* handle, unsigned int index, int enable)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

index

Assign an index of the fan

enable

Set the enable state of the smart fan function

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.111 Feature – DIO

This feature is use to control the DIO.
Some of devices can use its DI to directly trigger ToE through its hardware or software. In these case, you can find the ToE trigger could be as a DO in the SDK.
For details, you can check the manual of each product.

4.1.48. AVS_DIOGetCapability

Retrieve the capabilities of the DIO.

Syntax

C/C++

int AVS_DIOGetCapability (void handle, AVS_DIOCAPABILITY_TYPE capList, unsigned int listsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

capList

Assign a list to retrieve the capabilities.

listsize

The number of list elements. The suggested list size is AVS_DIOCAPABILITY_NUMS.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.49. AVS_DIOGetDINums

Retrieve the number of the DI.

Syntax

C/C++

int AVS_DIOGetDINums(void* handle, unsigned int* nums)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

Nums

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Return the number of the DI.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.50. AVS_DIOGetDONums

Retrieve the number of the DO.

Syntax

C/C++

int AVS_DIOGetDONums(void* handle, unsigned int* nums)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

Nums

Return the number of the DO.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.51. AVS_DIOGetDICapability

Retrieve the capabilities of the DI.

Syntax

C/C++

int AVS_DIOGetDICapability(void* handle, unsigned int channel, AVS_DIODICAPABILITY_TYPE* capList, unsigned int listsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Assign a channel index.

capList

Assign a list to retrieve the capabilities of each channel.
The value is defined by AVS_DIODICAPABILITY_TYPE

listsize

The number of list elements. The suggested list size is AVS_DIODICAPABILITY_NUMS.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.52. AVS_DIOGetDOCapability

Retrieve the capabilities of the DO.

Syntax

C/C++

int AVS_DIOGetDOCapability(void* handle, unsigned int channel, AVS_DIODOCAPABILITY_TYPE* capList, unsigned int listsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

capList

Assign a list to retrieve the capabilities of each channel.
The value is defined by AVS_DIODOCAPABILITY_TYPE

listsize

The number of list elements. The suggested list size is AVS_DIODOCAPABILITY_NUMS.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.53. AVS_DIOSetDOState

Set the state of the DO. The function is supported only when AVS_DIODOCAPABILITY_SET is in the list which is got from AVS_DIOGetDOCapability()

Syntax

C/C++

int AVS_DIOSetDOState(void* handle,unsigned int channel, int state)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

state

The state of DO, it should be 0 or 1

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.54. AVS_DIOSetDOStates

Set all the state of the DO. If a channel don't support SET function, that channel will be skipped. The function is supported only when AVS_DIODOCAPABILITY_SET is in the list which is got from AVS_DIOGetDOCapability()

Syntax

C/C++

int AVS_DIOSetDOStates(void* handle, int* statelist, unsigned int listsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

statelist

Assign a list to set all the DO states. The value is 0 or 1

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

listsize

The number of list elements. The suggested list size could be got from AVS_DIOGetDONums()

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.55. AVS_DIOGetDOState

Retrieve the state of the DO. The function is supported only when AVS_DIODOCAPABILITY_GET is in the list which is got from AVS_DIOGetDOCapability()

Syntax

C/C++

int AVS_DIOGetDOState(void* handle,unsigned int channel, int* state)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

state

The current state of DO, it should be 0 or 1

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.56. AVS_DIOGetDOStates

Retrieve all the state of the DO. The function is supported only when AVS_DIODOCAPABILITY_GET is in the list which is got from AVS_DIOGetDOCapability()

Syntax

C/C++

int AVS_DIOGetDOStates(void* handle, int* statelist, unsigned int listsize)

Parameter(s)

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

statelist

Assign a list to retrieve all the DO states.
* The value is 0 or 1 or -1 (Not support)

listsize

The number of list elements. The suggested list size could be got from AVS_DIOGetDONums()

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.57. AVS_DIOGetDIState

Retrieve the state of the DI. The function is supported only when AVS_DIODICAPABILITY_GET is in the list which is got from AVS_DIOGetDICapability()

Syntax

C/C++

int AVS_DIOGetDIState(void* handle,unsigned int channel, int* state)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

state

The current state of DI, it should be 0 or 1

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.58. AVS_DIOGetDIStates

Retrieve all the state of the DI. The function is supported only when AVS_DIODICAPABILITY_GET is in the list which is got from AVS_DIOGetDICapability()

Syntax

C/C++

int AVS_DIOGetDIStates(void* handle, int* statelist, unsigned int listsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

statelist

Assign a list to retrieve all the DI states.
The value is 0 or 1

listsize

The number of list elements. The suggested list size could be got from AVS_DIOGetDINums()

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.59. AVS_DIOSetDIDebounceTime

Set the debounce time of the DI. The unit please refer AVS_DIOGetDIDebounceTimeUnit(). The function is supported only when AVS_DIODICAPABILITY_DEBOUNCETIME is in the list which is got from AVS_DIOGetDICapability()

Syntax

C/C++

int AVS_DIOSetDIDebounceTime(void* handle, unsigned int channel, unsigned long units)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

channel

Assign a channel index.

units

The number of unit.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.60. AVS_DIOGetDIDebounceTime

Retrieve the debounce time of the DI. The unit please refer AVS_DIOGetDIDebounceTimeUnit(). The function is supported only when AVS_DIODICAPABILITY_DEBOUNCETIME is in the list which is got from AVS_DIOGetDICapability()

Syntax

C/C++

int AVS_DIOGetDIDebounceTime(void* handle, unsigned int channel, unsigned long* units)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

units

The number of unit.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.61. AVS_DIOGetDIDebounceTimeUnit

Retrieve how many nano seconds the DI debounce time unit is. The function is supported only when AVS_DIODICAPABILITY_DEBOUNCETIME is in the list which is got from AVS_DIOGetDICapability()

Syntax

C/C++

int AVS_DIOGetDIDebounceTimeUnit(void* handle, unsigned long* ns)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

ns

The nano seconds.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.62. AVS_DIOSetDOInvertState

Set the enable state of the DO inverted output. The function is supported only when AVS_DIODOCAPABILITY_INVERTSTATE is in the list which is got from AVS_DIOGetDOCapability()

Syntax

C/C++

int AVS_DIOSetDOInvertState(void* handle, unsigned int channel, int enable)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

enable

Enable/disable the inverted state

Return Value

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.63. AVS_DIOGetDOInvertState

Retrieve the enable state of the DO inverted output. The function is supported only when AVS_DIODOCAPABILITY_INVERTSTATE is in the list which is got from AVS_DIOGetDOCapability()

Syntax

C/C++

int AVS_DIOGetDOInvertState(void* handle, unsigned int channel, int* enable)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

enable

Enable/disable the inverted state

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.64. AVS_DIOSetDITriggerEnable

Enable/Disable the DI trigger. The function may be supported when AVS_DIODICAPABILITY_TYPE::AVS_DIODICAPABILITY_TRIGGEROFFSWITCH is in the list which is got from getDICapability()

Syntax

C/C++

int AVS_DIOSetDITriggerEnable(void* handle, unsigned int channel, unsigned int enable)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

channel

Assign a channel index.

enable

1:Enable, 0:Disable

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.65. AVS_DIOGetDITriggerEnable

Retrieve the enable state of the DO inverted output. The function is supported only when AVS_DIODOCAPABILITY_INVERTSTATE is in the list which is got from AVS_DIOGetDOCapability()

Syntax

C/C++

int AVS_DIOGetDITriggerEnable(void* handle, unsigned int channel, unsigned int* enable)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

enable

1:Enable, 0:Disable

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.66. AVS_DIOSetDOTriggerEnable

Enable/Disable the DO trigger. The function may be supported when AVS_DIODICAPABILITY_TYPE::AVS_DIODOCAPABILITY_TRIGGEROFFSWITCH is in the list which is got from getDICapability()

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Syntax

C/C++

int AVS_DIOSetDOTriggerEnable(void* handle, unsigned int channel, unsigned int enable)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

enable

1:Enable, 0:Disable

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.67. AVS_DIOGetDOTriggerEnable

Retrieve the enable state of a DO trigger. The function is supported only when AVS_DIODICAPABILITY_TYPE::AVS_DIODOCAPABILITY_TRIGGEROFFSWITCH is in the list which is got from getDICapability()

Syntax

C/C++

int AVS_DIOGetDOTriggerEnable(void* handle, unsigned int channel, unsigned int* enable)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

enable

1:Enable, 0:Disable

Return Value

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.68. AVS_DIOSetDITriggerCallback

Set the callback function. When DI is triggered, the function will be executed. The function may be supported when AVS_DIODICAPABILITY_TRIGGERDETECTION is in the list which is got from AVS_DIOGetDICapability()

Syntax

C/C++

int AVS_DIOSetDITriggerCallback(void* handle, AVS_DITRIGGER_CALLBACK callback)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

callback

Assign a callback function pointer.
typedef void(* AVS_DITRIGGER_CALLBACK) (unsigned int index, int currentState)

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.69. AVS_DIOSetDIEdgeTrigger

Set the edge trigger type of the DI. The function is supported only when AVS_DIODICAPABILITY_EDGETIRGGERTYPE is in the list which is got from AVS_DIOGetDICapability()

Syntax

C/C++

int AVS_DIOSetDIEdgeTrigger(void* handle, unsigned int channel, AVS_EDGETRIGGER_TYPE edgeTriggerType)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Assign a channel index.

edgeTriggerType

The edge trigger type. Please refer AVS_EDGETRIGGER_TYPE

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.70. AVS_DIOGetDIEdgeTrigger

Retrieve the edge trigger type of the DI. The function is supported only when AVS_DIODICAPABILITY_EDGETIRGGERTYPE is in the list which is got from AVS_DIOGetDICapability()

Syntax

C/C++

int AVS_DIOGetDIEdgeTrigger(void* handle, unsigned int channel, AVS_EDGETRIGGER_TYPE *edgeTriggerType)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

edgeTriggerType

The edge trigger type. Please refer AVS_EDGETRIGGER_TYPE

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.71. AVS_DIOSetDITriggerPulseWidth

Set the trigger pulse width of the DI. The function is supported only when AVS_DIODICAPABILITY_TRIGGERPULSEWIDTH is in the list which is got from AVS_DIOGetDICapability()

Syntax

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

C/C++

int AVS_DIOSetDITriggerPulseWidth(void* handle, unsigned int channel, unsigned long unit)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

unit

The number of unit

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.72. AVS_DIOGetDITriggerPulseWidth

Retrieve the trigger pulse width of the DI. The function is supported only when AVS_DIODICAPABILITY_TRIGGERPULSEWIDTH is in the list which is got from AVS_DIOGetDICapability()

Syntax

C/C++

int AVS_DIOGetDITriggerPulseWidth(void* handle, unsigned int channel, unsigned long* unit)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

unit

The number of unit

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.73. AVS_DIOGetDITriggerPulseWidthUnit

Retrieve how many micro seconds the DI trigger pulse width unit is.

Syntax

C/C++

int AVS_DIOGetDITriggerPulseWidthUnit(void* handle, unsigned long* us)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

us

The micro seconds.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.74. AVS_DIOSetDOTriggerDelay

Set the delay trigger time of the DI. The function is supported only when AVS_DIODOCAPABILITY_TRIGGERDELAY is in the list which is got from AVS_DIOGetDICapability()

Syntax

C/C++

int AVS_DIOSetDOTriggerDelay(void* handle, unsigned int channel, unsigned long unit)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

unit

The number of unit

Return Value

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.75. AVS_DIOGetDOTriggerDelay

Retrieve the delay trigger time of the DI. The function is supported only when AVS_DIODOCAPABILITY_TRIGGERDELAY is in the list which is got from AVS_DIOGetDOCapability()

Syntax

C/C++

int AVS_DIOGetDOTriggerDelay(void* handle, unsigned int channel, unsigned long* unit)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

unit

The number of unit

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.76. AVS_DIOGetDOTriggerDelayUnit

Retrieve how many micro seconds the DO delay trigger time is.

Syntax

C/C++

int AVS_DIOGetDOTriggerDelayUnit(void* handle, unsigned long* us)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

us

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

The micro seconds.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.77. AVS_DIOSetDOTriggerPulseWidth

Set the trigger pulse width of the DO. The function is supported only when AVS_DIODOCAPABILITY_TRIGGERPULSEWIDTH is in the list which is got from AVS_DIOGetDOCapability()

Syntax

C/C++

int AVS_DIOSetDOTriggerPulseWidth(void* handle, unsigned int channel, unsigned long unit)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

unit

The number of unit

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.78. AVS_DIOGetDOTriggerPulseWidth

Retrieve the trigger pulse width of the DO. The function is supported only when AVS_DIODOCAPABILITY_TRIGGERPULSEWIDTH is in the list which is got from AVS_DIOGetDOCapability()

Syntax

C/C++

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

int AVS_DIOGetDOTriggerPulseWidth(void* handle, unsigned int channel, unsigned long* unit)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

unit

The number of unit

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.79. AVS_DIOGetDOTriggerPulseWidthUnit

Retrieve how many micro seconds the DO trigger pulse width unit is.

Syntax

C/C++

int AVS_DIOGetDOTriggerPulseWidthUnit (void* handle, unsigned long* us)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

us

The micro seconds.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.80. AVS_DIOTriggerDO

SW trigger DO. The function is supported only when AVS_DIODICAPABILITY_SWTRIGGER is in the list which is got from AVS_DIOGetDOCapability()

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Syntax

C/C++

int AVS_DIOTriggerDO(void* handle, unsigned int channel)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

channel

Assign a channel index.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.81. AVS_DIOTriggerDOs

SW trigger DO. The function is supported only when AVS_DIODICAPABILITY_SWTRIGGER is in the list which is got from AVS_DIOGetDOCapability()

Syntax

C/C++

int AVS_DIOTriggerDOs(void* handle, unsigned int* triggerlist, unsigned int listsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

triggerlist

Assign a list, if a element value bigger than 0, it menas trigger the corresponding DO.

listsize

The number of list elements. The suggested list size could be got from AVS_DIOGetDONums()

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.82. AVS_DIOSetTrigger1vs1MappingTable

Set the 1vs1 trigger mapping table. The function is supported only when AVS_DIOCAPABILITY_1VS1TRIGGER or AVS_DIOCAPABILITY_1VSMULTIPLETRIGGER is in the list which is got from AVS_DIOGetCapability()

Syntax

C/C++

int AVS_DIOSetTrigger1vs1MappingTable(void* handle, unsigned int* maplist, unsigned int listsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

maplist

Assign a list for configing the 1vs1 mapping table. If the index 0 is 2, it means the DO2 is triggered when DI0 is triggered by HW.

listsize

The number of list elements. The suggested list size could be got from AVS_DIOGetDINums()

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.83. AVS_DIOGetTrigger1vs1MappingTable

Retrieve the 1vs1 trigger mapping table. The function is supported only when AVS_DIOCAPABILITY_1VS1TRIGGER or AVS_DIOCAPABILITY_1VSMULTIPLETRIGGER is in the list which is got from AVS_DIOGetCapability()

Syntax

C/C++

int AVS_DIOGetTrigger1vs1MappingTable(void* handle, unsigned int* maplist, unsigned int listsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

maplist

Assign a list for retrieving 1vs1 mapping table. If the index 0 is 2, it means the DO2 is triggered when DI0 is triggered by HW.

listsize

The number of list elements. The suggested list size could be got from AVS_DIOGetDINums()

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.84. AVS_DIOSetTrigger1vsMultipleMappingTable

Set the 1vsMultipple trigger mapping table. The function is supported only when AVS_DIOCAPABILITY_1VSMULTIPLETRIGGER is in the list which is got from AVS_DIOGetCapability()

Syntax

C/C++

int AVS_DIOSetTrigger1vsMultipleMappingTable(void* handle, unsigned int dichannel, unsigned int* maplist, unsigned int listsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

dichannel

Assign a DI channel

maplist

Assign a list for setting 1vsMultiple mapping table. The table is the DO trigger table for the specified DI channel. The index 0 means DO0 and so on. If the dichannel is 3, the do will be triggered according to the value is non-zero in the maplist array when DI3 is triggered by HW.

listsize

The number of list elements. The suggested list size could be got from AVS_DIOGetDONums()

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.85. AVS_DIOGetTrigger1vsMultipleMappingTable

Retrieve the 1vsMultipple trigger mapping table. The function is supported only when AVS_DIOCAPABILITY_1VSMULTIPLETRIGGER is in the list which is got from AVS_DIOGetCapability()

Syntax

C/C++

int AVS_DIOGetTrigger1vsMultipleMappingTable(void* handle, unsigned int dichannel, unsigned int* maplist, unsigned int listsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

maplist

Assign a list for retrieving 1vsMultiple mapping table.The table is the DO trigger table for the specified DI channel. The index 0 means DO0 and so on. If the dichannel is 3, the do will be triggered according to the value is non-zero in the maplist array when DI3 is triggered by HW

listsize

The number of list elements. The suggested list size could be got from AVS_DIOGetDONums()

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.112 Feature – FrameGrabber

This feature is use to control the framegrabber device
For details, you can check the manual of each product.

4.1.86. AVS_FrameGrabberGetDSNums

Get the number of the data streaming.

Syntax

C/C++

int AVS_FrameGrabberGetDSNums(void* handle, unsigned int* nums)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

nums

Return the number of data streaming.
Note: Data stream mapping one channel index (PCIe-CPL64V Full have one datastream)

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.87. AVS_FrameGrabberStartAquisition

Start a frame grabber to retrieve data streaming. The data could be retrieved from AVS_FrameGrabberGetFrame() or AVS_FrameCallback routine

Syntax

C/C++

int AVS_FrameGrabberStartAquisition(void* handle, unsigned int dsindex)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

dsindex

Assign the data streaming index. (Zero based)

Return Value

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.88. AVS_FrameGrabberStopAquisition

Stop a frame grabber to retrieve the data streaming.

Syntax

C/C++

int AVS_FrameGrabberStopAquisition(void* handle, unsigned int dsindex)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

dsindex

Assign the data streaming index. (Zero based)

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.89. AVS_FrameGrabberGetFrame

Get a frame from the framegrabber device. This function will be blocked until timeout. Once received a frame, copy the data to a buffer first if you need to process the frame. Otherwise, you should execute AVS_FrameGrabberQueueBuffer() as soon as possible.

Syntax

C/C++

int AVS_FrameGrabberGetFrame(void* handle, unsigned int* dsindex, void **phBuffer, size_t* iSize, unsigned long mstimeout)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

dsindex

Received the data streaming index. (Zero based)

phBuffer

Return a buffer pointer if a frame is retrieved. Don't modify the buffer content, please copy out to another user space buffer for the advanced processing.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

iSize

The size of the frame data

mstimeout

The timeout value. (MilliSecond)

Return Value

Return ::AVS_FUN_SUCCESS if success to retrieve a frame. ::AVS_FUN_FAILED if failed, or other values please refer ::AVS_FUN_RETURNVAL in the I_common.h.

4.1.90. AVS_FrameGrabberQueueBuffer

Queue a buffer to insert into the waiting queue to retrieve the streaming data. Use this function to restore the buffer state after retrieving a frame with AVS_FrameGrabberGetFrame().

Syntax

C/C++

int AVS_FrameGrabberQueueBuffer(void* handle, void *phBuffer)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

phBuffer

Buffer pointer from AVS_FrameGrabberGetFrame(). Specified the buffer to start retrieving the data streaming

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.91. AVS_FrameGrabberGetDropFrame

This function gets dropped frames after start capturing. The dropped frames will be reset when call AVS_FrameGrabberStartAquisition function.

The system capture frames as soon as it can. But sometimes user's get grame routine spends too much time that the next frame will be dropped because the frame buffer queue is full. To solve this problem, user needs to shorten the process time of the thread processing.

Syntax

C/C++

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

int AVS_FrameGrabberGetDropFrame(void* handle, unsigned int* dsindex, unsigned int* dropcount, unsigned int* frameevent, unsigned int* framedropevent)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

dsindex

Received the data streaming index. (Zero based)

dropcount

Indicates how many frames were dropped.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.92. AVS_FrameGrabberSetCallback

This function set user's defined frame callback routine. Callback routine will be called when event is ready.

Remarks: The callback runs inside an internal acquisition thread. Avoid long-running or blocking operations (such as file I/O, GUI operations, or complex image processing). If heavy work is required, copy the data and process it in another worker thread.

AVS_FrameGrabberSetCallback and AVS_FrameGrabberGetFrame must not be used at the same time on the same data stream.

Once a callback is registered by calling AVS_FrameGrabberSetCallback, the frame grabber operates in callback mode, and all frame notifications are delivered through the user callback. In this mode, calling AVS_FrameGrabberGetFrame on the same handle leads to undefined behavior and is not supported.

Syntax

C/C++

int AVS_FrameGrabberSetCallback(void* handle, AVS_FrameCallback cb, unsigned long mstimeout) , ~~callback routine as void(AVS_CALL *AVS_FrameCallback)(int eventType, void* buffer, int size);~~

Callback Prototype

```
typedef void (AVS_CALL *AVS_FrameCallback)(
  int eventType,
  const void* buffer,
  int size
```


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

const void* frameinfo,
);

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

cb

Pointer to the callback routine. The callback is invoked from an internal grabbing thread when a frame event occurs.

eventType: 0 (frame ready) ,
1 (Missing line data or tap configuration is incorrect),
2 (timeout),
3 (function failed, System internal error.)
4 (Stop acquisition)

buffer

Return a buffer pointer if a frame is retrieved. Don't modify the buffer content, please assign other point to process. ⚠ **Do NOT modify the content of this buffer.**

size

The size of the frame data

frameinfo

The frame information (Reserved)

mstimeout

The timeout value. (MilliSecond)

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.1.93. AVS_FrameGrabberSetBufferCount

This function configures how many frame buffers are pre-allocated and used for image acquisition. The buffer will be allocated when AVS_FrameGrabberStartAquisition function is called.

Syntax

C/C++

int AVS_FrameGrabberSetBufferCount(void* handle, unsigned int buffercount)

Parameter(s)

handle

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Assign a product handle. Get from AVS_GetProductHandle()

buffercount

Assign number of buffer (default = 4, max=256)

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.113 Feature – **AdvancedConfig**

This feature is use to config the parameters of a device.
For details, you can check the manual of each product or the configuration field definition section.

4.2.1. AVS_AdvancedConfigWritePars

Write the configuration parameter into the device

Syntax

C/C++

int AVS_AdvancedConfigWritePars(void* handle, unsigned long long address, void *pBuffer, unsigned int iSize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

address

Specified an address to set parameters. The suppored address could be retrieved from AVS_AdvancedConfigGetSupportedAdr(). For a device, these addresses should be defined the strings.

pBuffer

The configuration data

iSize

The size of the configuration data.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.2.2. AVS_AdvancedConfigReadPars

Read the configuration parameter into the device

Syntax

C/C++

int AVS_AdvancedConfigReadPars(void* handle, unsigned long long address, void *pBuffer, unsigned int *piSize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

address

Specified an address to get parameters. The suppored address could be retrieved from AVS_AdvancedConfigGetSupportedAdr(). For a device, these addresses should be defined the strings.

pBuffer

The allocated buffer for put the configuration data

piSize

Input the buffer size and output the configuration data size

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.2.3. AVS_AdvancedConfigGetSupportedAdr

Retrieved the supported addresses for the device configuration. Each address is the defined string

Syntax

C/C++

int AVS_AdvancedConfigGetSupportedAdr(void* handle, char* pBuffer, unsigned int bufsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

pBuffer

The allocated buffer for retrieving the data

bufsize

Input the buffer size and output the data size

Return Value

Return AVS_FUN_SUCCESS if success, AVS_FUN_BUFFERSIZEISTOOSMALL if the specified buffer or buffer size is too smaller for the operation or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.114 Feature – **CameraLink**

This feature is use to config the cameralink device.
For details, you can check the manual of each product or the configuration field definition section.

4.2.4. AVS_CameraLinkImportCamFile

Import the cam file into CameraLink device

Syntax

C/C++

int AVS_CameraLinkImportCamFile(void* handle, const char* filepath)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

filepath

The specified file path.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.2.5. AVS_CameraLinkExportCamFile

Export the camfile from the device

Syntax

C/C++

int AVS_CameraLinkExportCamFile(void* handle, const char* filepath)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

filepath

The specified file path.

Return Value

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.2.6. AVS_CameraLinkGetSupportedSectionKey

Retrieve the suppored list of the section, key and how to assign the value. The list of supported CAM-file sections, keys, and valid assignment formats for the Camera Link configuration. Camfile is described in section 4.2.2.1

This function returns a serialized text block describing all supported sections and keys, allowing users to know which CAM file parameters are recognized by the frame grabber.

Syntax

C/C++

int AVS_CameraLinkGetSupportedSectionKey(void* handle, void *pBuffer, unsigned int bufsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

pBuffer

The allocated buffer for retrieving the data

bufsize

Input the buffer size and output the data size

Return Value

Return AVS_FUN_SUCCESS if success, AVS_FUN_BUFFERSIZEISTOOSMALL if the specified buffer or buffer size is too smaller for the operation or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.2.7. AVS_CameraLinkReadParameter

Read a string by the camfile's section and key

Syntax

C/C++

int AVS_CameraLinkReadParameter(void* handle, const char* section, const char* key, char* strval, unsigned int strsize)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

section

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Name of the CAM file section to read from. A section corresponds to the bracket-enclosed category in the CAM file. Example: [ImageFormat]

key

The parameter name inside the specified section.
Example keys inside
[ImageFormat]
PixelFormat

strval

Pointer to a character buffer that receives the retrieved value

strsize

Assign the buffer size of the retrieved string

Return Value

Return AVS_FUN_SUCCESS if success, AVS_FUN_CL_SECTIONORKEYERROR if the cmeralink section or key is invalid, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

4.2.8. AVS_CameraLinkWriteParameter

Write a string by the section and key

Syntax

C/C++

int AVS_CameraLinkWriteParameter(void* handle, const char* section, const char* key, const char* strval)

Parameter(s)

handle

Assign a product handle. Get from AVS_GetProductHandle()

section

Name of the CAM file section to read from. A section corresponds to the bracket-enclosed category in the CAM file. Example: [ImageFormat]

key

The parameter name inside the specified section.
Example keys inside: PixelFormat

strval

Input the value string

Return Value

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Return AVS_FUN_SUCCESS if success, or other values please refer AVS_FUN_RETURNVAL in the I_common.h.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

5. Serial Communication Functions

Serial communication functions are frame grabber-independent API. Allow users to write a camera-specific configuration utility. The following API offers standard asynchronous serial communi-cation API followed by camera link specification.

4.115 Defined Data Type

Defined Data Type	C++ Type	C# Type	VB.Net Type
hSerRef	void*	IntPtr	IntPtr
INT32	int	int	integer
UINT32	unsigned int	uint	UInt32
INT8	char	Byte	Byte

4.116 clFlushPort

Purpose

This function discards any bytes that are available in the input buffer.

Prototype

C/C++ INT32 clFlushPort(hSerRef serialRef)
C# int clFlushPort(IntPtr serialRef)
VB.Net clFlushPort(ByVal serialRef As IntPtr) As Integer

Parameters

serialRef
The value obtained by the clSerialInit function.

Return Value

CL_ERR_NO_ERR
CL_ERR_INVALID_REFERENCE
Refer to Section 5.13 for more information about status codes.

4.117 clGetErrorText

Purpose

This function converts an error code to error text for display in a dialog box or in a standard I/O window.

Prototype

C/C++ INT32 clGetErrorText(const INT8* manuName, INT32 errorCode, INT8* errorText, UINT32* errorTextSize)
C# int clGetErrorText(string manuName, int errorCode, byte[] errorText, ref uint errorTextSize)
VB.Net clGetErrorText(ByVal manuName As String, ByVal errorCode As Integer, ByVal errorText() As Byte, ByRef errorTextSize As UInt32) As Integer

Parameters

manufacturerName
The manufacturer name in a NULL-terminated buffer. Manufacturer name is returned from clGetPortInfo.
errorCode
The error code used to find the appropriate error text.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

errorText
A caller-allocated buffer which contains the NULL-terminated error text on function return.
errorTextSize
On success, contains the number of bytes written into the buffer, including the NULL-termination character. This value should be the size in bytes of the error text buffer passed in. On CL_ERR_BUFFER_TOO_SMALL, contains the size of the buffer needed to write the error text.

Return Value

CL_ERR_NO_ERR
CL_ERR_MANU_DOES_NOT_EXIST
CL_ERR_BUFFER_TOO_SMALL
CL_ERR_ERROR_NOT_FOUND
Refer to Section 5.13 for more information about status codes.

4.118 clGetNumBytesAvail

Purpose

This function outputs the number of bytes that are received at the port specified by serialRef but are not yet read out.

Prototype

C/C++	INT32 clGetNumBytesAvail(hSerRef serialRef, UINT32* numBytes)
C#	int clGetNumBytesAvail(IntPtr serialRef, ref uint numBytes)
VB.Net	clGetNumBytesAvail (ByVal serialRef As IntPtr, ByRef numBytes As UInt32) As Integer

Parameters

serialRef
The value obtained by the clSerialInit function.
numBytes
The number of bytes currently available to be read from the port.

Return Value

CL_ERR_NO_ERR
CL_ERR_INVALID_REFERENCE
Refer to Section 5.13 for more information about status codes.

4.119 clGetNumPorts

Purpose

This function returns the total number of Camera Link serial ports in your system.

Prototype

C/C++	INT32 clGetNumPorts(UINT32* Ports)
C#	int clGetNumPorts(ref uint Ports)
VB.Net	clGetNumPorts (ByRef Ports As UInt32) As Integer

Parameters

numPorts
The number of Camera Link serial ports in your system.

Return Value

CL_ERR_NO_ERR
Refer to Section 5.13 for more information about status codes.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.120 clGetPortInfo

Purpose

This function provides information about the port specified by the index.

Prototype

C/C++ INT32 clGetPortInfo(UINT32 serialIndex, INT8* manufacturerName, UINT32* nameBytes, INT8* portID, UINT32* IDBytes, UINT32* version)

C# int clGetPortInfo(uint serialIndex, byte[] manufacturerName, ref uint nameBytes, byte[] portID, byte[] IDBytes, ref uint version)

VB.Net clGetPortInfo (ByVal serialIndex As UInt32, ByVal manufacturerName() As Byte, ByRef nameBytes As UInt32, ByVal portID() As Byte, ByVal IDBytes() As Byte, ByRef version As UInt32) As Integer

Parameters

serialIndex

The index of the port for which you want information. The valid range for this index is 0 to (n-1) where n is the value of numPorts returned by clGetNumPorts.

manufacturerName

Pointer to a user-allocated buffer into which the function copies the manufacturer name. The returned name is NULL-terminated. In the case that the DLL conforms to the October 2000 specification, this parameter will contain the file name of the DLL rather than the manufacturer name.

nameBytes

As an input, this value should be the size of the buffer that is passed. On successful return, this parameter contains the number of bytes written into the buffer, including the NULL termination character.
On CL_ERR_BUFFER_TOO_SMALL , this parameter contains the size of the buffer needed to write the data text..

portID

A manufacturer-specific identifier for the serial port. In the case that the manufacturer DLL conforms to the October 2000 specification, on return this parameter will be Port n, where n is a unique index for the port.

IDBytes

As an input, this value should be the size of the buffer that is passed. On successful return, this parameter contains the number of bytes written into the buffer, including the NULL-termination character.
On CL_ERR_BUFFER_TOO_SAMLL, this parameter contains the size of the buffer needed to write the data text.

Version

The version of the Camera Link specifications with which this frame grabber software complies. The Camera Link Constants are defined below:
CL_DLL_VERSION_NO_VERSION 1
CL_DLL_VERSION_1_0 2
CL_DLL_VERSION_1_1 3

Return Value

CL_ERR_NO_ERR
CL_ERR_BUFFER_TOO_SMALL
CL_ERR_INVALID_INDEX
Refer to Section 5.13 for more information about status codes.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.121 clGetSupportedBaudRates

Purpose

This function returns the valid baud rates of the interface. This function returns the valid baud rates of the current interface. This function is required for clseradl.dll and is available from clallserial.dll.

Prototype

C/C++	INT32 clGetSupportedBaudRates(hSerRef serialRef, UINT32* baudRates)
C#	int clGetSupportedBaudRates(IntPtr serialRef, ref uint baudRates)
VB.Net	clGetSupportedBaudRates (ByVal serialRef As IntPtr, ByRef baudRates As UInt32) As Integer

Parameters

serialRef

The value obtained from the clSerialInit function, which describes the port being queried for baud rates.

baudRates

Bitfield that describes all supported baud rates of the serial port as described by serialRef. The supported baud rates are defined below:
CL_BAUDRATE_9600 1
CL_BAUDRATE_19200 2
CL_BAUDRATE_38400 4
CL_BAUDRATE_57600 8
CL_BAUDRATE_115200 16
CL_BAUDRATE_230400 32
CL_BAUDRATE_460800 64

Return Value

CL_ERR_NO_ERR CL_ERR_INVALID_REFERENCEL
CL_ERR_FUNCTION_NOT_FOUND
Refer to Section 5.13 for more information about status codes.

4.122 clSerialClose

Purpose

This function closes the serial device and cleans up resources associated with serialRef. Upon return, serialRef is no longer usable.

Prototype

C/C++	void clSerialClose(hSerRef serialRef)
C#	void clSerialClose(IntPtr serialRef)
VB.Net	clSerialClose (ByVal serialRef As IntPtr)

Parameters

serialRef

The value obtained from the clSerialInit function, for cleanup .

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.123 clSerialInit

Purpose

This function initializes the device referred to by serialIndex, and returns a pointer to an internal serial reference structure.

Prototype

C/C++ INT32 clSerialInit (UINT32 serialIndex, hSerRef* serialRefPtr)
C# int clSerialInit (uint serialIndex, ref IntPtr serialRefPtr)
VB.Net clSerialInit (ByVal serialIndex As UInt32, ByRef serialRefPtr As IntPtr) As Integer

Parameters

serialIndex
A zero-based index value. If there are n serial devices in the system that is supported by this library, the range of serialIndex is 0 to (n-1). The order of the serial devices is vendor specific. The number of serial ports supported by this DLL is output by the clGetNumSerialPorts function.

serialRefPtr
Upon a successful call, a pointer to the vendor specific reference to the current serial session will be put into the value pointed to by serialRefPtr.

Return Value

CL_ERR_NO_ERR
CL_ERR_PORT_IN_USE
CL_ERR_INVALID_INDEX
CL_ERR_UNABLE_TO_OPEN_PORT
Refer to Section 5.13 for more information about status codes.

4.124 clSerialRead

Purpose

This function reads from the serial device referenced by serialRef.

Prototype

C/C++ INT32 clSerialRead(hSerRef serialRef, INT8* buffer, UINT32* bufferSize, UINT32 serialTimeout)
C# int clSerialRead(IntPtr serialRef, byte[] buffer, ref uint bufferSize, uint serialTimeout)
VB.Net clSerialRead (ByVal serialRef As IntPtr, ByVal buffer() As Byte, ByRef bufferSize As UInt32, ByVal serialTimeout As UInt32) As Integer

Parameters

serialRef
The value obtained from clSerialInit function.

Buffer
Points to a user-allocated buffer. Upon a successful call, contains the data read from the serial device.

bufferSize
As an input parameter, bufferSize contains the buffer size to indicate the maximum number of bytes that the buffer can accomodate. Upon a successful call, bufferSize is the number of bytes that were read successfully from the serial device.

serialTimeout
Indicates the timeout, in milliseconds.

Return Value

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

CL_ERR_NO_ERR
CL_ERR_TIMEOUT
CL_ERR_INVALID_REFERENCE
Refer to Section 5.13 for more information about status codes.

4.125 clSerialWrite

Purpose

This function writes data in buffer to the serial device referenced by serialRef.

Prototype

C/C++ INT32 clSerialWrite(hSerRef serialRef, INT8* buffer, UINT32* bufferSize, UINT32 serialTimeout)

C# int clSerialWrite(IntPtr serialRef, string buffer, ref uint bufferSize, uint serialTimeout)
Or
int clSerialWrite(IntPtr serialRef, byte[] buffer, ref uint bufferSize, uint serialTimeout)

VB.Net clSerialWrite (ByVal serialRef As IntPtr, ByVal buffer As String, ByRef bufferSize As UInt32, ByVal serialTimeout As UInt32) As Integer
Or
clSerialWrite (ByVal serialRef As IntPtr, ByVal buffer() As Byte, ByRef bufferSize As UInt32, ByVal serialTimeout As UInt32) As Integer

Parameters

serialRef
The value obtained from clSerialInit function.

buffer
Contains data to write to the serial port.

bufferSize
Contains the buffer size indicating the maximum number of bytes to be written. Upon a successful call, bufferSize contains the number of bytes written to the serial device.

serialTimeout
Indicates the timeout in milliseconds.

Return Value

CL_ERR_NO_ERR
CL_ERR_TIMEOUT
CL_ERR_INVALID_REFERENCE
Refer to Section 5.13 for more information about status codes.

4.126 clSetBaudRate

Purpose

This function sets the Baud Rate for the serial port of the selected port. Use clGetSupportedBaudRates to determine supported baud rates.

Prototype

C/C++ INT32 clSetBaudRate(hSerRef serialRef, UINT32 baudRate)

C# int clSetBaudRate(IntPtr serialRef, uint baudRate)

VB.Net clSetBaudRate (ByVal serialRef As IntPtr, ByVal baudRate As UInt32) As Integer

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Parameters

serialRef

The value obtained from clSerialInit function.

baudRate

The baud rate you want to use. This input expects the values represented by the CL_BAUDRATE constants.

Return Value

CL_ERR_NO_ERR
CL_ERR_INVALID_REFERENCE
CL_ERR_BAUD_RATE_NOT_SUPPORTED
Refer to Section 5.13 for more information about status codes.

4.127 Status Codes

Error Code	Definition Name	Meaning
0	CL_ERR_NO_ERR	Function returned successfully.
-10001	CL_ERR_BUFFER_TOO_SMALL	User buffer not large enough to hold data.
-10002	CL_ERR_MANU_DOES_NOT_EXIST	The requested manufacturer's DLL does not exist on your system.
-10003	CL_ERR_PORT_IN_USE	Port is valid but cannot be opened because it is in use.
-10004	CL_ERR_TIMEOUT	Operation not completed within specified timeout period.
-10005	CL_ERR_INVALID_INDEX	Not a valid index
-10006	CL_ERR_INVALID_REFERENCE	The serial reference is not valid.
-10007	CL_ERR_ERROR_NOT_FOUND	Could not find the error description for this error code.
-10008	CL_ERR_BAUD_RATE_NOT_SUPPORTED	Requested baud rate is not supported by this interface.
-10009	CL_ERR_OUT_OF_MEMORY	System is out of memory and could not perform required action.
-10010	CL_ERR_REGISTRY_KEY_NOT_FOUND	Could not found cameralink registry key in system registry.
-10098	CL_ERR_UNABLE_TO_LOAD_DLL	The DLL was unable to load due to a lack of memory or because it does not export all required functions.
-10099	CL_ERR_FUNCTION_NOT_FOUND	Function does not exist in the manufacturer's library.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000